

```
1 package br.com.uniamerica.estacionamento.controller;
2
3 import br.com.uniamerica.estacionamento.entity.Condutor;
4 import br.com.uniamerica.estacionamento.entity.Marca;
5 import br.com.uniamerica.estacionamento.repository.
6 MarcaRepository;
7 import org.springframework.beans.factory.annotation.
8 Autowired;
9 import org.springframework.dao.
10 DataIntegrityViolationException;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.stereotype.Controller;
13 import org.springframework.web.bind.annotation.*;
14
15 @Controller
16 @RequestMapping(value = "/api/marca")
17
18 public class MarcaController {
19     @Autowired
20         public MarcaRepository marcaRepository;
21     @GetMapping //outra forma de buscar por id
22     public ResponseEntity<?> findByIdRequest(@RequestParam
("id")final Long id ){
23         final Marca marca = this.marcaRepository.findById
(id).orElse(null);
24         return marca == null
25             ? ResponseEntity.badRequest().body("nenhum
26 valor encontrado" :
27                 ResponseEntity.ok(marca);
28
29     @GetMapping("/lista")
30         public ResponseEntity<?> retornaLista (){
31             return ResponseEntity.ok(this.marcaRepository.
32             findAll());
33
34     @GetMapping("/listaAtiva")
35         public ResponseEntity<?> retornaListaAtiva (){
36             List <Marca> listaMarca = this.marcaRepository.
37             findAll();
```

```
37         List<Marca> checaLista = new ArrayList<>();
38         for (Marca valor:listaMarca
39             ) {
40             if (valor.isAtivo()){
41                 checaLista.add(valor);
42             }
43
44         }
45     }
46     return ResponseEntity.ok(checaLista);
47 }
48
49 @PostMapping
50     public ResponseEntity<?> cadastraMarca(@
51 RequestBody final Marca marca){
52     this.marcaRepositorio.save(marca);
53     return ResponseEntity.ok("Marca cadastrada com
54 sucesso");
55 }
56     @PutMapping("/{id}")
57     public ResponseEntity<?> atualizaMarca(@
58 PathVariable("id") final Long id,
59     @
60 RequestBody final Marca marca)
61     {
62         try {
63             final Marca marcaBanco = this.marcaRepositorio
64 .findById(id).orElse(null);
65             if (marcaBanco == null || !marcaBanco.getId()
66 .equals(marca.getId())){
67                 throw new RuntimeException("Não foi
68     possível localizar");
69             }
70             this.marcaRepositorio.save(marca);
71             return ResponseEntity.ok("Marca atualizada!");
72         } catch (DataIntegrityViolationException e){
73             return ResponseEntity.internalServerError()
74 .body("Error" + e.getCause().getCause().getMessage());
75         }
76         catch (RuntimeException e){
77             return ResponseEntity.internalServerError()
78 .body(e.getMessage());
79         }
80     }
81 }
```

```
72      }
73
74      @DeleteMapping ("/{id}")
75      public ResponseEntity <?> inativar(@PathVariable("id"
76      ) Long id){
77          try {
78              final Marca marcaBanco = this.
79              marcaRepositorio.findById(id).orElse(null);
80              if (marcaBanco == null ){
81                  throw new RuntimeException("Não foi
82                  possível identificar o registro");
83              }
84              catch (DataIntegrityViolationException e){
85                  return ResponseEntity.internalServerError().
86                  body("Error"+e + e.getCause().getCause().getMessage());
87              }
88              catch (Exception e){
89                  final Marca marcaBanco = this.
90                  marcaRepositorio.findById(id).orElse(null);
91                  if (marcaBanco == null ){
92                      throw new RuntimeException("Não foi
93                      possível identificar o registro");
94                  }
95                  marcaBanco.setAtivo(false);
96                  this.marcaRepositorio.save(marcaBanco);
97                  return ResponseEntity.ok("Registro Inativado"
98 );
99
100
101
102 }
103
```

```
1 package br.com.uniamerica.estacionamento.controller;
2
3 import br.com.uniamerica.estacionamento.entity.Modelo;
4 import br.com.uniamerica.estacionamento.entity.
5     Movimentacao;
6 import br.com.uniamerica.estacionamento.repository.
7     ModeloRepositorio;
8 import org.springframework.dao.
9     DataIntegrityViolationException;
10 import org.springframework.http.ResponseEntity;
11 import org.springframework.stereotype.Controller;
12 import org.springframework.stereotype.Repository;
13 import org.springframework.web.bind.annotation.*;
14
15 @Controller
16 @RequestMapping(value = "/api/modelo")
17
18 public class ModeloController {
19     private ModeloRepositorio modeloRepositorio;
20     public ModeloController(ModeloController
21         modeloController ){
22         this.modeloRepositorio = modeloRepositorio;
23     }
24     @GetMapping
25     public ResponseEntity<?> buscaModelo(@RequestParam("id
") final Long id){
26         final Modelo modelo = this.modeloRepositorio.
27             findById(id).orElse(null);
28         return modelo == null ? ResponseEntity.badRequest
29             ().body("Não localizado") :
30             ResponseEntity.ok(modelo);
31     }
32     @GetMapping("/lista")
33     public ResponseEntity<?> buscaListaModelo(){
34         return ResponseEntity.ok(this.modeloRepositorio.
35             findAll());
36     }
37     @GetMapping("/listaativo")
38     public ResponseEntity<?> buscaListaAtivo(){
```

```
37         List <Modelo> modeloLista = this.modeloRepositorio
38             .findAll();
39         List <Modelo> checaLista      = new ArrayList<>();
40         for (Modelo valor:modeloLista
41             ) {
42             if ( valor.isAtivo()){
43                 checaLista.add(valor);
44             }
45         }
46         return ResponseEntity.ok(checaLista);
47     }
48
49     @PutMapping("/{id}")
50     public ResponseEntity<?> atualizaModelo(@PathVariable
51     final Long id, @RequestBody Modelo modelo) {
52         try {
53             Modelo valorBanco = this.modeloRepositorio.
54             findById(id).orElse(null);
55             if (valorBanco == null || !valorBanco.getId().
56             equals(modelo.getId())) {
57                 throw new RuntimeException("Não foi
58                 possível localizar");
59             }
60             this.modeloRepositorio.save(modelo);
61             return ResponseEntity.ok("Modelo Atualizado
62             com sucesso.");
63         } catch (DataIntegrityViolationException e) {
64             return ResponseEntity.internalServerError().
65             body("Error" + e.getCause().getCause().getMessage());
66         } catch (RuntimeException e) {
67             return ResponseEntity.internalServerError().
68             body("Error");
69         }
70     }
71
72     @DeleteMapping("/{id}")
73     public ResponseEntity<?> excluiModelo(@PathVariable
74     final Long id) {
75         try {
76             Modelo valorBanco = this.modeloRepositorio.
77             findById(id).orElse(null);
78             if (valorBanco == null ) {
79                 throw new RuntimeException("Não foi
80                 possível localizar");
```

```
70          }
71          this.modeloRepositorio.delete(valorBanco);
72          return ResponseEntity.ok("Modelo deletado com
    sucesso.");
73      } catch (DataIntegrityViolationException e) {
74          return ResponseEntity.internalServerError().
    body("Error" + e.getCause().getCause().getMessage());
75      } catch (Exception e){
76          final Modelo valorBanco = this.
    modeloRepositorio.findById(id).orElse(null);
77          if (valorBanco == null){
78              throw new RuntimeException("Não foi
    possível identificar o registro");
79          }
80          valorBanco.setAtivo(false);
81          this.modeloRepositorio.save(valorBanco);
82          return ResponseEntity.ok("Registro Inativado"
    );
83      }
84  }
85
86
87
88 }
89
```

```
1 package br.com.uniamerica.estacionamento.controller;
2
3 import br.com.uniamerica.estacionamento.entity.Modelo;
4 import br.com.uniamerica.estacionamento.entity.Veiculo;
5 import br.com.uniamerica.estacionamento.repository.
6 VeiculoRepository;
7 import org.springframework.dao.
8 DataIntegrityViolationException;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.stereotype.Controller;
11 import org.springframework.web.bind.annotation.*;
12
13
14 @Controller
15 @RequestMapping(value = "/api/veiculo")
16
17 public class VeiculoController {
18     private VeiculoRepository veiculoRepository;
19
20     public VeiculoController(VeiculoRepository
21 veiculoRepository) {
22         this.veiculoRepository = veiculoRepository;
23     }
24
25     @GetMapping
26     public ResponseEntity<?> buscaVeiculoPorId(@
27 RequestParam("id") final Long id) {
28         final Veiculo veiculo = this.veiculoRepository.
29 findById(id).orElse(null);
30         return ResponseEntity.ok(veiculo);
31     }
32
33     @GetMapping
34     public ResponseEntity<?> buscaLista() {
35         return ResponseEntity.ok(this.veiculoRepository.
36 findAll());
37     }
38
39     @GetMapping("/listaativo")
40     public ResponseEntity<?> buscaListaAtivo() {
41         List<Veiculo> listaVeiculo = this.
42 veiculoRepository.findAll();
```

```
38         List<Veiculo> checaLista = new ArrayList();
39         for (Veiculo valor : checaLista
40 ) {
41             if (valor.isAtivo()) {
42                 checaLista.add(valor);
43             }
44         }
45         return ResponseEntity.ok(checaLista);
46     }
47
48     @PostMapping
49     public ResponseEntity<?> cadastraVeiculo(@RequestBody
final Veiculo veiculo) {
50         this.veiculoRepositorio.save(veiculo);
51         return ResponseEntity.ok("Cadastrado com sucesso.");
52     }
53
54     @PutMapping("/{id}")
55     public ResponseEntity<?> atualizaVeiculo(@PathVariable
final Long id, @RequestBody Veiculo veiculo) {
56         try {
57             Veiculo valorBanco = this.veiculoRepositorio.
findById(id).orElse(null);
58             if (valorBanco == null || !valorBanco.getId().
equals(veiculo.getId())) {
59                 throw new RuntimeException("Não localizado
");
60             }
61             this.veiculoRepositorio.save(veiculo);
62             return ResponseEntity.ok("Veiculo atualizado.");
63         } catch (DataIntegrityViolationException e) {
64             return ResponseEntity.internalServerError().
body("Error" + e.getCause().getCause().getMessage());
65         } catch (RuntimeException e) {
66             return ResponseEntity.internalServerError().
body("Error");
67         }
68     }
69
70     @DeleteMapping("/{id}")
71     public ResponseEntity<?> atualizaVeiculo(@PathVariable
final Long id) {
```

```
72     try {
73         Veiculo valorBanco = this.veiculoRepositorio.
74             findById(id).orElse(null);
75         if (valorBanco == null) {
76             throw new RuntimeException("Não
77                 localizado");
78         }
79         this.veiculoRepositorio.delete(valorBanco);
80         return ResponseEntity.ok("Veiculo Deletado."
81 );
82     } catch (DataIntegrityViolationException e) {
83         return ResponseEntity.internalServerError().
84             body("Error" + e.getCause().getCause().getMessage());
85     } catch (RuntimeException e) {
86         Veiculo valorBanco = this.veiculoRepositorio.
87             findById(id).orElse(null);
88         if (valorBanco == null) {
89             throw new RuntimeException("Não
90                 localizado");
91         }
92     }
93 }
```

```
1 package br.com.uniamerica.estacionamento.controller;
2
3 import br.com.uniamerica.estacionamento.entity.Condutor;
4 import br.com.uniamerica.estacionamento.repository.
CondutorRepository;
5 import br.com.uniamerica.estacionamento.repository.
ModeloRepository;
6 import org.springframework.beans.factory.annotation.
Autowired;
7 import org.springframework.dao.
DataIntegrityViolationException;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.stereotype.Controller;
10 import org.springframework.stereotype.Repository;
11 import org.springframework.web.bind.annotation.*;
12
13 import java.util.ArrayList;
14 import java.util.List;
15
16 @Controller
17 @RequestMapping(value = "/api/condutor")
18 public class CondutorController {
19
20     // @Autowired
21     // private CondutorRepository condutorRepository;
22     @Autowired
23     private CondutorRepository condutorRepository;
24     public CondutorController (CondutorRepository
condutorRepository){
25         this.condutorRepository = condutorRepository;
26     }
27
28     /*
29      http://localhost:8080/api/condutor/1
30
31     */
32 /**
33     @GetMapping("/{id}") //
34     public ResponseEntity <?> findByIdRequest(@
PathVariable("id")final long id ){
35         //return ResponseEntity.ok(new Condutor());
36
37         return ResponseEntity.ok(this.condutorRepository.
findById(id).orElse(null));

```

```
38 }*/  
39 /*  
40     http://localhost:8080/api/condutor?id=1  
41  
42     */  
43  
44     @GetMapping //outra forma de buscar po id  
45     public ResponseEntity <?> findByIdRequest(@  
        RequestParam("id")final Long id ){  
46         final Condutor condutor = this.condutorRepositorio  
            .findById(id).orElse(null);  
47         return condutor == null ? ResponseEntity.  
            badRequest().body("nenhum valor encontrado") :  
48             ResponseEntity.ok(condutor);  
49     }  
50  
51     @GetMapping("/listaativo")  
52     public ResponseEntity <?> listaCondutorAtivo (){  
53         List<Condutor> condutor = this.  
            condutorRepositorio.findAll();  
54         List <Condutor> condutorAtivo= new ArrayList();  
55         for (Condutor valor: condutor  
56 ) {  
57             if (valor.isAtivo())  
58             {  
59                 condutorAtivo.add(valor);  
60             }  
61         }  
62         return ResponseEntity.ok(condutorAtivo) ;  
63     }  
64  
65  
66     @GetMapping("/lista")  
67     public ResponseEntity <?> listaCompletaCondutor (){  
68         return ResponseEntity.ok(this.condutorRepositorio.  
            findAll());  
69     }  
70  
71     @PostMapping  
72     public ResponseEntity <?> cadastrar (@RequestBody  
        final Condutor condutor){  
73         this.condutorRepositorio.save(condutor);  
74         return ResponseEntity.ok("Registro cadastrado com  
            sucesso");
```

```
75      }
76
77      @PutMapping("/{id}")
78      public ResponseEntity <?> editar(
79          @PathVariable("id") Long id,
80          @RequestBody final Condutor condutor
81      ){
82          try {
83              final Condutor condutorBanco = this.
84                  condutorRepositorio.findById(id).orElse(null);
85              if (condutorBanco == null || !condutorBanco.
86                  getId().equals(condutor.getId())){
87                  throw new RuntimeException("Não foi possível
88                      identificar o registro");
89              }
90              this.condutorRepositorio.save(condutor);
91              return ResponseEntity.ok("Registro Atualizado");
92          }
93          catch (DataIntegrityViolationException e){
94              return ResponseEntity.internalServerError().
95                  body("Error" + e.getCause().getCause().getMessage());
96          }
97
98
99
100     @DeleteMapping ("/{id}")
101     public ResponseEntity <?> inativar(@PathVariable("id"
102 ) Long id){
103         try {
104             final Condutor condutorBanco = this.
105                 condutorRepositorio.findById(id).orElse(null);
106             if (condutorBanco == null ){
107                 throw new RuntimeException("Não foi
108                     possível identificar o registro");
109             }
110             this.condutorRepositorio.delete(condutorBanco
111 );
112             return ResponseEntity.ok("Registro Deletado"
113 );
```

```
109         }
110         catch (DataIntegrityViolationException e){
111             return ResponseEntity.internalServerError().
112             body("Error"+e + e.getCause().getCause().getMessage());
113         }
114         catch (Exception e){
115             final Condutor condutorBanco = this.
116             condutorRepositorio.findById(id).orElse(null);
117             if (condutorBanco == null){
118                 throw new RuntimeException("Não foi
119                 possível identificar o registro");
120             }
121             condutorBanco.setAtivo(false);
122             this.condutorRepositorio.save(condutorBanco);
123             return ResponseEntity.ok("Registro Inativado"
124 );
125     }
126
127     /* @DeleteMapping ("/{id}")
128
129     public ResponseEntity <?> inativar(
130         @PathVariable("id") Long id
131     ){
132         try {
133             final Condutor condutorBanco = this.
134             condutorRepositorio.findById(id).orElse(null);
135             if (condutorBanco == null){
136                 throw new RuntimeException("Não foi
137                 possível identificar o registro");
138             }
139             condutorBanco.setAtivo(false);
140             this.condutorRepositorio.save(condutorBanco);
141             return ResponseEntity.ok("Registro deletado
142 ");
143         }
144         catch (DataIntegrityViolationException e){
145             return ResponseEntity.internalServerError().
146             body("Error" + e.getCause().getCause().getMessage());
147         }
148     }
```

```
145         catch (RuntimeException e){
146             return ResponseEntity.internalServerError().
147                 body("Error");
148         }
149     }
150 */
151
152
153 }
154
```

```
1 package br.com.uniamerica.estacionamento.controller;
2
3 import br.com.uniamerica.estacionamento.entity.
4 Configuracao;
5 import br.com.uniamerica.estacionamento.repository.
6 ConfiguracaoRepository;
7 import org.springframework.dao.
8 DataIntegrityViolationException;
9 import org.springframework.http.ResponseEntity;
10 import org.springframework.stereotype.Controller;
11 import org.springframework.web.bind.annotation.*;
12
13 @Controller
14 @RequestMapping(value = "/api/configuracao")
15 public class ConfiguracaoController {
16
17     private ConfiguracaoRepository
18     configuracaoRepository;
19     public ConfiguracaoController(ConfiguracaoController
20     configuracaoController){
21         this.configuracaoRepository =
22         configuracaoRepository;
23     }
24
25     @GetMapping
26     public ResponseEntity <?> buscaid(@RequestParam("id")
27     final Long id){
28         final Configuracao configuracao = this.
29         configuracaoRepository.findById(id).orElse(null);
30
31         return configuracao == null ? ResponseEntity.
32             badRequest().body("Não localizado") :
33             ResponseEntity.ok(configuracao);
34     }
35
36     @PostMapping
37     public ResponseEntity<?> cadastracurso(@
38     RequestBody final Configuracao configuracao){
39         configuracaoRepository.save(configuracao);
40         return ResponseEntity.ok("Configuracao ");
41     }
42
43     @PutMapping("/{id}")
44     public ResponseEntity <?> editaConfiguracao(@
```

```
34 PathVariable final Long id, @RequestBody Configuracao
  configuracao){
35     try{
36         Configuracao valorBanco = this.
  configuracaoRepositorio.findById(id).orElse(null);
37         if (valorBanco == null || !valorBanco.getId().
  equals(configuracao.getId())){
38             throw new RuntimeException("Não foi
  possível localizar.");
39         }
40         this.configuracaoRepositorio.save(configuracao
  );
41         return ResponseEntity.ok("Configuracao
  cadastrada com sucesso.");
42     } catch (DataIntegrityViolationException e) {
43         return ResponseEntity.internalServerError().
  body("Error" + e.getCause().getCause().getMessage());
44     } catch (RuntimeException e) {
45         return ResponseEntity.internalServerError().
  body("Error");
46     }
47 }
48
49 }
50
```

```
1 package br.com.uniamerica.estacionamento.controller;
2
3 import br.com.uniamerica.estacionamento.entity.Marca;
4 import br.com.uniamerica.estacionamento.entity.
5     Movimentacao;
6 import br.com.uniamerica.estacionamento.repository.
7     CondutorRepository;
8 import br.com.uniamerica.estacionamento.repository.
9     MarcaRepository;
10 import br.com.uniamerica.estacionamento.repository.
11     MovimentacaoRepository;
12 import org.apache.catalina.connector.Response;
13 import org.springframework.dao.
14     DataIntegrityViolationException;
15 import org.springframework.http.ResponseEntity;
16 import org.springframework.stereotype.Controller;
17 import org.springframework.web.bind.annotation.*;
18
19
20 public class MovimentacaoController {
21
22     private MovimentacaoRepository
23         movimentacaoRepository;
24     public MovimentacaoController (MovimentacaoRepository
25         movimentacaoRepository){
26         this.movimentacaoRepository =
27             movimentacaoRepository;
28     }
29
30     @GetMapping("/{id}")
31     public ResponseEntity<?> buscaMovimentacao(@
32         RequestParam("id") final Long id){
33         final Movimentacao movimentacao = this.
34             movimentacaoRepository.findById(id).orElse(null);
35         return movimentacao == null ?
36             ResponseEntity.badRequest().body("Valor
37             não encontrado") :
38             ResponseEntity.ok(movimentacao);
39     }
40 }
```

```
34     @GetMapping("/lista")
35     public ResponseEntity <?> buscaLista(){
36         return ResponseEntity.ok(this.
37             movimentacaoRepository.findAll());
38     }
39
39     @GetMapping("/listaAtiva")
40     public ResponseEntity<?> listaAtivo(){
41         List<Movimentacao> listaMovimentacao = this.
42             movimentacaoRepository.findAll();
42         List <Movimentacao> checaLista = new ArrayList
42 <>();
43         for (Movimentacao valor: listaMovimentacao
44             ) {
45             if(valor.isAtivo()){
46                 checaLista.add(valor);
47             }
48         }
49         return ResponseEntity.ok(checaLista) ;
50     }
51
52     @GetMapping("/listaAtivaSemSaida")
53     public ResponseEntity<?> listaAtivoSemSaida() {
54         List<Movimentacao> listaMovimentacao = this.
54             movimentacaoRepository.findAll();
55         List<Movimentacao> checaLista = new ArrayList<>();
56         for (Movimentacao valor : listaMovimentacao
57             ) {
58             if (valor.getSaida().equals(null) && valor.
59             isAtivo()) {
60                 checaLista.add(valor);
61             }
62         }
63         return ResponseEntity.ok(listaMovimentacao);
64     }
65
66
67     @PostMapping
68     public ResponseEntity<?> cadastramovimentacao (@
69     RequestBody final Movimentacao movimentacao){
70         this.movimentacaoRepository.save(movimentacao);
71         return ResponseEntity.ok("Carro estacionado");
71     }
```

```
72
73     @PutMapping("/{id}")
74     public ResponseEntity<?> atualizaMovimentacao(@
75         PathVariable("id") final Long id, @RequestBody final
76         Movimentacao movimentacao) {
77         try {
78             final Movimentacao movimentacaoBanco = this.
79             movimentacaoRepositorio.findById(id).orElse(null);
80             if (movimentacaoBanco == null || !
81                 movimentacaoBanco.getId().equals(movimentacao.getId())) {
82                 throw new RuntimeException("Não foi
83                     possível localizar");
84             }
85             this.movimentacaoRepositorio.save(
86             movimentacao);
87             return ResponseEntity.ok("Movimentacao
88                 atualizada!");
89         } catch (DataIntegrityViolationException e) {
90             return ResponseEntity.internalServerError().
91                 body("Error" + e.getCause().getCause().getMessage());
92         } catch (RuntimeException e) {
93             return ResponseEntity.internalServerError().
94                 body("Error");
95         }
96     }
97
98     @DeleteMapping ("/{id}")
99     public ResponseEntity <?> inativar(@PathVariable("id"
100 ) Long id){
101         try {
102             final Movimentacao valorBanco = this.
103             movimentacaoRepositorio.findById(id).orElse(null);
104             if (valorBanco == null ){
105                 throw new RuntimeException("Não foi
106                     possível identificar o registro");
107             }
108             valorBanco.setAtivo(false);
109             this.movimentacaoRepositorio.delete(
110             valorBanco);
111             return ResponseEntity.ok("Registro Deletado"
112 );
113         } catch (DataIntegrityViolationException e){
114             return ResponseEntity.internalServerError().
115                 body("Error");
116         }
117     }
118 }
```

```
101 body("Error"+e + e.getCause().getCause().getMessage());
102     }
103     catch (Exception e){
104         final Movimentacao valorBanco = this.
105             movimentacaoRepositorio.findById(id).orElse(null);
106         if (valorBanco == null ){
107             throw new RuntimeException("Não foi
108             possível identificar o registro");
109             valorBanco.setAtivo(false);
110             this.movimentacaoRepositorio.save(valorBanco
111 );
112         }
113     }
114 }
115
116
117
118 }
119
```