```java
package br.com.padon.application.controllers;

import br.com.padon.application.dtos.ProdutoFornecedorDto;
import br.com.padon.application.dtos.ProdutoLojaDto;
import br.com.padon.application.dtos.ProdutoVendaDto;
import br.com.padon.application.models.Conter;
import br.com.padon.application.models.Fornece;
import br.com.padon.application.models.Pertence;
import br.com.padon.application.models.Produto;
import br.com.padon.application.repositorys.ConterRepository;
import br.com.padon.application.repositorys.ForneceRepository;
import br.com.padon.application.repositorys.PertenceRepository;
import br.com.padon.application.repositorys.ProdutoRepository;
import com.fasterxml.jackson.databind.JsonNode;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

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

@RestController
@CrossOrigin
@RequestMapping("/produto")
public class ProdutoController {

    @Autowired
    private ProdutoRepository produto;
    @Autowired
    private ForneceRepository fornece;
    @Autowired
    private PertenceRepository pertence;
    @Autowired
    private ConterRepository conter;

    @PostMapping("/create")
    public Produto createProduto(@RequestBody JsonNode node) {
        return produto.save(new Produto(
```

```
39                  node.get("nome").asText(),
40                  node.get("fabricante").asText(),
41                  node.get("image").asText(),
42                  node.get("codigoDeBarras").asDouble(),
43                  node.get("bloqueado").asBoolean(),
44                  node.get("precoPorQuilo").asDouble(),
45                  node.get("precoPorUnidade").asDouble(),
46                  node.get("porQuilo").asBoolean()
47          ));
48      }
49
50      @GetMapping("/get")
51      public List<Produto> getAllProdutos() {
52          return produto.findAll();
53      }
54
55      @PostMapping("/save")
56      public Produto saveProduto(@RequestBody JsonNode node
    ) {
57          Produto produtoById = produto.findById(node.get("
    id").asInt()).orElseThrow();
58          produtoById.setNome(node.get("nome").asText());
59          produtoById.setFabricante(node.get("fabricante").
    asText());
60          produtoById.setImage(node.get("image").asText());
61          produtoById.setCodigoDeBarras(node.get("
    codigoDeBarras").asDouble());
62          produtoById.setBloqueado(node.get("bloqueado").
    asBoolean());
63          produtoById.setPrecoPorQuilo(node.get("
    precoPorQuilo").asDouble());
64          produtoById.setPrecoPorUnidade(node.get("
    precoPorUnidade").asDouble());
65          produtoById.setPorQuilo(node.get("porQuilo").
    asBoolean());
66          return produto.save(produtoById);
67      }
68
69      @PostMapping("/byid")
70      public Optional<Produto> getProduto(@RequestBody
    JsonNode node) {
71          return produto.findById(node.get("id").asInt());
72      }
73
```

```
 74     @PostMapping("/delete")
 75     public boolean deleteProduto(@RequestBody JsonNode
    node) {
 76         produto.deleteById(node.get("id").asInt());
 77         return true;
 78     }
 79
 80     @PostMapping("/fromcategoria")
 81     public List<Produto> getProdutoFromCategoria(@
    RequestBody JsonNode node) {
 82         return produto.getProdutosFromCategoria(node.get(
    "id").asInt());
 83     }
 84
 85     @PostMapping("/outcategoria")
 86     public List<Produto> getProdutoOutCategoria(@
    RequestBody JsonNode node) {
 87         return produto.getProdutosOutCategoria(node.get("
    id").asInt());
 88     }
 89
 90     @PostMapping("/fromfornecedor")
 91     public List<ProdutoFornecedorDto>
    getProdutoFromFornecedor(@RequestBody JsonNode node) {
 92         int fornecedorId = node.get("id").asInt();
 93         return produto.getProdutosFromFornecedor(
    fornecedorId).stream().map(p -> {
 94             Fornece forneceModel = fornece.getFornece(p.
    getProdutoId(), fornecedorId);
 95             return new ProdutoFornecedorDto(
 96                     p.getProdutoId(),
 97                     p.getNome(),
 98                     p.getFabricante(),
 99                     p.getImage(),
100                     p.getCodigoDeBarras(),
101                     p.isBloqueado(),
102                     p.getPrecoPorQuilo(),
103                     p.getPrecoPorUnidade(),
104                     p.isPorQuilo(),
105                     forneceModel != null ? forneceModel.
    getPreco() : 0
106             );
107         }).collect(Collectors.toList());
108     }
```

```
109
110     @PostMapping("/outfornecedor")
111     public List<ProdutoFornecedorDto>
    getProdutoOutFornecedor(@RequestBody JsonNode node) {
112         int fornecedorId = node.get("id").asInt();
113         return produto.getProdutosOutFornecedor(
    fornecedorId).stream().map(p -> {
114             Fornece forneceModel = fornece.getFornece(p.
    getProdutoId(), fornecedorId);
115             return new ProdutoFornecedorDto(
116                     p.getProdutoId(),
117                     p.getNome(),
118                     p.getFabricante(),
119                     p.getImage(),
120                     p.getCodigoDeBarras(),
121                     p.isBloqueado(),
122                     p.getPrecoPorQuilo(),
123                     p.getPrecoPorUnidade(),
124                     p.isPorQuilo(),
125                     forneceModel != null ? forneceModel.
    getPreco() : 0
126             );
127         }).collect(Collectors.toList());
128     }
129
130     @PostMapping("/fromvenda")
131     public List<ProdutoVendaDto> getProdutoFromVenda(@
    RequestBody JsonNode node) {
132         int vendaId = node.get("id").asInt();
133         return produto.getProdutosFromVenda(vendaId).
    stream().map(p -> {
134             Pertence pertenceModel = pertence.getPertence
    (p.getProdutoId(), vendaId);
135             return new ProdutoVendaDto(
136                     p.getProdutoId(),
137                     p.getNome(),
138                     p.getFabricante(),
139                     p.getImage(),
140                     p.getCodigoDeBarras(),
141                     p.isBloqueado(),
142                     p.getPrecoPorQuilo(),
143                     p.getPrecoPorUnidade(),
144                     p.isPorQuilo(),
145                     pertenceModel.getPrecoTotal(),
```

```
146                     pertenceModel.getQuantidade(),
147                     pertenceModel.getPrecoAtual()
148             );
149         }).collect(Collectors.toList());
150     }
151 
152     @PostMapping("/outvenda")
153     public List<ProdutoVendaDto> getProdutoOutVenda(@
    RequestBody JsonNode node) {
154         int vendaId = node.get("id").asInt();
155         return produto.getProdutosOutVenda(vendaId).
    stream().map(p -> {
156             Pertence pertenceModel = pertence.getPertence
    (p.getProdutoId(), vendaId);
157             return new ProdutoVendaDto(
158                     p.getProdutoId(),
159                     p.getNome(),
160                     p.getFabricante(),
161                     p.getImage(),
162                     p.getCodigoDeBarras(),
163                     p.isBloqueado(),
164                     p.getPrecoPorQuilo(),
165                     p.getPrecoPorUnidade(),
166                     p.isPorQuilo(),
167                     pertenceModel != null ? pertenceModel
    .getPrecoTotal() : 0,
168                     pertenceModel != null ? pertenceModel
    .getQuantidade() : 0,
169                     pertenceModel != null ? pertenceModel
    .getPrecoAtual() : 0
170             );
171         }).collect(Collectors.toList());
172     }
173 
174     @PostMapping("/fromloja")
175     public List<ProdutoLojaDto> getProdutoFromLoja(@
    RequestBody JsonNode node) {
176         int lojaId = node.get("id").asInt();
177         return produto.getProdutosFromLoja(lojaId).stream
    ().map(p -> {
178             Conter conterModel = conter.getConter(p.
    getProdutoId(), lojaId);
179             return new ProdutoLojaDto(
180                     p.getProdutoId(),
```

```
181                     p.getNome(),
182                     p.getFabricante(),
183                     p.getImage(),
184                     p.getCodigoDeBarras(),
185                     p.isBloqueado(),
186                     p.getPrecoPorQuilo(),
187                     p.getPrecoPorUnidade(),
188                     p.isPorQuilo(),
189                     conterModel != null ? conterModel.
    getEstoque() : 0,
190                     conterModel != null ? conterModel.
    getQuantidadeMinima() : 0
191             );
192         }).collect(Collectors.toList());
193     }
194 
195     @PostMapping("/outloja")
196     public List<ProdutoLojaDto> getProdutoOutLoja(@
    RequestBody JsonNode node) {
197         int lojaId = node.get("id").asInt();
198         return produto.getProdutosOutLoja(lojaId).stream
    ().map(p -> {
199             Conter conterModel = conter.getConter(p.
    getProdutoId(), lojaId);
200             return new ProdutoLojaDto(
201                     p.getProdutoId(),
202                     p.getNome(),
203                     p.getFabricante(),
204                     p.getImage(),
205                     p.getCodigoDeBarras(),
206                     p.isBloqueado(),
207                     p.getPrecoPorQuilo(),
208                     p.getPrecoPorUnidade(),
209                     p.isPorQuilo(),
210                     conterModel != null ? conterModel.
    getEstoque() : 0,
211                     conterModel != null ? conterModel.
    getQuantidadeMinima() : 0
212             );
213         }).collect(Collectors.toList());
214     }
215 }
216 
```