

Product.java

```
package com.example.product_api.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "products")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String productName;
    private String category;
    private Double price;
    private int quantity;

}
```

ProductDTO.java

```
package com.example.product_api.dto;

import jakarta.validation.constraints.Min;
import jakarta.validation.constraints.NotBlank;
```

```
import lombok.Data;

@Data
public class ProductDTO {

    @NotBlank(message = "Product name cannot be empty")
    private String productName;

    @NotBlank(message = "Category cannot be empty")
    private String category;

    @Min(value = 1, message = "Price must be positive")
    private Double price;

    @Min(value = 1, message = "Quantity must be positive")
    private int quantity;
}
```

ProductRepository.java

```
package com.example.product_api.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.product_api.entity.Product;
import java.util.List;
import java.util.Optional;

@Repository
public interface ProductRepository extends JpaRepository<Product,
Long> {

    List<Product> findByCategoryIgnoreCase(String category);
    Optional<Product> findByProductNameIgnoreCase(String productName);
```

```
}
```

ProductService.java

```
package com.example.product_api.service;

import java.util.List;

import com.example.product_api.dto.ProductDTO;

public interface ProductService {

    List<ProductDTO> getAllProducts();
    List<ProductDTO> searchProductsByCategory(String category);
    ProductDTO addProduct(ProductDTO productDTO);
    void deleteProductByName(String name);

}
```

ProductServiceImpl.java

```
package com.example.product_api.service.impl;

import java.util.List;
import java.util.stream.Collectors;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.product_api.dto.ProductDTO;
import com.example.product_api.entity.Product;
import com.example.product_api.repository.ProductRepository;
import com.example.product_api.service.ProductService;

@Service

```

```
public class ProductServiceImpl implements ProductService {  
  
    @Autowired  
    private ProductRepository productRepository;  
  
    private ProductDTO convertToDTO(Product product) {  
        ProductDTO dto = new ProductDTO();  
        dto.setProductName(product.getProductName());  
        dto.setCategory(product.getCategory());  
        dto.setPrice(product.getPrice());  
        dto.setQuantity(product.getQuantity());  
        return dto;  
    }  
  
    @Override  
    public ProductDTO addProduct(ProductDTO productDTO) {  
  
        Product product = new Product();  
        product.setProductName(productDTO.getProductName());  
        product.setCategory(productDTO.getCategory());  
        product.setPrice(productDTO.getPrice());  
        product.setQuantity(productDTO.getQuantity());  
  
        Product savedProduct = productRepository.save(product);  
  
        return convertToDTO(savedProduct);  
    }  
  
    @Override  
    public List<ProductDTO> getAllProducts() {  
        return productRepository.findAll()  
            .stream()  
            .map(this::convertToDTO)  
            .collect(Collectors.toList());  
    }  
}
```

```
@Override
public List<ProductDTO> searchProductsByCategory(String category)
{
    return productRepository.findByCategoryIgnoreCase(category)
        .stream()
        .map(this::convertToDTO)
        .collect(Collectors.toList());
}

@Override
public void deleteProductByName(String name) {
    Product product =
productRepository.findByProductNameIgnoreCase(name)
        .orElseThrow(() -> new RuntimeException("Product not
found with name: " + name));
    productRepository.delete(product);
}
}
```

ProductController.java

```
package com.example.product_api.controller;

import com.example.product_api.dto.ProductDTO;
import com.example.product_api.service.ProductService;
import jakarta.validation.Valid;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/products")
public class ProductController {

    @Autowired
```

```
private ProductService productService;

@PostMapping
public ResponseEntity<ProductDTO> addProduct(@Valid @RequestBody
ProductDTO productDTO) {
    ProductDTO newProductDTO =
productService.addProduct(productDTO);
    return new ResponseEntity<>(newProductDTO,
HttpStatus.CREATED);
}

@GetMapping
public ResponseEntity<List<ProductDTO>> getAllProducts() {
    List<ProductDTO> products = productService.getAllProducts();
    return ResponseEntity.ok(products);
}

@GetMapping("/category/{category}")
public ResponseEntity<List<ProductDTO>>
getProductsByCategory(@PathVariable String category) {
    List<ProductDTO> products =
productService.searchProductsByCategory(category);
    return ResponseEntity.ok(products);
}

@DeleteMapping("/{name}")
public ResponseEntity<String> deleteProductByName(@PathVariable
String name) {
    try {
        productService.deleteProductByName(name);
        return ResponseEntity.ok("Product '" + name + "' deleted
successfully.");
    } catch (RuntimeException e) {
        return new ResponseEntity<>(e.getMessage(),
HttpStatus.NOT_FOUND);
    }
}
```

LAB 04

EC/2021/006

W.K.G.K JAYAWARDANA

The screenshot shows the Postman application interface. On the left, the sidebar displays collections, environments, flows, and history. The main workspace shows a collection named "ApeBodima Api" with several endpoints listed under "Vehicle Api". A specific POST request titled "Create a new vehicle" is selected. The request details show a POST method to "http://localhost:8080/api/products". The request body is set to "JSON" and contains the following JSON payload:

```
1 {  
2   "productName": "Gaming Mouse",  
3   "category": "Electronics",  
4   "price": 75.50,  
5   "quantity": 200  
6 }
```

Below the request details, the response section shows a green status bar indicating "201 Created" with a timestamp of "455 ms" and a size of "259 B". The response body is also displayed in JSON format:

```
1 {  
2   "id": 1,  
3   "productName": "Gaming Mouse",  
4   "category": "Electronics",  
5   "price": 75.5,  
6   "quantity": 200  
7 }
```

At the bottom of the interface, there are various icons for runner, vault, and other tools.

LAB 04

EC/2021/006

W.K.G.K JAYAWARDANA

The screenshot shows the Postman application interface. On the left, the sidebar displays collections, environments, flows, and history. The main area shows a collection named "ApeBodima Api" with several endpoints listed under "Vehicle Api". A POST request titled "Create a new vehicle" is selected. The request URL is `http://localhost:8080/api/products`. The request method is `GET`, and the response status is `200 OK` with a response time of `359 ms` and a size of `544 B`. The response body is a JSON array containing four product objects:

```
[{"productName": "Sapiens: A Brief History of Humankind", "category": "Books", "price": 3500.00, "quantity": 25}, {"productName": "4K Ultra HD Monitor", "category": "Electronics", "price": 75000.0, "quantity": 80}, {"productName": "The Alchemist", "category": "Books", "price": 2500.0, "quantity": 300}, {"productName": "Espresso Coffee Machine", "category": "Home & Kitchen", "price": 45000.0, "quantity": 50}]
```

LAB 04

EC/2021/006

W.K.G.K JAYAWARDANA

The screenshot shows the Postman application interface. On the left, the sidebar displays collections like 'ApeBodima Api', 'Employee Api', 'Movies Api', and 'Vehicle Api'. The 'Vehicle Api' collection is currently selected, showing methods such as 'POST Create a new vehicle', 'DEL New Request', 'GET Get vehicles by year', 'GET Get all Vehicles', and 'GET Get Vehicle Type by Service ID'. The main workspace shows a request to 'http://localhost:8080/api/products/category/Electronics'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "productName": "Sapiens: A Brief History of Humankind",  
3   "category": "Books",  
4   "price": 3500.00,  
5   "quantity": 250  
6 }
```

The response status is '200 OK' with a duration of '71 ms' and a size of '258 B'. Below the response, there's a preview of the JSON data:

```
1 [  
2   {  
3     "productName": "4K Ultra HD Monitor",  
4     "category": "Electronics",  
5     "price": 75000.0,  
6     "quantity": 80  
7   }  
8 ]
```

LAB 04

EC/2021/006

W.K.G.K JAYAWARDANA

The screenshot shows the Postman application interface. On the left, the sidebar displays collections like 'ApeBodima Api' and 'Vehicle Api'. The main workspace shows a 'DELETE' request to `http://localhost:8080/api/products/Gaming Mouse`. The request body is set to `JSON` and contains the following JSON payload:

```
1 {  
2     "productName": "Sapiens: A Brief History of Humankind",  
3     "category": "Books",  
4     "price": 3500.00,  
5     "quantity": 250  
6 }  
7
```

The response section shows a successful `200 OK` status with a response message: `1 Product 'Gaming Mouse' deleted successfully.`