

REPORT EXERCISE LAB 7

Student Name: Nguyễn Tiến Anh

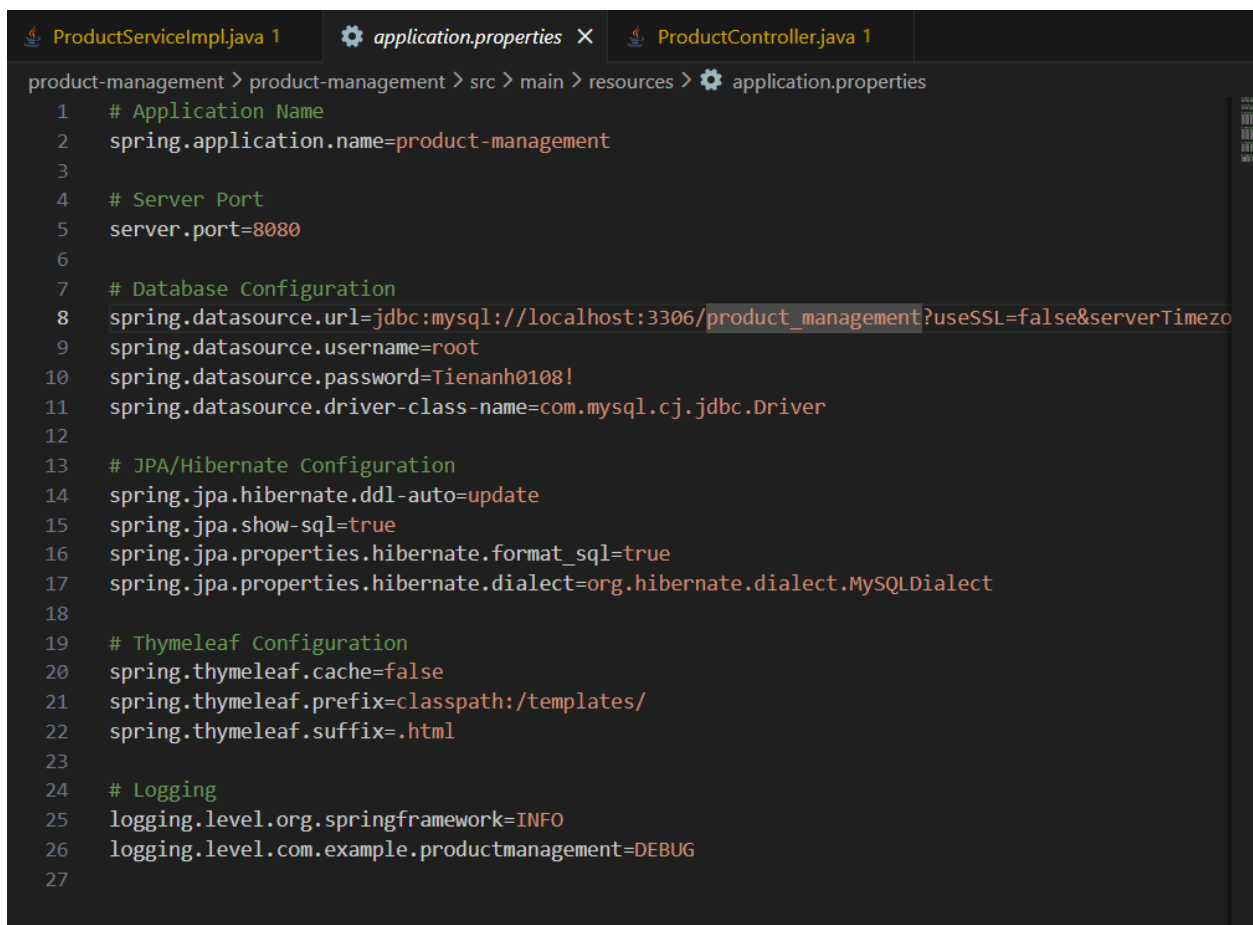
Student ID: ITITDK22128

Course: Web Application Development

Lab 7: SPRING BOOT & JPA CRUD

Github: https://github.com/TienAnh0108/Lab7_Exercise.git

Ex 1:

The image shows a screenshot of an IDE with three tabs: 'ProductServiceImpl.java', 'application.properties', and 'ProductController.java'. The 'application.properties' tab is active, showing the following configuration:

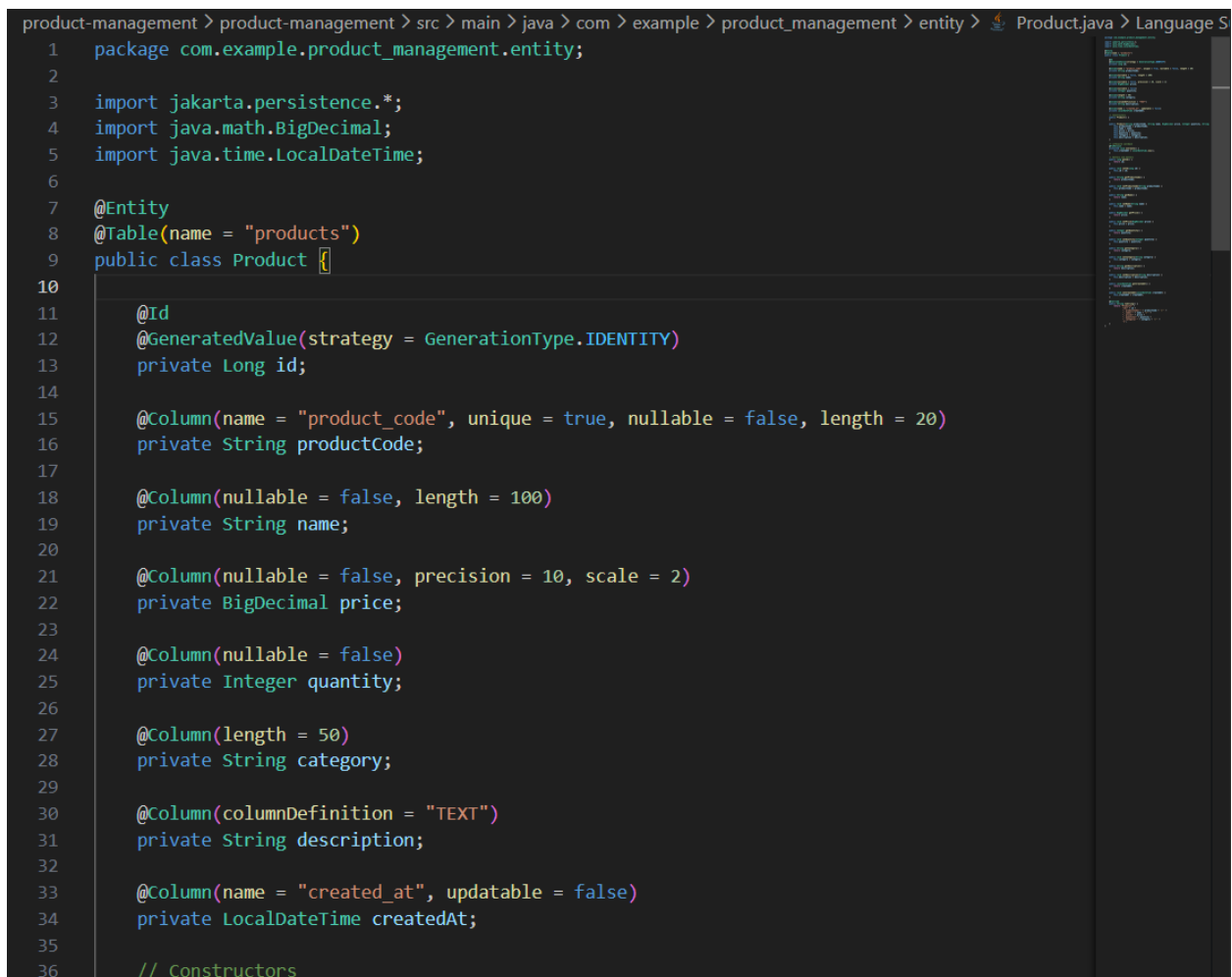
```
product-management > product-management > src > main > resources > application.properties
1  # Application Name
2  spring.application.name=product-management
3
4  # Server Port
5  server.port=8080
6
7  # Database Configuration
8  spring.datasource.url=jdbc:mysql://localhost:3306/product_management?useSSL=false&serverTimezo
9  spring.datasource.username=root
10 spring.datasource.password=Tienanh0108!
11 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
12
13 # JPA/Hibernate Configuration
14 spring.jpa.hibernate.ddl-auto=update
15 spring.jpa.show-sql=true
16 spring.jpa.properties.hibernate.format_sql=true
17 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
18
19 # Thymeleaf Configuration
20 spring.thymeleaf.cache=false
21 spring.thymeleaf.prefix=classpath:/templates/
22 spring.thymeleaf.suffix=.html
23
24 # Logging
25 logging.level.org.springframework=INFO
26 logging.level.com.example.productmanagement=DEBUG
27
```

File: application.properties **Role:** Defines the runtime environment settings.

- **Database:** Configures the MySQL connection details (url, username, password) used by the Hikari connection pool.

- **JPA/Hibernate:** spring.jpa.hibernate.ddl-auto=update allows Hibernate to automatically modify the table structure based on entity changes, simplifying development.
- **Thymeleaf:** Disables caching (spring.thymeleaf.cache=false) for fast template reloading during development.

Ex 2:



```

product-management > product-management > src > main > java > com > example > product_management > entity > Product.java > Language S
1  package com.example.product_management.entity;
2
3  import jakarta.persistence.*;
4  import java.math.BigDecimal;
5  import java.time.LocalDateTime;
6
7  @Entity
8  @Table(name = "products")
9  public class Product {
10
11      @Id
12      @GeneratedValue(strategy = GenerationType.IDENTITY)
13      private Long id;
14
15      @Column(name = "product_code", unique = true, nullable = false, length = 20)
16      private String productCode;
17
18      @Column(nullable = false, length = 100)
19      private String name;
20
21      @Column(nullable = false, precision = 10, scale = 2)
22      private BigDecimal price;
23
24      @Column(nullable = false)
25      private Integer quantity;
26
27      @Column(length = 50)
28      private String category;
29
30      @Column(columnDefinition = "TEXT")
31      private String description;
32
33      @Column(name = "created_at", updatable = false)
34      private LocalDateTime createdAt;
35
36      // Constructors

```

File: Product.java **Role:** Represents the data structure and the corresponding database table.

- **Mapping:**
 - @Entity, @Table(name = "products"): Maps this class to the products table.
 - @Id, @GeneratedValue(strategy = GenerationType.IDENTITY): Defines the primary key with auto-increment generation.

- **Fields:** Contains essential product fields: productCode, name, price, quantity, category, and description.
- **Timestamp:** Uses @PrePersist and @Column(name = "created_at", updatable = false) to automatically record the creation time when the entity is saved for the first time.

```
product-management > product-management > src > main > java > com > example > product_management > repository > ProductRepository.java >
1  package com.example.product_management.repository;
2
3  import com.example.product_management.entity.Product;
4  import org.springframework.data.jpa.repository.JpaRepository;
5  import org.springframework.stereotype.Repository;
6
7  import java.math.BigDecimal;
8  import java.util.List;
9
10 @Repository
11 public interface ProductRepository extends JpaRepository<Product, Long> {
12
13     // Spring Data JPA generates implementation automatically!
14
15     // Custom query methods (derived from method names)
16     List<Product> findByCategory(String category);
17
18     List<Product> findByNameContaining(String keyword);
19
20     List<Product> findByPriceBetween(BigDecimal minPrice, BigDecimal maxPrice);
21
22     List<Product> findByCategoryOrderByPriceAsc(String category);
23
24     boolean existsByProductCode(String productCode);
25
26     // All basic CRUD methods inherited from JpaRepository:
27     // - findAll()
28     // - findById(Long id)
29     // - save(Product product)
30     // - deleteById(Long id)
31     // - count()
32     // - existsById(Long id)
33 }
```

File: ProductRepository.java **Role:** Directly interfaces with the database.

- **Inheritance:** Extends **JpaRepository<Product, Long>**, which automatically provides basic CRUD methods (findAll, save, findById, deleteById) without requiring boilerplate code.
- **Custom Queries (Derived Queries):**
 - findByCategory(String category): Searches products based on their category.
 - findByNameContaining(String keyword): Searches by a substring within the product name (used by the Controller for searching).

- existsByProductCode(String productCode): Checks if a product code already exists

Ex 3:

```
product-management > product-management > src > main > java > com > example > product_management > service > ProductService.java > Java
1  package com.example.product_management.service;
2
3  import com.example.product_management.entity.Product;
4
5  import java.util.List;
6  import java.util.Optional;
7
8  public interface ProductService {
9
10     List<Product> getAllProducts();
11
12     Optional<Product> getProductById(Long id);
13
14     Product saveProduct(Product product);
15
16     void deleteProduct(Long id);
17
18     List<Product> searchProducts(String keyword);
19
20     List<Product> getProductsByCategory(String category);
21 }
```

```
product-management > product-management > src > main > java > com > example > product_management > service > ProductServiceImpl.java > ...
1  package com.example.product_management.service;
2
3  import com.example.product_management.entity.Product;
4  import com.example.product_management.repository.ProductRepository;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.stereotype.Service;
7  import org.springframework.transaction.annotation.Transactional;
8
9  import java.util.List;
10 import java.util.Optional;
11
12 @Service
13 @Transactional
14 public class ProductServiceImpl implements ProductService {
15
16     private final ProductRepository productRepository;
17
18     @Autowired
19     public ProductServiceImpl(ProductRepository productRepository) {
20         this.productRepository = productRepository;
21     }
22
23     @Override
24     public List<Product> getAllProducts() {
25         return productRepository.findAll();
26     }
27
28     @Override
29     public Optional<Product> getProductById(Long id) {
30         return productRepository.findById(id);
31     }
32
33     @Override
34     public Product saveProduct(Product product) {
35         // Validation logic can go here
36         return productRepository.save(product);
37     }
38 }
```

```

39     @Override
40     public void deleteProduct(Long id) {
41         productRepository.deleteById(id);
42     }
43
44     @Override
45     public List<Product> searchProducts(String keyword) {
46         return productRepository.findByNameContaining(keyword);
47     }
48
49     @Override
50     public List<Product> getProductsByCategory(String category) {
51         return productRepository.findByCategory(category);
52     }
53 }

```

Files: ProductService.java (Interface) & ProductServiceImpl.java (Implementation) **Role:** Holds the core business logic, acting as the intermediary between the Controller and the Repository.

- **Core Functionality:**
 - getAllProducts(): Retrieves all products.
 - getProductById(Long id): Retrieves a product by ID (returning an Optional).
 - saveProduct(Product product): Executes the insert or update operation.
 - deleteProduct(Long id): Executes the deletion operation.
 - searchProducts(String keyword): Searches products by name.
- **Data Integrity:** The ProductServiceImpl class uses **@Transactional** to ensure that database operations are executed safely and atomically (all or nothing)

Ex 4:

```
product-management > product-management > src > main > java > com > example > product_management > controller > ProductController.java
1  package com.example.product_management.controller;
2
3  import com.example.product_management.entity.Product;
4  import com.example.product_management.service.ProductService;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.stereotype.Controller;
7  import org.springframework.ui.Model;
8  import org.springframework.web.bind.annotation.*;
9  import org.springframework.web.servlet.mvc.support.RedirectAttributes;
10
11  import java.util.List;
12
13  @Controller
14  @RequestMapping("/products")
15  public class ProductController {
16
17      private final ProductService productService;
18
19      @Autowired
20      public ProductController(ProductService productService) {
21          this.productService = productService;
22      }
23
24      // List all products
25      @GetMapping
26      public String listProducts(Model model) {
27          List<Product> products = productService.getAllProducts();
28          model.addAttribute("products", products);
29          return "product-list"; // Returns product-list.html
30      }
31
32      // Show form for new product
33      @GetMapping("/new")
34      public String showNewForm(Model model) {
35          Product product = new Product();
36          model.addAttribute("product", product);
37          return "product-form";
38      }
39
40      // Show form for editing product
41      @GetMapping("/edit/{id}")
42      public String showEditForm(@PathVariable Long id, Model model, RedirectAttributes redirectAttributes) {
43          return productService.getProductById(id)
44              .map(product -> {
45                  model.addAttribute("product", product);
46                  return "product-form";
47              })
48              .orElseGet(() -> {
49                  redirectAttributes.addFlashAttribute("error", "Product not found");
50                  return "redirect:/products";
51              });
52      }
53
54      // Save product (create or update)
55      @PostMapping("/save")
56      public String saveProduct(@ModelAttribute("product") Product product, RedirectAttributes redirectAttributes) {
57          try {
58              productService.saveProduct(product);
59              redirectAttributes.addFlashAttribute("message",
60                  product.getId() == null ? "Product added successfully!" : "Product updated");
61          } catch (Exception e) {
62              redirectAttributes.addFlashAttribute("error", "Error saving product: " + e.getMessage());
63          }
64          return "redirect:/products";
65      }
66
67      // Delete product
68      @GetMapping("/delete/{id}")
69      public String deleteProduct(@PathVariable Long id, RedirectAttributes redirectAttributes) {
70          try {
71              productService.deleteProduct(id);
72              redirectAttributes.addFlashAttribute("message", "Product deleted successfully!");
73          } catch (Exception e) {
74              redirectAttributes.addFlashAttribute("error", "Error deleting product: " + e.getMessage());
75          }
76          return "redirect:/products";
77      }
78  }
```

```

// 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";
}

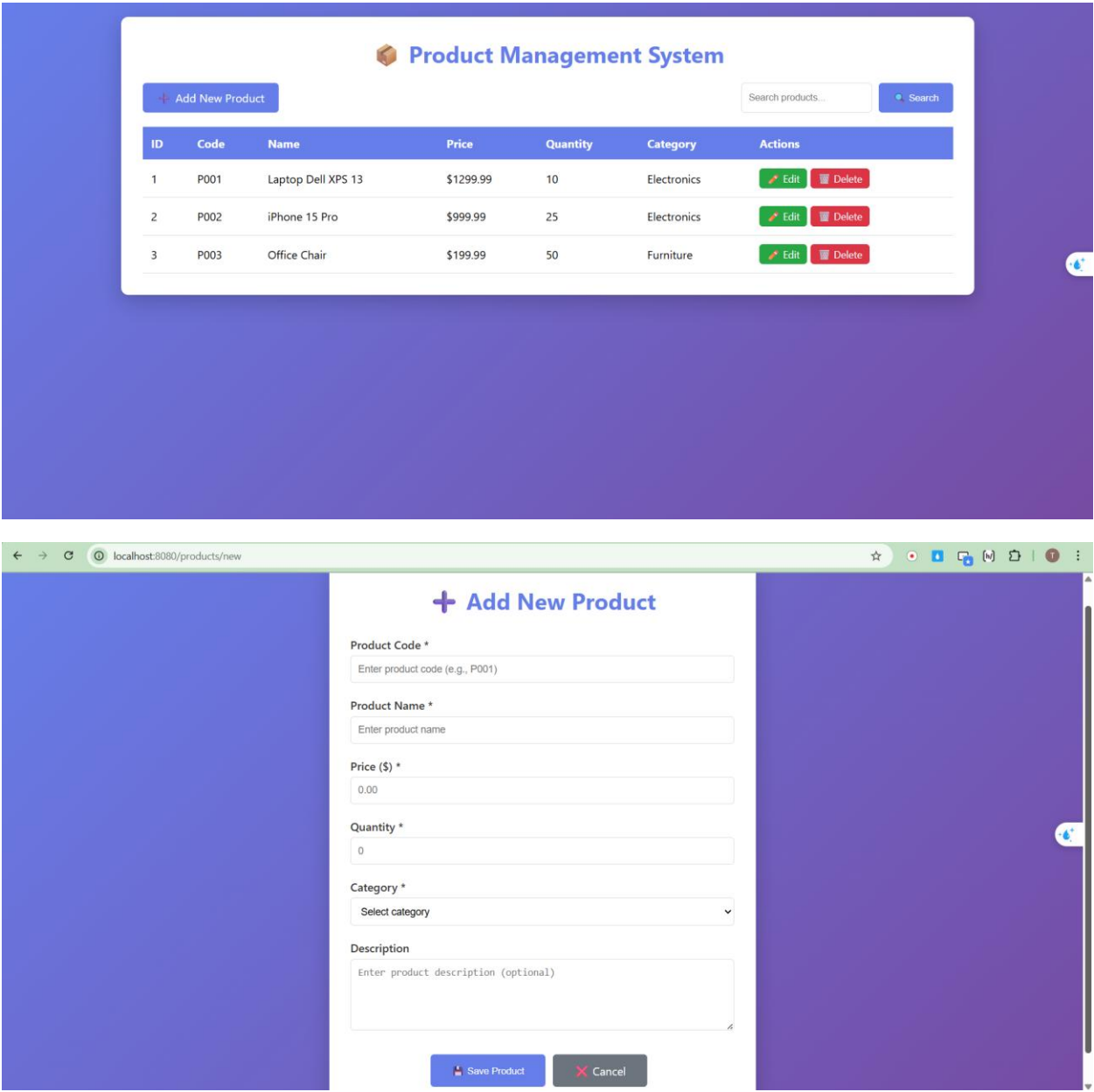
// Search products
@GetMapping("/search")
public String searchProducts(@RequestParam("keyword") String keyword, Model model) {
    List<Product> products = productService.searchProducts(keyword);
    model.addAttribute("products", products);
    model.addAttribute("keyword", keyword);
    return "product-list";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

```


File: ProductController.java **Role:** Handles HTTP requests and manages the application flow.


- **Class Mapping:** @RequestMapping("/products") – All URLs starting with /products are handled here.
- **List Products:** @GetMapping (/products) – Fetches all products, adds them to the Model, and returns the **product-list** view.
- **New Form:** @GetMapping("/new") – Prepares an empty Product object and returns the **product-form** view.
- **Edit Form:** @GetMapping("/edit/{id}") – Finds the product by ID; if found, it loads the product into the **product-form**; otherwise, it redirects with an error message.
- **Save (Create/Update):** @PostMapping("/save") – Receives the submitted Product object (@ModelAttribute), calls productService.saveProduct(), and redirects to /products with a success message.
- **Delete:** @GetMapping("/delete/{id}") – Calls productService.deleteProduct(), then redirects to /products with a success message.
- **Search:** @GetMapping("/search") – Receives a keyword (@RequestParam("keyword")), calls productService.searchProducts(), and returns the filtered results in the **product-list** view.


RESULT:





← → ↻ localhost:8080/products/search?keyword=Laptop ☆ 🇯🇵 📄 🗨️ 📁 📌 ⋮


 **Product Management System**

 Add New Product

 Search

ID	Code	Name	Price	Quantity	Category	Actions
1	P001	Laptop Dell XPS 13	\$1299.99	10	Electronics	 Edit  Delete

← → ↻ localhost:8080/products/edit/1 ☆ 🇯🇵 📄 🗨️ 📁 📌 ⋮

 **Edit Product**

Product Code *

Product Name *

Price (\$) *

Quantity *

Category *

Description