**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

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

**LÂM PHÁT THỊNH**

**XÂY DỰNG HỆ THỐNG QUẢN LÝ NHÀ TRỌ VÀ SỐ HÓA CÁC TIỆN ÍCH**

**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**

**LÂM PHÁT THỊNH**

**XÂY DỰNG HỆ THỐNG QUẢN LÝ NHÀ TRỌ VÀ SỐ HÓA CÁC TIỆN ÍCH**

**Mã số sinh viên: 2151013094**

**ĐỒ ÁN NGÀNH**

**NGÀNH KHOA HỌC MÁY TÍNH**

**Giảng viên hướng dẫn: ThS. DƯƠNG HỮU THÀNH**

**TP. HỒ CHÍ MINH, 2024**

LỜI CẢM ƠN

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TÓM TẮT ĐỐ ÁN NGÀNH

ABSTRACT

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc176731193)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc176731194)

[TÓM TẮT ĐỐ ÁN NGÀNH 3](#_Toc176731195)

[ABSTRACT 4](#_Toc176731196)

[DANH MỤC TỪ VIẾT TẮT 7](#_Toc176731197)

[DANH MỤC HÌNH VẼ 8](#_Toc176731198)

[DANH MỤC BẢNG 9](#_Toc176731199)

[MỞ ĐẦU 10](#_Toc176731200)

[Chương 1. TỔNG QUAN ĐỀ TÀI 11](#_Toc176731201)

[1.1. Giới thiệu đề tài 11](#_Toc176731202)

[1.2. Lý do chọn đề tài 11](#_Toc176731203)

[1.3. Mục tiêu và phạm vi đề tài 11](#_Toc176731204)

[1.4. Phương pháp nghiên cứu 11](#_Toc176731205)

[1.5. Bố cục báo cáo 12](#_Toc176731206)

[Chương 2. CƠ SỞ LÝ THUYẾT 13](#_Toc176731207)

[2.1. Spring Boot 13](#_Toc176731208)

[2.1.1. Giới thiệu về Spring Boot 13](#_Toc176731209)

[2.1.2. Kiến trúc Spring Boot 13](#_Toc176731210)

[2.1.3. Cấu trúc dự án Spring Boot 14](#_Toc176731211)

[2.2. Reactjs 14](#_Toc176731212)

[2.2.1. Giới thiệu về ReactJS 14](#_Toc176731213)

[2.2.2. Kiến trúc ReactJS 15](#_Toc176731214)

[2.2.3. Cấu trúc dự án ReactJS 16](#_Toc176731215)

[2.3. Kiến trúc Microservice 16](#_Toc176731216)

[2.3.1. Giới thiệu Microservice 16](#_Toc176731217)

[2.3.2. Cấu trúc dự án microservice 17](#_Toc176731218)

[2.4. Các dịch vụ và công cụ hỗ trợ 18](#_Toc176731219)

[2.4.1. Docker 18](#_Toc176731220)

[Chương 3. HỆ THỐNG QUẢN LÝ NHÀ TRỌ VÀ SỐ HÓA CÁC TIỆN ÍCH 19](#_Toc176731221)

[3.1. Giới thiệu bài toán 19](#_Toc176731222)

[3.2. Phân tích hệ thống 19](#_Toc176731223)

[3.3. Thiết kế hệ thống 22](#_Toc176731224)

[3.3.1. Lược đồ tuần tự (Sequence diagram) 22](#_Toc176731225)

[3.3.2. Lược đồ hoạt động (Activity diagram [option]) 22](#_Toc176731226)

[3.3.3. Thiết kế cơ sở dữ liệu 22](#_Toc176731227)

[3.3.4. Thiết kế giao diện 22](#_Toc176731228)

[3.4. Kiến trúc hệ thống 22](#_Toc176731229)

[3.5. Kết quả thực hiện 23](#_Toc176731230)

[Chương 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 24](#_Toc176731231)

[4.1. Kết luận 24](#_Toc176731232)

[4.2. Hướng phát triển 24](#_Toc176731233)

[TÀI LIỆU THAM KHẢO 25](#_Toc176731234)

[PHỤ LỤC 26](#_Toc176731235)

DANH MỤC TỪ VIẾT TẮT

DANH MỤC HÌNH VẼ

[Hình 2.1 Kiến trúc Spring Boot 13](#_Toc176730653)

[Hình 2.2 Cấu trúc của một dự án Spring Boot 14](#_Toc176730654)

[Hình 2.3 Kiến trúc Reactjs 15](#_Toc176730655)

[Hình 2.4 Cấu trúc dự án ReactJS 16](#_Toc176730656)

[Hình 2.5 Kiến trúc microservice 17](#_Toc176730657)

[Hình 2.6 Cấu trúc một dự án microservice 17](#_Toc176730658)

[Hình 3.1 Use case hệ thống 20](#_Toc176730659)

DANH MỤC BẢNG

[Bảng 3.1 Đặc tả use case “thêm dữ liệu gương mặt” 21](#_Toc176730782)

[Bảng 3.2 Đặc tả use case “Xem danh sách cảnh báo” 21](#_Toc176730783)

MỞ ĐẦU

# TỔNG QUAN ĐỀ TÀI

## Giới thiệu đề tài

Trong thời đại số hóa hiện nay, việc quản lý nhà trọ không chỉ đơn thuần là ghi chép thông tin cư trú và thu tiền thuê, mà còn yêu cầu một hệ thống hiện đại để nâng cao hiệu quả và trải nghiệm cho cả chủ nhà lẫn người thuê. Đề tài "Xây dựng hệ thống quản lý nhà trọ và số hóa các tiện ích" nhằm phát triển một nền tảng trực tuyến hỗ trợ quản lý toàn diện các hoạt động liên quan đến kinh doanh nhà trọ.

## Lý do chọn đề tài

Trong bối cảnh công nghệ phát triển mạnh mẽ, số hóa và tự động hóa quy trình là xu hướng tất yếu. Việc quản lý nhà trọ bằng các phương pháp truyền thống như ghi chép tay, quản lý qua giấy tờ dần trở nên lỗi thời và kém hiệu quả. Hệ thống quản lý số hóa giúp tối ưu hóa quy trình, tăng tính chính xác và giảm thiểu sai sót. Người thuê cũng mong muốn một quy trình thuê nhà tiện lợi hơn, từ tìm kiếm, thanh toán đến quản lý các dịch vụ và tiện ích đi kèm. Các phương pháp quản lý thủ công thường gây ra nhiều bất tiện như khó khăn trong việc lưu trữ và tra cứu thông tin, không có cảnh báo kịp thời về thời hạn thanh toán, và việc tính toán sai lệch các chi phí tiện ích như điện, nước. Một hệ thống số hóa sẽ giúp khắc phục những nhược điểm này, giúp các bên liên quan làm việc hiệu quả hơn.

## Mục tiêu và phạm vi đề tài

Phát triển một hệ thống quản lý hiện đại giúp chủ nhà trọ theo dõi, quản lý tất cả các hoạt động liên quan đến nhà trọ một cách dễ dàng và hiệu quả. Hệ thống sẽ bao gồm quản lý thông tin người thuê, hợp đồng, thanh toán cũng như tăng cường an ninh bằng giải pháp nhận diện gương mặt phát hiện người lạ trong khu và đưa ra cảnh báo tức thời.

## Phương pháp nghiên cứu

Tiến hành khảo sát thực tế đối với chủ nhà trọ và người thuê nhà để thu thập thông tin về nhu cầu và khó khăn trong việc quản lý nhà trọ, quy trình thanh toán, và quản lý tiện ích. Khảo sát này giúp xác định rõ các vấn đề cần giải quyết và các tính năng cần có trong hệ thống. Nghiên cứu quy trình quản lý nhà trọ truyền thống để hiểu rõ các bước quản lý từ ghi nhận thông tin người thuê, quản lý hợp đồng, tính toán chi phí tiện ích, và thu tiền. Điều này giúp hệ thống được thiết kế hợp lý và có thể cải tiến các bước quản lý. Sử dụng công nghệ ReactJS để dễ dàng tạo ra giao diện thân thiện, dễ sử dụng cho cả chủ nhà và người thuê. Và Spring Boot để xây dựng server an toàn và linh hoạt.

## Bố cục báo cáo

Chương 1: Tổng quan đề tài, khái quát về đề tài và phương pháp nghiên cứu.

Chương 2: Cơ sở lý thuyết.

Chương 3: Hệ thống quản lý nhà trọ và số hóa các tiện ích.

Chương 4: Kết luận và hướng phát triển.

# CƠ SỞ LÝ THUYẾT

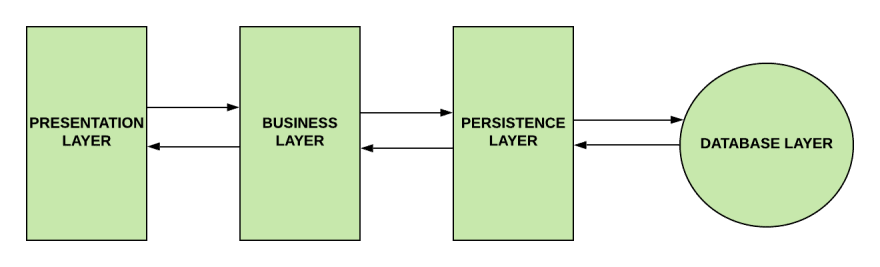
## Spring Boot

### Giới thiệu về Spring Boot

S**pring Boot** là một framework của Java được phát triển dựa trên Spring Framework. Mục tiêu của Spring Boot là giúp các nhà phát triển xây dựng và triển khai các ứng dụng web một cách nhanh chóng, dễ dàng mà không cần phải cấu hình phức tạp. Spring Boot tự động cấu hình các thành phần, giúp rút ngắn thời gian phát triển ứng dụng và cung cấp một cơ sở hạ tầng mạnh mẽ để xây dựng các ứng dụng microservices, API RESTful, hoặc ứng dụng doanh nghiệp phức tạp.

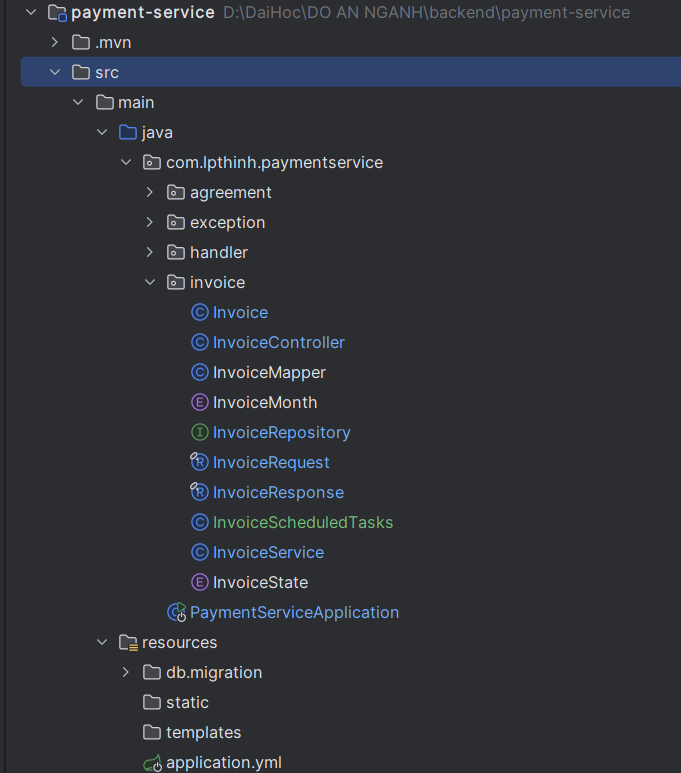
### Kiến trúc Spring Boot

Kiến trúc của Spring Boot bao gồm các lớp chính như **Presentation Layer** (Lớp trình bày) xử lý các yêu cầu HTTP thông qua các Controller, **Service Layer** (Lớp dịch vụ) chứa logic nghiệp vụ và quản lý tương tác giữa Controller và lớp dữ liệu, **Data Access Layer** (Lớp truy cập dữ liệu) xử lý việc giao tiếp với cơ sở dữ liệu hoặc các nguồn dữ liệu khác, và **Database Layer** (Lớp cơ sở dữ liệu) tương tác trực tiếp với cơ sở dữ liệu. Spring Boot hoạt động dựa trên nguyên tắc **Inversion of Control (IoC)** và **Dependency Injection (DI)**, giúp quản lý các thành phần của ứng dụng một cách hiệu quả và linh hoạt.



Hình 2.1 Kiến trúc Spring Boot

### Cấu trúc dự án Spring Boot



Hình 2.2 Cấu trúc của một dự án Spring Boot

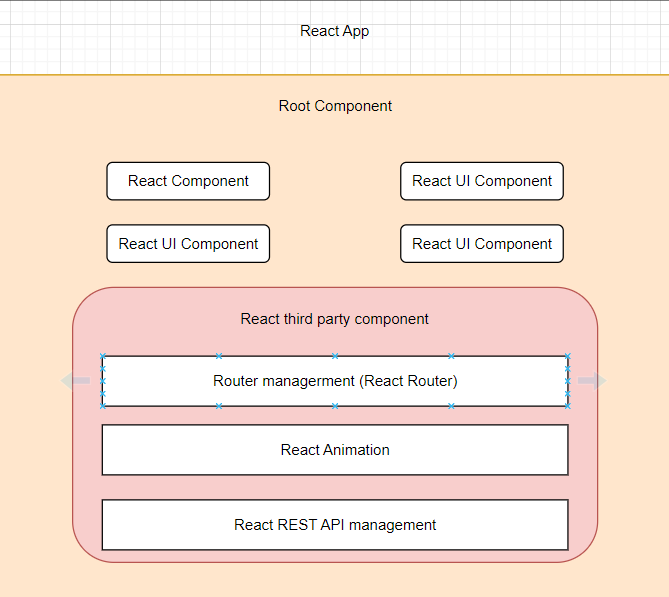
## Reactjs

### Giới thiệu về ReactJS

**ReactJS** là một thư viện JavaScript phát triển bởi Facebook, nổi bật với khả năng xây dựng giao diện người dùng (UI) một cách hiệu quả và linh hoạt. ReactJS sử dụng “**Component-Based Architecture**”, cho phép chia nhỏ giao diện thành các thành phần (components) tái sử dụng được, quản lý trạng thái và logic riêng biệt cho từng phần của ứng dụng. Thư viện này sử dụng “**Virtual DOM**” để cải thiện hiệu suất bằng cách chỉ cập nhật những phần giao diện có thay đổi, thay vì toàn bộ trang. ReactJS hỗ trợ “**JSX”**, một cú pháp kết hợp giữa HTML và JavaScript, giúp viết mã giao diện dễ hiểu và gần gũi với các nhà phát triển.

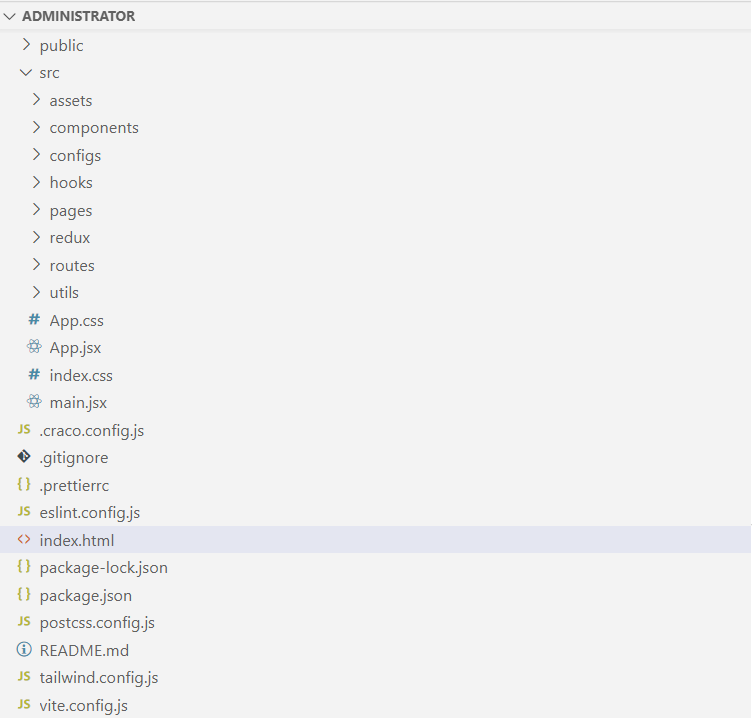
### Kiến trúc ReactJS

Ứng dụng React bắt đầu với một thành phần gốc duy nhất, từ đó xây dựng các thành phần khác để tạo ra giao diện người dùng. Thành phần gốc được tạo ra bằng cách sử dụng một hoặc nhiều thành phần nhỏ hơn, và các thành phần này có thể được lồng vào nhau ở bất kỳ mức độ nào. Thành phần là một trong những khái niệm cốt lõi của React, và thay vì kế thừa, mỗi thành phần thường được xây dựng bằng cách kết hợp các thành phần nhỏ hơn. Hầu hết các thành phần trong React đóng vai trò là các thành phần giao diện người dùng. Để mở rộng chức năng của ứng dụng, React cũng cho phép tích hợp các thành phần của bên thứ ba, như các công cụ định tuyến, hoạt ảnh, và quản lý trạng thái.



Hình 2.3 Kiến trúc Reactjs

### Cấu trúc dự án ReactJS



Hình 2.4 Cấu trúc dự án ReactJS

## Kiến trúc Microservice

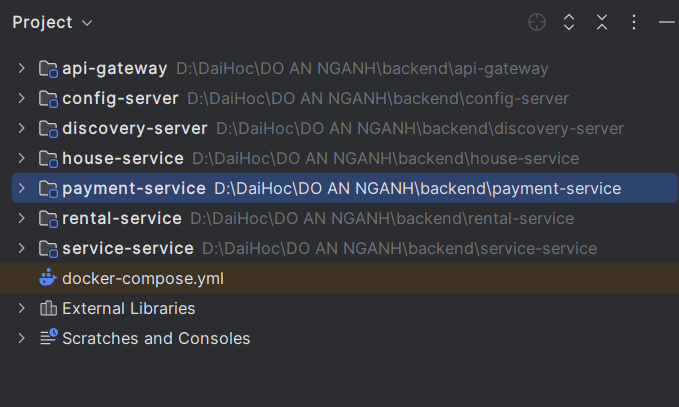
### Giới thiệu Microservice

**Microservice** là một kiến trúc phần mềm trong đó ứng dụng được chia thành các dịch vụ nhỏ, độc lập, mỗi dịch vụ đảm nhiệm một chức năng cụ thể. Các microservice có thể được phát triển, triển khai, và mở rộng độc lập, giúp cải thiện tính linh hoạt và khả năng bảo trì của hệ thống. Chúng giao tiếp với nhau qua các giao thức nhẹ như HTTP/REST hoặc thông qua các message broker như Kafka. Mô hình này mang lại nhiều lợi ích như dễ mở rộng, tăng tốc độ phát triển và bảo trì dễ dàng hơn, nhưng cũng đi kèm thách thức về quản lý, theo dõi lỗi và hiệu suất nếu không được tối ưu hóa.



Hình 2.5 Kiến trúc microservice

### Cấu trúc dự án microservice



Hình 2.6 Cấu trúc một dự án microservice

## Các dịch vụ và công cụ hỗ trợ

### Docker

**Docker** là nền tảng mã nguồn mở cho phép tạo, triển khai và chạy các ứng dụng trong các **container**, cung cấp môi trường ảo hóa nhẹ chứa tất cả các thành phần cần thiết cho ứng dụng, bao gồm mã nguồn, thư viện, và phụ thuộc. Docker đảm bảo tính nhất quán giữa các môi trường phát triển, kiểm thử và triển khai bằng cách đóng gói ứng dụng vào một container duy nhất, giúp triển khai dễ dàng trên nhiều nền tảng như máy chủ vật lý, máy ảo, và dịch vụ đám mây. Với **Docker Compose**, quản lý nhiều container cùng lúc trở nên đơn giản, hỗ trợ các ứng dụng phức tạp bao gồm frontend, backend, và các dịch vụ phụ trợ. Docker giúp quy trình phát triển, kiểm thử và triển khai trở nên hiệu quả hơn bằng cách cung cấp tính nhất quán, khả năng mở rộng linh hoạt và triển khai dễ dàng.

# HỆ THỐNG QUẢN LÝ NHÀ TRỌ VÀ SỐ HÓA CÁC TIỆN ÍCH

## Giới thiệu bài toán

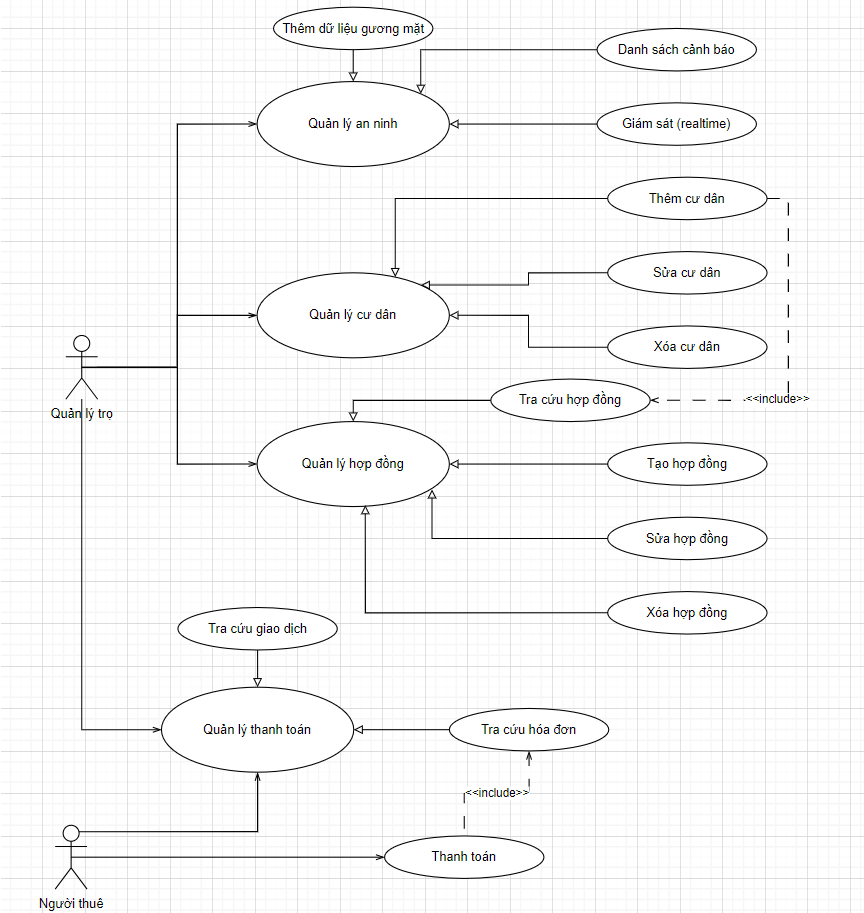
**Hệ thống Quản lý Nhà trọ và Số hóa các Tiện ích** là một giải pháp nhằm hiện đại hóa việc quản lý các cơ sở lưu trú như nhà trọ hoặc căn hộ cho thuê. Hệ thống này sử dụng kiến trúc **microservice** với **Spring Boot** cho backend và **ReactJS** cho frontend, giúp quản lý các chức năng như quản lý người thuê, hợp đồng, thanh toán và các tiện ích liên quan.

Với **Spring Boot** microservice, hệ thống được phân chia thành nhiều dịch vụ nhỏ độc lập như dịch vụ quản lý người thuê, dịch vụ quản lý hợp đồng, và dịch vụ thanh toán, mỗi dịch vụ có thể phát triển và triển khai một cách riêng biệt. Điều này mang lại tính linh hoạt, khả năng mở rộng, và dễ bảo trì cho toàn bộ hệ thống. Các dịch vụ này giao tiếp qua giao thức HTTP/REST hoặc sử dụng message broker như Kafka để đảm bảo trao đổi dữ liệu mượt mà giữa các thành phần.

Ở phía frontend, **ReactJS** cung cấp giao diện người dùng hiện đại và thân thiện, cho phép người dùng dễ dàng tương tác với hệ thống. Chủ nhà và người thuê có thể quản lý hợp đồng, theo dõi trạng thái thanh toán, và sử dụng các tiện ích trực tuyến như yêu cầu bảo trì, xem hóa đơn điện nước, và nhận thông báo.

## Phân tích hệ thống

1. Lược đồ Use case



Hình 3.1 Use case hệ thống

1. Đặc tả use case

|  |  |
| --- | --- |
| **Use case Id** | UC-01 |
| **Tên use Case** | Thêm dữ liệu gương mặt |
| **Mô tả** | Use case này cho phép quản lý trọ lấy hình ảnh gương mặt của người thuê để đưa vào dữ liệu hệ thống. |
| **Actor chính** | Quản lý trọ |
| **Actor phụ** | Không có |
| **Tiền điều kiện** | Thông tin người thuê đã được đưa vào dữ liệu hệ thống |
| **Hậu điều kiện** | Không có |
| **Luồng hoạt động** | 1. Tra cứu cư dân  2. Lấy hình ảnh của cư dân thông qua camera  3. Hoàn tất thêm dữ liệu |
| **Luồng thay thế** | Không có |
| Luồng ngoại lệ | Không có |

Bảng 3.1 Đặc tả use case “Thêm dữ liệu gương mặt”

|  |  |
| --- | --- |
| **Use case Id** | UC-02 |
| **Tên use Case** | Xem danh sách cảnh báo |
| **Mô tả** | Use case này cho phép quản lý trọ xem chi tiết các cảnh báo phát hiện người lạ trong khu vực |
| **Actor chính** | Quản lý trọ |
| **Actor phụ** | Không có |
| **Tiền điều kiện** | Không có |
| **Hậu điều kiện** | Không có |
| **Luồng hoạt động** | 1. Hiển thị danh sách các cảnh báo |
| **Luồng thay thế** | Không có |
| Luồng ngoại lệ | Không có |

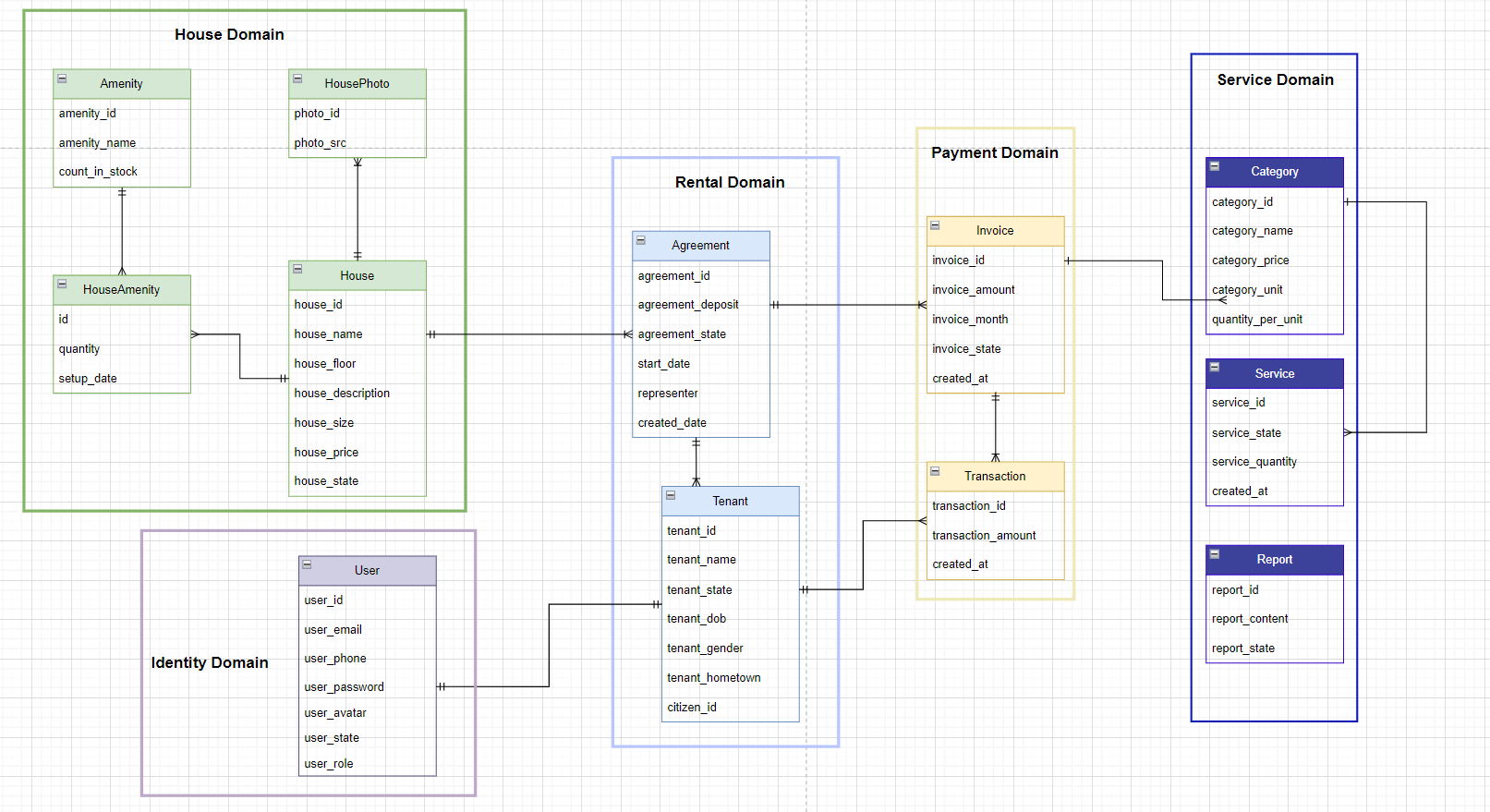
Bảng 3.2 Đặc tả use case “Xem danh sách cảnh báo”

## Thiết kế hệ thống

### Lược đồ tuần tự (Sequence diagram)

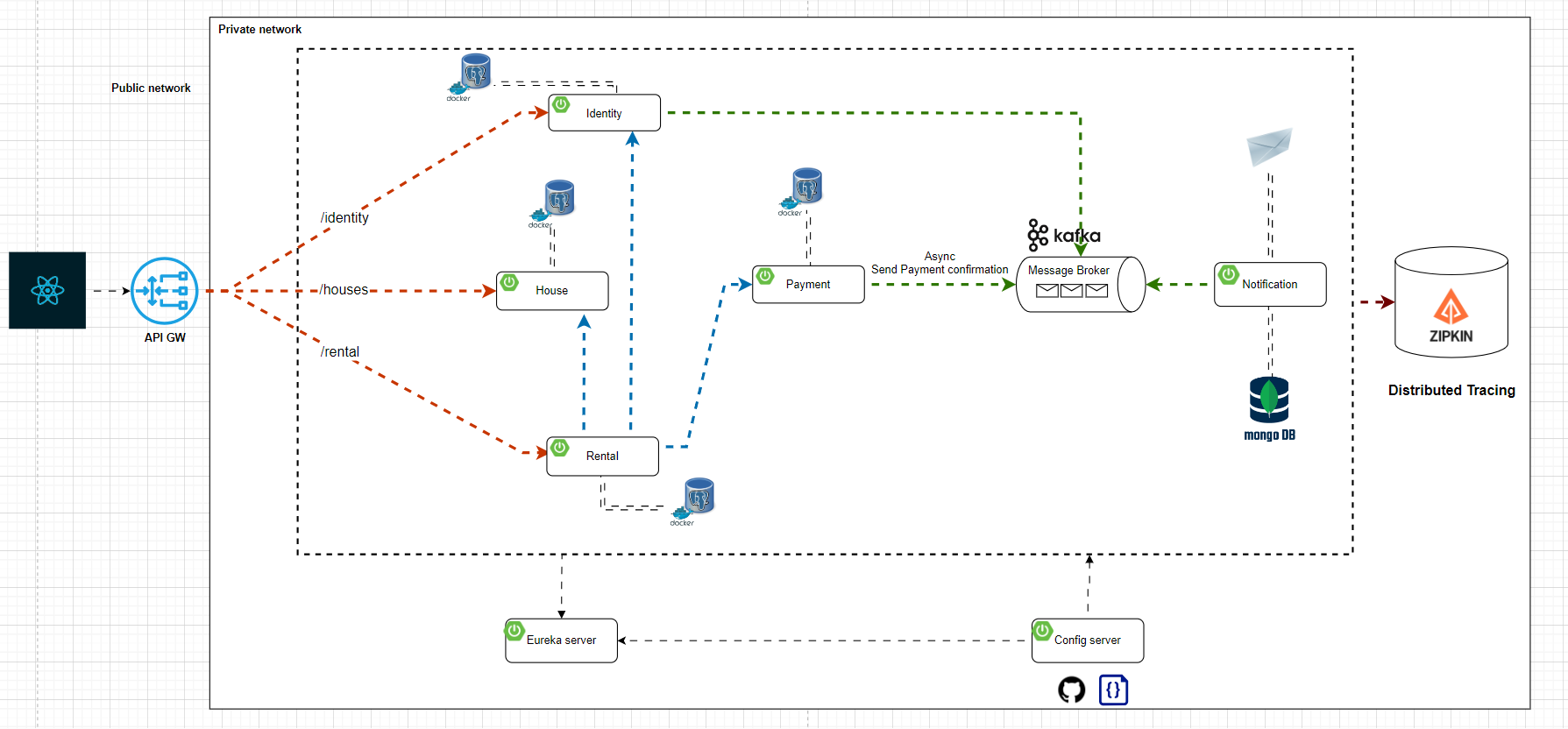
### Lược đồ hoạt động (Activity diagram [option])

### Thiết kế cơ sở dữ liệu



### Thiết kế giao diện

## Kiến trúc hệ thống



Hình 3.2 Kiến trúc hệ thống

## Kết quả thực hiện

# 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