

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM



ĐỒ ÁN 1
XÂY DỰNG WEBSITE ĐẶT PHÒNG KHÁCH SẠN
BẰNG KIẾN TRÚC MICROSERVICE
KẾT HỢP SỬ DỤNG GRPC

GV HƯỚNG DẪN: ThS. Trần Thị Hồng Yến
SV THỰC HIỆN: Lê Đoàn Tấn Trí – 21522702

TP. HỒ CHÍ MINH, 2024

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành đến ThS. Trần Thị Hồng Yến, một giảng viên tâm huyết và chu toàn, đã luôn giúp đỡ và động viên em trong suốt quá trình thực hiện đồ án. Cô cũng là người tận tâm, tận tình, tận tụy chăm lo và nắn nót cho từng sản phẩm của sinh viên, tỉ mỉ chỉ dẫn cho dù là những tiểu tiết.

Em cũng xin cảm ơn Trường Đại học Công nghệ Thông tin và Khoa Công nghệ Phần mềm đã tạo cơ hội cho sinh viên chúng em được thực hành môn *Đồ Án 1*, một môn học mang tính ứng dụng và chuyên môn cao để phần nào giúp chúng em trang bị những kiến thức hữu dụng cho công việc tương lai sau này.

Xin cảm ơn những người bạn tốt, những người cộng sự luôn kề vai sát cánh đưa ra những lời khuyên và giải pháp kịp thời để đồ án của tôi được hoàn thành một cách trọn vẹn.

Xin cảm ơn ba mẹ vì đã luôn ủng hộ cả tinh thần và vật chất để con có thể hoàn thiện đồ án môn học của mình.

Hơn thế nữa, xin cảm ơn những nhân sự có liên quan mà có thể tôi chưa đề cập đến.

Một lần nữa, xin cảm ơn rất nhiều! Mong chúc những điều tốt đẹp sẽ luôn đồng hành với mọi người.

TP. Hồ Chí Minh, tháng 3 năm 2024

Lê Đoàn Tấn Trí

MỤC LỤC

Chương 1. GIỚI THIỆU ĐỀ TÀI	3
1.1. Lý do chọn đề tài	3
1.2. Mục tiêu đề tài	4
1.3. Phạm vi	4
1.3.1. Phạm vi môi trường	4
1.3.2. Phạm vi chức năng	4
1.4. Đối tượng sử dụng	5
1.5. Phương pháp thực hiện	5
1.6. Công nghệ sử dụng	6
1.7. Kết quả mong đợi	6
Chương 2. CƠ SỞ LÝ THUYẾT	7
2.1. Những hướng nghiên cứu đã có	7
2.1.1. Các nghiên cứu ngoài nước	7
2.1.2. Các nghiên cứu trong nước	7
2.1.3. Đánh giá	8
2.2. Những vấn đề còn tồn tại	8
2.3. Những vấn đề cần tập trung và giải quyết	8
2.4. Kiến trúc hệ thống	9
2.4.1. Các kiểu kiến trúc	9
2.4.1.1. Khái niệm	9
2.4.1.2. Phân loại	9
2.4.2. Mô hình Khách-Chủ (Client-Server)	9
2.4.2.1. Khái niệm	9

2.4.2.2.	Ưu điểm và nhược điểm	10
2.4.3.	Kiến trúc hướng dịch vụ (Service-Oriented Architecture)	11
2.4.3.1.	Khái niệm	11
2.4.3.2.	Ưu điểm và nhược điểm	11
2.4.4.	Kiến trúc vi dịch vụ (Microservice Architecture).....	12
2.4.4.1.	Khái niệm	12
2.4.4.2.	Ưu điểm và nhược điểm	12
2.5.	Công nghệ áp dụng.....	13
2.5.1.	Next.js	13
2.5.2.	Java Spring Boot	14
2.5.3.	MongoDB	15
2.5.4.	RESTful API.....	16
2.5.5.	gRPC	17
2.5.6.	Clerk.....	18
2.5.7.	Docker.....	18
2.5.8.	Zipkin.....	19
2.5.9.	JUnit5	20
Chương 3.	PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG	21
3.1.	Khảo sát hiện trạng.....	21
3.2.	Xác định yêu cầu	21
3.2.1.	Yêu cầu chức năng.....	21
3.2.2.	Yêu cầu phi chức năng.....	21
3.2.3.	Yêu cầu nghiệp vụ	24
3.3.	Mô hình hóa yêu cầu	27

3.3.1.	Lược đồ Use Case	27
3.3.2.	Danh sách chức năng	28
3.3.3.	Danh sách tác nhân (Actor).....	30
3.3.4.	Danh sách trường hợp sử dụng (Use Case)	30
3.3.5.	Đặc tả Use Case	34
3.4.	Thiết kế hệ thống.....	53
3.4.1.	Kiến trúc hệ thống.....	53
3.4.2.	Mô tả các thành phần trong hệ thống.....	54
3.5.	Thiết kế hướng đối tượng.....	55
3.5.1.	Sơ đồ lớp (Class Diagram).....	55
3.5.2.	Sơ đồ tuần tự (Sequence Diagram)	57
3.6.	Thiết kế dữ liệu.....	58
3.6.1.	Sơ đồ cơ sở dữ liệu (Database Diagram)	58
3.6.2.	Mô tả các Collection trong cơ sở dữ liệu	58
3.7.	Thiết kế giao diện	63
3.7.1.	Danh sách các màn hình	63
3.7.2.	Sơ đồ luồng màn hình (Screen Flow Diagram)	68
Chương 4.	XÂY DỰNG WEBSITE.....	69
4.1.	Cài đặt.....	69
4.1.1.	Môi trường và công cụ.....	69
4.1.2.	Hướng dẫn cài đặt	69
4.2.	Thiết lập dự án.....	69
4.2.1.	Thiết lập Discovery Server và API Gateway	69
4.2.2.	Thiết lập gRPC.....	73

4.2.3.	Thiết lập Zipkin.....	76
4.2.4.	Kết quả thực thi.....	77
4.3.	Triển khai	80
4.3.1.	Back-end (Server)	80
4.3.2.	Front-end (Client)	81
4.4.	Kiểm thử.....	83
4.4.1.	Kiểm thử thủ công.....	83
4.4.2.	Kiểm thử tự động	85
4.4.3.	Kiểm thử API.....	87
Chương 5.	KẾT LUẬN.....	89
5.1.	Công việc đã hoàn thành	89
5.2.	Kết quả đồ án.....	89
5.3.	Ưu điểm và hạn chế.....	90
5.4.	Hướng phát triển.....	91
TÀI LIỆU THAM KHẢO.....		93

DANH SÁCH HÌNH ẢNH

<i>Hình 2.1 Mô hình Client-Server</i>	10
<i>Hình 2.2 Điểm lỗi duy nhất</i>	11
<i>Hình 2.3 Logo Next.js</i>	13
<i>Hình 2.4 Logo Spring Framework</i>	14
<i>Hình 2.5 Logo MonogoDB</i>	15
<i>Hình 2.6 Cách MongoDB hoạt động</i>	15
<i>Hình 2.7 Logo gRPC</i>	17
<i>Hình 2.8 Logo Clerk</i>	18
<i>Hình 2.9 Logo Docker</i>	18
<i>Hình 2.10 Logo Zipkin</i>	19
<i>Hình 2.11 Logo JUnit5</i>	20
<i>Hình 3.1 Lược đồ Use Case</i>	27
<i>Hình 3.2 Sơ đồ kiến trúc hệ thống</i>	53
<i>Hình 3.3 Sơ đồ các lớp ở mức khái niệm</i>	55
<i>Hình 3.4 Sơ đồ cơ sở dữ liệu</i>	58
<i>Hình 3.5 Sơ đồ luồng màn hình</i>	68
<i>Hình 4.1 Tạo module discovery-server</i>	70
<i>Hình 4.2 Cấu hình file pom.xml của discovery-server</i>	70
<i>Hình 4.3 Cấu hình file application.properties của discovery-server</i>	70
<i>Hình 4.4 Cấu hình file pom.xml của api-gateway</i>	71
<i>Hình 4.5 Cấu hình file application.properties của api-gateway</i>	71
<i>Hình 4.6 Cấu hình file pom.xml cho các service là eureka-client</i>	72
<i>Hình 4.7 Cấu hình file application.properties cho các service</i>	72
<i>Hình 4.8 Giao diện trang Spring Eureka</i>	72
<i>Hình 4.9 Cấu hình các dependencies của gRPC cho hotel-service</i>	73
<i>Hình 4.10 Thư mục (package) proto</i>	73
<i>Hình 4.11 File hotel.proto</i>	74
<i>Hình 4.12 Các files do protobuf tạo tự động khi build chương trình</i>	75

Hình 4.13 Cấu hình file pom.xml cho các service	76
Hình 4.14 Cấu hình file application.properties cho các services	76
Hình 4.15 Zipkin đang chạy trong Docker	77
Hình 4.16 Giao diện Zipkin.....	77
Hình 4.17 getHotelById bằng HTTP GET	78
Hình 4.18 getHotelById bằng gRPC.....	78
Hình 4.19 Tổng thời gian thực thi bởi các service.....	79
Hình 4.20 Truy vết các dịch vụ	79
Hình 4.21 Cây thư mục của code back-end	80
Hình 4.22 Biểu tượng Services.....	80
Hình 4.23 Khởi chạy chương trình Spring Boot	81
Hình 4.24 Cây thư mục của code front-end	81
Hình 4.25 Mở New Terminal	82
Hình 4.26 Client đang chạy	82
Hình 4.27 Trạng thái của các trường và nút khi chưa nhập gì	83
Hình 4.28 Trạng thái của các trường và nút khi đã nhập dữ liệu đầy đủ.....	83
Hình 4.29 Các trường và nút trên trang đặt phòng khi chưa nhập gì	84
Hình 4.30 Các trường báo lỗi khi thông tin chưa được nhập.....	85
Hình 4.31 Thiết lập module kiểm thử tích hợp.....	85
Hình 4.32 Viết test cho hàm shouldCreateHotel().....	86
Hình 4.33 Phần test đã pass.....	86
Hình 4.34 Test container được tạo ra trong Docker	87
Hình 4.35 Giao diện Postman khi chưa gửi yêu cầu	87
Hình 4.36 Kết quả trả về danh sách các khách sạn.....	88
Hình 5.1 Các công việc đã hoàn thành	89

DANH SÁCH BẢNG

<i>Bảng 2.1 Bảng phân loại các kiểu kiến trúc</i>	<i>9</i>
<i>Bảng 3.1 Danh sách các yêu cầu phi chức năng</i>	<i>21</i>
<i>Bảng 3.2 Bảng phân rã chi tiết các yêu cầu phi chức năng</i>	<i>22</i>
<i>Bảng 3.3 Danh sách các yêu cầu nghiệp vụ</i>	<i>24</i>
<i>Bảng 3.4 Bảng danh sách các chức năng</i>	<i>28</i>
<i>Bảng 3.5 Bảng danh sách tác nhân.....</i>	<i>30</i>
<i>Bảng 3.6 Bảng danh sách Use Case chung cho các Actor</i>	<i>31</i>
<i>Bảng 3.7 Bảng danh sách Use Case cho Người dùng có tài khoản (RU)</i>	<i>31</i>
<i>Bảng 3.8 Bảng danh sách Use Case cho Quản trị viên khách sạn (HA).....</i>	<i>32</i>
<i>Bảng 3.9 Đặc tả Use Case “Đăng ký”</i>	<i>34</i>
<i>Bảng 3.10 Đặc tả Use Case “Tìm kiếm phòng”</i>	<i>35</i>
<i>Bảng 3.11 Đặc tả Use Case “Sắp xếp danh sách Phòng”</i>	<i>36</i>
<i>Bảng 3.12 Đặc tả Use Case “Lọc Phòng”</i>	<i>37</i>
<i>Bảng 3.13 Đặc tả Use Case “Xem thông tin Phòng”</i>	<i>38</i>
<i>Bảng 3.14 Đặc tả Use Case “Đăng nhập”</i>	<i>39</i>
<i>Bảng 3.15 Đặc tả Use Case “Đăng xuất”</i>	<i>40</i>
<i>Bảng 3.16 Đặc tả Use Case “Quên mật khẩu”</i>	<i>41</i>
<i>Bảng 3.17 Đặc tả Use Case “Xem lịch sử đặt phòng”</i>	<i>42</i>
<i>Bảng 3.18 Đặc tả Use Case “Đặt phòng”</i>	<i>44</i>
<i>Bảng 3.19 Đặc tả Use Case “Hủy đặt phòng”</i>	<i>46</i>
<i>Bảng 3.20 Đặc tả Use Case “Thanh toán trực tuyến”</i>	<i>47</i>
<i>Bảng 3.21 Đặc tả Use Case “Đánh giá Phòng”</i>	<i>49</i>
<i>Bảng 3.22 Đặc tả Use Case “Sửa thông tin tài khoản”</i>	<i>51</i>
<i>Bảng 3.23 Đặc tả Use Case “Đổi mật khẩu”</i>	<i>52</i>
<i>Bảng 3.24 Bảng mô tả chi tiết các thành phần của hệ thống</i>	<i>54</i>
<i>Bảng 3.25 Mô tả các thành phần trong Sơ đồ lớp</i>	<i>55</i>
<i>Bảng 3.26 Danh sách các sơ đồ tuần tự</i>	<i>57</i>
<i>Bảng 3.27 Các trường trong collection provinces.....</i>	<i>58</i>

<i>Bảng 3.28 Các trường trong collection “cities”</i>	59
<i>Bảng 3.29 Các trường trong collection “hotels”</i>	59
<i>Bảng 3.30 Các trường trong collection “rooms”</i>	59
<i>Bảng 3.31 Các trường trong collection “attributes”</i>	60
<i>Bảng 3.32 Các trường trong collection “reviews”</i>	60
<i>Bảng 3.33 Các trường trong collection “beds”</i>	61
<i>Bảng 3.34 Các trường trong collection “accounts”</i>	61
<i>Bảng 3.35 Các trường trong collection “review_components”</i>	61
<i>Bảng 3.36 Các trường trong collection “guests”</i>	62
<i>Bảng 3.37 Các trường trong collection “bookings”</i>	62
<i>Bảng 3.38 Các trường trong collection “payments”</i>	63
<i>Bảng 3.39 Bảng danh sách các màn hình giao diện</i>	63
<i>Bảng 4.1 Môi trường và công cụ cần cài đặt</i>	69

TÓM TẮT ĐỒ ÁN

Đặt vấn đề:

Ngành du lịch ở Việt Nam hiện nay đang duy trì tốc độ tăng trưởng mạnh trong nhiều năm, điều đó cũng góp phần phát triển kinh tế - xã hội và thúc đẩy giao lưu văn hóa.¹ Vì thế, các hệ thống đặt phòng khách sạn trực tuyến là giải pháp hiệu quả giúp du khách trong và ngoài nước dễ dàng tiếp cận với việc tìm kiếm một nơi nghỉ dưỡng uy tín trong các dịp du lịch. Các website thị trường hiện nay có thể đáp ứng được các nhu cầu cơ bản của người dùng, tuy nhiên, khi lượng người dùng truy cập quá lớn, hệ thống sẽ dễ bị tắc nghẽn, độ trễ cao và giảm tốc độ truyền tải dữ liệu. Một hệ thống nếu được xây dựng bởi kiến trúc Microservice (tức là chia các mô-đun của hệ thống thành các dịch vụ nhỏ hơn, nhằm giảm thiểu lỗi toàn bộ hệ thống) và ứng dụng gRPC để tối ưu giao tiếp giữa các service sẽ phần nào giải quyết vấn đề đó. Website đặt phòng khách sạn “Hotelligence” được thực hiện nhằm đáp ứng tiêu chí nêu trên.

Vấn đề nghiên cứu:

- Nghiên cứu quy trình nghiệp vụ về các hệ thống đặt phòng khách sạn hiện có để áp dụng thiết kế website của đồ án theo chuẩn quy trình;
- Nghiên cứu áp dụng kiến trúc Microservice và gRPC vào đồ án.

Nội dung:

Cuốn báo cáo sau đây sẽ trình bày về các nghiên cứu, quy trình thiết kế, cài đặt và triển khai sản phẩm đồ án thông qua các chương sau:

- **Chương 1 – Giới thiệu đề tài:** Trình bày lý do chọn đề tài, mục tiêu, phạm vi, đối tượng sử dụng, phương pháp thực hiện, công nghệ sử dụng và kết quả mong đợi.
- **Chương 2 – Cơ sở lý thuyết:** Trình bày một số vấn đề mà các nghiên cứu trong và ngoài nước đưa ra, từ đó đưa ra giải pháp. Đồng thời trình bày về các kiến thức nền tảng sử dụng để phục vụ cho đồ án.

¹ <https://vietnamtourism.gov.vn/post/32527>

- **Chương 3 – Phân tích và thiết kế hệ thống:** Đặc tả và mô hình hóa các yêu cầu phần mềm; trình bày các bản thiết kế của đồ án, bao gồm thiết kế hệ thống, thiết kế đối tượng, thiết kế dữ liệu và thiết kế giao diện.
- **Chương 4 – Xây dựng Website:** Trình bày quy trình cài đặt, kiểm thử và triển khai sản phẩm.
- **Chương 5 – Kết luận:** Kết luận lại những vấn đề đã giải quyết và trình bày ở các phần trước và nêu ra hướng phát triển cho sản phẩm trong tương lai.

Chương 1. GIỚI THIỆU ĐỀ TÀI

1.1. Lý do chọn đề tài

Hàng thập kỷ qua, nhu cầu du lịch và nghỉ dưỡng luôn đóng vai trò thiết yếu đối với mỗi cá nhân trong cuộc sống. Vì vậy, việc có được một địa điểm nghỉ ngơi an tâm và uy tín luôn là mối quan tâm hàng đầu của hầu hết du khách, đặc biệt là khi đặt chân đến các địa điểm nổi bật và đông đúc quanh năm như các thành phố biển hay khu du lịch trọng điểm.

Song song với đó, ngành du lịch và khách sạn đang phát triển rất mạnh mẽ, luôn hằng ngày tìm giải pháp để cải thiện chất lượng dịch vụ để thỏa mãn nhu cầu của du khách đến và đi. Qua đó, ta có thể thấy việc ứng dụng công nghệ thông tin để đặt phòng khách sạn không chỉ mang lại hiệu suất cao mà còn tối ưu hóa trải nghiệm của du khách, đặc biệt hơn là trong bối cảnh thế giới ngày càng trở nên kỹ thuật số hóa như hiện nay. Vậy nên, đề tài **“Xây dựng website đặt phòng khách sạn bằng kiến trúc Microservice kết hợp sử dụng gRPC”** được đề ra nhằm đáp ứng tiêu chí đó.

Một lý do quan trọng khác là tính linh hoạt và tiện lợi mà website đặt phòng khách sạn mang lại. Nhờ vào việc này, khách hàng cũng được hưởng lợi từ sự thuận tiện khi có thể đặt phòng, kiểm tra tình trạng đặt phòng, và thậm chí là lựa chọn dịch vụ khác thông qua website một cách nhanh chóng.

Việc ứng dụng kiến trúc Microservice vào sản phẩm không chỉ giúp cải thiện độ mượt mà về trải nghiệm người dùng mà còn giúp những nhà phát triển phần mềm có thể phát triển, vận hành và bảo trì dễ dàng hơn trong tương lai. Không những thế, khi kết hợp cùng công nghệ gRPC lại càng củng cố hiệu quả đó hơn.

1.2. Mục tiêu đề tài

Xây dựng một website hỗ trợ việc đặt phòng khách sạn với các nội dung cốt lõi, giao diện thân thiện, trực quan, hài hòa, đáp ứng được các chức năng cơ bản và cần thiết.

Đồng thời, sử dụng kiến trúc Microservice để chia nhỏ các thành phần của ứng dụng thành các dịch vụ (service) con, giúp ứng dụng trở nên dễ bảo trì và sửa chữa, đồng thời kết hợp gRPC để cải thiện hiệu suất giao tiếp giữa các service.

1.3. Phạm vi

1.3.1. Phạm vi môi trường

Triển khai sản phẩm đề tài trên môi trường web.

1.3.2. Phạm vi chức năng

- Các chức năng liên quan đến tài khoản:
 - Đăng nhập
 - Đăng ký
 - Đăng xuất
 - Quên mật khẩu
 - Chính sửa thông tin tài khoản
- Các chức năng liên quan đến phòng khách sạn:
 - Tìm kiếm, Sắp xếp, Lọc
 - Đánh giá và Bình luận
- Các chức năng liên quan đến đặt phòng:
 - Thêm phòng vào giỏ hàng
 - Xem và chỉnh sửa thông tin phòng cần đặt
- Các chức năng liên quan đến thanh toán:
 - Có thể kết hợp thanh toán thông qua các cổng thanh toán điện tử
- Các chức năng quản lý website:

- Quản lý tài khoản
- Quản lý bài viết
- Quản lý phòng
- Quản lý đơn hàng
- Xem báo cáo và thống kê doanh thu
- Và các chức năng khác liên quan (có thể phát sinh trong quá trình phát triển).

1.4. Đối tượng sử dụng

- Người dùng khách
- Người dùng có tài khoản
- Quản trị viên khách sạn

1.5. Phương pháp thực hiện

- Tìm hiểu công nghệ xây dựng website gồm: Figma, ReactJS, MongoDB, Java, v.v.
- Tìm hiểu các kiến trúc hiện hành để đưa ra so sánh và lựa chọn phù hợp cho đề tài.
- Xác định yêu cầu từ phía khách hàng, từ đó phân tích tính năng cụ thể cho đề tài. Đồng thời tham khảo các hệ thống đặt phòng khách sạn sẵn có trên thị trường.
- Phân tích, thiết kế hệ thống website.
- Thiết kế:
 - Thiết kế đối tượng.
 - Thiết kế dữ liệu.
 - Thiết kế giao diện.
- Cài đặt.
- Triển khai và kiểm thử.
- Hoàn thiện sản phẩm.

1.6. Công nghệ sử dụng

- Công cụ thiết kế UI/UX: Figma.
- Ngôn ngữ sử dụng: Javascript, Java.
- Xây dựng website: HTML, CSS.
- Thư viện/Framework sử dụng chính:
 - Front-end: ReactJS.
 - Back-end: Spring Boot.
- Database: MongoDB.
- Source Control: GitHub.
- Các công cụ hỗ trợ khác (có thể có): Postman, Docker, Kubernetes, v.v.

1.7. Kết quả mong đợi

- Nắm bắt và áp dụng các công nghệ mới để hoàn thành sản phẩm đề tài.
- Hiểu rõ các nghiệp vụ, chức năng của một website thương mại điện tử, cụ thể là đặt phòng khách sạn.
- Xây dựng website hoàn chỉnh đáp ứng yêu cầu về giao diện và chức năng đã đề ra.
- Có thể thay đổi giao diện một cách linh động và mở rộng thêm các chức năng mới cho website đề tài để phù hợp với nhu cầu thực tiễn trong tương lai.

Chương 2. CƠ SỞ LÝ THUYẾT

2.1. Những hướng nghiên cứu đã có

2.1.1. Các nghiên cứu ngoài nước

Một bài nghiên cứu ở nước ngoài đã chỉ ra rằng, việc đặt phòng khách sạn thủ công bằng cách đi bộ đến quầy lễ tân hoặc gọi điện thông qua dịch vụ bên thứ ba và các yêu cầu đặt phòng này sau đó sẽ được chuyển tiếp về phía khách sạn. Tuy nhiên, đôi khi việc quản lý và tiếp nhận không được chặt chẽ có thể dẫn đến những báo cáo sai lệch liên quan đến quy trình làm việc của khách sạn cũng như việc giá cả không phải lúc nào cũng được thông tin một cách chính xác tới khách hàng khi có sự thay đổi theo thời gian.²

Tương tự, bài nghiên cứu cũng nói rằng, thông tin chi tiết về khách hàng hầu như không được sử dụng trong quy trình làm việc và cũng không được lưu trữ đúng cách; khi những sổ sách về thông tin khách hàng bị đầy thì chúng cũng sẽ bị xử lý. Điều đó có thể sẽ là mối nguy hiểm tiềm tàng về khía cạnh bảo mật thông tin khách hàng.

2.1.2. Các nghiên cứu trong nước

Ở một khía cạnh khác, theo một kết quả nghiên cứu được đăng trên Tạp chí Tài chính³, có 6 yếu tố ảnh hưởng đến ý định đặt phòng trực tuyến của khách du lịch nội địa, (trường hợp hệ thống khách sạn 4-5 sao TP. Đà Nẵng), xếp theo thứ tự ảnh hưởng giảm dần bao gồm: Giá cả; Yếu tố niềm tin; Nhận thức tính hữu dụng; Hệ thống thanh toán; Truyền miệng điện tử; Nhận thức tính dễ sử dụng.

² Bemile, Richard & Achampong, Akwasi & Danquah, Emmanuel. (2014). Online Hotel Reservation System. International Journal of Innovative Science, Engineering & Technology.

³ Thang, Nguyen. (2019). Yếu tố ảnh hưởng đến ý định đặt phòng trực tuyến của khách du lịch nội địa.

Qua đó, tác giả cũng gợi ý rằng chúng ta có thể nâng cao ý định đặt phòng trực tuyến thông qua những giải pháp như:

- Đưa ra các chương trình ưu đãi hoặc tích điểm;
- Tạo dựng thương hiệu doanh nghiệp minh bạch, công khai để tạo lòng tin cho khách hàng cũng như bảo mật thông tin khách hàng và thông tin đặt phòng;
- Mở rộng kết nối với khách hàng thông qua các kênh truyền thông xã hội như Facebook, Zalo... để khách hàng dễ dàng tìm kiếm;
- Triển khai nhiều hình thức thanh toán phù hợp thói quen tiêu dùng;
- Khách hàng tin rằng việc tham khảo ý kiến của người dùng khác trên website là linh hoạt, hữu ích và đáng tin cậy. Vì vậy việc xây dựng chất lượng về dịch vụ và sản phẩm là điều thiết yếu;
- Doanh nghiệp cần cải thiện quy trình để đặt phòng khách sạn trực tuyến đơn giản, giao diện thân thiện, dễ sử dụng, không đòi hỏi nhiều nỗ lực gây phức tạp.

2.1.3. Đánh giá

Qua các cuộc khảo sát nêu trên, ta có thể dễ dàng nhận thấy sự quan trọng và cần thiết của một hệ thống đặt phòng khách sạn. Đó không chỉ là tạo niềm tin mà còn là phương thức nâng cao nhận thức cho người dùng hiện đại trong thời kỳ kỹ thuật số.

2.2. Những vấn đề còn tồn tại

Ở phần trên, ta có thể nêu ra một số điểm hạn chế mà hệ thống quản lý khách sạn cũng như đặt phòng trực tuyến hiện hữu như: ***tính an toàn, tính bảo mật và tính tiện dụng***.

2.3. Những vấn đề cần tập trung và giải quyết

Vì vậy, hiện tại, để đảm bảo chất lượng phần mềm cơ bản, ta cần đáp ứng trước hết là ***tính đúng đắn*** và ***tính tiến hóa***, kết hợp những tiêu chí đã nêu ở

2.2; đồng thời, ta sẽ có giải pháp để giải quyết thêm một số vấn đề liên quan đến **tính hiệu quả**, nhằm đáp ứng cho người dùng nhu cầu về tốc độ truy cập cũng như có thể nhập/xuất dữ liệu đa hình thức.

2.4. Kiến trúc hệ thống

2.4.1. Các kiểu kiến trúc

2.4.1.1. Khái niệm

Kiểu kiến trúc (hay mẫu kiến trúc) là một tập các nguyên lý, mẫu thô, cung cấp khung trù tượng cho một tập các hệ thống. Lợi ích nhằm tạo ra ngôn ngữ chung và phục vụ cho việc tái sử dụng.

2.4.1.2. Phân loại

Kiểu kiến trúc được phân loại dựa vào phương diện phục vụ của kiểu đó, có thể tham khảo bảng sau:

Bảng 2.1 Bảng phân loại các kiểu kiến trúc

Phương diện phục vụ	Kiểu kiến trúc
Giao tiếp (Communication)	Service-Oriented Architecture (SOA), Message Bus
Triển khai (Deployment)	Client/Server, N-Tier, 3-Tier
Miền (Domain)	Domain Driven Design
Cấu trúc (Structure)	Component-Based, Object-Oriented, Layered Architecture

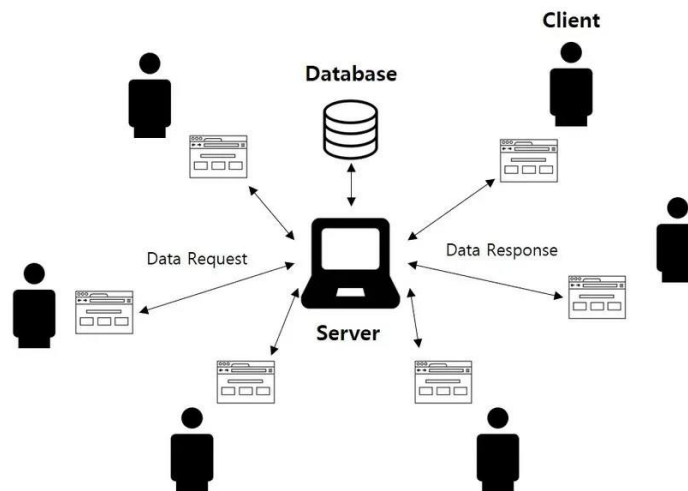
2.4.2. Mô hình Khách-Chủ (Client-Server)

2.4.2.1. Khái niệm

Kiểu Client-Server (Khách-Chủ) cho hệ thống phân tán với hai thành phần riêng biệt là Khách (Client) và Chủ (Server). Trong mô hình này, client và server tương tác với nhau thông qua mạng hoặc Internet.

Trong mô hình client-server, server đóng vai trò như là một trung tâm điều khiển, cung cấp các dịch vụ và tài nguyên cho các client. Các client được cấp quyền truy cập vào các tài nguyên và dịch vụ của server thông qua các giao thức và phương thức truy cập như HTTP, FTP, SSH, Telnet, và nhiều hơn nữa.

Các ứng dụng client-server rất phổ biến trong các môi trường mạng và Internet, bao gồm các ứng dụng web, email, trò chơi trực tuyến, hệ thống quản lý cơ sở dữ liệu, và nhiều hơn nữa. Mô hình client-server cho phép các ứng dụng phân tán trên nhiều máy tính và cho phép các người dùng truy cập và chia sẻ dữ liệu và tài nguyên một cách dễ dàng.⁴



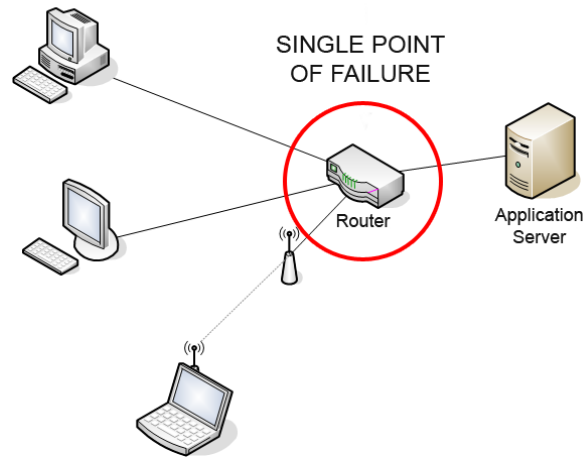
Hình 2.1 Mô hình Client-Server

2.4.2.2. Ưu điểm và nhược điểm

- Ưu điểm:
 - Chia sẻ tài nguyên
 - Thuận tiện cho việc bảo trì
 - An ninh: dữ liệu được giữ ở server nên được bảo vệ tốt hơn
- Nhược điểm:
 - Nghiệp vụ và dữ liệu thường được đề chung phía server

⁴ <https://helpdesk.inet.vn/blog/mo-hinh-client-server>

- Điểm lỗi duy nhất (Single point of failure)⁵



Hình 2.2 Điểm lỗi duy nhất

2.4.3. Kiến trúc hướng dịch vụ (Service-Oriented Architecture)

2.4.3.1. Khái niệm

Theo AWS, kiến trúc hướng dịch vụ (SOA) là một phương pháp phát triển phần mềm sử dụng các thành phần của phần mềm được gọi là dịch vụ để tạo ra các ứng dụng dành cho doanh nghiệp. Mỗi dịch vụ cung cấp một tính năng doanh nghiệp, đồng thời các dịch vụ cũng có thể giao tiếp với nhau giữa nhiều nền tảng và ngôn ngữ. Nhà phát triển tận dụng SOA để tái sử dụng các dịch vụ trong nhiều hệ thống khác nhau hoặc kết hợp một số dịch vụ độc lập để thực hiện các tác vụ phức tạp.⁶

2.4.3.2. Ưu điểm và nhược điểm

- Ưu điểm:
 - Giảm thiểu một khoản chi phí trong quá trình phát triển
 - Giảm thiểu các yêu cầu về đào tạo và kỹ năng.
 - Khoản phí bảo hành thấp
 - Chu trình phát triển phần mềm nhanh chóng và dễ dàng hơn.

⁵ Một điểm lỗi duy nhất là một phần của hệ thống, nếu nó bị lỗi, toàn bộ hệ thống sẽ ngừng hoạt động.

⁶ Nguồn: <https://aws.amazon.com/vi/what-is/service-oriented-architecture/>

- Định hướng kinh doanh: SOA giúp cho việc cấu trúc lại quy trình hoạt động của một tổ chức dựa trên nền tảng công nghệ
- Nhược điểm:
 - Khả năng điều chỉnh quy mô hạn chế
 - Gia tăng sự phụ thuộc lẫn nhau
 - Điểm lỗi chí mạng đơn lẻ

2.4.4. Kiến trúc vi dịch vụ (Microservice Architecture)

2.4.4.1. Khái niệm

Microservices là các module trong hệ thống được chia thành nhiều services nhỏ. Mỗi service sẽ thực hiện các chức năng chuyên biệt, như quản lý đơn hàng hoặc quản lý khách hàng,... và được đặt tại một server riêng, cho phép nâng cấp chỉnh sửa một cách độc lập. Các server này có thể giao tiếp thông qua các phương thức như gRPC, Rest API, lambda và không bị ảnh hưởng bởi nhau.

Việc áp dụng kiến trúc microservices cho phép chia nhỏ chức năng của ứng dụng thành các dịch vụ nhỏ, tối ưu hóa trải nghiệm và tốc độ cho từng người dùng. Thiết kế giao diện dựa trên từng đối tượng giúp cải thiện tương thích và tốc độ, đồng thời giảm thiểu các chức năng không cần thiết.⁷

2.4.4.2. Ưu điểm và nhược điểm

- Ưu điểm:
 - Thiết kế kiến trúc tốt hơn cho những ứng dụng lớn
 - Nhanh nhạy, linh động với những dự án dài
 - Dễ học, dễ tuyển dụng để triển khai
 - Khả năng kiểm soát khủng hoảng (lỗi), các mô đun được cách ly, không ảnh hưởng đến các dịch vụ khác

⁷ Nguồn: <https://www.linkedin.com/pulse/microservices-l%C3%A0-g%C3%AC-%E1%BB%A9ng-d%E1%BB%A5ng-c%E1%BB%A7a-ki%E1%BA%BFn-tr%C3%BAC-n%C3%A0y-nh%C6%B0-th%E1%BA%BF-n%C3%A0o>

- Nhược điểm:
 - Nhiều phần phải di chuyển
 - Các yêu cầu liên quan đến cơ sở hạ tầng phức tạp
 - Tính nhất quán và Tính sẵn sàng trong hệ thống
 - Khó kiểm thử

2.5. Công nghệ áp dụng

2.5.1. Next.js



Hình 2.3 Logo Next.js

Next.js (hay NextJS, Nextjs) là framework mã nguồn mở được xây dựng dựa trên nền tảng React, cho phép xây dựng các website nhanh hơn với các thư viện và components có sẵn (như NextUI).

Một số tính năng nổi bật có thể kể đến như:

- Automatic Routing: Tự động định tuyến các trang đích dựa vào tên thư mục. Giả sử muốn đi đến trang “/blog” thì chỉ cần tạo thư mục tên “blog” và định nghĩa trang “page.js” bên trong đó. Next sẽ tự hiểu và routing tới.
- Nested Routing/Dynamic Routing: Có thể lồng các router lại bên trong thư mục cha, giả sử muốn đến trang “/blog/{id}” thì tạo thư mục tên “blog” rồi tiếp tục tạo thư mục “[id]” bên trong, tạo trang page.js bên trong nữa thì Next sẽ tự hiểu và routing tới.
- Server-side Rendering (SSR): Cho phép tài nguyên thực hiện các tác vụ ở bên phía server, không phải mất thời gian xử lý ở phía client nữa nên khi người dùng sử dụng trang web, những thành phần trong đó sẽ được render ra nhanh hơn. Ngoài ra còn có Client-side Rendering (CSR), ngược lại với SSR.

- Và còn nhiều tính năng khác ở các bản cập nhật tiếp theo.

2.5.2. Java Spring Boot



Hình 2.4 Logo Spring Framework

Spring Boot là một trong số các module của Spring framework chuyên cung cấp các tính năng RAD (Rapid Application Development) cho phép tạo ra và phát triển các ứng dụng độc lập dựa trên Spring một cách nhanh chóng.

Spring Boot ra đời với mục đích loại bỏ những cấu hình phức tạp của Spring, nó không yêu cầu cấu hình XML và nâng cao năng suất cho các nhà phát triển. Với sự góp mặt của Spring Boot, hệ sinh thái Spring đã trở nên mạnh mẽ, phổ biến và hiệu quả hơn bao giờ hết.⁸

Những ưu điểm có thể kể đến:

- Hội tụ đầy đủ các tính năng của Spring framework.
- Đơn giản hóa cấu hình và xây dựng được các ứng dụng độc lập có khả năng chạy bằng “java -jar” nhờ các dependency starter.
- Dễ dàng deploy vì các ứng dụng server được nhúng trực tiếp vào ứng dụng để tránh những khó khăn khi triển khai lên môi trường production mà không cần thiết phải tải file WAR.
- Cấu hình ít, tự động hỗ trợ bất cứ lúc nào cho chức năng giống với Spring như tăng năng suất, giảm thời gian viết code và không yêu cầu XML config.
- Cung cấp nhiều plugin, số liệu, cấu hình ứng dụng từ bên ngoài.

⁸ <https://stringee.com/vi/blog/post/java-spring-boot-la-gi>

2.5.3. MongoDB



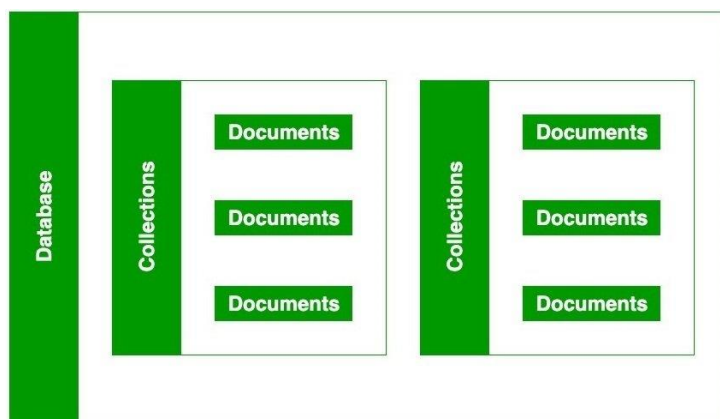
Hình 2.5 Logo MonogoDB

MongoDB là một hệ thống cơ sở dữ liệu phi quan hệ (NoSQL), mã nguồn mở, được phát hành lần đầu vào tháng 2 năm 2009 bởi MongoDB Inc và quản lý theo SSPL (Server Side Public License).

MongoDB lưu trữ dữ liệu dưới dạng tài liệu (document) JSON, cho phép các ứng dụng lưu trữ và truy vấn dữ liệu một cách linh hoạt và hiệu quả.

MongoDB có nhiều ưu điểm như khả năng lưu trữ dữ liệu phân tán, linh hoạt trong cấu trúc dữ liệu, có thể mở rộng, tốc độ truy vấn nhanh và hỗ trợ các tính năng như indexing, replication, sharding và map-reduce.

MongoDB hoạt động dưới dạng một hệ thống cơ sở dữ liệu phi quan hệ, lưu trữ dữ liệu dưới dạng tài liệu (document) JSON. Dữ liệu được lưu trữ trong collections và documents. Do đó, database, collection và documents sẽ có liên quan với nhau như hình dưới đây:⁹



Hình 2.6 Cách MongoDB hoạt động

⁹ <https://200lab.io/blog/mongodb-la-gi/>

2.5.4. RESTful API

2.5.4.1. Định nghĩa

- **API (Application Programming Interface)** là một tập các quy tắc và cơ chế mà theo đó, một ứng dụng hay một thành phần sẽ tương tác với một ứng dụng hay thành phần khác. API có thể trả về dữ liệu mà bạn cần cho ứng dụng của mình ở những kiểu dữ liệu phổ biến như JSON hay XML.
- **REST (REpresentational State Transfer)** là một dạng chuyển đổi cấu trúc dữ liệu, một kiểu kiến trúc để viết API. Nó sử dụng phương thức HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.
- **RESTful API** là một tiêu chuẩn dùng trong việc thiết kế các API cho các ứng dụng web để quản lý các resource. RESTful là một trong những kiểu thiết kế API được sử dụng phổ biến ngày nay để cho các ứng dụng (web, mobile...) khác nhau giao tiếp với nhau. Nó chú trọng vào tài nguyên hệ thống (tệp văn bản, ảnh, âm thanh, video, hoặc dữ liệu động...), bao gồm các trạng thái tài nguyên được định dạng và được truyền tải qua HTTP.

2.5.4.2. Cách thức hoạt động

REST hoạt động chủ yếu dựa vào giao thức HTTP. Các hoạt động cơ bản nêu trên sẽ sử dụng những phương thức HTTP riêng.

- GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
- POST (CREATE): Tạo mới một Resource.
- PUT (UPDATE): Cập nhật thông tin cho Resource.
- DELETE (DELETE): Xóa một Resource.

Những phương thức hay hoạt động này thường được gọi là CRUD tương ứng với Create, Read, Update, Delete – Tạo, Đọc, Sửa, Xóa.

2.5.5. gRPC



Hình 2.7 Logo gRPC

gRPC là một framework RPC mã nguồn mở, hiện đại và hiệu năng cao mà có thể chạy trên bất kỳ môi trường nào. Framework này được Google khởi công phát triển vào năm 2015, đến 08/2016 thì được phát hành chính thức. Đây được cho là một thế hệ tiếp theo của RPC (Remote Procedure Calls) đặc biệt là trong mô hình Microservices.

Để tăng tải cho cả hệ thống nhiều services (hay Microservices), Google đã phát triển 2 thứ:

1. Một giao thức mới để tối ưu các connection, đảm bảo dữ liệu đi trao đổi liên tục với ít băng thông nhất có thể.
2. Một định dạng dữ liệu mới để 2 đầu service (hoặc client và server) có thể hiểu được các message của nhau mà ít phải encode/decode.

Đầu tiên Google phát triển một giao thức thay thế cho HTTP/1.1 với tên gọi SPDY. Sau này giao thức này được open source thậm chí chuẩn hoá, lấy làm nền móng cho giao thức HTTP/2. Khi có HTTP/2 rồi thì giao thức SPDY ngừng phát triển. gRPC chính thức hoạt động trên HTTP/2 luôn từ sau năm 2015.

2.5.6. Clerk



Hình 2.8 Logo Clerk

Clerk (clerk.com) là nền tảng cho phép tích hợp vào ứng dụng để thực hiện các tính năng liên quan đến xác thực và quản lý người dùng, đồng thời cung cấp những components về Đăng nhập, Đăng ký,... có sẵn để tối ưu hóa quy trình phát triển phần mềm.

Một số tính năng của Clerk có thể kể đến như:

- Xác thực nhiều yếu tố: Người dùng có thể xác thực thông qua Email (code hoặc link), hay qua các nền tảng mạng xã hội khác như Google, GitHub, v.v.
- Bảo mật nâng cao: Clerk được chứng nhận SOC 2 Type 2 tuân thủ theo CCPA, đồng thời cũng thực hiện kiểm toán bên thứ ba (third-party) thường xuyên để đảm bảo yêu cầu bảo mật.
- Quản lý phiên làm việc (Session): Cho phép quản lý các phiên đăng nhập, đăng ký, đăng xuất, ở thiết bị nào,... của người dùng.

Hơn hết, Clerk rất phù hợp để tích hợp phát triển cho các dự án của Next.js vì nó hỗ trợ rất mạnh cho framework này, giúp giảm thiểu thời gian khi không cần phải định nghĩa service riêng để quản lý người dùng.

2.5.7. Docker



Hình 2.9 Logo Docker

Docker là nền tảng cho phép phát triển và vận hành phần mềm thông qua các container, nó hoạt động bằng cách tạo các môi trường ảo (như cách máy ảo ảo hóa phần cứng máy chủ) để phần mềm có thể đa dạng thích ứng hoạt động trên các môi trường đó.

Docker có khá nhiều thuật ngữ cần tìm hiểu, tuy nhiên chúng ta chỉ cần hiểu khái niệm của một số thứ như sau:

- **Image:** Image là một tệp chỉ đọc chứa mọi thứ cần thiết để chạy một ứng dụng, bao gồm mã nguồn, thư viện, biến môi trường và file cấu hình. Images được sử dụng để tạo ra containers. Một image có thể được dựa trên một image khác, cộng thêm một số tùy chỉnh.
- **Container:** Container là một instance của image. Nó là một môi trường thực thi ứng dụng cô lập, nơi các ứng dụng được chạy.
- **Dockerfile:** Dockerfile là một tệp văn bản chứa các chỉ dẫn để xây dựng một Docker image. Dockerfile mô tả các bước cần thiết để tạo ra một image từ một base image.
- **Docker Hub:** Docker Hub là một registry công cộng nơi Docker images có thể được lưu trữ và chia sẻ. Người dùng có thể tải lên hoặc tải xuống các images từ Docker Hub.

2.5.8. Zipkin



Hình 2.10 Logo Zipkin

Zipkin là hệ thống truy vết phân tán, giúp thu thập và tra cứu dữ liệu thời gian để khắc phục sự cố độ trễ trong kiến trúc dịch vụ. Người dùng có

thể truy vấn dữ liệu theo nhiều thuộc tính và xem các biểu đồ phụ thuộc để xác định hành vi tổng hợp. Các ứng dụng cần được cấu hình để báo cáo dữ liệu tới Zipkin, thường qua HTTP hoặc Kafka, với nhiều tùy chọn khác như Apache ActiveMQ, gRPC và RabbitMQ. Dữ liệu được lưu trữ trong bộ nhớ hoặc với backend như Apache Cassandra hoặc Elasticsearch.

2.5.9. JUnit5



Hình 2.11 Logo JUnit5

JUnit là một framework kiểm thử cho Java, giúp viết và chạy các unit test. Các tính năng chính của JUnit bao gồm hỗ trợ các annotation như `@Test`, `@Before`, `@After`, và `@Ignore` để đánh dấu và quản lý các phương thức kiểm thử, cùng với các phương thức assertion như `assertEquals` và `assertTrue` để kiểm tra kết quả.

JUnit còn cung cấp khả năng tạo Test Suites để chạy nhiều kiểm thử cùng lúc, hỗ trợ kiểm thử tham số hóa để kiểm thử với nhiều bộ dữ liệu, và tích hợp tốt với các IDE và công cụ xây dựng như Maven và Gradle. Điều này giúp việc kiểm thử phần mềm trở nên thuận tiện và hiệu quả.

Hiện tại JUnit đang ở phiên bản mới nhất là JUnit5.

Chương 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1. Khảo sát hiện trạng

Đề tài có tham khảo và khảo sát một số chức năng trên một số sản phẩm hiện hành như Agoda, Trivago, Booking.com, Traveloka, Airbnb và đảm bảo được các chức năng cơ bản như các website nêu trên.

3.2. Xác định yêu cầu

3.2.1. Yêu cầu chức năng

Đảm bảo đầu ra cho các chức năng nêu ở mục **4.3.2. Danh sách chức năng**.

3.2.2. Yêu cầu phi chức năng

Bảng 3.1 Danh sách các yêu cầu phi chức năng

STT	Mã yêu cầu phi chức năng	Tên yêu cầu phi chức năng
1	NFR01	Bảo mật (Security)
2	NFR02	Hiệu suất (Performance)
3	NFR03	Dễ sử dụng (Usability)
4	NFR04	Giao diện (User Interface)
5	NFR05	Tuân thủ (Compliance)
6	NFR06	Khả năng mở rộng (Scalability)
7	NFR07	Mở rộng phạm vi chức năng (Extensibility)

Bảng 3.2 Bảng phân rã chi tiết các yêu cầu phi chức năng

STT	Mã yêu cầu phi chức năng	Mã yêu cầu chi tiết	Mô tả
1	NFR01	NFR01-UC01	Hệ thống phải đảm bảo tính bảo mật thông tin cá nhân dùng để đăng ký của khách hàng.
2		NFR01-UC06	Hệ thống phải đảm bảo tính bảo mật thông tin cá nhân dùng để đăng nhập của khách hàng.
3		NFR01-UC08	Hệ thống phải đảm bảo tính bảo mật thông tin cá nhân dùng để khôi phục tài khoản và mật khẩu mới của khách hàng.
4		NFR01-UC11	Hệ thống phải đảm bảo tính bảo mật thông tin cá nhân dùng để đặt phòng của khách hàng.
5		NFR01-UC13	Hệ thống phải đảm bảo tính bảo mật thông tin cá nhân dùng để thanh toán của khách hàng.
6		NFR01-UC15	Hệ thống phải đảm bảo tính bảo mật thông tin cá nhân mới của khách hàng.
7		NFR01-UC16	Hệ thống phải đảm bảo tính bảo mật thông tin báo cáo và thống kê doanh thu của khách sạn.
8		NFR01-UC24	Hệ thống phải đảm bảo tính bảo mật thông tin cá nhân dùng để

			xác thực tài khoản và mật khẩu mới của khách hàng.
9	NFR02		Hệ thống phải có khả năng xử lý nhiều yêu cầu đồng thời và hoạt động ổn định, liên tục.
10	NFR03		Hệ thống phải có giao diện dễ hiểu và dễ sử dụng.
11	NFR04		Hệ thống phải có giao diện đẹp và thân thiện với người dùng.
12	NFR05	NRF05-UC14	Hệ thống phải đảm bảo các bình luận được hiển thị không vi phạm tiêu chuẩn cộng đồng và chính sách của tổ chức.
13		NRF05-UC23	Hệ thống phải đảm bảo các bình luận trả lời của quản trị viên khách sạn không vi phạm tiêu chuẩn cộng đồng và chính sách của tổ chức.
14	NRF06	NRF06-UC02	Hệ thống phải đảm bảo khả năng mở rộng phạm vi và từ khóa tìm kiếm trong tương lai.
15		NRF06-UC03	Hệ thống phải đảm bảo khả năng mở rộng để đáp ứng các tiêu chí sắp xếp phòng khác trong tương lai.
16		NRF06-UC04	Hệ thống phải đảm bảo khả năng mở rộng để đáp ứng các tiêu chí lọc phòng khác trong tương lai.

17		NRF06-UC06	Hệ thống phải đảm bảo khả năng mở rộng các hình thức đăng nhập khác trong tương lai.
18	NFR07	NFR07-UC06	Hệ thống phải đảm bảo khả năng tích hợp với các bên thứ ba để khách hàng có thể đăng nhập vào website.
19		NFR07-UC08	Hệ thống phải đảm bảo khả năng tích hợp với hệ thống hỗ trợ gửi email xác thực khách hàng để thay đổi mật khẩu của tài khoản.
20		NRF07-UC16	
21		NFR07-UC13	Hệ thống phải đảm bảo khả năng tích hợp với các hệ thống thanh toán trực tuyến.

3.2.3. Yêu cầu nghiệp vụ

Bảng 3.3 Danh sách các yêu cầu nghiệp vụ

STT	Mã yêu cầu	Mô tả
1	BR01	Email được sử dụng phải là một email hợp lệ
2	BR02	Mật khẩu của tài khoản phải có ít nhất 8 ký tự và bao gồm ít nhất một chữ số và một chữ cái viết hoa.
3	BR03	“Ngày checkin” mặc định là “[Ngày hiện tại] + 1” và “Ngày checkout” mặc định là “[Ngày checkin] + 1”
4	BR04	“Số lượng khách” mặc định là 2 khách, “Loại khách” là “Người lớn”

5	BR05	Tiêu chí “Giá từ thấp đến cao” và “Giá từ cao đến thấp” là bắt buộc phải có.
6	BR06	Tiêu chí lọc theo số sao của khách sạn là bắt buộc phải có.
7	BR07	Tên phòng và giá phòng là ưu tiên hàng đầu để hiển thị.
8	BR08	Tài khoản đăng nhập của khách hàng phải là tài khoản hợp lệ đã được lưu trên hệ thống.
9	BR09	Chỉ người dùng có tài khoản mới có thể xem lịch sử đặt phòng của mình.
10	BR10	Khách hàng chỉ có thể chỉnh sửa thông tin phòng nếu phòng đó vẫn còn trống trong thời gian đặt.
11	BR11	Khách hàng không thể chỉnh sửa thông tin phòng nếu đã thanh toán cho đơn đặt phòng.
12	BR12	Khách hàng chỉ có thể đặt phòng nếu phòng đó còn trống trong thời gian đặt.
13	BR13	Khách hàng chỉ có thể đặt phòng sau sau khi liên kết thẻ tín dụng thành công.
14	BR14	Nếu khách hàng chọn phương thức thanh toán trực tuyến và [thời gian nhận phòng] - [thời gian đặt phòng] \geq 96 (giờ), khách hàng phải thanh toán đầy đủ tiền phòng trong vòng 72 giờ tiếp theo nếu không phòng đã đặt sẽ tự động được hủy.
15	BR15	Nếu khách hàng chọn phương pháp thanh toán trực tuyến và [thời gian nhận phòng] - [thời gian đặt phòng] $<$ 96 (giờ), khách hàng phải thanh toán đầy đủ tiền phòng trong vòng 24 giờ tiếp theo nếu không phòng đã đặt sẽ tự động được hủy.

16	BR16	Khách hàng chỉ có thể hủy phòng đã đặt của chính mình.
17	BR17	Việc hoàn tiền sau khi hủy phòng sẽ dựa theo Chính sách hủy đã được khách hàng và khách sạn thông qua ở mục Đặt phòng.
18	BR18	Mỗi khách hàng chỉ được đánh giá một lần cho mỗi phòng.
19	BR19	Khách hàng chỉ được phép đánh giá các phòng mà họ đã từng đặt.
20	BR20	Đánh giá của khách hàng phải tuân thủ các quy định về nội dung của Hotelligence.
21	BR21	Khách hàng chỉ có thể bình luận tối đa 256 ký tự.
22	BR22	Khách hàng chỉ có thể sửa đổi thông tin cá nhân của họ.
23	BR23	Hệ thống phải đảm bảo tính chính xác về định dạng của các thông tin cần cập nhật.
24	BR24	Khách hàng chỉ có thể đổi mật khẩu sau khi nhận được email xác thực tài khoản của họ.

3.3. Mô hình hóa yêu cầu

3.3.1. Lược đồ Use Case



Hình 3.1 Lược đồ Use Case

3.3.2. Danh sách chức năng

Bảng 3.4 Bảng danh sách các chức năng

STT	Chức năng	Bao gồm	Mô tả
1	Xác thực tài khoản	<ul style="list-style-type: none">• Đăng ký• Đăng nhập• Quên mật khẩu• Đăng xuất	Cho phép người dùng sử dụng tài khoản để tiến vào hệ thống, thực hiện các chức năng cao hơn; cho phép người dùng lấy lại mật khẩu đã quên thông qua công xác thực.
2	Duyệt phòng khách sạn	<ul style="list-style-type: none">• Tìm kiếm Khách sạn• Sắp xếp Danh sách Khách sạn• Lọc Danh sách Khách sạn• Xem thông tin Khách sạn• Xem thông tin Phòng	Cho phép người dùng tìm kiếm, sắp xếp, lọc và xem thông tin khách sạn trong danh sách khách sạn có sẵn.
3	Đặt phòng khách sạn	<ul style="list-style-type: none">• Đặt phòng• Hủy đặt phòng• Xem lịch sử đặt phòng	Cho phép người dùng có tài khoản có thể đặt, hủy và xem lại các lịch sử đặt phòng của mình trên website.

4	Thanh toán	<ul style="list-style-type: none"> • Thanh toán trực tuyến 	Cho phép người dùng thanh toán tiền phòng đã đặt.
5	Đánh giá Phòng	<ul style="list-style-type: none"> • Đánh giá Phòng 	Cho phép người dùng có thể đánh giá và bình luận những phòng đã đặt.
6	Tùy chỉnh tài khoản	<ul style="list-style-type: none"> • Sửa thông tin tài khoản • Đổi mật khẩu 	Cho phép người dùng có thể sửa thông tin tài khoản và đổi mật khẩu tài khoản hiện hữu.
7	Quản lý Khách sạn	<ul style="list-style-type: none"> • Thêm Phòng • Xóa Phòng • Sửa thông tin Phòng 	Cho phép quản trị viên khách sạn có thể tùy chỉnh danh sách phòng và thông tin phòng như thêm, xóa, sửa.

3.3.3. Danh sách tác nhân (Actor)

Bảng 3.5 Bảng danh sách tác nhân

STT	Mã Actor	Tên Actor	Mô tả
1	GS	Người dùng khách/chưa có tài khoản (Guest)	Người dùng khách có thể xem danh sách phòng khách sạn. Họ có thể đăng ký để trở thành RU và thực hiện các chức năng bên trong.
2	RU	Người dùng có tài khoản (Registered User)	Người dùng có tài khoản có thể đăng nhập vào hệ thống để thực hiện chức năng đặt phòng và các chức năng bên trong có liên quan.
3	HA	Quản trị viên khách sạn (Hotel Admin)	Quản trị viên khách sạn là những người trực tiếp quản lý danh sách các phòng cũng như đơn hàng của khách sạn.

3.3.4. Danh sách trường hợp sử dụng (Use Case)

Bảng 3.6 Bảng danh sách Use Case chung cho các Actor

STT	Mã Use Case	Tên Use Case	Mô tả
1	GS-UC01	Đăng ký	Người dùng chưa có tài khoản có thể đăng ký để trở thành RU hoặc HA.
2	GS-UC02	Tìm kiếm Phòng	Người dùng có thể tìm kiếm phòng thông qua từ khóa như thành phố hoặc tên khách sạn.
3	GS-UC03	Sắp xếp Danh sách Phòng	Người dùng có thể sắp xếp danh sách theo các tiêu chí có sẵn.
4	GS-UC04	Lọc Phòng	Người dùng có thể lọc danh sách phòng theo các tiêu chí có sẵn.
5	GS-UC05	Xem thông tin Phòng	Người dùng có thể xem chi tiết thông tin phòng khi nhấn chọn vào phòng bất kỳ.

Bảng 3.7 Bảng danh sách Use Case cho Người dùng có tài khoản (RU)

STT	Mã Use Case	Tên Use Case	Mô tả
1	RU-UC06	Đăng nhập	Người dùng có thể đăng nhập vào hệ thống thông qua tài khoản đã đăng ký.
2	RU-UC07	Đăng xuất	Người dùng có thể đăng xuất (hay thoát) khỏi hệ thống.
3	RU-UC08	Quên mật khẩu	Người dùng có thể lấy lại tài khoản thông qua xác thực email trong trường hợp người dùng đã quên.

4	RU-UC09	Xem lịch sử đặt phòng	Người dùng có thể xem danh sách các phòng đã đặt trên website Hotelligence.
5	RU-UC10	Đặt phòng	Người dùng nhập thông tin đặt phòng và xác nhận đặt phòng để đặt phòng.
6	RU-UC11	Hủy đặt phòng	Người dùng có thể hủy phòng đã đặt nhưng chưa thanh toán.
7	RU-UC12	Thanh toán trực tuyến	Người dùng thanh toán tiền phòng trực tuyến qua các cổng thanh toán hoặc thẻ ngân hàng.
8	RU-UC13	Đánh giá Phòng	Người dùng đã lưu trú tại khách sạn có thể đánh giá và bình luận về phòng mà họ đã sử dụng.
9	RU-UC14	Sửa thông tin tài khoản	Người dùng có thể sửa thông tin tài khoản như tên hiển thị, email.
10	RU-UC15	Đổi mật khẩu	Người dùng đổi mật khẩu của tài khoản.

Bảng 3.8 Bảng danh sách Use Case cho Quản trị viên khách sạn (HA)

STT	Mã Use Case	Tên Use Case	Mô tả
1	HA-UC17	Thêm phòng	Quản trị viên khách sạn có thể thêm phòng và các thông tin liên quan vào danh sách phòng do khách sạn quản lý.
2	HA-UC18	Xóa phòng	Quản trị viên khách sạn có thể xóa phòng và các thông tin liên

			quan trong danh sách phòng do khách sạn quản lý.
3	HA-UC19	Sửa thông tin phòng	Quản trị viên khách sạn có thể sửa các thông tin liên quan đến phòng trong danh sách phòng do khách sạn quản lý.
4	HA-UC20	Xem tình trạng Đặt phòng	Quản trị viên khách sạn có thể xem trạng thái Đặt phòng của các phòng khách sạn.
5	HA-UC21	Cập nhật tình trạng phòng	Quản trị viên khách sạn có thể cập nhật tình trạng phòng, tình trạng thanh toán và các thông tin liên quan được đề cập trong đơn hàng.
6	HA-UC22	Xem bình luận	Quản trị viên khách sạn có thể xem bình luận từ những người sử dụng dịch vụ phòng.
7	HA-UC23	Trả lời bình luận	Quản trị viên khách sạn có thể trả lời bình luận từ những người sử dụng dịch vụ phòng.
8	HA-UC24	Xem báo cáo và thống kê doanh thu	Quản trị viên khách sạn có thể xem báo cáo và thống kê doanh thu của khách sạn.

3.3.5. Đặc tả Use Case

Bảng 3.9 Đặc tả Use Case “Đăng ký”

Use Case ID	GS-UC01
Use Case Name	Đăng ký
Description	Cho phép người dùng chưa có tài khoản có thể đăng ký tài khoản để đăng nhập vào hệ thống.
Actor(s)	Người dùng Khách (GS)
Priority	Cao
Trigger	Người dùng muốn đăng ký tài khoản trên website Hotelligence
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet khi thực hiện đăng ký. Người dùng mở được trang web Hotelligence.
Post-Condition(s)	Người dùng đăng ký tài khoản thành công Hệ thống ghi nhận hoạt động của người dùng vào Activity Log.
Basic Flow	<ol style="list-style-type: none">1. Người dùng truy cập vào website Hotelligence.2. Người dùng chọn nút “Đăng ký” trên màn hình hiển thị.3. Website hiển thị trang “Đăng ký” kèm biểu mẫu theo quy định.4. Người dùng nhập các trường thông tin vào biểu mẫu.5. Người dùng xác nhận thông tin qua email.5. Người dùng đăng ký tài khoản thành công.6. Hệ thống ghi nhận hoạt động đăng ký thành công của người dùng vào Activity Log.

Alternate Flow	4a. Người dùng chọn phương thức đăng ký bằng Gmail. Use case tiếp tục bước 5.
Exception Flow	5a. Người dùng không xác nhận thông qua email. 6. Hệ thống không thay đổi.
Business Rules	BR01, BR02
Non-Functional Requirements	NFR01-UC01, NRF02, NRF03, NRF04

Bảng 3.10 Đặc tả Use Case “Tìm kiếm phòng”

Use Case ID	GS-UC02
Use Case Name	Tìm kiếm phòng
Description	Cho phép người dùng chưa có tài khoản hoặc đã có tài khoản có thể tìm kiếm thông tin phòng khách sạn.
Actor(s)	Người dùng Khách (GS) Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng tìm kiếm thông tin phòng khách sạn trên website Hotelligence
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet khi thực hiện đăng ký. Người dùng mở được trang web Hotelligence.
Post-Condition(s)	Trang web hiển thị các kết quả liên quan đến từ khóa tìm kiếm của người dùng
Basic Flow	1. Người dùng truy cập vào website Hotelligence. 2. Người dùng bấm vào thanh tìm kiếm trên màn hình hiển thị. 3. Người dùng nhập từ khóa tìm kiếm.

	4. Người dùng bấm nút “Tìm kiếm”
Alternate Flow	3a. Người dùng bấm nút tìm kiếm khi chưa nhập từ khóa tìm kiếm, trang web hiển thị thông báo lỗi dưới thanh tìm kiếm.
Exception Flow	None.
Business Rules	None.
Non-Functional Requirements	NRF02, NRF03, NRF04

Bảng 3.11 Đặc tả Use Case “Sắp xếp danh sách Phòng”

Use Case ID	GS-UC03
Use Case Name	Sắp xếp danh sách Phòng
Description	Cho phép người dùng có thể sắp xếp danh sách phòng khách sạn theo các tiêu chí có sẵn.
Actor(s)	Người dùng Khách (GS) Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng muốn sắp xếp danh sách phòng khách sạn.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng mở được trang web Hotelligence. Người dùng đang ở trang “Kết quả tìm kiếm”.
Post-Condition(s)	Hệ thống hiển thị danh sách các phòng đã được sắp xếp theo tiêu chí.
Basic Flow	1. Người dùng bấm chọn nút “Sắp xếp theo” hiển thị trên màn hình. 2. Người dùng chọn một trong các tiêu chí sắp xếp.

	3. Website hiển thị danh sách các phòng được sắp xếp theo tiêu chí mà người dùng chọn.
Alternate Flow	Không có.
Exception Flow	Không có.
Business Rules	BR05
Non-Functional Requirements	NFR06-UC03, NRF02, NRF03, NRF04

Bảng 3.12 Đặc tả Use Case “Lọc Phòng”

Use Case ID	GS-UC04
Use Case Name	Lọc Phòng
Description	Cho phép người dùng có thể lọc phòng khách sạn theo các tiêu chí có sẵn.
Actor(s)	Người dùng Khách (GS) Người dùng có tài khoản (RU)
Priority	Trung bình.
Trigger	Người dùng muốn lọc phòng khách sạn.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng mở được trang web Hotelligence. Người dùng đang ở trang “Kết quả tìm kiếm”.
Post-Condition(s)	Hệ thống hiển thị danh sách các phòng đã được lọc theo tiêu chí.
Basic Flow	1. Người dùng bấm chọn các ô tiêu chí lọc hiển thị trên màn hình. 2. Website hiển thị danh sách các phòng được lọc theo tiêu chí mà người dùng chọn.
Alternate Flow	Không có.
Exception Flow	Không có.

Business Rules	BR06
Non-Functional Requirements	NFR06-UC04, NRF02, NRF03, NRF04

Bảng 3.13 Đặc tả Use Case “Xem thông tin Phòng”

Use Case ID	GS-UC05
Use Case Name	Xem thông tin Phòng
Description	Cho phép người dùng có thể xem thông tin phòng khách sạn theo các tiêu chí có sẵn.
Actor(s)	Người dùng Khách (GS) Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng muốn xem thông tin phòng khách sạn.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng mở được trang web Hotelligence. Người dùng đang ở trang “Kết quả tìm kiếm”.
Post-Condition(s)	Hệ thống hiển thị thông tin phòng mà người dùng muốn xem.
Basic Flow	1. Người dùng bấm chọn một trong các phòng khách sạn đang hiển thị trên màn hình. 2. Website hiển thị thông tin chi tiết của phòng mà người dùng chọn.
Alternate Flow	Không có.
Exception Flow	Không có.
Business Rules	BR07
Non-Functional Requirements	NRF02, NRF03, NRF04

Bảng 3.14 Đặc tả Use Case “Đăng nhập”

Use Case ID	GS-UC06
Use Case Name	Đăng nhập
Description	Cho phép người dùng có thể đăng nhập vào website Hotelligence.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng muốn đăng nhập vào website.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng mở được trang web Hotelligence.
Post-Condition(s)	Người dùng có thể đăng nhập vào website.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng bấm nút “Đăng nhập” hiển thị trên màn hình. 2. Hệ thống chuyển tới trang “Đăng nhập” và hiển thị biểu mẫu đăng nhập. 3. Người dùng nhập email và bấm “Tiếp tục”. 4. Hệ thống gửi mail xác nhận. 5. Người dùng xác thực qua email. 6. Người dùng đăng nhập thành công. 7. Hệ thống ghi lại hoạt động của người dùng vào activity log.
Alternate Flow	<ol style="list-style-type: none"> 3a. Người dùng chọn phương thức đăng nhập thông qua Gmail. 4. Hệ thống chuyển tiếp đến hệ thống xác thực của Google. 5. Người dùng chọn tài khoản Google muốn đăng nhập. <p>Tiếp tục bước 6 ở Basic Flow.</p>

Exception Flow	3b. Người dùng chọn phương thức đăng nhập thông qua Gmail nhưng không tiếp tục. 4. Hệ thống giữ nguyên.
Business Rules	BR08
Non-Functional Requirements	NFR01-UC06, NRF06-UC06, NFR07-UC06, NFR06-UC04, NRF02, NRF03, NRF04

Bảng 3.15 Đặc tả Use Case “Đăng xuất”

Use Case ID	RU-UC07
Use Case Name	Đăng xuất
Description	Cho phép người dùng có thể đăng xuất khỏi website Hotelligence.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng muốn đăng xuất khỏi website.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng mở được trang web Hotelligence. Người dùng đã đăng nhập vào website.
Post-Condition(s)	Người dùng có thể đăng xuất ra khỏi website.
Basic Flow	1. Người dùng bấm vào tên tài khoản hiển thị ở góc phải. 2. Website hiển thị các lựa chọn. 3. Người dùng chọn “Đăng xuất”.
Alternate Flow	Không có.
Exception Flow	Không có.
Business Rules	Không có.
Non-Functional Requirements	Không có.

Bảng 3.16 Đặc tả Use Case “Quên mật khẩu”

Use Case ID	RU-UC08
Use Case Name	Quên mật khẩu
Description	Cho phép người dùng có thể tạo mới mật khẩu đã quên.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng đã quên mật khẩu của tài khoản và muốn đăng nhập vào website.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng mở được trang web Hotelligence.
Post-Condition(s)	Người dùng có thể tạo mới mật khẩu để đăng nhập vào website.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng bấm vào nút “Đăng nhập” hiển thị trên màn hình. 2. Hệ thống chuyển tới trang “Đăng nhập”. 3. Người dùng chọn “Quên mật khẩu”. 4. Người dùng nhập email đã quên mật khẩu và bấm “Tiếp tục” 5. Hệ thống gửi mail xác thực. 6. Người dùng xác thực và tạo mật khẩu mới.
Alternate Flow	Không có.
Exception Flow	Không có.
Business Rules	BR01, BR02
Non-Functional Requirements	NFR01-UC08, , NRF03, NRF04

Bảng 3.17 Đặc tả Use Case “Xem lịch sử đặt phòng”

Use Case ID	RU-UC09
Use Case Name	Xem lịch sử đặt phòng
Description	Cho phép người dùng xem lại thông tin về các đặt phòng đã thực hiện trên hệ thống.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Trung bình
Trigger	Người dùng muốn xem lịch sử đặt phòng của mình.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng mở được trang web Hotelligence. Người dùng đã đăng nhập vào hệ thống.
Post-Condition(s)	Hệ thống hiển thị danh sách các đặt phòng đã thực hiện của khách hàng. Khách hàng có thể xem thông tin chi tiết đặt phòng như mã đặt phòng, ngày đặt phòng, ngày nhận – trả phòng, tình trạng thanh toán.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng truy cập vào hệ thống Hotelligence. 2. Người dùng nhấn nút "Lịch sử đặt phòng" ở màn hình “Trang chủ”. 3. Hệ thống chuyển sang màn hình “Lịch sử đặt phòng”. 4. Hệ thống hiển thị danh sách các đặt phòng đã thực hiện của người dùng. 5. Các đặt phòng đã thực hiện được hiển thị các thông tin bao gồm: Mã đặt phòng Ngày đặt phòng Ngày nhận phòng

	Ngày trả phòng Loại phòng Số lượng phòng Giá phòng Trạng thái đặt phòng Thông tin thanh toán (tùy chọn)
Alternate Flow	N/A
Exception Flow	N/A
Business Rules	BR09
Non-Functional Requirements	NRF02, NRF03, NRF04

Bảng 3.18 Đặc tả Use Case “Đặt phòng”

Use Case ID	RU-UC10
Use Case Name	Đặt phòng
Description	Cho phép người dùng nhập thông tin đặt phòng và xác nhận đặt phòng để đặt phòng trên website Hotelligence.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng muốn đặt phòng khách sạn tại hệ thống.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng đã đăng nhập vào tài khoản Hotelligence.
Post-Condition(s)	Đơn đặt phòng được tạo thành công. Hệ thống hiển thị thông báo xác nhận đặt phòng và thông tin chi tiết về đơn đặt phòng.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng truy cập vào hệ thống “Hotelligence”. 2. Ở màn hình “Trang chủ”, người dùng chọn số ngày muốn đi, số lượng khách, số phòng, sau đó nhấn nút “Tìm”. 3. Hệ thống chuyển sang màn hình “Kết quả tìm kiếm”. 4. Hệ thống hiển thị ra danh sách các khách sạn tương ứng với nhu cầu của người dùng. 5. Người dùng chọn một “Khách sạn”. 6. Hệ thống chuyển sang màn hình “Thông tin khách sạn” 7. Người dùng nhấn nút “Đặt” ở một phòng”.

	<p>8. Hệ thống chuyển sang màn hình “Thông tin đặt phòng”.</p> <p>9. Người dùng điền vào các trường thông tin cần thiết như họ tên, email, SĐT, phương thức thanh toán.</p> <p>10. Sau khi hoàn thành nhập thông tin, người dùng nhấn vào nút "Đặt phòng".</p> <p>11. Hệ thống chuyển sang màn hình “Xác nhận đặt phòng”.</p> <p>12. Hệ thống gửi mã xác nhận gồm 6 chữ số đến email để người dùng nhập vào.</p> <p>13. Người dùng nhập mã xác nhận trùng khớp.</p> <p>14. Người dùng nhấn nút “Tiếp tục”.</p> <p>15. Hệ thống chuyển sang màn hình “Tình trạng đặt phòng”.</p> <p>16. Hệ thống ghi nhận đặt phòng thành công và hiển thị thông báo.</p>
Alternate Flow	N/A
Exception Flow	<p>13a. Người dùng nhập sai mã xác nhận.</p> <p>14a. Người dùng nhấn nút “Tiếp tục”.</p> <p>14a1. Hệ thống hiển thị thông báo sai mã xác nhận và hủy bỏ lệnh đặt phòng.</p> <p>Use case dừng lại.</p>
Business Rules	BR09, BR12, BR13, BR14, BR15
Non-Functional Requirements	NFR01-UC11, NRF02, NRF03, NRF04

Bảng 3.19 Đặc tả Use Case “Hủy đặt phòng”

Use Case ID	RU-UC11
Use Case Name	Hủy đặt phòng
Description	Cho phép người dùng hủy đặt phòng đã thực hiện nhưng chưa thanh toán.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng muốn hủy đặt phòng đã đặt trước đó.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng đã đăng nhập vào tài khoản Hotelligence. Người dùng đã thực hiện đặt phòng và chưa thanh toán.
Post-Condition(s)	Phòng đã đặt của người dùng được hủy thành công. Hệ thống hiển thị thông báo xác nhận hủy đặt phòng. Người dùng nhận được email xác nhận hủy đặt phòng (tùy chọn).
Basic Flow	1. Người dùng truy cập vào hệ thống Hotelligence. 2. Người dùng nhấn nút “Lịch sử đặt phòng” ở màn hình “Trang chủ”. 3. Người dùng nhấn nút “Hủy đặt phòng” ở một phòng. 4. Hệ thống hiển thị thông báo “Xác nhận hủy phòng” và người dùng nhấn nút “Đồng ý”. 5. Hệ thống ghi nhận hủy đặt phòng thành công và hiển thị thông báo.
Alternate Flow	N/A
Exception Flow	3a. Chưa có đặt phòng được thực hiện từ người dùng.

	Use case dừng lại. 3b. Tất cả đặt phòng đã được thanh toán. Use case dừng lại
Business Rules	BR16, BR17
Non-Functional Requirements	NRF02, NRF03, NRF04

Bảng 3.20 Đặc tả Use Case “Thanh toán trực tuyến”

Use Case ID	RU-UC12
Use Case Name	Thanh toán trực tuyến
Description	Cho phép người dùng thanh toán tiền phòng trực tuyến qua các cổng thanh toán hoặc thẻ ngân hàng.
Actor(s)	Người dùng có tài khoản (RU) Hệ thống thanh toán trực tuyến bằng Ví điện tử
Priority	Cao
Trigger	Người dùng muốn thanh toán trực tuyến sau khi đặt phòng.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng đã đăng nhập vào hệ thống Hotelligence. Người dùng có đơn đặt phòng chưa thanh toán.
Post-Condition(s)	Đơn đặt phòng được thanh toán thành công. Người dùng nhận được thông báo xác nhận thanh toán từ ví trực tuyến đã lựa chọn.
Basic Flow	1. Người dùng truy cập hệ thống Hotelligence. 2. Người dùng thực hiện đặt phòng và xác nhận đặt phòng thành công. 3. Hệ thống chuyển sang màn hình “Thanh toán trực tuyến”.

	<p>4. Người dùng nhấn nút “Thẻ tín dụng” để áp dụng phương thức thanh toán bằng thẻ tín dụng.</p> <p>5. Hệ thống chuyển sang màn hình “Xác nhận thanh toán bằng thẻ tín dụng”.</p> <p>6. Người dùng nhấn nút “Đồng ý” để xác nhận thanh toán.</p> <p>7. Hệ thống chuyển sang màn hình “Tình trạng thanh toán”.</p> <p>9. Hệ thống ghi nhận thanh toán thành công và hiển thị thông báo.</p>
Alternate Flow	<p>4a. Người dùng nhấn nút “Ví điện tử” để áp dụng phương thức thanh toán bằng ví điện tử.</p> <p>5a. Hệ thống chuyển sang màn hình “Xác nhận thanh toán bằng ví điện tử”.</p> <p>6a. Người dùng nhấn nút “ Ví VNPay” để áp dụng thanh toán bằng ví VNPay.</p> <p>6a1. Hệ thống chuyển sang màn hình “Chờ tiến trình thanh toán bằng ví điện tử”.</p> <p>Use case tiếp tục bước 7.</p> <p>4b. Người dùng nhấn nút “Ví điện tử” để áp dụng phương thức thanh toán bằng ví điện tử.</p> <p>5b. Hệ thống chuyển sang màn hình “Xác nhận thanh toán bằng ví điện tử”.</p> <p>6b. Người dùng nhấn nút “ Ví Momo” để áp dụng thanh toán bằng ví VNPay.</p> <p>6b1. Hệ thống chuyển sang màn hình “Chờ tiến trình thanh toán bằng ví điện tử”.</p> <p>Use case tiếp tục bước 7.</p>

Exception Flow	<p>4c. Người dùng nhấn nút “Quay về trang chủ” để hủy thanh toán trực tuyến.</p> <p>4c1. Hệ thống chuyển sang màn hình “Trang chủ”.</p> <p>Use case dừng lại.</p> <p>6d. Người dùng nhấn nút “Quay về trang chủ” để hủy thanh toán trực tuyến.</p> <p>6d1. Hệ thống chuyển sang màn hình “Trang chủ”.</p> <p>Use case dừng lại.</p>
Business Rules	N/A
Non-Functional Requirements	NFR01-UC13, NFR07-UC13, NRF02, NRF03, NRF04

Bảng 3.21 Đặc tả Use Case “Đánh giá Phòng”

Use Case ID	RU-UC13
Use Case Name	Đánh giá Phòng
Description	Cho phép người dùng đã lưu trú tại khách sạn đánh giá và bình luận về phòng mà họ đã sử dụng.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Trung bình
Trigger	Người dùng muốn đánh giá về phòng mà họ đã sử dụng.
Pre-Condition(s)	<p>Thiết bị của người dùng đã kết nối internet.</p> <p>Người dùng đã đăng nhập vào hệ thống Hotelligence.</p> <p>Người dùng đã từng sử dụng phòng khách sạn tại hệ thống Hotelligence.</p>
Post-Condition(s)	Đánh giá và bình luận của người dùng được lưu trữ trong hệ thống.

	Người dùng có thể xem lại đánh giá của mình sau khi gửi đánh giá.
Basic Flow	<p>1. Người dùng nhấn nút “Đánh giá” ở 1 phòng đã ở trong màn hình “Lịch sử đặt phòng”.</p> <p>2. Hệ thống hiển thị modal “Đánh giá phòng”.</p> <p>3. Có 5 trường tiêu chí đánh giá bắt buộc người dùng đánh giá (thang điểm từ 1 – 10, mặc định là 10) bao gồm:</p> <p>Sạch sẽ</p> <p>Tiện nghi và dịch vụ</p> <p>Nhân viên</p> <p>Cơ sở vật chất</p> <p>Thân thiện với môi trường</p> <p>4. Người dùng nhấn nút “Gửi đánh giá”.</p> <p>5. Hệ thống ghi nhận đánh giá thành công, đóng modal và hiển thị thông báo.</p>
Alternate Flow	<p>3a. Người dùng đánh giá 5 tiêu chí bắt buộc như bước 3 ở trên.</p> <p>3a1. Người dùng viết bình luận.</p> <p>Use case tiếp tục bước 4.</p>
Exception Flow	<p>3b. Người dùng nhấn biểu tượng “Thoát”.</p> <p>3b1. Hệ thống đóng modal.</p> <p>Use case dừng lại.</p>
Business Rules	BR18, BR19, BR20, BR21
Non-Functional Requirements	NRF05-UC14, NRF02, NRF03, NRF04

Bảng 3.22 Đặc tả Use Case “Sửa thông tin tài khoản”

Use Case ID	RU-UC14
Use Case Name	Sửa thông tin tài khoản
Description	Cho phép người dùng sửa đổi các thông tin tài khoản như tên hiển thị và email.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Trung bình
Trigger	Người dùng muốn sửa đổi các thông tin tài khoản như tên hiển thị và email.
Pre-Condition(s)	Thiết bị của người dùng đã kết nối internet. Người dùng đã đăng nhập vào hệ thống Hotelligence.
Post-Condition(s)	Thông tin tài khoản của người dùng được cập nhật thành công. Hệ thống hiển thị thông báo xác nhận cập nhật.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng truy cập vào hệ thống “Hotelligence”. 2. Người dùng nhấn nút “Tài khoản” ở màn hình “Trang chủ”. 3. Ở form “Thông tin tài khoản”, người dùng nhập dữ liệu hợp lệ vào một hoặc tất cả trường thông tin. 4. Người dùng nhấn nút “Lưu”. 5. Hệ thống ghi nhận sửa thông tin tài khoản thành công và hiển thị thông báo.
Alternate Flow	N/A
Exception Flow	<ol style="list-style-type: none"> 3a. Thông tin người dùng nhập vào chưa hợp lệ. 4a. Người dùng nhấn nút “Lưu”. 4a1. Hệ thống hiển thị thông báo lỗi và hủy bỏ lệnh lưu thông tin. <p>Use case dừng lại.</p>

Business Rules	BR22, BR23
Non-Functional Requirements	NRF01-UC15, NRF02, NRF03, NRF04

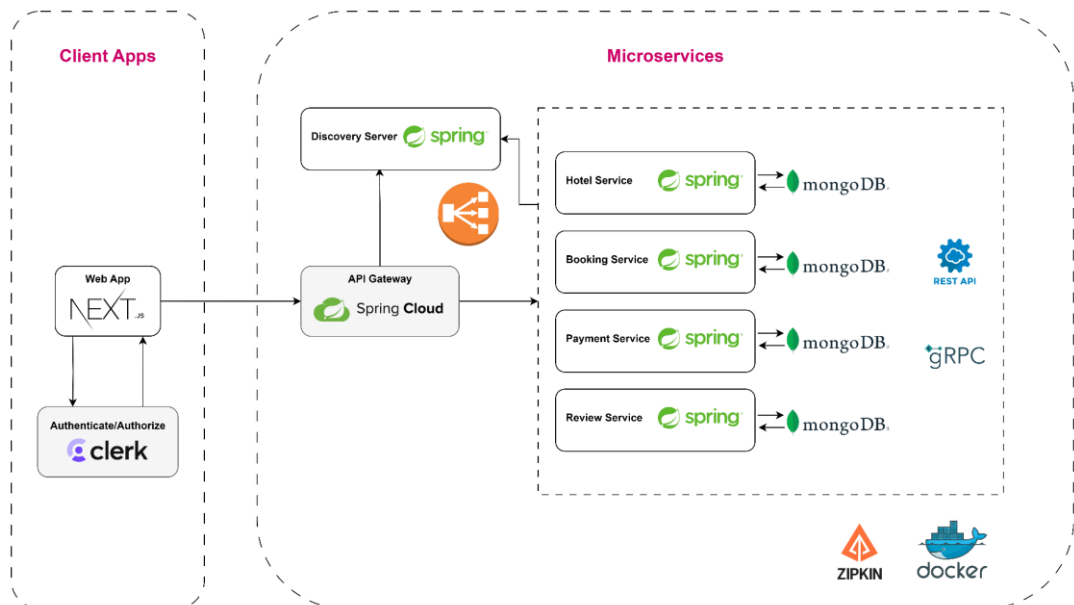
Bảng 3.23 Đặc tả Use Case “Đổi mật khẩu”

Use Case ID	RU-UC15
Use Case Name	Đổi mật khẩu
Description	Cho phép người dùng thay đổi mật khẩu cho tài khoản Hotelligence của họ.
Actor(s)	Người dùng có tài khoản (RU)
Priority	Cao
Trigger	Người dùng muốn đổi mật khẩu tài khoản.
Pre-Condition(s)	Người dùng đã đăng nhập vào hệ thống Hotelligence. Người dùng biết mật khẩu hiện tại của họ. Thiết bị của người dùng đã kết nối Internet.
Post-Condition(s)	Mật khẩu của người dùng được thay đổi thành công. Hệ thống hiển thị thông báo thay đổi mật khẩu thành công. Người dùng được đăng xuất khỏi tài khoản và phải đăng nhập lại với mật khẩu mới.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng đã truy cập vào hệ thống Hotelligence 2. Người dùng nhấn nút “Tài khoản” ở màn hình “Trang chủ”. 3. Ở form “Thay đổi mật khẩu”, người dùng nhập dữ liệu hợp lệ vào 3 trường: 4. Mật khẩu hiện tại 5. Mật khẩu mới

	6. Nhập lại mật khẩu mới 7. Người dùng nhấn nút “Lưu”. 8. Hệ thống ghi nhận thay đổi mật khẩu thành công và hiển thị thông báo.
Alternate Flow	N/A
Exception Flow	3a. Người dùng nhập dữ liệu không hợp lệ vào một hoặc nhiều trường. 4a. Người dùng nhấn nút “Lưu”. 4a1. Hệ thống hiển thị thông báo lỗi và hủy bỏ lệnh thay đổi mật khẩu. Use case dừng lại.
Business Rules	BR2, BR24
Non-Functional Requirements	NRF01-UC16, NRF07-UC16, NRF02, NRF03, NRF04

3.4. Thiết kế hệ thống

3.4.1. Kiến trúc hệ thống



Hình 3.2 Sơ đồ kiến trúc hệ thống

3.4.2. Mô tả các thành phần trong hệ thống

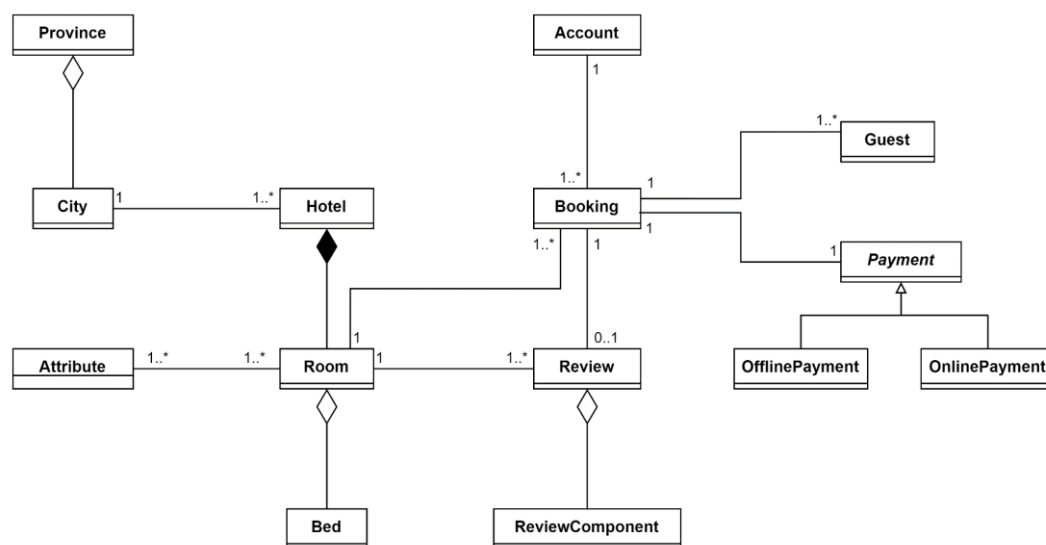
Bảng 3.24 Bảng mô tả chi tiết các thành phần của hệ thống

STT	Thành phần		Mô tả
1	Client Apps	Web App	Hiển thị giao diện người dùng, giúp người dùng tương tác với hệ thống thông qua website, ở đây sử dụng thư viện ReactJS.
		Authentication Authorization User Management	Tích hợp Clerk (clerk.com) hỗ trợ việc xác thực, phân quyền và quản lý người dùng.
2	Microservices	API Gateway	API Gateway đóng vai trò là điểm truy cập duy nhất cho các APIs, điều hướng các APIs đến điểm đích cụ thể.
		Discovery Server	Cho phép các dịch vụ có thể tìm thấy nhau thông qua API Gateway.
		Hotel Service	Cung cấp các dịch vụ liên quan đến khách sạn và phòng khách sạn.
		Booking Service	Cung cấp các dịch vụ liên quan đến đặt phòng.
		Payment Service	Cung cấp dịch vụ xử lý thanh toán.
		Review Service	Cung cấp dịch vụ về đánh giá và bình luận phòng.
		Database	Cơ sở dữ liệu của hệ thống, mỗi service sẽ có database riêng để

			phục vụ dịch vụ riêng lẻ. Sử dụng MongoDB cho mỗi service.
--	--	--	------------------------------------------------------------

3.5. Thiết kế hướng đối tượng

3.5.1. Sơ đồ lớp (Class Diagram)



Hình 3.3 Sơ đồ các lớp ở mức khái niệm

Bảng 3.25 Mô tả các thành phần trong Sơ đồ lớp

STT	Lớp	Mô tả
1	Province	Lớp “Tỉnh”, chứa tên các tỉnh thành ở Việt Nam.
2	City	Lớp “Thành Phố” chứa tên các thành phố ở Việt Nam.
3	Hotel	Lớp “Khách sạn” chứa các thông tin về khách sạn tọa lạc ở thành phố nào đó.
4	Room	Lớp “Phòng” chứa các thông tin về phòng thuộc một khách sạn nào đó.

5	Attribute	Lớp “Thuộc tính” chứa các thuộc tính của phòng khách sạn.
6	Bed	Lớp “Giường” chứa các thông tin về giường của phòng khách sạn.
7	Account	Lớp “Tài khoản” chứa thông tin về tài khoản của người dùng trong hệ thống.
8	Booking	Lớp “Đặt phòng” chứa thông tin đặt phòng của người dùng.
9	Review	Lớp “Đánh giá” chứa các thông tin về đánh giá (bình luận) về phòng.
10	ReviewComponent	Lớp “Thành phần Đánh giá” chứa các thành phần của tiêu chí đánh giá.
11	Guest	Lớp “Khách” chứa các thông tin về khách đặt phòng.
12	Payment	Lớp “Thanh toán” là lớp trừu tượng, cho phép các lớp liên quan đến thanh toán kế thừa.
13	OfflinePayment	Lớp “Thanh toán trực tiếp” chứa các thông tin về hình thức thanh toán trực tiếp.
14	OnlinePayment	Lớp “Thanh toán trực tuyến” chứa các thông tin về hình thức thanh toán trực tuyến.

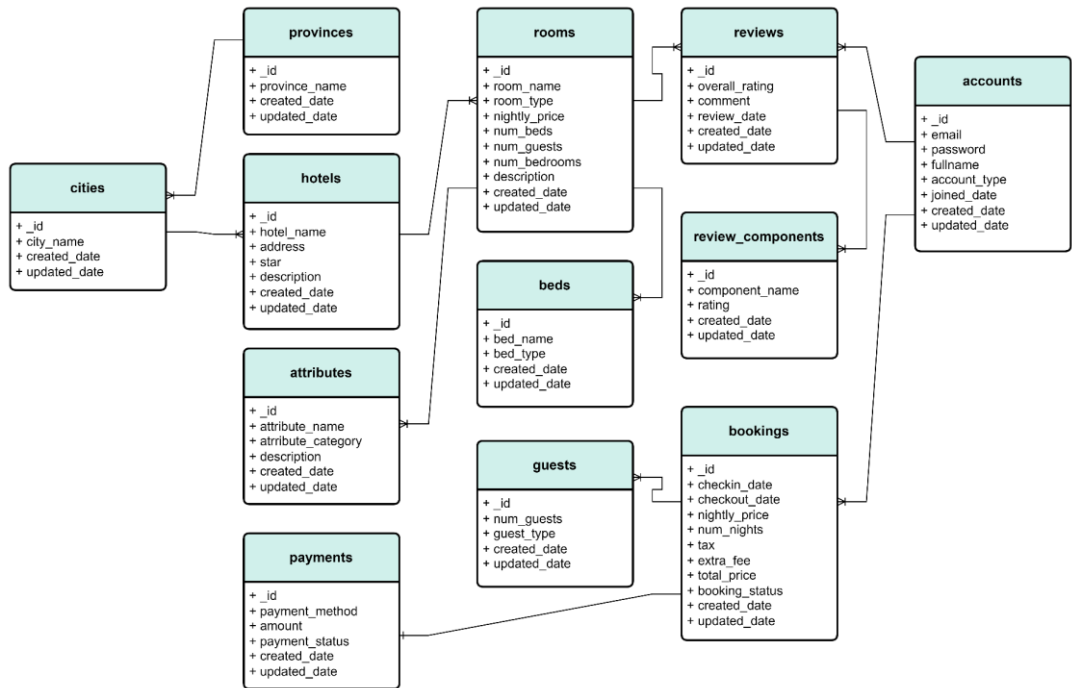
3.5.2. Sơ đồ tuần tự (Sequence Diagram)

Bảng 3.26 Danh sách các sơ đồ tuần tự

STT	Tên sơ đồ tuần tự	Hình sơ đồ
1	Đăng ký	Link
2	Đăng nhập	Link
3	Đăng xuất	Link
4	Quên mật khẩu	Link
5	Tìm kiếm khách sạn	Link
6	Sắp xếp Danh sách Khách sạn	Link
7	Lọc Danh sách Khách sạn	Link
8	Xem thông tin khách sạn	Link
9	Xem thông tin Phòng	Link
10	Xem thông tin Hỗ trợ	Link
11	Xem thông tin trang web	Link
12	Đặt phòng	Link
13	Hủy đặt phòng	Link
14	Thanh toán	Link
15	Đánh giá phòng	Link
16	Xem lịch sử đặt phòng	Link
17	Sửa thông tin tài khoản	Link
18	Đổi mật khẩu	Link

3.6. Thiết kế dữ liệu

3.6.1. Sơ đồ cơ sở dữ liệu (Database Diagram)



Hình 3.4 Sơ đồ cơ sở dữ liệu

3.6.2. Mô tả các Collection trong cơ sở dữ liệu

Vì cơ sở dữ liệu được lưu trữ dưới dạng Cơ sở dữ liệu không quan hệ (NoSQL), nên các Collection được hiểu tương đương với Table trong SQL. Trong Collection sẽ bao gồm các Documents (tương ứng với Row trong SQL).

3.6.2.1. Collection “provinces”

Bảng 3.27 Các trường trong collection provinces

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	province_name	string	Tên tỉnh
3	created_date	date	Ngày khởi tạo
4	updated_date	date	Ngày cập nhật

3.6.2.2. Collection “cities”

Bảng 3.28 Các trường trong collection “cities”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	city_name	string	Tên thành phố
3	created_date	date	Ngày khởi tạo
4	updated_date	date	Ngày cập nhật

3.6.2.3. Collection “hotels”

Bảng 3.29 Các trường trong collection “hotels”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	hotel_name	string	Tên khách sạn
3	address	string	Địa chỉ của khách sạn
4	star	int	Số sao của khách sạn
5	description	string	Mô tả khách sạn
6	created_date	date	Ngày khởi tạo
7	updated_date	date	Ngày cập nhật

3.6.2.4. Collection “rooms”

Bảng 3.30 Các trường trong collection “rooms”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	room_name	string	Tên phòng
3	room_type	string	Loại phòng
4	nightly_price	int	Giá phòng mỗi đêm
5	num_beds	int	Số lượng giường

6	num_guests	string	Số khách có thể chứa
7	num_bedrooms	int	Số phòng trống sẵn có
8	description	int	Mô tả phòng
9	created_date	date	Ngày khởi tạo
10	updated_date	date	Ngày cập nhật

3.6.2.5. Collection “attributes”

Bảng 3.31 Các trường trong collection “attributes”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	attribute_name	string	Tên thuộc tính
3	attribute_category	string	Loại thuộc tính
4	description	string	Mô tả thuộc tính
5	created_date	date	Ngày khởi tạo
6	updated_date	date	Ngày cập nhật

3.6.2.6. Collection “reviews”

Bảng 3.32 Các trường trong collection “reviews”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	overall_rating	float	Điểm đánh giá trung bình
3	comment	string	Bình luận
4	review_date	date	Ngày đánh giá
5	created_date	date	Ngày khởi tạo
6	updated_date	date	Ngày cập nhật

3.6.2.7. Collection “beds”

Bảng 3.33 Các trường trong collection “beds”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	bed_name	string	Tên giường
3	bed_type	string	Loại giường
4	created_date	date	Ngày khởi tạo
5	updated_date	date	Ngày cập nhật

3.6.2.8. Collection “accounts”

Bảng 3.34 Các trường trong collection “accounts”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	email	string	Địa chỉ email
3	password	string	Mật khẩu
4	fullname	string	Họ tên người dùng
5	account_type	string	Loại tài khoản
6	joined_date	date	Ngày tham gia
7	created_date	date	Ngày khởi tạo
8	updated_date	date	Ngày cập nhật

3.6.2.9. Collection “review_components”

Bảng 3.35 Các trường trong collection “review_components”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	component_name	string	Tên thành phần đánh giá
3	rating	int	Số điểm đánh giá
7	created_date	date	Ngày khởi tạo
8	updated_date	date	Ngày cập nhật

3.6.2.10. Collection “guests”

Bảng 3.36 Các trường trong collection “guests”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	Num_guests	Int	Số lượng khách hàng
3	Guest_type	String	Loại khách hàng
7	created_date	date	Ngày khởi tạo
8	updated_date	date	Ngày cập nhật

3.6.2.11. Collection “bookings”

Bảng 3.37 Các trường trong collection “bookings”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	checkin_date	date	Ngày nhận phòng
3	checkout_date	date	Ngày trả phòng
4	nightly_price	float	Giá mỗi đêm
5	num_nights	int	Số lượng đêm
6	tax	float	Thuế
7	extra_fee	float	Phí phát sinh
8	total_price	float	Tổng tiền
9	booking_status	string	Trạng thái đặt phòng
10	created_date	date	Ngày khởi tạo
11	updated_date	date	Ngày cập nhật

3.6.2.12. Collection “payments”

Bảng 3.38 Các trường trong collection “payments”

STT	Tên trường	Kiểu dữ liệu	Mô tả
1	_id	int	Mã document
2	payment_method	string	Phương thức thanh toán
3	amount	float	Số tiền thanh toán
4	payment_status	string	Tình trạng thanh toán
5	created_date	date	Ngày khởi tạo
6	updated_date	date	Ngày cập nhật

3.7. Thiết kế giao diện

3.7.1. Danh sách các màn hình

Bảng 3.39 Bảng danh sách các màn hình giao diện

STT	Tên màn hình	Tên Tiếng Việt	Mô tả	Giao diện
1	Home	Trang chủ	Cho phép người dùng tìm kiếm khách sạn với số ngày và số lượng khách. Ngoài ra còn hiển thị một số truy cập nhanh như các khách sạn có ưu đãi, v.v.	Link (Chưa đăng ký tài khoản) Link (Đã có tài khoản)
2	SignUp or Login	Đăng ký hoặc Đăng nhập	Cho phép người dùng đăng ký hoặc đăng nhập thông qua tài khoản Google hoặc qua Email.	Link
3	Login by OTP	Đăng nhập bằng OTP	Sau khi người dùng nhập Email đã có trong hệ	Link

			thống, hệ thống gửi mã OTP thông qua email. Màn hình cho phép người dùng nhập mã OTP đó.	
4	Login by Password	Đăng nhập bằng mật khẩu	Thay vì xác nhận qua Email, người dùng có thể sử dụng mật khẩu để đăng nhập.	Link
5	SignUp by OTP	Đăng ký bằng OTP	Nếu người dùng nhập Email chưa có trong hệ thống, hệ thống sẽ gửi mã OTP thông qua mail. Màn hình cho phép người dùng nhập mã OTP đó.	Link
6	Name Register	Tạo tên tài khoản	Người dùng tạo tên tài khoản cho tài khoản mới.	Link
7	Password Register	Tạo mật khẩu	Người dùng tạo mật khẩu cho tài khoản mới.	Link
8	ForgetPassword by OTP	Quên mật khẩu	Khi người dùng chọn quên mật khẩu, người dùng nhập Email và hệ thống gửi mã OTP thông qua Email đó. Màn hình cho phép người dùng nhập OTP.	Link
9	Create New Password	Tạo mật khẩu mới	Sau khi đã xác nhận quên mật khẩu qua OTP, người dùng tiến hành tạo lại mật khẩu mới.	Link

10	Search Result	Kết quả tìm kiếm	Hiển thị danh sách các phòng khách sạn khớp với tiêu chí tìm kiếm của người dùng.	Link
11	Hotel Details	Thông tin chi tiết Khách sạn	Hiển thị thông tin của khách sạn khi người dùng chọn một khách sạn bất kỳ.	Link
12	Room Details	Thông tin chi tiết Phòng	Hiển thị thông tin của phòng khi người dùng chọn một phòng trong khách sạn.	Link
13	Booking Details	Chi tiết đặt phòng	Hiển thị biểu mẫu phiếu đặt phòng để người dùng điền vào trước khi xác nhận đặt phòng.	Link
14	Booking Confirm	Xác nhận đặt phòng	Người dùng xác nhận đặt phòng thông qua mã OTP đã gửi qua Email.	Link
15	Booking Status	Tình trạng đặt phòng	Hiển thị tình trạng đặt phòng sau khi người dùng xác nhận đặt phòng.	Link
16	Pay Offline	Thanh toán trực tiếp	Nếu người dùng chọn hình thức “Thanh toán tại khách sạn” ở bước đặt phòng, màn hình hiển thị cho người dùng biết nếu người dùng chọn “Thanh toán”.	Link

17	Pay Online	Thanh toán trực tuyến	Nếu người dùng chọn hình thức “Thanh toán trực tuyến” ở bước đặt phòng, màn hình hiển thị cho người dùng biết nếu người dùng chọn “Thanh toán”. Cho phép người dùng chọn 1 trong 2 phương thức thanh toán bằng “Thẻ tín dụng” hoặc “Ví điện tử”.	Link
18	Pay via Card Confirm	Xác nhận thanh toán bằng thẻ tín dụng	Hỏi xem người dùng có chắc chắn muốn dùng thẻ tín dụng để thanh toán không.	Link
19	Pay via eWallet Confirm	Xác nhận thanh toán bằng ví điện tử	Hỏi xem người dùng muốn dùng ví điện tử nào để thanh toán.	Link
20	Pay via eWallet Processing	Xử lý thanh toán bằng ví điện tử	Quy trình thanh toán bằng ví điện tử đang được xử lý.	Link
21	Payment Status	Tình trạng thanh toán	Hiển thị tình trạng thanh toán.	Link
22	FAQ	Hỗ trợ - Câu hỏi thường gặp	Hiển thị các câu hỏi thường gặp nhằm hỗ trợ người dùng.	Link

23	Direct Contact	Hỗ trợ - Liên hệ trực tiếp	Hiển thị số đường dây nóng và email cho người dùng liên hệ.	Link
24	About Us	Thông tin trang web (“Về chúng tôi”)	Hiển thị thông tin về website Hotelligence.	Link
25	Booking History	Lịch sử đặt phòng	Hiển thị lịch sử đặt phòng cho người dùng.	Link
26	Cancel Booking Confirm	Xác nhận hủy đặt phòng	Hỏi xem người dùng có chắc chắn muốn hủy phòng không.	Link
27	Account	Tài khoản	Cho phép người dùng thay đổi tên hiển thị hoặc đổi mật khẩu.	Link
28	Terms of Use	Điều khoản sử dụng	Hiển thị điều khoản sử dụng cho người dùng.	Link
29	Account Menu	Danh mục tài khoản	Hiển thị các lựa chọn: “Tài khoản”, “Lịch sử đặt phòng” và “Đăng xuất”.	Link

3.7.2. Sơ đồ luồng màn hình (Screen Flow Diagram)



Hình 3.5 Sơ đồ luồng màn hình

Chương 4. XÂY DỰNG WEBSITE

4.1. Cài đặt

4.1.1. Môi trường và công cụ

Bảng 4.1 Môi trường và công cụ cần cài đặt

STT	Tên môi trường/công cụ	Link cài đặt	Yêu cầu phiên bản
1	NodeJs	Link	Node 18 trở lên
2	IntelliJ	Link	Community hoặc Ultimate
3	Docker Desktop	Link	
4	MongoDB Compass	Link	
5	Postman	Link	
6	Visual Studio Code	Link	

4.1.2. Hướng dẫn cài đặt

Có thể đọc hướng dẫn và cài đặt theo chỉ dẫn của các đường dẫn gắn ở mục **6.1.1**.

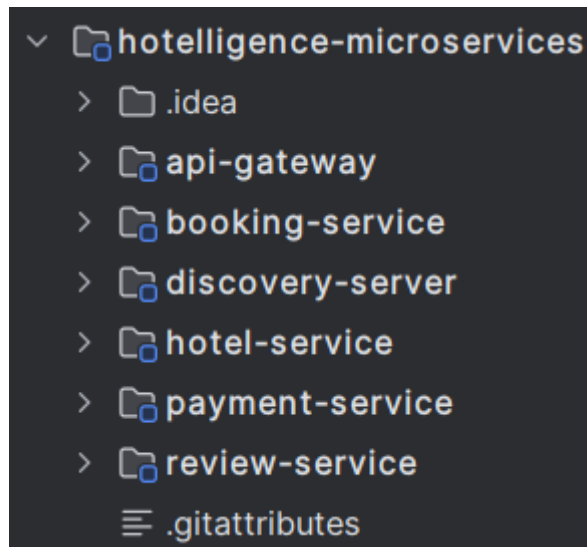
Link source code: <https://github.com/orgs/Hotelligence/repositories>

4.2. Thiết lập dự án

4.2.1. Thiết lập Discovery Server và API Gateway

Trước khi bắt đầu, chúng ta giả định rằng đã cài đặt các dependencies cho dự án Spring, ở đây chúng ta sẽ chỉ tiếp tục thiết lập Discovery Server bằng Eureka Netflix và API Gateway, sau đó quan sát các tình trạng hoạt động của các service để đảm bảo các service vẫn hoạt động bình thường.

Bước 1: Tạo 1 module mới tên là “discovery-server”, cấu hình file pom.xml và application.properties như hình dưới:



Hình 4.1 Tạo module discovery-server

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
</dependency>
```

Hình 4.2 Cấu hình file pom.xml của discovery-server

```
eureka.instance.hostname=localhost
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
server.port=8761

management.tracing.sampling.probability=1.0
```

Hình 4.3 Cấu hình file application.properties của discovery-server

Bước 2: Tạo module “api-gateway” tương tự như ở Bước 1, cấu hình ở pom.xml và application.properties như sau:


```

<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>

```

Hình 4.4 Cấu hình file pom.xml của api-gateway

```

1  spring.application.name=api-gateway
2  eureka.client.service-url.defaultZone=http://localhost:8761/eureka
3
4  logging.level.root=INFO
5  logging.level.org.springframework.cloud.gateway.route.RouteDefinitionRouteLocator=INFO
6  logging.level.org.springframework.cloud.gateway=TRACE
7
8  management.endpoints.web.exposure.include=health,info,env,beans
9  management.endpoint.health.show-details=always
10
11  ## Hotel Service Route
12  spring.cloud.gateway.routes[0].id=hotel-service
13  spring.cloud.gateway.routes[0].uri=lb://hotel-service
14  spring.cloud.gateway.routes[0].predicates[0]=Path=/api/hotels/**, /api/rooms/**
15
16  ## Booking Service Route
17  spring.cloud.gateway.routes[1].id=booking-service
18  spring.cloud.gateway.routes[1].uri=lb://booking-service
19  spring.cloud.gateway.routes[1].predicates[0]=Path=/api/bookings/**
20
21  ##Discovery Server Route
22  spring.cloud.gateway.routes[2].id=discovery-service
23  spring.cloud.gateway.routes[2].uri=http://localhost:8761
24  spring.cloud.gateway.routes[2].predicates[0]=Path=/eureka/web
25  spring.cloud.gateway.routes[2].filters[0]=SetPath=/
26
27  ##Discovery Server Static Resources Route
28  spring.cloud.gateway.routes[3].id=discovery-service-static
29  spring.cloud.gateway.routes[3].uri=http://localhost:8761
30  spring.cloud.gateway.routes[3].predicates[0]=Path=/eureka/**
31
32  ##Payment Service Route
33  spring.cloud.gateway.routes[4].id=payment-service
34  spring.cloud.gateway.routes[4].uri=lb://payment-service
35  spring.cloud.gateway.routes[4].predicates[0]=Path=/api/payments/**
36
37  ##Review Service Route
38  spring.cloud.gateway.routes[5].id=review-service
39  spring.cloud.gateway.routes[5].uri=lb://review-service
40  spring.cloud.gateway.routes[5].predicates[0]=Path=/api/reviews/**
41
42  management.tracing.sampling.probability=1.0

```

Hình 4.5 Cấu hình file application.properties của api-gateway

Bước 3: Cấu hình cho các service còn lại như “hotel-service”, “booking-service”, “payment-service” và “review-service”

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

Hình 4.6 Cấu hình file pom.xml cho các service là eureka-client

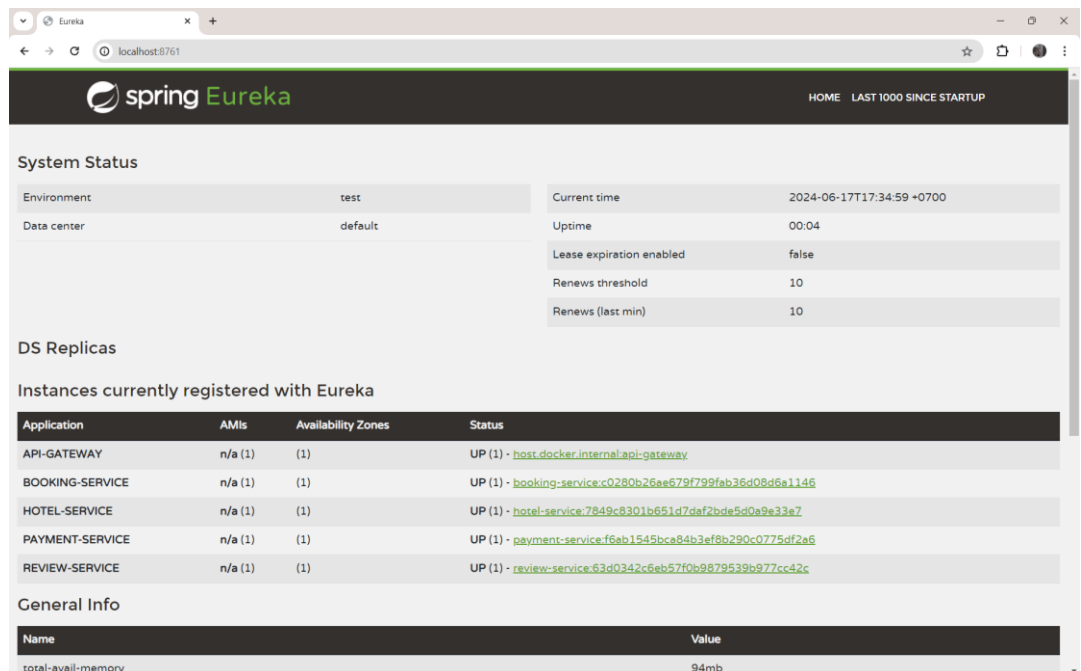
Ở file application.properties ở mỗi service, cấu hình như sau:

```
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka/
eureka.instance.instanceId=${spring.application.name}:${vcap.application.instance_id:${spring.application.instance_id:${random.value}}}
```

Hình 4.7 Cấu hình file application.properties cho các service

Bước 4: Mở trình duyệt trang <http://localhost:8761/>, ta dễ dàng thấy được tình trạng hoạt động của các service (UP: Đang hoạt động / DOWN: Không hoạt động).

Hình bên dưới cho ta thấy rằng các service đang hoạt động bình thường.



The screenshot shows the Spring Eureka dashboard in a web browser. The page has a dark header with the 'spring Eureka' logo and navigation links 'HOME' and 'LAST 1000 SINCE STARTUP'. The main content area is divided into several sections:

- System Status:** A table showing environment details.

Environment	test	Current time	2024-06-17T17:34:59 +0700
Data center	default	Uptime	00:04
		Lease expiration enabled	false
		Renews threshold	10
		Renews (last min)	10
- DS Replicas:** A section indicating the number of data store replicas.
- Instances currently registered with Eureka:** A table listing the services and their status.

Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1) - host-docker.internal-api-gateway
BOOKING-SERVICE	n/a (1)	(1)	UP (1) - booking-service:c0280b26ae679f799fab36d08d6a1146
HOTEL-SERVICE	n/a (1)	(1)	UP (1) - hotel-service:7849c8301b651d7daf2bde5d0a9e33e7
PAYMENT-SERVICE	n/a (1)	(1)	UP (1) - payment-service:f6ab1545bca84b3ef8b290c0775df2a6
REVIEW-SERVICE	n/a (1)	(1)	UP (1) - review-service:63d0342c6eb57f0b9879538b977cc42c
- General Info:** A table showing general system information.

Name	Value
total-avail-memory	94mb

Hình 4.8 Giao diện trang Spring Eureka

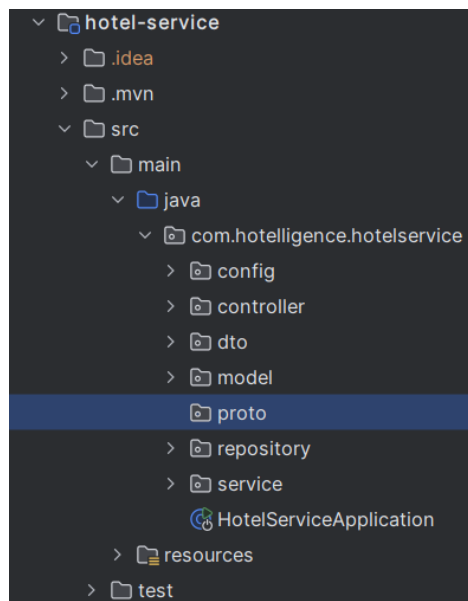
4.2.2. Thiết lập gRPC

Bước 1: Đầu tiên, chúng ta cài đặt các dependencies của gRPC cho hotel-service.

```
<dependency>
  <groupId>io.grpc</groupId>
  <artifactId>grpc-netty</artifactId>
</dependency>
<dependency>
  <groupId>io.grpc</groupId>
  <artifactId>grpc-protobuf</artifactId>
</dependency>
<dependency>
  <groupId>io.grpc</groupId>
  <artifactId>grpc-services</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>io.grpc</groupId>
  <artifactId>grpc-stub</artifactId>
</dependency>
<dependency>
  <groupId>io.zipkin.brave</groupId>
  <artifactId>brave</artifactId>
  <version>6.0.3</version>
</dependency>
```

Hình 4.9 Cấu hình các dependencies của gRPC cho hotel-service

Bước 2: Tạo package mới tên “**proto**” và tạo file **hotel.proto** trong đó:



Hình 4.10 Thư mục (package) proto



```

1  syntax = "proto3";
2
3  package com.hotelligence;
4
5  option java_multiple_files = true;
6  option java_package = "com.hotelligence";
7
8  service HotelService {
9      rpc GetHotelByID (HotelRequest) returns (HotelResponse);
10 }
11
12 message HotelRequest {
13     string id = 1;
14 }
15
16 message HotelResponse {
17     string id = 1;
18     string hotelName = 2;
19     string address = 3;
20     int32 star = 4;
21     string description = 5;
22     repeated string images = 6;
23     string city = 7;
24     string province = 8;
25     double ratingScore = 9;
26     string ratingCategory = 10;
27     int32 numOfReviews = 11;
28     double originPrice = 12;
29     double discount = 13;
30     double discountPrice = 14;
31     double taxPercentage = 15;
32     double taxPrice = 16;
33     double extraFee = 17;
34     double totalPrice = 18;
35 }
36

```

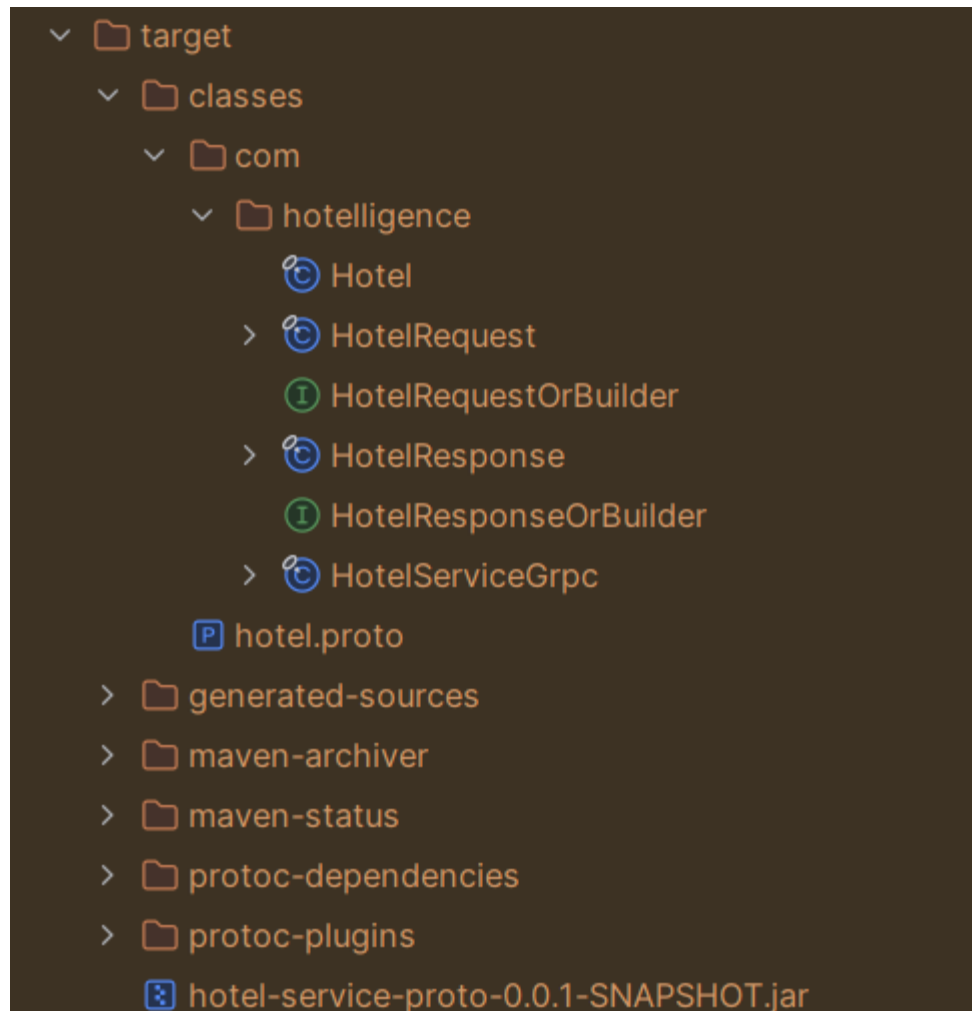
Hình 4.11 File *hotel.proto*

Có thể thấy, cú pháp của file proto khá đơn giản:

- **syntax = “proto3”**: cho biết ta sẽ sử dụng proto3 (nếu để mặc định là proto2)
- **package**: cho biết gói sẽ đặt vào

- **service:** Định nghĩa 1 service, trong đó bao gồm các hàm được đánh dấu bằng chữ “rpc” phía trước (ta có thể hiểu tương đương như class có các method bên trong)
- **message:** định nghĩa các fields cần gửi/nhận
*Ví dụ: **rpc GetHotelById (HotelRequest) returns (HotelResponse)** nghĩa là phương thức **rpc** tên **GetHotelById** có đầu vào là **HotelRequest** và đầu ra là **HotelResponse***

Bước 3: Build chương trình và ta sẽ thấy các files được protobuf sẽ tạo tự động:



Hình 4.12 Các files do protobuf tạo tự động khi build chương trình

Bước 4: Bây giờ ta chỉ cần tạo các class implements từ các class được tạo sẵn, và override các hàm cần thiết

Bước 5: Chúng ta chạy lại dự án Spring Boot và quan sát lại tình trạng hoạt động của service như ở 4.2.1.

4.2.3. Thiết lập Zipkin

Để có thể trực quan hóa (visualize) về thời gian thực thi của giao thức REST với gRPC, ta tiếp tục thiết lập cài đặt Zipkin để có thể quan sát.

Bước 1: Cài đặt Zipkin theo trang <https://zipkin.io/pages/quickstart.html> ở đây chúng ta sẽ pull Docker image của Zipkin xuống để đảm bảo Zipkin sẽ chạy được trong môi trường của Docker.

Bước 2: Thêm cấu hình của Zipkin vào các service ở file pom.xml và application.properties

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
  <groupId>io.micrometer</groupId>
  <artifactId>micrometer-tracing-bridge-brave</artifactId>
</dependency>
<dependency>
  <groupId>io.zipkin.reporter2</groupId>
  <artifactId>zipkin-reporter-brave</artifactId>
</dependency>
```




Hình 4.13 Cấu hình file pom.xml cho các service

```
management.endpoints.web.exposure.include=health,info,env,beans
management.endpoint.health.show-details=always

management.tracing.sampling.probability=1.0
```

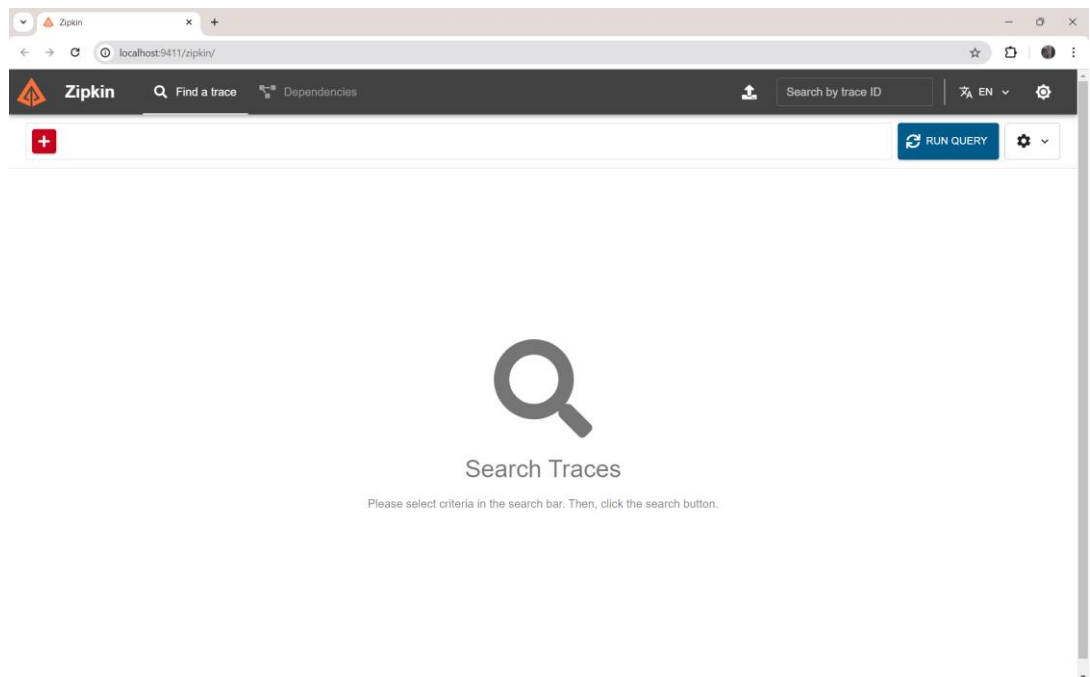
Hình 4.14 Cấu hình file application.properties cho các services

Bước 3: Khởi chạy Zipkin bằng Docker Desktop (ta có thể tải Docker Desktop tại <https://www.docker.com/products/docker-desktop/>)

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	nice_chaum 473f73658b26	openzipkin/zipkin	Running	N/A	9411:9411	4 seconds ago	  

Hình 4.15 Zipkin đang chạy trong Docker

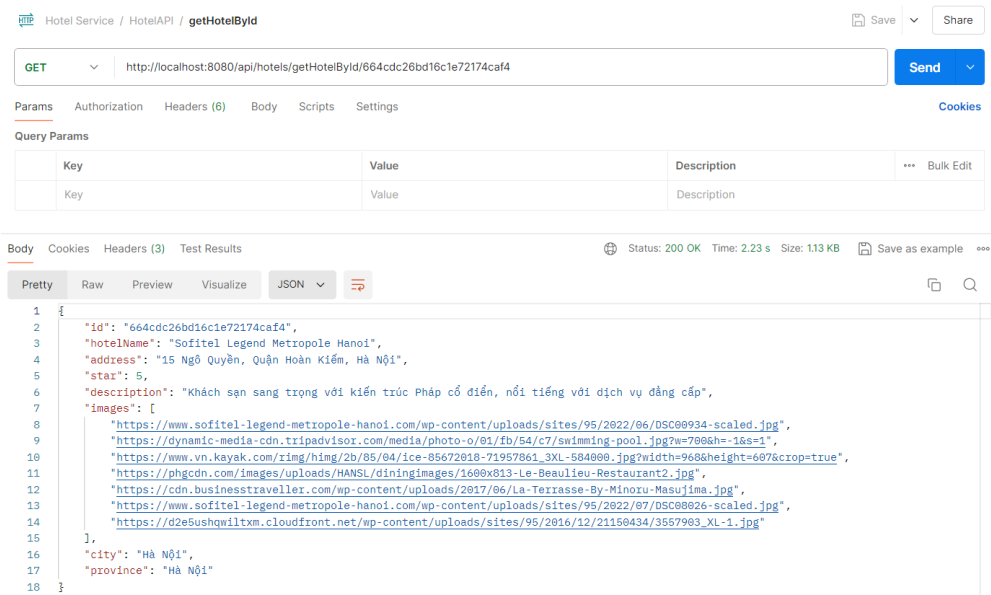
Bước 4: Mở trình duyệt trang <http://localhost:9411/zipkin/> và Zipkin sẽ hiển thị giao diện:



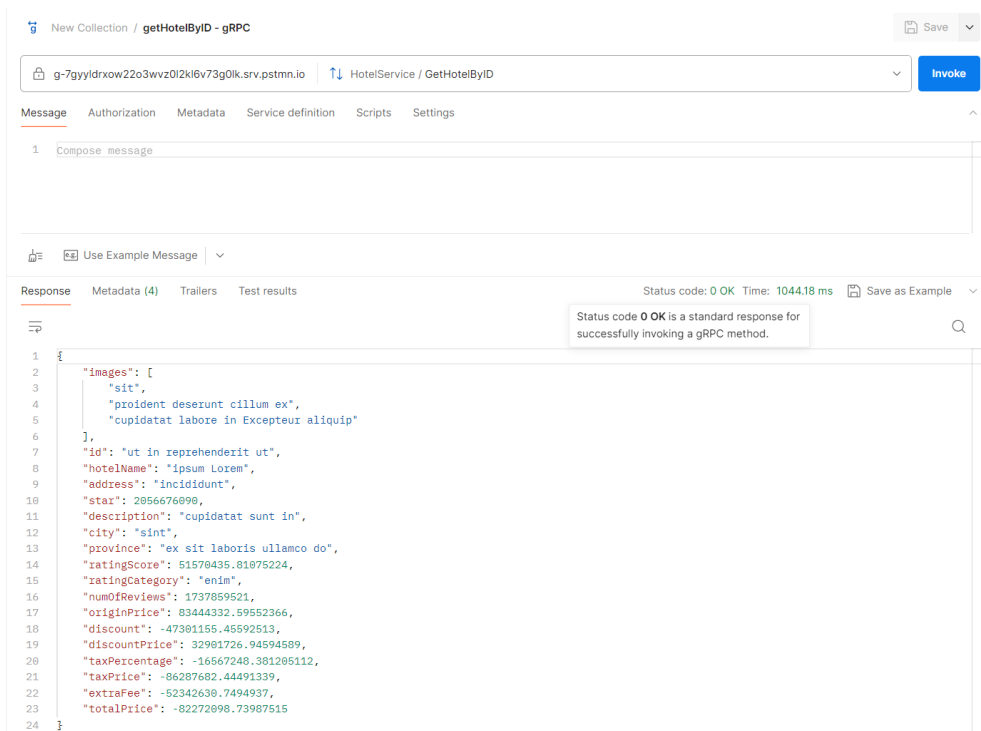
Hình 4.16 Giao diện Zipkin

4.2.4. Kết quả thực thi

Để thực hiện kiểm thử tốc độ truyền nhận của hai giao thức, ta sẽ thử gửi 2 request bằng Postman, 1 request bằng HTTP GET và 1 request bằng gRPC.



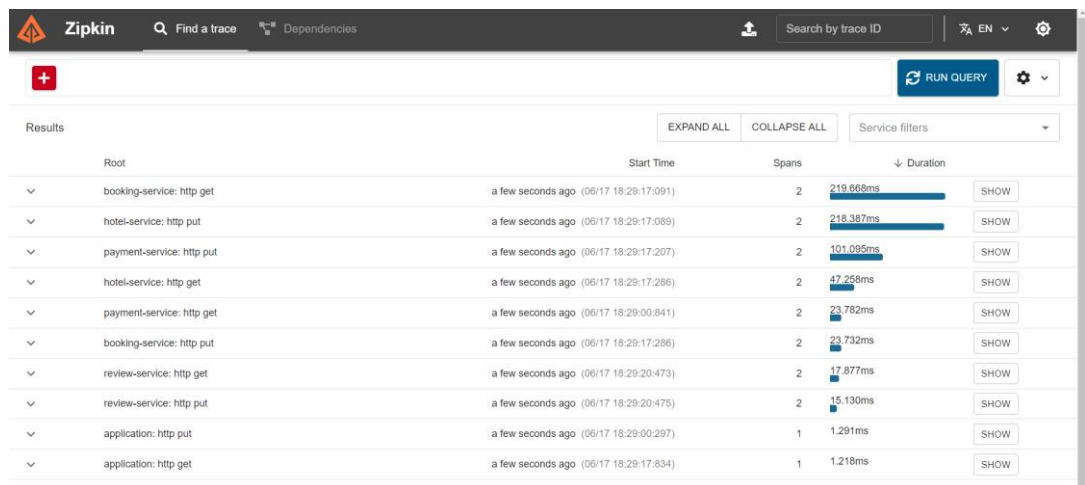
Hình 4.17 *getHotelById* bằng HTTP GET



Hình 4.18 *getHotelById* bằng gRPC

Với HTTP, 200 OK là tín hiệu cho ta biết request được gửi thành công, còn với gRPC thì 0 OK là tín hiệu cho việc gửi thành công.

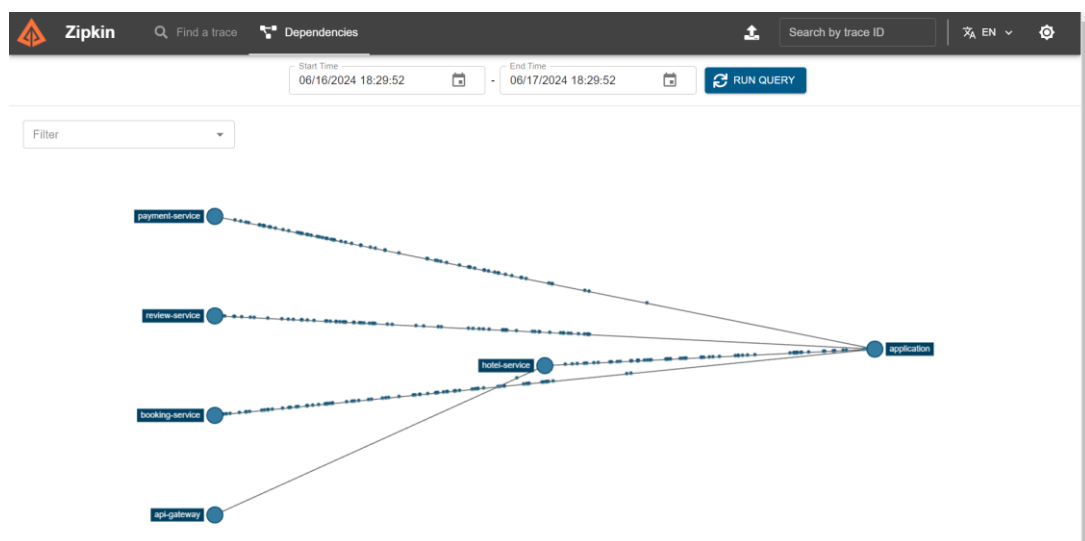
Đồng thời chúng ta cũng có thể xem được tổng thời gian thực thi của các phương thức khi được gửi bởi các service:



Root	Start Time	Spans	Duration
booking-service: http get	a few seconds ago (06/17 18:29:17:091)	2	219.668ms
hotel-service: http put	a few seconds ago (06/17 18:29:17:089)	2	218.387ms
payment-service: http put	a few seconds ago (06/17 18:29:17:207)	2	101.095ms
hotel-service: http get	a few seconds ago (06/17 18:29:17:286)	2	47.258ms
payment-service: http get	a few seconds ago (06/17 18:29:00:841)	2	23.782ms
booking-service: http put	a few seconds ago (06/17 18:29:17:286)	2	23.732ms
review-service: http get	a few seconds ago (06/17 18:29:20:473)	2	17.877ms
review-service: http put	a few seconds ago (06/17 18:29:20:475)	2	15.130ms
application: http put	a few seconds ago (06/17 18:29:00:297)	1	1.291ms
application: http get	a few seconds ago (06/17 18:29:17:834)	1	1.218ms

Hình 4.19 Tổng thời gian thực thi bởi các service

Và ta cũng có thể xem mật độ các yêu cầu/phản hồi tương tác với nhau thông qua tab Dependencies:



Hình 4.20 Truy vết các dịch vụ

Như ta có thể thấy, chỉ qua ví dụ trên, rõ ràng với **HTTP GET** ta nhận được kết quả thực thi là **2.23s** trong khi với **gRPC** chỉ với **1044.18ms**.

Kết quả cho thấy với gRPC thì ta tối ưu được tốc độ gửi đi tới **2.13564711 lần!**

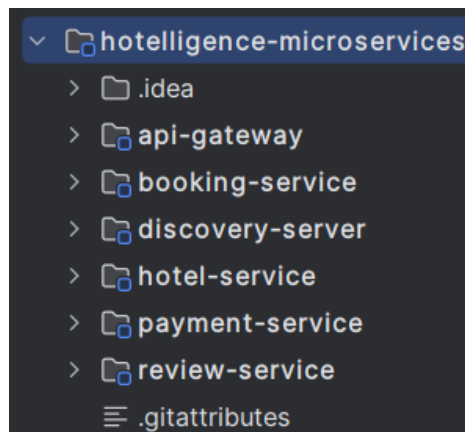
4.3. Triển khai

Hiện tại ứng dụng đang triển khai trên môi trường local nên người dùng cần cài đặt thủ công để có thể chạy được.

4.3.1. Back-end (Server)

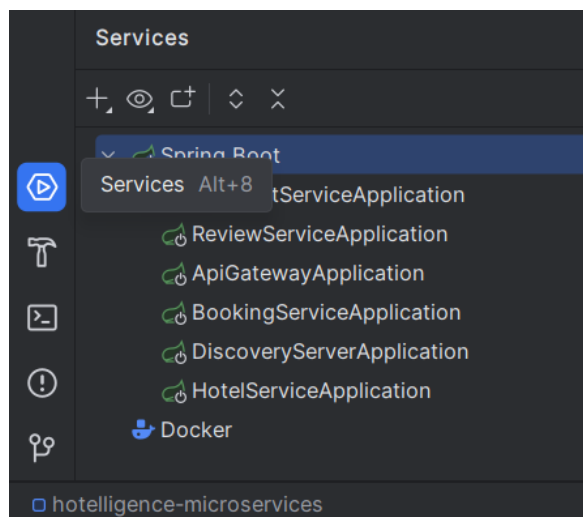
Bước 1: Mở thư mục “**hotelligence-microservices**” ở IntelliJ.

Ta có thể thấy cây thư mục như sau:



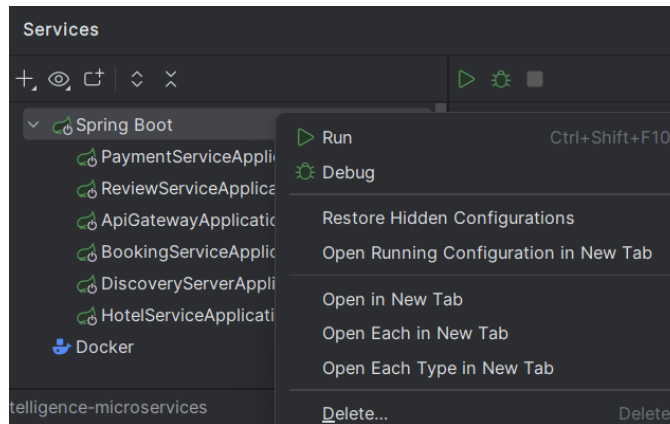
Hình 4.21 Cây thư mục của code back-end

Bước 2: Ở góc dưới màn hình, ta bấm vào biểu tượng **Services** như sau (hoặc dùng tổ hợp Alt + 8):



Hình 4.22 Biểu tượng Services

Bước 3: Nhấp chuột phải vào “**Spring Boot**” rồi nhấn “**Run**” để chạy chương trình server.



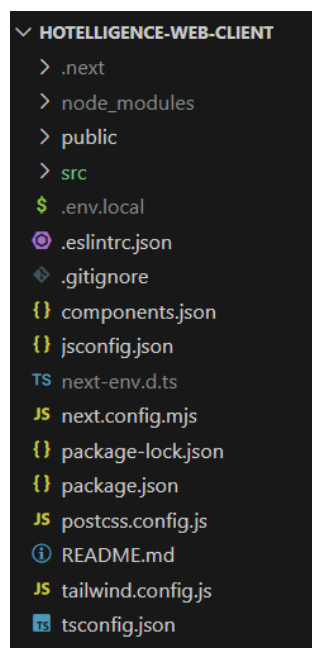
Hình 4.23 Khởi chạy chương trình Spring Boot

Bước 4: Chạy code front-end ở phần 6.2.2

4.3.2. Front-end (Client)

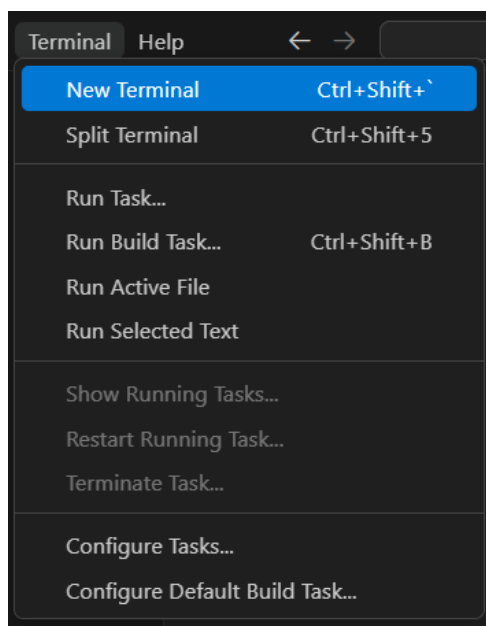
Bước 1: Mở thư mục “**hotelligence-web-client**” ở Visual Studio Code.

Ta có thể thấy cây thư mục như sau:



Hình 4.24 Cây thư mục của code front-end

Bước 2: Mở Terminal bằng tab trên cùng của ứng dụng, hoặc ấn tổ hợp phím **Ctrl + Shift + `**



Hình 4.25 Mở New Terminal

Bước 3: Gõ lệnh **npm run dev** và chờ trong giây lát, ta sẽ thấy kết quả như sau:

```
> hotellintelligence-web-client@0.1.0 dev
> next dev

▲ Next.js 14.1.4
- Local:      http://localhost:3000
- Environments: .env.local

✓ Ready in 2.7s
```

Hình 4.26 Client đang chạy

Bước 4: Mở trang **<http://localhost:3000>** và xem kết quả, trang web sẽ hiển thị.

4.4. Kiểm thử

4.4.1. Kiểm thử thủ công

Áp dụng phương pháp Kiểm thử hộp đen (Blackbox Testing) để kiểm tra kết quả của các components hiển thị trên trang web.

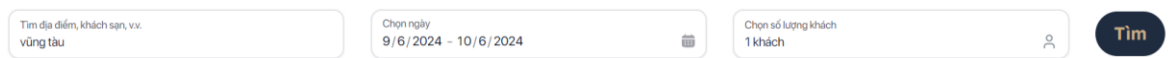
❖ Ví dụ 1: Trang chủ

Giả sử ta có 3 trường trên trang web như sau, trường hợp nếu các trường chưa nhập đầy đủ thì nút “**Tìm**” sẽ không thể bấm được.

Ở đây thì trường “**Chọn ngày**” và “**Chọn số lượng khách**” đã mặc định có giá trị, nên ta chỉ cần thêm trường “**Tìm địa điểm, khách sạn, v.v.**” có dữ liệu thì nút “**Tìm**” sẽ bấm được.



Hình 4.27 Trạng thái của các trường và nút khi chưa nhập gì



Hình 4.28 Trạng thái của các trường và nút khi đã nhập dữ liệu đầy đủ

❖ Ví dụ 2: Trang “Đặt phòng”

Ở trang đặt phòng ta có thể thấy các trường đều yêu cầu là bắt buộc (dấu * màu đỏ)

Chi tiết đặt phòng

Bước 1: Thông tin cá nhân

Vui lòng cho chúng tôi biết tên khách sẽ lưu trú tại phòng này chính xác như trên giấy tờ tùy thân sẽ sử dụng khi nhận phòng. Vui lòng nhập đầy đủ nếu khách mang họ kép (như Nguyễn Phước, Tôn Nữ, Lê Đoàn, v.v.).

Họ và tên *

Email *

Số điện thoại *

Bước 2: Kiểm tra thông tin phòng

◆ Classic Room

✓ Bao gồm bữa sáng cho 2 người

Bước 3: Phương thức thanh toán

☒ Thanh toán tại khách sạn

☐ Thanh toán trực tuyến

Chấp nhận thanh toán thông qua các thẻ tín dụng và ví điện tử hiện hành:



Để đảm bảo quyền lợi giữa Quý khách và khách sạn, chúng tôi yêu cầu thêm thông tin về thẻ tín dụng. Chúng tôi cam kết sử dụng phương thức truyền tải an toàn để bảo vệ thông tin cá nhân của Quý khách.

Số thẻ *

Ngày hết hạn *

Mã số CVV *

Bằng việc bấm "Đặt phòng", chúng tôi mặc định Quý khách xác nhận đã đọc và đồng ý Điều khoản & Điều kiện, Chính sách bảo mật và Hướng dẫn du lịch của chính phủ của Hotelligence.

Đặt phòng

Hình 4.29 Các trường và nút trên trang đặt phòng khi chưa nhập gì

Khi chưa nhập thông tin nhưng bấm nút “Đặt phòng”, các trường sẽ cảnh báo đỏ và yêu cầu nhập đầy đủ các trường bắt buộc:

Chi tiết đặt phòng

Bước 1: Thông tin cá nhân

Vui lòng cho chúng tôi biết tên khách sẽ lưu trú tại phòng này chính xác như trên giấy tờ tùy thân sẽ sử dụng khi nhận phòng. Vui lòng nhập đầy đủ nếu khách mang họ kép (như Nguyễn Phước, Tôn Nữ, Lê Đoàn, v.v.).

Họ và tên *

Please fill out this field.

Email *

Please fill out this field.

Số điện thoại *

Please fill out this field.

Hình 4.30 Các trường báo lỗi khi thông tin chưa được nhập

Chúng ta cũng làm tương tự cho các nút bấm và các trường ở các trang khác để đảm bảo tính đúng đắn của dữ liệu.

4.4.2. Kiểm thử tự động

Để kiểm thử tích hợp, chúng ta cần kiểm thử trên môi trường Docker, để nó độc lập với môi trường ứng dụng đang sử dụng. Ở đây chúng ta có Docker Image là mongoDB và Framework cho testing là JUnit5.

```
@SpringBootTest
@Testcontainers
@AutoConfigureMockMvc
class HotelServiceApplicationTests {

    @Container
    static MongoDBContainer mongoDBContainer = new MongoDBContainer("mongo:4.4.2");

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @Autowired
    private HotelRepository hotelRepository;

    @DynamicPropertySource
    static void setProperties(DynamicPropertyRegistry dynamicPropertyRegistry){
        dynamicPropertyRegistry.add("spring.data.mongodb.uri", mongoDBContainer::getReplicaSetUrl);
    }
}
```

Hình 4.31 Thiết lập module kiểm thử tích hợp

Tiếp theo chúng ta viết test cho hàm **shouldCreateHotel()**, nếu pass thì nó sẽ trả về 1 Hotel mới giả định được tạo ra trong database.

```
class HotelServiceApplicationTests {

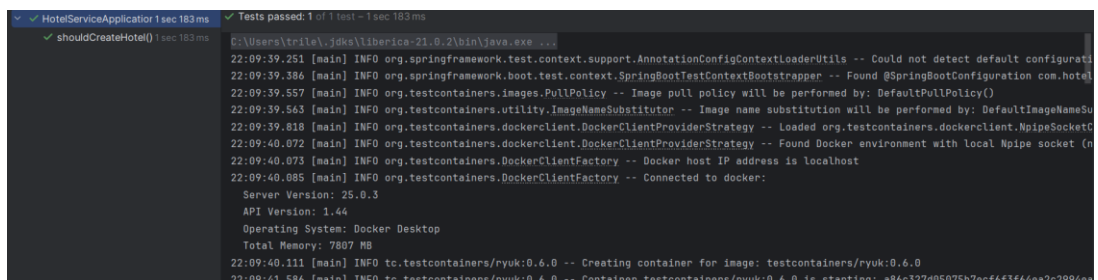
    @Test
    void shouldCreateHotel() throws Exception {
        HotelRequest hotelRequest = getHotelRequest();
        String hotelRequestString = objectMapper.writeValueAsString(hotelRequest);

        mockMvc.perform(MockMvcRequestBuilders.post( uriTemplate: "/api/hotels")
            .contentType(MediaType.APPLICATION_JSON)
            .content(hotelRequestString))
            .andExpect(status().isCreated());
        Assertions.assertEquals( expected: 1, hotelRepository.findAll().size());
    }

    private HotelRequest getHotelRequest() {
        return HotelRequest.builder()
            .hotelName("Fusion Suites")
            .address("2 Trương Công Định")
            .star(4)
            .description("Khách sạn gần biển")
            .city("Vũng Tàu")
            .province("Bà Rịa - Vũng Tàu")
            .ratingScore(8.80)
            .ratingCategory("Tuyệt vời")
            .numOfReviews(4)
            .originPrice(600000)
            .discount(0.50)
            .discountPrice(300000)
            .taxPercentage(0.10)
            .taxPrice(30000)
            .extraFee(0)
            .totalPrice(300000)
            .build();
    }
}
```









Hình 4.32 Viết test cho hàm *shouldCreateHotel()*

Nếu như kết quả pass, nó sẽ hiển thị như vậy:



Hình 4.33 Phần test đã pass

Ta cũng có thể quan sát 1 testcontainer trong Docker được tạo ra khi chạy test

<input type="checkbox"/>	Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
<input type="checkbox"/>	 testcontainers-ryuk-f5d496da-2 2b8933ab62c9	testcontainers/ryuk:0.6.0	Running	0%	53605:8080	1 second ago	  
<input type="checkbox"/>	 boring_hugle 4ccf5de8524e	mongo:4.4.2	Running	0%	53607:27017	1 second ago	  

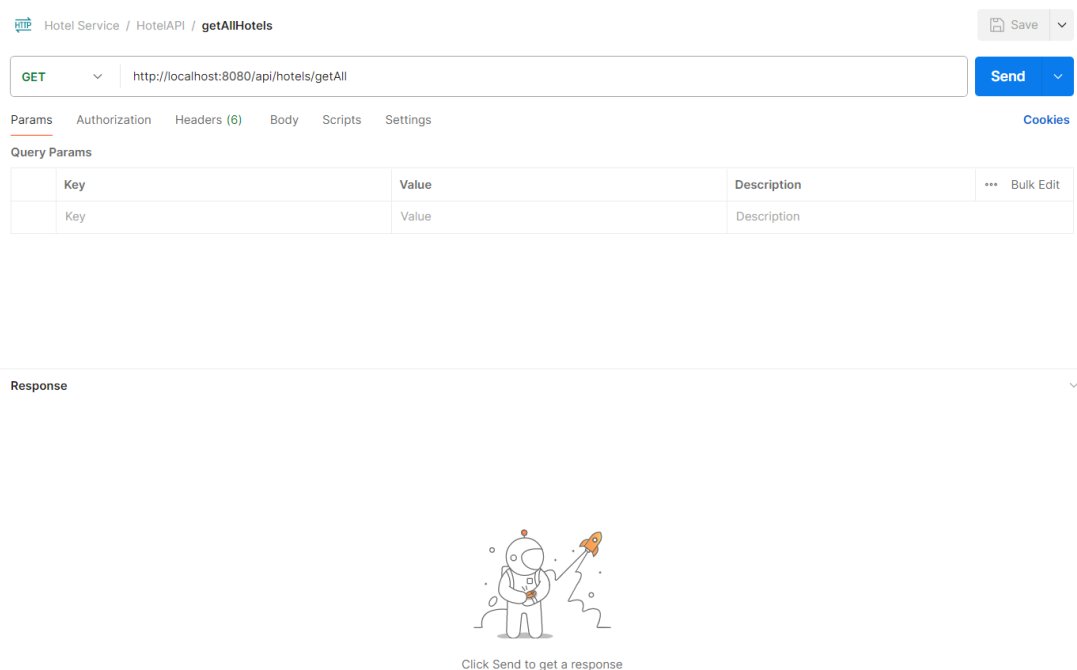
Hình 4.34 Test container được tạo ra trong Docker

Ta thực hiện tương tự với các hàm khác để có thể xem các kết quả tương ứng.

4.4.3. Kiểm thử API

Như đã đề cập ở phần cài đặt, ta sẽ kiểm thử API bằng công cụ Postman.

Giả sử ta sẽ kiểm tra xem danh sách các khách sạn có được trả về hay không, ta sử dụng hàm **GET** để lấy các khách sạn theo route **/getAll** đã định nghĩa trước.



Hình 4.35 Giao diện Postman khi chưa gửi yêu cầu

Ta thực hiện gửi yêu cầu đến server để kiểm tra xem server có trả về danh sách khách sạn hay không, nhấn nút “Send” để xem kết quả.

Rõ ràng ta sẽ thấy danh sách các khách sạn theo các trường được định nghĩa đã được trả về:

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/hotels/getAll`. The response status is 200 OK, and the body is a JSON array containing one hotel object. The JSON response is as follows:

```
[
  {
    "id": "664cdc26bd16c1e72174caf4",
    "hotelName": "Sofitel Legend Metropole Hanoi",
    "address": "15 Ngõ Quyền, Quận Hoàn Kiếm, Hà Nội",
    "star": 5,
    "description": "Khách sạn sang trọng với kiến trúc Pháp cổ điển, nổi tiếng với dịch vụ đẳng cấp",
    "images": [
      "https://www.sofitel-legend-metropole-hanoi.com/wp-content/uploads/sites/95/2022/06/DSC00934-scaled.jpg",
      "https://dynamic-media-cdn.tripadvisor.com/media/photo-o/01/fb/54/c7/swimming-pool.jpg?w=700&h=-1&s=1",
      "https://www.vn.kayak.com/img/himg/2b/85/04/ice-85672018-71957861_3XL-584000.jpg?width=968&height=607&crop=true",
      "https://phgcdn.com/images/uploads/HANSL/diningimages/1600x813-Le-Beaulieu-Restaurant2.jpg",
      "https://cdn.business traveller.com/wp-content/uploads/2017/06/La-Terrasse-By-Minoru-Masujima.jpg",
      "https://www.sofitel-legend-metropole-hanoi.com/wp-content/uploads/sites/95/2022/07/DSC08026-scaled.jpg",
      "https://d2e5ushqwiltxm.cloudfront.net/wp-content/uploads/sites/95/2016/12/21150434/3557903_XL-1.jpg"
    ],
    "city": "Hà Nội",
    "province": "Hà Nội",
    "ratingScore": 9.5,
    "ratingCategory": "Tuyệt vời",
    "numOfReviews": 1023,
    "originPrice": 3000000,
    "discount": 0.1,
    "discountPrice": 2700000,
    "taxPercentage": 0.08
  }
]
```

Hình 4.36 Kết quả trả về danh sách các khách sạn

Chương 5. KẾT LUẬN

5.1. Công việc đã hoàn thành

STT	Tên Use Case	Đặc tả	UI/UX	Frontend	Backend	Database
1	Đăng ký	✓	✓	✓	-	-
2	Tìm kiếm Khách sạn	✓	✓	✓	✓	✓
3	Sắp xếp Danh sách Khách sạn	✓	✓	✓	✓	✓
4	Lọc Danh sách Khách sạn	✓	✓	✓	✓	✓
5	Xem thông tin Khách sạn	✓	✓	✓	✓	✓
6	Xem thông tin Phòng	✓	✓	✓	✓	✓
7	Xem thông tin Hỗ trợ	✓	✓	✓	-	-
8	Xem thông tin trang web	✓	✓	✓	-	-
9	Đăng nhập	✓	✓	✓	-	-
10	Đăng xuất	✓	✓	✓	-	-
11	Quên mật khẩu	✓	✓	✓	-	-
12	Điền thông tin Phiếu đặt phòng	✓	✓	✓	✓	✓
13	Đặt phòng	✓	✓	✓	✓	✓
14	Hủy đặt phòng	✓	✓	✓	✓	✓
15	Thanh toán trực tuyến	✓	✓	✓	✓	✓
16	Đánh giá Phòng	✓	✓	✓	✓	✓
17	Xem lịch sử đặt phòng	✓	✓	✓	✓	✓
18	Sửa thông tin tài khoản	✓	✓	✓	-	-
19	Đổi mật khẩu	✓	✓	✓	-	-

Hình 5.1 Các công việc đã hoàn thành

Có thể nói, sản phẩm đồ án tuy chưa phải là hoàn hảo nhất, nhưng kết quả và quá trình thực hiện các công việc vẫn đảm bảo đầu ra cho môn học, tạo ra sản phẩm cuối có thể dùng được theo sát với các tiêu chí đã định hướng trước đó.

Các công việc được phân chia hợp lý và thực hiện theo từng giai đoạn, đảm bảo mỗi giai đoạn đều hoàn thành một chỉ tiêu nhất định để công việc được hoàn thành đúng hạn.

5.2. Kết quả đồ án

Sau khi thực hiện, đồ án đã đạt được những kết quả sau:

- **Yêu cầu:** Đồ án thỏa mãn yêu cầu đề bài đưa ra, đảm bảo các luồng cơ bản được thực hiện đúng đắn.

- **Thiết kế:** Các yêu cầu được đặc tả để thuận tiện cho việc thiết kế, các thiết kế có thể kể đến như thiết kế giao diện, thiết kế cơ sở dữ liệu, thiết kế hệ thống và các sơ đồ kèm theo.
- **Cài đặt:** Ứng dụng được cài đặt để thích ứng với các công nghệ sử dụng.
- **Kiểm thử:** Áp dụng được các phương pháp kiểm thử để đảm bảo chất lượng phần mềm.

Ngoài ra, đồ án không chỉ đảm bảo quy trình theo vòng đời phát triển phần mềm như các bước trên mà còn là sự vận dụng và lĩnh hội những kiến thức mới, những công nghệ mới có thể kể đến như Next.js, Java Spring Boot và đặc biệt là gRPC.

Với Next.js được sử dụng ở phía front-end giúp cải thiện hiệu suất hiển thị các thành phần tương tác cho người dùng cuối, hỗ trợ mạnh mẽ các tác vụ giúp tối ưu hóa giao diện và trải nghiệm người dùng hơn.

Với Spring Boot được sử dụng ở phía back-end giúp tách bạch các thành phần hạ tầng của ứng dụng, giảm thiểu lỗi có thể xảy ra trong quá trình xây dựng và sử dụng nhằm đảm bảo chất lượng phần mềm được viết ra, tạo mối liên hệ chặt chẽ giữa thực tiễn và lý thuyết của đề tài.

Hơn nữa, việc áp dụng gRPC là phương án mới mẻ giúp các thành phần bên dưới ứng dụng có cơ hội tối ưu giao tiếp với nhau, tăng cường sự linh hoạt về dung lượng lẫn tốc độ truyền tải thông tin trong bối cảnh cần chinh phục dữ liệu lớn như ngày nay.

5.3. Ưu điểm và hạn chế

❖ Ưu điểm:

- Quá trình thực hiện:
 - Được trải nghiệm và học hỏi thêm nhiều kiến thức mới.
 - Được sự chỉ dẫn tận tình của GVHD.

- Đồ án mang tính ứng dụng.
- Áp dụng được nhiều công nghệ mới.
- Nguồn tài liệu dồi dào để tham khảo.
- Sản phẩm đồ án:
 - Thực hiện được các chức năng cơ bản và các luồng cơ bản khi phân tích yêu cầu.
 - Đảm bảo hạn chế lỗi nhờ thiết kế được kiến trúc hệ thống phù hợp kết hợp với nhiều phương pháp kiểm thử.
 - Tối thiểu được các chức năng của một website thương mại điện tử: Đặt phòng, Hủy đặt phòng, Thanh toán, Đánh giá, Quản lý tài khoản.
 - Áp dụng được gRPC để giảm thiểu thời gian đáp ứng khi truyền tải.

❖ Hạn chế:

- Quá trình thực hiện:
 - Cần học nhiều kiến thức mới trong thời gian hạn chế.
 - Thời gian đầu có những thay đổi bất cập.
 - Nguồn tài liệu nhiều nhưng cần chất lọc vì khó đọc.
- Sản phẩm đồ án:
 - Các dữ liệu chưa được hiển thị hết và các chức năng chưa được tối ưu ở mức tối đa.

5.4. Hướng phát triển

Website đặt phòng khách sạn theo mô típ của các hệ thống thương mại điện tử, nên số lượng người dùng và độ lớn của dữ liệu ngày càng lớn hơn. Vì thế, ứng dụng có tiềm năng để phát triển và mở rộng theo chiều ngang, đồng thời cũng có thể áp dụng những công nghệ tiên tiến và nâng cấp, bảo trì thường xuyên để không bị lỗi thời so với nhu cầu của người dùng hiện đại.

Hệ thống có thể phát triển thêm những tính năng như sau:

- Tích hợp chatbot để khách đặt phòng và phía khách sạn có thể tương tác được với nhau.
- Phát triển theo khuynh hướng hệ khuyến nghị, dựa vào hành vi người dùng để đề xuất các khách sạn theo ưu đãi và nhu cầu.
- Tích hợp đa dạng phương thức thanh toán để phù hợp với nhu cầu của người dùng.
- Cải thiện hiệu suất của hệ thống về hạ tầng.
- Cải thiện giao diện người dùng và nâng cao trải nghiệm người dùng.
- Có thể tích hợp trí tuệ nhân tạo (AI) giúp đưa ra cho người dùng những sự lựa chọn linh hoạt.

TÀI LIỆU THAM KHẢO

- [1]. "Core Concepts, architecture and Lifecycle," gRPC, <https://grpc.io/docs/what-is-grpc/core-concepts/> (Truy cập lần cuối ngày 09/6/2024).
- [2]. "Gioi-thieu-ve-protocol-buffers," Viblo, <https://viblo.asia/p/gioi-thieu-ve-protocol-buffers-PwIVmx3wL5Z> (Truy cập lần cuối ngày 09/6/2024).
- [3]. "Hiểu rõ VỀ JSON LÀ GÌ? cách lấy dữ liệu TỪ JSON," TopDev, <https://topdev.vn/blog/json-la-gi/> (Truy cập lần cuối ngày 09/6/2024).
- [4]. "HTTP/2," <https://http2.github.io/> (Truy cập lần cuối ngày 09/6/2024).
- [5]. "HTTP/2 LÀ GÌ? so Sánh HTTP/2 và HTTP/1," 200Lab Blog, <https://200lab.io/blog/http2-la-gi/> (Truy cập lần cuối ngày 09/6/2024).
- [6]. "Introducing grpc to our Hotels.com platform - part 1," Medium, N. Katirtzis, <https://medium.com/expedia-group-tech/introducing-grpc-to-our-hotels-com-platform-part-1-61716af50b13> (Truy cập lần cuối ngày 01/5/2024).
- [7]. "Introducing grpc to our Hotels.com platform - part 2," Medium, N. Katirtzis, <https://medium.com/expedia-group-tech/introducing-grpc-to-our-hotels-com-platform-part-2-8024a1dda0aa> (Truy cập lần cuối ngày 01/5/2024).
- [8]. "Java Spring Boot là gì," Stringee, <https://stringee.com/vi/blog/post/java-spring-boot-la-gi> (Truy cập lần cuối ngày 09/6/2024).
- [9]. "Kiến Trúc Nguyên Khối (monolithic) LÀ GÌ?," TIGO Software, <https://tigosoftware.com/vi/kien-truc-nguyen-khoi-monolithic-la-gi/> (Truy cập lần cuối ngày 09/6/2024).
- [10]. "Microservices communication using GRPC protocol," LinkedIn, P. Pankaj, <https://www.linkedin.com/pulse/microservices-communication-using-grpc-protocol-prabhat-pankaj> (Truy cập lần cuối ngày 01/5/2024).
- [11]. "Microservices là gì? Ứng dụng của kiến trúc này như thế nào," LinkedIn, <https://www.linkedin.com/pulse/microservices-l%C3%A0-g%C3%AC-%E1%BB%A9ng-d%E1%BB%A5ng-c%E1%BB%A7a-ki%E1%BA%BFn-tr%C3%BAC-n%C3%A0y-nh%C6%B0-th%E1%BA%BF-n%C3%A0o> (Truy cập lần cuối ngày 09/6/2024).

- [12]. "Monolithic vs microservices architecture," GeeksforGeeks, <https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture/#differences-between-monolithic-and-microservices-architecture> (Truy cập lần cuối ngày 09/6/2024).
- [13]. "NextJS quản lý người dùng rất đơn giản nhất với Clerk," Viblo, <https://viblo.asia/p/nextjs-quan-ly-nguoi-dung-rat-don-gian-nhat-voi-clerk-yZjJYKeOVOE> (Truy cập lần cuối ngày 09/6/2024).
- [14]. "RESTful API LÀ GÌ? Cách thiết KẾ RESTful API," TopDev, <https://topdev.vn/blog/restful-api-la-gi/> (Truy cập lần cuối ngày 14/5/2024).
- [15]. "Secureframe Blog," Secureframe, <https://secureframe.com/blog/soc-2-type-ii> (Truy cập lần cuối ngày 09/6/2024).
- [16]. "Thanh toán Pay," VNPayment, <https://sandbox.vnpayment.vn/apis/docs/thanh-toan-pay/pay.html> (Truy cập lần cuối ngày 09/6/2024).
- [17]. "What is Docker?," AWS, <https://aws.amazon.com/vi/docker/> (Truy cập lần cuối ngày 09/6/2024).
- [18]. "What is service-oriented architecture?," AWS, <https://aws.amazon.com/vi/what-is/service-oriented-architecture/> (Truy cập lần cuối ngày 09/6/2024).
- [19]. "A practical example of applying attribute-driven design (ADD) version 2.0," SEI Insights, <https://insights.sei.cmu.edu/library/a-practical-example-of-applying-attribute-driven-design-add-version-20/> (Truy cập lần cuối ngày 09/6/2024).
- [20]. "Yếu tố ảnh hưởng đến ý định đặt phòng trực tuyến của khách du lịch nội địa," Vietnam Tourism, <https://vietnamtourism.gov.vn/post/32527> (Truy cập lần cuối ngày 09/6/2024).
- [21]. "gRPC là gì? Vũ khí tối thượng tăng tải Microservices," 200Lab Blog, <https://200lab.io/blog/grpc-la-gi/> (Truy cập lần cuối ngày 09/6/2024).

- [22]. Bemile, R., Achampong, A., & Danquah, E., "Online Hotel Reservation System," International Journal of Innovative Science, Engineering & Technology, 2014.
- [23]. N. Katirtzis, "Introducing grpc to our Hotels.com platform - part 1," Medium, <https://medium.com/expedia-group-tech/introducing-grpc-to-our-hotels-com-platform-part-1-61716af50b13> (Truy cập lần cuối ngày 01/5/2024).
- [24]. N. Katirtzis, "Introducing grpc to our Hotels.com platform - part 2," Medium, <https://medium.com/expedia-group-tech/introducing-grpc-to-our-hotels-com-platform-part-2-8024a1dda0aa> (Truy cập lần cuối ngày 01/5/2024).
- [25]. P. Pankaj, "Microservices communication using GRPC protocol," LinkedIn, <https://www.linkedin.com/pulse/microservices-communication-using-grpc-protocol-prabhat-pankaj> (Truy cập lần cuối ngày 01/5/2024).
- [26]. Richard Bemile, Akwasi Achampong, & Emmanuel Danquah, "Online Hotel Reservation System," International Journal of Innovative Science, Engineering & Technology, 2014.
- [27]. "gRPC," <https://grpc.io/> (Truy cập lần cuối ngày 09/6/2024).
- [28]. "Microservices.io," <https://microservices.io/> (Truy cập lần cuối ngày 09/6/2024).
- [29]. "MongoDB Documentation," MongoDB, <https://www.mongodb.com/docs> (Truy cập lần cuối ngày 09/6/2024).
- [30]. "Next.js Documentation," Next.js, <https://nextjs.org/docs> (Truy cập lần cuối ngày 09/6/2024).
- [31]. "NextUI Documentation," NextUI, <https://nextui.org/docs> (Truy cập lần cuối ngày 09/6/2024).