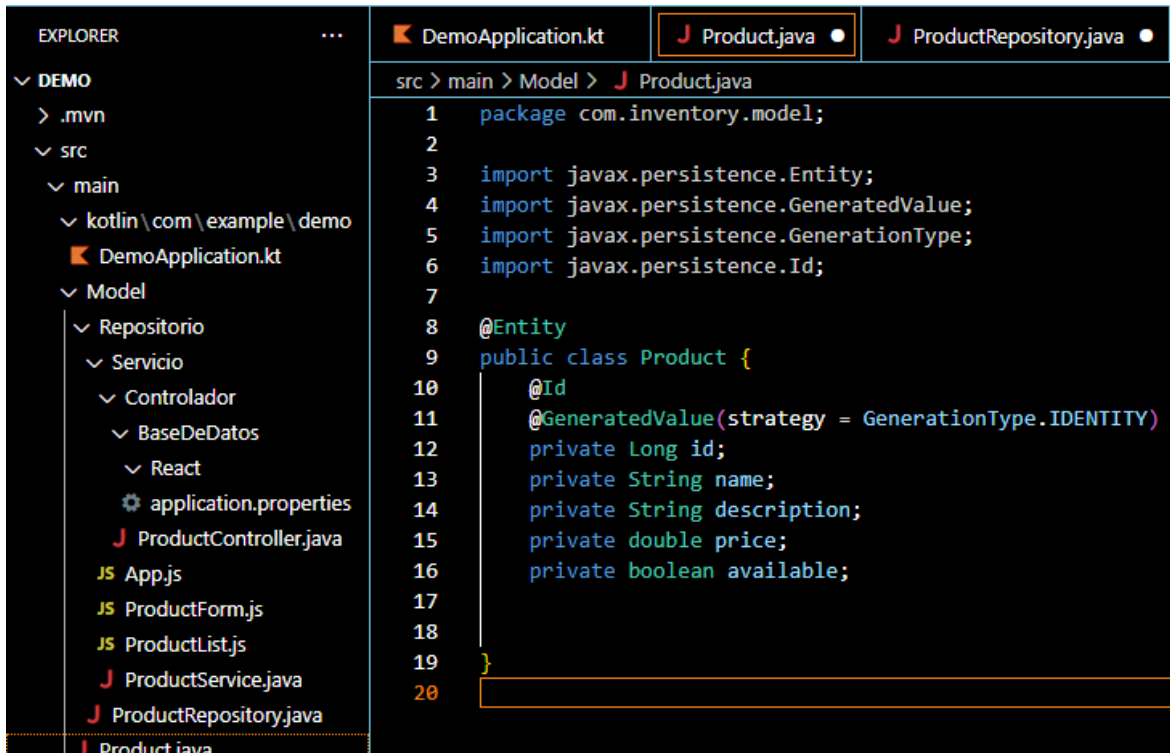


ALEJANDRA JASMIN MARROQUIN AGUILAR

0907-23-25100

SEGUNDO PARCIAL DE PROGRAMACION



```
1 package com.inventory.model;
2
3 import javax.persistence.Entity;
4 import javax.persistence.GeneratedValue;
5 import javax.persistence.GenerationType;
6 import javax.persistence.Id;
7
8 @Entity
9 public class Product {
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private Long id;
13     private String name;
14     private String description;
15     private double price;
16     private boolean available;
17
18 }
19
20
```

La clase `Product` define un producto en el sistema de inventario, mapeándose a una tabla en la base de datos. Incluye:

Entity: Marca la clase como entidad JPA.

Id: Define el campo `id` como clave primaria única.

GeneratedValue: Indica que `id` se genera automáticamente.

Campos:

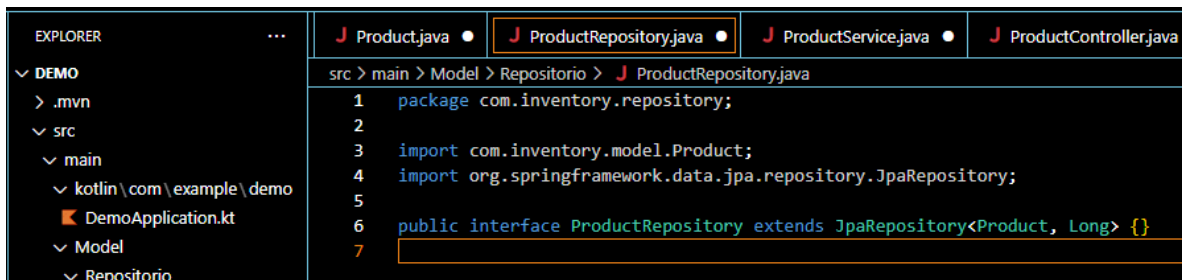
name: Nombre del producto.

description: Descripción del producto.

price: Precio del producto.

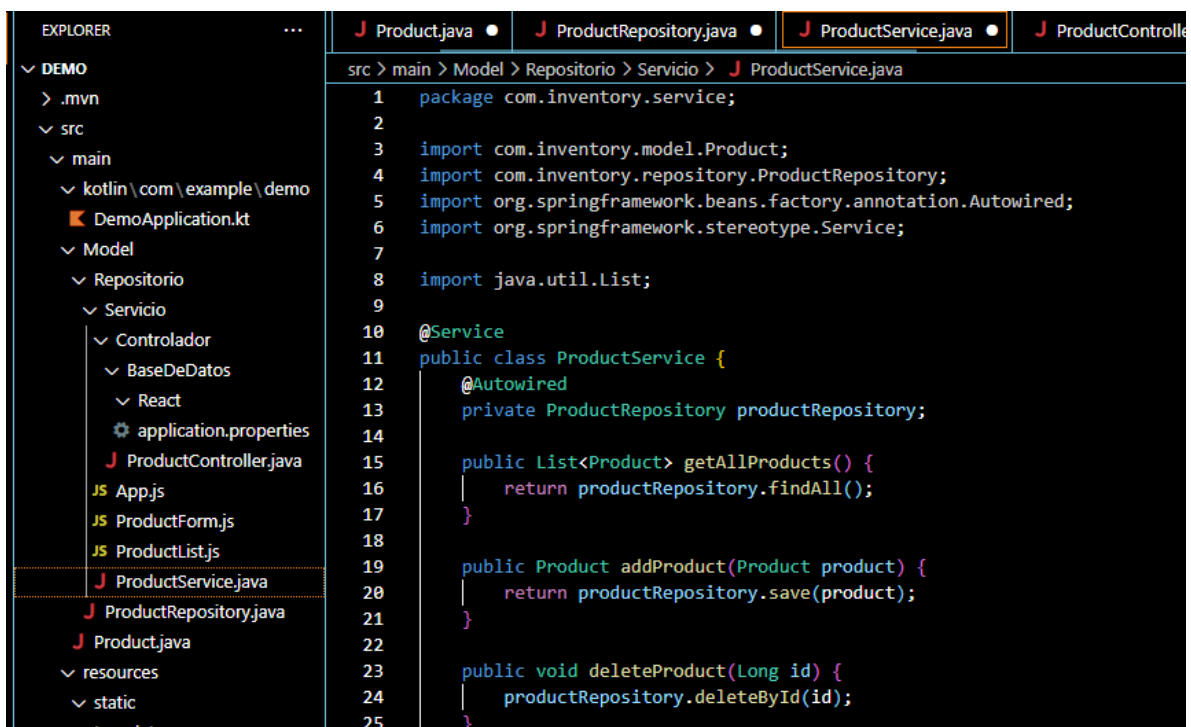
available: Indica si el producto está disponible.

Representa un producto con su nombre, descripción, precio y disponibilidad en la base de datos.



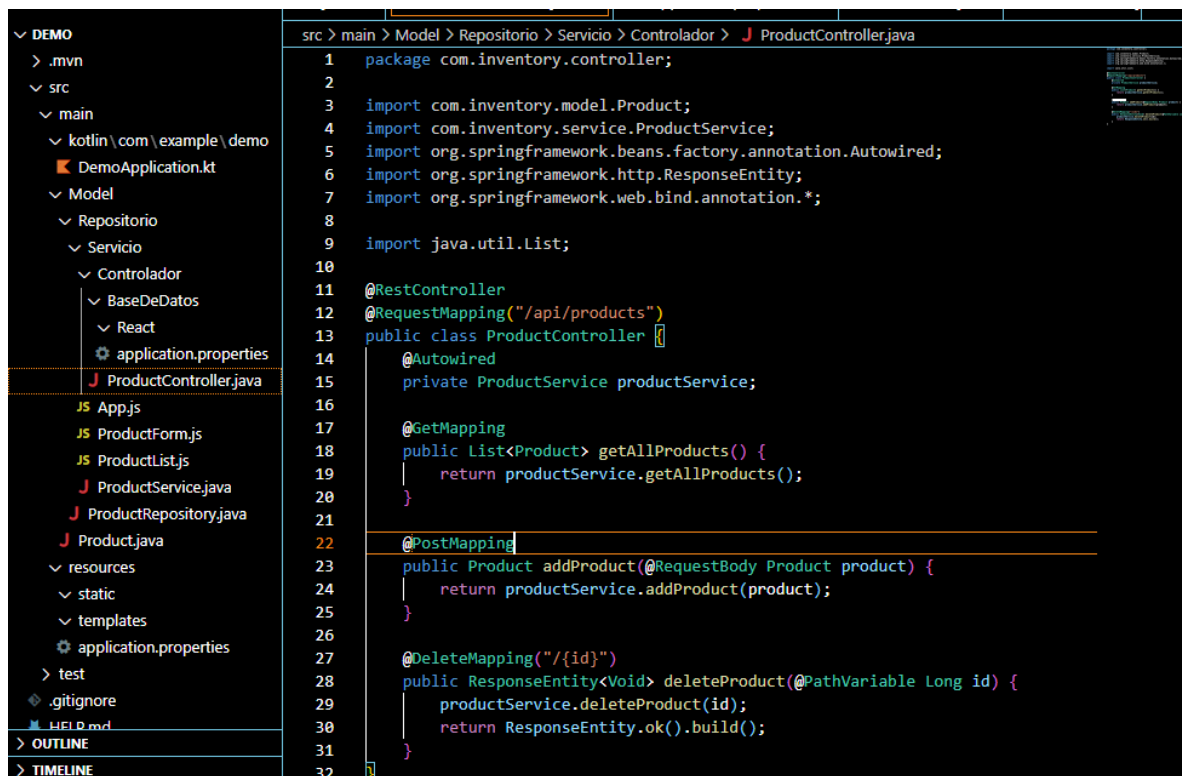
```
1 package com.inventory.repository;
2
3 import com.inventory.model.Product;
4 import org.springframework.data.jpa.repository.JpaRepository;
5
6 public interface ProductRepository extends JpaRepository<Product, Long> {}
7
```

permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la entidad Product.



```
1 package com.inventory.service;
2
3 import com.inventory.model.Product;
4 import com.inventory.repository.ProductRepository;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import java.util.List;
9
10 @Service
11 public class ProductService {
12     @Autowired
13     private ProductRepository productRepository;
14
15     public List<Product> getAllProducts() {
16         return productRepository.findAll();
17     }
18
19     public Product addProduct(Product product) {
20         return productRepository.save(product);
21     }
22
23     public void deleteProduct(Long id) {
24         productRepository.deleteById(id);
25     }
26 }
```

Gestiona la lógica de negocio para obtener, agregar y eliminar productos usando ProductRepository.



```
src > main > Model > Repositorio > Servicio > Controlador > J ProductController.java
1  package com.inventory.controller;
2
3  import com.inventory.model.Product;
4  import com.inventory.service.ProductService;
5  import org.springframework.beans.factory.annotation.Autowired;
6  import org.springframework.http.ResponseEntity;
7  import org.springframework.web.bind.annotation.*;
8
9  import java.util.List;
10
11  @RestController
12  @RequestMapping("/api/products")
13  public class ProductController {
14      @Autowired
15      private ProductService productService;
16
17      @GetMapping
18      public List<Product> getAllProducts() {
19          return productService.getAllProducts();
20      }
21
22      @PostMapping
23      public Product addProduct(@RequestBody Product product) {
24          return productService.addProduct(product);
25      }
26
27      @DeleteMapping("/{id}")
28      public ResponseEntity<Void> deleteProduct(@PathVariable Long id) {
29          productService.deleteProduct(id);
30          return ResponseEntity.ok().build();
31      }
32  }
```

Gestiona las solicitudes API para obtener, agregar y eliminar productos, utilizando ProductService para la lógica de negocio.

```
src > main > Model > Repositorio > Servicio > JS ProductList.js > [Ⓢ] default
1  import React, { useEffect, useState } from 'react';
2  import axios from 'axios';
3
4  const ProductList = () => {
5      const [products, setProducts] = useState([]);
6
7      useEffect(() => {
8          const fetchProducts = async () => {
9              const response = await axios.get('/api/products');
10             setProducts(response.data);
11         };
12         fetchProducts();
13     }, []);
14
15     return (
16         <div>
17             <h1>Lista de Productos</h1>
18             <ul>
19                 {products.map(product => (
20                     <li key={product.id}>{product.name} - {product.available ?
21                 ))}
22             </ul>
23         </div>
24     );
25 };
26
27 export default ProductList;
28
```

Muestra una lista de productos, recuperándolos del backend mediante una solicitud API.

```
... ctController.java • application.properties • JS ProductList.js • JS ProductForm.js • JS App.js •
src > main > Model > Repositorio > Servicio > JS ProductForm.js > ...
1 import React, { useState } from 'react';
2 import axios from 'axios';
3
4 const ProductForm = () => {
5   const [product, setProduct] = useState({ name: '', description: '', price:
6
7   const handleChange = (e) => {
8     setProduct({ ...product, [e.target.name]: e.target.value });
9   };
10
11   const handleSubmit = async (e) => {
12     e.preventDefault();
13     await axios.post('/api/products', product);
14     setProduct({ name: '', description: '', price: 0, available: true });
15   };
16
17   return (
18     <form onSubmit={handleSubmit}>
19       <input name="name" placeholder="Nombre" onChange={handleChange} rec
20       <input name="description" placeholder="Descripción" onChange={handl
21       <input name="price" type="number" placeholder="Precio" onChange={h
22       <button type="submit">Agregar Producto</button>
23     </form>
24   );
25 };
26
27 export default ProductForm;
28
```

Proporciona un formulario para agregar un producto, gestionando la entrada del usuario y enviando los datos al backend.