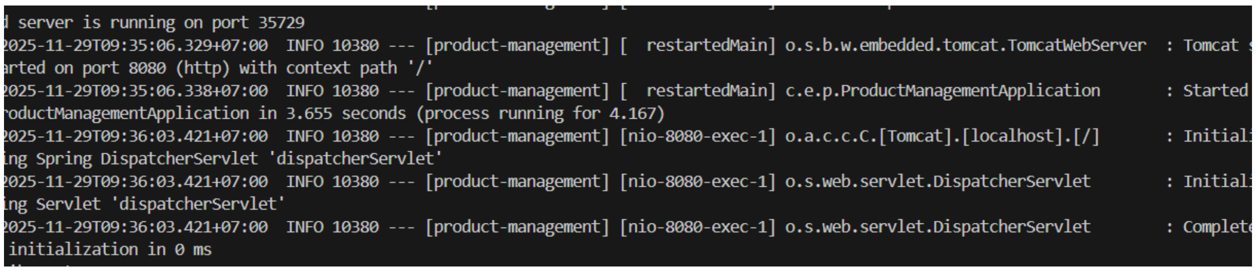
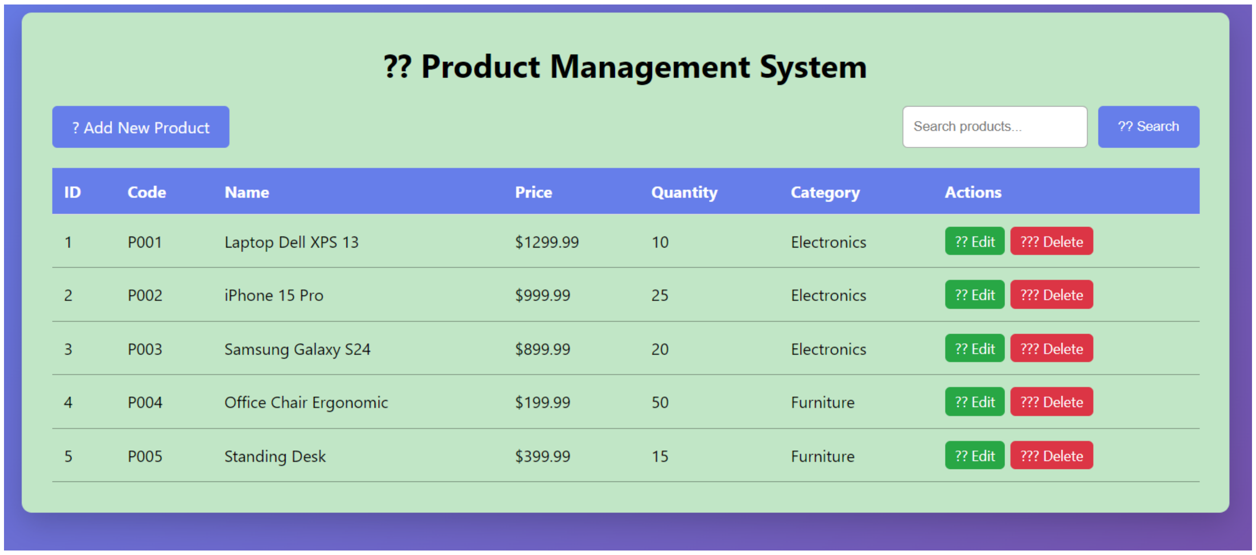
Successful Initialization in console, Entity and Repository work correctly.



```
d server is running on port 35729
2025-11-29T09:35:06.329+07:00  INFO 10380 --- [product-management] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat s
arted on port 8080 (http) with context path '/'
2025-11-29T09:35:06.338+07:00  INFO 10380 --- [product-management] [  restartedMain] c.e.p.ProductManagementApplication       : Started
roductManagementApplication in 3.655 seconds (process running for 4.167)
2025-11-29T09:36:03.421+07:00  INFO 10380 --- [product-management] [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initiali
ing Spring DispatcherServlet 'dispatcherServlet'
2025-11-29T09:36:03.421+07:00  INFO 10380 --- [product-management] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initiali
ing Servlet 'dispatcherServlet'
2025-11-29T09:36:03.421+07:00  INFO 10380 --- [product-management] [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Complete
 initialization in 0 ms
```



ID	Code	Name	Price	Quantity	Category	Actions
1	P001	Laptop Dell XPS 13	$1299.99	10	Electronics	?? Edit ??? Delete
2	P002	iPhone 15 Pro	$999.99	25	Electronics	?? Edit ??? Delete
3	P003	Samsung Galaxy S24	$899.99	20	Electronics	?? Edit ??? Delete
4	P004	Office Chair Ergonomic	$199.99	50	Furniture	?? Edit ??? Delete
5	P005	Standing Desk	$399.99	15	Furniture	?? Edit ??? Delete

The code used to design the page can be found in templates/product-list.html

The product list page shows the list of products, available. You can delete, edit, search and add new products here.

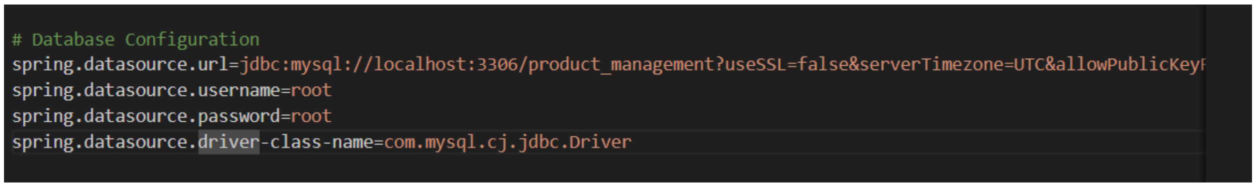
:8080/products send a GET request to the ProductController.java when accessed

(@Controller

@RequestMapping("/products"))

@GetMapping in ProductController directs the user to webpages based on what is written after it, such as the new product form, edit product form, or just the product list if it is empty. It also calls for the value that will be used in the pages

It uses data from the local mySQL, connected through the line in application.properties

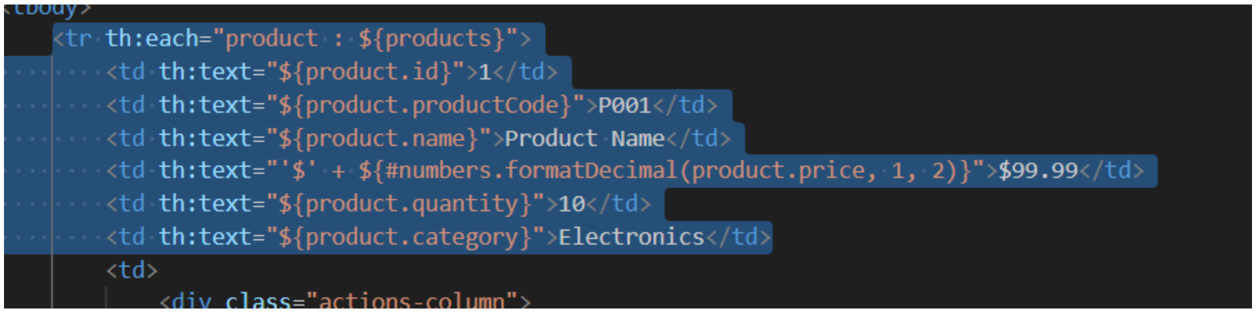


```
# Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/product_management?useSSL=false&serverTimezone=UTC&allowPublicKey
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

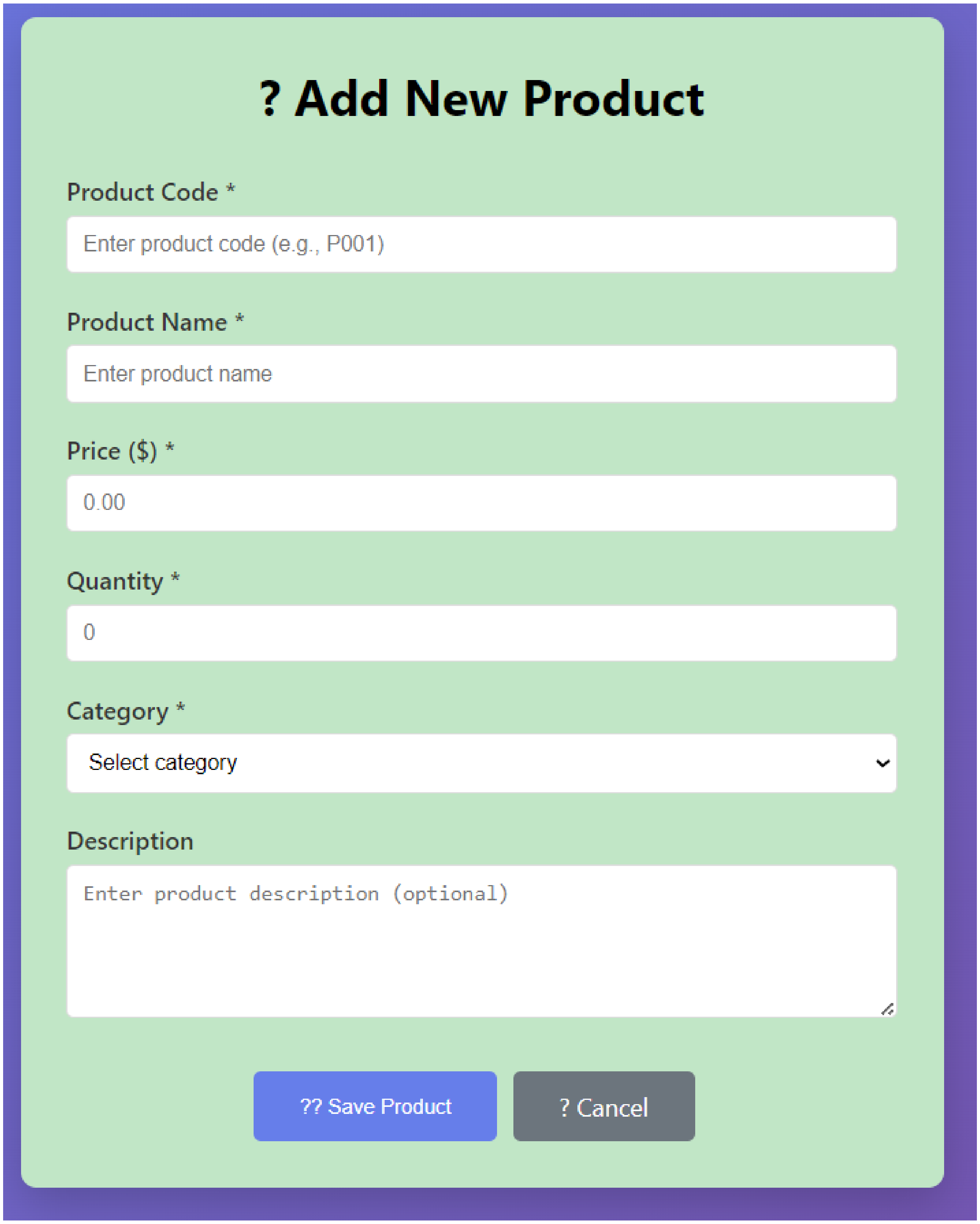
List<Product> products = productService.getAllProducts(); requests the data of all of the product in the form of a list, then getAllProducts() uses the findAll() method with the productRepository inherited from JpaRepository

productRepository is just an interface that extends JpaRepository to inherit its methods

The name and content of the products are defined by the line



```
<tbody>
    <tr th:each="product : ${products}">
        <td th:text="${product.id}">1</td>
        <td th:text="${product.productCode}">P001</td>
        <td th:text="${product.name}">Product Name</td>
        <td th:text="'$' + ${#numbers.formatDecimal(product.price, 1, 2)}">$99.99</td>
        <td th:text="${product.quantity}">10</td>
        <td th:text="${product.category}">Electronics</td>
        <td>
            <div class="actions-column">
```


? Add New Product

Product Code *

Enter product code (e.g., P001)

Product Name *

Enter product name

Price ($) *

0.00

Quantity *

0

Category *

Select category

Description

Enter product description (optional)

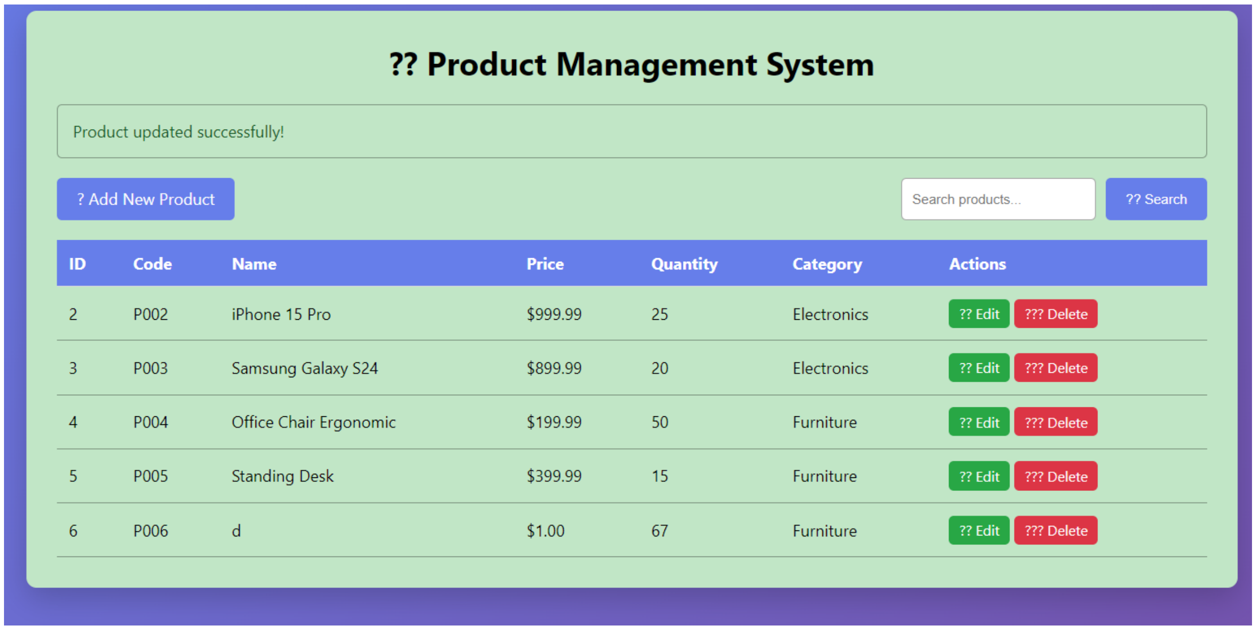
Pressing add new product directs the user to the new product page (th:href="@{/products/new}")

```
@GetMapping("/new")
  public String showNewForm(Model model) {
    Product product = new Product();
    model.addAttribute("product", product);
    return "product-form";
  }
```

Creates a new product, adding product attribute

And then directs to product-form.html

product-form.html uses a form for user input, pressing save product sends the user to the product list with the new product saved(th:action="@{/products/save}") and pressing cancel send the user to the product list without anything changed(th:href="@{/products}").

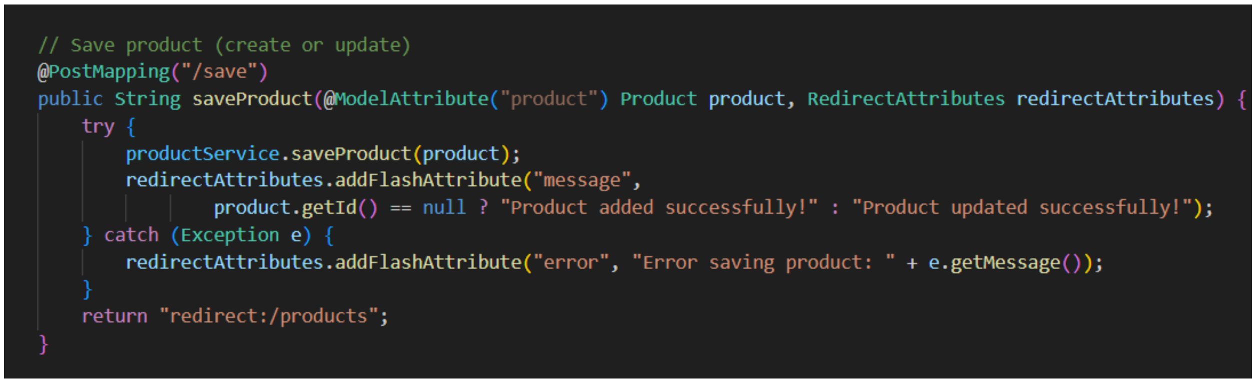


?? Product Management System

Product updated successfully!

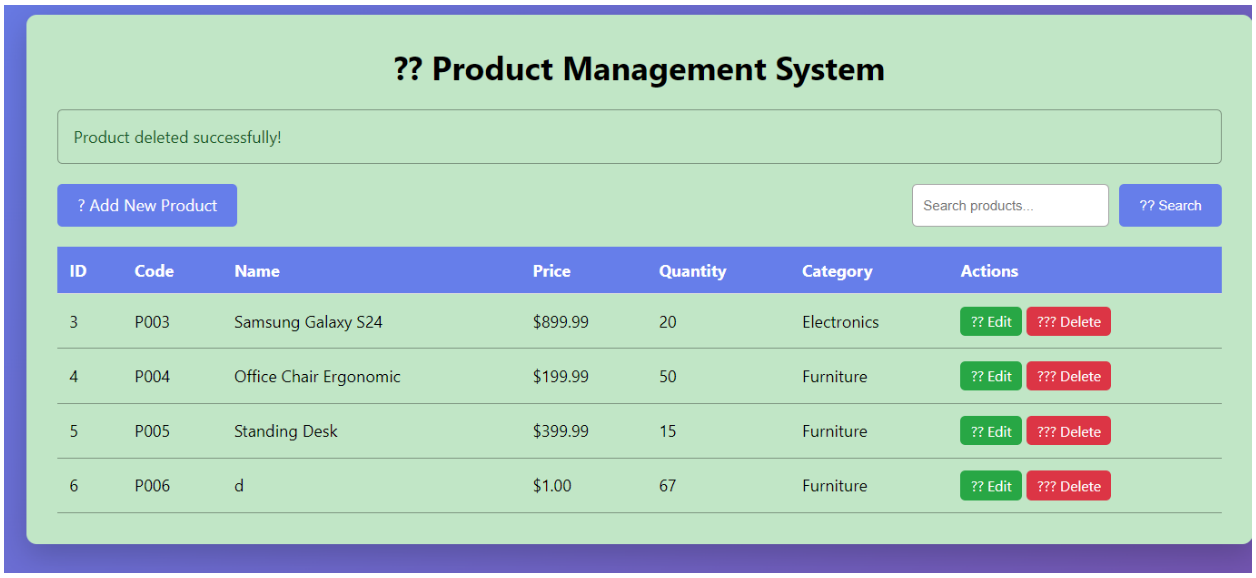
ID	Code	Name	Price	Quantity	Category	Actions
2	P002	iPhone 15 Pro	$999.99	25	Electronics	?? Edit ??? Delete
3	P003	Samsung Galaxy S24	$899.99	20	Electronics	?? Edit ??? Delete
4	P004	Office Chair Ergonomic	$199.99	50	Furniture	?? Edit ??? Delete
5	P005	Standing Desk	$399.99	15	Furniture	?? Edit ??? Delete
6	P006	d	$1.00	67	Furniture	?? Edit ??? Delete

Product list with a new product saved, the result is either success or error, checked by the controller



```java
// Save product (create or update)
@PostMapping("/save")
public String saveProduct(@ModelAttribute("product") Product product, RedirectAttributes redirectAttributes) {
    try {
        productService.saveProduct(product);
        redirectAttributes.addFlashAttribute("message",
                product.getId() == null ? "Product added successfully!" : "Product updated successfully!");
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("error", "Error saving product: " + e.getMessage());
    }
    return "redirect:/products";
}
```

The notification at the top of the list is also determined by this

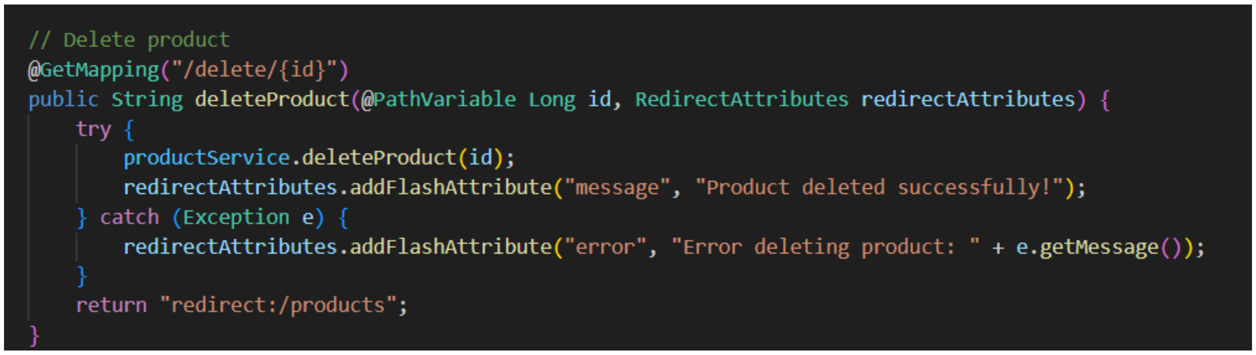


?? Product Management System

Product deleted successfully!

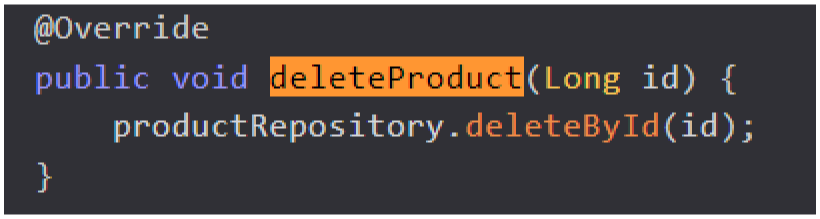
ID	Code	Name	Price	Quantity	Category	Actions
3	P003	Samsung Galaxy S24	$899.99	20	Electronics	?? Edit ??? Delete
4	P004	Office Chair Ergonomic	$199.99	50	Furniture	?? Edit ??? Delete
5	P005	Standing Desk	$399.99	15	Furniture	?? Edit ??? Delete
6	P006	d	$1.00	67	Furniture	?? Edit ??? Delete

Pressing delete directs the user to the product list with the ID of the pressed product (th:href="@{/products/delete/{id}(id=${product.id})})



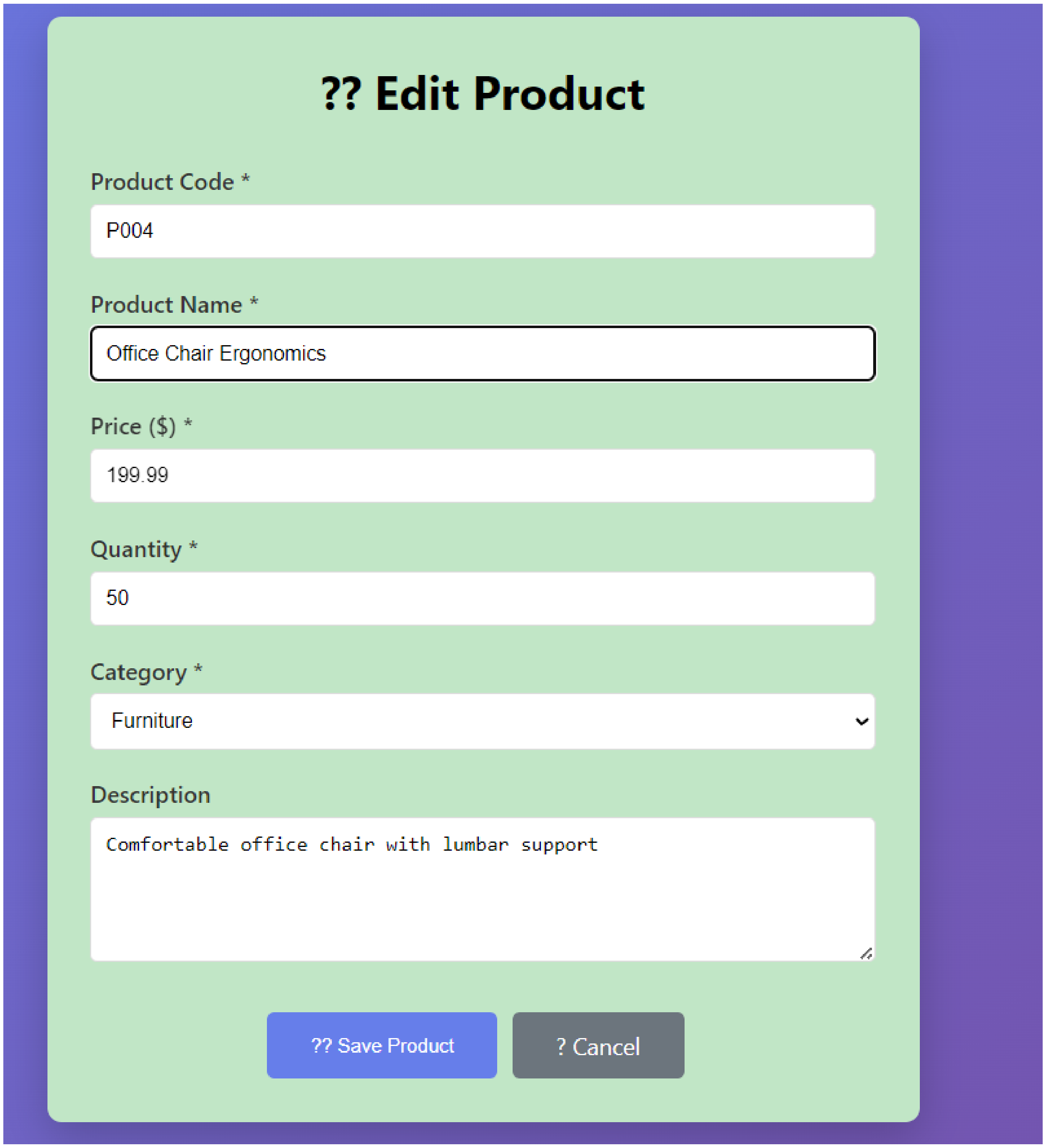
```java
// Delete product
@GetMapping("/delete/{id}")
public String deleteProduct(@PathVariable Long id, RedirectAttributes redirectAttributes) {
    try {
        productService.deleteProduct(id);
        redirectAttributes.addFlashAttribute("message", "Product deleted successfully!");
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("error", "Error deleting product: " + e.getMessage());
    }
    return "redirect:/products";
}
```

GetMapping then tries to use the deleteProduct() method, sending a message to the user at the top of the list of it being successful/an error



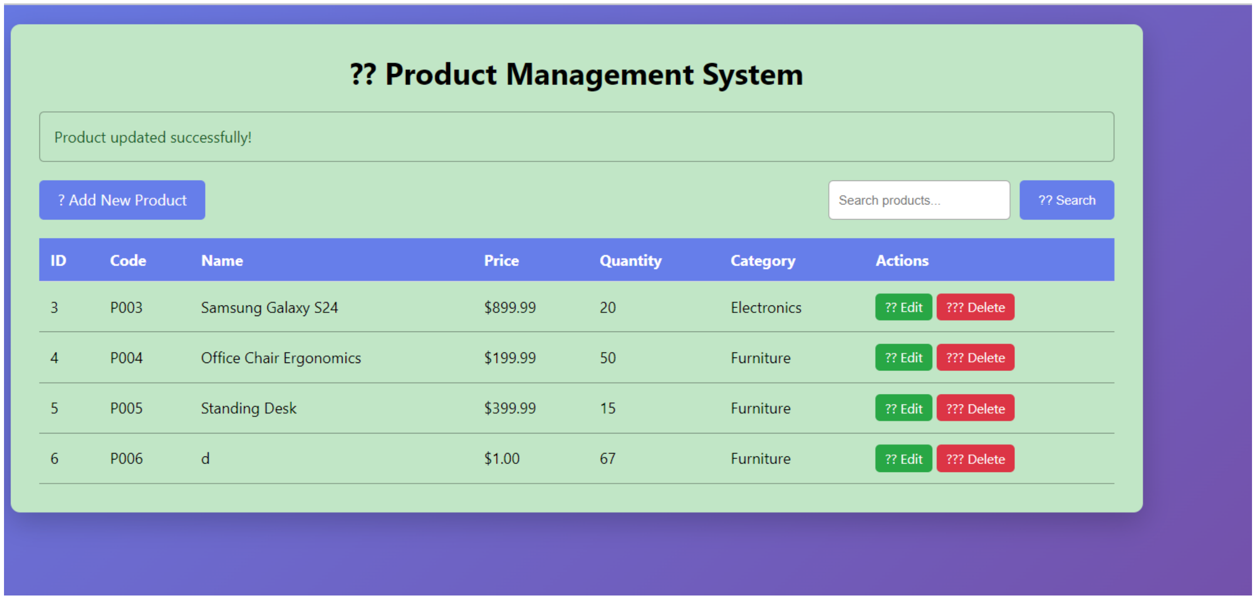
```
@Override
public void deleteProduct(Long id) {
    productRepository.deleteById(id);
}
```

deleteProduct uses the deleteById method inherited from JpaRepository by calling productRepository to delete a product based on the id.



?? Edit Product

Product Code *	P004
Product Name *	Office Chair Ergonomics
Price ($) *	199.99
Quantity *	50
Category *	Furniture
Description	Comfortable office chair with lumbar support



?? Product Management System

Product updated successfully!

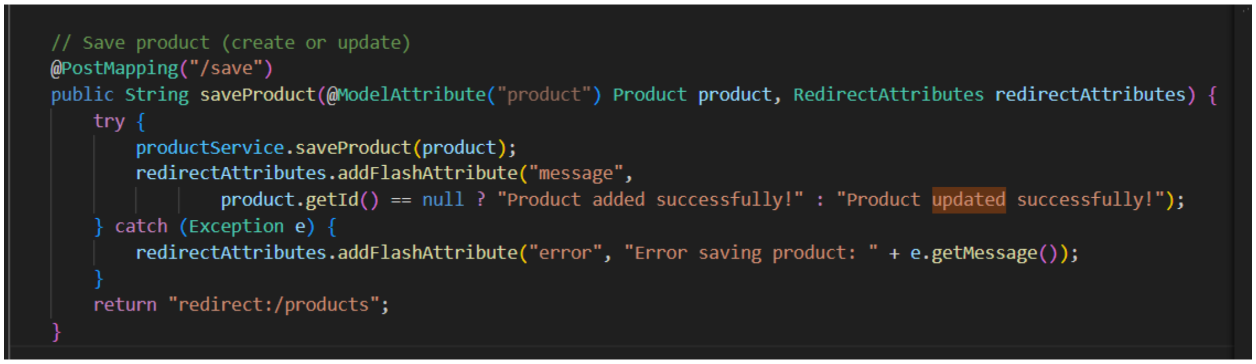
ID	Code	Name	Price	Quantity	Category	Actions
3	P003	Samsung Galaxy S24	$899.99	20	Electronics	?? Edit ??? Delete
4	P004	Office Chair Ergonomics	$199.99	50	Furniture	?? Edit ??? Delete
5	P005	Standing Desk	$399.99	15	Furniture	?? Edit ??? Delete
6	P006	d	$1.00	67	Furniture	?? Edit ??? Delete

Updating a product successfully, pressing on edit will send the user to the product form with the product id(th:href="@{/products/edit/{id}(id=${product.id})}")

@GetMapping("/edit/{id}")

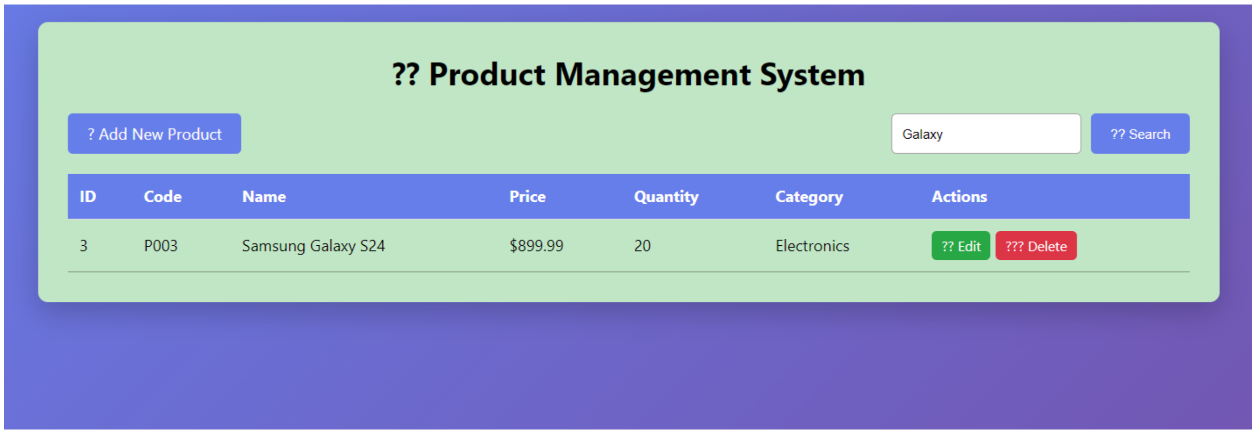
Checks if the product with the id exists, sending an error if it does not

Similar to add new product, saving the product directs to the product list with new information saved, cancel returns to the page unchanged



```java
// Save product (create or update)
@PostMapping("/save")
public String saveProduct(@ModelAttribute("product") Product product, RedirectAttributes redirectAttributes) {
    try {
        productService.saveProduct(product);
        redirectAttributes.addFlashAttribute("message",
                product.getId() == null ? "Product added successfully!" : "Product updated successfully!");
    } catch (Exception e) {
        redirectAttributes.addFlashAttribute("error", "Error saving product: " + e.getMessage());
    }
    return "redirect:/products";
}
```

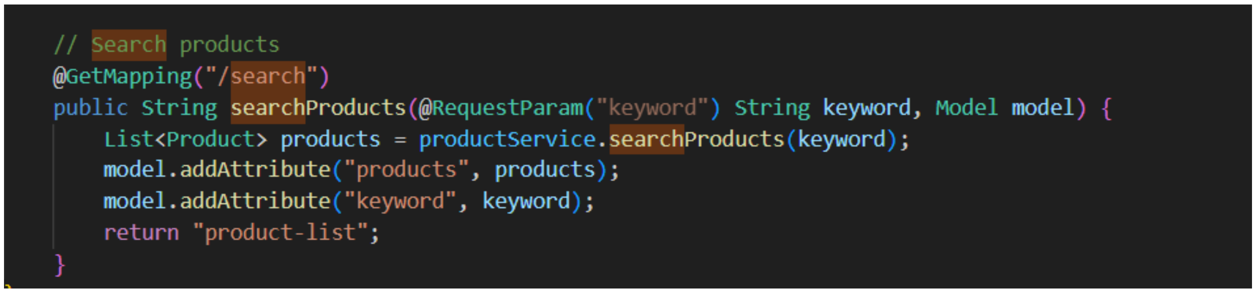
Saves the product, using addFlashAttribute to send an error/success message.



?? Product Management System

ID	Code	Name	Price	Quantity	Category	Actions
3	P003	Samsung Galaxy S24	$899.99	20	Electronics	?? Edit ??? Delete

The search bar is a form of its own, pressing the search button will send the user to the product list page again(th:action="@{/products/search}")



```
// Search products
@GetMapping("/search")
public String searchProducts(@RequestParam("keyword") String keyword, Model model) {
    List<Product> products = productService.searchProducts(keyword);
    model.addAttribute("products", products);
    model.addAttribute("keyword", keyword);
    return "product-list";
}
```

Request the keyword parameter entered into the form, then use the searchProducts() to search for the products, which uses findByNameContaining inherited from JpaRepository to search for products matching the search input.

GetMapping then directs the product list after adding the searched products and the entered keyword as attributes, making the data available to view.