**LAPORAN HASIL PRAKTIKUM 5**

**PEMROGRAMAN BERORIENTASI OBJEK LANJUT**

**“Spring Boot-1”**

**Dosen Pengampu :**

**Sri Hartati Wijono, M.Kom.**



Oleh

Nama : Maria Gresia Plena Br Purba

NIM : 235314094

Kelas : DP

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS SANATA DHARMA**

**YOGYAKARTA**

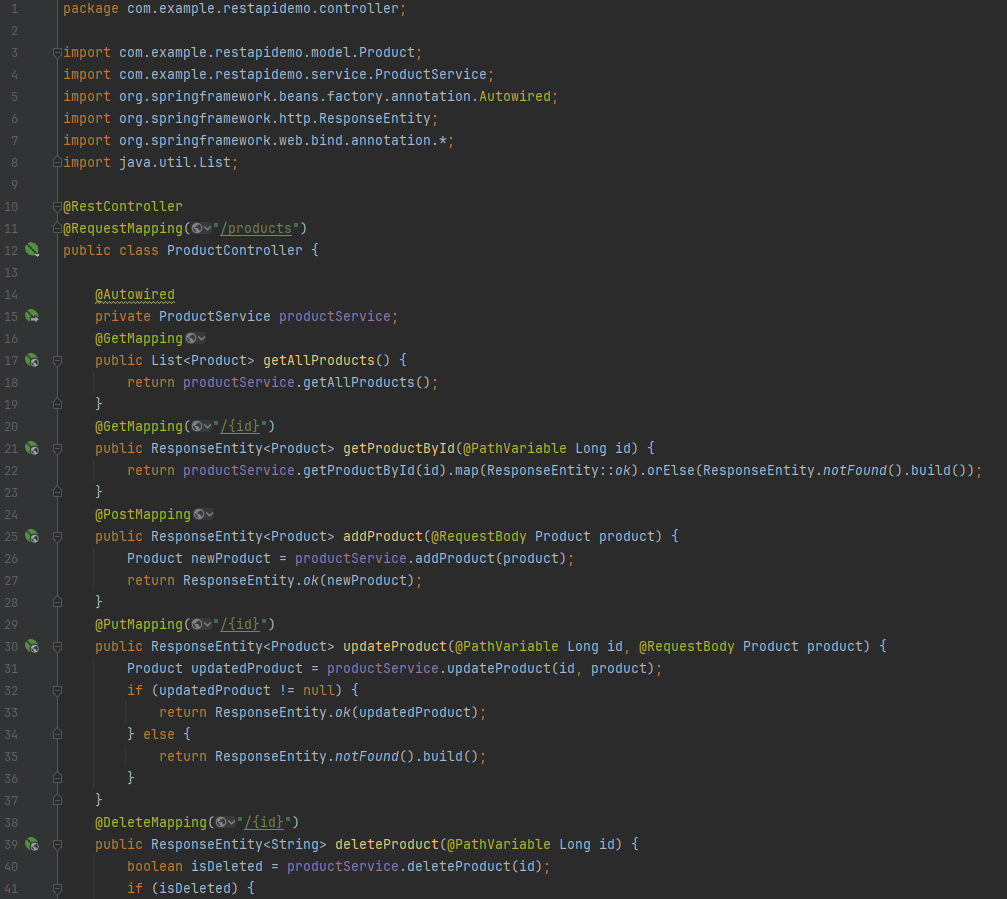
**2024**

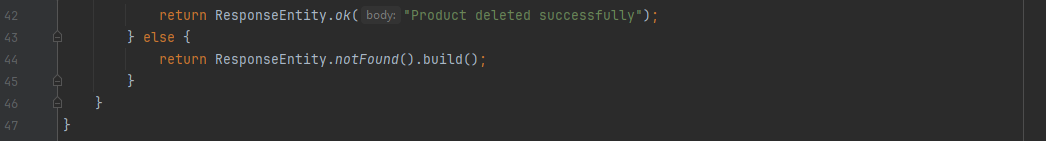
1. **TUJUAN**

1. Mahasiswa memahami cara menginisiasi proyek **Spring Boot** dan membangun REST API.

2. Mahasiswa mampu membuat REST API sederhana yang menangani operasi CRUD (Create, Read, Update, Delete).

1. **PRAKTIKUM**
   1. **Capture code class Product Controller**

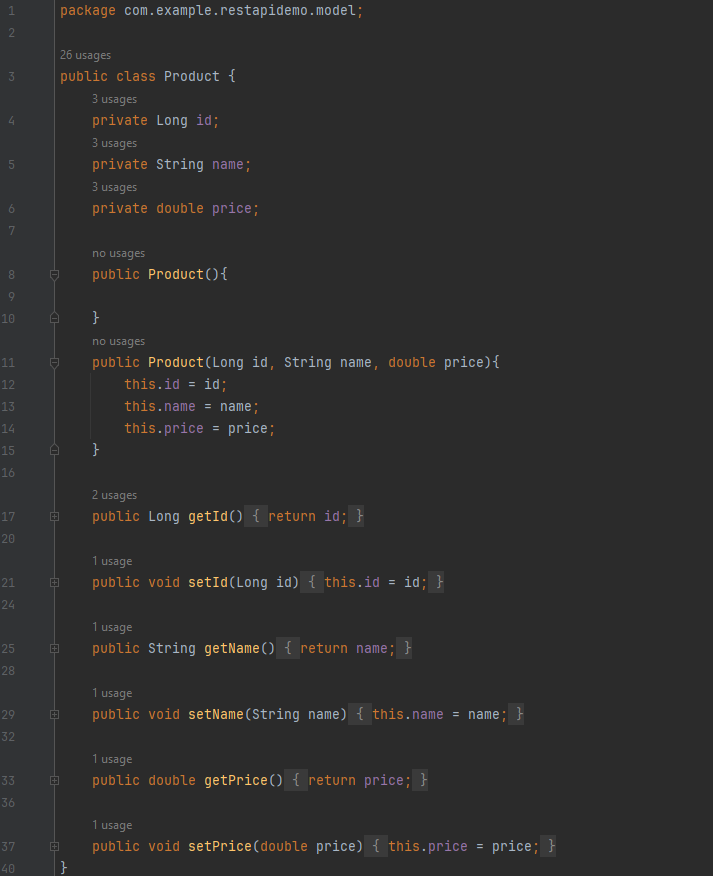
****

****

**Penjelasan :**

@RestController berfungsi untuk menangani permintaan HTTP. @RequestMapping("/products") menandakan bahwa setiap request yang berhubungan dengan product akan menggunakan ("/products") . Pada class ini terdapat deklarasi variabel bernama productRepository bertipe ProductRepository yang bersifat private. @GetMapping berfungsi untuk mendapatkan product dan method getAllProducts() mengembalikan list product dengan memanggil method getAllProducts() pada productService . @GetMapping("/{id}") berfungsi untuk mendapatkan product berdasarkan id dan method getProductById() mengembalikan product yang ditemukan dan status “ok”. Jika tidak, maka akan mengembalikan status not found. @PostMapping berfungsi untuk menambahkan product baru dan method addProduct() akan mengambil objek dari @RequstBody. Kemudian, mengembalikan produk baru yang sudah ditambahkan dan status “ok”. @PutMapping("/{id}") berfungsi untuk memperbarui product berdasarkan id dan method updateProduct() akan mengambil id dan objek dari @RequstBody. Jika product yang berhasil diupdate, maka akan mengembalikan product yang telah diperbarui dan status “ok”. Jika tidak, maka akan mengembalikan status “not found”. @DeleteMapping("/{id}") berfungsi untuk menghapus product berdasarkan id dan method deleteProduct() akan mendeklarasikan variabel bernama isDeleted yang bertipe boolean dan menginisialisasinya dengan memanggil method deletProduct(id) pada productService. Jika product terhapus, maka akan menampilkan status “ok”. Jika tidak, maka akan menampilkan status “not found”.

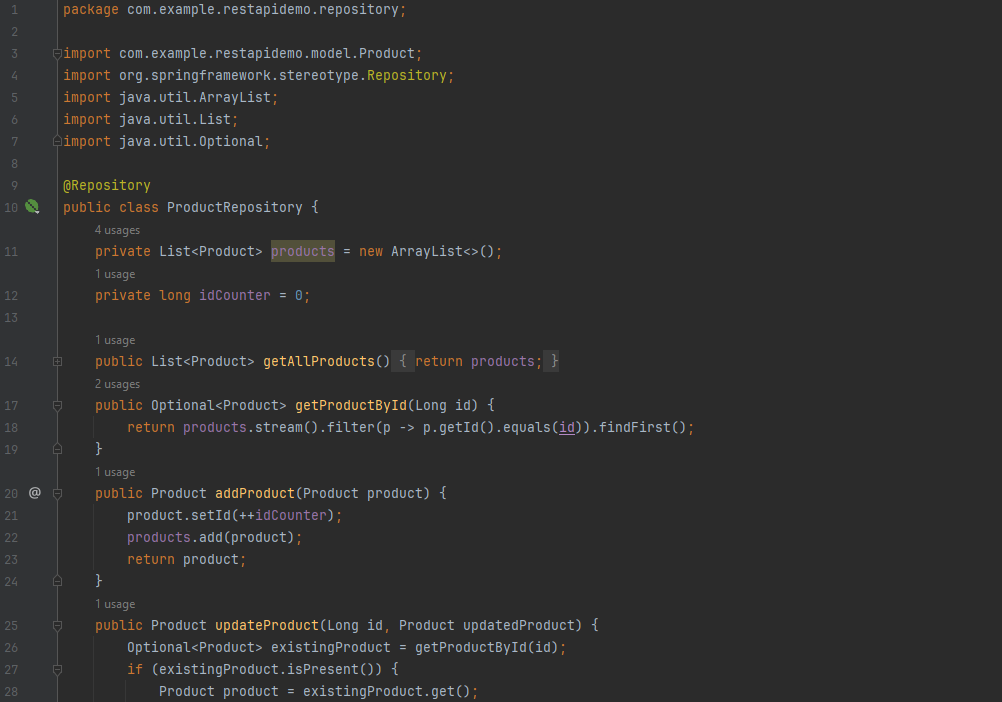
* 1. **Capture code class Product**

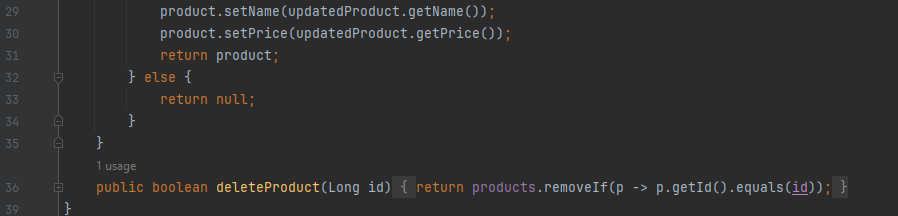


**Penjelasan :**

Pada class ini terdapat penggunaan method setter getter yang berfungsi untuk mengeset dan mengambil data dari id, name, dan price yang bersifat private. Terdapat constructor dengan isian parameter id bertipe Long, name bertipe String, dan price bertipe double.

* 1. **Capture code class Product Repository**

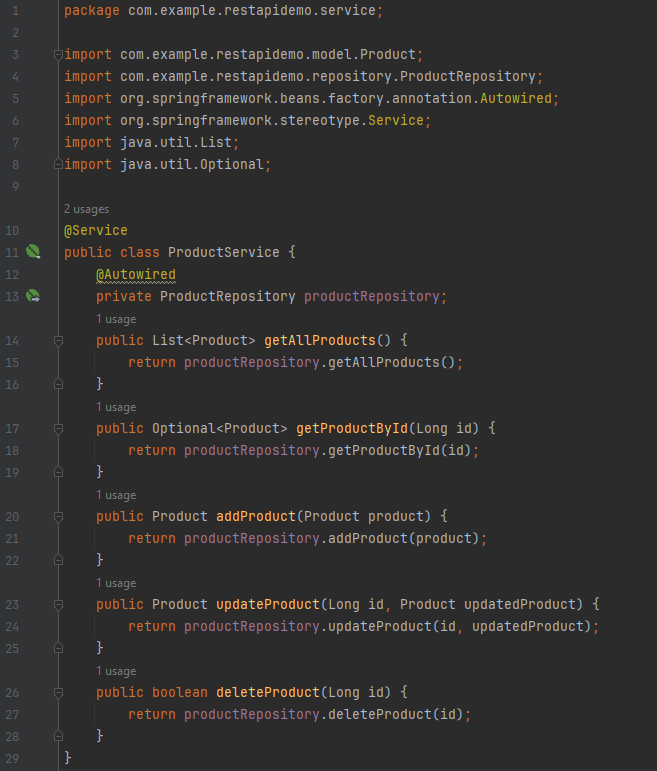
****

****

**Penjelasan :**

Mendeklarasikan variabel bernama products bertipe List<product> dan menginisialisasi idCounter dengan 0 yang bertipe long. Method getAllProducts() akan mengembalikan data “products”. Method getProductById() akan mengembalikan objek Optional<product> jika id yang dicari ditemukan. Jika tidak, akan mengembalikan Optional.empty(). Method addProduct() akan mengeset id baru dengan idCounter untuk product, memanggil method add(product) pada products, dan mengembalikannya. Method updateProduct() akan memperbarui product berdasarkan id yang dengan memanggil method getProductById(). Lalu, data nama dan harga akan di update. Jika tidak ditemukan, maka akan mengembalikan null. Method deleteProduct() akan mengembalikan data dari products jika id yang dicari sama dengan id yang ingin dihapus.

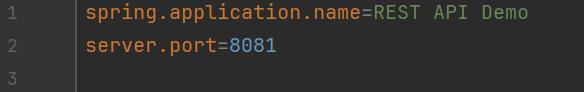
* 1. **Capture code class Product Service**

****

**Penjelasan :**

Pada class ini terdapat deklarasi variabel bernama productRepository bertipe ProductRepository yang bersifat private. Terdapat method getAllProducts() yang berfungsi untuk mendapatkan semua produk yang ada dengan mengembalikan list product. Method getProductById() berfungsi untuk mencari product dengan id tertentu dan mengembalikan optional yang berisi product jika ditemukan. Method addProduct berfungsi untuk menambahkan product baru dan mengembalikan produk baru yang sudah ditambahkan. Method updateProduct() berfungsi untuk mengupdate product yang ada berdasarkan id nya dan mengembalikan product yang telah diupdate . Method deleteProduct() berfungsi untuk menghapus product berdasarkan id nya.

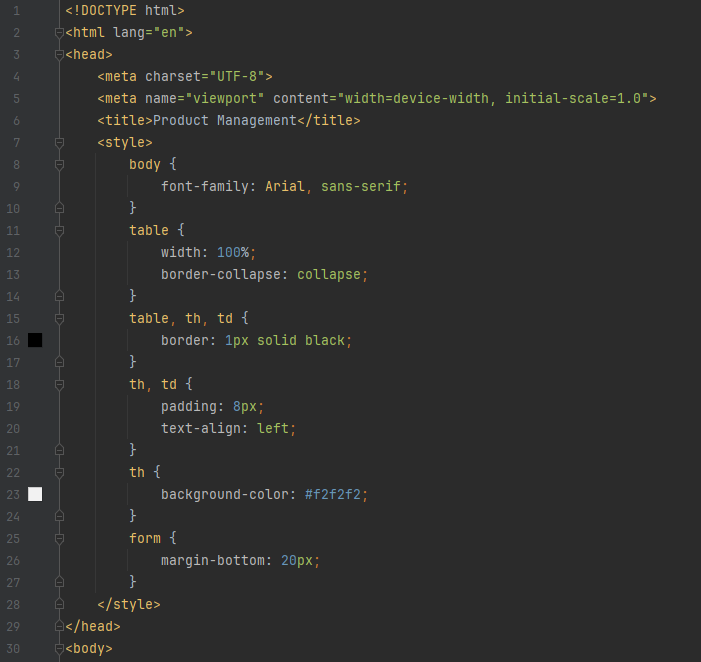
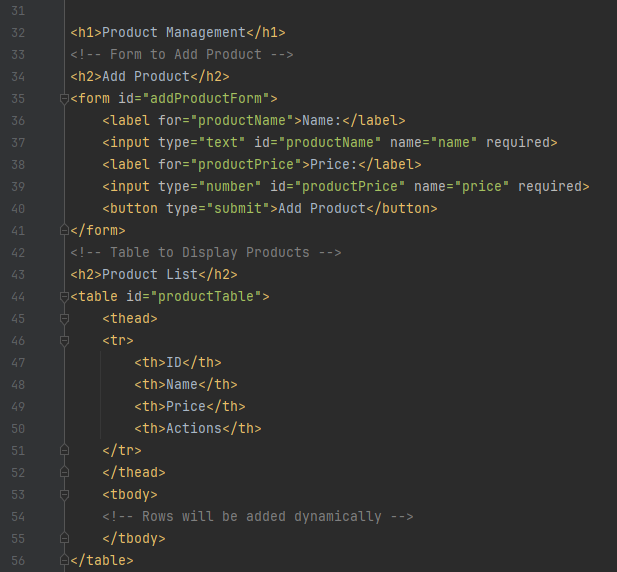
* 1. **Capture Application.properties**

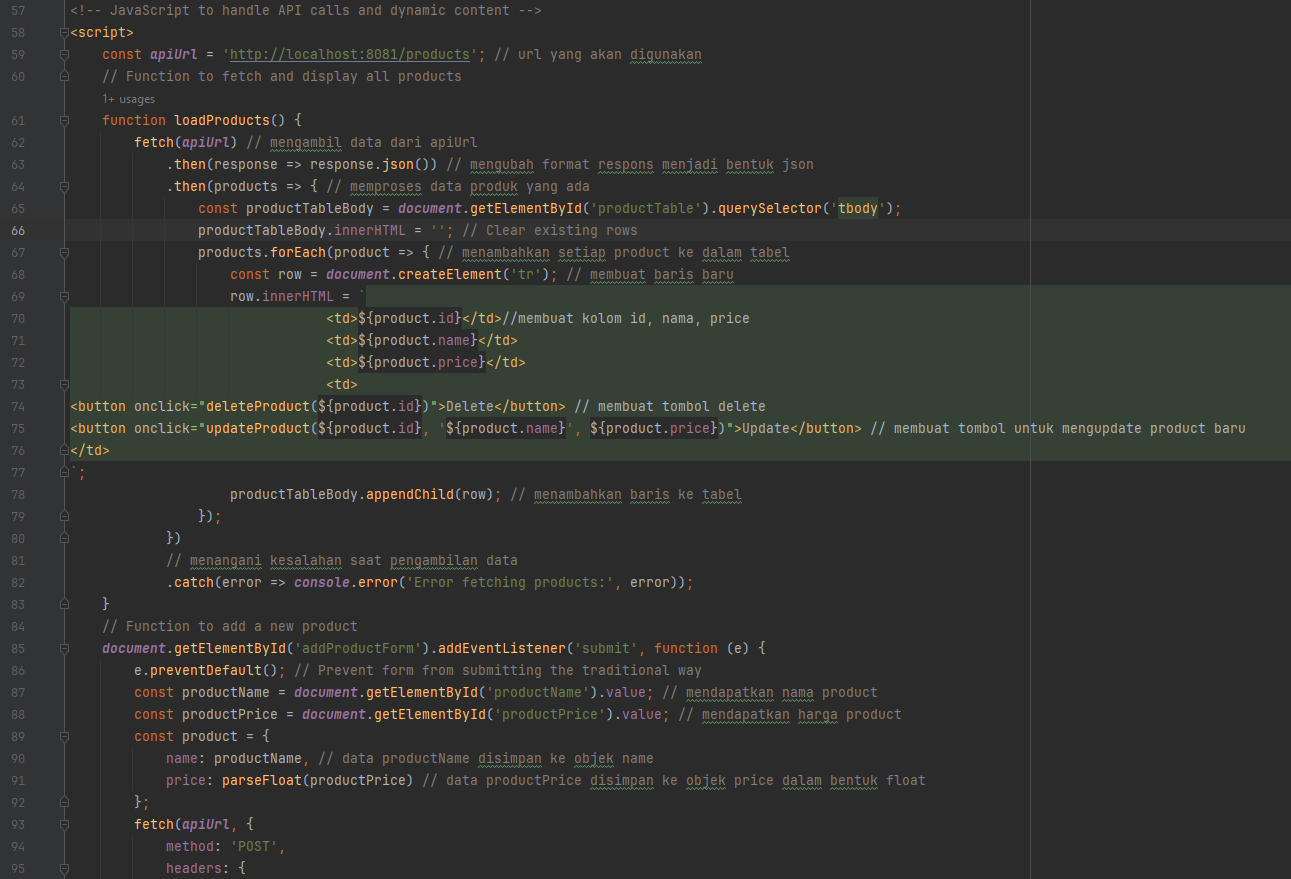
****

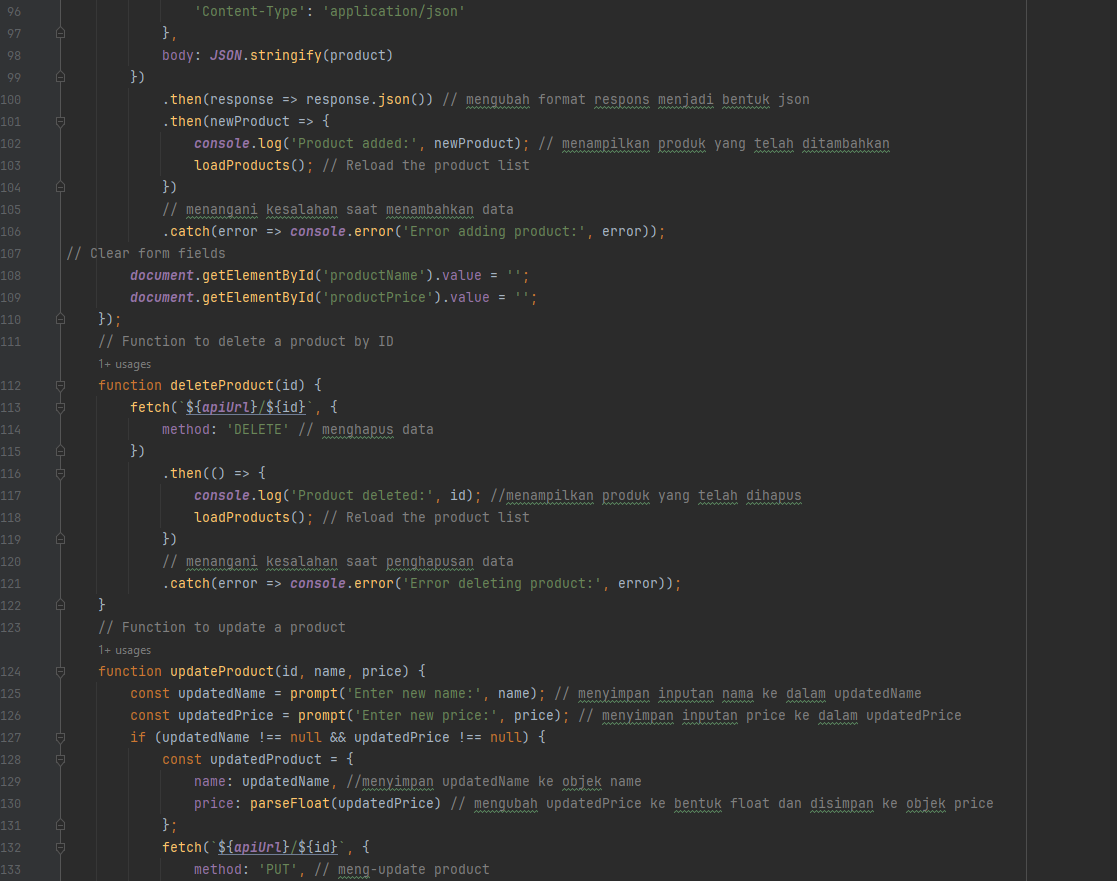
**Penjelasan :**

Menentukan server port yang akan digunakan, yaitu 8081. Jika server port telah digunakan, dapat diganti dengan server port yang belum digunakan selama tidak melewati batasan port nya.

* 1. **Capture index.html**





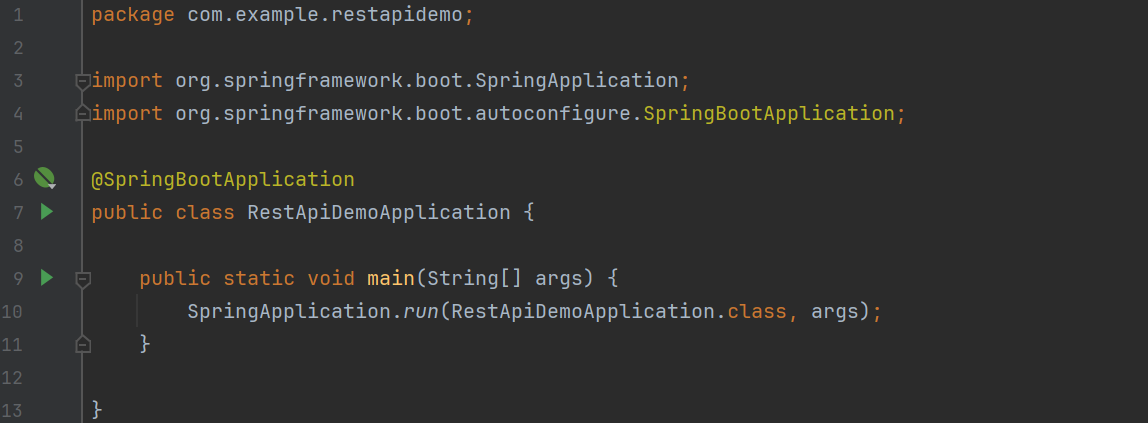




**Penjelasan :**

Menentukan style yang diinginkan dan membuat form untuk menambahkan product. Lalu, membuat tabel untuk menampilkan product yang sudah ditambahkan. Terdapat pengguanan JavaScript untuk meng-handle API dan content nya. Menyimpan url yang akan digunakan pada apiUrl. Kemudian membuat suatu function untuk menampilkan semua product yang ada. Pertama, mengambil data dari apiUrl. Kemudian, mengubah format respons menjadi bentuk json, memproses data produk yang ada, menghapus kolom yang ada dan menambahkan setiap product ke dalam tabel. Lalu, membuat baris baru, membuat tombol delete dan tombol update. menambahkan baris ke tabel dan terdapat catch untuk menangani kesalahan saat pengambilan data. Dilanjut dengan membuat function untuk menambahkan product baru. Pertama, menggunakan preventDefault() untuk mencegah submintting form secara traditional. Kemudian, menyimpan nama product dan harga product yang telah diinput ke dalam productName dan productPrice. Lalu, data dari productName akan disimpan ke name dan productPrice disimpan ke objek price dalam bentuk float. Menambahkan data dengan menggunakan method ‘POST’. Content akan dikirim dalam bentuk json, response akan diubah menjadi format json, dan akan memuat ulang daftar produk. Kemudian, terdapat catch untuk menangani kesalahan saat menambahkan data. Lalu, membuat suatu function untuk menghapus suatu product dengan menggunakan method ‘DELETE’. Kemudian, menampilkan data yang telah dihapus dan memuat ulang daftar produk. Kemudian, terdapat catch untuk menangani kesalahan saat penghapusan. Dilanjut dengan membuat suatu function untuk meng-update suatu product. Menyimpan inputan nama dan harga ke updatedName dan updatedPrice. Jika updatedName dan updatedPrice tidak null, maka data akan disimpan ke name dan price dalam bentuk float. Lalu, menggunakan method ‘PUT’ untuk meng-update product, mengirim dalam format json, dan mengubah updatedProduct ke dalam format json. Lalu, mengubah format respons menjadi bentuk json, menampilkan product yang telah di update dan memuat ulang daftar produk. Kemudian, terdapat catch untuk menangani kesalahan saat meng-update data.

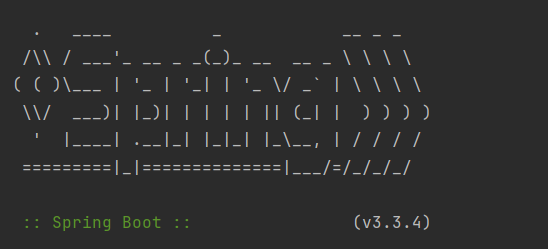
* 1. **Capture RestApiDemoApplication**

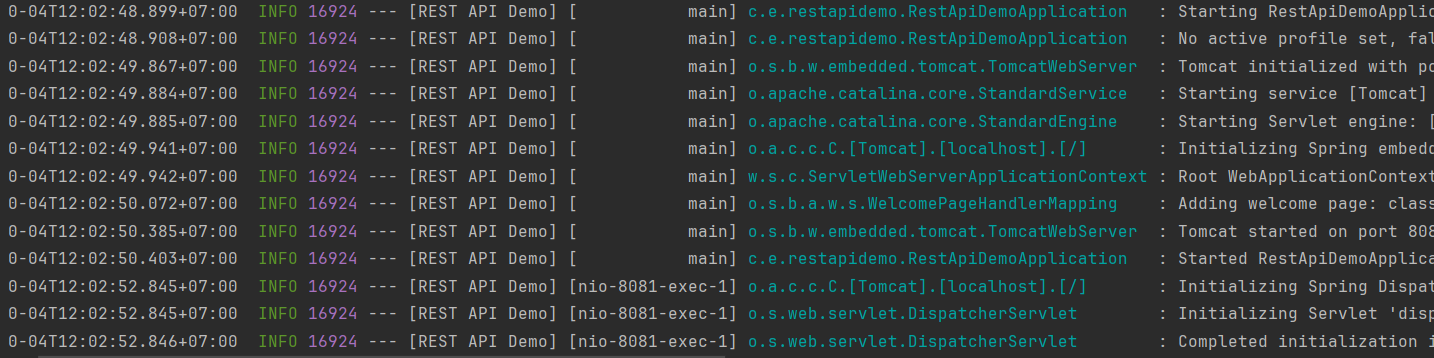
****

**Penjelasan :**

Menggunakan Spring Boot pada class RestApiDemoApplication yang ditandai dengan @SpringBootApplication (akan mengkonfigurasi secara otomatis). Lalu, SpringApplication.*run*() berfungsi untuk menginisialisai Spring dan mengelola konfigurasi yang diperlukan.

* 1. **Capture Output**

****

****

