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

import br.com.padon.application.models.Funcionario;
import br.com.padon.application.repositorys.
FuncionarioRepository;
import com.fasterxml.jackson.databind.JsonNode;
import org.springframework.beans.factory.annotation.
Autowired;
import org.springframework.security.crypto.bcrypt.
BCryptPasswordEncoder;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@CrossOrigin
@RequestMapping("/funcionario")
public class FuncionarioController {

    @Autowired
    private FuncionarioRepository funcionario;
    private final BCryptPasswordEncoder senha = new
BCryptPasswordEncoder();

    @PostMapping("/create")
    public Funcionario createFuncionario(@RequestBody
JsonNode node) {
        return funcionario.save(new Funcionario(
                node.get("cpf").asText(),
                node.get("usuario").asText(),
                node.get("nome").asText(),
                senha.encode(node.get("senha").asText()),
                node.get("email").asText(),
                node.get("telefone").asText(),
                node.get("gerente").asBoolean()
        ));
    }

    @GetMapping("/get")
    public List<Funcionario> getAllFuncionarios() {
        return funcionario.findAll();
    }

```

```
40      @PostMapping("/save")
41      public Funcionario saveFuncionario(@RequestBody
    JsonNode node) {
42          Funcionario funcionarioById = funcionario.findById
    (node.get("cpf").asText()).orElseThrow();
43          funcionarioById.setUsuario(node.get("usuario").
    asText());
44          funcionarioById.setNome(node.get("nome").asText
    ());
45          funcionarioById.setSenha(senha.encode(node.get("
    senha").asText()));
46          funcionarioById.setEmail(node.get("email").asText
    ());
47          funcionarioById.setTelefone(node.get("telefone").
    asText());
48          funcionarioById.setGerente(node.get("gerente").
    asBoolean());
49          return funcionario.save(funcionarioById);
50      }
51
52      @PostMapping("/byid")
53      public Optional<Funcionario> getFuncionario(@
    RequestBody JsonNode node) {
54          return funcionario.findById(node.get("cpf").asText
    ());
55      }
56
57      @PostMapping("/delete")
58      public boolean deleteFuncionario(@RequestBody JsonNode
     node) {
59          funcionario.deleteById(node.get("cpf").asText());
60          return true;
61      }
62
63      @PostMapping("/fromloja")
64      public List<Funcionario> getProdutoFromLoja(@
    RequestBody JsonNode node) {
65          return funcionario.getFuncionarioFromLoja(node.get
    ("id").asInt());
66      }
67
68      @PostMapping("/outloja")
69      public List<Funcionario> getProdutoOutLoja(@
    RequestBody JsonNode node) {
```

```
70          return funcionario.getFuncionarioOutLoja(node.get
   ("id").asInt());
71      }
72 }
73
```