

Báo cáo dự án PRJ301

TÊN DỰ ÁN: WEBSITE LAPTOPSHOP

ĐOÀN KIẾN QUỐC – SE1888

MỤC LỤC

PHẦN I: CƠ SỞ LÝ THUYẾT	3
I. Mô hình MVC	3
1. MVC là gì ? Đặc điểm của MVC	3
2. Các thành phần trong mô hình MVC.....	3
II. Sơ lược về Servlet	3
1. doGet và doPost.....	3
2. CRUD cơ bản với Servlet và Microsoft SQL Server:.....	4
3. Session.....	5
4. Cookies	6
5. Filter	6
III. JSP (Jakarta Server Pages)	8
1. JSP và HTML	8
2. EL và JSTL.....	8
IV. Framework Bootstrap dùng cho Frontend	9
1. Hệ thống 12-grid system với Bootstrap 5.3	9
2. Card	10
3. Modal	11
V. Javascript với jQuery và AJAX	13
1. jQuery, AJAX và JSON là gì ?	13
2. Tại sao nên dùng jQuery để xử lý form với AJAX	13
3. Ví dụ đơn giản về sử dụng AJAX để xử lý form	14
PHẦN II: KHẢO SÁT, PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG	17
I. Khảo sát	17
1. Giới thiệu website Laptopshop	17
2. Đánh giá	17
3. Mục đích của website	17
II. Phân tích và thiết kế hệ thống.....	17
1. Yêu cầu của khách hàng về website.....	17
2. Phân tích, thiết kế cơ sở dữ liệu	18
3. Sơ đồ Use Case.....	20
PHẦN III: THIẾT KẾ WEBSITE	20

I. Thiết kế giao diện phía User(người dùng)	20
1. Thiết kế trang home gồm: header, footer và các banner:	20
2. Hiển thị, sắp xếp, lọc sản phẩm theo tiêu chí	21
3. Giao diện giỏ hàng, thanh toán:.....	22
II. Thiết kế giao diện Admin và Shipper	23
1. Phần sidebar.....	23
2. Thiết kế trang giao diện quản lí.....	23
III. Kỹ thuật phân trang sử dụng SQL Server và Servlet	26
1. Tại sao phải phân trang ?	27
2. Công thức phân trang và cách áp dụng:	27
IV. Thiết kế giỏ hàng dùng AJAX và Database	32
1. Tại sao dùng AJAX ?.....	32
2. Thêm sản phẩm vào giỏ hàng	32
3. Giỏ hàng tự động cập nhật cùng với database	33
PHẦN IV: KẾT LUẬN	36

PHẦN I: CƠ SỞ LÝ THUYẾT

I. Mô hình MVC

1. MVC là gì ? Đặc điểm của MVC

MVC là viết tắt của Model-View-Controller. Cấu trúc Model-View-Controller (MVC) là một mẫu kiến trúc/mẫu thiết kế (design pattern) tách ứng dụng thành ba thành phần logic chính: Model, View và Controller. Mỗi thành phần kiến trúc được xây dựng để xử lý các khía cạnh phát triển cụ thể của một ứng dụng.

2. Các thành phần trong mô hình MVC

Model: Model là các thành phần của ứng dụng tương ứng với tất cả logic liên quan đến miền dữ liệu (data domain), hoặc nói ngắn gọn đây là phần back-end chứa tất cả logic dữ liệu của ứng dụng. Dữ liệu ở đây có thể là dữ liệu đang được truyền giữa các thành phần View và Controller hoặc bất kỳ dữ liệu nào khác liên quan đến logic của doanh nghiệp.

Ví dụ: Giả sử bạn đang phát triển một ứng dụng mua sắm. Ở đây, Model sẽ chỉ định giỏ hàng sẽ bao gồm những dữ liệu nào — như mặt hàng, giá cả, v.v. — và những dữ liệu nào đã có sẵn trong giỏ hàng.

View là các thành phần hiển thị giao diện người dùng (UI) của ứng dụng. Thông thường, giao diện người dùng này được tạo từ dữ liệu Model.

Ví dụ: Trong ứng dụng mua sắm, View sẽ xác định cách hiển thị giỏ hàng cho người dùng và nhận dữ liệu từ Model để hiển thị. View sẽ bao gồm tất cả các thành phần UI như hiển thị nút bấm, danh sách thả xuống, v.v. mà người dùng cuối cùng tương tác.

Controller: là các thành phần xử lý tương tác của người dùng để làm việc với Model (cập nhật logic dữ liệu) hoặc/ và với View (cập nhật hiển thị giao diện người dùng).

Ví dụ: Trong ứng dụng mua sắm, ở giỏ hàng của người dùng, bạn có thể thêm các button cho phép người dùng thêm hoặc xóa các mặt hàng.

II. Sơ lược về Servlet

1. doGet và doPost

GET	POST
Dữ liệu được gửi qua phần header, giới hạn lượng dữ liệu (tối đa 2048 ký tự).	Dữ liệu được gửi qua phần body, có thể gửi lượng lớn dữ liệu.
Không an toàn vì dữ liệu hiển thị trên thanh URL.	An toàn hơn do dữ liệu không hiển thị trên URL.

Có thể đánh dấu trang.	Không thể đánh dấu trang.
Idempotent: các yêu cầu lặp lại không thay đổi kết quả.	Non-idempotent: yêu cầu lặp lại có thể thay đổi kết quả.
Hiệu quả hơn và thường được dùng nhiều hơn POST.	Kém hiệu quả hơn và ít được sử dụng.

Trong giao tiếp request-response giữa máy chủ và máy khách, hai phương thức phổ biến là:

- **GET**: Được sử dụng để yêu cầu dữ liệu từ một tài nguyên cụ thể.
- **POST**: Được sử dụng để gửi dữ liệu đã xử lý đến một tài nguyên cụ thể.

Phương thức **GET** chủ yếu để truy vấn và nhận thông tin, còn **POST** dùng khi cần gửi dữ liệu để máy chủ xử lý.

2. CRUD cơ bản với Servlet và Microsoft SQL Server:

Dưới đây là ví dụ cơ bản về các thao tác **CRUD** (Create, Read, Update, Delete) với **Servlet** và **MS SQL Server**.

0. Kết nối cơ sở dữ liệu:

```
public Connection getConnection() throws ClassNotFoundException, SQLException {
    Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
    return DriverManager.getConnection(
        "jdbc:sqlserver://localhost:1433;databaseName=mydb", "username",
        "password");
}
```

1. Create(thêm dữ liệu)

```
public void createProduct(String name, int price) throws SQLException, ClassNotFoundException {
    String sql = "INSERT INTO Products (name, price) VALUES (?, ?)";
    try (Connection conn = getConnection(); PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setString(1, name);
        stmt.setInt(2, price);
        stmt.executeUpdate();
    }
}
```

2. Read(đọc dữ liệu):

```

public List<Product> getAllProducts() throws SQLException, ClassNotFoundException
{
    String sql = "SELECT * FROM Products";
    List<Product> products = new ArrayList<>();
    try (Connection conn = getConnection();
        Statement stmt = conn.createStatement()) {
        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) {
            String name = rs.getString("name");
            int price = rs.getInt("price");
            products.add(new Product(name, price));
        }
    }
    return products;
}

```

3. Update (cập nhật dữ liệu)

```

public void updateProduct(int id, String name, int price) throws SQLException,
ClassNotFoundException {
    String sql = "UPDATE Products SET name = ?, price = ? WHERE id = ?";
    try (Connection conn = getConnection();
        PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setString(1, name);
        stmt.setInt(2, price);
        stmt.setInt(3, id);
        stmt.executeUpdate();
    }
}

```

4. Delete (Xóa dữ liệu)

```

public void deleteProduct(int id) throws SQLException, ClassNotFoundException {
    String sql = "DELETE FROM Products WHERE id = ?";
    try (Connection conn = getConnection(); PreparedStatement stmt = conn.pre-
pareStatement(sql)) {
        stmt.setInt(1, id);
        stmt.executeUpdate();
    }
}

```

3. Session

Trong Servlet, **session** là một cơ chế giúp lưu trữ dữ liệu của người dùng trong suốt phiên làm việc giữa máy khách và máy chủ. Mỗi người dùng khi truy cập sẽ được cấp một **session ID** duy

nhất, và dữ liệu có thể được lưu trữ dưới dạng các thuộc tính (attributes) trong đối tượng HttpSession. Session tồn tại trong suốt thời gian người dùng tương tác với ứng dụng (cho đến khi đóng trình duyệt hoặc hết thời gian, hay còn gọi là timeout).

- Tạo session: HttpSession session = request.getSession();
- Lưu thuộc tính: session.setAttribute("key", value);
- Lấy thuộc tính: session.getAttribute("key");

Trong dự án này, Session được sử dụng để duy trì phiên đăng nhập của người dùng.

4. Cookies

Cookies trong Servlet là cách lưu trữ dữ liệu người dùng trên trình duyệt. Máy chủ gửi một cookie đến trình duyệt thông qua phản hồi HTTP, và trình duyệt lưu trữ cookie đó để gửi lại trong các yêu cầu HTTP tiếp theo. Cookies được sử dụng để lưu trữ thông tin trạng thái như phiên đăng nhập hoặc giỏ hàng.

Tạo cookie:

```
Cookie cookie = new Cookie("key", "value");
response.addCookie(cookie);
```

Đọc cookie:

```
Cookie[] cookies = request.getCookies();
for (Cookie c : cookies) {
    if (c.getName().equals("key")) {
        String value = c.getValue();
    }
}
```

Các thuộc tính của Cookie

```
cookie.setMaxAge(60 * 60 * 24); // Cookie tồn tại 1 ngày
cookie.setPath("/");             // Cookie hợp lệ trên toàn bộ ứng dụng
```

Cookies giúp theo dõi thông tin như đăng nhập, sở thích người dùng, và phiên làm việc.

Trong dự án này, Cookies được sử dụng cho phần Remember Me trong module đăng nhập (tự động điền email và mật khẩu cho người dùng đã tích vào ô Duy trì đăng nhập).

5. Filter

Filter trong Servlet là một thành phần xử lý trước hoặc sau các yêu cầu (requests) hoặc phản hồi (responses) từ client. Nó có thể được dùng để thực hiện các tác vụ như kiểm tra xác thực, logging mà không cần sửa đổi logic chính của Servlet.

Ví dụ về Filter

```
@WebFilter("/restricted/*") // Áp dụng filter cho các URL bắt đầu bằng /re-
stricted
public class AuthFilter implements Filter {
    public void doFilter(ServletRequest request, ServletResponse response, Fil-
terChain chain)
        throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) request;
        HttpSession session = httpRequest.getSession(false);

        // Kiểm tra xem người dùng đã đăng nhập hay chưa
        if (session == null || session.getAttribute("user") == null) {
            ((HttpServletResponse) response).sendRedirect("/login.jsp"); //
Chuyển hướng về trang đăng nhập
        } else {
            chain.doFilter(request, response); // Cho phép yêu cầu tiếp tục đến
Servlet
        }
    }
}
```

Cấu hình trong web.xml:

```
<filter>
    <filter-name>AuthFilter</filter-name>
    <filter-class>com.example.AuthFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>AuthFilter</filter-name>
    <url-pattern>/restricted/*</url-pattern>
</filter-mapping>
```

Chức năng chính của filter:

- Kiểm tra quyền truy cập: Ví dụ như kiểm tra session để xác nhận người dùng đã đăng nhập.
- Logging: Ghi lại thông tin yêu cầu, phản hồi.
- Nén nội dung: Nén dữ liệu phản hồi trước khi gửi về phía client.
- Thay đổi yêu cầu/ phản hồi: Ví dụ thay đổi header hoặc nội dung.

Filter không phải là Servlet, nhưng hoạt động tương tác với Servlet để quản lý các yêu cầu và phản hồi một cách linh hoạt.

Trong dự án này, Filter được dùng để xác thực vai trò của người dùng, kiểm tra người dùng có quyền truy cập vào 1 trang cụ thể nào đó, kiểm tra phiên đăng nhập có đang được duy trì hay không.

III. JSP (Jakarta Server Pages)

1. JSP và HTML

JSP (Jakarta Server Pages) là một công nghệ của Jakarta EE, cho phép tạo các trang web động bằng cách nhúng mã Java vào các tệp HTML. JSP giúp dễ dàng phát triển các ứng dụng web bằng cách tách phần giao diện và phần xử lý logic phía server. Các tệp JSP được biên dịch thành các **Servlet** để xử lý yêu cầu từ client.

Đặc điểm chính của JSP:

- **Tích hợp với Java:** Mã Java có thể được viết trực tiếp trong tệp JSP.
- **Các thẻ JSP:** Sử dụng các thẻ như `<%= %>` để nhúng mã Java.
- **Tái sử dụng mã:** Cho phép sử dụng **custom tags** và **JSP includes** để tái sử dụng các đoạn mã.

JSP thường được dùng để tạo các trang web động và dễ dàng duy trì.

So với **HTML**, JSP có một số lợi thế:

1. **Tạo nội dung động:** JSP có thể nhúng mã Java vào trang HTML, cho phép xử lý logic và truy vấn cơ sở dữ liệu để tạo ra nội dung động, trong khi HTML chỉ có thể hiển thị nội dung tĩnh.
2. **Tích hợp dễ dàng với Java:** JSP tích hợp trực tiếp với các công nghệ Java như **Servlet**, **JDBC**.
3. **Quản lý dễ dàng:** Tách biệt giữa giao diện (HTML) và logic (Java), giúp bảo trì dễ dàng hơn.
4. **Tái sử dụng code:** Sử dụng **taglibs** và **JSP includes** để tái sử dụng các đoạn mã.

2. EL và JSTL

EL giúp truy xuất dữ liệu từ các JavaBeans, request, session, hoặc application mà không cần mã Java phức tạp.

```
<p>Hello, ${user.name}!</p> <!-- Truy xuất thuộc tính "name" của đối tượng "user" -->
```

Expression Language (EL) phổ biến:

Truy cập thuộc tính JavaBean: `${user.name}` - Lấy thuộc tính name của đối tượng user.

Truy cập đối tượng phạm vi: `${sessionScope.cart}` - Lấy đối tượng cart từ sessionScope.

Toán tử logic: $\${user.age > 18 ? 'Adult' : 'Minor'}$ - Biểu thức điều kiện.

JSTL (Jakarta Standard Tag Library): JSTL cung cấp một bộ thẻ chuẩn cho các tác vụ như điều khiển luồng, xử lý biểu thức, ...

```
<!-- Thiết lập một biến trong JSP -->
<c:set var="total" value="100"/>

<!-- Kiểm tra điều kiện với thẻ <c:if> -->
<c:if test="${total > 50}">
  <p>Tổng lớn hơn 50: ${total}</p>
</c:if>

<!-- Sử dụng <c:forEach> để lặp qua một danh sách sản phẩm -->
<ul>
  <c:forEach var="product" items="${productList}">
    <li>${product.name} - ${product.price}</li>
  </c:forEach>
</ul>

<!-- Kiểm tra điều kiện phức tạp với toán tử trong EL -->
<c:choose>
  <c:when test="${user.age >= 18}">
    <p>Người dùng đã đủ tuổi: ${user.name}</p>
  </c:when>
  <c:otherwise>
    <p>Người dùng chưa đủ tuổi: ${user.name}</p>
  </c:otherwise>
</c:choose>
```

IV. Framework Bootstrap dùng cho Frontend

(Bài viết này chỉ đề cập đến những phần được sử dụng trong project)

1. Hệ thống 12-grid system với Bootstrap 5.3

Grid system trong Bootstrap 5.3 cho phép bạn xây dựng bố cục linh hoạt và đáp ứng trên nhiều kích cỡ màn hình khác nhau, sử dụng hệ thống lưới 12 cột. Bạn có thể tạo bố cục bằng các lớp CSS như `.container`, `.row`, và `.col`. Một số tính năng chính:

1. **Responsive:** Tự động thay đổi bố cục dựa trên kích cỡ màn hình (xs, sm, md, lg, xl, xxl).
2. **Gutter (khoảng cách giữa các cột):** Dùng để tạo khoảng cách giữa các cột.
3. **Breakpoint:** Cho phép kiểm soát layout theo từng kích thước cụ thể.

Dưới đây là một ví dụ:

```

<div class="container text-center">
  <div class="row">
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
    <div class="col">
      Column
    </div>
  </div>
</div>

```



Ở ví dụ trên, mỗi cột chiếm 4 đơn vị (ta sẽ có $4 \times 3 = 12$) và được như trên.

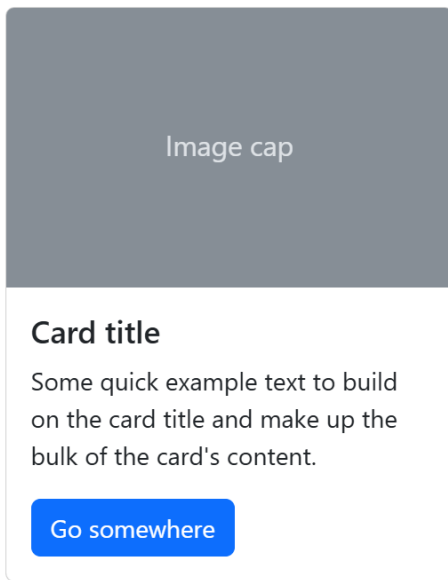
Bootstrap Grid System giúp dễ dàng tạo bố cục linh hoạt và đáp ứng cho các ứng dụng web hiện đại.

2. Card

Trong **Bootstrap**, **card** là một thành phần mạnh mẽ để hiển thị thông tin có định dạng hộp (box), thường sử dụng cho các bố cục có hình ảnh, văn bản và các nút hành động.

Các thành phần chính của một card:

- **.card-body:** Chứa nội dung chính của card.
- **.card-title** và **.card-text:** Hiển thị tiêu đề và nội dung.
- **.card-img-top:** Thêm hình ảnh ở phía trên.
- **.card-footer:** Thêm thông tin ở cuối.



```
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text to build on the card title and
make up the bulk of the card's content.</p>
    <a href="#" class="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Trong dự án này, Card được sử dụng để hiển thị nhanh thông tin sản phẩm, như ảnh sản phẩm, Title là tên sản phẩm, card-text là giá và có 2 tùy chọn: xem chi tiết hoặc thêm vào giỏ hàng.

3. Modal

Modal trong Bootstrap là một thành phần giao diện (UI component) hiển thị các cửa sổ hộp thoại nổi bật lên trên nội dung trang. Nó có thể chứa các thông báo, biểu mẫu, hoặc nút hành động mà không cần tải lại trang.

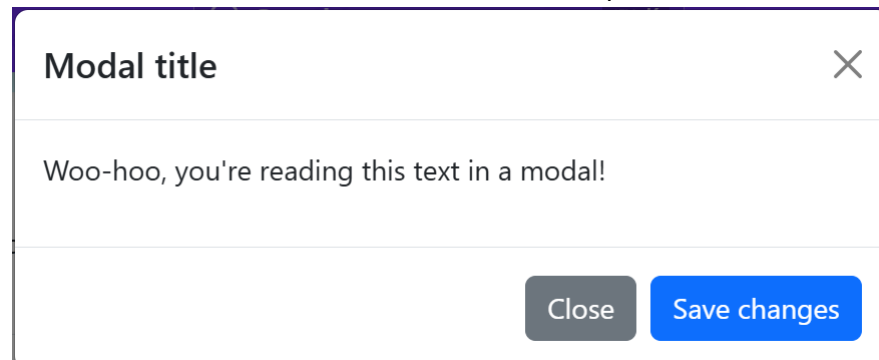
Dưới đây là một ví dụ cơ bản về Modal

Launch demo modal

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#exampleModal">
  Launch demo modal
</button>

<!-- Modal -->
<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModallabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h1 class="modal-title fs-5" id="exampleModallabel">Modal title</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        ...
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```

Khi ấn vào nút “Launch demo modal”, một hộp thoại sẽ xuất hiện:



Các thành phần chính:

- **.modal**: Container chính của modal.
- **.modal-dialog**: Điều chỉnh kích thước và bố cục của modal.
- **.modal-content**: Chứa nội dung chính của modal.
- **.modal-header, .modal-body, .modal-footer**: Các phần tử tương ứng để chứa tiêu đề, nội dung và chân trang.

Tính năng:

- Modal tự động xuất hiện khi người dùng nhấp vào nút mở modal.
- Có thể đóng modal bằng cách nhấn nút đóng hoặc nhấp ra ngoài khu vực modal.

Trong dự án này, Modal được sử dụng làm các module để CRUD sản phẩm, nhãn hiệu, danh mục, tài khoản người dùng trên trang mà không cần phải sang trang khác, tăng trải nghiệm sử dụng.

V. Javascript với jQuery và AJAX

1. jQuery, AJAX và JSON là gì ?

a. jQuery:

- **jQuery** là một thư viện JavaScript giúp đơn giản hóa việc xử lý DOM, sự kiện, và hiệu ứng.
- Ưu điểm chính là cú pháp ngắn gọn, dễ sử dụng, và tương thích tốt với các trình duyệt khác nhau.

b. AJAX:

- **AJAX** (Asynchronous JavaScript and XML) là kỹ thuật cho phép gửi và nhận dữ liệu từ server mà không cần tải lại toàn bộ trang.
- Thường được sử dụng để cập nhật một phần trang web mà không gây gián đoạn trải nghiệm người dùng.

c. JSON:

- **JSON** (JavaScript Object Notation) là định dạng dữ liệu nhẹ để trao đổi thông tin giữa client và server, dễ đọc và phân tích bằng JavaScript.

2. Tại sao nên dùng jQuery để xử lý form với AJAX

Không cần tải lại trang: jQuery kết hợp với AJAX giúp gửi dữ liệu form tới server và nhận phản hồi mà không cần reload trang, cải thiện trải nghiệm người dùng.

Đơn giản hóa thao tác DOM: jQuery giúp dễ dàng thao tác với các thành phần của form như lấy dữ liệu input, kiểm tra hợp lệ, và gửi dữ liệu.

Xử lý sự kiện dễ dàng: jQuery hỗ trợ bắt sự kiện form như submit và xử lý các callback sau khi nhận dữ liệu từ server (success, error).

Trong dự án này, AJAX được sử dụng để handle form đăng nhập thay vì gửi dữ liệu lên server và xử lý bằng doPost (khiến cho trang phải reload lại). Nó cũng được dùng cho các hoạt động tương tác với giỏ hàng của người dùng mà không cần tải lại trang, giúp người dùng có trải nghiệm mượt mà và nhanh chóng hơn.

3. Ví dụ đơn giản về sử dụng AJAX để xử lý form

<h1>Gửi Dữ Liệu Đến Servlet</h1>

```
<!-- Form -->
<form id="dataForm">
    <label for="inputData">Nhập dữ liệu:</label>
    <input type="text" id="inputData" name="inputData" required>
    <button type="submit">Gửi</button>
</form>
<!-- Kết quả phản hồi -->
<div id="response"></div>
<script>
    $(document).ready(function () {
        // Lắng nghe sự kiện submit của form
        $('#dataForm').on('submit', function (event) {
            event.preventDefault(); // Ngăn chặn việc reload trang
            // AJAX call
            $.ajax({
                url: 'process-data', // URL servlet
                type: 'POST', // Phương thức POST
                data: $(this).serialize(), // Lấy toàn bộ dữ liệu form
                success: function (response) {
                    // Xử lý phản hồi JSON từ servlet
                    $('#response').html(
                        '<p>Trạng thái: ' + response.status + '</p>' +
                        '<p>Thông báo: ' + response.message + '</p>' +
                        '<p>Dữ liệu nhận được: ' + response.inputData + '</p>'
                    );
                },
                error: function (error) {
                    // Xử lý khi có lỗi xảy ra
                    $('#response').html('<p>Đã xảy ra lỗi: ' + error + '</p>');
                }
            });
        });
    });
</script>
```

Xử lý back-end servlet , ProcessData.java


```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    // Lấy dữ liệu từ frontend (client)
    String inputData = request.getParameter("inputData");
    // Xử lý logic tại đây (ví dụ: lưu vào database, tính toán,...)
    // Chuẩn bị phản hồi
    JSONObject jsonResponse = new JSONObject();
    jsonResponse.put("status", "success");
    jsonResponse.put("message", "Data processed successfully");
    jsonResponse.put("inputData", inputData);

    // Đặt kiểu dữ liệu phản hồi là JSON
    response.setContentType("application/json");
    response.setCharacterEncoding("UTF-8");

    // Gửi phản hồi JSON về cho client
    response.getWriter().write(jsonResponse.toString());
}

```

Gửi Dữ Liệu Đến Servlet

Nhập dữ liệu:

Trạng thái: success

Thông báo: Data processed successfully

Dữ liệu nhận được: Đây là dữ liệu đã được gửi

Ta có thể ấn nút “gửi” dữ liệu nhận được từ server đã thông qua Javascript xử lí và không phải load lại trang

serialize() giúp lấy toàn bộ dữ liệu form dưới dạng chuỗi truy vấn (query string) để gửi lên server một cách dễ dàng.

Dữ liệu ở đây được gửi đến server dưới dạng **query string**, nhờ sử dụng phương thức `.serialize()` của jQuery. Điều này có nghĩa là dữ liệu sẽ được gửi dưới dạng cặp key-value giống như trong URL với phương thức GET, nhưng ở đây sử dụng POST.

Server gửi dữ liệu về dưới dạng JSON, Trong ví dụ trên server gửi về { "status": "success", "message": "Data processed successfully" }

PHẦN II: KHẢO SÁT, PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

I. Khảo sát

1. Giới thiệu website Laptopshop

LaptopShop là một trang web thương mại điện tử đơn giản dành cho việc bán máy tính xách tay trực tuyến. Người dùng có thể duyệt các sản phẩm, tạo tài khoản, thêm sản phẩm vào giỏ hàng, và thực hiện thanh toán. Hệ thống cũng có các chức năng quản lý tài khoản người dùng, đánh giá sản phẩm, và theo dõi đơn hàng. Quản trị viên có thể kiểm soát các sản phẩm, người dùng và theo dõi tình trạng đơn hàng. Giao diện đơn giản, thân thiện, đáp ứng nhu cầu mua sắm trực tuyến.

2. Đánh giá

Ưu điểm:

- **Giao diện thân thiện:** Giao diện đơn giản, dễ sử dụng, phù hợp với người dùng mua sắm máy tính xách tay.
- **Chức năng đa dạng:** Hỗ trợ các tính năng như giỏ hàng, giỏ hàng được lưu vào trong CSDL nên có thể truy cập bất cứ lúc nào, trên mọi thiết bị, theo dõi đơn hàng, quản lý tài khoản.
- **Phân quyền:** Cho phép quản trị viên quản lý sản phẩm và người dùng dễ dàng.

Nhược điểm:

- **Chưa tối ưu hiệu suất:** Website có thể chưa tối ưu cho khối lượng lớn người dùng cùng lúc.
- **Tính năng bảo mật:** Có thể cần cải thiện thêm bảo mật trong giao dịch và quản lý dữ liệu khách hàng.

3. Mục đích của website

- **Mua bán sản phẩm:** Cung cấp nền tảng để khách hàng mua các mẫu máy tính xách tay đa dạng, ngoài ra còn có các sản phẩm khác như phụ kiện, linh kiện máy tính, TV,...
- **Thúc đẩy nhu cầu mua hàng online:** Phát triển logistics, mua hàng mọi lúc mọi nơi, không cần phải đến tận cửa hàng

II. Phân tích và thiết kế hệ thống

1. Yêu cầu của khách hàng về website

Về phía người dùng:

- Đăng nhập, đăng kí tài khoản (nếu quên mật khẩu vui lòng liên hệ với admin)
- Trình bày sản phẩm dưới dạng Card để hiển thị các thông tin cần thiết, có 2 lựa chọn: thêm vào giỏ hàng, hoặc xem chi tiết.
- Cho phép người dùng lọc sản phẩm theo các tiêu chí: tầm giá, theo nhãn hiệu, theo danh mục, sắp xếp theo giá tăng dần giảm dần,...
- Giỏ hàng có thể truy cập bất kì lúc nào, trên bất kì thiết bị nào, có thể chọn sản phẩm nhất định để tiến hành thanh toán.
- Có thể theo dõi hành trình đơn hàng, lựa chọn nhận hàng hoặc huỷ đơn hàng.
- Có thể gửi feedback về cho quản trị viên

Về phía Admin:

- CRUD với các đối tượng: danh mục, nhãn hiệu, đơn hàng, sản phẩm, có thể thêm cả 3 loại tài khoản: Admin, User hoặc Shipper.
- Có thể theo dõi được tất cả các feedback, tất cả các đơn hàng, chi tiết đơn hàng, ...

Về phía Shipper (có thể là một người giao hàng, hoặc một dịch vụ giao hàng):

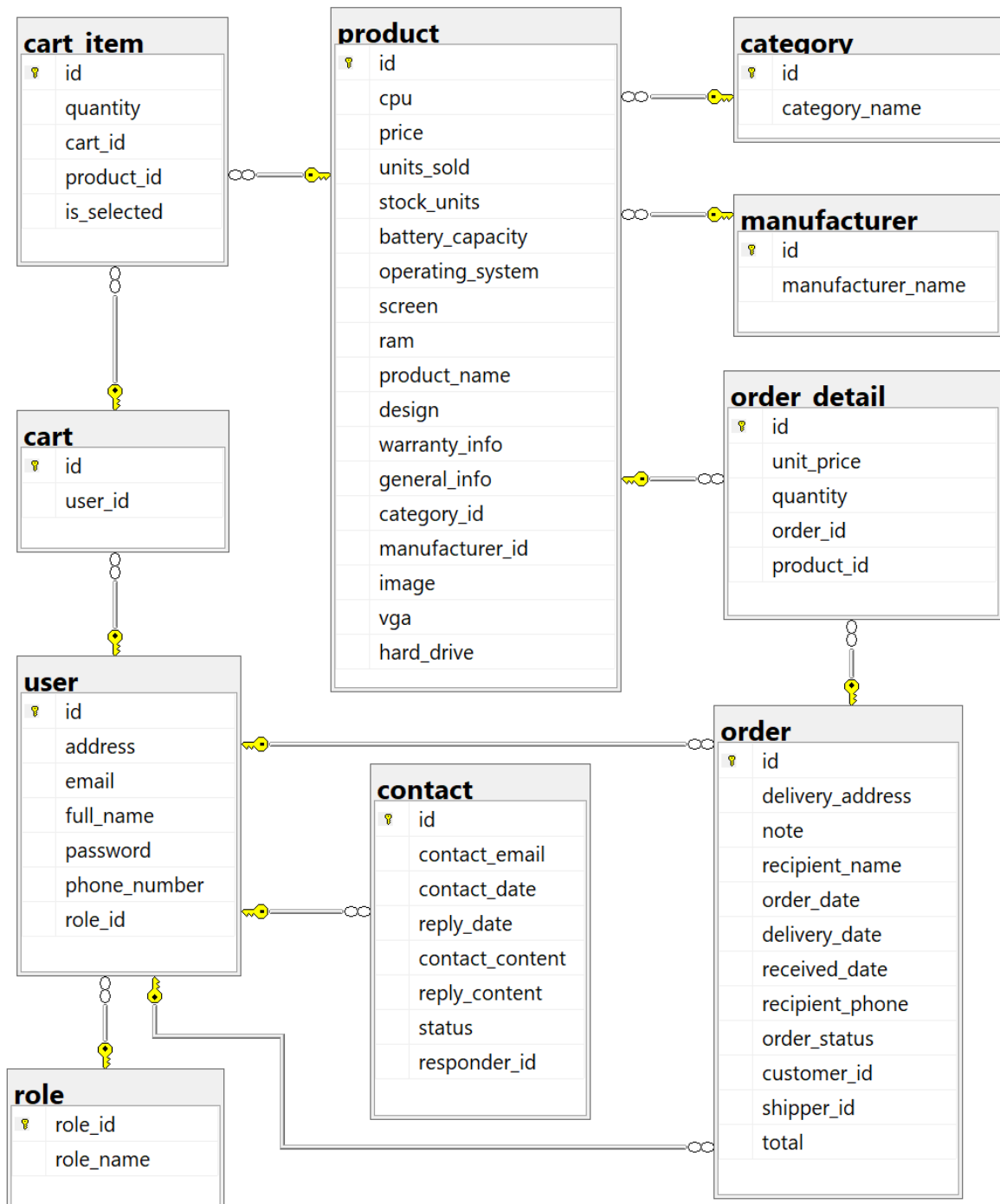
- Có thể quản lí đơn hàng đã được phân công.
- Có thể cập nhật tình hình đơn hàng, ghi chú thông tin chuyển về cho Admin.

2. Phân tích, thiết kế cơ sở dữ liệu

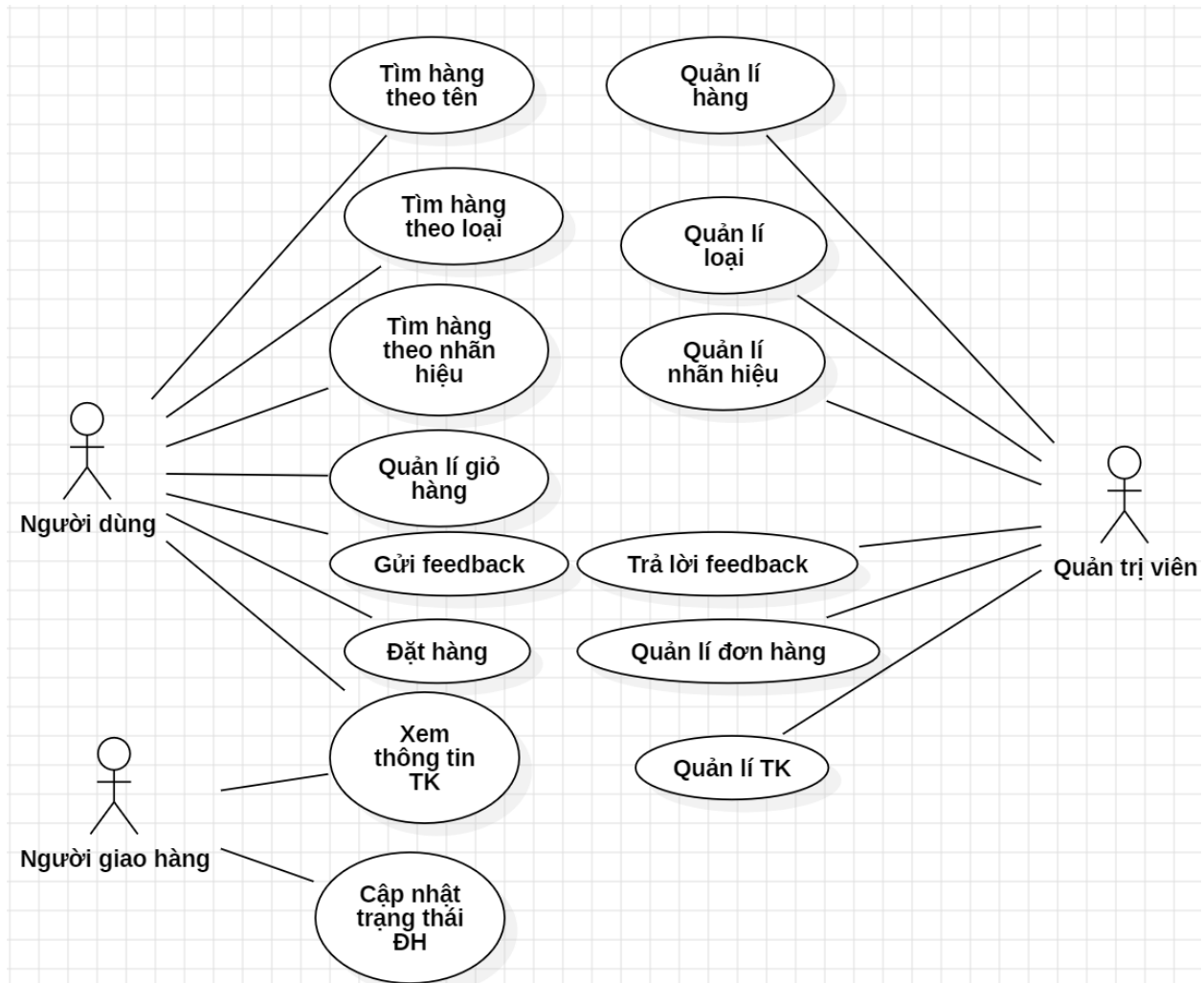
a. Cơ sở dữ liệu sẽ gồm các bảng sau:

- **User**(id, address, email, full_name, password, phone_number, role_id).
- **Role**(role_id, role_name).
- **Contact**(id, contact_email, contact_date, reply_date, contact_content, reply_content, status, responder_id).
- **Cart**(id, user_id).
- **Category**(id, category_name).
- **Manufacturer**(id, manufacturer_name).
- **Product**(id, cpu, price, units_sold, stock_units, battery_capacity, operating_system, screen, ram, product_name, design, warranty_info, general_info, category_id, manufacturer_id, image, vga, hard_drive).
- **Order**(id, delivery_address, note, recipient_name, order_date, delivery_date, received_date, recipient_phone, order_status, customer_id, shipper_id, total).
- Giữa Cart và Product là mối quan hệ nhiều-nhiều(do một sản phẩm có thể có trong nhiều giỏ hàng, và ngược lại), nên sinh ra một bảng mới **Cart_Item**(id, quantity, cart_id, product_id, is_selected).
- Giữa product và order là mối quan hệ nhiều – nhiều, nên sinh ra một bảng mới **Order_Detail**(id, unit_price, quantity, order_id, product_id).

Dưới đây là Database Diagram của dự án này:



3. Sơ đồ Use Case



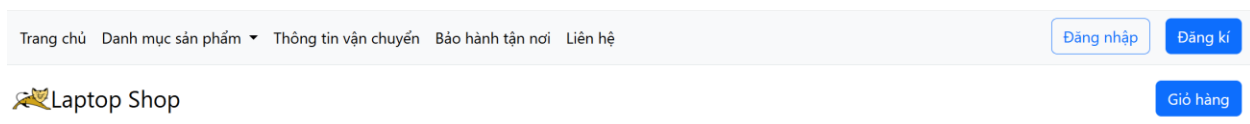
PHẦN III: THIẾT KẾ WEBSITE

I. Thiết kế giao diện phía User(người dùng)

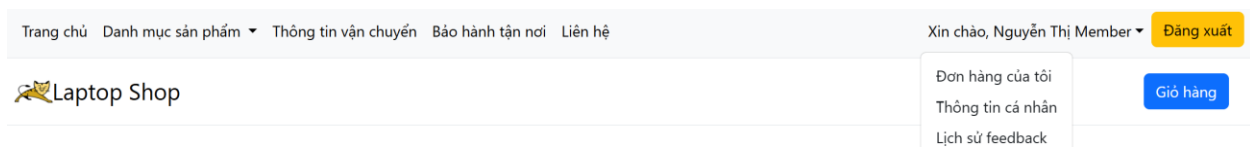
1. Thiết kế trang home gồm: header, footer và các banner:

a. Header:

Khi chưa đăng nhập:



Khi đã đăng nhập:



Khi đã đăng nhập người dùng mới có thể sử dụng các chức năng chính như theo dõi đơn hàng, chỉnh sửa thông tin cá nhân, xem đơn feedback.

b. Carousel:




c. Người dùng có thể xem sản phẩm dưới dạng các Card chứa thông tin, trình bày website sử dụng 12-grid system với Bootstrap:




2. Hiển thị, sắp xếp, lọc sản phẩm theo tiêu chí

a. Hiển thị sản phẩm dưới dạng Card


Back to school 2024




R7-5700U | AMD Radeon
16GB | 512GB | 14" Full HD
Laptop Acer Aspire 3
Giá sốc: 11.990.000 VNĐ
[Xem](#) [Giỏ hàng](#)



R5-7235HS | RTX 3050
12GB | 512GB | 15.6" Full HD
Laptop Lenovo LOQ 15ARP9
Giá sốc: 21.390.000 VNĐ
[Xem](#) [Giỏ hàng](#)



i5-12450H | Intel UHD
16GB | 512GB | 14" WUXGA
Laptop Lenovo Ideapad Slim 5
Giá sốc: 15.990.000 VNĐ
[Xem](#) [Giỏ hàng](#)



i5-12500H | RTX 3050
16GB | 512GB | 16" WUXGA
ASUS Gaming VivoBook
Giá sốc: 19.290.000 VNĐ
[Xem](#) [Giỏ hàng](#)

Card là một thành phần gồm ảnh, title và content (content ở đây là 2 nút: xem sản phẩm, hoặc thêm sản phẩm vào giỏ hàng). Ta chỉ cần định nghĩa một file Card.jsp, chỉ cần thay đổi nội dung trong card, ta có thể hiển thị các sản phẩm khác nhau (như hình).

b. Sắp xếp sản phẩm theo các tiêu chí:

Theo danh mục:

Laptop, PC, máy đồng bộ

Theo nhãn hiệu:

Tất cả các nhãn hiệu

Theo tầm giá:

Tất cả tầm giá

Nhập từ khóa

Tìm kiếm theo tên

Theo tầm giá:

Sắp xếp theo mức giá

3. Giao diện giỏ hàng, thanh toán:

a. Giao diện giỏ


Giỏ hàng

<input type="checkbox"/>	Tên sản phẩm	Đơn giá	Số lượng	Tổng	
<input checked="" type="checkbox"/>	Macbook Pro 14 M3 Pro 36GB - 512GB	58.590.000 VNĐ	<input type="text" value="1"/>	58.590.000 VNĐ	Xoá
<input type="checkbox"/>	Laptop Lenovo LOQ 15ARP9	21.390.000 VNĐ	<input type="text" value="1"/>	21.390.000 VNĐ	Xoá
<input type="checkbox"/>	Laptop gaming Acer Nitro 16 Phoenix	40.000.000 VNĐ	<input type="text" value="1"/>	40.000.000 VNĐ	Xoá
<input checked="" type="checkbox"/>	ASUS TUF Gaming F15 FX507ZC4-HN095W	20.290.000 VNĐ	<input type="text" value="1"/>	20.290.000 VNĐ	Xoá
<input checked="" type="checkbox"/>	MacBook Pro 14 M3 Pro	49.190.000 VNĐ	<input type="text" value="1"/>	49.190.000 VNĐ	Xoá
Mua thêm sản phẩm khác		Thanh toán	Thành tiền: 128.070.000 VNĐ		

Giao diện giỏ hàng gồm tên sản phẩm, đơn giá, số lượng, và thành tiền. Người dùng có thể thêm sửa xoá sản phẩm trong giỏ hàng, lựa chọn sản phẩm để thanh toán hoặc mua sản phẩm khác.

b. Thanh toán:

Sau khi đã chọn những sản phẩm cần mua, người dùng sẽ được dẫn đến giao diện thanh toán:

 Laptop Shop Giỏ hàng

Thông tin đặt hàng

Tên người nhận

Đoàn Kiến Quốc

Địa chỉ nhận hàng

Số 67, đường Phạm Hùng, Mỹ Đình, Nam Từ Liêm, HN

Số điện thoại người nhận

0359462146

Lưu ý dành cho người giao hàng

Chỉ giao hàng vào giờ hành chính

Tiến hành đặt hàng

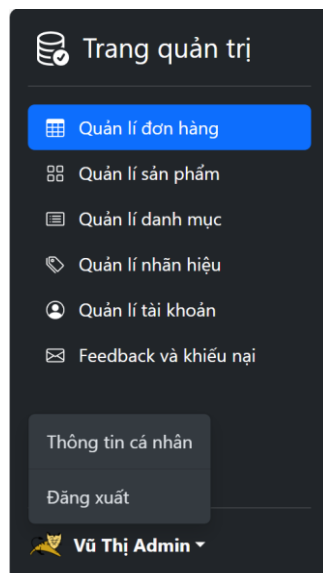
Giỏ hàng

Macbook Pro 14 M3 Pro 36GB - 512GB	58.590.000 VNĐ
Số lượng: 1	
ASUS TUF Gaming F15 FX507ZC4-HN095W	20.290.000 VNĐ
Số lượng: 1	
MacBook Pro 14 M3 Pro	49.190.000 VNĐ
Số lượng: 1	
Tổng:	128.070.000 VNĐ

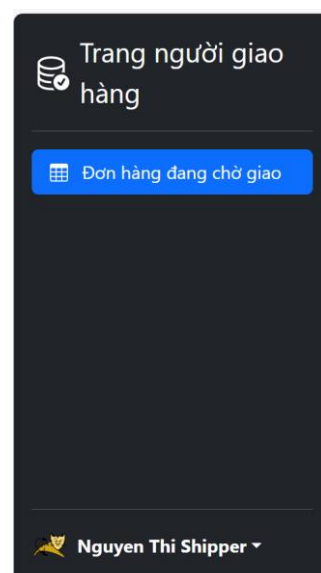
II. Thiết kế giao diện Admin và Shipper

1. Phần sidebar

Về cơ bản, giao diện của Admin và Shipper sẽ giống nhau, đều có bố cục: sidebar bên trái, phần chính ở bên phải. Admin sẽ có các menu : quản lý đơn hàng, sản phẩm, danh mục, nhãn hiệu, tài khoản, feedback, nhấn vào phần avatar để quản lý tài khoản của mình.



Sidebar trang quản trị



Sidebar trang cho shipper

2. Thiết kế trang giao diện quản lý

a. Hiển thị dưới dạng bảng:

Theo danh mục:







Laptop, PC, máy đồng bộ

Theo nhãn hiệu:

Tất cả các nhãn hiệu

Theo tầm giá:

Tất cả tầm giá

Hình ảnh	Tên SP	Danh mục	Hãng sản xuất	Đơn giá	Số lượng			
	Apple MacBook Air M2 2022 8GB 256GB	Laptop, PC, máy đồng bộ	Apple	23.590.000 VNĐ	20	Chi tiết	Cập nhật	Xóa
	MacBook Pro 14 M3 Pro	Laptop, PC, máy đồng bộ	Apple	49.190.000 VNĐ	20	Chi tiết	Cập nhật	Xóa
	Macbook Pro 14 M3 Pro 36GB - 512GB	Laptop, PC, máy đồng bộ	Apple	58.590.000 VNĐ	20	Chi tiết	Cập nhật	Xóa
	Laptop Acer Aspire 3	Laptop, PC, máy đồng bộ	Acer	11.990.000 VNĐ	20	Chi tiết	Cập nhật	Xóa
	Laptop Lenovo LOQ 15ARP9	Laptop, PC, máy đồng bộ	Lenovo	21.390.000 VNĐ	20	Chi tiết	Cập nhật	Xóa
	Laptop gaming Acer Nitro 16 Phoenix	Laptop, PC, máy đồng bộ	Acer	40.000.000 VNĐ	20	Chi tiết	Cập nhật	Xóa

[Thêm mới PC, Laptop](#)[Thêm mới sản phẩm khác](#)

Previous

1

2

3

Next

Trang quản lí sản phẩm

Trang quản lí tài khoản

Loại tài khoản

Tất cả

ID	Họ và tên	Địa chỉ	Email	Số điện thoại	Vai trò	
1	Vũ Thị Admin	Hà Nội	admin@gmail.com	012345678	Admin	Xóa
2	Nguyen Thi Shipper	Hà Nội	shipper@gmail.com	123456	Shipper	Xóa
3	Bưu điện Việt Nam	Hà Nội	VNPost@gmail.com	1234567890	Shipper	Xóa
4	Nguyễn Thị Member	19, 190 Nguyen Du, Quỳnh Côi Town	member@gmail.com	123456789	Customer	Xóa
5	dũng	hà nội	dung@gmail.com	123456789	Admin	Xóa

[Thêm mới người dùng](#)

Previous

1

Next

Trang quản lí tài khoản

b. Các thao tác CRUD, (chi tiết, cập nhật, xóa) được tích hợp vào modal

Cập nhật

X

Tên sản phẩm

Đơn giá

Apple MacBook Air M2 2022 8GB 256GB

23590000

Hãng sản xuất

CPU

RAM

VGA

Apple

Apple M2

8GB

8 nhân GPU, 16 nhân Neural Engi

Lưu trữ

Màn hình

Thiết kế

Hệ điều hành

256GB (M.2 NVMe)

13.6 inches 2560 x 1664 pixels

Nhôm

Mac OS

Thông tin chung

Macbook Air 2022 - Ví xử lý Apple M2 mạnh mẽ

Sau thành công của dòng Macbook M1 thì Apple lại chuẩn bị mang đến cho người dùng dòng sản phẩm Macbook Air 2022 với những điểm nâng cấp đáng chú ý. Bên cạnh đó mức giá thành lại thấp hơn so với hiện tại, chắc chắn rằng các fan đang rất nóng lòng chờ đón sự xuất hiện của dòng sản phẩm mới này.

Dung lượng pin

Bảo hành

Đơn vị kho

52,6 Wh

3 năm

20


Hình ảnh

Choose File

No file chosen

Đóng

Lưu thay đổi



Apple M2 | 8 nhân GPU

8GB | 256GB | 13.6" 2K

Mã sản phẩm: 1

Tên sản phẩm: Apple MacBook Air M2 2022 8GB 256GB

Hãng sản xuất: Apple

CPU: Apple M2

RAM: 8GB

VGA: 8 nhân GPU, 16 nhân Neural Engine

Lưu trữ: 256GB (M.2 NVMe)

Thiết kế: Nhôm

Hệ điều hành: Mac OS

Màn hình: 13.6 inches 2560 x 1664 pixels

Dung lượng pin: 52,6 Wh

Thông tin chung: Macbook Air 2022 - Vì xử lý Apple M2 mạnh mẽ Sau thành công của dòng Macbook M1 thì Apple lại chuẩn bị mang đến cho người dùng dòng sản phẩm Macbook Air 2022 với những điểm nâng cấp đáng chú ý. Bên cạnh đó mức giá thành lại thấp hơn so với hiện tại, chắc chắn rằng các iFan đang rất nóng lòng chờ đón sự xuất hiện của dòng sản phẩm mới này.

Bảo hành: 3 năm

Đơn giá: 23.590.000 VNĐ

Kho: 20

Đã bán: 0

Chi tiết đơn hàng

Thông tin người nhận:

Người nhận: Đoàn Kiến Quốc

Địa chỉ: Số 67, đường Phạm Hùng, Mỹ Đình, Nam Từ Liêm, HN

Số điện thoại: 0359462146

Thông tin người đặt hàng:

Người đặt: Nguyễn Thị Member

Email: member@gmail.com

Số điện thoại: 123456789

Thông tin đơn hàng:

Trạng thái đơn hàng: Shipping

Ngày đặt: 2024-10-20 16:42:17.94

Ngày giao: 2024-10-20 18:50:32.61

Ngày nhận:

STT	Tên sản phẩm	Đơn giá	Số lượng đặt
1	Macbook Pro 14 M3 Pro 36GB - 512GB	58590000	1
2	ASUS TUF Gaming F15 FX507ZC4-HN095W	20290000	1
3	MacBook Pro 14 M3 Pro	49190000	1
Tổng			128070000

Ghi chú

Chỉ giao hàng vào giờ hành chính

Thông tin người giao hàng:

Tên: Nguyen Thi Shipper

Id: 2

Số điện thoại: 123456

Đóng

Modal chi tiết đơn hàng

Cập nhật trạng thái đơn hàng

STT	Tên sản phẩm	Đơn giá	Số lượng đặt
1	Macbook Pro 14 M3 Pro 36GB - 512GB	58590000	1
2	ASUS TUF Gaming F15 FX507ZC4-HN095W	20290000	1
3	MacBook Pro 14 M3 Pro	49190000	1
Tổng			128070000

Ghi chú thêm:

Giao hàng thành công

Giao hàng thành công

Giao hàng thất bại

Đóng

Modal cập nhật trạng thái đơn hàng (trang shipper)

III. Kỹ thuật phân trang sử dụng SQL Server và Servlet
(Tham khảo: [Kỹ thuật phân trang với PHP và MySQL](#))

1. Tại sao phải phân trang ?

a. Phân trang là một phần quan trọng với bất cứ ngôn ngữ lập trình web nào. Phân trang giúp giảm thời gian tải cho website, nâng cao hiệu suất. Trong dự án này, chúng ta sẽ phân trang bằng SQL Server với câu lệnh OFFSET và FETCH NEXT ? ROWS ONLY

2. Công thức phân trang và cách áp dụng:

a. Các tham số cần phải có:

- recordsPerPage(số bản ghi mỗi trang, chọn một số phù hợp)
- currentPage(trang hiện tại, lấy từ tham số trang từ request.getParameter(), nếu trang load lần đầu, tự động là 1)
- offset (bỏ qua bao nhiêu bản ghi). Công thức: **offset = (page - 1) * recordsPerPage;**
- totalRecords(tổng số bản ghi) – truy vấn từ SQL
- totalPages(tổng số trang) **totalPages = (int) Math.ceil(totalRecords * 1.0 / recordsPerPage);**

b. Trong thực tế, việc phân trang có thể kết hợp với các điều kiện lọc khác (khoảng giá, nhãn hiệu, danh mục) , ví dụ, ở trong dự án này, chúng ta sẽ phân trang đếm tổng số bản ghi kết hợp với việc lọc như sau:

```

public int countTotalRecords(int brandId, int categoryId,
    String priceRange, String searchKey) {
    String sql = ""
        SELECT COUNT(*)
        FROM [LaptopShop].[dbo].[product]
        WHERE 1=1
        "";
    int count = 0;
    String range = getPriceRange(priceRange);
    if (brandId != 0) {
        sql += " AND [manufacturer_id] = " + brandId;
    }
    if (categoryId != 0) {
        sql += " AND [category_id] = " + categoryId;
    }
    if (searchKey != null && !searchKey.isEmpty()) {
        sql += " AND [product_name] LIKE '%" + searchKey + "%'";
    }
    sql += range;
    try {
        PreparedStatement st = connection.prepareStatement(sql);
        ResultSet rs = st.executeQuery();
        if (rs.next()) {
            count = rs.getInt(1);
        }
    } catch (SQLException ex) {
        Logger.getLogger(ProductDAO.class.getName()).log(Level.SEVERE, null, ex);
    }
    return count;
}

```

Hàm đếm tổng số bản ghi

```

public List<Product> getProducts(int brandId, int categoryId, int offset,
int recordsPerPage, String priceRange,
String searchKey, String sortOrder) {
    List<Product> listProduct = new ArrayList<>();
    String sql = ""
        SELECT *
        FROM [LaptopShop].[dbo].[product]
        WHERE 1=1
        "";

    // Tìm kiếm theo brandId nếu có
    if (brandId != 0) {
        sql += " AND [manufacturer_id] = " + brandId;
    }
    // Tìm kiếm theo categoryId nếu có
    if (categoryId != 0) {
        sql += " AND [category_id] = " + categoryId;
    }
    // Tìm kiếm theo tên sản phẩm nếu có
    if (searchKey != null && !searchKey.isEmpty()) {
        sql += " AND [product_name] LIKE '%" + searchKey + "%'";
    }
    // Thêm điều kiện lọc giá nếu có
    sql += getPriceRange(priceRange);
    // Sắp xếp giá theo thứ tự tăng hoặc giảm
    if ("asc".equalsIgnoreCase(sortOrder)) {
        sql += " ORDER BY [price] ASC ";
    } else if ("desc".equalsIgnoreCase(sortOrder)) {
        sql += " ORDER BY [price] DESC ";
    } else {
        sql += " ORDER BY [id]"; // Mặc định nếu không chỉ định sắp xếp
    }
    // Phân trang
    String pagination = ""
        OFFSET ? ROWS
        FETCH NEXT ? ROWS ONLY;
        "";
    sql += pagination;
    // Dưới đây là các câu truy vấn SQL để lấy dữ liệu ..., rồi thêm vào
listProduct
    // (...)
    return listProduct;
}

```

Hàm truy vấn trả về danh sách các sản phẩm, kết hợp phân trang và các tiêu chí lọc

```

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    CategoryDAO cdao = new CategoryDAO();
    BrandDAO bdao = new BrandDAO();
    ProductDAO pdao = new ProductDAO();
    List<Category> categoryList = cdao.getAllCategories();
    List<Brand> brandList = bdao.getAllBrands();
    String brandSelect = request.getParameter("brandSelect");
    String categorySelect = request.getParameter("categorySelect");
    String pageParam = request.getParameter("page");
    String priceRange = request.getParameter("priceRange");
    try {
        int page = (pageParam == null) ? 1 : Integer.parseInt(pageParam);
        int brandId = (brandSelect == null) ? 0 : Integer.parseInt(brandSelect);
        int categoryId = (categorySelect == null) ? 1 : Integer.parseInt(categorySelect);
        int recordsPerPage = 6;
        int offset = (page - 1) * recordsPerPage;
        int totalRecords = pdao.countTotalRecords(brandId, categoryId, priceRange, null);
        int totalPages = (int) Math.ceil(totalRecords * 1.0 / recordsPerPage);
        List<Product> productList = pdao.getProducts(brandId, categoryId, offset, recordsPerPage, priceRange, null, null);
        request.setAttribute("productList", productList);
        request.setAttribute("currentPage", page);
        request.setAttribute("totalPages", totalPages);
        request.setAttribute("currentBrand", brandId);
        request.setAttribute("currentCategory", categoryId);
        request.setAttribute("currentPriceRange", priceRange);
    } catch (NumberFormatException ex) {
        System.out.println(ex);
    }
    System.out.println(getServletContext().getRealPath(""));
    request.setAttribute("categoryList", categoryList);
    request.setAttribute("brandList", brandList);
    request.setAttribute("pageName", "product-manager");
    request.getRequestDispatcher("/admin/productManager.jsp").forward(request, response);
}

```

Hàm doGet dùng để phân trang trang sản phẩm kèm theo các tiêu chí search

```

<nav aria-label="Page navigation">
  <ul class="pagination justify-content-center">
    <li class="page-item ${currentPage == 1 ? 'disabled' : ''}">
      <a class="page-link" href="?page=${currentPage} -
1}&brandSelect=${currentBrand}&categorySelect=${currentCategory}&priceRange=${currentPriceRange}"
        tabindex="-1">Previous</a>
    </li>

    <c:forEach var="i" begin="1" end="${totalPages}">
      <li class="page-item ${i == currentPage ? 'active' : ''}">
        <a class="page-link"
href="?page=${i}&brandSelect=${currentBrand}&categorySelect=${currentCategory}&priceRange=${currentPriceRange}">${i}</a>
      </li>
    </c:forEach>

    <li class="page-item ${currentPage == totalPages ? 'disabled' : ''}">
      <a class="page-link" href="?page=${currentPage} +
1}&brandSelect=${currentBrand}&categorySelect=${currentCategory}&priceRange=${currentPriceRange}">Next</a>
    </li>
  </ul>
</nav>







```

Thanh phân trang

Trang quản lí sản phẩm

Theo danh mục:
Theo nhãn hiệu:
Theo tầm giá:

Laptop, PC, máy đồng bộ
Tất cả các nhãn hiệu
Tất cả tầm giá

Hình ảnh	Tên SP	Danh mục	Hãng sản xuất	Đơn giá	Số lượng	
	Apple MacBook Air M2 2022 8GB 256GB	Laptop, PC, máy đồng bộ	Apple	23.590.000 VNĐ	20	Chi tiết Cập nhật Xóa
	MacBook Pro 14 M3 Pro	Laptop, PC, máy đồng bộ	Apple	49.190.000 VNĐ	20	Chi tiết Cập nhật Xóa
	Macbook Pro 14 M3 Pro 36GB - 512GB	Laptop, PC, máy đồng bộ	Apple	58.590.000 VNĐ	20	Chi tiết Cập nhật Xóa
	Laptop Acer Aspire 3	Laptop, PC, máy đồng bộ	Acer	11.990.000 VNĐ	20	Chi tiết Cập nhật Xóa
	Laptop Lenovo LOQ 15ARP9	Laptop, PC, máy đồng bộ	Lenovo	21.390.000 VNĐ	20	Chi tiết Cập nhật Xóa
	Laptop gaming Acer Nitro 16 Phoenix	Laptop, PC, máy đồng bộ	Acer	40.000.000 VNĐ	20	Chi tiết Cập nhật Xóa

Thêm mới PC, Laptop
Thêm mới sản phẩm khác

Previous
1
2
3
Next

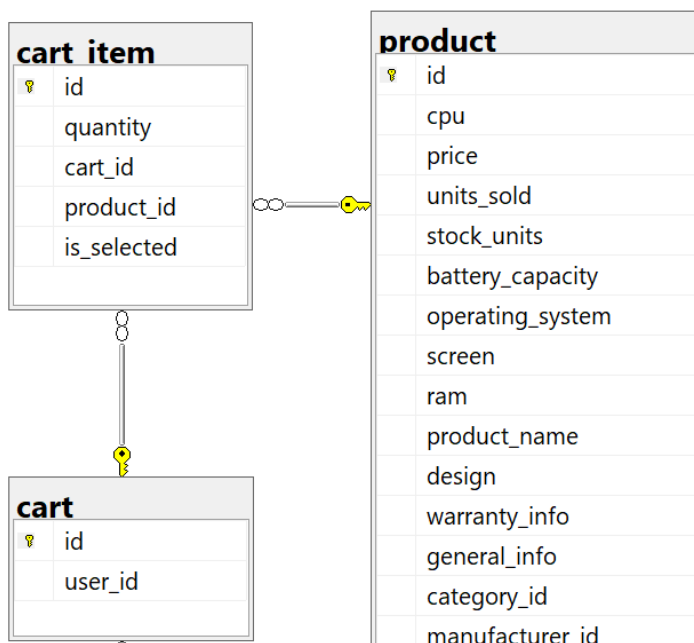
Kết quả

IV. Thiết kế giỏ hàng dùng AJAX và Database

1. Tại sao dùng AJAX ?

- a. **Tăng trải nghiệm người dùng:** AJAX cho phép cập nhật nội dung của trang mà không cần phải tải lại toàn bộ trang. Điều này giúp người dùng có trải nghiệm mượt mà và nhanh chóng hơn.
- b. **Tối ưu hoá băng thông:** Thay vì tải lại toàn bộ trang, AJAX chỉ gửi và nhận dữ liệu cần thiết. Điều này giúp giảm băng thông và tốc độ tải trang.
- c. **Giao diện tương tác hơn:** AJAX cho phép tạo ra các ứng dụng web tương tác hơn, chẳng hạn như các biểu mẫu tự động làm mới, tự động gợi ý tìm kiếm, và cập nhật dữ liệu theo thời gian thực.
- d. **Tăng hiệu suất ứng dụng:** Khi sử dụng AJAX, bạn có thể chỉ cập nhật các phần của trang mà cần thiết (như thông báo, giỏ hàng, bảng dữ liệu, v.v.), giúp ứng dụng hoạt động hiệu quả hơn.

2. Thêm sản phẩm vào giỏ hàng



Vì người dùng có thể truy cập bằng điện thoại, máy tính, mọi thiết bị, và có thể mua hàng bất kì lúc nào họ muốn, nên database được sử dụng làm nơi lưu trữ giỏ hàng.

Handle hành động thêm vào giỏ hàng của người dùng bằng AJAX:

```

$(document).ready(function () {
    $('.cart').click(function () {
        var productId = $(this).data('product-id');
        console.log(productId);

        $.ajax({
            url: 'add-to-cart',
            type: 'POST',
            data: {productId: productId},
            success: function (response) {
                alert(response.message);
            },
            error: function () {
                alert("Bạn vui lòng đăng nhập để thực hiện chức năng này");
                window.location.href = "./login";
            }
        });
    });
});

```

Thay vì gửi dữ liệu một cách thủ công lên server dùng form và phương thức doPost, AJAX sẽ tương tác với Servlet và xử lý dữ liệu mà không phải load lại trang.

localhost:8080 says

Thêm sản phẩm vào giỏ hàng thành công

OK

3. Giỏ hàng tự động cập nhật cùng với database

Thông thường, khi cập nhật giỏ hàng, chúng ta sẽ bắt sự kiện “change” của ô “input”, khi người dùng thay đổi thông tin sẽ tự động submit form để thao tác với database. Nhưng phương pháp này có nhược điểm là trang phải tải lại rất nhiều lần chỉ với một vài thao tác đơn giản.

AJAX giúp chúng ta giảm thiểu việc tải lại trang không cần thiết mà vẫn có thể kết nối đến server và thao tác với database một cách dễ dàng. Dưới đây là cách thực hiện:

Bắt sự kiện “change” của phần tử có class là “.quantity”

```
$('.quantity').change(function () {  
    var productId = $(this).data("product-id");  
    var quantity = $(this).val();  
    var quantityCell = $(this);  
    if (quantity === '0') {  
        if (window.confirm('Bạn có muốn xoá mục này ? ')) {  
            deleteAnOrder(quantityCell);  
        } else {  
            quantityCell.val('1');  
        }  
        return;  
    }  
    calculatePrices(); //Sau khi người dùng thay đổi thông tin quantity,  
    //chúng ta cần phải tính lại tổng giá tiền  
    // Phần dưới để tương tác với database  
    $.ajax({  
        url: 'update-cart',  
        type: 'POST',  
        data: {  
            action: 'change-quantity',  
            productId: productId,  
            quantity: quantity  
        },  
        success: function (response) {  
            if (response.success) {  
                console.log('done');  
            } else {  
                quantityCell.val(response.stockUnits);  
                alert(response.message);  
            }  
        },  
        error: function () {  
            alert('Lỗi kết nối đến server, vui lòng thử lại sau.');        }  
    });  
});
```

Servlet nhận thông tin từ ajax:

```
int productId = Integer.parseInt(request.getParameter("productId"));
int quantity = Integer.parseInt(request.getParameter("quantity"));
int currentStockUnits = pdao.getCurrentStockUnits(productId);
if (currentStockUnits < quantity) {
    json.put("success", false);
    json.put("message", "Số lượng sản phẩm không hợp lệ do vượt quá đơn vị kho");
    json.put("stockUnits", currentStockUnits);
    cdao.changeQuantity(quantity, userId, productId);
} else {
    cdao.changeQuantity(quantity, userId, productId);
    json.put("success", true);
}
```

Servlet gửi thông tin về dưới dạng json, nếu có lỗi xảy ra sẽ báo về phía người dùng.

Kết quả: khi người dùng cập nhật giỏ hàng:

Đơn giá	Số lượng	Tổng
21.390.000 VNĐ	<input type="text" value="3"/>	64.170.000 VNĐ
h toán		Thành tiền: 64.170.000 VNĐ

Database cũng đã được tự động cập nhật theo, mà không phải tải lại trang:

	id	quantity	cart_id	product_id	is_selected
1	20	3	1	5	1

PHẦN IV: KẾT LUẬN

Kết quả đạt được:

- **Quảng bá sản phẩm:** Hệ thống giúp cửa hàng quảng bá sản phẩm hiệu quả hơn thông qua việc tiếp cận khách hàng trực tuyến, đồng thời tạo điều kiện để khách hàng dễ dàng xây dựng và quản lý giỏ hàng. Điều này mang lại sự tiện lợi cho cả cửa hàng và khách hàng, tối ưu hóa thời gian xử lý mà không làm giảm hiệu quả.
- **Giao diện thân thiện:** Giao diện của hệ thống được thiết kế đơn giản, trực quan, theo xu hướng hiện đại, đảm bảo dễ sử dụng cho cả quản trị viên và khách hàng.

Hạn chế:

- **Thiếu tính năng và giao diện:** Mặc dù hệ thống đã hoàn thành những chức năng cơ bản, nhưng vẫn còn thiếu nhiều tính năng cần thiết cho một trang thương mại điện tử toàn diện, chẳng hạn như đánh giá đơn hàng (mới khắc phục bằng cách gửi feedback), hỗ trợ nhiều phương thức thanh toán, và các công cụ phân tích dữ liệu. Giao diện có thể cần thêm các tính năng cải tiến về thiết kế để thu hút người dùng hơn.
- **Khả năng mở rộng:** Hệ thống hiện tại chưa được tối ưu hóa cho sự mở rộng hoặc tích hợp với các hệ thống khác, điều này có thể gây khó khăn khi lượng người dùng và đơn hàng tăng lên.

Khắc phục và phát triển:

- **Hoàn thiện chức năng:** Nâng cấp và bổ sung các tính năng quan trọng còn thiếu như, thanh toán trực tuyến, hỗ trợ khách hàng, và báo cáo thống kê. Các chức năng cơ bản hiện tại cần được hoàn chỉnh để đáp ứng tốt hơn nhu cầu người dùng.
- **Tối ưu hóa hiệu năng:** Tinh chỉnh mã nguồn để cải thiện tốc độ tải trang và trải nghiệm người dùng, đặc biệt là khi hệ thống mở rộng hoặc có lượng truy cập cao.
- **Phát triển thanh toán trực tuyến:** Tích hợp các cổng thanh toán điện tử, cho phép khách hàng thực hiện giao dịch nhanh chóng và bảo mật. Cải thiện quy trình thanh toán để nâng cao sự tiện lợi.
- **Tăng cường bảo mật và khả năng chịu lỗi:** Tăng cường các biện pháp bảo mật như mã hóa dữ liệu, xác thực người dùng mạnh mẽ, và kiểm tra các lỗ hổng an ninh. Khả năng khôi phục sau lỗi hoặc khi hệ thống gặp sự cố cũng cần được cải thiện để tránh mất mát dữ liệu.
- **Mở rộng ứng dụng:** Triển khai các module hoặc plugin bổ sung, chẳng hạn như các công cụ marketing, phân tích dữ liệu, tích hợp mạng xã hội. Triển khai website trên môi trường host chính thức với tên miền phù hợp, giúp nâng cao tính chuyên nghiệp và tiếp cận người dùng dễ dàng hơn.