**LAPORAN HASIL PRAKTIKUM 7**

**PEMROGRAMAN BERORIENTASI OBJEK LANJUT**

**“Spring Boot-3”**

**Dosen Pengampu :**

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



Oleh

Nama : Maria Gresia Plena Br Purba

NIM : 235314094

Kelas : DP

**PROGRAM STUDI INFORMATIKA**

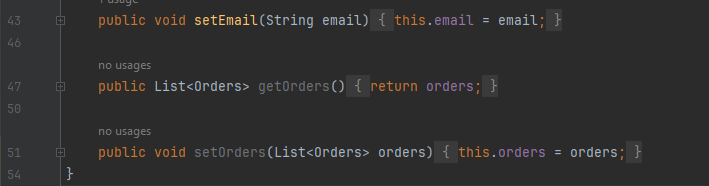
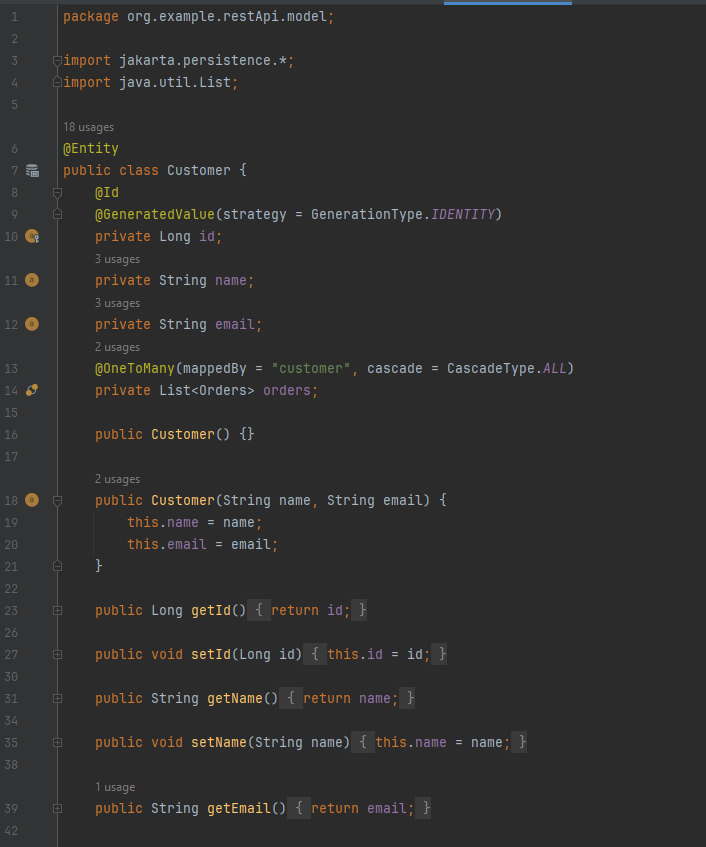
**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS SANATA DHARMA**

**YOGYAKARTA**

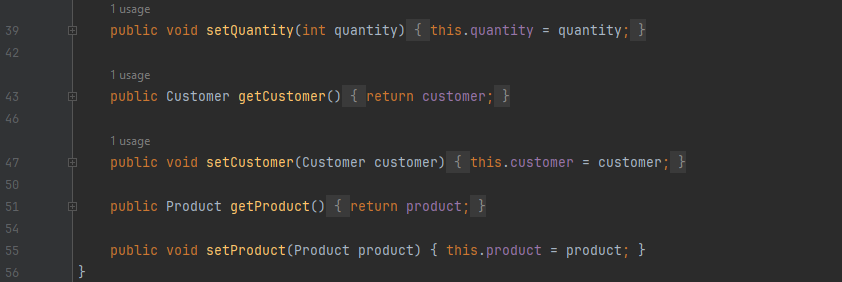
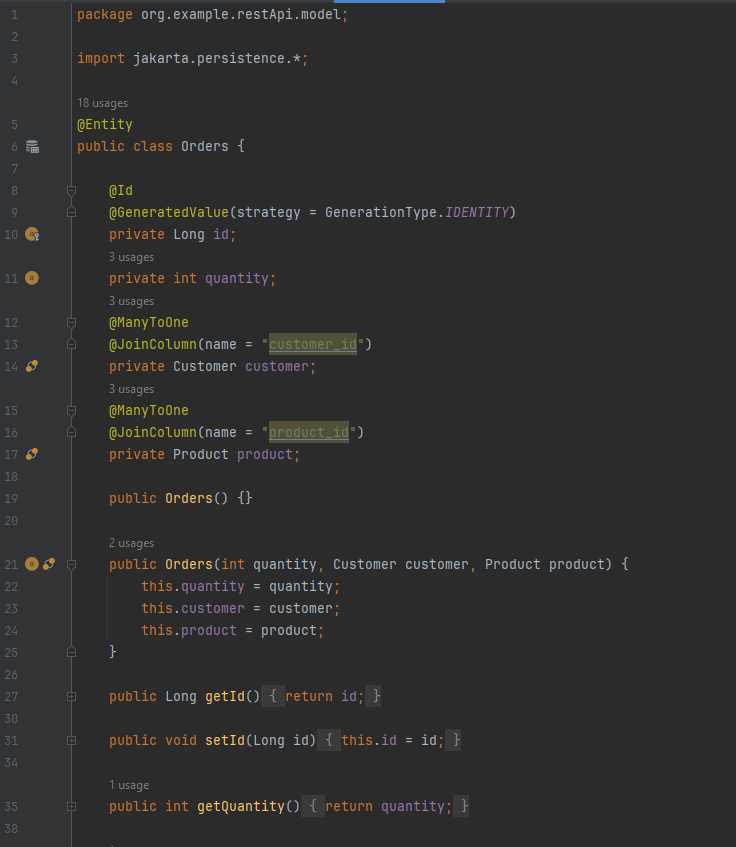
**2024**

1. **TUJUAN** 
   1. Memahami cara membuat aplikasi Spring Boot REST API.
   2. Memahami konsep CRUD (Create, Read, Update, Delete) yang lebih kompleks.
   3. Memahami konsep MVC.
2. **PRAKTIKUM**
   1. **Package model**
      1. **Customer**

****

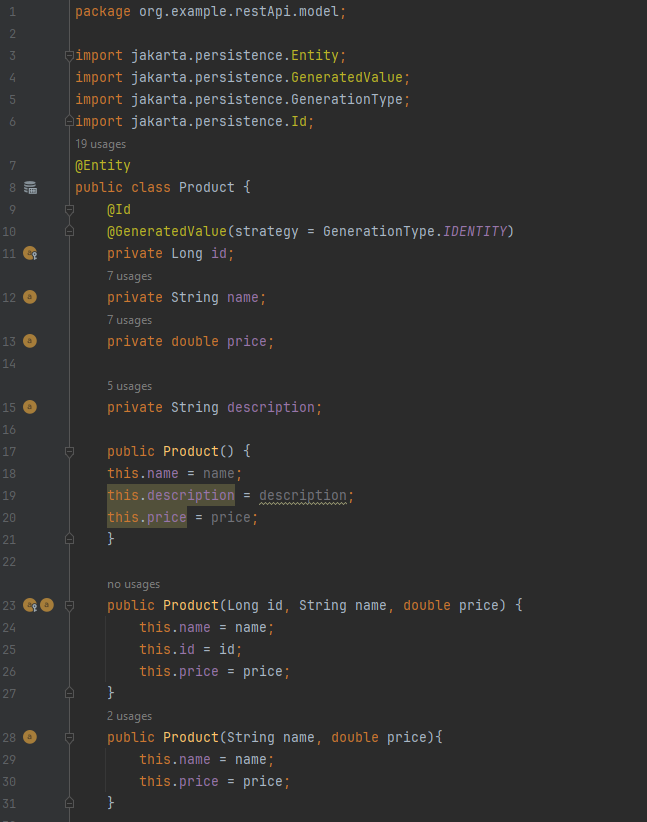
**Penjelasan :** class inimemiliki primary key berupa “id” yang menggunakan GenerationType.IDENTITY. kemudian, terdapat variabel name, email, dan orders yang bersifat OneToMany dengan class Orders. Relasi OneToMany pada class ini berarti setiap order terhubung ke satu customer. Lalu, terdapat penggunaan getter setter untuk mendapatkan dan mengeset nilai dari id, name, email, dan orders.

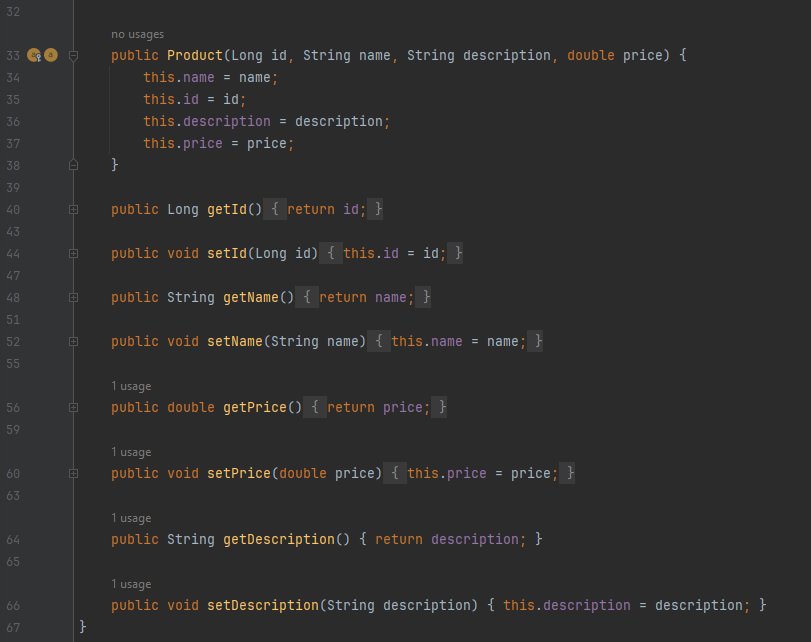
* + 1. **Orders**

****

**Penjelasan :** class inimemiliki primary key berupa “id” yang menggunakan GenerationType.IDENTITY. Kemudian, terdapat variabel quantity, customer, dan product. Variabel customer dan product bersifat ManyToOne di mana customer berelasi dengan class Customer, sedangkan product berelasi dengan class Product. @JoinColumn pada customer\_id dan product\_id berfungsi untuk menghubungkan order dengan pelanggan dan produk. Lalu, terdapat penggunaan getter setter untuk mendapatkan dan mengeset nilai dari id, quantity, customer, dan product.

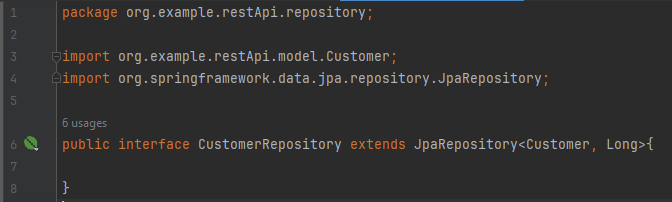
* + 1. **Product**

****

****

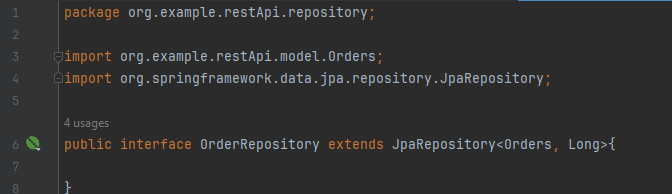
**Penjelasan :** class inimemiliki primary key berupa “id” yang menggunakan GenerationType.IDENTITY. Kemudian, terdapat variabel name, price, dan description. Lalu, terdapat penggunaan getter setter untuk mendapatkan dan mengeset nilai dari id, name, price, dan description.

* 1. **Package repository**
     1. **CustomerRepository**

****

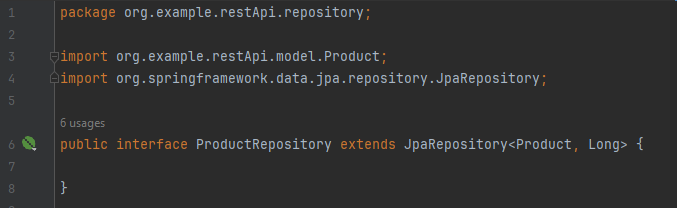
**Penjelasan :** interface mewarisi JpaRepository yang akan secara otomatis menyediakan method untuk penyimpanan, pembaruan, penghapusan, dan pengambilan data.

* + 1. **OrderRepository**

****

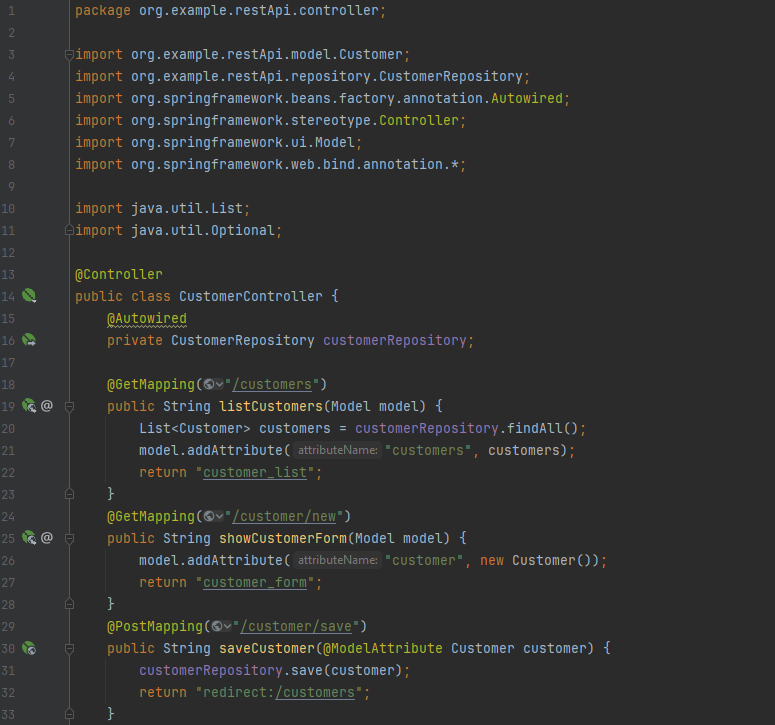
**Penjelasan :** interface mewarisi JpaRepository yang akan secara otomatis menyediakan method untuk penyimpanan, pembaruan, penghapusan, dan pengambilan data.

* + 1. **ProductRepository**

****

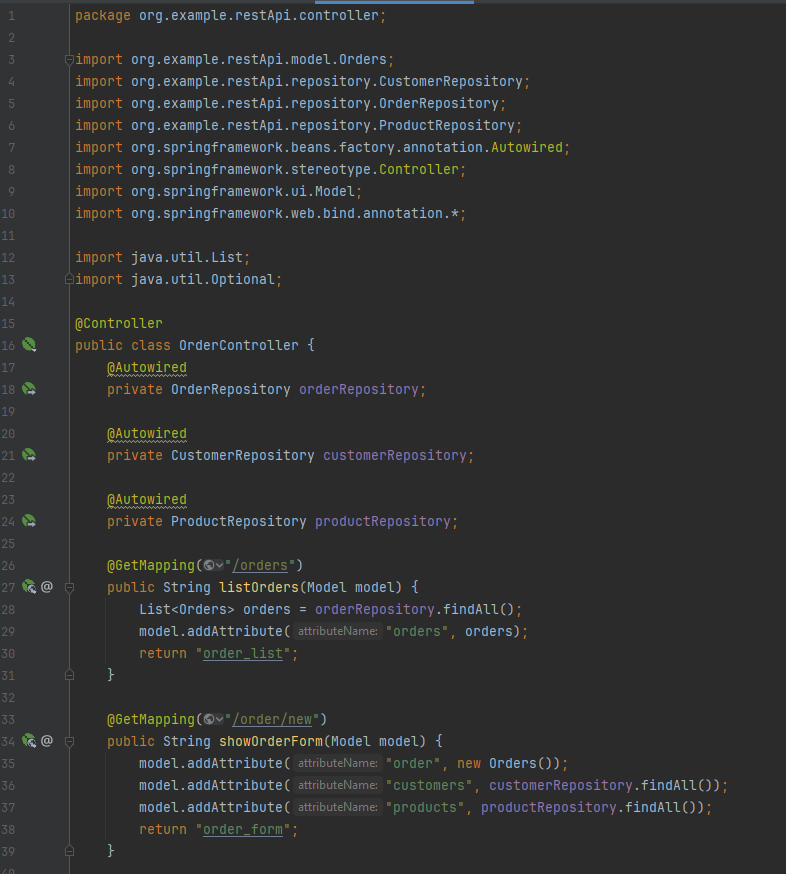
**Penjelasan :** interface mewarisi JpaRepository yang akan secara otomatis menyediakan method untuk penyimpanan, pembaruan, penghapusan, dan pengambilan data.

* 1. **Package controller** 
     1. **CustomerController**

**** ****

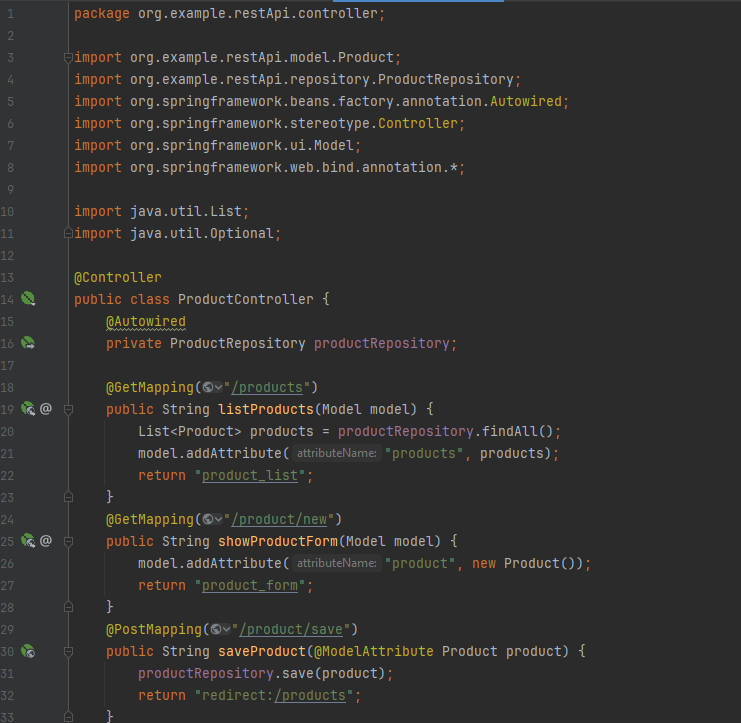
**Penjelasan :** menggunakan @Autowired pada CustomerRepository. Method listCustomers berfungsi untuk menampilkan semua data yang ada, pada halaman “customer\_list”. Method showCustomerForm berfungsi untuk menampilkan form untuk menambahkan data baru berupa objek customer. Method saveCustomer berfungsi untuk menyimpan data baru dan pengguna akan diarahkan ke halaman “/customers”. Method updateCustomer berfungsi untuk meng-update informasi customer yang sudah ada berdasarkan customer ID nya dan akan kembali ke “/customers” setelah meng-update informasi . Method deleteCustomer berfungsi untuk menghapus customer berdasarkan id-nya dan akan kembali ke “/customers” setelah penghapusan.

* + 1. **OrderController**

**** ****

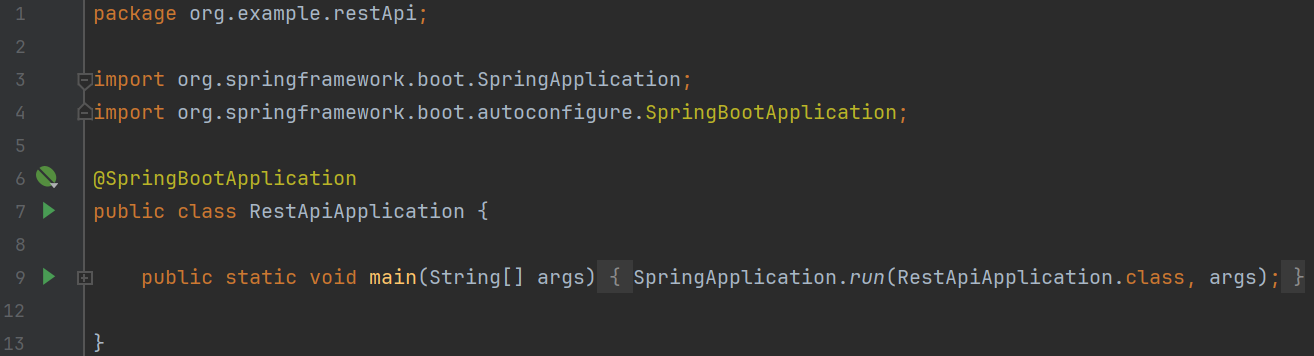
**Penjelasan :** menggunakan @Autowired pada OrderRepository, CustomerRepository, dan ProductRepository. Method listOrders berfungsi untuk menampilkan semua data yang ada, pada halaman “order\_list.html”. method showOrderForm berfungsi untuk menampilkan form untukk menambahkan data baru berupa objek order, customers, dan products. Method saveOrder berfungsi untuk menyimpan data baru dan pengguna akan diarahkan ke halaman “/orders”. Method updateOrder berfungsi untuk meng-update informasi order yang sudah ada berdasarkan order ID nya dan akan kembali ke “/orders” setelah meng-update informasi . Method deleteOrder berfungsi untuk menghapus order berdasarkan id nya dan akan kembali ke “/orders” setelah penghapusan.

* + 1. **ProductController**

**** ****

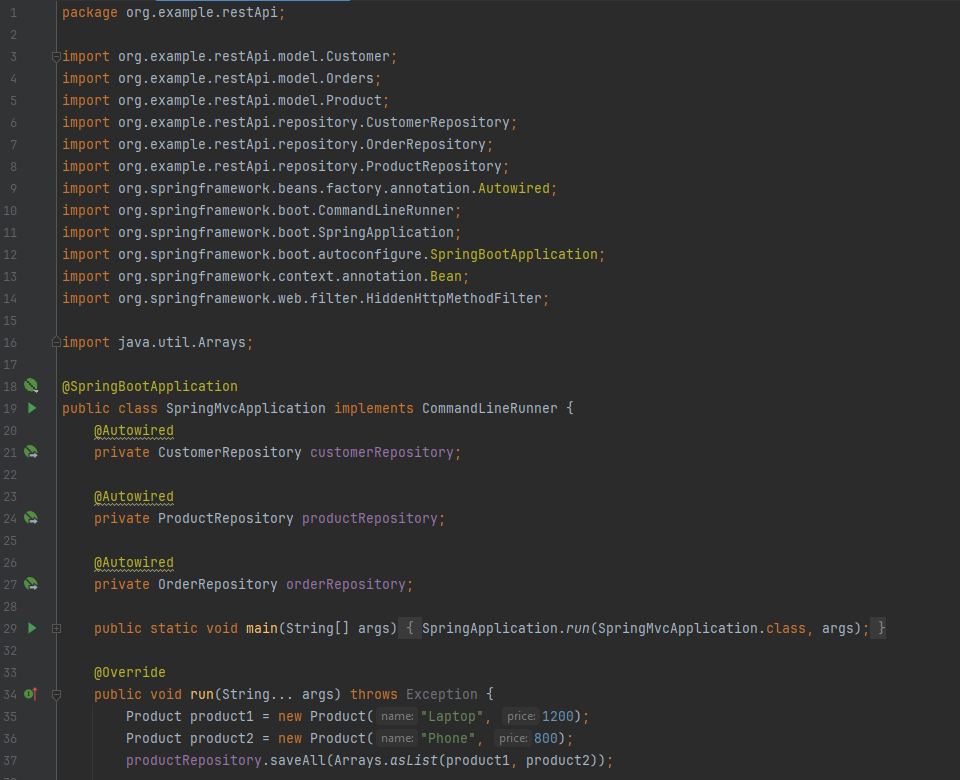
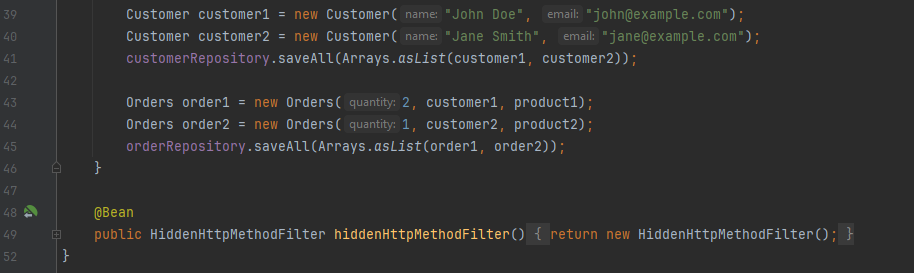
**Penjelasan :** menggunakan @Autowired pada ProductRepository. Method listProducts berfungsi untuk menampilkan semua data yang ada, pada halaman “product\_list”. Method showProductForm berfungsi untuk menampilkan form untuk menambahkan data baru berupa objek product. Method saveProduct berfungsi untuk menyimpan data baru dan pengguna akan diarahkan ke halaman “/products”. Method updateProduct berfungsi untuk meng-update informasi product yang sudah ada berdasarkan product ID nya dan akan kembali ke “/products” setelah meng-update informasi . Method deleteProducts berfungsi untuk menghapus product berdasarkan id-nya dan akan kembali ke “/products” setelah penghapusan.

* 1. **restApiApplication**

****

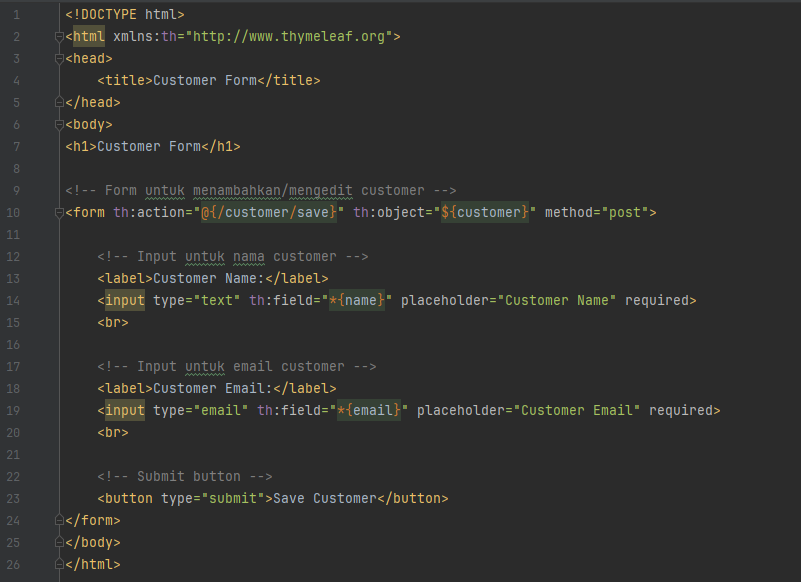
**Penjelasan :** class ini berfungsi sebagai titik awal saat ingin menjalankan aplikasi Spring Boot. Anotasi @SpringBootApplication menunjukkan bahwa ini adalah kelas utama untuk Spring Boot. Method main() memanggil SpringApplicatio.run() untuk menjalankan aplikasi-aplikasi yang ada.

* 1. **SpringMvcApplication**

****

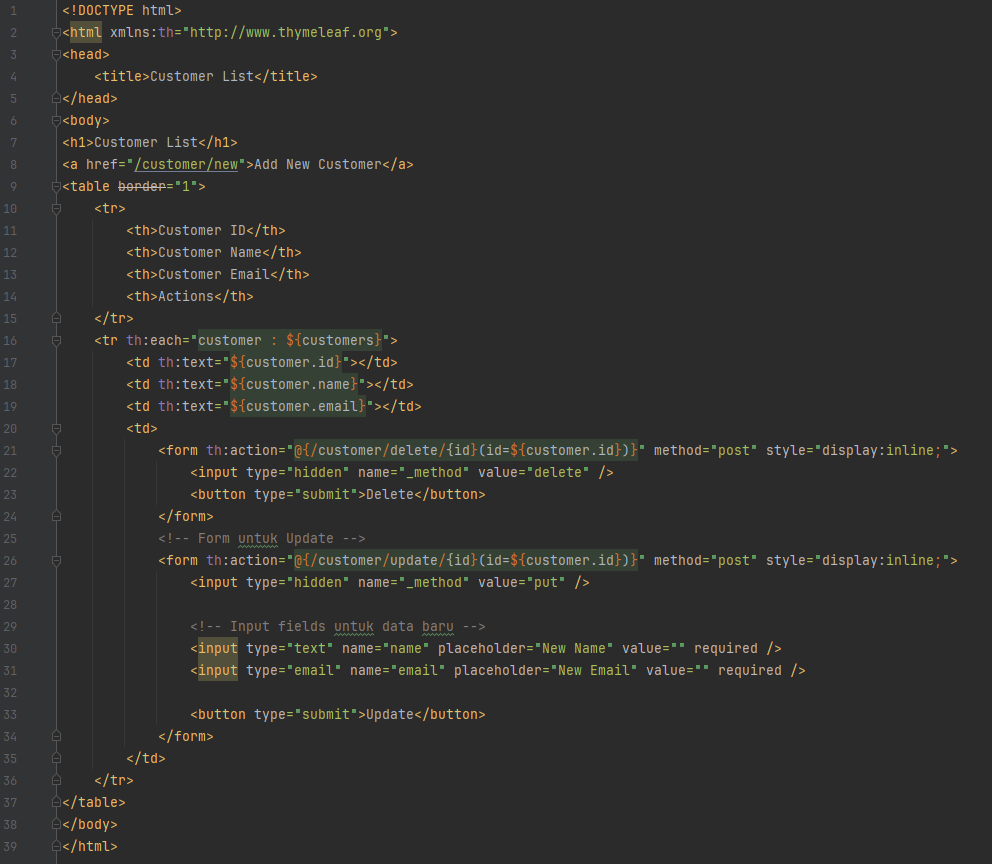
**Penjelasan :** class ini menggunakan implemnetasi CommandLineRunner untuk mengeksekusi kode. Pada method run(), terdapat pembuatan objek dari Product, Customer, dan Orders. Kemudian, mengisi data pada parameternya. Lalu, menggunakan saveAll() untuk menyimpan objek Product ke ProductRepository, Customer ke CustomerRepository, dan objek Order ke OrderRepository. Method HiddenHttpMethodFilter() merupakan bean yang memungkinkan dukungan untuk HTTP method.

* 1. **Html**
     1. **customer\_form.html**

****

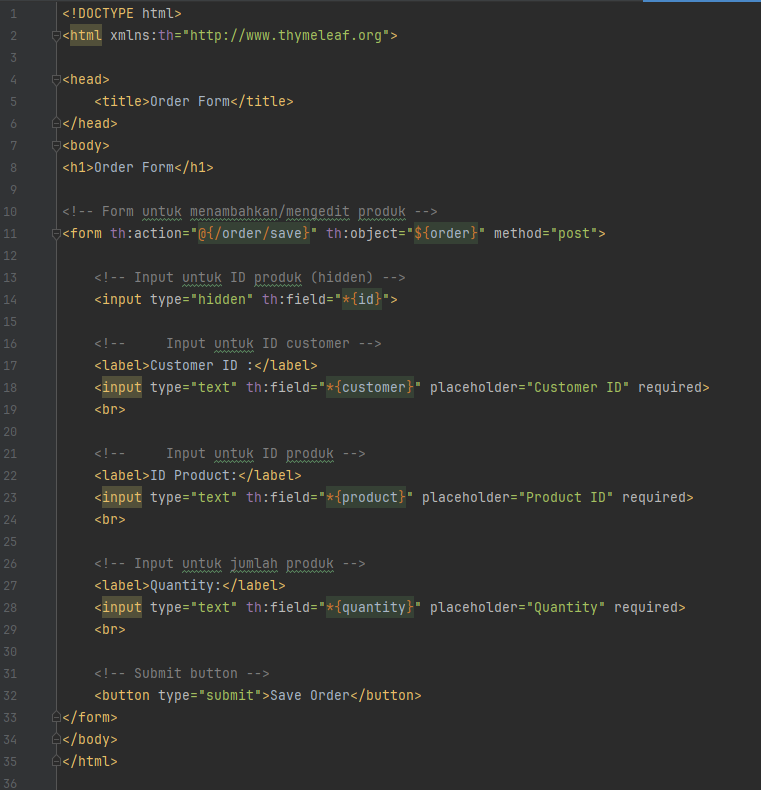
**Penjelasan :** form ini berfungsi agar customer dapat menambahkan atau mengedit data customer dengan meng-input data nama dan email. Lalu, membuat tombol dengan nama submit agar pengguna dapat menyimpan data-data tersebut.

* + 1. **customer\_list.html**

****

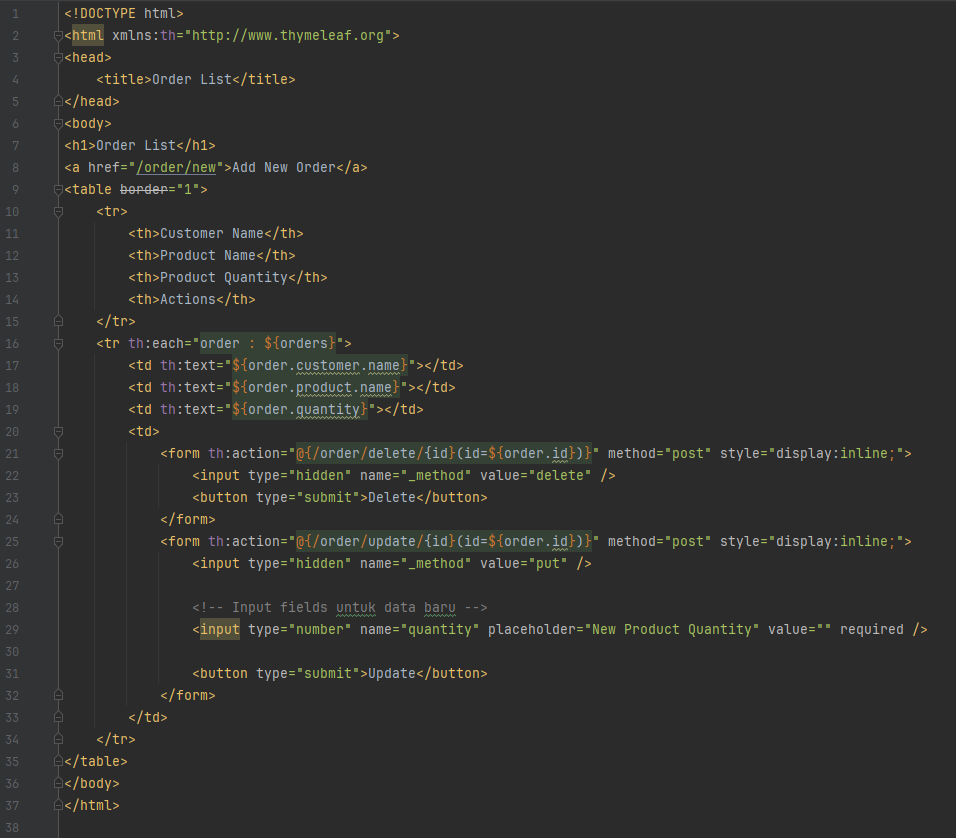
**Penjelasan :** halaman ini bernama “Customer List” dan memiliki tautan “<a href="/customer/new">Add New Customer</a>” yang mengarah ke halaman untuk menambah customer baru. Membuat kolom pada tabel dengan nama “Customer ID”, “Customer Name”, “Customer Email”, dan “Actions”. Pada kolom Actions, terdapat tombol Delete dengan method DELETE dan terdapat form untuk meng-update nama dan email customer dengan method PUT.

* + 1. **order\_from.html**

****

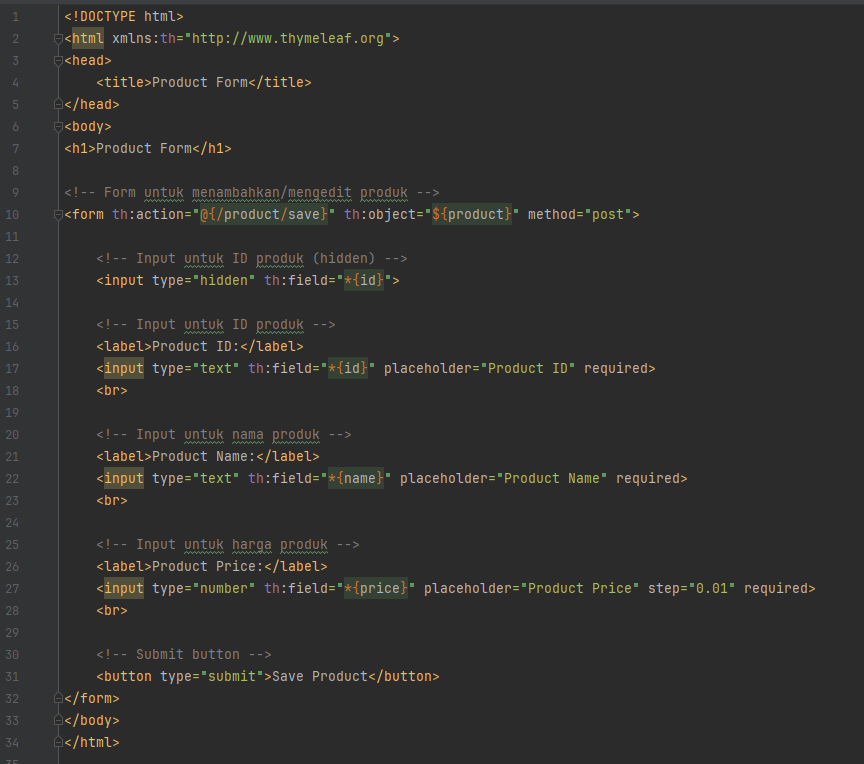
**Penjelasan :** form ini berfungsi untuk menambahkan atau mengedit data order dengan meng-input data dari id customer, product, dan quantity-nya. Lalu, membuat tombol dengan nama submit agar pengguna dapat menyimpan data-data tersebut.

* + 1. **order\_list.html**

****

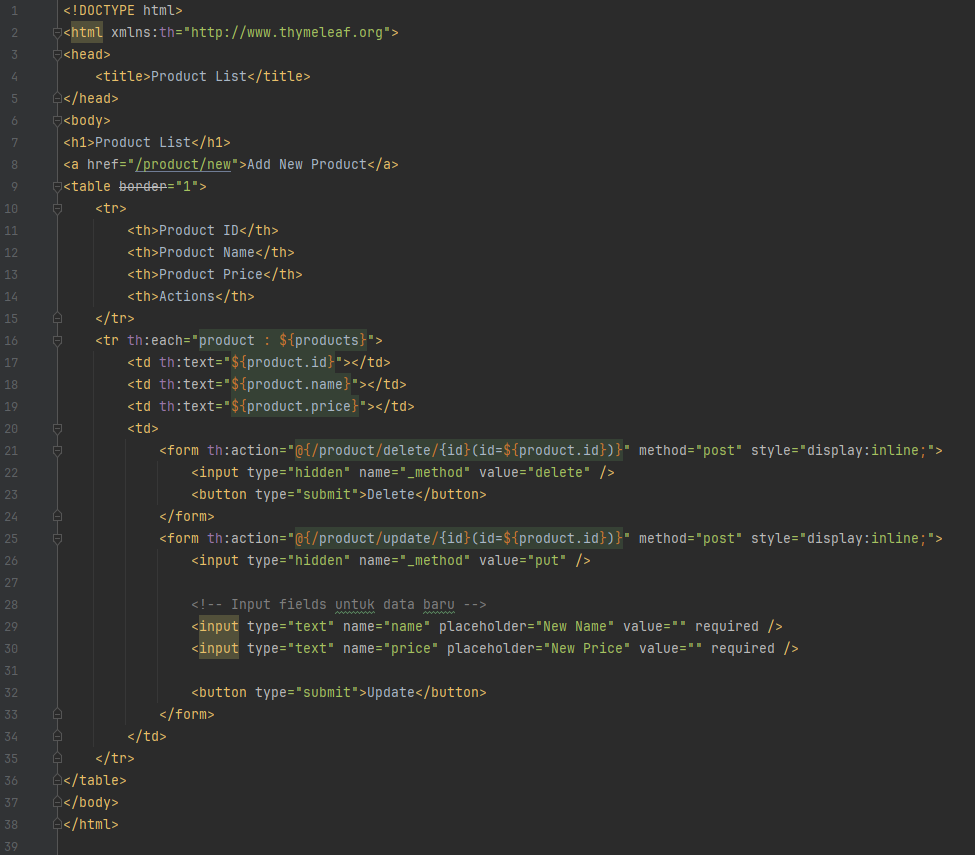
**Penjelasan :** halaman ini bernama “Order List” dan memiliki tautan <a href="/order/new">Add New Order</a> yang mengarah ke halaman untuk menambah order baru. Membuat kolom pada tabel dengan nama “Customer ID”, “Product Name”, “Product Quantity”, dan “Actions”. Pada kolom Actions, terdapat tombol Delete dengan method DELETE dan terdapat form untuk meng-update jumlah product dengan method PUT.

* + 1. **product\_form.html**

****

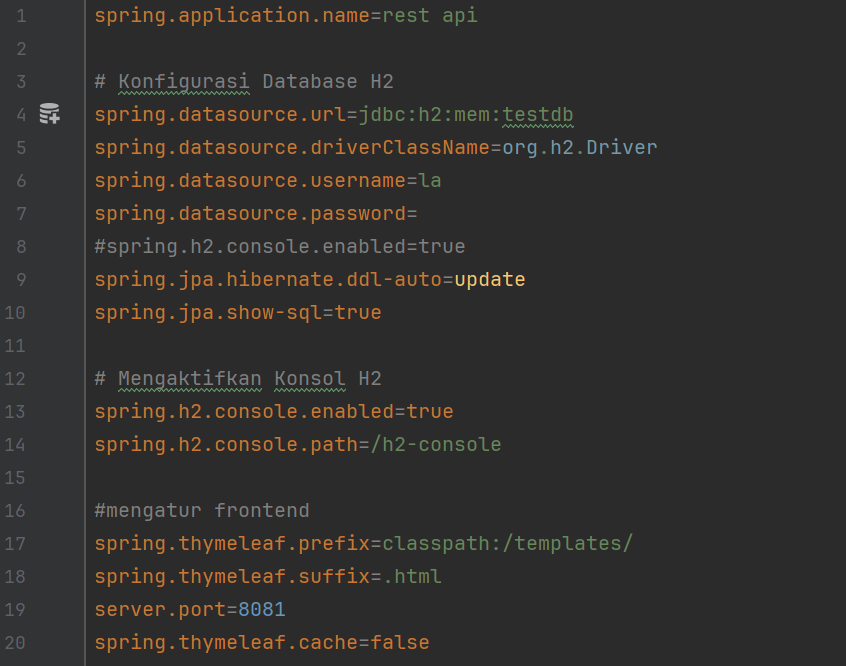
**Penjelasan :** form ini berfungsi untuk menambahkan atau mengedit data product dengan meng-input data id produk, nama produk, dan harga produk. Lalu, membuat tombol dengan nama submit untuk menyimpan data-data tersebut.

* + 1. **product \_list.html**

****

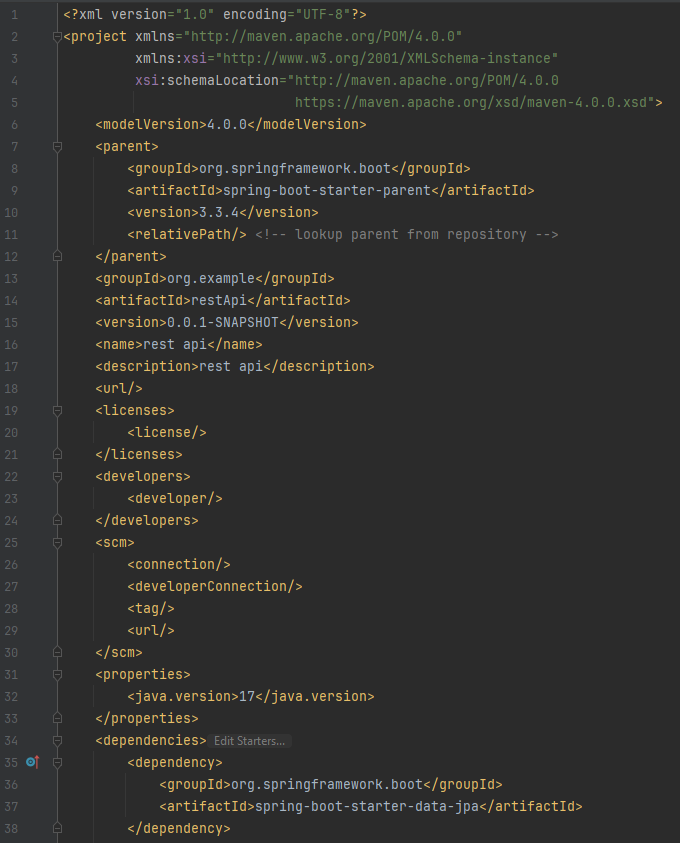
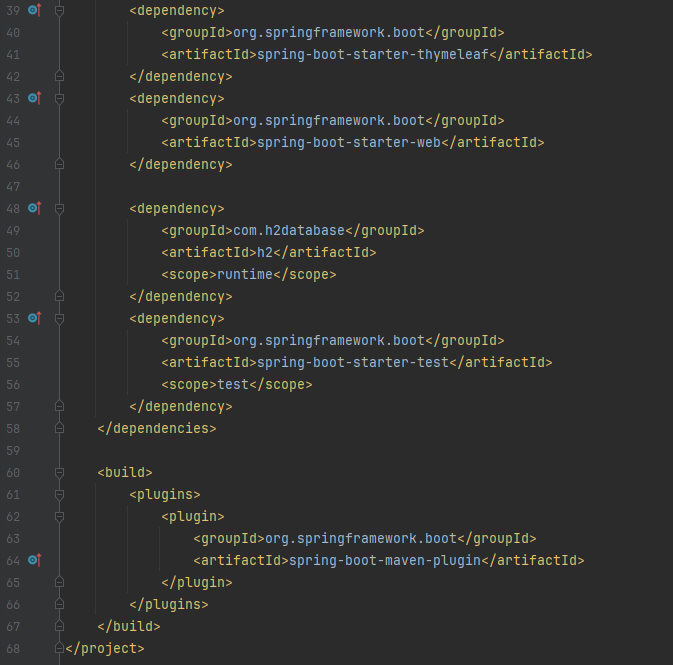
**Penjelasan :** halaman ini bernama “Product List” dan memiliki tautan “<a href="/product/new">Add New Product</a>” yang mengarah ke halaman untuk menambah product baru. Membuat kolom pada tabel dengan nama “Product ID”, “Product Name”, “Product Price”, dan “Actions”. Pada kolom Actions, terdapat tombol Delete dengan method DELETE dan terdapat form untuk meng-update nama dan price product dengan method PUT.

* 1. **application.properties**

****

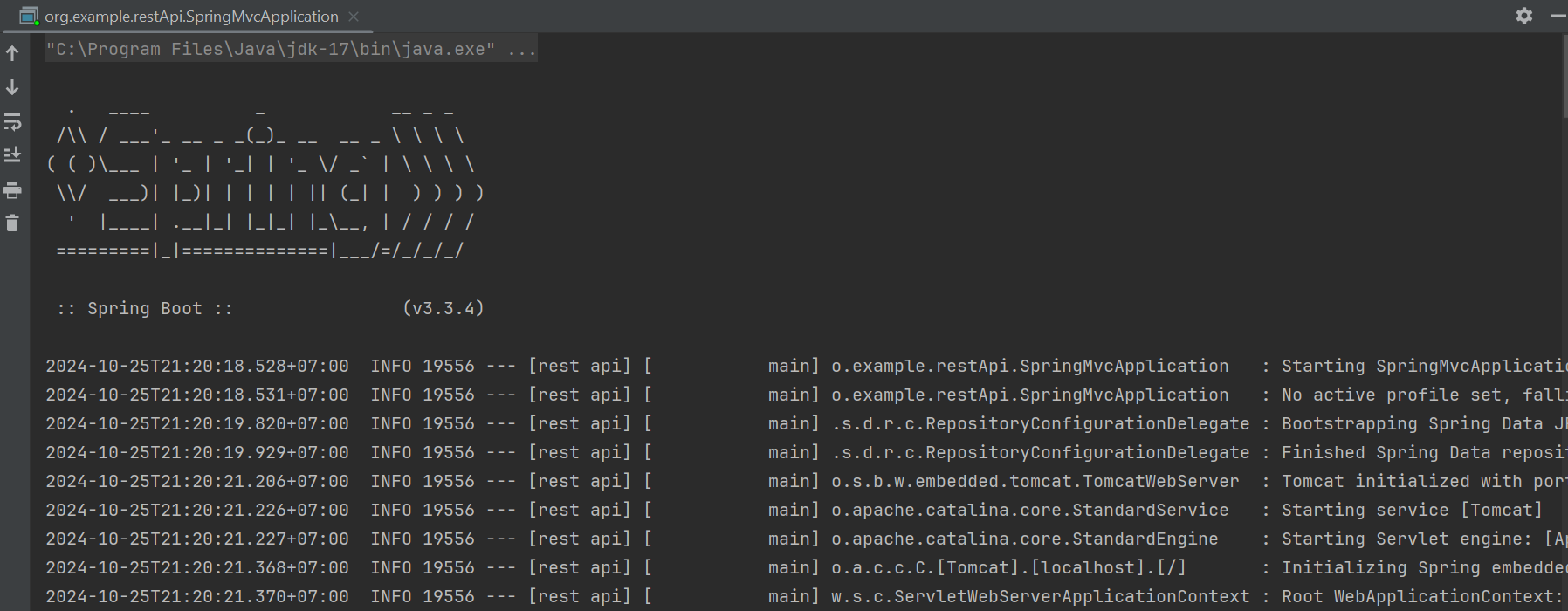
**Penjelasan :** Database h2 disetting menggunakan “jdbc:h2:mem:testdb” dengan driverClassName “org.h2.Driver”. Kemudian, memberikan data username, yaitu “sa” tanpa password. Mengaktifkan konsol h2 dengan menggunakan “spring.h2.console.enabled=true” dan memiliki path “path=/h2-console”. Lalu, mengatur frontend dengan menggunakan “spring.thymeleaf” yang di mana “.prefix=classpath:/templates/” untuk menentukan lokasi template dan “.suffix=.html” untuk format file template. Menggunakan port 8081 dan “spring.thymeleaf.cache=false” untuk memudahkan pengembangan.

* 1. **pom.xml**

**** ****

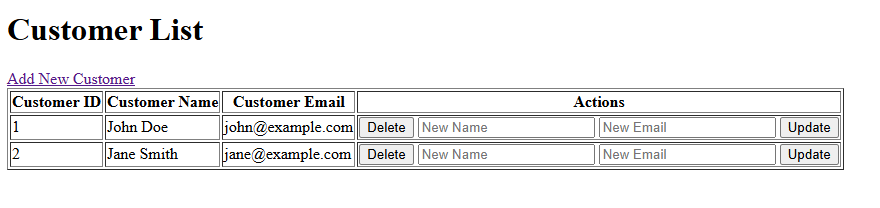
**Penjelasan :** Memiliki groupId org.example dan artifactId restApi, Menggunakan java dengan versi 17. Pada dependencies terdapat spring-boot-starter-data-jpa yang berfungsi untuk mengimplementasikan JPA repositories, spring-boot-starter-thymeleaf yang berfungsi untuk mengonfigurasi template Thymeleaf secara otomatis dalam aplikasi web, spring-boot-starter-web yang berfungsi untuk menyediakan aplikasi web Spring Boot yang siap produksi, com.h2database:h2 yang berfugsi untuk pengujian, dan spring-boot-starter-test yang berfungsi menyediakan alat untuk menguji. Pada build plugin terdapat spring-boot-maven-plugin yang berfungsi untuk menyediakan dukungan Spring Boot di Apache Maven.

* 1. **Output**

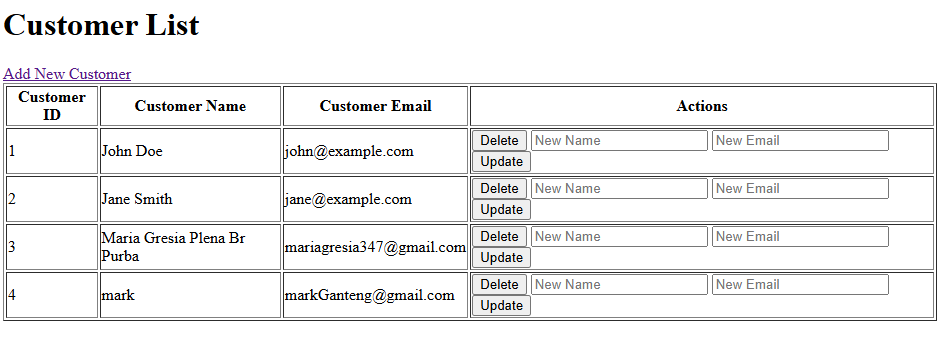
****

* + 1. **Customer**

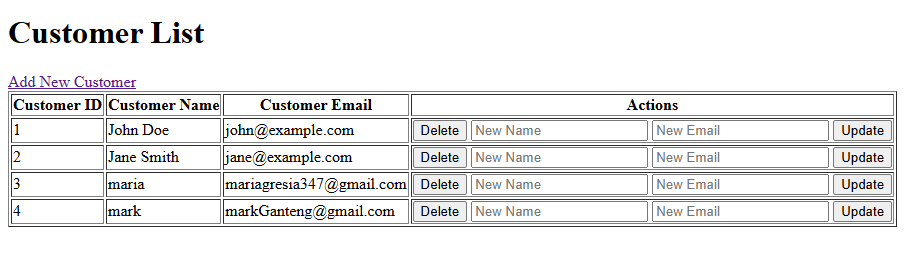
Menampilkan list dari customer

****

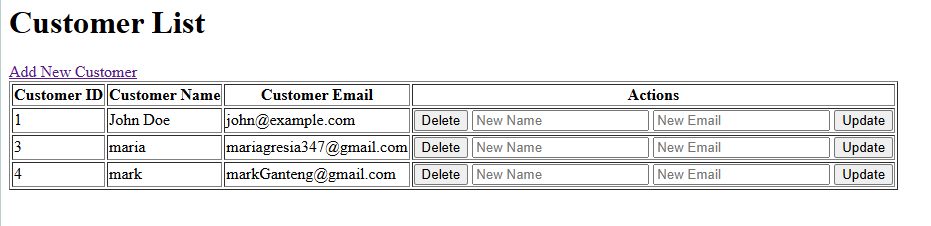
Menambahkan data baru

****

Mengganti data nama dan email pada Customer ID 3

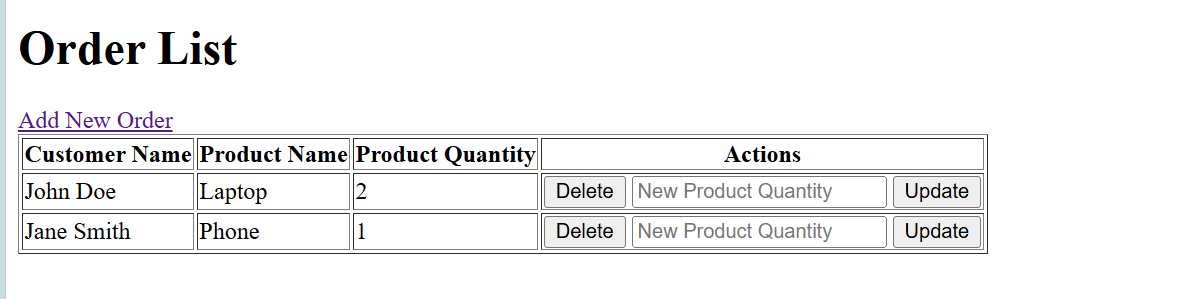
****

Menghapus data dengan customer ID 2

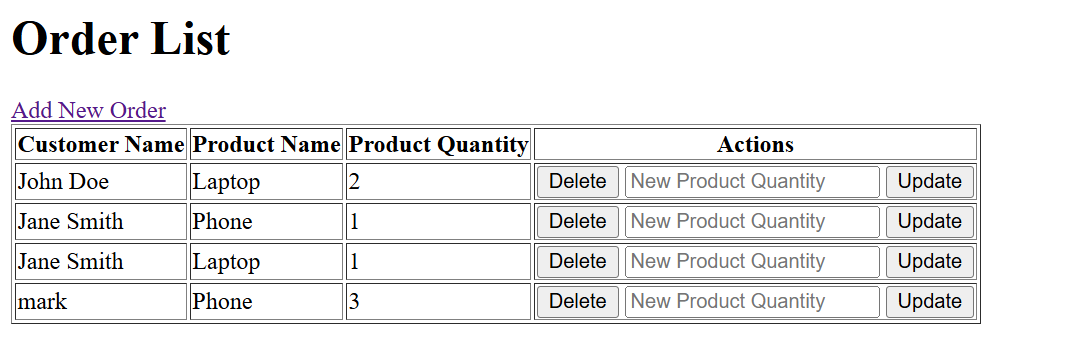
****

* + 1. **Order**

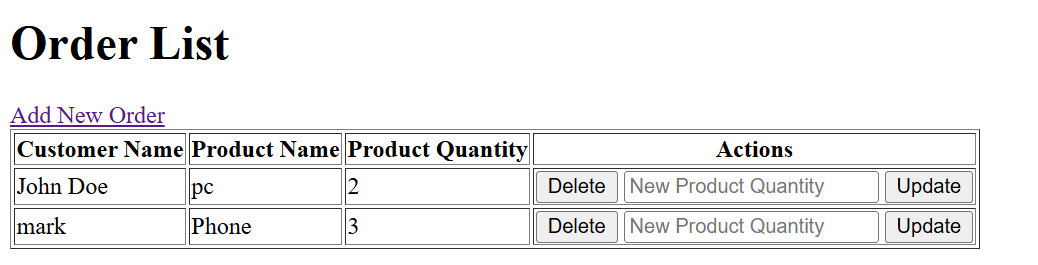
Menampilkan list dari order



Menambahkan data baru

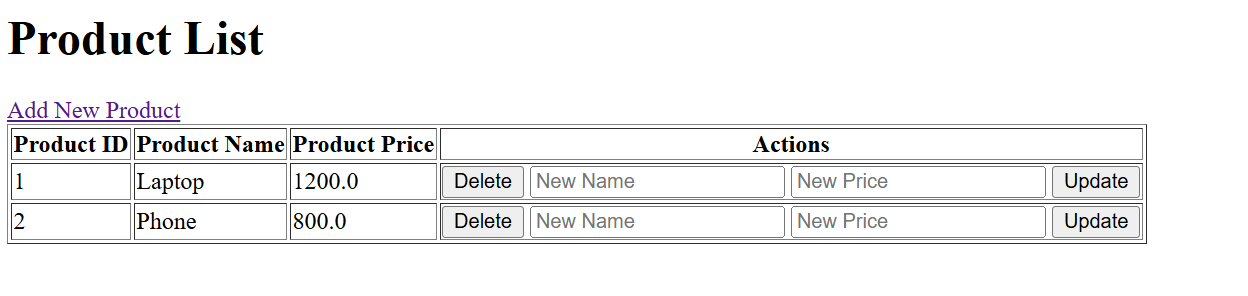


Mengganti data product name Laptop dengan pc dan menghapus data dengan customer name “Jane Smith”

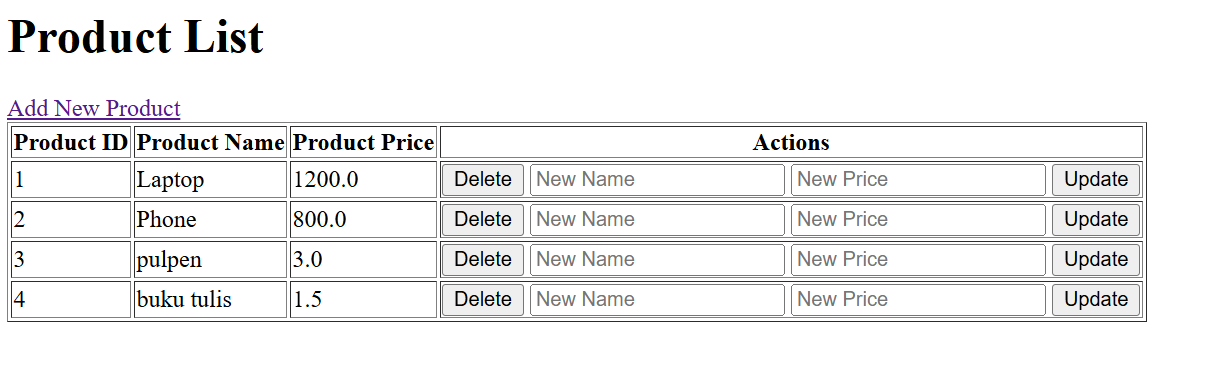


* + 1. **Product**

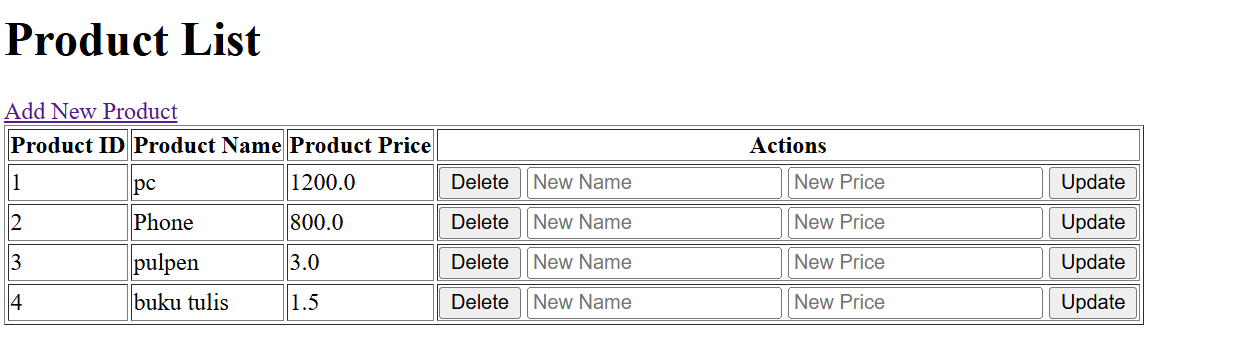
Menampilkan list dari product



Menambahkan data baru



Mengganti data product name laptop dengan pc



Menghapus data dengan product id 3