

```
1  @RestController
2  @ResponseBody
3  public class Controller {
4
5      @Autowired
6      AdministradorService administradorService;
7
8      @Autowired
9      EnderecoService enderecoService;
10
11     @Autowired
12     MusicaService musicaService;
13
14     @Autowired
15     MusicoService musicoService;
16
17     @RequestMapping(value = "/cadastrarPessoa", method = RequestMethod.POST)
18     public ResponseEntity<String> cadastrarPessoa(@RequestParam("nome") String nome, @RequestParam("cpf") String cpf,
19         @RequestParam("telefone") String telefone, @RequestParam("celular") String celular, @RequestParam("email") String email,
20         @RequestParam("rua") String rua, @RequestParam("bairro") String bairro, @RequestParam("numero") int numero,
21         @RequestParam("cidade") String cidade, @RequestParam("complemento") String complemento,
22         @RequestParam("estado") String estado, @RequestParam("cep") String cep, @RequestParam("tipo") String tipo) {
23
24         try {
25             // verifica se o endereco ja existe no banco de dados
26             boolean enderecoExistente = enderecoService.verificaExistencia(cep);
27
28             // se o endereco nao existe, cadastra o endereco
29             if (!enderecoExistente) {
30                 enderecoService.cadastrarEndereco(rua, bairro, numero, cidade, complemento, estado, cep);
31             }
32
33             // cadastra a pessoa
34             administradorService.cadastrarPessoa(nome, cpf, telefone, celular, email, enderecoService.findByCep(cep), tipo);
35
36             return new ResponseEntity<>("Pessoa cadastrada com sucesso", HttpStatus.OK);
37         } catch (RuntimeException e) {
38             return new ResponseEntity<>("Erro: " + e.getMessage(), HttpStatus.BAD_REQUEST);
39         } catch (Exception e) {
40             return new ResponseEntity<>("Erro no banco de dados: " + e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
41         }
42     }
```



```
1 public class EnderecoService {
2     @Autowired
3     private EnderecoRepository enderecoRepository;
4
5     public void cadastrarEndereco(String rua, String bairro, int numero, String cidade,
6     String complemento, String estado, String cep) {
7
8         // verifica se o endereco ja existe no banco de dados
9         Endereco enderecoExistente = enderecoRepository.findByCep(cep);
10
11         // se o endereco ja existe, lanca uma excecao
12         if (enderecoExistente != null) {
13             throw new RuntimeException("Endereco já existe");
14         }
15
16         // Cria a instancia de Endereco
17         Endereco endereco = new Endereco(rua, bairro, numero, cidade, complemento, estado, cep);
18
19         // Salva o endereco no banco de dados
20         enderecoRepository.save(endereco.getId(), endereco.getRua(), endereco.getNumero(),
21         endereco.getBairro(), endereco.getCidade(), endereco.getEstado(), endereco.getCep());
22     }
23
24     // verifica se o endereco ja existe no banco de dados
25     public boolean verificaExistencia(String cep) {
26         Endereco enderecoExistente = enderecoRepository.findByCep(cep);
27         if (enderecoExistente != null) {
28             return true;
29         }
30         return false;
31     }
32
33     // retorna o endereco pelo cep
34     public Endereco findByCep(String cep) {
35         return enderecoRepository.findByCep(cep);
36     }
37 }
```






```
1 public interface EnderecoRepository extends JpaRepository<Endereco, Long>{
2
3     // query que vai salvar um endereco no banco de dados
4     @Query("INSERT INTO endereco (rua, numero, bairro, cidade, estado, cep) VALUES (:rua, :numero, :bairro, :cidade, :estado, :cep)")
5     void save(@Param("id") Long id ,@Param("rua") String rua, @Param("numero") int numero, @Param("bairro") String bairro,
6         @Param("cidade") String cidade, @Param("estado") String estado, @Param("cep") String cep);
7
8     // query findByRuaAndNumero
9     @Query("SELECT e FROM endereco e WHERE e.rua = :rua AND e.numero = :numero")
10    Endereco findByRuaAndNumero(String rua, int numero);
11
12    // query findByCep
13    @Query("SELECT e FROM endereco e WHERE e.cep = :cep")
14    Endereco findByCep(String cep);
15 }
```



```
1 public class AdministradorService {
2     @Autowired
3     private AdministradorRepository administradorRepository;
4
5     public String cadastrarPessoa(String nome, String cpf, String telefone, String celular,
6         String email, Endereco endereco, String tipo) {
7
8         try {
9             // verifica se a pessoa ja existe no banco de dados
10            Administrador administradorExistente = administradorRepository.findByCpf(cpf);
11
12            // se a pessoa ja existe, lanca uma excecao
13            if (administradorExistente != null) {
14                throw new RuntimeException("Pessoa já existe");
15            }
16
17            // verifica o tipo de pessoa
18            if (tipo.equals("administrador")) {
19
20                // Cria a instancia de Administrador
21                Administrador administrador = new Administrador(nome, cpf, telefone, celular, email, endereco);
22
23                // Salva o administrador no banco de dados
24                administradorRepository.saveAdministrador(administrador.getId(), administrador.getNome(),
25                    administrador.getCpf(), administrador.getTelefone(), administrador.getCelular(),
26                    administrador.getEmail(), administrador.getEndereco());
27
28            } else if (tipo.equals("musico")) {
29                // Salva o musico no banco de dados
30                Musico musico = new Musico(nome, cpf, telefone, celular, email, endereco);
31                administradorRepository.saveMusico(musico.getId(), musico.getNome(), musico.getCpf(),
32                    musico.getTelefone(), musico.getCelular(), musico.getEmail(), musico.getEndereco());
33            }
34
35            return "Sucesso";
36        } catch (RuntimeException e) {
37            return "Erro: " + e.getMessage();
38        } catch (Exception e) {
39            return "Erro no banco de dados: " + e.getMessage();
40        }
41    }
```





```
1  @Repository
2  public interface AdministradorRepository extends JpaRepository<Administrador, Long> {
3
4      // query que vai salvar um administrador ou um musico no banco de dados
5      @Query("INSERT INTO administrador (nome, cpf, telefone, celular, email, endereco) VALUES (:nome, :cpf, :telefone, :celular, :email, :endereco)")
6      void saveAdministrador(@Param("id") Long id ,@Param("nome") String nome, @Param("cpf") String cpf, @Param("telefone")
7          String telefone, @Param("celular") String celular, @Param("email") String email, @Param("endereco") Endereco endereco);
8
9      // query que vai salvar um musico no banco de dados
10     @Query("INSERT INTO musico (nome, cpf, telefone, celular, email, endereco) VALUES (:nome, :cpf, :telefone, :celular, :email, :endereco)")
11     void saveMusico(@Param("id") Long id ,@Param("nome") String nome, @Param("cpf") String cpf, @Param("telefone") String telefone,
12         @Param("celular") String celular, @Param("email") String email, @Param("endereco") Endereco endereco);
13
14     // query que vai buscar uma pessoa pelo cpf
15     @Query("SELECT p FROM pessoa p WHERE p.cpf = :cpf")
16     Administrador findByCpf(@Param("cpf") String cpf);
17 }
```



```
1 @RequestMapping(value = "/gravarMusica", method = RequestMethod.POST)
2 public ResponseEntity<String> cadastrarMusica(@RequestParam("id_estiloMusical") Long id_estiloMusical, @RequestParam("nome") String nome,
3     @RequestParam("duracao") String duracao, @RequestParam("data_Criacao") String data_Criacao, @RequestParam("autores_letra") String autores_letra,
4     @RequestParam("autores_musica") String autores_musica, @RequestParam("instrumentos") String instrumentos,
5     @RequestParam("produtor") String produtor, @RequestParam("musicoQueGravou") String musicoQueGravou,
6     @RequestParam("id_banda") String id_banda) {
7
8     // transforma os parametros em listas
9     List<Instrumento> listaInstrumentos = musicaService.transformaInstrumentoLista(instrumentos);
10    List<Musico> listaAutoresLetra = musicaService.transformaMusicoLista(autores_letra);
11    List<Musico> listaAutoresMusica = musicaService.transformaMusicoLista(autores_musica);
12    Map<Long, Long> listaMusicoQueGravou = musicaService.transformaMap(musicoQueGravou);
13
14    // faz a verificação se os musicos ja estao associados ao instrumento
15    // verifica o id do musico e o id do instrumento presentes em listaMusicoQueGravou
16    for (Map.Entry<Long, Long> entry : listaMusicoQueGravou.entrySet()) {
17        Long idMusico = entry.getKey();
18        Long idInstrumento = entry.getValue();
19
20        // verifica se o musico ja esta associado ao instrumento
21        boolean musicoAssociado = musicoService.verificaInstrumento(idMusico, idInstrumento);
22
23        // se o musico nao esta associado ao instrumento, cadastra a associacao
24        if (!musicoAssociado) {
25            musicoService.associarInstrumento(idMusico, idInstrumento);
26        }
27    }
28
29    try {
30        // cadastra a musica
31        musicaService.gravarMusica(id_estiloMusical, nome, duracao, data_Criacao, listaAutoresLetra, listaAutoresMusica,
32            listaInstrumentos, produtor, listaMusicoQueGravou, id_banda);
33
34        return new ResponseEntity<>("Musica cadastrada com sucesso", HttpStatus.OK);
35    } catch (RuntimeException e) {
36        return new ResponseEntity<>("Erro: " + e.getMessage(), HttpStatus.BAD_REQUEST);
37    } catch (Exception e) {
38        return new ResponseEntity<>("Erro no banco de dados: " + e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
39    }
40 }
```



```
1 public class MusicoService {
2
3     @Autowired
4     private MusicoRepository musicoRepository;
5
6     public boolean verificaInstrumento(Long id_musico, Long id_instrumento) {
7
8         // verifica no banco de dados se o musico tem aquele instrumento na sua lista de instrumentos
9         boolean resposta = musicoRepository.findById(id_musico).get().getInstrumentos().contains(id_instrumento);
10        if (resposta) {
11            return true;
12        } else {
13            return false;
14        }
15    }
16
17    // associa um instrumento a um musico
18    public boolean associarInstrumento(Long id_musico, Long id_instrumento) {
19        // verifica se o musico ja tem aquele instrumento
20        if (verificaInstrumento(id_musico, id_instrumento)) {
21            return false;
22        } else {
23            // adiciona o instrumento na lista de instrumentos do musico
24            musicoRepository.findById(id_musico).get().getInstrumentos().add(id_instrumento);
25            return true;
26        }
27    }
28 }
```