**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

Dự án "Xây dựng hệ thống quản lý nhà trọ và số hóa các tiện ích" có mục tiêu chính là phát triển một nền tảng quản lý toàn diện cho các hoạt động liên quan đến việc cho thuê nhà trọ, nhằm mang lại sự tiện lợi cho cả chủ trọ và người thuê. Hệ thống được xây dựng dựa trên kiến trúc microservices hiện đại, sử dụng Spring Boot làm nền tảng phát triển backend và kết hợp giữa hai cơ sở dữ liệu PostgreSQL và MongoDB để lưu trữ dữ liệu, giúp tối ưu hóa quá trình quản lý và truy xuất thông tin. Các chức năng chính của hệ thống bao gồm quản lý hóa đơn hàng tháng, phòng trọ, thông tin người thuê, hợp đồng thuê, và đặc biệt là tích hợp công nghệ nhận diện gương mặt bằng OpenCV để tự động phát hiện và cảnh báo khi có sự xuất hiện của người lạ trong khu vực nhà trọ.

Phía giao diện người dùng, ứng dụng dành cho admin được phát triển bằng ReactJS, giúp quản lý các tác vụ CRUD (tạo, đọc, cập nhật, xóa) liên quan đến các đối tượng như phòng, hợp đồng, người thuê, và hóa đơn một cách dễ dàng và trực quan. Bên cạnh đó, hệ thống cũng cung cấp một ứng dụng di động phát triển bằng React Native dành cho người thuê nhà, giúp họ truy cập vào các tiện ích trực tuyến như thanh toán hóa đơn, theo dõi thông tin hợp đồng, và nhận các thông báo quan trọng từ chủ trọ về tình trạng phòng và hóa đơn ngay trên điện thoại di động của mình.

Kết quả của dự án là hoàn thành một hệ thống quản lý nhà trọ thông minh với đầy đủ các chức năng cần thiết, đáp ứng nhu cầu quản lý của chủ trọ và mang lại sự tiện lợi cho người thuê thông qua các tính năng trực tuyến. Hệ thống không chỉ tối ưu hóa quy trình quản lý mà còn giúp nâng cao tính an ninh cho khu vực nhà trọ thông qua công nghệ cảnh báo người lạ, tạo nên một giải pháp toàn diện và hiện đại cho việc quản lý nhà trọ.

ABSTRACT

**MỤC LỤC**

[DANH MỤC TỪ VIẾT TẮT 6](#_Toc51061474)

[DANH MỤC HÌNH VẼ 7](#_Toc51061475)

[DANH MỤC BẢNG 8](#_Toc51061476)

[MỞ ĐẦU 9](#_Toc51061477)

[Chương 1. TÊN CHƯƠNG 1 10](#_Toc51061478)

[1.1. Chủ đề cấp độ 2 10](#_Toc51061479)

[1.1.1. Chủ đề cấp độ 3 10](#_Toc51061480)

[1.1.2. Chủ đề cấp độ 3 10](#_Toc51061481)

[Chương 2. TÊN CHƯƠNG 2 11](#_Toc51061482)

[2.1. Chủ đề cấp độ 2 11](#_Toc51061483)

[2.1.1. Chủ đề cấp độ 3 11](#_Toc51061484)

[2.2. Chủ đề cấp độ 2 11](#_Toc51061485)

[2.2.1. Chủ đề cấp độ 3 11](#_Toc51061486)

[Chương 3. TÊN CHƯƠNG 3 12](#_Toc51061487)

[3.1. Chủ đề cấp độ 2 12](#_Toc51061488)

[3.1.1. Chủ đề cấp độ 3 12](#_Toc51061489)

[3.2. Chủ đề cấp độ 2 12](#_Toc51061490)

[TÀI LIỆU THAM KHẢO 13](#_Toc51061491)

[PHỤ LỤC 14](#_Toc51061492)

DANH MỤC TỪ VIẾT TẮT

DANH MỤC HÌNH VẼ

[Hình 2.1 Kiến trúc Spring Boot 12](#_Toc6979)

[Hình 2.2 Cấu trúc của một dự án Spring Boot 16](#_Toc17016)

DANH MỤC BẢNG

[Bảng 1.1: Tên bảng 1 10](#_Toc367742567)

[Bảng 2.1: Tên bảng 1 11](#_Toc367742568)

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

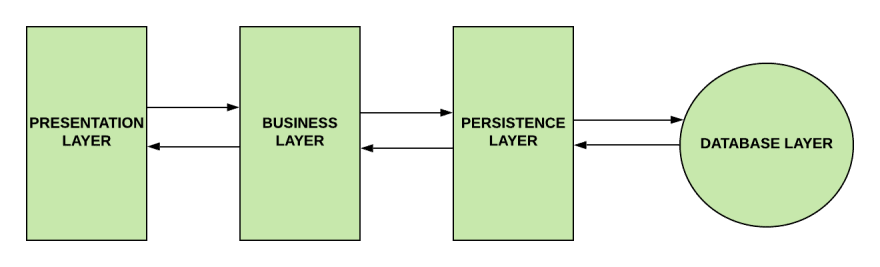
## Spring Boot

### 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

#### Presentation Layer

1. Mục đích: Lớp này chịu trách nhiệm xử lý các yêu cầu từ người dùng, thường thông qua giao thức HTTP. Đây là nơi mà các API RESTful hoặc giao diện người dùng tương tác với ứng dụng.
2. Các thành phần chính: Controller, Views (nếu sử dụng MVC với các công cụ như Thymeleaf hoặc JSP).
3. Chi tiết:

Controller: Là thành phần nhận các yêu cầu HTTP từ phía người dùng hoặc client. Controller chuyển tiếp yêu cầu này tới Service Layer để xử lý nghiệp vụ và trả về kết quả cho người dùng thông qua response. Trong Spring Boot, các Controller được định nghĩa bằng annotation @RestController hoặc @Controller.

Views (optional): Nếu sử dụng mô hình MVC, lớp này sẽ chứa các view template để hiển thị dữ liệu tới người dùng (ví dụ: các tệp HTML với Thymeleaf).

1. Vai trò chính: Điều phối và trả về dữ liệu đã xử lý dưới dạng JSON, HTML, hoặc XML (tùy thuộc vào loại ứng dụng).

#### Service Layer (Lớp dịch vụ)

1. Mục đích: Chứa các logic nghiệp vụ chính của ứng dụng. Lớp này là nơi tất cả các quy tắc và điều kiện nghiệp vụ được thực hiện.
2. Các thành phần chính: Service classes, thường được đánh dấu bằng annotation @Service.
3. Chi tiết:

Business Logic: Service Layer là nơi toàn bộ logic nghiệp vụ của ứng dụng diễn ra. Nó lấy dữ liệu từ Presentation Layer (thông qua các request từ Controller), xử lý, và giao tiếp với Data Access Layer để lưu hoặc lấy dữ liệu.

Transaction Management: Đây cũng là nơi mà các giao dịch dữ liệu được quản lý. Spring hỗ trợ quản lý giao dịch thông qua các annotation như @Transactional.

1. Vai trò chính: Xử lý toàn bộ logic của ứng dụng, giữ cho Controller gọn gàng và tập trung vào việc xử lý yêu cầu từ client.

#### Data Access Layer (Lớp truy cập dữ liệu)

1. Mục đích: Lớp này chịu trách nhiệm tương tác với cơ sở dữ liệu hoặc các nguồn dữ liệu khác (API, file, v.v.). Nó cung cấp một lớp abstraction giữa Service Layer và Database Layer, đảm bảo việc truy xuất dữ liệu được tách biệt và dễ quản lý.
2. Các thành phần chính: Repository classes, đánh dấu bằng annotation @Repository.
3. Chi tiết:

Data Access Object (DAO): Trong Spring Boot, các DAO hoặc Repository đảm nhận việc thực hiện các hoạt động CRUD (Create, Read, Update, Delete) trên cơ sở dữ liệu. Spring Boot cung cấp Spring Data JPA giúp giảm bớt việc viết mã bằng cách cung cấp các interface như CrudRepository hoặc JpaRepository.

ORM (Object Relational Mapping): Spring Boot thường sử dụng Hibernate, một framework ORM, để ánh xạ các đối tượng Java thành các bản ghi trong cơ sở dữ liệu.

1. Vai trò chính: Cung cấp phương tiện để Service Layer tương tác với cơ sở dữ liệu mà không phải lo lắng về chi tiết triển khai.

#### Database Layer (Lớp cơ sở dữ liệu)

1. Mục đích: Tương tác trực tiếp với cơ sở dữ liệu thực tế, nơi lưu trữ dữ liệu ứng dụng.
2. Các thành phần chính: Hệ quản trị cơ sở dữ liệu (MySQL, PostgreSQL, Oracle, v.v.).
3. Chi tiết:

Schema: Định nghĩa cấu trúc của cơ sở dữ liệu như bảng, cột, khóa chính, khóa ngoại, v.v.

SQL: Là các câu truy vấn SQL được tự động tạo bởi ORM framework (Hibernate) hoặc có thể do người lập trình viết tay cho các truy vấn phức tạp.

Connection Pooling: Spring Boot thường sử dụng các kết nối đã được tối ưu hóa bằng cách quản lý pool các kết nối tới cơ sở dữ liệu (thông qua các thư viện như HikariCP).

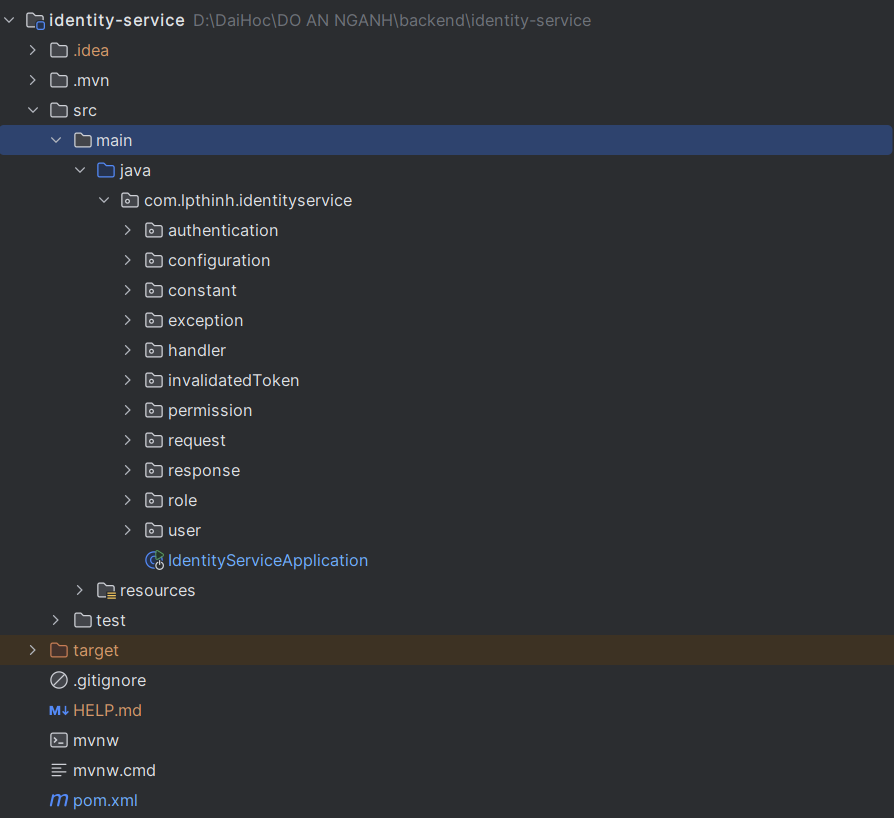
1. Vai trò chính: Lưu trữ và quản lý dữ liệu một cách nhất quán và hiệu quả, đảm bảo tính toàn vẹn của dữ liệu.

#### Nguyên tắc kiến trúc chính

Inversion of Control (IoC): Trong Spring Boot, nguyên tắc IoC cho phép framework kiểm soát luồng ứng dụng thay vì để các lớp tự kiểm soát. Thông qua IoC Container, Spring tự động khởi tạo các đối tượng và quản lý các phụ thuộc của chúng.

Dependency Injection (DI): DI là một trong những cơ chế của IoC, cho phép Spring Boot "inject" các đối tượng vào trong các lớp phụ thuộc vào chúng, thay vì để các lớp tự khởi tạo các đối tượng cần thiết. Điều này giúp mã nguồn dễ bảo trì và kiểm thử hơn.

### Cấu trúc dự án Spring Boot



Hình 2.2 Cấu trúc của một dự án Spring Boot

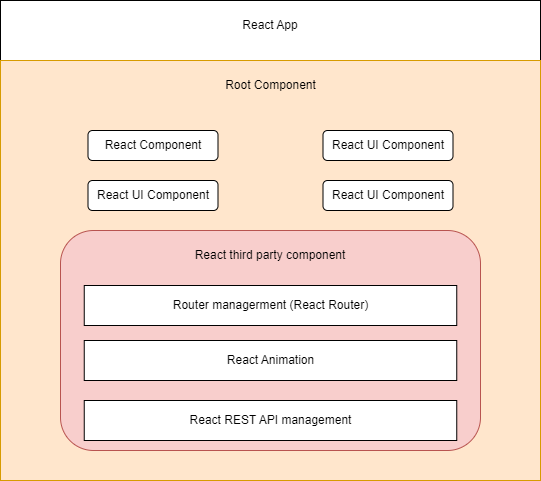
## React

### Giới thiệu về React

**React** (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. React 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. React 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.

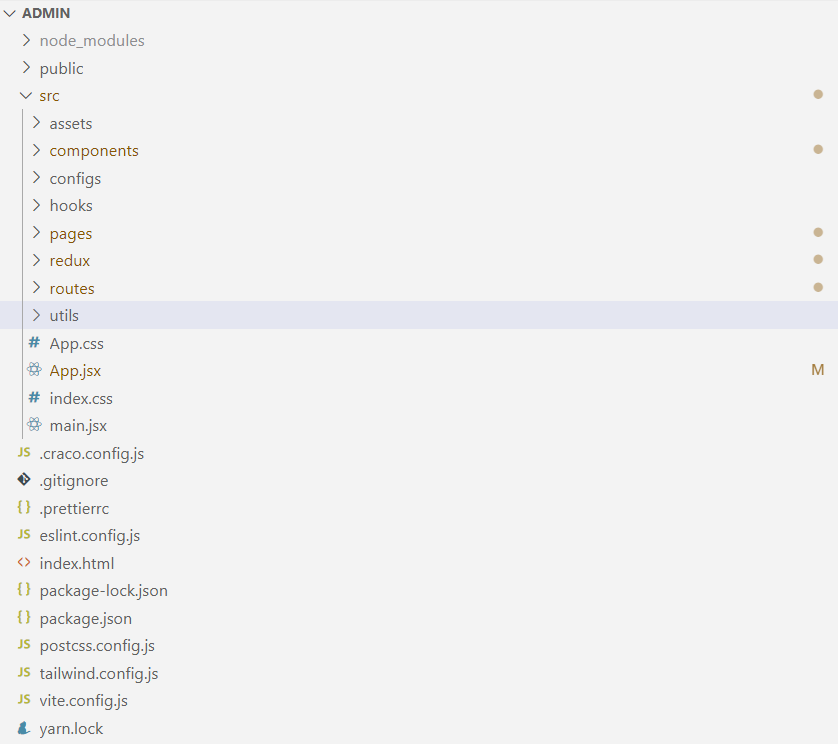
### Kiến trúc React

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 React

### Cấu trúc dự án React

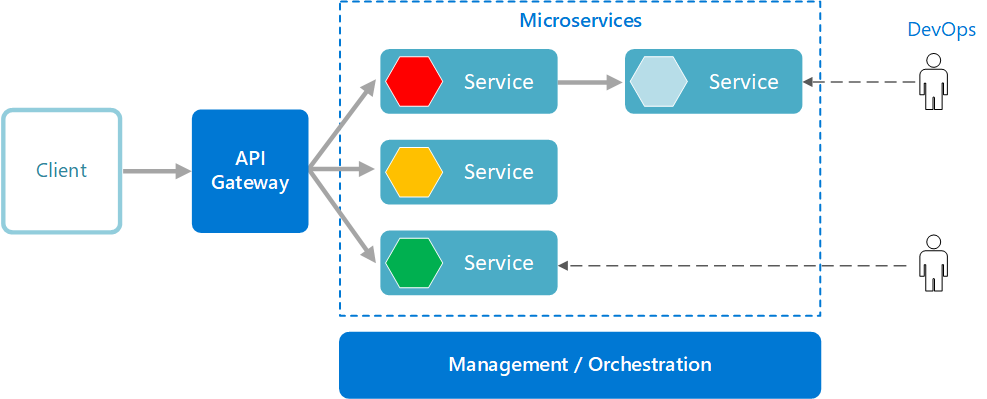


Hình 2.4 Cấu trúc dự án React

## Kiến trúc Microservice

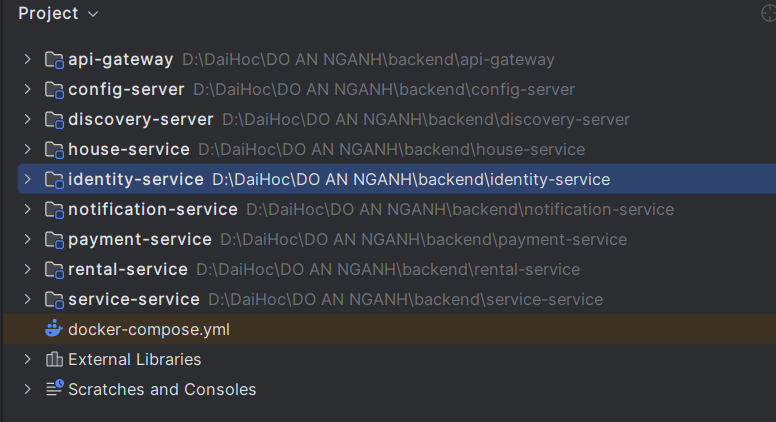
### Giới thiệu Microservice

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
2. Đặc tả Use case

## Thiết kế hệ thống

### Lược đồ tuần tự (Sequence diagram)

### Thiết kế cơ sở dữ liệu

### Thiết kế giao diện

## Kiến trúc hệ thống

## Kết quả thực hiện

# KẾ 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