# Código Completo del Proyecto - Tienda Online

## Índice



## Estructura del Proyecto

```
OnlineShopProject/
├── src/
│   └── main/
│       ├── java/
│       │   └── com/
│       │       └── onlineshop/
│       │           ├── model/    # Clases de modelo
│       │           ├── dao/      # Objetos de acceso a datos
│       │           └── servlet/  # Controladores servlet
│       └── webapp/
│           ├── WEB-INF/          # Configuración web
│           ├── css/              # Hojas de estilo
│           ├── js/               # JavaScript
│           ├── images/           # Imágenes
│           │   └── products/     # Imágenes de productos
│           └── *.jsp             # Páginas JSP
└── database/                     # Scripts SQL
```

## Modelo de Datos

### User.java

```java
package com.onlineshop.model;

/**
 * User class representing customer or seller in the system
 */
public class User {
    private int userId;
    private String username;
    private String password;
    private String userType; // "customer" or "seller"
```

```
    // Default constructor
    public User() {
    }

    // Constructor with parameters
    public User(int userId, String username, String password, String userType) {
        this.userId = userId;
        this.username = username;
        this.password = password;
        this.userType = userType;
    }

    // Constructor without userId (for new user registration)
    public User(String username, String password, String userType) {
        this.username = username;
        this.password = password;
        this.userType = userType;
    }

    // Getters and setters
    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUserType() {
        return userType;
    }

    public void setUserType(String userType) {
        this.userType = userType;
    }

    public boolean isSeller() {
```

```
        return "seller".equals(userType);
    }

    @Override
    public String toString() {
        return "User [userId=" + userId + ", username=" + username + ", userType="
+ userType + "]";
    }
}
```

Product.java

```
package com.onlineshop.model;

/**
 * Product class representing items available for purchase
 */
public class Product {
    private int productId;
    private String name;
    private String description;
    private double price;
    private String imageUrl;

    // Default constructor
    public Product() {
    }

    // Constructor with parameters
    public Product(int productId, String name, String description, double price,
String imageUrl) {
        this.productId = productId;
        this.name = name;
        this.description = description;
        this.price = price;
        this.imageUrl = imageUrl;
    }

    // Constructor without productId (for new product creation)
    public Product(String name, String description, double price, String imageUrl)
{
        this.name = name;
        this.description = description;
        this.price = price;
        this.imageUrl = imageUrl;
    }

    // Getters and setters
    public int getProductId() {
        return productId;
    }
```

```
    public void setProductId(int productId) {
        this.productId = productId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public String getImageUrl() {
        return imageUrl;
    }

    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }

    @Override
    public String toString() {
        return "Product [productId=" + productId + ", name=" + name + ", price=" +
price + "]";
    }
}
```

## Order.java

```
package com.onlineshop.model;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
```

```java
/**
 * Order class representing a customer's purchase
 */
public class Order {
    private int orderId;
    private int userId;
    private Date orderDate;
    private double totalPrice;
    private double shippingCost;
    private String trackingNumber;
    private List<OrderItem> orderItems;

    // Default constructor
    public Order() {
        this.orderItems = new ArrayList<>();
    }

    // Constructor with parameters
    public Order(int orderId, int userId, Date orderDate, double totalPrice,
                 double shippingCost, String trackingNumber) {
        this.orderId = orderId;
        this.userId = userId;
        this.orderDate = orderDate;
        this.totalPrice = totalPrice;
        this.shippingCost = shippingCost;
        this.trackingNumber = trackingNumber;
        this.orderItems = new ArrayList<>();
    }

    // Constructor without orderId (for new order creation)
    public Order(int userId, Date orderDate, double totalPrice,
                 double shippingCost, String trackingNumber) {
        this.userId = userId;
        this.orderDate = orderDate;
        this.totalPrice = totalPrice;
        this.shippingCost = shippingCost;
        this.trackingNumber = trackingNumber;
        this.orderItems = new ArrayList<>();
    }

    // Getters and setters
    public int getOrderId() {
        return orderId;
    }

    public void setOrderId(int orderId) {
        this.orderId = orderId;
    }

    public int getUserId() {
        return userId;
    }
```

```
    public void setUserId(int userId) {
        this.userId = userId;
    }

    public Date getOrderDate() {
        return orderDate;
    }

    public void setOrderDate(Date orderDate) {
        this.orderDate = orderDate;
    }

    public double getTotalPrice() {
        return totalPrice;
    }

    public void setTotalPrice(double totalPrice) {
        this.totalPrice = totalPrice;
    }

    public double getShippingCost() {
        return shippingCost;
    }

    public void setShippingCost(double shippingCost) {
        this.shippingCost = shippingCost;
    }

    public String getTrackingNumber() {
        return trackingNumber;
    }

    public void setTrackingNumber(String trackingNumber) {
        this.trackingNumber = trackingNumber;
    }

    public List<OrderItem> getOrderItems() {
        return orderItems;
    }

    public void setOrderItems(List<OrderItem> orderItems) {
        this.orderItems = orderItems;
    }

    // Add an order item to this order
    public void addOrderItem(OrderItem item) {
        this.orderItems.add(item);
    }

    // Calculate shipping cost based on items (2€ + 1€ per item)
    public double calculateShippingCost() {
        int totalItems = 0;
        for (OrderItem item : orderItems) {
            totalItems += item.getQuantity();
```

```
        }
        return 2.0 + (totalItems * 1.0);
    }

    // Calculate total price of all items
    public double calculateTotalPrice() {
        double total = 0.0;
        for (OrderItem item : orderItems) {
            total += item.getItemPrice() * item.getQuantity();
        }
        return total;
    }

    // Calculate final price including shipping
    public double calculateFinalPrice() {
        return calculateTotalPrice() + calculateShippingCost();
    }

    @Override
    public String toString() {
        return "Order [orderId=" + orderId + ", userId=" + userId + ", orderDate="
+ orderDate +
                ", totalPrice=" + totalPrice + ", trackingNumber=" + trackingNumber
+ "]";
    }
}
```

## OrderItem.java

```
package com.onlineshop.model;

/**
 * OrderItem class representing a product in an order with its quantity
 */
public class OrderItem {
    private int orderItemId;
    private int orderId;
    private int productId;
    private int quantity;
    private double itemPrice;

    // For display purposes
    private String productName;

    // Default constructor
    public OrderItem() {
    }

    // Constructor with parameters
    public OrderItem(int orderItemId, int orderId, int productId, int quantity,
double itemPrice) {
```

```
        this.orderItemId = orderItemId;
        this.orderId = orderId;
        this.productId = productId;
        this.quantity = quantity;
        this.itemPrice = itemPrice;
    }

    // Constructor without orderItemId (for new order item creation)
    public OrderItem(int orderId, int productId, int quantity, double itemPrice) {
        this.orderId = orderId;
        this.productId = productId;
        this.quantity = quantity;
        this.itemPrice = itemPrice;
    }

    // Constructor for cart items (no orderId yet)
    public OrderItem(int productId, int quantity, double itemPrice, String
productName) {
        this.productId = productId;
        this.quantity = quantity;
        this.itemPrice = itemPrice;
        this.productName = productName;
    }

    // Getters and setters
    public int getOrderItemId() {
        return orderItemId;
    }

    public void setOrderItemId(int orderItemId) {
        this.orderItemId = orderItemId;
    }

    public int getOrderId() {
        return orderId;
    }

    public void setOrderId(int orderId) {
        this.orderId = orderId;
    }

    public int getProductId() {
        return productId;
    }

    public void setProductId(int productId) {
        this.productId = productId;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
```

```
        this.quantity = quantity;
    }

    public double getItemPrice() {
        return itemPrice;
    }

    public void setItemPrice(double itemPrice) {
        this.itemPrice = itemPrice;
    }

    public String getProductName() {
        return productName;
    }

    public void setProductName(String productName) {
        this.productName = productName;
    }

    // Calculate total for this item
    public double getTotal() {
        return quantity * itemPrice;
    }

    @Override
    public String toString() {
        return "OrderItem [orderItemId=" + orderItemId + ", productId=" +
productId +
                ", quantity=" + quantity + ", itemPrice=" + itemPrice + "]";
    }
}
```

## Acceso a Datos (DAO)

### DBConnection.java

```
package com.onlineshop.dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 * Database connection utility class
 */
public class DBConnection {
    private static final String URL = "jdbc:mysql://localhost:3306/onlineshop";
    private static final String USER = "root";
    private static final String PASSWORD = "password"; // Change as needed

    static {
```

```
        try {
            // Load MySQL JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
    }

    /**
     * Get a database connection
     * @return Connection object
     * @throws SQLException if connection fails
     */
    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }

    /**
     * Close the database connection
     * @param connection Connection to close
     */
    public static void closeConnection(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

UserDAO.java

```
package com.onlineshop.dao;

import com.onlineshop.model.User;
import java.sql.*;

/**
 * Data Access Object for User entities
 */
public class UserDAO {

    /**
     * Register a new user
     * @param user User object with username, password, and user type
     * @return true if registration successful, false otherwise
     */
    public boolean registerUser(User user) {
        String sql = "INSERT INTO users (username, password, user_type) VALUES (?,
```

```
?, ?)";
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

            pstmt.setString(1, user.getUsername());
            pstmt.setString(2, user.getPassword()); // In a real application,
password should be hashed
            pstmt.setString(3, user.getUserType());

            int affectedRows = pstmt.executeUpdate();

            if (affectedRows > 0) {
                ResultSet rs = pstmt.getGeneratedKeys();
                if (rs.next()) {
                    user.setUserId(rs.getInt(1));
                }
                return true;
            }

            return false;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }

    /**
     * Authenticate user login
     * @param username Username
     * @param password Password
     * @return User object if authentication successful, null otherwise
     */
    public User authenticateUser(String username, String password) {
        String sql = "SELECT * FROM users WHERE username = ? AND password = ?";
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);

            pstmt.setString(1, username);
            pstmt.setString(2, password); // In a real application, password
should be hashed

            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                User user = new User();
```

```
                user.setUserId(rs.getInt("user_id"));
                user.setUsername(rs.getString("username"));
                user.setPassword(rs.getString("password"));
                user.setUserType(rs.getString("user_type"));

                return user;
            }

            return null;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }

    /**
     * Get user by ID
     * @param userId User ID
     * @return User object if found, null otherwise
     */
    public User getUserById(int userId) {
        String sql = "SELECT * FROM users WHERE user_id = ?";
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, userId);

            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                User user = new User();
                user.setUserId(rs.getInt("user_id"));
                user.setUsername(rs.getString("username"));
                user.setPassword(rs.getString("password"));
                user.setUserType(rs.getString("user_type"));

                return user;
            }

            return null;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }

    /**
```

```
     * Check if username already exists
     * @param username Username to check
     * @return true if username exists, false otherwise
     */
    public boolean usernameExists(String username) {
        String sql = "SELECT COUNT(*) FROM users WHERE username = ?";
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);

            pstmt.setString(1, username);

            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                return rs.getInt(1) > 0;
            }

            return false;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }
}
```

ProductDAO.java

```
package com.onlineshop.dao;

import com.onlineshop.model.Product;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * Data Access Object for Product entities
 */
public class ProductDAO {

    /**
     * Get all products from the database
     * @return List of Product objects
     */
    public List<Product> getAllProducts() {
        String sql = "SELECT * FROM products";
        List<Product> products = new ArrayList<>();
```

```
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                Product product = new Product();
                product.setProductId(rs.getInt("product_id"));
                product.setName(rs.getString("name"));
                product.setDescription(rs.getString("description"));
                product.setPrice(rs.getDouble("price"));
                product.setImageUrl(rs.getString("image_url"));

                products.add(product);
            }

            return products;
        } catch (SQLException e) {
            e.printStackTrace();
            return products;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }

    /**
     * Get product by ID
     * @param productId Product ID
     * @return Product object if found, null otherwise
     */
    public Product getProductById(int productId) {
        String sql = "SELECT * FROM products WHERE product_id = ?";
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, productId);

            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                Product product = new Product();
                product.setProductId(rs.getInt("product_id"));
                product.setName(rs.getString("name"));
                product.setDescription(rs.getString("description"));
                product.setPrice(rs.getDouble("price"));
                product.setImageUrl(rs.getString("image_url"));

                return product;
            }
```

```
            return null;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }

    /**
     * Add a new product
     * @param product Product object with name, description, price, and image URL
     * @return true if addition successful, false otherwise
     */
    public boolean addProduct(Product product) {
        String sql = "INSERT INTO products (name, description, price, image_url)
VALUES (?, ?, ?, ?)";
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

            pstmt.setString(1, product.getName());
            pstmt.setString(2, product.getDescription());
            pstmt.setDouble(3, product.getPrice());
            pstmt.setString(4, product.getImageUrl());

            int affectedRows = pstmt.executeUpdate();

            if (affectedRows > 0) {
                ResultSet rs = pstmt.getGeneratedKeys();
                if (rs.next()) {
                    product.setProductId(rs.getInt(1));
                }
                return true;
            }

            return false;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }
}
```

## OrderDAO.java

```
package com.onlineshop.dao;

import com.onlineshop.model.Order;
import com.onlineshop.model.OrderItem;
import com.onlineshop.model.Product;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 * Data Access Object for Order entities
 */
public class OrderDAO {

    /**
     * Create a new order
     * @param order Order object with user ID, total price, and shipping cost
     * @return true if creation successful, false otherwise
     */
    public boolean createOrder(Order order) {
        String sql = "INSERT INTO orders (user_id, total_price, shipping_cost,
tracking_number) VALUES (?, ?, ?, ?)";
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            // Start transaction
            conn.setAutoCommit(false);

            PreparedStatement pstmt = conn.prepareStatement(sql,
Statement.RETURN_GENERATED_KEYS);

            pstmt.setInt(1, order.getUserId());
            pstmt.setDouble(2, order.getTotalPrice());
            pstmt.setDouble(3, order.getShippingCost());
            pstmt.setString(4, order.getTrackingNumber());

            int affectedRows = pstmt.executeUpdate();

            if (affectedRows > 0) {
                ResultSet rs = pstmt.getGeneratedKeys();
                if (rs.next()) {
                    int orderId = rs.getInt(1);
                    order.setOrderId(orderId);

                    // Insert order items
                    boolean itemsInserted = insertOrderItems(conn, order);

                    if (itemsInserted) {
                        conn.commit();
                        return true;
                    } else {
```

```
                        conn.rollback();
                        return false;
                    }
                }
            }

            conn.rollback();
            return false;
        } catch (SQLException e) {
            try {
                if (conn != null) {
                    conn.rollback();
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
            e.printStackTrace();
            return false;
        } finally {
            try {
                if (conn != null) {
                    conn.setAutoCommit(true);
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            DBConnection.closeConnection(conn);
        }
    }

    /**
     * Insert order items for an order
     * @param conn Database connection
     * @param order Order with order items
     * @return true if insertion successful, false otherwise
     */
    private boolean insertOrderItems(Connection conn, Order order) throws
SQLException {
        String sql = "INSERT INTO order_items (order_id, product_id, quantity,
item_price) VALUES (?, ?, ?, ?)";

        PreparedStatement pstmt = conn.prepareStatement(sql);

        for (OrderItem item : order.getOrderItems()) {
            pstmt.setInt(1, order.getOrderId());
            pstmt.setInt(2, item.getProductId());
            pstmt.setInt(3, item.getQuantity());
            pstmt.setDouble(4, item.getItemPrice());

            pstmt.addBatch();
        }

        int[] results = pstmt.executeBatch();
```

```
            for (int result : results) {
                if (result <= 0) {
                    return false;
                }
            }

            return true;
        }

        /**
         * Get the next tracking number
         * @return Next tracking number
         */
        public String getNextTrackingNumber() {
            String sql = "SELECT MAX(CAST(tracking_number AS UNSIGNED)) AS
    max_tracking FROM orders";
            Connection conn = null;

            try {
                conn = DBConnection.getConnection();
                Statement stmt = conn.createStatement();
                ResultSet rs = stmt.executeQuery(sql);

                if (rs.next()) {
                    String maxTracking = rs.getString("max_tracking");
                    if (maxTracking == null) {
                        return "100001"; // Starting tracking number
                    } else {
                        // Increment the tracking number
                        int nextTracking = Integer.parseInt(maxTracking) + 1;
                        return String.valueOf(nextTracking);
                    }
                }

                return "100001"; // Starting tracking number
            } catch (SQLException e) {
                e.printStackTrace();
                return "100001"; // Starting tracking number on error
            } finally {
                DBConnection.closeConnection(conn);
            }
        }

        /**
         * Get order by ID with order items
         * @param orderId Order ID
         * @return Order object if found, null otherwise
         */
        public Order getOrderById(int orderId) {
            String sql = "SELECT * FROM orders WHERE order_id = ?";
            Connection conn = null;

            try {
                conn = DBConnection.getConnection();
```

```
            PreparedStatement pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, orderId);

            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                Order order = new Order();
                order.setOrderId(rs.getInt("order_id"));
                order.setUserId(rs.getInt("user_id"));
                order.setOrderDate(rs.getTimestamp("order_date"));
                order.setTotalPrice(rs.getDouble("total_price"));
                order.setShippingCost(rs.getDouble("shipping_cost"));
                order.setTrackingNumber(rs.getString("tracking_number"));

                // Get order items
                List<OrderItem> orderItems = getOrderItemsByOrderId(orderId);
                order.setOrderItems(orderItems);

                return order;
            }

            return null;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }

    /**
     * Get order items by order ID
     * @param orderId Order ID
     * @return List of OrderItem objects
     */
    private List<OrderItem> getOrderItemsByOrderId(int orderId) {
        String sql = "SELECT oi.*, p.name AS product_name FROM order_items oi " +
                     "JOIN products p ON oi.product_id = p.product_id " +
                     "WHERE oi.order_id = ?";
        List<OrderItem> orderItems = new ArrayList<>();
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, orderId);

            ResultSet rs = pstmt.executeQuery();

            while (rs.next()) {
                OrderItem item = new OrderItem();
                item.setOrderItemId(rs.getInt("order_item_id"));
```

```
                item.setOrderId(rs.getInt("order_id"));
                item.setProductId(rs.getInt("product_id"));
                item.setQuantity(rs.getInt("quantity"));
                item.setItemPrice(rs.getDouble("item_price"));
                item.setProductName(rs.getString("product_name"));

                orderItems.add(item);
            }

            return orderItems;
        } catch (SQLException e) {
            e.printStackTrace();
            return orderItems;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }

    /**
     * Get all orders for a user
     * @param userId User ID
     * @return List of Order objects
     */
    public List<Order> getOrdersByUserId(int userId) {
        String sql = "SELECT * FROM orders WHERE user_id = ? ORDER BY order_date
DESC";
        List<Order> orders = new ArrayList<>();
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql);

            pstmt.setInt(1, userId);

            ResultSet rs = pstmt.executeQuery();

            while (rs.next()) {
                Order order = new Order();
                order.setOrderId(rs.getInt("order_id"));
                order.setUserId(rs.getInt("user_id"));
                order.setOrderDate(rs.getTimestamp("order_date"));
                order.setTotalPrice(rs.getDouble("total_price"));
                order.setShippingCost(rs.getDouble("shipping_cost"));
                order.setTrackingNumber(rs.getString("tracking_number"));

                orders.add(order);
            }

            return orders;
        } catch (SQLException e) {
            e.printStackTrace();
            return orders;
        } finally {
```

```
            DBConnection.closeConnection(conn);
        }
    }

    /**
     * Get all orders (for seller)
     * @return List of Order objects
     */
    public List<Order> getAllOrders() {
        String sql = "SELECT o.*, u.username FROM orders o JOIN users u ON
o.user_id = u.user_id ORDER BY o.order_date DESC";
        List<Order> orders = new ArrayList<>();
        Connection conn = null;

        try {
            conn = DBConnection.getConnection();
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(sql);

            while (rs.next()) {
                Order order = new Order();
                order.setOrderId(rs.getInt("order_id"));
                order.setUserId(rs.getInt("user_id"));
                order.setOrderDate(rs.getTimestamp("order_date"));
                order.setTotalPrice(rs.getDouble("total_price"));
                order.setShippingCost(rs.getDouble("shipping_cost"));
                order.setTrackingNumber(rs.getString("tracking_number"));

                orders.add(order);
            }

            return orders;
        } catch (SQLException e) {
            e.printStackTrace();
            return orders;
        } finally {
            DBConnection.closeConnection(conn);
        }
    }
}
```

## Servlets

### LoginServlet.java

```
package com.onlineshop.servlet;

import com.onlineshop.dao.UserDAO;
import com.onlineshop.model.User;

import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * Servlet for handling user login
 */
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Check if user is already logged in
        HttpSession session = request.getSession(false);
        if (session != null && session.getAttribute("user") != null) {
            // User is already logged in, redirect to product list
            response.sendRedirect("products");
            return;
        }

        // Forward to login page
        request.getRequestDispatcher("/login.jsp").forward(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");

        // Validate input
        if (username == null || username.trim().isEmpty() ||
            password == null || password.trim().isEmpty()) {
            request.setAttribute("errorMessage", "Username and password are
required");
            request.getRequestDispatcher("/login.jsp").forward(request, response);
            return;
        }

        // Authenticate user
        UserDAO userDAO = new UserDAO();
        User user = userDAO.authenticateUser(username, password);

        if (user != null) {
            // Authentication successful
            HttpSession session = request.getSession();
            session.setAttribute("user", user);

            // Set cookie for customer ID (non-session cookie)
            if ("customer".equals(user.getUserType())) {
                Cookie userIdCookie = new Cookie("userId",
```

```
String.valueOf(user.getUserId()));
                userIdCookie.setMaxAge(30 * 24 * 60 * 60); // 30 days
                userIdCookie.setPath("/");
                response.addCookie(userIdCookie);
            }

            // Redirect based on user type
            if ("seller".equals(user.getUserType())) {
                response.sendRedirect("seller/orders");
            } else {
                response.sendRedirect("products");
            }
        } else {
            // Authentication failed
            request.setAttribute("errorMessage", "Invalid username or password");
            request.getRequestDispatcher("/login.jsp").forward(request, response);
        }
    }
}
```

RegisterServlet.java

```
package com.onlineshop.servlet;

import com.onlineshop.dao.UserDAO;
import com.onlineshop.model.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * Servlet for handling user registration
 */
@WebServlet("/register")
public class RegisterServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Forward to registration page
        request.getRequestDispatcher("/register.jsp").forward(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        String username = request.getParameter("username");
```

```
        String password = request.getParameter("password");
        String confirmPassword = request.getParameter("confirmPassword");

        // Validate input
        if (username == null || username.trim().isEmpty() ||
            password == null || password.trim().isEmpty() ||
            confirmPassword == null || confirmPassword.trim().isEmpty()) {
            request.setAttribute("errorMessage", "All fields are required");
            request.getRequestDispatcher("/register.jsp").forward(request,
response);
            return;
        }

        // Check if passwords match
        if (!password.equals(confirmPassword)) {
            request.setAttribute("errorMessage", "Passwords do not match");
            request.getRequestDispatcher("/register.jsp").forward(request,
response);
            return;
        }

        // Check if username already exists
        UserDAO userDAO = new UserDAO();
        if (userDAO.usernameExists(username)) {
            request.setAttribute("errorMessage", "Username already exists");
            request.getRequestDispatcher("/register.jsp").forward(request,
response);
            return;
        }

        // Create new user
        User user = new User(username, password, "customer"); // Default to
customer type
        boolean registered = userDAO.registerUser(user);

        if (registered) {
            // Registration successful
            HttpSession session = request.getSession();
            session.setAttribute("user", user);

            // Set cookie for customer ID (non-session cookie)
            Cookie userIdCookie = new Cookie("userId",
String.valueOf(user.getUserId()));
            userIdCookie.setMaxAge(30 * 24 * 60 * 60); // 30 days
            userIdCookie.setPath("/");
            response.addCookie(userIdCookie);

            // Redirect to product list
            response.sendRedirect("products");
        } else {
            // Registration failed
            request.setAttribute("errorMessage", "Registration failed");
            request.getRequestDispatcher("/register.jsp").forward(request,
response);
```

```
        }
    }
}
```

ProductListServlet.java

```
package com.onlineshop.servlet;

import com.onlineshop.dao.ProductDAO;
import com.onlineshop.model.Product;
import com.onlineshop.model.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * Servlet for displaying product list
 */
@WebServlet("/products")
public class ProductListServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Check if user is logged in
        HttpSession session = request.getSession(false);
        if (session == null || session.getAttribute("user") == null) {
            // User is not logged in, redirect to login
            response.sendRedirect("login");
            return;
        }

        // Get products from database
        ProductDAO productDAO = new ProductDAO();
        List<Product> products = productDAO.getAllProducts();

        // Set products as request attribute
        request.setAttribute("products", products);

        // Forward to product list page
        request.getRequestDispatcher("/product-list.jsp").forward(request,
response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
```

```
response)
            throws ServletException, IOException {
        // Check if user is logged in
        HttpSession session = request.getSession(false);
        if (session == null || session.getAttribute("user") == null) {
            // User is not logged in, redirect to login
            response.sendRedirect("login");
            return;
        }

        // Get product IDs and quantities from request
        String[] productIds = request.getParameterValues("productId");
        String[] quantities = request.getParameterValues("quantity");

        if (productIds == null || quantities == null || productIds.length !=
quantities.length) {
            // Invalid request, redirect to product list
            response.sendRedirect("products");
            return;
        }

        // Flag to check if any product was selected
        boolean anyProductSelected = false;

        // Get cart from session or create new one
        @SuppressWarnings("unchecked")
        List<com.onlineshop.model.OrderItem> cart =
(List<com.onlineshop.model.OrderItem>) session.getAttribute("cart");

        if (cart == null) {
            cart = new ArrayList<>();
        }

        // Process each product
        ProductDAO productDAO = new ProductDAO();

        for (int i = 0; i < productIds.length; i++) {
            int productId = Integer.parseInt(productIds[i]);
            int quantity = Integer.parseInt(quantities[i]);

            if (quantity > 0) {
                anyProductSelected = true;

                // Get product from database
                Product product = productDAO.getProductById(productId);

                if (product != null) {
                    // Check if product is already in cart
                    boolean productInCart = false;

                    for (com.onlineshop.model.OrderItem item : cart) {
                        if (item.getProductId() == productId) {
                            // Update quantity
                            item.setQuantity(item.getQuantity() + quantity);
```

```
                            productInCart = true;
                            break;
                        }
                    }

                    if (!productInCart) {
                        // Add new item to cart
                        com.onlineshop.model.OrderItem item = new
com.onlineshop.model.OrderItem(
                            productId, quantity, product.getPrice(),
product.getName());
                        cart.add(item);
                    }
                }
            }
        }

        // Save cart to session
        session.setAttribute("cart", cart);

        if (anyProductSelected) {
            // Redirect to cart
            response.sendRedirect("cart");
        } else {
            // No product selected, redirect to product list
            request.setAttribute("errorMessage", "Please select at least one
product");
            doGet(request, response);
        }
    }
}
```

## CartServlet.java

```
package com.onlineshop.servlet;

import com.onlineshop.model.OrderItem;
import com.onlineshop.model.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

/**
 * Servlet for handling shopping cart
 */
@WebServlet("/cart")
```

```
public class CartServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Check if user is logged in
        HttpSession session = request.getSession(false);
        if (session == null || session.getAttribute("user") == null) {
            // User is not logged in, redirect to login
            response.sendRedirect("login");
            return;
        }

        // Get cart from session
        @SuppressWarnings("unchecked")
        List<OrderItem> cart = (List<OrderItem>) session.getAttribute("cart");

        // Inicializar los valores por defecto aunque el carrito esté vacío
        double totalPrice = 0.0;
        double shippingCost = 0.0;
        double finalPrice = 0.0;

        // Si el carrito existe y tiene elementos, calculamos los totales
        if (cart != null && !cart.isEmpty()) {
            int totalItems = 0;

            for (OrderItem item : cart) {
                totalPrice += item.getTotal();
                totalItems += item.getQuantity();
            }

            // Calculate shipping cost (2€ + 1€ per item)
            shippingCost = 2.0 + (totalItems * 1.0);

            // Calculate final price
            finalPrice = totalPrice + shippingCost;
        } else {
            // Si el carrito está vacío, inicializamos uno vacío
            cart = new ArrayList<>();
            session.setAttribute("cart", cart);
        }

        // Set attributes for JSP
        request.setAttribute("cart", cart);
        request.setAttribute("totalPrice", totalPrice);
        request.setAttribute("shippingCost", shippingCost);
        request.setAttribute("finalPrice", finalPrice);

        // Forward to cart page (siempre vamos a cart.jsp, incluso con carrito
vacío)
        request.getRequestDispatcher("/cart.jsp").forward(request, response);
    }
```

```
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        // Check if user is logged in
        HttpSession session = request.getSession(false);
        if (session == null || session.getAttribute("user") == null) {
            // User is not logged in, redirect to login
            response.sendRedirect("login");
            return;
        }

        // Check action parameter
        String action = request.getParameter("action");

        if ("update".equals(action)) {
            // Update cart quantities
            updateCart(request, session);
        } else if ("checkout".equals(action)) {
            // Proceed to checkout
            response.sendRedirect("checkout");
            return;
        } else if ("continue".equals(action)) {
            // Continue shopping
            response.sendRedirect("products");
            return;
        } else if ("remove".equals(action)) {
            // Remove item from cart
            removeItem(request, session);
        }

        // Redirect back to cart
        response.sendRedirect("cart");
    }

    /**
     * Update cart quantities
     * @param request HTTP request
     * @param session HTTP session
     */
    private void updateCart(HttpServletRequest request, HttpSession session) {
        String[] productIds = request.getParameterValues("productId");
        String[] quantities = request.getParameterValues("quantity");

        if (productIds == null || quantities == null || productIds.length !=
quantities.length) {
            return;
        }

        @SuppressWarnings("unchecked")
        List<OrderItem> cart = (List<OrderItem>) session.getAttribute("cart");

        if (cart == null || cart.isEmpty()) {
            return;
```

```
        }

        // Crear una lista temporal con productos a eliminar
        List<OrderItem> itemsToRemove = new ArrayList<>();

        // Update quantities
        for (int i = 0; i < productIds.length; i++) {
            int productId = Integer.parseInt(productIds[i]);
            int quantity = Integer.parseInt(quantities[i]);

            for (OrderItem item : cart) {
                if (item.getProductId() == productId) {
                    if (quantity > 0) {
                        item.setQuantity(quantity);
                    } else {
                        // Marcar para eliminar en lugar de eliminar directamente
                        // (evitar ConcurrentModificationException)
                        itemsToRemove.add(item);
                    }
                    break;
                }
            }
        }

        // Eliminar los items marcados
        cart.removeAll(itemsToRemove);

        // Save updated cart to session
        session.setAttribute("cart", cart);
    }

    /**
     * Remove item from cart
     * @param request HTTP request
     * @param session HTTP session
     */
    private void removeItem(HttpServletRequest request, HttpSession session) {
        String productId = request.getParameter("productId");

        if (productId == null) {
            return;
        }

        @SuppressWarnings("unchecked")
        List<OrderItem> cart = (List<OrderItem>) session.getAttribute("cart");

        if (cart == null) {
            return;
        }

        // Find and remove item using iterator to avoid
ConcurrentModificationException
        Iterator<OrderItem> iterator = cart.iterator();
        while (iterator.hasNext()) {
```

```
            OrderItem item = iterator.next();
            if (item.getProductId() == Integer.parseInt(productId)) {
                iterator.remove();
                break;
            }
        }

        // Save updated cart to session
        session.setAttribute("cart", cart);
    }
}
```

CheckoutServlet.java

```
package com.onlineshop.servlet;

import com.onlineshop.dao.OrderDAO;
import com.onlineshop.model.Order;
import com.onlineshop.model.OrderItem;
import com.onlineshop.model.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.Date;
import java.util.List;

/**
 * Servlet for handling checkout process
 */
@WebServlet("/checkout")
public class CheckoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Check if user is logged in
        HttpSession session = request.getSession(false);
        if (session == null || session.getAttribute("user") == null) {
            // User is not logged in, redirect to login
            response.sendRedirect("login");
            return;
        }

        // Get cart from session
        @SuppressWarnings("unchecked")
        List<OrderItem> cart = (List<OrderItem>) session.getAttribute("cart");

        if (cart == null || cart.isEmpty()) {
```

```
            // Cart is empty, redirect to product list
            request.setAttribute("errorMessage", "Your cart is empty");
            request.getRequestDispatcher("/product-list.jsp").forward(request,
response);
            return;
        }

        // Calculate total price
        double totalPrice = 0.0;
        int totalItems = 0;

        for (OrderItem item : cart) {
            totalPrice += item.getTotal();
            totalItems += item.getQuantity();
        }

        // Calculate shipping cost (2€ + 1€ per item)
        double shippingCost = 2.0 + (totalItems * 1.0);

        // Calculate final price
        double finalPrice = totalPrice + shippingCost;

        // Set attributes for JSP
        request.setAttribute("cart", cart);
        request.setAttribute("totalPrice", totalPrice);
        request.setAttribute("shippingCost", shippingCost);
        request.setAttribute("finalPrice", finalPrice);

        // Forward to checkout page
        request.getRequestDispatcher("/checkout.jsp").forward(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        // Check if user is logged in
        HttpSession session = request.getSession(false);
        if (session == null || session.getAttribute("user") == null) {
            // User is not logged in, redirect to login
            response.sendRedirect("login");
            return;
        }

        // Get user from session
        User user = (User) session.getAttribute("user");

        // Get cart from session
        @SuppressWarnings("unchecked")
        List<OrderItem> cart = (List<OrderItem>) session.getAttribute("cart");

        if (cart == null || cart.isEmpty()) {
            // Cart is empty, redirect to product list
            request.setAttribute("errorMessage", "Your cart is empty");
```

```
            request.getRequestDispatcher("/product-list.jsp").forward(request,
response);
            return;
        }

        // Calculate total price and shipping cost
        double totalPrice = 0.0;
        int totalItems = 0;

        for (OrderItem item : cart) {
            totalPrice += item.getTotal();
            totalItems += item.getQuantity();
        }

        double shippingCost = 2.0 + (totalItems * 1.0);

        // Get next tracking number
        OrderDAO orderDAO = new OrderDAO();
        String trackingNumber = orderDAO.getNextTrackingNumber();

        // Create order
        Order order = new Order(user.getUserId(), new Date(), totalPrice,
shippingCost, trackingNumber);

        // Add items to order
        for (OrderItem item : cart) {
            order.addOrderItem(item);
        }

        // Save order to database
        boolean orderCreated = orderDAO.createOrder(order);

        if (orderCreated) {
            // Clear cart
            session.removeAttribute("cart");

            // Set order in session for confirmation page
            session.setAttribute("order", order);

            // Redirect to confirmation page
            response.sendRedirect("confirmation");
        } else {
            // Order creation failed
            request.setAttribute("errorMessage", "Failed to create order");
            doGet(request, response);
        }
    }
}
```

## ConfirmationServlet.java

```
package com.onlineshop.servlet;

import com.onlineshop.model.Order;
import com.onlineshop.model.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * Servlet for order confirmation
 */
@WebServlet("/confirmation")
public class ConfirmationServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Check if user is logged in
        HttpSession session = request.getSession(false);
        if (session == null || session.getAttribute("user") == null) {
            // User is not logged in, redirect to login
            response.sendRedirect("login");
            return;
        }

        // Get order from session
        Order order = (Order) session.getAttribute("order");

        if (order == null) {
            // No order in session, redirect to product list
            response.sendRedirect("products");
            return;
        }

        // Forward to confirmation page
        request.getRequestDispatcher("/confirmation.jsp").forward(request,
response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
        // Remove order from session
        HttpSession session = request.getSession(false);
        if (session != null) {
            session.removeAttribute("order");
        }

        // Redirect to product list
```

```
            response.sendRedirect("products");
        }
    }
```

SellerOrdersServlet.java

```
package com.onlineshop.servlet;

import com.onlineshop.dao.OrderDAO;
import com.onlineshop.model.Order;
import com.onlineshop.model.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.List;

/**
 * Servlet for seller to view all orders
 */
@WebServlet("/seller/orders")
public class SellerOrdersServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Check if user is logged in
        HttpSession session = request.getSession(false);
        if (session == null || session.getAttribute("user") == null) {
            // User is not logged in, redirect to login
            response.sendRedirect("../login");
            return;
        }

        // Check if user is a seller
        User user = (User) session.getAttribute("user");
        if (!"seller".equals(user.getUserType())) {
            // User is not a seller, redirect to product list
            response.sendRedirect("../products");
            return;
        }

        // Get all orders
        OrderDAO orderDAO = new OrderDAO();
        List<Order> orders = orderDAO.getAllOrders();

        // Set orders as request attribute
        request.setAttribute("orders", orders);
```

```
        // Forward to seller orders page
        request.getRequestDispatcher("/seller-orders.jsp").forward(request,
response);
    }
}
```

CustomerOrdersServlet.java

```
package com.onlineshop.servlet;

import com.onlineshop.dao.OrderDAO;
import com.onlineshop.dao.UserDAO;
import com.onlineshop.model.Order;
import com.onlineshop.model.User;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;
import java.util.List;

/**
 * Servlet for customer to view their orders
 */
@WebServlet("/my-orders")
public class CustomerOrdersServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Try to get user ID from session
        HttpSession session = request.getSession(false);
        User user = null;

        if (session != null && session.getAttribute("user") != null) {
            user = (User) session.getAttribute("user");
        } else {
            // Try to get user ID from cookie
            Cookie[] cookies = request.getCookies();
            Integer userId = null;

            if (cookies != null) {
                for (Cookie cookie : cookies) {
                    if ("userId".equals(cookie.getName())) {
                        try {
                            userId = Integer.parseInt(cookie.getValue());
                        } catch (NumberFormatException e) {
                            // Invalid cookie value
                        }
                        break;
```

```
                    }
                }
            }

            if (userId != null) {
                // Get user from database
                UserDAO userDAO = new UserDAO();
                user = userDAO.getUserById(userId);
            }
        }

        if (user == null) {
            // No user found, redirect to login
            response.sendRedirect("login");
            return;
        }

        // Get orders for user
        OrderDAO orderDAO = new OrderDAO();
        List<Order> orders = orderDAO.getOrdersByUserId(user.getUserId());

        // Set orders as request attribute
        request.setAttribute("orders", orders);

        // Forward to customer orders page
        request.getRequestDispatcher("/customer-orders.jsp").forward(request,
response);
    }
}
```

## LogoutServlet.java

```
package com.onlineshop.servlet;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
import java.io.IOException;

/**
 * Servlet for handling user logout
 */
@WebServlet("/logout")
public class LogoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Get current session
        HttpSession session = request.getSession(false);
```

```
        // Invalidate session if it exists
        if (session != null) {
            session.invalidate();
        }

        // Redirect to login page
        response.sendRedirect("login");
    }
}
```

## Páginas JSP

### login.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Login - Online Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">
    <div class="container">
        <div class="row justify-content-center mt-5">
            <div class="col-md-6">
                <div class="card shadow">
                    <div class="card-header bg-primary text-white">
                        <h3 class="mb-0">Login</h3>
                    </div>
                    <div class="card-body">
                        <% if(request.getAttribute("errorMessage") != null) { %>
                            <div class="alert alert-danger" role="alert">
                                <%= request.getAttribute("errorMessage") %>
                            </div>
                        <% } %>

                        <form action="login" method="post">
                            <div class="mb-3">
                                <label for="username" class="form-
label">Username</label>
                                <input type="text" class="form-control"
id="username" name="username" required>
                            </div>
                            <div class="mb-3">
                                <label for="password" class="form-
```

```
label">Password</label>
                                <input type="password" class="form-control"
id="password" name="password" required>
                            </div>
                            <div class="d-grid gap-2">
                                <button type="submit" class="btn btn-
primary">Login</button>
                            </div>
                        </form>

                        <div class="mt-3 text-center">
                            <p>Don't have an account? <a href="register">Register
here</a></p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

register.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Register - Online Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">
    <div class="container">
        <div class="row justify-content-center mt-5">
            <div class="col-md-6">
                <div class="card shadow">
                    <div class="card-header bg-primary text-white">
                        <h3 class="mb-0">Register</h3>
                    </div>
                    <div class="card-body">
                        <% if(request.getAttribute("errorMessage") != null) { %>
                            <div class="alert alert-danger" role="alert">
```

```
                                <%= request.getAttribute("errorMessage") %>
                            </div>
                        <% } %>

                        <form action="register" method="post">
                            <div class="mb-3">
                                <label for="username" class="form-
label">Username</label>
                                <input type="text" class="form-control"
id="username" name="username" required>
                            </div>
                            <div class="mb-3">
                                <label for="password" class="form-
label">Password</label>
                                <input type="password" class="form-control"
id="password" name="password" required>
                            </div>
                            <div class="mb-3">
                                <label for="confirmPassword" class="form-
label">Confirm Password</label>
                                <input type="password" class="form-control"
id="confirmPassword" name="confirmPassword" required>
                            </div>
                            <div class="d-grid gap-2">
                                <button type="submit" class="btn btn-
primary">Register</button>
                            </div>
                        </form>

                        <div class="mt-3 text-center">
                            <p>Already have an account? <a href="login">Login
here</a></p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## product-list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ page import="java.util.List" %>
<%@ page import="com.onlineshop.model.Product" %>
```

```
<%@ page import="com.onlineshop.model.User" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Products - League of Legends Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .product-image {
            height: 200px;
            object-fit: contain;
            margin-bottom: 15px;
        }
        .card {
            height: 100%;
            display: flex;
            flex-direction: column;
        }
        .card-body {
            flex: 1;
            display: flex;
            flex-direction: column;
        }
        .card-text {
            flex-grow: 1;
        }
        .navbar-logo {
            height: 40px;
            margin-right: 10px;
        }
    </style>
</head>
<body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="#">
                <img src="${pageContext.request.contextPath}/images/logo.png"
alt="Shop Logo" class="navbar-logo">
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link active" href="products">Products</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="cart">Cart</a>
```

```
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="my-orders">My Orders</a>
                    </li>
                </ul>
                <div class="d-flex">
                    <% User user = (User) session.getAttribute("user"); %>
                    <span class="navbar-text me-3">
                        Welcome, <%= user.getUsername() %>
                    </span>
                    <a href="logout" class="btn btn-outline-light">Logout</a>
                </div>
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        <h2>Champions & Characters</h2>

        <% if(request.getAttribute("errorMessage") != null) { %>
            <div class="alert alert-danger" role="alert">
                <%= request.getAttribute("errorMessage") %>
            </div>
        <% } %>

        <form action="products" method="post">
            <div class="row">
                <%
                List<Product> products = (List<Product>)
request.getAttribute("products");
                if(products != null && !products.isEmpty()) {
                    for(Product product : products) {
                %>
                <div class="col-md-3 mb-4">
                    <div class="card h-100">
                        <img src="<%= product.getImageUrl() %>" class="card-img-
top product-image" alt="<%= product.getName() %>">
                        <div class="card-body">
                            <h5 class="card-title"><%= product.getName() %></h5>
                            <p class="card-text"><%= product.getDescription() %>
</p>
                            <p class="card-text text-primary fw-bold">€<%=
String.format("%.2f", product.getPrice()) %></p>
                            <div class="d-flex align-items-center mt-auto">
                                <input type="hidden" name="productId" value="<%=
product.getProductId() %>">
                                <label for="quantity-<%= product.getProductId()
%>" class="me-2">Quantity:</label>
                                <select class="form-select form-select-sm w-auto"
id="quantity-<%= product.getProductId() %>" name="quantity">
                                    <option value="0">0</option>
                                    <option value="1">1</option>
                                    <option value="2">2</option>
                                    <option value="3">3</option>
```

```
                                    <option value="4">4</option>
                                    <option value="5">5</option>
                                </select>
                            </div>
                        </div>
                    </div>
                </div>
                <%
                    }
                } else {
                %>
                <div class="col-12">
                    <p>No products available.</p>
                </div>
                <% } %>
            </div>

            <div class="mt-3 mb-5">
                <button type="submit" class="btn btn-primary">Add to Cart</button>
            </div>
        </form>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## cart.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ page import="java.util.List" %>
<%@ page import="com.onlineshop.model.OrderItem" %>
<%@ page import="com.onlineshop.model.User" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Cart - League of Legends Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .navbar-logo {
            height: 40px;
            margin-right: 10px;
        }
    </style>
```

```
</head>
<body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="#">
                <img src="${pageContext.request.contextPath}/images/logo.png"
alt="Shop Logo" class="navbar-logo">
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="products">Products</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" href="cart">Cart</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="my-orders">My Orders</a>
                    </li>
                </ul>
                <div class="d-flex">
                    <% User user = (User) session.getAttribute("user"); %>
                    <span class="navbar-text me-3">
                        Welcome, <%= user.getUsername() %>
                    </span>
                    <a href="logout" class="btn btn-outline-light">Logout</a>
                </div>
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        <h2>Your Shopping Cart</h2>

        <%
        List<OrderItem> cart = (List<OrderItem>) session.getAttribute("cart");
        Double totalPrice = (Double) request.getAttribute("totalPrice");
        Double shippingCost = (Double) request.getAttribute("shippingCost");
        Double finalPrice = (Double) request.getAttribute("finalPrice");

        if(cart != null && !cart.isEmpty()) {
        %>

        <!-- Formulario principal para Update/Checkout -->
        <form id="mainCartForm" action="cart" method="post">
            <table class="table">
                <thead>
                    <tr>
                        <th>Product</th>
```

```
                        <th>Price</th>
                        <th>Quantity</th>
                        <th>Total</th>
                        <th>Action</th>
                    </tr>
                </thead>
                <tbody>
                    <% for(OrderItem item : cart) { %>
                    <tr>
                        <td><%= item.getProductName() %></td>
                        <td>€<%= String.format("%.2f", item.getItemPrice()) %>
</td>
                        <td>
                            <input type="hidden" name="productId" value="<%=
item.getProductId() %>">
                            <select class="form-select form-select-sm w-auto"
name="quantity">
                                <% for(int i = 1; i <= 10; i++) { %>
                                <option value="<%= i %>" <%= item.getQuantity() ==
i ? "selected" : "" %>><%= i %></option>
                                <% } %>
                            </select>
                        </td>
                        <td>€<%= String.format("%.2f", item.getTotal()) %></td>
                        <td>
                            <!-- Botón que activa JavaScript para eliminar -->
                            <button type="button" class="btn btn-sm btn-danger"
                                    onclick="removeItem(<%= item.getProductId()
%>)">Remove</button>
                        </td>
                    </tr>
                    <% } %>
                </tbody>
                <tfoot>
                    <tr>
                        <td colspan="3" class="text-end fw-bold">Subtotal:</td>
                        <td>€<%= String.format("%.2f", totalPrice) %></td>
                        <td></td>
                    </tr>
                    <tr>
                        <td colspan="3" class="text-end fw-bold">Shipping Cost:
</td>
                        <td>€<%= String.format("%.2f", shippingCost) %></td>
                        <td></td>
                    </tr>
                    <tr>
                        <td colspan="3" class="text-end fw-bold">Total:</td>
                        <td class="fw-bold text-primary">€<%=
String.format("%.2f", finalPrice) %></td>
                        <td></td>
                    </tr>
                </tfoot>
            </table>
```

```
            <!-- Campo oculto para la acción -->
            <input type="hidden" id="cartAction" name="action" value="">

            <div class="d-flex justify-content-between mt-4 mb-5">
                <button type="button" onclick="setAction('update')" class="btn
btn-secondary">Update Cart</button>
                <div>
                    <button type="button" onclick="setAction('continue')"
class="btn btn-outline-primary me-2">Continue Shopping</button>
                    <button type="button" onclick="setAction('checkout')"
class="btn btn-primary">Proceed to Checkout</button>
                </div>
            </div>
        </form>

        <!-- Formulario oculto para eliminar elementos -->
        <form id="removeForm" action="cart" method="post" style="display: none;">
            <input type="hidden" name="action" value="remove">
            <input type="hidden" id="removeProductId" name="productId" value="">
        </form>

        <!-- JavaScript para manejar los diferentes formularios -->
        <script>
            function setAction(action) {
                document.getElementById('cartAction').value = action;
                document.getElementById('mainCartForm').submit();
            }

            function removeItem(productId) {
                document.getElementById('removeProductId').value = productId;
                document.getElementById('removeForm').submit();
            }
        </script>

        <% } else { %>
        <div class="alert alert-info" role="alert">
            Your cart is empty. <a href="products" class="alert-link">Go
shopping</a>
        </div>
        <% } %>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## checkout.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ page import="java.util.List" %>
<%@ page import="com.onlineshop.model.OrderItem" %>
<%@ page import="com.onlineshop.model.User" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Checkout - Online Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .navbar-logo {
            height: 40px;
            margin-right: 10px;
        }
    </style>
</head>
<body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="#">
                <img src="${pageContext.request.contextPath}/images/logo.png"
alt="Shop Logo" class="navbar-logo">
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="products">Products</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="cart">Cart</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="my-orders">My Orders</a>
                    </li>
                </ul>
                <div class="d-flex">
                    <% User user = (User) session.getAttribute("user"); %>
                    <span class="navbar-text me-3">
                        Welcome, <%= user.getUsername() %>
                    </span>
                    <a href="logout" class="btn btn-outline-light">Logout</a>
                </div>
            </div>
```

```
        </div>
    </nav>

    <div class="container mt-4">
        <div class="row">
            <div class="col-md-8">
                <div class="card mb-4">
                    <div class="card-header bg-primary text-white">
                        <h4 class="mb-0">Order Summary</h4>
                    </div>
                    <div class="card-body">
                        <table class="table">
                            <thead>
                                <tr>
                                    <th>Product</th>
                                    <th>Price</th>
                                    <th>Quantity</th>
                                    <th>Total</th>
                                </tr>
                            </thead>
                            <tbody>
                                <%
                                List<OrderItem> cart = (List<OrderItem>)
session.getAttribute("cart");
                                for(OrderItem item : cart) {
                                %>
                                <tr>
                                    <td><%= item.getProductName() %></td>
                                    <td>€<%= String.format("%.2f",
item.getItemPrice()) %></td>
                                    <td><%= item.getQuantity() %></td>
                                    <td>€<%= String.format("%.2f",
item.getTotal()) %></td>
                                </tr>
                                <% } %>
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>

            <div class="col-md-4">
                <div class="card">
                    <div class="card-header bg-primary text-white">
                        <h4 class="mb-0">Order Total</h4>
                    </div>
                    <div class="card-body">
                        <%
                        Double totalPrice = (Double)
request.getAttribute("totalPrice");
                        Double shippingCost = (Double)
request.getAttribute("shippingCost");
                        Double finalPrice = (Double)
request.getAttribute("finalPrice");
```

```
                        %>
                        <div class="d-flex justify-content-between mb-2">
                            <span>Subtotal:</span>
                            <span>€<%= String.format("%.2f", totalPrice) %></span>
                        </div>
                        <div class="d-flex justify-content-between mb-2">
                            <span>Shipping Cost:</span>
                            <span>€<%= String.format("%.2f", shippingCost) %>
</span>
                        </div>
                        <hr>
                        <div class="d-flex justify-content-between fw-bold">
                            <span>Total:</span>
                            <span class="text-primary">€<%= String.format("%.2f",
finalPrice) %></span>
                        </div>

                        <form action="checkout" method="post" class="mt-4">
                            <div class="d-grid gap-2">
                                <button type="submit" class="btn btn-
primary">Confirm Purchase</button>
                                <a href="cart" class="btn btn-outline-
secondary">Back to Cart</a>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## confirmation.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ page import="com.onlineshop.model.Order" %>
<%@ page import="com.onlineshop.model.User" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Order Confirmation - Online Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
```

```
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .navbar-logo {
            height: 40px;
            margin-right: 10px;
        }
    </style>
</head>
<body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="#">
                <img src="${pageContext.request.contextPath}/images/logo.png"
alt="Shop Logo" class="navbar-logo">
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="products">Products</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="cart">Cart</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="my-orders">My Orders</a>
                    </li>
                </ul>
                <div class="d-flex">
                    <% User user = (User) session.getAttribute("user"); %>
                    <span class="navbar-text me-3">
                        Welcome, <%= user.getUsername() %>
                    </span>
                    <a href="logout" class="btn btn-outline-light">Logout</a>
                </div>
            </div>
        </div>
    </nav>

    <div class="container mt-5">
        <div class="row justify-content-center">
            <div class="col-md-8">
                <div class="card shadow">
                    <div class="card-body text-center py-5">
                        <h1 class="display-4 text-success mb-4">Thank You!</h1>
                        <p class="lead">Your order has been placed successfully.
</p>

                        <% Order order = (Order) session.getAttribute("order"); %>
                        <div class="card mt-4 mb-4">
```

```
                            <div class="card-header bg-light">
                                <h5 class="mb-0">Order Details</h5>
                            </div>
                            <div class="card-body">
                                <div class="row">
                                    <div class="col-md-6 text-start">
                                        <p><strong>Order ID:</strong> #<%=
order.getOrderId() %></p>
                                        <p><strong>Date:</strong> <%=
order.getOrderDate() %></p>
                                    </div>
                                    <div class="col-md-6 text-start">
                                        <p><strong>Tracking Number:</strong> <%=
order.getTrackingNumber() %></p>
                                        <p><strong>Total Price:</strong> €<%=
String.format("%.2f", order.getTotalPrice() + order.getShippingCost()) %></p>
                                    </div>
                                </div>
                            </div>
                        </div>

                        <p>You can check your order status in the "My Orders"
section.</p>

                        <form action="confirmation" method="post" class="mt-4">
                            <button type="submit" class="btn btn-primary btn-
lg">Continue Shopping</button>
                        </form>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## seller-orders.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ page import="java.util.List" %>
<%@ page import="com.onlineshop.model.Order" %>
<%@ page import="com.onlineshop.model.User" %>
<%@ page import="java.text.SimpleDateFormat" %>
<!DOCTYPE html>
<html>
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Seller Dashboard - Online Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .navbar-logo {
            height: 40px;
            margin-right: 10px;
        }
    </style>
</head>
<body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="#">
                <img src="${pageContext.request.contextPath}/images/logo.png"
alt="Shop Logo" class="navbar-logo">
                Seller Dashboard
            </a>
            <div class="collapse navbar-collapse" id="navbarNav">
                <div class="d-flex ms-auto">
                    <% User user = (User) session.getAttribute("user"); %>
                    <span class="navbar-text me-3">
                        Welcome, <%= user.getUsername() %>
                    </span>
                    <a href="../logout" class="btn btn-outline-light">Logout</a>
                </div>
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        <h2>All Orders</h2>

        <%
        List<Order> orders = (List<Order>) request.getAttribute("orders");
        if(orders != null && !orders.isEmpty()) {
            SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy
HH:mm");
        %>

        <div class="table-responsive">
            <table class="table table-striped">
                <thead>
                    <tr>
                        <th>Order ID</th>
                        <th>Customer ID</th>
                        <th>Date</th>
                        <th>Items Total</th>
                        <th>Shipping</th>
                        <th>Total</th>
```

```
                        <th>Tracking Number</th>
                        <th>Details</th>
                    </tr>
                </thead>
                <tbody>
                    <% for(Order order : orders) { %>
                    <tr>
                        <td><%= order.getOrderId() %></td>
                        <td><%= order.getUserId() %></td>
                        <td><%= dateFormat.format(order.getOrderDate()) %></td>
                        <td>€<%= String.format("%.2f", order.getTotalPrice()) %>
</td>
                        <td>€<%= String.format("%.2f", order.getShippingCost()) %>
</td>
                        <td>€<%= String.format("%.2f", order.getTotalPrice() +
order.getShippingCost()) %></td>
                        <td><%= order.getTrackingNumber() %></td>
                        <td>
                            <button class="btn btn-sm btn-outline-primary"
                                    data-bs-toggle="modal"
                                    data-bs-target="#orderModal<%=
order.getOrderId() %>">
                                View Details
                            </button>
                        </td>
                    </tr>
                    <% } %>
                </tbody>
            </table>
        </div>

        <!-- Order Details Modals -->
        <% for(Order order : orders) { %>
        <div class="modal fade" id="orderModal<%= order.getOrderId() %>"
tabindex="-1" aria-hidden="true">
            <div class="modal-dialog modal-lg">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title">Order #<%= order.getOrderId() %>
Details</h5>
                        <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                    </div>
                    <div class="modal-body">
                        <p><strong>Order ID:</strong> <%= order.getOrderId() %>
</p>
                        <p><strong>Customer ID:</strong> <%= order.getUserId() %>
</p>
                        <p><strong>Date:</strong> <%=
dateFormat.format(order.getOrderDate()) %></p>
                        <p><strong>Tracking Number:</strong> <%=
order.getTrackingNumber() %></p>
                        <p><strong>Total Price:</strong> €<%=
String.format("%.2f", order.getTotalPrice()) %></p>
```

```
                        <p><strong>Shipping Cost:</strong> €
                        <p><strong>Shipping Cost:</strong> €<%=
String.format("%.2f", order.getShippingCost()) %></p>
                        <p><strong>Final Price:</strong> €<%=
String.format("%.2f", order.getTotalPrice() + order.getShippingCost()) %></p>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
                    </div>
                </div>
            </div>
        </div>
        <% } %>

        <% } else { %>
        <div class="alert alert-info" role="alert">
            No orders found.
        </div>
        <% } %>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

customer-orders.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ page import="java.util.List" %>
<%@ page import="com.onlineshop.model.Order" %>
<%@ page import="com.onlineshop.model.User" %>
<%@ page import="java.text.SimpleDateFormat" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>My Orders - Online Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        .navbar-logo {
            height: 40px;
            margin-right: 10px;
        }
    </style>
```

```
</head>
<body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="#">
                <img src="${pageContext.request.contextPath}/images/logo.png"
alt="Shop Logo" class="navbar-logo">
            </a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarNav">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav me-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="products">Products</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="cart">Cart</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link active" href="my-orders">My Orders</a>
                    </li>
                </ul>
                <div class="d-flex">
                    <%
                    User user = (User) session.getAttribute("user");
                    if(user != null) {
                    %>
                    <span class="navbar-text me-3">
                        Welcome, <%= user.getUsername() %>
                    </span>
                    <a href="logout" class="btn btn-outline-light">Logout</a>
                    <% } else { %>
                    <a href="login" class="btn btn-outline-light">Login</a>
                    <% } %>
                </div>
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        <h2>My Orders</h2>

        <%
        List<Order> orders = (List<Order>) request.getAttribute("orders");
        if(orders != null && !orders.isEmpty()) {
            SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy
HH:mm");
        %>

        <div class="row">
            <% for(Order order : orders) { %>
```

```
            <div class="col-md-6 mb-4">
                <div class="card">
                    <div class="card-header bg-primary text-white d-flex justify-
content-between align-items-center">
                        <h5 class="mb-0">Order #<%= order.getOrderId() %></h5>
                        <span class="badge bg-light text-dark"><%=
dateFormat.format(order.getOrderDate()) %></span>
                    </div>
                    <div class="card-body">
                        <p><strong>Tracking Number:</strong> <%=
order.getTrackingNumber() %></p>
                        <p><strong>Items Total:</strong> €<%=
String.format("%.2f", order.getTotalPrice()) %></p>
                        <p><strong>Shipping Cost:</strong> €<%=
String.format("%.2f", order.getShippingCost()) %></p>
                        <p><strong>Total:</strong> €<%= String.format("%.2f",
order.getTotalPrice() + order.getShippingCost()) %></p>
                    </div>
                </div>
            </div>
            <% } %>
        </div>

        <% } else { %>
        <div class="alert alert-info" role="alert">
            You have no orders yet. <a href="products" class="alert-link">Start
shopping</a>
        </div>
        <% } %>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

error.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8" isErrorPage="true" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Error - Online Shop</title>
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
```

```
<body class="bg-light">
    <div class="container">
        <div class="row justify-content-center mt-5">
            <div class="col-md-6">
                <div class="card shadow">
                    <div class="card-header bg-danger text-white">
                        <h3 class="mb-0">Error</h3>
                    </div>
                    <div class="card-body text-center py-5">
                        <h1 class="display-1 text-danger mb-4">Oops!</h1>
                        <p class="lead">Something went wrong.</p>

                        <% if(response.getStatus() == 404) { %>
                            <p>The page you are looking for could not be found.
</p>
                        <% } else { %>
                            <p>We're experiencing some technical difficulties.</p>
                        <% } %>

                        <div class="mt-4">
                            <a href="login" class="btn btn-primary">Go to
Login</a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Bootstrap JS -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

## Configuración

### web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
         version="4.0">

    <display-name>Online Shop</display-name>

    <!-- Session Configuration -->
    <session-config>
        <session-timeout>30</session-timeout> <!-- 30 minutes -->
```

```
    </session-config>

    <!-- Welcome File List -->
    <welcome-file-list>
        <welcome-file>login</welcome-file>
    </welcome-file-list>

    <!-- Error Pages -->
    <error-page>
        <error-code>404</error-code>
        <location>/error.jsp</location>
    </error-page>

    <error-page>
        <error-code>500</error-code>
        <location>/error.jsp</location>
    </error-page>

</web-app>
```

## Base de Datos

### create_database.sql

```
-- Create database
CREATE DATABASE onlineshop;
USE onlineshop;

-- Create users table
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    user_type ENUM('customer', 'seller') NOT NULL DEFAULT 'customer'
);

-- Create products table
CREATE TABLE products (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    image_url VARCHAR(255)
);

-- Create orders table
CREATE TABLE orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    total_price DECIMAL(10, 2) NOT NULL,
```

```
    shipping_cost DECIMAL(10, 2) NOT NULL,
    tracking_number VARCHAR(20) NOT NULL UNIQUE,
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);

-- Create order_items table
CREATE TABLE order_items (
    order_item_id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    item_price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);

-- Insert sample products
INSERT INTO products (name, description, price, image_url) VALUES
('Ashe', 'Frost Archer with precise ice arrows. Classic marksman with global
ultimate ability.', 29.99, 'images/products/Ashe.avif'),
('Caitlyn', 'Piltover Sheriff with long-range sniper rifle. Known for precision
headshots and trap placements.', 34.99, 'images/products/Cait.avif'),
('Corki', 'Daring bombardier with his flying machine. Special delivery expert with
missile packages.', 24.99, 'images/products/Corki.avif'),
('Gwen', 'Enchanted seamstress with magical scissors. Once a doll, now animated
with the magic of the mist.', 39.99, 'images/products/Gwen.avif'),
('Jinx', 'Loose cannon with a love for explosions. Carries a variety of weapons
including her shark rocket launcher.', 42.99, 'images/products/Jinx.avif'),
('Kai\'Sa', 'Daughter of the void who survived and adapted. Wears a living void
suit with evolving abilities.', 44.99, 'images/products/Kaisa.avif'),
('Lux', 'Lady of Luminosity with light-bending powers. Demacian mage hiding her
abilities from persecution.', 32.99, 'images/products/Lux.avif'),
('Mel', 'Shrewd political mastermind from Noxus. Known for her diplomatic skills
and strategic thinking.', 27.99, 'images/products/Mel.avif'),
('Poro', 'Fluffy creature from the Freljord. Adorable companion known for their
love of Poro-Snax.', 19.99, 'images/products/Poro.avif'),
('Viktor', 'Machine herald pursuing glorious evolution. Augments himself with
technology to transcend human limitations.', 36.99,
'images/products/Viktor.avif'),
('Yone', 'Unforgotten swordsman and Yasuo\'s half-brother. Wields two blades after
returning from death.', 46.99, 'images/products/Yone.avif'),
('Yuumi', 'Magical cat with a floating book companion. Attaches to allies to
provide support and protection.', 22.99, 'images/products/Yuumi.avif');

-- Insert a seller account
INSERT INTO users (username, password, user_type) VALUES
('admin', 'admin123', 'seller');
```