**SOA4 Project Document**

**By: Wye Zhen Ng**

**Student Number: A00290501**

**Project Overview**

This project implements two microservices: Category Service (producer) and Product Service (consumer). The Category Service manages categories, while the Product Service manages products and fetches category details asynchronously from the Category Service using Feign. JPA is used for database persistence in both services, and HTTP caching with ETags is implemented on the /products collection endpoint in the Product Service. A JavaScript/HTML client demonstrates the communication and caching functionality.

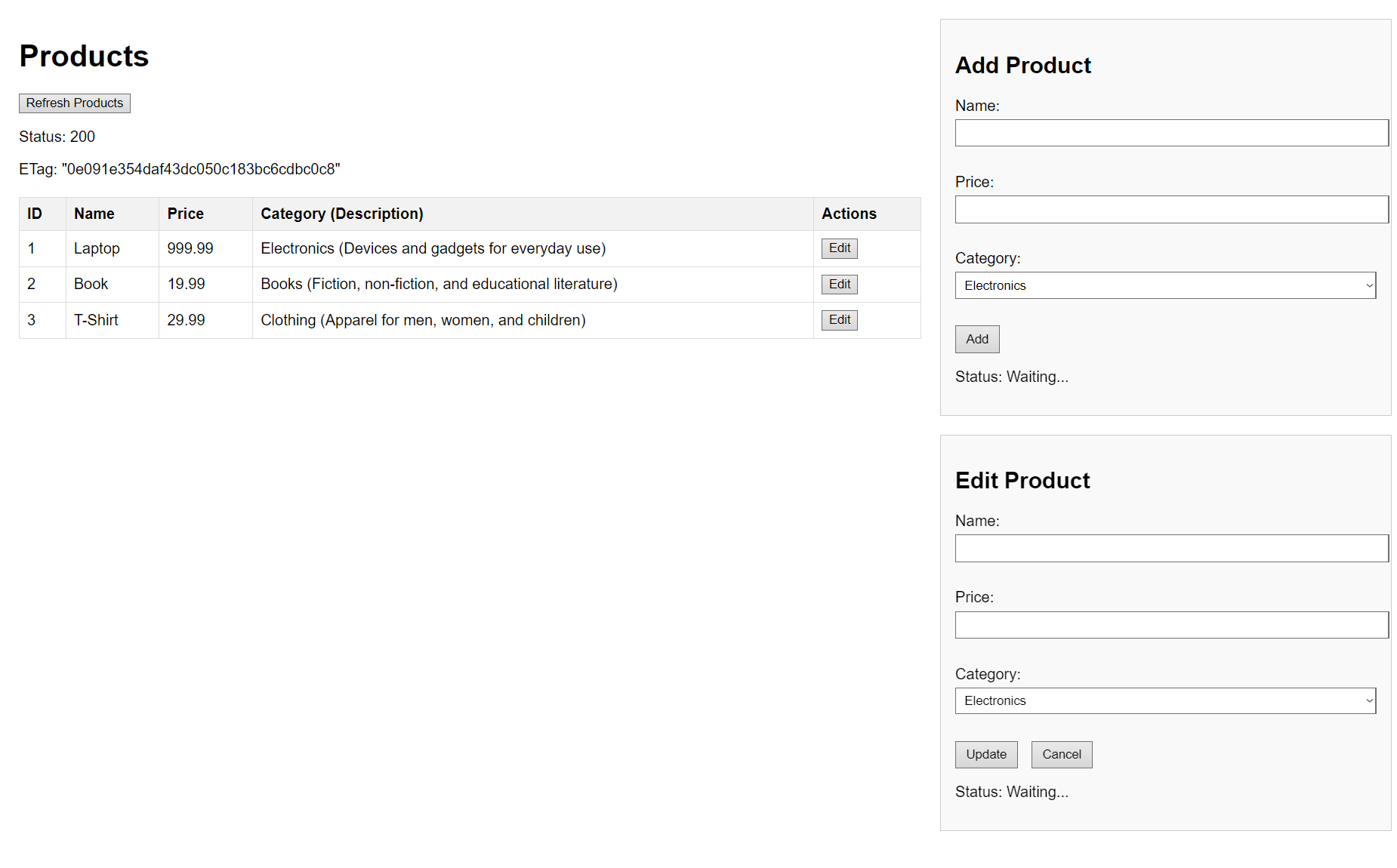
**Service Descriptions**

1. **Category Service (Producer):** 
   * Manages categories with attributes: id, name, and description.
   * Exposes REST endpoints: GET /categories and GET /categories/{id}.
   * Uses an in-memory H2 database, seeded with initial data (Electronics, Books, Clothing).
2. **Product Service (Consumer):** 
   * Manages products with attributes: id, name, price, and categoryId.
   * Connects to the Category Service via Feign for category details.
   * Implements ETag caching on the /products endpoint.
   * Provides a client interface for viewing, adding, and editing products.

**Screenshots Demonstrating Functionality**

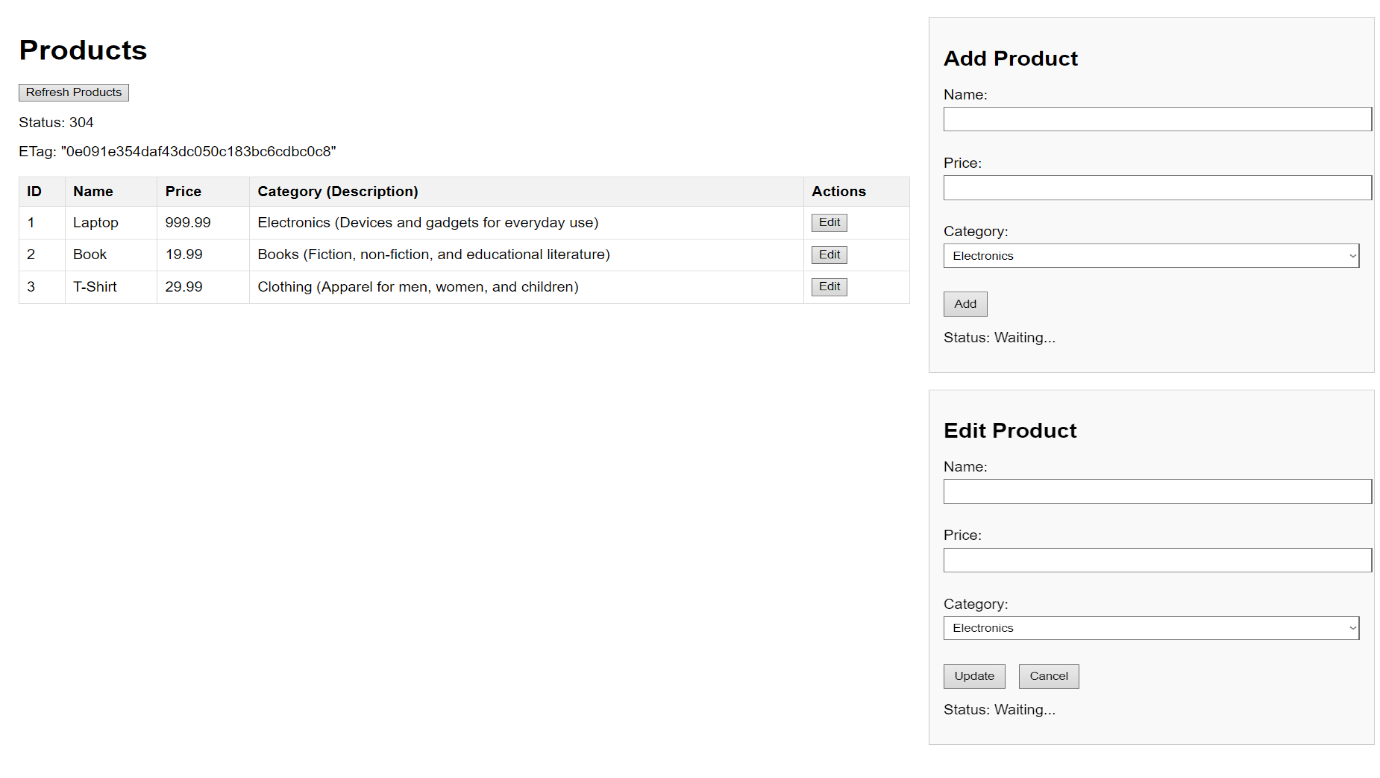
The following screenshots demonstrate the client interface, caching, and add/edit functionality:

1. **Initial Screen**:



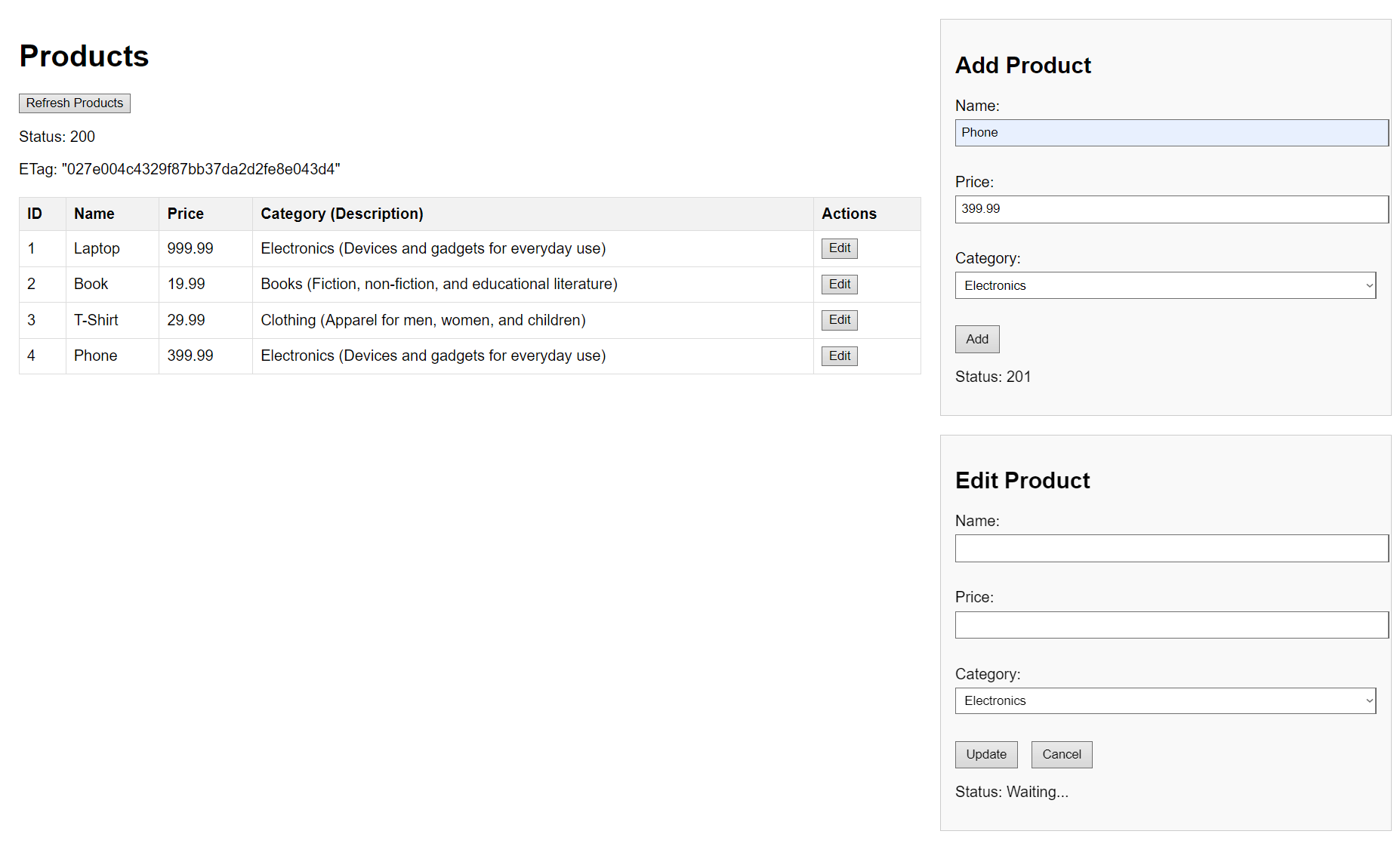
* + The page loads with the seeded products (Laptop, Book, T-Shirt). The initial GET /products request returns a 200 OK status with an ETag.

1. **After Clicking "Refresh Products"– 304 Status:**



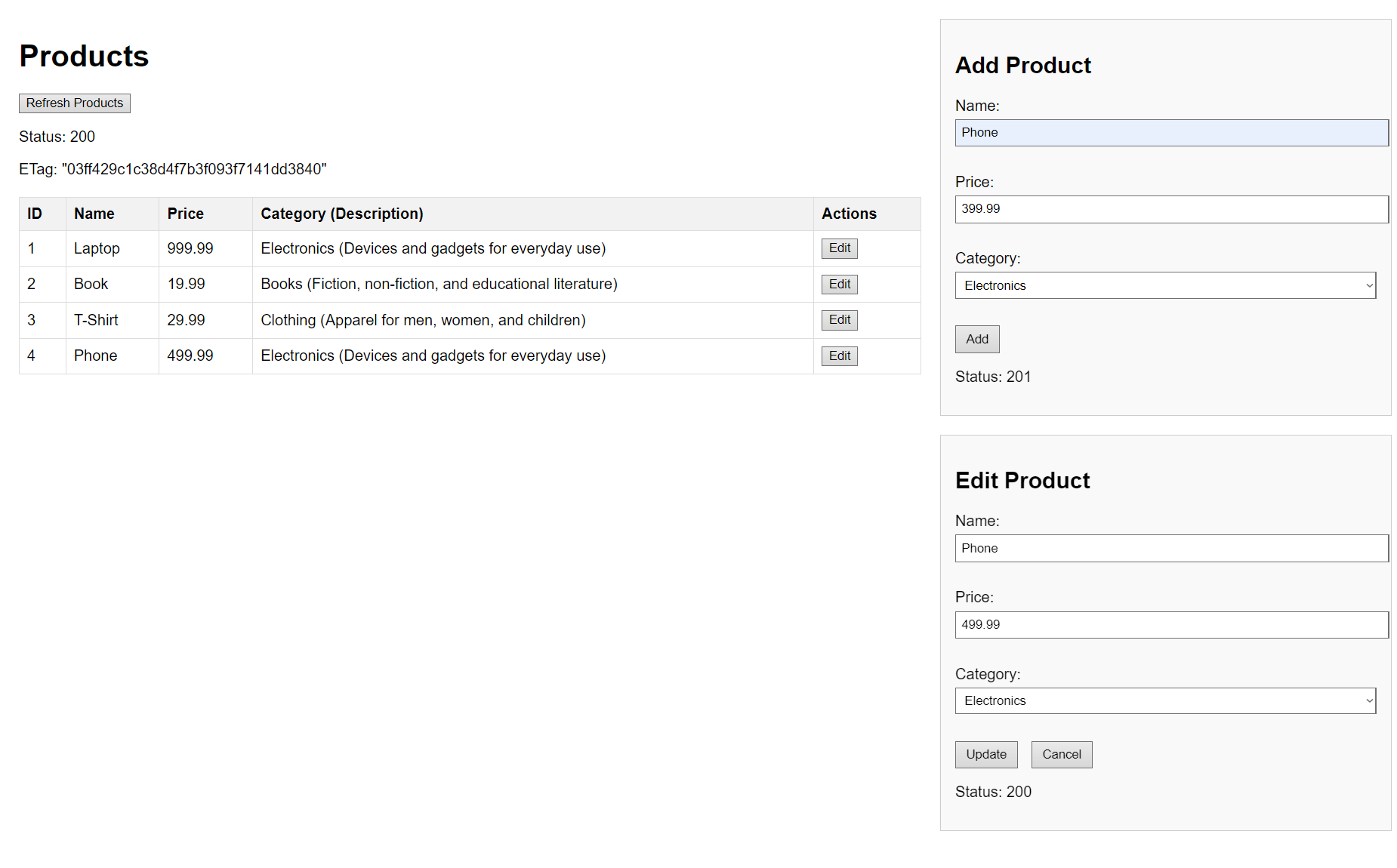
* + Clicking "Refresh Products" (without data changes) results in a 304 Not Modified status, showing the caching in action.

1. **After Adding a New Product:**



* + A new product "Phone" (price 399.99, category Electronics) is added, resulting in a 201 Created status. After refreshing, the table updates with a 200 OK status and a new ETag

1. **Editing a Product**:

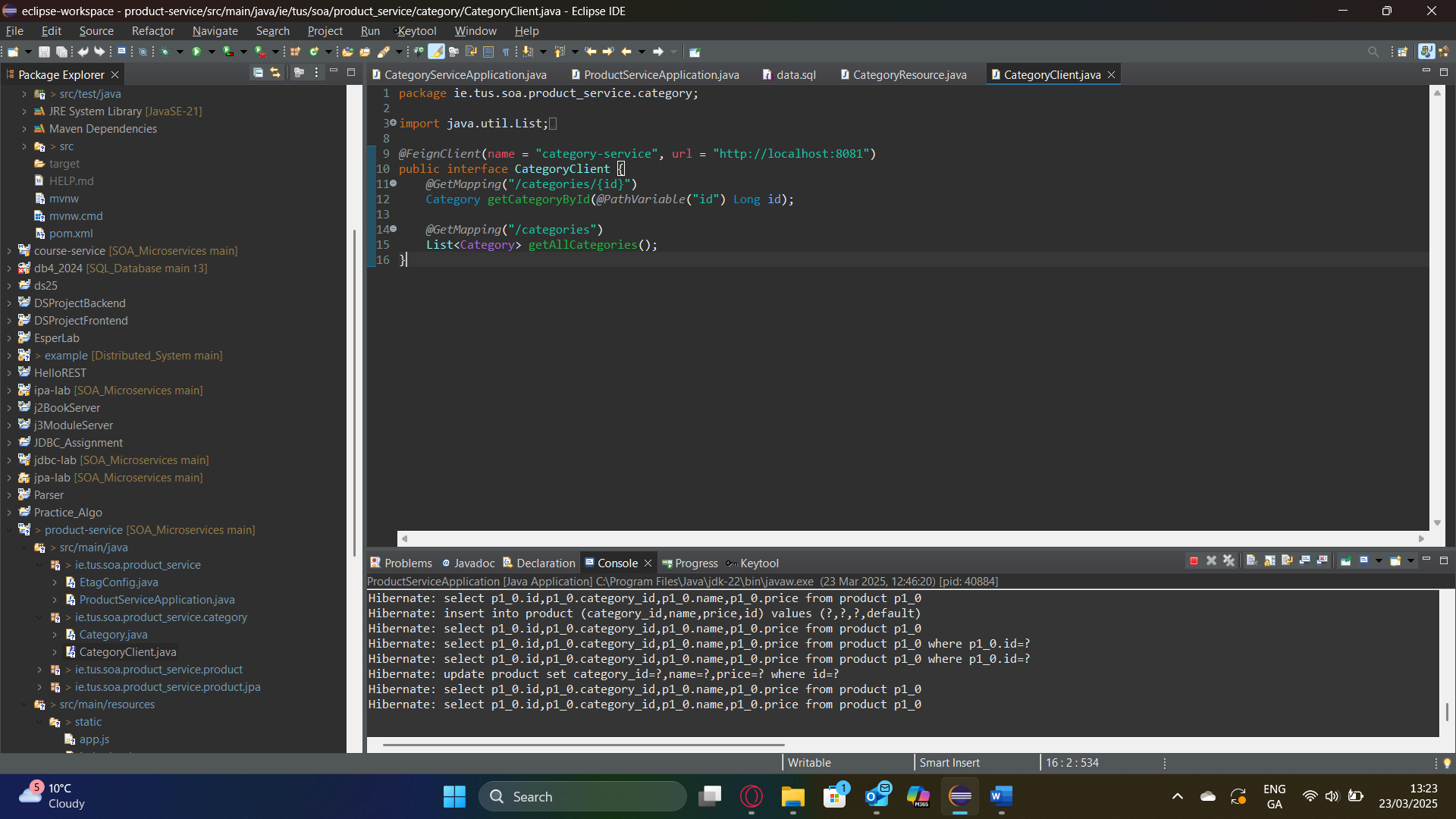


* + The "Phone" product’s price is updated to 499.99. After refreshing, the table reflects the change with a 200 OK status and a new ETag.

**Connection Between Services**

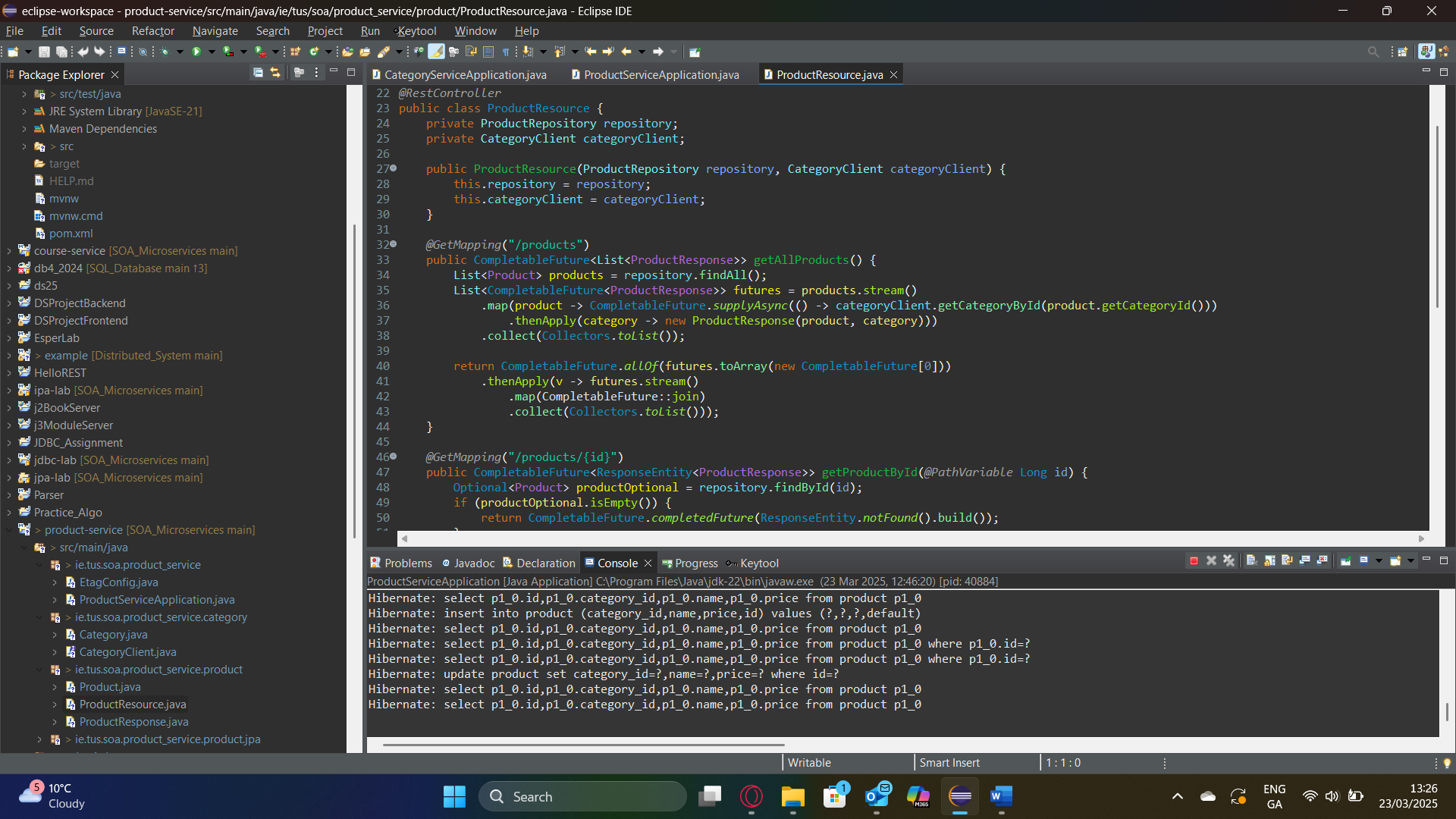
The Product Service (consumer) connects to the Category Service (producer) using Feign for asynchronous nonblocking communication. Below is the key code implementing this connection:

**Feign Client (CategoryClient.java)**



* This interface defines methods to fetch category data from the Category Service.
* The @FeignClient annotation specifies the producer’s URL (<http://localhost:8081>).

**Usage in Product Resource (ProductResource.java)**



* This endpoint fetches all products and asynchronously retrieves their corresponding categories using CompletableFuture.
* categoryClient.getCategoryById() is called for each product’s categoryId, ensuring nonblocking communication.
* The results are combined into a ProductResponse object, which includes category details (name and description).

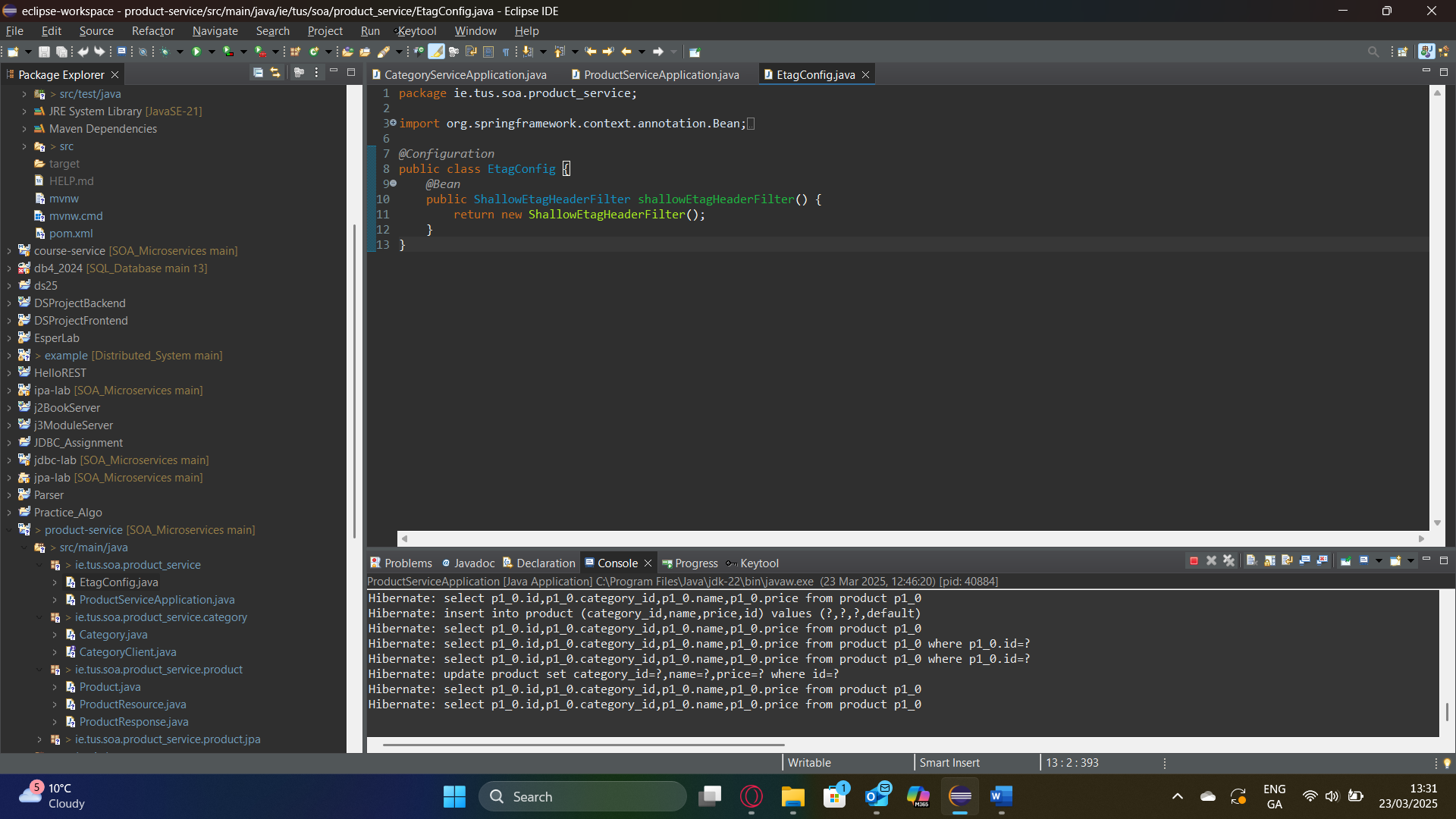
**Explanation**:

* The @EnableFeignClients and @EnableAsync annotations in ProductServiceApplication.java enable Feign and asynchronous processing.
* This setup ensures the Product Service consumes data from the Category Service efficiently without blocking the main thread.

**HTTP Caching Implementation**

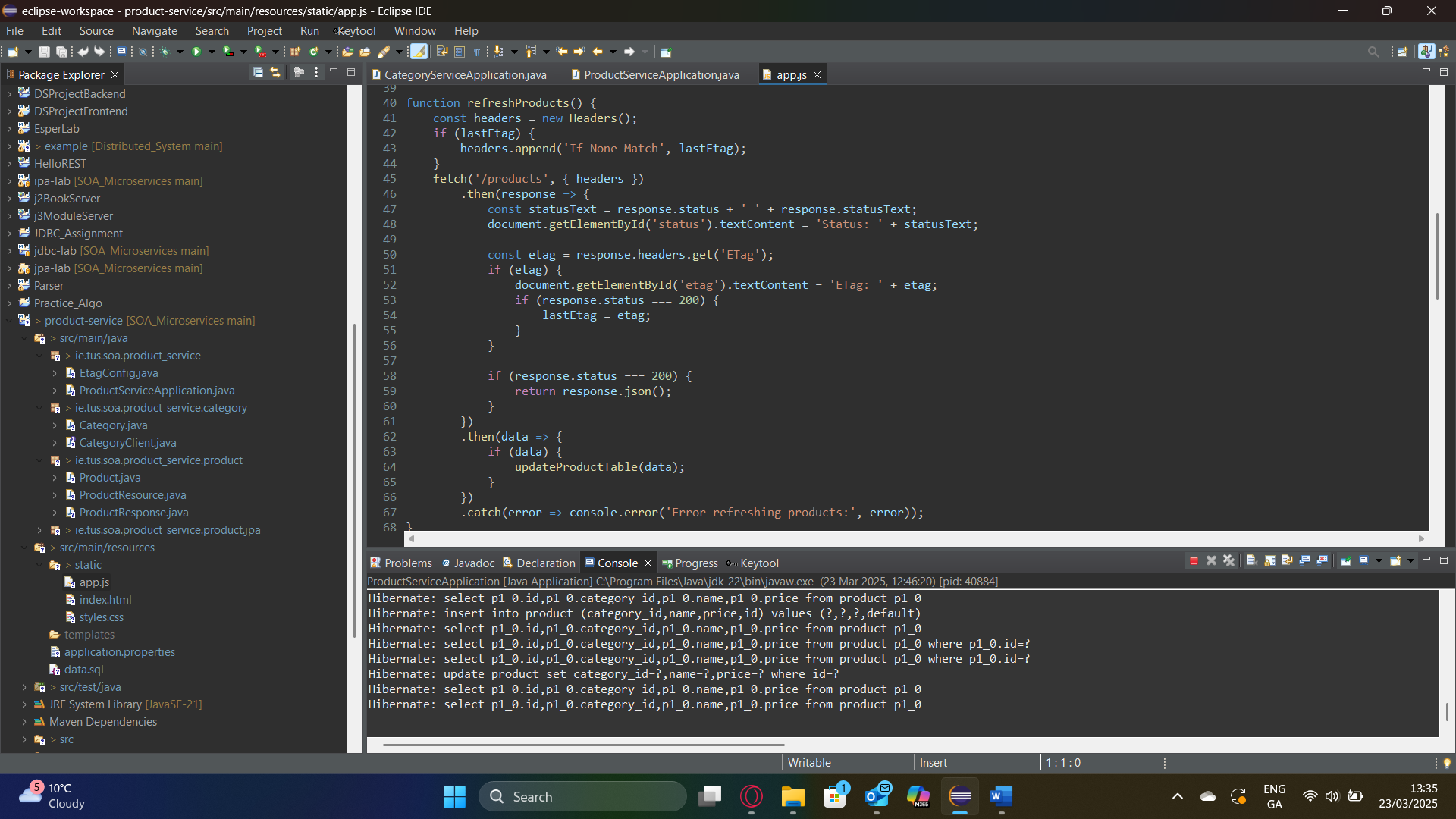
HTTP caching is implemented in the Product Service using ETags on the /products collection endpoint.

**ETag Configuration (EtagConfig.java)**



* The ShallowEtagHeaderFilter automatically generates ETags based on the response body of the /products endpoint and handles If-None-Match requests.

**Client-Side Handling (app.js)**



* The client sends an If-None-Match header with the last received ETag.
* If the server data has not changed, it returns 304 Not Modified, and the client skips updating the table.
* If the data has changed (e.g., after adding/editing), the server returns 200 OK with a new ETag, and the table is updated.

**Explanation**:

* The ShallowEtagHeaderFilter computes an ETag for the entire /products response. When the client re-requests the data, the server checks the ETag against the If-None-Match header, enabling efficient caching.