**ĐẠI HỌC ĐÀ NẴNG**

**ĐẠI HỌC BÁCH KHOA**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PBL3: DỰ ÁN LẬP TRÌNH ỨNG DỤNG**

**PBL3:**  **XÂY DỰNG HỆ THỐNG QUẢN LÝ VÀ THÔNG BÁO TÌNH TRẠNG AN TOÀN VỆ SINH THỰC PHẨM CHO THÀNH PHỐ ĐÀ NẴNG**

**GIẢNG VIÊN HƯỚNG DẪN: NGUYỄN VĂN HIỆU**

**SINH VIÊN THỰC HIỆN:**

**Tên sinh viên 1 : NGUYỄN QUÂN**

**LỚP: 20TCLC\_NHAT1 NHÓM: 21N16**

**Tên sinh viên 2 : NGUYỄN ĐỨC DŨNG**

**LỚP: 21TCLC\_NHAT1 NHÓM: 21N16**

**Đà Nẵng 05-06-2023**

**MỤC LỤC**

[**LỜI MỞ ĐẦU** 1](#_heading=h.qsh70q)

[**1.**](#_heading=h.23ckvvd) **TỔNG QUAN ĐỀ TÀI** 2

[**2.**](#_heading=h.32hioqz) **CƠ SỞ LÝ THUYẾT** 2

[**2.1.**](#_heading=h.1hmsyys) **Ý tưởng** 2

[**2.2.**](#_heading=h.1664s55) **Kỹ thuật và công nghệ** 5

[**3.**](#_heading=h.3dy6vkm) **PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG** 5

[**3.1.**](#_heading=h.3q5sasy) **Phát biểu yêu cầu** 5

[**3.2.**](#_heading=h.3jtnz0s) **Các biểu đồ thiết kế** 8

[**3.3.**](#_heading=h.3x8tuzt) **Biểu đồ cơ sở dữ liệu quan hệ** 9

[**3.4.**](#_heading=h.3u2rp3q) **Kiến trúc hệ thống** 11

[**3.4.1.**](#_heading=h.2981zbj) **Kiến trúc hệ thống** 11

[**3.4.2.**](#_heading=h.38czs75) **Components** 12

[**4.**](#_heading=h.1t3h5sf) **CHƯƠNG TRÌNH VÀ KẾT QUẢ** 14

[**4.1.**](#_heading=h.47hxl2r) **Giao diện chính** 14

[**4.2.**](#_heading=h.3ls5o66) **Kết quả thực thi** 15

[**4.3.**](#_heading=h.3qwpj7n) **Nhận xét** 18

[**1. Ưu Điểm** 18](#_heading=h.261ztfg)

[**2. Nhược điểm** 18](#_heading=h.2jh5peh)

[**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 19](#_heading=h.jlnnj2cfep5a)

[**PHỤ LỤC** 22](#_heading=h.1c1lvlb)

**DANH MỤC HÌNH**

Hình 1: Biểu đồ đăng nhập 8

Hình 2: Biểu đồ admin 8

Hình 3: Biểu đồ thanh tra 8

Hình 4: Cơ sở dữ liệu 9

Hình 5: Bảng kế hoạch và thanh tra 9

Hình 6: Bảng người dùng 9

Hình 7: Bảng thông báo 10

Hình 8: Bảng cửa hàng 10

Hình 9: Bảng chứng nhận 10

Hình 10: Bảng chi cục 10

Hình 11: Bảng phân quyền quản trị 11

Hình 12: Bảng phân quyền 11

Hình 13: Chi tiết chứng nhận 11

Hình 14: Trang chủ 14

Hình 15: Trang chủ 14

Hình 16: Đăng nhập 15

Hình 17: Đăng xuất tài khoản 15

Hình 18: Quản lý tài khoản 16

Hình 19: Quản lý cửa hàng 16

Hình 20: Các loại chứng nhận an toàn vệ sinh thực phẩm 16

Hình 21: Chỉnh sửa và bổ sung chứng nhận 17

Hình 22: Chỉnh sửa và bổ sung chứng nhận 17

Hình 22: Chỉnh sửa và bổ sung chứng nhận

Hình 23: Quản lý kết quả thanh tra các cửa hàng thực phẩm trên địa bàn 18

Hình 24: Gửi thông báo cho thanh tra

Hình 25: Gửi thông báo cho thanh tra

Hình 26: Lập báo cáo

Hình 27: Nhận thông báo từ người dân

Hình 28: Đóng góp ý kiến

**LỜI MỞ ĐẦU**

Công nghệ thông tin là một trong những ngành phát triển vượt bậc trong những năm gần đây. Ngày nay với sự phát triển nhanh chóng của xã hội thì công nghệ thông tin được ứng dụng rộng rãi ở hầu hết tất cả các lĩnh vực và ngày càng đóng vai trò quan trọng, trở thành một phần thiết yếu trong đời sống hằng ngày. Công nghệ thông tin là một ngành đòi hỏi ở người học một nền tảng kiến thức vững chắc, sự tư duy logic cao, hiểu biết sâu rộng trên nhiều lĩnh vực. Với chúng em hiện đang là những sinh viên công nghệ thông tin cần phải có sự đầu tư, không ngừng học hỏi để nâng cao kiến thức. Do đó để củng cố lại kiến thức đã học, đề tài mà em chọn để thực hiện đồ án là: **Xây dựng hệ thống quản lý và thông báo tình trạng an toàn vệ sinh thực phẩm cho thành phố Đà Nẵng**

**Hệ thống quản lý và thông báo tình trạng an toàn vệ sinh thực phẩm cho thành phố Đà Nẵng** là một đề tài khá mới với các bài toán quản lý. Việc đưa tin học vào ứng dụng để quản lý là rất hữu ích, vì chúng ta cần bỏ ra rất ít thời gian mà lại thu được hiệu quả cao, chính xác, tiện lợi và nhanh chóng.

Trong qua quá trình thực hiện đồ án, nhóm chúng em đã vận dụng các kiến thức đã học và cả những kiến thức thông qua tìm hiểu trên mạng để thực hiện đề tài. Từ đó, chúng em đã rút ra được các kỹ thuật quan trọng sao cho tối ưu nhất đối với một dự án web cơ bản như “Quản lý và thông báo tình trạng an toàn vệ sinh thực phẩm”. Chúng em xin chân thành cảm ơn thầy Nguyễn Văn Hiệu đã hổ trợ và giúp đỡ tận tình nhóm chúng em thực hiện đồ án này.

Trong quá trình thực hiện, không tránh khỏi sai sót. Chúng em rất mong nhận được sự góp ý từ phía thầy, cô để bài làm của chúng em được hoàn thiện hơn.

Đà Nẵng, ngày 05 tháng 6 năm 2023

1. **TỔNG QUAN ĐỀ TÀI**

Quản lý và thông báo tình trạng an toàn vệ sinh thực phẩm là công việc phức tạp. Công việc quản lý đòi hỏi tính tỉ mỉ, cẩn thận, rõ ràng và chính xác. Trước đây công nghệ thông tin chưa phát triển mạnh mẽ, các công việc được xử lý thủ công, chủ yếu là ghi chép bằng bút, sổ sách chính vì vậy rất tốn công sức, mất khá nhiều thời gian và có thể gây thiếu sót. Ngày nay khi mà khoa học kỹ thuật phát triển, đặc biệt là sự bùng nổ công nghệ thông tin thì việc quản lý sẽ trở nên dễ dàng hơn . Xuất phát từ nhu cầu đó mà nhóm chúng em đã quyết định thực hiện đề tài **Quản lý và thông báo tình trạng an toàn vệ sinh của thành phố**. Yêu cầu của đồ án là tạo ra sản phẩm thiện có giao diện tương tác với người dùng và sản phẩm có thể kết nối CSDL. Có các đầy đủ các chức năng cơ bản của một trang web quản lý hệ thống. Chương trình được viết bằng ngôn ngữ java, html và css. Với chương trình “**quản lý và thông báo tình trạng an toàn vệ sinh của thành phố**” này chúng em hy vọng sẽ giúp cho công tác quản lý của các nhân viên sẽ diễn ra đơn giản và tiết kiệm thời gian hơn.

1. **CƠ SỞ LÝ THUYẾT**
   1. **Ý tưởng**
2. **Chức năng đăng nhập:** Hệ thống cho phép người dùng sử dụng tài khoản và mật khẩu để đăng nhập vào hệ thống
3. **Chức năng đăng xuất:** Hệ thống cho phép người dùng đăng xuất khỏi hệ thống
4. **Chức năng quản lý tài khoản:** Hệ thống cho phép người dùng là admin và thanh tra có thể quản lý các tài khoản hiện có

## 3.1: Hệ thống cho phép admin có thể xem tất cả các thông tin của tất cả các tài khoản hiện có

## 3.2: Hệ thống cho phép admin có thể thêm tài khoản cho thanh tra

## 3.3: Hệ thống cho phép admin có thể thêm xóa khoản của thanh tra

## 3.4: Hệ thống cho phép admin có thể thay đổi mật khẩu tài khoản của thanh tra

## 3.5: Hệ thống cho phép admin có thể thay đổi thông tin tài khoản của thanh tra

## 3.6: Hệ thống cho phép thanh tra có thể xem thông tin tài khoản của chính mình

## 3.7: Hệ thống cho phép thanh tra có thể thay đổi thông tin tài khoản của chính mình

1. **Chức năng quản lý cửa hàng sản xuất và kinh doanh thực phẩm:** Hệ thống cho phép người dùng là admin và thanh tra có thể quản lý được thông tin các cửa hàng sản xuất và kinh doanh thực phẩm

## 4.1: Hệ thống cho phép admin và thanh tra có thể xem thông tin các cửa hàng

## 4.2: Hệ thống cho phép admin và thanh tra có thể thay đổi thông tin cửa hàng

## 4.3: Hệ thống cho phép admin và thanh tra nhập các thông tin cửa hàng mới

## 4.4: Hệ thống cho phép admin và thanh tra xóa các cửa hàng đã dừng hoạt động trong thời gian dài

1. **Chức năng quản lý các loại chứng nhận an toàn vệ sinh thực phẩm:** Hệ thống cho phép người dùng là admin và thanh tra có thể quản lý các loại chứng nhận an toàn vệ sinh thực phẩm

## 5.1: Hệ thống cho phép admin và thanh tra có thể xem thông tin về các loại chứng nhận an toàn vệ sinh thực phẩm

## 5.2: Hệ thống cho phép admin và thanh tra có thể nhập thêm các loại chứng nhận an toàn thực phẩm mới theo quy định của Nhà nước

## 5.3: Hệ thống cho phép admin và thanh tra có thể sửa các thông tin về các loại chứng nhận an toàn thực phẩm theo quy định của Nhà nước

## 5.4: Hệ thống cho phép admin và thanh tra có thể xóa các loại chứng nhận an toàn thực phẩm không còn được sử dụng

1. **Chức năng quản lý thông tin việc cấp các loại chứng nhận an toàn thực phẩm cho các cửa hàng:** Hệ thống cho phép người dùng là admin và thanh tra quản lý thông tin việc cấp các loại chứng nhận an toàn thực phẩm

## 6.1: Hệ thống cho phép admin và thanh tra nhập thông tin các loại chứng nhận an toàn thực phẩm cho các cửa hàng trên hệ thống

## 6.2: Hệ thống cho phép admin và thanh tra sửa thông tin các loại chứng nhận an toàn thực phẩm đã cấp cho các cửa hàng trên hẹ thông

## 6.3: Hệ thống cho phép admin và thanh tra xóa thông tin các loại chứng nhận an toàn thực phẩm đã cấp cho các cửa hàng trên hệ thống

1. **Chức năng tạo kế hoạch thanh tra các cửa hàng thực phẩm trên địa bàn:** Hệ thống cho phép người dùng là admin có thể lên kế hoạch thanh tra các cửa hàng thực phẩm để các thanh tra thực hiện đi thanh tra theo đúng kế hoạch

## 7.1: Hệ thống cho phép admin có thể tạo kế hoạch thanh tra

## 7.2: Hệ thống cho phép admin có thể chỉ định các thanh tra thuộc chi cục để thực hiện các kế hoạch thanh tra

## 7.3: Hệ thống cho phép admin có thể điều chỉnh thông tin của các kế hoạch thanh tra

1. **Chức năng quản lý kết quả thanh tra các cửa hàng thực phẩm trên địa bàn:** Hệ thống cho phép người dùng là admin và thanh tra có thể quản lý kết quả của các cuộc thanh tra

## 8.1: Hệ thống cho phép người dùng là admin và thanh tra có thể nhập mới thông tin kết quả thanh tra các cửa hàng

## 8.2: Hệ thống cho phép người dùng là admin và thanh tra có thể chỉnh sửa kết quả thanh tra các cửa hàng

1. **Chức năng gửi thông báo:** Hệ thống cho phép người dùng là admin tạo và gửi thông báo đến các thanh tra về các kế hoạch thanh tra, các công việc của chi cục
2. **Chức năng thống kê:** Hệ thống cho phép người dùng là admin và thanh tra có thể thống kê

## 10.1: Hệ thống cho phép người dùng là admin và thanh tra có thể thống kê số lượng cửa hàng thực phẩm trên địa bàn, các cửa hàng đã được cấp chứng nhận an toàn thực phẩm, cửa hàng chưa được chứng nhận an toàn thực phẩm

## 10.2: Hệ thống cho phép người dùng là admin và thanh tra có thể thống kê số lượng và mật độ các kế hoạch thanh tra các cửa hàng thực phẩm trên địa bàn

1. **Chức năng lập báo cáo:** Hệ thống cho phép người dùng là admin và thanh tra lập báo cáo tình hình vệ sinh an toàn thực phẩm trên địa bàn cho cấp trên
2. **Chức năng nhận thông báo từ người dân:** Hệ thống cho phép người dùng là admin và thanh tra nhận thông báo của người dân về tình hình vệ sinh an toàn thực phẩm trên địa bàn
3. **Chức năng đóng góp ý kiến:** Hệ thống cho phép người dùng là người dân có thể ghi lại những ý kiến phản hồi về hoạt động của chi cục

## 13.1: Hệ thống cho phép người dùng là người dân có thể nhập những ý kiến phản hồi

## 13.2: Hệ thống cho phép người dùng là người dân liên hệ để làm việc với chi cục

* 1. **Kỹ thuật và công nghệ**

Frontend: ReactJS

Backend: Java(Servlet)

Database: MySQL

File Storage: Microsoft onedrive

1. **PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**
   1. **Phát biểu yêu cầu**

+ Input:

1. Đăng nhập:

- Tên người dùng và mật khẩu.

2. Đăng xuất: Không yêu cầu đầu vào.

3. Quản lý tài khoản:

- Xem thông tin tất cả các tài khoản hiện có: Không yêu cầu đầu vào.

- Thêm tài khoản cho thanh tra:

- Thông tin tài khoản mới (tên, email, v.v.).

- Xóa tài khoản của thanh tra:

- Tên tài khoản cần xóa.

- Thay đổi mật khẩu tài khoản của thanh tra:

- Tên tài khoản và mật khẩu mới.

- Thay đổi thông tin tài khoản của thanh tra:

- Tên tài khoản và thông tin cần thay đổi.

- Xem thông tin tài khoản của thanh tra:

- Tên tài khoản.

4. Quản lý cửa hàng sản xuất và kinh doanh thực phẩm:

- Xem thông tin các cửa hàng: Không yêu cầu đầu vào.

- Thay đổi thông tin cửa hàng:

- Tên cửa hàng và thông tin cần thay đổi.

- Nhập thông tin cửa hàng mới:

- Thông tin cửa hàng mới (tên, địa chỉ, v.v.).

- Xóa cửa hàng đã dừng hoạt động:

- Tên cửa hàng.

5. Quản lý các loại chứng nhận an toàn vệ sinh thực phẩm:

- Xem thông tin về các loại chứng nhận: Không yêu cầu đầu vào.

- Nhập thêm các loại chứng nhận mới:

- Thông tin loại chứng nhận mới (tên, mô tả, v.v.).

- Sửa thông tin các loại chứng nhận:

- Tên loại chứng nhận và thông tin cần sửa đổi.

- Xóa các loại chứng nhận không còn được sử dụng:

- Tên loại chứng nhận.

6. Quản lý thông tin việc cấp chứng nhận an toàn thực phẩm cho các cửa hàng:

- Nhập thông tin chứng nhận cho cửa hàng:

- Tên cửa hàng và thông tin chứng nhận.

- Sửa thông tin chứng nhận đã cấp cho cửa hàng:

- Tên cửa hàng và thông tin cần sửa đổi.

- Xóa thông tin chứng nhận đã cấp cho cửa hàng:

- Tên cửa hàng.

7. Tạo kế hoạch thanh tra các cửa hàng:

- Tạo kế hoạch thanh tra:

- Thông tin kế hoạch (thời gian, địa điểm, v.v.).

- Chỉ định thanh tra viên:

- Tên kế hoạch và danh sách thanh tra viên.

- Điều chỉnh thông tin kế hoạch thanh tra:

- Tên kế hoạch và thông tin cần điều chỉnh.

8. Quản lý kết quả thanh tra các cửa hàng:

- Nhập thông tin kết quả thanh tra:

- Thông tin kết quả thanh tra (điểm, ghi chú, v.v.).

- Chỉnh sửa kết quả thanh tra:

- Thông tin kết quả cần chỉnh sửa.

9. Gửi thông báo:

- Tạo thông báo:

- Nội dung thông báo và danh sách người nhận.

10. Thống kê:

- Thống kê số lượng cửa hàng trên địa bàn, các cửa hàng đã được cấp chứng nhận an toàn, cửa hàng chưa được chứng nhận an toàn: Không yêu cầu đầu vào.

- Thống kê số lượng và mật độ kế hoạch thanh tra các cửa hàng trên địa bàn: Không yêu cầu đầu vào.

11. Lập báo cáo:

- Lập báo cáo tình hình vệ sinh an toàn thực phẩm trên địa bàn.

12. Nhận thông báo từ người dân: Không yêu cầu đầu vào.

13. Đóng góp ý kiến:

- Ghi lại ý kiến phản hồi.

+ Output:

1. Kết quả đăng nhập: Xác nhận đăng nhập thành công hoặc thông báo lỗi đăng nhập.

2. Kết quả đăng xuất: Xác nhận đăng xuất thành công.

3. Kết quả quản lý tài khoản: Thông tin tài khoản đã được thay đổi hoặc xem.

4. Kết quả quản lý cửa hàng sản xuất và kinh doanh thực phẩm: Thông tin cửa hàng đã được thay đổi hoặc xem.

5. Kết quả quản lý các loại chứng nhận an toàn vệ sinh thực phẩm: Thông tin chứng nhận đã được thay đổi hoặc xem.

6. Kết quả quản lý thông tin việc cấp chứng nhận an toàn thực phẩm cho các cửa hàng: Thông tin chứng nhận đã được thay đổi hoặc xóa.

7. Kết quả tạo kế hoạch thanh tra các cửa hàng: Kế hoạch thanh tra đã được tạo.

8. Kết quả quản lý kết quả thanh tra các cửa hàng: Kết quả thanh tra đã được nhập hoặc chỉnh sửa.

9. Kết quả gửi thông báo: Thông báo đã được gửi thành công.

10. Kết quả thống kê: Thống kê số liệu.

11. Kết quả lập báo cáo: Báo cáo đã được tạo.

12. Kết quả nhận thông báo từ người dân: Thông báo từ người dân đã được nhận.

13. Kết quả đóng góp ý kiến: Ý kiến đã được ghi lại.

* 1. **Các biểu đồ thiết kế**

A picture containing text, screenshot, font, line

Description automatically generated

**Hình 1: Biểu đồ đăng nhập**

A picture containing text, screenshot, line, rectangle

Description automatically generated

**Hình 2: Biểu đồ admin**

A picture containing text, screenshot, rectangle, line

Description automatically generated

**Hình 3: Biểu đồ thanh tra**

* 1. **Biểu đồ cơ sở dữ liệu quan hệ**

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Phông chữ

Mô tả được tạo tự động

**Hình 4: Cơ sở dữ liệu**

Ảnh có chứa văn bản, số, ảnh chụp màn hình, hàng

Mô tả được tạo tự động

**Hình 5: Bảng kế hoạch và thanh tra**

Ảnh có chứa văn bản, hàng, số, ảnh chụp màn hình

Mô tả được tạo tự động

**Hình 6: Bảng người dùng**

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, hàng

Mô tả được tạo tự động

**Hình 7: Bảng thông báo**

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động

**Hình 8: Bảng cửa hàng**

Ảnh có chứa văn bản, ảnh chụp màn hình, hàng, số

Mô tả được tạo tự động

**Hình 9: Bảng chứng nhận**

Ảnh có chứa văn bản, hàng, số, Phông chữ

Mô tả được tạo tự động

**Hình 10: Bảng chi cục**

Ảnh có chứa văn bản, hàng, Phông chữ, số

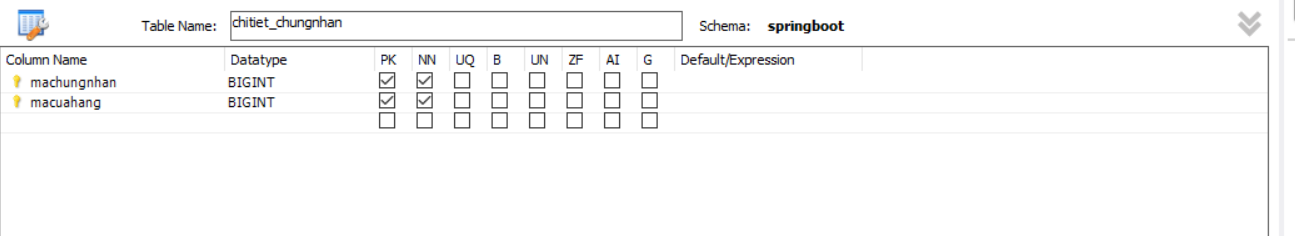
Mô tả được tạo tự động

**Hình 11: Bảng phân quền quản trị**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

**Hình 12: Bảng phân quyền**



**Hình 13: Chi tiết chứng nhận**

* 1. **Kiến trúc hệ thống**
     1. **Kiến trúc hệ thống**

Client-Side:

- Người dùng (Admin, Thanh tra, Người dân): Đại diện cho các vai trò khác nhau trong hệ thống.

- Giao diện người dùng: Cung cấp giao diện tương tác cho người dùng để thao tác với hệ thống.

- Server-Side:

- Web Server: Đảm nhận việc xử lý các yêu cầu từ phía người dùng và gửi lại kết quả tương ứng.

- Application Server: Chứa logic xử lý nghiệp vụ của hệ thống.

- Database Server: Lưu trữ và quản lý cơ sở dữ liệu của hệ thống.

* + 1. **Components**

Hệ thống xác thực và phân quyền: Đảm bảo chỉ người dùng hợp lệ và có quyền truy cập vào hệ thống.

- Quản lý tài khoản:

- Xem tất cả thông tin tài khoản hiện có (Admin).

- Thêm tài khoản thanh tra (Admin).

- Xóa tài khoản thanh tra (Admin).

- Thay đổi mật khẩu của tài khoản thanh tra (Admin).

- Thay đổi thông tin tài khoản thanh tra (Admin).

- Xem thông tin tài khoản của chính mình (Thanh tra).

- Thay đổi thông tin tài khoản của chính mình (Thanh tra).

- Quản lý cửa hàng sản xuất và kinh doanh thực phẩm:

- Xem thông tin các cửa hàng (Admin, Thanh tra).

- Thay đổi thông tin cửa hàng (Admin, Thanh tra).

- Thêm mới cửa hàng (Admin, Thanh tra).

- Xóa cửa hàng không hoạt động (Admin, Thanh tra).

- Quản lý chứng nhận an toàn vệ sinh thực phẩm:

- Xem thông tin về các loại chứng nhận (Admin, Thanh tra).

- Thêm mới loại chứng nhận (Admin, Thanh tra).

- Sửa thông tin các loại chứng nhận (Admin, Thanh tra).

- Xóa các loại chứng nhận không còn sử dụng (Admin, Thanh tra).

- Quản lý việc cấp chứng nhận an toàn vệ sinh thực phẩm cho các cửa hàng:

- Nhập thông tin chứng nhận cho các cửa hàng (Admin, Thanh tra).

- Sửa thông tin chứng nhận đã cấp cho các cửa hàng (Admin, Thanh tra).

- Xóa thông tin chứng nhận đã cấp cho các cửa hàng (Admin, Thanh tra).

- Tạo kế hoạch thanh tra các cửa hàng thực phẩm:

- Tạo kế hoạch thanh tra (Admin).

- Chỉ định thanh tra cho kế hoạch (Admin).

- Điều chỉnh thông tin kế hoạch thanh tra (Admin).

- Quản lý kết quả thanh tra các cửa hàng thực phẩm:

- Nhập thông tin kết quả thanh tra các cửa hàng (Admin, Thanh tra).

- Chỉnh sửa thông tin kết quả thanh tra các cửa hàng (Admin, Thanh tra).

- Gửi thông báo:

- Tạo và gửi thông báo cho thanh tra về kế hoạch thanh tra, công việc của chi cục (Admin).

- Thống kê:

- Thống kê số lượng cửa hàng trên địa bàn, cửa hàng đã được cấp chứng nhận, cửa hàng chưa được chứng nhận (Admin, Thanh tra).

- Thống kê số lượng và mật độ kế hoạch thanh tra các cửa hàng trên địa bàn (Admin, Thanh tra).

- Lập báo cáo:

- Lập báo cáo tình hình vệ sinh an toàn thực phẩm trên địa bàn cho cấp trên (Admin, Thanh tra).

- Nhận thông báo từ người dân:

- Nhận thông báo của người dân về tình hình vệ sinh an toàn thực phẩm trên địa bàn (Admin, Thanh tra).

- Đóng góp ý kiến:

- Ghi lại ý kiến phản hồi về hoạt động của chi cục (Người dân).

1. **CHƯƠNG TRÌNH VÀ KẾT QUẢ**
   1. **Giao diện chính**

Ảnh có chứa văn bản, Thức ăn tự nhiên, sản vật, Nhóm thực phẩm

Mô tả được tạo tự động

**Hình 14: Trang chủ**

Ảnh có chứa văn bản, ảnh chụp màn hình, thiết kế

Mô tả được tạo tự động

**Hình 15: Trang chủ**

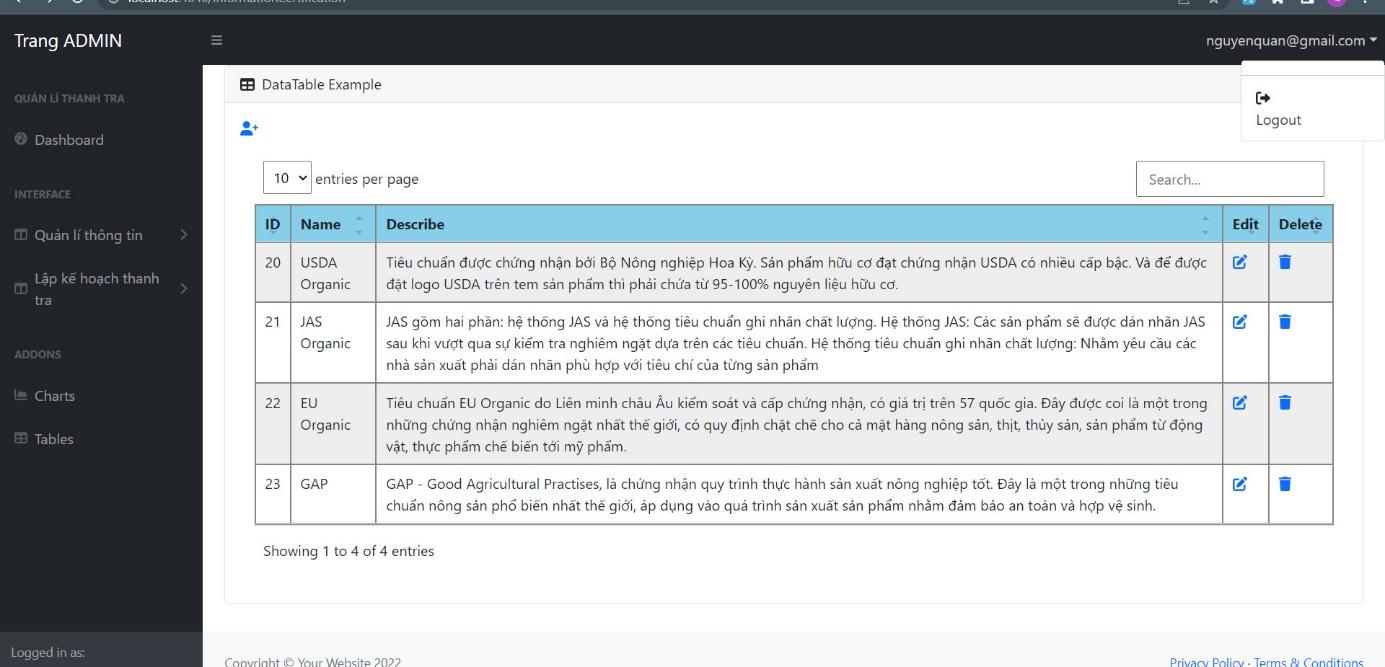
* 1. **Kết quả thực thi**
  2. **Đăng nhập**

Ảnh có chứa văn bản, ảnh chụp màn hình, thiết kế, Phông chữ

Mô tả được tạo tự động

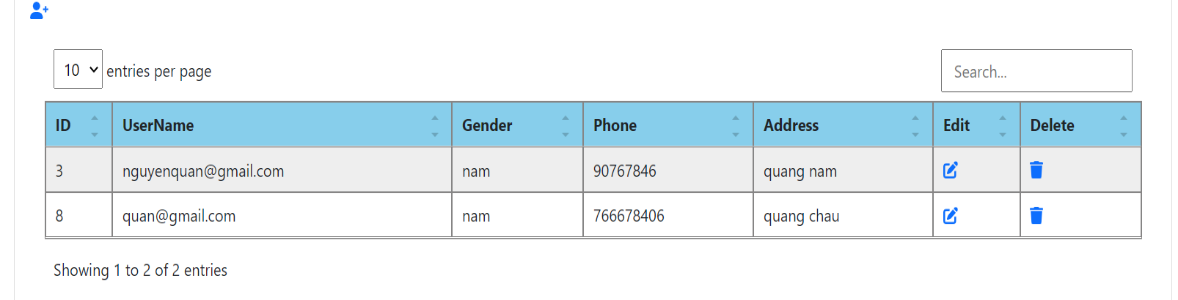
**Hình 16: Đăng nhập**

* 1. **Đăng xuất**



**Hình 17: Đăng xuất tài khoản**

* 1. **Quản lý tài khoản**



**Hình 18: Quản lý tài khoản**

Ở đây admin có thể thêm, sửa, xóa thông tin của người dùng

* 1. **Quản lý cửa hàng sản xuất và kinh doanh thực phẩm**

Ảnh có chứa văn bản, hàng, Phông chữ, số

Mô tả được tạo tự động

**Hình 19: Quản lý cửa hàng**

Chức năng này cho phép người dùng xem cửa hàng nào đã dừng hoạt động và còn hoạt động

* 1. **Quản lý chứng nhận an toàn vệ sinh thực phẩm**

Ảnh có chứa văn bản, thực đơn, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

**Hình 20: Các loại chứng nhận an toàn vệ sinh thực phẩm**

Chức năng này cho phép người dùng xem các chứng nhận hiện có và còn hiệu lực trên thị trường

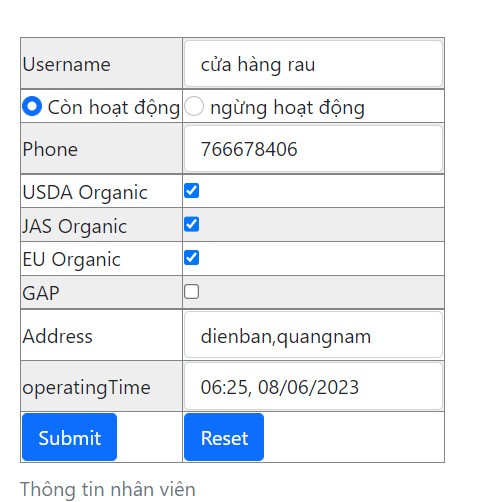
* 1. **Quản lý thông tin việc cấp các loại chứng nhận an toàn thực phẩm cho các cửa hàng**

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tạo tự động

**Hình 21: Chỉnh sửa và bổ sung chứng nhận**

Chức năng cho phép admin quản lý các chứng nhận đang có trên thị trường và cấp cho các cửa hàng



**Hình 21.1: Chứng nhận mà cửa hàng có thể nhìn thấy**

* 1. **Tạo kế hoạch thanh tra các cửa hàng thực phẩm trên địa bàn**

Ảnh có chứa văn bản, ảnh chụp màn hình, số, Phông chữ

Mô tả được tạo tự động

**Hình 22: Tạo kế hoạch thanh tra các cửa hàng thực phẩm trên địa bàn**

Chức năng này cho phép thanh tra xem kế hoạch mà admin đã tạo cho thanh tra

* 1. **Quản lý kết quả thanh tra các cửa hàng thực phẩm trên địa bàn**

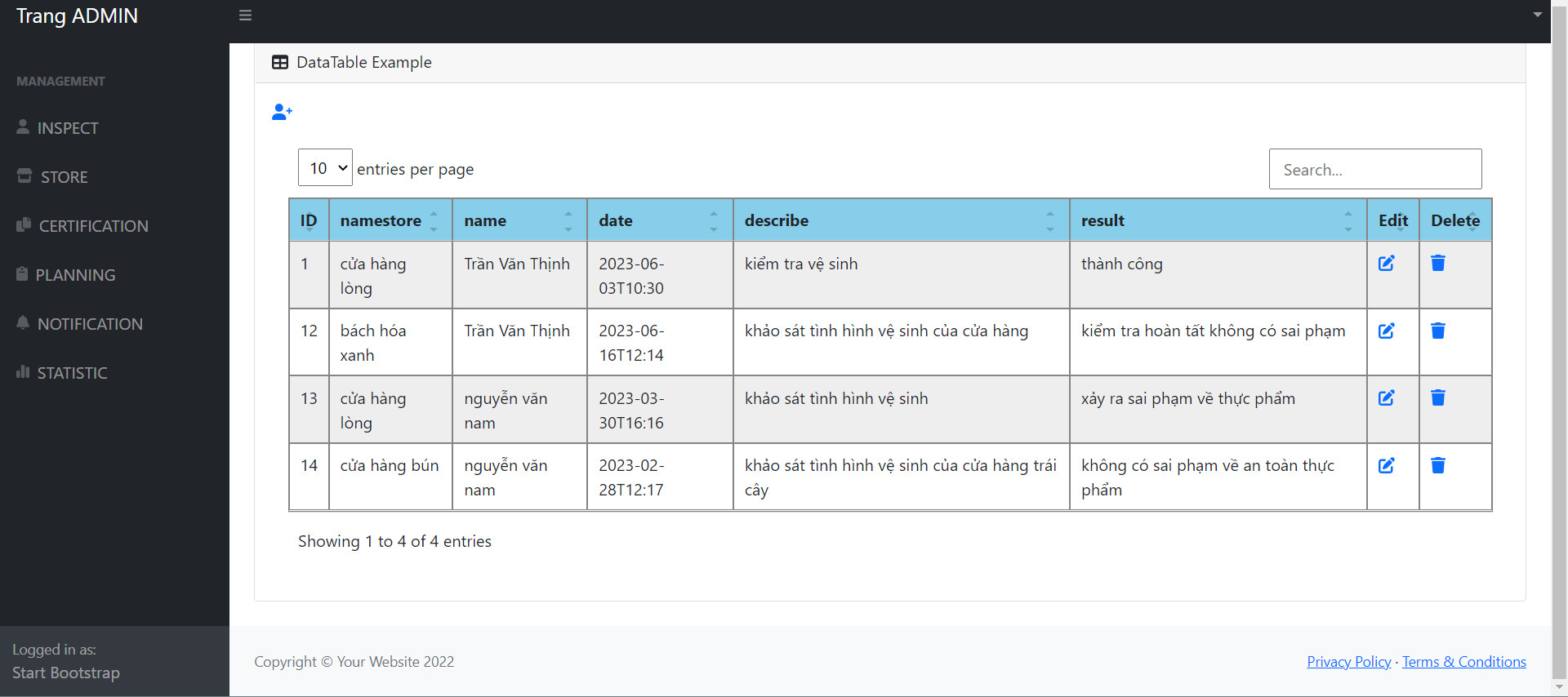
Ảnh có chứa văn bản, số, Phông chữ, hàng

Mô tả được tạo tự động

**Hình 23: Quản lý kết quả thanh tra các cửa hàng thực phẩm trên địa bàn**

Ở chức năng này admin có thể nhìn thấy kết quả thanh tra của các cửa hàng

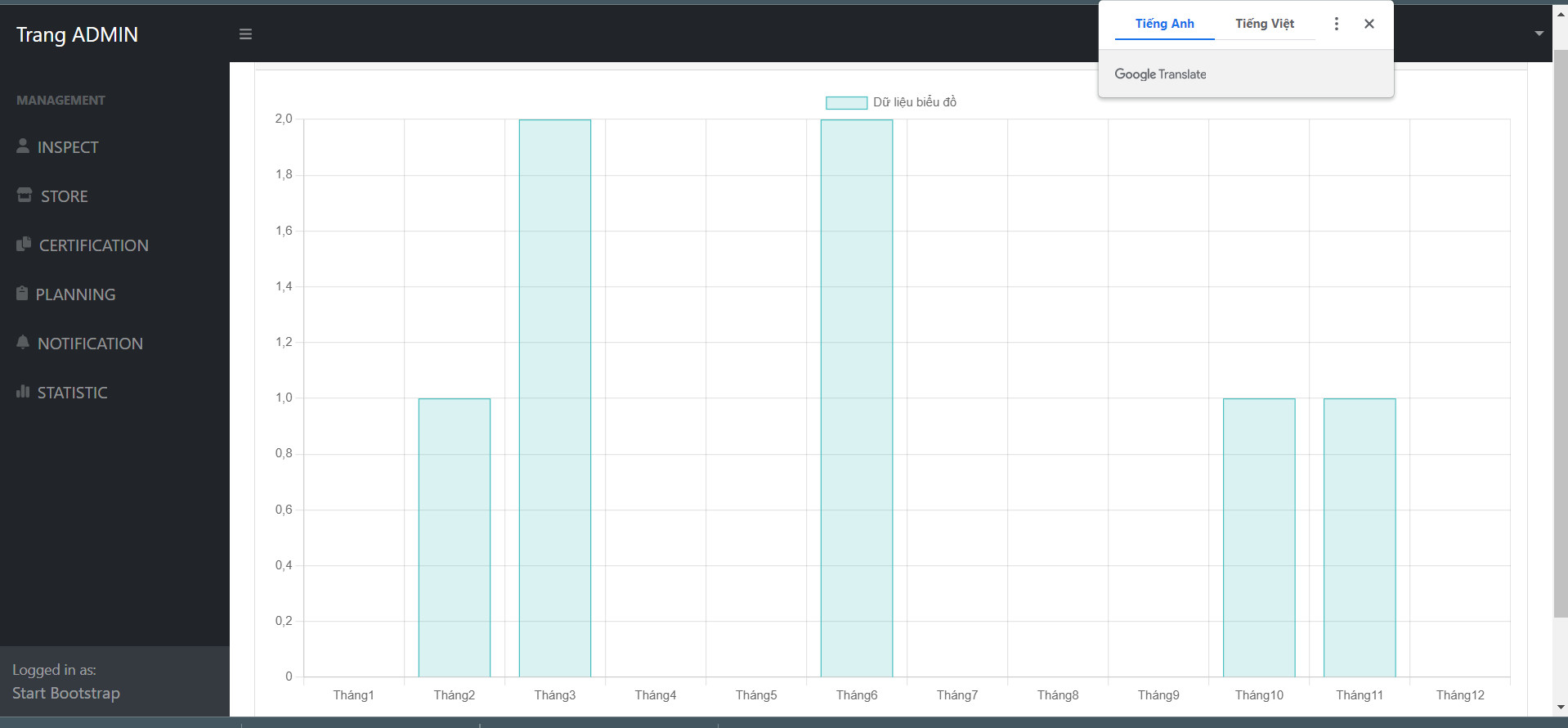
* 1. **Gửi thông báo cho thanh tra**



**Hình 24: Gửi thông báo cho thanh tra**

Ở chức năng này admin có thể gửi thông báo cho các thanh tra

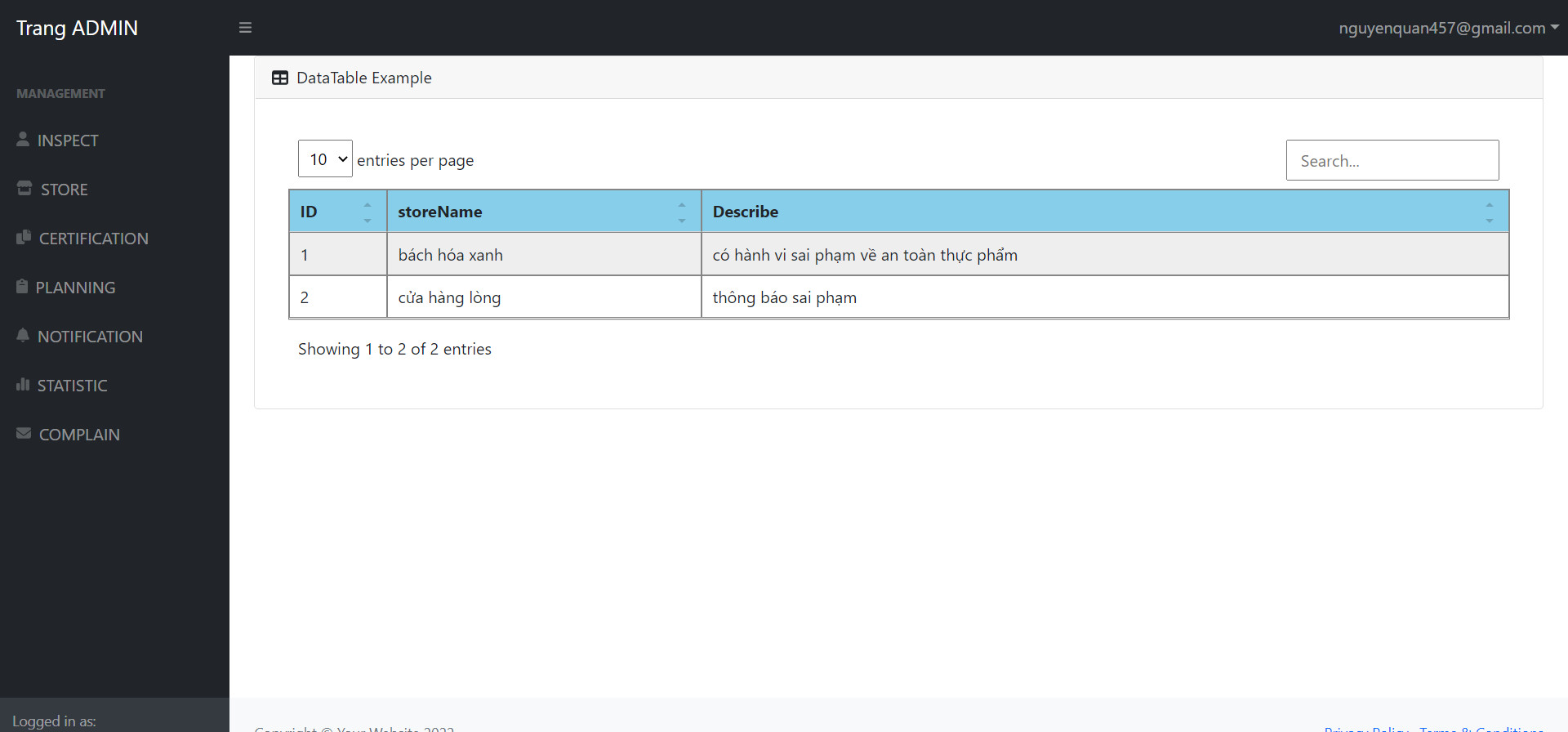
* 1. **Thống kê**

****

**Hình 25: Gửi thông báo cho thanh tra**

Ở chức năng này admin và thanh tra có thể thống kê

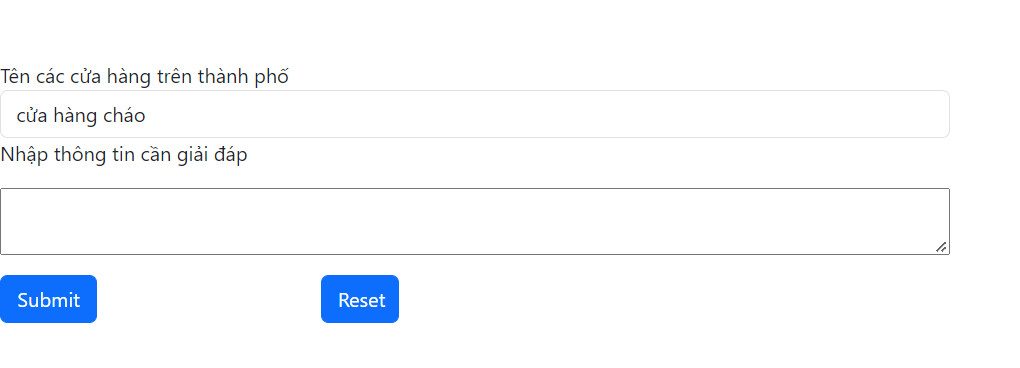
* 1. **Lập báo cáo**

****

**Hình 26: Lập báo cáo**

Ở chức năng này admin và thanh tra nhận thông báo của người dân về tình hình vệ sinh an toàn thực phẩm trên địa bàn

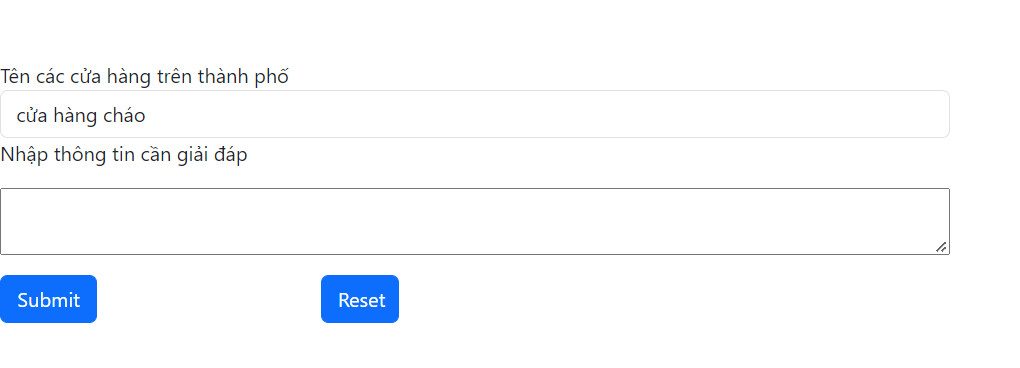
* 1. **Nhận thông báo từ người dân**

****

**Hình 27: Nhận thông báo từ người dân**

Ở chức năng này admin và thanh tra nhận thông báo của người dân về tình hình vệ sinh an toàn thực phẩm trên địa bàn

* 1. **Đóng góp ý kiến**

****

**Hình 28: Đóng góp ý kiến**

Ở chức năng này người dùng là người dân có thể ghi lại những ý kiến phản hồi về hoạt động của chi cục

* 1. ***Nhận xét***

***1. Ưu Điểm***

***+*** Giao diện rõ ràng.

+ Có đầy đủ các chức năng cơ bản mà một trang web yêu cầu.

+ Xử lý được một số ngoại lệ thường gặp.

+ Đơn giản và dễ sử dụng.

***2. Nhược điểm***

***+*** Các bước chuyển qua lại các chức năng còn phức tạp và có độ trễ.

+ Thiết kế giao diện còn sơ sài.

+ Còn nhiều lỗi nhỏ chưa xử lý được.

**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

1. **Kết luận**

Trong thời gian thực hiện đồ án PBL: Dự án lập trình ứng dụng, nhóm em đã tìm hiểu và học được phương pháp lập trình ra một trang web cơ bản về quản lý.

1. **Hướng phát triển**

1. Giao diện người dùng đa nền tảng: Phát triển giao diện người dùng thân thiện, dễ sử dụng và đáp ứng trên nhiều nền tảng, bao gồm ứng dụng di động và trang web. Điều này cho phép người dùng dễ dàng truy cập và sử dụng hệ thống từ mọi thiết bị.

2. Tích hợp hệ thống thông báo: Xây dựng chức năng thông báo để cung cấp thông tin liên quan đến tình trạng an toàn vệ sinh thực phẩm. Hệ thống có thể gửi thông báo tự động đến người dùng, bao gồm cả thanh tra và người dân, về các kế hoạch thanh tra, thông tin cửa hàng, thay đổi quy định, và các thông báo khẩn cấp khác.

3. Hệ thống báo cáo và thống kê: Cung cấp chức năng lập báo cáo và thống kê tình hình vệ sinh an toàn thực phẩm trên địa bàn thành phố. Hệ thống sẽ tổng hợp dữ liệu từ các cuộc thanh tra, thông báo từ người dân, và thông tin về cửa hàng để tạo ra báo cáo chi tiết và thống kê có ích cho việc đánh giá và quản lý.

4. Tích hợp công nghệ IoT (Internet of Things): Kết hợp với công nghệ IoT, hệ thống có thể theo dõi và ghi nhận các thông số về vệ sinh an toàn thực phẩm như nhiệt độ lưu trữ, hạn sử dụng, và các thông số quan trọng khác. Dữ liệu này sẽ được gửi về hệ thống và sử dụng để đánh giá và quản lý tình trạng an toàn vệ sinh của cửa hàng.

5. Mở rộng tích hợp với cơ sở dữ liệu và hệ thống khác: Để nâng cao tính linh hoạt và hiệu quả, hệ thống có thể tích hợp với các cơ sở dữ liệu và hệ thống khác, bao gồm hệ thống quản lý khu vực, hệ thống y tế, hệ thống giám sát môi trường, để chia sẻ thông tin và tối ưu hóa quá trình quản lý.

6. Nâng cao tính bảo mật:

Đảm bảo tính bảo mật cho hệ thống là rất quan trọng. Áp dụng các biện pháp bảo mật như mã hóa dữ liệu, xác thực người dùng, và kiểm soát quyền truy cập để bảo vệ thông tin nhạy cảm và ngăn chặn các cuộc tấn công mạng.

7. Đánh giá và cải tiến liên tục: Liên tục đánh giá hiệu suất và phản hồi từ người dùng để cải tiến hệ thống theo thời gian. Sự phản hồi này có thể thu thập thông qua khảo sát, đánh giá và ghi nhận ý kiến từ người dùng, và áp dụng các cải tiến để tối ưu hóa trải nghiệm người dùng và hiệu quả của hệ thống.

Tổng thể, hướng phát triển của đề tài này nhằm xây dựng một hệ thống Quản lý và Thông báo Tình trạng An toàn Vệ sinh linh hoạt, tiện lợi và hiệu quả để quản lý và cung cấp thông tin về vệ sinh an toàn thực phẩm trong một thành phố.

**TÀI LIỆU THAM KHẢO**

**[1]** Mai Văn Hà, tài liệu môn lập trình Java, đại học bách khoa Đà Nẵng

**[2]** Trương Ngọc Châu, Giáo trình môn Cơ sở dữ liệu, đại học bách khoa Đà Nẵng

**[3]** Java Programming Basics - GeeksforGeeks

**[4]** Learn Servlet Tutorial - javatpoint

**[5]** Thông tin tìm kiếm trên các trang web trên google và các video trên youtube

**[6]** Tìm hiểu mọi thứ với chat GPT.

<https://chat.openai.com/>

**PHỤ LỤC**

Link github: [nguyenquan23/quanlivesinh (github.com)](https://github.com/nguyenquan23/quanlivesinh/tree/foodsafety)

package com.example.quanlivesinh.controller;  
  
import com.example.quanlivesinh.persistence.Certification;  
import com.example.quanlivesinh.persistence.Inspect;  
import com.example.quanlivesinh.service.CertificationService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.util.StringUtils;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.multipart.MultipartFile;  
  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.nio.file.StandardCopyOption;  
  
@Controller  
public class CertificationController {  
 private CertificationService certificationService;  
 @Autowired  
 public void setInspectService(CertificationService certificationService) {  
 this.certificationService = certificationService;  
 }  
 @RequestMapping(value = "/informationCertification")  
 public String getCertification(Model model){  
 model.addAttribute("listCertification",certificationService.getCertificationAll());  
 return "html/home";  
 }  
 @RequestMapping("/certification/edit/{id}")  
  
 public String editCertification(@PathVariable Long id, Model model){  
 System.*out*.println(certificationService.getOneCertification(id));  
 model.addAttribute("certification", certificationService.getOneCertification(id) );  
 return "html/home";  
 }  
 @RequestMapping("/certification/delete/{id}")  
 public String deleteInspect(@PathVariable Long id){  
 certificationService.deleteInspect(id);  
 return "redirect:/informationCertification";  
  
 }  
 @RequestMapping("/add/certification")  
 public String addCertification(Model model){  
 model.addAttribute("certificationNew",new Certification());  
 return "html/home";  
  
 }  
 @PostMapping("add/NewCertification")  
 public String saveCertification(@ModelAttribute("certificationNew") Certification certification , @RequestParam("hinhAnh") MultipartFile hinhAnh) throws IOException {  
 String fileName = StringUtils.*cleanPath*(hinhAnh.getOriginalFilename());  
 Path path = Paths.*get*("src/main/resources/static/images/certification/" + fileName);  
 Files.*copy*(hinhAnh.getInputStream(), path, StandardCopyOption.*REPLACE\_EXISTING*);  
 Certification certificationNew = new Certification();  
 certificationNew.setName(certification.getName());  
 certificationNew.setDescribe(certification.getDescribe());  
 certificationNew.setImage("/images/certification/" + fileName);  
 certificationService.save(certificationNew);  
 return "redirect:/informationCertification";  
 }  
  
 @RequestMapping(value = "/getInformationCertification")  
 public String getAllCertification(Model model){  
 model.addAttribute("listCertification",certificationService.getCertificationAll());  
 return "html/home-page";  
 }  
  
 @PostMapping("/certification")  
 public String Update(Certification certification, Model model){  
 certificationService.updateStudent(certification.getId(),certification.getName(),certification.getDescribe());  
 model.addAttribute("listCertification",certificationService.getCertificationAll());  
 return "redirect:/informationCertification";  
 }  
  
 }

package com.example.quanlivesinh.controller;  
  
  
import com.example.quanlivesinh.persistence.Inspect;  
import com.example.quanlivesinh.persistence.Role;  
  
import com.example.quanlivesinh.service.InspectService;  
import com.example.quanlivesinh.service.RoleService;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
  
@Controller  
public class InspectController {  
 private InspectService inspectService;  
 private RoleService roleService;  
 private final PasswordEncoder passwordEncoder = new BCryptPasswordEncoder();  
  
  
 @Autowired  
 public void setInspectService(InspectService inspectService) {  
 this.inspectService = inspectService;  
 }  
 @Autowired  
 public void setRoleService(RoleService roleService){this.roleService = roleService;}  
  
 @RequestMapping(value = "/informationInspect")  
 public String getInspect(Model model){  
  
 System.*out*.println(inspectService.listAllInspect());  
  
 model.addAttribute("listInspect", inspectService.listAllInspect());  
  
 return "html/home";  
 }  
 @RequestMapping("/inspect/edit/{id}")  
  
 public String editInspect(@PathVariable Long id, Model model){  
  
 model.addAttribute("inspect", inspectService.getOneInspect(id) );  
 return "html/home";  
 }  
 @RequestMapping("/inspect/delete/{id}")  
 public String deleteInspect(@PathVariable Long id){  
 inspectService.deleteInspect(id);  
 return "redirect:/informationInspect";  
  
 }  
  
  
 @PostMapping("/inspect")  
 public String update(Inspect inspect, Model model){  
  
 inspectService.updateInspect(inspect.getId(), inspect.getUsername(),inspect.getGender(),inspect.getPhone(),inspect.getAddress());  
 model.addAttribute("listInspect", inspectService.listAllInspect());  
 return "redirect:/informationInspect";  
  
 }  
 @RequestMapping("/add")  
 public String add(Model model){  
 model.addAttribute("inspectNew",new Inspect());  
 return "html/home";  
 }  
 @PostMapping("inspect/add")  
 public String save(Inspect inspect){  
  
 Role role = roleService.getRole(2L);  
 inspect.getRoles().add(role);  
 String passCode = passwordEncoder.encode(inspect.getPassword());  
 inspect.setPassword(passCode);  
 inspectService.save(inspect);  
 return "redirect:/informationInspect";  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
@Controller  
public class LoginController {  
 public LoginController() {  
 }  
  
 @RequestMapping({"/login"})  
 public String login() {  
 return "html/login";  
 }  
  
 @RequestMapping({"/home"})  
 public String loginSuccess() {  
 return "html/home";  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.controller;  
  
import com.example.quanlivesinh.persistence.Store;  
import java.io.IOException;  
import java.nio.file.CopyOption;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.nio.file.StandardCopyOption;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.util.StringUtils;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.multipart.MultipartFile;  
  
@Controller  
public class StoreController {  
 public StoreController() {  
 }  
  
 @PostMapping({"/add/shop"})  
 public String sd(@ModelAttribute("shopNew") Store store, @RequestParam("hinhAnh") MultipartFile hinhAnh) throws IOException {  
 String fileName = StringUtils.*cleanPath*(hinhAnh.getOriginalFilename());  
 System.*out*.println(fileName);  
 Path path = Paths.*get*("src/main/resources/static/images/store/" + fileName);  
 Files.*copy*(hinhAnh.getInputStream(), path, new CopyOption[]{StandardCopyOption.*REPLACE\_EXISTING*});  
 System.*out*.println(store.getAddress());  
 System.*out*.println(store.getName());  
 System.*out*.println(store.isStatus());  
 Store shopNew = new Store();  
 shopNew.setName(store.getName());  
 shopNew.setImage("/images/store/" + fileName);  
 shopNew.setAddress(store.getAddress());  
 shopNew.setStatus(store.isStatus());  
 shopNew.setOperatingTime(store.getOperatingTime());  
 shopNew.setPhone(store.getPhone());  
 return "html/home";  
 }  
  
 @RequestMapping({"form-add"})  
 public String addShop(Model model) {  
 model.addAttribute("shopNew", new Store());  
 return "html/form-add-shop";  
 }  
  
 @RequestMapping({"informationStore"})  
 public String getStore() {  
 return "html/home";  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.persistence;  
  
import java.util.HashSet;  
import java.util.Set;  
import javax.persistence.CascadeType;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.JoinTable;  
import javax.persistence.ManyToMany;  
import javax.persistence.Table;  
  
@Entity  
@Table(  
 name = "chungnhan"  
)  
public class Certification {  
 @Id  
 @GeneratedValue(  
 strategy = GenerationType.*IDENTITY* )  
 @Column(  
 name = "machungnhan"  
 )  
 private Long id;  
 @Column(  
 name = "tenchungnhan"  
 )  
 private String name;  
 @Column(  
 name = "mota"  
 )  
 private String describe;  
 @Column(  
 name = "image"  
 )  
 private String image;  
 @ManyToMany(  
 fetch = FetchType.*EAGER*,  
 cascade = {CascadeType.*ALL*}  
 )  
 @JoinTable(  
 name = "chitiet\_chungnhan",  
 joinColumns = {@JoinColumn(  
 name = "machungnhan",  
 referencedColumnName = "machungnhan"  
)},  
 inverseJoinColumns = {@JoinColumn(  
 name = "macuahang",  
 referencedColumnName = "macuahang"  
)}  
 )  
 private Set<Store> stores = new HashSet();  
  
 public Certification() {  
 }  
  
 public Certification(Long id, String name, String describe, Set<Store> stores) {  
 this.id = id;  
 this.name = name;  
 this.describe = describe;  
 this.stores = stores;  
 }  
  
 public String getImage() {  
 return this.image;  
 }  
  
 public void setImage(String image) {  
 this.image = image;  
 }  
  
 public Long getId() {  
 return this.id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return this.name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getDescribe() {  
 return this.describe;  
 }  
  
 public void setDescribe(String describe) {  
 this.describe = describe;  
 }  
  
 public Set<Store> getStores() {  
 return this.stores;  
 }  
  
 public void setStores(Set<Store> stores) {  
 this.stores = stores;  
 }  
  
 public String toString() {  
 return "Certification{id=" + this.id + ", name='" + this.name + "', describe='" + this.describe + "'}";  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.persistence;  
  
import java.util.HashSet;  
import java.util.Set;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.JoinTable;  
import javax.persistence.ManyToMany;  
import javax.persistence.Table;  
  
@Entity  
@Table(  
 name = "users"  
)  
public class Inspect {  
 @Id  
 @GeneratedValue(  
 strategy = GenerationType.*IDENTITY* )  
 @Column(  
 name = "user\_id"  
 )  
 private Long id;  
 @Column(  
 name = "USER\_NAME"  
 )  
 private String username;  
 @Column(  
 name = "PASSWORD"  
 )  
 private String password;  
 @Column(  
 name = "GENDER"  
 )  
 private Boolean gender;  
 @Column(  
 name = "PHONE"  
 )  
 private Integer phone;  
 @Column(  
 name = "ADDRESS"  
 )  
 private String address;  
 @ManyToMany  
 @JoinTable(  
 name = "users\_role",  
 joinColumns = {@JoinColumn(  
 name = "USER\_ID",  
 referencedColumnName = "USER\_ID"  
)},  
 inverseJoinColumns = {@JoinColumn(  
 name = "ROLE\_ID",  
 referencedColumnName = "ROLE\_ID"  
)}  
 )  
 private Set<Role> roles = new HashSet();  
  
 public Inspect() {  
 }  
  
 public Inspect(Long id, String username, String password, Boolean gender, Integer phone, String address, Set<Role> roles) {  
 this.id = id;  
 this.username = username;  
 this.password = password;  
 this.gender = gender;  
 this.phone = phone;  
 this.address = address;  
 this.roles = roles;  
 }  
  
 public Long getId() {  
 return this.id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getUsername() {  
 return this.username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return this.password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public Boolean getGender() {  
 return this.gender;  
 }  
  
 public void setGender(Boolean gender) {  
 this.gender = gender;  
 }  
  
 public Integer getPhone() {  
 return this.phone;  
 }  
  
 public void setPhone(Integer phone) {  
 this.phone = phone;  
 }  
  
 public String getAddress() {  
 return this.address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public Set<Role> getRoles() {  
 return this.roles;  
 }  
  
 public void setRoles(Set<Role> roles) {  
 this.roles = roles;  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.persistence;  
  
import com.example.quanlivesinh.enums.ERole;  
import java.util.HashSet;  
import java.util.Set;  
import javax.persistence.CascadeType;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.EnumType;  
import javax.persistence.Enumerated;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.ManyToMany;  
import javax.persistence.Table;  
  
@Entity  
@Table(  
 name = "role"  
)  
public class Role {  
 @Id  
 @GeneratedValue(  
 strategy = GenerationType.*IDENTITY* )  
 @Column(  
 name = "ROLE\_ID"  
 )  
 private Long id;  
 @Enumerated(EnumType.*STRING*)  
 @Column(  
 name = "ROLE\_NAME"  
 )  
 private ERole name;  
 @ManyToMany(  
 mappedBy = "roles",  
 cascade = {CascadeType.*ALL*}  
 )  
 Set<User> users = new HashSet();  
  
 public Role() {  
 }  
  
 public Role(Long id, ERole name) {  
 this.id = id;  
 this.name = name;  
 }  
  
 public Long getId() {  
 return this.id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public ERole getName() {  
 return this.name;  
 }  
  
 public void setName(ERole name) {  
 this.name = name;  
 }  
  
 public boolean equals(Object o) {  
 if (this == o) {  
 return true;  
 } else if (o != null && this.getClass() == o.getClass()) {  
 Role developer = (Role)o;  
 return !this.id.equals(developer.id) ? false : this.name.equals(developer.name);  
 } else {  
 return false;  
 }  
 }  
  
 public int hashCode() {  
 int result = this.id.hashCode();  
 result = 31 \* result + this.name.hashCode();  
 return result;  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.persistence;  
  
import java.time.LocalDateTime;  
import java.util.HashSet;  
import java.util.Set;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.ManyToMany;  
import javax.persistence.Table;  
  
@Entity  
@Table(  
 name = "cuahang"  
)  
public class Store {  
 @Id  
 @GeneratedValue(  
 strategy = GenerationType.*IDENTITY* )  
 @Column(  
 name = "macuahang"  
 )  
 private Long id;  
 @Column(  
 name = "tencuahang"  
 )  
 private String name;  
 @Column(  
 name = "diachi"  
 )  
 private String address;  
 @Column(  
 name = "hotline"  
 )  
 private String phone;  
 @Column(  
 name = "tinhtrang"  
 )  
 private boolean status;  
 @Column(  
 name = "thoigianhoatdong"  
 )  
 private LocalDateTime operatingTime;  
 @Column(  
 name = "image"  
 )  
 private String image;  
 @ManyToMany(  
 mappedBy = "stores",  
 fetch = FetchType.*EAGER* )  
 private Set<Certification> certifications = new HashSet();  
  
 public Store() {  
 }  
  
 public Store(Long id, String name, String address, String phone, boolean status, LocalDateTime operatingTime) {  
 this.id = id;  
 this.name = name;  
 this.address = address;  
 this.phone = phone;  
 this.status = status;  
 this.operatingTime = operatingTime;  
 }  
  
 public Long getId() {  
 return this.id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return this.name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getAddress() {  
 return this.address;  
 }  
  
 public String getImage() {  
 return this.image;  
 }  
  
 public void setImage(String image) {  
 this.image = image;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public String getPhone() {  
 return this.phone;  
 }  
  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
  
 public boolean isStatus() {  
 return this.status;  
 }  
  
 public void setStatus(boolean status) {  
 this.status = status;  
 }  
  
 public LocalDateTime getOperatingTime() {  
 return this.operatingTime;  
 }  
  
 public void setOperatingTime(LocalDateTime operatingTime) {  
 this.operatingTime = operatingTime;  
 }  
  
 public Set<Certification> getCertifications() {  
 return this.certifications;  
 }  
  
 public void setCertifications(Set<Certification> certifications) {  
 this.certifications = certifications;  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.persistence;  
  
import java.util.HashSet;  
import java.util.Set;  
import javax.persistence.CascadeType;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
import javax.persistence.JoinColumn;  
import javax.persistence.JoinTable;  
import javax.persistence.ManyToMany;  
import javax.persistence.Table;  
  
@Entity  
@Table(  
 name = "users"  
)  
public class User {  
 @Id  
 @GeneratedValue(  
 strategy = GenerationType.*IDENTITY* )  
 @Column(  
 name = "USER\_ID"  
 )  
 private Long id;  
 @Column(  
 name = "USER\_NAME"  
 )  
 private String username;  
 @Column(  
 name = "PASSWORD"  
 )  
 private String password;  
 @Column(  
 name = "GENDER"  
 )  
 private boolean gender;  
 @Column(  
 name = "PHONE"  
 )  
 private Integer phone;  
 @Column(  
 name = "ADDRESS"  
 )  
 private String address;  
 @ManyToMany(  
 fetch = FetchType.*LAZY*,  
 cascade = {CascadeType.*ALL*}  
 )  
 @JoinTable(  
 name = "users\_role",  
 joinColumns = {@JoinColumn(  
 name = "USER\_ID",  
 referencedColumnName = "USER\_ID"  
)},  
 inverseJoinColumns = {@JoinColumn(  
 name = "ROLE\_ID",  
 referencedColumnName = "ROLE\_ID"  
)}  
 )  
 private Set<Role> roles = new HashSet();  
  
 public User() {  
 }  
  
 public User(Long id, String username, String password, boolean gender, Integer phone, String address, Set<Role> roles) {  
 this.id = id;  
 this.username = username;  
 this.password = password;  
 this.gender = gender;  
 this.phone = phone;  
 this.address = address;  
 this.roles = roles;  
 }  
  
 public Long getId() {  
 return this.id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getUsername() {  
 return this.username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return this.password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public boolean isGender() {  
 return this.gender;  
 }  
  
 public void setGender(boolean gender) {  
 this.gender = gender;  
 }  
  
 public Integer getPhone() {  
 return this.phone;  
 }  
  
 public void setPhone(Integer phone) {  
 this.phone = phone;  
 }  
  
 public String getAddress() {  
 return this.address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public Set<Role> getRoles() {  
 return this.roles;  
 }  
  
 public void setRoles(Set<Role> roles) {  
 this.roles = roles;  
 }  
  
 public String toString() {  
 return "User{id=" + this.id + ", username='" + this.username + "', password='" + this.password + "', gender=" + this.gender + ", phone=" + this.phone + ", address='" + this.address + "', roles=" + this.roles + "}";  
 }  
  
 public boolean equals(Object o) {  
 if (this == o) {  
 return true;  
 } else if (o != null && this.getClass() == o.getClass()) {  
 User developer = (User)o;  
 return !this.id.equals(developer.id) ? false : this.username.equals(developer.username);  
 } else {  
 return false;  
 }  
 }  
  
 public int hashCode() {  
 int result = this.id.hashCode();  
 result = 31 \* result + this.username.hashCode();  
 return result;  
 }  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.repository;  
  
import com.example.quanlivesinh.persistence.Certification;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Modifying;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface CertificationRepository extends JpaRepository<Certification, Long> {  
 Certification getCertificationById(Long id);  
  
 void deleteById(Long id);  
  
 @Modifying  
 @Query("UPDATE Certification c SET c.name = :name ,c.describe = :describe WHERE c.id = :id")  
 void updateCertifition(@Param("id") Long id, @Param("name") String name, @Param("describe") String describe);  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.repository;  
  
import com.example.quanlivesinh.persistence.Inspect;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Modifying;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface InspectRepository extends JpaRepository<Inspect, Long> {  
 Inspect getInspectsById(Long id);  
  
 Inspect saveAndFlush(Inspect inspect);  
  
 @Modifying  
 @Query("update Inspect u set u.username = :newName , u.gender = :newGender , u.phone = :newPhone , u.address = :newAddress where u.id = :newId")  
 void updateInspect(@Param("newId") Long id, @Param("newName") String username, @Param("newGender") boolean gender, @Param("newPhone") Integer phone, @Param("newAddress") String address);  
  
 void deleteById(Long id);  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.repository;  
  
import com.example.quanlivesinh.enums.ERole;  
import com.example.quanlivesinh.persistence.Role;  
import java.util.Optional;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface RoleRepository extends JpaRepository<Role, Long> {  
 Optional<Role> findByName(ERole name);  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.repository;  
  
import com.example.quanlivesinh.persistence.Store;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface StoreRepository extends JpaRepository<Store, Long> {  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.repository;  
  
import com.example.quanlivesinh.persistence.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface UserRepository extends JpaRepository<User, Long> {  
 User findByUsername(String username);  
}

//  
// Source code recreated from a .class file by IntelliJ IDEA  
// (powered by FernFlower decompiler)  
//  
  
package com.example.quanlivesinh.security;  
  
import com.example.quanlivesinh.service.impl.UserDetailServiceImpl;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;  
import org.springframework.security.config.annotation.web.configurers.AuthorizeHttpRequestsConfigurer;  
import org.springframework.security.config.annotation.web.configurers.FormLoginConfigurer;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig extends WebSecurityConfigurerAdapter {  
 @Autowired  
 UserDetailServiceImpl userDetailServiceImpl;  
  
 public SecurityConfig() {  
 }  
  
 @Bean  
 public static BCryptPasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder();  
 }  
  
 @Autowired  
 public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {  
 auth.userDetailsService(this.userDetailServiceImpl).passwordEncoder(*passwordEncoder*());  
 }  
  
 protected void configure(HttpSecurity http) throws Exception {  
 ((FormLoginConfigurer)((FormLoginConfigurer)((FormLoginConfigurer)((HttpSecurity)((AuthorizeHttpRequestsConfigurer.AuthorizedUrl)((AuthorizeHttpRequestsConfigurer.AuthorizedUrl)http.authorizeHttpRequests().antMatchers(new String[]{"/"})).permitAll().anyRequest()).authenticated().and()).formLogin().loginPage("/login").loginProcessingUrl("/j\_spring\_security\_check")).defaultSuccessUrl("/home")).failureUrl("/login?suces=fail")).passwordParameter("password").usernameParameter("username").permitAll();  
 }  
}

Link github: [nguyenquan23/quanlivesinh (github.com)](https://github.com/nguyenquan23/quanlivesinh/tree/foodsafety)