ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ


**Trần Tuấn Thịnh**

# PHÁT TRIỂN HỆ THỐNG HỖ TRỢ QUẢN LÝ NHÀ HÀNG VỪA VÀ NHỎ

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**
**Ngành: Công nghệ thông tin**

**HÀ NỘI - 2024**

**ĐẠI HỌC QUỐC GIA HÀ NỘI**
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**

**Trần Tuấn Thịnh**

# PHÁT TRIỂN HỆ THỐNG HỖ TRỢ QUẢN LÝ NHÀ HÀNG VỪA VÀ NHỎ

**KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**
**Ngành: Công nghệ thông tin**

**Cán bộ hướng dẫn: TS. Võ Đình Hiếu**

**HÀ NỘI - 2024**

**VIETNAM NATIONAL UNIVERSITY, HANOI**
**UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

**Tran Tuan Thinh**

# DEVELOP A MANAGEMENT SYSTEM FOR SMALL AND MEDIUM-SIZED EATERIES

**BACHELOR'S THESIS**
**Major: Information Technology**

**Supervisor: Dr. Vo Dinh Hieu**

**Hanoi - 2024**

# LỜI CAM ĐOAN

Tôi tên là Trần Tuấn Thịnh, sinh viên của lớp QH-2020-I/CQ-C-CLC, ngành Công nghệ thông tin hệ chất lượng cao. Tôi xin cam đoan rằng đề tài khóa luận tốt nghiệp của tôi với chủ đề "Phát triển hệ thống hỗ trợ quản lý nhà hàng vừa và nhỏ" là sản phẩm nghiên cứu của bản thân tôi và chưa bao giờ được nộp làm khóa luận tốt nghiệp cho bất kỳ trường đại học nào, trong đó có Trường Đại học Công nghệ. Tôi cam đoan rằng tôi không sao chép bất kỳ nội dung nào từ các tài liệu khác, cũng như không sử dụng kết quả của bất kỳ người nào khác mà không có trích dẫn cụ thể. Đây là công trình nghiên cứu do cá nhân tôi thực hiện với sự hướng dẫn của TS Võ Đình Hiếu, không chứa bất kỳ mã nguồn nào được sao chép, chỉnh sửa từ cá nhân, tổ chức nào khác mà không được cho phép. Nếu vi phạm những điều trên, tôi hoàn toàn chấp nhận mọi trách nhiệm theo quy định của Trường Đại học Công nghệ.

Hà Nội, ngày 27 tháng 5 năm 2024

Sinh viên

Trần Tuấn Thịnh

# LỜI CẢM ƠN

Lời đầu tiên cho phép em được gửi lời cảm ơn tới Khoa Công Nghệ Thông Tin – Trường Đại học Công nghệ - ĐHQG Hà Nội đã tạo điều kiện thuận lợi cho em được học tập, nghiên cứu và thực hiện đề tài tốt nghiệp này.

Em cũng xin được bày tỏ lòng biết ơn sâu sắc tới thầy Võ Đình Hiếu đã tận tình hướng dẫn, đóng góp những ý kiến xác đáng để em có thể hoàn thành khóa luận một cách tốt nhất. Thời gian được làm việc và nghiên cứu tại phòng thí nghiệm thật sự đã đem lại những kinh nghiệm, những kiến thức quý báu và vô giá đối với bản thân em.

Cuối cùng, em cũng vô cùng biết ơn những thầy cô trong trường đã tận tình giảng dạy, trang bị cho em những kiến thức quan trọng để em có đầy đủ kiến thức và kỹ năng để có thể tự tin đi tiếp những chặng đường tiếp theo.

Chúc mọi người luôn luôn vui vẻ và gặt hái được nhiều thành công trong cuộc sống.

# TÓM TẮT

***Tóm tắt:*** Khóa luận trình bày về quá trình phát triển một nền tảng tập trung hỗ trợ quản lý nhà hàng và hỗ trợ đặt đồ ăn dành riêng cho các nhà hàng vừa và nhỏ.

Hệ thống này sẽ cung cấp một ứng dụng quản lý cho một doanh nghiệp với các chức năng đầy đủ và toàn diện như quản lý bán hàng, doanh thu, nhân viên, khách hàng cũng như hỗ trợ quảng bá cho nhà hàng. Điều này giúp tối ưu hóa hoạt động kinh doanh của nhà hàng, giúp giảm thiểu chi phí nhân viên và vận hành, quản lý. Ngoài ra, hệ thống cũng tích hợp tính năng đặt đồ ăn thông qua quét mã QR với giao diện thân thiện, dễ sử dụng, cho phép khách hàng dễ dàng đặt món và thanh toán tại nhà hàng trên nền tảng.

Bằng việc áp dụng kiến trúc vi dịch vụ (microservice) khi phát triển và triển khai hệ thống, từng dịch vụ như thông báo, đặt đồ ăn, quản lý nhà hàng, v.v. sẽ được tách biệt riêng với nhau giúp tăng tính sẵn sàng cũng như là khả năng chịu lỗi của hệ thống. Ngoài ra về sau khi số lượng người dùng tăng, kiến trúc này cũng đảm bảo khả năng nâng cấp linh hoạt, dễ dàng và đặc biệt trong những khung giờ cao điểm khi số lượng người dùng vượt quá mức sử dụng thường thấy, hệ thống hoàn toàn có thể tự động mở rộng mà không cần tới sự can thiệp của quản trị viên hệ thống.

***Từ khóa:*** *Microservice, tự động mở rộng, quản lý quán ăn, nhà hàng, nền tảng tập trung*

# ABSTRACT

***Abstract:*** This thesis presents the development process of a centralized system to support restaurant management and online food ordering specifically for small and medium-sized restaurants.

The system will provide a management application for businesses with comprehensive and full-fledged functionalities such as sales, revenue, staff, and customer management, as well as restaurant promotion support. This will help optimize restaurant operations, minimize staff and operational costs, and management. In addition, the system also integrates food ordering feature through QR scanning with a friendly and easy-to-use interface, allowing customers to easily order and pay for food at the restaurant on the platform.

By applying a microservice architecture to the development and deployment of the system, each service such as notifications, food ordering, restaurant management, etc., will be separated from each other, increasing the availability and fault tolerance of the system. Additionally, as the number of users increases, this architecture also ensures flexible and easy upgradeability, and especially during peak hours when the number of users exceeds the usual usage, the system can automatically scale without the need for system administrator intervention.

***Keywords:*** *Microservice, auto-scaling, restaurant management, restaurants, centralized platform*

# Mục lục

# Danh sách hình vẽ

# Danh sách bảng

# Danh sách đoạn mã

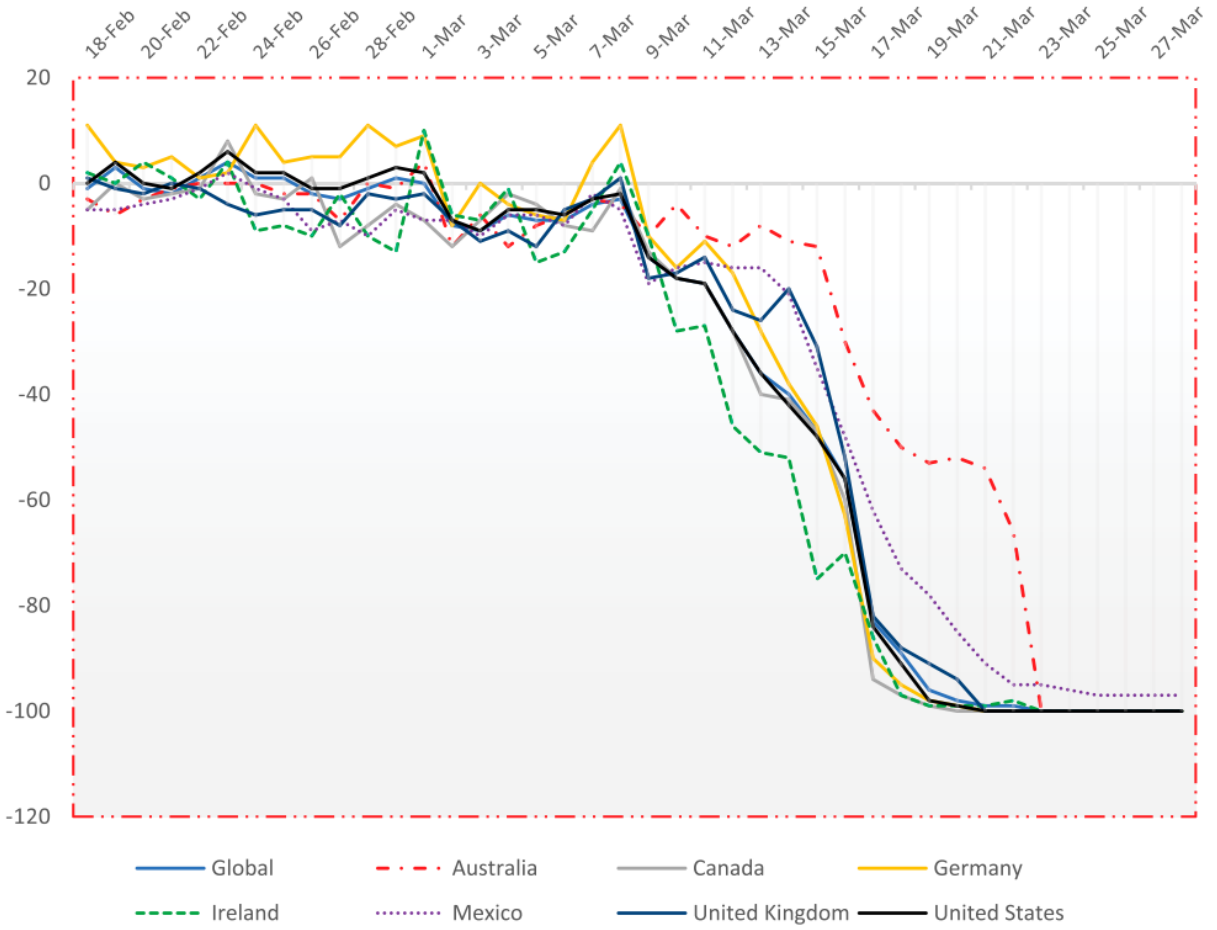
# Thuật ngữ

| Thuật ngữ | | Ý nghĩa tiếng Việt |
|---|---|---|
| **Từ viết tắt** | **Từ đầy đủ tiếng Anh** | |
| K8s | Kubernetes | Cây cú pháp trừu tượng |
| HPA | Horizontal Pod Autoscale | Tự động mở rộng Pod theo chiều ngang |
| HA | High Availability | Tính sẵn sàng cao |
| CSP | Cloud Service Provider | Đồ thị dòng điều khiển |
| GCP | Google Cloud Platform | Dòng mã |
| GKE | Google Kubernetes Engine | Ngôn ngữ đánh dấu siêu văn bản |
| HTML | Hypertext Markup Language | Lý thuyết mô-đun thỏa mãn |
| JSX | JavaScript XML | Lý thuyết mô-đun thỏa mãn |
| GLSB | Global Server Load Balancers | Lý thuyết mô-đun thỏa mãn |
| API | Application Programming Interface | GIao diện lập trình ứng dụng |
| | Microservice | kiến trúc vi dịch vụ |
| | Message Queue | hàng đợi tin nhắn |
| | API Gateway | Cổng API |
| | Request | yêu cầu |
| | Autoscale | Tự động mở rộng |
| | Container | Tương trưng cho một máy ảo (VM), mỗi container đều có CPU, cây thư mục hệ thống, không gian xử lý, bộ nhớ, v.v. |

# Đặt vấn đề

Hiện nay, trong thời đại chuyển đổi số khi mà mọi giao dịch, thanh toán đều được thực hiện trên thiết bị di động của người dùng, các nhà phát triển phần mềm đứng trước một cơ hội để tối ưu hóa các quy trình rườm rà đã có nhằm giảm thiểu tương tác trực tiếp giữa nhà cung cấp dịch vụ và người dùng từ đó đẩy nhanh quy trình nghiệp vụ. Một trong các lĩnh vực được quan tâm và chuyển đổi mạnh mẽ trong những năm gần đây đó là về lĩnh vực dịch vụ ăn uống tại Việt Nam. Trong đại dịch Covid-19, đã có một sự dịch chuyển lớn trong ngành ẩm thực. Năm 2020, một khảo sát đã chỉ ra rằng số lượng người đến ăn tại nhà hàng đã giảm trên 90% do lệnh phong tỏa của các quốc gia và nỗi sợ nguy cơ lây nhiễm từ người qua người [1].

Mặc dù các ứng dụng đặt đồ ăn trực tuyến đã tồn tại từ lâu từ trước khi đại dịch xảy ra, nhưng khi khách hàng không còn lựa chọn ra ngoài ăn, số lượng người dùng mới đăng ký tham gia những nền tảng này tăng đáng kể xuyên suốt thời gian này [2]. Các khảo sát sau đó cũng cho thấy rằng 64% [4] người Việt Nam dự định tiếp tục sử dụng các ứng dụng đặt đồ ăn trong tương lai. Như vậy từ sau đại dịch, đã có sự gia tăng một cách nhanh chóng trong thói quen đặt đồ ăn trực tuyến và mua hàng online của người dùng. Khảo sát cũng cho thấy người dùng đặt ưu tiên về độ sạch sẽ của nhà hàng lên cao ngoài chất lượng món ăn khi họ đi ăn [2]. Từ tất cả những điều trên, các quán ăn, nhà hàng mới chớm nở trong những năm trở lại đây đều gặp phải những trở ngại, thách thức nếu mong muốn mở rộng, phát triển và thu hút khách hàng tiềm năng đến với địa chỉ của họ. Các quán ăn giờ đây cần phải chuyển dịch cơ cấu bán hàng của họ qua hình thức bán hàng trực tuyến [3]. Điều này tức là mở rộng độ phủ của họ, tham gia vào các nền tảng bán hàng trực tuyến như Shopee Food, Grab Food, v.v. Ngoài ra, việc thực khách giờ đây đặt ưu tiên sự sạch sẽ lên cao cũng tạo áp lực cho nhà hàng cần quản lý chặt chẽ nguồn cung của mình. Đối với các quán ăn nhỏ lẻ, việc phát triển đồng thời hai phương




Hình 0.1: Số liệu khách đến nhà hàng tại các nơi trên thế giới [1]

thức bán hàng đó là bán hàng trực tuyến và bán hàng truyền thống rất tốn kém do chi phí đi kèm từ việc tham gia các nền tảng bán hàng trực tuyến đó là phí duy trì thành viên lên tới 25% cho Shopee Food[4]. Từ đây nhiều quán ăn chọn cách chuyển dịch cơ cấu sang chỉ buôn bán online nhằm giảm thiểu tối đa chi phí phát sinh cho việc thuê mặt bằng, chỗ ngồi cho khách hàng cũng như là chi phí cho các ứng dụng, hệ thống quản lý quán ăn.

Khóa luận này phát triển một giải pháp cho vấn đề nói trên. Một hệ thống hỗ trợ quản lý nhà hàng, quán ăn cho các quán ăn tham gia dịch vụ. Hệ thống cung cấp một ứng dụng cho phép các nhà hàng quản lý quán ăn của họ bao gồm quản lý nhân viên, thực đơn, thống kê doanh thu, hóa đơn theo các mốc thời gian cụ thể. Ngoài ra nền tảng còn cung cấp giải pháp giúp các bên tham gia xây dựng nên một trang giới thiệu cho chính nhà hàng của họ đi kèm các dịch vụ như đặt bàn, hỗ trợ khách hàng, v.v. Trong tương

[4]https://merchant.shopeefood.vn/edu/article/phi-mo-gian-hang-va-chiet-khau-tren-shopeefood-la-bao-nhieu

lai, hệ thống cũng hướng đến việc cho phép người dùng tạo lập tài khoản cho chính họ và cung cấp các tính năng giúp người dùng quản lý những thông tin như lịch sử ăn tại những nhà hàng, quán ăn họ đã đến và tìm kiếm các quán ăn dựa theo khẩu vị, địa điểm của người dùng. Một khi người dùng đã có tài khoản trên nền tảng, các quán ăn, nhà hàng có thể biết được thông tin của người đến ăn nếu họ có trong hệ thống, từ đó đưa ra các chương trình khuyến mãi hợp lý cho bất kỳ thực khách quen nào tại chuỗi nhà hàng, quán ăn đó.

Để đáp ứng được đầy đủ các chức năng nghiệp vụ được đặt ra với số lượng người dùng lớn từ các nhà hàng, quán ăn trải dài khắp cả nước, hệ thống của nền tảng quản lý nhà hàng, quán ăn cần đáp ứng số lượng người dùng lớn, đặt biệt trong những khung giờ cao điểm. Khả năng dễ dàng mở rộng về mặt phần cứng cũng là một yếu tố quan trọng được chú ý đến khi nền tảng tận dụng khả năng mở rộng hệ thống tự động của GKE (Google Kubernetes Engine) thông qua tính năng autoscale (tự động mở rộng) khi hệ thống trải qua một lượng truy cập lớn. Chi tiết về thiết kế của hệ thống sẽ được nói rõ hơn trong Chương Thiết kế hệ thống

Phần còn lại của khóa luận được trình bày như sau. Chương 1 trình bày giới thiệu về thiết kế tổng quan của hệ thống và các chức năng chính cũng như đi qua biểu đồ ca sử dụng của một vài chức năng cụ thể. Tiếp theo, Chương 2 mô tả chi tiết thêm về từng thành phần cấu thành hệ thống, từ thiết kế của các cơ sở dữ liệu cho đến luồng người dùng cho nhà hàng, quán ăn, khách hàng, v.v. Chương 3 sẽ giới thiệu về các công nghệ được sử dụng trong quá trình xây dựng hệ thống, chia sẻ về quá trình cài đặt và triển khai hệ thống lên môi trường đám mây của Google, tích hợp CI/CD (Continuous Integration/Continuous Delivery) cho dự án giúp tự động hóa quá trình cài đặt cũng như đảm bảo tính sẵn sàng cao cho hệ thống khi chạy thực nghiệm. Ngoài ra Chương 3 cũng mô tả luồng kỹ thuật dưới dạng sơ đồ tuần tự khi người dùng thực hiện các chức năng cụ thể trên ứng dụng. Cuối cùng Chương 4 thực hiện đánh giá hệ thống, hiệu quả hoạt động thông qua các bộ kiểm thử cụ thể cho từng tính năng được chọn.

Nên trình bày 1 số hệ thống quản lý tương tự

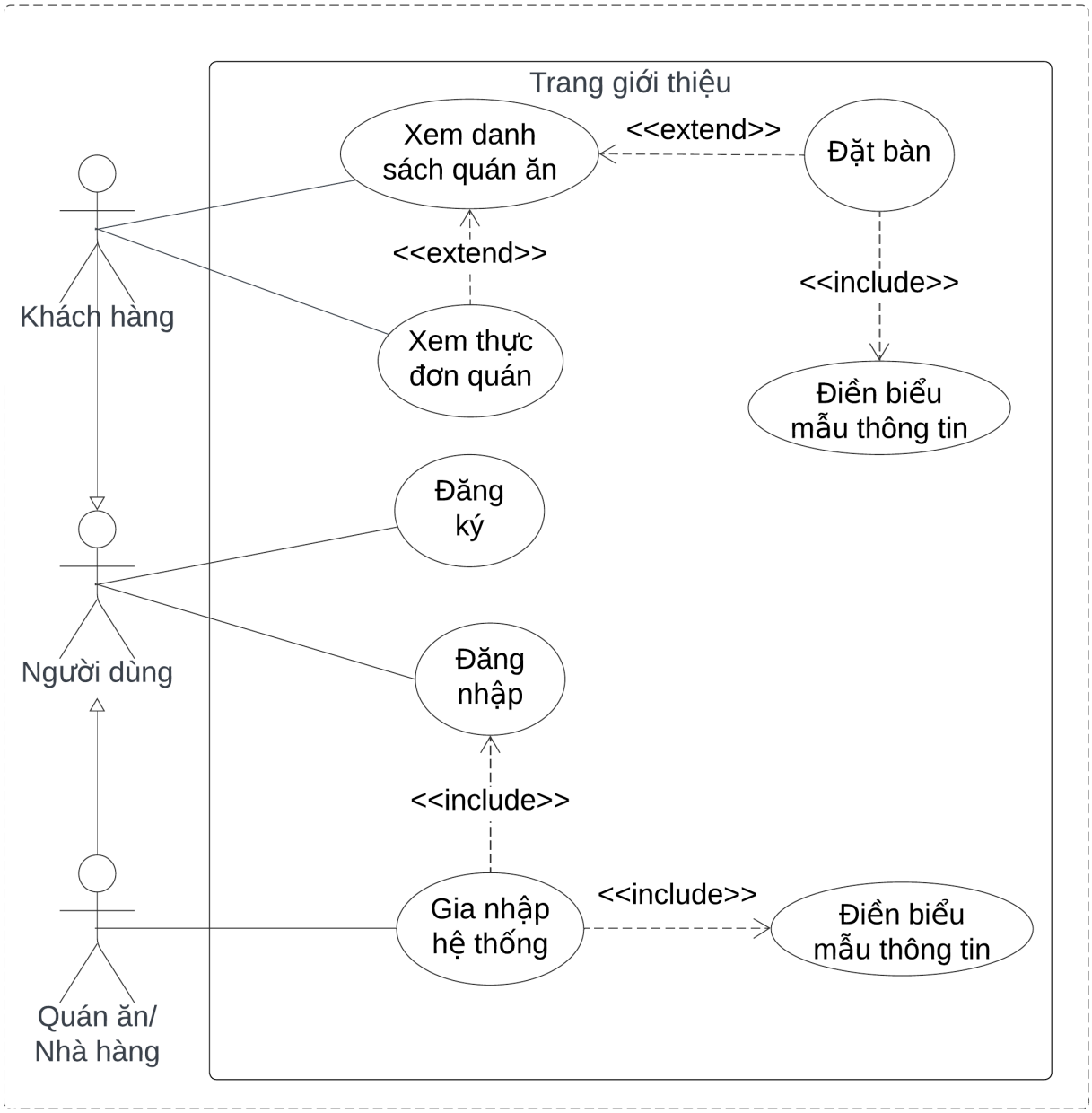
# Chương 1

# Giới thiệu hệ thống

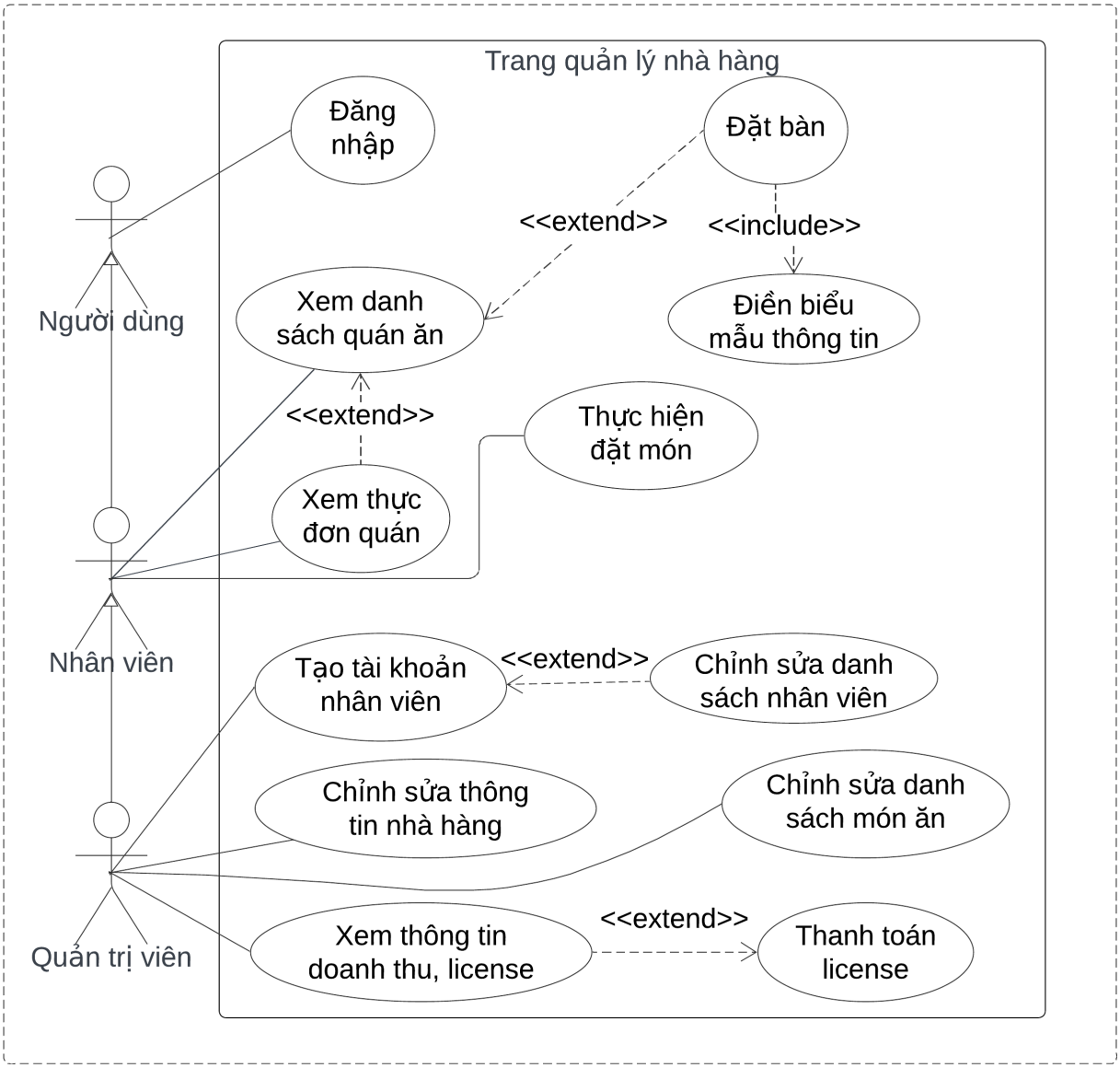
Trước khi đi sâu về thiết kế kiến trúc của hệ thống quản lý quán ăn và các dịch vụ liên quan, chương này sẽ trình bày các chức năng chính của hệ thống theo thiết kế cũng như là đi qua luồng nghiệp vụ của một số tính năng cụ thể.

## Các chức năng chính

Mô hình quản lý quán ăn được phát triển trong khóa luận này sẽ tập trung vào các chức năng chính giúp nhà hàng, quán ăn quản lý thực đơn, khách và bàn, v.v. Từ các chức năng chính như quản lý thực đơn sẽ có các chức năng nhỏ hơn bổ trợ được gọi là các tính năng phụ. Ví dụ như đối với chức năng quản lý bàn, ngoài các chức năng chính như tạo bàn mới, xóa bàn, thay đổi trạng thái của bàn, v.v., các nhánh chức năng phụ còn bao gồm các chức năng như ghép bàn, phân bàn cho những người đặt bàn trước, v.v. Danh sách các chức năng chính của hệ thống quản lý quán ăn được mô tả tại Hình 1.1 và Hình 1.2 bao gồm đặt bàn trực tuyến, gọi món thông qua việc quét mã QR, thanh toán hóa đơn, v.v.




Hình 1.1: Biểu đồ ca sử dụng cho trang giới thiệu `landing-page`




Hình 1.2: Biểu đồ ca sử dụng cho trang quản lý nhà hàng `admin-page`

Luồng đặt bàn trực tuyến và đặt món qua QR sẽ được tập trung thảo luận và mô tả xuyên suốt các chương về sau. Chương này sẽ tập trung mô tả qua các tác nhân tham gia vào hệ thống và một luồng nghiệp vụ tổng quan giữa các bên tham gia. Mô tả kỹ thuật các lời gọi API cũng như là các tương tác giữa các vi dịch vụ với nhau sẽ được mô tả rõ hơn tại Mục 3.3

## Luồng đặt bàn trực tuyến

Luồng người dùng đặt bàn có thể được tương tác bởi cả người dùng đã đăng ký đăng nhập cũng như là người dùng vãng lai của hệ thống. Khách hàng có thể sử dụng chức

năng đặt bàn tại trang giới thiệu *landing-page* hoặc là trang giới thiệu *customer-page* của nhà hàng, quán ăn đã gia nhập vào nền tảng chung.

Bảng 1.1: Bảng mô tả luồng nghiệp vụ của chức năng đặt bàn trực tuyến

| STT | Màn hình | Hoạt động người dùng | Tương tác hệ thống |
|---|---|---|---|
| 1 | Trang giới thiệu | Mở ứng dụng/truy cập vào website | Hiển thị giao diện trang giới thiệu |
| 2 | Trang giới thiệu | Chọn chức năng đặt bàn | Chuyển hướng đến trang đặt bàn |
| 3 | Trang đặt bàn | Nhập các thông tin cần thiết | Kiểm tra tính khả dụng của bàn |
| 4 | | | Hiển thị kết quả và trạng thái: bàn trống hoặc không có bàn |
| 5 | | Nếu có bàn trống: Chọn bàn, xác nhận đặt bàn | |
| 6 | | | Ghi nhận thông tin đặt bàn, gửi thông báo xác nhận, cập nhật trạng thái bàn |
| 7 | Ứng dụng quản lý nhà hàng | | Hiện thị thông báo đặt bàn mới |
| 8 | | Nhân viên nhà hàng phân bàn và xác nhận với khách | |

Ở bước 8 Bảng 1.1, nhân viên xác nhận với khách hàng, điều này có thể được thực hiện thông qua nhân viên gọi điện hoặc gửi email xác nhận thông tin đặt bàn. Các trường thông tin của khách hàng sẽ được lấy từ thông tin tài khoản nếu khách hàng có đăng nhập vào hệ thống hoặc tại biểu mẫu được nhập khi yêu cầu đặt bàn của khách hàng mới.

## Luồng đặt món thông qua quét mã QR

Tương tự với Bảng 1.1, cả khách hàng vãng lai và khách hàng đã đăng ký đều có thể thực hiện luồng đặt món, luồng bắt đầu từ trang `customer-page` hoặc trang giới thiệu `landing-page`.

Với tính năng đặt món, tạo đơn hàng mới, v.v., không chỉ có khách hàng mới sử dụng các tính năng đó mà các tính năng này cũng được tích hợp để gọi từ phía quản lý nhà hàng bởi các nhân viên của cửa hàng. Điều này giúp cho nhân viên có thể thay mặt khách hàng thực hiện các thao tác thay họ. Ví dụ đối với chức năng đặt món qua việc

Bảng 1.2: Bảng mô tả luồng nghiệp vụ của chức năng đặt món qua QR

| STT | Màn hình | Hoạt động người dùng | Tương tác hệ thống |
|---|---|---|---|
| 1 | Mở ứng dụng quét mã QR | Quét mã QR gọi món (có thể ở trên bàn hoặc từ lúc đặt bàn) | API tạo đơn hàng lấy thông tin đặt bàn (nếu có), kiểm tra trạng thái bàn, gửi thông báo đơn hàng mới |
| 2 | Trang chọn món ăn | Chọn món ăn từ thực đơn | API lấy thông tin chi tiết món ăn, tính giá tiền, trả về thông tin order, gửi thông báo mới về món ăn đã thêm |
| 3 | | Xác nhận danh sách món | API xác nhận gọi món, chuyển trạng thái đơn sang đang xử lý, gửi thông báo mới về trang quản lý nhà hàng |
| 4 | Ứng dụng quản lý nhà hàng | | Nhân viên nhận thông báo đơn hàng mới, chuyển trạng thái món ăn qua từng giai đoạn của món, gửi thông báo cho người dùng |
| 5 | Trang chọn món ăn | | Hiển thị thông báo cho các món ăn |

quét mã QR, nếu khách hàng chẳng may không mang thiết bị di động bên người hoặc khách hàng không truy cập được mạng, nhân viên hoàn toàn có thể thay mặt khách hàng quét QR và tạo đơn hàng cho bàn của khách.

Tuy nhiên vẫn sẽ có trường thông tin trong cơ sở dữ liệu giúp nhận biết nhân viên phụ trách cho việc tạo đơn hàng giúp nhận biết, kiểm soát xem nhân viên có tạo đơn hàng hay không nhằm tránh các tình huống không mong muốn phát sinh. Ngoài ra sau khi đơn hàng được hoàn thành, thực khách của quán cũng có lựa chọn đánh giá dịch vụ của quán ăn, nhà hàng giúp hỗ trợ tăng cường quảng bá nhà hàng trên trang giới thiệu của nền tảng. Chi tiết về các trường trong cơ sở dữ liệu thuộc từng dịch vụ trong hệ thống quản lý nhà hàng được đề cập ở Mục 2.2.

Yên cái phi chuẩ này ?

# Chương 2

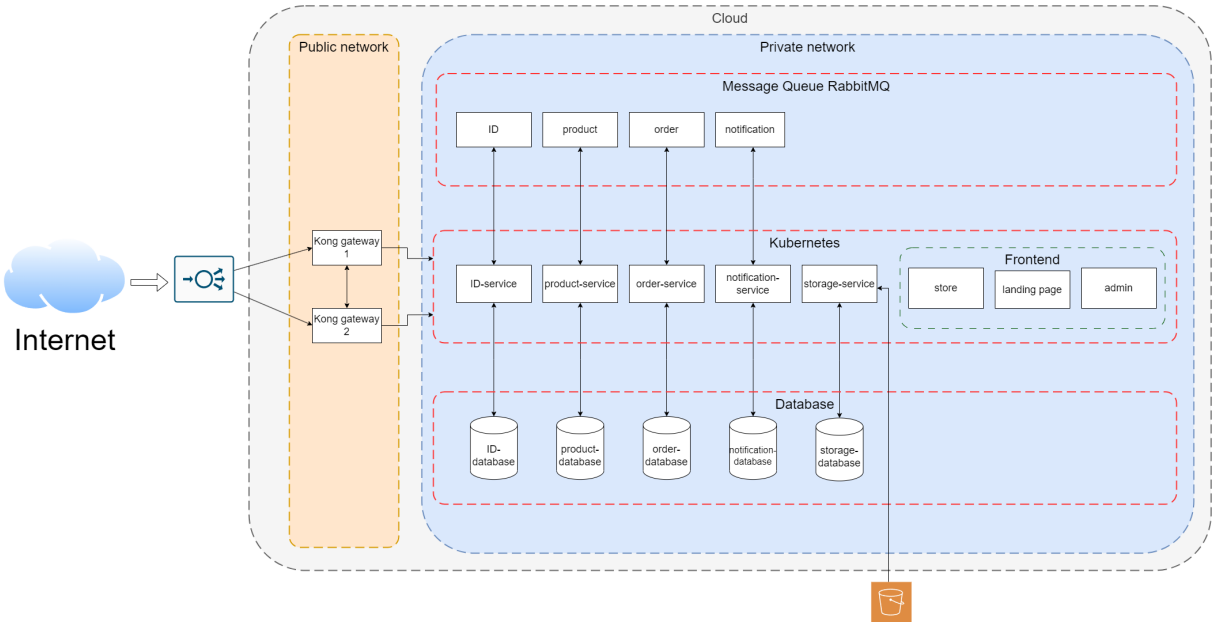
# Thiết kế hệ thống

## 2.1. Kiến trúc hệ thống

Nền tảng hạ tầng của ứng dụng quản lý quán ăn được thiết kế giúp hướng đến tính ổn định cao và phát triển cho sau này. Khi hệ thống đạt một số lượng người dùng nhất định, và có thể những người dùng này trải dài khắp nơi thay vì tập trung chủ yếu tại một địa điểm cố định. Khi đó, việc duy trì một máy chủ vật lý sẽ không còn là lựa chọn tối ưu do sự tốn kém về nhân lực cũng như chi phí phát sinh trong việc bảo trì và nâng cấp cho máy chủ. Từ đó mô hình kiến trúc của dự án hướng tới triển khai trên môi trường đám mây, tức sử dụng máy chủ của các nhà cung cấp dịch vụ đám mây rải rác ở khắp nơi trên thế giới. Điều này sẽ giúp giảm đáng kể thời gian cấu hình máy chủ vật lý của hệ thống cũng như thời gian cần phải bỏ ra giúp duy trì và bảo dưỡng do các tác nhân bên ngoài gây nên.

Luồng dữ liệu đi vào hệ thống bắt nguồn từ ngoài Internet, nơi tất cả các yêu cầu của người dùng đi qua một GLSB (Global Server Load Balancers). GSLB của Cloudflare giúp phân phối người dùng ở bất cứ nơi nào trên thế giới đến máy chủ khỏe mạnh gần nhất đối với họ. Điều này sẽ giúp tối ưu hóa thời gian tải trang, giảm thiểu độ trễ, và đảm bảo trải nghiệm truy cập website của người dùng nhanh chóng, mượt mà nhất. Ngoài ra Cloudflare còn đóng vai trò như một hệ thống phân giải tên miền (DNS).

Khi người dùng truy cập vào trang giới thiệu hoặc trang quản lý nhà hàng lần đầu tiên trên trình duyệt web, trình duyệt sẽ gửi truy vấn tới máy chủ DNS cục bộ (resolver), thông thường sẽ là máy chủ của nhà mạng. Máy chủ DNS cục bộ sẽ kiểm tra liệu tên




Hình 2.1: Kiến trúc tổng quan của hệ thống quản lý quán ăn

miền đang được yêu cầu có địa chỉ IP tương ứng không, nếu không tìm thấy máy chủ DNS của nhà mạng sẽ thực hiện một lời gọi khác tới máy chủ DNS tại các cấp cao hơn. Ở đây cấp độ ngay trên máy chủ cục bộ của nhà mạng sẽ là các máy chủ DNS nơi máy chủ được triển khai như Cloudflare hoặc Google Cloud. Lời gọi sẽ được chuyển đến DNS của nhà cung cấp mạng của người dùng và sau đó là đến Cloudflare DNS. Cloudflare sẽ nhận lời gọi của người dùng với tên miền của hệ thống đã được đăng ký trên Namecheap[1] và người dùng đến địa chỉ IP (Internet Protocol) của máy chủ phù hợp nhất với người gọi.

Bên trong mỗi máy chủ được triển khai trên Google Cloud sẽ có hai lớp mạng là mạng công cộng được phơi ra ngoài Internet và lớp mạng nội bộ nơi triển khai các thành phần chức năng chính. Tại lớp mạng công cộng triển khai hai cổng API (API Gateway) phụ trợ cho nhau. Ngoài những lợi ích chính của một cổng API đó là giúp thống nhất một điểm truy cập đến máy chủ thông qua API Gateway và cải thiện hiệu suất hệ thống nhờ cơ chế cache kết quả truy vấn API cũng như là giới hạn lưu lượng truy cập tránh quá tải, việc sử dụng song song hai cổng API cũng giúp phân phối lưu lượng truy cập giữa các cổng và giúp đảm bảo tính sẵn sàng cao.

Các cổng API sau đó sẽ chuyển tiếp lời gọi từ ngoài Internet đến hệ thống mạng

[1]https://www.namecheap.com/

nội bộ của máy chủ, nơi sẽ triển khai các thành phần chính của mô hình quản lý quán ăn. Hệ thống ở đây được chia thành ba lớp chính bao gồm lớp hàng đợi tin nhắn sử dụng RabbitMQ, lớp K8s bao gồm các dịch vụ phụ trách xử lý lời gọi của người dùng, và cuối cùng là lớp cơ sở dữ liệu lưu trữ các thông tin quan trọng như dữ liệu của người dùng, nhà hàng, quán ăn, v.v. Hệ thống được thiết kế theo hướng kiến trúc vi dịch vụ tức từng thành phần, chức năng chính sẽ được triển khai và hoạt động trên các môi trường độc lập với nhau giúp tăng cường khả năng chịu lỗi, bảo trì dễ dàng, và tiện lợi trong việc nâng cấp, mở rộng sau này. Mỗi vi dịch vụ có trách nhiệm thực hiện một chức năng cụ thể, ví dụ như dịch vụ thông báo sẽ hoạt động với mục đích duy nhất là nhận thông tin được đẩy đến từ một hoặc nhiều dịch vụ cụ thể và phân phối nó đến với các dịch vụ đã đăng ký nhận thông báo từ các dịch vụ đó. Việc chia nhỏ hệ thống thành các dịch vụ độc lập mang lại nhiều lợi ích như giúp dễ dàng triển khai và bảo trì cho hệ thống, tăng tính sẵn sàng cũng như là khả năng chịu lỗi, giúp tự động mở rộng hệ thống. Về mặt phát triển hệ thống, bởi vì mỗi vi dịch vụ đều hoàn toàn độc lập với nhau nên các đội ngũ phát triển của từng chức năng có thể chọn bộ công nghệ phù hợp nhất với nhu cầu và khả năng của đội.

Bằng việc triển khai nền tảng quản lý quán ăn trên Kubernetes (K8s) và Google Cloud Platform (GCP), nhà phát triển sẽ tránh được việc phải cấu hình thủ công cho từng dịch vụ K8s và giảm tải cho quản trị viên hệ thống trong quá trình vận hành và quản lý ứng dụng. Các ưu điểm công nghệ của K8s sẽ được đề cập rõ hơn trong Mục Công nghệ sử dụng.

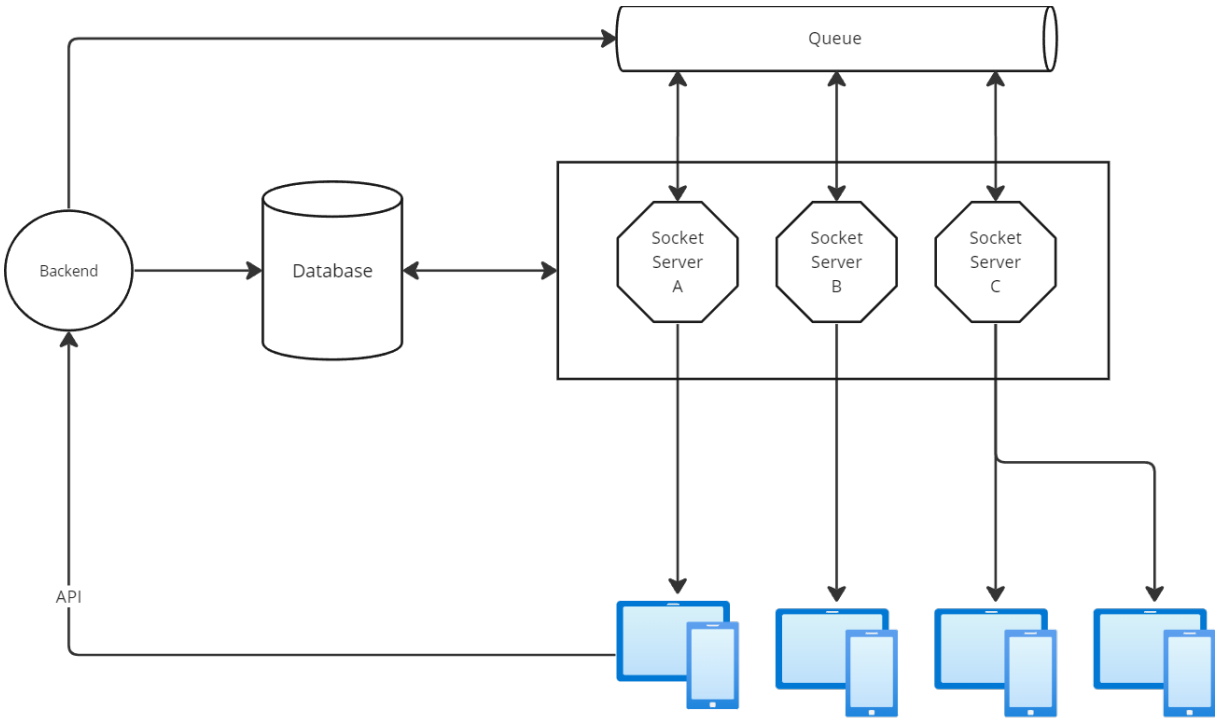
GCP là một nền tảng dịch vụ điện toán đám mây với gói chức năng đa dạng kèm theo các công cụ hỗ trợ tốt cho hệ thống với kiến trúc vi dịch vụ giúp dễ dàng triển khai và vận hành. Có thể kể đến nền tảng GKE giúp người dùng cấu hình K8s trên GCP một cách dễ dàng, Firebase giúp bảo mật, xác thực người dùng truy cập vào hệ thống thông qua việc tạo lập và quản lý tài khoản cả người dùng. GCP cũng hỗ trợ người sử dụng tránh được các vấn đề liên quan đến cấu hình cơ sở hạ tầng của hệ thống. Với các tùy chọn đa dạng, người dùng có thể dựa theo nhu cầu thực tế tại từng giai đoạn và có thể đưa ra các lựa chọn cài đặt cấu hình hợp lý.

Tất cả thông tin của người dùng cũng như là nhà hàng, quán ăn tham gia vào nền tảng đều sẽ được lưu trong cơ sở dữ liệu của hệ thống. Các thông tin này có thể là về những lần đặt bàn, thực đơn của quán ăn, những lần gọi món của người dùng, thông tin

người dùng, v.v., hoặc thông tin của hệ thống như các chỉ số sức khỏe, các bản ghi hoạt động, v.v. Nền tảng quản lý quán ăn hiện tại sử dụng MongoDB[2], một cơ sở dữ liệu dạng NoSQL. Chi tiết về MongoDB sẽ được nói rõ hơn trong Mục Công nghệ sử dụng và thiết kế chi tiết của cơ sở dữ liệu tại Mục Cấu trúc cơ sở dữ liệu.

## Giao tiếp giữa các vi dịch vụ

Trong các hệ thống quản lý nhà hàng, có những sự kiện từ phía khách hàng cần được tiếp nhận và xử lý trong thời gian thực ví dụ như tính năng đặt món, đặt bàn, thay đổi trạng thái đặt bàn. Một công nghệ phổ biến được sử dụng rộng rãi nhằm giao tiếp trong thời gian thực đó là Socket.IO.



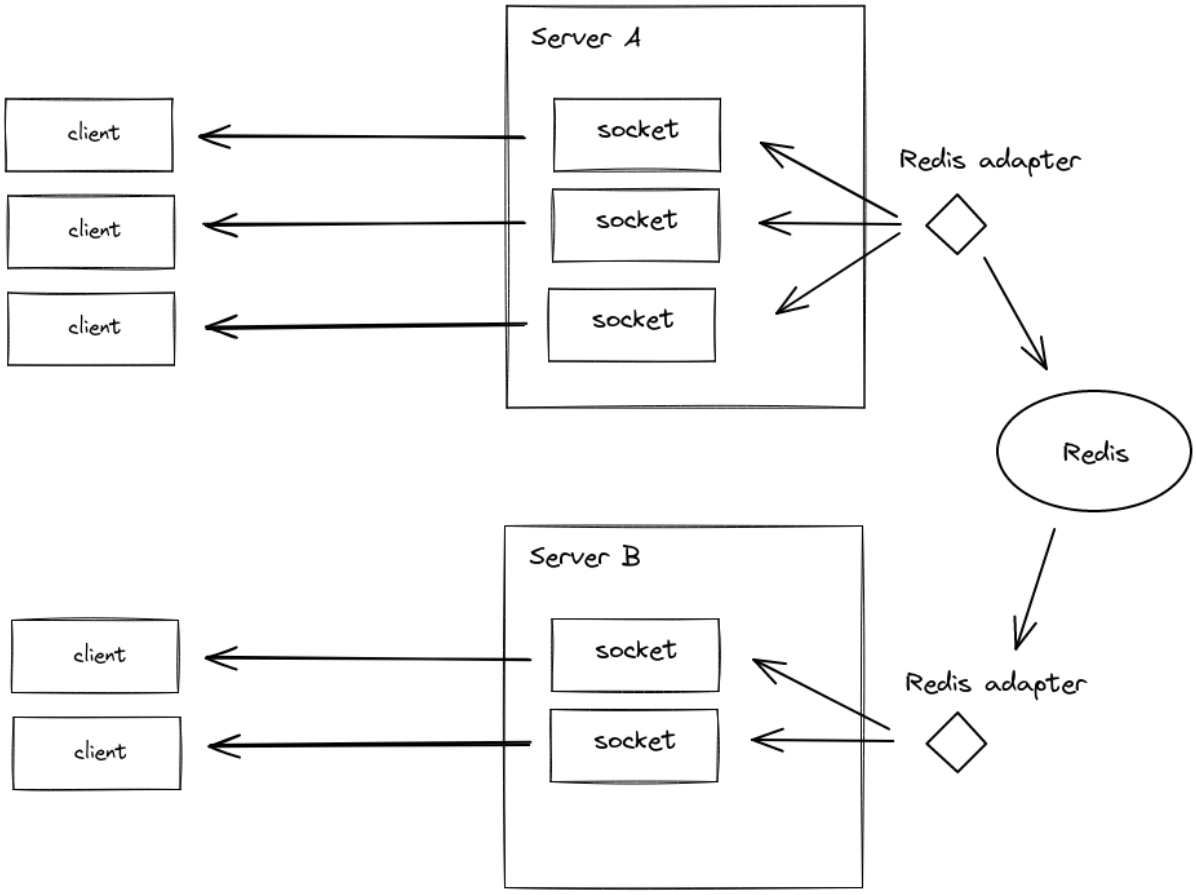

Hình 2.2: Kiến trúc tổng quan của hệ thống thông báo

Socket.IO là một thư viện JavaScript cho phép giao tiếp theo thời gian thực, hai chiều trên máy chủ và máy khách bằng cách thiết lập một kênh truyền dữ liệu (kết nối socket) giữa máy chủ và máy khách, cho phép truyền dữ liệu theo cả hai hướng. Áp dụng quy tắc đó ở các hệ thống lớn với kiến trúc vi dịch vụ, các vi dịch vụ sẽ mở các kết nối socket với nhau nhằm tạo ra một kênh liên lạc giúp thông báo các sự kiện từ phía người

[2]https://www.mongodb.com/

dùng đến các vi dịch vụ khác để xử lý nó.

Khi sử dụng Socket.IO, kiến trúc của socket sẽ có thêm một adapter đi kèm như trên Hình 2.3. Tuy nhiên đối với mô hình này, khi hệ thống có mở rộng và có nhiều máy chủ hơn, máy chủ được sử dụng làm adpter (ở đây là Redis) sẽ cần phải liên tục gửi yêu cầu đến tất cả các adapter còn lại mỗi khi một máy chủ gửi thông tin đến do máy chủ gửi tin không hề biết là máy khách cần nhận tin đang kết nối tới máy chủ nào. Điều này dẫn đến giảm tải hiệu năng của đáng kể khi mở rộng hệ thống do các kết nối không cần thiết giữa các máy chủ với nhau.




Hình 2.3: Kiến trúc của socket adapter [3]

Ngoài ra, cơ chế Publisher/Subscriber (Pub/Sub) giúp gửi và nhận dữ liệu giữa các máy chủ thông qua socket cũng mang một hạn chế đó là máy chủ gửi dữ liệu đi không có cơ chế đảm bảo dữ liệu đến được với máy nhận.

Cho nên để đảm bảo tin nhắn được gửi đến đúng máy khách, thiết kế của hệ thống

[3]https://socket.io/docs/v4/redis-adapter/

thông báo đã được thay đổi, tận dụng thêm một cơ sở dữ liệu giúp lưu trữ thông tin của các máy chủ và máy khách kết nối với nhau. Điều này giúp tránh việc phải gọi đến mọi máy sub khi một máy gửi tin nhắn lên do cơ sở dữ liệu lưu trữ mọi hoạt động kết nối và ngắt kết nối giữa máy khách và các máy chủ socket. Thiết kế chi tiết về cơ sở dữ liệu của hệ thống quản lý tin nhắn áp dụng với hệ thống thông báo của nền tảng quản lý quán ăn sẽ được đề cập trong Mục Cấu trúc cơ sở dữ liệu.

Ngoài ra trong Hình 2.2, hệ thống cũng sử dụng một hàng đợi, ở đây là RabbitMQ nhằm cân bằng tải tới các máy chủ socket, phân loại và xử lý tin nhắn giúp chuyển tiếp tin nhắn đến đúng máy khách cần nhận tin. Ngoài ra việc sử dụng hàng đợi tin nhắn (message queue) còn hỗ trợ lưu trữ tin nhắn khi máy khách ngắt kết nối khỏi hệ thống thông báo và đợi cho đến khi máy khách kết nối trở lại nhằm thực hiện lại tác vụ.

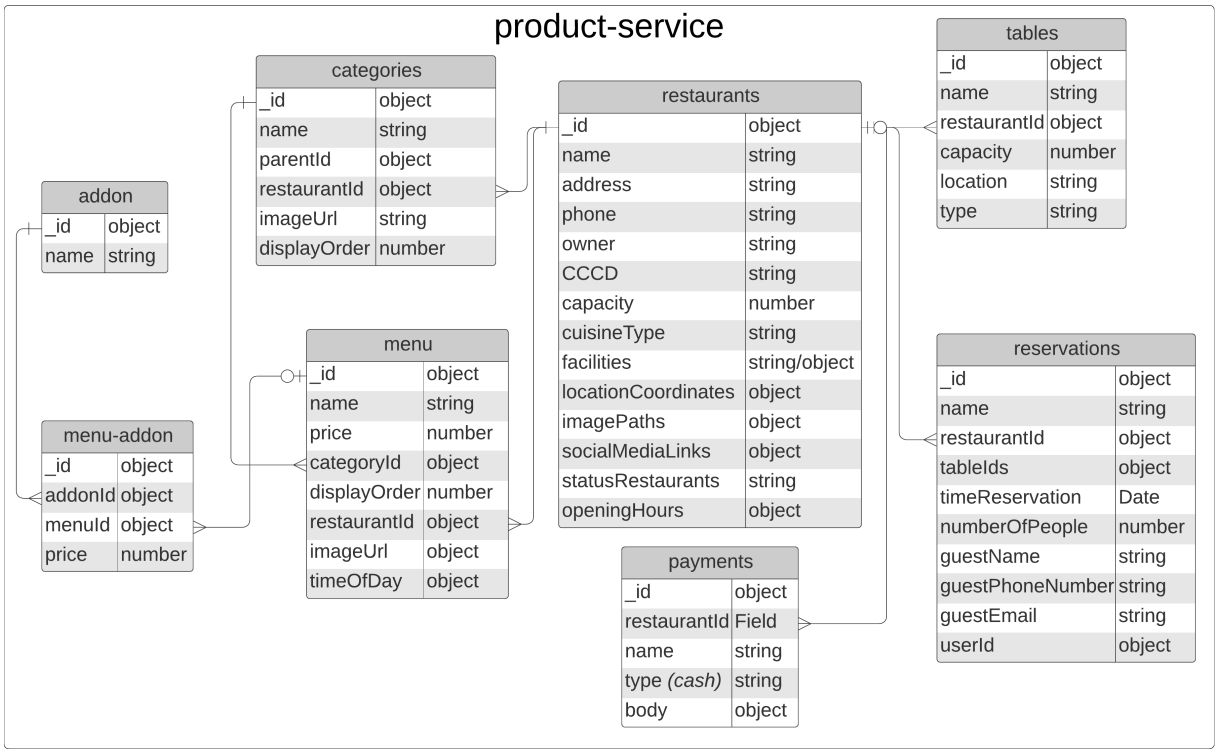
## 2.2. Cấu trúc cơ sở dữ liệu

Tại bất kỳ dự án phát triển phần mềm nào, có một giai đoạn đóng vai trò không thể thiếu đó là thiết kế cơ sở dữ liệu. Việc thiết kế cơ sở dữ liệu sẽ bắt dầu trước mọi quá trình xây dựng sản phẩm, giai đoạn này sẽ đi cùng với giai đoạn thiết kế các màn hình, các ca sử dụng của hệ thống. Đối với nền tảng quản lý quán ăn, tổng cộng có bốn cơ sở dữ liệu trong đó đối mỗi cơ sở dữ liệu sẽ đảm nhiệm vai trò lưu trữ cho từng vi dịch vụ trong hệ thống. Mỗi cơ sở dữ liệu sẽ có các trường được thiết kế đảm bảo việc lưu trữ các trường thông tin phục vụ cho luồng nghiệp vụ.

### 2.2.1. Dịch vụ quản lý cửa hàng

Cơ sở dữ liệu của dịch vụ quản lý cửa hàng được thiết kế với mục đích quản lý thông tin về sản phẩm, các loại dịch vụ và hoạt động chung của nhà hàng, quán ăn vừa và nhỏ. Trong đó, các trường thông tin được tổ chức thành các bảng chính như Hình 2.4.

Các bảng tại Hình 2.4 bao gồm bảng `restaurants` lưu thông tin của nhà hàng như tên, địa chỉ, số điện thoại, thông tin liên hệ của chủ sở hữu, v.v. giúp khách hàng có thể dễ dàng tìm kiếm, liên hệ khi cần đặt bàn, trợ giúp. Tại đây, một nhà hàng sẽ bao gồm nhiều thông tin khác như thông tin về đặt bàn được lưu trữ ở bảng `reservations` bao gồm tên khách hàng, số điện thoại, thời gian đặt bàn, v.v., thông tin về các bàn của quán ăn nằm ở bảng `tables` lưu sức chứa và vị trí của bàn. Thực đơn của quán ăn sẽ được lưu

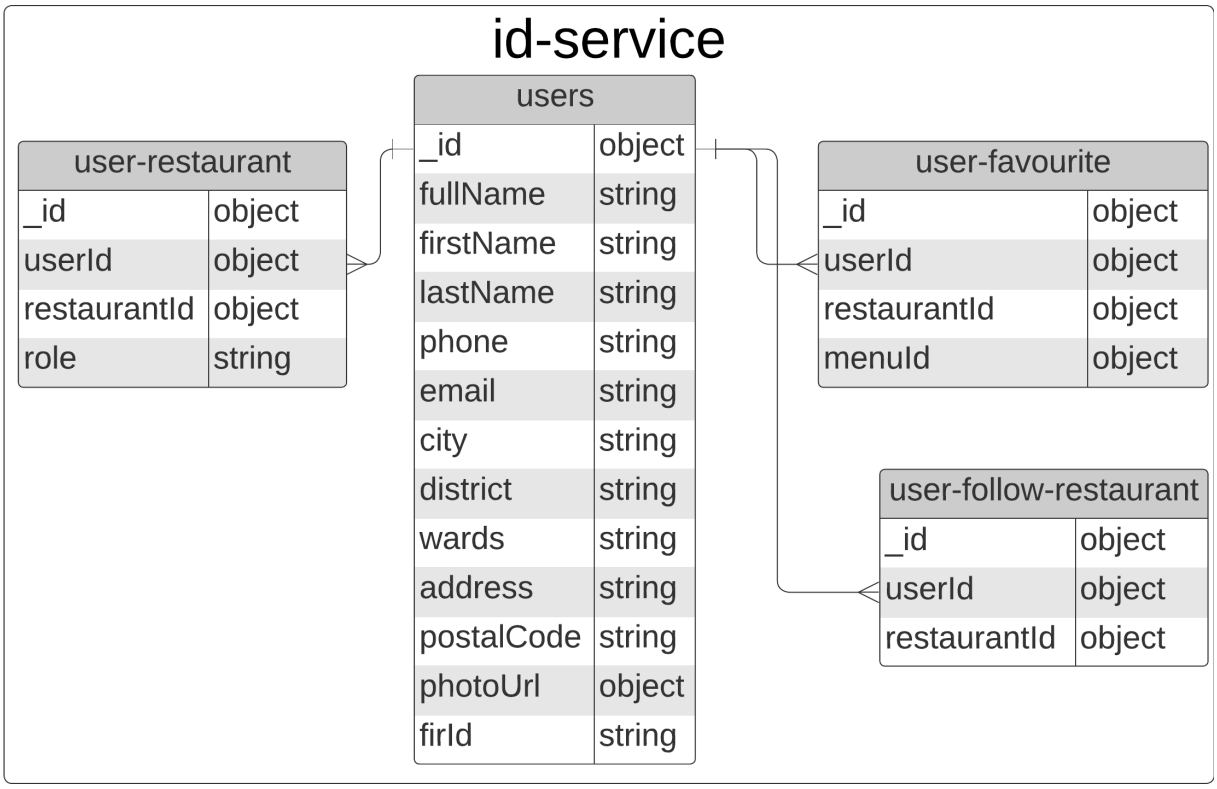



Hình 2.4: Thiết kế cơ sở dữ liệu cho dịch vụ quản lý cửa hàng

trong một tập hợp các bảng bao gồm bảng `menu` lưu các thông tin như tên, giá, loại món ăn cũng như là thời gian bán của món trong ngày. Mỗi món sẽ được phân loại vào một bảng `categories`, bảng này bao gồm trường `parentId` hỗ trợ lồng các thể loại món với nhau giúp dễ dàng quản lý. Cuối cùng, bảng `payments` lưu trữ thông tin về các phương thức thanh toán nhà hàng hỗ trợ như tiền mặt, chuyển khoản, v.v.

### 2.2.2. Dịch vụ quản lý định danh khách hàng

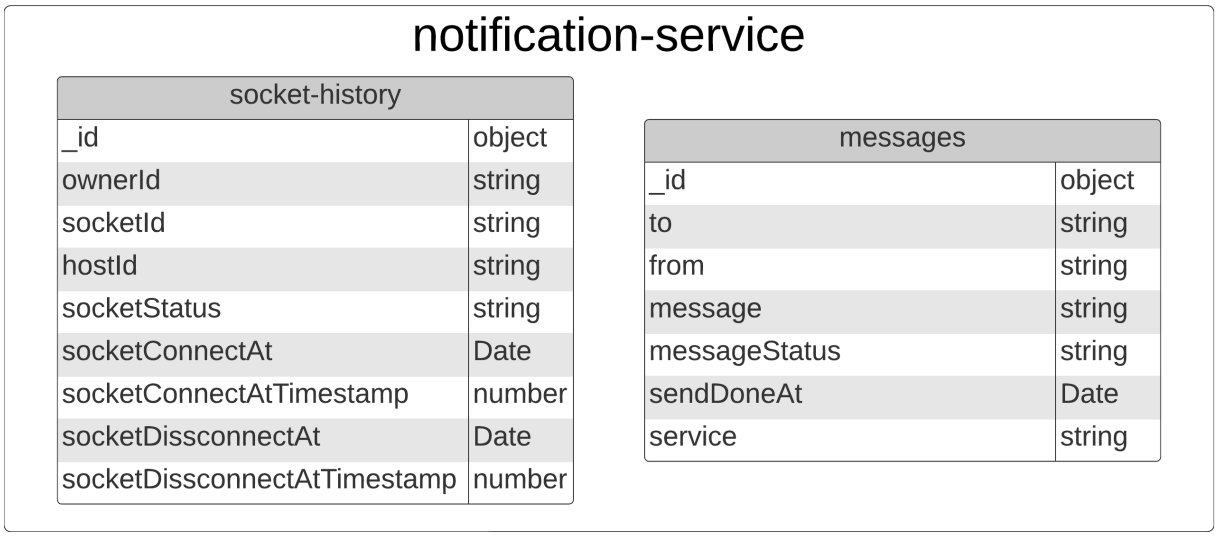
Cơ sở dữ liệu `id-service` được thiết kế nhằm quản lý thông tin của người dùng trong hệ thống và các mối quan hệ liên quan đến nhà hàng. Hình 2.5 mô tả kiến trúc cơ sở dữ liệu mô hình quản lý định danh khách hàng bao gồm bảng `user` lưu các thông tin cơ bản về người dùng bao gồm tên, tuổi, số điện thoại, địa chỉ, v.v. Ngoài thông tin của người dùng còn có các bảng lưu danh sách các nhà hàng người dùng đã từng đến ăn, nhà hàng người dùng đang theo dõi và các nhà hàng mà người dùng thích. Các trường thông tin tại bảng `users` được người dùng điền khi chỉnh sửa thông tin hồ sơ của bản thân hoặc trong trường hợp người dùng đăng nhập thông qua một bên thứ ba như Google, Facebook, hệ thống sẽ truy cập các thông tin công khai của người dùng trên nền tảng đó.




Hình 2.5: Thiết kế cơ sở dữ liệu cho dịch vụ quản lý định danh khách hàng

### 2.2.3. Dịch vụ thông báo

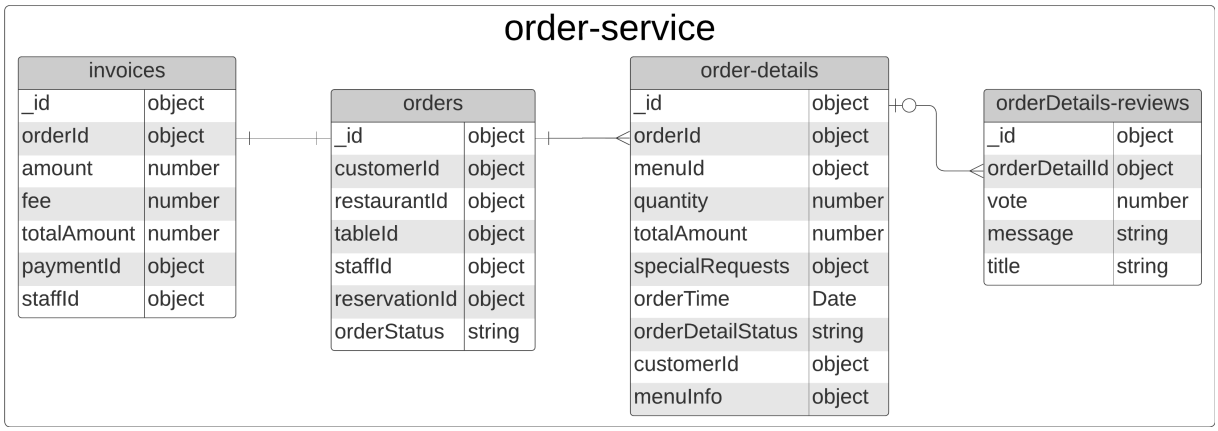
Mục đích của cơ sở dữ liệu `notification-service` là quản lý các thông báo, lời gọi giữa các dịch vụ với nhau thông qua socket trong hệ thống. Cơ sở dữ liệu này bao gồm 2 bảng là `socket-history` và `messages`. Trên Hình 2.6, bảng `socket-history` lưu trữ lịch sử kết nối socket, bao gồm thông tin về khách hàng `ownerId`, `socketId` lưu trữ thông tin của phiên kết nối đó, và các trường lưu thông tin của máy chủ cũng như trạng thái, thời gian máy khách kết nối, ngắt kết nối khỏi hệ thống.




Hình 2.6: Thiết kế cơ sở dữ liệu cho dịch vụ thông báo

### 2.2.4. Dịch vụ đặt bàn

Cơ sở dữ liệu `order-service` quản lý các khía cạnh trong quy trình đặt món và thanh toán trong hệ thống quản lý nhà hàng. Cơ sở dữ liệu cũng được thiết kế với cơ chế đánh giá giúp khách hàng có thể nêu lên ý kiến của họ với nhà hàng giúp nhà hàng nắm được cảm nhận của khách. Hình 2.7 bắt đầu từ bảng `orders` ở giữa lưu trữ thông tin tổng quan về mỗi đơn hàng bao gồm `customerId` (Id của khách hàng), `restaurantId` (Id nhà hàng), `tableId` (Id bàn ăn), người phục vụ của bàn `staffId` và thời gian cũng như trạng thái của đơn hàng. Bảng `order-details` đi sâu vào chi tiết từng món ăn trong mỗi đơn hàng bao gồm Id của món `menuId`, số lượng và tổng giá tiền cũng như là các tùy chọn khác tại trường `specialRequests` Để quản lý thông tin hóa đơn cho mỗi đơn hàng, bảng `invoices` sẽ lưu thông tin về giá trị đơn hàng, phí dịch vụ, thuế cũng như là người phụ trách thanh toán cho đơn hàng đó.




Hình 2.7: Thiết kế cơ sở dữ liệu cho dịch vụ đặt bàn

## 2.3. Luồng người dùng

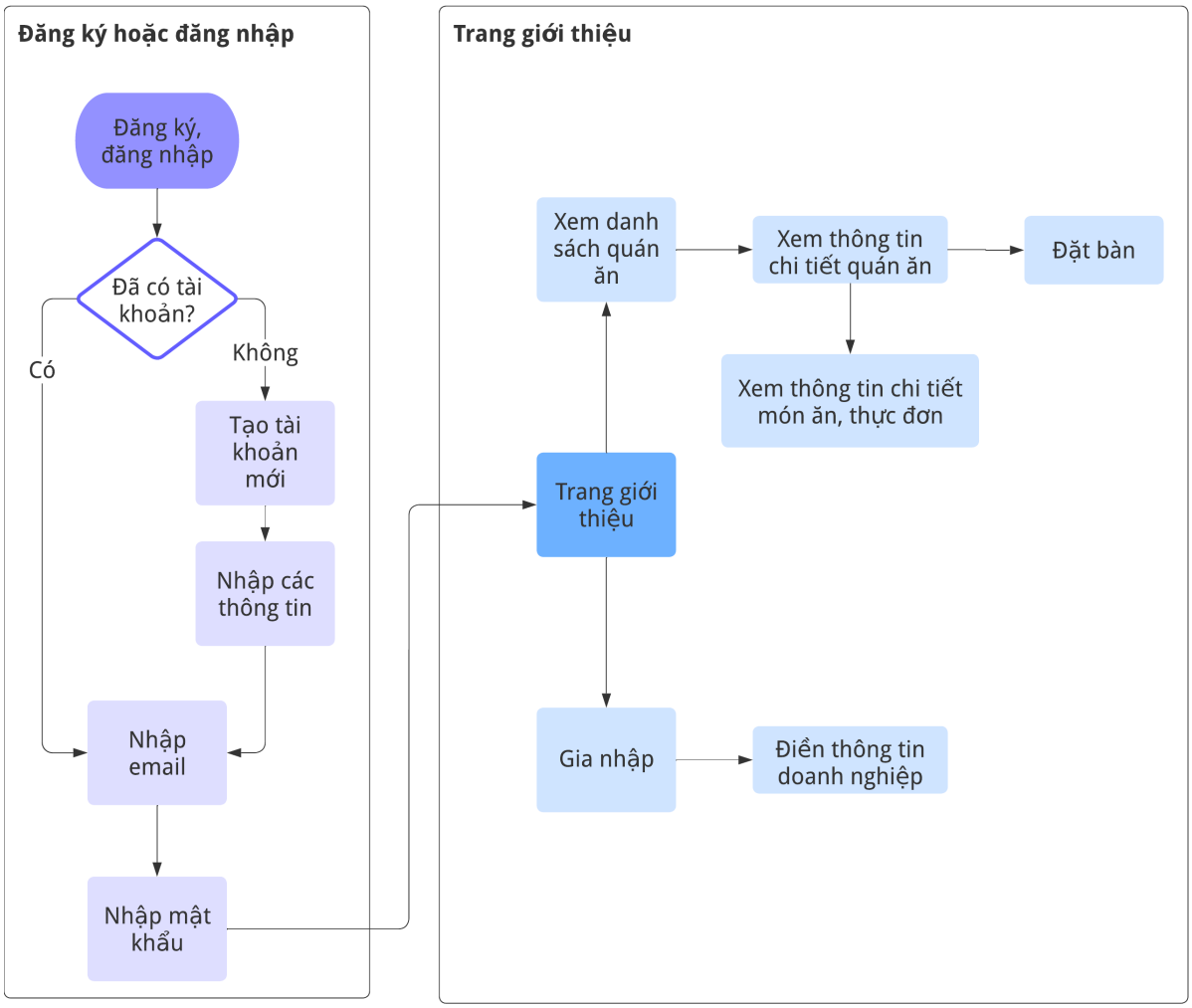
Thiết kế luồng người dùng (use journey) là một quá trình trực quan hóa quan giúp đội ngũ phát triển hiểu rõ hành vi, nhu cầu của người dùng khi tương tác với giao diện sản phẩm hệ thống. Việc xây dựng luồng người dùng đầy đủ và chi tiết cho phép người xem có cái nhìn toàn diện, đẩy đủ nhất về trải nghiệm của người dùng (UX) từ lúc bắt đầu tương tác với hệ thống cho đến khi hoàn thành chức năng đề ra, từ đó đưa giúp nhà phát triển ra quyết định điều chỉnh thiết kế phù hợp và tối ưu hóa trải nghiệm người dùng. Luồng người dùng thường được thể hiện dưới dạng sơ đồ, mô tả các bước mà người dùng thực hiện, các hành động của họ trong từng giai đoạn. Đối với hệ thống quản lý quán ăn, sẽ có ba giao diện chính đó là giao diện trang giới thiệu hệ thống `landing-page`, trang quản lý nhà hàng `admin-page` và trang giao diện cho cửa hàng `customer-page`. Trang `customer-page` sẽ không được mô tả luồng người dùng trong Mục này do nhu cầu tính năng và giao diện của giao diện cửa hàng có thể tùy biến theo yêu cầu của nhà hàng, quán ăn khi họ sử dụng dịch vụ của nền tảng.

### 2.3.1. Trang giới thiệu

Luồng người dùng trên trang giới thiệu của hệ thống quản lý nhà hàng đi theo hai hướng chính đó là đặt bàn và gia nhập hệ thống với tư cách nhà hàng. Với tính năng đặt bàn, người dùng truy cập trang giới thiệu, nơi hiển thị danh sách các quán ăn, nhà hàng đã gia nhập nền tảng. Họ có thể xem danh sách quán, lựa chọn một quán ăn cụ thể để xem thông tin chi tiết cũng như là thực đơn và tiến hành đặt bàn. Quá trình đặt bàn không

yêu cầu người dùng cần phải đăng nhập để thực hiện hành động.

Nếu người dùng quyết định gia nhập hệ thống với tư cách là một chủ nhà hàng, quán ăn, họ sẽ cần phải đăng nhập vào hệ thống. Sau đó người dùng sẽ truy cập vào trang gia nhập và điền thông tin chi tiết về doanh nghiệp của họ. Thông tin người dùng nhập sau đó sẽ được xác thực bởi quản trị viên của hệ thống và sẽ liên lạc lại với người đăng ký để xác nhận thủ tục và cấp tài khoản vào hệ thống quản lý nhà hàng.

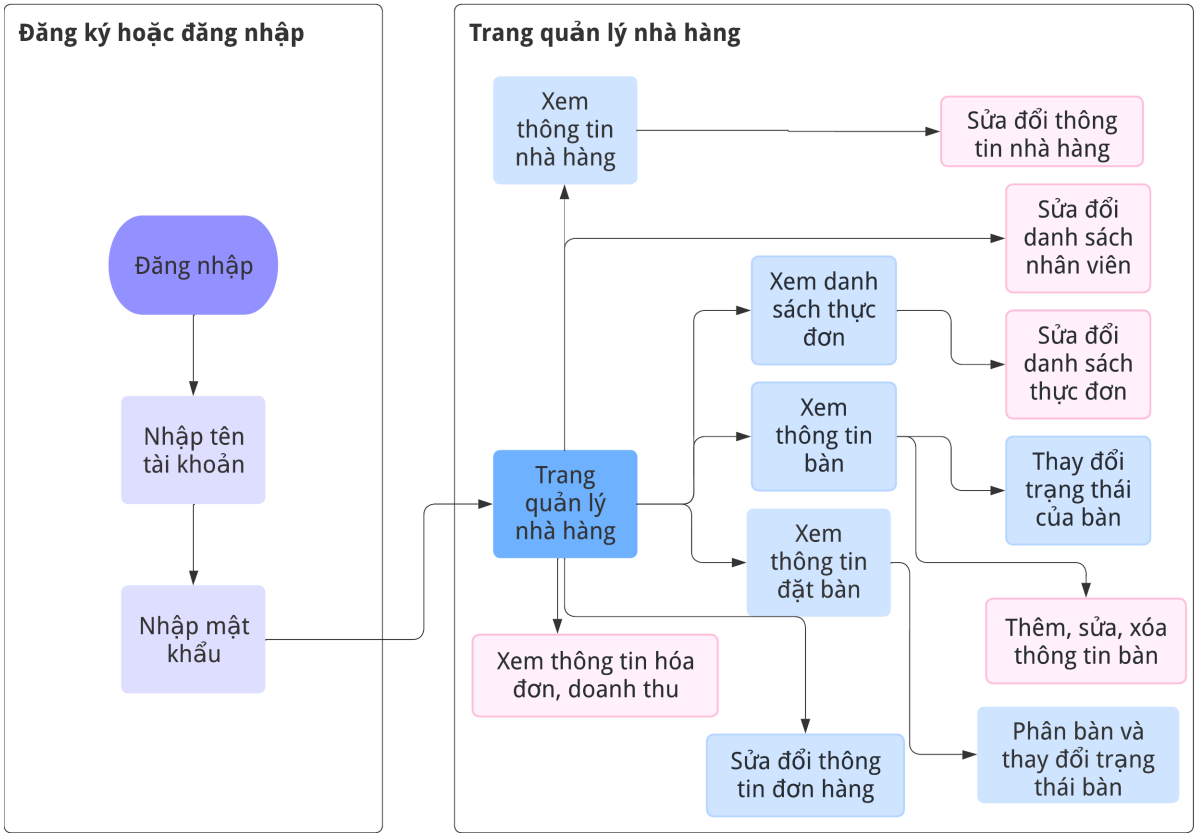



Hình 2.8: Luồng người dùng trang giới thiệu `landing-page`

### 2.3.2. Quản lý nhà hàng

Ở trang quản lý nhà hàng, người dùng không thể tự tạo tài khoản mà chỉ có thể đăng nhập dưới các tài khoản được tạo bởi quản trị viên của trang hoặc là quản trị viên hệ thống. Sau khi đăng nhập vào trang quản lý nhà hàng, các chức năng sẽ được chia ra

cho từng vai trò cụ thể. Nhân viên của nhà hàng, quán ăn được phép xem các thông tin của quán cũng như là thông tin về thực đơn, bàn, đặt bàn, cũng như là thông tin về hóa đơn, doanh thu. Nhân viên cũng có thể sửa đổi thông tin Đối với quản trị viên của trang tức quản lý của cửa hàng, họ có mọi quyền hạn của nhân viên và quản trị viên còn có thể sửa đổi thông tin của nhà hàng, sửa đổi danh sách nhân viên như thêm nhân viên mới, sửa đổi thông tin cá nhân của nhân viên. Ngoài ra quản lý của nhà hàng, quán ăn cũng có thể sửa đổi danh sách các món ăn đang bày bán tại cửa hàng cũng như thay đổi bố cục bàn của quán ăn.




Hình 2.9: Luồng người dùng trang quản lý quán ăn `admin-page`

# Chương 3

# Triển khai hệ thống

Ở Chương Thiết kế hệ thống khóa luận đã đề cập đến một số công nghệ được sử dụng trong quá trình thiết kế như Kubernetes, RabbitMQ, MongoDB, v.v. Mỗi công nghệ được lựa chọn và sử dụng trong quá trình phát triển hệ thống đều đóng một vai trò quan trọng giúp đảm bảo luồng nghiệp vụ phía người dùng diễn ra một cách mượt mà và toàn vẹn. Chương này sẽ đi chi tiết hơn vào diễn giải cách các công nghệ này tương tác, hoạt động với nhau, lợi ích của từng công nghệ cụ thể. Tiếp sau đó, Mục Quy trình triển khai sẽ trình bày cách triển khai hệ thống quản lý quán ăn lên môi trường đám mây của Google Cloud quá trình cài đặt HPA giúp hệ thống tự động mở rộng (autoscale) khi có lưu lượng truy cập lớn.

## 3.1. Công nghệ sử dụng

### 3.1.1. Kubernetes (K8s)

Đối với mô hình quản lý quán ăn tập trung, việc đảm bảo khả năng chịu lỗi và tính sẵn sàng cao khi các quán ăn sử dụng hệ thống là điều thiết yếu. Điều này đặc biệt đúng trong giờ cao điểm khi các nhà hàng, quán ăn đón một lượng lớn khách hàng đến quán. Từ đó khóa luận này đưa ra phương án thiết kế hệ thống đi theo mô hình hệ thống phân tán (distributed system) với kiến trúc vi dịch vụ sử dụng K8s nhằm đạt được những yêu cầu đề ra và hơn thế nữa.

Khi sử dụng mô hình phân tán, hệ thống sẽ tránh được việc có một điểm lỗi duy

nhất (single point of failure), khi một dịch vụ ngừng hoạt động, chỉ có riêng phần chức năng đó của hệ thống dừng phản hồi trong thời gian quản trị viên sửa chữa dịch vụ lỗi. Về phía khách hàng cũng như là quán ăn, trải nghiệm sử dụng của họ trên toàn bộ hệ thống sẽ dường như không thay đổi và đây là một ưu điểm lớn trong mô hình phân tán. Ngoài ra các hệ thống/dịch vụ độc lập với nhau thuộc hệ phân tán còn mang lại lợi ích cho khả năng mở rộng, phát triển nhờ sự tách biệt về mặt công nghệ. Các đội ngũ kỹ thuật hoàn toàn có thể phát triển hệ thống/dịch vụ của đội mà không gây ảnh hưởng đến các đội khác. Một ví dụ cho lợi ích này đó là dịch vụ thông báo. Như đã nói đề cập ở Mục Kiến trúc hệ thống, dịch vụ thông báo sẽ đóng vai trò giao tiếp với mọi dịch vụ khác trong hệ thống mỗi khi có một đơn đặt bàn mới, một lời gọi món mới, v.v. Khi vào giờ cao điểm, dịch vụ thông báo sẽ cần phải điều phối lượng thông tin của mọi dịch vụ khác gửi đến và có thể gây ra hiện tượng tắc nghẽn cho toàn bộ hệ thống nếu không mở rộng dịch vụ này. Nếu hệ thống được triển khai theo mô hình kiến trúc một khối (monolithic), việc nâng cấp chỉ riêng tính năng thông báo sẽ đồng nghĩa với việc phải nâng cấp cho toàn bộ máy chủ của toàn bộ tất cả các dịch vụ khác. Quá trình này không những khó khăn và tốn kém hơn mà còn khiến cho toàn bộ hệ thống chịu ảnh hưởng từ một dịch vụ. Bằng cách triển khai dịch vụ thông báo trên một máy chủ độc lập, ta sẽ loại bỏ được những phiền toái ban đầu liên quan đến phát triển, nâng cấp cũng như là bảo trì, sửa chữa.

Một trong những kiến trúc phổ biến nhất của mô hình hệ thống phân tán đó là kiến trúc vi dịch vụ. Các lợi ích chính của K8s có thể kể đến như là tự động hóa quá trình triển khai, mở rộng và quản lý các ứng dụng đóng gói (containerized). Các ứng dụng đóng gói trong ngữ cảnh này là từng dịch vụ riêng lẻ đảm nhiệm cho từng chức năng chính trong hệ thống kiến trúc vi dịch vụ. Trong quá trình phát triển phần mềm, mã nguồn luôn thay đổi và cập nhật liên tục để đáp ứng nhu cầu mới và sửa lỗi. Việc triển khai thủ công các phiên bản mới có thể tốn thời gian và dễ xảy ra sai sót. Vì vậy, việc sử dụng quy trình tự động triển khai của Kubernetes (K8s) là một giải pháp hợp lý.

Trong Kubernetes, việc triển khai ứng dụng được thực hiện trên một cụm (cluster), là một tập hợp các máy chủ (node) làm việc cùng nhau để cung cấp tài nguyên tính toán và lưu trữ [1]. Đơn vị triển khai cơ bản của K8s là Pod, thông thường bên trong một Pod sẽ chỉ có duy nhất một container [2] chính nhưng Pod cũng được thiết kế cho nhiều container

[1] https://kubernetes.io/docs/concepts/overview/components/

[2] https://kubernetes.io/docs/reference/glossary/?fundamental=true#term-container

với liên kết chặt chẽ với nhau để thực hiện một chức năng cụ thể [3]. Các Pod được quản lý bởi Deployment, chịu trách nhiệm duy trì số lượng pod mong muốn và đảm bảo chúng luôn hoạt động. Deployment được nhắc đến ở đây như một khái niệm trừu tượng, trên thực tế các Deployment không trực tiếp chạy trên bất kỳ máy chủ vật lý nào. Chúng hoạt động như một bản thiết kế, mô tả trạng thái mong muốn của ứng dụng, bao gồm số lượng bản sao pod, phiên bản image và các cấu hình khác.

Các Pod được triển khai trên Worker Node, là các máy chủ vật lý trong cluster K8s. Worker Node cung cấp tài nguyên tính toán và lưu trữ cho các Pod hoạt động. Ngoài ra ta cũng có Master Node chịu trách nhiệm quản lý toàn bộ cluster, bao gồm việc lên lịch triển khai Pod, giám sát trạng thái Pod và đảm bảo tính sẵn sàng của hệ thống. Trong một cụm hoàn toàn có thể có Master Node, tuy nhiên các Master Node sẽ tiến hành bầu cử để chọn ra một Master Node chính chịu trách nhiệm quản lý cụm và các Master Node còn lại sẽ đóng vai trò như bản sao của Node chính này. Quá trình bầu cử này dựa vào thuật toán đồng thuận phân tán như Raft[4] giúp đảm bảo tính công bằng, nhất quán và độ tin cậy của hệ thống.

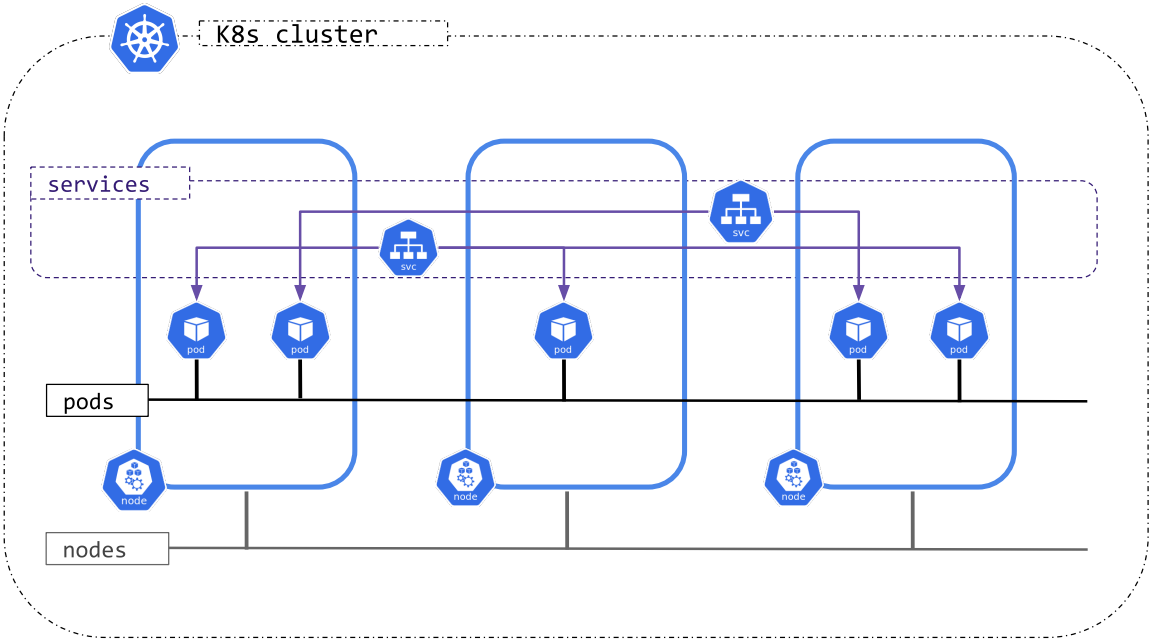
Bên trong một cụm (cluster) K8s, các Pod sẽ giao tiếp với nhau bằng mạng nội bộ. Như trên Hình 3.1, các Nodes tức các Worker Node và bên trong đó chứa các Pod/ứng dụng. Các Pod này được quản lý bởi các Service nằm trong cụm K8s đó. Mỗi Pod trong mạng sẽ được gán một địa chỉ IP (Internet Protocol) ảo và port (cổng) độc nhất trong mạng nội bộ bởi Service. Service hoạt động như một bộ cân bằng tải nội bộ và giúp quản lý các Pod trong hệ thống. Khi có yêu cầu đến Service, tùy thuộc vào cấu hình cân bằng tải, Master Node sẽ chỉ định một trong các Pod có trong Service xử lý yêu cầu đó.

Trước khi các yêu cầu từ bên ngoài đến được tới Service, K8s sử dụng Ingress để phân loại và điều hướng đến các Service phù hợp. Ingress hoạt động như Controller (bộ điều khiển) ở tầng ứng dụng, tiếp nhận các yêu cầu từ ngoài Internet và dựa theo cấu hình được định nghĩa trước để chuyển hướng các yêu cầu đó tới các Service phù hợp. Như trong Hình 3.2 mô tả cách client gọi đến Ingress từ ngoài Internet sau đó được điều hướng về các Service dựa theo HTTP Header `Host` được truyền vào. Bản thân Ingress cũng có thể đóng vai trò như một bộ cân bằng tải nhưng trong hệ thống khóa luận này

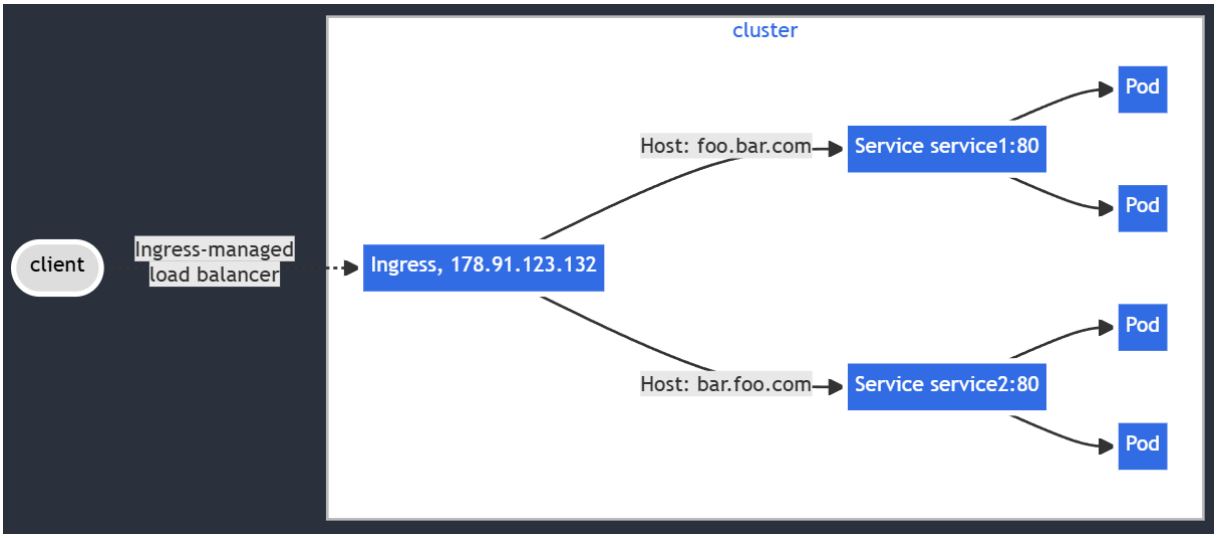
[3] https://kubernetes.io/docs/reference/glossary/?fundamental=true#term-pod

[4] Để giúp hiểu thêm về thuật toán đồng thuận phân tán, tham khảo trang http://thesecretlivesofdata.com/raft/

phát triển sẽ không dùng tới nó. Chi tiết về việc triển khai và các cấu hình của K8s trên hệ thống quản lý quán ăn sẽ được trình bày ở Mục Quy trình triển khai.




Hình 3.1: Các dải mạng khác nhau trong một cụm Kubernetes [5]




Hình 3.2: Hình minh họa phân bố yêu cầu gọi đến các Service trong cụm [6]

[5]https://kubernetes.io/docs/concepts/cluster-administration/networking/#kubernetes-ip-address-ranges
[6]https://kubernetes.io/docs/concepts/services-networking/ingress/#name-based-virtual-hosting

### 3.1.2. RabbitMQ

Đối với một hệ thống phân tán nói chung mà kiến trúc vi dịch vụ nói riêng, việc giao tiếp giữa các dịch vụ với nhau đóng vai trò then chốt. Khác với các ứng dụng theo kiến trúc một khối (monolithic) truyền thống khi mà các thành phần giao tiếp với nhau thông qua các lời gọi hàm hoặc chia sẻ bộ nhớ, các ứng dụng theo kiến trúc vi dịch vụ giao tiếp với nhau qua hai hình thức chính:

- Giao tiếp đồng bộ (Synchronous communication) là khi các dịch vụ gửi yêu cầu đến các dịch vụ đích và chờ phản hồi từ đó trước khi tiếp tục thực hiện tác vụ tiếp theo. Các giao thức phổ biến cho giao tiếp đồng bộ bao gồm RESTful API và gRPC.
- Giao tiếp bất đồng bộ (Asynchronous communication) khác với giao tiếp đồng bộ ở điểm dịch vụ gửi yêu cầu không cần chờ phản hồi ngay lập tức từ các dịch vụ đích. Điều này cho phép dịch vụ tiếp tục thực hiện các tác vụ khác trong khi chờ phản hồi. Giao tiếp không đồng bộ thường được thực hiện thông qua việc sử dụng hàng chờ tin nhắn (message queue).

Việc chọn ra một chuẩn giao tiếp phù hợp với ứng dụng trong quá trình thiết kế mang ý nghĩa quan trọng khi mà mỗi cách giao tiếp đều đi kèm những lợi ích và giới hạn riêng cho từng hệ thống cũng như là đội phát triển ứng dụng. Hệ thống quản lý quán ăn yêu cầu khả năng xử lý các yêu cầu từ khách hàng một cách nhanh chóng, hiệu quả, từ đó khi thiết kế hệ thống, hàng đợi tin nhắn như một lựa chọn tốt nhất với những yêu cầu nghiệp vụ nói trên. Điều này giúp các dịch vụ giảm sự phụ thuộc vào nhau khi chỉ cần biết địa chỉ của hàng đợi tin nhắn mà không cần biết địa chỉ của các dịch vụ khác trong cùng mạng. Hàng đợi tin nhắn cũng giúp tránh thất thoát dữ liệu, một trường hợp có thể xảy ra là khi một dịch vụ không hoạt động, các tin nhắn vẫn được lưu trữ ở trong hàng đợi và sẽ được xử lý một khi dịch vụ đó trở lại hoạt động.

Các công nghệ hàng đợi tin nhắn (message broker) phổ biến trên thị trường hiện nay có thể kể đến bao gồm Kafka, RabbitMQ, Amazon Simple Queue Service, v.v. Ở đây, RabbitMQ được lựa chọn nhờ sự đơn giản trong quá trình triển khai và cài đặt đơn giản hơn các dịch vụ khác. Chi tiết về cách hàng đợi tin nhắn được sử dụng trong hệ thống quản lý quán ăn được đề cập ở Mục Kiến trúc hệ thống.

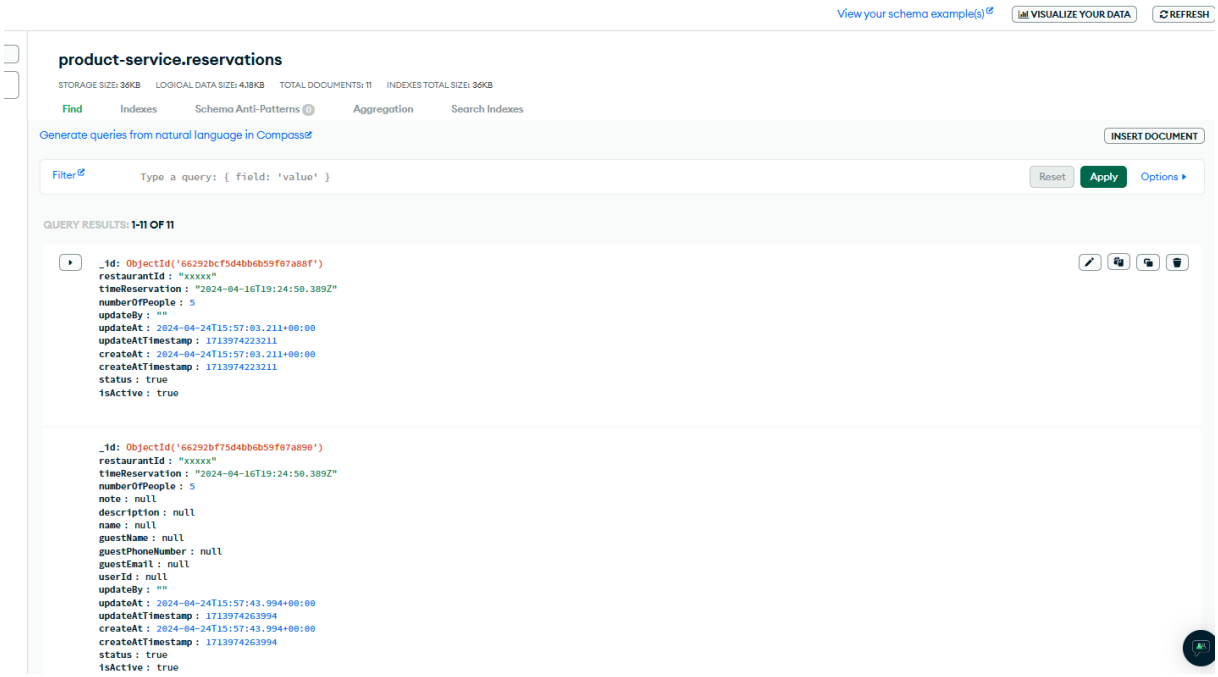
### 3.1.3. MongoDB

Trong hệ thống quản lý nhà hàng phức tạp, việc lưu trữ và truy vấn dữ liệu hiệu quả là rất quan trọng. Khác với các cơ sở dữ liệu truyền thống, MongoDB thuộc dạng cơ sở dữ liệu NoSQL, tức MongoDB có khả năng lưu trữ dữ liệu dạng tài liệu hay các dữ liệu phi cấu trúc một cách linh hoạt dưới dạng JSON. Điều này khiến cho cấu trúc bảng, dữ liệu có thể dễ dàng thích ứng với những thay đổi bất ngờ trong mô hình dữ liệu và yêu cầu nghiệp vụ. Trong MongoDB, khái niệm khóa chính (primary key) và khóa ngoại (foreign key) không tồn tại theo cách hiểu truyền thống như trong các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Tuy nhiên, MongoDB có những cơ chế riêng để đảm bảo tính toàn vẹn dữ liệu và tạo mối quan hệ giữa các bản ghi. Đối với khóa chính, MongoDB tự động tạo một trường `_id` cho mỗi tài liệu ngoại trừ những trường hợp người dùng chỉ định rõ. Trường này có giá trị là một `ObejctId` duy nhất trên toàn bộ cơ sở dữ liệu. Giá trị của `ObejctId` là một chuỗi `byte` dài 12 ký tự trong đó 4 `byte` mang giá trị là thời gian tạo `Id`, 5 `byte` là giá trị ngẫu nhiên được tạo ra dựa theo `Id` của tiến trình và của máy, cuối cùng là 3 `byte` được tạo với một bộ đếm tăng dần, bộ đếm này sẽ được khởi tạo với một giá trị ngẫu nhiên [7].

Ngoài ra, MongoDB còn có một phiên bản dịch vụ đám mây gọi là MongoDB Atlas, mang lại nhiều lợi ích như khả năng mở rộng tài nguyên tự động, sao lưu và phục hồi dữ liệu, giám sát và cảnh báo, giúp giảm thiểu công việc quản trị và đảm bảo hệ thống luôn sẵn sàng phục vụ.

Đối với một cơ sở dữ liệu thông thường, việc giám sát thông tin thường tập trung vào các loại như hiệu suất truy vấn, mức độ sử dụng tài nguyên, và trạng thái hoạt động. Các chức năng này đều được MongoDB Atlas cung cấp trên nền tảng website gồm các chức năng như ghi nhật ký, các cảnh báo tùy chỉnh theo nhu cầu người dùng, giám sát tài nguyên hoạt động theo thời gian thực, v.v. Từ đó khóa luận này quyết định sử dụng MongoDB cũng như là MongoDB Atlas làm cơ sở dữ liệu của hệ thống quản lý quán ăn. Về chi tiết về mặt thiết kế cơ sở dữ liệu của hệ thống quản lý nhà hàng, quán ăn được đề cập ở Mục Cấu trúc cơ sở dữ liệu.

[7] https://www.mongodb.com/docs/manual/reference/method/ObjectId/




Hình 3.3: Giao diện người dùng của MongoDB Atlas

### 3.1.4. Kong

Các hệ thống lớn hàng ngày sẽ phải tiếp nhận các lời gọi với tần suất rất lớn từ người dùng, đặc biệt là trong các giờ cao điểm trong ngày. Không những thế, những hệ thống này còn có thể phải đối mặt với nguy cơ bị DDOS (Distributed Denial-of-Service), tức khi hệ thống bị ngợp bởi một lượng truy cập lớn có thể gây ra tê liệt hệ thống. Cổng API (API Gateway) từ đây mang lại tác dụng như một điểm truy cập duy nhất cho ứng dụng, từ đó cấu hình chi tiết bên trong của nền tảng được ẩn dưới lớp API Gateway này. Một trong những cổng API phổ biến đó là Kong Gateway. Kong cung cấp các tiện ích khi sử dụng như chuyển đổi giao thức, đánh số, kiểm soát phiên bản API, giới hạn tốc độ truy cập và tích hợp các dịch vụ xác thực ở bên ngoài, v.v.

Khóa luận này thực hiện triển khai Kong Gateway trên một máy chủ thuộc GCP với Kong Manager. Kiến trúc nền tảng của công gồm 3 thành phần bao gồm bộ định tuyến (router), dịch vụ (service), và máy chủ đích (upstream). Khi nhận được yêu cầu từ ngoài vào, Kong Gateway sẽ khớp yêu cầu với Route phù hợp và chuyển tiếp yêu cầu đến Service tương ứng hoặc là tới các cân bằng tải đến cụm các Service trong Upstream. Nhằm quản lý Kong dễn dàng và hiệu quả hơn, khóa luận sử dụng Kong Manager. Kong Manager là bản thân Kong Gateway đã được trực quan hóa trên nền tảng web, điều này

giúp người dùng dễ dàng tạo, chỉnh sửa và xóa các Route, Service và Upstream, phát triển, cấu hình các plugin nhằm mở rộng chức năng của Kong Gateway, giám sát lưu lượng truy cập API và hiệu suất của Kong.

## 3.2. Quy trình triển khai

Trong bối cảnh phát triển các phần mềm hiện đại, khi mỗi dự án bao gồm nhiều đội ngũ phát triển phần mềm khác nhau với tần suất thay đổi mã nguồn ngày. Nhằm lưu trữ, theo dõi các thay đổi một cách hiệu quả cũng như tăng cường hiệu quả phối hợp giữa các thành viên trong nhóm phát triển, các phần mềm quản lý mã nguồn (VCS - Version Control System) dần trở thành công cụ không thể thiếu trong giai đoạn phát triển cũng như là triển khai hệ thống. Các VCS tiêu biểu có thể kể đến như là GitLab và GitHub, mặc dù cả hai nền tảng đều cung cấp đầy đủ các phương pháp quản lý mã nguồn hệ thống, cấu hình quyền người dùng linh hoạt và đều tích hợp các công cụ hỗ trợ kiểm thử hệ thống mạnh mẽ, GitLab nổi trội hơn GitHub về mặt tích hợp và triển khai liên tục (CI/CD) tốt hơn GitHub do cung cấp nhiều chức năng như bỏ qua các quy trình triển khai khi cần thiết, kết quả chạy đồng nhất và hiệu quả hơn, v.v. nên GitLab được lựa chọn là nền tảng quản lý mã nguồn cho hệ thống quản lý quán ăn.

Nền tảng quản lý quán ăn được chia làm tám kho lưu trữ mã nguồn (repository) bao gồm năm kho lưu trữ các mã nguồn thuộc về mặt logic của hệ thống, mỗi bộ mã nguồn tương dương với một vi dịch vụ và ba kho lưu trữ bao gồm trang giới thiệu (landing-page), trang quản lý nhà hàng (admin-page), và một trang mô phỏng phần đặt hàng của người dùng (customer-page).

Khi một thay đổi đối với mã nguồn được đẩy lên kho lưu trữ của GitLab, một quá trình kiểm tra và triển khai được cấu hình trước sẽ được chạy bởi GitLab Runner. GitLab Runner là một ứng dụng chuyên biệt được phát triển bởi GitLab tích hợp với kho lưu trữ giúp chạy các câu lệnh cấu hình, kiểm thử và triển khai mã nguồn. Ngoài lựa chọn sử dụng GitLab runner của riêng GitLab, việc tự triển khai riêng các GitLab Runner mang giúp khả năng tuỳ chỉnh cấu hình cụ thể của hạ tầng CI/CD, từ đó giúp tiết kiệm chi phí vận hành.

Có hai pha chính sẽ được thực hiện mỗi khi nhà phát triển phần mềm đẩy code lên kho mã nguồn của GitLab do GitLab Runner phụ trách đó là pha biên dịch mã nguồn

và triển khai mã nguồn trên K8s và một pha phụ đó là cấu hình tệp `.env` cho mã nguồn trước đó giúp đẩy các biến môi trường giúp biên dịch mã nguồn. Khi một thay đổi mới được đẩy lên GitLab, GitLab Runner sẽ tìm đọc tệp cấu hình `.gitlab-ci.yml` để xác nhận những chỉ dẫn cần được thực thi.

```
build_env:
  tags:
    - ${RUNNER_TAG}
  stage: build_env
  image: alpine/k8s:1.23.16
  script:
    - cat $PATH_DOCKER_BUILD_ENV_FILE > .env
  artifacts:
    paths:
      - .env
```

Đoạn mã 3.1: Đoạn mã cấu hình cài đặt biến môi trường cho GitLab Runner.

Đoạn mã 3.1 cho thấy cách đẩy tệp `.env` vào môi trường biên dịch. Biến môi trường `$PATH_DOCKER_BUILD_ENV_FILE` được sử dụng trong đoạn mã được cấu hình trong như một biến môi trường của dự án trên GitLab bên cạnh các biến môi trường khác được GitLab cung cấp sẵn khi chạy CI/CD. Ở đây giá trị của biến là nội dung tệp `.env` là các biến hệ thống cần giúp cho hệ thống có thể biên dịch thành công bao gồm địa chỉ của kho lưu trữ ảnh và tệp trên Amazon S3, các biến trỏ đến đường dẫn các dịch vụ khác của hệ thống.

Sau khi mã nguồn trên kho lưu trữ của GitLab có tệp `env` cần thiết, GitLab Runner sẽ bắt đầu quá trình biên dịch mã nguồn hệ thống. Đoạn mã 3.2 sử dụng `kaniko` giúp chạy lệnh biên dịch mã nguồn sang ảnh Docker. Kaniko [8] là một công cụ giúp xây dựng các ảnh Docker từ Dockerfile, có thể chạy bên trong một container hoặc một cụm K8s. Kaniko giúp giải quyết hai vấn đề khi build Docker trong Docker (Docker-in-Docker) [9] đó là tránh việc phải chạy Docker ở quyền quản trị trong ảnh Docker, vốn là một điều phải có khi cần quyền truy cập vào Docker daemon của máy chủ, và giúp tối ưu quá trình chạy Docker-in-Docker (DinD) thông qua caching hiệu quả các lớp (layer) ảnh Docker [10].

[8] https://github.com/GoogleContainerTools/kaniko

[9] https://docs.gitlab.com/ee/ci/docker/using_docker_build.html#use-docker-in-docker

[10] https://cloud.google.com/blog/products/containers-kubernetes/introducing-kaniko-build-container-

```
build:
  stage: build
  tags:
    - ${RUNNER_TAG}
  image:
    name: gcr.io/kaniko-project/executor:debug
    entrypoint: [""]
  before_script:
    # - echo $IMAGE_REGISTRY_URL
    - echo $IMAGE_REGISTRY_USER
    # - echo $IMAGE_REGISTRY_PASS
    - echo "{\"auths\":{\"$IMAGE_REGISTRY_URL\":{\"auth\":\"$(printf "%s:%s"
        "$IMAGE_REGISTRY_USER" "$IMAGE_REGISTRY_PASS" | base64 | tr -d
        '\n')\"}}}" > /kaniko/.docker/config.json
    # - cat /kaniko/.docker/config.json
  script:
    - ls -la
    - cat .env
    - echo $IMAGE_NAME_BUILD
    - echo $IMAGE_NAME_BUILD_LATEST
    - /kaniko/executor
      --context "${CI_PROJECT_DIR}"
      --dockerfile "${CI_PROJECT_DIR}/Dockerfile"
      --cache=true
      --destination "$IMAGE_REGISTRY_URL/$IMAGE_NAME_BUILD_LATEST"
```

Đoạn mã 3.2: Đoạn mã cấu hình quá trình biên dịch mã cho GitLab Runner.

Trong Đoạn mã 3.2, phần `before_script` giúp người dùng đăng nhập vào một kho lưu trữ ảnh Docker (Docker Registry) giúp dễ dàng quản lý và triển khai trên K8s. Sau đó GitLab Runner chạy phần `script` để biên dịch hệ thống dựa theo tệp `Dockerfile` được định nghĩa trước trong quá trình phát triển và kiểm thử hệ thống.

Cuối cùng pha triển khai (deploy) mã lên K8s của GCP, GitLab Runner được cấu hình chạy một tệp lệnh chương trình sử dụng kubectl, một công cụ giúp giao tiếp với các Master Node (Control Plane) [11]. Các cấu hình cho môi trường K8s nằm trong tệp `deployment-gke.yml` bao gồm các thành phần cài đặt cấu hình cho Ingress, Service

images-in-kubernetes-and-google-container-builder-even-without-root-access

[11] https://kubernetes.io/docs/reference/kubectl/

và Deployment của hệ thống. Đối với phần cấu hình cho Ingress và Service, K8s sẽ chỉ cần có số cổng mà ứng dụng mở ra ngoài và tên của kho mã nguồn, tuy nhiên ở phần cấu hình cho Deployment và autoscaling (tự động mở rộng), các trường thông tin sẽ ảnh hưởng nhiều đến hiệu năng của ứng dụng hơn như thông số của API kiểm tra tình trạng hoạt động, các thông số cấu hình phần cứng liên quan đến bộ nhớ và CPU cho mỗi Pod, và giới hạn số lượng pod tối đa khi hệ thống mở rộng cũng như là ngưỡng hệ thống quá tải cần mở rộng thêm.

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: $CI_PROJECT_NAME
  namespace: $CI_PROJECT_ROOT_NAMESPACE
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: $CI_PROJECT_NAME
  minReplicas: 1
  maxReplicas: 5
  metrics:
    - resource:
        name: memory
        target:
          averageUtilization: 80
          type: Utilization
      type: Resource
    - resource:
        name: cpu
        target:
          averageUtilization: 50
          type: Utilization
      type: Resource
```

Đoạn mã 3.3: Đoạn mã cấu hình cài đặt tự động mở rộng Kubernetes cho GitLab Runner.

Nhằm phục vụ quá trình kiểm thử sản phẩm ở Chương Đánh giá hệ thống, ngưỡng tự động mở rộng trong Đoạn mã 3.3 được đặt tại giới hạn sử dụng 80% bộ nhớ và 50%

CPU. Khi bất kỳ một Pod nào thuộc Deployment đạt tới ngưỡng, GKE sẽ tự động tạo thêm một Pod mới giúp giãn tải trên các Pod đang hoạt động, tới một giới hạn số lượng Pod tối đa là năm như được cấu hình tại Dòng 12, Đoạn mã 3.3.

## 3.3. Một số luồng chức năng chính

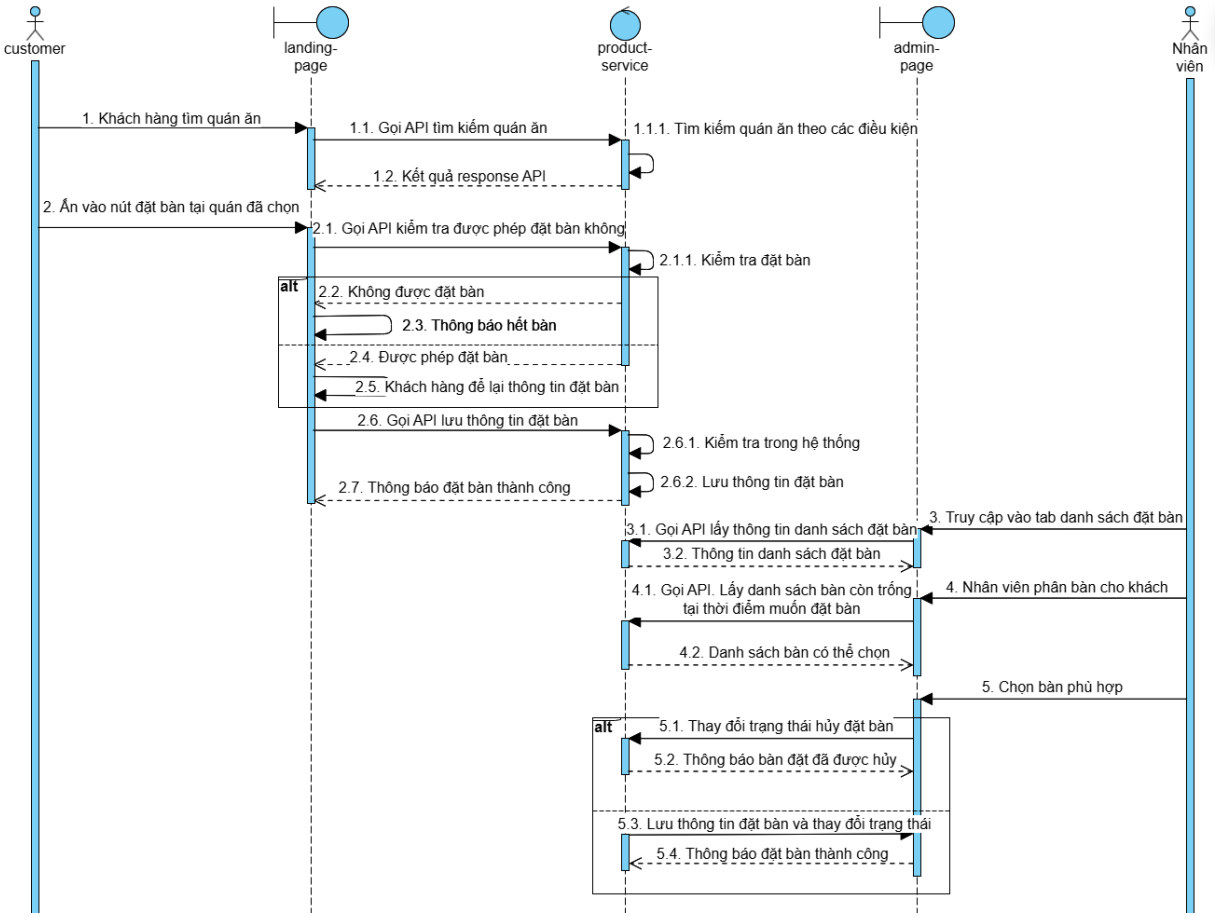
Như đã nói ở Mục 1, các luồng chính của nền tảng quản lý nhà hàng, quán ăn được mô tả sẽ bao gồm luồng đặt bàn của người dùng khách hàng tại quán ăn và luồng đặt đồ ăn thông qua việc quét mã QR từ phía người dùng. Ở Chương này, ta sẽ đi sâu vào luồng di chuyển của người dùng cũng như là các lời gọi dữ liệu giữa các dịch vụ với nhau trong hệ thống.

### 3.3.1. Luồng đặt bàn

Sơ đồ tuần tự ở Hình 3.4 mô tả cách người dùng tương tác với tính năng đặt hàng trên giao diện tại trang giới thiệu. Sơ đồ tồn tại các tác nhân chính là người dùng/thực khách có nhu cầu đặt bàn, giao diện người dùng tương tác khi đặt bàn có thể từ trang giới thiệu nền tảng quản lý nhà hàng, quán ăn hoặc các trang riêng của nhà hàng có liên kết với hệ thống quản lý. Tiếp theo đó là `product-service` tiếp nhận các lời gọi từ phía người dùng và xử lý chúng. Hai tác nhân cuối gồm `admin-page` và nhân viên của cửa hàng phụ trách việc tiếp nhận, xác nhận lại yêu cầu đặt bàn của người dùng.

Luồng người dùng bắt đầu khi người dùng thực hiện tìm kiếm quán ăn trên trang giới thiệu của hệ thống hoặc của từng cửa hàng, khách hàng sẽ có lựa chọn đặt bàn tại quán ăn. Sau khi khách hàng đã chọn xong bàn và điền các thông tin được yêu cầu, hệ thống sẽ kiểm tra tình trạng hiện tại của quán ăn và xác nhận lại với khách hàng tình trạng đặt bàn trên giao diện. Một khi người dùng thực hiện đặt bàn thành công, thông tin sẽ được đẩy đến trung tâm thông báo và chuyển tiếp đến các dịch vụ khác, ở đây là trang quản lý nhà hàng. Nhân viên của quán ăn sẽ xác nhận yêu cầu đặt bàn của khách hàng và tiến hành phân bàn cho khách dựa trên số lượng, yêu cầu khi đặt bàn. Nếu không thể chọn được bàn phù hợp cho yêu cầu của thực khách, nhân viên của quán ăn sẽ thay đổi trạng thái của phần đặt bàn thành đã hủy và ngược lại là thành công nếu có bàn.

Ở Hình 3.4 không có luồng thông báo cho người dùng về trạng thái đặt bàn do người đặt bàn có thể chưa đăng ký tham gia hệ thống, điều đó khiến cho việc thông báo



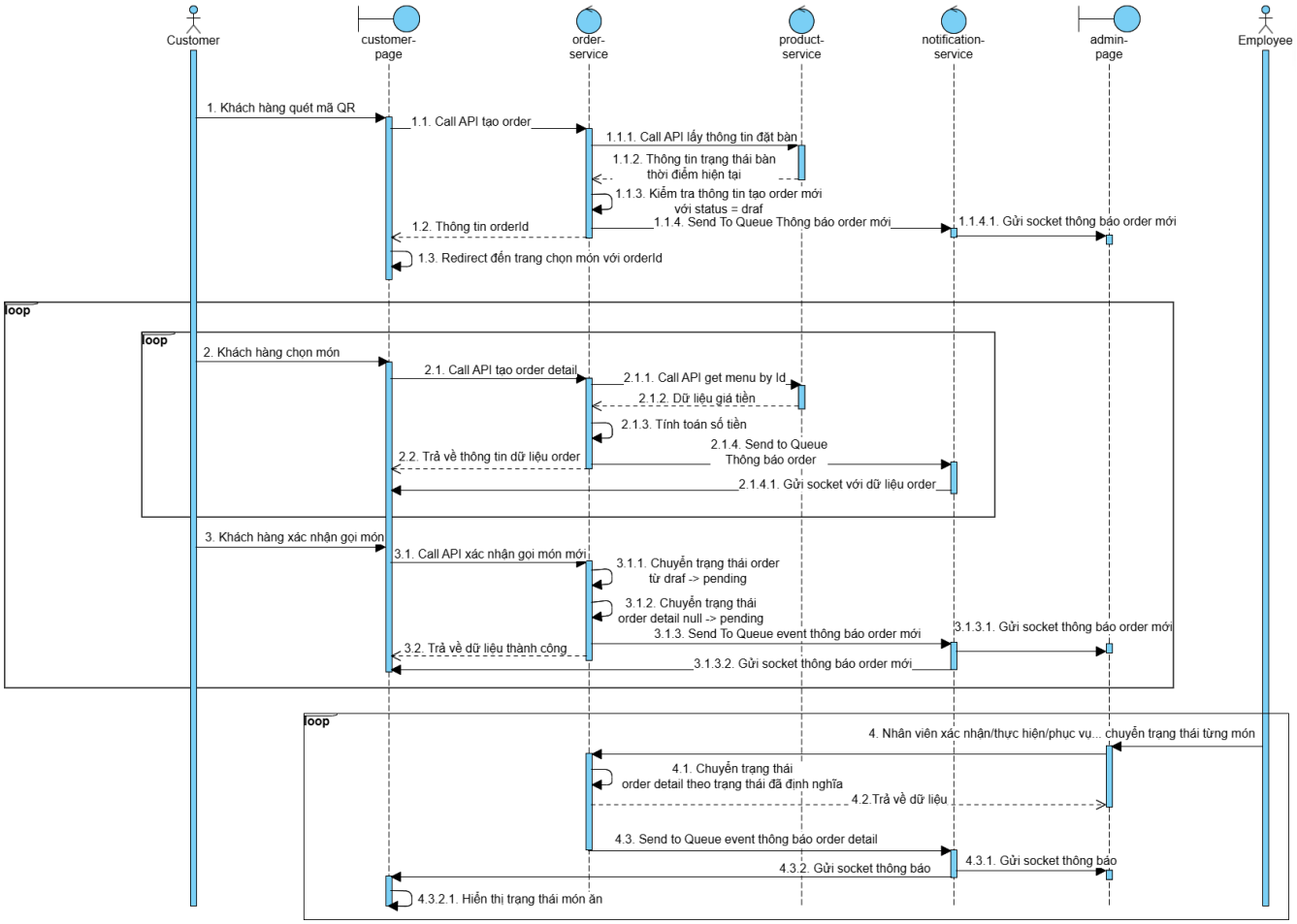

Hình 3.4: Sơ đồ tuần tự mô tả luồng nghiệp vụ đặt bàn trực tuyến

cho người dùng trở nên khó khăn bởi hệ thống thông báo sẽ không biết thông tin của người dùng để hiển thị. Nhân viên của quán sau khi thay đổi trạng thái của yêu cầu đặt bàn sẽ cần xác nhận, thông tin tới khách hàng theo dựa theo biểu mẫu mà khách cung cấp khi yêu cầu đặt bàn trên website.

### 3.3.2. Luồng gọi món

Khi nhà hàng, quán ăn tham gia vào hệ thống quản lý sẽ được hỗ trợ tạo QR gán cho mỗi bàn thuộc nhà hàng giúp khách hàng có thể gọi món mà không cần tương tác với nhân viên của quán. Hình 3.5 mô tả quá trình từ lúc người dùng sử dụng mã QR để đặt món và nhân viên, phục vụ của nhà hàng, quán ăn thay đổi trạng thái của đơn gọi món. Các tác nhân trong sơ đồ bao gồm người dùng và nhân viên là hai tác nhân chính tương tác với hệ thống, trang `customer-page` giúp người dùng tương tác và thực hiện lời gọi đến hệ thống. Các tác nhân đảm nhiệm việc xử lý bao gồm `order-service`,

`product-service`, và `notification-service`.




Hình 3.5: Sơ đồ tuần tự mô tả luồng nghiệp vụ đặt món thông qua quét mã QR

Sau khi người dùng thực hiện quét mã QR, hệ thống sẽ thực hiện xác nhận tạo order mới cho số bàn đang ngồi với trạng thái là `draf`. Khách hàng sẽ được chuyển hướng đến trang chọn món sau khi nhận lại `orderId`, ở đấy khách hàng sẽ chọn các món ăn từ thực đơn của quán. Sau khi người dùng hoàn thành quá trình chọn đồ ăn cũng như xác nhận danh sách các món ăn cho đơn hàng, nhân viên, phục vụ, đầu bếp, v.v. sẽ chuyển trạng thái từng món cụ thể dựa theo tình trạng hiện tại của món ăn như *đang chuẩn bị*, *dã xong*, *đã phục vụ*, v.v. Toàn bộ luồng nghiệp vụ sẽ lặp lại cho đến khi người dùng chọn tính năng thanh toán và hoàn thành bữa ăn.

# Chương 4

# Đánh giá hệ thống

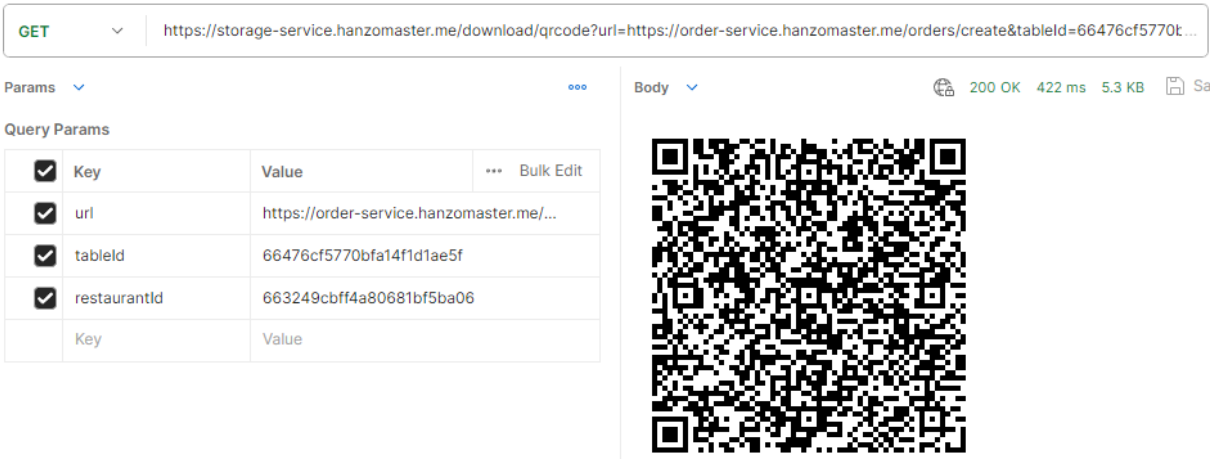
Việc thực hiện đánh giá hệ thống sau khi đã triển khai trên môi trường thực nghiệm là một bước quan trọng giúp đảm bảo hệ thống hoạt động đúng như mong đợi, đáp ứng các yêu cầu về hiệu suất độ tin cậy và khả năng mở rộng. Chương này sẽ đánh giá hiệu quả hoạt động thông qua các bộ kiểm thử được cấu hình sẵn. Ngoài ra ở Chương này đề cập đến một số vấn đề gặp phải trong quá trình phát triển nên tảng quản lý quán ăn.

## 4.1. Hiệu quả hoạt động

Để đánh giá hiệu quả hoạt động của hệ thống một cách toàn diện, hai bộ kiểm thử được tiến hành gồm kiểm thử tự động mở rộng và kiểm thử tính sẵn sàng cao. Kiểm thử tự động mở rộng tập trung vào khả năng GKE tự động điều chỉnh tài nguyên, số lượng Node, Pod để đáp ứng với sự biến động trong lưu lượng truy cập. Kiểm thử tính sẵn sàng cao sẽ đánh giá xem liệu hệ thống có thể duy trì hoạt động liên tục ngay cả khi gặp sự cố

### 4.1.1. Kiểm thử tự động mở rộng

Bằng việc sử dụng HPA của K8s trên môi trường đám mây của GCP, cấu hình phần cứng của các Pod được định nghĩa ở Đoạn mã 4.1. `limits` là cấu hình phần cứng tối đa (bộ nhớ và CPU) mà Pod đó được phép tiêu thụ. Đây là giá trị ngưỡng cố định được đặt ra bởi K8s. `requests` là số lượng bộ nhớ và CPU tối thiểu mà Pod yêu cầu để hoạt động và K8s sẽ đảm bảo lượng tài nguyên này sẽ luôn khả dụng cho Pod. Theo cấu hình tại Dòng 12, Đoạn mã 3.3, mỗi deployment sẽ có tối đa năm Pod hoạt động cùng lúc nhưng

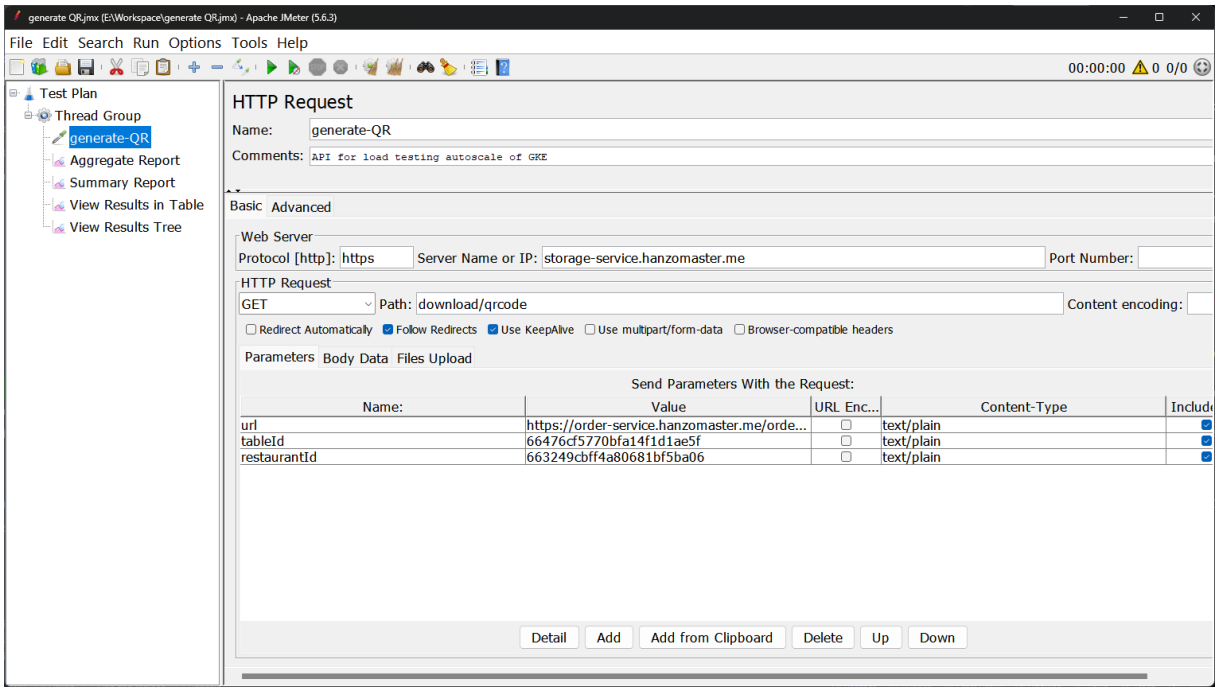



Hình 4.1: API lấy mã QR thanh toán của nhà hàng, quán ăn

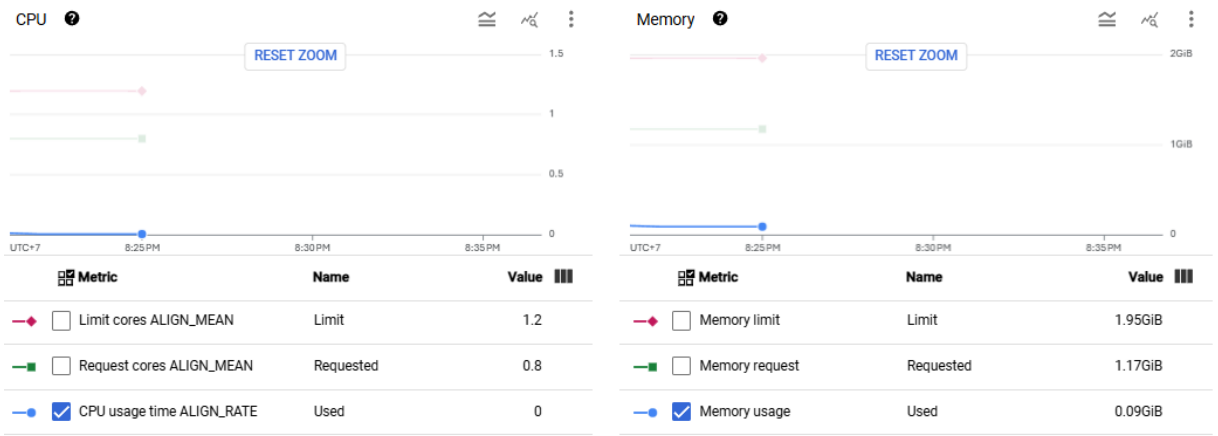
để phục vụ mục đích kiểm thử với lưu lượng thực tế của một hệ thống lớn, số lượng Pod tối đa đã được điều chỉnh lên 20 Pod. Dịch vụ được chọn để chạy kiểm thử sẽ là dịch vụ lưu trữ tệp `storage-service`, dịch vụ này phụ trách việc lưu trữ ảnh và các tệp do người dùng tải lên một S3 bucket của AWS (Amazon Web Service). API được gọi cho mục đích kiểm thử sẽ là một API lấy mã QR thanh toán của nhà hàng đã được tải lên S3.

```
resources :
    limits :
        cpu: 300m
        memory: 500Mi
    requests :
        cpu: 200m
        memory: 300Mi
```

Đoạn mã 4.1: Đoạn mã cấu hình phần cứng cho `storage-service` trên GKE.



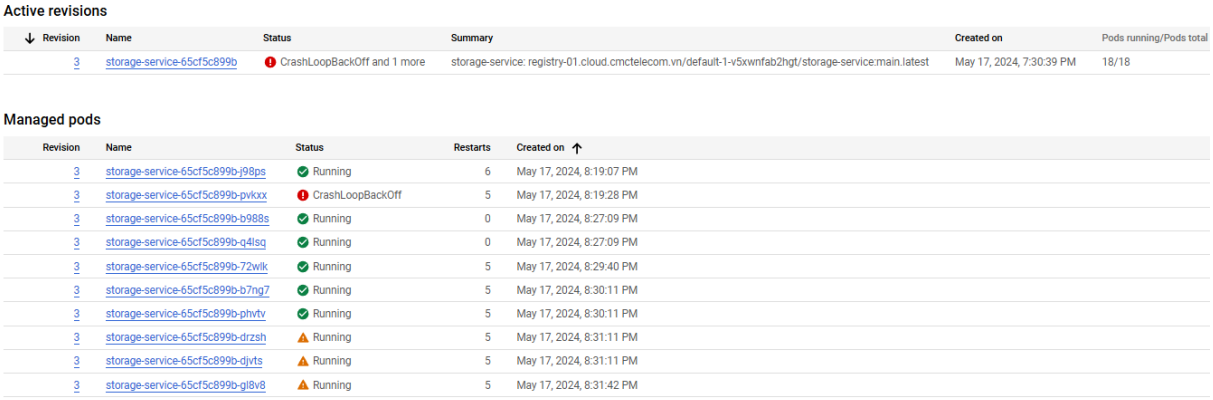

Hình 4.3: Bộ kiểm thử API của Jmeter




Hình 4.2: Lượng tài nguyên trước khi bắt đầu kiểm thử tự động mở rộng

Hình 4.2 cho thấy lượng tài nguyên Deployment trước khi bắt đầu gọi API có số lượng bộ nhớ tối đa có thể truy cập được là `1.95GiB` tương đương với 4 Pod đang hoạt động. Công cụ được sử dụng để chạy kiểm thử là Jmeter, một công cụ kiểm thử hiệu năng mã nguồn mở nổi tiếng thường xuyên được dùng trong các công việc liên quan đến đo lường và phân tích hiệu suất của cá ứng dụng web cũng như là các dịch vụ. Cấu hình Jmeter chạy bộ kiểm thử API với 200 luồng tương đương với 200 người dùng và thời gian tăng trưởng (ramp-up period) 20 giây trong 15 phút, ta có được các chỉ số như sau. Ở thời


Hình 4.4: Lượng tài nguyên sử dụng trong quá trình chạy Jemter kiểm tra tải




Hình 4.5: Số lượng Pod trong quá trình chạy Jmeter kiểm tra tải

điểm số lượng lời gọi vào hệ thống nhiều nhất, hệ thống mở rộng lên tới 11 Pod và lượng tài nguyên được các Pod yêu cầu tiêu thụ là`3.22GiB`. Ngoài ra số lượng Node cũng được K8s cấu hình tăng cường lên trong quá trình này. Một số Pod khi hệ thống hứng chịu một lượng yêu cầu lớn có biểu hiện lỗi do Pod dùng quá bộ nhớ giới hạn của K8s khiến cho Pod bị buộc tắt đi và bật lại. Sau khi quá trình chạy Jmeter hoàn tất, các Pod đều trở lại hoạt động bình thường và K8s tự động giảm số lượng Node và Pod về cấu hình mặc định.

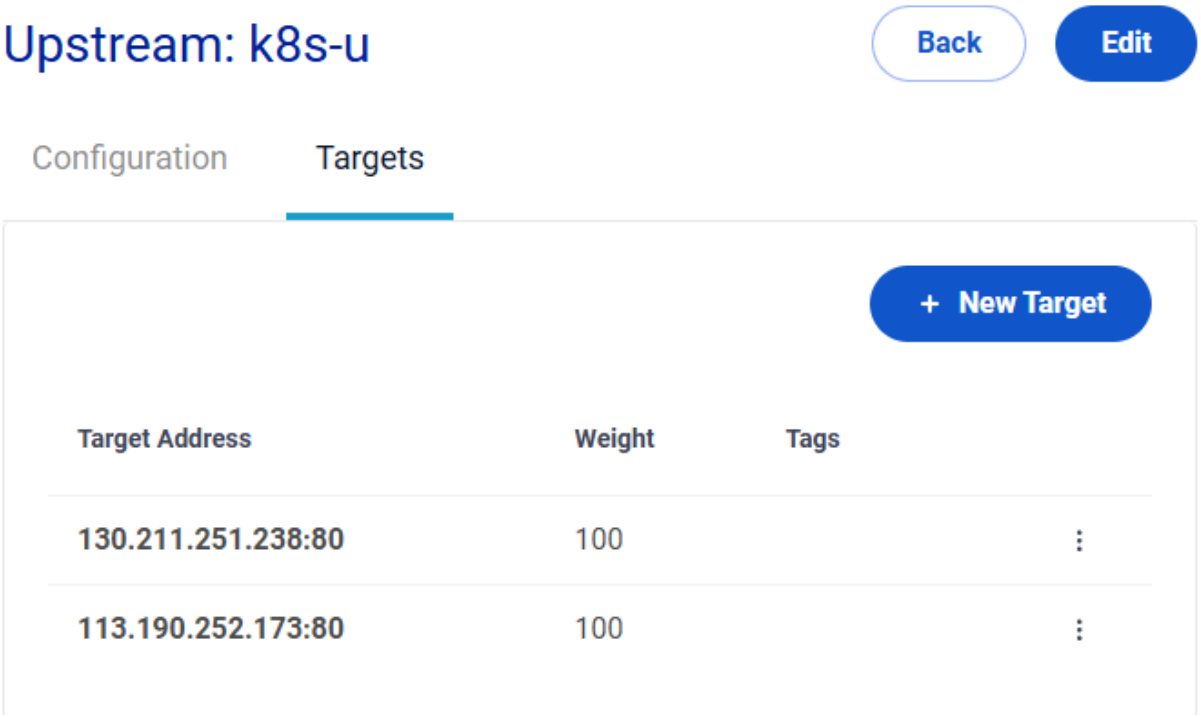
### 4.1.2. Kiểm thử tính sẵn sàng cao

Bộ kiểm thử này được sử dụng với mục tiêu xác định khả năng phục hồi của hệ thống khi gặp sự cố cũng như là tính sẵn sàng cao của hệ thống, tức khi một hệ thống không hoạt động thì các lời gọi yêu cầu đến vẫn có thể được xử lý tại một mức nhất định.

Có nhiều cách nhằm đảm bảo tính sẵn sàng cao của hệ thống, các phương pháp truyền thống vẫn hay được sử dụng như DC/DR (Data Center/Diaster Recovery) kèm theo cân bằng tải giúp giảm thiểu rủi ro khi một thành phần gặp sự cố. Ngoài ra, việc sao lưu dữ liệu thường xuyên cũng là những biện pháp cần thiết giúp duy trì tính liên tục của dịch vụ.

Trong kiến trúc vi dịch vụ, việc sử dụng nhiều cụm K8s đã trở thành một giải pháp hiệu quả để duy trì HA. Với cách tiếp cận này, ứng dụng sẽ được triển khai trên nhiều cụm K8s độc lập, phân tán trên các vùng địa lý khác nhau. Khi một cụm gặp sự cố, lưu lượng truy cập sẽ được tự động chuyển hướng đến các cụm khác, đảm bảo tính liên tục của dịch vụ. Trên thực tế, việc duy trì tính liên tục của hệ thống còn gặp nhiều khó khăn do vấn đề không chỉ nằm ở cụm K8s chạy các dịch vụ xử lý chính của hệ thống mà lỗi có thể rải rác ở bất cứ vị trí nào từ Cổng API, cân bằng tải của Clouflare, cơ sở dữ liệu MongoDB. Ở mỗi vùng này ta đều cần có các biện pháp sao lưu, dự phòng phù hợp giúp tránh hệ thống gặp sự cố khiến cho dịch vụ ngừng hoạt động.

Hệ thống quản lý nhà hàng, quán ăn được cấu hình chia làm hai cụm chạy độc lập với nhau trên GKE của GCP. Từ đó ta sẽ cấu hình Cổng API của Kong giúp cân bằng tải đến cả hai cụm. Ở đây ta sẽ điều chỉnh thêm một Upstream mới với IP của máy chủ đích đến là cụm dự phòng của hệ thống với cùng một thông số `Weight`. Điều này có nghĩa là lưu lượng truy cập sẽ được phân phối điều cho cả hai máy chủ. Sau khi đã cấu hình xong, giờ đây thông qua các API kiểm tra tình trạng của hệ thống (health check), Kong Gateway sẽ định kỳ gọi đến các API này nhằm phát hiện hệ thống có gặp phải sự cố hay không. Khi một hệ thống bị đánh dấu là ngừng hoạt động, Kong Gateway sẽ ngừng chuyển hướng lưu lượng truy cập máy chủ đó mà chuyển các yêu cầu gọi API đến máy chủ còn đang hoạt động. Khi máy chủ gặp sự cố nhận yêu cầu gọi trở lại, Kong Gateway sẽ tự động phát hiện thông qua API kiểm tra tình trạng hoạt động của hệ thống và chuyển hướng lưu lượng truy cập trở lại máy chủ đó. Điều này giúp ứng dụng đạt tính sẵn sàng cao khi hạn chế được thời gian ngừng hoạt động của hệ thống bằng cách phân phối dữ liệu trên nhiều máy chủ khác nhau.




Hình 4.6: Các máy chủ được cấu hình trỏ đến trong Kong Gateway

# Kết luận

Khóa luận này đã trình bày chi tiết về quá trình nghiên cứu, thiết kế, và triển khai một hệ thống quản lý nhà hàng toàn diện, đáp ứng nhu cầu của cả chủ nhà hàng và khách hàng. Hệ thống không chỉ cung cấp các công cụ quản lý hiệu quả cho nhà hàng, quán ăn mà còn tạo ra một nền tảng tương tác thuận tiện cho khách hàng thông qua các tính năng đặt bàn trực tuyến và đặt món trực tuyến thông qua quét mã QR và quản lý thông tin cá nhân.

Việc áp dụng kiến trúc vi dịch vụ cùng với việc tận dụng khả năng mở rộng tự động của GKE đã giúp hệ thống đạt được tính linh hoạt, khả năng chịu lỗi cao và khả năng đáp ứng nhu cầu sử dụng biến động. Các bộ kiểm thử đã chứng minh hiệu quả hoạt động của hệ thống trong việc tự động mở rộng và duy trì tính sẵn sàng cao, đảm bảo trải nghiệm người dùng mượt mà và không bị gián đoạn.

Tuy còn một vài hạn chế trong thiết kế và tích hợp hoàn chỉnh các luồng nghiệp vụ của hệ thống, sau khi các vấn đề được khắc phục trong tương lai gần, hệ thống sẽ tiếp tục được mở rộng và phát triển với các tính năng mới nhằm nâng cao trải nghiệm người dùng và đáp ứng tốt hơn nhu cầu thị trường. Một số các chức năng trong số đó ao gồm tính năng đặt món trực tuyến trên trang giới thiệu của nhà hàng, quán ăn, quản lý chế độ dinh dưỡng của người dùng, gợi ý món ăn tại trang đặt món, phát triển ứng dụng di động, v.v. Với những định hướng phát triển này, hệ thống quản lý nhà hàng với hy vọng trở thành một nền tảng không thể thiếu cho cả nhà hàng và khách hàng, mang lại giá trị thực tiễn và đóng góp tích cực vào sự phát triển của ngành dịch vụ ăn uống tại Việt Nam.

# Tài liệu tham khảo

## Tiếng Anh


[1] Kaitano Dube, Godwell Nhamo, and David Chikodzi (2021). "COVID-19 cripples global restaurant and hospitality industry". In: *Current Issues in Tourism* 24.11, pp. 1487–1490.

[2] Thao Hoang and Javed Suleri (2021). "Customer behaviour in restaurants before and during COVID-19: A study in Vietnam". In: *Research in Hospitality Management* 11.3, pp. 205–214.

[3] Oleksandr Mykhailovych Matsenko et al. (2021). "Transformation of the Restaurant Business as a Result of the COVID-19 Pandemic: Improving the Security of Service and Maintaining the Health of Human Capital". In.

[4] Van Kien Pham, Thu Ha Do Thi, and Thu Hoai Ha Le (2020). "A study on the COVID-19 awareness affecting the consumer perceived benefits of online shopping in Vietnam". In: *Cogent Business & Management* 7.1, p. 1846882.