**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG – HCM**

**KHOA ĐIỆN TỬ - VIỄN THÔNG**



**BÁO CÁO ĐỒ ÁN NHÓM JAVA**

***Đề tài:***

**SmartAutoStore**

**Xây dựng hệ thống quản lý và bán hàng trực tuyến**

***Danh sách sinh viên:***

|  |  |
| --- | --- |
| *Võ Minh Trung* | *21200365* |
| *Nguyễn Hồng Quân* | *21200337* |
| *Trần Nguyễn Tường Vũ* | *21200372* |
| *Nguyễn Thành Trung* | *20200381* |
| *Võ Tư Hải* | *21200285* |
| *Trần Quốc Việt* | *21200370* |

***Giảng viên:***

*ThS.* [*Huỳnh Thị Minh Tuyến*](https://courses.hcmus.edu.vn/user/view.php?id=28309&course=9354)

**TP. HỒ CHÍ MINH – NĂM 2025**

**MỤC LỤC**

[I. PHẦN MỞ ĐẦU 1](#_Toc186759134)

[II. TỔNG QUAN CÔNG NGHỆ 2](#_Toc186759135)

[1. Spring boot 2](#_Toc186759136)

[2. Servlet 4](#_Toc186759137)

[3. Spring Security 4](#_Toc186759138)

[4. JWT (JSON Web Token) 4](#_Toc186759139)

[5. JPA (Java Persistence API) 4](#_Toc186759140)

[6. MySQL Driver 4](#_Toc186759141)

[7. ReactJS 4](#_Toc186759142)

[8. JavaFX 4](#_Toc186759143)

[III. Phân tích yêu cầu và thiết kế hệ thống 4](#_Toc186759144)

[IV. Triển khai hệ thống 4](#_Toc186759145)

[V. Kiểm thử hệ thống 4](#_Toc186759146)

[VI. Kết luận và hướng phát triển 4](#_Toc186759147)

# PHẦN MỞ ĐẦU

1. **Lí do chọn đề tài**

Trong bối cảnh công nghệ thông tin ngày càng phát triển, việc xây dựng các hệ thống quản lý và bán hàng trực tuyến đã trở thành nhu cầu thiết yếu trong nhiều lĩnh vực kinh doanh. Ngành công nghiệp ô tô, với sự gia tăng nhanh chóng của nhu cầu mua sắm và quản lý xe hơi, cũng không nằm ngoài xu hướng này.  
Việc tạo ra một hệ thống quản lý bán hàng trực tuyến không chỉ giúp doanh nghiệp tiếp cận khách hàng dễ dàng hơn mà còn hỗ trợ nâng cao hiệu quả trong quản lý và vận hành. Đề tài 'Xây dựng hệ thống quản lý và bán hàng trực tuyến - Application Car Store' ra đời nhằm giải quyết các vấn đề thực tiễn trong hoạt động kinh doanh xe ô tô, mang lại một nền tảng hiện đại, tiện ích và chuyên nghiệp.

1. **Mục tiêu của đề tài**

Mục tiêu của đề tài là xây dựng một hệ thống Application Car Store với các tính năng chính:  
- Quản lý sản phẩm: Lưu trữ và quản lý thông tin xe ô tô như loại xe, hãng xe, kích thước, màu sắc, giá cả.  
- Quản lý người dùng: Lưu trữ thông tin khách hàng, lịch sử mua hàng.  
- Xử lý đơn hàng: Tạo, cập nhật và quản lý các đơn đặt hàng.  
- Hệ thống xác thực và phân quyền: Đảm bảo an toàn và bảo mật thông tin người dùng thông qua Spring Security và JWT.  
- Giao diện thân thiện: Cung cấp giao diện web hiện đại cho khách hàng và giao diện desktop tiện lợi cho quản trị viên.

1. **Phạm vi nghiên cứu**

Hệ thống được xây dựng dựa trên kiến trúc đa tầng và áp dụng các công nghệ mới như:  
- Backend: Sử dụng Spring Boot kết hợp với Spring Security, JWT, và JPA để quản lý cơ sở dữ liệu MySQL.  
- Frontend: Phát triển giao diện người dùng bằng ReactJS, hỗ trợ tìm kiếm, lọc sản phẩm, và đặt hàng.  
- Admin: Xây dựng ứng dụng JavaFX dành cho quản trị viên để quản lý sản phẩm, đơn hàng và thông tin người dùng.

1. **Ý nghĩa thực tiễn**

Đề tài mang lại những giá trị thực tiễn như:  
- Tạo ra một nền tảng bán hàng xe ô tô trực tuyến hiệu quả, hiện đại và an toàn.  
- Giúp doanh nghiệp tối ưu hóa quy trình quản lý và tiếp cận khách hàng tiềm năng.  
- Tích hợp các công nghệ tiên tiến để đáp ứng nhu cầu thực tế của doanh nghiệp trong ngành ô tô.

1. **Cấu trúc tài liệu**

Tài liệu được chia thành các phần chính như sau:  
1. Giới thiệu và lý do chọn đề tài.  
2. Phân tích và tổng quan công nghệ.  
3. Thiết kế và triển khai hệ thống.  
4. Kết quả thực nghiệm và đánh giá.  
5. Kết luận và hướng phát triển trong tương lai.

# TỔNG QUAN CÔNG NGHỆ

1. Spring boot

**Spring Boot** là một framework phát triển ứng dụng Java phổ biến, giúp việc xây dựng và triển khai ứng dụng trở nên đơn giản và nhanh chóng. Nó cung cấp các cấu hình tự động và tiện ích giúp giảm thiểu việc cấu hình thủ công trong Spring.

* **Mục tiêu của Spring Boot**

Giảm thiểu cấu hình: Spring Boot tự động cấu hình ứng dụng dựa trên các phụ thuộc.

Tạo ứng dụng độc lập: Các ứng dụng Spring Boot có thể chạy mà không cần triển khai vào máy chủ bên ngoài.

Cung cấp các tính năng tích hợp: Các tính năng như bảo mật, kiểm thử, và theo dõi đã được tích hợp sẵn.

* **Cấu trúc một dự án Spring Boot**

Maven/Gradle:

* Maven: Dùng để quản lý phụ thuộc và xây dựng dự án.
* Gradle: Là công cụ build hiện đại hỗ trợ nhiều loại dự án và dễ dàng mở rộng.

Cấu trúc thư mục:

* src/main/java: Chứa mã nguồn Java.
* src/main/resources: Chứa tài nguyên như application.properties, các file cấu hình.
* src/test/java: Chứa mã nguồn kiểm thử.
* **File chính:**

File chính trong ứng dụng Spring Boot thường có tên là Application.java, với annotation @SpringBootApplication để đánh dấu đây là lớp chính khởi động ứng dụng.

|  |
| --- |
| @SpringBootApplication  public class Application {  public static void main(String[] args) {  SpringApplication.run(Application.class, args);  }  } |

* **Các tính năng chính của Spring Boot**
* **Auto Configuration:** Spring Boot tự động cấu hình ứng dụng dựa trên các thư viện có sẵn trong classpath. Ví dụ, nếu bạn sử dụng thư viện spring-boot-starter-web, Spring Boot sẽ tự động cấu hình một máy chủ Tomcat cho bạn.
* **Embedded Servers**: Spring Boot tích hợp các máy chủ nhúng như Tomcat, Jetty, Undertow. Bạn có thể chạy ứng dụng mà không cần phải triển khai vào một máy chủ ngoài.
* **Spring Boot Starter:**
* Các starter là các bộ phụ thuộc đã được Spring Boot chuẩn bị sẵn cho các tính năng như web, cơ sở dữ liệu, bảo mật, v.v.
* Ví dụ: spring-boot-starter-web cho ứng dụng web, spring-boot-starter-data-jpa cho ứng dụng dùng JPA.
* **Spring Boot Actuator:** Cung cấp các endpoint để theo dõi và giám sát ứng dụng, như /actuator/health, /actuator/metrics

1. Servlet

Servlet là một công nghệ trong Java giúp phát triển các ứng dụng web động. Servlet có khả năng xử lý yêu cầu HTTP từ phía client và trả lại kết quả dưới dạng một phản hồi. Servlet hoạt động như một thành phần trên server, giúp quản lý và điều phối các yêu cầu HTTP từ client, chủ yếu là các trình duyệt web. Được triển khai trong môi trường máy chủ ứng dụng web (ví dụ: Tomcat, Jetty), Servlet có thể xử lý các yêu cầu HTTP và thực hiện các tác vụ như truy vấn cơ sở dữ liệu, kiểm tra thông tin đăng nhập, hoặc xử lý các form nhập liệu.

Servlet được triển khai dưới dạng các lớp Java đơn giản, có thể được cấu hình để xử lý các loại yêu cầu HTTP cụ thể như GET, POST, PUT, DELETE. Servlet nhận yêu cầu từ client thông qua đối tượng HttpServletRequest và trả về kết quả qua đối tượng HttpServletResponse.

* Ứng dụng của Servlet trong hệ thống

Servlet có thể được sử dụng trong nhiều tình huống khác nhau trong các ứng dụng web Java:

1. Xử lý yêu cầu HTTP và phân phối chúng tới các controller trong Spring Boot: Trong hệ thống sử dụng Spring Boot, Servlet có thể đóng vai trò là phần mềm trung gian, xử lý các yêu cầu HTTP trước khi chuyển chúng đến các controller của Spring Boot. Servlet giúp phân loại các yêu cầu, ví dụ như xử lý các yêu cầu đăng nhập hoặc các yêu cầu kiểm tra quyền truy cập, sau đó chuyển các yêu cầu hợp lệ tới các controller của Spring Boot để xử lý.
2. Quản lý phiên làm việc (Session Management): Servlet có thể dễ dàng sử dụng để quản lý các phiên làm việc trong ứng dụng web. Thông qua đối tượng HttpSession, Servlet có thể lưu trữ và truy xuất thông tin liên quan đến phiên làm việc của người dùng như thông tin đăng nhập, quyền truy cập và các trạng thái tạm thời của người dùng.
3. Xử lý các yêu cầu không cần Spring Boot: Dù Spring Boot là một framework mạnh mẽ, trong một số trường hợp, các ứng dụng web có thể yêu cầu xử lý các yêu cầu HTTP trực tiếp mà không cần thông qua Spring Boot. Ví dụ, khi cần xử lý các yêu cầu RESTful hoặc SOAP đơn giản, Servlet có thể thực hiện công việc này mà không cần Spring Boot, chỉ với các thư viện hỗ trợ như Java EE.
4. Spring Security

Spring Security là một framework bảo mật mạnh mẽ và linh hoạt cho các ứng dụng Java, được tích hợp sẵn trong hệ sinh thái Spring. Nó cung cấp các tính năng bảo mật toàn diện như xác thực (authentication), phân quyền (authorization), bảo vệ các endpoint, và bảo mật HTTP. Spring Security cho phép lập trình viên dễ dàng triển khai các cơ chế bảo mật phức tạp mà không cần phải xây dựng từ đầu.

* Các tính năng chính của Spring Security bao gồm:

1. Xác thực người dùng: Xác minh danh tính của người dùng qua các phương thức như username/password, OAuth2, LDAP, v.v.
2. Phân quyền: Cung cấp các cơ chế để phân chia quyền truy cập cho người dùng dựa trên vai trò (role), quyền (authority), hoặc các đặc tính của người dùng.
3. Bảo vệ ứng dụng web: Tự động cấu hình bảo mật cho các ứng dụng web, bao gồm mã hóa dữ liệu và bảo vệ khỏi các cuộc tấn công như CSRF, XSS.

* Ứng dụng của Spring Security trong hệ thống

1. Bảo vệ các endpoint REST API khỏi các truy cập trái phép: Spring Security giúp bảo vệ các API trong hệ thống bằng cách xác thực người dùng và phân quyền truy cập cho các endpoint. Chỉ người dùng có quyền truy cập hợp lệ mới có thể gọi các API nhất định, chẳng hạn như API yêu cầu quyền quản trị viên để sửa đổi dữ liệu.

Ví dụ, có thể sử dụng Spring Security để yêu cầu người dùng phải đăng nhập với một token JWT (JSON Web Token) hợp lệ để truy cập vào các API cần bảo mật.

1. Xử lý đăng nhập, đăng ký, và phân quyền người dùng: Spring Security cung cấp các cơ chế để xử lý các yêu cầu đăng nhập, đăng ký và phân quyền người dùng. Việc phân quyền người dùng dựa trên vai trò như admin, khách hàng, hay người dùng bình thường giúp đảm bảo rằng chỉ những người dùng có quyền thích hợp mới có thể thực hiện các thao tác nhất định.

**Đăng nhập**: Spring Security hỗ trợ các phương thức đăng nhập như form login, Basic Auth, hoặc OAuth.

Phân quyền: Dựa vào vai trò người dùng (ví dụ: ROLE\_ADMIN, ROLE\_USER), các API có thể bị hạn chế chỉ cho phép người dùng có quyền truy cập tương ứng.

1. JWT (JSON Web Token)

JWT (JSON Web Token) là một phương thức phổ biến để xác thực và truyền thông tin giữa các hệ thống dưới dạng một token mã hóa. JWT sử dụng một chuỗi JSON được mã hóa và ký số để đảm bảo tính bảo mật khi truyền tải thông tin. Thông qua JWT, các hệ thống có thể xác thực người dùng và chia sẻ thông tin mà không cần phải lưu trữ session trên server, giúp giảm tải cho server và cải thiện hiệu suất của ứng dụng.

* JWT bao gồm ba phần chính:
  1. Header: Chứa thông tin về thuật toán mã hóa và loại token (thường là "JWT").
  2. Payload: Chứa thông tin người dùng hoặc các dữ liệu liên quan, chẳng hạn như thông tin đăng nhập, quyền hạn, v.v.
  3. Signature: Được tạo ra bằng cách mã hóa Header và Payload với một khóa bí mật, giúp đảm bảo tính toàn vẹn của token và xác thực nguồn gốc của nó.
* Ứng dụng của JWT trong hệ thống
  1. Xử lý việc đăng nhập, đăng ký, và tạo token cho người dùng: JWT thường được sử dụng trong các hệ thống để xử lý việc đăng nhập và đăng ký người dùng. Sau khi người dùng đăng nhập thành công, server sẽ tạo một JWT và gửi lại cho client. Client sẽ lưu token này và gửi nó cùng với mỗi yêu cầu đến server trong header Authorization để xác thực người dùng.
  2. Quản lý token xác thực trong suốt phiên làm việc của người dùng: Trong suốt phiên làm việc của người dùng, JWT giúp server không cần phải lưu trữ session của người dùng. Mỗi khi client gửi một yêu cầu, JWT trong header sẽ được kiểm tra để xác thực quyền truy cập của người dùng. Nếu token hợp lệ và chưa hết hạn, người dùng sẽ được phép truy cập vào các API bảo mật.

1. JPA (Java Persistence API)

****JPA (Java Persistence API)**** là một API trong Java giúp quản lý và thao tác với cơ sở dữ liệu, đặc biệt là trong các ứng dụng Java. JPA là một phần của Java EE (Enterprise Edition) và cung cấp một cách tiếp cận tiêu chuẩn để lưu trữ và truy xuất dữ liệu từ cơ sở dữ liệu quan hệ. JPA không trực tiếp tương tác với cơ sở dữ liệu mà thay vào đó cung cấp một lớp trừu tượng thông qua các đối tượng Java, giúp đơn giản hóa việc thao tác với dữ liệu.

JPA cho phép lập trình viên sử dụng các entity (thực thể) Java để ánh xạ dữ liệu từ cơ sở dữ liệu vào các đối tượng Java thông qua các annotations như @Entity, @Table, @Id, @Column. Ngoài ra, JPA cũng hỗ trợ thực hiện các truy vấn dữ liệu thông qua **JPQL** (Java Persistence Query Language), giúp việc truy vấn dữ liệu trở nên dễ dàng hơn và độc lập với cơ sở dữ liệu.

* Ứng dụng của JPA trong hệ thống

1. Quản lý và thao tác với cơ sở dữ liệu MySQL: JPA cho phép kết nối và thao tác với cơ sở dữ liệu MySQL một cách hiệu quả. Thông qua JPA, các ứng dụng có thể dễ dàng lưu trữ và truy xuất dữ liệu từ MySQL mà không cần phải viết các câu lệnh SQL phức tạp. JPA giúp ánh xạ các đối tượng Java vào các bảng trong cơ sở dữ liệu và ngược lại, giúp quản lý dữ liệu một cách tự động. Ví dụ, một entity User có thể được ánh xạ tới một bảng users trong MySQL, với các trường tương ứng là các thuộc tính của entity.
2. Thực hiện các truy vấn dữ liệu: JPA cung cấp một cách tiếp cận mạnh mẽ để thực hiện các truy vấn dữ liệu. Thông qua JPQL (Java Persistence Query Language), bạn có thể viết các câu truy vấn để tìm kiếm, cập nhật hoặc xóa dữ liệu mà không phải lo lắng về chi tiết SQL. JPQL cho phép sử dụng các đối tượng Java trong truy vấn thay vì tên bảng và cột cơ sở dữ liệu.

* Tìm kiếm sản phẩm: Giả sử có một bảng products, bạn có thể sử dụng JPA để truy vấn các sản phẩm theo tên hoặc giá trị.

|  |
| --- |
| @Query("SELECT p FROM Product p WHERE p.name = :name")  List<Product> findByName(@Param("name") String name); |

* Lưu trữ thông tin người dùng và đơn hàng: Bạn có thể sử dụng JPA để lưu thông tin người dùng và đơn hàng vào cơ sở dữ liệu. JPA hỗ trợ các thao tác cơ bản như save, findById, và delete.

1. MySQL Driver

****MySQL Driver**** là một thư viện phần mềm cho phép ứng dụng Java kết nối và giao tiếp với cơ sở dữ liệu MySQL. Được cung cấp bởi Oracle, MySQL Driver (còn được gọi là **MySQL Connector/J**) là một JDBC (Java Database Connectivity) driver cho phép Java ứng dụng kết nối với MySQL và thực hiện các thao tác cơ sở dữ liệu như truy vấn, cập nhật và quản lý dữ liệu.

MySQL Driver thực hiện các tác vụ truyền thông giữa ứng dụng Java và cơ sở dữ liệu MySQL, thông qua các kết nối JDBC. Điều này giúp các ứng dụng Java có thể tương tác với MySQL một cách dễ dàng và hiệu quả. Thư viện này hỗ trợ đầy đủ các tính năng của MySQL, bao gồm mã hóa, kết nối SSL, và hỗ trợ các truy vấn SQL chuẩn.

* Ứng dụng của MySQL Driver trong hệ thống

1. Lưu trữ dữ liệu sản phẩm, người dùng và đơn hàng: MySQL Driver cho phép ứng dụng Java kết nối với cơ sở dữ liệu MySQL để lưu trữ và truy xuất các dữ liệu liên quan đến sản phẩm, người dùng và đơn hàng. Khi sử dụng MySQL trong hệ thống, bạn có thể tạo các bảng như products, users, và orders để lưu trữ thông tin sản phẩm, người dùng, và các đơn hàng của họ. MySQL Driver giúp Java ứng dụng tương tác với cơ sở dữ liệu này thông qua các phương thức JDBC như prepareStatement(), executeQuery(), và executeUpdate().
2. Kết nối và thực hiện các truy vấn SQL trong hệ thống: MySQL Driver cũng cho phép ứng dụng Java thực hiện các truy vấn SQL như SELECT, UPDATE, DELETE, v.v., giúp hệ thống có thể truy xuất và cập nhật dữ liệu khi cần thiết. Các truy vấn này có thể được thực hiện thông qua các đối tượng như Statement, PreparedStatement, và CallableStatement của JDBC.
3. ReactJS
4. JavaFX

# Phân tích yêu cầu và thiết kế hệ thống

# Triển khai hệ thống

# Kiểm thử hệ thống

# Kết luận và hướng phát triển