

TRƯỜNG ĐẠI HỌC TRÀ VINH
KHOA KỸ THUẬT VÀ CÔNG NGHỆ



ISO 9001:2015

NGUYỄN MINH ĐĂNG

**XÂY DỰNG HỆ THỐNG QUẢNG BÁ ĐẶC SẢN
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 ĐẶC SẢN
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: **Nguyễn Minh Đăng**

Lớp: **DA20TTB**

MSSV: **110120013**

GVHD: **ThS. Phan Thị Phương Nam**

TRÀ VINH, NĂM 2024

LỜI MỞ ĐẦU

Hiện nay, công nghệ thông tin là một ngành được ứng dụng trong rất nhiều lĩnh vực ngành nghề. Sự phát triển của nó ngày càng một đa dạng, nó được ứng dụng trong nhiều lĩnh vực từ y tế, giáo dục, thương mại điện tử đến các ngành nghề giải trí, du lịch,... góp phần vào sự phát triển của một xã hội tiên tiến, thuận lợi và hiệu quả. Trong bối cảnh công nghệ số, ứng dụng số trong việc giới thiệu sản phẩm, hàng hóa nhanh đến người tiêu dùng. Đặc biệt giới thiệu sản phẩm đặc sản địa phương góp phần giới thiệu nhanh các đặc sản đến người tiêu dùng bằng công nghệ là một ứng dụng thực tế, phù hợp với nhu cầu của nhà sản xuất sản phẩm và người tiêu dùng.

Cơ sở dữ liệu phi quan hệ là một loại cơ sở dữ liệu được thiết kế để lưu trữ và truy xuất dữ liệu một cách linh hoạt, không yêu cầu cấu trúc cố định như các cơ sở dữ liệu quan hệ truyền thống. Với khả năng mở rộng ngang dễ dàng và hiệu suất cao trong việc truy xuất và xử lý dữ liệu lớn, NoSQL trở thành lựa chọn lý tưởng cho nhiều ứng dụng hiện đại yêu cầu tính linh hoạt và hiệu suất vượt trội.

Sau quá trình tìm hiểu, lựa chọn đề tài tôi nhận thấy đề tài “Xây dựng hệ thống quảng bá đặc sản các tỉnh miền Tây Nam Bộ với cơ sở dữ liệu phi quan hệ” rất phù hợp với khả năng hiện tại của. Việc quảng bá đặc sản sẽ góp phần phát triển cho lĩnh vực du lịch và phát triển nông nghiệp bền vững.

Đồ án tốt nghiệp gồm 5 chương:

Chương 1. Đặt vấn đề

Chương 2. Cơ sở lý thuyết

Chương 3. Hiện thực hoá nghiên cứu

Chương 4. Kết quả nghiên cứu

Chương 5. Kết luận và hướng phát triển

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành nhất đến Ban giám hiệu Trường Đại học Trà Vinh, các giảng viên trong Khoa Kỹ thuật và Công nghệ cũng như toàn thể các giảng viên trong Trường Đại học Trà Vinh đã truyền đạt những kiến thức quý báu cho em trong thời gian học tập tại trường. Đặc biệt, Em xin chân thành cảm ơn Cô Phan Thị Phương Nam. Nhờ sự giúp đỡ tận tình và những chỉ bảo của Cô từ lúc bắt đầu cho tới lúc kết thúc đồ án mà em đã hoàn thành đúng thời hạn quy định và tích lũy được cho mình một lượng nền tảng kiến thức quý báu.

Tuy nhiên trong quá trình hoàn thành đề tài, do kiến thức chuyên ngành còn hạn chế nên em vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong nhận được sự quan tâm, góp ý của các thầy, cô giảng viên trong bộ môn đề tài của em được hoàn thiện hơn.

Xin chân thành cảm ơn !

Trà Vinh, ngày tháng 7 năm 2024

Sinh viên thực hiện

Nguyễn Minh Đăng

NHẬN XÉT
(Của giảng viên hướng dẫn trong đề 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 ý nghĩa thực tiễn cao, có khả năng áp dụng trong thực tế.

[illegible]

Giảng viên hướng dẫn

Phan Thị Phương Nam

BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP
(Của giảng viên hướng dẫn)

Họ và tên sinh viên: Nguyễn Minh Đăng

MSSV: 110120013

Ngành: Công nghệ thông tin

Khóa: 2020 – 2024

Tên đề tài: Xây dựng hệ thống quảng bá đặc sản các tỉnh miền Tây Nam Bộ với cơ sở dữ liệu phi quan hệ.

Họ và tên Giảng 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.

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

.....
.....

2. Ưu điểm:

Sinh viên Nguyễn Minh Đăng có tinh thần, thái độ học tập, làm việc tốt, luôn hoàn thành 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 loại đặc sản 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 tắc thiết kế giao diện.

.....
.....

4. Điểm mới đề tài:

Đề tài có tính mới là thiết kế web kết nối 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 đặc sản 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 hoặc một địa phương cụ thể.

5. Giá trị thực trên đề tài:

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

.....

.....

.....

.....

7. Đề nghị sửa chữa bổ sung:

Sinh viên Nguyễn Minh Đăng đã thực hiện chỉnh sửa theo yêu cầu của hội đồng bảo vệ Đồ án tốt nghiệp

.....

.....

.....

.....

8. Đánh giá:

Đạt yêu cầu của đồ án tốt nghiệp ngành công nghệ thông tin

.....

.....

.....

Trà Vinh, ngày 31 tháng 07 năm 2024
Giảng viên hướng dẫn
(Ký & ghi rõ họ tên)

Phan Thị Phương Nam

MỤC LỤC

CHƯƠNG 1. ĐẶT VẤN ĐỀ.....	1
1.1. Lý do chọn đề tài.....	1
1.2. Mục tiêu	1
1.3. Nội dung.....	1
1.4. Đối tượng và phạm vi nghiên cứu	2
1.5. Phương pháp nghiên cứu	2
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	4
2.1. Tìm hiểu về các tỉnh Tây Nam Bộ và các đặc sản của Tây Nam Bộ.	4
2.1.1. Tổng quan Tây Nam Bộ	4
2.1.2. Tổng quan đặc sản Tây Nam Bộ.....	5
2.2. Tìm hiểu về cơ sở dữ liệu phi quan hệ.....	7
2.2.1. Tổng quan NoSQL	7
2.2.2. Đặc điểm chính	8
2.2.3. Các dạng cơ sở dữ liệu NoSQL	8
2.2.4. Ưu điểm của NoSQL	9
2.2.5. Hạn chế của NoSQL	9
2.3. Tìm hiểu về MongoDB	10
2.3.1. Tổng quan về MongoDB	10
2.3.2. Ưu điểm của MongoDB	11
2.3.3. Nhược điểm của MongoDB	11
2.4. Tìm hiểu về NodeJS.....	12
2.4.1. Tổng quan về NodeJS	12
2.4.2. Ưu điểm của NodeJS.....	13
2.4.3. Nhược điểm của NodeJS.....	14
2.5. Tìm hiểu về Resful API	15
2.5.1. Tổng quan về Resful API	15

2.5.2. Các thành phần của Resful API.....	16
2.5.3. Các phương thức của Resful API.....	16
2.5.4. Các trạng thái của Resful API.....	17
2.5.5. Ưu điểm của Resful API.....	18
2.5.6. Nhược điểm của Resful API.....	19
2.6. Tìm hiểu Framework ReactJS.....	20
2.6.1. Tổng quan về ReactJS.....	20
2.6.2. Ưu điểm của ReactJS.....	21
2.6.3. Nhược điểm của ReactJS.....	22
CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU.....	23
3.1. Mô tả bài toán.....	23
3.2. Đặc tả hệ thống.....	24
3.3. Lược đồ dữ liệu của đặc sản.....	26
3.4. Cài đặt MongoDB và nhập dữ liệu.....	29
3.4.1. Cài đặt MongoDB.....	29
3.4.2. Cài đặt MongoDB Compass.....	29
3.4.3. Nhập dữ liệu vào MongoDB Compass.....	30
3.5. Xây dựng Back-end.....	32
3.5.1. Định nghĩa model.....	32
3.5.2. Định nghĩa các Controller.....	33
3.5.3. Định nghĩa các Router.....	47
3.6. Xây dựng giao diện.....	50
3.6.1. Phác thảo giao diện người dùng.....	50
3.6.2. Phác thảo giao diện trang quản trị.....	52
CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU.....	53
4.1. Thử nghiệm các API để lấy dữ liệu với Postman.....	53
4.2. Giao diện chức năng người dùng.....	56

4.2.1. Giao diện trang giới thiệu hệ thống.....	56
4.2.2. Giao diện trang chủ	58
4.2.3. Giao diện thông tin đặc sản	61
4.3. Giao diện chức năng người quản trị.....	62
4.3.1. Giao diện quản lý đặc sản	62
4.3.2. Giao diện quản lý loại đặc sản	64
4.3.3. Giao diện quản lý nhà sản xuất	66
4.3.4. Giao diện quản lý hạn sử dụng	68
4.3.5. Giao diện quản lý nơi bán	70
CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	72
5.1. Kết luận.....	72
5.2. Hướng phát triển	72
DANH MỤC TÀI LIỆU THAM KHẢO.....	74

DANH MỤC CÁC BẢNG

Bảng 3.1 Mô tả Actor.....	25
Bảng 3.2 Mô tả Use-case	25
Bảng 3.3 Mô tả Colletion “specialty”	27

DANH MỤC HÌNH ẢNH

Hình 2.1. Mô hình trang web sử dụng Restful API	16
Hình 3.1. Sơ đồ Use-case.....	24
Hình 3.2. Giao diện cài đặt MongoDB Community Server	29
Hình 3.3. Giao diện cài đặt MongoDB Compass	30
Hình 3.4. Giao diện tạo cơ sở dữ liệu	30
Hình 3.5. Tạo cơ sở dữ liệu.....	31
Hình 3.6. Thêm Document và Collection	31
Hình 3.7. Document sau khi thêm thành công.....	31
Hình 3.8. Phác thảo giao diện người dùng.....	50
Hình 3.9. Phác thảo giao diện trang thông tin đặc sản	51
Hình 3.10. Phác thảo giao diện trang quản trị	52
Hình 4.1. Sử dụng API lấy danh sách tất cả đặc sản với Postman	53
Hình 4.2. Sử dụng API lấy thông tin đặc sản theo ID với Postman	53
Hình 4.3. Sử dụng API lấy ra thông tin về loại của một đặc sản với Postman.....	54
Hình 4.4. Sử dụng API lấy ra thông tin nhà sản xuất của một đặc sản với Postman	54
Hình 4.5. Sử dụng API lấy ra thông tin hạn sử dụng của một đặc sản với Postman.....	55
Hình 4.6. Sử dụng API lấy ra danh sách các địa điểm bán của một đặc sản với Postman.....	55
Hình 4.7. Sử dụng API lấy ra thông tin nơi bán theo ID của một đặc sản với Postman ...	56
Hình 4.8. Giao diện giới thiệu Tây Nam Bộ.....	56
Hình 4.9. Giao diện giới thiệu một số đặc sản loại thức ăn.....	57
Hình 4.10. Giao diện giới thiệu một số đặc sản loại trái cây.....	57
Hình 4.11. Giao diện giới thiệu một số đặc sản loại rượu	57
Hình 4.12. Giao diện trang chủ.....	58
Hình 4.13. Danh sách tất cả đặc sản	58
Hình 4.14. Chức năng tìm kiếm theo tên	59
Hình 4.15. Chức năng tìm kiếm theo tỉnh thành.....	59
Hình 4.16. Tìm kiếm theo loại đặc sản	60
Hình 4.17. Chức năng tìm kiếm đồng thời	60
Hình 4.18. Giao diện thông tin của đặc sản và thông tin về loại đặc sản	61
Hình 4.19. Giao diện thông tin hạn sử dụng, nơi bán, nhà sản xuất của đặc sản	61
Hình 4.20. Giao diện quản lý đặc sản	62

Hình 4.21. Giao diện chức năng thêm đặc sản mới	62
Hình 4.22. Giao diện chức năng sửa thông tin đặc sản	63
Hình 4.23. Chức năng xoá nhiều đặc sản	63
Hình 4.24. Giao diện quản lý loại đặc sản	64
Hình 4.25. Giao diện chức năng thêm loại cho đặc sản.....	64
Hình 4.26. Giao diện chức năng sửa thông tin về loại cho đặc sản.....	65
Hình 4.27. Chức năng xoá loại của đặc sản.....	65
Hình 4.28. Giao diện quản lý nhà sản xuất.....	66
Hình 4.29. Giao diện chức năng thêm nhà sản xuất	66
Hình 4.30. Giao diện chức năng sửa thông tin nhà sản xuất	67
Hình 4.31. Chức năng xoá nhà sản xuất	67
Hình 4.32. Giao diện quản lý hạn sử dụng	68
Hình 4.33. Giao diện chức năng thêm hạn sử dụng.....	68
Hình 4.34. Giao diện chức năng sửa hạn sử dụng	69
Hình 4.35. Chức năng xoá hạn sử dụng của đặc sản	69
Hình 4.36. Giao diện quản lý địa điểm bán	70
Hình 4.37. Giao diện chức năng thêm địa điểm bán.....	70
Hình 4.38. Giao diện chức năng sửa địa điểm bán	71
Hình 4.39. Chức năng xoá địa điểm bán.....	71

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa
ĐBSCL	Đồng bằng sông Cửu Long
NoSQL	Not Only SQL

CHƯƠNG 1. ĐẶT VẤN ĐỀ

1.1. Lý do chọn đề tài

Miền Tây Nam Bộ, vùng đất nổi tiếng với vô số đặc sản phong phú và đa dạng, đang đối mặt với thách thức trong việc quảng bá các sản phẩm này ra thị trường rộng lớn hơn. Mặc dù những đặc sản của vùng đất này rất đặc biệt và có giá trị cao, việc giới thiệu và tiếp thị chúng vẫn còn hạn chế. Để giải quyết vấn đề này, cần một hệ thống quảng bá hiệu quả nhằm nâng cao nhận thức của người tiêu dùng và mở rộng thị trường tiêu thụ.

Thay vì tiếp tục sử dụng cơ sở dữ liệu quan hệ truyền thống, việc áp dụng cơ sở dữ liệu phi quan hệ sẽ mang lại nhiều lợi ích vượt trội. Cơ sở dữ liệu phi quan hệ có khả năng xử lý và lưu trữ thông tin đa dạng về các đặc sản của Miền Tây Nam Bộ một cách linh hoạt và hiệu quả hơn, giúp hệ thống quảng bá hoạt động trơn tru và đáng tin cậy.

Việc đẩy mạnh quảng bá các đặc sản không chỉ góp phần tăng doanh thu mà còn mở ra nhiều cơ hội kinh doanh mới, kết nối chặt chẽ hơn giữa người sản xuất và người tiêu dùng. Điều này không chỉ mang lại lợi ích kinh tế mà còn thúc đẩy sự phát triển bền vững của kinh tế địa phương. Hơn nữa, hệ thống quảng bá sẽ cung cấp những dữ liệu quan trọng, hỗ trợ xây dựng chiến lược phát triển nông nghiệp và du lịch một cách bền vững, tạo đà cho sự phát triển toàn diện của vùng đất Miền Tây Nam Bộ trong tương lai.

1.2. Mục tiêu

- Tìm hiểu cụ thể về các đặc sản của 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

Phân tích và thiết kế hệ thống:

- Tìm hiểu các đặc sản miền Tây Nam bộ.
- Nghiên cứu cách thiết kế mô hình dữ liệu phi quan hệ.

Triển khai và thử nghiệm:

- Xây dựng mô hình dữ liệu phi quan hệ cho các loại đặc sản.

- Chuẩn bị dữ liệu và triển khai hệ thống trên môi trường thử nghiệm.
- Xây dựng hệ thống thống web quảng bá các đặc sản với thông tin đã tìm hiểu và mô hình dữ liệu đã có.
- Tiến hành cài đặt thử nghiệm và kiểm thử hệ thống.

Kết luận và đề xuất:

- Tóm tắt kết quả nghiên cứu và đánh giá ưu nhược điểm của hệ thống.
- Đề xuất các hướng phát triển và cải thiện cho hệ thống trong tương lai

1.4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu:

- Các đặc sản ở 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

- Đặc sản miền Tây Nam Bộ.
- Hệ quản trị cơ sở dữ liệu MongoDB.
- NodeJS, Resful API, Framework ReactJS.

1.5. Phương pháp nghiên cứu

Phương pháp nghiên cứu lý thuyết:

- Nghiên cứu tài liệu về vị trí địa lý, văn hoá, đặc sản của các tỉnh Tây Nam Bộ;
- Nghiên cứu tài liệu về xây dựng cơ sở dữ liệu phi quan hệ;
- Nghiên cứu cách cài đặt và vận hành của Hệ quản trị Cơ sở dữ liệu phi quan hệ;
- Nghiên cứu các tài liệu thiết kế và cài đặt hệ thống web.

Phương pháp thực nghiệm:

- Xây dựng mô hình cơ sở dữ liệu phi quan hệ;
- Cài đặt các công cụ cần thiết để hiện thực hóa hệ thống thiết kế;
- Phát triển các tính năng và giao diện của hệ thống theo đặt ta đã thiết kế;
- Kiểm thử và triển khai thử nghiệm

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Tìm hiểu về các tỉnh Tây Nam Bộ và các đặc sản của Tây Nam Bộ.

2.1.1. Tổng quan Tây Nam Bộ

Tây Nam Bộ, hay còn gọi là vùng Đồng bằng sông Cửu Long (ĐBSCL), là một trong những khu vực quan trọng và đặc trưng của Việt Nam. Đây là vùng đất trù phú và có vai trò quan trọng trong phát triển kinh tế, văn hóa và du lịch của cả nước. Dưới đây là một số đặc điểm chính về Tây Nam Bộ: [5]

Vị trí địa lý: Tây Nam Bộ nằm ở phía nam của Việt Nam, giáp Campuchia ở phía tây bắc và Biển Đông ở phía đông nam. Vùng này bao gồm 13 tỉnh, bao gồm Long An, Tiền Giang, Bến Tre, Vĩnh Long, Đồng Tháp, Trà Vinh, Hậu Giang, Sóc Trăng, An Giang, Kiên Giang, Cần Thơ, Bạc Liêu và Cà Mau.

Khí hậu: Tây Nam Bộ có khí hậu nhiệt đới gió mùa, với hai mùa rõ rệt: mùa mưa từ tháng 5 đến tháng 11 và mùa khô từ tháng 12 đến tháng 4. Lượng mưa lớn và nguồn nước phong phú từ sông Cửu Long tạo điều kiện thuận lợi cho nông nghiệp.

Nông nghiệp: Tây Nam Bộ là vựa lúa lớn nhất của Việt Nam, chiếm phần lớn sản lượng lúa gạo cả nước. Ngoài lúa, khu vực này còn trồng nhiều loại cây ăn quả như xoài, bưởi, chôm chôm, và dừa.

Thủy sản: Với hệ thống sông ngòi và kênh rạch chằng chịt, thủy sản là một ngành kinh tế quan trọng. Khu vực này nổi tiếng với nuôi tôm, cá tra, cá basa, và các loại hải sản khác.

Công nghiệp và Dịch vụ: Tây Nam Bộ cũng đang phát triển các khu công nghiệp và khu chế xuất, cũng như ngành dịch vụ và du lịch.

Dân tộc và Ngôn ngữ: Tây Nam Bộ là nơi sinh sống của nhiều dân tộc khác nhau, bao gồm người Kinh, người Khmer, người Hoa, và người Chăm. Mỗi dân tộc có văn hóa, phong tục tập quán và lễ hội riêng.

Ẩm thực: Ẩm thực Tây Nam Bộ nổi tiếng với các món ăn đậm đà hương vị miền sông nước như cá lóc nướng trui, lẩu mắm, bánh xèo, và các loại trái cây tươi ngon.

Lễ hội: Khu vực này có nhiều lễ hội độc đáo, tiêu biểu như lễ hội Óoc Om Bóc của người Khmer, lễ hội đua ghe ngo, và lễ hội Bà Chúa Xứ ở Châu Đốc.

Điểm du lịch nổi bật: Tây Nam Bộ hấp dẫn du khách với nhiều điểm đến như Cần Thơ (chợ nổi Cái Răng), Bến Tre (du lịch sinh thái), An Giang (rừng tràm Trà Sư), và Phú Quốc (bãi biển đẹp, khu du lịch sinh thái).

Du lịch sông nước: Du khách có thể trải nghiệm cuộc sống miền sông nước qua các tour du lịch trên sông, thăm chợ nổi, và khám phá các cù lao.

Đặc sản: nổi tiếng với nhiều đặc sản ẩm thực hấp dẫn, phản ánh nét văn hóa ẩm thực đặc trưng của vùng sông nước.

2.1.2. Tổng quan đặc sản Tây Nam Bộ

Tây Nam Bộ có 13 tỉnh thành, mỗi tỉnh có nhiều đặc sản khác nhau: [4]

Trà Vinh:

- **Bánh tét Trà Cuôn:** Bánh tét có lớp vỏ xanh từ lá cẩm, nhân đậu xanh và thịt mỡ.
- **Dừa sáp Cầu Kè:** Dừa có lớp cơm dày, dẻo và nước cốt đậm đà.
- **Chả hoa Năm Thuy:** Chả có lớp vỏ ngoài mịn, bên trong là nhân thịt và các loại rau củ.
- **Bún suông:** Bún với nước lèo đậm đà, tôm giã nhuyễn tạo thành hình sông dài, ăn kèm với rau sống.
- **Bún nước lèo:** Món bún với nước dùng từ mắm bò hóc đặc trưng của Trà Vinh.

Long An:

- **Dưa hấu Long Trì:** dưa có lớp vỏ mỏng, ruột đỏ, vị ngọt thanh.
- **Rượu gò đen:** Rượu gạo truyền thống, nổi tiếng với hương vị đặc trưng, nồng nàn.
- **Lạp xưởng tươi:** Lạp xưởng Long An có vị ngọt, béo, thường được chế biến từ thịt nạc và mỡ heo.
- **Gạo Nàng Thơm Chợ Đào:** Gạo đặc sản nổi tiếng với hạt dài, thơm ngon.

Tiền Giang:

- **Hủ tiếu Mỹ Tho:** Món ăn có sợi hủ tiếu dai, nước dùng thơm ngon từ xương heo và tôm khô.
- **Vú sữa Lò Rèn Vĩnh Kim:** Loại vú sữa nổi tiếng với trái to, ngọt và thơm.
- **Bánh bèo chợ Hàng Bông:** Bánh bèo mềm mịn, ăn kèm với nước cốt dừa và tôm khô.

Hậu Giang:

- **Chả cá thác lác:** Chả cá được làm từ cá thác lác, có vị ngọt, dai, thơm ngon.
- **Sỏi mằm:** Món ăn độc đáo, thường được chế biến từ gạo nếp và các loại đậu.
- **Đọt choại:** Rau đọt choại thường được dùng để làm món luộc hoặc xào.
- **Khô cá lóc:** Cá lóc được phơi khô, thường được nướng hoặc chiên giòn.

Bến Tre:

- **Kẹo dừa Bến Tre:** Kẹo được làm từ dừa và đường, có nhiều hương vị khác nhau.
- **Rượu dừa:** Rượu được ủ từ nước dừa, có hương vị đặc trưng.
- **Bánh tráng Mỹ Lồng:** Bánh tráng dày, giòn, thơm mùi dừa.
- **Bánh tráng sữa:** Bánh tráng được làm từ bột gạo và nước cốt dừa, thơm ngon và mềm mịn.

Vĩnh Long:

- **Bưởi Năm Roi:** Bưởi có vị ngọt, mọng nước và thơm ngon.
- **Cam sành Tam Bình:** Cam có vỏ sần sùi, màu vàng tươi, vị ngọt thanh.
- **Khoai lang Bình Tân:** Khoai lang có vị ngọt, bùi, thường được chế biến thành nhiều món ăn ngon.

Cần Thơ:

- **Bánh tét lá cẩm:** Bánh tét có màu tím từ lá cẩm, nhân đậu xanh và thịt mỡ.
- **Vú sữa Cần Thơ:** Loại vú sữa có vị ngọt, mọng nước và thơm ngon.
- **Nem nướng Cái Răng:** Nem nướng từ thịt heo, ăn kèm bánh tráng và rau sống.

Sóc Trăng:

- **Bánh pía Sóc Trăng:** Bánh pía có lớp vỏ mỏng, nhân đậu xanh, sầu riêng và trứng muối.
- **Lạp xưởng Sóc Trăng:** Lạp xưởng có vị ngọt, béo và thơm ngon.
- **Bánh ông:** Được làm từ bột gạo và dừa nạo, có vị béo, ngọt thanh.

Bạc Liêu:

- **Bánh củ cải:** Bánh được làm từ củ cải trắng, nhân tôm thịt, chiên giòn.
- **Mắm ba khía:** Mắm làm từ ba khía, thường được ăn kèm với cơm hoặc bún.
- **Bánh xèo Bạc Liêu:** Bánh xèo có lớp vỏ giòn, nhân tôm, thịt, giá đỗ, ăn kèm rau sống.
- **Nhân da bò:** Nhân có vỏ mỏng, cơm thịt dày, hạt nhỏ.

Cà Mau:

- **Cua Cà Mau:** Cua biển Cà Mau nổi tiếng với thịt ngọt, chắc.
- **Mắm cá lóc:** Mắm được làm từ cá lóc, ủ thơm ngon, dùng để ăn kèm với cơm hoặc bún.
- **Tôm khô Rạch Gốc:** Tôm khô có vị ngọt tự nhiên, thường được dùng làm nguyên liệu cho nhiều món ăn.
- **Mật ong rừng U Minh:** Mật ong được khai thác từ rừng U Minh, có hương vị đặc trưng và nhiều giá trị dinh dưỡng.

Đồng Tháp:

- **Nem Lai Vung:** Nem chua có vị chua ngọt, thường được gói trong lá chuối.
- **Hồng sen Đồng Tháp:** Hạt sen to, bùi, thường được dùng để nấu chè hoặc ăn sống.
- **Bánh phồng tôm Sa Giang:** Bánh phồng tôm giòn, thơm, thường được chiên giòn trước khi ăn.
- **Hủ tiếu Sa Đéc:** Sợi hủ tiếu dai, nước dùng nấu từ xương và tôm khô, ăn kèm với thịt heo, tôm, lòng đỏ trứng và rau thơm.

An Giang:

- **Mắm Châu Đốc:** Các loại mắm cá linh, cá sặc được ủ thơm ngon, đậm đà.
- **Bánh bò thốt nốt:** Bánh bò làm từ bột gạo và đường thốt nốt, có vị ngọt, thơm.
- **Gỏi sầu đâu:** Gỏi được làm từ lá sầu đâu, tôm, thịt và các loại rau thơm.

Kiên Giang:

- **Nước mắm Phú Quốc:** Nước mắm được làm từ cá cơm và muối, có hương vị đậm đà.
- **Bánh canh ghẹ Hà Tiên:** Món bánh canh với ghẹ tươi, nước dùng ngọt từ hải sản.
- **Hạt tiêu Phú Quốc:** Hạt tiêu thơm, cay nồng, nổi tiếng trên khắp cả 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") được giới thiệu lần đầu tiên vào năm 1998 để chỉ các cơ sở dữ liệu không sử dụng giao diện SQL truyền thống. NoSQL đã phát triển mạnh mẽ trong suốt những năm 2000, đặc biệt trong bối

cảnh sự mở rộng nhanh chóng của Internet. Sự phát triển này phản ánh nhu cầu ngày càng cao về các hệ thống cơ sở dữ liệu linh hoạt, có khả năng mở rộng và xử lý khối lượng lớn dữ liệu không cấu trúc hoặc bán cấu trúc, điều mà các cơ sở dữ liệu quan hệ truyền thống gặp khó khăn trong việc đáp ứng.

Cơ sở dữ liệu NoSQL là một Hệ thống quản lý dữ liệu không quan hệ (non-relational Data Management System) có lược đồ (schema) linh hoạt, dễ mở rộng. Mục đích chính của việc sử dụng cơ sở dữ liệu NoSQL là dành cho các kho dữ liệu phân tán với nhu cầu lưu trữ dữ liệu lớn. NoSQL được sử dụng cho dữ liệu lớn và ứng dụng web thời gian thực. Chẳng hạn các công ty như Twitter, Facebook và Google thu thập hàng terabyte dữ liệu người dùng mỗi ngày. [1]

2.2.2. Đặc điểm chính

Linh hoạt về mô hình dữ liệu: NoSQL hỗ trợ nhiều mô hình dữ liệu khác nhau, bao gồm key-value, document, column-family, và graph. Điều này cho phép bạn chọn mô hình phù hợp nhất với nhu cầu cụ thể của ứng dụng hoặc dữ liệu bạn đang xử lý.

Khả năng mở rộng: NoSQL dễ dàng mở rộng quy mô bằng cách thêm nhiều máy chủ hơn, thay vì chỉ nâng cấp phần cứng của máy chủ hiện tại. Điều này giúp hệ thống có thể mở rộng linh hoạt để đáp ứng nhu cầu ngày càng tăng mà không gặp phải các hạn chế về phần cứng.

Hiệu suất cao: NoSQL được tối ưu hóa để xử lý khối lượng lớn dữ liệu với tốc độ cao. Nó phù hợp cho các ứng dụng yêu cầu hiệu suất cao và khả năng xử lý dữ liệu lớn nhanh chóng.

Tính linh hoạt và phi cấu trúc: NoSQL cho phép lưu trữ và quản lý dữ liệu phi cấu trúc như JSON, XML, hoặc các định dạng khác mà không cần định nghĩa schema cố định trước. Điều này cung cấp sự linh hoạt cao trong việc lưu trữ và truy vấn dữ liệu mà không cần phải tuân theo cấu trúc bảng cứng nhắc của cơ sở dữ liệu quan hệ.

2.2.3. Các dạng cơ sở dữ liệu NoSQL

Có nhiều hệ thống cơ sở dữ liệu NoSQL như: MongoDB, RavenDB, Redis,... Xong có thể chia NoSQL thành 4 loại: [1]

- Key-Value stores;
- Document stores;
- Column-Family Databases;
- Graph Databases;

2.2.4. Ưu điểm của NoSQL

Khả năng mở rộng cao:

Mở rộng ngang: Dễ dàng mở rộng bằng cách thêm nhiều máy chủ thay vì nâng cấp phần cứng của một máy chủ duy nhất.

Tính linh hoạt: Dễ dàng thêm hoặc bớt các nút trong cụm mà không ảnh hưởng đến hiệu suất.

Hiệu suất cao:

Xử lý dữ liệu lớn: Tối ưu hóa cho việc xử lý khối lượng lớn dữ liệu và truy vấn tốc độ cao.

Tối ưu cho dữ liệu phi cấu trúc: Hiệu quả trong việc lưu trữ và truy vấn dữ liệu phi cấu trúc hoặc bán cấu trúc.

Linh hoạt về mô hình dữ liệu:

Không cần Schema cố định: Không cần định nghĩa trước schema, cho phép thay đổi cấu trúc dữ liệu linh hoạt.

Hỗ trợ nhiều mô hình dữ liệu: Đa dạng mô hình như key-value, document, column-family, và graph, phù hợp với nhiều loại ứng dụng.

Khả năng chịu lỗi tốt:

Phân tán dữ liệu: Dữ liệu được phân tán trên nhiều nút, giúp hệ thống chịu lỗi tốt hơn.

Khả năng phục hồi nhanh: Nhanh chóng phục hồi sau các sự cố hệ thống.

2.2.5. Hạn chế của NoSQL

Thiếu tính nhất quán:

Nhất quán cuối (Eventual Consistency): Một số hệ thống NoSQL ưu tiên tính khả dụng và phân tán hơn là tính nhất quán ngay lập tức, dẫn đến trạng thái dữ liệu không đồng nhất tạm thời.

Hạn chế về ngôn ngữ truy vấn:

Không có chuẩn thống nhất: Không có ngôn ngữ truy vấn chuẩn như SQL, mỗi hệ thống NoSQL có ngôn ngữ và API riêng, gây khó khăn trong việc học và sử dụng.

Giới hạn truy vấn phức tạp: Không tối ưu cho các truy vấn phức tạp và liên kết giữa nhiều bảng dữ liệu.

Cần quản lý phức tạp:

Quản trị hệ thống: Yêu cầu kiến thức sâu về quản trị hệ thống và khả năng cấu hình phức tạp để đảm bảo hiệu suất và tính ổn định.

Khó khăn trong việc thay đổi mô hình dữ liệu: Thay đổi cấu trúc dữ liệu hoặc mô hình lưu trữ có thể phức tạp và tốn kém.

Hỗ trợ giới hạn từ cộng đồng:

Tài liệu và hỗ trợ: So với SQL, NoSQL có ít tài liệu, hướng dẫn và hỗ trợ từ cộng đồng hơn, có thể gây khó khăn trong việc triển khai và khắc phục sự cố.

2.3. Tìm hiểu về MongoDB

2.3.1. Tổng quan về MongoDB

MongoDB là một cơ sở dữ liệu đa nền tảng sử dụng các khái niệm Collection và Document, lưu trữ dữ liệu dưới dạng tài liệu JSON-like (BSON) trong các bộ sưu tập. Nó cung cấp hiệu suất cao, tính khả dụng và mở rộng dễ dàng, với khả năng sao chép dữ liệu trên nhiều nodes để đảm bảo độ tin cậy và khả năng chịu lỗi. MongoDB hỗ trợ sharding để phân tán tải công việc và mở rộng dễ dàng mà không yêu cầu cấu trúc dữ liệu cố định. [1]

Lịch sử hình thành của MongoDB: [6]

Công ty phần mềm 10gen, có trụ sở tại Mỹ, bắt đầu phát triển MongoDB vào năm 2007 như một phần của sản phẩm nền tảng dưới dạng dịch vụ, một dự án đã được lên kế hoạch từ trước. Đến năm 2009, họ chuyển sang mô hình phát triển mã nguồn mở và bắt đầu cung cấp hỗ trợ thương mại cùng với các dịch vụ khác. Đến năm 2013, 10gen chính thức đổi tên thành MongoDB Inc.

Vào ngày 20 tháng 10 năm 2017, MongoDB trở thành một công ty niêm yết công khai trên sàn NASDAQ với tên mã giao dịch là MDB và giá IPO là 24 USD mỗi cổ phiếu. Với bản phát hành ổn định 4.0.4 vào ngày 8 tháng 11 năm 2018, giấy phép phần mềm của MongoDB đã chuyển từ AGPL 3.0 sang SSPL.

Vào ngày 30 tháng 10 năm 2019, MongoDB hợp tác với Alibaba Cloud để cung cấp giải pháp MongoDB dưới dạng dịch vụ cho khách hàng của Alibaba Cloud, cho phép họ tiếp cận với các dịch vụ được quản lý từ các trung tâm dữ liệu toàn cầu của Alibaba.

2.3.2. Ưu điểm của MongoDB

Mô hình dữ liệu linh hoạt:

MongoDB sử dụng mô hình dữ liệu tài liệu, cho phép lưu trữ dữ liệu phi cấu trúc và có thể thay đổi cấu trúc dễ dàng mà không cần thay đổi schema.

Khả năng mở rộng tốt:

MongoDB hỗ trợ mở rộng theo chiều ngang (horizontal scaling) thông qua sharding, cho phép phân chia dữ liệu trên nhiều máy chủ và xử lý tải cao.

Hiệu suất cao:

Dữ liệu được lưu trữ dưới dạng tài liệu, giúp truy xuất dữ liệu nhanh chóng và hiệu quả hơn, đặc biệt với dữ liệu lớn và phi cấu trúc.

Tính linh hoạt trong truy vấn:

MongoDB cung cấp ngôn ngữ truy vấn mạnh mẽ và linh hoạt, cho phép thực hiện các truy vấn phức tạp và điều kiện đa dạng trên dữ liệu.

Tính sẵn sàng cao và bền bỉ:

MongoDB được thiết kế để đảm bảo tính nhất quán và bền bỉ của dữ liệu, với khả năng sao lưu và phục hồi dữ liệu tự động.

2.3.3. Nhược điểm của MongoDB

Khả năng tăng tải nhờ mở rộng ngang:

Mặc dù MongoDB hỗ trợ mở rộng ngang, nhưng quá trình này có thể gặp khó khăn khi dữ liệu tăng lên đột ngột và cần sự mở rộng. Có thể yêu cầu quản lý phân phối dữ liệu và xử lý transaction một cách cẩn thận để tránh sự phức tạp.

Sử dụng lượng bộ nhớ đáng kể:

MongoDB cần sử dụng lượng bộ nhớ đáng kể để hoạt động hiệu quả. Điều này có thể tạo ra vấn đề nếu ứng dụng của người dùng đang chạy trên các máy chủ có tài nguyên hạn chế.

Chưa hỗ trợ Transactions nguồn gốc (Native transactions):

Trước phiên bản MongoDB 4.0, hệ thống này không hỗ trợ transactions nguồn gốc (native transactions) qua nhiều document, điều này có thể tạo ra khó khăn đối với các ứng dụng yêu cầu giao dịch phức tạp.

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 người dùng 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.

Vấn đề bảo mật:

Có thể gặp phải vấn đề về bảo mật nếu MongoDB không được cấu hình và triển khai đúng cách. Cần phải thiết lập các biện pháp bảo mật để đảm bảo an toàn cho dữ liệu.

2.4. Tìm hiểu về NodeJS

2.4.1. Tổng quan về NodeJS

Node.js là một môi trường thực thi mã JavaScript dựa trên Chrome V8 JavaScript engine, được xây dựng để chạy trên máy chủ. Đặc điểm nổi bật của Node.js là khả năng xử lý sự kiện theo hướng không đồng bộ (asynchronous event-driven), giúp ứng dụng xử lý hàng ngàn kết nối đồng thời mà không cần tạo ra một luồng mới cho mỗi kết nối. Điều này làm cho Node.js trở thành một lựa chọn phổ biến cho việc xây dựng các ứng dụng mạng thời gian thực và ứng dụng web có hiệu suất cao. [8]

Node.js không chỉ hỗ trợ nhiều nền tảng mà còn giúp đơn giản hóa quy trình phát triển và triển khai ứng dụng. Sự xuất hiện của các framework nổi tiếng như Express.js và công nghệ như Socket.io cung cấp những công cụ mạnh mẽ để phát triển ứng dụng web và real-time. Được đánh giá cao với tính linh hoạt và hiệu suất, Node.js đang ngày càng trở thành lựa chọn hàng đầu cho việc xây dựng các dự án phức tạp và ứng dụng đa nhiệm trên nền tảng web.

Lịch sử hình thành của NodeJS: [8]

Node.js ban đầu được viết bởi Ryan Dahl vào năm 2009, khoảng 13 năm sau khi giới thiệu môi trường JavaScript phía máy chủ đầu tiên, LiveWire Pro Web của Netscape. Phiên bản ban đầu chỉ hỗ trợ Linux và Mac OS X. Quá trình phát triển và bảo trì được Ryan Dahl dẫn đầu và sau đó được Joyent tài trợ.

Ryan Dahl đã chỉ trích khả năng giới hạn của Apache HTTP Server trong việc xử lý nhiều kết nối đồng thời (10,000+), cũng như mô hình lập trình tuần tự chiếm ưu thế, trong đó ứng dụng có thể chặn toàn bộ quy trình hoặc tạo nên nhiều ngăn xếp thực thi cho các kết nối đồng thời. Dahl trình diễn dự án này tại hội nghị European JSConf đầu tiên vào ngày 8 tháng 11 năm 2009. Node.js kết hợp giữa Google's V8 JavaScript engine, một vòng lặp sự kiện và một API I/O cấp thấp.

Tháng 1 năm 2010, một trình quản lý gói dành cho môi trường Node.js được giới thiệu với tên là npm. Trình quản lý gói này cho phép các lập trình viên xuất bản và chia sẻ gói Node.js, cũng như mã nguồn đi kèm, và được thiết kế để đơn giản hóa quá trình cài đặt, cập nhật và gỡ cài đặt các gói.

Tháng 6 năm 2011, Microsoft và Joyent triển khai một phiên bản Windows native của Node.js. Phiên bản đầu tiên hỗ trợ Windows được phát hành vào tháng 7 năm 2011.

Tháng 1 năm 2012, Ryan Dahl nhường quyền quản lý dự án cho tác giả npm Isaac Schlueter. Tháng 1 năm 2014, Schlueter thông báo rằng Timothy J. Fontaine sẽ lãnh đạo dự án.

Tháng 12 năm 2014, Fedor Indutny tạo ra io.js, một nhánh của Node.js do không hài lòng với quản lý của Joyent, với ý định tạo ra một cộng đồng mở rộng với một ủy ban kỹ thuật độc lập. Node.js Foundation được thành lập để hòa giải Node.js và io.js dưới một bức màn duy nhất và được công bố vào tháng 2 năm 2015. Sáp nhập được thực hiện vào tháng 9 năm 2015, mang theo Node.js v0.12 và io.js v3.3 để tạo ra Node v4.0.

Năm 2019, JS Foundation và Node.js Foundation hợp nhất để tạo ra OpenJS Foundation.

Ngày 6 tháng 9 năm 2023, Node.js 20.6.0 được phát hành với sự thêm mới của hỗ trợ tích hợp cho tệp chứa biến môi trường .env.

2.4.2. Ưu điểm của NodeJS

Hiệu suất cao:

Node.js sử dụng JavaScript để xử lý cả yêu cầu client và server trong một ngôn ngữ duy nhất, giúp giảm thời gian phản hồi và tăng hiệu suất.

Khả năng đồng thời (concurrency):

Node.js sử dụng kiến trúc không đồng bộ và sự kiện lớn, cho phép xử lý hàng nghìn kết nối đồng thời mà không cần tạo ra các luồng mới. Điều này giúp cải thiện khả năng

mở rộng và hiệu suất của ứng dụng. Node.js thích hợp cho việc xây dựng ứng dụng real-time như chat, trò chơi trực tuyến, cập nhật thời gian thực và ứng dụng streaming.

Khả năng mở rộng:

Node.js hỗ trợ mô hình không đồng bộ và xử lý sự kiện non-blocking I/O, cho phép xử lý nhiều yêu cầu cùng một lúc mà không cần tạo ra các luồng mới, giúp tăng khả năng mở rộng.

Cộng đồng mạnh mẽ:

Node.js có một cộng đồng phát triển lớn, đam mê và năng động. Sự hỗ trợ từ cộng đồng làm cho việc giải quyết vấn đề và tìm giải pháp trở nên dễ dàng. Node.js thường xuyên nhận được cập nhật và cải tiến từ cộng đồng, giúp đảm bảo rằng nó luôn duy trì được sự hiện đại và an toàn.

Tính linh hoạt:

Node.js hỗ trợ việc xây dựng ứng dụng web, ứng dụng thời gian thực, API và nhiều loại ứng dụng khác, giúp phù hợp với nhiều loại dự án khác nhau.

2.4.3. Nhược điểm của NodeJS

Xử lý công việc tính toán nặng kém:

Node.js không phải là sự lựa chọn tốt nếu ứng dụng của người dùng đòi hỏi nhiều tính toán nặng, chẳng hạn như xử lý hình ảnh lớn hay tính toán phức tạp. Do cách Node.js xử lý đơn luồng, một công việc lâu dài có thể tạo ra độ trễ cho toàn bộ ứng dụng.

Chạy đơn luồng và chờ I/O:

Node.js hoạt động trong mô hình chạy đơn luồng, điều này có nghĩa là nó chỉ xử lý một công việc tại một thời điểm. Khi phải thực hiện các công việc chờ đợi như đọc dữ liệu từ cơ sở dữ liệu, nó có thể tạo ra độ trễ cho các công việc khác.

Callback Hell:

Việc sử dụng callback trong Node.js có thể dẫn đến hiện tượng "callback hell", khi mã trở nên khó đọc và khó bảo trì do sự lồng ghép của nhiều callback.

Quản lý Dependency khó khăn:

Node.js sử dụng một công cụ quản lý gói gọi là npm, nhưng với sự phổ biến của nó, quản lý các phụ thuộc có thể trở nên phức tạp. Sự phụ thuộc giữa các gói có thể tạo ra khó khăn khi cần duy trì và cập nhật ứng dụng.

Thay đổi liên tục trong API:

Có thể xuất hiện những thay đổi không dự kiến trong API của Node.js khi chuyển sang các phiên bản mới. Điều này có thể làm cho mã nguồn không tương thích và yêu cầu thêm công việc để cập nhật mã nguồn.

Thiếu sự đồng nhất trong thư viện và Framework:

Trái ngược với một số ngôn ngữ khác như Java hoặc .NET, Node.js thiếu sự đồng nhất trong các thư viện và framework, làm cho quá trình phát triển và bảo trì ứng dụng có thể phức tạp hơn.

Vấn đề bảo mật:

Với khả năng thực hiện các hoạt động không an toàn, quản lý bảo mật có thể trở nên khó khăn. Nếu triển khai và cấu hình không an toàn, có thể tăng rủi ro về bảo mật cho ứng dụng.

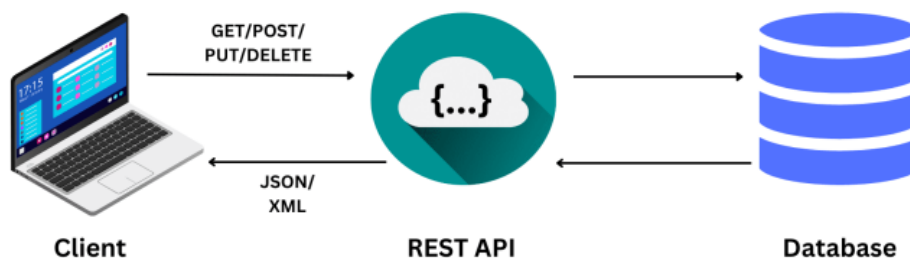
2.5. Tìm hiểu về Resful API

2.5.1. Tổng quan về Resful API

RESTful API (Representational State Transfer API) là một kiến trúc thiết kế cho việc xây dựng các dịch vụ web linh hoạt, đơn giản và dễ dàng tích hợp với nhiều hệ thống khác nhau. Khái niệm này được đưa ra bởi Roy Fielding trong luận văn tiến sĩ của ông vào năm 2000. RESTful API dựa trên các nguyên tắc cơ bản của REST (Representational State Transfer), một mô hình truyền thông không trạng thái, có khả năng mở rộng và dễ dàng quản lý.

RESTful API hoạt động dựa trên các phương thức HTTP tiêu chuẩn như GET, POST, PUT, DELETE, giúp việc truy cập và thao tác dữ liệu trở nên dễ dàng hơn. Với mô hình truyền thông không trạng thái, mỗi yêu cầu từ client đến server chứa tất cả thông tin cần thiết để server có thể hiểu và xử lý yêu cầu đó. Điều này giúp RESTful API trở nên rất linh hoạt và dễ dàng mở rộng, vì không cần phải duy trì trạng thái của client giữa các yêu cầu.

Sự đơn giản và hiệu quả của RESTful API đã khiến nó trở thành lựa chọn phổ biến cho việc xây dựng các dịch vụ web hiện đại. Với RESTful API, các nhà phát triển có thể dễ dàng tích hợp các hệ thống khác nhau, tạo ra các ứng dụng web và mobile mạnh mẽ, và cung cấp trải nghiệm người dùng mượt mà và nhất quán. [9]



Hình 2.1. Mô hình trang web sử dụng Restful API

2.5.2. Các thành phần của Resful API

API (Application Programming Interface) là một tập hợp các quy tắc và cơ chế cho phép các ứng dụng hoặc các thành phần phần mềm tương tác với nhau. API cung cấp cách để một ứng dụng truy cập và sử dụng dữ liệu hoặc chức năng từ một ứng dụng khác. Thông qua API, các ứng dụng có thể trả về dữ liệu cần thiết cho người dùng dưới các định dạng phổ biến như JSON hoặc XML.

REST (REpresentational State Transfer) là một phong cách kiến trúc cho việc thiết kế API. REST sử dụng các phương thức HTTP đơn giản để tạo ra các giao tiếp hiệu quả giữa các máy tính. Thay vì sử dụng một URL để xử lý thông tin người dùng, REST thực hiện các yêu cầu HTTP như GET, POST, DELETE, v.v., tới các URL cụ thể để xử lý và tương tác với dữ liệu. REST giúp xây dựng các API linh hoạt và dễ sử dụng, cho phép các ứng dụng giao tiếp với nhau một cách hiệu quả thông qua các phương thức HTTP chuẩn.

2.5.3. Các phương thức của Resful API

[GET]: Phương thức này được sử dụng để đọc và truy xuất dữ liệu từ một tài nguyên cụ thể. Khi một yêu cầu GET được gửi đến máy chủ, nó sẽ trả về thông tin chi tiết của tài nguyên được yêu cầu, thường dưới dạng JSON hoặc XML.

[POST]: Được sử dụng để tạo mới một tài nguyên trên máy chủ. Khi một yêu cầu POST được gửi, dữ liệu đi kèm theo yêu cầu sẽ được sử dụng để tạo một tài nguyên mới. Ví dụ, khi bạn gửi một biểu mẫu đăng ký trên một trang web, yêu cầu POST sẽ gửi thông tin của bạn đến máy chủ để tạo một tài khoản mới.

[PUT]: Phương thức này được sử dụng để cập nhật một tài nguyên hiện có hoặc tạo mới tài nguyên nếu nó chưa tồn tại. Dữ liệu đi kèm theo yêu cầu PUT sẽ thay thế hoàn toàn thông tin hiện có của tài nguyên hoặc tạo một tài nguyên mới với thông tin đó.

[DELETE]: Được sử dụng để xóa một tài nguyên cụ thể trên máy chủ. Khi một yêu cầu DELETE được gửi, nó sẽ yêu cầu máy chủ xóa tài nguyên được xác định.

[PATCH]: Phương thức này được sử dụng để cập nhật một phần nhỏ của tài nguyên thay vì thay thế toàn bộ thông tin như với PUT. Điều này giúp giảm lượng dữ liệu được gửi đi, làm cho quá trình cập nhật hiệu quả hơn.

[OPTIONS]: Yêu cầu này được sử dụng để lấy thông tin về các phương thức HTTP được hỗ trợ trên một tài nguyên cụ thể hoặc trên toàn bộ ứng dụng. Điều này giúp người dùng biết được những hành động nào có thể thực hiện trên tài nguyên đó.

[HEAD]: Tương tự như GET, nhưng yêu cầu HEAD chỉ trả về phần tiêu đề của phản hồi mà không có phần nội dung thực tế. Điều này thường được sử dụng để kiểm tra trạng thái của một tài nguyên hoặc để xem metadata mà không cần tải toàn bộ nội dung.

2.5.4. Các trạng thái của Resful API

Các trạng thái HTTP trong RESTful API thường được sử dụng để chỉ định kết quả của các yêu cầu đến máy chủ. Dưới đây là các trạng thái phổ biến: [9]

- **200 (OK):** Yêu cầu đã được thực hiện thành công, và máy chủ trả về kết quả. Đây là mã trạng thái thường được sử dụng để cho biết rằng mọi thứ diễn ra đúng như mong đợi.
- **201 (Created):** Yêu cầu đã thành công và dẫn đến việc tạo ra một tài nguyên mới. Thường được trả về sau khi thực hiện một yêu cầu POST để tạo tài nguyên.
- **204 (No Content):** Yêu cầu đã được xử lý thành công, nhưng không có dữ liệu để trả về. Thường được sử dụng khi tài nguyên đã được xóa thành công hoặc khi một yêu cầu PUT không cần trả về dữ liệu.
- **304 (Not Modified):** Dữ liệu từ yêu cầu trước đó vẫn còn hợp lệ, và máy chủ không cần gửi lại dữ liệu. Đây là cách để chỉ định rằng client có thể sử dụng dữ liệu đã được lưu trữ trong cache.
- **400 (Bad Request):** Yêu cầu không hợp lệ hoặc không thể được xử lý bởi máy chủ. Thường xảy ra khi có lỗi trong cú pháp của yêu cầu.

- **401 (Unauthorized):** Yêu cầu cần phải được xác thực. Máy chủ yêu cầu client cung cấp thông tin xác thực (như token hoặc mật khẩu) để tiếp tục.
- **403 (Forbidden):** Máy chủ hiểu yêu cầu, nhưng từ chối thực hiện. Đây thường là do thiếu quyền truy cập hoặc hạn chế quyền của người dùng.
- **404 (Not Found):** Tài nguyên yêu cầu không được tìm thấy tại URI được chỉ định. Điều này có thể do tài nguyên không tồn tại hoặc URI không đúng.
- **405 (Method Not Allowed):** Phương thức HTTP được sử dụng trong yêu cầu không được phép cho tài nguyên cụ thể. Ví dụ, nếu một tài nguyên chỉ hỗ trợ phương thức GET, thì yêu cầu POST sẽ nhận được mã trạng thái này.
- **410 (Gone):** Tài nguyên không còn tồn tại và không còn được hỗ trợ. Khác với 404, trạng thái này cho biết rằng tài nguyên đã bị xóa và sẽ không trở lại.
- **415 (Unsupported Media Type):** Loại tài nguyên được gửi trong yêu cầu không được máy chủ hỗ trợ. Điều này thường xảy ra khi định dạng của dữ liệu không tương thích với yêu cầu của máy chủ.
- **422 (Unprocessable Entity):** Dữ liệu trong yêu cầu không thể được xử lý do lỗi xác thực hoặc cấu trúc dữ liệu không chính xác. Thường được sử dụng trong các yêu cầu POST hoặc PUT khi dữ liệu không đáp ứng các quy tắc yêu cầu.
- **429 (Too Many Requests):** Client đã gửi quá nhiều yêu cầu trong một khoảng thời gian ngắn, dẫn đến việc máy chủ từ chối các yêu cầu thêm. Điều này thường xảy ra do áp dụng giới hạn tần suất yêu cầu để bảo vệ tài nguyên máy chủ..

2.5.5. Ưu điểm của *Resful API*

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:

Đơn giản hóa việc phát triển:

Restful API sử dụng các phương thức HTTP như GET, POST, PUT, DELETE để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên tài nguyên. Điều này làm giảm đáng kể sự phức tạp trong việc phát triển ứng dụng bởi vì các lập trình viên có thể tập trung vào logic kinh doanh chính mà không cần lo lắng về cách thức tương tác giữa các thành phần.

Tính mở rộng:

Restful API cho phép các ứng dụng có thể kết nối với nhau một cách linh hoạt và dễ dàng. Nhờ vào cấu trúc của nó, các ứng dụng có thể mở rộng chức năng và dịch vụ một cách hiệu quả bằng cách tương tác với các API từ các ứng dụng khác.

Tính độc lập của ứng dụng:

Việc sử dụng Restful API giúp các ứng dụng hoạt động độc lập với nhau. Điều này làm cho việc bảo trì và nâng cấp từng ứng dụng trở nên dễ dàng hơn mà không cần phải ảnh hưởng đến các thành phần khác trong hệ thống.

Khả năng tương thích:

Restful API được xây dựng trên các tiêu chuẩn mở và sử dụng các giao thức chuẩn như HTTP và JSON, làm cho nó tương thích với hầu hết các nền tảng và ngôn ngữ lập trình. Điều này giúp cho việc tích hợp các ứng dụng với nhau trở nên dễ dàng và hiệu quả.

2.5.6. Nhược điểm của Restful API**Chi phí:**

Việc phát triển và triển khai API đôi khi rất tốn kém và đòi hỏi bảo trì cũng như hỗ trợ cao từ các nhà phát triển.

Vấn đề bảo mật:

Việc sử dụng API sẽ thêm một tầng khác trong kiến trúc, khiến hệ thống dễ bị tấn công và do đó vấn đề rủi ro bảo mật thường xảy ra trong API. Ngoài ra, việc chỉ có thể chạy trên giao thức HTTP cũng khiến các API dễ dàng bị tấn công.

Hiệu suất:

RESTful API thường yêu cầu nhiều request/response vòng lặp để hoàn thành một tác vụ, điều này có thể làm giảm hiệu suất, đặc biệt là trên mạng chậm hoặc không ổn định.

Giới hạn bởi HTTP:

RESTful API thường bị giới hạn bởi các đặc điểm của HTTP, ví dụ như trạng thái kết nối và kích thước payload, điều này có thể không phù hợp cho các ứng dụng cần giao tiếp thời gian thực hoặc truyền tải dữ liệu lớn.

Quản lý phiên bản:

Khi API phát triển và thay đổi, việc quản lý phiên bản có thể trở nên phức tạp. Đảm bảo rằng các phiên bản cũ vẫn hoạt động đồng thời cung cấp các tính năng mới có thể là một thách thức.

2.6. Tìm hiểu Framework ReactJS

2.6.1. Tổng quan về ReactJS

ReactJS là một thư viện JavaScript mã nguồn mở do Facebook phát triển, được thiết kế để tạo ra các ứng dụng web hấp dẫn, nhanh chóng và hiệu quả với mã hóa tối thiểu. Mục đích cốt lõi của ReactJS không chỉ là tạo ra các trang web mượt mà, mà còn phải đảm bảo chúng nhanh chóng, dễ mở rộng và đơn giản trong việc phát triển và bảo trì.

ReactJS sử dụng một kiến trúc dựa trên các thành phần (components), cho phép các nhà phát triển xây dựng các phần giao diện người dùng độc lập và tái sử dụng được. Điều này giúp tăng cường tính tổ chức và khả năng quản lý của mã nguồn, đồng thời giảm thiểu sự phức tạp khi phát triển các ứng dụng lớn.

Một trong những tính năng nổi bật của ReactJS là Virtual DOM (Document Object Model). Thay vì thao tác trực tiếp với DOM thật, ReactJS sử dụng một phiên bản ảo của DOM, cho phép nó xác định những thay đổi cần thiết và cập nhật DOM thật một cách tối ưu nhất. Kết quả là, hiệu suất của ứng dụng được cải thiện đáng kể, đặc biệt là khi xử lý các ứng dụng có giao diện phức tạp và nhiều thay đổi theo thời gian thực.

Ngoài ra, ReactJS cũng dễ dàng tích hợp với các thư viện và frameworks khác, cho phép các nhà phát triển sử dụng nó trong nhiều loại dự án khác nhau. Với sự hỗ trợ mạnh mẽ từ cộng đồng và tài liệu phong phú, ReactJS đã trở thành một công cụ phổ biến và đáng tin cậy trong việc phát triển các ứng dụng web hiện đại. [7]

Lịch sử hình thành ReactJS: [7]

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

React Native (Phiên bản của React dành cho mobile) , cho phép phát triển Android, iOS và UWP gốc với React, đã 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 đã công bố React Fiber, một bộ thuật toán nội bộ mới để kết xuất, trái ngược với thuật toán kết xuất cũ của React, Stack. React Fiber đã trở thành nền tảng của mọi cải tiến trong tương lai và phát triển tính năng của thư viện

React. Cú pháp lập trình thực tế với React không thay đổi; chỉ có cách mà cú pháp được thực thi đã thay đổi. Hệ thống kết xuất cũ của React, Stack, được phát triển vào thời điểm mà hệ thống không tập trung vào thay đổi động. Ví dụ, Stack chậm về hoạt ảnh phức tạp khi cố gắng hoàn thành tất cả hoạt ảnh đó trong một đoạn. Sợi chia hoạt ảnh thành các phân đoạn có thể trải rộng trên nhiều khung hình. Tương tự như vậy, cấu trúc của một trang có thể được chia thành các phân đoạn có thể được duy trì và cập nhật riêng biệt. Các hàm JavaScript và đối tượng DOM ảo được gọi là "sợi" và mỗi hàm có thể được vận hành và cập nhật riêng biệt, cho phép hiển thị trên màn hình mượt mà hơn.

Vào ngày 26 tháng 9 năm 2017, React 16.0 đã được phát hành ra công chúng.

Vào ngày 10 tháng 8 năm 2020, nhóm React đã 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.

Vào ngày 29 tháng 3 năm 2022, React 18 đã 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.

2.6.2. Ưu điểm của ReactJS

Virtual DOM:

React sử dụng Virtual DOM để tối ưu hóa hiệu suất. Virtual DOM là một bản sao của DOM thực tế được React duy trì và cập nhật một cách hiệu quả. Khi dữ liệu thay đổi, React so sánh Virtual DOM cũ và mới để chỉ cập nhật những phần tử thực sự thay đổi trên DOM, giúp giảm thiểu tài nguyên và tăng tốc độ render.

JSX:

JSX là một phần mở rộng của JavaScript cho phép viết HTML trong JavaScript. Điều này giúp cho việc phát triển và bảo trì mã nguồn trở nên dễ dàng hơn bằng cách kết hợp logic UI và logic ứng dụng trong cùng một tệp tin, cũng như cho phép lập trình viên sử dụng các tính năng của JavaScript như biến, vòng lặp, điều kiện trong phần giao diện.

Cộng đồng lớn và hỗ trợ tốt:

React có một cộng đồng lớn và sự hỗ trợ mạnh mẽ từ Facebook và cộng đồng người dùng. 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.

Dễ dàng sửa lỗi:

Dễ dàng cho việc tìm và sửa lỗi, vì hầu hết các mã nguồn đều được thực hiện bằng JavaScript chứ không phải bằng HTML.

Mã nguồn mở:

ReactJS là một opensource, vì vậy cũng rất dễ cho tiếp cận với những người mới bắt đầu tìm hiểu.

2.6.3. Nhược điểm của ReactJS**Thời gian học tập và làm quen:**

ReactJS yêu cầu các nhà phát triển phải học nhiều khái niệm mới và các công nghệ liên quan như JSX, Redux, và Hooks. Điều này có thể làm tăng thời gian làm quen, đặc biệt đối với những người mới.

Phức tạp trong việc cấu hình:

ReactJS tự nó không phải là một framework hoàn chỉnh, vì vậy việc cấu hình và tích hợp các công cụ khác (như Redux, Router, và các công cụ xây dựng) có thể trở nên phức tạp và tốn thời gian.

Hiệu suất:

Đối với các ứng dụng lớn và phức tạp, ReactJS có thể gặp vấn đề về hiệu suất nếu không được tối ưu hóa đúng cách. Sử dụng quá nhiều state hoặc quản lý state không hiệu quả có thể làm giảm hiệu suất.

Trải nghiệm học tập và làm quen:

Vì hầu hết code được viết dưới dạng JSX, tức là HTML và CSS là một phần của JavaScript, nó không giống như những framework khác vẫn tách biệt giữa HTML và CSS nên những người mới làm quen với ReactJS sẽ hơi lúng túng và dễ nhầm lẫn giữa JSX và HTML.

CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU

3.1. Mô tả bài toán

Nghiệp vụ quản lý thông tin các đặc sản thuộc 13 tỉnh Tây Nam Bộ. 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 của các đặc sản gồm: mã, tên đặc sản, mã tỉnh, nhà sản xuất, loại, mô tả, hình ảnh, thành phần, địa chỉ nơi bán, cách sử dụng, chứng nhận an toàn thực phẩm, hạn sử dụng,... 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.

Yêu cầu chức năng

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

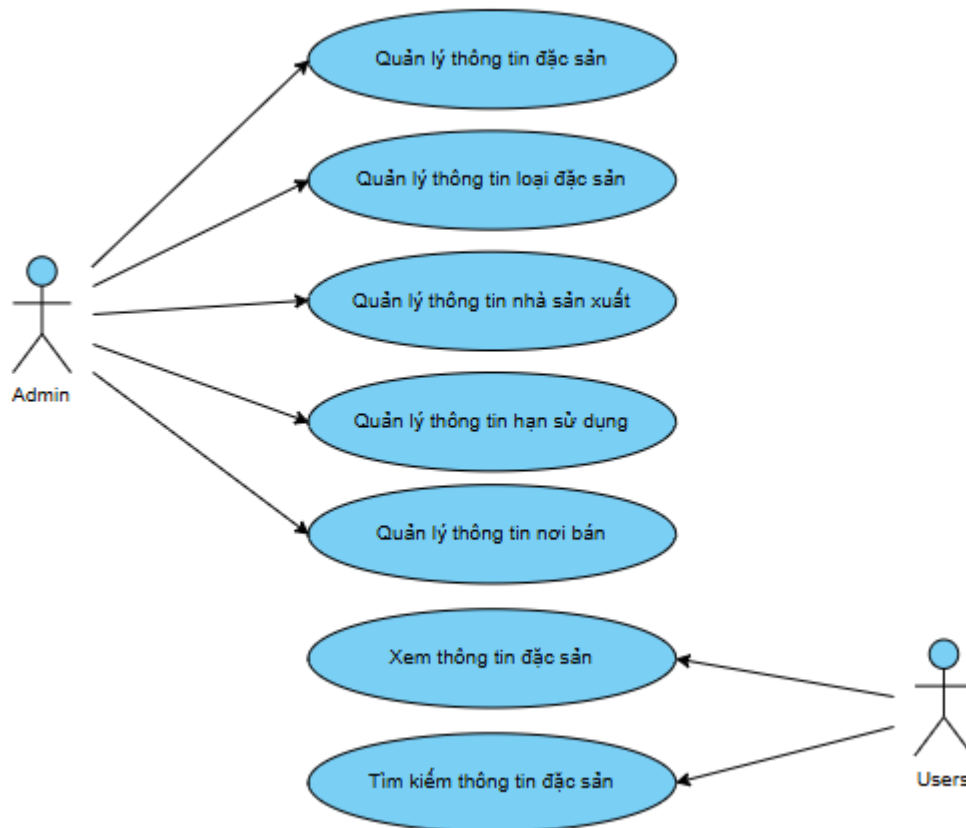
- Người dùng có thể tìm kiếm các đặc sản theo từ khóa.
- Hỗ trợ tìm kiếm theo tên tỉnh thành.
- Thông tin chi tiết: Cung cấp thông tin chi tiết về mỗi đặc sản.

Yêu cầu chức năng

- Khả năng mở rộng:
 - Thiết kế hệ thống với kiến trúc linh hoạt, cho phép dễ dàng thêm mới các đặc sản và tính năng mà không ảnh hưởng đến hiệu suất hiện tại.
 - Sử dụng các công nghệ và frameworks có khả năng mở rộng tốt, như ReactJS cho frontend và Node.js cho backend.
- Cơ sở dữ liệu tối ưu:
 - Lựa chọn cơ sở dữ liệu phù hợp để lưu trữ thông tin đặc sản. Ví dụ, MongoDB có thể là một lựa chọn tốt do khả năng lưu trữ dữ liệu không cấu trúc và dễ dàng mở rộng.
 - Tối ưu hóa cơ sở dữ liệu để đảm bảo hiệu suất cao trong việc truy vấn và cập nhật dữ liệu.

- Hiệu suất:
 - Đảm bảo trang web tải nhanh và phản hồi nhanh chóng, ngay cả khi số lượng người dùng truy cập lớn.
 - Sử dụng các phương pháp tối ưu hóa như lazy loading hình ảnh và giảm thiểu kích thước tài nguyên tải về.
- Bảo mật:
 - Thường xuyên cập nhật và kiểm tra các lỗ hổng bảo mật.
- Trải nghiệm người dùng:
 - Thiết kế giao diện người dùng thân thiện, dễ sử dụng, và trực quan.
 - Đảm bảo trang web tương thích tốt với nhiều loại thiết bị, từ máy tính để bàn đến điện thoại di động.

3.2. Đặc tả hệ thống



Hình 3.1. Sơ đồ Use-case

Bảng 3.1 Mô tả Actor

STT	Tên Actor	Ý nghĩa
1	Users	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 đặc sản	Quản trị viên quản lý thông tin các loại đặc sản trên trang web và thêm, sửa, xóa nếu cần thiết
2	Quản lý thông tin loại đặc sản	Quản trị viên có thể quản lý thông tin loại cho đặc sản
3	Quản lý thông tin nhà sản xuất	Quản trị viên có thể quản lý thông tin của nhà sản xuất cho từng loại đặc sản khác nhau
4	Quản lý thông tin hạn sử dụng	Quản trị viên có thể quản lý thông tin về hạn sử dụng, mỗi đặc sản khác nhau sẽ có hạn sử dụng khác nhau
5	Quản lý thông tin nơi bán	Quản trị viên có thể quản lý thông tin về nơi bán, mỗi đặc sản có thể có một hoặc nhiều nơi bán khác nhau
6	Xem thông tin đặc sản	Người dùng có thể xem thông tin của các đặc sản có trên hệ thống
7	Tìm kiếm thông tin đặc sản	Người dùng có thể tìm kiếm các đặc sản có trên hệ thống, có thể tìm kiếm theo tên, tìm kiếm theo tỉnh thành, tìm kiếm theo loại

3.3. Lược đồ dữ liệu của đặc sản

```
"_id": ObjectId(),
"name",
"province",
"manufacturer":
    {
        "name",
        "address",
        "contact",
    }
"flavor",
"description",
"category":
    {
        "name",
        "specifications",
        "storage_method",
        "components",
    }
"img",
"market":
    {
        "name",
        "address",
        "contact",
    }
"instructions",
"food_safety_standard",
"expiry_date":
    {
        "name",
        "production_date",
        "expiration_date",
    }
```


Bảng 3.3 Mô tả Colletion “specialties”

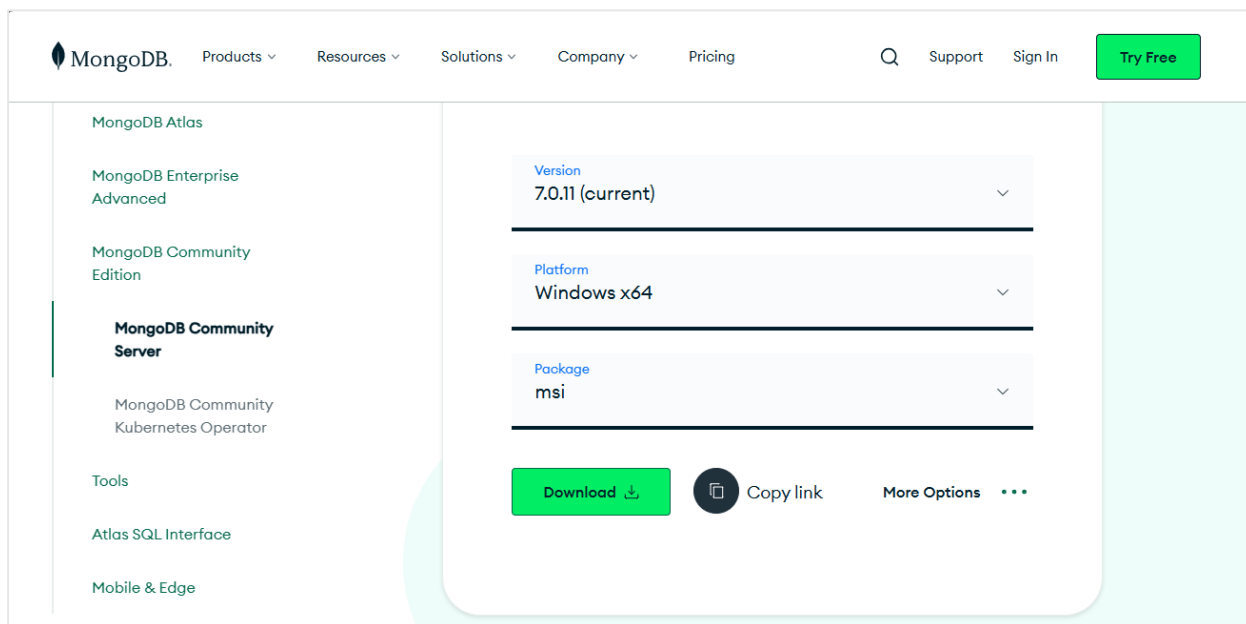
STT	Tên thuộc tính	Kiểu dữ liệu				Ý nghĩa
1	_id	ObjectId (MongoDB)				Mã của đặc sản
2	name	String				Tên của đặc sản
3	province	String				Tên của tỉnh thành
4	manufacturer	STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa	Nhà sản xuất
		1	_id	ObjectId (MongoDB)	Mã của nhà sản xuất	
		2	name	String	Tên nhà sản xuất	
		3	address	String	Địa chỉ nhà sản xuất	
		4	contact	String	Thông tin liên lạc	
5	flavor	String				Hương vị của đặc sản
6	description	String				Mô tả về đặc sản
7	category	STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa	Loại của đặc sản
		1	_id	ObjectId (MongoDB)	Mã của loại đặc sản	
		2	name	String	Tên loại	
		3	specifications	String	Quy cách	
		4	storage_method	String	Cách bảo quản	
		5	components	String	Thành phần chính	

STT	Tên thuộc tính	Kiểu dữ liệu				Ý nghĩa
8	img	String				Hình ảnh của đặc sản
9	market	STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa	Nơi bán của đặc sản
		1	_id	ObjectId (MongoDB)	Mã của nơi bán	
		2	name	String	Tên nơi bán	
		3	address	String	Địa chỉ bán	
		4	contact	String	Số thông tin liên lạc	
10	instructions	String				Hướng dẫn sử dụng đặc sản
11	food_safety_standard	String				Chứng nhận an toàn thực phẩm
12	expiry_date	STT	Tên thuộc tính	Kiểu dữ liệu	Ý nghĩa	Hạn sử dụng của đặc sản
		1	_id	ObjectId (MongoDB)	Mã của hạn sử dụng	
		2	name	String	Lô sản xuất	
		3	production_date	String	Ngày sản xuất	
		4	expiration_date	String	Hạn sử dụng của đặc sản	

3.4. Cài đặt MongoDB và nhập dữ liệu

3.4.1. Cài đặt MongoDB

Bước 1: Truy cập <https://www.mongodb.com/try/download/community> và chọn phiên bản phù hợp với hệ điều hành.



Hình 3.2. Giao diện cài đặt MongoDB Community Server

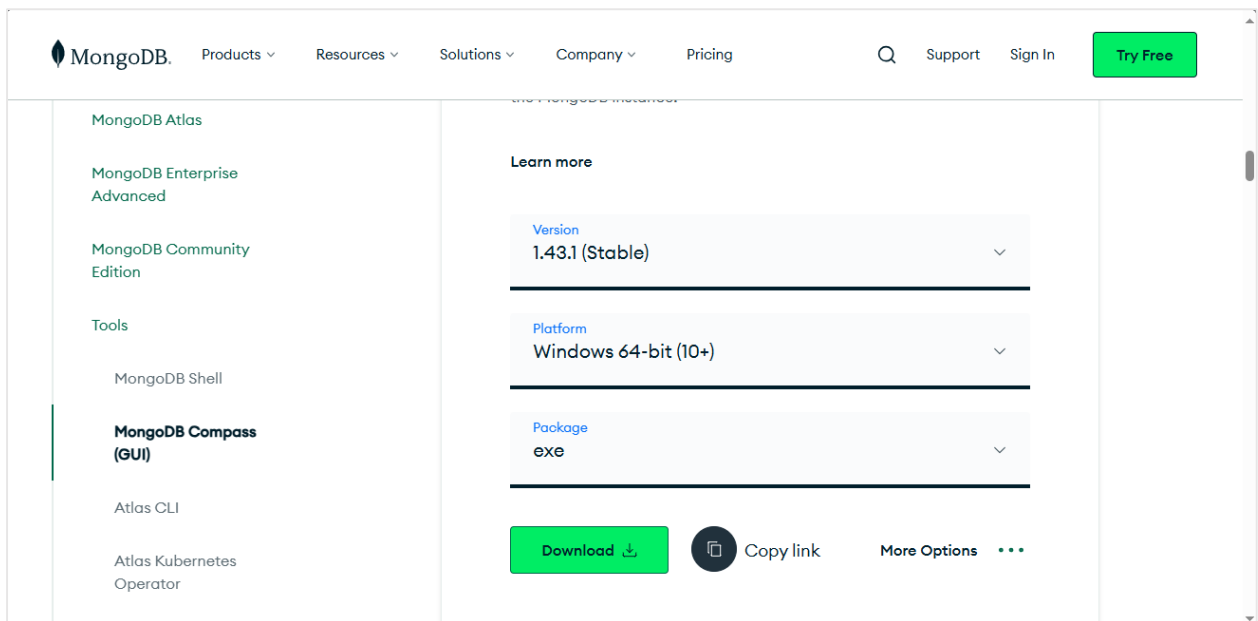
Bước 2: Tải về tệp msi và tiến hành cài đặt.

Bước 3: Mở “Control Panel” -> “System and Security” -> “System” -> “Advanced system settings”. Sau đó chọn “Environment Variables”, tìm biến “Path” trong mục “System variables”, chọn và nhấp vào “Edit”. Thêm đường dẫn đến thư mục bin của MongoDB.

Sau khi cài đặt hoàn tất, MongoDB sẽ tự động chạy như một dịch vụ. Có thể kiểm tra và quản lý dịch vụ này qua “Services” trong Windows. Kiểm tra phiên bản đã cài đặt bằng cách Mở Command Prompt và nhập “mongo –version”.

3.4.2. Cài đặt MongoDB Compass

Bước 1: Truy cập <https://www.mongodb.com/try/download/compass> và chọn phiên bản phù hợp với hệ điều hành.



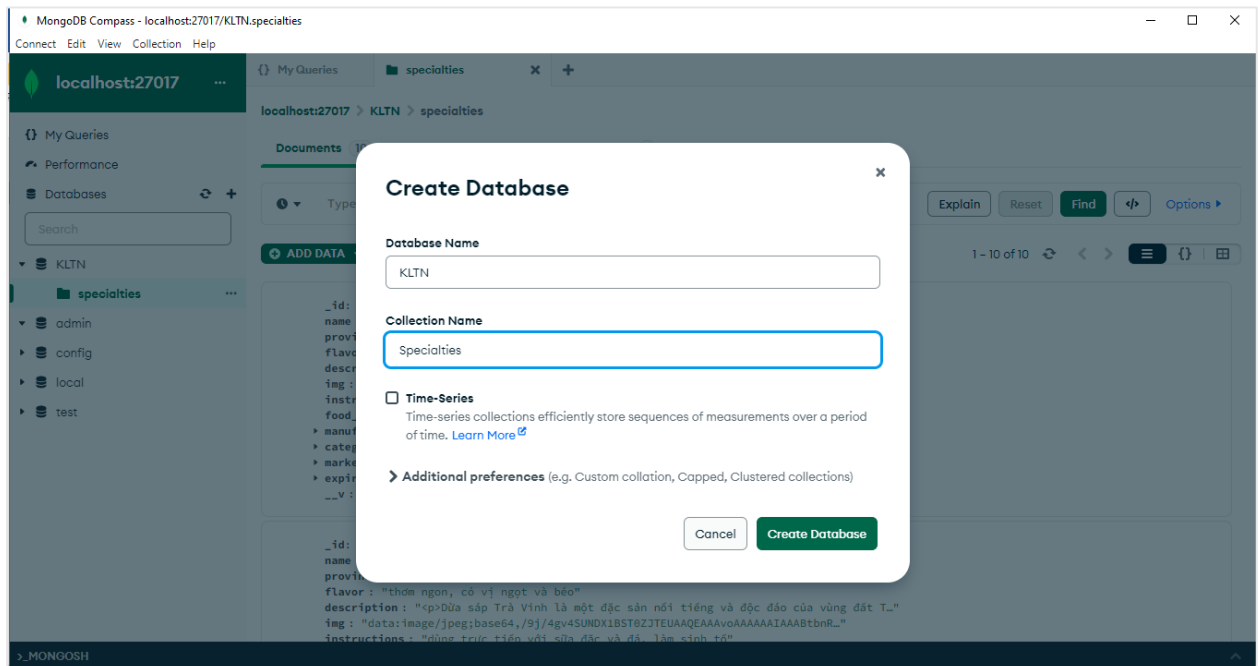
Hình 3.3. Giao diện cài đặt MongoDB Compass

Bước 2: Tải về và tiến hành cài đặt.

Bước 3: Mở MongoDB Compass, nhập chuỗi kết nối `mongodb://localhost:27017`, để kết nối đến MongoDB server đang chạy trên máy. Cuối cùng nhấn "Connect" để kết nối và bắt đầu sử dụng MongoDB Compass để quản lý cơ sở dữ liệu.

3.4.3. Nhập dữ liệu vào MongoDB Compass

Bước 1: Chọn Create Database.



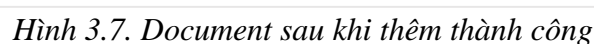
Hình 3.4. Giao diện tạo cơ sở dữ liệu

Ngoài ra, có thể tạo cơ sở dữ liệu bằng cách sử dụng lệnh **use** với cú pháp: “use <DATABASE_NAME>” và tạo collection với phương thức: “db.createCollection(name, options)”.

Hình 3.5. Tạo cơ sở dữ liệu

```
> MONGODB
}
DATN> db.specialties.insertOne({
  _id: ObjectId(),
  name: 'Bánh Pía nhân sầu riêng Tân Huê Viên',
  province: 'Sóc Trăng',
  Manufacturer: [
    {
      _id: ObjectId(),
      name: 'CÔNG TY TNHH CHẾ BIẾN THỰC PHẨM BÁNH PÍA - LẠP XƯỜNG TÂN HUÊ VIÊN',
      address: '153 Quốc lộ 1A, ấp Phụng Hiệp, xã An Hiệp, huyện Châu Thành, tỉnh Sóc Trăng',
      contract: '(0299) 3623700'
    }
  ],
  category: [
    {
      _id: ObjectId(),
      name: 'Bánh',
      specifications: 'Bịch 480g x 12 cái',
      storage_method: 'Giữ khô ráo, Bảo quản ở nhiệt độ phù hợp'
    }
  ],
  flavor: 'Hương vị thơm, ngon',
  instructions: 'Dùng ngay sau khi mở bao bì'
})
```

Hình 3.6. Thêm Document và Collection



3.5. Xây dựng Back-end

3.5.1. Định nghĩa model

```
const mongoose = require('mongoose');
//Model market
const marketSchema = new mongoose.Schema({
  name: String,
  address: String,
  contact: String,
  price: String
});
//Model manufacturer
const manufacturerSchema = new mongoose.Schema({
  name: String,
  address: String,
  contact: String,
});
// Model expiry_date
const expiry_dateSchema = new mongoose.Schema({
  name: String,
  production_date: String,
  expiration_date: String,
});
//Model category
const categorySchema = new mongoose.Schema({
  name: String,
  specifications: String,
  storage_method: String,
  components: String,
});
//Model specialty
const specialtySchema = new mongoose.Schema({
  name: {
    type: String,
```

```

        required: true
    },
    province: String,
    manufacturer: [manufacturerSchema],
    category: [categorySchema],
    flavor: String,
    description: String,
    img: String,
    market: [marketSchema],
    instructions: String,
    food_safety_standard: String,
    expiry_date: [expiry_dateSchema]
  });

const Specialty = mongoose.model('Specialty', specialtySchema);

module.exports = Specialty;

```

3.5.2. Định nghĩa các Controller

SpecialtyController (định nghĩa các hàm thêm, sửa, xóa các đặc sản)

- Controller lấy ra tất cả đặc sản:

```

exports.getAllSpecialties = async (req, res) => {
  try {
    const specialties = await Specialty.find();
    res.json(specialties);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};

```

- Controller lấy ra một đặc sản theo ID đặc sản:

```
exports.getSpecialtyById = async (req, res) => {
  try {
    const specialty = await Specialty.findById(req.params.id);
    if (!specialty) {
      return res.status(404).json({ message: "Specialty not found" });
    }
    res.json(specialty);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

- Controller thêm một đặc sản mới:

```
exports.createSpecialty = async (req, res) => {
  try {
    const specialty = new Specialty(req.body);
    await specialty.save();
    res.status(201).json(specialty);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

- Controller cập nhật đặc sản theo ID:

```
exports.updateSpecialtyById = async (req, res) => {
  try {
    await Specialty.findByIdAndUpdate(req.params.id, req.body);
    res.json({ message: "Specialty updated successfully" });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```


- Controller xoá một đặc sản theo ID:

```
exports.deleteSpecialtyById = async (req, res) => {
  try {
    await Specialty.findByIdAndDelete(req.params.id);
    res.json({ message: "Specialty deleted successfully" });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

- Controller xoá nhiều đặc sản:

```
exports.deleteMultipleSpecialties = async (req, res) => {
  try {
    const { ids } = req.body;
    await Specialty.deleteMany({ _id: { $in: ids } });
    res.status(200).json({ message: 'Specialties deleted successfully' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
};
```

- Controller lấy ra tên của đặc sản:

```
exports.getSpecialtyNames = async (req, res) => {
  try {
    const specialties = await Specialty.find({}, 'name');
    res.json(specialties);
  } catch (error) {
    console.error('Error fetching specialty names:', error);
    res.status(500).json({ error: 'Error fetching specialty names' });
  }
};
```

CategoryController (định nghĩa các hàm thêm, sửa, xóa loại cho đặc sản)

- Controller tạo ra một loại mới:

```
exports.createCategory = async (req, res) => {  
  try {  
    const specialtyId = req.params.id;  
    const category = req.body;  
    const specialty = await Specialty.findById(specialtyId);  
    if (!specialty) {  
      return res.status(404).json({ message: 'Specialty not found'  
});  
    }  
    specialty.category.push(category);  
    await specialty.save();  
    res.status(201).json({ message: 'Category created  
successfully', category: category });  
  } catch (error) {  
    res.status(500).json({ error: error.message });  
  }  
};
```

- Controller lấy ra tất cả loại của đặc sản:

```
exports.getAllCategories = async (req, res) => {  
  try {  
    const specialtyId = req.params.id;  
    const specialty = await Specialty.findById(specialtyId);  
    if (!specialty) {  
      return res.status(404).json({ message: 'Specialty not  
found' });  
    }  
    res.json(specialty.category);  
  } catch (error) {  
    res.status(500).json({ error: error.message });  
  }  
};
```

- Controller lấy ra một loại trong một đặc sản theo ID

```
exports.getCategoryById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const categoryId = req.params.categoryId;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    const category = specialty.category.id(categoryId);
    if (!category) {
      return res.status(404).json({ message: 'Category not found' });
    }
    res.json(category);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

- Controller cập nhật thông tin của một loại trong một đặc sản theo ID

```
exports.updateCategoryById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const categoryId = req.params.categoryId;
    const updatedCategory = req.body;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    const category = specialty.category.id(categoryId);
    if (!category) {
      return res.status(404).json({ message: 'Category not found' });
    }
  }
};
```

```

    }

    category.set(updatedCategory);
    await specialty.save();

    res.json({ message: 'Category updated successfully', category:
category });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};

```

- Controller xoá một loại trong một đặc sản theo ID

```

exports.deleteCategoryById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const categoryId = req.params.categoryId;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not
found' });
    }

    specialty.category.pull(categoryId); // Sử dụng pull để xoá
category từ mảng

    await specialty.save();
    res.json({ message: 'Category deleted successfully' });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};

```

ManufacturerController (định nghĩa các hàm thêm, sửa, xóa nhà sản xuất cho đặc sản)

- Controller tạo một nhà sản xuất mới

```
exports.createManufacturer = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const manufacturer = req.body;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    specialty.manufacturer.push(manufacturer);
    await specialty.save();
    res.status(201).json({ message: 'Manufacturer created successfully', manufacturer: manufacturer });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

- Controller lấy tất cả các nhà sản xuất của một đặc sản

```
exports.getAllManufacturers = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    res.json(specialty.manufacturer);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

- Controller lấy thông tin một nhà sản xuất trong một đặc sản theo ID

```
exports.getManufacturerById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const manufacturerId = req.params.manufacturerId;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    const manufacturer =
      specialty.manufacturer.id(manufacturerId);
    if (!manufacturer) {
      return res.status(404).json({ message: 'Manufacturer not found' });
    }
    res.json(manufacturer);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

- Controller cập nhật thông tin một nhà sản xuất trong một đặc sản theo ID

```
exports.updateManufacturerById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const manufacturerId = req.params.manufacturerId;
    const updatedManufacturer = req.body;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
  }
```

```

        const manufacturer =
specialty.manufacturer.id(manufacturerId);

        if (!manufacturer) {

            return res.status(404).json({ message: 'Manufacturer not
found' });

        }

        manufacturer.set(updatedManufacturer);

        await specialty.save();

        res.json({ message: 'Manufacturer updated successfully',
manufacturer: manufacturer });

    } catch (error) {

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

    }

};

```

- Controller xoá một nhà sản xuất trong một đặc sản theo ID

```

exports.deleteManufacturerById = async (req, res) => {

    try {

        const specialtyId = req.params.id;

        const manufacturerId = req.params.manufacturerId;

        const specialty = await Specialty.findById(specialtyId);

        if (!specialty) {

            return res.status(404).json({ message: 'Specialty not
found' });

        }

        specialty.manufacturer.pull(manufacturerId);

        await specialty.save();

        res.json({ message: 'Manufacturer deleted successfully' });

    } catch (error) {

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

    }

};

```

MarketController (định nghĩa các hàm thêm, sửa, xóa nơi bán cho đặc sản)

- Controller thêm một nơi bán mới

```
exports.createMarket = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const market = req.body;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    specialty.market.push(market);
    await specialty.save();
    res.status(201).json({ message: 'Market created successfully', market: market });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

- Controller lấy tất cả các nơi bán của một đặc sản

```
exports.getAllMarkets = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    res.json(specialty.market);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```


- Controller lấy thông tin một market trong một specialty theo ID

```
exports.getMarketById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const marketId = req.params.marketId;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    const market = specialty.market.id(marketId);
    if (!market) {
      return res.status(404).json({ message: 'Market not found' });
    }
    res.json(market);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

- Controller cập nhật thông tin một market trong một specialty theo ID

```
exports.updateMarketById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const marketId = req.params.marketId;
    const updatedMarket = req.body;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    const market = specialty.market.id(marketId);
    if (!market) {
```

```

        return res.status(404).json({ message: 'Market not found'
    });

    }

    market.set(updatedMarket);

    await specialty.save();

    res.json({ message: 'Market updated successfully', market:
market });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
};

```

- Controller xoá một market trong một specialty theo ID

```

exports.deleteMarketById = async (req, res) => {
    try {
        const specialtyId = req.params.id;
        const marketId = req.params.marketId;
        const specialty = await Specialty.findById(specialtyId);
        if (!specialty) {
            return res.status(404).json({ message: 'Specialty not
found' });
        }

        specialty.market.pull(marketId);

        await specialty.save();

        res.json({ message: 'Market deleted successfully' });
    } catch (error) {
        res.status(500).json({ error: error.message });
    }
};

```

ExpiryDateController (định nghĩa các hàm thêm, sửa, xóa hạn sử dụng cho đặc sản)

- **Controller tạo một hạn sử dụng mới**

```
exports.createExpiryDate = async (req, res) => {  
  try {  
    const specialtyId = req.params.id;  
    const expiryDate = req.body;  
    const specialty = await Specialty.findById(specialtyId);  
    if (!specialty) {  
      return res.status(404).json({ message: 'Specialty not  
found' });  
    }  
    specialty.expiry_date.push(expiryDate);  
    await specialty.save();  
    res.status(201).json({ message: 'Expiry Date created  
successfully', expiryDate: expiryDate });  
  } catch (error) {  
    res.status(500).json({ error: error.message });  
  }  
};
```

- **Controller lấy tất cả hạn sử dụng của một đặc sản**

```
exports.getAllExpiryDates = async (req, res) => {  
  try {  
    const specialtyId = req.params.id;  
    const specialty = await Specialty.findById(specialtyId);  
    if (!specialty) {  
      return res.status(404).json({ message: 'Specialty not  
found' });  
    }  
    res.json(specialty.expiry_date);  
  } catch (error) {  
    res.status(500).json({ error: error.message });  
  }  
};
```

- Controller lấy thông tin một hạn sử dụng trong một đặc sản theo ID

```
exports.getExpiryDateById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const expiryDateId = req.params.expiryDateId;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    const expiryDate = specialty.expiry_date.id(expiryDateId);
    if (!expiryDate) {
      return res.status(404).json({ message: 'Expiry Date not found' });
    }
    res.json(expiryDate);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
};
```

- Controller cập nhật thông tin một hạn sử dụng trong một đặc sản theo ID

```
exports.updateExpiryDateById = async (req, res) => {
  try {
    const specialtyId = req.params.id;
    const expiryDateId = req.params.expiryDateId;
    const updatedExpiryDate = req.body;
    const specialty = await Specialty.findById(specialtyId);
    if (!specialty) {
      return res.status(404).json({ message: 'Specialty not found' });
    }
    const expiryDate = specialty.expiry_date.id(expiryDateId);
    if (!expiryDate) {
      return res.status(404).json({ message: 'Expiry Date not found' });
    }
  }
};
```

```

    }

    expiryDate.set(updatedExpiryDate);

    await specialty.save();

    res.json({ message: 'Expiry Date updated successfully',
expiryDate: expiryDate });

    } catch (error) {

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

    }

};

```

- Controller xoá một hạn sử dụng trong một đặc sản theo ID

```

exports.deleteExpiryDateById = async (req, res) => {

    try {

        const specialtyId = req.params.id;

        const expiryDateId = req.params.expiryDateId;

        const specialty = await Specialty.findById(specialtyId);

        if (!specialty) {

            return res.status(404).json({ message: 'Specialty not
found' });

        }

        specialty.expiry_date.pull(expiryDateId);

        await specialty.save();

        res.json({ message: 'Expiry Date deleted successfully' });

    } catch (error) {

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

    }

};

```

3.5.3. Định nghĩa các Router

- SpecialtyRoutes (chứa các API xử lý việc thêm, xoá, sửa các đặc sản):

```

const express = require('express');

const router = express.Router();

const specialtyController = require('../controllers/specialtyController');

router.post('/', specialtyController.createSpecialty);

```

```

router.get('/', specialtyController.getAllSpecialties);
router.get('/names', specialtyController.getSpecialtyNames);
router.get('/:id', specialtyController.getSpecialtyById);
router.get('/', specialtyController.getSpecialtiesByProvince);
router.put('/:id', specialtyController.updateSpecialtyById);
router.delete('/:id', specialtyController.deleteSpecialtyById);
router.post('/delete-multiple', specialtyController.deleteMultipleSpecialties);
router.delete('/:specialtyId/category/:categoryId', specialtyController.deleteCategoryFromSpecialty);
router.put('/:specialtyId/category/:categoryId', specialtyController.updateCategory);
module.exports = router;

```

- **CategoryRoutes** (Chứa các API xử lý việc thêm, xóa, sửa các loại của đặc sản):

```

const express = require('express');
const router = express.Router();
const categoryController = require('../controllers/categoryController');
router.post('/:id/category', categoryController.createCategory);
router.get('/:id/category', categoryController.getAllCategories);
router.get('/:id/category/:categoryId', categoryController.getCategoryById);
router.put('/:id/category/:categoryId', categoryController.updateCategoryById);
router.delete('/:id/category/:categoryId', categoryController.deleteCategoryById);
router.delete('/:specialtyId/categories', categoryController.deleteMultipleCategories);
module.exports = router;

```

- **ManufacturerRoutes** (Chứa các API xử lý việc thêm, xóa, sửa nhà sản xuất):

```

const express = require('express');
const router = express.Router();
const manufacturerController = require('../controllers/manufacturerController');

router.post('/:id/manufacturer', manufacturerController.createManufacturer);

```

```

router.get('/:id/manufacturers',manufacturerController.getAllManufacturers);

router.get('/:id/manufacturers/:manufacturerId',manufacturerController.getManufacturerById);

router.put('/:id/manufacturers/:manufacturerId',manufacturerController.updateManufacturerById);

router.delete('/:id/manufacturers/:manufacturerId',manufacturerController.deleteManufacturerById);

module.exports = router;

```

- **MarketRoutes** (Chứa các API xử lý việc thêm, xóa, sửa các địa điểm bán)

```

const express = require('express');
const router = express.Router();
const marketController = require('../controllers/marketController');
router.post('/:id/market', marketController.createMarket);
router.get('/:id/market', marketController.getAllMarkets);
router.get('/:id/market/:marketId', marketController.getMarketById);
router.put('/:id/market/:marketId',marketController.updateMarketById);
router.delete('/:id/market/:marketId',marketController.deleteMarketById);

module.exports = router;

```

- **ExpiryDateRoutes** (Chứa các API xử lý việc thêm, xóa, sửa thời hạn sử dụng):

```

const express = require('express');
const router = express.Router();
const expiryDateController = require('../controllers/expiryDateController');
router.post('/:id/expiry_date',expiryDateController.createExpiryDate);
router.get('/:id/expiry_date',expiryDateController.getAllExpiryDates);
router.get('/:id/expiry_date/:expiryDateId',expiryDateController.getExpiryDateById);
router.put('/:id/expiry_date/:expiryDateId',expiryDateController.updateExpiryDateById);
router.delete('/:id/expiry_date/:expiryDateId',expiryDateController.deleteExpiryDateById);

module.exports = router;

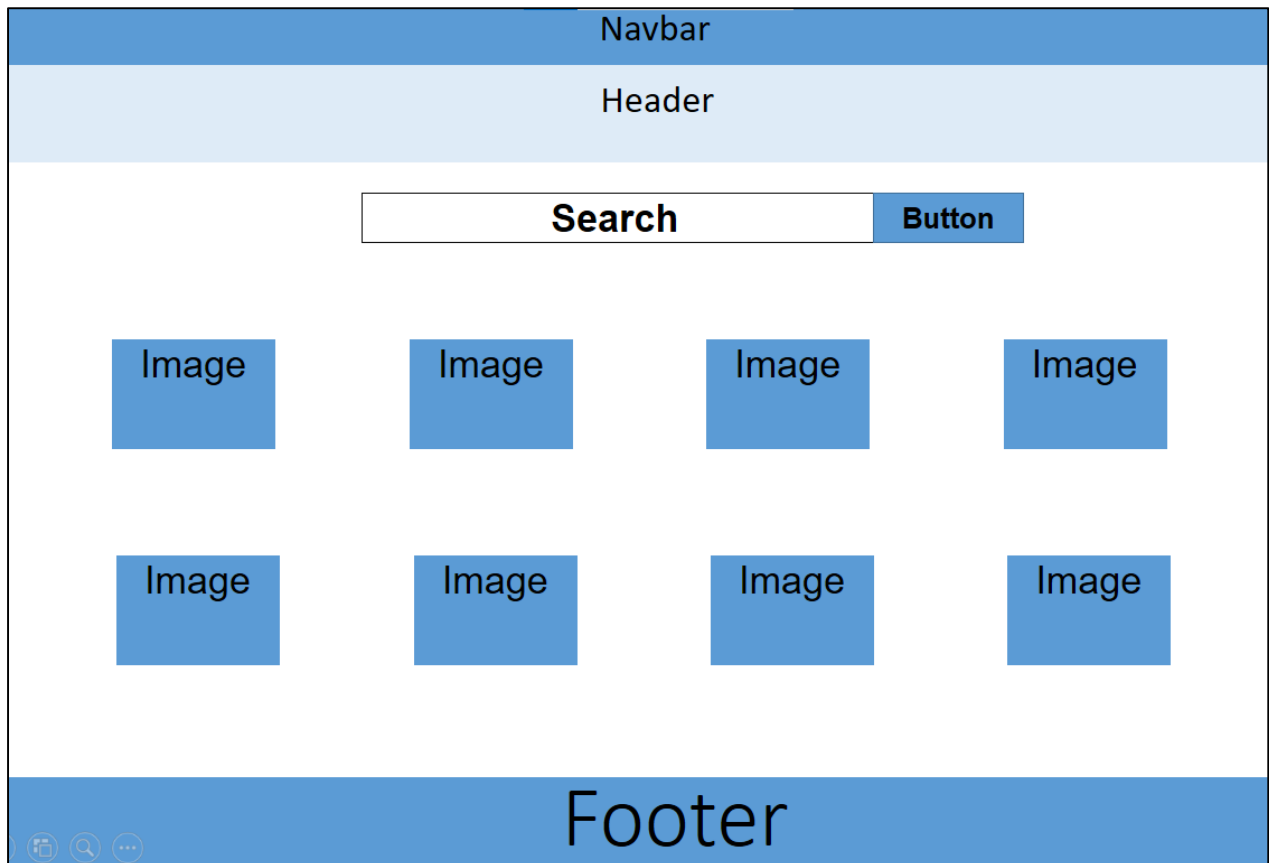
```

3.6. Xây dựng giao diện

Phát thảo giao diện để tiến hành xây dựng gồm có:

- Giao diện người dùng.
- Giao diện trang quản trị.

3.6.1. Phác thảo giao diện người dùng.



Hình 3.8. Phác thảo giao diện người dùng

Trong đó:

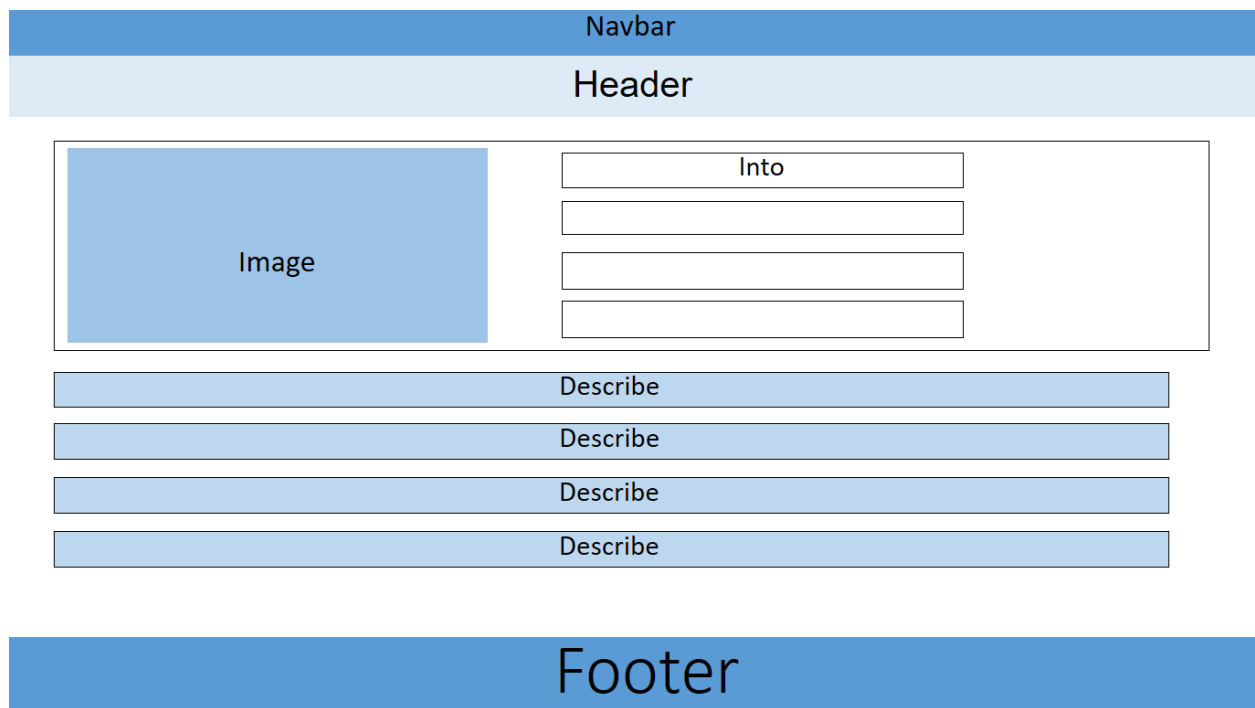
Navbar: chứa các lựa chọn đi đến các trang khác nhau.

Header: phần đầu của trang web.

Search: thanh tìm kiếm để người dùng dễ dàng tìm được.

Image: các đặc sản có trên trang web.

Footer: phần chân của trang web.



Hình 3.9. Phác thảo giao diện trang thông tin đặc sản

Trong đó:

Navbar: chứa các lựa chọn đi đến các trang khác nhau.

Header: phần đầu của trang web.

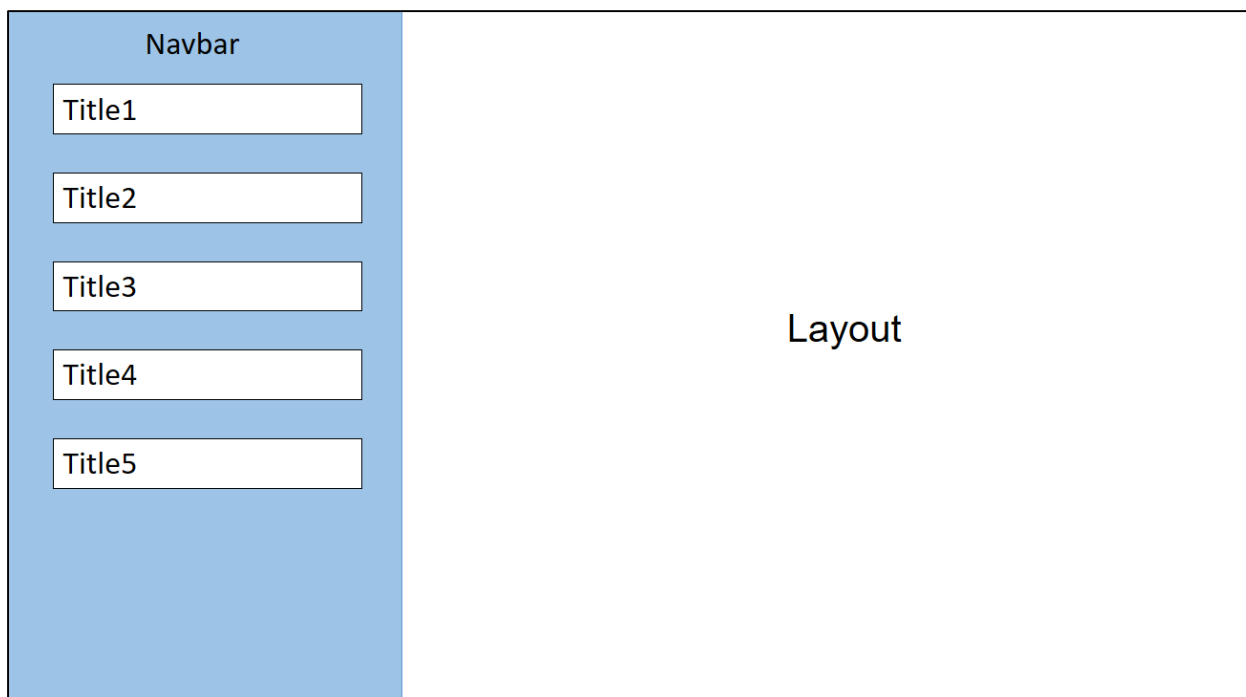
Image: hình ảnh của đặc sản.

Info: chứa các thông tin của đặc sản.

Describe: chứa các mô tả của đặc sản.

Footer: phần chân của trang web.

3.6.2. Phác thảo giao diện trang quản trị.



Hình 3.10. Phác thảo giao diện trang quản trị

Trong đó:

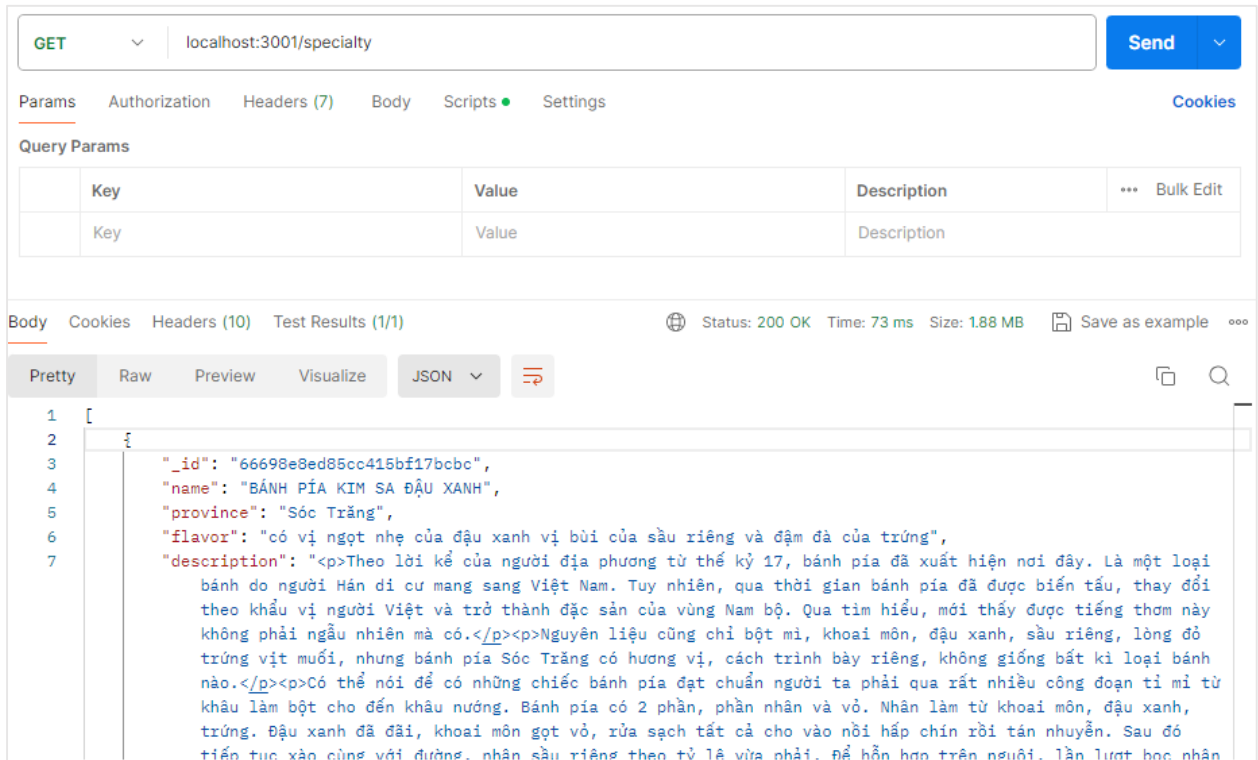
Navbar: chứa các lựa chọn để admin tương tác.

Title: các lựa chọn để người quản trị thực hiện việc thêm, sửa, xóa.

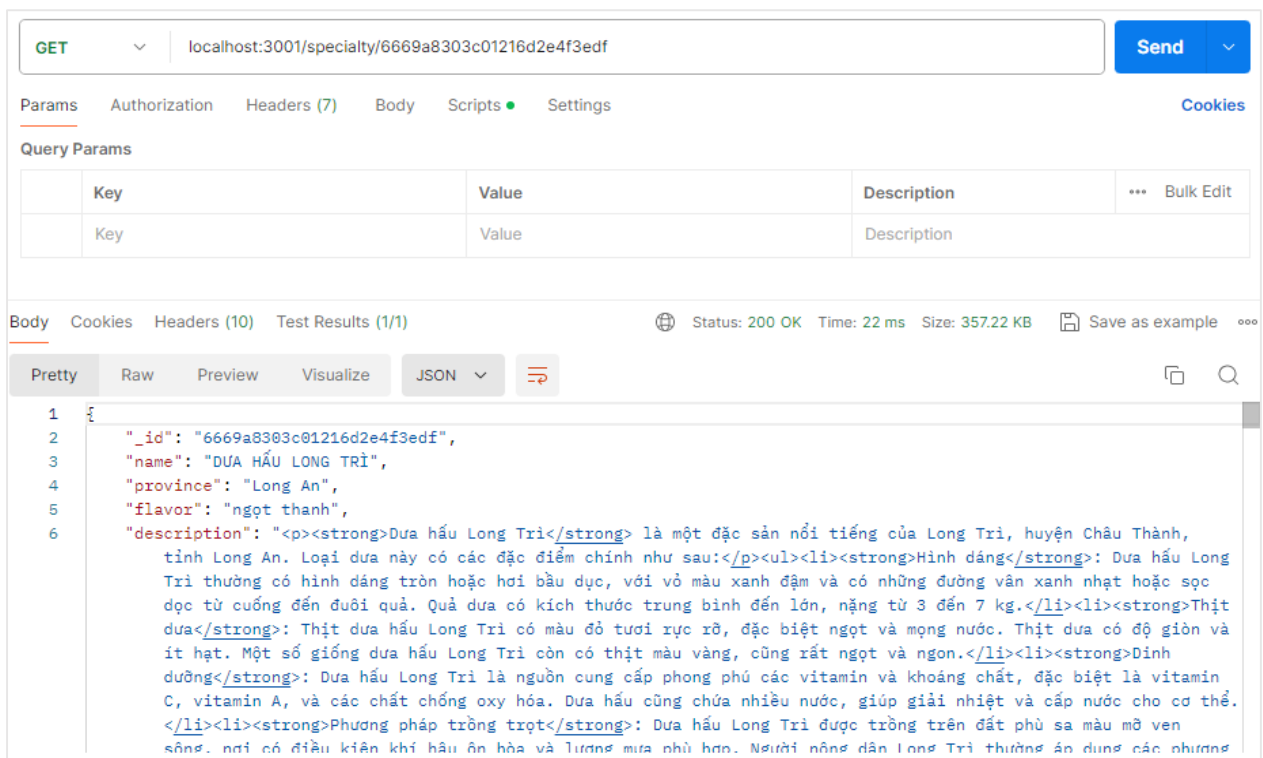
Layout: hiển thị thông tin tương ứng với title đã chọn để tiến hành thêm, sửa, xóa.

CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU

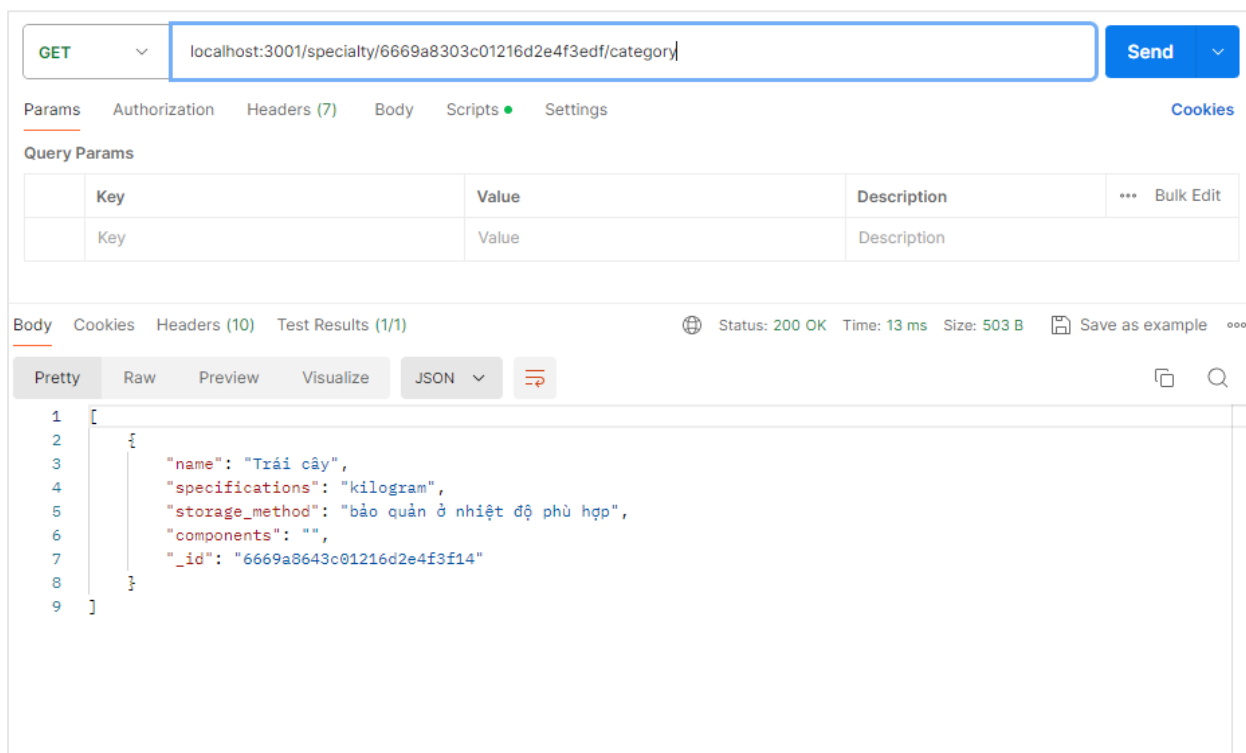
4.1. Thử nghiệm các API để lấy dữ liệu với Postman



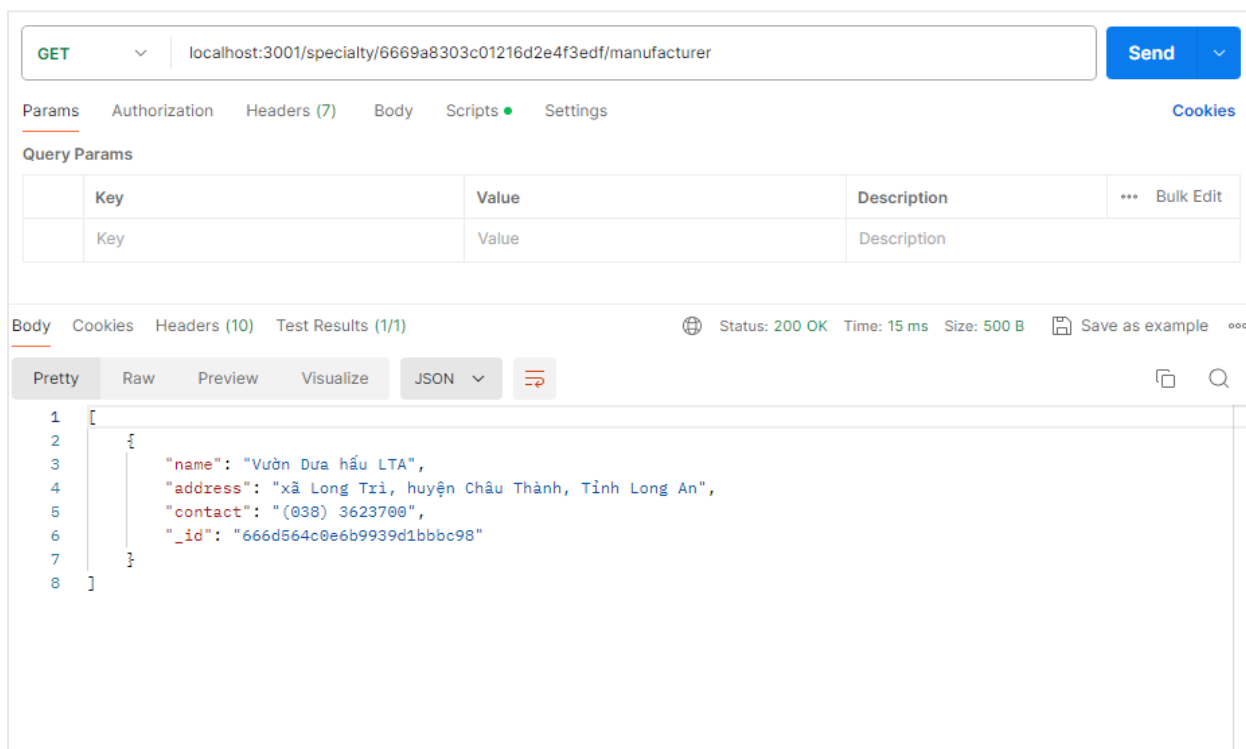
Hình 4.1. Sử dụng API lấy danh sách tất cả đặc sản với Postman



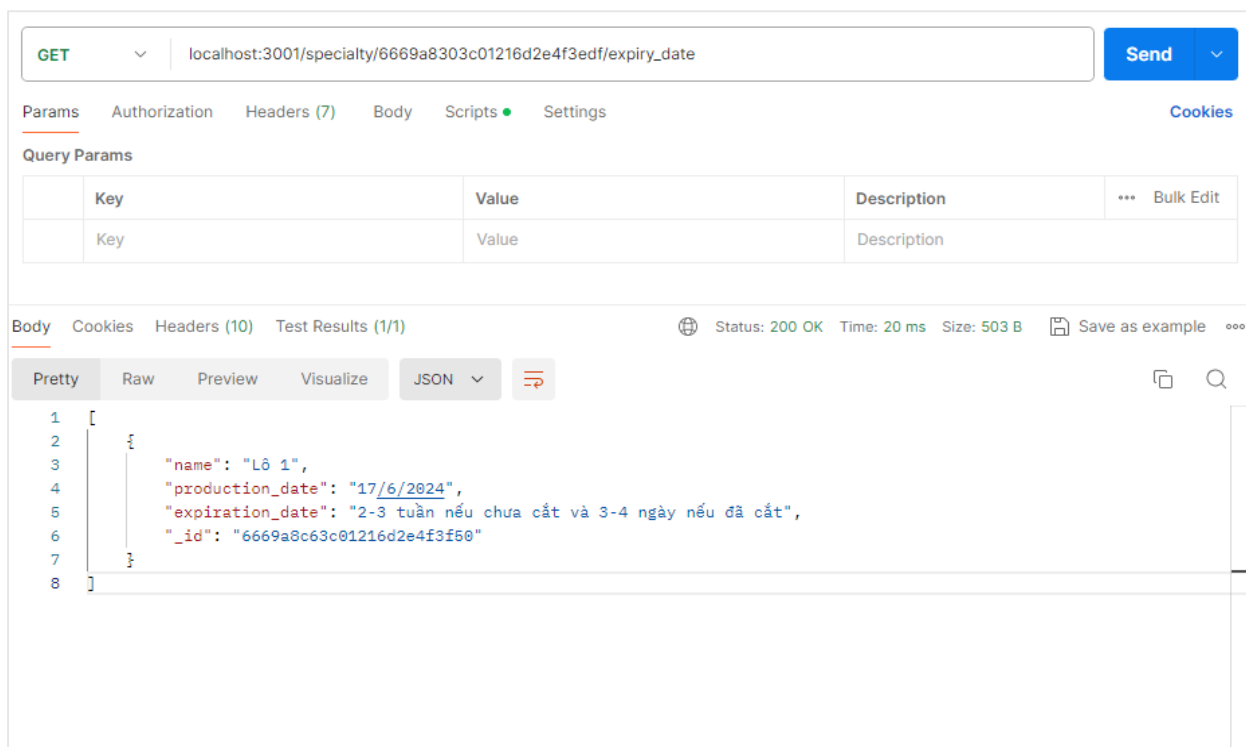
Hình 4.2. Sử dụng API lấy thông tin đặc sản theo ID với Postman



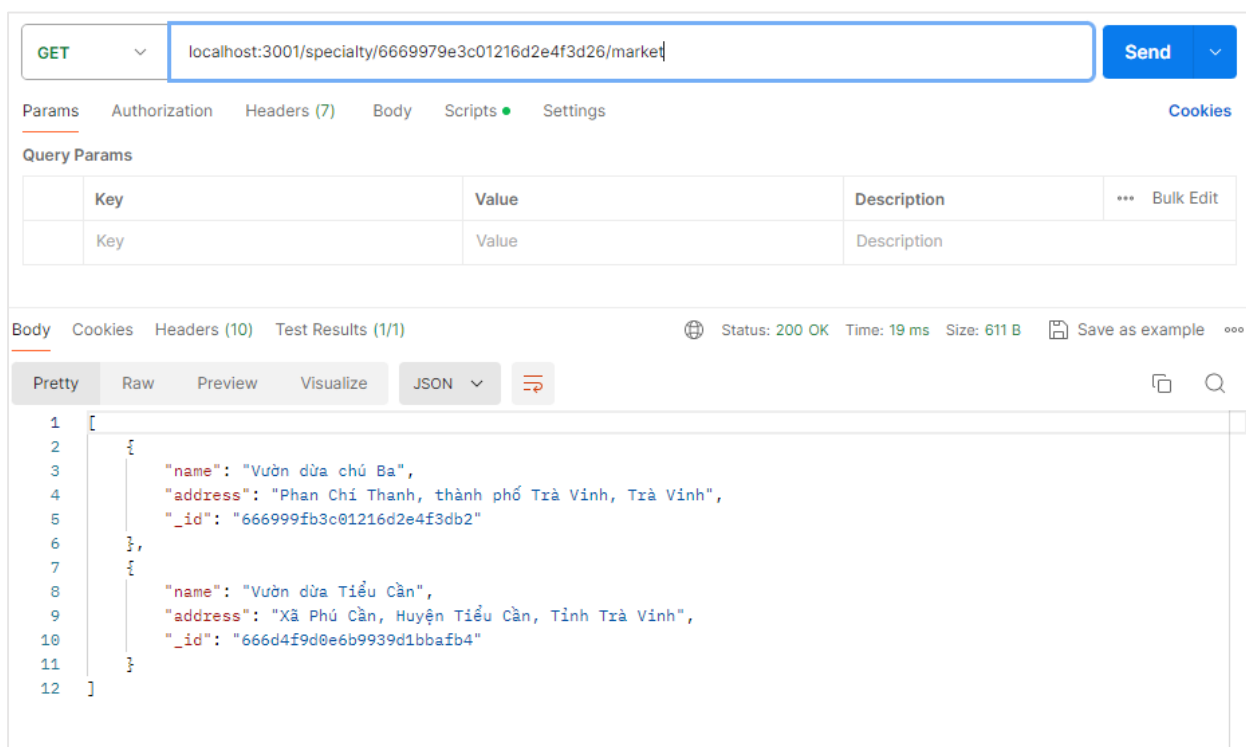
Hình 4.3. Sử dụng API lấy ra thông tin về loại của một đặc sản với Postman



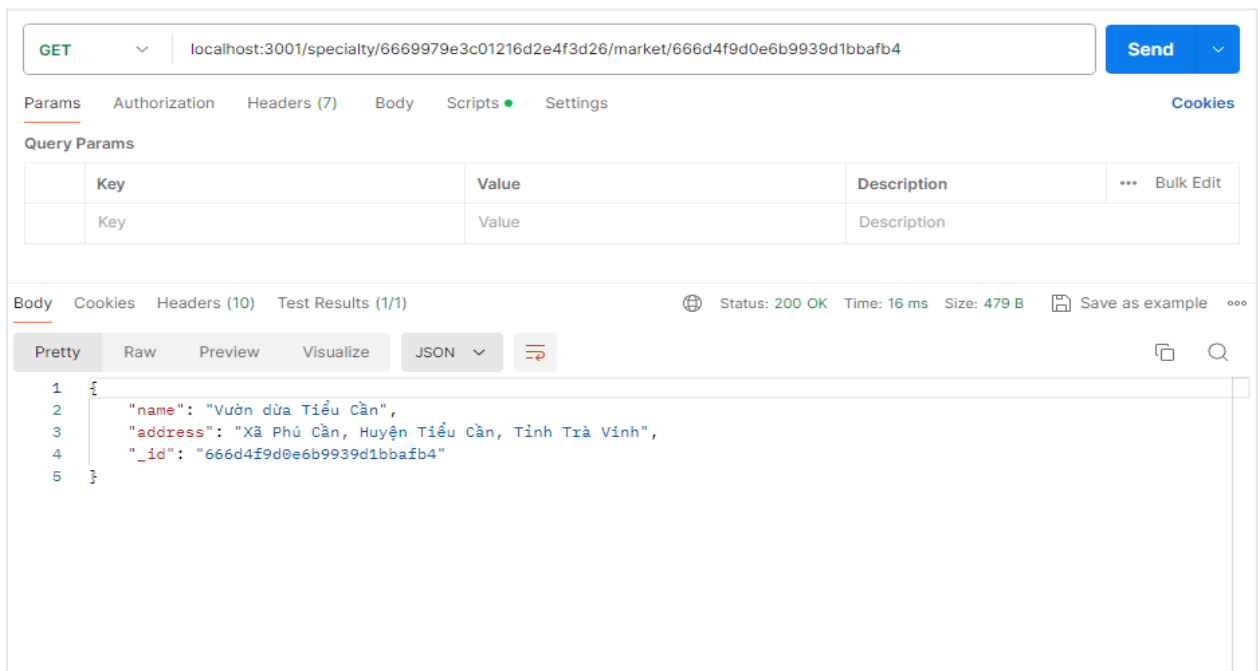
Hình 4.4. Sử dụng API lấy ra thông tin nhà sản xuất của một đặc sản với Postman



Hình 4.5. Sử dụng API lấy ra thông tin hạn sử dụng của một đặc sản với Postman



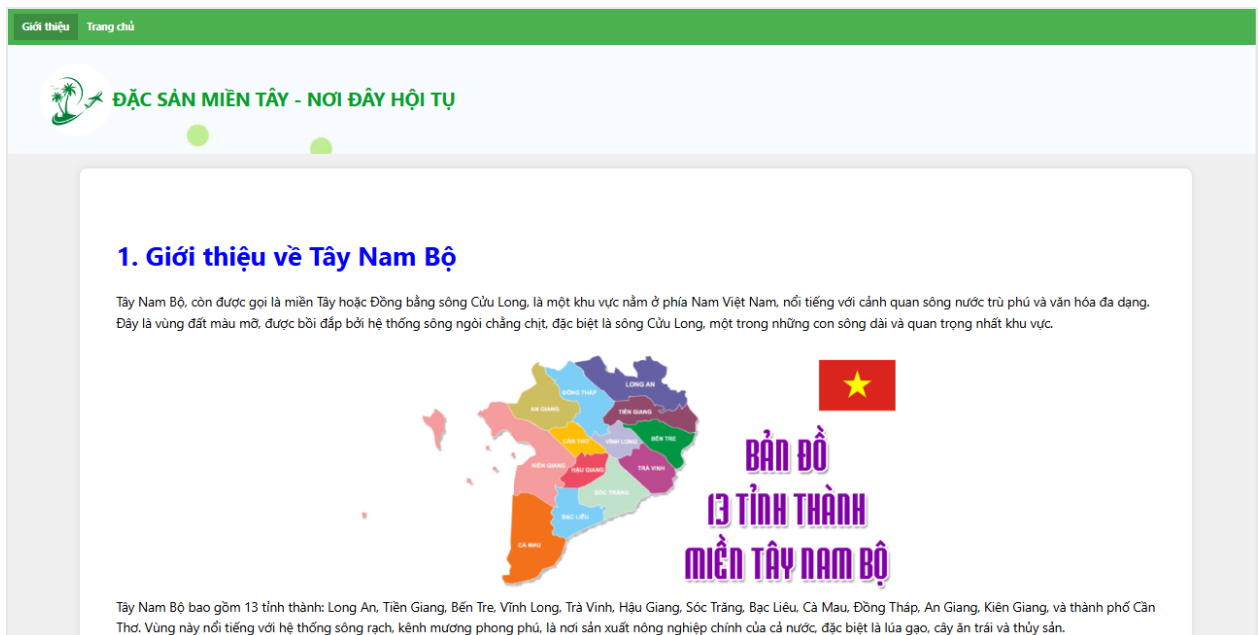
Hình 4.6. Sử dụng API lấy ra danh sách các địa điểm bán của một đặc sản với Postman



Hình 4.7. Sử dụng API lấy ra thông tin nơi bán theo ID của một đặc sản với Postman

4.2. Giao diện chức năng người dùng

4.2.1. Giao diện trang giới thiệu hệ thống



Hình 4.8. Giao diện giới thiệu Tây Nam Bộ

* Thức ăn

Bánh pía Sóc Trăng: Bánh pía là loại bánh ngọt làm từ bột mì, đậu xanh, sầu riềng và lòng đỏ trứng muối. Vỏ bánh mỏng nhiều lớp, nhân bánh thơm ngon, béo ngậy.



Hình 4.9. Giao diện giới thiệu một số đặc sản loại thức ăn

* Trái cây

Dừa sáp: Dừa sáp có lớp cùi dày, mềm dẻo và có vị béo ngậy, khác biệt so với các loại dừa thông thường.



Hình 4.10. Giao diện giới thiệu một số đặc sản loại trái cây

* Rượu

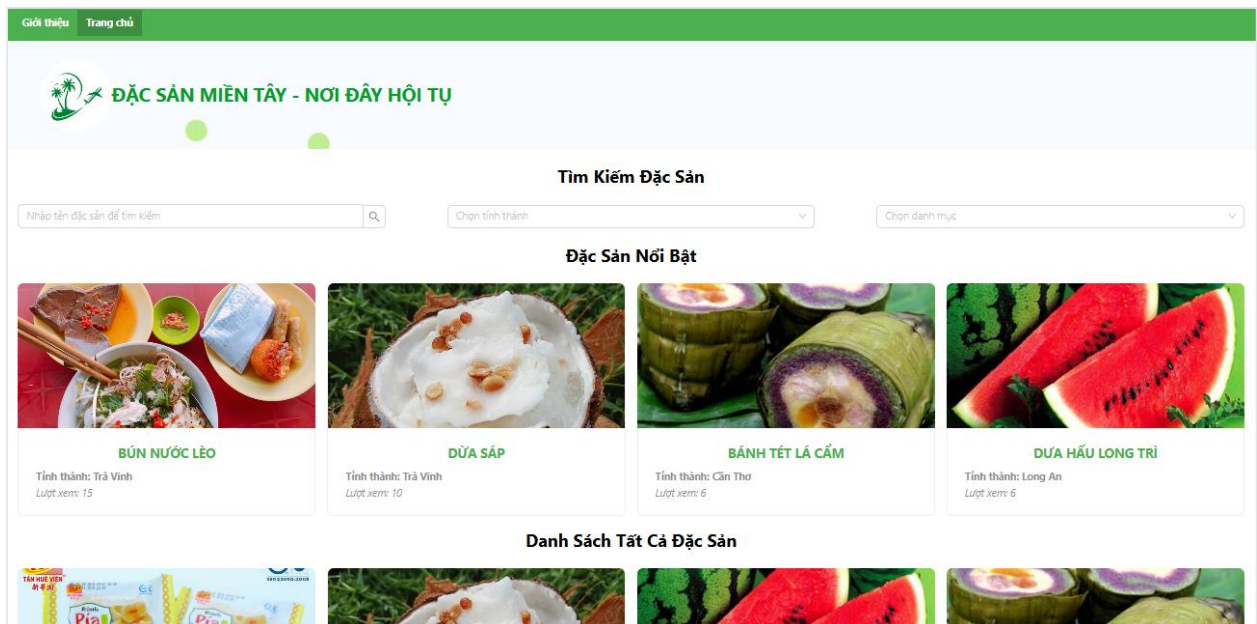
Rượu xuân thanh: Rượu Xuân Thanh không chỉ là đồ uống phổ biến mà còn là món quà ý nghĩa trong các dịp lễ tết của người dân vùng sông nước.



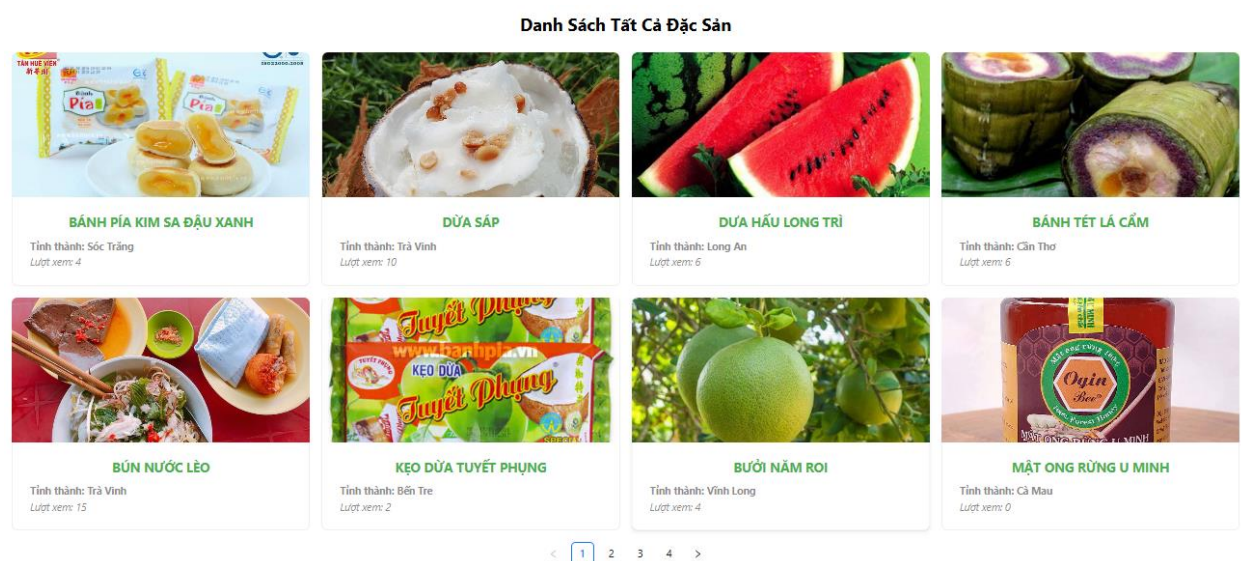
Hình 4.11. Giao diện giới thiệu một số đặc sản loại rượu

4.2.2. Giao diện trang chủ

Trên giao diện trang chủ, người dùng sẽ có thể dễ dàng xem tất cả các đặc sản có trên hệ thống. Mỗi đặc sản sẽ được trình bày với ảnh minh họa sắc nét, giúp người dùng có cái nhìn trực quan về sản phẩm. Bên cạnh mỗi ảnh, tên đặc sản và tỉnh thành xuất xứ và số lượt xem sẽ được hiển thị rõ ràng. Đồng thời người dùng có thể thấy được những đặc sản nổi bật, đó là những sản phẩm được nhiều người dùng quan tâm nhất được hiển thị lên trước với số lượt xem sản phẩm được hiển thị cùng. Thiết kế trang chủ được tối ưu hóa để người dùng có thể nhanh chóng duyệt qua, tìm kiếm các đặc sản một cách thuận tiện và hiệu quả.

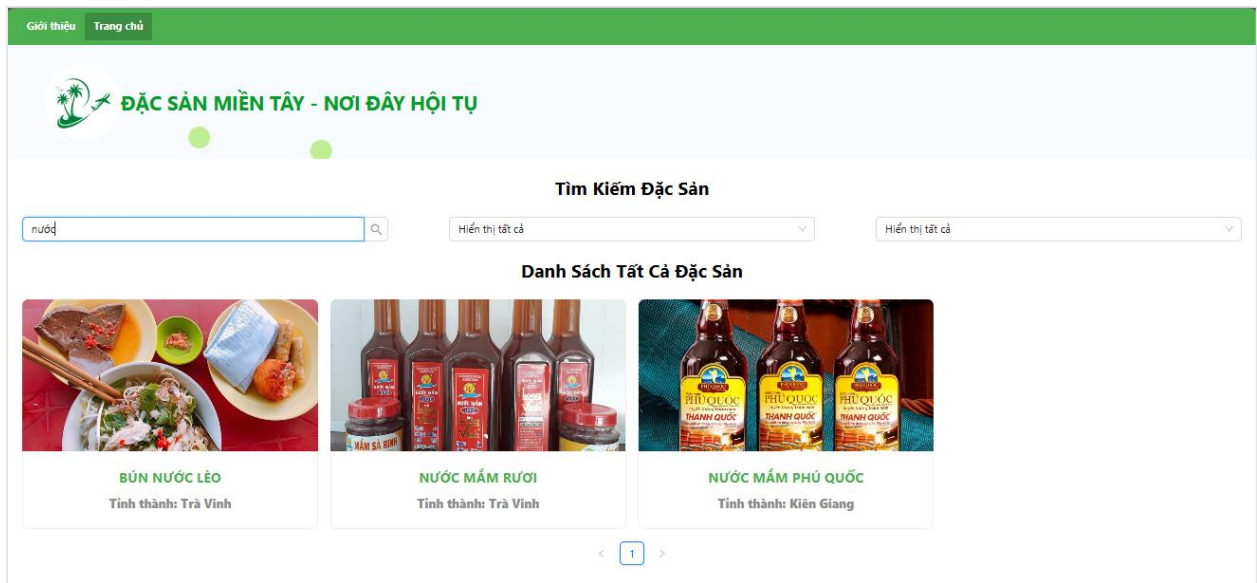


Hình 4.12. Giao diện trang chủ



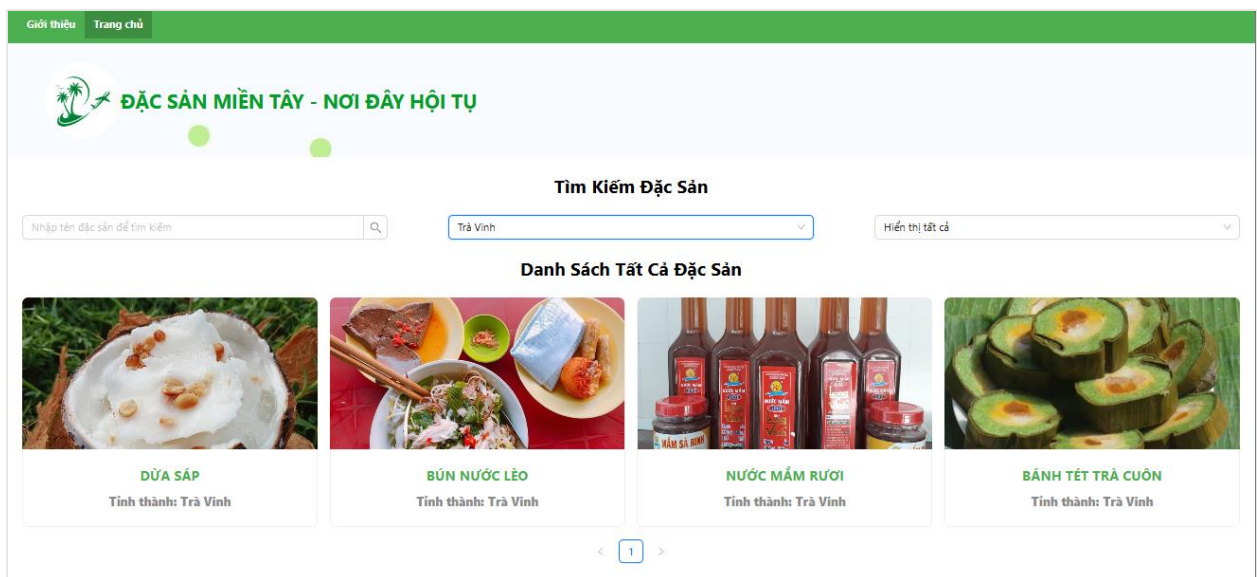
Hình 4.13. Danh sách tất cả đặc sản

Trang chủ còn cung cấp một thanh tìm kiếm để người dùng có thể tra cứu đặc sản theo tên, người dùng có thể nhập từ khoá vào thanh tìm kiếm, sau đó các đặc sản có tên trùng với từ khoá đó sẽ hiện ra.



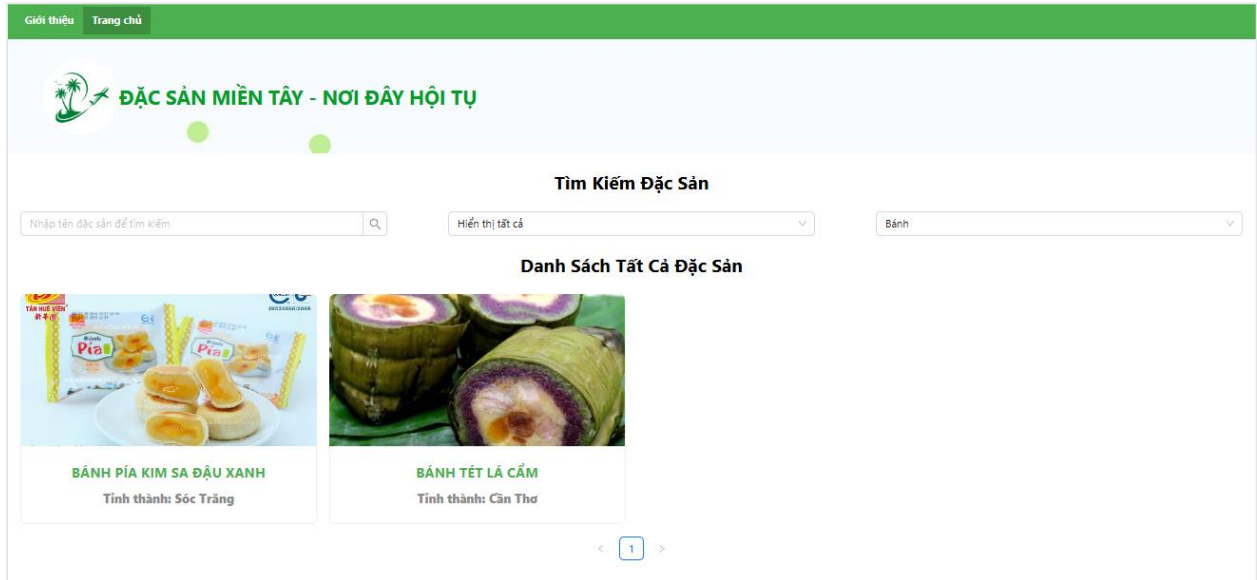
Hình 4.14. Chức năng tìm kiếm theo tên

Người dùng có thể dễ dàng tìm kiếm các đặc sản theo tỉnh thành bằng cách chọn một tỉnh thành từ danh sách thả xuống (dropdown list) nằm bên cạnh thanh tìm kiếm. Khi người dùng chọn một tỉnh thành cụ thể từ danh sách, hệ thống sẽ tự động lọc và hiển thị các đặc sản tương ứng với tỉnh thành đó, giúp người dùng nhanh chóng tìm ra những sản phẩm đến từ khu vực mình quan tâm.



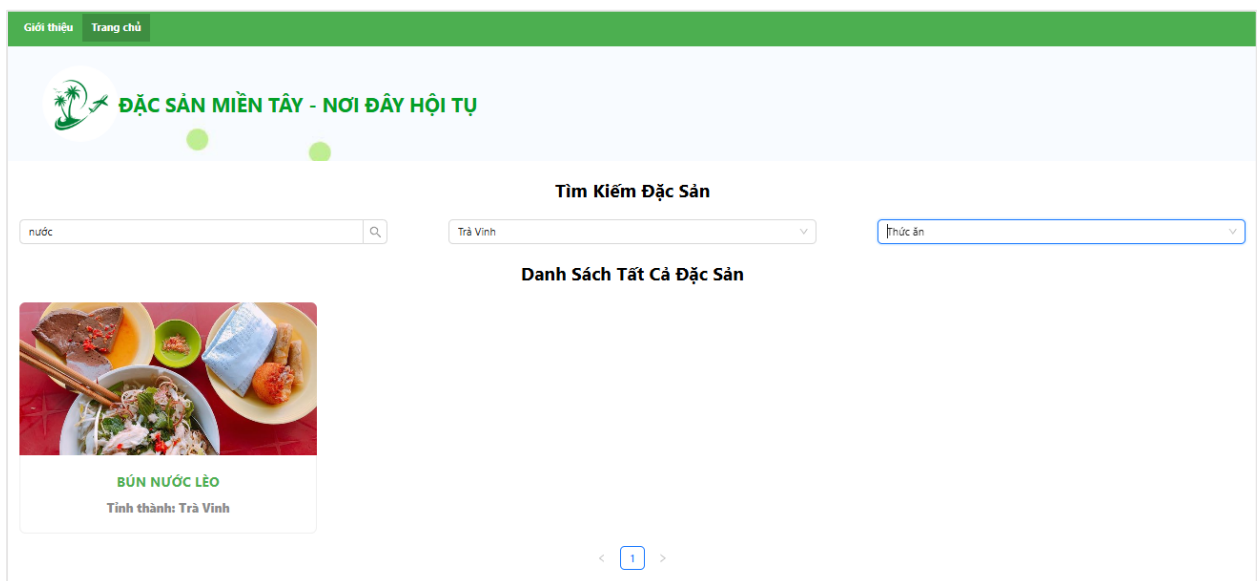
Hình 4.15. Chức năng tìm kiếm theo tỉnh thành

Bên cạnh đó, người dùng còn có thể tìm kiếm các đặc sản theo loại của đặc sản bằng cách chọn loại thành từ danh sách thả xuống bên cạnh thanh tìm kiếm theo tỉnh thành. Khi một loại cụ thể được chọn, hệ thống sẽ tự động lọc và hiển thị các đặc sản tương ứng với loại đặc sản đó, giúp người dùng nhanh chóng tìm thấy những sản phẩm mà họ cần thiết.



Hình 4.16. Tìm kiếm theo loại đặc sản

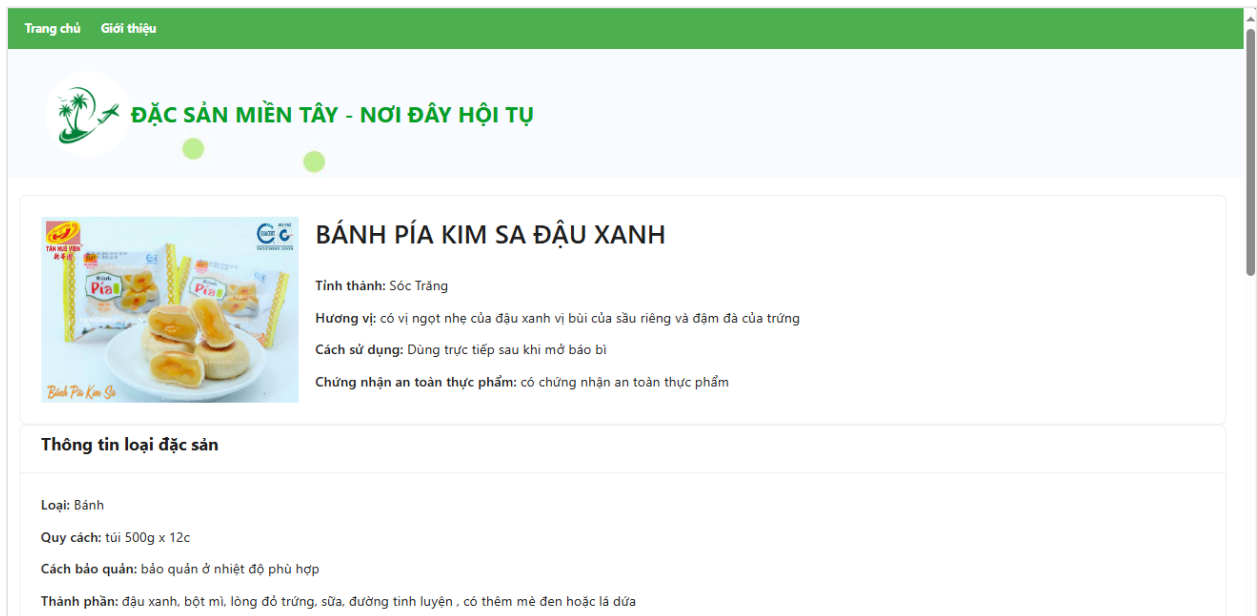
Đặc biệt, người dùng có thể kết hợp tìm kiếm theo cả tên đặc sản, tỉnh thành và loại cùng lúc để thuận tiện hơn trong việc tra cứu. Bằng cách nhập tên đặc sản vào thanh tìm kiếm đồng thời chọn một tỉnh thành và loại đặc sản từ các danh sách thả xuống bên cạnh, hệ thống sẽ lọc và hiển thị các kết quả phù hợp với cả ba tiêu chí này. Điều này giúp người dùng nhanh chóng tìm ra các đặc sản chính xác theo tên, khu vực và loại mong muốn, nâng cao hiệu quả và độ chính xác của quá trình tìm kiếm.



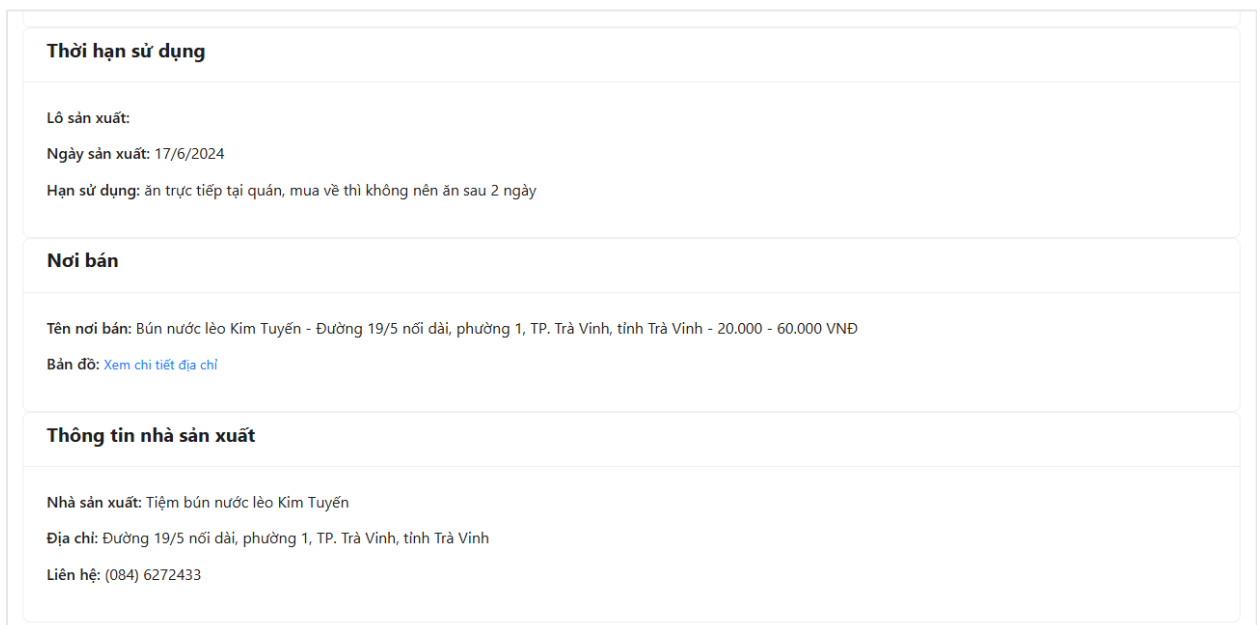
Hình 4.17. Chức năng tìm kiếm đồng thời

4.2.3. Giao diện thông tin đặc sản

Khi người dùng nhấn vào một đặc sản tại trang chủ, giao diện thông tin chi tiết của đặc sản đó sẽ xuất hiện. Tại đây, người dùng có thể xem các thông tin chi tiết như tên đặc sản, tỉnh thành xuất xứ, hương vị, cách sử dụng, chứng nhận an toàn thực phẩm, mô tả chi tiết, loại đặc sản, nhà sản xuất, các địa điểm bán và hạn sử dụng.



Hình 4.18. Giao diện thông tin của đặc sản và thông tin về loại đặc sản

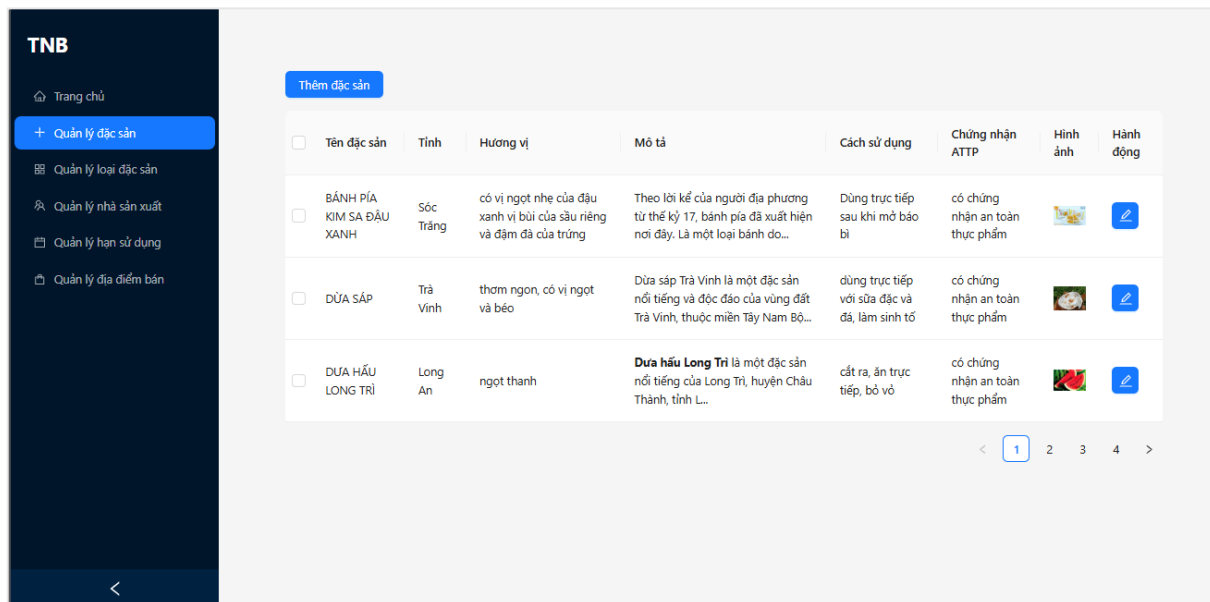


Hình 4.19. Giao diện thông tin hạn sử dụng, nơi bán, nhà sản xuất của đặc sản

4.3. Giao diện chức năng người quản trị

4.3.1. Giao diện quản lý đặc sản

Người quản trị có thể sử dụng trang quản lý đặc sản để thực hiện các thao tác với đặc sản như thêm, sửa, xóa đặc sản.



Hình 4.20. Giao diện quản lý đặc sản

Khi người quản trị chọn chức năng thêm đặc sản, giao diện sẽ chuyển sang trang nhập liệu chi tiết. Tại đây, người quản trị cần điền đầy đủ thông tin về đặc sản, bao gồm tên, tỉnh thành, mô tả và các thông tin liên quan khác. Sau khi hoàn tất, chọn nút "Thêm" để lưu lại. Đặc sản mới này sẽ ngay lập tức xuất hiện trên trang quản lý đặc sản, hiển thị tất cả các thông tin vừa được nhập.

Hình 4.21. Giao diện chức năng thêm đặc sản mới

Khi người quản trị chọn chức năng sửa đặc sản, giao diện sẽ chuyển sang trang nhập liệu với các thông tin có sẵn của đặc sản. Người quản trị có thể chỉnh sửa thông tin của đặc sản. Cuối cùng chọn “Lưu” để lưu lại.

TNB

Trang chủ

+ Quản lý đặc sản

Quản lý loại đặc sản

Quản lý nhà sản xuất

Quản lý hạn sử dụng

Quản lý địa điểm bán

Tên đặc sản

BÁNH PÍA KIM SA ĐẬU XANH

Tỉnh

Sóc Trăng

Hương vị

có vị ngọt nhẹ của đậu xanh vị bùi của sầu riêng và đậm đà của trứng

Mô tả

Theo lời kể của người địa phương từ thế kỷ 17, bánh pía đã xuất hiện nơi đây. Là một loại bánh do người Hán di cư mang sang Việt Nam. Tuy nhiên, qua thời gian bánh pía đã được biến tấu, thay đổi theo khẩu vị người Việt và trở thành đặc sản của vùng Nam bộ. Qua tìm hiểu, mới thấy được tiếng thơm này không phải ngẫu nhiên mà có. Nguyên liệu cũng chỉ bột mì, khoai môn, đậu xanh, sầu riêng, lòng đỏ trứng vịt muối, nhưng bánh pía Sóc Trăng có hương vị, cách trình bày riêng, không giống bất kì loại bánh nào. Có thể nói để có những chiếc bánh pía đạt chuẩn người ta phải qua rất nhiều công đoạn tỉ mỉ từ khâu làm bột cho đến khâu nướng. Bánh pía có 2 phần, phần nhân và vỏ. Nhân làm từ khoai môn, đậu xanh, trứng. Đậu xanh đã đãi, khoai môn gọt vỏ, rửa sạch tất cả cho vào nồi hấp chín rồi tán nhuyễn. Sau đó tiếp tục xào cùng với đường, nhân sầu riêng theo tỷ lệ vừa phải. Để hỗn hợp trên nguội, lăn lượt bột nhân quanh từng lòng đỏ trứng vịt. Ngoài ra, muốn tăng vị béo đậm đà có thể thêm thịt heo vào phần nhân.

Cách sử dụng

Dùng trực tiếp sau khi mở bảo bì

Chứng nhận ATTP

có chứng nhận an toàn thực phẩm

Hình ảnh

Chọn hình ảnh

Lưu

Hình 4.22. Giao diện chức năng sửa thông tin đặc sản

Người quản trị có thể xóa đặc sản bằng cách chọn một hoặc nhiều mục cần xóa. Sau đó, nhấn nút "Xóa" và một hộp thoại xác nhận sẽ xuất hiện để hỏi xem có chắc chắn muốn xóa các mục đã chọn hay không. Nếu xác nhận, các đặc sản này sẽ được xóa khỏi hệ thống.

TNB

Trang chủ

+ Quản lý đặc sản

Quản lý loại đặc sản

Quản lý nhà sản xuất

Quản lý hạn sử dụng

Quản lý địa điểm bán

Thêm đặc sản Xóa 3 đặc sản đã chọn

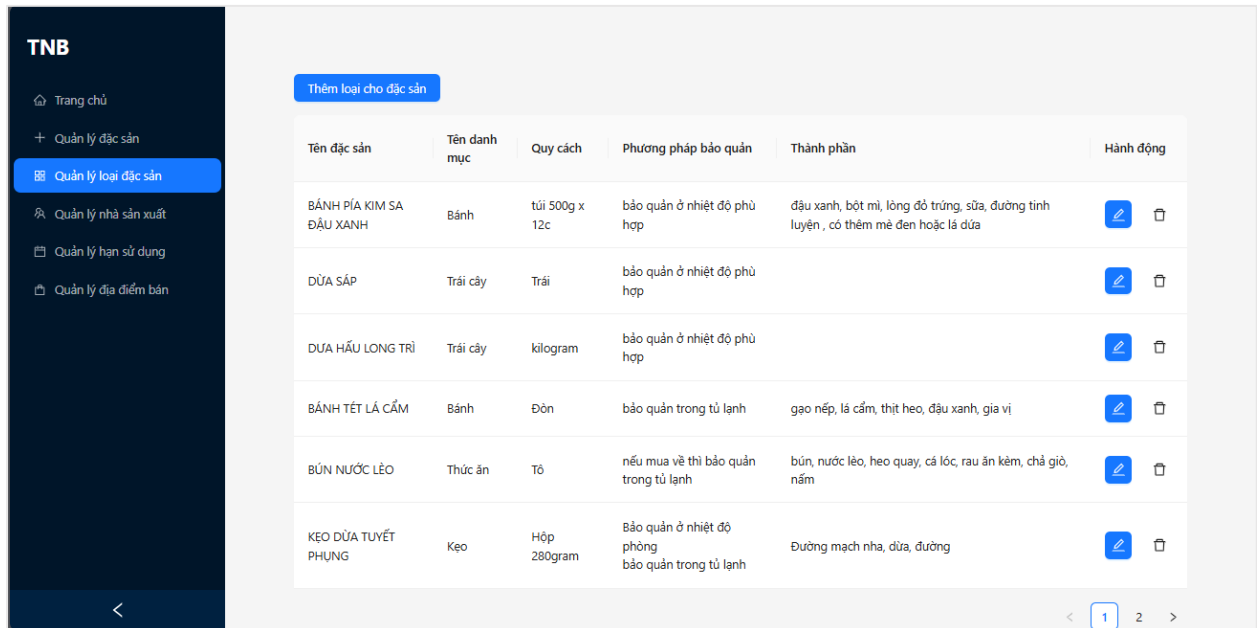
<input checked="" type="checkbox"/>	Tên đặc sản	Tỉnh	Hương vị	Mô tả	Cách sử dụng	Chứng nhận ATTP	Hình ảnh	Hành động
<input checked="" type="checkbox"/>	BÁNH PÍA KIM SA ĐẬU XANH	Sóc Trăng	có vị ngọt nhẹ của đậu xanh vị bùi của sầu riêng và đậm đà của trứng	Theo lời kể của người địa phương từ thế kỷ 17, bánh pía đã xuất hiện nơi đây. Là một loại bánh do...	Dùng trực tiếp sau khi mở bảo bì	có chứng nhận an toàn thực phẩm		
<input checked="" type="checkbox"/>	DỪA SÁP	Trà Vinh	thơm ngon, có vị ngọt và béo	Dừa sáp Trà Vinh là một đặc sản nổi tiếng và độc đáo của vùng đất Trà Vinh, thuộc miền Tây Nam Bộ...	dùng trực tiếp với sữa đặc và đá, làm sinh tố	có chứng nhận an toàn thực phẩm		
<input checked="" type="checkbox"/>	DỪA HẤU LONG TRỊ	Long An	ngọt thanh	Dừa hấu Long Trị là một đặc sản nổi tiếng của Long Trị, huyện Châu Thành, tỉnh L...	cắt ra, ăn trực tiếp, bỏ vỏ	có chứng nhận an toàn thực phẩm		

< 1 2 3 4 >

Hình 4.23. Chức năng xoá nhiều đặc sản

4.3.2. Giao diện quản lý loại đặc sản

Người quản trị có thể sử dụng trang quản lý loại đặc sản để chọn một đặc sản và thêm loại cho đặc sản đó, đồng thời người quản trị còn có thể sửa, xoá loại của đặc sản.

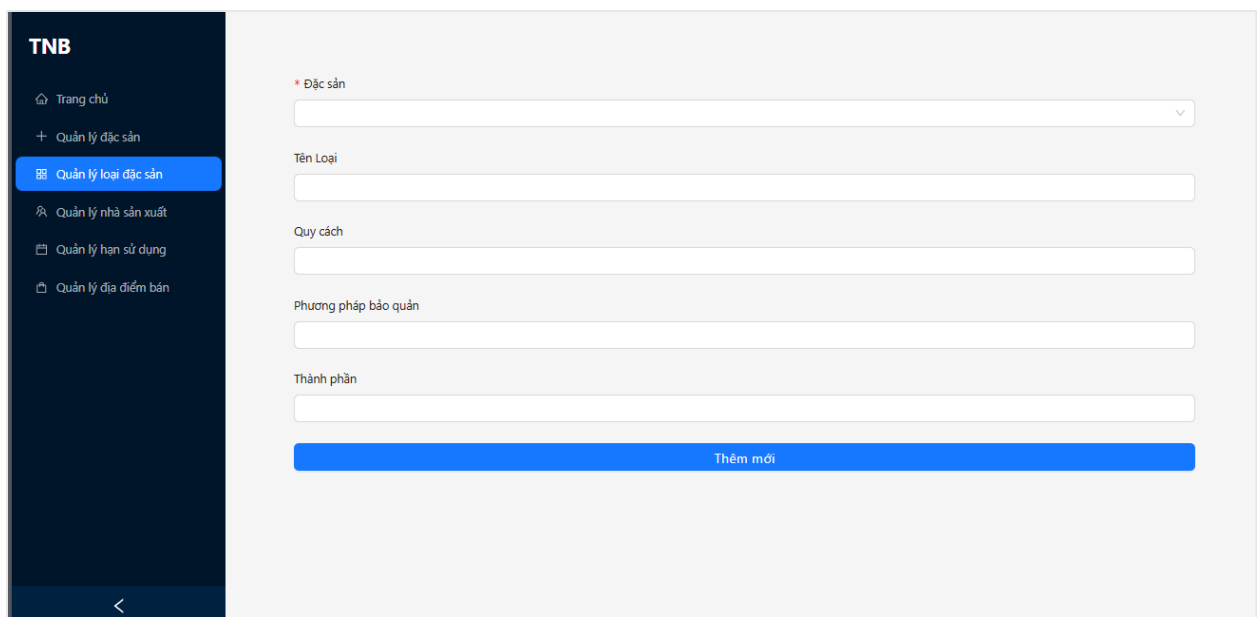


The screenshot displays a web application interface for managing special products. On the left is a dark sidebar with the logo 'TNB' and a menu containing: 'Trang chủ', 'Quản lý đặc sản', 'Quản lý loại đặc sản' (highlighted), 'Quản lý nhà sản xuất', 'Quản lý hạn sử dụng', and 'Quản lý địa điểm bán'. The main content area has a light gray background. At the top left of this area is a blue button labeled 'Thêm loại cho đặc sản'. Below it is a table with the following columns: 'Tên đặc sản', 'Tên danh mục', 'Quy cách', 'Phương pháp bảo quản', 'Thành phần', and 'Hành động'. The table contains six rows of data. Each row has two icons in the 'Hành động' column: a blue pencil for editing and a gray trash can for deleting. At the bottom right of the main area, there are pagination controls showing '< 1 2 >'.

Tên đặc sản	Tên danh mục	Quy cách	Phương pháp bảo quản	Thành phần	Hành động
BÁNH PÍA KIM SA ĐẦU XANH	Bánh	túi 500g x 12c	bảo quản ở nhiệt độ phù hợp	đậu xanh, bột mì, lòng đỏ trứng, sữa, đường tinh luyện, có thêm mè đen hoặc lá dứa	
DỪA SÁP	Trái cây	Trái	bảo quản ở nhiệt độ phù hợp		
DỪA HẦU LONG TRÌ	Trái cây	kilogram	bảo quản ở nhiệt độ phù hợp		
BÁNH TÉT LÁ CẨM	Bánh	Đòn	bảo quản trong tủ lạnh	gạo nếp, lá cẩm, thịt heo, đậu xanh, gia vị	
BÚN NƯỚC LÈO	Thức ăn	Tô	nếu mua về thì bảo quản trong tủ lạnh	bún, nước lèo, heo quay, cá lóc, rau ăn kèm, chả giò, nấm	
KEO DỪA TUYẾT PHỤNG	Kẹo	Hộp 280gram	Bảo quản ở nhiệt độ phòng, bảo quản trong tủ lạnh	Đường mạch nha, dừa, đường	

Hình 4.24. Giao diện quản lý loại đặc sản

Khi người quản trị chọn chức năng thêm loại cho đặc sản, giao diện sẽ chuyển sang trang nhập thông tin chi tiết. Tại đây, người quản trị cần lựa chọn đặc sản cụ thể, sau đó điền đầy đủ thông tin về loại cho đặc sản đó. Sau khi nhập liệu xong, người quản trị nhấn nút "Thêm" để lưu lại. Loại của đặc sản được chọn sẽ được thêm vào với tất cả thông tin đã được nhập.



The screenshot shows the 'TNB' management interface with the 'Quản lý loại đặc sản' menu item selected. The main area displays a form for adding a new type for a special product. At the top left of the form is a red asterisk and the text '* Đặc sản'. Below this is a dropdown menu. The form contains five text input fields labeled: 'Tên Loại', 'Quy cách', 'Phương pháp bảo quản', and 'Thành phần'. At the bottom of the form is a large blue button labeled 'Thêm mới'.

Hình 4.25. Giao diện chức năng thêm loại cho đặc sản

Khi người quản trị chọn chức năng sửa loại của đặc sản, giao diện sẽ chuyển sang trang nhập liệu với các thông tin loại có sẵn của đặc sản. Người quản trị có thể chỉnh sửa thông tin loại của đặc sản. Cuối cùng chọn “Lưu” để lưu lại.

TNB

- Trang chủ
- Quản lý đặc sản
- Quản lý loại đặc sản**
- Quản lý nhà sản xuất
- Quản lý hạn sử dụng
- Quản lý địa điểm bán

* Tên Loại

Bánh

Quy cách

túi 500g x 12c

Phương pháp bảo quản

bảo quản ở nhiệt độ phù hợp

Thành phần

đậu xanh, bột mì, lòng đỏ trứng, sữa, đường tinh luyện, có thêm mè đen hoặc lá dứa

Cập nhật

Hình 4.26. Giao diện chức năng sửa thông tin về loại cho đặc sản

Người quản trị có thể xóa loại của đặc sản bằng cách chọn biểu tượng xóa. Sau đó, một hộp thoại xác nhận sẽ xuất hiện để hỏi xem có chắc chắn muốn xóa không. Nếu xác nhận, loại đặc sản đó sẽ được xóa khỏi hệ thống.

TNB

- Trang chủ
- Quản lý đặc sản
- Quản lý loại đặc sản**
- Quản lý nhà sản xuất
- Quản lý hạn sử dụng
- Quản lý địa điểm bán

Thêm loại cho đặc sản

Tên đặc sản	Tên danh mục	Quy cách	Phương pháp bảo quản	Thành phần		
BÁNH PÍA KIM SA ĐẬU XANH	Bánh	túi 500g x 12c	bảo quản ở nhiệt độ phù hợp	đậu xanh, bột mì, lòng đỏ trứng, sữa, đường tinh luyện, có thêm mè đen hoặc lá dứa		
DỪA SÁP	Trái cây	Trái	bảo quản ở nhiệt độ phù hợp			
DỪA HẦU LONG TRÌ	Trái cây	kilogram	bảo quản ở nhiệt độ phù hợp			
BÁNH TẾT LÁ CẨM	Bánh	Đòn	bảo quản trong tủ lạnh	gạo nếp, lá cẩm, thịt heo, đậu xanh, gia vị		
BÚN NƯỚC LÈO	Thức ăn	Tô	nếu mua về thì bảo quản trong tủ lạnh	bún, nước lèo, heo quay, cá lóc, rau ăn kèm, chả giò, nấm		
KEO DỪA TUYẾT PHỤNG	Kẹo	Hộp 280gram	Bảo quản ở nhiệt độ phòng bảo quản trong tủ lạnh	Đường mạch nha, dừa, đường		

Bạn có chắc chắn muốn xóa danh mục này không?

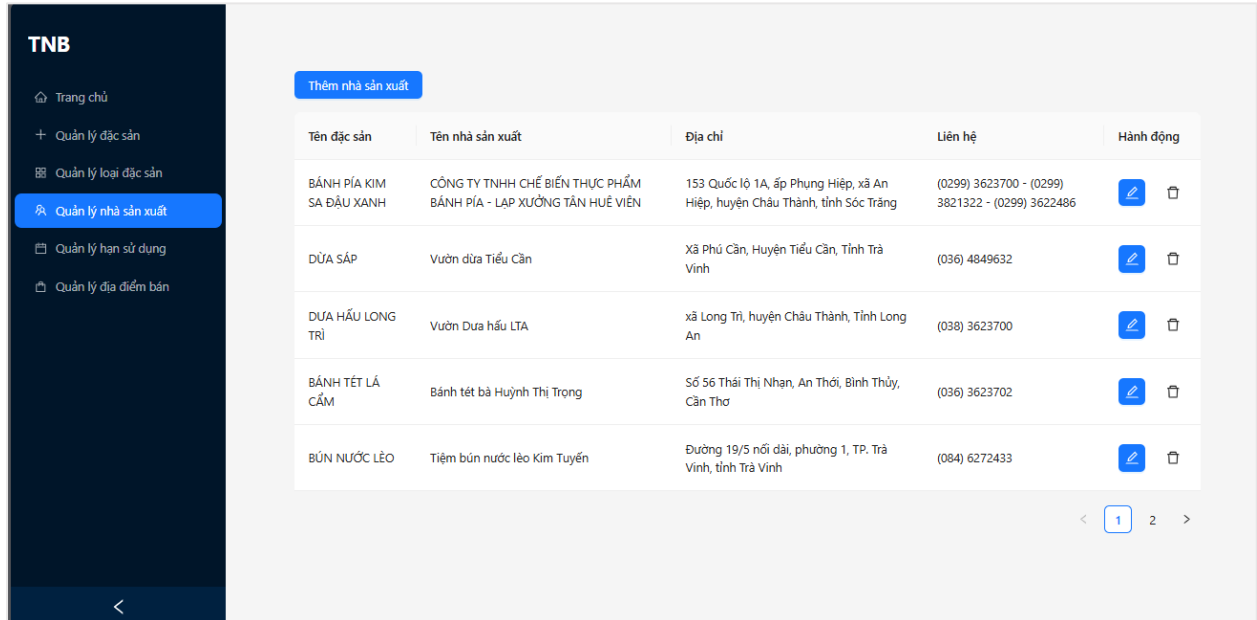
Không Có











< 1 2 >

Hình 4.27. Chức năng xóa loại của đặc sản

4.3.3. Giao diện quản lý nhà sản xuất

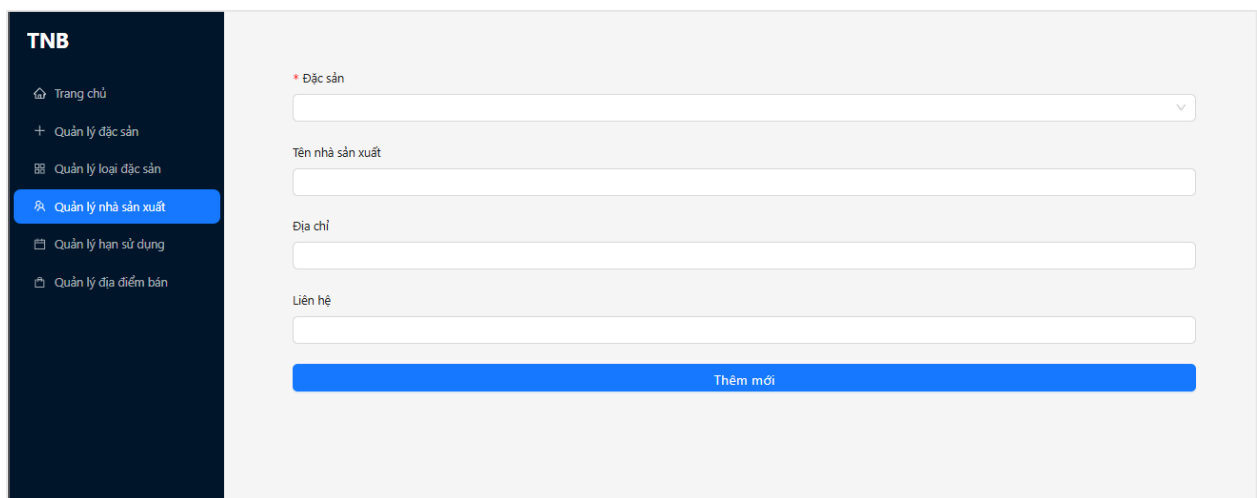
Người quản trị có thể sử dụng trang quản lý nhà sản xuất để chọn một đặc sản và thêm nhà sản xuất cho đặc sản đó, đồng thời người quản trị còn có thể sửa, xóa nhà sản xuất của đặc sản.



Thêm nhà sản xuất				
Tên đặc sản	Tên nhà sản xuất	Địa chỉ	Liên hệ	Hành động
BÁNH PÍA KIM SA ĐÀU XANH	CÔNG TY TNHH CHẾ BIẾN THỰC PHẨM BÁNH PÍA - LẠP XƯỚNG TÂN HUỆ VIÊN	153 Quốc lộ 1A, ấp Phụng Hiệp, xã An Hiệp, huyện Châu Thành, tỉnh Sóc Trăng	(0299) 3623700 - (0299) 3821322 - (0299) 3622486	 
DỪA SÁP	Vườn dừa Tiểu Cần	Xã Phú Cần, Huyện Tiểu Cần, Tỉnh Trà Vinh	(036) 4849632	 
DỪA HẦU LONG TRÌ	Vườn Dừa hầu LTA	xã Long Trì, huyện Châu Thành, Tỉnh Long An	(038) 3623700	 
BÁNH TẾT LÁ CẨM	Bánh tết bà Huỳnh Thị Trọng	Số 56 Thái Thị Nhan, An Thới, Bình Thủy, Cần Thơ	(036) 3623702	 
BÚN NƯỚC LÈO	Tiệm bún nước lèo Kim Tuyến	Đường 19/5 nối dài, phường 1, TP. Trà Vinh, tỉnh Trà Vinh	(084) 6272433	 

Hình 4.28. Giao diện quản lý nhà sản xuất

Khi người quản trị chọn chức năng thêm nhà sản xuất, giao diện sẽ chuyển sang trang nhập thông tin chi tiết. Tại đây, người quản trị cần lựa chọn đặc sản cụ thể, sau đó điền đầy đủ thông tin về nhà sản xuất cho đặc sản đó. Sau khi nhập liệu xong, người quản trị nhấn nút "Thêm" để lưu lại. Nhà sản xuất của đặc sản được chọn sẽ được thêm vào với tất cả thông tin đã được nhập.



TNB

Trang chủ

Quản lý đặc sản

Quản lý loại đặc sản

Quản lý nhà sản xuất

Quản lý hạn sử dụng

Quản lý địa điểm bán

* Đặc sản

Tên nhà sản xuất

Địa chỉ

Liên hệ

Thêm mới

Hình 4.29. Giao diện chức năng thêm nhà sản xuất

Khi người quản trị chọn chức năng sửa nhà sản xuất của đặc sản, giao diện sẽ chuyển sang trang nhập liệu với các thông tin nhà sản xuất có sẵn của đặc sản. Người quản trị có thể chỉnh sửa thông tin nhà sản xuất của đặc sản. Cuối cùng chọn “Lưu” để lưu lại.

TNB

- Trang chủ
- Quản lý đặc sản
- Quản lý loại đặc sản
- Quản lý nhà sản xuất**
- Quản lý hạn sử dụng
- Quản lý địa điểm bán

* Tên nhà sản xuất

CÔNG TY TNHH CHẾ BIẾN THỰC PHẨM BÁNH PÍA - LAP XƯỜNG TÂN HUẾ VIÊN

Địa chỉ

153 Quốc lộ 1A, ấp Phụng Hiệp, xã An Hiệp, huyện Châu Thành, tỉnh Sóc Trăng

Liên hệ

(0299) 3623700 - (0299) 3821322 - (0299) 3622486

Cập nhật

Hình 4.30. Giao diện chức năng sửa thông tin nhà sản xuất

Người quản trị có thể xóa nhà sản xuất của đặc sản bằng cách chọn biểu tượng xóa. Sau đó, một hộp thoại xác nhận sẽ xuất hiện để hỏi xem có chắc chắn muốn xóa không. Nếu xác nhận, nhà sản xuất của đặc sản đó sẽ được xóa khỏi hệ thống.

TNB

- Trang chủ
- Quản lý đặc sản
- Quản lý loại đặc sản
- Quản lý nhà sản xuất**
- Quản lý hạn sử dụng
- Quản lý địa điểm bán

Thêm nhà sản xuất

Tên đặc sản	Tên nhà sản xuất	Địa chỉ	Liên	
BÁNH PÍA KIM SA ĐẬU XANH	CÔNG TY TNHH CHẾ BIẾN THỰC PHẨM BÁNH PÍA - LAP XƯỜNG TÂN HUẾ VIÊN	153 Quốc lộ 1A, ấp Phụng Hiệp, xã An Hiệp, huyện Châu Thành, tỉnh Sóc Trăng	(0299) 3623700 - (0299) 3821322 - (0299) 3622486	
DỪA SÁP	Vườn dừa Tiều Cẩn	Xã Phú Cẩn, Huyện Tiều Cẩn, Tỉnh Trà Vinh	(036) 4849632	
DỪA HẦU LONG TRÌ	Vườn Dừa hầu LTA	xã Long Trĩ, huyện Châu Thành, Tỉnh Long An	(038) 3623700	
BÁNH TÉT LÁ CẨM	Bánh tét bà Huỳnh Thị Trọng	Số 56 Thái Thị Nhạn, An Thới, Bình Thủy, Cần Thơ	(036) 3623702	
BÚN NƯỚC LEO	Tiệm bún nước leo Kim Tuyến	Đường 19/5 nối dài, phường 1, TP. Trà Vinh, tỉnh Trà Vinh	(084) 6272433	

Bạn có chắc chắn muốn xóa danh mục này không?

< 1 2 >

Hình 4.31. Chức năng xóa nhà sản xuất

4.3.4. Giao diện quản lý hạn sử dụng

Người quản trị có thể sử dụng trang quản lý hạn sử dụng để chọn một đặc sản và thêm hạn sử dụng cho đặc sản đó, đồng thời người quản trị còn có thể sửa, xóa hạn sử dụng của đặc sản.

Tên đặc sản	Lô sản xuất	Ngày sản xuất	Hạn sử dụng	Hành động
BÁNH PÍA KIM SA ĐẬU XANH	Lô 1	15/6/2024	6 tháng sau ngày sản xuất	✎ 🗑
DỪA SÁP		15/6/2024	25-30 nếu bảo quản đúng cách	✎ 🗑
DỪA HẦU LONG TRÌ	Lô 1	17/6/2024	2-3 tuần nếu chưa cắt và 3-4 ngày nếu đã cắt	✎ 🗑
BÁNH TÉT LÁ CẨM		12/6/2024	3-5 ngày bảo quản trong tủ lạnh	✎ 🗑
BÚN NƯỚC LÈO		17/6/2024	ăn trực tiếp tại quán, mua về thì không nên ăn sau 2 ngày	✎ 🗑
KEO DỪA TUYẾT PHỤNG	Lô 2	18/6/2024	1-2 tháng nếu bảo quản đúng cách	✎ 🗑
BƯỞI NẤM ROI	đợt 1	18/6/2024	1-2 tháng nếu bảo quản đúng cách	✎ 🗑

Hình 4.32. Giao diện quản lý hạn sử dụng

Khi người quản trị chọn chức năng thêm hạn sử dụng, giao diện sẽ chuyển sang trang nhập thông tin chi tiết. Tại đây, người quản trị cần lựa chọn đặc sản cụ thể, sau đó điền đầy đủ thông tin về hạn sử dụng cho đặc sản đó. Sau khi nhập liệu xong, người quản trị nhấn nút "Thêm" để lưu lại. Hạn sản xuất của đặc sản được chọn sẽ được thêm vào với tất cả thông tin đã được nhập.

* Đặc sản

Lô sản xuất

Ngày sản xuất

Hạn sử dụng

Thêm mới

Hình 4.33. Giao diện chức năng thêm hạn sử dụng

Khi người quản trị chọn chức năng sửa hạn sử dụng của đặc sản, giao diện sẽ chuyển sang trang nhập liệu với các thông tin hạn sử dụng có sẵn của đặc sản. Người quản trị có thể chỉnh sửa thông tin hạn sử dụng của đặc sản. Cuối cùng chọn “Lưu” để lưu lại.

TNB

- Trang chủ
- Quản lý đặc sản
- Quản lý loại đặc sản
- Quản lý nhà sản xuất
- Quản lý hạn sử dụng**
- Quản lý địa điểm bán

* Tên lô sản xuất

Lô 1

Ngày sản xuất

15/6/2024

Hạn sử dụng

6 tháng sau ngày sản xuất

Cập nhật

Hình 4.34. Giao diện chức năng sửa hạn sử dụng

Người quản trị có thể xóa nhà sản xuất của đặc sản bằng cách chọn biểu tượng xóa. Sau đó, một hộp thoại xác nhận sẽ xuất hiện để hỏi xem có chắc chắn muốn xóa không. Nếu xác nhận, nhà sản xuất của đặc sản đó sẽ được xóa khỏi hệ thống.

TNB

- Trang chủ
- Quản lý đặc sản
- Quản lý loại đặc sản
- Quản lý nhà sản xuất
- Quản lý hạn sử dụng**
- Quản lý địa điểm bán

Thêm ngày sản xuất

Tên đặc sản	Lô sản xuất	Ngày sản xuất	Hạn sử dụng	
BÁNH PÍA KIM SA ĐẦU XANH	Lô 1	15/6/2024	6 tháng sau ngày sản xuất	
DỪA SẮP		15/6/2024	25-30 nếu bảo quản đúng cách	
DƯA HẦU LONG TRÌ	Lô 1	17/6/2024	2-3 tuần nếu chưa cắt và 3-4 ngày nếu đã cắt	
BÁNH TẾT LÁ CẨM		12/6/2024	3-5 ngày bảo quản trong tủ lạnh	
BÚN NƯỚC LÈO		17/6/2024	ăn trực tiếp tại quán, mua về thì không nên ăn sau 2 ngày	
KẸO DỪA TUYẾT PHỤNG	Lô 2	18/6/2024	1-2 tháng nếu bảo quản đúng cách	
BƯỞI NĂM ROI	đợt 1	18/6/2024	1-2 tháng nếu bảo quản đúng cách	

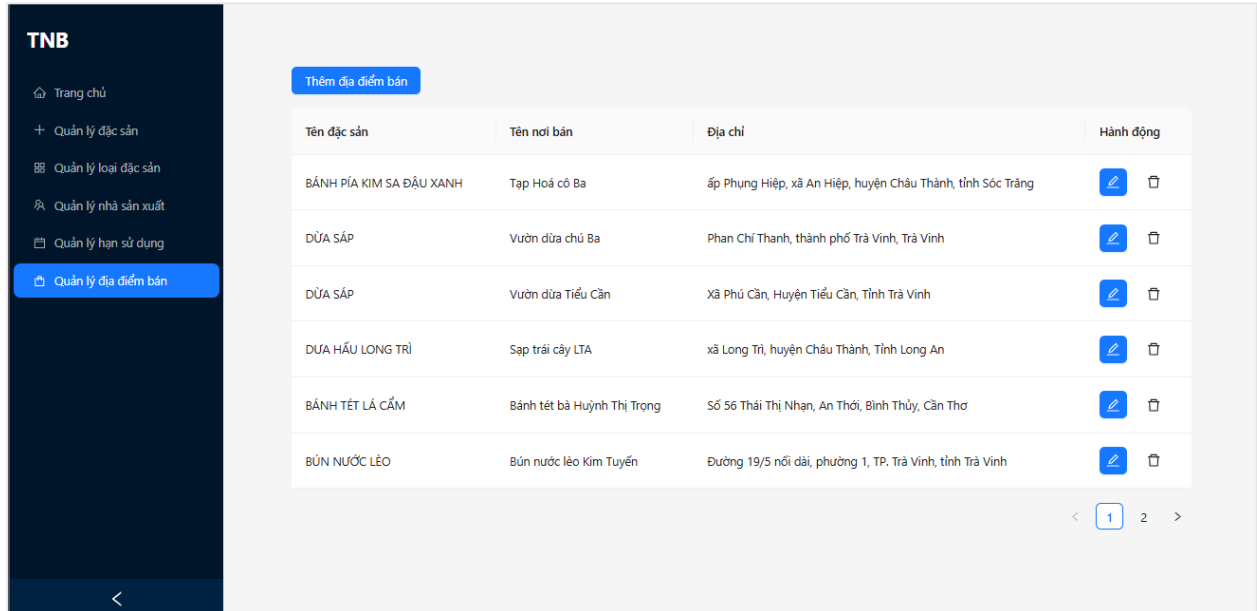
Bạn có chắc chắn xóa hạn sử dụng này?

< 1 2 >

Hình 4.35. Chức năng xóa hạn sử dụng của đặc sản

4.3.5. Giao diện quản lý nơi bán

Người quản trị có thể sử dụng trang quản lý địa điểm bán để chọn một đặc sản và thêm địa điểm bán cho đặc sản đó, đồng thời người quản trị còn có thể sửa, xóa địa điểm bán của đặc sản.

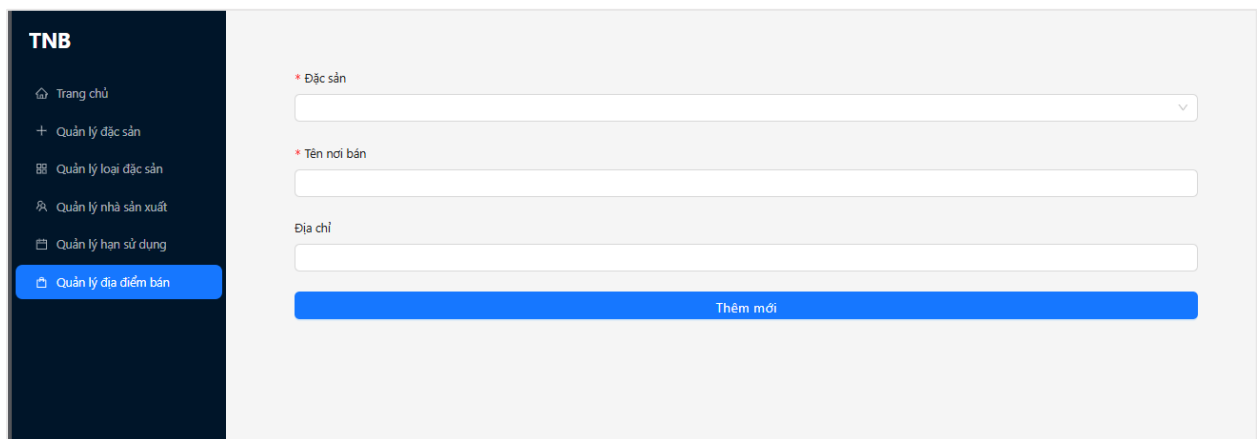


The screenshot displays the 'TNB' (Quản lý địa điểm bán) interface. On the left is a dark sidebar with navigation links: Trang chủ, Quản lý đặc sản, Quản lý loại đặc sản, Quản lý nhà sản xuất, and Quản lý hạn sử dụng. The 'Quản lý địa điểm bán' link is highlighted in blue. The main content area features a 'Thêm địa điểm bán' button at the top left. Below it is a table with four columns: Tên đặc sản, Tên nơi bán, Địa chỉ, and Hành động. The table lists six products with their respective selling locations and actions (edit and delete icons). At the bottom right of the table is a pagination control showing '1' and '2'.

Tên đặc sản	Tên nơi bán	Địa chỉ	Hành động
BÁNH PÍA KIM SA ĐẬU XANH	Tạp Hoá cô Ba	ấp Phụng Hiệp, xã An Hiệp, huyện Châu Thành, tỉnh Sóc Trăng	
DỪA SẤP	Vườn dừa chú Ba	Phan Chí Thanh, thành phố Trà Vinh, Trà Vinh	
DỪA SẤP	Vườn dừa Tiểu Cần	Xã Phú Cần, Huyện Tiểu Cần, Tỉnh Trà Vinh	
DỪA HẦU LONG TRÌ	Sạp trái cây LTA	xã Long Trì, huyện Châu Thành, Tỉnh Long An	
BÁNH TÉT LÁ CẨM	Bánh tét bà Huỳnh Thị Trọng	Số 56 Thái Thị Nhạn, An Thới, Bình Thủy, Cần Thơ	
BÚN NƯỚC LÈO	Bún nước lèo Kim Tuyến	Đường 19/5 nổi dài, phường 1, TP. Trà Vinh, tỉnh Trà Vinh	

Hình 4.36. Giao diện quản lý địa điểm bán

Khi người quản trị chọn chức năng thêm địa điểm bán, giao diện sẽ chuyển sang trang nhập thông tin chi tiết. Tại đây, người quản trị cần lựa chọn đặc sản cụ thể, sau đó điền đầy đủ thông tin về địa điểm bán cho đặc sản đó. Sau khi nhập liệu xong, người quản trị nhấn nút "Thêm" để lưu lại. Nơi bán của đặc sản được chọn sẽ được thêm vào với tất cả thông tin đã được nhập.



The screenshot shows the 'TNB' form for adding a new selling location. The left sidebar is identical to the previous screenshot, with 'Quản lý địa điểm bán' highlighted. The main form area contains three input fields: 'Đặc sản' (with a dropdown arrow), 'Tên nơi bán', and 'Địa chỉ'. Each field has a red asterisk indicating it is required. At the bottom of the form is a blue button labeled 'Thêm mới'.

Hình 4.37. Giao diện chức năng thêm địa điểm bán

Khi người quản trị chọn chức năng sửa địa điểm bán của đặc sản, giao diện sẽ chuyển sang trang nhập liệu với các thông tin địa điểm bán có sẵn của đặc sản. Người quản trị có thể chỉnh sửa thông tin địa điểm bán của đặc sản. Cuối cùng chọn “Lưu” để lưu lại

TNB

- Trang chủ
- Quản lý đặc sản
- Quản lý loại đặc sản
- Quản lý nhà sản xuất
- Quản lý hạn sử dụng
- Quản lý địa điểm bán**

* Tên nơi bán

Tập Hoà cổ Ba

Địa chỉ

ấp Phụng Hiệp, xã An Hiệp, huyện Châu Thành, tỉnh Sóc Trăng

Cập nhật

Hình 4.38. Giao diện chức năng sửa địa điểm bán

Người quản trị có thể xóa địa điểm của đặc sản bằng cách chọn biểu tượng xóa. Sau đó, một hộp thoại xác nhận sẽ xuất hiện để hỏi xem có chắc chắn muốn xóa không. Nếu xác nhận, địa điểm bán của đặc sản đó sẽ được xóa khỏi hệ thống.

TNB

- Trang chủ
- Quản lý đặc sản
- Quản lý loại đặc sản
- Quản lý nhà sản xuất
- Quản lý hạn sử dụng
- Quản lý địa điểm bán**

Thêm địa điểm bán

Tên đặc sản	Tên nơi bán	Địa chỉ		
BÁNH PÍA KIM SA ĐẦU XANH	Tập Hoà cổ Ba	ấp Phụng Hiệp, xã An Hiệp, huyện Châu Thành, tỉnh Sóc Trăng		
DỪA SÁP	Vườn dừa chú Ba	Phan Chí Thanh, thành phố Trà Vinh, Trà Vinh		
DỪA SÁP	Vườn dừa Tiểu Cần	Xã Phú Cần, Huyện Tiểu Cần, Tỉnh Trà Vinh		
DỪA HẦU LONG TRÌ	Sạp trái cây LTA	xã Long Trĩ, huyện Châu Thành, Tỉnh Long An		
BÁNH TÉT LÁ CẨM	Bánh tét bà Huỳnh Thị Trọng	Số 56 Thái Thị Nhạn, An Thới, Bình Thủy, Cần Thơ		
BÚN NƯỚC LÈO	Bún nước lèo Kim Tuyến	Đường 19/5 nổi dài, phường 1, TP. Trà Vinh, tỉnh Trà Vinh		

Bạn có chắc chắn muốn xóa địa điểm bán này không?

Không Có

< 1 2 >

Hình 4.39. Chức năng xóa địa điểm bán

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

- Về mặt kiến thức: Tôi mong muốn được hỗ trợ để có thêm nhiều kiến thức sâu rộng về cơ sở dữ liệu phi quan hệ, đặc biệt là MongoDB. Điều này không chỉ giúp tôi hiểu rõ cách thức hoạt động của các hệ thống cơ sở dữ liệu phi quan hệ mà còn trang bị cho tôi khả năng áp dụng MongoDB một cách hiệu quả trong các dự án thực tế. Bên cạnh đó, tôi cũng cần nắm vững các công nghệ liên quan như Node.js, RESTful API, và ReactJS. Việc hiểu rõ và sử dụng thành thạo những công nghệ này sẽ giúp tôi phát triển các ứng dụng web hiện đại, đảm bảo tính linh hoạt, khả năng mở rộng, và hiệu suất cao. Mục tiêu của tôi là có thể áp dụng những kiến thức này một cách hiệu quả để xây dựng thành công hệ thống quảng bá đặc sản Tây Nam Bộ, từ đó mang lại trải nghiệm tốt nhất cho người dùng và giúp nâng cao giá trị của các sản phẩm đặc sản. Việc này không chỉ đóng góp vào việc bảo tồn và phát triển văn hóa ẩm thực địa phương mà còn mở ra cơ hội kinh doanh mới cho các sản phẩm đặc sản.

- Về mặt kỹ năng: Tôi hy vọng sẽ được giúp đỡ để cải thiện nhiều kỹ năng quan trọng. Trước tiên là kỹ năng tự học, giúp tôi có thể tiếp thu kiến thức mới một cách nhanh chóng và hiệu quả. Khả năng tự học sẽ cho phép tôi luôn cập nhật và nắm bắt được những xu hướng và công nghệ mới nhất trong lĩnh vực công nghệ thông tin. Tiếp theo, tôi muốn nâng cao khả năng đọc và hiểu các tài liệu chuyên ngành bằng tiếng Anh, điều này rất quan trọng để tôi có thể cập nhật và áp dụng các công nghệ mới. Việc nắm vững tiếng Anh chuyên ngành sẽ giúp tôi tiếp cận được nguồn tài liệu phong phú và chất lượng, từ đó nâng cao kiến thức và kỹ năng của bản thân. Ngoài ra, tôi cần phát triển kỹ năng quản lý và sắp xếp thời gian để có thể hoàn thành công việc một cách hiệu quả nhất. Quản lý thời gian tốt sẽ giúp tôi đảm bảo tiến độ công việc và đạt được mục tiêu đề ra. Cuối cùng, kỹ năng viết báo cáo cũng rất cần thiết để tôi có thể trình bày kết quả và tiến độ công việc một cách rõ ràng và chuyên nghiệp. Việc này không chỉ giúp tôi giao tiếp hiệu quả với đồng nghiệp và khách hàng mà còn tạo ấn tượng tốt trong các dự án và báo cáo công việc.

5.2. Hướng phát triển

Để làm cho trang web trở nên hấp dẫn và thu hút hơn, tôi sẽ tiến hành cải thiện giao diện người dùng một cách toàn diện, đảm bảo rằng mọi chi tiết đều được tối ưu hóa về mặt thẩm mỹ và tính năng. Tôi sẽ chú trọng vào việc thiết kế các phần tử giao diện sao cho đẹp

mắt, dễ nhìn, và thuận tiện cho người dùng thao tác. Bên cạnh đó, tôi sẽ phát triển và tích hợp thêm nhiều tính năng mới để trang web không chỉ phong phú về nội dung mà còn đa dạng về trải nghiệm người dùng. Những tính năng mới này sẽ được thiết kế nhằm tăng cường sự tương tác và tính tiện ích cho người dùng, giúp họ cảm thấy hứng thú và dễ dàng sử dụng trang web.

Đồng thời, tôi sẽ cập nhật và bổ sung thêm nhiều thông tin chi tiết và hữu ích về từng đặc sản. Những thông tin này sẽ bao gồm lịch sử, nguồn gốc, cách chế biến, và các công dụng của từng đặc sản, giúp người dùng có cái nhìn toàn diện và sâu sắc hơn về các sản phẩm. Điều này không chỉ giúp trang web trở nên đầy đủ và hấp dẫn hơn mà còn giúp người dùng dễ dàng tìm kiếm và tiếp cận thông tin một cách thuận tiện và nhanh chóng. Nhờ vào sự cải tiến này, tôi hy vọng trang web sẽ trở thành một nguồn thông tin tin cậy và hữu ích cho người dùng, từ đó thu hút nhiều lượt truy cập hơn và tạo dựng được uy tín trong lòng khách hàng.

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] Lâm Văn Hải (2020), Cẩm nang đặc sản miền Tây Nam Bộ, NXB Lao động xã hội.
- [5] Wikipedia, "Đồng bằng sông Cửu Long". 2024, [Online]. Từ:
https://vi.wikipedia.org/wiki/%C4%90%E1%BB%93ng_b%E1%BA%B1ng_s%C3%B4ng_C%E1%BB%ADu_Long. [Ngày truy cập: 5/6/2024]
- [6] Wikipedia, "MongoDB". 2024, [Online]. Từ:
<https://en.wikipedia.org/wiki/MongoDB>. [Ngày truy cập: 10/6/2024]
- [7] Wikipedia, "React (software) ". 2024, [Online]. Từ:
[https://en.wikipedia.org/wiki/React_\(software\)](https://en.wikipedia.org/wiki/React_(software)). [Ngày truy cập: 11/6/2024]
- [8] Wikipedia, "Node.js". 2024, [Online]. Từ:
<https://en.wikipedia.org/wiki/Node.js>. [Ngày truy cập: 10/6/2024]
- [9] AWS, "Resful API". 2024, [Online]. Từ:
<https://aws.amazon.com/vi/what-is/restful-api/>. [Ngày truy cập: 10/6/2024]