TRƯỜNG ĐẠI HỌC TRÀ VINH
KHOA KỸ THUẬT VÀ CÔNG NGHỆ

ISO 9001:2015

KIM THANH ÁI NHÂN

XÂY DỰNG HỆ THỐNG QUẢNG BÁ DU LỊCH CÁC TỈNH MIỀN TÂY NAM BỘ VỚI CƠ SỞ DỮ LIỆU PHI QUAN HỆ

ĐỒ ÁN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN

TRÀ VINH, NĂM 2024

TRƯỜNG ĐẠI HỌC TRÀ VINH
KHOA KỸ THUẬT VÀ CÔNG NGHỆ

XÂY DỰNG HỆ THỐNG QUẢNG BÁ DU LỊCH CÁC TỈNH MIỀN TÂY NAM BỘ VỚI CƠ SỞ DỮ LIỆU PHI QUAN HỆ

ĐỒ ÁN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ THÔNG TIN

Sinh viên: **Kim Thanh Ái Nhân**
Lớp: **DA20TTB**
MSSV: **110120146**
GVHD: **Th.S Phan Thị Phương Nam**

TRÀ VINH, NĂM 2024

LỜI MỞ ĐẦU

Một hệ thống quảng bá du lịch hiệu quả không chỉ cần cung cấp thông tin đầy đủ, chính xác và hấp dẫn về các điểm tham quan, văn hóa, ẩm thực và dịch vụ du lịch mà còn phải đảm bảo tính linh hoạt, dễ dàng mở rộng và cập nhật liên tục. Với sự phát triển mạnh mẽ của công nghệ thông tin, cơ sở dữ liệu phi quan hệ nổi lên như một giải pháp lý tưởng cho việc lưu trữ và quản lý lượng lớn dữ liệu đa dạng, không có cấu trúc cố định. Khả năng xử lý nhanh chóng và linh hoạt của cơ sở dữ liệu phi quan hệ sẽ giúp hệ thống truy cập thông tin một cách hiệu quả, đồng thời dễ dàng đáp ứng nhu cầu mở rộng trong tương lai.

Đề tài nghiên cứu gồm 5 chương:

Chương 1. Đặt vấn đề.

Chương 2. Cơ sở lý thuyết.

Chương 3. Hiện thực hóa nghiên cứu.

Chương 4. Kết quả nghiên cứu.

Chương 5. Kết luận và hướng phát triển.

LỜI CẢM ƠN

Tôi xin được gửi lời cảm ơn chân thành và sâu sắc nhất đến Ban Giám hiệu Trường Đại học Trà Vinh đã tạo điều kiện cho tôi được học tập và rèn luyện tại Trường Đại học Trà Vinh trong suốt thời gian qua. Tôi xin gửi lời tri ân đến quý thầy cô Khoa Kỹ thuật và Công nghệ, những người đã truyền cho tôi những kiến thức quý báu, giúp tôi trưởng thành và phát triển trong suốt quá trình học tập tại Trường. Lời cảm ơn sâu sắc nhất tôi xin dành cho Cô Phan Thị Phương Nam, người đã trực tiếp hướng dẫn tôi thực hiện đồ án tốt nghiệp. Nhờ sự tận tâm, nhiệt tình và những lời khuyên quý giá của Cô, tôi đã hoàn thành được đồ án đúng thời hạn và tích lũy được cho mình nhiều kiến thức và kinh nghiệm bổ ích.

Tuy nhiên, trong quá trình thực hiện đồ án, do kiến thức và kinh nghiệm còn hạn chế, tôi không tránh khỏi những thiếu sót và sai sót. Rất mong nhận được sự quan tâm, góp ý của quý thầy, cô giảng viên trong bộ môn để đề tài của tôi được hoàn thiện hơn.

Tôi xin chân thành cảm ơn !

Trà Vinh, ngày …… tháng 7 năm 2024

Sinh viên thực hiện

Kim Thanh Ái Nhân

NHẬN XÉT
(Của giảng viên hướng dẫn trong đồ án, khoá luận của sinh viên)

Đáp ứng yêu cầu về nội dung thực hiện đã đề ra trong đề cương,

Đạt yêu cầu về đồ án tốt nghiệp của sinh viên ngành Công nghệ thông tin,

Đề tài có thể là tài liệu tham khảo cho sinh viên ngành Công nghệ thông tin các khóa sau.

……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……
……

Giảng viên hướng dẫn

Phan Thị Phương Nam

UBND TỈNH TRÀ VINH
TRƯỜNG ĐẠI HỌC TRÀ VINH

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh Phúc

BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP

(Của giảng viên hướng dẫn)

Họ và tên sinh viên: Kim Thanh Ái Nhân MSSV: 110120146
Ngành: Công nghệ thông tin Khóa: 2020 - 2024
Tên đề tài: Xây dựng hệ thống quảng bá du lịch các tỉnh miền Tây Nam Bộ với cơ sở dữ liệu phi quan hệ
Họ và tên Giáo viên hướng dẫn: Phan Thị Phương Nam
Chức danh: Giảng viên chính Học vị: Thạc sĩ

NHẬN XÉT

1. Nội dung đề tài:
   - Đáp ứng yêu cầu về nội dung thực hiện đã đề ra trong đề cương,
   - Đạt yêu cầu về đồ án tốt nghiệp của sinh viên ngành Công nghệ thông tin,
   - Đề tài có giá trị thực tiễn, có thể áp dụng trong thực tế,
   - Đề tài có thể là tài liệu tham khảo cho sinh viên ngành Công nghệ thông tin các khóa sau.

.....
.....

2. Ưu điểm:

Sinh viên Kim Thanh Ái Nhân có tinh thần, thái độ học tập, làm việc tốt, luôn thực hiện công việc giảng viên hướng dẫn giao đúng thời hạn, đạt yêu cầu, có tinh thần cầu thị, học hỏi, có khả năng tự học tập, nghiên cứu tốt, có tinh thần đoàn kết.

Dữ liệu minh họa demo phong phú, đa dạng các điểm du lịch của các tỉnh trong phạm vi nghiên cứu của đề tài.

3. Khuyết điểm:

Chưa có kinh nghiệm trong thiết kế giao diện nên thiết kế giao diện demo có một vài mục chưa đảm bảo theo nguyên tắt thiết kế giao diện.

.....
.....

4. Điểm mới đề tài:

Đề tài có tính mới là thiết kế web với cơ sở dữ liệu NoSQL, đặc biệt phần NoSQL trong chương trình học sinh viên chưa được học, nên sinh viên phải mất nhiều thời gian tự học về thiết kế lược đồ cho cơ sở dữ liệu theo cấu trúc phi quan hệ, cài đặt và sử dụng hệ quản trị cơ sở dữ liệu phi quan hệ.

Phạm vi nghiên cứu là các điểm du lịch các tỉnh Tây Nam Bộ là phạm vi tương đối rộng so với một số nghiên cứu chỉ tìm hiểu trong phạm vi một tỉnh.

5. Giá trị thực trên đề tài:

Có thể triển khai thực tế khi có đơn vị đặt hàng sử dụng.

Đề tài có giá trị thực tiễn cao, có khả năng áp dụng vào thực tế.

.....

.....

.....

.....

.....

.....

7. Đề nghị sửa chữa bổ sung:

Sinh viên đã chỉnh sửa theo góp ý của Hội đồng bảo vệ khóa luận

.....

.....

.....

.....

.....

.....

8. Đánh giá:

Đạt yêu cầu về đồ án tốt nghiệp của sinh viên ngành Công nghệ thông tin.

Dữ liệu demo phong phú, đa dạng.

.....

.....

.....

Trà Vinh, *ngày 31 tháng 07 năm 2024*

Giảng viên hướng dẫn

Phan Thị Phương Nam

MỤC LỤC

DANH MỤC CÁC BẢNG

DANH MỤC HÌNH ẢNH

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa
ACID	Atomicity, Consistency, Isolation, Durability
AI	Artificial Intelligence
CSDL	Cơ sở dữ liệu
IoT	Internet of Things

CHƯƠNG 1. ĐẶT VẤN ĐỀ

1.1. Lý do chọn đề tài.

Miền Tây Nam Bộ có tiềm năng du lịch lớn với hệ sinh thái sông nước độc đáo, văn hóa miệt vườn phong phú và con người hồn hậu, chất phác. Ngành du lịch đóng vai trò quan trọng trong phát triển kinh tế - xã hội của khu vực, tạo việc làm và thúc đẩy các ngành dịch vụ khác. Tuy nhiên, du lịch miền Tây Nam Bộ hiện nay còn nhiều hạn chế, chưa được khai thác hiệu quả so với tiềm năng. Hệ thống quảng bá du lịch hiệu quả có vai trò quan trọng trong việc thu hút du khách đến với miền Tây Nam Bộ. Hệ thống cần cung cấp thông tin đầy đủ, chính xác và hấp dẫn về các điểm tham quan, văn hóa, ẩm thực và dịch vụ du lịch. Cơ sở dữ liệu phi quan hệ có khả năng lưu trữ và xử lý lượng lớn dữ liệu đa dạng, không có cấu trúc cố định, phù hợp với nhu cầu lưu trữ thông tin du lịch. Hệ thống cho phép truy cập dữ liệu nhanh chóng, linh hoạt, dễ dàng mở rộng và cập nhật.

Với những lý do trên, đề tài "Xây dựng hệ thống quảng bá du lịch các tỉnh miền Tây Nam Bộ với cơ sở dữ liệu phi quan hệ" góp phần thúc đẩy phát triển du lịch miền Tây Nam Bộ một cách hiệu quả và bền vững. Ngoài ra, đề tài còn có thể góp phần bảo tồn và phát huy bản sắc văn hóa độc đáo của miền Tây Nam Bộ.

1.2. Mục tiêu.

- Tìm hiểu cụ thể về các điểm du lịch tại các tỉnh miền Tây Nam Bộ,
- Thiết kế mô hình dữ liệu phi quan hệ;
- Sử dụng Hệ quản trị CSDL phi quan hệ để lưu trữ dữ liệu;
- Thiết kế website kết nối với CSDL phi quan hệ.

1.3. Nội dung.

- Tìm hiểu các địa điểm du lịch miền Tây Nam bộ:
    - Danh sách các địa điểm du lịch phổ biến ở miền Tây Nam bộ,
    - Đặc điểm địa lý, văn hóa, và lịch sử của các địa điểm này,
    - Hoạt động và trải nghiệm du lịch độc đáo ở từng địa điểm,
- Nghiên cứu cách thiết kế môn hình dữ liệu phi quan hệ:
    - Phân tích các yếu tố quan trọng trong thiết kế mô hình dữ liệu phi quan hệ.
    - Đặc điểm và ứng dụng của mô hình dữ liệu phi quan hệ.

- Phương pháp và công cụ để thiết kế mô hình dữ liệu phi quan hệ.

- Xây dựng mô hình dữ liệu phi quan hệ cho các địa điểm du lịch:
  - Phân tích yêu cầu dữ liệu cho hệ thống quản lý địa điểm du lịch.
  - Thiết kế mô hình dữ liệu phi quan hệ phù hợp với các yêu cầu cụ thể.
  - Các quan hệ và thuộc tính cần thiết để mô tả thông tin về địa điểm du lịch.
- Xây dựng hệ thống Web quảng bá các địa điểm du lịch với thông tin tìm hiểu và mô hình dữ liệu:
  - Thiết kế giao diện người dùng trực quan và dễ sử dụng cho hệ thống Web.
  - Tích hợp thông tin tìm hiểu về các địa điểm du lịch vào hệ thống.
  - Triển khai mô hình dữ liệu phi quan hệ vào hệ thống Web và quản lý dữ liệu hiệu quả.

1.4. Đối tượng và phạm vi nghiên cứu.

Đối tượng nghiên cứu:

- Các địa điểm du lịch tại các tỉnh miền Tây Nam bộ
- Hệ quản trị cơ sở dữ liệu MongoDB.
- NodeJS và thư viện ExpressJS.
- Resful API.
- FrameWork ReactJS.

Phạm vi nghiên cứu

- Các địa điểm du lịch tại các tỉnh miền Tây Nam bộ
- Hệ quản trị cơ sở dữ liệu MongoDB,
- Xây dựng mô hình dữ liệu phi quan hệ,
- NodeJS, Resful API, Framework ReactJS Phương pháp nghiên cứu.

Phương pháp nghiên cứu lý thuyết:

- Nghiên cứu tài liệu về địa lý, các địa điểm du lịch của các tỉnh Tây Nam Bộ,
- Tìm hiểu các tài liệu hướng dẫn và tài liệu tham khảo liên quan đến xây dựng hệ thống quảng bá du lịch với cơ sở dữ liệu phi quan hệ,
- Nghiên cứu các báo cáo và nghiên cứu trước đó liên quan đến điểm du lịch của miền Tây Nam Bộ, cũng như các hệ thống quảng bá du lịch.

Phương pháp thực nghiệm

- Cài đặt các công cụ cần thiết, tùy chỉnh theo kế hoạch thiết kế đã xác định,
- Xây dựng mô hình cơ sở dữ liệu phi quan hệ,
- Phát triển các tính năng và giao diện theo yêu cầu đã đặt ra,
- Cài đặt thử nghiệm và test hệ thống.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về Tây Nam Bộ.

2.1.1. *Vị trí địa lý* .

- Miền Tây Nam Bộ, còn được gọi là Đồng bằng Sông Cửu Long, nằm ở phía Nam Việt Nam. Khu vực này bao gồm 13 tỉnh thành: An Giang, Bạc Liêu, Bến Tre, Cà Mau, Cần Thơ, Đồng Tháp, Hậu Giang, Kiên Giang, Long An, Sóc Trăng, Tiền Giang, Trà Vinh và Vĩnh Long. Mỗi tỉnh đều có nét đặc trưng văn hóa và lịch sử riêng biệt, tạo nên một vùng đất đa dạng và phong phú về cả tự nhiên lẫn con người.

- Phía Đông và Nam của miền Tây Nam Bộ giáp biển Đông và vịnh Thái Lan, với đường bờ biển dài và nhiều cửa sông lớn như sông Tiền và sông Hậu. Những dòng sông này không chỉ là nguồn cung cấp nước ngọt mà còn đóng vai trò quan trọng trong việc vận chuyển hàng hóa và phát triển kinh tế của vùng.

- Địa hình chủ yếu của miền Tây Nam Bộ là đồng bằng phẳng thấp, được bồi đắp bởi lượng phù sa dồi dào từ các con sông. Đây là yếu tố quan trọng giúp cho nông nghiệp ở khu vực này phát triển mạnh mẽ, đặc biệt là sản xuất lúa gạo, cây ăn trái và thủy sản.

- Khí hậu của vùng này là khí hậu nhiệt đới gió mùa, với hai mùa rõ rệt: mùa mưa kéo dài từ tháng 5 đến tháng 11 và mùa khô từ tháng 12 đến tháng 4 năm sau. Nhiệt độ trung bình hàng năm dao động từ 25 đến 30 độ C, tạo điều kiện thuận lợi cho sự phát triển của nhiều loại cây trồng và vật nuôi.

- Hệ thống sông ngòi chằng chịt, kênh rạch dày đặc của miền Tây Nam Bộ đóng vai trò quan trọng trong giao thông, sinh hoạt và nông nghiệp. Các con sông, kênh rạch này không chỉ là tuyến đường giao thông chính mà còn cung cấp nguồn nước tưới tiêu, hỗ trợ cho hoạt động nuôi trồng thủy sản và là nơi diễn ra các hoạt động sinh hoạt, văn hóa của người dân địa phương.[7]

2.1.2. *Đặc điểm du lịch.*

- Phong phú và đa dạng: Du lịch miền Tây Nam Bộ mang đậm dấu ấn văn hóa sông nước độc đáo với sự phong phú và đa dạng. Các chợ nổi như Cái Răng, Phong Điền ở Cần Thơ, chợ nổi Ngã Bảy ở Hậu Giang hay chợ nổi Cái Bè ở Tiền Giang là những điểm đến hấp dẫn, nơi du khách có thể tận mắt chứng kiến và trải nghiệm cảnh buôn bán tấp nập trên sông. Bên cạnh đó, miền Tây còn nổi tiếng với những vườn trái cây trĩu quả như vườn

chôm chôm, sầu riêng, măng cụt ở Vĩnh Long, Tiền Giang, Bến Tre, cho phép du khách tự tay hái và thưởng thức trái cây tươi ngon ngay tại vườn. Những miệt vườn xanh mướt, những rừng tràm bạt ngàn ở Tràm Chim (Đồng Tháp) hay U Minh Hạ (Cà Mau) cũng là điểm đến không thể bỏ qua, cùng với các di tích lịch sử như khu di tích Xẻo Quýt, di tích văn hóa Óc Eo, tạo nên một bức tranh du lịch đa dạng và phong phú.

- Sự hấp dẫn: Du lịch miền Tây còn nằm ở những trải nghiệm độc đáo mà du khách có thể có. Khi đến đây, du khách sẽ được trải nghiệm cuộc sống bình dị của người dân địa phương, từ việc chèo thuyền trên sông, câu cá, đến tham gia vào các hoạt động sản xuất nông nghiệp. Những món ăn đặc sản thơm ngon như cá lóc nướng trui, lẩu mắm, bánh xèo, hủ tiếu Mỹ Tho, hay các loại bánh dân gian như bánh tét lá cẩm, bánh ít, bánh phồng Sơn Đốc sẽ làm say lòng bất kỳ ai yêu thích ẩm thực. Tham gia các hoạt động dân gian thú vị như hát đờn ca tài tử, một loại hình nghệ thuật truyền thống đã được UNESCO công nhận là Di sản Văn hóa Phi vật thể, cũng là một trải nghiệm không thể bỏ lỡ.

- Tiềm năng phát triển lớn: Miền Tây Nam Bộ có tiềm năng phát triển lớn nhờ vào hệ sinh thái đa dạng, cảnh quan thiên nhiên tươi đẹp, văn hóa độc đáo và con người thân thiện. Hệ sinh thái phong phú với các khu rừng ngập mặn, rừng tràm, các khu bảo tồn thiên nhiên như Vườn Quốc gia Tràm Chim, Vườn Quốc gia U Minh Thượng không chỉ là nơi bảo tồn động thực vật mà còn là điểm đến du lịch sinh thái lý tưởng. Cảnh quan thiên nhiên với những cánh đồng lúa bạt ngàn, những dòng sông hiền hòa, những khu miệt vườn xanh tươi, cùng với văn hóa sông nước đặc trưng và con người miền Tây thân thiện, hiếu khách là những lợi thế quan trọng để phát triển du lịch bền vững. Với những điều kiện thuận lợi này, miền Tây Nam Bộ hứa hẹn sẽ trở thành một điểm đến du lịch hấp dẫn và ngày càng phát triển. [7]

2.1.3. *Tầm quan trọng của du lịch miền Tây Nam Bộ.*

- Du lịch đóng vai trò quan trọng trong sự phát triển kinh tế - xã hội của miền Tây Nam Bộ. Ngành du lịch không chỉ tạo ra nguồn thu lớn từ các hoạt động du lịch, lưu trú và dịch vụ mà còn góp phần thúc đẩy các ngành dịch vụ khác như giao thông, thương mại, ẩm thực và văn hóa. Khi du lịch phát triển, nhu cầu về các dịch vụ đi kèm như vận chuyển, lưu trú, ăn uống và mua sắm cũng tăng lên, tạo điều kiện cho các ngành kinh tế khác phát triển theo. Đặc biệt, du lịch giúp tạo việc làm cho người dân địa phương, từ những công việc trực tiếp trong ngành như hướng dẫn viên, nhân viên khách sạn, nhà hàng, đến các

công việc gián tiếp như sản xuất hàng thủ công, nông sản đặc sản. Điều này không chỉ nâng cao đời sống vật chất mà còn cải thiện đời sống tinh thần của người dân, tạo ra một môi trường sống và làm việc tốt hơn.

- Tuy nhiên, hiện nay du lịch miền Tây Nam Bộ vẫn chưa được khai thác hết tiềm năng. Một số thách thức lớn đang cản trở sự phát triển của ngành du lịch khu vực này. Trước tiên, cơ sở hạ tầng du lịch ở một số nơi còn hạn chế. Nhiều tuyến đường giao thông vẫn chưa được nâng cấp, các cơ sở lưu trú và dịch vụ còn thiếu và chưa đáp ứng được nhu cầu ngày càng cao của du khách. Dịch vụ du lịch cũng chưa thực sự chuyên nghiệp, từ chất lượng phục vụ đến kỹ năng của nhân viên còn nhiều hạn chế. Thêm vào đó, công tác quảng bá du lịch chưa đủ mạnh mẽ và hiệu quả để thu hút du khách trong và ngoài nước. Nhiều điểm đến du lịch tiềm năng vẫn chưa được biết đến rộng rãi, các chương trình quảng bá còn thiếu sáng tạo và chưa tận dụng được các kênh truyền thông hiện đại.

- Để khắc phục những hạn chế này và phát huy tối đa tiềm năng du lịch, cần có sự đầu tư mạnh mẽ hơn vào cơ sở hạ tầng, nâng cao chất lượng dịch vụ và đẩy mạnh công tác quảng bá. Các chiến lược phát triển du lịch bền vững, bảo vệ môi trường và giữ gìn bản sắc văn hóa địa phương cũng cần được chú trọng để du lịch miền Tây Nam Bộ thực sự trở thành động lực phát triển kinh tế - xã hội của vùng. [7]

2.1.4. *Kết luận.*

Miền Tây Nam Bộ không chỉ sở hữu vị trí địa lý thuận lợi mà còn có nhiều địa điểm du lịch hấp dẫn và đặc sắc. Từ những chợ nổi sầm uất, các khu du lịch sinh thái, đến những bãi biển tuyệt đẹp và các di tích lịch sử văn hóa, khu vực này mang đến cho du khách những trải nghiệm độc đáo và phong phú. Các chợ nổi như Cái Răng, Phong Điền, các khu du lịch sinh thái như Tràm Chim, Vườn Quốc gia U Minh Thượng, cùng với các bãi biển tuyệt đẹp ở Kiên Giang và Cà Mau, đều là những điểm đến nổi bật.

Việc phát triển và quảng bá du lịch miền Tây Nam Bộ sẽ góp phần thúc đẩy kinh tế xã hội, bảo tồn và phát huy các giá trị văn hóa, đồng thời nâng cao đời sống vật chất và tinh thần cho người dân địa phương. Đầu tư vào cơ sở hạ tầng, cải thiện chất lượng dịch vụ và tạo ra các chiến lược quảng bá hiệu quả sẽ là những bước đi quan trọng để khu vực này khai thác hết tiềm năng du lịch, trở thành điểm đến hấp dẫn cho du khách trong và ngoài nước.

2.2. Tìm hiểu về cơ sở dữ liệu phi quan hệ.

2.2.1. *Tổng quan NoSQL.*

Thuật ngữ NoSQL (viết tắt của "Not Only SQL" hoặc "Not SQL") lần đầu tiên được giới thiệu vào năm 1998 để chỉ các cơ sở dữ liệu không sử dụng giao diện SQL truyền thống. Sự phát triển mạnh mẽ của NoSQL trong những năm 2000 gắn liền với sự mở rộng nhanh chóng của Internet và sự gia tăng khối lượng dữ liệu. Khả năng xử lý dữ liệu lớn và nhu cầu mở rộng hệ thống đã thúc đẩy sự phổ biến của các cơ sở dữ liệu NoSQL.

Cơ sở dữ liệu NoSQL, còn được biết đến với tên gọi Hệ thống Quản lý Dữ liệu Phi Quan hệ (Non-Relational Data Management System), nổi bật với khả năng linh hoạt trong việc quản lý và lưu trữ dữ liệu. Điểm mạnh chính của NoSQL là lược đồ linh hoạt, cho phép người dùng dễ dàng thay đổi cấu trúc dữ liệu mà không cần phải thay đổi toàn bộ cơ sở dữ liệu. Điều này đặc biệt hữu ích trong các hệ thống cần xử lý và lưu trữ dữ liệu khổng lồ, đặc biệt là trong các kho dữ liệu phân tán.

Mục đích chính của việc sử dụng cơ sở dữ liệu NoSQL là để phục vụ các ứng dụng có yêu cầu về lưu trữ và xử lý dữ liệu lớn, chẳng hạn như các nền tảng web thời gian thực và các hệ thống phân tích dữ liệu quy mô lớn. Các công ty công nghệ lớn như Twitter, Facebook và Google thường xuyên thu thập và phân tích hàng terabyte dữ liệu người dùng mỗi ngày, và NoSQL cung cấp nền tảng vững chắc để quản lý lượng dữ liệu khổng lồ này.

Các đặc tính nổi bật của cơ sở dữ liệu NoSQL bao gồm:

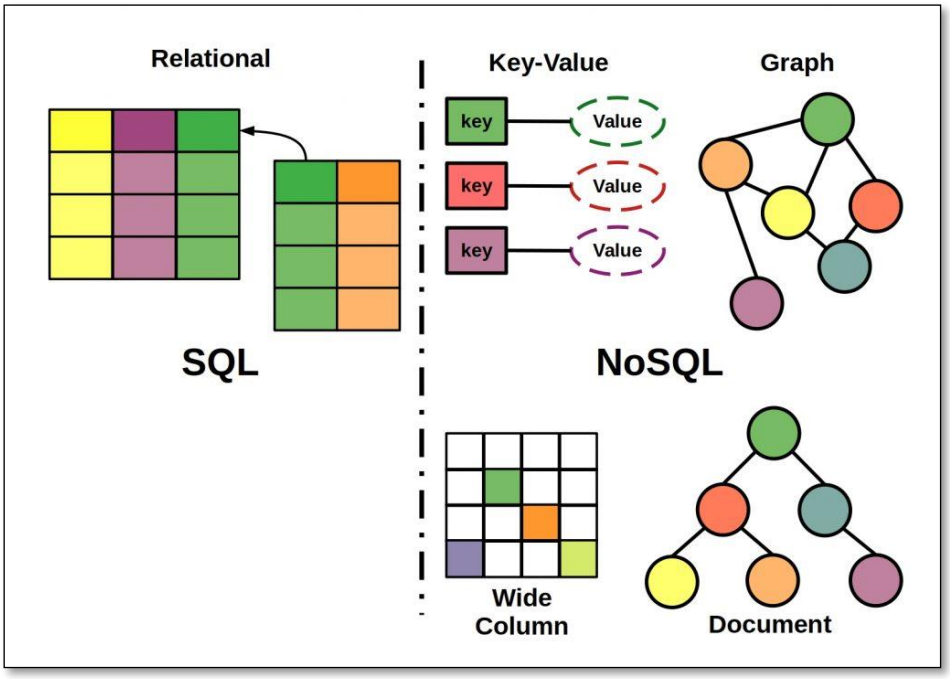
- Mô hình cơ sở dữ liệu phi quan hệ: Không sử dụng cấu trúc bảng quan hệ như trong các hệ thống cơ sở dữ liệu quan hệ (RDBMS).
- Khả năng phân tán và mở rộng thuộc tính theo chiều rộng: Hệ thống NoSQL được thiết kế để dễ dàng phân tán dữ liệu trên nhiều nút mạng, giúp mở rộng quy mô hệ thống một cách hiệu quả.
- Quy tắc lược đồ yếu hoặc không có: Lược đồ trong NoSQL không yêu cầu cấu trúc dữ liệu cứng nhắc, cho phép người dùng dễ dàng thay đổi định dạng và cấu trúc dữ liệu khi cần thiết.
- Sao chép dữ liệu dễ dàng: NoSQL hỗ trợ sao chép dữ liệu giữa các nút để đảm bảo tính sẵn sàng và độ tin cậy cao.

- Truy cập dữ liệu thông qua API: Cung cấp giao diện lập trình ứng dụng đơn giản để truy cập và quản lý dữ liệu.

- Mô hình nhất quán không phải là ACID: NoSQL thường không tuân theo các nguyên tắc ACID của các hệ thống cơ sở dữ liệu quan hệ, thay vào đó, sử dụng các mô hình nhất quán khác để đảm bảo hiệu suất và khả năng mở rộng.

Ngược lại, cơ sở dữ liệu quan hệ (RDBMS - Relational Database Management System) sử dụng cú pháp SQL để lưu trữ và truy xuất dữ liệu theo mô hình bảng quan hệ. Trong khi đó, hệ thống cơ sở dữ liệu NoSQL bao gồm nhiều công nghệ khác nhau có khả năng lưu trữ dữ liệu cấu trúc, bán cấu trúc, phi cấu trúc và đa hình. Sự khác biệt giữa cơ sở dữ liệu NoSQL và cơ sở dữ liệu SQL thể hiện rõ qua các mô hình dữ liệu, cách xử lý và khả năng mở rộng.

Sơ đồ dưới đây minh họa rõ ràng sự khác biệt giữa cơ sở dữ liệu NoSQL và cơ sở dữ liệu SQL, giúp làm nổi bật những điểm mạnh và hạn chế của từng loại hệ thống.[1]



Hình 2.1. Sơ đồ cấu trúc của CSDL SQL và CSDL NoSQL. [1]

2.2.2. *Đặc điểm chính.*

- Linh hoạt về mô hình dữ liệu:

- Key-value: Lưu trữ dữ liệu dưới dạng cặp khóa-giá trị, đơn giản và hiệu quả cho truy vấn dữ liệu dựa trên khóa.

- Document: Lưu trữ dữ liệu dạng tài liệu JSON hoặc BSON, linh hoạt cho dữ liệu phi cấu trúc và có thể nhúng lồng các trường.

- Column-family: Lưu trữ dữ liệu dạng bảng với nhiều cột, tối ưu cho truy vấn dữ liệu theo cột và có thể mở rộng theo hàng.

- Graph: Lưu trữ dữ liệu dạng đồ thị với các mối quan hệ giữa các thực thể, hiệu quả cho việc phân tích dữ liệu liên kết và truy vấn theo mối quan hệ.

Sự linh hoạt này giúp NoSQL thích ứng tốt với nhiều loại dữ liệu khác nhau, từ dữ liệu có cấu trúc đơn giản đến dữ liệu phức tạp và phi cấu trúc, đáp ứng nhu cầu đa dạng của các ứng dụng hiện đại.[1]

2.2.3. *Ưu điểm của NoSQL.*

- Tính linh hoạt và mở rộng: Nhờ kiến trúc phân tán, NoSQL có khả năng linh hoạt cao trong việc điều chỉnh cấu trúc hệ thống. Việc thêm hoặc bớt các nút (node) trong cụm có thể thực hiện dễ dàng mà không ảnh hưởng đáng kể đến hiệu suất hoạt động, đáp ứng nhu cầu thay đổi của ứng dụng theo thời gian. NoSQL được thiết kế để mở rộng theo chiều ngang, cho phép dễ dàng thêm nhiều máy chủ vào hệ thống khi nhu cầu lưu trữ và truy vấn dữ liệu tăng lên. Điều này giúp loại bỏ rào cản về hiệu suất thường gặp trong RDBMS khi xử lý khối lượng lớn dữ liệu.

- Hiệu suất cao: NoSQL được tối ưu hóa để xử lý hiệu quả khối lượng lớn dữ liệu với tốc độ truy vấn nhanh chóng. Nhờ khả năng phân tán dữ liệu và tận dụng các thuật toán xử lý song song, NoSQL có thể đáp ứng nhu cầu của các ứng dụng đòi hỏi hiệu suất cao và thời gian phản hồi ngắn. NoSQL lưu trữ và truy vấn dữ liệu phi cấu trúc hoặc bán cấu trúc một cách hiệu quả hơn so với hệ quản trị cơ sỡ dữ liệu quan hệ. Điều này đặc biệt hữu ích cho các ứng dụng cần xử lý các định dạng dữ liệu đa dạng như JSON, XML, hoặc dữ liệu từ các mạng xã hội, cảm biến IoT, v.v..

- Linh hoạt về mô hình dữ liệu: NoSQL không yêu cầu định nghĩa trước schema dữ liệu, mang lại sự linh hoạt cao trong việc thay đổi cấu trúc dữ liệu khi cần thiết. Điều này giúp đơn giản hóa quá trình phát triển và mở rộng ứng dụng, phù hợp với các trường hợp dữ liệu thay đổi thường xuyên. NoSQL cung cấp đa dạng mô hình dữ liệu như key-value, document, column-family và graph, đáp ứng nhu cầu lưu trữ và truy vấn dữ liệu khác nhau của nhiều loại ứng dụng.

- Khả năng chịu lỗi tốt: NoSQL lưu trữ dữ liệu trên nhiều nút trong cụm, giúp tăng khả năng chịu lỗi của hệ thống. Khi một nút gặp sự cố, dữ liệu vẫn được lưu trữ và truy vấn trên các nút còn lại, đảm bảo tính sẵn sàng và truy cập liên tục cho ứng dụng. Kiến trúc phân tán và các cơ chế sao chép dữ liệu, NoSQL có khả năng phục hồi nhanh chóng sau các sự cố hệ thống. Khi một nút bị lỗi, dữ liệu có thể được phục hồi từ các bản sao lưu trên các nút khác trong cụm, giúp giảm thiểu thời gian gián đoạn và đảm bảo tính ổn định cho hệ thống.[9]

2.2.4. *Hạn chế của NoSQL.*

- Thiếu tính nhất quán: Nhiều hệ thống NoSQL áp dụng mô hình nhất quán cuối cùng (eventual consistency) để đạt được khả năng mở rộng cao. Tuy nhiên, điều này có thể gây ra sự không đồng bộ tạm thời trong dữ liệu, gây khó khăn cho các ứng dụng yêu cầu tính nhất quán ngay lập tức.

- Khả năng tương thích hạn chế: CSDL NoSQL có thể gặp khó khăn trong việc tương thích với các hệ thống và công cụ hiện có sử dụng cơ sở dữ liệu SQL. Việc chuyển đổi từ hệ thống SQL sang NoSQL có thể yêu cầu nỗ lực lớn và thay đổi cơ bản trong kiến trúc hệ thống.

- Phụ thuộc vào loại dữ liệu và ứng dụng: NoSQL rất hiệu quả với các ứng dụng cần xử lý dữ liệu phi cấu trúc hoặc có mô hình dữ liệu phức tạp. Tuy nhiên, đối với các ứng dụng cần xử lý dữ liệu có cấu trúc chặt chẽ và phức tạp, RDBMS có thể vẫn là lựa chọn tốt hơn do tính năng mạnh mẽ về quản lý giao dịch và truy vấn phức tạp.

- Hạn chế về công cụ và hệ sinh thái: So với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS), hệ sinh thái của NoSQL vẫn chưa phát triển bằng. Các công cụ hỗ trợ, như quản lý, giám sát, và phân tích, có thể hạn chế hơn, dẫn đến việc khó khăn trong việc quản trị và tối ưu hệ thống.[9]

2.3. Tìm hiểu về MongoDB.

2.3.1. *Tổng quan về MongoDB.*

MongoDB là hệ quản trị cơ sở dữ liệu NoSQL, lưu trữ dữ liệu dưới dạng các tài liệu (document) thay vì bảng (table) như các hệ quản trị quan hệ (RDBMS) truyền thống. Nó nổi bật bởi sự linh hoạt, khả năng mở rộng và hiệu suất cao, thu hút đông đảo người dùng trong các ứng dụng web hiện đại và dự án Big Data. Tuy nhiên, việc lựa chọn MongoDB

cần đánh giá kỹ lưỡng cả ưu điểm và nhược điểm của nó trong bối cảnh khoa học.

Ngoài ra, MongoDB cung cấp các công cụ và thư viện hỗ trợ cho nhiều ngôn ngữ lập trình, giúp dễ dàng tích hợp với các ứng dụng được phát triển bằng nhiều ngôn ngữ khác nhau. Cộng đồng lớn và sự hỗ trợ từ MongoDB, Inc. làm cho MongoDB trở thành một lựa chọn phổ biến trong cộng đồng phần mềm mã nguồn mở và doanh nghiệp, đặc biệt là trong lĩnh vực của các ứng dụng web, phân tích dữ liệu, và các dự án yêu cầu tính linh hoạt và mở rộng.[1]

2.3.2. *Ưu điểm của MongoDB.*

- Mô hình dữ liệu linh hoạt: MongoDB không áp đặt cấu trúc dữ liệu cố định, cho phép lưu trữ dữ liệu một cách tự nhiên, mô phỏng các mối quan hệ thực tế trong ứng dụng. Cấu trúc dữ liệu có thể dễ dàng thay đổi theo thời gian mà không ảnh hưởng đến dữ liệu hiện có, đáp ứng nhu cầu phát triển linh hoạt của ứng dụng.

- Khả năng mở rộng tốt: MongoDB có thể mở rộng theo chiều ngang bằng cách thêm nhiều máy chủ vào cụm, đáp ứng nhu cầu lưu trữ và truy cập dữ liệu ngày càng tăng. Khả năng phân chia dữ liệu (sharding) giúp phân tán dữ liệu trên nhiều máy chủ, tối ưu hóa hiệu suất truy vấn và giảm thiểu chi phí phần cứng.

- Hiệu suất tối ưu cao: MongoDB sử dụng nhiều kỹ thuật tối ưu hóa như lưu trữ nhị phân BSON, kỹ thuật truy vấn tinh vi và bộ nhớ cache hiệu quả để cung cấp tốc độ truy vấn nhanh chóng, ngay cả với lượng dữ liệu lớn. MongoDB phù hợp cho các ứng dụng đòi hỏi truy cập dữ liệu nhanh và hiệu quả, ví dụ như ứng dụng web thời gian thực và hệ thống phân tích dữ liệu lớn.

- Dễ dàng sử dụng: MongoDB cung cấp giao diện lập trình ứng dụng (API) đơn giản và trực quan cho nhiều ngôn ngữ lập trình phổ biến như Java, Python, C++, JavaScript, v.v. Điều này giúp các nhà phát triển dễ dàng học tập, sử dụng và tích hợp MongoDB vào ứng dụng của họ.[10]

2.3.3. *Nhược điểm của MongoDB.*

- Khó khăn khi mở rộng: Mặc dù có khả năng mở rộng ngang, MongoDB có thể gặp khó khăn khi xử lý lượng dữ liệu tăng đột ngột, đòi hỏi quản lý phức tạp và cẩn thận để đảm bảo hiệu quả và tính nhất quán.

- Tốn nhiều bộ nhớ: Hoạt động hiệu quả của MongoDB đòi hỏi lượng bộ nhớ đáng

kể, gây áp lực lên hệ thống, đặc biệt là khi triển khai trên máy chủ có cấu hình thấp.

- Thiếu công cụ quản lý và thống kê: MongoDB thiếu một số công cụ quản lý và thống kê tích hợp mạnh mẽ so với một số hệ quản trị cơ sở dữ liệu quan hệ. Điều này có thể làm cho quá trình giám sát và tối ưu hóa hiệu suất trở nên khó khăn hơn.

- Khó khăn khi thực hiện những truy vấn phức tạp: Mặc dù MongoDB mạnh mẽ khi thực hiện các truy vấn cơ bản, nhưng nếu bạn cần thực hiện các truy vấn phức tạp và lớn, nó có thể gặp khó khăn và yêu cầu việc quản lý chỉ mục một cách cẩn thận.[3]

2.4. Tìm hiểu về NodeJS.

2.4.1. *Tổng quan về NodeJS.*

- Node.js là một môi trường runtime mã nguồn mở và đa nền tảng, cho phép thực thi mã JavaScript bên ngoài trình duyệt web, điều này tạo ra sự linh hoạt lớn trong phát triển ứng dụng. Node.js được xây dựng trên nền tảng công cụ V8 JavaScript Engine của Google Chrome, nổi bật với khả năng xử lý non-blocking (không chặn) và mô hình event-driven (dựa trên sự kiện). Những đặc điểm này giúp Node.js tối ưu hóa hiệu suất và khả năng mở rộng của các ứng dụng mạng, đặc biệt trong các tình huống yêu cầu xử lý đồng thời nhiều kết nối mà không gặp phải tắc nghẽn.

- Một trong những ưu điểm nổi bật của Node.js là khả năng xử lý hàng nghìn kết nối đồng thời trên một luồng đơn mà không làm giảm hiệu suất. Điều này là nhờ vào cơ chế xử lý sự kiện và các thao tác bất đồng bộ, giúp các ứng dụng mạng có thể quản lý nhiều kết nối cùng một lúc mà không bị cản trở bởi các tác vụ đang chạy. Chính vì lý do này, Node.js trở thành lựa chọn lý tưởng cho các ứng dụng thời gian thực, bao gồm các hệ thống chat, game online, và các dịch vụ API cần phản hồi nhanh và liên tục.

- Node.js không chỉ đơn thuần là một môi trường runtime mà còn đi kèm với NPM (Node Package Manager), một hệ thống quản lý gói mạnh mẽ. NPM cung cấp một kho lưu trữ khổng lồ các thư viện và module mà các nhà phát triển có thể dễ dàng cài đặt, quản lý và chia sẻ. Điều này giúp đơn giản hóa quá trình phát triển và mở rộng các ứng dụng Node.js bằng cách cung cấp các công cụ và giải pháp đã được kiểm chứng sẵn có, từ các thư viện hỗ trợ HTTP, kết nối cơ sở dữ liệu, đến các công cụ phát triển và kiểm thử.

- Với sự hỗ trợ mạnh mẽ từ cộng đồng và khả năng tích hợp với nhiều công nghệ khác, Node.js không chỉ phù hợp với các ứng dụng thời gian thực mà còn được sử dụng

rộng rãi trong việc xây dựng các API RESTful, các dịch vụ web, và các công cụ phát triển phần mềm hiện đại. Điều này chứng tỏ sự linh hoạt và sức mạnh của Node.js trong việc đáp ứng nhu cầu phát triển ứng dụng trong thời đại số.[5]

2.4.2. *Lịch sử hình thành của NodeJS.*

- Node.js, được phát triển vào năm 2009 dưới sự dẫn dắt của Ryan Dahl, là một môi trường runtime mã nguồn mở được thiết kế để giải quyết các hạn chế về hiệu suất của các máy chủ web truyền thống như Apache HTTP Server. Ryan Dahl nhận thấy rằng các máy chủ này gặp khó khăn trong việc xử lý nhiều kết nối đồng thời do mô hình xử lý blocking (chặn) và tiêu tốn tài nguyên cao. Để khắc phục vấn đề này, Dahl đã chọn công cụ V8 JavaScript Engine của Google Chrome làm nền tảng cho Node.js, nhờ vào hiệu suất xử lý cao và khả năng tương thích tốt với JavaScript – ngôn ngữ lập trình phổ biến trong phát triển web.

- Phiên bản đầu tiên của Node.js đã thu hút sự chú ý của cộng đồng lập trình viên nhờ khả năng xử lý hiệu quả các tác vụ I/O và dễ dàng xây dựng các ứng dụng mạng mở rộng. Vào năm 2011, công ty công nghệ Joyent đã nhận thấy tiềm năng của Node.js và thuê Ryan Dahl để tiếp tục phát triển nó. Sự hỗ trợ từ Joyent đã giúp Node.js cải tiến và mở rộng, đồng thời thu hút sự quan tâm từ cộng đồng mã nguồn mở.

- Một cột mốc quan trọng là việc thành lập Node.js Foundation vào năm 2015, nhằm thúc đẩy sự phát triển độc lập và bền vững của Node.js. Vào năm 2019, Node.js Foundation đã hợp nhất với JS Foundation để thành lập OpenJS Foundation, một tổ chức quản lý mạnh mẽ hơn cho các dự án JavaScript mã nguồn mở. Node.js đã chứng tỏ giá trị của mình qua việc được sử dụng rộng rãi bởi các công ty lớn như Netflix, LinkedIn và Walmart. Với một hệ sinh thái phong phú và cộng đồng phát triển mạnh mẽ, Node.js tiếp tục khẳng định vai trò quan trọng trong phát triển ứng dụng web và mạng. [5]

2.4.3. *Ưu điểm của NodeJS.*

Node.js nổi bật với nhiều ưu điểm đáng chú ý, làm cho nó trở thành lựa chọn hàng đầu cho phát triển ứng dụng web và mạng:

- Hiệu suất cao: Node.js sử dụng V8 JavaScript Engine của Google, cho phép thực thi mã JavaScript với tốc độ nhanh chóng. Điều này giúp các ứng dụng Node.js xử lý các yêu cầu hiệu quả và nhanh chóng.

- Khả năng mở rộng: Node.js hỗ trợ xây dựng các ứng dụng mở rộng dễ dàng nhờ vào clustering module, giúp khai thác tối đa khả năng của các CPU đa nhân và cải thiện hiệu suất tổng thể.

- Mô hình non-blocking và event-driven: Node.js hoạt động dựa trên mô hình non-blocking I/O và event-driven, cho phép xử lý nhiều yêu cầu đồng thời mà không bị chặn. Điều này đặc biệt hữu ích cho các ứng dụng thời gian thực như chat, game online và dịch vụ phát trực tuyến.

- Cộng đồng mạnh mẽ và tài liệu phong phú: Node.js được hỗ trợ bởi một cộng đồng phát triển sôi nổi và nhiều tài liệu phong phú. Điều này giúp các nhà phát triển dễ dàng tiếp cận, học hỏi và phát triển ứng dụng.

- Khả năng xử lý JSON hiệu quả: Node.js, với JavaScript là ngôn ngữ chính, xử lý dữ liệu JSON rất hiệu quả và thuận tiện, phù hợp với sự phổ biến của JSON trong trao đổi dữ liệu.

- Tính linh hoạt: Node.js cho phép xây dựng nhiều loại ứng dụng khác nhau một cách hiệu quả và dễ dàng. Nó hỗ trợ nhiều hệ điều hành như Windows, Linux, và macOS, giúp triển khai ứng dụng trên nhiều môi trường mà không cần thay đổi mã nguồn.[5]

2.4.4. *Nhược điểm của NodeJS.*

Mặc dù Node.js có nhiều ưu điểm, nó cũng có một số nhược điểm cần lưu ý:

- Không phù hợp cho các tác vụ nặng về CPU: Node.js không phải là lựa chọn tốt nhất cho các ứng dụng cần xử lý tính toán phức tạp hoặc tác vụ nặng về CPU, như xử lý hình ảnh, video hoặc các thuật toán máy học, vì môi trường đơn luồng của nó có thể làm giảm hiệu suất chung.

- Khả năng mở rộng: Mặc dù Node.js có thể xử lý nhiều kết nối đồng thời, việc mở rộng ứng dụng để xử lý các yêu cầu nặng về tài nguyên có thể phức tạp và yêu cầu nhiều kiến thức về kiến trúc hệ thống.

- Quản lý Dependency khó khăn: Node.js thường phụ thuộc vào nhiều gói thư viện, làm tăng độ phức tạp trong việc cập nhật và duy trì. Các vấn đề tương thích phiên bản có thể phát sinh khi cập nhật các gói, và các dependency không trực tiếp (transitive dependencies) có thể tạo ra một cây phụ thuộc khó theo dõi.

- Thay đổi liên tục trong API: Các API của gói thư viện có thể thay đổi nhanh chóng và không nhất quán giữa các phiên bản, gây ra vấn đề tương thích và làm gián đoạn quá trình phát triển. Những thay đổi này có thể yêu cầu điều chỉnh mã nguồn hoặc thay đổi toàn bộ kiến trúc ứng dụng.

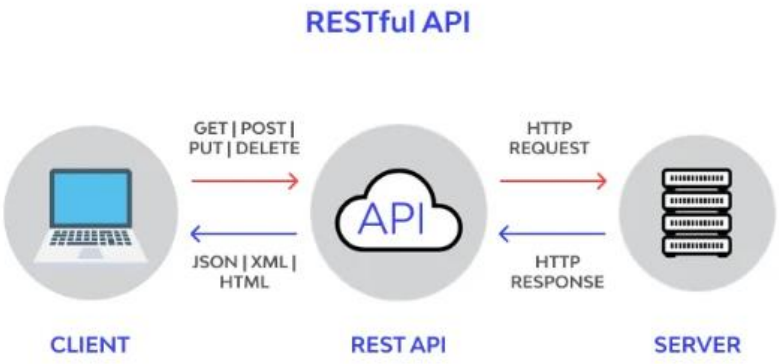
- Xử lý công việc tính toán nặng kém: Node.js có thể gặp vấn đề về hiệu suất khi xử lý các tác vụ tính toán phức tạp và tiêu tốn nhiều tài nguyên CPU, dẫn đến việc ứng dụng trở nên chậm chạp hoặc không phản hồi kịp thời.

- Vấn đề bảo mật: Quản lý các gói và phiên bản trong Node.js có thể trở nên phức tạp, và các lỗ hổng bảo mật trong các gói phụ thuộc có thể gây ra rủi ro bảo mật nếu không được cập nhật kịp thời. [5]

2.5. Tìm hiểu về Resful API.

2.5.1. *Tổng quan về Resful API.*

RESTful API (Representational State Transfer Application Programming Interface) là một phong cách kiến trúc được sử dụng để thiết kế các dịch vụ web. RESTful API tuân theo các nguyên tắc của REST, một kiến trúc do Roy Fielding giới thiệu trong luận án tiến sĩ của ông vào năm 2000. Các dịch vụ web RESTful cho phép các hệ thống khác nhau giao tiếp với nhau thông qua giao thức HTTP, sử dụng các phương thức tiêu chuẩn như GET, POST, PUT, DELETE.[8]



Hình 2.2. Mô hình trang web sử dụng Restful API [8]

2.5.2. *Các thành phần của Resful API.*

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 người dùng 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,… đến một URL để xử lý dữ liệu.[8]

2.5.3. *Các phương thức của Resful API.*

RESTful API sử dụng các phương thức HTTP tiêu chuẩn để thực hiện các thao tác trên tài nguyên. Dưới đây là các phương thức chính của RESTful API:

- [GET]: sử dụng để truy xuất thông tin về một tài nguyên hoặc một tập hợp tài nguyên từ máy chủ. Nó không thay đổi trạng thái của tài nguyên và được coi là phương thức "đọc".
- [POST]: Được sử dụng để tạo mới một tài nguyên. Thông thường, dữ liệu gửi kèm theo yêu cầu POST được sử dụng để tạo mới tài nguyên trên máy chủ.
- [PUT]: Sử dụng để cập nhật một tài nguyên hiện có hoặc tạo một tài nguyên mới nếu nó chưa tồn tại. Dữ liệu cập nhật được gửi trong phần thân của yêu cầu.
- [DELETE]: Dùng để xóa một tài nguyên. Yêu cầu DELETE sẽ gửi thông điệp đến máy chủ để yêu cầu xóa tài nguyên được xác định.
- [PATCH]: sử dụng để cập nhật một phần của tài nguyên hiện có. Nó cho phép thay đổi một hoặc một số thuộc tính của tài nguyên mà không cần gửi toàn bộ dữ liệu.
- [OPTIONS]: Sử dụng để lấy thông tin về các phương thức HTTP mà máy chủ hỗ trợ cho một URL cụ thể. Nó thường được sử dụng trong các yêu cầu kiểm tra trước (preflight requests) của CORS.
- [HEAD]: Tương tự như GET nhưng không trả về phần thân của phản hồi. Nó chỉ trả về các tiêu đề HTTP. Điều này hữu ích để kiểm tra thông tin về tài nguyên mà không cần tải toàn bộ tài nguyên.[8]

2.5.4. *Các trạng thái của Resful API.*

RESTful API sử dụng mã trạng thái HTTP (HTTP status codes) để cung cấp thông tin về kết quả của các yêu cầu. Mã trạng thái HTTP là một phần quan trọng trong giao tiếp giữa máy khách và máy chủ, giúp xác định xem yêu cầu đã thành công, gặp lỗi hay cần thêm hành động. Dưới đây là các nhóm mã trạng thái chính và một số mã trạng thái phổ biến:
Mã Trạng Thái 1xx (Thông tin - Informational Responses):

- 100 Continue: Máy chủ đã nhận được phần đầu của yêu cầu và khách hàng nên tiếp tục gửi phần thân của yêu cầu.
- 101 Switching Protocols: Máy chủ đang chuyển đổi giao thức theo yêu cầu của khách hàng.

Mã Trạng Thái 2xx (Thành công - Successful Responses):

- 200 OK: Yêu cầu đã thành công và máy chủ trả về tài nguyên (thường dùng với GET).
- 201 Created: Yêu cầu đã thành công và tài nguyên mới đã được tạo (thường dùng với POST hoặc PUT).
- 202 Accepted: Yêu cầu đã được chấp nhận để xử lý, nhưng xử lý chưa hoàn thành.
- 204 No Content: Yêu cầu đã thành công nhưng không có nội dung nào để trả về (thường dùng với DELETE hoặc PUT).

Mã Trạng Thái 3xx (Chuyển hướng - Redirection Messages):

- 301 Moved Permanently: Tài nguyên đã được di chuyển vĩnh viễn đến một URL mới.
- 302 Found: Tài nguyên tạm thời được di chuyển đến một URL mới.
- 304 Not Modified: Tài nguyên chưa được thay đổi kể từ lần cuối cùng nó được truy xuất.

Mã Trạng Thái 4xx (Lỗi từ phía khách hàng - Client Error Responses):

- 400 Bad Request: Yêu cầu không hợp lệ hoặc bị lỗi cú pháp.
- 401 Unauthorized: Yêu cầu yêu cầu xác thực người dùng.
- 403 Forbidden: Máy chủ hiểu yêu cầu nhưng từ chối cho phép thực hiện.
- 404 Not Found: Tài nguyên được yêu cầu không tồn tại trên máy chủ.
- 405 Method Not Allowed: Phương thức HTTP được sử dụng không được hỗ trợ cho tài nguyên được yêu cầu.
- 409 Conflict: Có xung đột với trạng thái hiện tại của tài nguyên.

Mã Trạng Thái 5xx (Lỗi từ phía máy chủ - Server Error Responses):

- 500 Internal Server Error: Máy chủ gặp lỗi và không thể hoàn thành yêu cầu.
- 501 Not Implemented: Máy chủ không hỗ trợ phương thức yêu cầu.
- 502 Bad Gateway: Máy chủ nhận phản hồi không hợp lệ từ một máy chủ ngược dòng.
- 503 Service Unavailable: Máy chủ hiện không thể xử lý yêu cầu (do quá tải hoặc bảo trì).
- 504 Gateway Timeout: Máy chủ không nhận được phản hồi kịp thời từ máy chủ ngược dòng.[8]

2.5.5. *Ưu điểm của ResfulAPI.*

Restful API (Representational State Transfer API) là một kiến trúc được thiết kế để xây dựng các dịch vụ web có khả năng tương tác một cách đơn giản và hiệu quả. Dưới đây là một số ưu điểm quan trọng của Restful API trong việc phát triển ứng dụng web:

- Thiết kế và Triển khai Dễ Dàng: RESTful API sử dụng các phương thức HTTP chuẩn như GET, POST, PUT, DELETE, PATCH, OPTIONS, và HEAD. Các phương thức này đều có nghĩa rõ ràng và tuân theo các quy tắc HTTP, giúp các nhà phát triển dễ dàng thiết kế và triển khai API. Mỗi phương thức tương ứng với một loại hành động cụ thể, làm cho việc đọc và hiểu tài liệu API trở nên trực quan hơn.
- Cấu Trúc URL Rõ Ràng: RESTful API yêu cầu các tài nguyên phải được định danh thông qua các URL có cấu trúc rõ ràng và có thể đoán trước. Ví dụ, một URL như /users/{userId} cho phép dễ dàng hiểu rằng nó liên quan đến thông tin của người dùng cụ thể. Cấu trúc này không chỉ giúp trong việc phát triển mà còn làm giảm sự nhầm lẫn khi sử dụng API.
- Sử Dụng Tiêu Chuẩn HTTP: RESTful API tuân theo các tiêu chuẩn HTTP, giúp nó trở nên tương thích với nhiều hệ thống và dịch vụ khác nhau. Điều này làm cho việc tích hợp với các dịch vụ bên ngoài hoặc hệ thống nội bộ trở nên trơn tru hơn, vì nhiều công cụ và thư viện đã được tối ưu hóa để làm việc với HTTP.
- Định Dạng Dữ Liệu Phổ Biến: RESTful API thường sử dụng JSON hoặc XML làm định dạng dữ liệu, cả hai đều rất phổ biến và dễ dàng tích hợp với nhiều công nghệ và ngôn ngữ lập trình khác nhau. JSON, đặc biệt, là định dạng dữ liệu nhẹ và dễ đọc, điều này giúp tăng tốc quá trình phát triển và giảm thiểu sự phức tạp trong việc chuyển đổi dữ liệu.
- Tái Sử Dụng Endpoint: Các endpoint trong RESTful API được thiết kế để phục vụ nhiều mục đích khác nhau và có thể được tái sử dụng cho nhiều ứng dụng hoặc dịch vụ khác nhau. Ví dụ, một endpoint như /products có thể được sử dụng để lấy danh sách sản phẩm, thêm sản phẩm mới, hoặc cập nhật thông tin sản phẩm, tùy thuộc vào phương thức HTTP được sử dụng.
- Tính Mở Rộng và Bảo Trì: Việc tổ chức các endpoint theo cách rõ ràng và nhất quán giúp việc mở rộng và bảo trì hệ thống trở nên dễ dàng hơn. Các thay đổi có thể được thực hiện mà không làm ảnh hưởng đến các phần còn lại của hệ thống, nhờ vào việc phân chia các chức năng và tài nguyên thành các endpoint riêng biệt.

- Tiêu Chuẩn Bảo Mật HTTP: RESTful API có thể tận dụng các tiêu chuẩn bảo mật của HTTP, chẳng hạn như HTTPS để mã hóa dữ liệu trong quá trình truyền tải, giúp bảo vệ dữ liệu khỏi bị rò rỉ hoặc đánh cắp. HTTPS đảm bảo rằng dữ liệu được truyền giữa máy khách và máy chủ là an toàn và bảo mật.
- Xác Thực và Cấp Quyền: RESTful API hỗ trợ các cơ chế xác thực mạnh mẽ như OAuth và JWT (JSON Web Tokens). OAuth cho phép cấp quyền truy cập cho các ứng dụng của bên thứ ba mà không cần phải tiết lộ thông tin xác thực của người dùng. JWT cung cấp một cách để xác thực và truyền thông tin một cách an toàn giữa các dịch vụ và hệ thống, giúp đảm bảo rằng chỉ những người dùng hoặc ứng dụng hợp lệ mới có quyền truy cập vào các tài nguyên. [8]

2.5.6. *Nhược điểm của ResfulAPI.*

- Hiệu suất thấp với các thao tác phức tạp: RESTful API có thể gặp khó khăn trong việc xử lý các thao tác phức tạp hoặc nặng về tài nguyên. Do mô hình truyền tải dữ liệu qua HTTP, các yêu cầu liên quan đến xử lý phức tạp hoặc khối lượng dữ liệu lớn có thể dẫn đến hiệu suất thấp hoặc thời gian phản hồi kéo dài.
- Thiếu quản lý trạng thái: RESTful API tuân theo nguyên tắc stateless (không duy trì trạng thái), điều này có thể gây khó khăn trong việc quản lý trạng thái người dùng và phiên làm việc. Các yêu cầu phải gửi tất cả thông tin cần thiết trong từng yêu cầu HTTP, điều này có thể làm phức tạp hóa việc quản lý và bảo mật thông tin phiên.
- Rủi ro bảo mật: RESTful API có thể dễ bị tổn thương nếu không áp dụng các biện pháp bảo mật đúng cách. Các lỗ hổng trong việc xác thực và phân quyền có thể dẫn đến rủi ro bảo mật, đặc biệt khi sử dụng HTTP không mã hóa hoặc khi các API không được bảo vệ đầy đủ bằng các phương pháp như HTTPS hoặc OAuth. [8]

2.6. Tìm hiểu Framework ReactJS.

2.6.1. *Tổng quan về ReactJS.*

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, dùng để xây dựng giao diện người dùng (UI) cho các ứng dụng web. React cho phép các nhà phát triển tạo ra các thành phần UI có thể tái sử dụng (reusable components), giúp quản lý và phát triển các giao diện phức tạp một cách dễ dàng và hiệu quả. Một trong những đặc điểm nổi bật của React là Virtual DOM (Document Object Model) - một cơ chế giúp tối ưu hóa việc cập nhật giao diện bằng cách chỉ thay đổi những phần cần thiết thay vì toàn bộ trang.

React hỗ trợ một cách tiếp cận hướng thành phần, cho phép xây dựng các phần nhỏ của giao diện người dùng rồi kết hợp chúng lại thành các phần phức tạp hơn. Điều này không chỉ giúp tối ưu hóa hiệu suất mà còn giúp cải thiện khả năng bảo trì và mở rộng mã nguồn. Nhờ vào tính linh hoạt và hiệu quả, ReactJS đã trở thành một trong những thư viện phổ biến nhất trong việc phát triển ứng dụng web hiện đại.[6]

2.6.2. *Lịch sử hình thành ReactJS.*

ReactJS được tạo ra bởi Jordan Walke, một kỹ sư phần mềm tại Facebook, và chịu ảnh hưởng từ XHP (một nền tảng thành phần HTML cho PHP). React lần đầu tiên được triển khai cho ứng dụng Newsfeed của Facebook vào năm 2011 và sau đó cho Instagram vào năm 2012. React được mã nguồn mở tại JSConf US vào tháng 5 năm 2013.

React Native, phiên bản của React dành cho phát triển ứng dụng di động, cho phép phát triển ứng dụng Android, iOS và UWP (Universal Windows Platform) gốc với React, được công bố tại React Conf của Facebook vào tháng 2 năm 2015 và mã nguồn mở vào tháng 3 năm 2015.

Vào ngày 18 tháng 4 năm 2017, Facebook công bố React Fiber, một thuật toán nội bộ mới để kết xuất, thay thế thuật toán kết xuất cũ của React, Stack. React Fiber đã trở thành nền tảng cho mọi cải tiến và phát triển tính năng trong tương lai của thư viện React. Cú pháp lập trình thực tế với React không thay đổi, chỉ có cách thực thi cú pháp thay đổi. Hệ thống kết xuất cũ, Stack, được phát triển trong bối cảnh không tập trung vào thay đổi động, chẳng hạn như xử lý hoạt ảnh phức tạp trong một đoạn duy nhất. Fiber chia hoạt ảnh thành các phân đoạn có thể trải rộng trên nhiều khung hình, cho phép hiển thị mượt mà hơn.

Ngày 26 tháng 9 năm 2017, React 16.0 được phát hành ra công chúng.

Ngày 10 tháng 8 năm 2020, nhóm React công bố ứng cử viên phát hành đầu tiên cho React v17.0, đáng chú ý là bản phát hành chính đầu tiên không có thay đổi lớn đối với API dành cho nhà phát triển React.

Ngày 29 tháng 3 năm 2022, React 18 được phát hành, giới thiệu trình kết xuất đồng thời mới, tạo nhóm tự động và hỗ trợ kết xuất phía máy chủ với Suspense.

2.6.3. *Ưu điểm của ReactJS.*

- Virtual DOM: React sử dụng một phiên bản ảo của DOM để quản lý và tối ưu hóa các thay đổi trong giao diện người dùng. Thay vì cập nhật toàn bộ DOM mỗi khi có thay

đổi, React chỉ cập nhật các phần tử cần thiết, giúp tăng hiệu suất và giảm tải cho trình duyệt. Cơ chế này giúp tăng tốc độ phản hồi và cải thiện trải nghiệm người dùng.

- JSX: React sử dụng JSX, một phần mở rộng cú pháp cho JavaScript, cho phép viết mã HTML trong JavaScript. Điều này giúp mã nguồn dễ đọc và dễ hiểu hơn, đặc biệt là đối với những người mới bắt đầu. JSX cũng cho phép kết hợp logic JavaScript với cấu trúc HTML, tạo điều kiện thuận lợi cho việc phát triển giao diện phức tạp.
- Hệ sinh thái mạnh mẽ: ReactJS không chỉ là một thư viện JavaScript mạnh mẽ để xây dựng giao diện người dùng, mà còn sở hữu một hệ sinh thái rộng lớn và phong phú. Hệ sinh thái này bao gồm nhiều thư viện, công cụ và tiện ích hỗ trợ các nhà phát triển trong việc xây dựng các ứng dụng web hiện đại và phức tạp
- Cộng đồng lớn và hỗ trợ tốt: React có một cộng đồng lớn và sự hỗ trợ mạnh mẽ từ Facebook cũng như cộng đồng người dùng toàn cầu. Điều này đảm bảo rằng có nhiều tài liệu, ví dụ, và các nguồn tài nguyên khác sẵn có để giúp các lập trình viên giải quyết vấn đề và tối ưu hóa ứng dụng của họ. Các diễn đàn và nhóm thảo luận trực tuyến cũng rất sôi động và hữu ích.
- Cập nhật và bảo trì dễ dàng: Tính năng hot reloading của React cho phép cập nhật mã nguồn và xem thay đổi ngay lập tức mà không cần tải lại toàn bộ trang. Điều này giúp tăng tốc độ phát triển và kiểm thử. React cũng duy trì sự tương thích ngược, giúp dễ dàng nâng cấp phiên bản mà không gặp nhiều vấn đề về tương thích.
- Mã nguồn mở: ReactJS là một thư viện mã nguồn mở, cho phép cộng đồng đóng góp và cải thiện liên tục. Điều này giúp cho những người mới bắt đầu có thể dễ dàng tiếp cận và học hỏi từ các dự án mẫu và tài liệu cộng đồng.
- Quản lý trạng thái hiệu quả: React quản lý trạng thái (state) và thuộc tính (props) một cách hiệu quả, giúp việc theo dõi và cập nhật giao diện dễ dàng hơn. Các công cụ như React Context và các thư viện quản lý trạng thái bên ngoài như Redux và MobX càng làm cho việc quản lý trạng thái trở nên mạnh mẽ và linh hoạt hơn. [6]

2.6.4. *Nhược điểm của ReactJS.*

- Tích hợp JSX: Một trong những nhược điểm của ReactJS là việc hầu hết mã nguồn được viết dưới dạng JSX, nơi HTML và CSS được tích hợp trực tiếp trong JavaScript. Điều này khác biệt so với các framework khác, nơi HTML và CSS thường được tách biệt rõ ràng. Sự kết hợp này có thể khiến những người mới làm quen với ReactJS cảm thấy lúng

túng và dễ nhầm lẫn giữa JSX và HTML thông thường. Việc học và áp dụng JSX đòi hỏi một khoảng thời gian nhất định để làm quen và thành thạo.

Dung lượng file phát triển lớn: Một nhược điểm khác của ReactJS là dung lượng của các file trong giai đoạn phát triển có thể khá lớn. Điều này có thể làm tăng thời gian tải trang và giảm hiệu suất, đặc biệt là đối với các ứng dụng lớn và phức tạp. Việc xử lý và tối ưu hóa mã nguồn để giảm kích thước file trong quá trình phát triển đòi hỏi thêm công sức và kỹ thuật từ phía nhà phát triển. Những yếu tố này có thể làm tăng độ phức tạp và thời gian cần thiết để triển khai và duy trì các dự án sử dụng ReactJS.

Thiếu một cách tiếp cận toàn diện: ReactJS chỉ là một thư viện cho việc xây dựng giao diện người dùng, do đó, nó không cung cấp một cách tiếp cận toàn diện cho phát triển ứng dụng web. Các nhà phát triển thường phải kết hợp ReactJS với các thư viện và công cụ khác để quản lý routing, state management, và các chức năng khác, điều này có thể dẫn đến một cấu trúc dự án phức tạp và khó duy trì. [6]

CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Mô tả bài toán.

Nghiệp vụ quản lý bao gồm quản lý địa điểm du lịch, nơi lưu trú,các địa điểm ăn uống, đặc sản, dịch vụ, quà lưu niệm. Quá trình quản lý bắt đầu bằng việc nhập thông tin của các địa điểm đã chuẩn bị và tìm hiểu sẵn. Điều này bao gồm cập nhật các thông tin chi tiết như là: tên địa điểm, mô tả, địa chỉ, hình ảnh, loại hình du lịch và tọa độ. Còn các nơi lưu trú sẽ là: tên nơi lưu trú, giá, địa chỉ, số điện thoại, mô tả, hình ảnh và tọa độ. Các địa điểm ăn uống bao gồm: tên địa điểm ăn uống, giá, địa chỉ, số điện thoại, mô tả, hình ảnh và tọa độ. Các đặc sản bao gồm: tên đặc sản, giá, mô tả, xuất xứ, hạn sử dụng, hình ảnh. Dịch vụ bao gồm: tên dịch vụ, giá, mô tả, hình ảnh. Quà lưu niệm bao gồm: tên dịch vụ, giá, mô tả, hình ảnh. Quá trình nhập liệu này là quan trọng để đảm bảo rằng cơ sở dữ liệu của website được duy trì một cách chính xác và đầy đủ.

3.2. Yêu cầu chức năng.

Để xây dựng một trang web quảng bá du lịch các tỉnh Tây Nam Bộ, cần một số yêu cầu chức năng cụ thể để đáp ứng nhu cầu người dùng:

- Hiển thị tất cả địa điểm du lịch,

- Lọc các địa điểm du lịch theo danh mục,

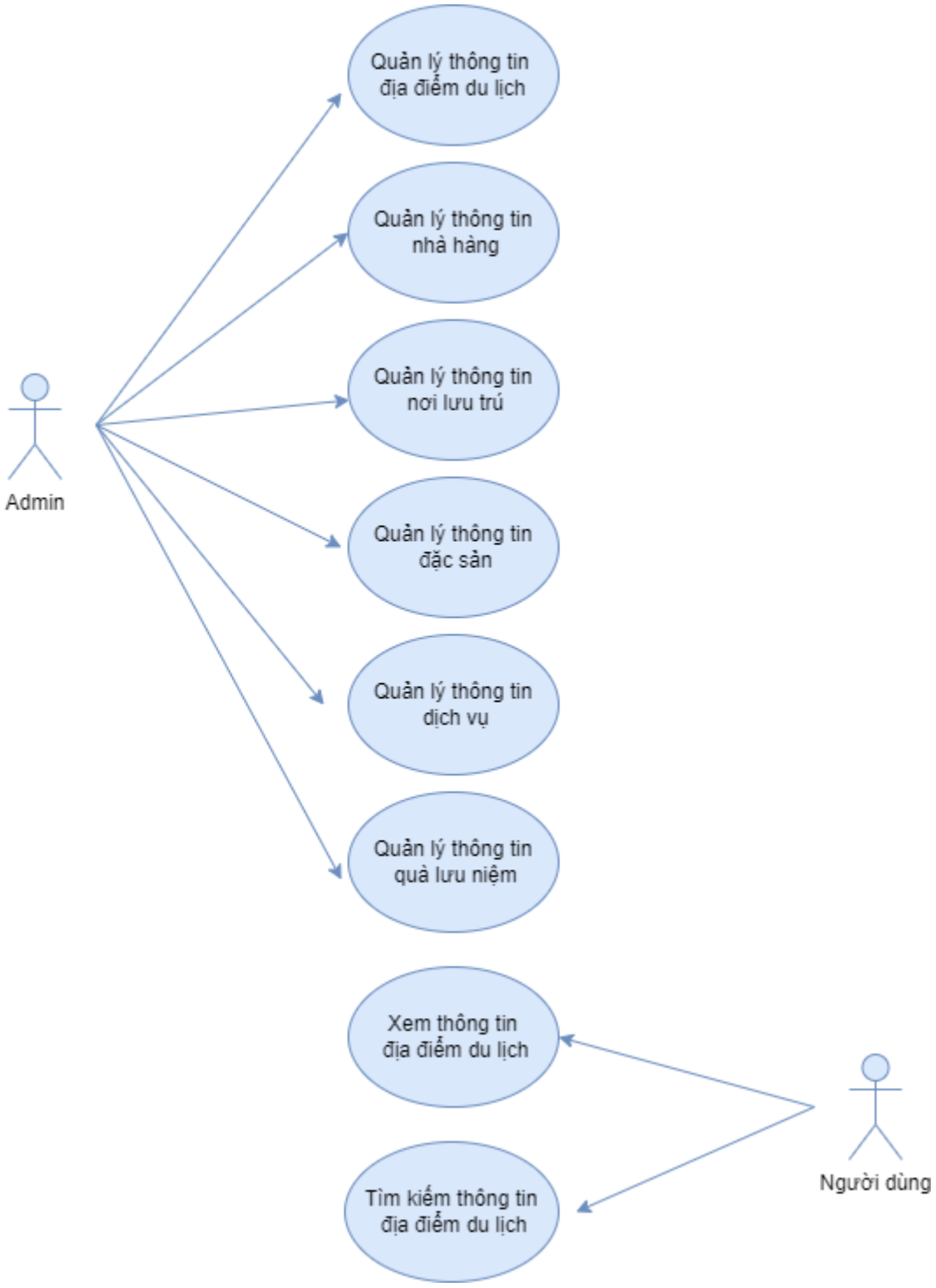
- Tìm kiếm địa điểm du lịch theo tên: Người dùng có thể tìm kiếm các địa điểm du lịch thuộc các tỉnh trong khu vực Tây Nam Bộ,

- Tìm kiếm địa điểm du lịch theo địa chỉ: Người dùng có thể tìm kiếm các địa điểm du lịch thuộc một tỉnh thành cụ thể trong khu vực Tây Nam Bộ,

- Xem thông tin chi tiết: Cung cấp thông tin chi tiết về mỗi địa điểm bao gồm: tên địa điểm, loại hình du lịch, mô tả, hình ảnh, địa chỉ và các phần liên quan đến địa điểm du lịch đó như: nơi lưu trú, nhà hàng, đặc sản, dịch vụ, quà lưu niệm,

- Bản đồ và vị trí: Hiển thị bản đồ với vị trí của các địa điểm du lịch các tỉnh của khu vực Tây Nam Bộ, cung cấp khả năng định vị các địa điểm,

3.3. Đặc tả hệ thống.



Hình 3.1. Sơ đồ Use-case

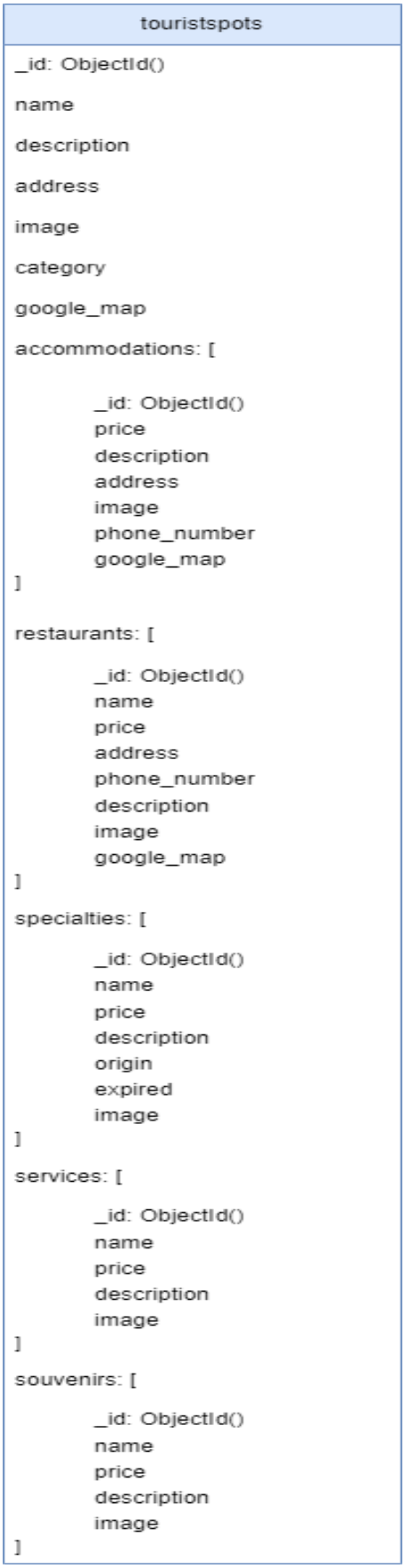
Bảng 3.1 Mô tả Actor

STT	Tên Actor	Ý nghĩa
1	Người dùng	Người dùng hệ thống
2	Admin	Người quản trị hệ thống

Bảng 3.2 Mô tả Use-case

STT	Tên Use-case	Ý Nghĩa
1	Quản lý thông tin địa điểm du lịch	Quản trị viên có thể quản lý thông tin của các địa điểm trên trang web và thêm, sửa, xoá nếu cần thiết.
2	Quản lý thông tin nơi lưu trú	Quản trị viên có thể quản lý thông tin của nơi lưu trú trên trang web và thêm, sửa, xoá nếu cần thiết.
3	Quản lý thông tin nhà hàng	Quản trị viên có thể quản lý thông tin của nhà hàng trang trên web và thêm, sửa, xoá nếu cần thiết.
4	Quản lý thông tin đặc sản	Quản trị viên có thể quản lý thông tin của đặc sản trên trang web và thêm, sửa, xoá nếu cần thiết.
5	Quản lý thông tin dịch vụ	Quản trị viên có thể quản lý thông tin của dịch vụ trên trang web và thêm, sửa, xoá nếu cần thiết.
6	Quản lý thông tin quà lưu niệm	Quản trị viên có thể quản lý thông tin của quà lưu niệm trên trang web và thêm, sửa, xoá nếu cần thiết.
7	Xem thông tin địa điểm du lịch	Người dùng có thể xe thông tin của các địa điểm có trên hệ thống và các phần liên quan đến địa điểm du lịch đó như: nơi lưu trú, nhà hàng, đặc sản, dịch vụ, quà lưu niệm.
8	Tìm kiếm thông tin địa điểm du lịch	Người dùng có thể tìm kiếm các địa điểm du lịch có trên hệ thống, có thể tìm kiếm theo tên, tìm kiếm địa chỉ địa điểm.

3.4. Lược đồ dữ liệu.



Hình 3.2. Cấu trúc document mô tả thông tin địa điểm du lịch

Bảng 3.3 Mô tả Colletion “touristspots”

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa		
1	_id	ObjectId (MongoDB)	Mã của địa điểm du lịch		
2	name	String	Tên của địa điểm du lịch		
3	description	String	Mô tả của địa điểm du lịch		
4	address	String	Địa chỉ của địa điểm du lịch		
5	image	String	Hình ảnh của địa điểm du lịch		
6	category	String	Danh mục của địa điểm		
7	google_map	String	Tọa độ của địa điểm		
8	accommodations: [ ]	**STT**	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ý nghĩa**
		1	_id	ObjectId (MongoDB)	Mã của nơi lưu trú
		2	name	String	Tên của nơi lưu trú
		3	price	String	Giá của nơi lưu trú
		4	address	String	Địa chỉ của nơi lưu trú
		5	phone_number	String	Số điện thoại của nơi lưu trú
		6	description	String	Mô tả của nơi lưu trú
		7	image	String	Hình ảnh của nơi lưu trú
		8	google_map	String	Tọa độ của nơi lưu trú

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa		
9	Restaurants: [ ]	**STT**	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ý nghĩa**
		1	_id	ObjectId (MongoDB)	Mã của nhà hàng
		2	name	String	Tên của nhà hàng
		3	price	String	Giá của nhà hàng
		4	address	String	Địa chỉ của nhà hàng
		5	phone_number	String	Số điện thoại của nhà hàng
		6	description	String	Mô tả của nhà hàng
		7	image	String	Hình ảnh của nhà hàng
		8	google_map	String	Tọa độ của nhà hàng
10	Specialties: [ ]	**STT**	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ý nghĩa**
		1	_id	ObjectId (MongoDB)	Mã của đặc sản
		2	name	String	Tên của đặc sản
		3	price	String	Giá của đặc sản
		4	description	String	Mô tả của đặc sản
		5	origin	String	Xuất xứ của đặc sản
		6	expired	String	Hạn sử dụng của đặc sản
		7	image	String	Hình ảnh của đặc sản

STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa		
11	Services: [ ]	**STT**	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ý nghĩa**
		1	_id	ObjectId (MongoDB)	Mã của dịch vụ
		2	name	String	Tên của dịch vụ
		3	price	String	Giá của dịch vụ
		4	description	String	Mô tả của dịch vụ
		5	image	String	Hình ảnh của dịch vụ
12	Souvenirs: [ ]	**STT**	**Tên thuộc tính**	**Kiểu dữ liệu**	**Ý nghĩa**
		1	_id	ObjectId (MongoDB)	Mã của quà lưu niệm
		2	name	String	Tên của quà lưu niệm
		3	price	String	Giá của quà lưu niệm
		4	description	String	Mô tả của quà lưu niệm
		5	image	String	Hình ảnh của quà lưu niệm

3.5. Xây dựng Back-end.

3.5.1. *Định nghĩa model.*

Model này định nghĩa một cấu trúc dữ liệu cho các điểm du lịch sử dụng Mongoose, một thư viện ODM (Object Data Modeling) cho MongoDB. Model bao gồm một schema chi tiết cho các điểm du lịch, với các trường như tên, mô tả, địa chỉ, hình ảnh, danh mục, và trường google_map để lưu tọa độ, tất cả đều được yêu cầu. Ngoài ra, nó còn định nghĩa các thông tin chi tiết về các tiện nghi như chỗ ở, nhà hàng, đặc sản, dịch vụ, và quà lưu niệm liên quan đến mỗi điểm du lịch, mỗi phần đều có các trường riêng như tên, giá cả, địa chỉ, số điện thoại, mô tả, hình ảnh và trường google_map để lưu tọa độ. Schema này đảm bảo rằng tất cả các dữ liệu liên quan đến điểm du lịch đều được lưu trữ một cách có cấu trúc và nhất quán trong cơ sở dữ liệu MongoDB, đồng thời cung cấp các phương thức để tương tác với dữ liệu này thông qua Mongoose. Model này sẽ hỗ trợ cho việc quản lý và truy xuất dữ liệu một cách hiệu quả, phục vụ các chức năng của ứng dụng web liên quan đến thông tin du lịch.

```
const mongoose = require('mongoose');


const touristSpotSchema = new mongoose.Schema({
    name: { type: String, required: true },
    description: { type: String, required: true },
    address: { type: String, required: true },
    image: { type: String, required: true },
    category: { type: String, required: true },
    google_map: { type: String, required: true },
    accommodations: [{
        name: { type: String, required: true },
        price: { type: String, required: true },
        address: { type: String, required: true },
        phone_number: { type: String, required: true },
        description: { type: String, required: true },
        image: { type: String, required: true },
        google_map: { type: String, required: true }
    }],
```

```
    restaurants: [{
        name: { type: String, required: true },
        price: { type: String, required: true },
        address: { type: String, required: true },
        phone_number: { type: String, required: true },
        description: { type: String, required: true },
        image: { type: String, required: true },
        google_map: { type: String, required: true }
    }],
    specialties: [{
        name: { type: String, required: true },
        price: { type: String, required: true },
        description: { type: String, required: true },
        origin: { type: String, required: true },
        expired: { type: String, required: true },
        image: { type: String, required: true }
    }],
    services: [{
        name: { type: String, required: true },
        price: { type: String, required: true },
        description: { type: String, required: true },
        image: { type: String, required: true }
    }],
    souvenirs: [{
        name: { type: String, required: true },
        price: { type: String, required: true },
        description: { type: String, required: true },
        image: { type: String, required: true }
    }]
});
const TouristSpot = mongoose.model('TouristSpot', touristSpotSchema);
module.exports = { TouristSpot };
```

3.5.2. *Định nghĩa các Controller.*

Controller chứa các hàm xử lý riêng cho từng công việc, lấy ví dụ với Touristspots Controller.

- Controller thêm mới một địa điểm du lịch:

```
// Controller function để tạo mới một địa điểm du lịch
const createTouristSpot = async (req, res) => {
  const touristSpot = req.body;
  try {
    const createdTouristSpot = await TouristSpot.create(touristSpot);
    res.status(201).json(createdTouristSpot);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

- Controller cập nhật thông tin của một địa điểm du lịch theo ID:

```
// Controller function để cập nhật thông tin của một địa điểm du lịch
theo ID
const updateTouristSpotById = async (req, res) => {
  const { id } = req.params;
  const updatedTouristSpot = req.body;
  try {
    const existingTouristSpot = await TouristSpot.findById(id);
    if (!existingTouristSpot) {
      return res.status(404).json({ message: 'Không tìm thấy địa điểm
du lịch để cập nhật' });
    }
    // Bảo toàn các trường lồng nhau nếu chúng không được cung cấp
trong yêu cầu
    const updatedFields = {
      name: updatedTouristSpot.name || existingTouristSpot.name,
      description: updatedTouristSpot.description ||
existingTouristSpot.description,
      address: updatedTouristSpot.address ||
existingTouristSpot.address,
```

```
      image: updatedTouristSpot.image || existingTouristSpot.image,
      category: updatedTouristSpot.category ||
existingTouristSpot.category,
      google_map: updatedTouristSpot.google_map ||
existingTouristSpot.google_map,
      accommodations: updatedTouristSpot.accommodations.length ?
updatedTouristSpot.accommodations :
existingTouristSpot.accommodations,
      restaurants: updatedTouristSpot.restaurants.length ?
updatedTouristSpot.restaurants : existingTouristSpot.restaurants,
      specialties: updatedTouristSpot.specialties.length ?
updatedTouristSpot.specialties : existingTouristSpot.specialties,
      services: updatedTouristSpot.services.length ?
updatedTouristSpot.services : existingTouristSpot.services,
      souvenirs: updatedTouristSpot.souvenirs.length ?
updatedTouristSpot.souvenirs : existingTouristSpot.souvenirs
    };
    const result = await TouristSpot.findByIdAndUpdate(id,
updatedFields, { new: true });
    res.status(200).json(result);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

- Controller xóa một địa điểm du lịch theo ID:

```
// Controller function để xóa một địa điểm du lịch theo ID
const deleteTouristSpotById = async (req, res) => {
  const { id } = req.params;
  try {
    const result = await TouristSpot.findByIdAndDelete(id);
    if (!result) {
      return res.status(404).json({ message: 'Không tìm thấy địa điểm
du lịch để xóa' });
    }
    res.status(200).json({ message: 'Đã xóa địa điểm du lịch thành
công' });
  } catch (error) {
```

```
    res.status(500).json({ message: error.message });
  }
};
```

3.5.3. *Định nghĩ các Router.*

- touristSpotRouter (chứa các API xử lý việc thêm, xoá, sửa các địa điểm du lịch):

```
const router = express.Router();
// Route để lấy tất cả danh mục
router.get('/categories', getCategories);
// Route để tìm kiếm địa điểm du lịch theo tên
router.get('/search', searchTouristSpots);
// Route để tìm kiếm địa điểm du lịch theo danh mục
router.get('/tourist-spots/category/:category',
getTouristSpotsByCategory);
// Route để tìm kiếm địa điểm du lịch theo địa chỉ
router.get('/searchByAddress', searchTouristSpotsByAddress);
// Route để lấy ra tất cả địa điểm du lịch
router.get('/tourist-spots', getAllTouristSpots);
// Route để lấy ra địa điểm du lịch theo id
router.get('/tourist-spots/:id', getTouristSpotById);
// Route để thêm mới địa điểm du lịch
router.post('/tourist-spots', createTouristSpot);
// Route để sửa địa điểm du lịch theo id
router.put('/tourist-spots/:id', updateTouristSpotById);
// Route để xóa địa điểm du lịch theo id
router.delete('/tourist-spots/:id', deleteTouristSpotById);
```

- accommodationRouter (Chứa các API xử lý việc thêm, xoá, sửa các nơi lưu trú):

```
const router = express.Router();
// Route để lấy tất cả dữ liệu accommodations từ tất cả tourist spots
router.get('/all/accommodations', getAllAccommodationsFromAllSpots);
// Route để lấy tất cả dữ liệu accommodations của một tourist spot
router.get('/:touristSpotId/accommodations', getAllAccommodations);
// Route để lấy dữ liệu accommodation theo ID
router.get('/accommodations/:accommodationId', getAccommodationById);
```

```
// Route để tạo mới một accommodation

router.post('/:touristSpotId/accommodations', createAccommodation);

// Route để cập nhật thông tin của một accommodation theo ID

router.put('/:touristSpotId/accommodations/:accommodationId',
updateAccommodationById);

// Route để xóa một accommodation theo ID

router.delete('/:touristSpotId/accommodations/:accommodationId',
deleteAccommodationById);
```

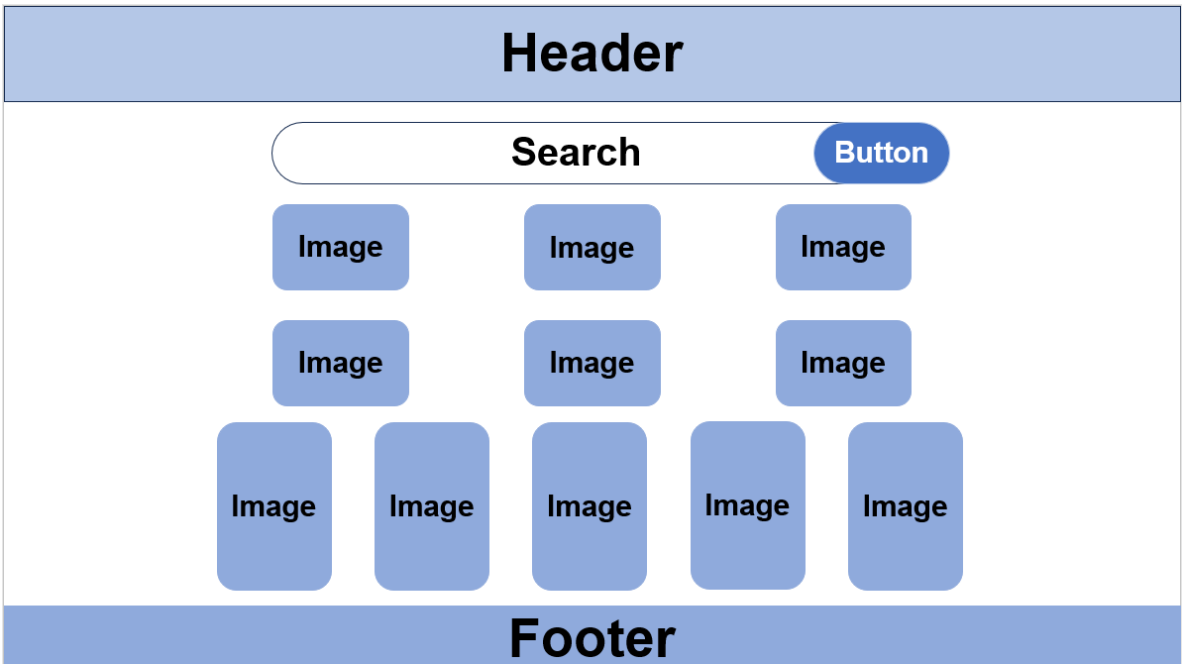
- restaurantRouter (Chứa các API xử lý việc thêm, xoá, sửa nhà hàng):

```
const router = express.Router();


// Route để lấy tất cả dữ liệu restaurants từ tất cả tourist spots

router.get('/all/restaurants', getAllRestaurantsFromAllSpots);

// Route để lấy tất cả dữ liệu restaurants của một tourist spot

router.get('/:touristSpotId/restaurants', getAllRestaurants);

// Route để lấy dữ liệu restaurant theo ID

router.get('/restaurants/:restaurantId', getRestaurantById);

// Route để tạo mới một restaurant

router.post('/:touristSpotId/restaurants', createRestaurant);

// Route để cập nhật thông tin của một restaurant theo ID

router.put('/:touristSpotId/restaurants/:restaurantId',
updateRestaurantById);

// Route để xóa một restaurant theo ID

router.delete('/:touristSpotId/restaurants/:restaurantId',
deleteRestaurantById);
```

3.6. Xây dựng giao diện.

Phát thảo giao diện để tiến hành xây dựng trang web gồm có: giao diện người dùng và giao diện trang quản trị.

3.6.1. *Phác thảo giao diện người dùng.*



Hình 3.3. Phác thảo giao diện trang chủ.

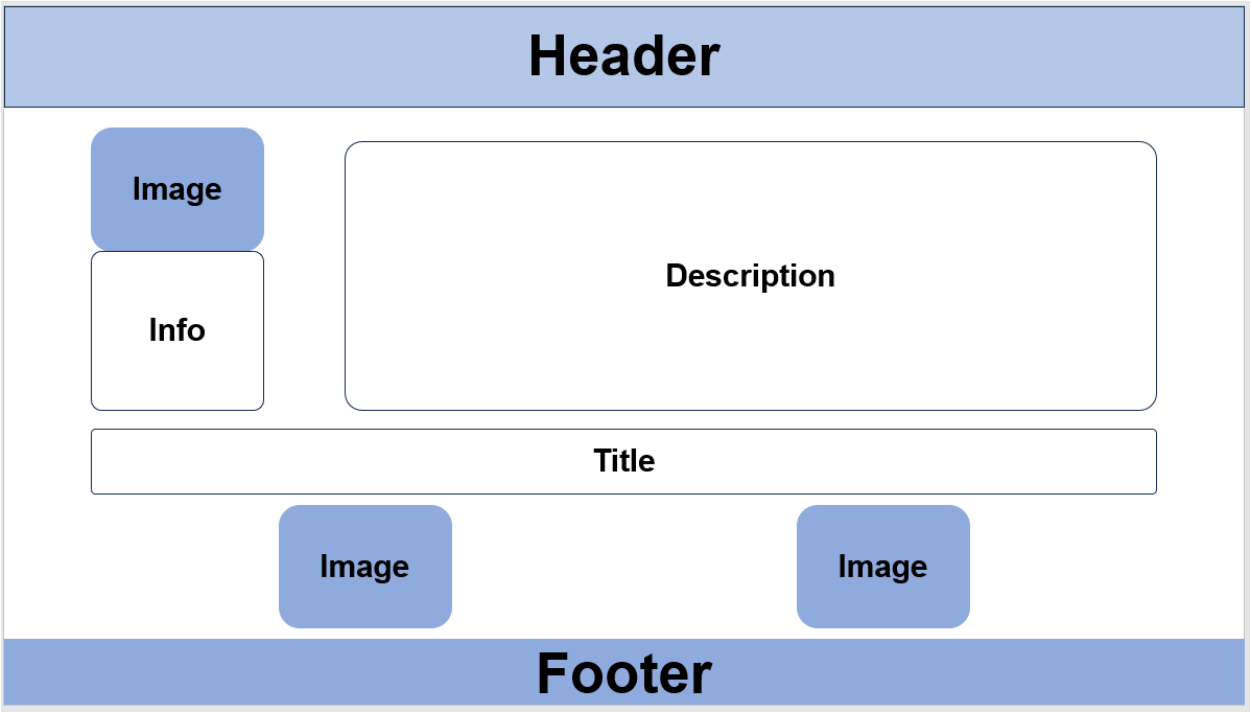
Trong đó:

Header: phần đầu của trang web.

Search: thanh tìm kiếm để người dùng dễ dàng tìm được các địa điểm du lịch.

Image: các địa điểm du lịch có trên trang web.

Footer: phần chân của trang web.



Hình 3.4, Phác thảo giao diện trang thông tin đị điểm du lịch.

Trong đó:

Header: phần đầu của trang web.

Image: hình ảnh của địa điểm du lịch.

Info: chứa các thông tin của địa điểm du lịch.

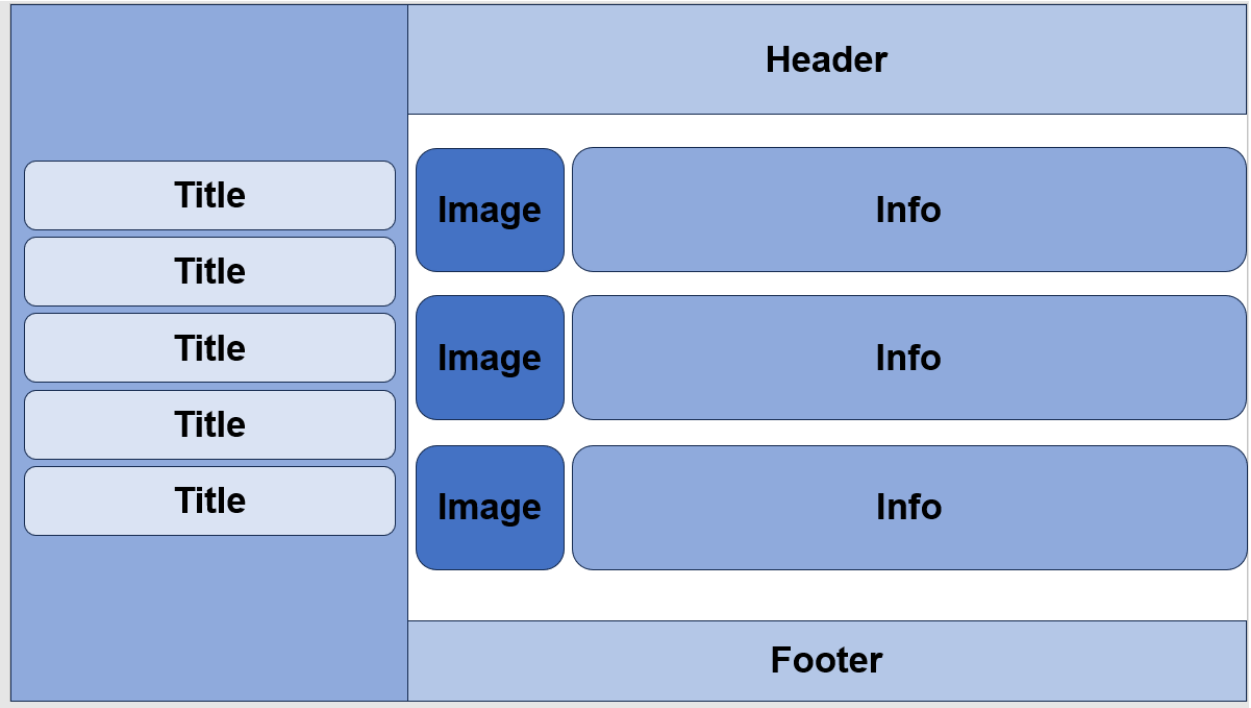
Description: chứa các mô tả của địa điểm du lịch.

Title: chứa các tiêu đề bảng con như: nơi lưu trú, nhà hàng,đặc sản, dịch vụ, quà lưu niệm.

Image: hình ảnh của các title.

Footer: phần chân của trang web.

3.6.2. *Phác thảo giao diện trang quản trị.*



Hình 3.5. Phác thảo giao diện trang quản trị.

Trong đó:

Header: phần đầu của trang web.

Title: các lựa chọn để người quản trị thực hiện tương tác với các bảng tương ứng với tiêu đề hiển thị.

Image: hình ảnh của các title.

Info: chứa các thông tin của các title tương ứng.

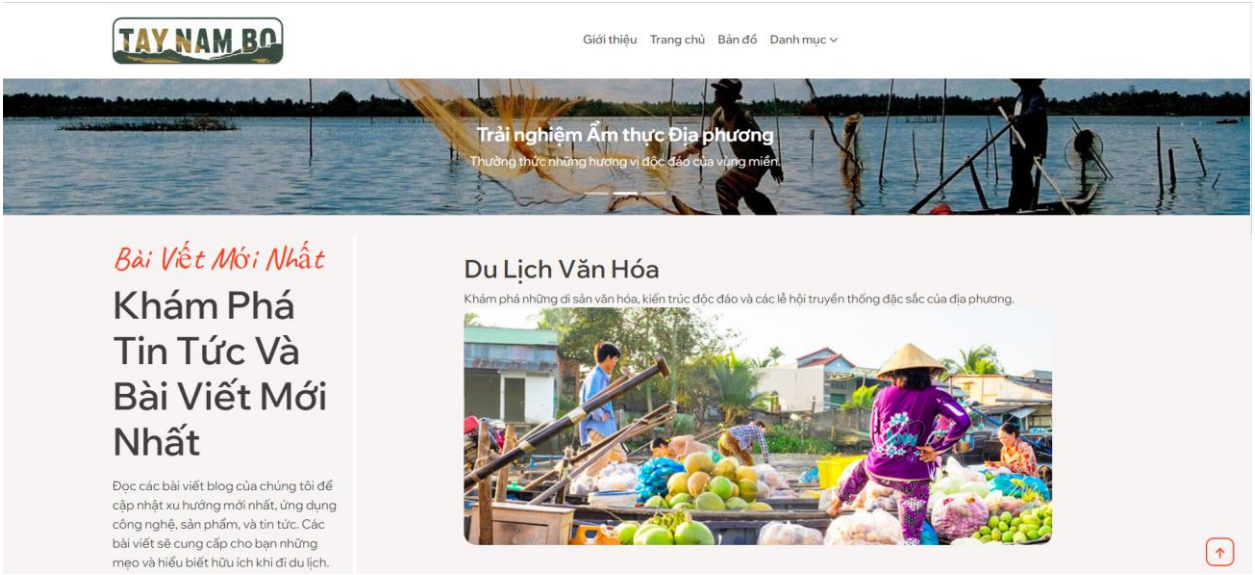
Footer: phần chân của trang web.

CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU

4.1. Giao diện chức năng phía người dùng.

4.1.1. *Trang giới thiệu.*

Trang giới thiệu mang đến cái nhìn tổng quan về vẻ đẹp của Tây Nam Bộ qua những hình ảnh sống động. Bạn sẽ thấy những khung cảnh thiên nhiên, trải nghiệm ẩm thực địa phương và tham gia các hoạt động phiêu lưu thú vị. Cung cấp các bài viết mới nhất về du lịch văn hóa, tôn giáo, sinh thái và làng nghề truyền thống. Mỗi bài viết đi kèm với hình ảnh minh họa hấp dẫn, giúp bạn dễ dàng tìm kiếm và lên kế hoạch cho những chuyến đi đáng nhớ.



Hình 4.1. Giao diện trang giới thiệu.

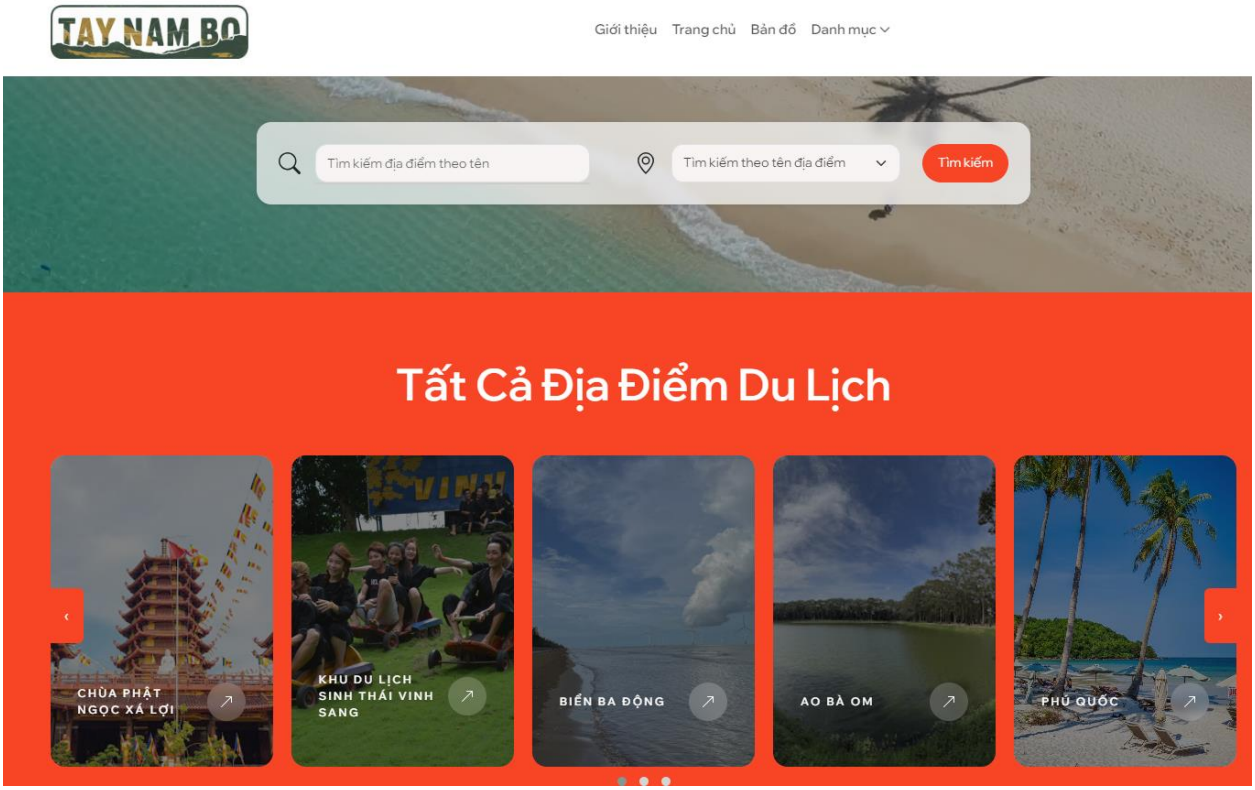
4.1.2. *Giao diện trang chủ.*

Phía trên cùng của trang là thanh header chứa các tùy chọn gồm "Giới Thiệu", "Trang Chủ", "Bản Đồ", và "Danh Mục". Thanh header này giúp dễ dàng điều hướng đến các phần chính của trang web.

Phần trang chủ bao gồm một thanh tìm kiếm cho phép tìm kiếm theo tên hoặc theo địa chỉ của các địa điểm du lịch. Thanh tìm kiếm này giúp nhanh chóng tìm thấy thông tin địa điểm du lịch cần một cách tiện lợi.

Dưới thanh tìm kiếm là phần hiển thị tất cả các địa điểm du lịch. Các địa điểm này được trình bày dưới dạng danh sách, với hình ảnh minh họa và thông tin ngắn gọn về mỗi địa điểm. Mỗi mục trong danh sách có thể được nhấp vào để xem chi tiết hơn về địa điểm

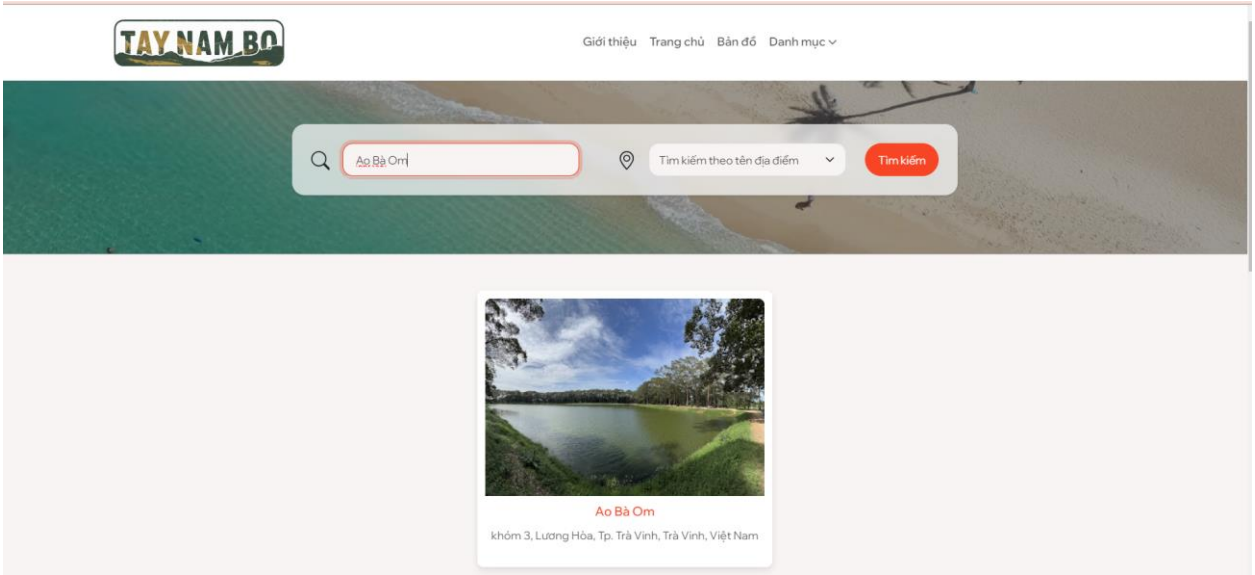
đó, bao gồm mô tả, hình ảnh, và các dịch vụ liên quan.



Hình 4.2. Giao diện trang chủ.

4.1.3. *Giao diện tìm kiếm theo tên địa điểm.*

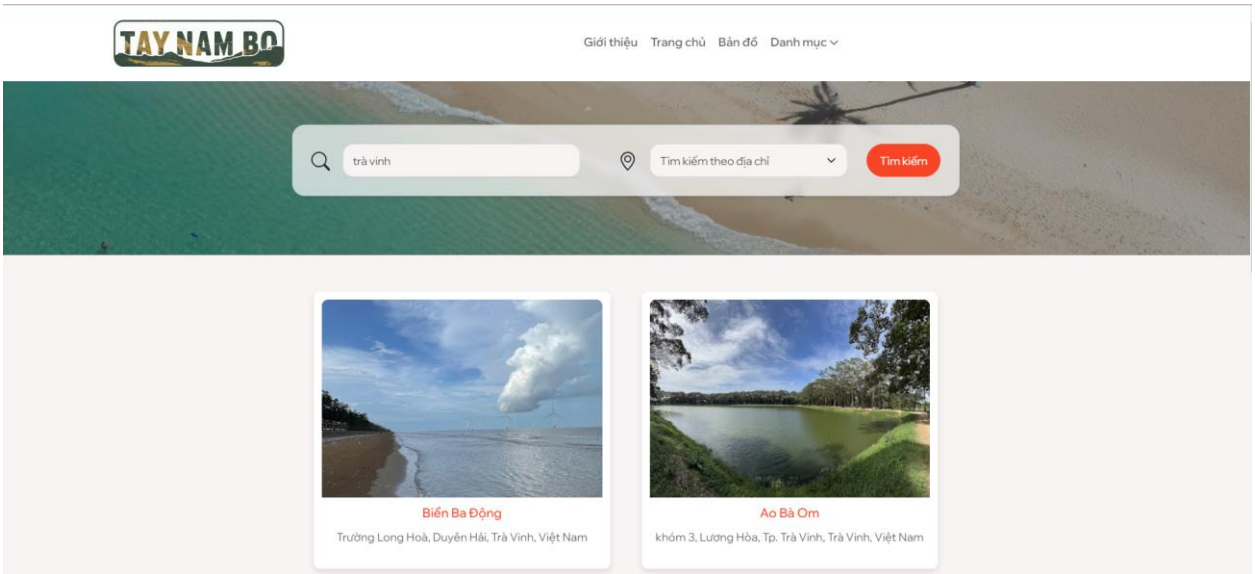
Người dùng có thể nhập tên địa điểm vào hộp tìm kiếm. Thanh tìm kiếm sẽ tự động gợi ý các kết quả phù hợp khi người dùng nhập liệu, giúp tăng tốc độ tìm kiếm. Khi người dùng nhập tên địa điểm và nhấn Enter hoặc nhấp vào nút tìm kiếm, trang sẽ hiển thị danh sách các địa điểm phù hợp ngay bên dưới thanh tìm kiếm. Mỗi kết quả tìm kiếm bao gồm một hình ảnh nhỏ, tên địa điểm, và một đoạn mô tả ngắn. Các kết quả được sắp xếp theo mức độ phù hợp với từ khóa tìm kiếm. Người dùng có thể nhấp vào bất kỳ kết quả nào để xem chi tiết hơn về địa điểm đó, bao gồm thông tin mô tả đầy đủ, hình ảnh, và các dịch vụ liên quan.



Hình 4.3. Giao diện tìm kiếm theo tên địa điểm.

4.1.4. *Giao diện tìm kiếm theo địa chỉ.*

Người dùng có thể nhập địa chỉ vào hộp tìm kiếm. Thanh tìm kiếm sẽ tự động gợi ý các kết quả phù hợp khi người dùng nhập liệu, giúp tăng tốc độ tìm kiếm. Khi người dùng nhập địa chỉ và nhấn Enter hoặc nhấp vào nút tìm kiếm, trang sẽ hiển thị danh sách các địa điểm phù hợp ngay bên dưới thanh tìm kiếm. Mỗi kết quả tìm kiếm bao gồm một hình ảnh nhỏ, tên địa điểm, và một đoạn mô tả ngắn. Các kết quả được sắp xếp theo mức độ phù hợp với từ khóa tìm kiếm. Người dùng có thể nhấp vào bất kỳ kết quả nào để xem chi tiết hơn về địa điểm đó, bao gồm thông tin mô tả đầy đủ, hình ảnh, và các dịch vụ liên quan.



Hình 4.4. Giao diện tìm kiếm theo địa chỉ.

4.1.5. *Giao diện thông tin chi tiết của địa điểm.*

Trang chi tiết địa điểm du lịch trên trang web được thiết kế để cung cấp mọi thông tin cần thiết về một địa điểm cụ thể. Giao diện trực quan, dễ sử dụng và chứa đầy đủ các thông tin chi tiết, giúp người dùng có cái nhìn tổng quan và sâu sắc về từng điểm đến.

- Tiêu đề: Được đặt ở đầu trang, tiêu đề bao gồm tên của địa điểm du lịch được hiển thị nổi bật.

- Hình ảnh chính: Một hình ảnh lớn và chất lượng cao của địa điểm, giúp người dùng hình dung rõ ràng về điểm đến.

- Nội dung mô tả: Phần này cung cấp một đoạn mô tả chi tiết về địa điểm, bao gồm lịch sử, điểm đặc biệt và những gì người dùng có thể mong đợi khi đến thăm.

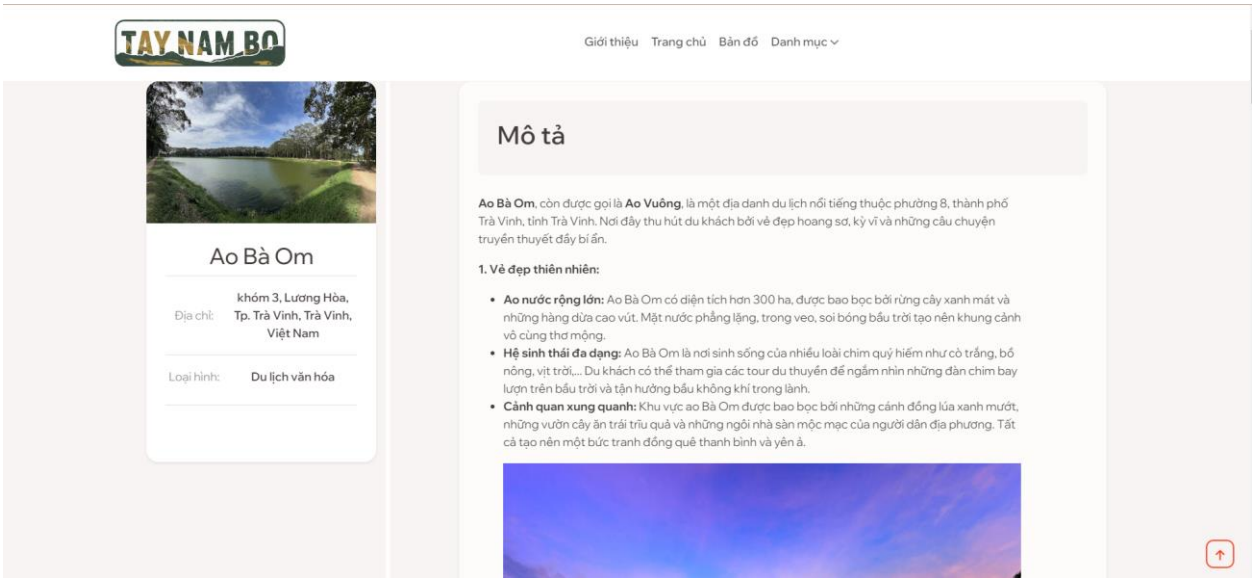
- Địa chỉ: Địa chỉ cụ thể của địa điểm, bao gồm đường, thành phố.

- Thông Tin Các Dịch Vụ Liên Quan:

  - Chỗ ở: Danh sách các khách sạn, nhà nghỉ gần địa điểm du lịch, kèm theo thông tin chi tiết như giá cả, địa chỉ, và số điện thoại liên hệ.

  - Nhà hàng: Các nhà hàng nổi bật gần địa điểm, cùng với mô tả ngắn về món ăn và dịch vụ.

  - Đặc sản, dịch vụ và quà lưu niệm: Thông tin về các đặc sản địa phương và những cửa hàng quà lưu niệm mà du khách có thể ghé thăm.



Hình 4.5. Giao diện thông tin chi tiết của địa điểm.

4.1.6. *Giao diện thông tin các dịch vụ liên quan.*

Phần "Thông Tin Các Dịch Vụ Liên Quan" trên trang chi tiết địa điểm du lịch cung cấp một cái nhìn tổng quan về các dịch vụ hỗ trợ du khách. Đầu tiên là mục "Khách Sạn", trong đó mỗi khách sạn đi kèm với hình ảnh minh họa, tên, và mô tả ngắn gọn. Giá phòng được hiển thị rõ ràng, cùng với địa chỉ và số điện thoại liên hệ của khách sạn. Người dùng cũng có thể sử dụng liên kết Google Maps để xem vị trí cụ thể của từng khách sạn.

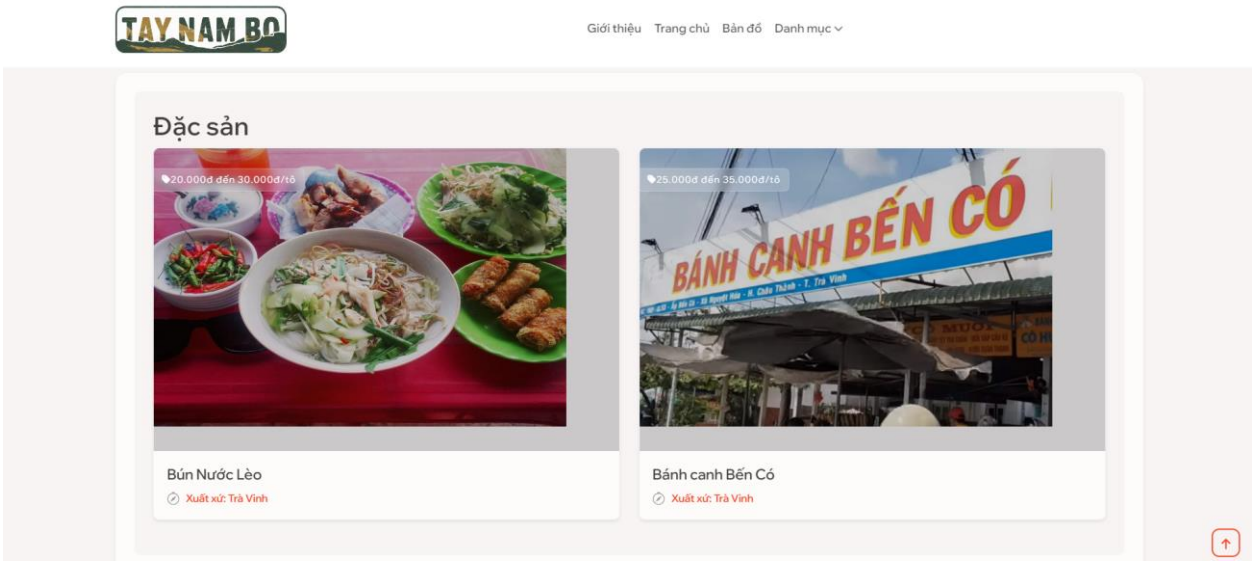
Tiếp theo là mục "Nhà Hàng", nơi người dùng có thể tìm thấy hình ảnh của các nhà hàng hoặc món ăn đặc trưng, kèm theo tên nhà hàng và giá cả được hiển thị rõ ràng. Địa chỉ và số điện thoại liên hệ của nhà hàng cũng được cung cấp để du khách dễ dàng liên lạc.

Phần "Đặc Sản" giới thiệu các sản phẩm đặc sản với hình ảnh minh họa, tên sản phẩm, giá cả rõ ràng, và thông tin về xuất xứ của đặc sản, giúp du khách có thể tìm hiểu và lựa chọn những món quà đặc trưng của địa phương.

Mục "Dịch Vụ Khác" bao gồm các dịch vụ như thuê xe, tour guide, với hình ảnh đại diện, tên dịch vụ và giá cả được hiển thị chi tiết, giúp du khách dễ dàng lựa chọn dịch vụ phù hợp cho chuyến đi của mình.

Cuối cùng là phần "Quà Lưu Niệm", nơi giới thiệu các sản phẩm quà lưu niệm với hình ảnh, tên sản phẩm và giá cả rõ ràng, giúp du khách dễ dàng chọn lựa những món quà ý nghĩa để mang về sau chuyến đi. Giao diện này không chỉ cung cấp đầy đủ thông tin mà còn mang lại trải nghiệm mượt mà và thân thiện, giúp người dùng dễ dàng khám phá và tận hưởng các dịch vụ liên quan tại địa điểm du lịch.

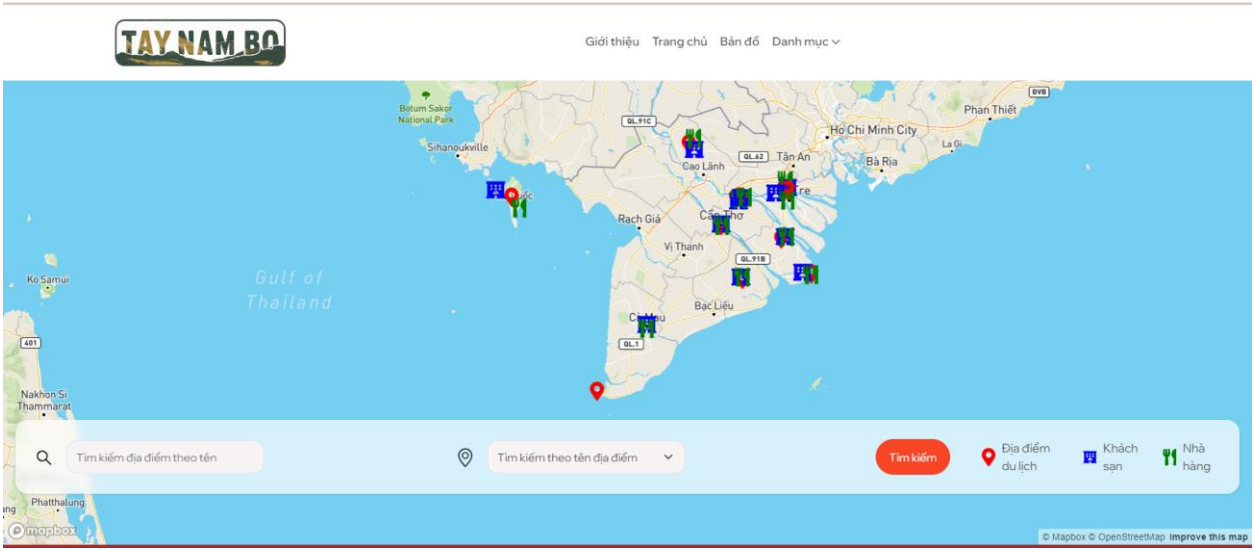
Người dùng có thể nhấp vào bất kỳ kết quả nào để xem chi tiết hơn về dịch vụ đó, bao gồm thông tin mô tả đầy đủ, hình ảnh, và các chi tiết liên quan.



Hình 4.6. Giao diện thông tin các dịch vụ liên quan.

4.1.7. *Giao diện bản đồ.*

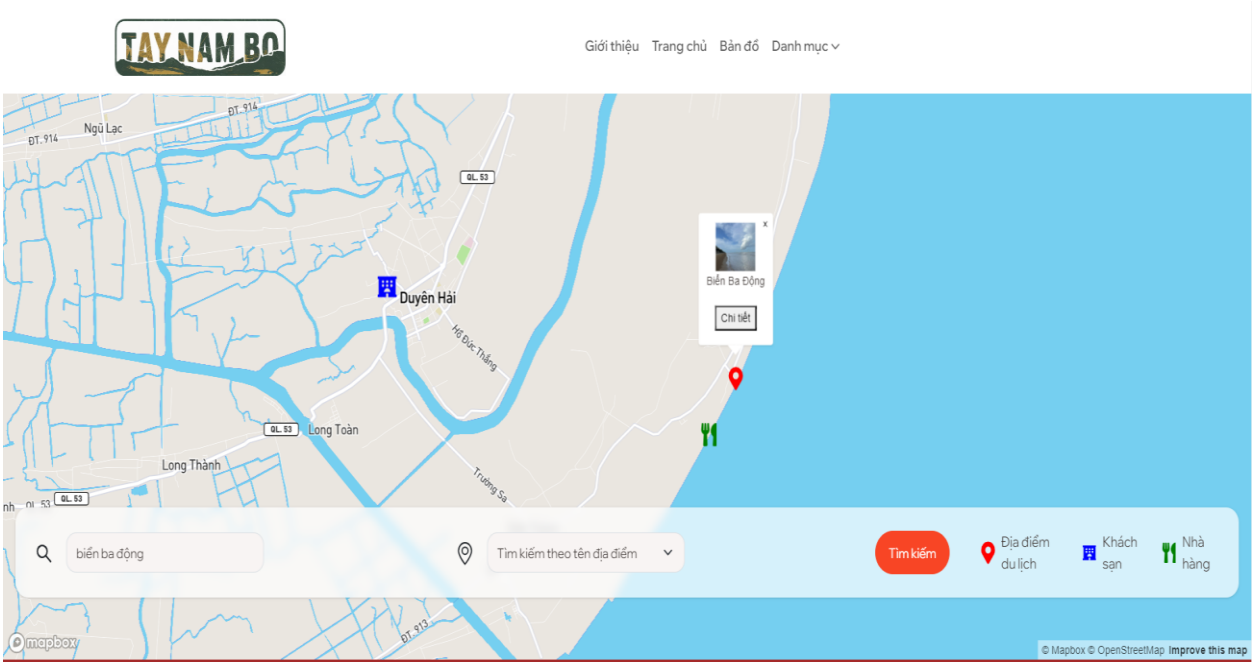
Giao diện bản đồ trên trang web được thiết kế trực quan và dễ điều hướng, giúp người dùng dễ dàng tìm kiếm và khám phá các địa điểm du lịch, khách sạn, và nhà hàng. Bản đồ sử dụng dữ liệu từ Mapbox và OpenStreetMap, hiển thị chi tiết khu vực với các biểu tượng rõ ràng: biểu tượng màu đỏ cho địa điểm du lịch, màu xanh dương cho khách sạn, và màu xanh lá cho nhà hàng. Dưới cùng của bản đồ là thanh tìm kiếm, cho phép người dùng nhập tên địa điểm với nút tìm kiếm để thực hiện tìm kiếm. Ở góc phải dưới của bản đồ, có chú thích giải thích ý nghĩa các biểu tượng. Giao diện này cung cấp cái nhìn toàn cảnh về các điểm đến, giúp người dùng dễ dàng lập kế hoạch và tìm kiếm các dịch vụ liên quan cho chuyến đi của mình.



Hình 4.7. Giao diện bản đồ.

4.1.8. *Giao diện tìm kiếm địa điểm bằng tên trên bản đồ.*

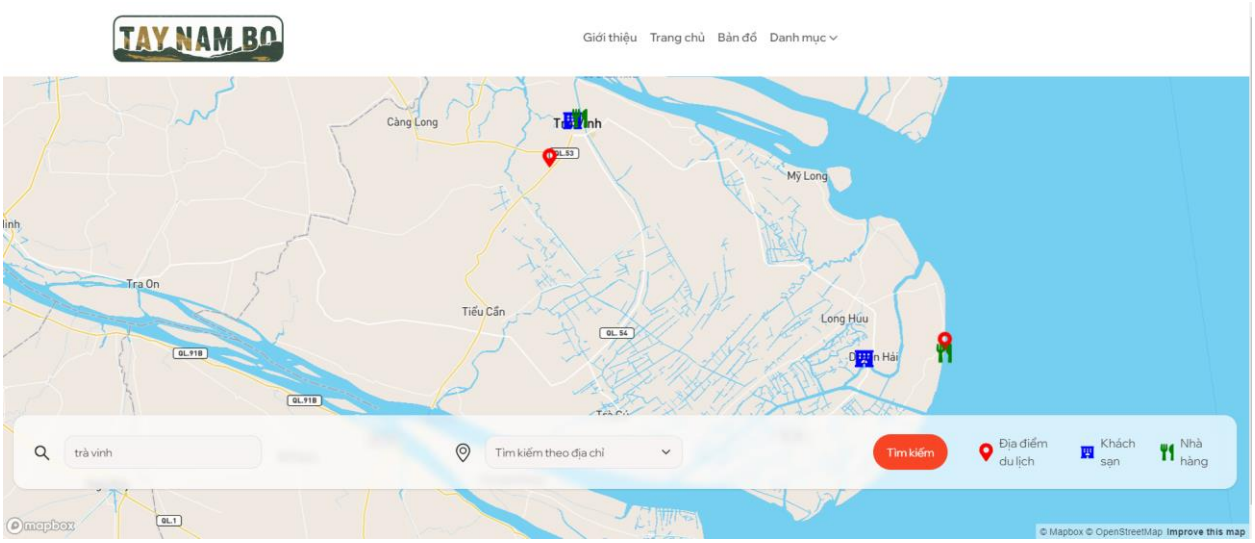
Khi tìm kiếm thành công bằng tên địa điểm, bản đồ sẽ chỉ hiển thị địa điểm đó cùng với các nhà hàng và khách sạn liên quan, các địa điểm khác sẽ bị ẩn. Khi bấm vào "Chi tiết" sẻ chuyển qua trang chi tiết của địa điểm tương ứng.



Hình 4.8. Giao diện tìm kiếm địa điểm bằng tên trên bản đồ.

4.1.9. *Giao diện tìm kiếm địa điểm bằng địa chỉ trên bản đồ.*

Khi tìm kiếm thành công bằng địa chỉ địa điểm, bản đồ sẽ chỉ hiển thị các địa điểm có địa chỉ trong khu vực đó cùng với các nhà hàng và khách sạn liên quan, các địa điểm khác sẽ bị ẩn. Khi bấm vào "Chi tiết" sẻ chuyển qua trang chi tiết của địa điểm tương ứng.



Hình 4.9. Giao diện tìm kiếm địa điểm bằng địa chỉ trên bản đồ.

4.1.10. *Giao diện chức năng lọc theo danh mục.*

Chức năng lọc theo danh mục trên trang web của tôi hiển thị chính xác các địa điểm thuộc danh mục mà người dùng lựa chọn. Khi người dùng chọn một danh mục từ menu "Danh Mục", trang sẽ hiển thị danh sách các địa điểm thuộc danh mục đó,gồm hình ảnh tên địa điểm và địa chỉ. Chỉ các địa điểm thuộc danh mục đã chọn mới được hiển thị, giúp người dùng nhanh chóng tìm thấy thông tin phù hợp. Người dùng có thể nhấp vào bất kỳ kết quả nào để xem chi tiết hơn về địa điểm đó, bao gồm thông tin mô tả đầy đủ, hình ảnh, và các dịch vụ liên quan.

Hình 4.10. Giao diện tìm kiếm địa điểm bằng tên trên bản đồ.

4.2. Giao diện chức năng phía người quản trị.

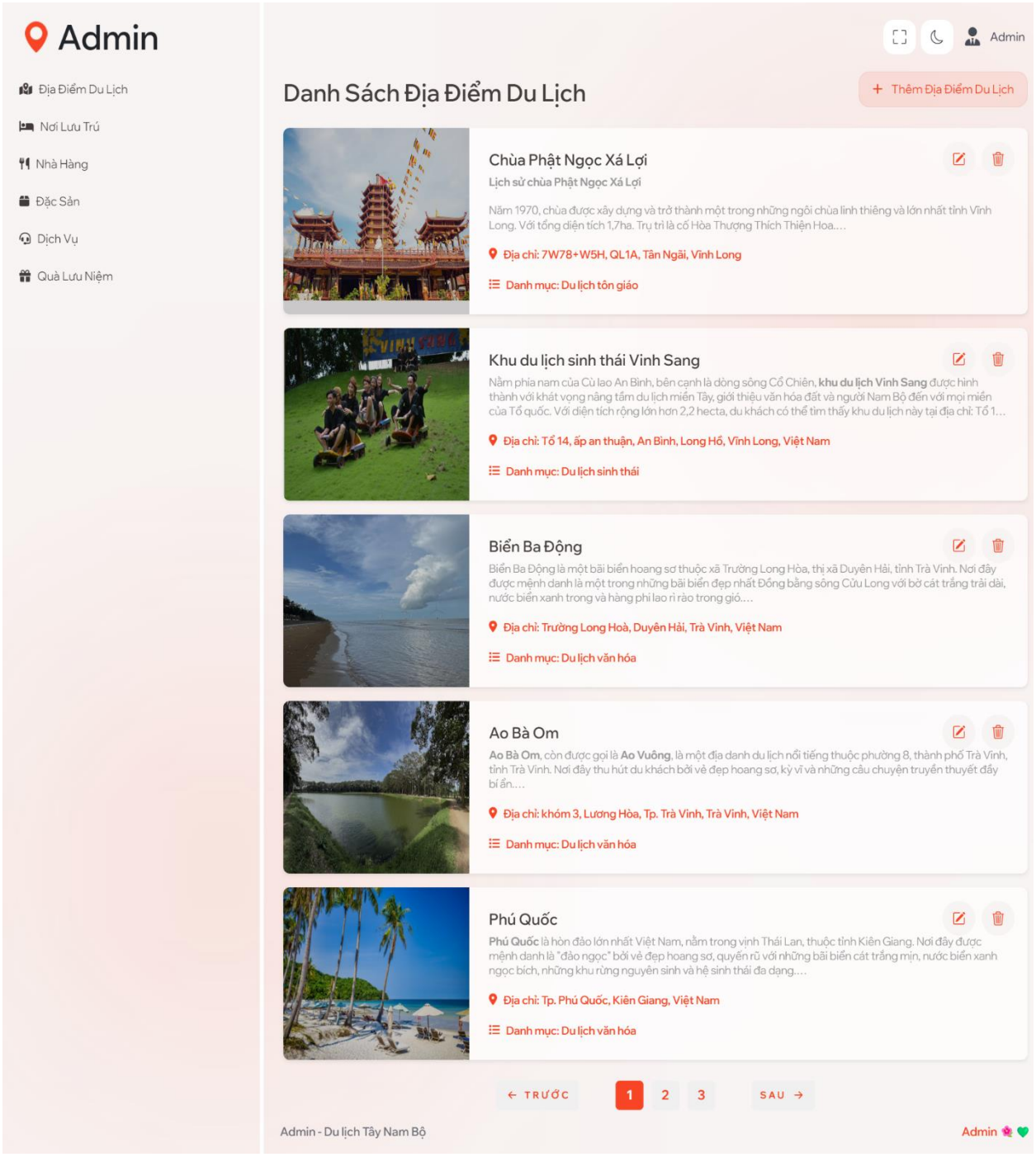
4.2.1. *Giao diện đăng nhập trang quản trị.*

Người dùng được hướng dẫn nhập "Tên Tài Khoản" và "Mật Khẩu" để tiếp tục, với các trường nhập liệu rõ ràng và có biểu tượng để hiển thị hoặc ẩn mật khẩu. Nút "Đăng Nhập giúp người dùng dễ dàng hoàn tất quá trình đăng nhập.

Hình 4.11. Giao diện đăng nhập trang quản trị.

4.2.2. *Giao diện danh sách tất cả địa điểm du lịch.*

Giao diện danh sách tất cả địa điểm du lịch trên trang quản trị giúp quản trị viên dễ dàng quản lý thông tin. Thanh điều hướng bên trái chứa các mục như "Địa Điểm Du Lịch", "Nơi Lưu Trú", và "Nhà Hàng", giúp chuyển đổi giữa các danh mục. Mỗi địa điểm được hiển thị dưới dạng thẻ thông tin với hình ảnh minh họa, tên, mô tả ngắn, địa chỉ, và danh mục. Các biểu tượng chỉnh sửa và xóa cho phép quản trị viên cập nhật hoặc xóa thông tin dễ dàng. Nút "Thêm Địa Điểm Du Lịch" ở góc trên bên phải giúp thêm mới địa điểm nhanh chóng.



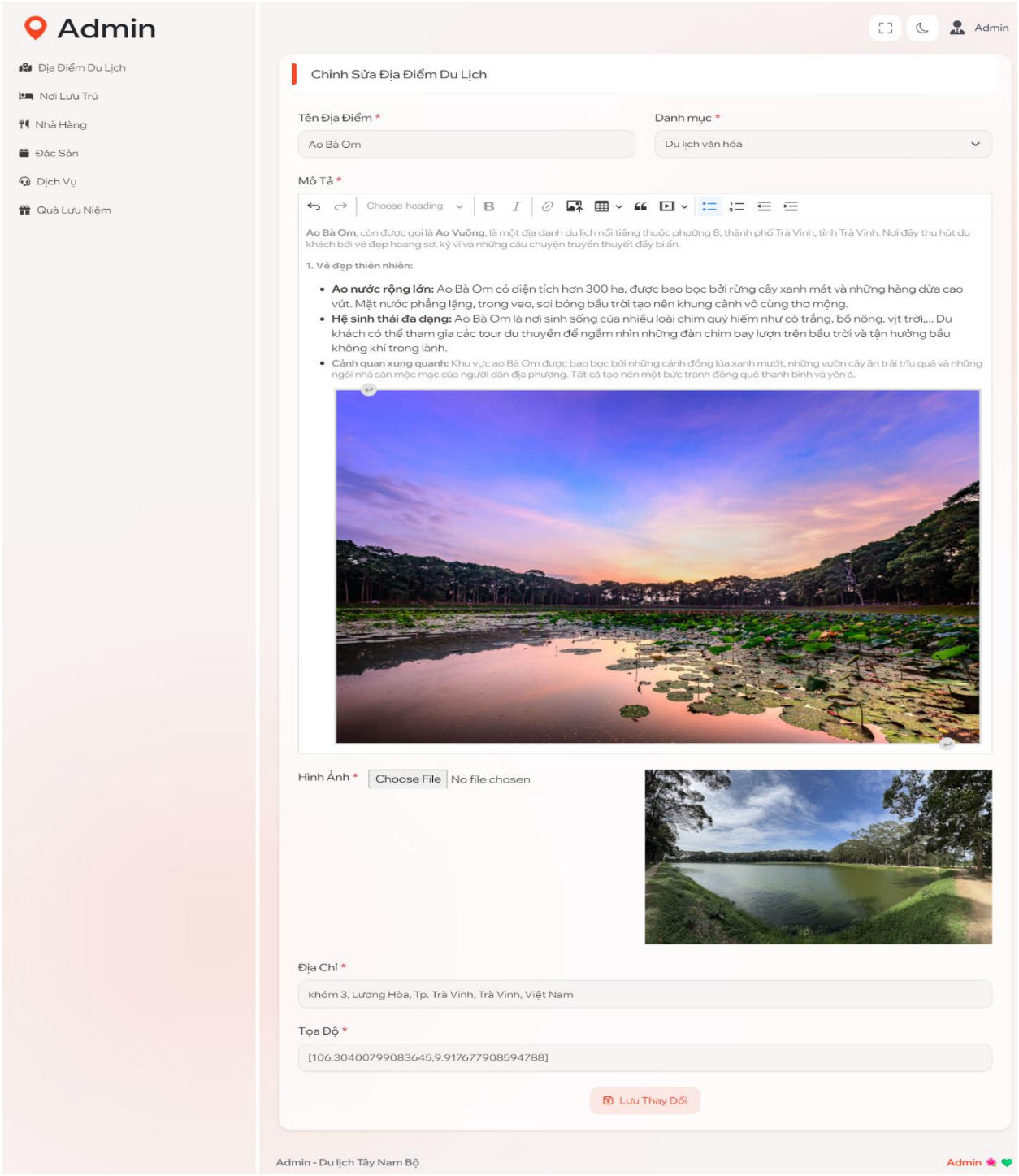
Hình 4.12. Giao diện danh sách tất cả địa điểm du lịch.

4.2.3. *Chức năng thêm mới địa điểm du lịch.*

Giao diện thêm mới địa điểm du lịch trên trang quản trị được thiết kế rõ ràng giúp quản trị viên dễ dàng nhập và quản lý thông tin về các địa điểm du lịch. Phần tiêu đề "Thêm Địa Điểm Du Lịch". Form nhập liệu bao gồm các trường bắt buộc như "Địa Điểm", "Danh Mục", "Mô Tả", "Hình Ảnh", "Địa Chỉ", và "Tọa Độ". Nút "Thêm Địa Điểm Du Lịch" nằm ở phía dưới form, giúp quản trị viên hoàn tất việc nhập liệu và thêm mới địa điểm du lịch vào hệ thống.

4.2.4. *Giao diện chỉnh sửa địa điểm du lịch.*

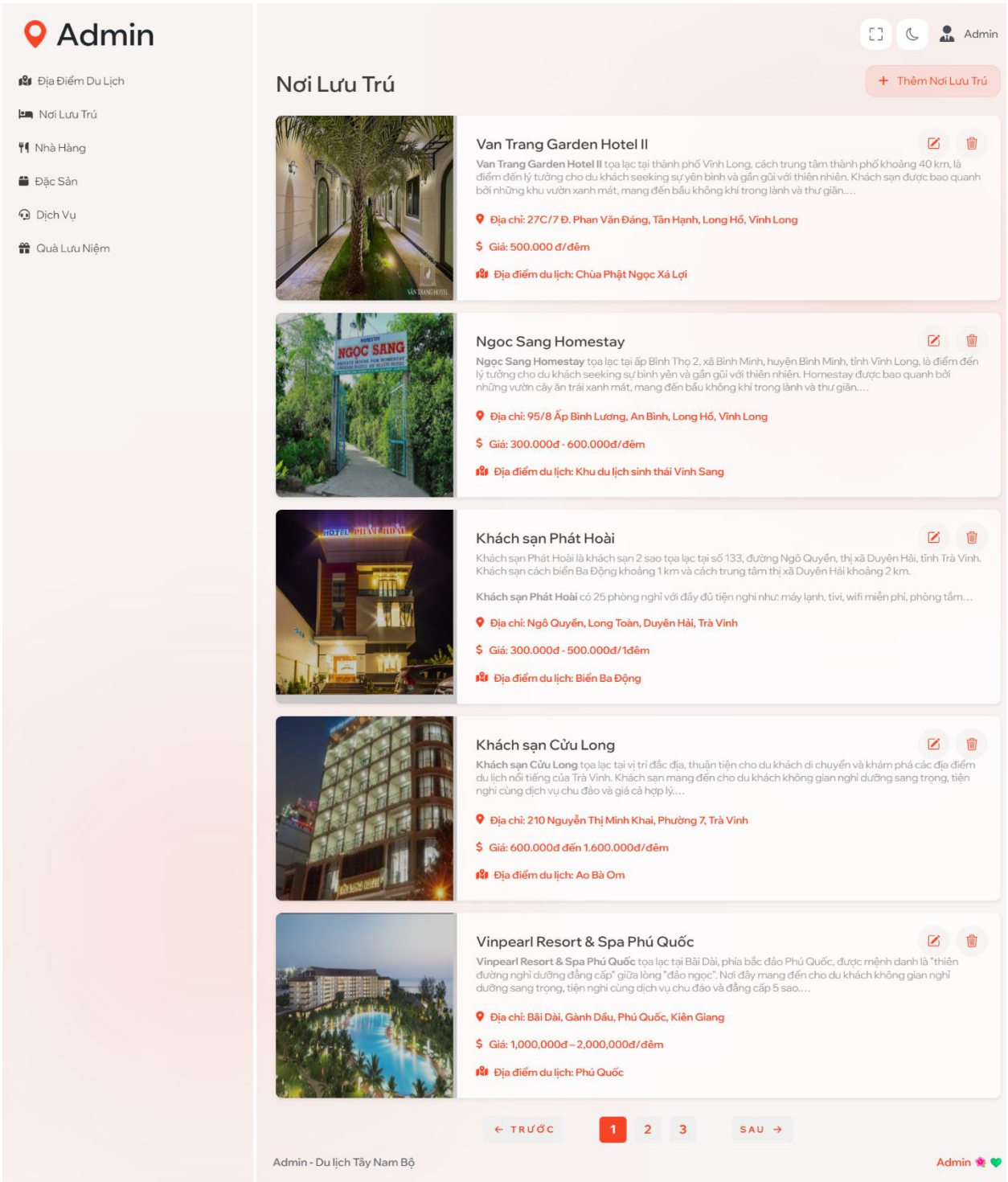
Form nhập liệu bao gồm các trường bắt buộc như "Địa Điểm", "Danh Mục", "Mô Tả", "Hình Ảnh", "Địa Chỉ", và "Tọa Độ". Sau khi thay đổi các thông tin quản trị viên có thể bấm nút "Lưu Thay Đổi" nằm ở phía dưới form, cho phép quản trị viên lưu lại những thay đổi đã thực hiện.



Hình 4.13. Giao diện chỉnh sửa địa điểm du lịch.

4.2.5. *Giao diện danh sách tất cả nơi lưu trú.*

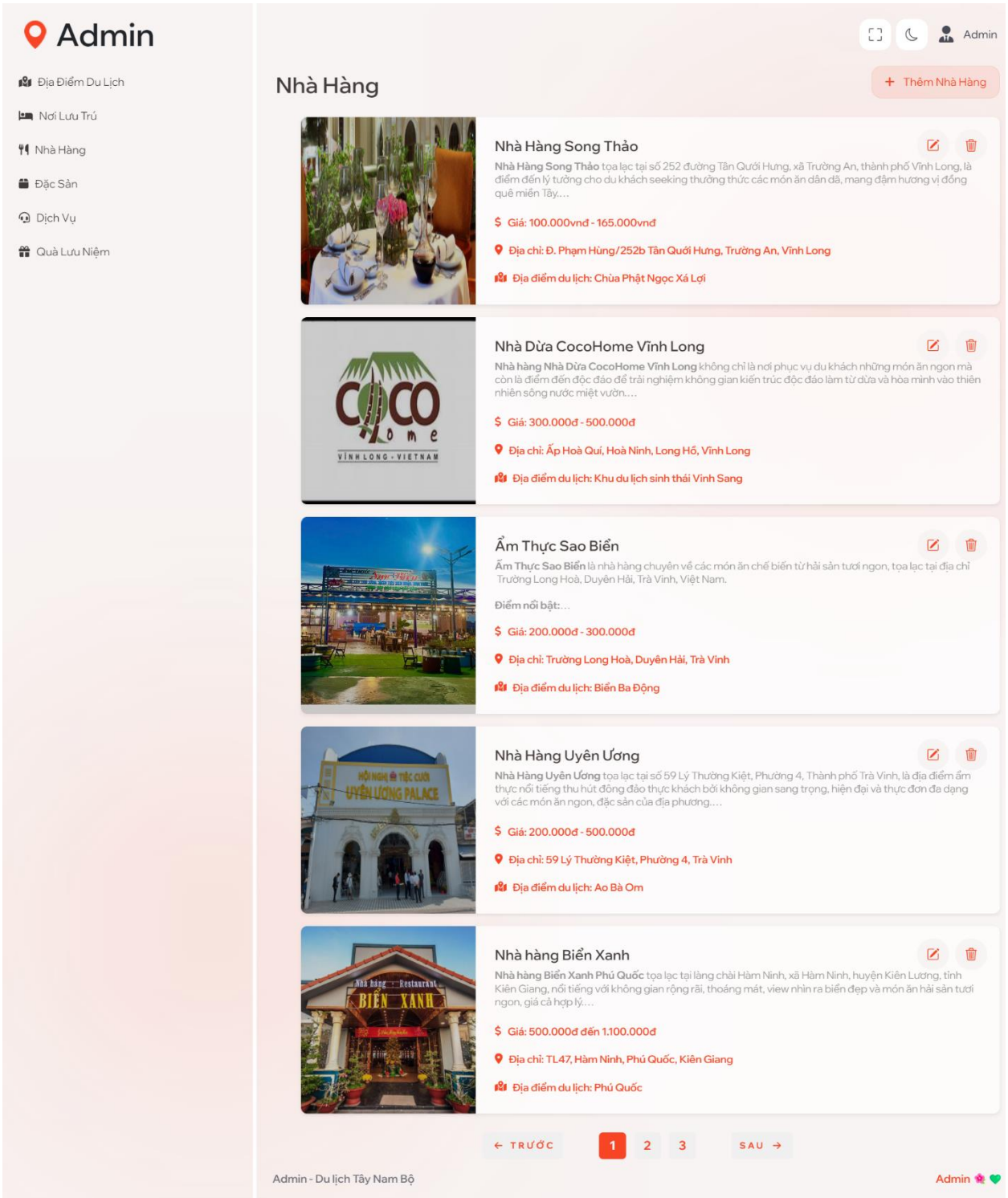
Giao diện danh sách nơi lưu trú trên trang quản trị được thiết kế để quản trị viên có thể dễ dàng quản lý và cập nhật thông tin về các khách sạn, homestay, và các loại hình lưu trú khác. Mỗi thẻ thông tin nơi lưu trú có các biểu tượng chỉnh sửa và xóa, cho phép quản trị viên dễ dàng cập nhật hoặc xóa thông tin. Giao diện này giúp quản trị viên quản lý thông tin nơi lưu trú một cách hiệu quả.



Hình 4.14. Giao diện danh sách tất cả nơi lưu trú.

4.2.6. *Giao diện danh sách tất cả nhà hàng.*

Giao diện danh sách nhà hàng trên trang quản trị được thiết kế để quản trị viên có thể dễ dàng quản lý và cập nhật thông tin về các nhà hàng. Mỗi thẻ thông tin nhà hàng có các biểu tượng chỉnh sửa và xóa, cho phép quản trị viên dễ dàng cập nhật hoặc xóa thông tin. Giao diện này giúp quản trị viên quản lý thông tin nhà hàng một cách hiệu quả.



Hình 4.15. Giao diện danh sách tất cả nhà hàng.

4.2.7. *Quản lý danh sách tất cả đặc sản.*

Giao diện quản trị danh sách đặc sản được thiết kế tối ưu để hỗ trợ quản trị viên trong việc quản lý và cập nhật thông tin. Mỗi mục đặc sản hiển thị các biểu tượng chỉnh sửa và xóa, giúp quản trị viên có thể thực hiện các thao tác cập nhật hoặc xóa thông tin một cách nhanh chóng và thuận tiện. Giao diện này đảm bảo việc quản lý thông tin về đặc sản được thực hiện hiệu quả và chính xác.

4.2.8. *Quản lý danh sách tất cả dịch vụ.*

Trang quản trị danh sách dịch vụ cung cấp một giao diện trực quan, cho phép quản trị viên dễ dàng quản lý và cập nhật thông tin dịch vụ. Từng mục dịch vụ đi kèm với các biểu tượng chỉnh sửa và xóa, tạo điều kiện thuận lợi cho việc cập nhật hoặc xóa thông tin. Giao diện này được thiết kế để tối ưu hóa quy trình quản lý, giúp quản trị viên duy trì thông tin dịch vụ một cách hiệu quả.

4.2.9. *Quản lý danh sách tất cả quà lưu niệm.*

Giao diện quản lý quà lưu niệm trên trang quản trị được thiết kế nhằm hỗ trợ quản trị viên trong việc cập nhật và quản lý thông tin về các quà lưu niệm. Mỗi mục quà lưu niệm có các biểu tượng chỉnh sửa và xóa, giúp việc cập nhật hoặc xóa thông tin trở nên dễ dàng và nhanh chóng. Giao diện này được xây dựng để nâng cao hiệu suất quản lý và đảm bảo thông tin về quà lưu niệm luôn được cập nhật kịp thời và chính xác.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận.

Cơ sở lý thuyết và công nghệ: Trong quá trình thực hiện dự án, tôi đã tiến hành nghiên cứu sâu rộng về các địa điểm du lịch tại các tỉnh Tây Nam Bộ. Ngoài việc tìm hiểu về các địa điểm du lịch, tôi còn nghiên cứu các công nghệ và công cụ hiện đại để áp dụng vào việc phát triển hệ thống. Những công nghệ chủ chốt bao gồm cơ sở dữ liệu NoSQL với MongoDB, môi trường runtime NodeJS, kiến trúc RESTful API, và framework ReactJS. Hiểu biết và sử dụng đúng cách các công nghệ này đã tạo nên một nền tảng vững chắc, giúp đảm bảo tính hiệu quả, khả năng mở rộng, và tính ổn định của hệ thống. Những kiến thức về công nghệ này không chỉ đóng vai trò nền tảng lý thuyết mà còn là công cụ thực tiễn giúp tôi hiện thực hóa các ý tưởng và chức năng của hệ thống.

Hiện thực hóa nghiên cứu: Dựa trên các nghiên cứu và hiểu biết đã có, tôi đã thiết kế và triển khai một hệ thống ứng dụng hoàn chỉnh với nhiều chức năng quan trọng. Các chức năng chính bao gồm khả năng tìm kiếm địa điểm du lịch, quản lý thông tin du lịch, và cung cấp giao diện người dùng trực quan cho cả du khách và quản trị viên. Giao diện của ứng dụng đã được thiết kế một cách chi tiết, thân thiện với người dùng, giúp việc tương tác trở nên dễ dàng và hiệu quả. Các công cụ quản lý thông tin mạnh mẽ giúp quản trị viên dễ dàng cập nhật và duy trì thông tin chính xác, đồng thời nâng cao trải nghiệm người dùng cuối.

5.2. Hướng phát triển.

Hướng dẫn đường đi đến các địa điểm du lịch: Tích hợp các công cụ chỉ đường chi tiết và chính xác, giúp du khách dễ dàng tìm đường đến các địa điểm du lịch một cách thuận tiện. Việc này có thể bao gồm chỉ đường bằng bản đồ trực tuyến, thông tin về giao thông công cộng, và các gợi ý về lộ trình ngắn nhất hoặc đẹp nhất.

Tích hợp các cổng thanh toán trực tuyến: Mở rộng hệ thống để cho phép du khách không chỉ tìm kiếm thông tin mà còn có thể đặt vé tham quan, đặt phòng khách sạn, và thực hiện các giao dịch thanh toán trực tiếp trên nền tảng. Điều này sẽ mang lại sự tiện lợi tối đa cho du khách, đồng thời giúp các doanh nghiệp du lịch tăng cường doanh thu và quản lý đặt chỗ hiệu quả hơn.

Tối ưu hóa giao diện: Nâng cao tính thẩm mỹ và dễ sử dụng Cải tiến thiết kế giao diện để không chỉ đẹp mắt mà còn trực quan, dễ sử dụng. Việc tối ưu hóa trải nghiệm người dùng bao gồm việc tạo ra các giao diện sạch sẽ, sử dụng các biểu tượng và hình ảnh chất lượng cao, và đảm bảo rằng các chức năng quan trọng luôn dễ dàng truy cập.

Sử dụng công nghệ AI: Tích hợp trí tuệ nhân tạo để cung cấp các gợi ý du lịch cá nhân hóa cho từng du khách dựa trên sở thích, hành vi tìm kiếm và lịch sử du lịch của họ. Công nghệ AI có thể giúp gợi ý các địa điểm mới, sự kiện địa phương, và thậm chí là các hành trình du lịch phù hợp với từng cá nhân.

Phát triển ứng dụng di động: Phát triển phiên bản ứng dụng di động để người dùng có thể truy cập thông tin và sử dụng các dịch vụ mọi lúc, mọi nơi. Ứng dụng di động có thể cung cấp các thông báo đẩy về các sự kiện, khuyến mãi, hoặc thay đổi thông tin quan trọng, giúp du khách luôn cập nhật và không bỏ lỡ bất kỳ thông tin hữu ích nào.

Hợp tác với các doanh nghiệp địa phương: Tạo mối quan hệ hợp tác với các nhà cung cấp dịch vụ du lịch địa phương như khách sạn, nhà hàng, và các công ty lữ hành để cung cấp các gói dịch vụ hấp dẫn và tiện ích cho du khách.

Tích hợp với các nền tảng xã hội: Cho phép người dùng chia sẻ trải nghiệm du lịch của họ lên các nền tảng mạng xã hội trực tiếp từ ứng dụng, giúp lan tỏa thông tin và thu hút nhiều du khách hơn.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Phan Thị Phương Nam (2022), Bài tập thực hành NoSQL – Môn: Cơ sở dữ liệu, Trường Đại học Trà Vinh.

[2] Andreas Meier, Michael Kaufmann (2019), SQL & NoSQL Databases_Models, Languages, Consistency Options and Architectures for Big Data Management.

[3] Shannon Bradshaw, Eoin Brazil, Kristina Chodorow (2019), MongoDB_ The Definitive Guide_ Powerful and Scalable Data Storage.

[4] Joe Morgan (2021), How To Code in React.js, DigitalOcean, New York, USA.

[5] Casciaro, M. (2020). Node.js Design Patterns. Packt Publishing.

[6] Lâm Thị Đông Vinh (2021), Sách hướng dẫn du lịch Tỉnh Trà Vinh, NXB Đồng Nai.

[7] Trần Ngọc Thêm (2022) Văn Hóa Người Việt Vùng Tây Nam Bộ, Nhà xuất bản Tổng Hợp TP.HCM.

[8] Fielding, R. T. (2000). Representational State Transfer (REST) API Method Overview.

[9]https://www.dataversity.net/nosql-databases-advantages-and-disadvantages, xem 25/07/2024.

[10] https://www.mongodb.com/docs/, xem 25/07/2024.

[11] https://vinpearl.com/vi/tron-bo-kinh-nghiem-du-lich-mien-tay-nam-bo-day-du-nhat, xem 30/04/2024.

[12] https://www.traveloka.com, xem 20/06/2024.