**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**PHẠM BÁ TRUNG HIẾU**

**PHÁT TRIỂN CÁC REST API CHO VÍ THANH TOÁN TRỰC TUYẾN SỬ DỤNG KIẾN TRÚC MICROSERVICES**

**ĐỒ ÁN NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**TP. HỒ CHÍ MINH, 2024BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC MỞ THÀNH PHỐ HỒ CHÍ MINH**

**PHẠM BÁ TRUNG HIẾU**

**PHÁT TRIỂN CÁC REST API CHO VÍ THANH TOÁN TRỰC TUYẾN SỬ DỤNG KIẾN TRÚC MICROSERVICES**

**Mã số sinh viên: 2151013024**

**ĐỒ ÁN NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Giảng viên hướng dẫn: TRƯƠNG HOÀNG VINH**

**TP. HỒ CHÍ MINH, 2024**

LỜI CẢM ƠN

Trước tiên, em xin được gửi lời cảm ơn trân thành tới ThS. Trương Hoàng Vinh – giảng viên hướng dẫn đã tận tình giúp đỡ trong quá trình hoàn thiện đồ án. Hướng dẫn và góp ý chi tiết của thầy đã giúp em nhận ra thiếu sót của bản thân. Đồng thời được mở rộng và nâng cao kiến thức.

Đồ án này không chỉ là kết quả đúc kết được sau quá trình học tập mà còn là niềm đam mê mãnh liệt với Công Nghệ Thông Tin. Em tin rằng trải nghiệm và kiến thức này sẽ là bước đệm vững chắc cho sự nghiệp sau này tại các công ty.

Do kiến thức và thời gian có hạn, đồ án này chắc chắn sẽ không tránh khỏi những thiếu sót. Em mong nhận được sự góp ý của quý thầy cô và bạn bè nhằm hoàn thiện hơn trong tương lai.

Cuối cùng, em xin gửi lời cảm ơn tới các quý thầy, cô và bạn bè đã luôn đồng hành và giúp đỡ trong quá trình học tập tại trường Đại học Mở Thành Phố Hồ Chí Minh.

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TÓM TẮT ĐỐ ÁN NGÀNH

ABSTRACT

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc179158243)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc179158244)

[TÓM TẮT ĐỐ ÁN NGÀNH 3](#_Toc179158245)

[ABSTRACT 4](#_Toc179158246)

[DANH MỤC TỪ VIẾT TẮT 7](#_Toc179158247)

[DANH MỤC HÌNH VẼ 8](#_Toc179158248)

[DANH MỤC BẢNG 9](#_Toc179158249)

[MỞ ĐẦU 10](#_Toc179158250)

[Chương 1. TỔNG QUAN ĐỀ TÀI 11](#_Toc179158251)

[1.1. Giới thiệu đề tài 11](#_Toc179158252)

[1.1.1. Mục tiêu và phạm vi đề tài 11](#_Toc179158253)

[1.1.2. Phương pháp nghiên cứu 12](#_Toc179158254)

[1.1.3. Bố cục báo cáo 12](#_Toc179158255)

[Chương 2. CƠ SỞ LÝ THUYẾT 13](#_Toc179158256)

[2.1. Giới thiệu Spring Boot 13](#_Toc179158257)

[2.1.1. Đặc điểm chính 13](#_Toc179158258)

[2.1.2. Kiến trúc 14](#_Toc179158259)

[2.1.3. Cách sử dụng 16](#_Toc179158260)

[2.2. Giới thiệu NextJS 18](#_Toc179158261)

[2.2.1. Đặc điểm chính 19](#_Toc179158262)

[2.2.2. Kiến trúc 20](#_Toc179158263)

[2.2.3. Cách sử dụng 20](#_Toc179158264)

[2.3. Cách dịch vụ và công cụ hỗ trợ 21](#_Toc179158265)

[2.3.1. MongoDB 21](#_Toc179158266)

[2.3.2. PostgreSQL 21](#_Toc179158267)

[2.3.3. Docker 21](#_Toc179158268)

[Chương 3. HỆ THỐNG THANH TOÁN RESTFUL API TÍCH HỢP ĐA PHƯƠNG THỨC 22](#_Toc179158269)

[3.1. Giới thiệu bài toán 22](#_Toc179158270)

[3.2. Phân tích hệ thống 22](#_Toc179158271)

[3.2.1. Lược đồ use case 22](#_Toc179158272)

[3.2.2. Đặc tả use case 23](#_Toc179158273)

[3.3. Thiết kế hệ thống 28](#_Toc179158274)

[3.3.1. Sơ đồ tuần tự (sequence diagram) 28](#_Toc179158275)

[3.3.2. Sơ đồ hoạt động (activity diagram) 31](#_Toc179158276)

[3.3.3. Thiết kế dữ liệu 35](#_Toc179158277)

[3.4. Kiến trúc hệ thống 37](#_Toc179158278)

[3.4.1. Dịch vụ chính: 37](#_Toc179158279)

[3.4.2. Công cụ hỗ trợ: 38](#_Toc179158280)

[3.5. Kết quả đề tài 38](#_Toc179158281)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 39](#_Toc179158282)

[4.1. Kết luận 39](#_Toc179158283)

[4.2. Hướng phát triển 39](#_Toc179158284)

[TÀI LIỆU THAM KHẢO 40](#_Toc179158285)

[PHỤ LỤC 41](#_Toc179158286)

DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **Từ viết tắt** | **Ý nghĩa** |
| 1 | CSDL | Cơ sở dữ liệu |
| 2 | MVC | Model – View - Controller |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

DANH MỤC HÌNH VẼ

[Hình 2‑1. Kiến trúc Spring Boot 14](#_Toc179158287)

[Hình 2‑2. Luồng đi của Spring Boot 15](#_Toc179158288)

[Hình 2‑3. Spring initializr 16](#_Toc179158289)

[Hình 2‑4. Cấu trúc dự án Spring Boot 17](#_Toc179158290)

[Hình 2‑5. Controller Spring Boot 18](#_Toc179158291)

[Hình 2‑6. Chương trình Spring Boot đầu tiên 18](#_Toc179158292)

[Hình 2‑7. NextJS Routing 19](#_Toc179158293)

[Hình 2‑8. NextJS dynamic routes 19](#_Toc179158294)

[Hình 2‑9. Kiến trúc NexJS 20](#_Toc179158295)

[Hình 2‑10. Cấu trúc NextJS 20](#_Toc179158296)

[Hình 3‑1. Lược đồ use case 23](#_Toc179158297)

[Hình 3‑2. Sơ đồ tuần tự tạo thanh toán 28](#_Toc179158298)

[Hình 3‑3. Sơ đồ tuần tự xác nhận giao dịch 28](#_Toc179158299)

[Hình 3‑4. Sơ đồ tuần tự tra cứu trạng thái giao dịch 29](#_Toc179158300)

[Hình 3‑5. Sơ đồ tuần tự hoàn tiền giao dịch 30](#_Toc179158301)

[Hình 3‑6. Sơ đồ hoạt động tạo thanh toán 31](#_Toc179158302)

[Hình 3‑7. Sơ đồ trạng thái xác nhận giao dịch 32](#_Toc179158303)

[Hình 3‑8. Sơ đồ hoạt động tra cứu trạng thái giao dịch 33](#_Toc179158304)

[Hình 3‑9. Sơ đồ hoạt động hoàn tiền giao dịch 34](#_Toc179158305)

[Hình 3‑10. Lược đồ cơ sở dữ liệu 35](#_Toc179158306)

[Hình 3‑11. Kiến trúc hệ thống 37](#_Toc179158307)

DANH MỤC BẢNG

[Bảng 3‑1. Use case tạo thanh toán 22](#_Toc179147854)

[Bảng 3‑2. Use case xác nhận giao dịch 23](#_Toc179147855)

[Bảng 3‑3. Use case tra cứu trạng thái 24](#_Toc179147856)

[Bảng 3‑4. Use case hoàn tiền giao dịch 25](#_Toc179147857)

MỞ ĐẦU

# TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

Trong bối cảnh phát triển mạnh của xã hội, các ví điện tử đã và đang được nhiều người tin tưởng và sử dụng để thanh toán cho các giao dịch mua sắm. Chúng đem lại sự tiện lợi, nhanh chóng và hợp xu hướng với giới trẻ. Các dịch vụ mua sắm cũng đang dần tích hợp để theo kịp sự phát triển của thời đại, góp phần tiện lợi cho khách hàng. Tuy nhiên, để có được các dịch vụ thanh toán thông qua ví điện tử đôi khi sẽ lại là khó khăn, nhất là trong việc tích hợp, triển khai và bảo trì, bên cạnh đó cũng tốn nhiều nguồn lực và nhân công.

Đối với các ví điện tử, doanh nghiệp phải tham khảo tài liệu kỹ thuật riêng biệt, tuân thủ các chuẩn và bảo mật của bên thanh toán. Tùy thuộc vào ngôn ngữ và kiến trúc hệ thống, việc hiện thực lại có thể khác nhau. Điều này gây ra khó khăn trong việc xây dựng, bảo trì và tiêu tốn nguồn lực đối với các doanh nghiệp.

Nắm bắt được nhu cầu, đề tài này đề xuất phát triển hệ thống thanh toán Restful API tích hợp đa phương thức. Hệ thống sẽ đóng vai trò là lớp trung gian, cung cấp bộ API chuẩn hóa cho phép doanh nghiệp dễ dàng tích hợp nhiều phương thức thanh toán khác nhau thông qua một hệ thống duy nhất.

### Mục tiêu và phạm vi đề tài

#### Mục tiêu

Xây dựng được hệ thống thanh toán RestFul API tích hợp đa phương thức, cụ thể:

* Phân quyền và chứng thực hai vai trò quan trọng là merchant và admin
* Bộ API chuẩn hóa gồm 4 loại: Tạo thanh toán, xác nhận thanh toán, tra cứu trạng thái và hoàn tiền giao dịch
* Doanh nghiệp có thể gửi yêu cầu tạo, xác nhận, tra cứu và hoàn tiền thanh toán tới các ví điện tử
* Trang Dashboard, cho phép merchant theo dõi, quản lý và thanh toán các giao dịch
* Trang Admin, cho phép quản lý các merchant và phương thức thanh toán

#### Phạm vi

Để đảm bảo được yêu cầu và tiến độ, đề tài sẽ tập trung vào các mục sau:

* Xây dựng hệ thống có kiến trúc có thể mở rộng và có khả năng thêm mới các hình thức thanh toán khác trong tương lai
* Tích hợp các ví điện tử phổ biến (MoMo, ZaloPay, VnPay và PayPal)
* Cung cấp bộ API thanh toán chuẩn với nhiều phương thức thanh toán cho doanh nghiệp
* Trang Dashboard giúp cho doanh nghiệp theo dõi và kiểm soát các giao dịch

### Phương pháp nghiên cứu

#### Phân tích yêu cầu

* Khảo sát thị trường và xu hướng của ví điện tử
* Xác định nhu cầu thanh toán hiện có

#### Xác định nghiệp vụ

#### Vẽ sơ đồ use case và đặc tả, mô tả các tính năng cần thiết

#### Vẽ sơ đồ sequence, mô tả quá trình chi tiết cách hệ thống hoạt động và tương tác

#### Vẽ sơ đồ activity, mô tả luồng đi của hệ thống

#### Thiết kế lược đồ cơ sở dữ liệu, cách hệ thống lưu trữ, tương tác

#### Lựa chọn công nghệ

* Spring Boot: triển khai theo kiến trúc microservices, đáp ứng được khả năng nâng cấp, mở rộng và phát triển
* NextJS: giao diện trực quan, linh hoạt và hiệu suất cao
* PostgreSQL: lưu trữ dữ liệu có cấu trúc, đảm bảo sự nhất quán và ổn định
* MongoDB: lưu trữ dữ liệu phi cấu trúc, đáp ứng khả năng mở rộng và hiệu suất cao.

### Bố cục báo cáo

Bao gồm 4 chương:

* Chương 1: Tổng Quan Đề Tài
* Chương 2: Cơ Sở Lý Thuyết
* Chương 3: Hệ Thống Thanh Toán RESTFUL API tích hợp đa phương thức
* Chương 4: Kết Luận Và Hướng Phát Triển

# CƠ SỞ LÝ THUYẾT

## Giới thiệu Spring Boot

Spring Boot là một framework giúp phát triển các ứng dụng trên nền Java. Nó cung cấp công cụ mạnh mẽ để phát triển các ứng dụng web và microservices.

Spring Boot là phần mở rộng được phát triển lên từ Spring framework, nhằm loại bỏ điểm yếu chính là cấu hình phức tạp và triển khai, giúp cho nhà phát triển có thể tập trung phát triển và xây dựng hệ thống nhanh chóng.

Ngoài ra, cộng đồng Spring rất mạnh mẽ, luôn cập nhật và hỗ trợ nhiều tiện ích khác nhau, có thể ví dụ như Spring Security, Spring Web MVC, Spring Cloud, … Tất cả đều hoàn toàn tương thích và dễ dàng tích hợp.

### Đặc điểm chính

* Auto Configurations

Một trong những ưu điểm to lớn của Spring Boot là khả năng tự cấu hình (Auto Configurations), Spring Boot sẽ cấu hình dựa vào các dependencies đã khai báo. Điều này lảm giảm bớt sự phức tạp của cấu hình thủ công, ví dụ như khởi tạo các bean hoặc các thuộc tính.

Khi cần thêm mới một dependency, Spring Boot sẽ tự động cung cấp cấu hình cần thiết như tạo bean, security hay kết nối CSDL. Ngoài ra, người dùng có thể ghi đè cấu hình mặc định thông qua tệp cấu hình như application.properties hoặc application.yaml, hoặc thông qqua mã nguồn.

* Standalone

Spring Boot mặc định đi kèm với máy chủ tích hợp (Tomcat, Jetty) và toàn bộ ứng dụng sẽ được đóng gói thành tệp JAR duy nhất.

Ưu điểm khác là loại bỏ cấu hình XML phức tạp, Spring Boot sử dụng các annotations để quản lý các cấu hình, giúp cho hệ thống đơn giản và dễ hiểu.

* Opinionated

Spring boot được thiết kế để đưa ra quyết định phù hợp nhất. Khi một dự án được khởi tạo. Spring Boot sẽ tự cấu hình mặc định thay cho người dùng dựa vào các phụ thuộc. Tuy nhiên người dùng vẫn có thể ghi đè và tự cấu hình theo cá nhân.

* Production-ready

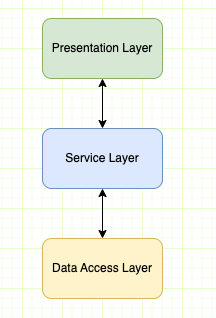
Spring Boot cung cấp bộ cung cụ sẵn sàn cho môi trường sản xuất.

Spring boot Actuator, công cụ giám sát và quản lý ứng dụng mạnh mẽ, giúp người dùng dễ quản lý ứng dụng trong môi trường Production.

Ngoài ra, Spring còn cung cấp công cụ về logging và monitoring, đảm bảo ứng dụng được hoạt động ổn dịnh.

### Kiến trúc

Spring Boot sử dụng mô hình kiến trúc đa tầng.



Hình 2‑. Kiến trúc Spring Boot

Presentation Layer: Tầng trình bày,xử lý tương tác với người dùng, nhận vào Http request và trả về response. Chứa các controller của hệ thống. Ngoài ra, có thêm lớp filter để lọc các request tới hệ thống.

Service Layer: xử lý các business logic và trung gian giữa Presentation layer và Data Access Layer.

Data Access Layer: Chịu trách nhiệm tương tác với CSDL, nhận lời gọi từ Service layer, truy vấn dữ liệu và trả về dữ liệu tương ứng. Ngoài ra còn chứa các model đại diện cho các thực thể trong CSDL.



Hình 2‑. Luồng đi của Spring Boot

Người dùng tạo yêu cầu HTTP (POST/GET/PUT/PATCH/DELETE)

Request sau đó sẽ được gửi đến hệ thống, trải qua các lớp security filter, tại đây request không hợp lệ sẽ bị chặn, những request hợp lệ sẽ đi tiếp và được map tới controller tương ứng.

Controller sẽ gọi đến các service phù hợp.

Service gọi lớp respository, yêu cầu truy vấn dữ liệu xuống CSDL.

Dữ liệu được trả về lớp Service, Service trả về Controller và trả về cho người dùng.

### Cách sử dụng

Một trong những cách để khởi tạo một dự án Spring Boot là thông qua Spring Initializer.

Spring Initializr là công cụ phổ biến giúp khởi tạo nhanh cấu trúc dự án Spring Boot một cách đơn giản.

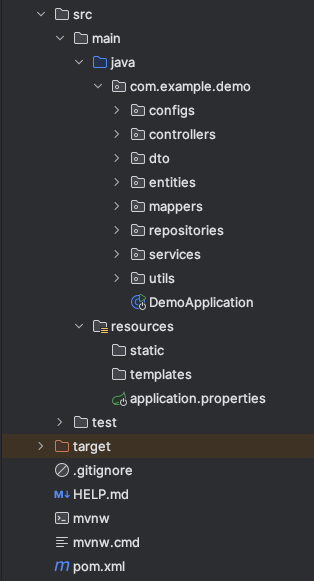


Hình 2‑. Spring initializr

Tại đây, ta có thể lựa chọn phiên bản Spring Boot, ngôn ngữ sử dụng và quản lý các dependencies.

Sau khi hoàn thành cấu hình, ta bấm generate để tải dự án.

Cấu trúc một dự án sử dụng Spring Boot sẽ bao gồm:

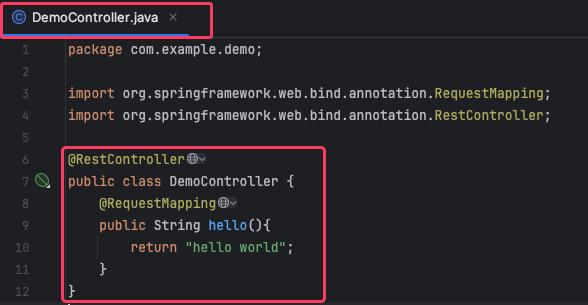


Hình 2‑. Cấu trúc dự án Spring Boot

Thành phần chính:

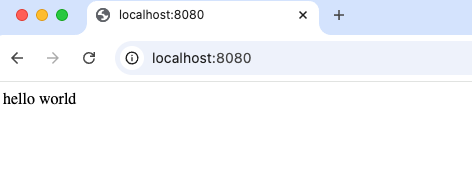
* **Src/main/java/:** Chứa toàn bộ mã nguồn chính của hệ thống, bao gồm các lớp như Controller, Repository, Service, Entity và các thành phần khác.
* **Src/main/resources/:** Chứa tài nguyên tĩnh và cấu hình của ứng dụng.
* **Src/test:** Chứa tệp kiểm thử của ứng dụng.
* **Pom.xml:** Nơi quản lý tất cả các dependencies.

Ta tạo một controller đơn giản



Hình 2‑. Controller Spring Boot

Chạy dự án và truy cập ***localhost:8080***



Hình 2‑. Chương trình Spring Boot đầu tiên

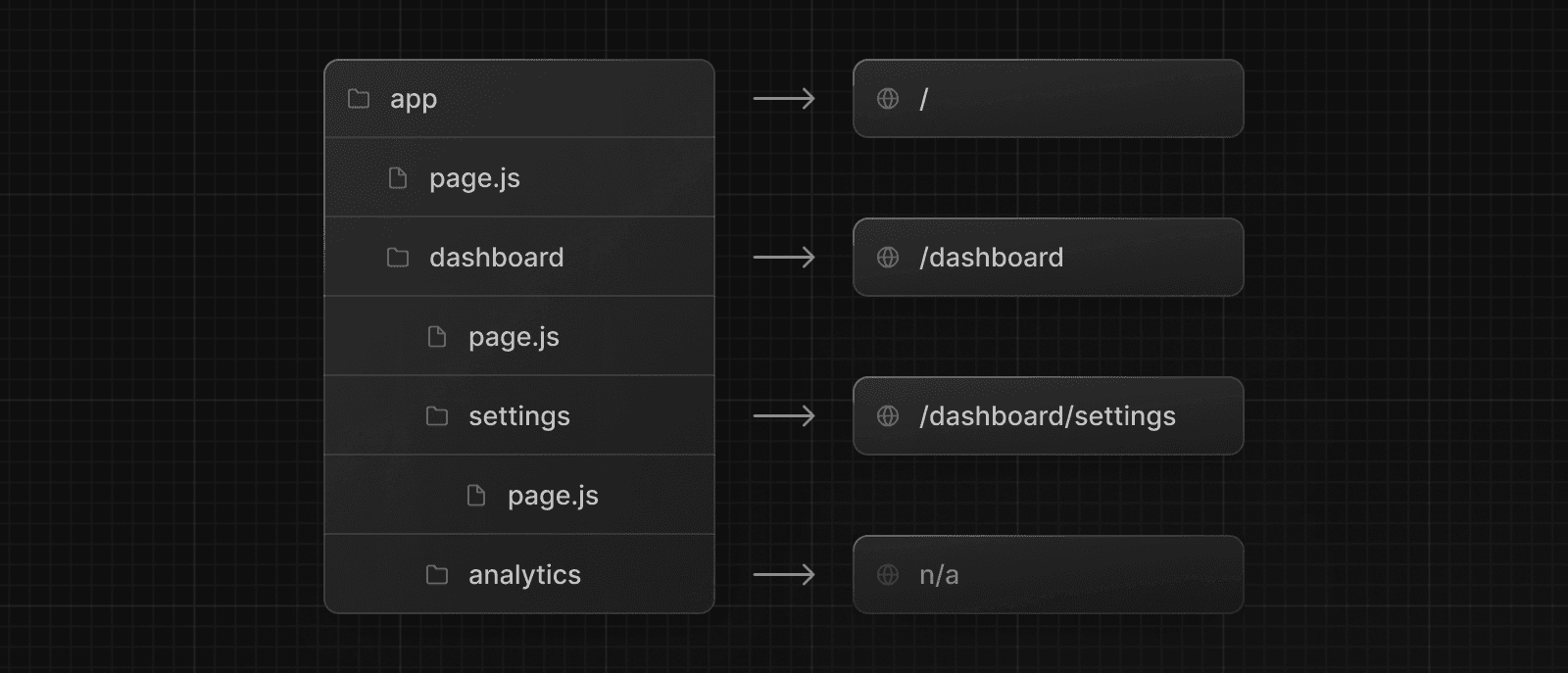
## Giới thiệu NextJS

NextJS là một framework React được sử dụng phổ biến bởi tính linh hoạt và gọn nhẹ. Nó hỗ trợ mạnh mẽ trong việc phát triển website một cách nhanh chóng, đồng thời tối ưu SEO và đem lại trải nghiệm người dùng tốt nhất.

Bên cạnh đó, NextJS thừa hưởng những tính năng mạnh mẽ của React, đồng thời bổ sung những tính năng nâng cao và tối ưu như routing, Server-Side Rendering (SSR) và Static Site Generation (SSG).

### Đặc điểm chính

* Routing: Cơ chế routing của NextJS vô cùng đặc biệt



Hình 2‑. NextJS Routing

NextJS sẽ lấy cấu trúc thư mục cho routing, ví dụ src/app thì ta sẽ có được đường dẫn API mặc định là /, bên cạnh đó NextJS cũng hỗ trợ lồng các folder lại với nhau tạo thành các nested routes.

Dynamic Routes: Là cơ chế routing động, bằng cách đặt tên trong dấu ngoặc vuông, đường dẫn API sẽ thay đổi tùy theo tham số truyền vào



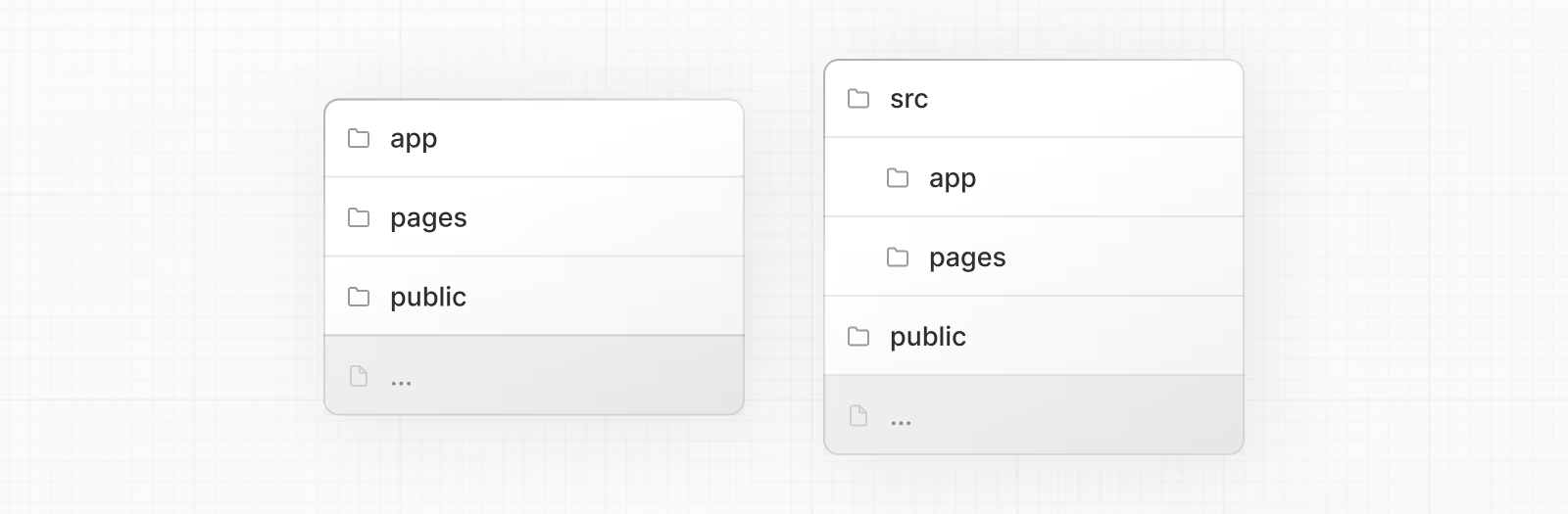
Hình 2‑8. NextJS dynamic routes

* Rendering: Hỗ trợ 2 cơ chế
* Server-Side Rendering (SSR): Tăng tốc độ tải trang, cập nhật dữ liệu mới một cách nhanh chóng, đồng thời tăng khả năng SEO, phù hợp với website cần thay đổi liên tục.
* Static Site Generation (SSG): Cho phép render trang tĩnh, tăng trải nghiệm người dùng, phù hợp với website ít sự thay đổi
* Data Fetching: Hỗ trợ mạnh mẽ trong việc fetching và caching dữ liệu. Kết hợp với SSG và SSR, giúp giảm tải cho server và nâng cao trải nghiệm người dùng.
* Styling: NextJS hỗ trợ đa dạng nhiều loại styling, bao gồm
  + CSS modules
  + Global CSS
  + Tailwind CSS
  + Sass
  + CSS-in-JS
  + Optimizations:
* Typescript: NextJS được tích hợp sẵn Typescript, hỗ trợ tốt trong việc kiểm tra dữ liệu.

### Kiến trúc

Hình 2‑. Kiến trúc NexJS

### Cách sử dụng



Hình 2‑. Cấu trúc NextJS

Cấu trúc một dự án sử dụng NextJS sẽ bao gồm:

* **Src/:** Chứa mã nguồn chính của hệ thống.
* **Public/:** Dùng để lưu trữ tài nguyên tình như hình ảnh, …

## Cách dịch vụ và công cụ hỗ trợ

### MongoDB

Cơ sở dữ liệu không quan hệ. Nổi bật với khả năng mở rộng và tốc độ truy vấn cao, thường được sử dụng vói hệ thống cần tốc độ và thời gian thực. Dữ liệu dược lưu với dạng documents (BSON, tương tự như JSON)

### PostgreSQL

Cơ sở dữ liệu quan hệ. Nổi bật với khả năng mở rộng, bảo mật và nhất quán dữ liệu. PostgreSQL có thể mở rộng về chiều ngang và dọc nhưng vẫn đảm bảo được tính chất ACID. Hiện tại nhiều công ty lớn đã và đang tin tưởng sử dụng PostgreSQL.

### Docker

Tách biệt môi trường developer và production, cung cấp khả năng cân bằng tải (load balancing), thích hợp cho hệ thống microservices.

# HỆ THỐNG THANH TOÁN RESTFUL API TÍCH HỢP ĐA PHƯƠNG THỨC

## Giới thiệu bài toán

Trong bối cảnh xã hội phát triển, các ví điện tử đang ngày càng phổ biến, đặc biệt là giới trẻ. Các dịch vụ mua sắm hiện nay đang dần sử dụng để bắt kịp xu hướng của xã hội. Tuy nhiên, để có thể tích hợp nhiều ví điện tử vào một hệ thống đôi khi sẽ rất tốn kém. Ngoài ra, mỗi ví điện tử sẽ có tài liệu kỹ thuật và chuẩn riêng, điều năng gây phức tạp hóa quy trình tích hợp và triển khai.

Giải quyết vấn đề trên, đề tài đề xuất giải pháp đóng vai trò là hệ thống trung gian giữa các doanh nghiệp và các ví điện tử. Hệ thống sẽ cung cấp giải pháp API thanh toán đa phương thức, cho phép doanh nghiệp dễ dàng kết nối với nhiều phương thức thanh toán khác nhau. Giải pháp sẽ giúp doanh nghiệp đơn giản hóa và tiết kiệm chi phí, đồng thời vẫn đảm bảo được hiệu năng và bảo mật.

## Phân tích hệ thống

### Lược đồ use case

Hệ thống sẽ bao gồm 2 actor:

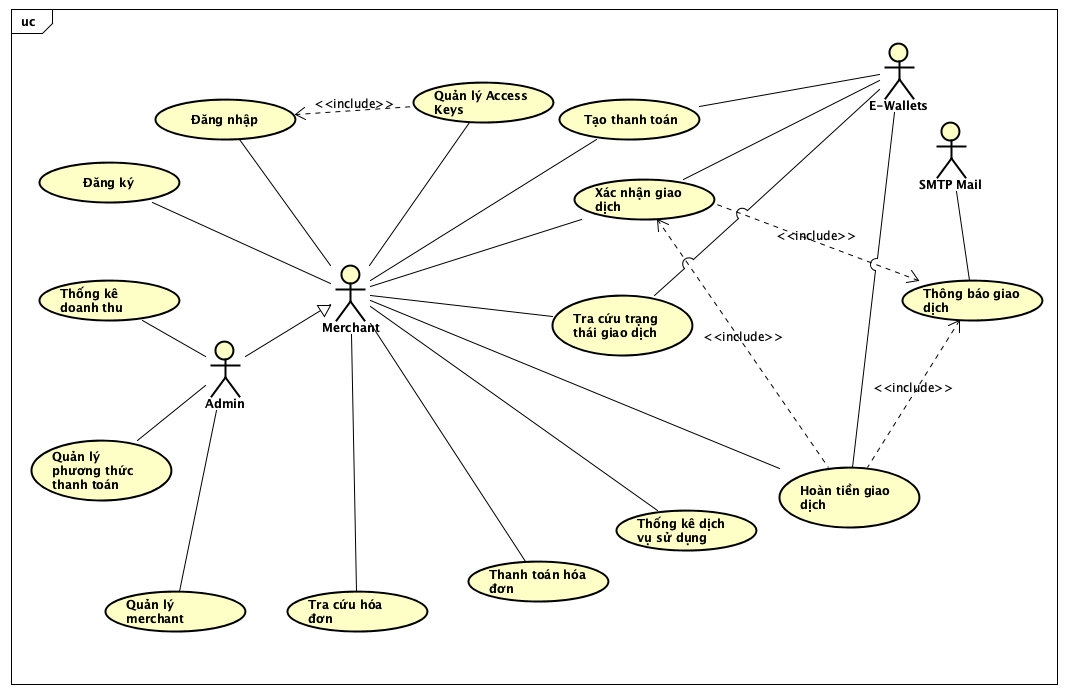
* Admin: Đảm nhiệm vai trò quản trị, có toàn quyền trong hệ thống.
* Merchant: Đảm nhiệm vai trò người sử dụng hệ thống.

Bên cạnh đó sẽ có 2 actor phụ:

* E-Wallets: Các ví điện tử
* SMTP Mail: Dịch vụ gửi mail thông báo

Các chính năng chính:

* Tạo thanh toán: Tạo thanh toán tới các ví điện tử tương ứng
* Tra cứu trạng thái giao dịch: Tra cứu trạng thái của giao dịch thanh toán các ví điện tử.
* Xác nhận giao dịch: Kiểm tra trạng thái giao dịch và chuyển trạng thái.
* Hoàn tiền giao dịch: Hoàn tiền một phần hoặc toàn bộ của giao dịch.
* Quản lý Access Keys: Thêm mới và cập nhật trạng thái của key.



Hình 3‑. Lược đồ use case

### Đặc tả use case

#### Use case tạo thanh toán

|  |  |
| --- | --- |
| **Use case Id** | UC-01 |
| **Tên use case** | Tạo thanh toán |
| **Mô tả** | Merchant gửi yêu cầu tạo thanh toán với phương thức tương ứng |
| **Actor chính** | Merchant |
| **Actor Phụ** | E-Wallets, SMTP Mail |
| **Tiền điều kiện** | Không |
| **Hậu điều kiện** | Nhận thông tin giao dịch thanh toán thông qua hệ thống |
| **Luồng hoạt động** | 1. Merchant gửi yêu cầu tạo thanh toán 2. Hệ thống xác thực thông tin merchant 3. Hệ thống lưu giao dịch mới 4. Hệ thống xác định ví điện tự theo yêu cầu của merchant 5. Hệ thống gửi yêu cầu thanh toán đến ví điện tử tương ứng 6. Ví điện tử xử lý yêu cầu và trả kết quả 7. Hệ thống cập nhật trạng thái giao dịch 8. Hệ thống trả thông tin thanh toán cho merchant |
| **Luồng thay thế** | 2a. Xác thực thất bại   1. Hệ thống báo lỗi cho merchant 2. Use case kết thúc   6a.Ví điện tử không phản hồi   1. Hệ thống thử lại 2. Chuyển sang luồng xử lý ngoại lệ   6b. Tạo thanh toán thất bại   1. Hệ thống cập nhật trạng thái giao dịch 2. Tiếp tục bước 8 |
| **Luồng ngoại lệ** | 1. Lưu request log 2. Thông báo cho merchant về trạng thái giao dịch |

Bảng 3‑. Use case tạo thanh toán

#### Use case xác nhận giao dịch

|  |  |
| --- | --- |
| **Use case Id** | UC-02 |
| **Tên use case** | Xác nhận giao dịch |
| **Mô tả** | Merchant xác nhận đã thanh toán với giao dịch của ví điện tử |
| **Actor chính** | Merchant |
| **Actor Phụ** | E-Wallets, SMTP Mail |
| **Tiền điều kiện** | Không |
| **Hậu điều kiện** | Xác nhận thành công thanh toán |
| **Luồng hoạt động** | 1. Merchant gửi yêu cầu xác nhận giao dịch 2. Hệ thống xác thực thông tin merchant 3. Hệ thống xác định ví điện tự theo yêu cầu của merchant 4. Hệ thống gửi yêu cầu xác nhận thanh toán 5. Ví điện tử xử lý yêu cầu và trả kết quả 6. Hệ thống cập nhật trạng thái giao dịch 7. Hệ thống trả kết quả thanh toán cho merchant 8. Hệ thống gửi thông báo về kết quả giao dịch thông qua mail |
| **Luồng thay thế** | 2a. Xác thực thất bại   1. Hệ thống báo lỗi cho merchant 2. Use case kết thúc   6a.Ví điện tử không phản hồi   1. Hệ thống thử lại 2. Chuyển sang luồng xử lý ngoại lệ   6b. Tạo thanh toán thất bại   1. Hệ thống cập nhật trạng thái giao dịch 2. Tiếp tục bước 8 |
| **Luồng ngoại lệ** | 1. Lưu request log 2. Thông báo cho merchant về trạng thái giao dịch |

Bảng 3‑. Use case xác nhận giao dịch

#### Use case tra cứu trạng thái giao dịch

|  |  |
| --- | --- |
| **Use case Id** | UC-03 |
| **Tên use case** | Tra cứu trạng thái giao dịch |
| **Mô tả** | Cho phép merchant tra cứu trạng thái giao dịch |
| **Actor chính** | Merchant |
| **Actor Phụ** | E-Wallets |
| **Tiền điều kiện** | Không |
| **Hậu điều kiện** | Nhận kết quả trạng thái của giao dịch |
| **Luồng hoạt động** | 1. Merchant gửi yêu cầu tra cứu 2. Hệ thống xác thực thông tin merchant 3. Hệ thống xác định ví điện tự theo yêu cầu của merchant 4. Hệ thống gửi yêu cầu tra cứu đến ví điện tử tương ứng 5. Ví điện tử xử lý yêu cầu và trả kết quả 6. Hệ thống trả kết quả tra cứu cho merchant |
| **Luồng thay thế** | 2a. Xác thực thất bại   1. Hệ thống báo lỗi cho merchant 2. Use case kết thúc   6a.Ví điện tử không phản hồi   1. Hệ thống thử lại 2. Chuyển sang luồng xử lý ngoại lệ |
| **Luồng ngoại lệ** | Lưu request log |

Bảng 3‑. Use case tra cứu trạng thái

#### Use case hoàn tiền giao dịch

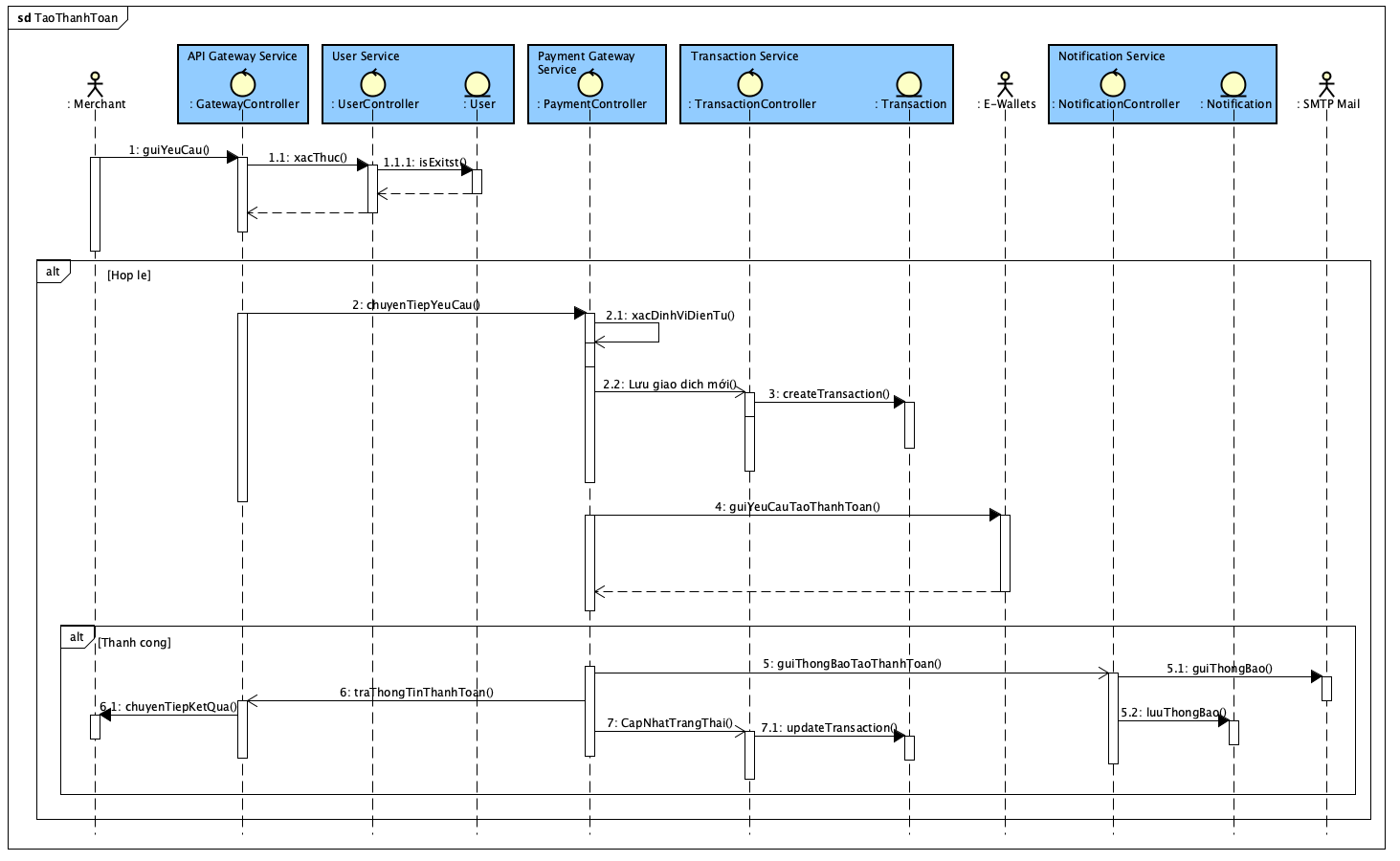
|  |  |
| --- | --- |
| **Use case Id** | UC-04 |
| **Tên use case** | Hoàn tiền giao dịch |
| **Mô tả** | Merchant gửi yêu cầu hoàn tiền giao dịch |
| **Actor chính** | Merchant |
| **Actor Phụ** | E-Wallets, SMTP Mail |
| **Tiền điều kiện** | Không |
| **Hậu điều kiện** | Merchant nhận được tiền hoàn |
| **Luồng hoạt động** | 1. Merchant gửi yêu cầu hoàn tiền 2. Hệ thống xác thực thông tin merchant 3. Hệ thống xác định ví điện tự theo yêu cầu của merchant 4. Hệ thống kiểm tra trạng thái giao dịch được lưu trong cơ sở dữ liệu 5. Hệ thống gửi yêu cầu hoàn tiền đến ví điện tử tương ứng 6. Ví điện tử xử lý yêu cầu và trả kết quả 7. Hệ thống cập nhật trạng thái giao dịch 8. Hệ thống trả kết quả thanh toán cho merchant 9. Hệ thống gửi thông báo về kết quả hoàn tiền |
| **Luồng thay thế** | 2a. Xác thực thất bại   1. Hệ thống báo lỗi cho merchant 2. Use case kết thúc   4a. Có trạng thái là “chưa thanh toán”   1. Hệ thống báo lỗi cho merchant 2. Use case kết thúc   6a.Ví điện tử không phản hồi   1. Hệ thống thử lại 2. Chuyển sang luồng xử lý ngoại lệ   6b. Hoàn tiền thất bại   1. Hệ thống cập nhật trạng thái giao dịch 2. Tiếp tục bước 8 |
| **Luồng ngoại lệ** | 1. Lưu request log 2. Thông báo cho merchant về trạng thái giao dịch |

Bảng 3‑. Use case hoàn tiền giao dịch

## Thiết kế hệ thống

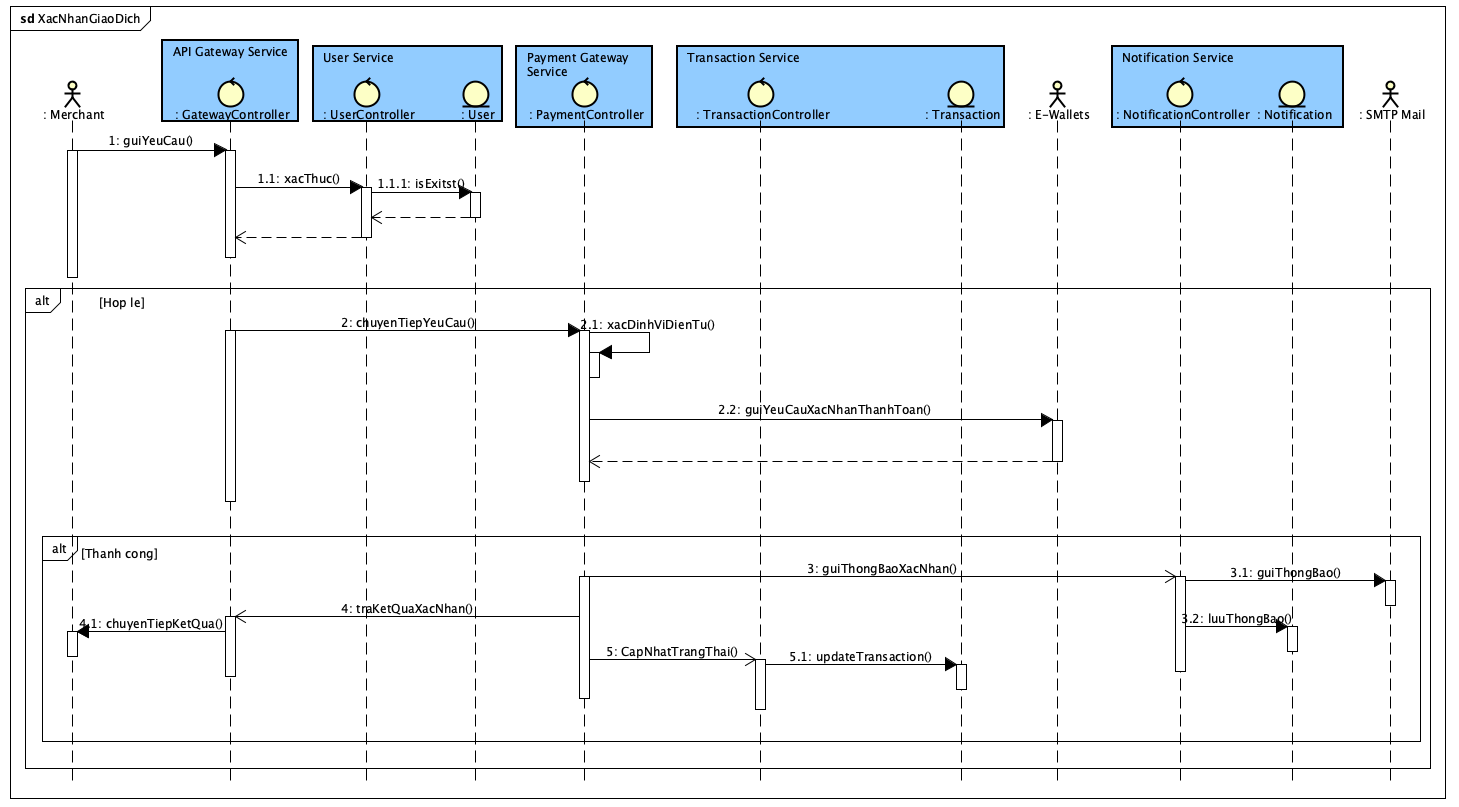
### Sơ đồ tuần tự (sequence diagram)

#### Tạo thanh toán



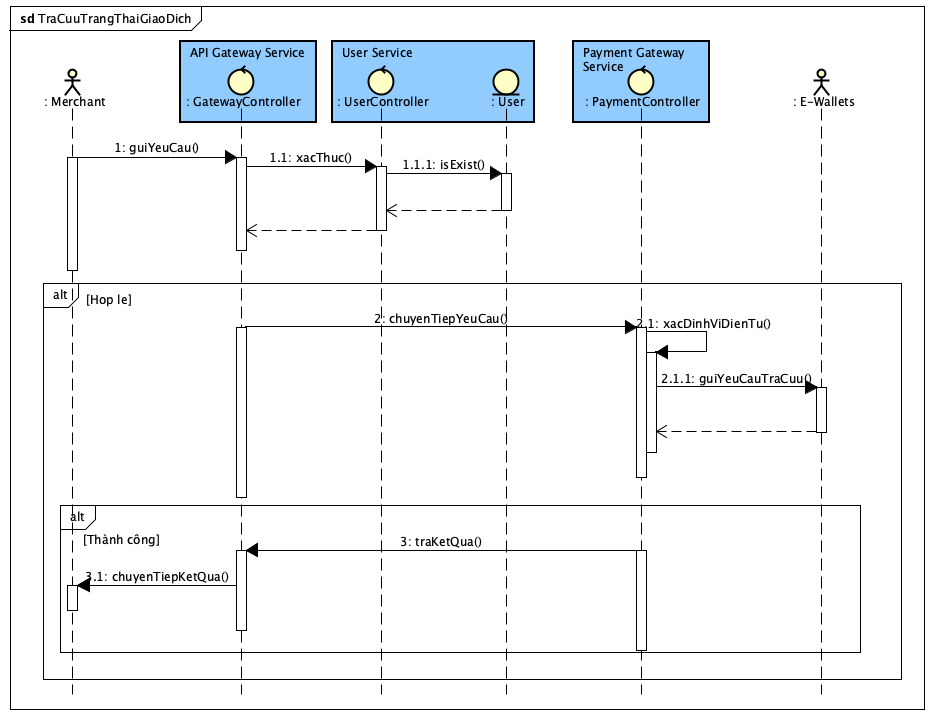
Hình 3‑. Sơ đồ tuần tự tạo thanh toán

#### Xác nhận giao dịch



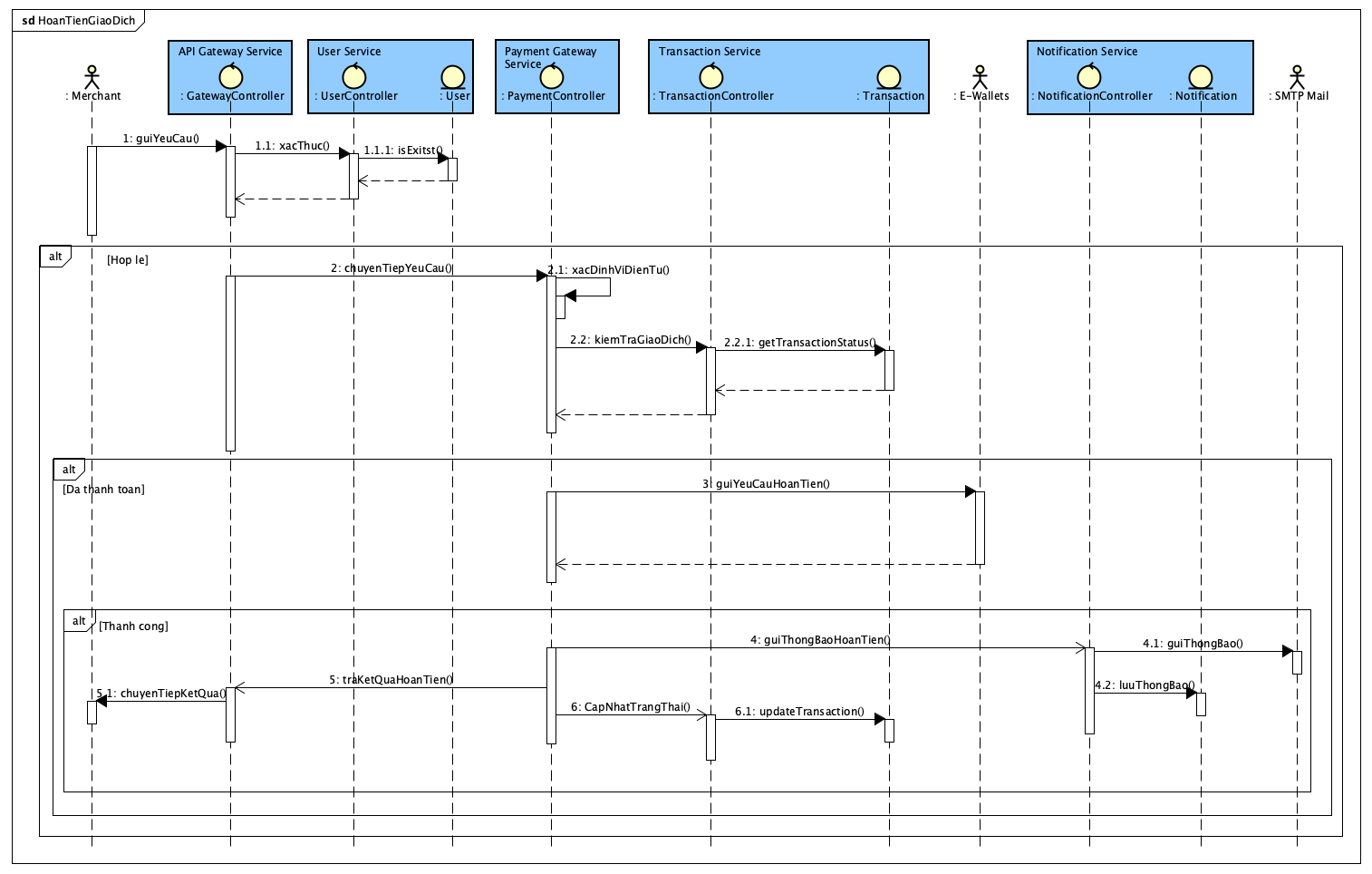
Hình 3‑. Sơ đồ tuần tự xác nhận giao dịch

#### Tra cứu trạng thái giao dịch



Hình 3‑. Sơ đồ tuần tự tra cứu trạng thái giao dịch

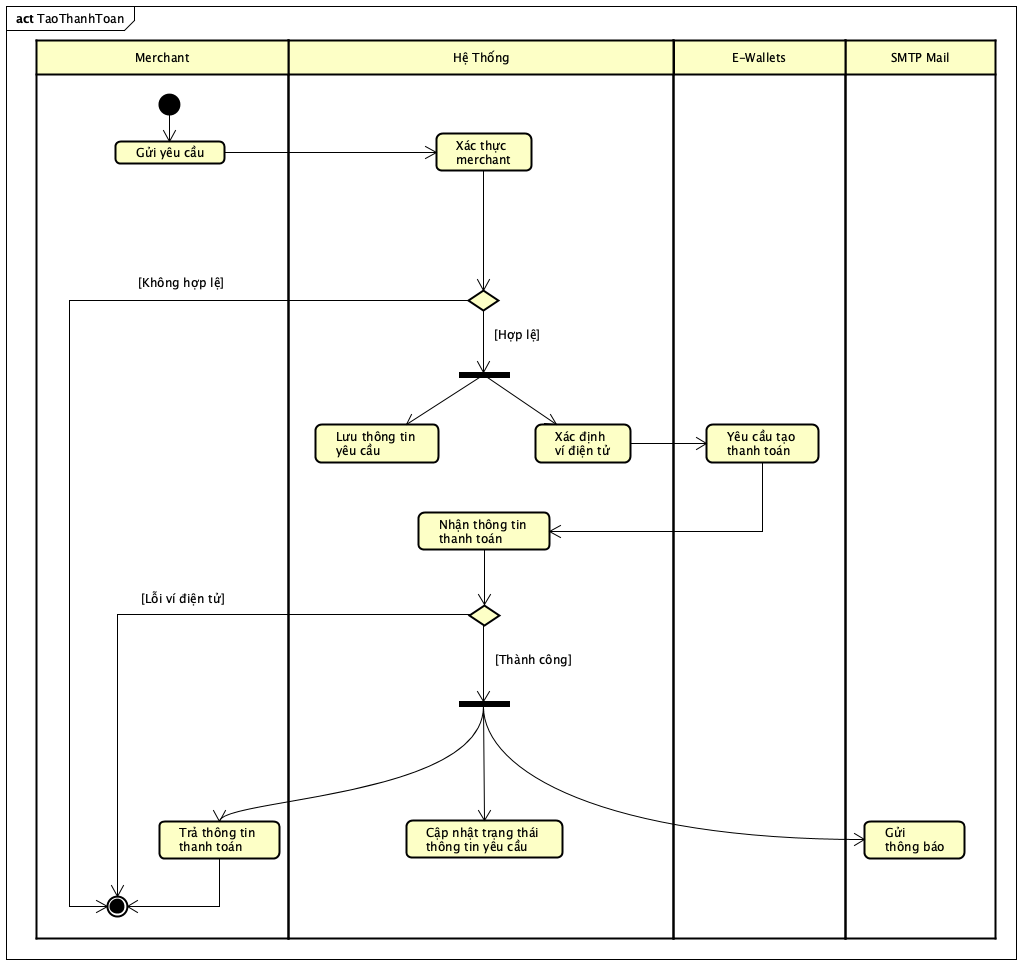
#### Hoàn tiền giao dịch



Hình 3‑. Sơ đồ tuần tự hoàn tiền giao dịch

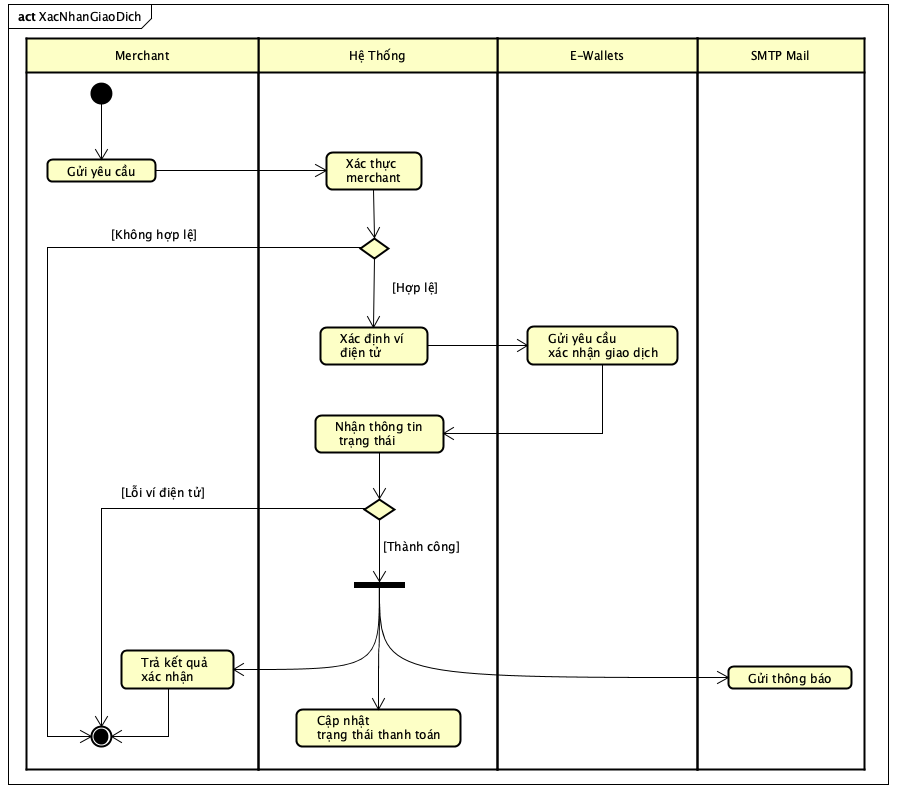
### Sơ đồ hoạt động (activity diagram)

#### Tạo thanh toán



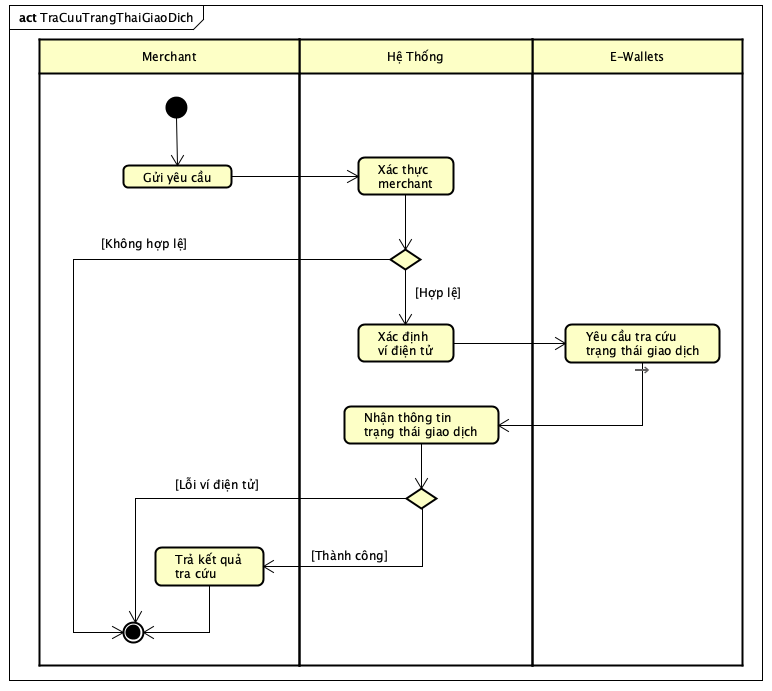
Hình 3‑. Sơ đồ hoạt động tạo thanh toán

#### Xác nhận giao dịch



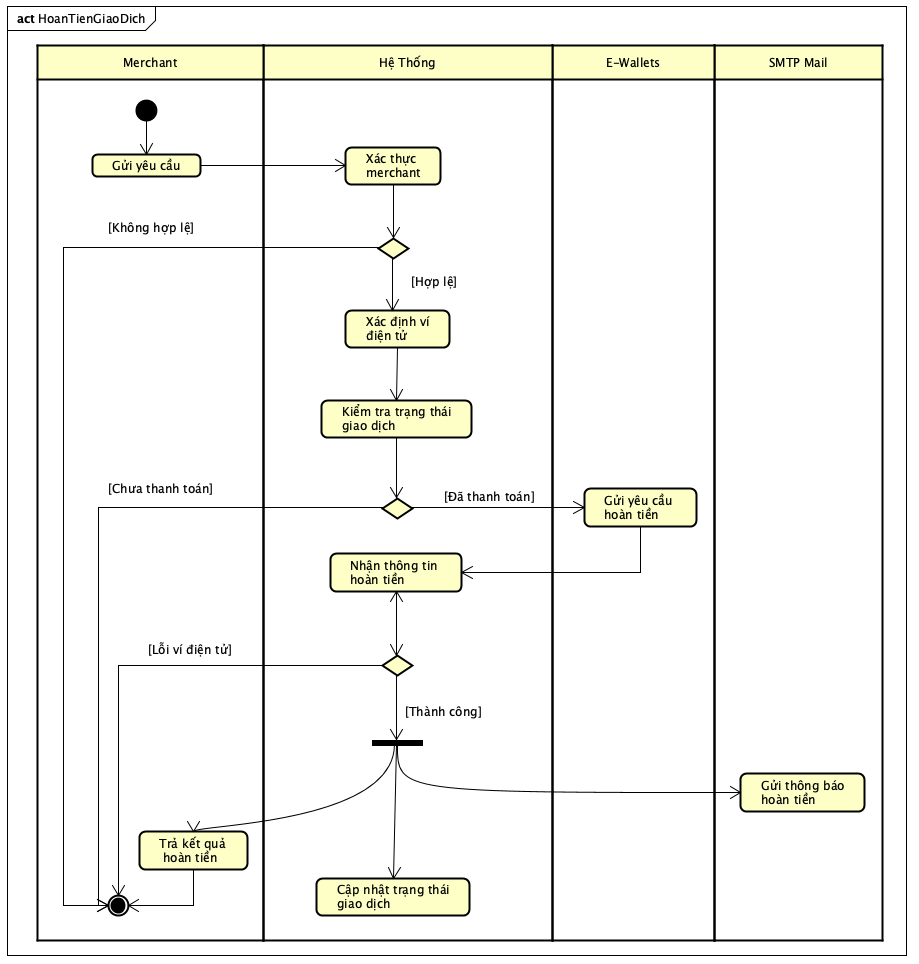
Hình 3‑. Sơ đồ trạng thái xác nhận giao dịch

#### Tra cứu trạng thái giao dịch



Hình 3‑. Sơ đồ hoạt động tra cứu trạng thái giao dịch

#### Hoàn tiền giao dịch



Hình 3‑. Sơ đồ hoạt động hoàn tiền giao dịch

### Thiết kế dữ liệu

­

Hình ‑. Lược đồ cơ sở dữ liệu

#### Đặc điểm

Hệ thống sẽ có 5 dịch vụ chính, tương đương với 5 cơ sở dữ liệu riêng biệt, tuân thủ đúng kiến trúc microservices.

* User service

Gồm 2 bảng

* User: Chứa thông tin người dùng
* Access Key: Chứa thông tin khóa truy cập của người dùng
* Transaction Service

Gồm 2 bảng

* Transaction: Chứa thông tin giao dịch tới các ví diện tử
* Payment Types: Chứa các phương thức thanh toán
* Notification Service

Gồm 2 bảng

* Notification: Chứa thông tin thông báo
* Notification Types: Chứa các loại thông báo
* Payment Gateway Service

Gồm 1 bảng

* Request log: Chứa lịch sử request tới các ví điện tử
* Billing Service

Gồm 3 bảng

* Billing: Chứa thông tin hóa đơn của người dùng
* Billing Details: Chứa thông tin chi tiết hóa đơn
* Fee Types: Chứa thông tin các loại phí

#### Mối quan hệ giữa các bảng

User - Access Key: (1-N) Một người dùng có thể có nhiều khóa truy cập, một khóa truy cập chỉ thuộc về một người dùng.

User – Transaction: (1-N) Một người dùng có thể thực hiện nhiều giao dịch, một giao dịch chỉ thuộc về một người dùng.

User – Billing: (1-N) Một người dùng có thể có nhiều hóa đơn, một hóa đơn chỉ thuộc một người dùng.

User – Notification: (1-N) Một người dùng có thể nhận nhiều thông báo, một thông báo chỉ thuộc về một người dùng.

Billing – Billing Details: (1-N) Một hóa đơn có thể có nhiều chi tiết hóa đơn, một chi tiết hóa đơn chỉ thuộc về một hóa đơn.

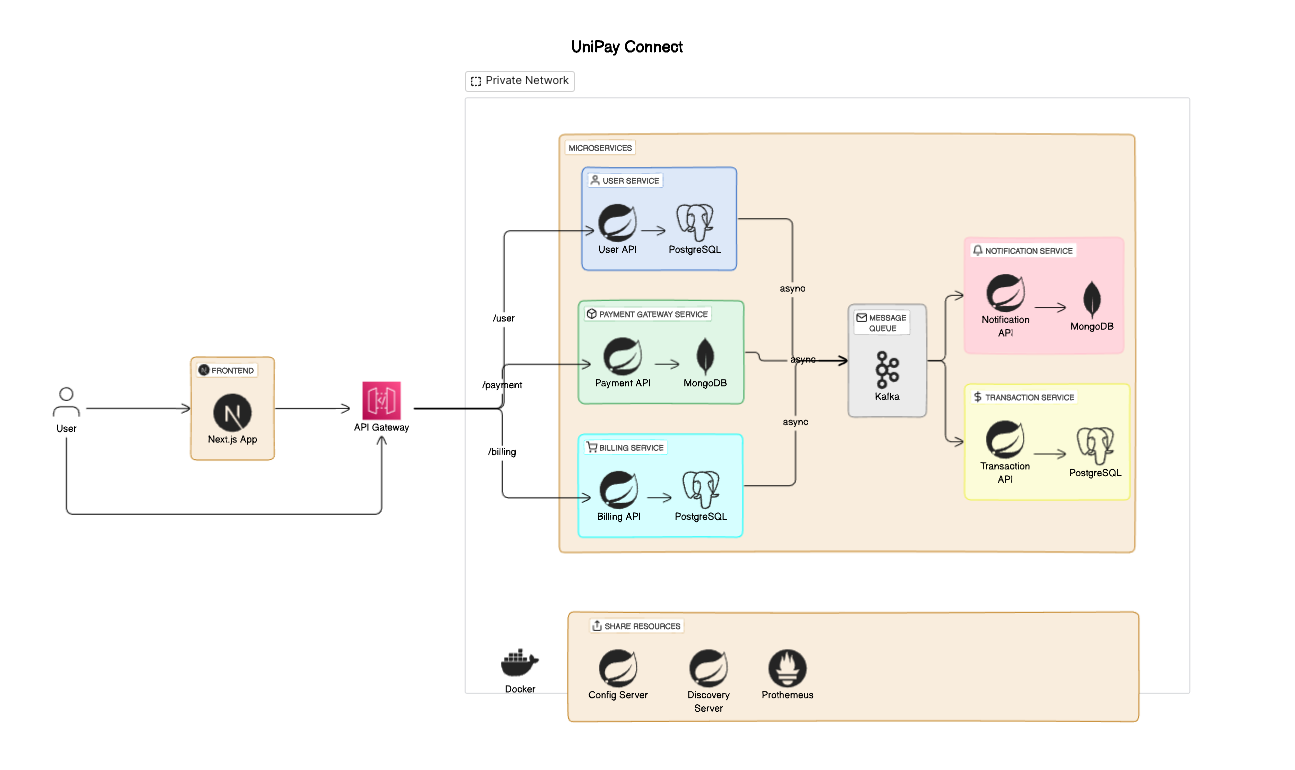
Fee Types – Billing Details: (1-N) Một loại phí có thể thuộc nhiều chi tiết hóa đơn, một chi tiết hóa đơn chỉ thuộc về một loại phí.

Transaction – Request Log: (1-N) Một giao dịch có thể có nhiều log request, một request log chỉ thuộc về một giao dịch.

Payment Types – Transaction: (1-N) Một loại thanh toán có thể thuộc nhiều giao dịch, một giao dịch chỉ thuộc về một loại thanh toán.

Notification Types – Notification: (1-N) Một loại thông báo có thể thuộc nhiều loại thông báo, một thông báo thuộc về một loại thông báo.

## Kiến trúc hệ thống



Hình 3‑. Kiến trúc hệ thống

Tuân thủ theo thiết kế của hệ thống microservices, mỗi hệ thống sẽ tách biệt với nhau, bao gồm có hệ thống riêng biệt, có từng database riêng và đều được xây dựng với Spring Boot.

### Dịch vụ chính:

1. User Service:
   * Đăng nhập/Đăng ký Merchant
   * Quản lý Merchant và Access Keys
   * Chứng thực và phân quyền
2. Payment Gateway Service:

* Trung gian giữa hệ thống và các ví điện tử. Tích hợp các phương thức thanh toán như tạo/truy vấn/xác nhận/hoàn tiền giao dịch Đồng thời,
* Lưu trữ toàn bộ lịch sử request tới ví điện tử của Merchant.

1. Billing Service:

* Quản lý thanh toán về số lượng request của Merchant
* Thống kê doanh thu theo tháng, quý, năm

1. Notification Service:

* Quản lý và lưu trữ thông báo

1. Transaction Service:

* Lưu trữ chi tiết các giao dịch của Merchant

### Công cụ hỗ trợ:

* **Kafka:** Đóng vai trò message broker, chịu trách nhiệm lắng nghe và phân tán các message tới các Service tương ứng.
* **API Gateway:** Điều hướng các lời gọi API của người dùng đến dịch vụ tương ứng. Tập trung hóa các API, đảm bảo người dùng chỉ gọi qua API Gateway
* **NextJS:** Trang Dashboard, giúp cho người dùng có thể xem thông tin về phí sử dụng, quản lý các API Key. Đồng thời admin cũng có thể quản lý các user và thống kê theo tháng, quý, năm.
* **Config Server:** Nơi lưu trữ toàn bộ cấu hình tập trung của hệ thống. Các dịch vụ sẽ lấy thông tin từ config server. Đảm bảo quản lý dễ dàng và giảm sự phức tạp khi thay đổi.
* **Discovery Server:** Phát hiện các dịch vụ hiện có trong hệ thống. Cung cấp cân bằng tải (load balancing), tăng/giảm dịch vụ tương ứng với lượng yêu cầu.
* **Prothemeus:** Giám sát các truy cập của API, đồng thời đo lường sức khỏe và kiểm tra lỗi.

## Kết quả đề tài

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

## Hướng phát triển

## 

TÀI LIỆU THAM KHẢO

(Theo chuẩn IEEE – *bỏ dòng này khi viết báo cáo*)

PHỤ LỤC