**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

Ví điện tử, được nhiều người tin tưởng và sử dụng vì sự tiện lợi, bảo mật và phù hợp xu hướng xã hội. Dịch vụ mua sắm từ trực tiếp hay trực tuyến cũng đang dần mong muốn có thêm ví điện tử làm phương thức thanh toán. Tuy nhiên, để có thể triển khai và tích hợp ví điện tử là một chuyện không hề đơn giản. Doanh nghiệp sẽ phải tuân theo từng tài liệu kĩ thuật tương ứng với ví điện tử và với các công nghệ khác nhau sẽ được phát triển khác nhau, điều này sẽ gây khó khăn cả về nhân công và nguồn lực công ty.

Đồ án này đề xuất giải pháp phát triển các Rest API cho các ví điện tử phổ biến như MoMo, ZaloPay. Hệ thống sẽ đóng vai trò trung gian giữa doanh nghiệp và các ví điện tử, cung cấp Rest API tới nhiều ví điện tử phổ biến hiện nay. Đơn giản tích hợp và sử dụng là phương châm của hệ thống này.

Hệ thống sẽ bao gồm hai vai trò chính là Merchant – người sử dụng dịch vụ, có quyền gửi yêu cầu tới các Rest API, Admin – quản trị viên, có toàn quyền trong hệ thống, quản lý người dùng và các dịch vụ. Ngoài ra, để tăng tính mở rộng và linh hoạt, ứng dụng sẽ được xây dựng theo kiến trúc microservices.

Kết quả đạt được là một hệ thống hoàn chỉnh được xây dựng theo kiến trúc microservices đảm bảo hệ thống có thể nâng cấp, bảo trì và triển khai dễ dàng trong tương lai. Các chính năng chính là một bộ Rest API chuẩn cho phép tạo thanh toán, truy vấn trạng thái, xác nhận giao dịch và hoàn tiền, ngoài ra, có giao diện dashboard giúp xử lý trong việc quản lý, cả về người dùng và các tiện ích khác.

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc179934207)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc179934208)

[TÓM TẮT ĐỐ ÁN NGÀNH 3](#_Toc179934209)

[DANH MỤC TỪ VIẾT TẮT 7](#_Toc179934210)

[DANH MỤC HÌNH VẼ 8](#_Toc179934211)

[DANH MỤC BẢNG 10](#_Toc179934212)

[MỞ ĐẦU 11](#_Toc179934213)

[Chương 1. TỔNG QUAN ĐỀ TÀI 12](#_Toc179934214)

[1.1. Giới thiệu đề tài 12](#_Toc179934215)

[1.2. Mục tiêu và phạm vi đề tài 12](#_Toc179934216)

[1.2.1. Mục tiêu 12](#_Toc179934217)

[1.2.2. Phạm vi 13](#_Toc179934218)

[1.3. Phương pháp nghiên cứu 13](#_Toc179934219)

[1.3.1. Phân tích yêu cầu 13](#_Toc179934220)

[1.3.2. Xác định nghiệp vụ 13](#_Toc179934221)

[1.3.3. Lựa chọn công nghệ 13](#_Toc179934222)

[1.4. Bố cục báo cáo 13](#_Toc179934223)

[Chương 2. CƠ SỞ LÝ THUYẾT 15](#_Toc179934224)

[2.1. Giới thiệu Microservices 15](#_Toc179934225)

[2.1.1. Đặc điểm chính 15](#_Toc179934226)

[2.1.2. Kiến trúc 16](#_Toc179934227)

[2.2. Giới thiệu Spring Boot 16](#_Toc179934228)

[2.2.1. Đặc điểm chính 17](#_Toc179934229)

[2.2.2. Kiến trúc 18](#_Toc179934230)

[2.2.3. Cách sử dụng 19](#_Toc179934231)

[2.3. Giới thiệu NextJS 22](#_Toc179934232)

[2.3.1. Đặc điểm chính 23](#_Toc179934233)

[2.3.2. Kiến trúc 25](#_Toc179934234)

[2.3.3. Cách sử dụng 26](#_Toc179934235)

[2.4. Cách dịch vụ và công cụ hỗ trợ 28](#_Toc179934236)

[2.4.1. MongoDB 28](#_Toc179934237)

[2.4.2. PostgreSQL 28](#_Toc179934238)

[2.4.3. Docker 29](#_Toc179934239)

[Chương 3. PHÁT TRIỂN CÁC REST API CHO VÍ THANH TOÁN TRỰC TUYẾN SỬ DỤNG KIẾN TRÚC MICROSERVICES 30](#_Toc179934240)

[3.1. Giới thiệu bài toán 30](#_Toc179934241)

[3.2. Phân tích hệ thống 30](#_Toc179934242)

[3.2.1. Lược đồ use case 30](#_Toc179934243)

[3.2.2. Đặc tả use case 31](#_Toc179934244)

[3.3. Thiết kế hệ thống 36](#_Toc179934245)

[3.3.1. Sơ đồ tuần tự (sequence diagram) 36](#_Toc179934246)

[3.3.2. Sơ đồ hoạt động (activity diagram) 39](#_Toc179934247)

[3.3.3. Thiết kế dữ liệu 43](#_Toc179934248)

[3.4. Kiến trúc hệ thống 44](#_Toc179934249)

[3.4.1. User Service: 45](#_Toc179934250)

[3.4.2. Payment Gateway Service: 45](#_Toc179934251)

[3.4.3. Billing Service: 45](#_Toc179934252)

[3.4.4. Notification Service: 46](#_Toc179934253)

[3.4.5. Transaction Service: 46](#_Toc179934254)

[3.4.6. Công cụ hỗ trợ: 46](#_Toc179934255)

[3.5. Kết quả đề tài 46](#_Toc179934256)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 53](#_Toc179934257)

[4.1. Kết luận 53](#_Toc179934258)

[4.1.1. Tổng quát 53](#_Toc179934259)

[4.1.2. Hạn chế 53](#_Toc179934260)

[4.2. Hướng phát triển 53](#_Toc179934261)

[TÀI LIỆU THAM KHẢO 54](#_Toc179934262)

DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **Từ viết tắt** | **Ý nghĩa** |
| 1 | CSDL | Cơ sở dữ liệu |
| 2 | MVC | Model – View - Controller |
| 3 | SSR | Server-Side Rendering |
| 4 | SSG | Static Site Generation |
| 5 | CI | Continuous Integration |
| 6 | CD | Continuous Development |
| 7 | SEO | Search Engine. Optimization |

DANH MỤC HÌNH VẼ

[Hình 2‑1. Kiến trúc ứng dụng Microservices 16](#_Toc179934175)

[Hình 2‑2. Kiến trúc ứng dụng Spring Boot 18](#_Toc179934176)

[Hình 2‑3. Luồng đi của Spring Boot 19](#_Toc179934177)

[Hình 2‑4. Spring initializr 20](#_Toc179934178)

[Hình 2‑5. Cấu trúc dự án Spring Boot 21](#_Toc179934179)

[Hình 2‑6. Controller Spring Boot 22](#_Toc179934180)

[Hình 2‑7. Chương trình Spring Boot đầu tiên 22](#_Toc179934181)

[Hình 2‑8. NextJS Routing 23](#_Toc179934182)

[Hình 2‑9. NextJS dynamic routes 23](#_Toc179934183)

[Hình 2‑10. Kiến trúc ứng dụng NexJS 25](#_Toc179934184)

[Hình 2‑11. Cấu trúc ứng dụng NextJS 26](#_Toc179934185)

[Hình 2‑12. Tạo dự án NextJS 26](#_Toc179934186)

[Hình 2‑13. Lựa chọn dự án NextJS 26](#_Toc179934187)

[Hình 2‑14. Cấu trức ứng dụng NextJS đầu tiên 27](#_Toc179934188)

[Hình 2‑15. Ứng dụng NextJS đầu tiên 28](#_Toc179934189)

[Hình 3‑1. Lược đồ use case 31](#_Toc179934190)

[Hình 3‑2. Sơ đồ tuần tự tạo thanh toán 36](#_Toc179934191)

[Hình 3‑3. Sơ đồ tuần tự xác nhận giao dịch 36](#_Toc179934192)

[Hình 3‑4. Sơ đồ tuần tự tra cứu trạng thái giao dịch 37](#_Toc179934193)

[Hình 3‑5. Sơ đồ tuần tự hoàn tiền giao dịch 38](#_Toc179934194)

[Hình 3‑6. Sơ đồ hoạt động tạo thanh toán 39](#_Toc179934195)

[Hình 3‑7. Sơ đồ trạng thái xác nhận giao dịch 40](#_Toc179934196)

[Hình 3‑8. Sơ đồ hoạt động tra cứu trạng thái giao dịch 41](#_Toc179934197)

[Hình 3‑9. Sơ đồ hoạt động hoàn tiền giao dịch 42](#_Toc179934198)

[Hình 3‑10. Lược đồ cơ sở dữ liệu 43](#_Toc179934199)

[Hình 3‑11. Kiến trúc hệ thống 45](#_Toc179934200)

[Hình 3‑12. API tạo thanh toán 47](#_Toc179934201)

[Hình 3‑13. API truy vấn trạng thái giao dịch 48](#_Toc179934202)

[Hình 3‑14. API xác nhận thanh toán 49](#_Toc179934203)

[Hình 3‑15. API hoàn tiền giao dịch 50](#_Toc179934204)

[Hình 3‑16. Giao diện quản lý Access Key 51](#_Toc179934205)

[Hình 3‑17. Giao diện quản lý Merchant 51](#_Toc179934206)

DANH MỤC BẢNG

[Bảng 3‑1. Use case tạo thanh toán 32](#_Toc179934171)

[Bảng 3‑2. Use case xác nhận giao dịch 33](#_Toc179934172)

[Bảng 3‑3. Use case tra cứu trạng thái 34](#_Toc179934173)

[Bảng 3‑4. Use case hoàn tiền giao dịch 35](#_Toc179934174)

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 các Rest API cho ví thanh toán trực tuyến sử dụng kiến trúc microservices. 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 rest api cho ví thanh toán trực tuyến sử dụng kiến trúc microservices, 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, …)

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: Phát triển các Rest API cho ví thanh toán trực tuyến sử dụng kiến trúc Microservices

Chương 4: Kết Luận Và Hướng Phát Triển

# CƠ SỞ LÝ THUYẾT

## Giới thiệu Microservices

Microservices là một thiết kế phần mềm, tập trung vào việc phát triển hệ thống thành những dịch vụ nhỏ, độc lập và không phụ thuộc lẫn nhau, tuy nhiên vẫn đảm bảo kết nối chặt chẽ khi cần thiết.

Microservices ra đời nhằm giải quyết vấn đề của thiết kế nguyên khối (monolithic). Kiến trúc này gói gọn toàn bộ vào một hệ thống duy nhất, gây khó khăn trong việc bảo trì, mở rộng và triển khai.

Microservices đưa ra giải pháp bằng cách phát triển theo từng module riêng biệt, tập trung vào một chức năng hoặc nghiệp vụ cụ thể. Cách này giúp tăng tốc độ phát triển, linh hoạt về công nghệ và triển khai độc lập.

### Đặc điểm chính

#### Độc lập

Mỗi dịch vụ là một đơn vị độc lập, có thể được phát triển và triển khai riêng biệt. Cho phép các chức năng được phát triển song song với nhau. Đồng thời, mỗi dịch vụ chỉ cần quan tâm về nghiệp vụ của riêng mình, đảm bảo việc nâng cấp và bảo trì không ảnh hưởng tới các dịch vụ khác.

#### Linh hoạt

Các dịch vụ có thể được phát triển trên nhiều công nghệ khác nhau, tùy thuộc vào chức năng của nghiệp vụ. Điều này giúp tăng tốc độ phát triển và bắt kịp xu hướng của công nghệ.

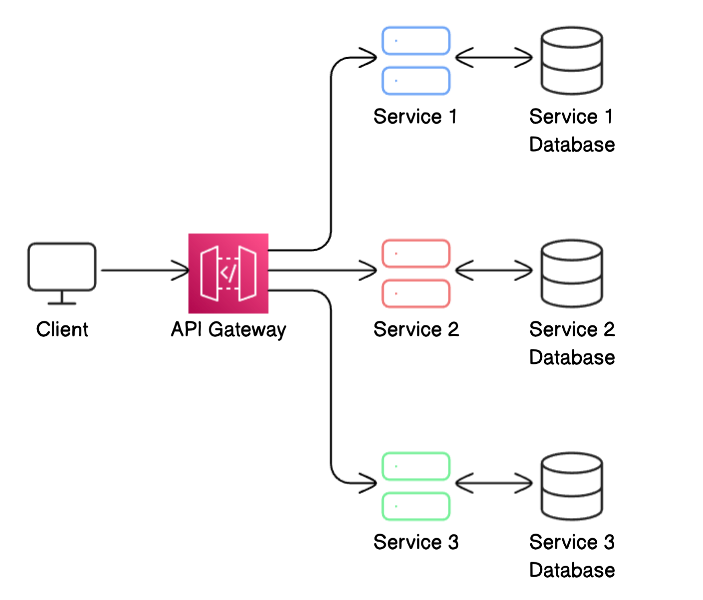
#### Mở rộng

Dễ dàng mở rộng quy mô tùy thuộc theo nhu cầu sử dụng, kết hợp với nhiều công cụ như Docker, Kubernetes, giúp hệ thống mở rộng linh hoạt.

#### Chịu lỗi

Các dịch vụ làm việc riêng lẻ với nhau, khi một dịch vụ cần bảo trì hoặc gặp sự cố sẽ không ảnh hưởng tới các dịch vụ khác, đảm bảo hệ thống luôn trong tình trạng hoạt động xuyên suốt.

### Kiến trúc



Hình ‑. Kiến trúc ứng dụng Microservices

Hệ thống Microservices sẽ gồm nhiều services con khác nhau, mỗi service sẽ đảm nhận một chức năng chuyên biệt, đồng thời sẽ có riêng CSDL. Để có thể giao tiếp các services với nhau, có thể sử dụng một số cách phổ biến như Rest API, gRPC, …

API Gateway là điểm truy cập duy nhất, khi client gửi yêu cầu tới các chức năng, API Gateway sẽ tự động mapping và gán vào các service tương ứng. Ngoài ra, có thể tích hợp thêm lớp bảo mật và cân bằng tải.

## 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

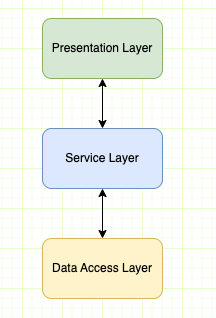
Spring Boot cung cấp bộ cung cụ sẵn sàn cho môi trường sản xuất.

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 ‑. Kiến trúc ứng dụng 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 các lớp filter để lọc các request và bảo vệ hệ thống.

**Service Layer:** xử lý các business logic, là 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 ‑. 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 tìm và 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.

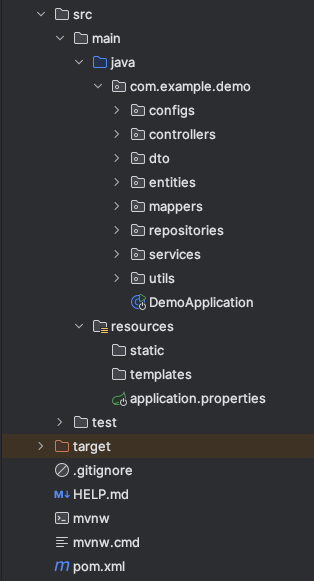


Hình ‑. Spring initializr

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 ‑. Cấu trúc dự án Spring Boot

Thành phần chính:

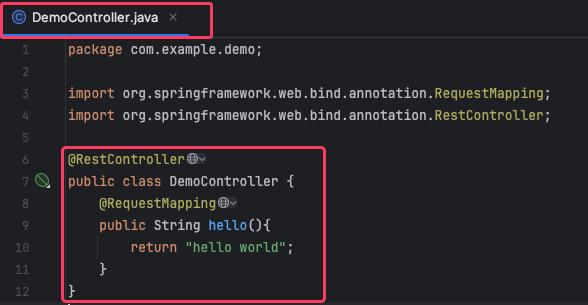
**Src/main/java/:** Chứa toàn bộ mã nguồn chính của hệ thống, bao gồm các lớp như Controller, Repository, Service, Entity và các thành phần khác.

**Src/main/resources/:** Chứa tài nguyên tĩnh và cấu hình của ứng dụng.

**Src/test:** Chứa tệp kiểm thử của ứng dụng.

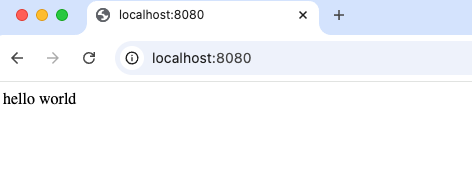
**Pom.xml:** Nơi quản lý tất cả các dependencies.

Ta tạo một controller đơn giản



Hình ‑. Controller Spring Boot

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



Hình ‑. Chương trình Spring Boot đầu tiên

## Giới thiệu NextJS

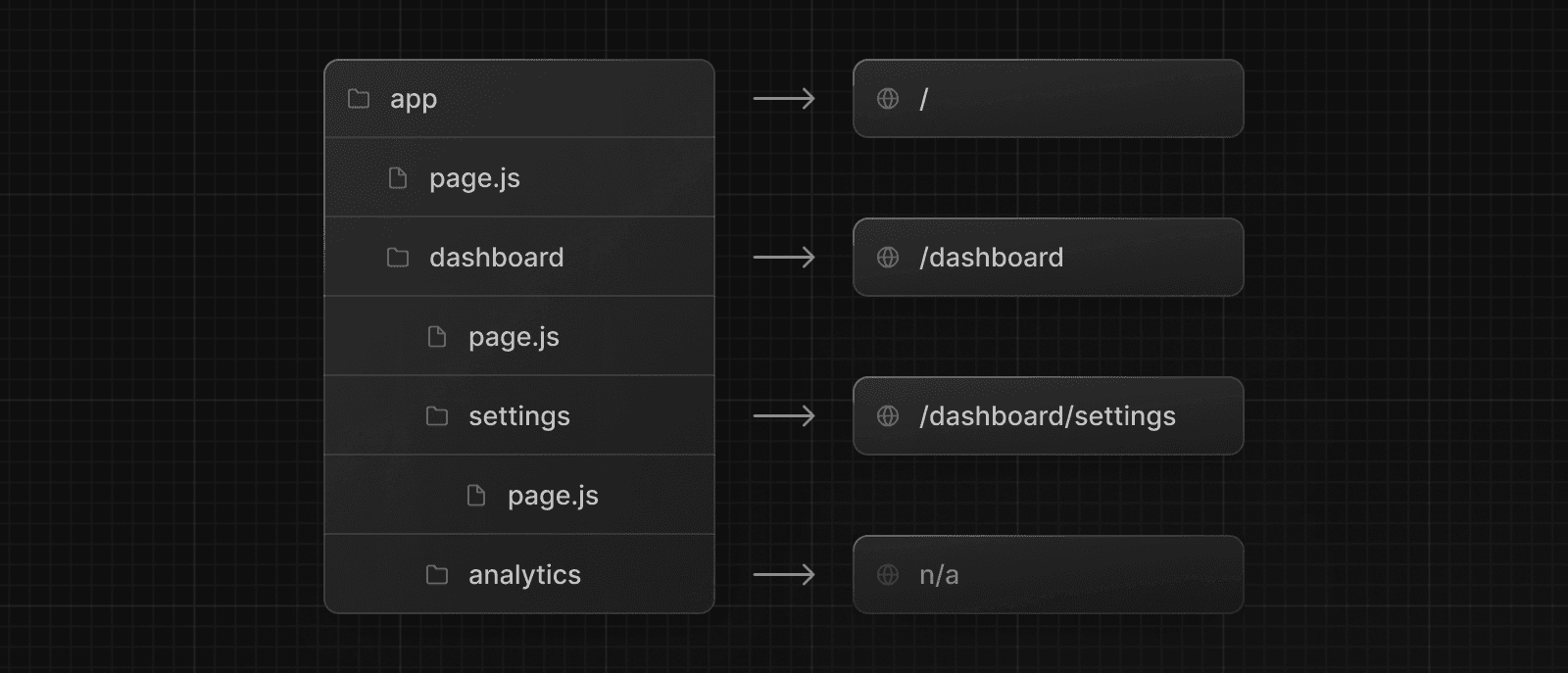
NextJS là một framework React được sử dụng phổ biến bởi tính linh hoạt và gọn nhẹ. Nó hỗ trợ mạnh mẽ trong việc phát triển website một cách nhanh chóng, đồng thời tối ưu SEO và đem lại trải nghiệm người dùng tốt nhất.

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 ‑. 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 ‑. 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.

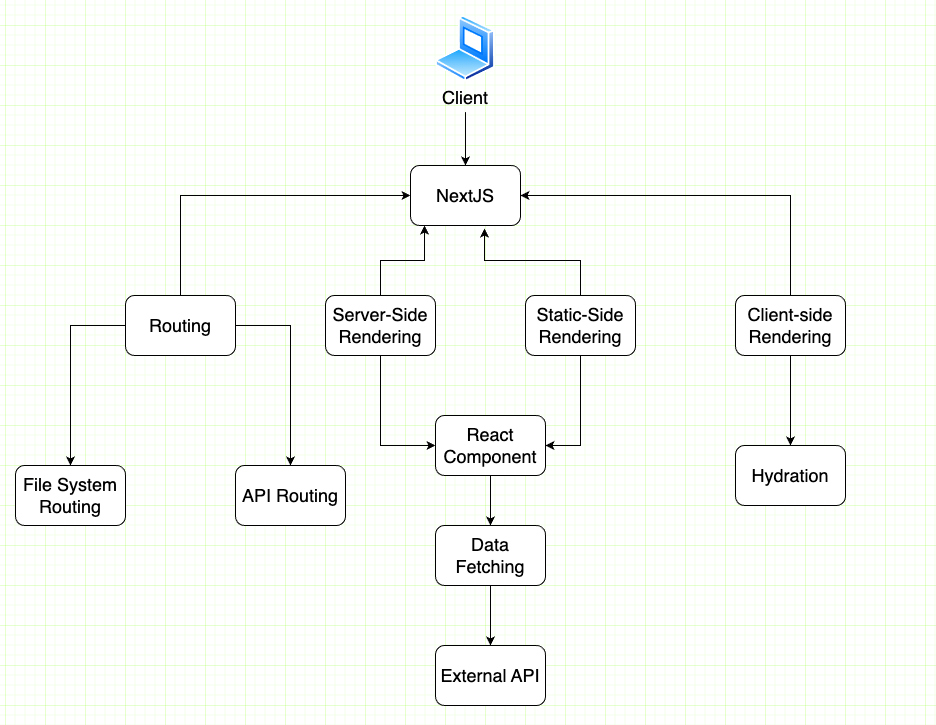
#### Styling

NextJS hỗ trợ đa dạng nhiều loại styling, bao gồm: CSS modules, Global CS, Tailwind CSS, Sass và CSS-in-JS.

#### Typescript

NextJS được tích hợp sẵn Typescript, hỗ trợ tốt trong việc kiểm tra kiểu dữ liệu.

### Kiến trúc



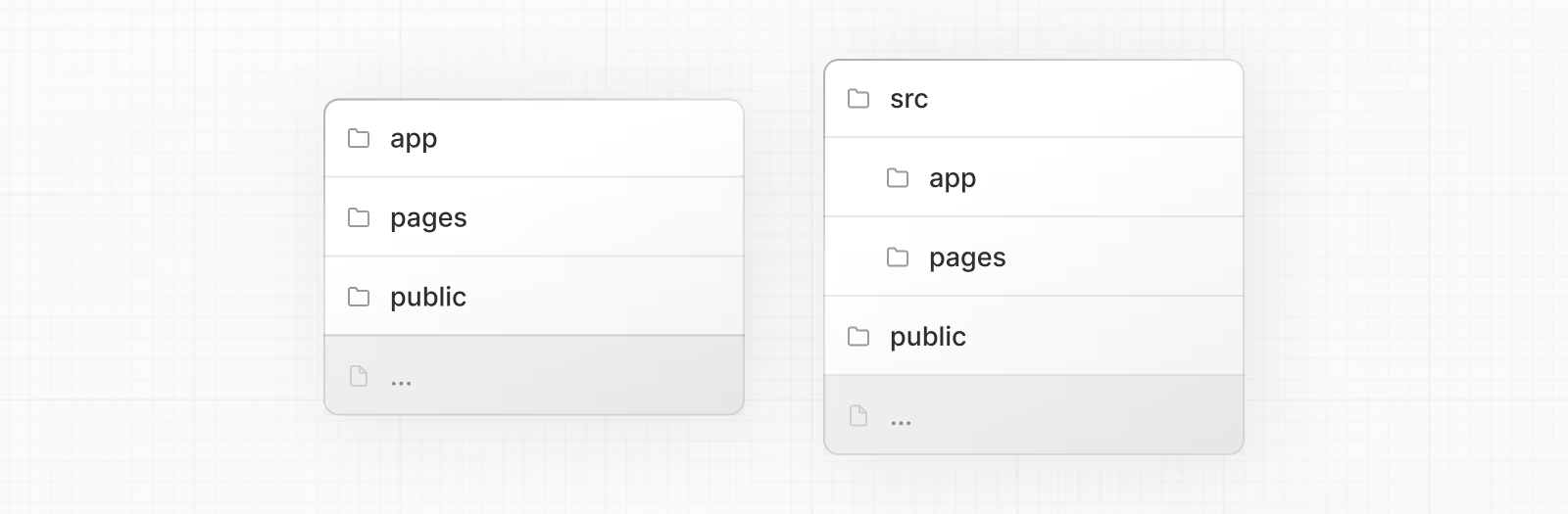
Hình ‑. Kiến trúc ứng dụng NexJS

NextJS sử dụng routing theo thư mục để định tuyến, mỗi tệp trong thư mục pages sẽ trở thành một route, ví dụ pages/about.js sẽ tương đương với /about. Ngoài ra, NextJS cho phép xây dựng các API endpoint trong cùng một cấu trúc thư mục.

NextJS hỗ trợ cơ chế rendering mạnh mẽ cả về client và server. Server-Side Rendering sẽ tạo một tranh HTML đầy đủ, điều này cải thiện được tốc độ tải trang và tăng khả năng SEO, kết hợp thêm Static-Side Rendering, sẽ tạo các trang tĩnh tại thời điểm build.

Phía Client, để giúp giảm tải cho server, NextJS sử dụng Client-Side Rendering, trả về các javascript bundle, giúp tương tác javascript trực tiếp trên client và fetching xuống server khi cần thay đổi động.

### Cách sử dụng



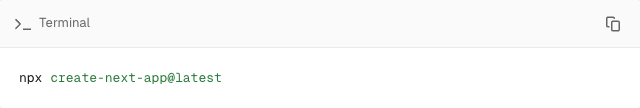
Hình ‑. Cấu trúc ứng dụng NextJS

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

**src/:** Chứa mã nguồn chính của hệ thống.

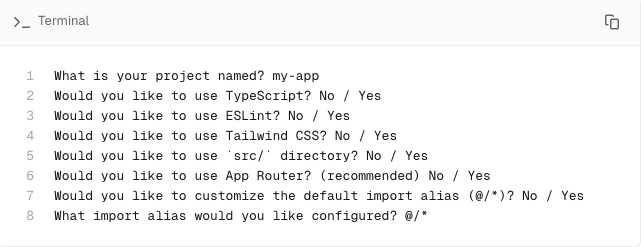
**public/:** Dùng để lưu trữ tài nguyên tình như hình ảnh, …

Để có thể sử dụng NextJS, ta cần cài đặt NodeJS, sau đó ta vào terminal và sử dụng lệnh



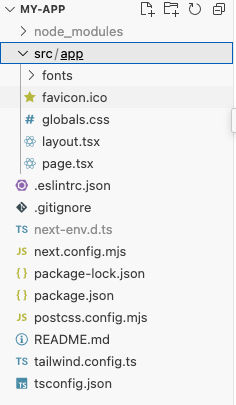
Hình ‑. Tạo dự án NextJS

Terminal sẽ hiện lên các options



Hình ‑. Lựa chọn dự án NextJS

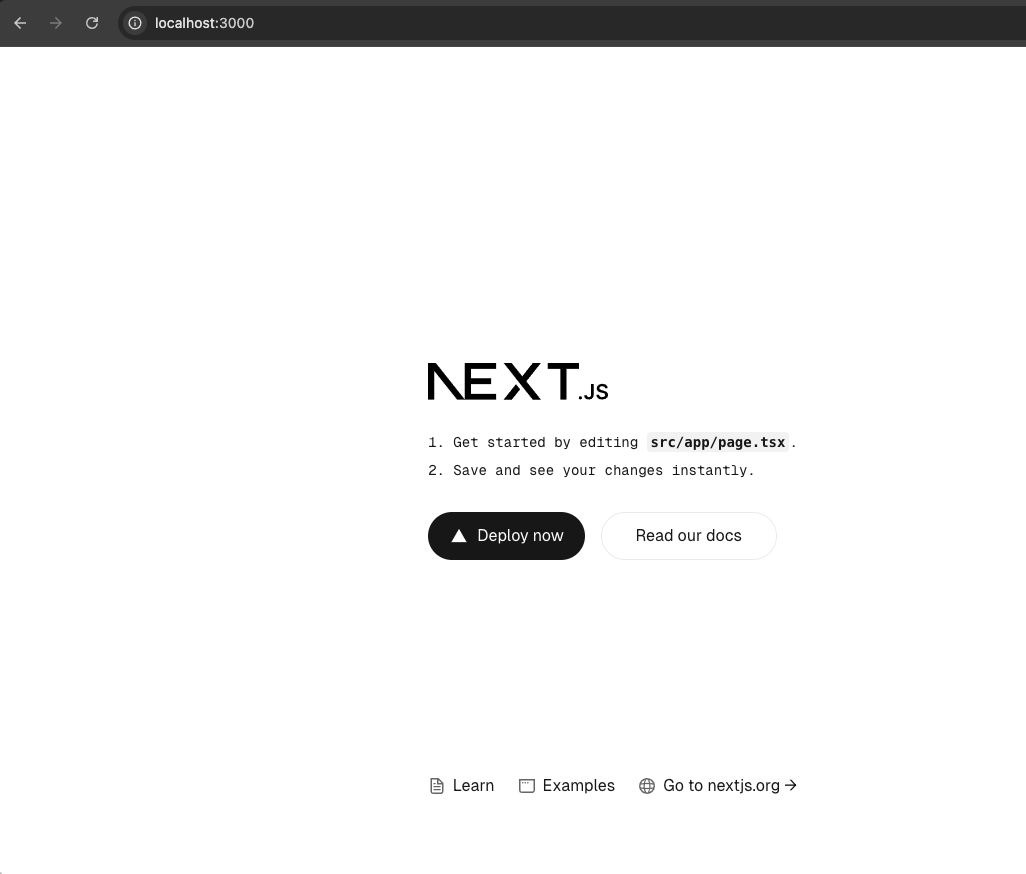
Ta lựa chọn theo nhu cầu, hoàn thành, ta sẽ có được cấu trúc thư mục đơn giản



Hình ‑. Cấu trức ứng dụng NextJS đầu tiên

Ta sử dụng npm install hoặc yarn để cài đặt các dependencies cần thiết

Để chạy ứng dụng, ta sử dụng npm run dev hoặc yarn dev



Hình ‑. Ứng dụng NextJS đầu tiên

## 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

Đóng gói ứng dụng vào container, giúp tách biệt môi trường developer và production, lý tưởng cho quy trình CI/CD. Bên cạnh đó docker cũng cung cấp khả năng cân bằng tải (load balancing), thích hợp cho hệ thống microservices.

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

## 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:

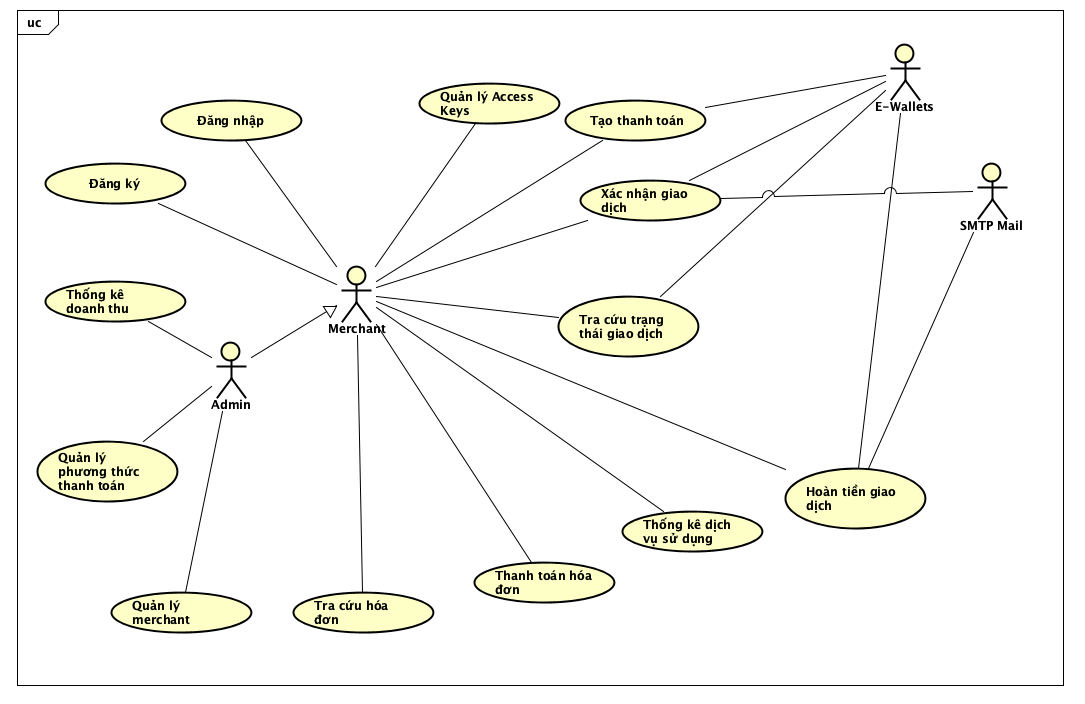
Tạo thanh toán: Tạo thanh toán tới các ví điện tử tương ứng

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 ‑. 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 ‑. 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 ‑. 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 ‑. Use case tra cứu trạng thái

#### Use case hoàn tiền giao dịch

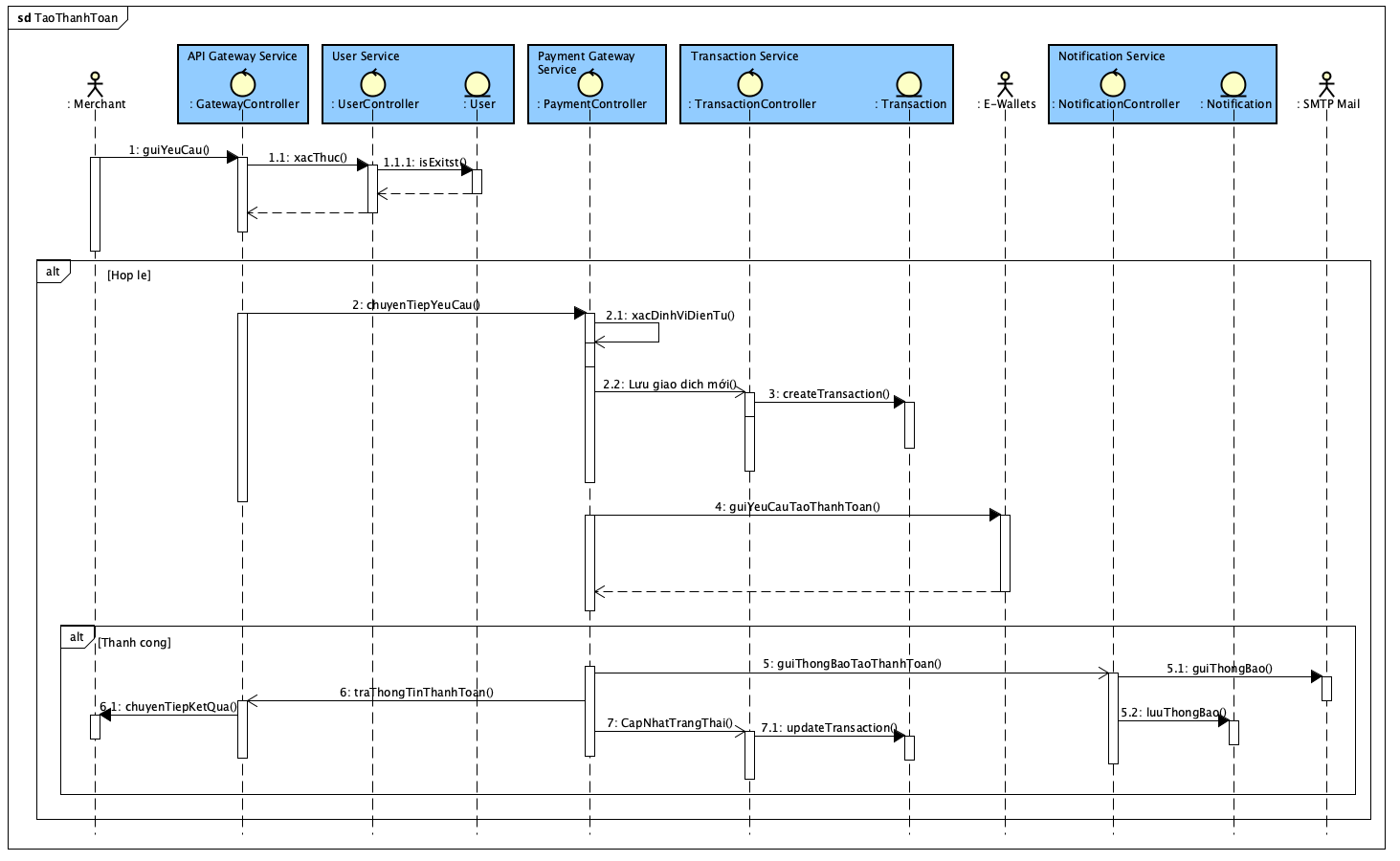
|  |  |
| --- | --- |
| **Use case Id** | UC-04 |
| **Tên use case** | Hoàn tiền giao dịch |
| **Mô tả** | Merchant gửi yêu cầu hoàn tiền giao dịch |
| **Actor chính** | Merchant |
| **Actor Phụ** | E-Wallets, SMTP Mail |
| **Tiền điều kiện** | Không |
| **Hậu điều kiện** | Merchant nhận được tiền hoàn |
| **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 ‑. Use case hoàn tiền giao dịch

## Thiết kế hệ thống

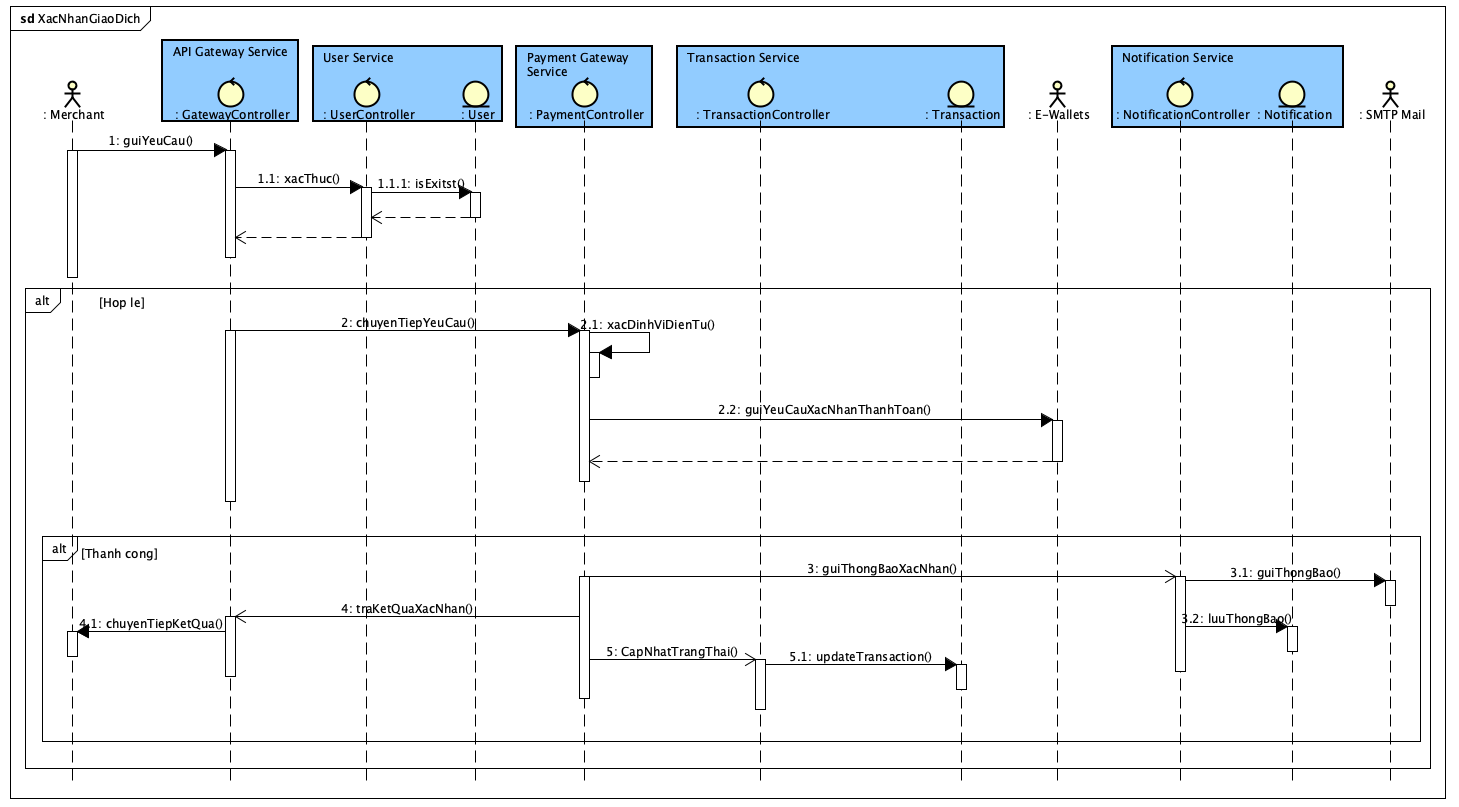
### Sơ đồ tuần tự (sequence diagram)

#### Tạo thanh toán



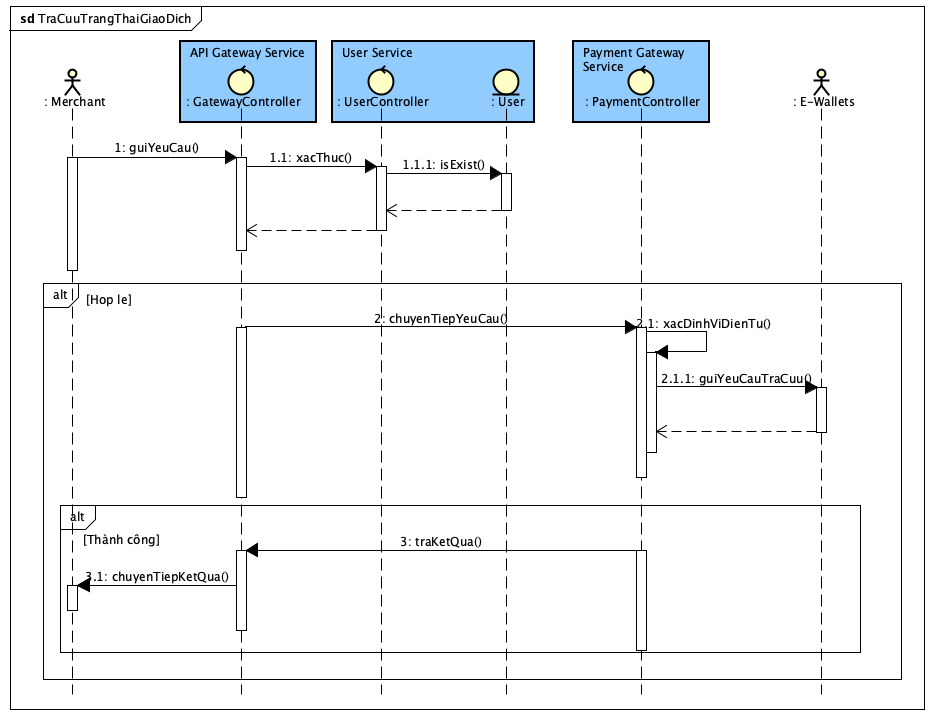
Hình ‑. Sơ đồ tuần tự tạo thanh toán

#### Xác nhận giao dịch



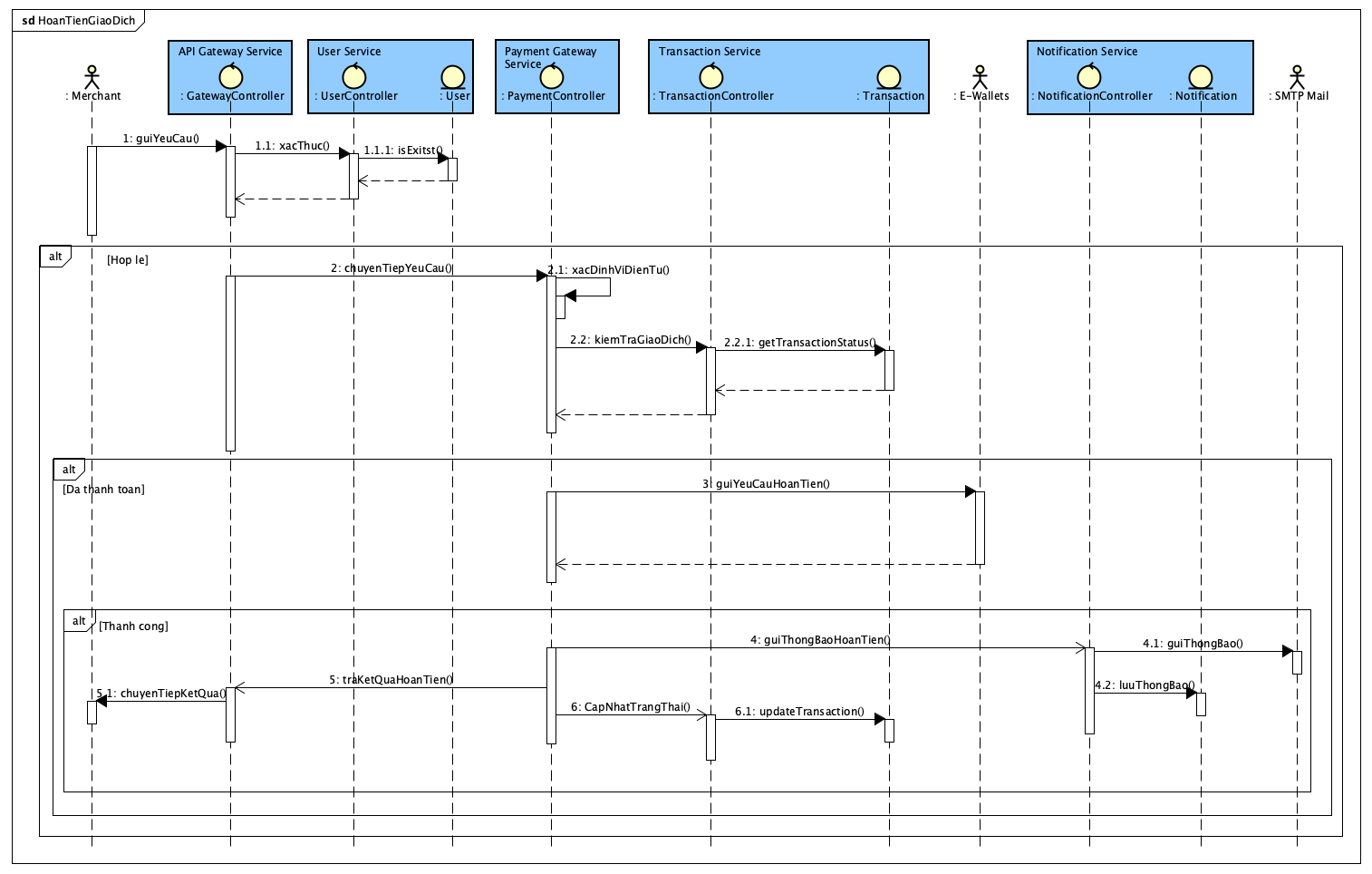
Hình ‑. Sơ đồ tuần tự xác nhận giao dịch

#### Tra cứu trạng thái giao dịch



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

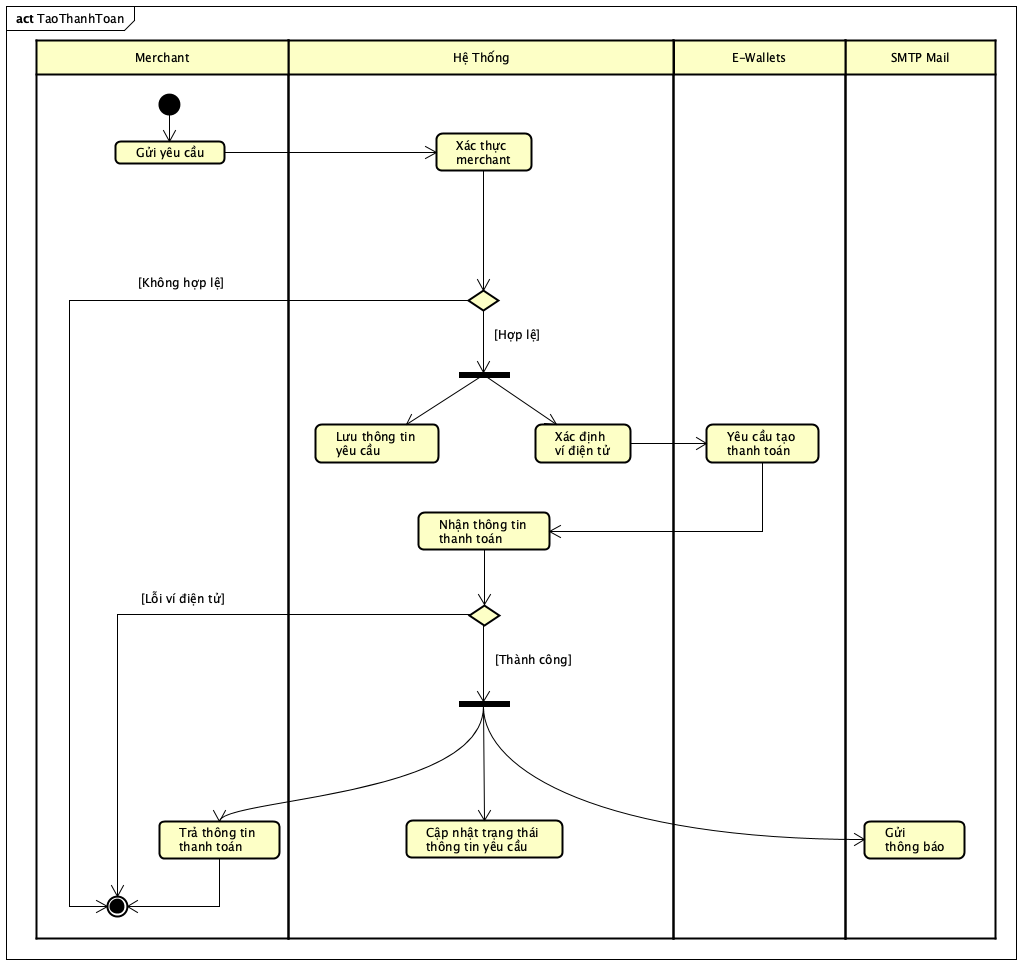
#### Hoàn tiền giao dịch



Hình ‑. Sơ đồ tuần tự hoàn tiền giao dịch

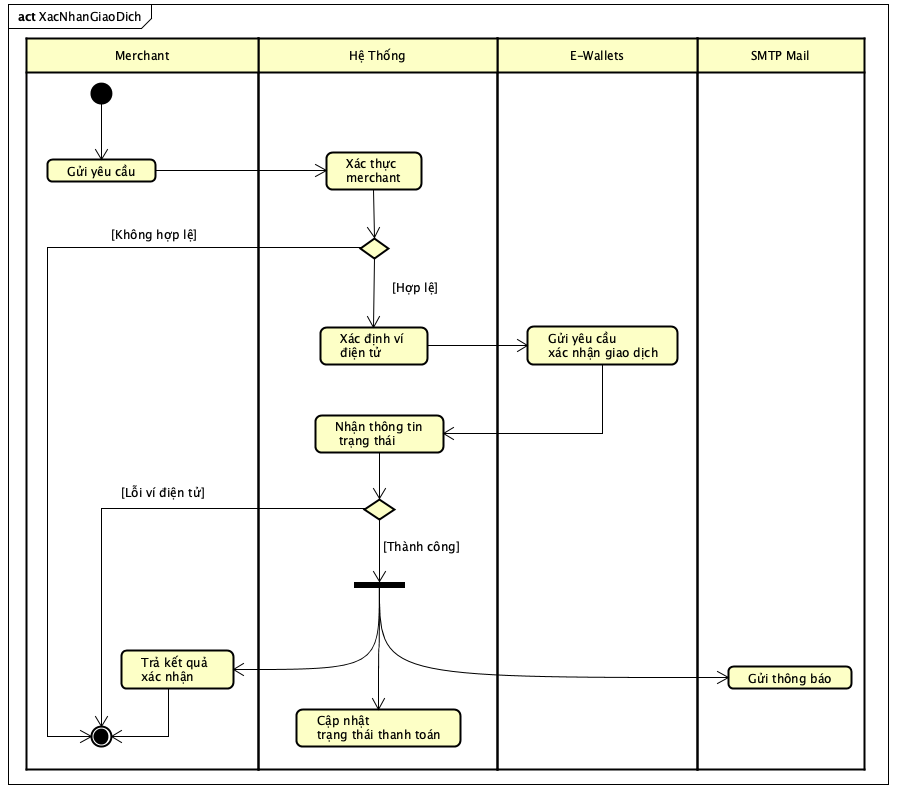
### Sơ đồ hoạt động (activity diagram)

#### Tạo thanh toán



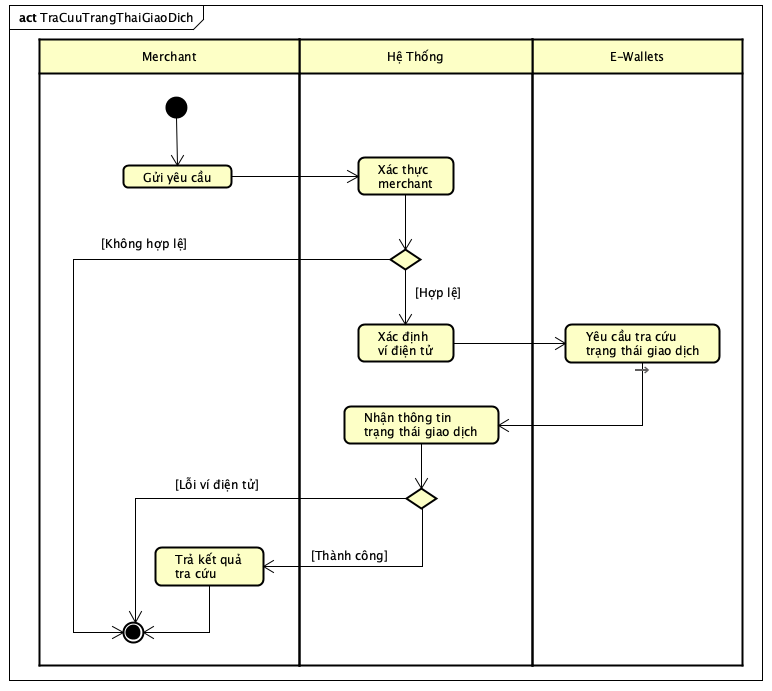
Hình ‑. Sơ đồ hoạt động tạo thanh toán

#### Xác nhận giao dịch



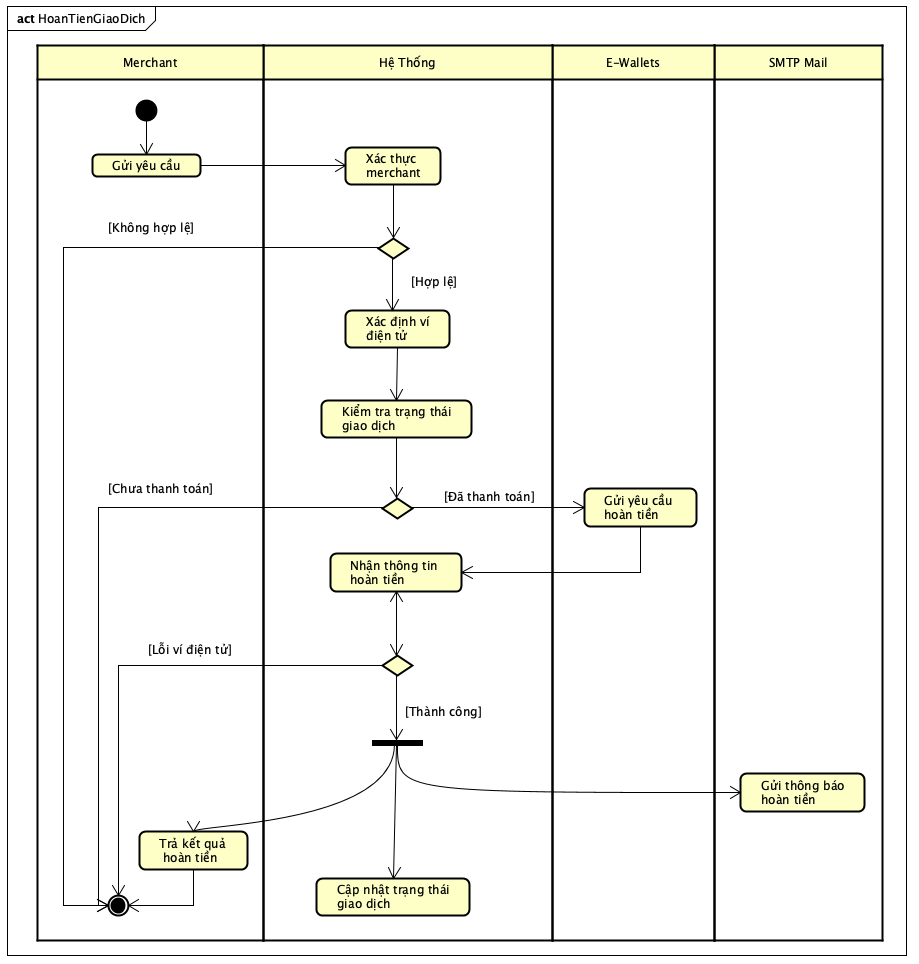
Hình ‑. Sơ đồ trạng thái xác nhận giao dịch

#### Tra cứu trạng thái giao dịch



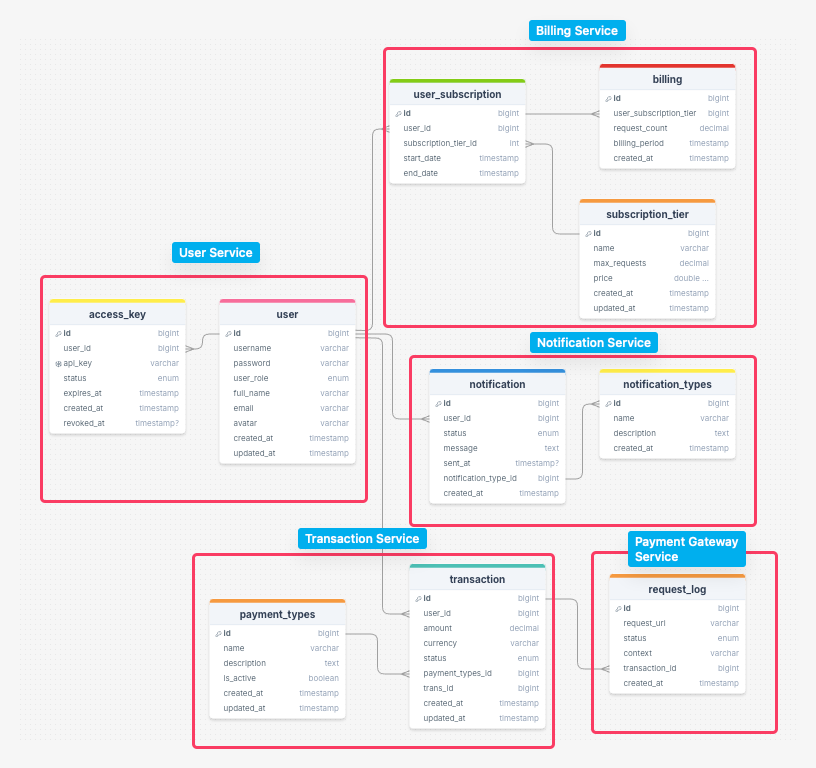
Hình ‑. Sơ đồ hoạt động tra cứu trạng thái giao dịch

#### Hoàn tiền giao dịch



Hình ‑. Sơ đồ hoạt động hoàn tiền giao dịch

### Thiết kế dữ liệu

­

Hình ‑. Lược đồ cơ sở dữ liệu

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

User: Chứa thông tin người dùng

Access Key: Chứa thông tin khóa truy cập của người dùng

#### Transaction Service

Transaction: Chứa thông tin giao dịch tới các ví diện tử

Payment Types: Chứa các phương thức thanh toán

#### Notification Service

Notification: Chứa thông tin thông báo

Notification Types: Chứa các loại thông báo

#### Payment Gateway Service

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

#### Billing Service

Billing: Chứa thông tin hóa đơn của người dùng

User Subscription: Chứa thông tin gói đăng ký của người dùng

Subscription Tier: Chứa thông tin các gói đăng ký

#### 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 – User Subscription: (1-N) Một người dùng có thể đăng ký nhiều gói, một gói đăng ký 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.

User Subscription – Billing: (1-N) Một gói đăng ký của người dùng có thể có nhiều hóa đơn, một hóa đơn chỉ thuộc về một gói đăng ký.

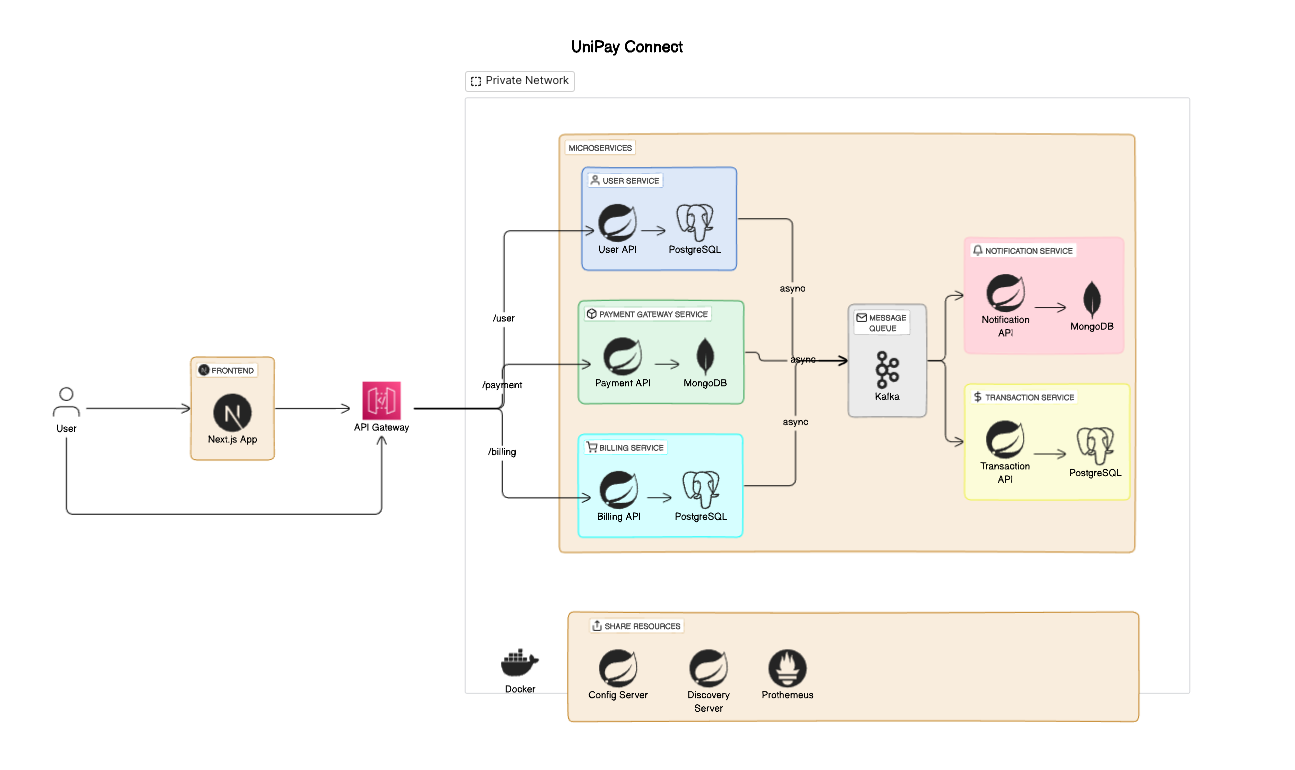
Subscription Tier – User Subscription: (1-N) Một gói đăng ký có thể được đăng ký bỏi nhiều người dùng, một gói đăng ký của người dùng chỉ thuộc về một gói đăng ký.

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 ‑. 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.

### User Service:

Có chức năng chính là lưu trữ người dùng, ngoài ra có thêm các chức năng về chứng thực và đăng ký người dùng

### Payment Gateway Service:

Trung gian giữa hệ thống và các ví điện tử. Dịch vụ này sẽ tích hợp các phương thức thanh toán tới các ví điện tử. Đồng thời, lưu trữ toàn bộ lịch sử request tới ví điện tử của người dùng

### Billing Service:

Quản lý về các gói đăng ký của Merchant. Admin có thể chỉnh sửa và tạo mới các gói đăng ký. Vào mỗi cuối chu kỳ đăng ký của người dùng, sẽ tạo hóa đơn tương ứng với gói đăng ký.

### Notification Service:

Lắng nghe các message từ các dịch vụ khác thông qua Kafka, sử dụng SMTP để gửi thông báo tới người dùng, đồng thời lưu trữ tất cả thông báo đã gửi đi.

### Transaction Service:

Chịu trách nhiệm lưu trữ toàn bộ giao dịch của người dùng tới các ví điện tử.

### Công cụ hỗ trợ:

**Kafka:** Đóng vai trò message broker, chịu trách nhiệm lắng nghe và phân tán các message tới các dịch vụ 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.

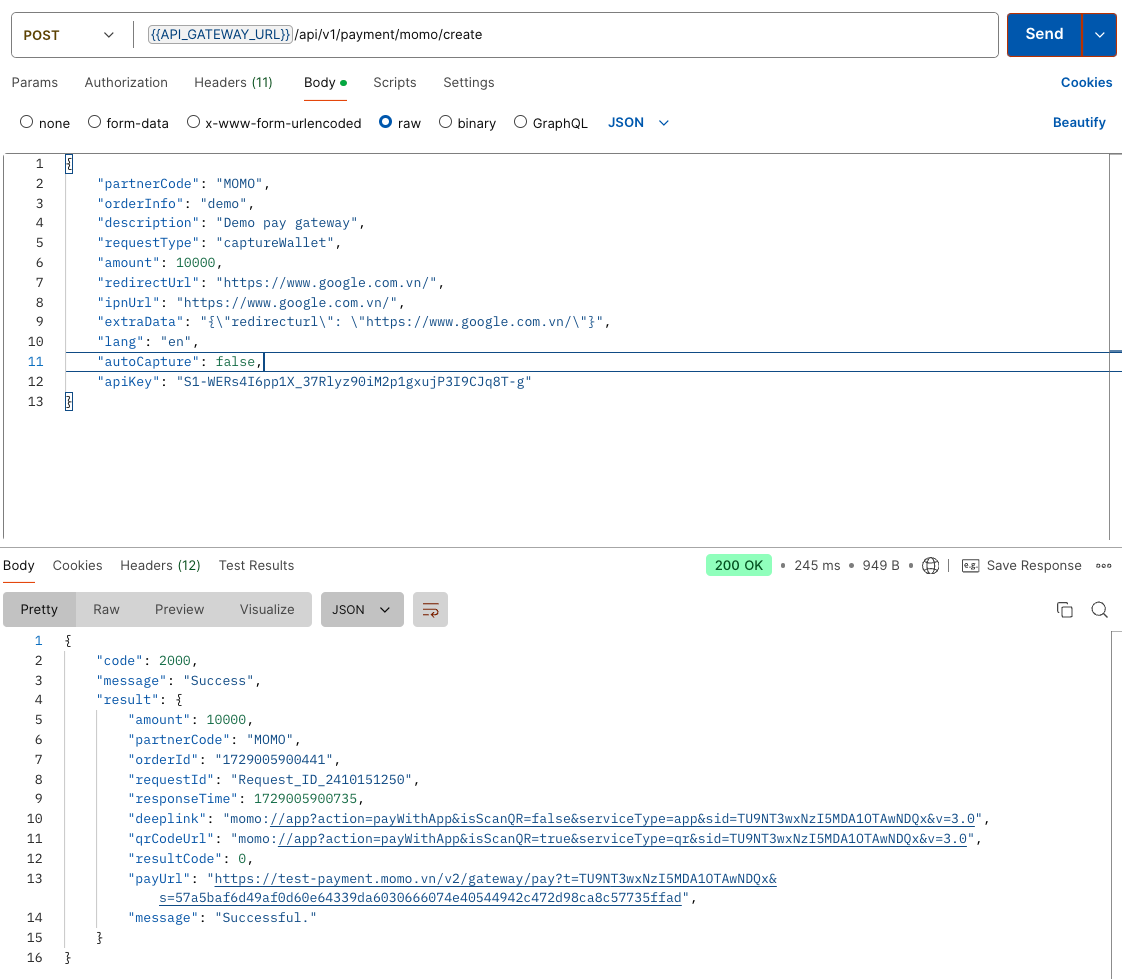
**Config Server:** Nơi lưu trữ toàn bộ cấu hình tập trung của hệ thống. Các dịch vụ sẽ lấy thông tin từ config server. Đảm bảo quản lý dễ dàng và giảm sự phức tạp khi thay đổi.

**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

Tất cả API sẽ được truy cập trực tiếp thông qua API Gateway, đảm bảo nhất quán và tập trung hóa, đồng thời bảo vệ và lọc các yêu cầu không hợp lệ.



Hình ‑. API tạo thanh toán

Request gửi lên hệ thống sẽ yêu cầu các thông tin:

**Partner Code:** Mã đối tác của ví điện tử

**Order Info:** Thông tin về đơn hàng

**Description:** Mô tả chi tiết về đơn hàng

**Request Type:** Loại request tới dịch vụ ví điện tử

**Amount:** số lượng tiền cần tạo

**Redirect Url:** Đường dẫn hệ thống ví điện tử sẽ chuyển tiếp sau khi thanh toán

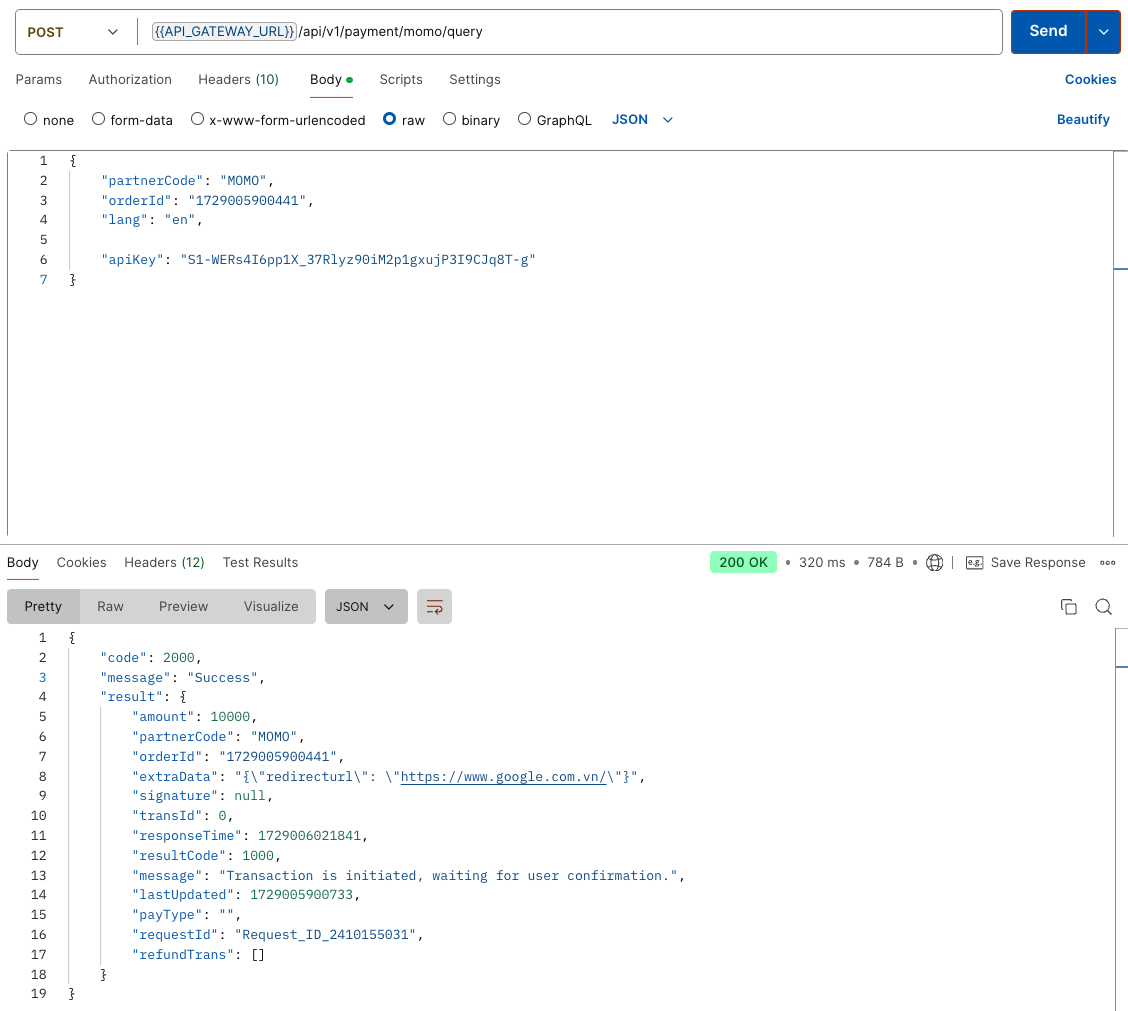
**Ipn Url:** Đường dẫn lắng nghe, Ví điện tử sẽ gọi đến đường dẫn này sau khi giao dịch thành công

**Extra Data:** Thông tin thêm về dữ liệu

**Lang:** Ngôn ngữ của response trả về

**Auto Capture:** True - nếu như muốn hệ thống ví điện tử tự động xác nhận, False – nếu như Merchant muốn tự xác nhận.

**Api Key:** Key bí mật, nhằm xác thực danh tính của Merchant



Hình ‑. API truy vấn trạng thái giao dịch

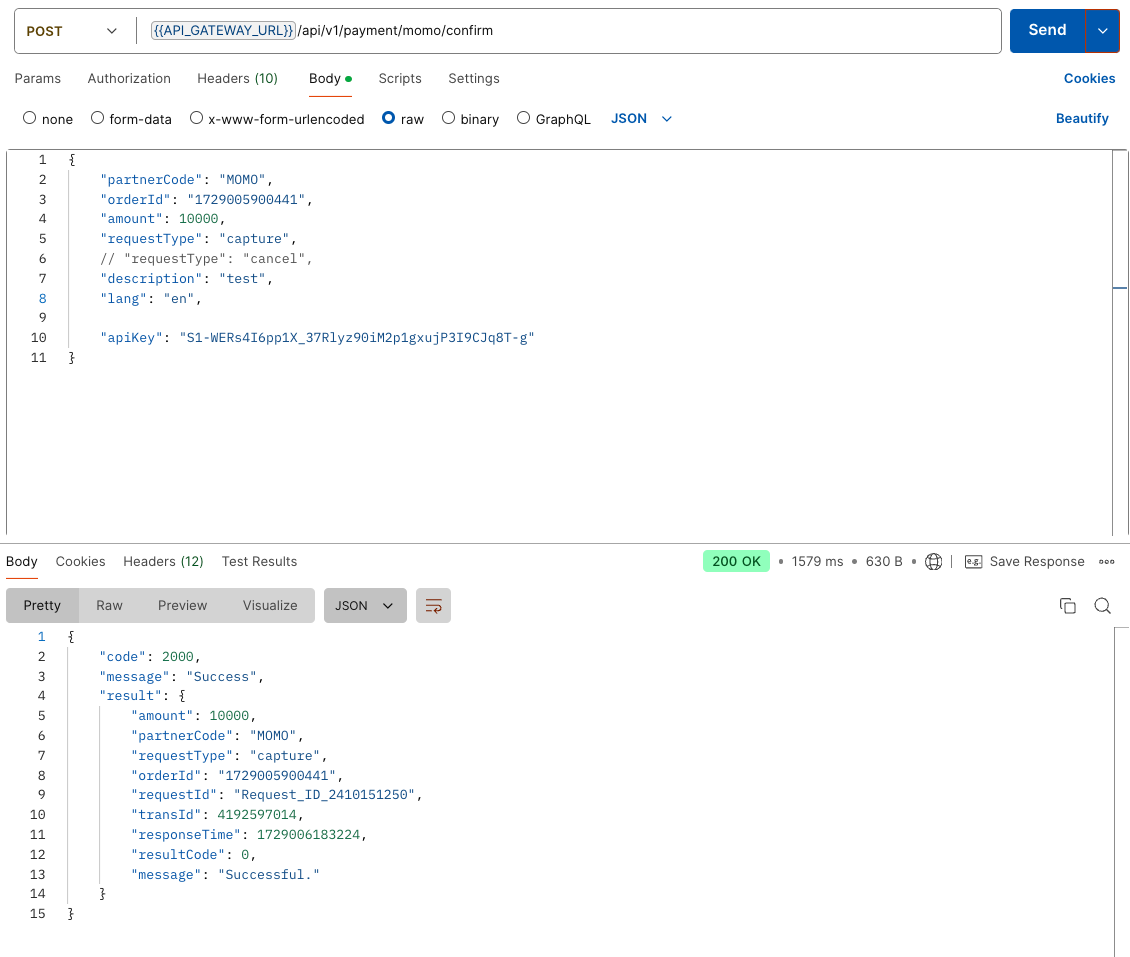
Request gửi lên hệ thống sẽ yêu cầu các thông tin:

**Partner Code:** Mã đối tác của ví điện tử

**Order Id:** Mã đơn hàng duy nhất của giao dịch

**Lang:** Ngôn ngữ của response trả về

**Api Key:** Key bí mật, nhằm xác thực danh tính của Merchant



Hình ‑. API xác nhận thanh toán

Request gửi lên hệ thống sẽ yêu cầu các thông tin:

**Partner Code:** Mã đối tác của ví điện tử

**Order Id:** Mã đơn hàng duy nhất của giao dịch

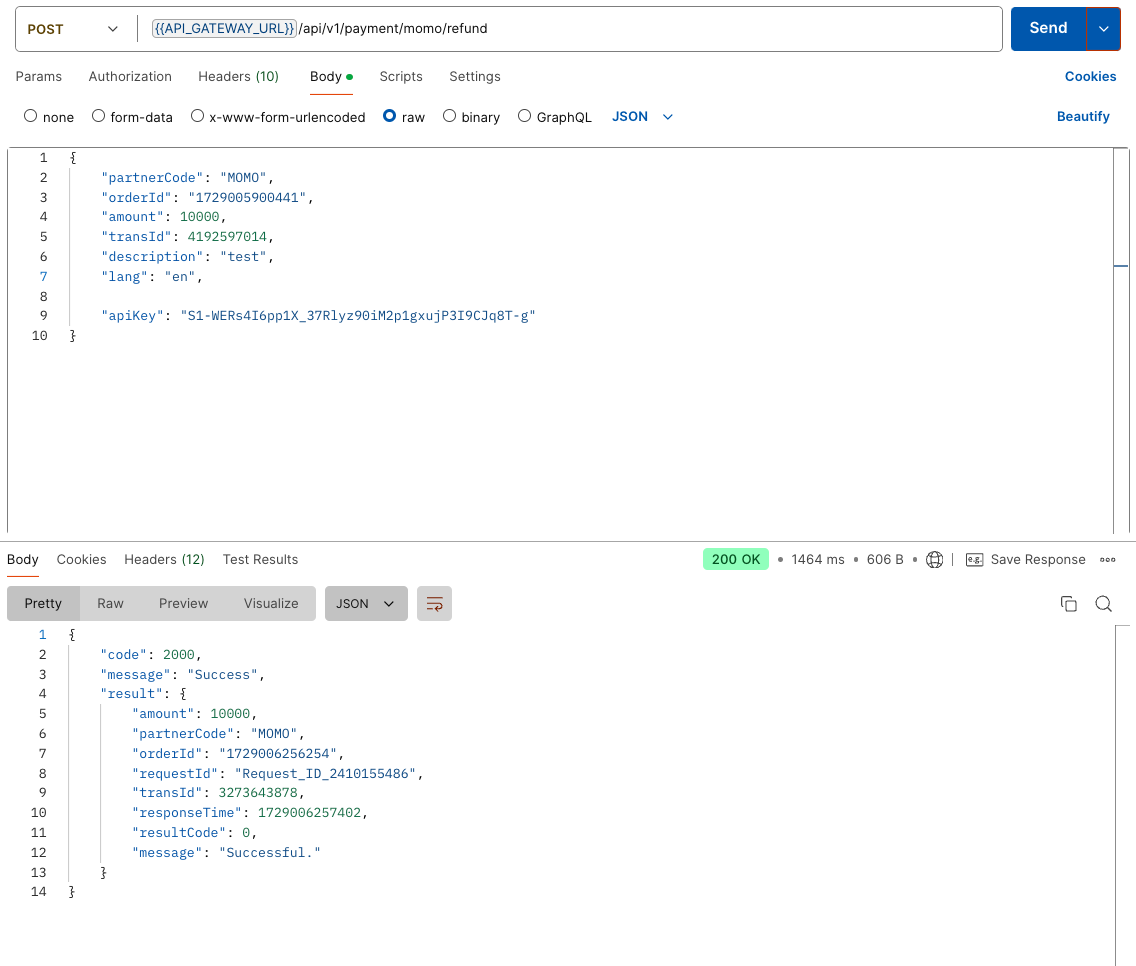
**Amount:** Số tiền cần xác nhận, có thể xác nhận toàn bộ hoặc một phần

**Request Type:** Loại request tới dịch vụ ví điện tử, có thể xác nhận đã thanh toán thành công hoặc hủy bỏ thanh toán

**Lang:** Ngôn ngữ của response trả về

**Description:** Mô tả chi tiết về giao dịch

**Api Key:** Key bí mật, nhằm xác thực danh tính của Merchant



Hình ‑. API hoàn tiền giao dịch

Request gửi lên hệ thống sẽ yêu cầu các thông tin:

**Partner Code:** Mã đối tác của ví điện tử

**Order Id:** Mã đơn hàng duy nhất của giao dịch

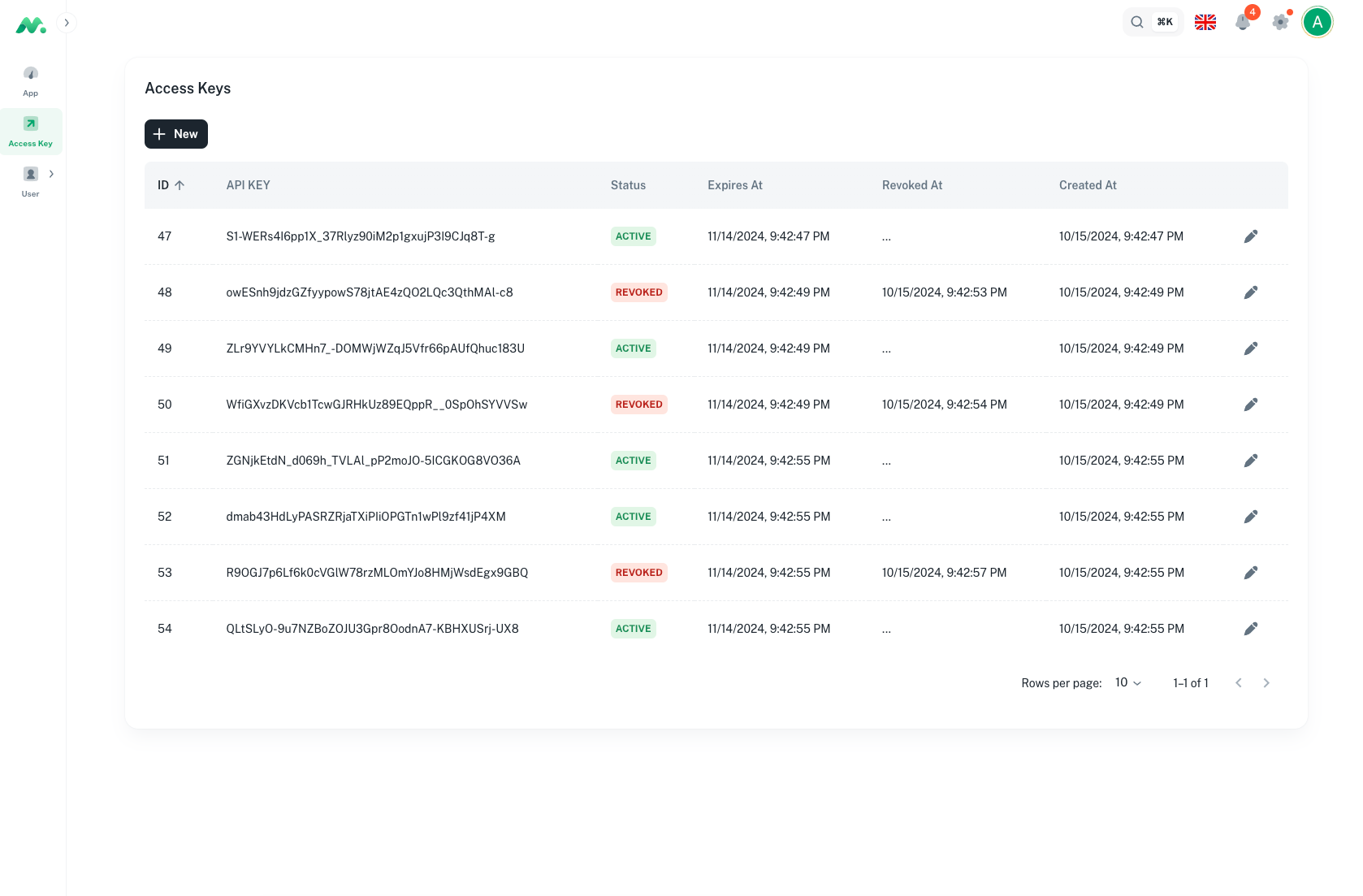
**Trans Id:** Mã giao dịch

**Amount:** Số tiền cần hoàn tiền, có thể hoàn tiền toàn bộ hoặc một phần

**Lang:** Ngôn ngữ của response trả về

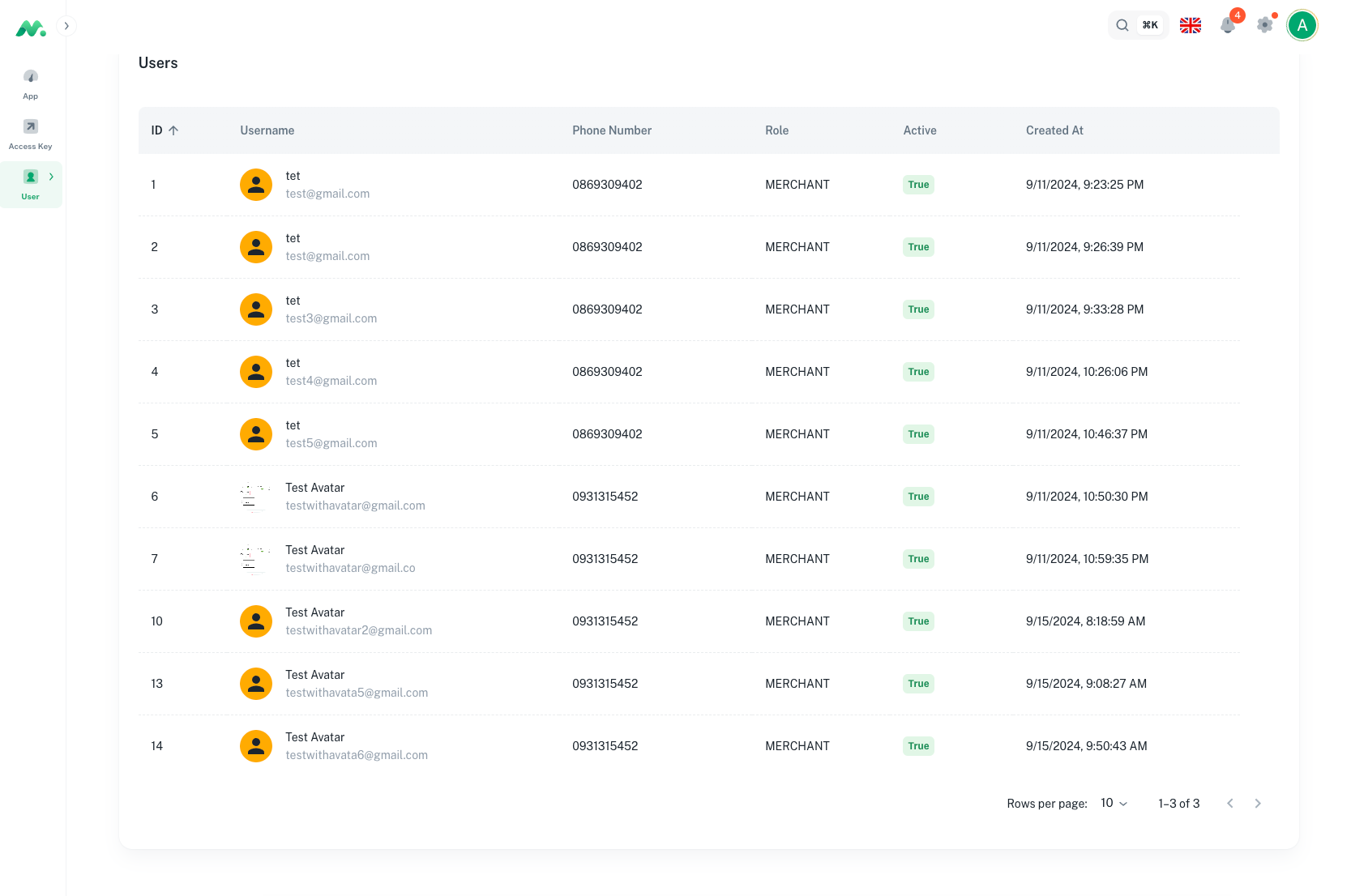
**Description:** Mô tả chi tiết về lí do hoàn tiền

**Api Key:** Key bí mật, nhằm xác thực danh tính của Merchant



Hình ‑. Giao diện quản lý Access Key

Cho phép Merchant quản lý các Access Key của mình, có thể tạo mới hoặc revoke trạng thái của các access key trước đó.



Hình ‑. Giao diện quản lý Merchant

Cho phép Admin quản lý các Merchant, xem trạng thái hoạt động của Merchant, đồng thời có thể thiết lập tài khoản thành không hoạt động.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

### Tổng quát

Dự án đã hoàn thiện gần như toàn bộ các mục tiêu đã đề ra.

Về mặt cấu trúc, đã đảm bảo được hệ thống phát triển theo kiến trúc microservices, các dịch vụ có thể tương tác qua lại với nhau.

Về mặt giao diện, người dùng chính trong hệ thống là Merchant và Admin, có thể tương tác và quản lý.

Các chức năng chính là Rest API cho các ví điện tử đã hoàn thành và có thể sử dụng.

### Hạn chế

Do thời gian có hạn, dự án sẽ gặp phải một số vấn đề liên quan về tối ưu hiệu năng, bugs về kết nối giữa các dịch vụ với nhau và bảo mật.

Toàn bộ dự án sẽ chạy trên môi trường developer, tức là cục bộ, không có lượng lớn người dùng thực tế để kiểm tra nếu như nhu cầu tăng cao thì hệ thống sẽ chịu tải như thế nào.

## Hướng phát triển

Hệ thống sẽ cần cải thiện thêm về mặt cấu trúc và hiệu năng. Đồng thời tích hợp thêm CI/CD để hệ thống có thể tự động cập nhật, xây dựng và triển khai, đảm bảo ứng dụng luôn hoạt động.

Trong tương lai, sẽ cần tích hợp và liên kết với các phương thức thanh toán ngân hàng nội địa và quốc tế**.**

Ngoài ra, các dịch vụ cloud như Amazon Web Services (AWS) là một lựa chọn tốt và tối ưu trong việc triển khai thực tế sản phẩm, nhằm đảm bảo mọi người đều có thể sử dụng.

TÀI LIỆU THAM KHẢO

[1] Pivotal Software, Inc., “Spring Boot Reference Documentation,” Spring.io. https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/ (truy cập: 10. 16, 2024).

[2] Vercel Inc., “Next.js Documentation,” Nextjs.org. https://nextjs.org/docs (truy cập: 10. 16, 2024).