SISTEMA DE GESTÃO DE GRAVADORA DE MÚSICA

Gustavo Mendes Santos
Heitor Guimarães da Fonseca Filho
Igor Augusto Reis Gomes

INTRODUÇÃO

- Problema
- Contextualização

ETAPA 1

Modelagem estrutural e dinâmica
em alto nível de abstração

DIAGRAMA DE CLASSES

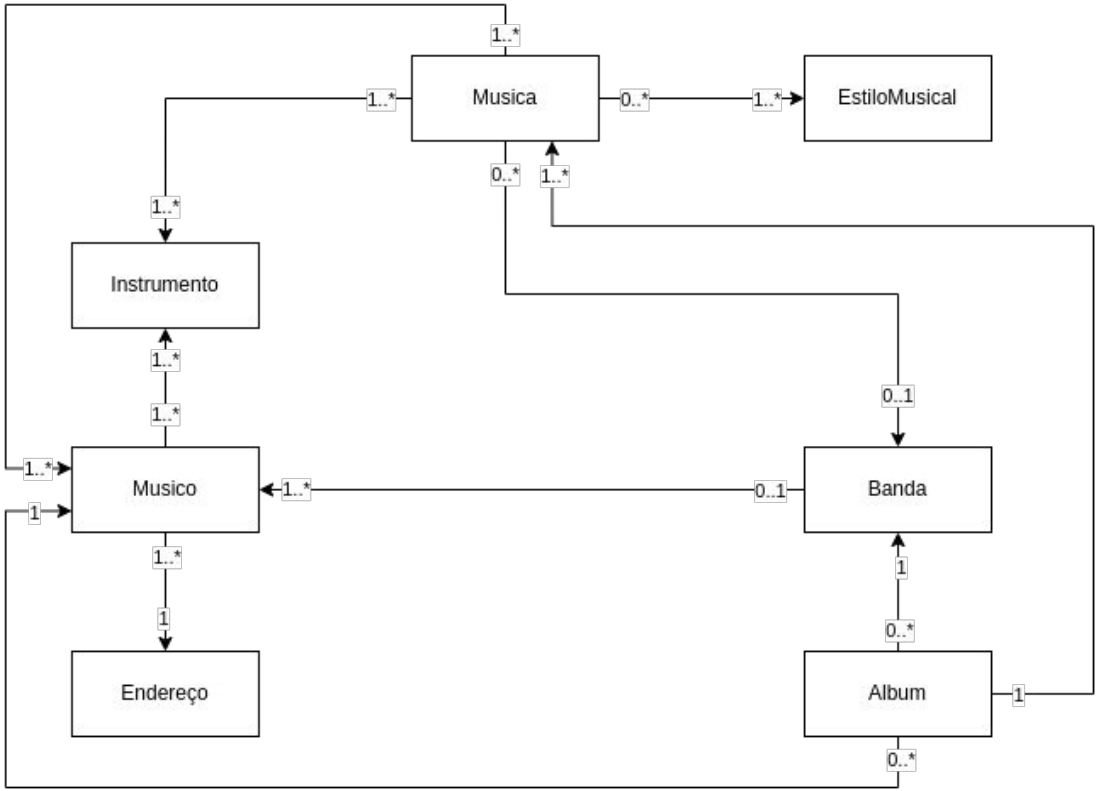
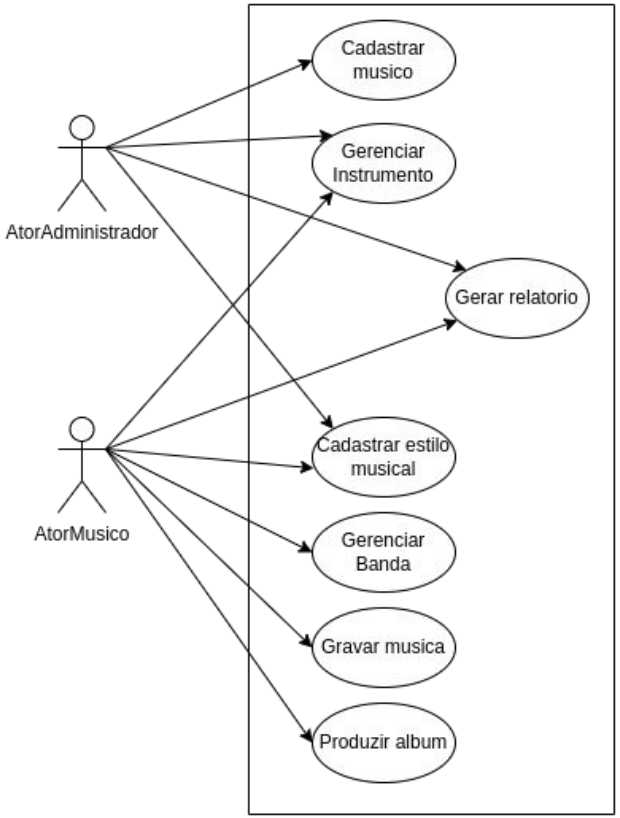


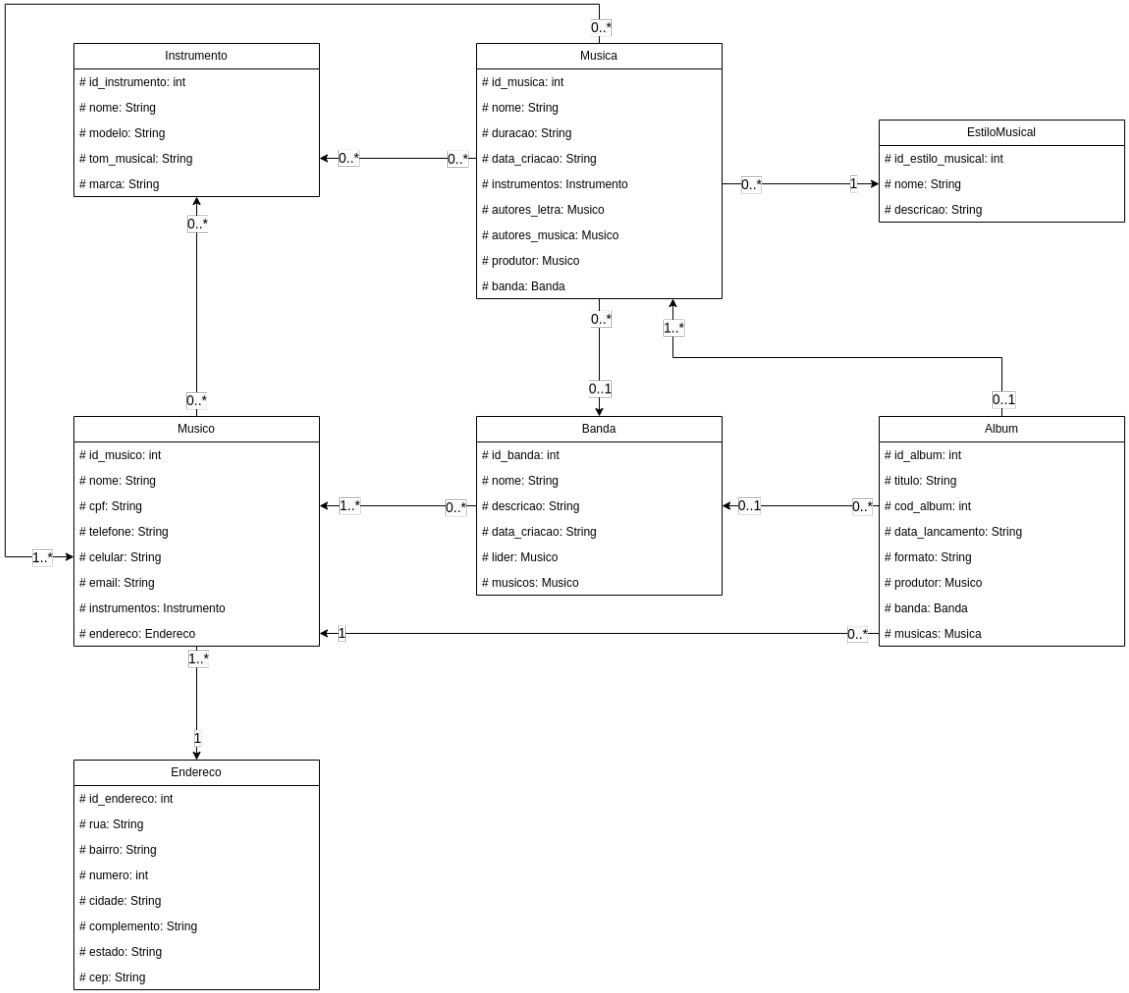
DIAGRAMA DE CASOS DE USO



ETAPA 2

Refinamento estrutural

DIAGRAMA DE CLASSES (COM ATRIBUTOS)



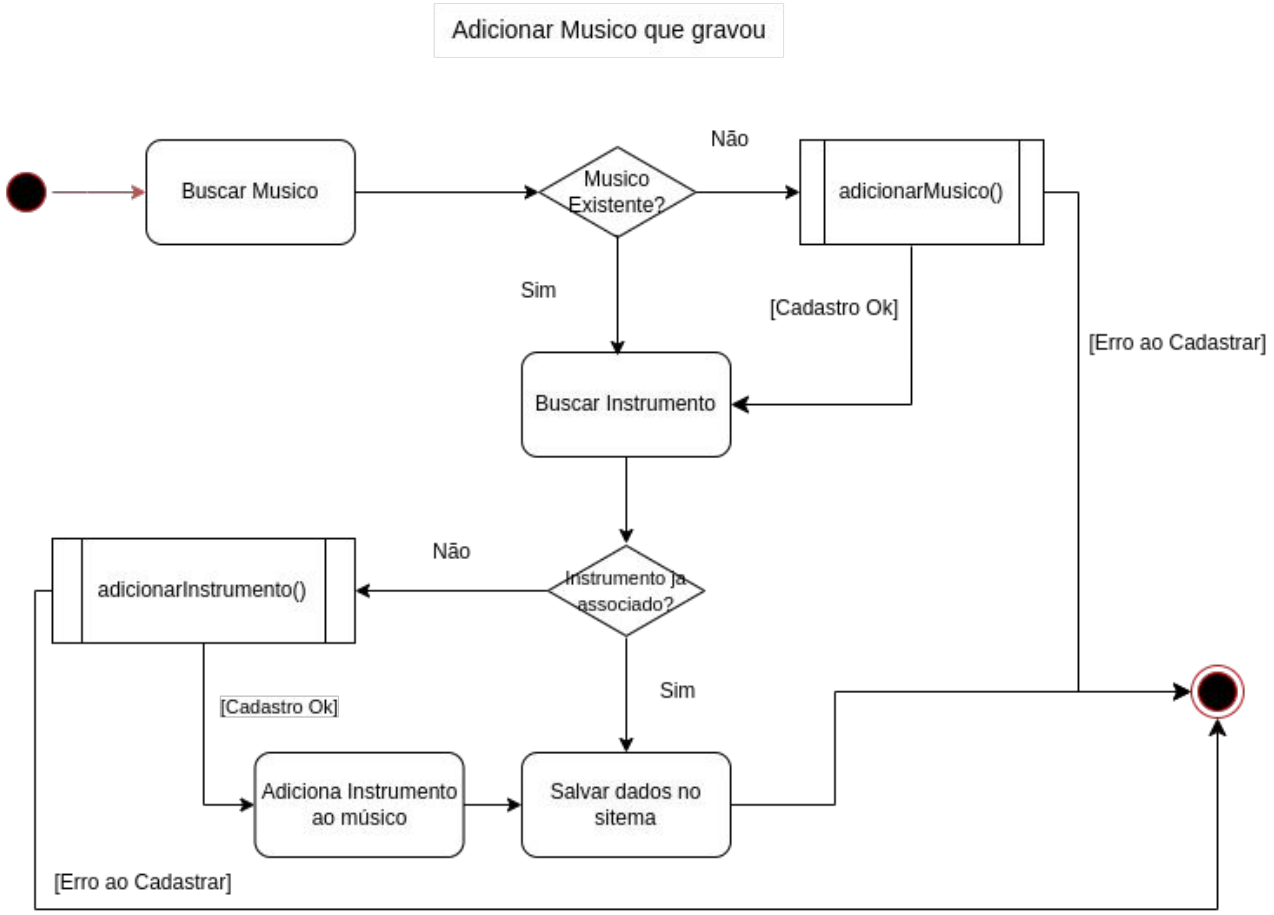
ETAPA 3

Refinamento de casos de uso

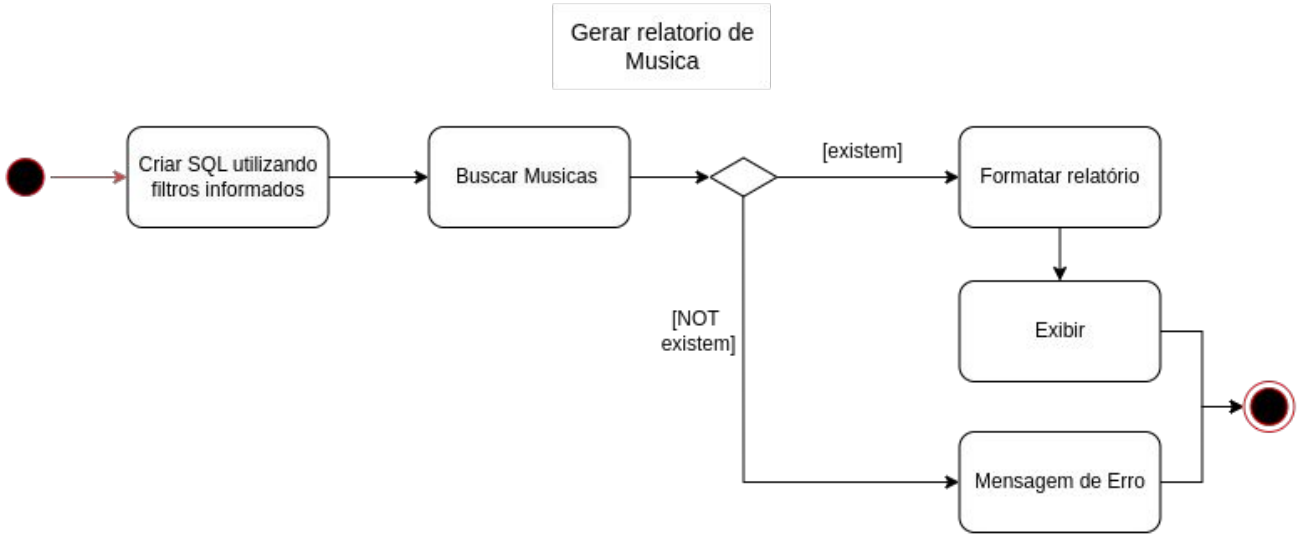
DIAGRAMAS DE ATIVIDADES

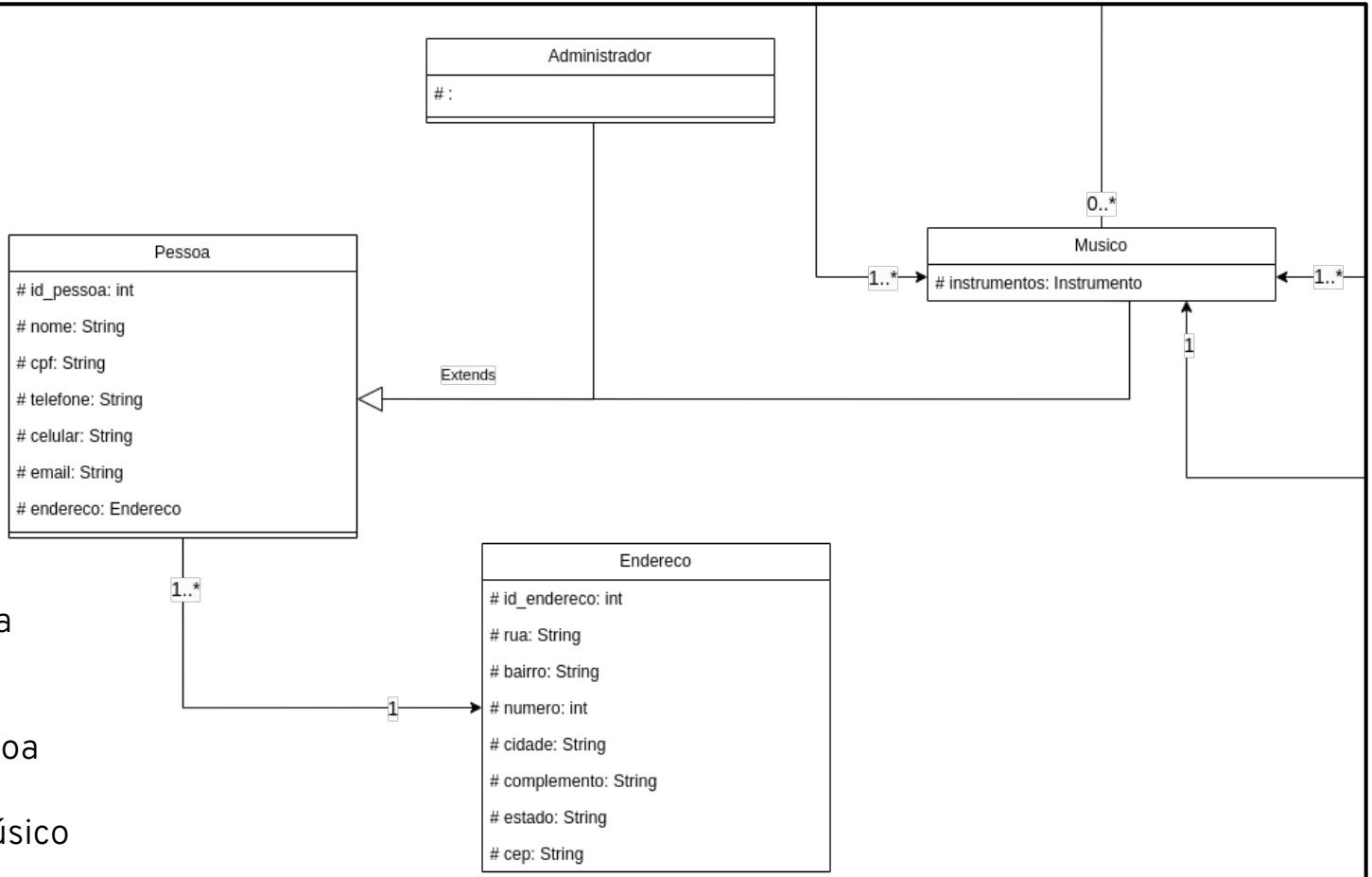
MUSICA

- Necessário saber qual músico tocou qual instrumento
- Um músico pode estar com vários instrumentos mas só utilizar um para gravar determinada música



- Gerar relatório é um caso de uso
- Será passado um filtro(consulta SQL) que vai fazer a busca
- Altera o caso de Uso do gerarRelatorio

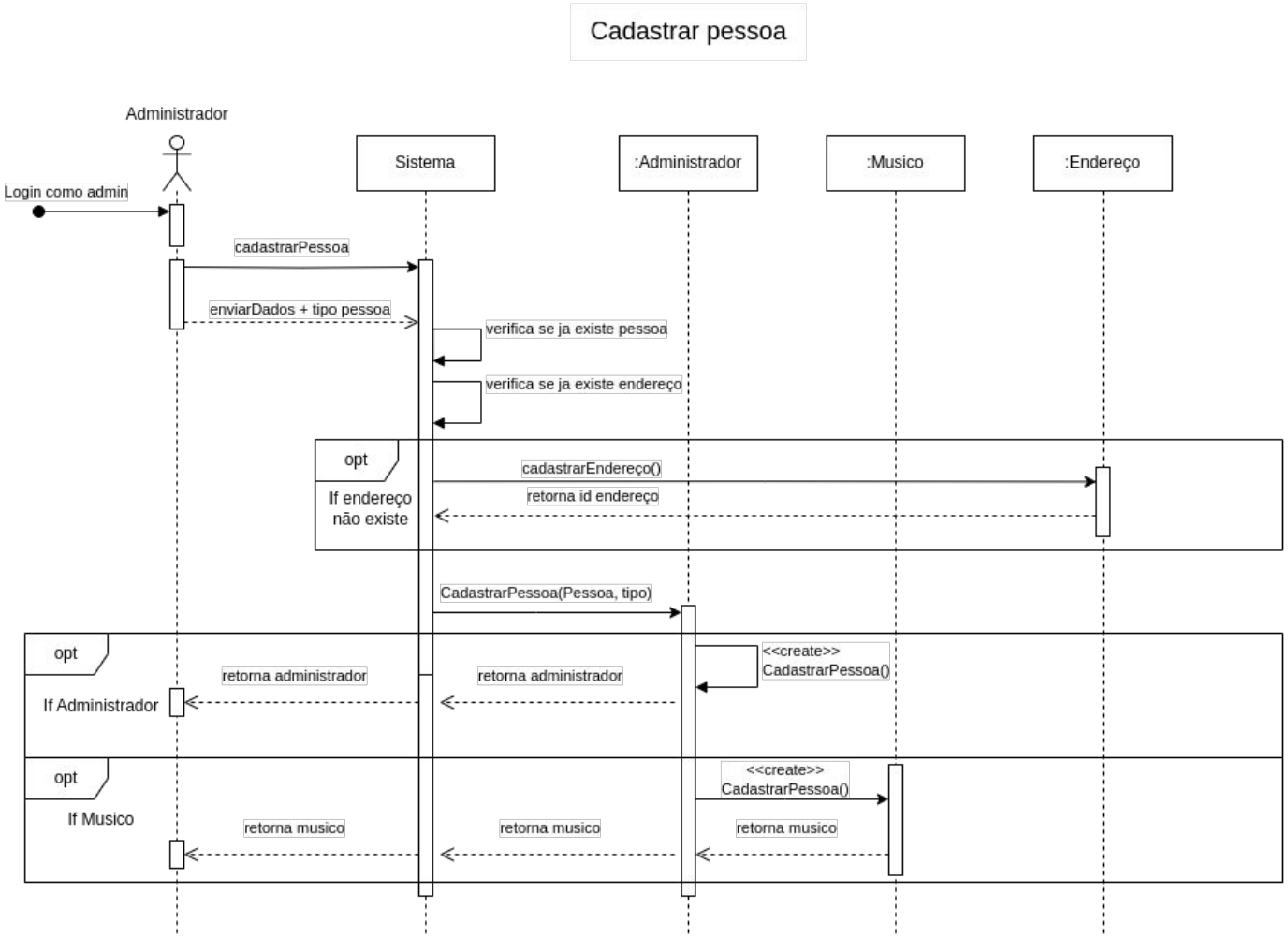




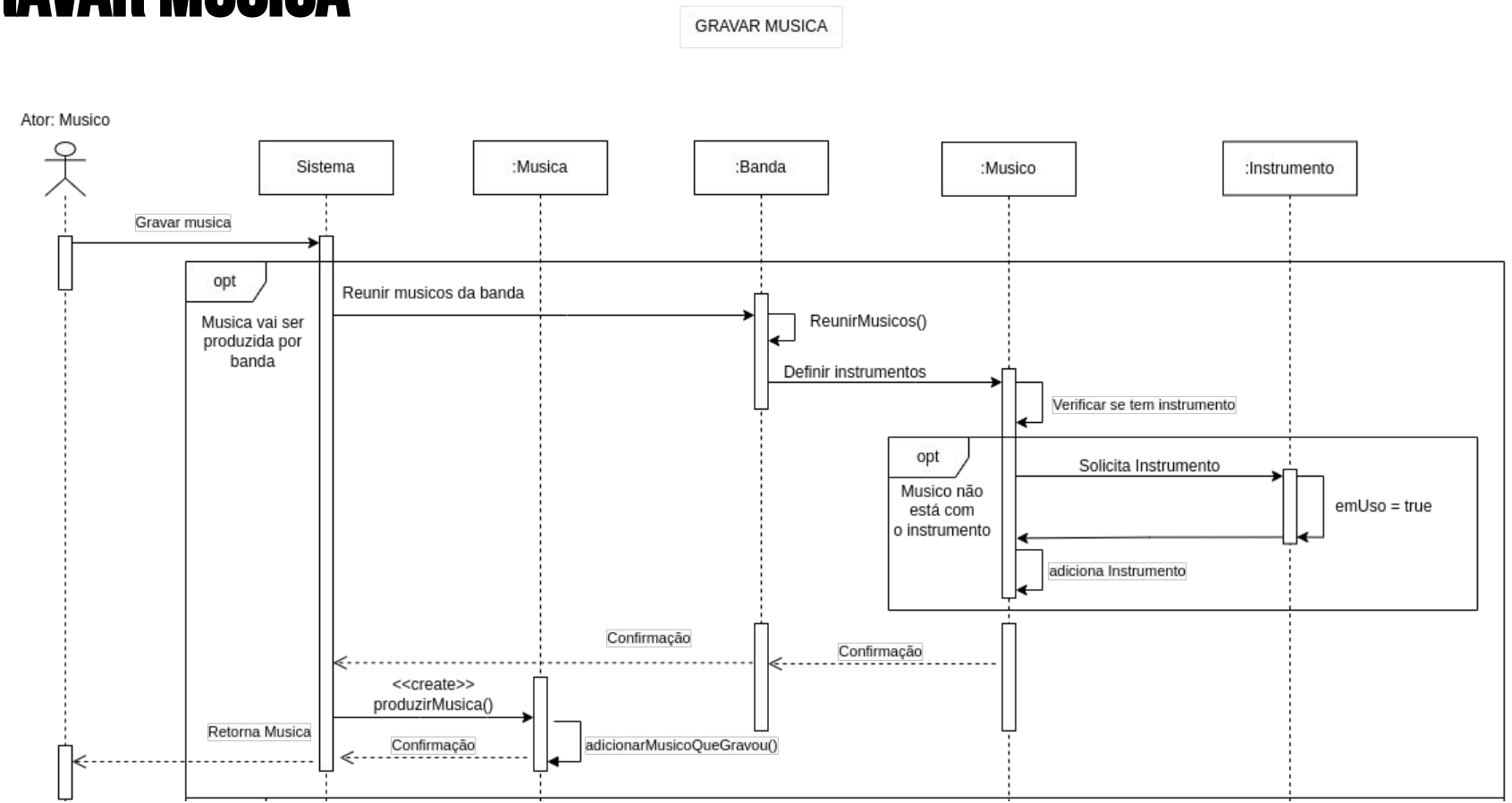
- Nova classe: Pessoa
- Endereço agora se relaciona com Pessoa
- Administrador e Músico herdam Pessoa

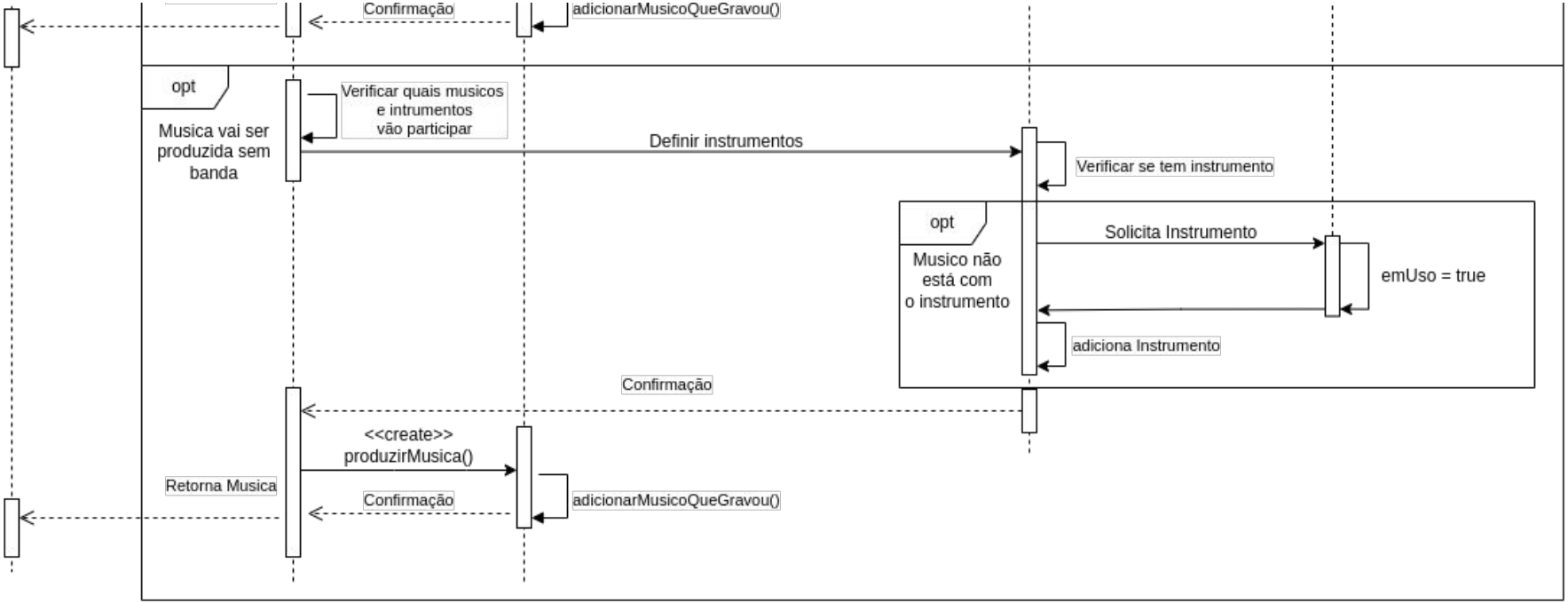
DIAGRAMAS DE SEQUÊNCIA

CADASTRAR PESSOA

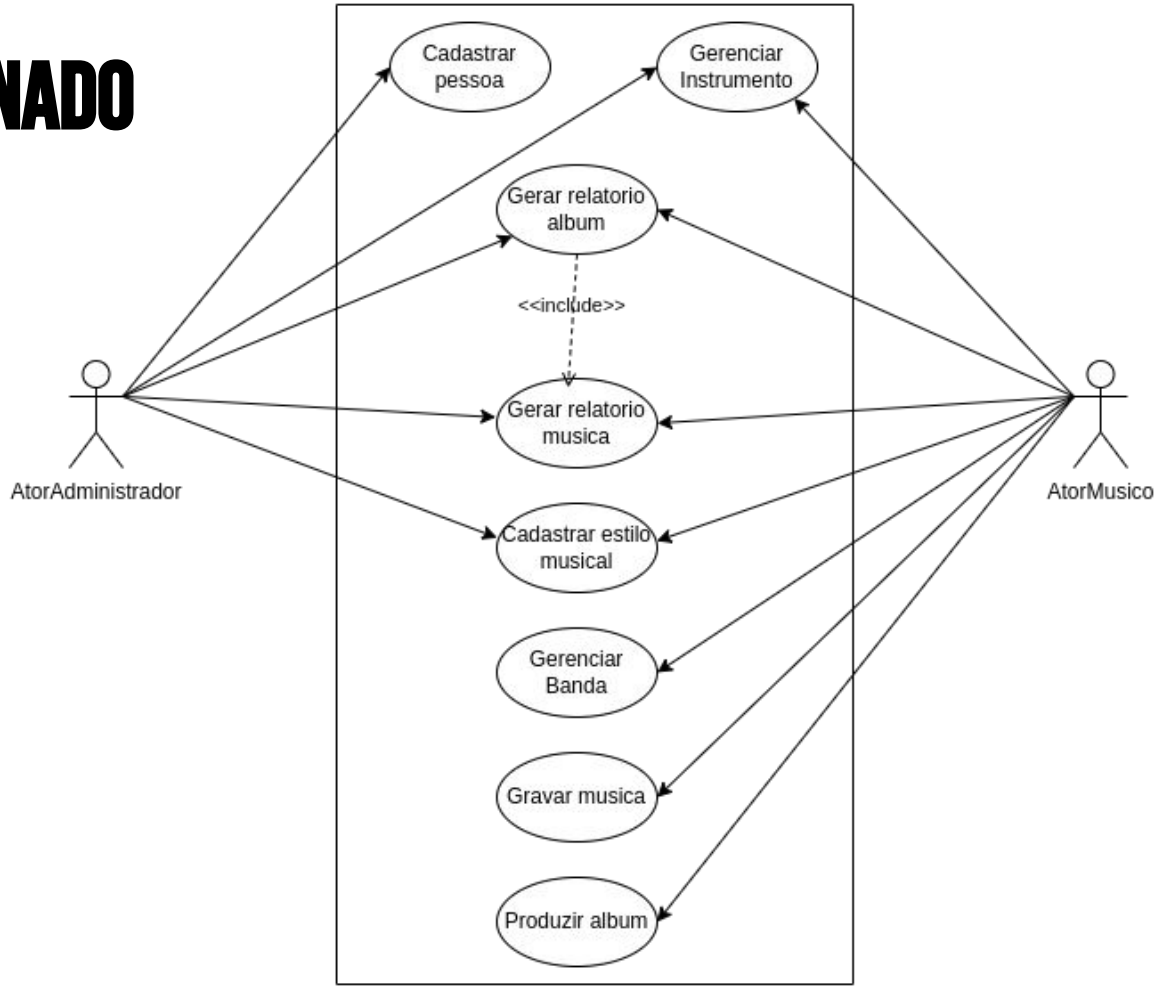


GRAVAR MÚSICA




Confirmação
adicionarMusicoQueGravou()
opt
Musica vai ser produzida sem banda
Verificar quais musicos e intrumentos vão participar
Definir instrumentos
Verificar se tem instrumento
opt
Musico não está com o instrumento
Solicita Instrumento
emUso = true
adiciona Instrumento
Confirmação
<<create>>
produzirMusica()
Retorna Musica
Confirmação
adicionarMusicoQueGravou()

CASO DE USO REFINADO



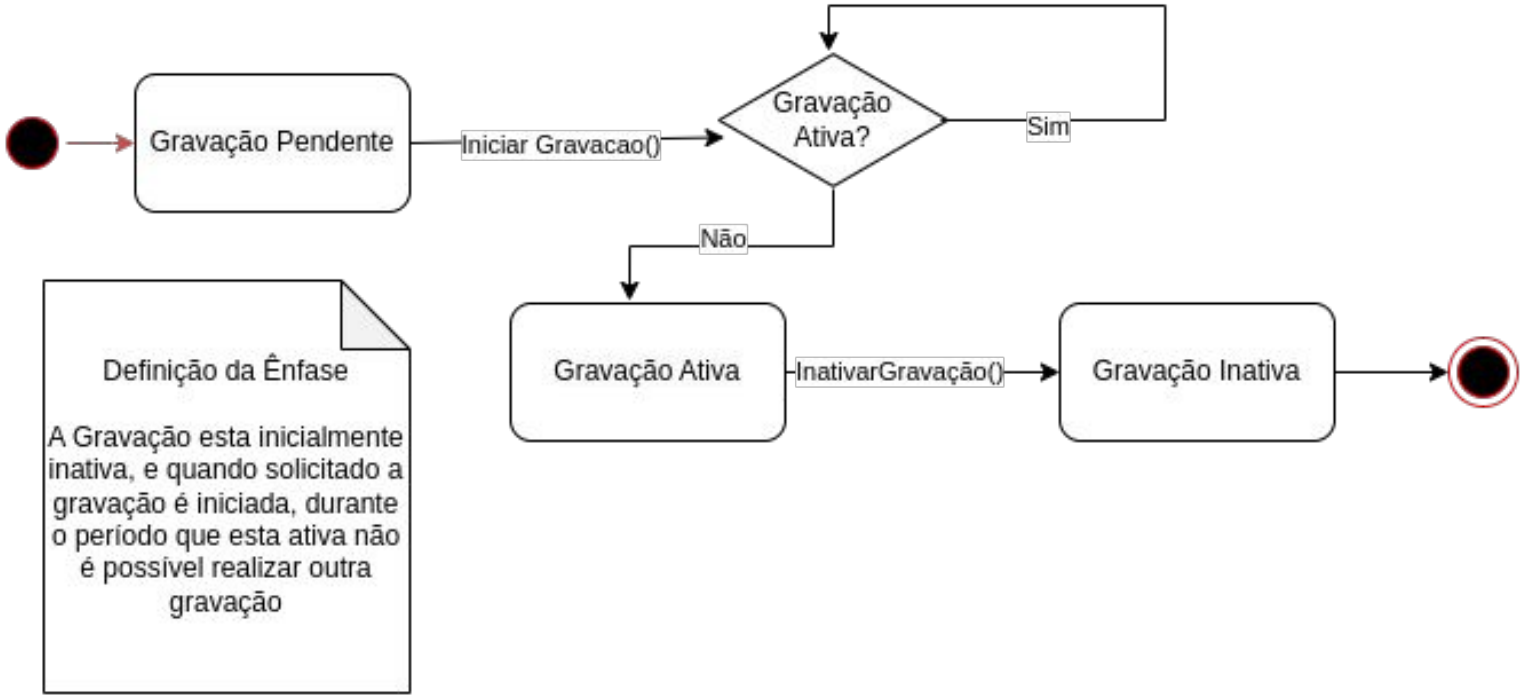
ETAPA 4

Modelagem de estados associada à classe

DIAGRAMAS DE MÁQUINA DE ESTADOS

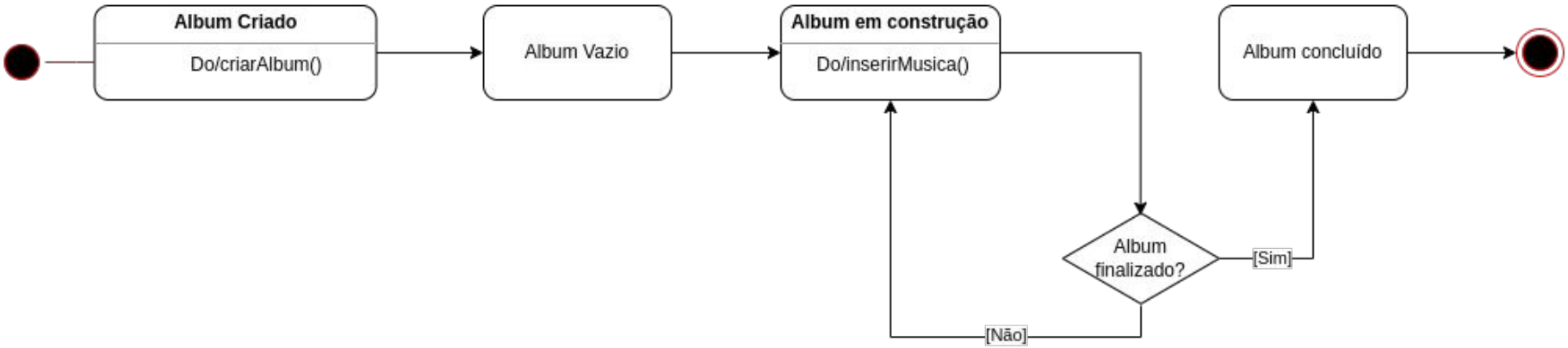
GRAVAÇÃO MÚSICA

Diagrama Maquina de Estado Gravação Musica

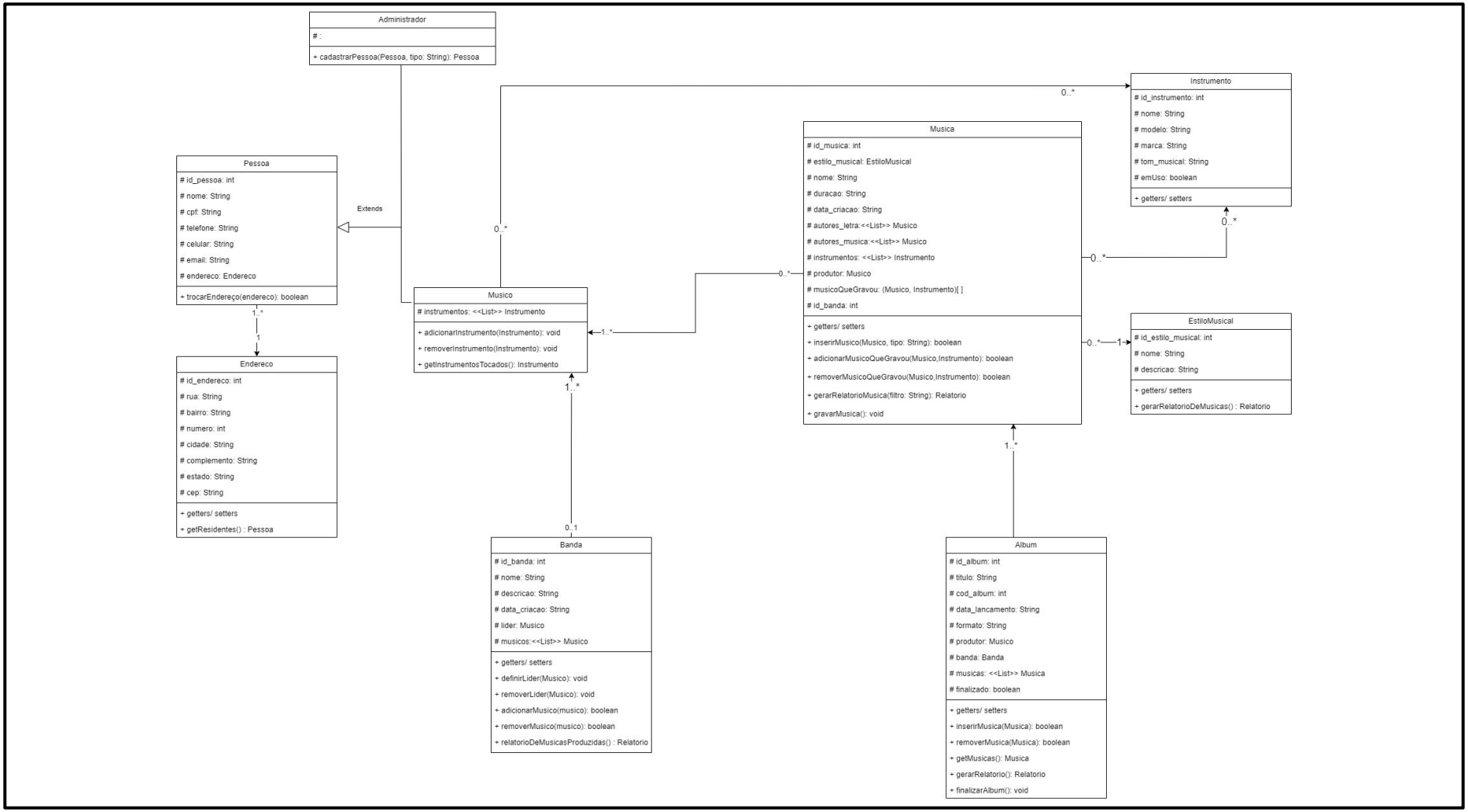


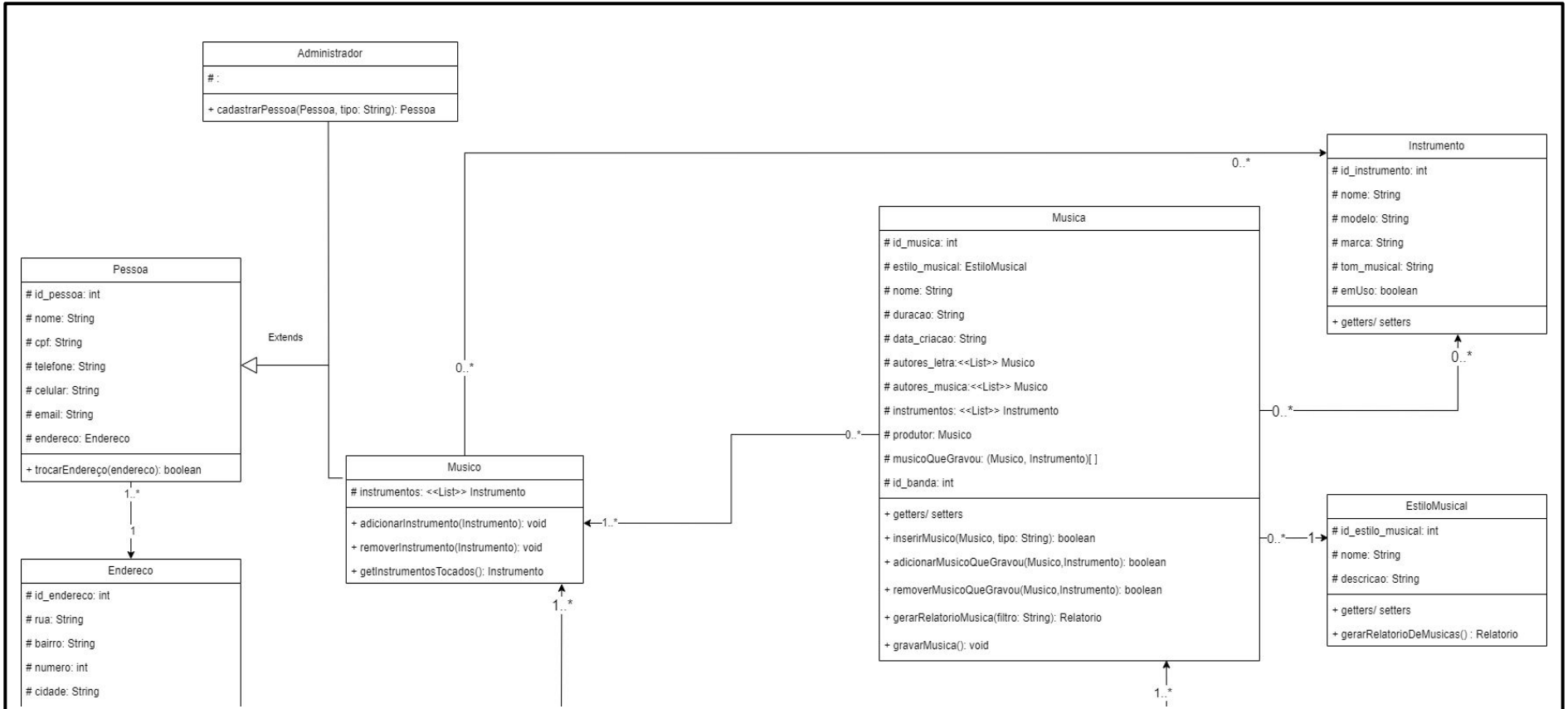
ÁLBUM

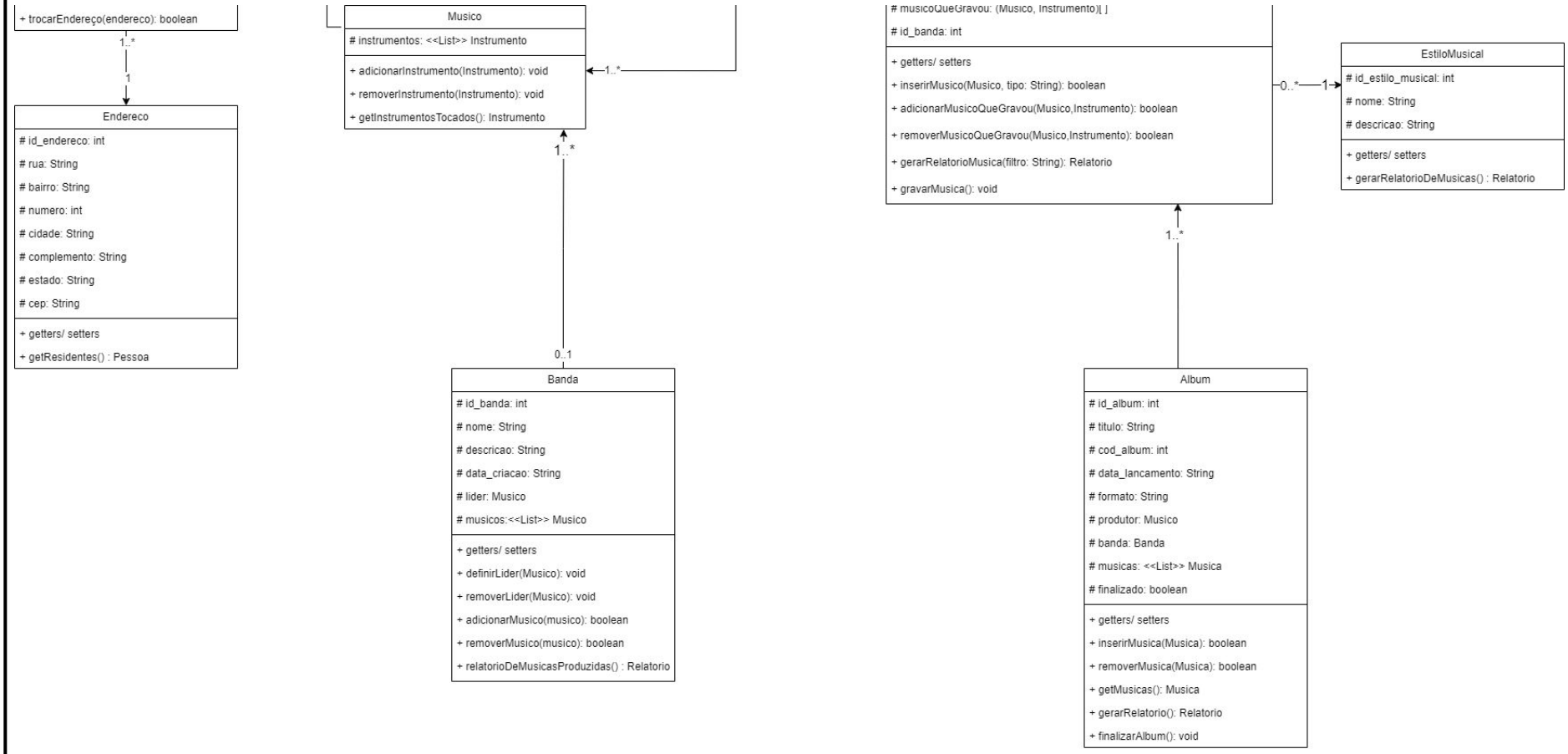
Diagrama Maquina de Estado Album



DIAGRAMAS DE CLASSES COM MÉTODOS


Administrador
:
+ cadastrarPessoa(Pessoa, tipo: String): Pessoa
Instrumento
id_instrumento: int
nome: String
modelo: String
marca: String
tom_musical: String
emUso: boolean
+ getters/ setters
Pessoa
id_pessoa: int
nome: String
cpf: String
telefone: String
celular: String
email: String
endereco: Endereco
+ trocarEndereço(endereco): boolean
Extends
Musica
id_musica: int
estilo_musical: EstiloMusical
nome: String
duracao: String
data_criacao: String
autores_letra:<<List>> Musico
autores_musica:<<List>> Musico
instrumentos: <<List>> Instrumento
produtor: Musico
musicoQueGravou: (Musico, Instrumento)[ ]
id_banda: int
+ getters/ setters
+ inserirMusico(Musico, tipo: String): boolean
+ adicionarMusicoQueGravou(Musico,Instrumento): boolean
+ removerMusicoQueGravou(Musico,Instrumento): boolean
+ gerarRelatorioMusica(filtro: String): Relatorio
+ gravarMusica(): void
Musico
instrumentos: <<List>> Instrumento
+ adicionarInstrumento(Instrumento): void
+ removerInstrumento(Instrumento): void
+ getInstrumentosTocados(): Instrumento
EstiloMusical
id_estilo_musical: int
nome: String
descricao: String
+ getters/ setters
+ gerarRelatorioDeMusicas() : Relatorio
Endereco
id_endereco: int
rua: String
bairro: String
numero: int
cidade: String
complemento: String
estado: String
cep: String
+ getters/ setters
+ getResidentes() : Pessoa
Banda
id_banda: int
nome: String
descricao: String
data_criacao: String
lider: Musico
musicos:<<List>> Musico
+ getters/ setters
+ definirLider(Musico): void
+ removerLider(Musico): void
+ adicionarMusico(musico): boolean
+ removerMusico(musico): boolean
+ relatorioDeMusicasProduzidas() : Relatorio
Album
id_album: int
titulo: String
cod_album: int
data_lancamento: String
formato: String
produtor: Musico
banda: Banda
musicas: <<List>> Musica
finalizado: boolean
+ getters/ setters
+ inserirMusica(Musica): boolean
+ removerMusica(Musica): boolean
+ getMusicas(): Musica
+ gerarRelatorio(): Relatorio
+ finalizarAlbum(): void
0..*
0..*
0..*
0..*
1..*
1
1..*
1..*
0..*
1
1..*
0..1


Administrador
:
+ cadastrarPessoa(Pessoa, tipo: String): Pessoa
Instrumento
id_instrumento: int
nome: String
modelo: String
marca: String
tom_musical: String
emUso: boolean
+ getters/ setters
Musica
id_musica: int
estilo_musical: EstiloMusical
nome: String
duracao: String
data_criacao: String
autores_letra:<<List>> Musico
autores_musica:<<List>> Musico
instrumentos: <<List>> Instrumento
produtor: Musico
musicoQueGravou: (Musico, Instrumento)[ ]
id_banda: int
+ getters/ setters
+ inserirMusico(Musico, tipo: String): boolean
+ adicionarMusicoQueGravou(Musico,Instrumento): boolean
+ removerMusicoQueGravou(Musico,Instrumento): boolean
+ gerarRelatorioMusica(filtro: String): Relatorio
+ gravarMusica(): void
Pessoa
id_pessoa: int
nome: String
cpf: String
telefone: String
celular: String
email: String
endereco: Endereco
+ trocarEndereço(endereco): boolean
Extends
Musico
instrumentos: <<List>> Instrumento
+ adicionarInstrumento(Instrumento): void
+ removerInstrumento(Instrumento): void
+ getInstrumentosTocados(): Instrumento
Endereco
id_endereco: int
rua: String
bairro: String
numero: int
cidade: String
EstiloMusical
id_estilo_musical: int
nome: String
descricao: String
+ getters/ setters
+ gerarRelatorioDeMusicas() : Relatorio
0..*
1..*
1
1..*
0..*
0..*
0..*
1
1..*
0..*
1..*

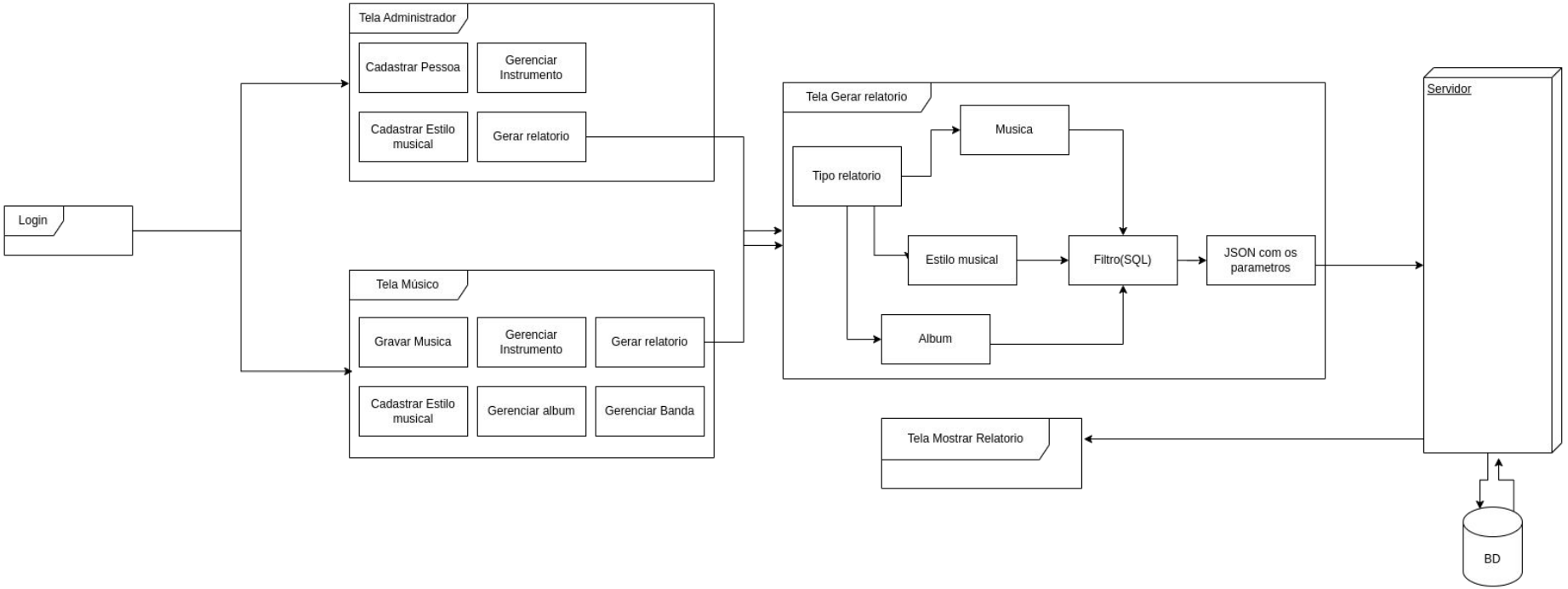

+ trocarEndereço(endereco): boolean
1..*
1
Endereco
id_endereco: int
rua: String
bairro: String
numero: int
cidade: String
complemento: String
estado: String
cep: String
+ getters/ setters
+ getResidentes() : Pessoa
Musico
instrumentos: <<List>> Instrumento
+ adicionarInstrumento(Instrumento): void
+ removerInstrumento(Instrumento): void
+ getInstrumentosTocados(): Instrumento
1..*
1..*
0..1
Banda
id_banda: int
nome: String
descricao: String
data_criacao: String
lider: Musico
musicos:<<List>> Musico
+ getters/ setters
+ definirLider(Musico): void
+ removerLider(Musico): void
+ adicionarMusico(musico): boolean
+ removerMusico(musico): boolean
+ relatorioDeMusicasProduzidas() : Relatorio
# musicoQueGravou: (Musico, Instrumento)[ ]
# id_banda: int
+ getters/ setters
+ inserirMusico(Musico, tipo: String): boolean
+ adicionarMusicoQueGravou(Musico,Instrumento): boolean
+ removerMusicoQueGravou(Musico,Instrumento): boolean
+ gerarRelatorioMusica(filtro: String): Relatorio
+ gravarMusica(): void
0..*
1
EstiloMusical
id_estilo_musical: int
nome: String
descricao: String
+ getters/ setters
+ gerarRelatorioDeMusicas() : Relatorio
1..*
Album
id_album: int
titulo: String
cod_album: int
data_lancamento: String
formato: String
produtor: Musico
banda: Banda
musicas: <<List>> Musica
finalizado: boolean
+ getters/ setters
+ inserirMusica(Musica): boolean
+ removerMusica(Musica): boolean
+ getMusicas(): Musica
+ gerarRelatorio(): Relatorio
+ finalizarAlbum(): void

ETAPA 6

Destaque de situações especiais na modelagem

FLUXO DE PÁGINAS

- Facilitar visualização da sequência de páginas



ETAPA 5

Introdução de elementos do
domínio da solução computacional

ELEMENTOS ESCOLHIDOS

APLICAÇÃO WEB: com utilização do framework Spring Boot 

LINGUAGEM: Java 

PERSISTÊNCIA DE DADOS: Java Persistence API (JPA) com banco de dados PostgreSQL (Supabase)

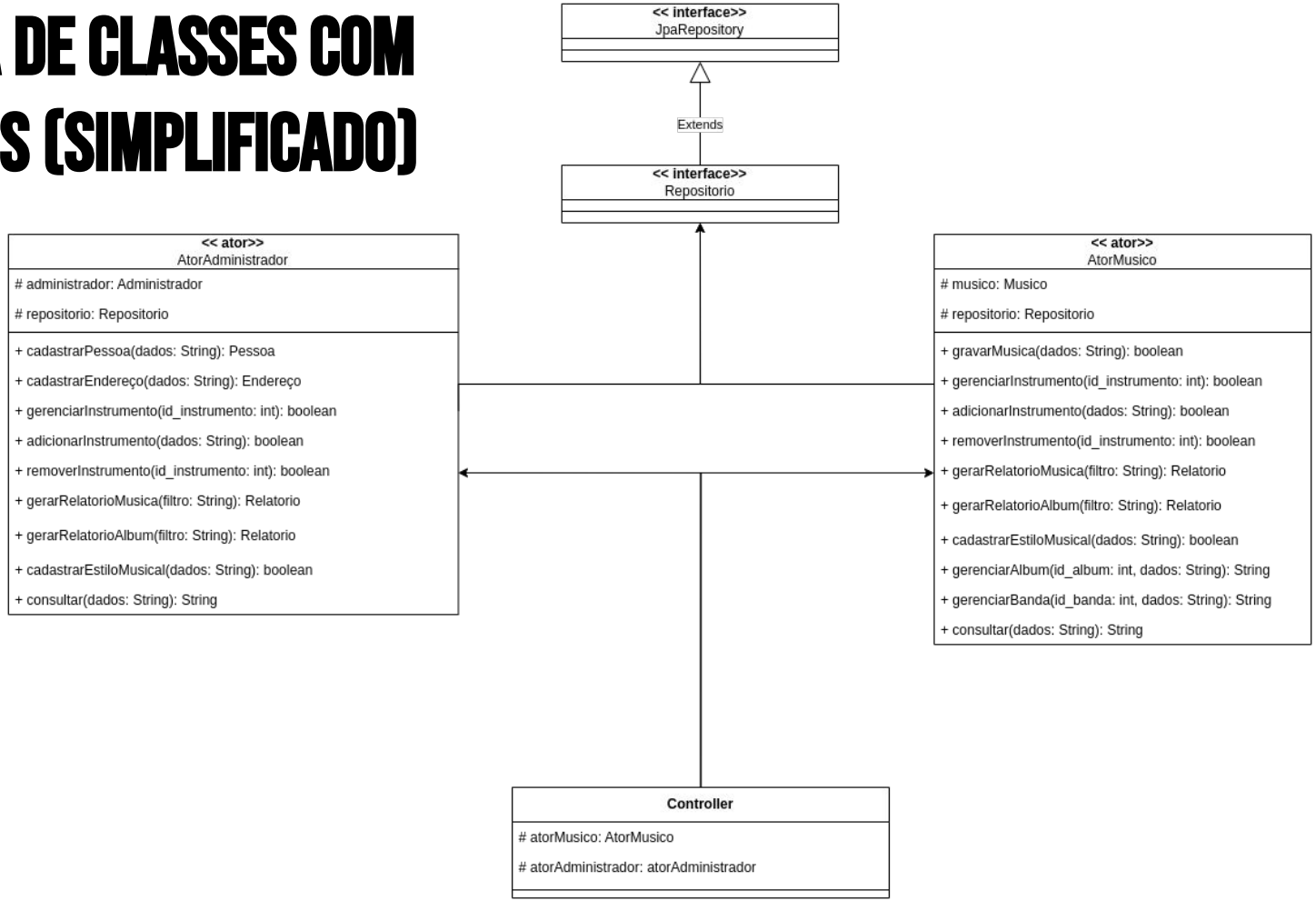
PADRÃO DE ARQUITETURA: Model-View-Controller (MVC) - separação clara de responsabilidades

LAYOUT: HTML (estruturação) 

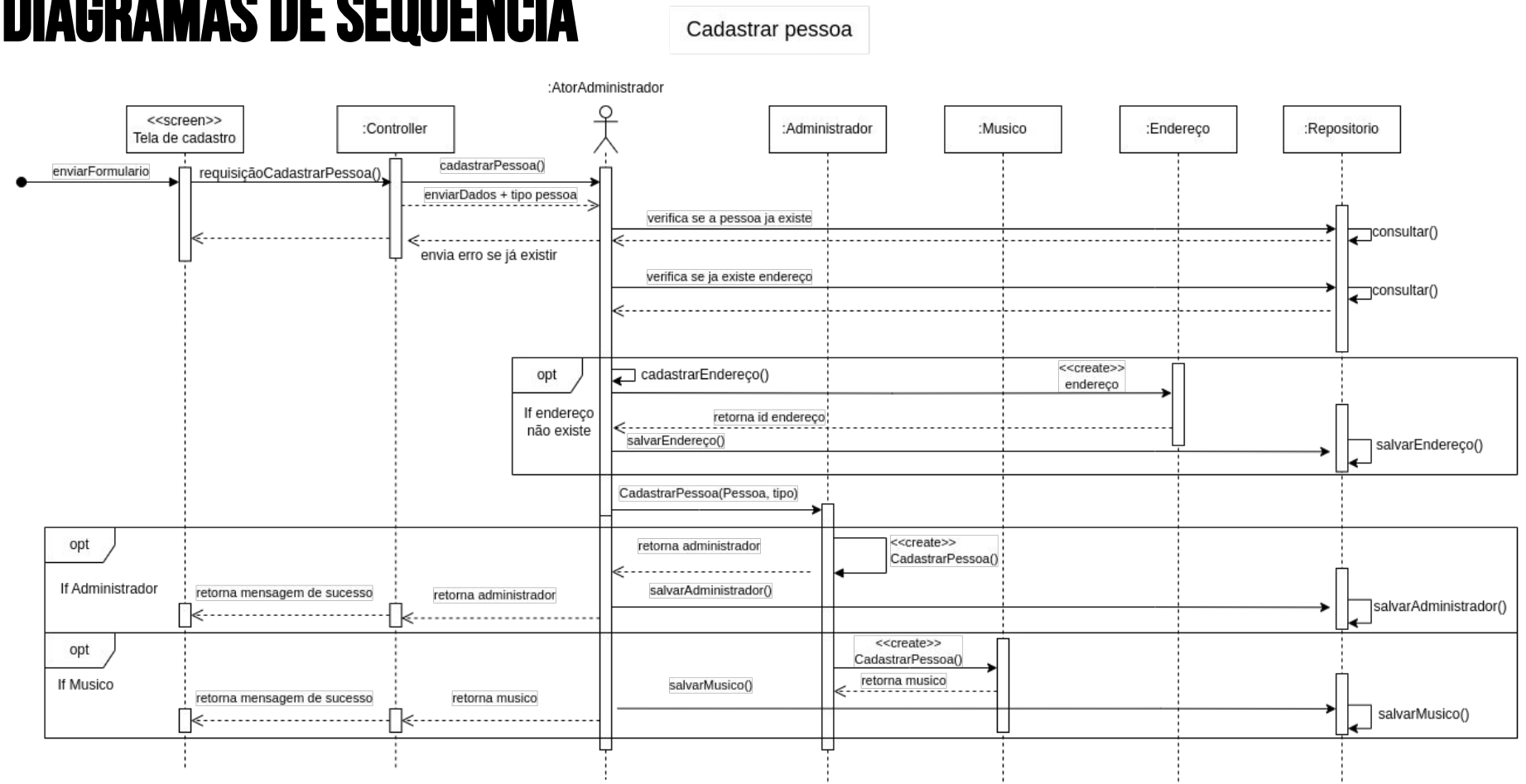
CSS (estilização) 

React (biblioteca) 

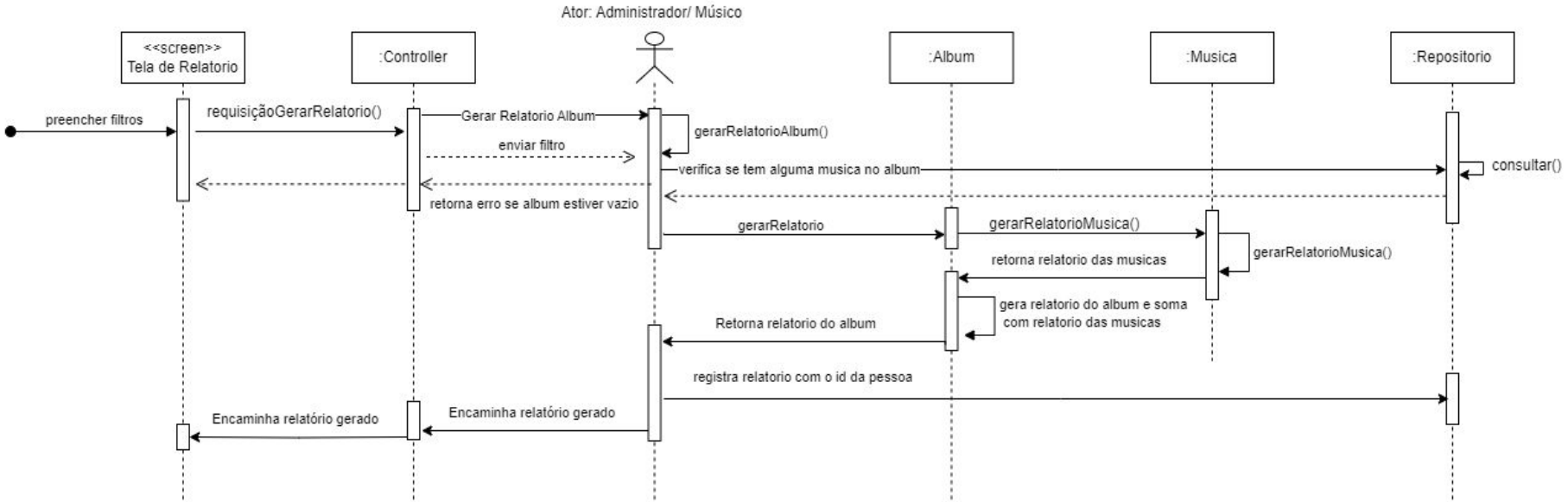
DIAGRAMA DE CLASSES COM ELEMENTOS (SIMPLIFICADO)



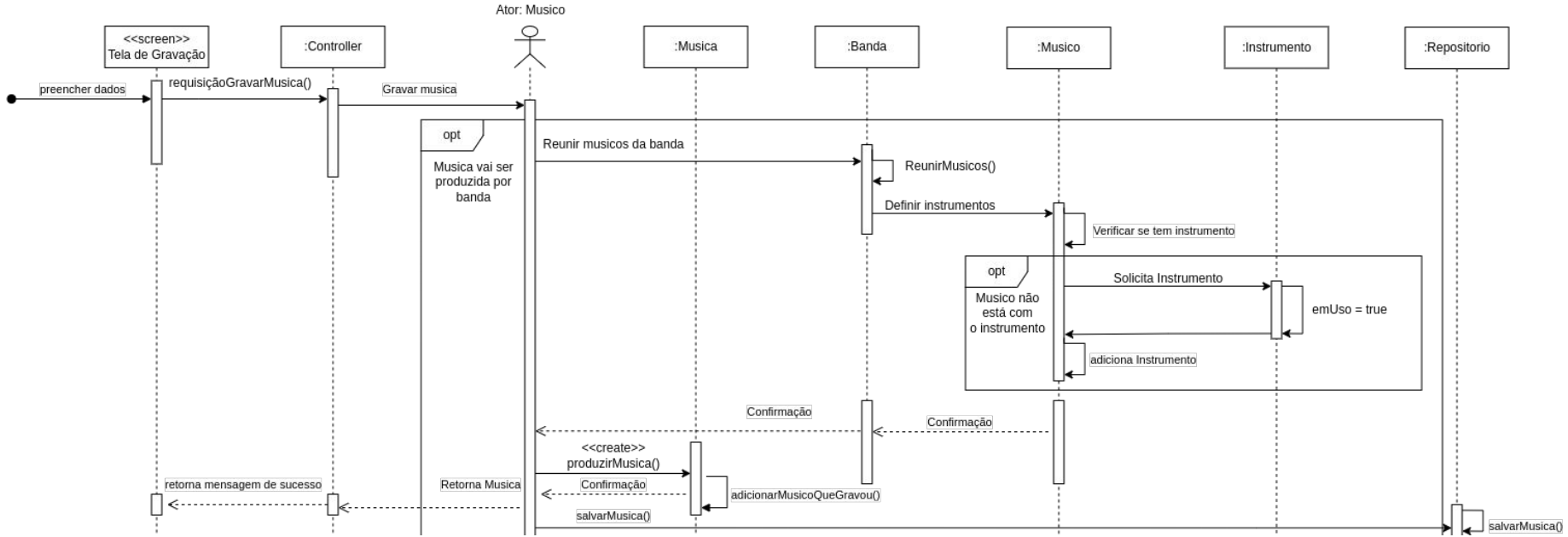
DIAGRAMAS DE SEQUÊNCIA



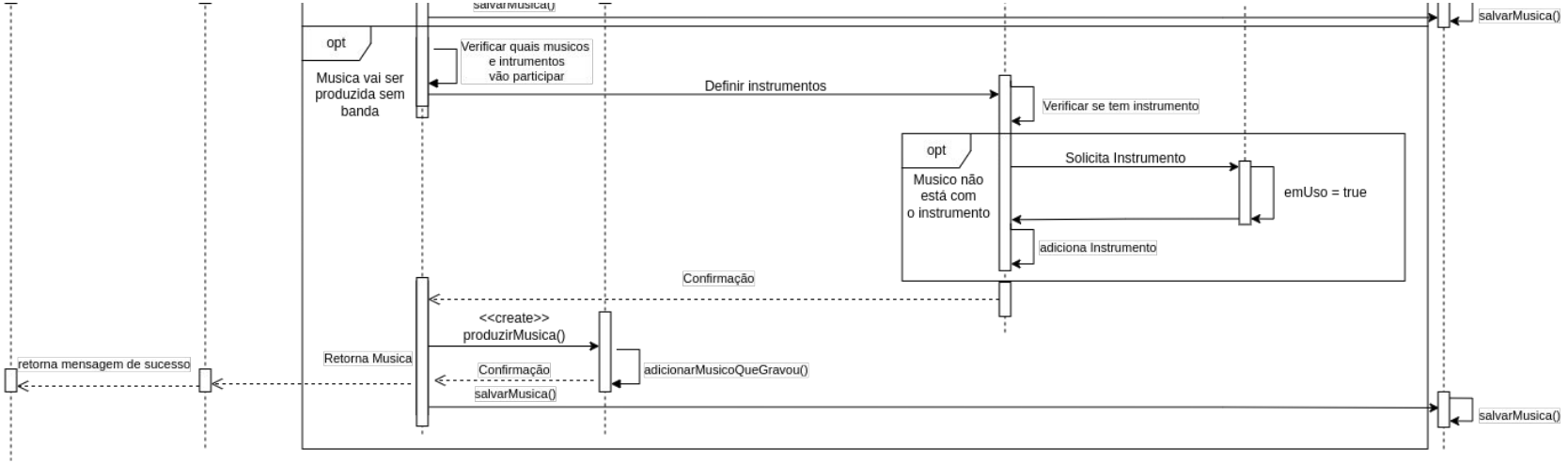
Gerar relatorio album



GRAVAR MUSICA



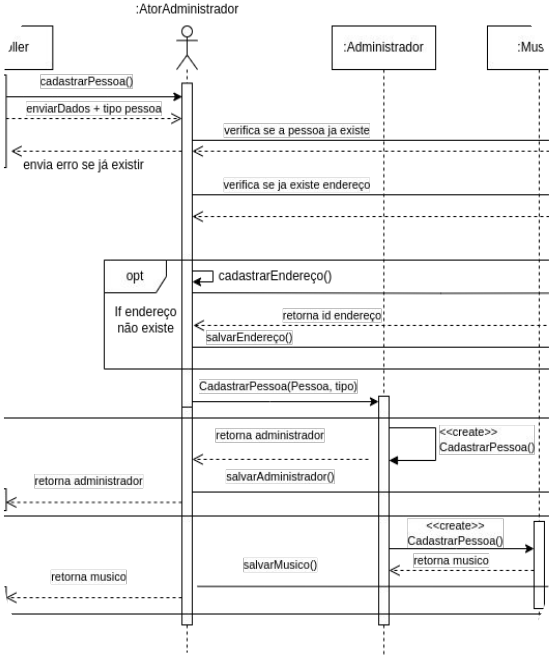
...continuação gravar música

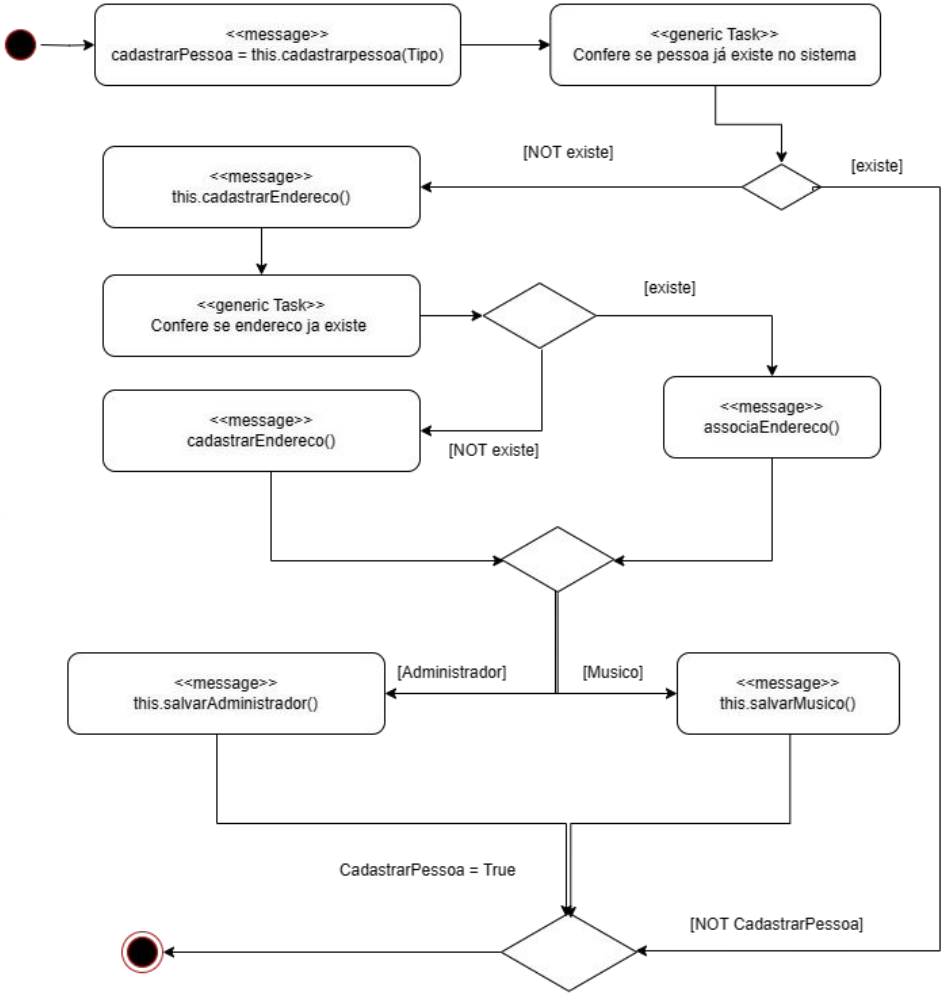


ETAPA 7

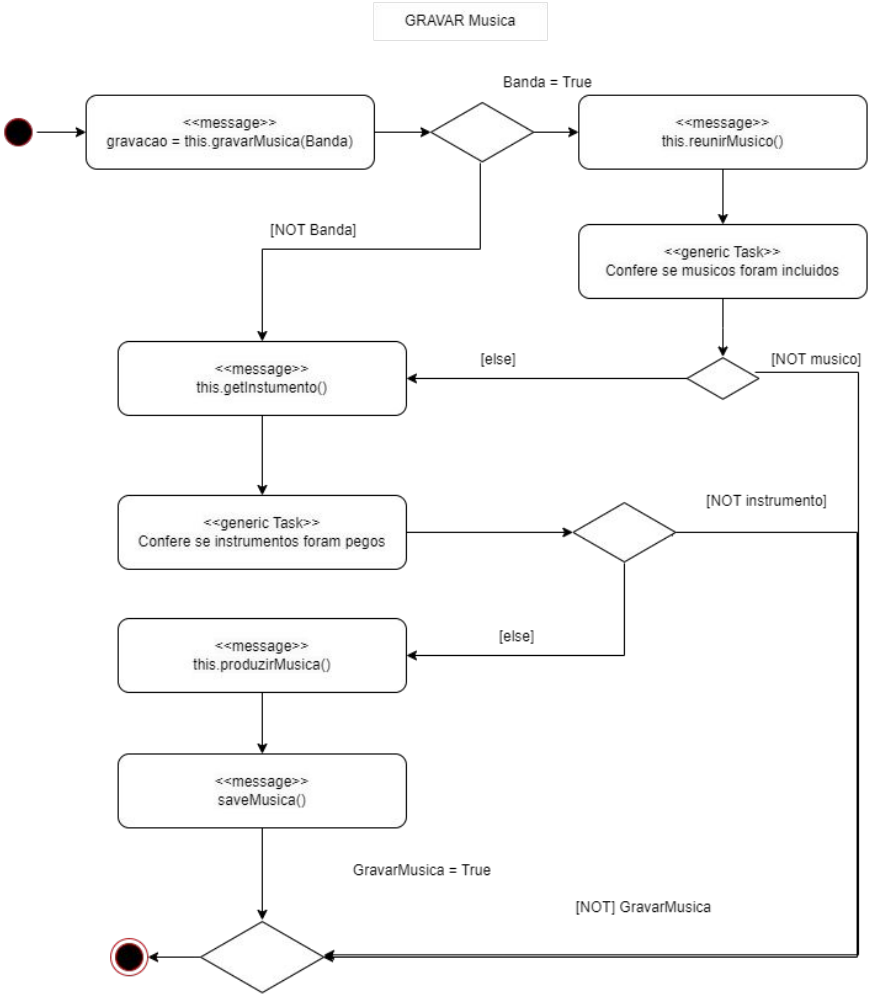
Modelagem de algoritmo de método

CADASTRAR PESSOA





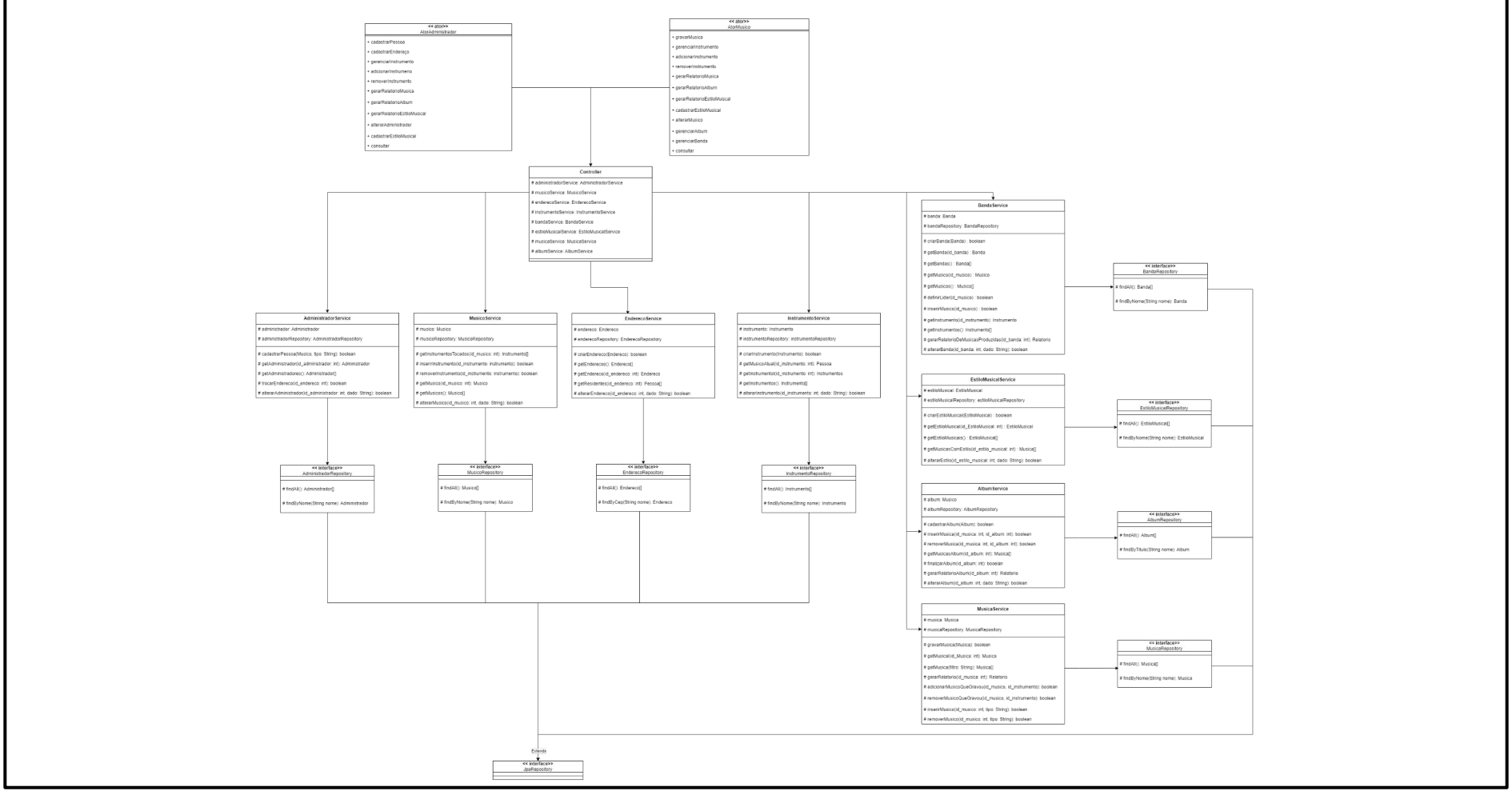
GRAVAR MUSICA

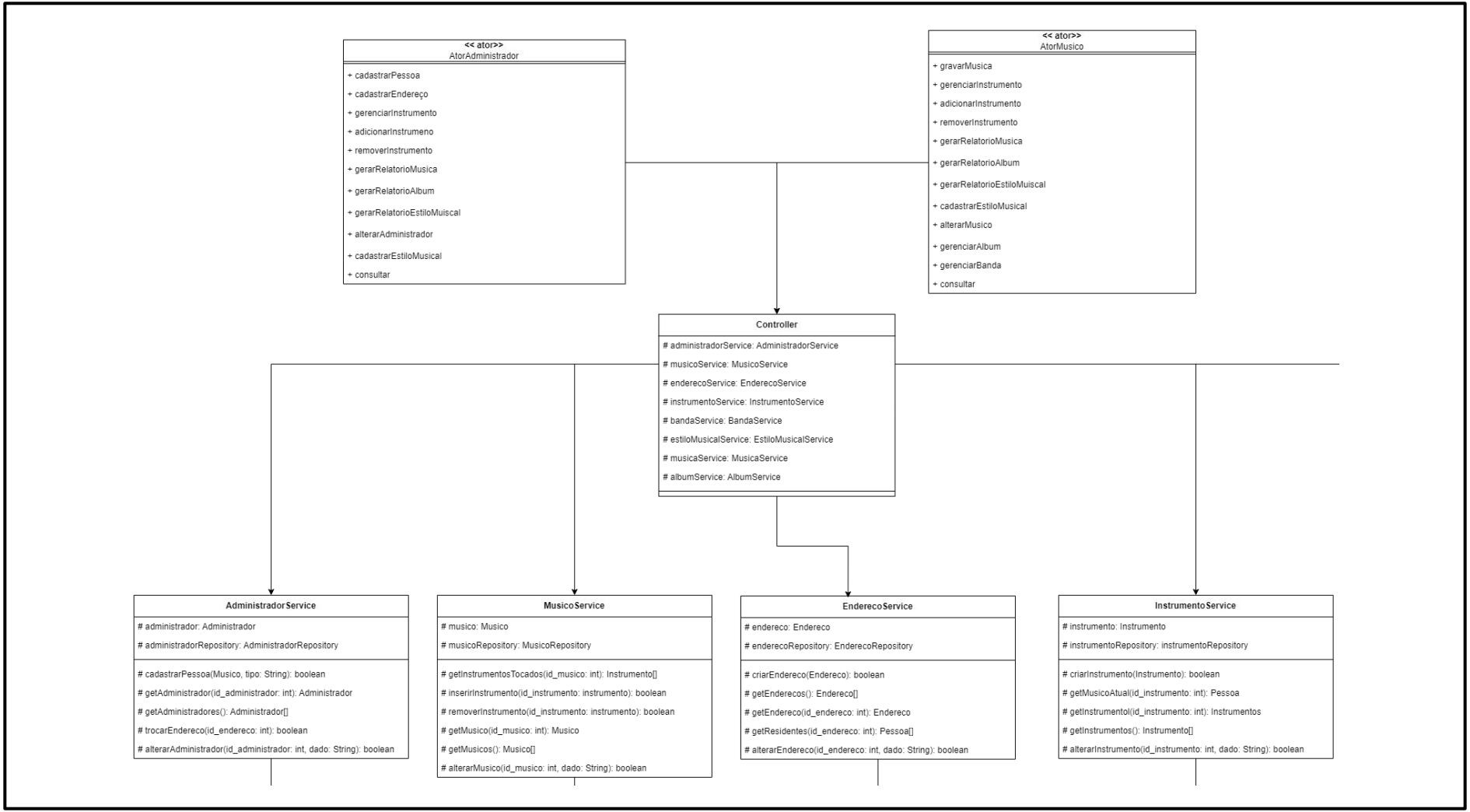


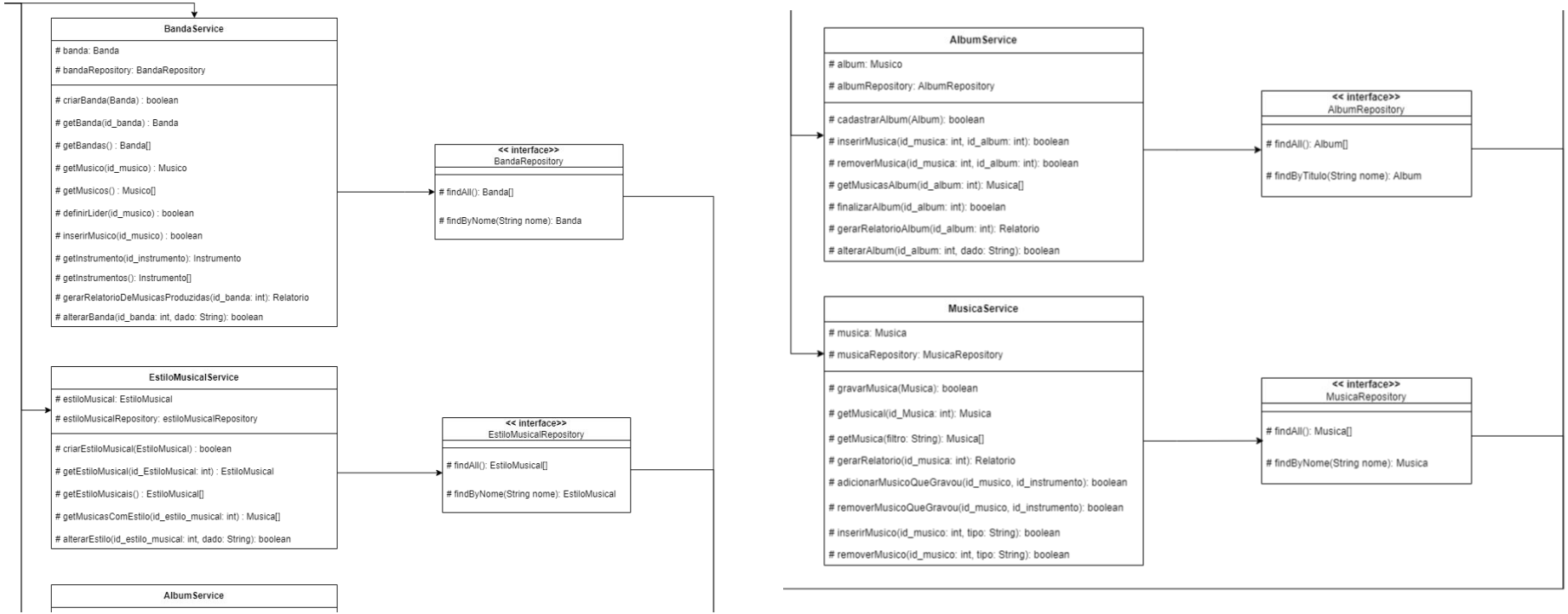
ETAPA 8

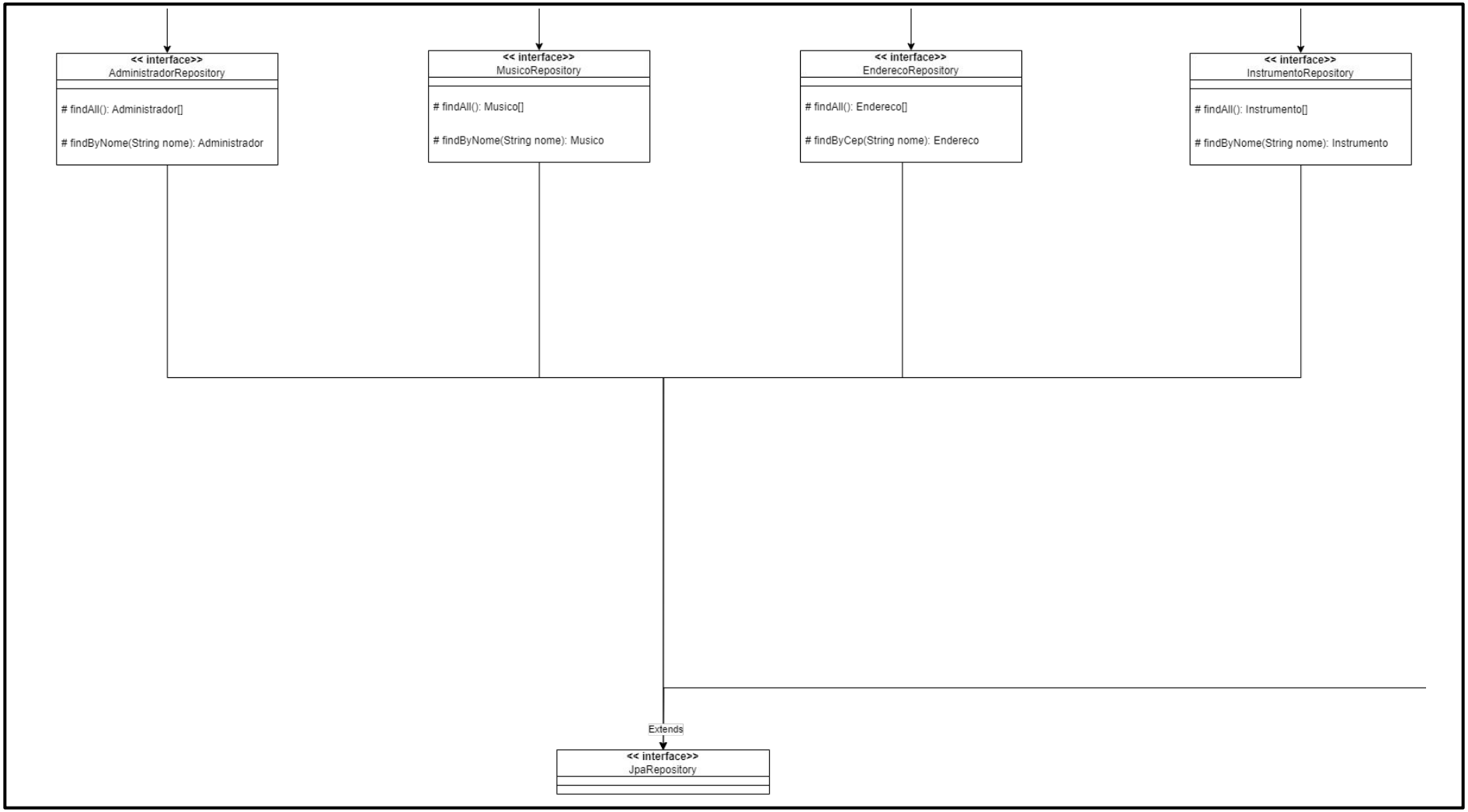
Geração de código e o
desenvolvimento iterativo

Diagrama de classes com elementos (completo)




<< ator>>
AtorAdministrador
+ cadastrarPessoa
+ cadastrarEndereço
+ gerenciarInstrumento
+ adicionarInstrumeno
+ removerInstrumento
+ gerarRelatorioMusica
+ gerarRelatorioAlbum
+ gerarRelatorioEstiloMuiscal
+ alterarAdministrador
+ cadastrarEstiloMusical
+ consultar
<< ator>>
AtorMusico
+ gravarMusica
+ gerenciarInstrumento
+ adicionarInstrumento
+ removerInstrumento
+ gerarRelatorioMusica
+ gerarRelatorioAlbum
+ gerarRelatorioEstiloMuiscal
+ cadastrarEstiloMusical
+ alterarMusico
+ gerenciarAlbum
+ gerenciarBanda
+ consultar
Controller
administradorService: AdministradorService
musicoService: MusicoService
enderecoService: EnderecoService
instrumentoService: InstrumentoService
bandaService: BandaService
estiloMusicalService: EstiloMusicalService
musicaService: MusicaService
albumService: AlbumService
AdministradorService
administrador: Administrador
administradorRepository: AdministradorRepository
cadastrarPessoa(Musico, tipo: String): boolean
getAdministrador(id_administrador: int): Administrador
getAdministradores(): Administrador[]
trocarEndereco(id_endereco: int): boolean
alterarAdministrador(id_administrador: int, dado: String): boolean
MusicoService
musico: Musico
musicoRepository: MusicoRepository
getInstrumentosTocados(id_musico: int): Instrumento[]
inserirInstrumento(id_instrumento: instrumento): boolean
removerInstrumento(id_instrumento: instrumento): boolean
getMusico(id_musico: int): Musico
getMusicos(): Musico[]
alterarMusico(id_musico: int, dado: String): boolean
EnderecoService
endereco: Endereco
enderecoRepository: EnderecoRepository
criarEndereco(Endereco): boolean
getEnderecos(): Endereco[]
getEndereco(id_endereco: int): Endereco
getResidentes(id_endereco: int): Pessoa[]
alterarEndereco(id_endereco: int, dado: String): boolean
InstrumentoService
instrumento: Instrumento
instrumentoRepository: instrumentoRepository
criarInstrumento(Instrumento): boolean
getMusicoAtual(id_instrumento: int): Pessoa
getInstrumentol(id_instrumento: int): Instrumentos
getInstrumentos(): Instrumento[]
alterarInstrumento(id_instrumento: int, dado: String): boolean


BandaService
banda: Banda
bandaRepository: BandaRepository
criarBanda(Banda) : boolean
getBanda(id_banda) : Banda
getBandas() : Banda[]
getMusico(id_musico) : Musico
getMusicos() : Musico[]
definirLider(id_musico) : boolean
inserirMusico(id_musico) : boolean
getInstrumento(id_instrumento): Instrumento
getInstrumentos(): Instrumento[]
gerarRelatorioDeMusicasProduzidas(id_banda: int): Relatorio
alterarBanda(id_banda: int, dado: String): boolean
<< interface>>
BandaRepository
findAll(): Banda[]
findByNome(String nome): Banda
EstiloMusicalService
estiloMusical: EstiloMusical
estiloMusicalRepository: estiloMusicalRepository
criarEstiloMusical(EstiloMusical) : boolean
getEstiloMusical(id_EstiloMusical: int) : EstiloMusical
getEstiloMusicais() : EstiloMusical[]
getMusicasComEstilo(id_estilo_musical: int) : Musica[]
alterarEstilo(id_estilo_musical: int, dado: String): boolean
<< interface>>
EstiloMusicalRepository
findAll(): EstiloMusical[]
findByNome(String nome): EstiloMusical
AlbumService
AlbumService
album: Musico
albumRepository: AlbumRepository
cadastrarAlbum(Album): boolean
inserirMusica(id_musica: int, id_album: int): boolean
removerMusica(id_musica: int, id_album: int): boolean
getMusicasAlbum(id_album: int): Musica[]
finalizarAlbum(id_album: int): booelan
gerarRelatorioAlbum(id_album: int): Relatorio
alterarAlbum(id_album: int, dado: String): boolean
<< interface>>
AlbumRepository
findAll(): Album[]
findByTitulo(String nome): Album
MusicaService
musica: Musica
musicaRepository: MusicaRepository
gravarMusica(Musica): boolean
getMusical(id_Musica: int): Musica
getMusica(filtro: String): Musica[]
gerarRelatorio(id_musica: int): Relatorio
adicionarMusicoQueGravou(id_musico, id_instrumento): boolean
removerMusicoQueGravou(id_musico, id_instrumento): boolean
inserirMusico(id_musico: int, tipo: String): boolean
removerMusico(id_musico: int, tipo: String): boolean
<< interface>>
MusicaRepository
findAll(): Musica[]
findByNome(String nome): Musica


<< interface>>
AdministradorRepository
findAll(): Administrador[]
findByNome(String nome): Administrador
<< interface>>
MusicoRepository
findAll(): Musico[]
findByNome(String nome): Musico
<< interface>>
EnderecoRepository
findAll(): Endereco[]
findByCep(String nome): Endereco
<< interface>>
InstrumentoRepository
findAll(): Instrumento[]
findByNome(String nome): Instrumento
Extends
<< interface>>
JpaRepository

CONTROLLER

- cadastrarPessoa

```java
@RestController
@ResponseBody
public class Controller {

    @Autowired
    AdministradorService administradorService;

    @Autowired
    EnderecoService enderecoService;

    @Autowired
    MusicaService musicaService;

    @Autowired
    MusicoService musicoService;

    @RequestMapping(value = "/cadastrarPessoa", method = RequestMethod.POST)
    public ResponseEntity<String> cadastrarPessoa(@RequestParam("nome") String nome, @RequestParam("cpf") String cpf,
        @RequestParam("telefone") String telefone, @RequestParam("celular") String celular, @RequestParam("email") String email,
            @RequestParam("rua") String rua, @RequestParam("bairro") String bairro, @RequestParam("numero") int numero,
                @RequestParam("cidade") String cidade, @RequestParam("complemento") String complemento,
                    @RequestParam("estado") String estado, @RequestParam("cep") String cep, @RequestParam("tipo") String tipo) {

        try {
            // verifica se o endereco ja existe no banco de dados
            boolean enderecoExistente = enderecoService.verificaExistencia(cep);

            // se o endereco nao existe, cadastra o endereco
            if (!enderecoExistente) {
                enderecoService.cadastrarEndereco(rua, bairro, numero, cidade, complemento, estado, cep);
            }

            // cadastra a pessoa
            administradorService.cadastrarPessoa(nome, cpf, telefone, celular, email, enderecoService.findByCep(cep), tipo);

            return new ResponseEntity<>("Pessoa cadastrada com sucesso", HttpStatus.OK);
        } catch (RuntimeException e) {
            return new ResponseEntity<>("Erro: " + e.getMessage(), HttpStatus.BAD_REQUEST);
        } catch (Exception e) {
            return new ResponseEntity<>("Erro no banco de dados: " + e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
```

ENDEREÇO SERVICE

- cadastrarEndereco

```java
public class EnderecoService {
    @Autowired
    private EnderecoRepository enderecoRepository;

    public void cadastrarEndereco(String rua, String bairro, int numero, String cidade,
    String complemento, String estado, String cep) {

        // verifica se o endereco ja existe no banco de dados
        Endereco enderecoExistente = enderecoRepository.findByCep(cep);

        // se o endereco ja existe, lanca uma excecao
        if (enderecoExistente != null) {
            throw new RuntimeException("Endereco já existe");
        }

        // Cria a instancia de Endereco
        Endereco endereco = new Endereco(rua, bairro, numero, cidade, complemento, estado, cep);

        // Salva o endereco no banco de dados
        enderecoRepository.save(endereco.getId(), endereco.getRua(), endereco.getNumero(),
        endereco.getBairro(), endereco.getCidade(), endereco.getEstado(), endereco.getCep());
    }

    // verifica se o endereco ja existe no banco de dados
    public boolean verificaExistencia(String cep) {
        Endereco enderecoExistente = enderecoRepository.findByCep(cep);
        if (enderecoExistente != null) {
            return true;
        }
        return false;
    }

    // retorna o endereco pelo cep
    public Endereco findByCep(String cep) {
        return enderecoRepository.findByCep(cep);
    }
}
```

ENDEREÇO REPOSITORY

- save
- findByRuaAndNumero
- findByCep

```java
public interface EnderecoRepository extends JpaRepository<Endereco, Long>{

    // query que vai salvar um endereco no banco de dados
    @Query("INSERT INTO endereco (rua, numero, bairro, cidade, estado, cep) VALUES (:rua, :numero, :bairro, :cidade, :estado, :cep)")
    void save(@Param("id") Long id ,@Param("rua") String rua, @Param("numero") int numero, @Param("bairro") String bairro,
        @Param("cidade") String cidade, @Param("estado") String estado, @Param("cep") String cep);

    // query findByRuaAndNumero
    @Query("SELECT e FROM endereco e WHERE e.rua = :rua AND e.numero = :numero")
    Endereco findByRuaAndNumero(String rua, int numero);

    // query findByCep
    @Query("SELECT e FROM endereco e WHERE e.cep = :cep")
    Endereco findByCep(String cep);
}
```

ADMINISTRADOR SERVICE

- cadastrarPessoa

```java
public class AdministradorService {
    @Autowired
    private AdministradorRepository administradorRepository;

    public String cadastrarPessoa(String nome, String cpf, String telefone, String celular,
        String email, Endereco endereco, String tipo) {

        try {
            // verifica se a pessoa ja existe no banco de dados
            Administrador administradorExistente = administradorRepository.findByCpf(cpf);

            // se a pessoa ja existe, lanca uma excecao
            if (administradorExistente != null) {
                throw new RuntimeException("Pessoa já existe");
            }

            // verifica o tipo de pessoa
            if (tipo.equals("administrador")) {

                // Cria a instancia de Administrador
                Administrador administrador = new Administrador(nome, cpf, telefone, celular, email, endereco);

                // Salva o administrador no banco de dados
                administradorRepository.saveAdministrador(administrador.getId(), administrador.getNome(),
                    administrador.getCpf(), administrador.getTelefone(), administrador.getCelular(),
                        administrador.getEmail(), administrador.getEndereco());

            } else if (tipo.equals("musico")) {
                // Salva o musico no banco de dados
                Musico musico = new Musico(nome, cpf, telefone, celular, email, endereco);
                administradorRepository.saveMusico(musico.getId(), musico.getNome(), musico.getCpf(),
                    musico.getTelefone(), musico.getCelular(), musico.getEmail(), musico.getEndereco());
            }

            return "Sucesso";
        } catch (RuntimeException e) {
            return "Erro: " + e.getMessage();
        } catch (Exception e) {
            return "Erro no banco de dados: " + e.getMessage();
        }
    }
```

ADMINISTRADOR REPOSITORY

- save
- findByRuaAndNumero
- findByCep

```java
@Repository
public interface AdministradorRepository extends JpaRepository<Administrador, Long> {

    // query que vai salvar um administrador ou um musico no banco de dados
    @Query("INSERT INTO administrador (nome, cpf, telefone, celular, email, endereco) VALUES (:nome, :cpf, :telefone, :celular, :email, :endereco)")
    void saveAdministrador(@Param("id") Long id ,@Param("nome") String nome, @Param("cpf") String cpf, @Param("telefone")
        String telefone, @Param("celular") String celular, @Param("email") String email, @Param("endereco") Endereco endereco);

    // query que vai salvar um musico no banco de dados
    @Query("INSERT INTO musico (nome, cpf, telefone, celular, email, endereco) VALUES (:nome, :cpf, :telefone, :celular, :email, :endereco)")
    void saveMusico(@Param("id") Long id ,@Param("nome") String nome, @Param("cpf") String cpf, @Param("telefone") String telefone,
        @Param("celular") String celular, @Param("email") String email, @Param("endereco") Endereco endereco);

    // query que vai buscar uma pessoa pelo cpf
    @Query("SELECT p FROM pessoa p WHERE p.cpf = :cpf")
    Administrador findByCpf(@Param("cpf") String cpf);
}
```

CONTROLLER

- gravarMusica

```java
// Recebe os dados necessarios para gravar uma musica
    @RequestMapping(value = "/gravarMusica", method = RequestMethod.POST)
    public ResponseEntity<String> gravarMusica(@RequestParam("id_estiloMusical") Long id_estiloMusical, @RequestParam("nome") String nome,
         @RequestParam("duracao") String duracao, @RequestParam("data_Criacao") String data_Criacao, @RequestParam("autores_letra") String autores_letra,
             @RequestParam("autores_musica") String autores_musica, @RequestParam("instrumentos") String instrumentos,
                @RequestParam("produtor") String produtor, @RequestParam("musicoQueGravou") String musicoQueGravou,
                    @RequestParam("id_banda") String id_banda) {

        // transforma os parametros em listas
        List<Instrumento> listaInstrumentos = musicaService.transformaInstrumentoLista(instrumentos);
        List<Musico> listaAutoresLetra = musicaService.transformaMusicoLista(autores_letra);
        List<Musico> listaAutoresMusica = musicaService.transformaMusicoLista(autores_musica);
        Map<Long, Long> listaMusicoQueGravou = musicaService.transformaMap(musicoQueGravou);

        // faz a verificação se os musicos ja estao associados ao instrumento
        // verifica o id do musico e o id do instrumento presentes em listaMusicoQueGravou
        for (Map.Entry<Long, Long> entry : listaMusicoQueGravou.entrySet()) {
            Long idMusico = entry.getKey();
            Long idInstrumento = entry.getValue();

            // verifica se o musico ja esta associado ao instrumento
            boolean musicoAssociado = musicoService.verificaInstrumento(idMusico, idInstrumento);

            // se o musico nao esta associado ao instrumento, cadastra a associacao
            if (!musicoAssociado) {
                musicoService.associarInstrumento(idMusico, idInstrumento);
            }
        }

        try {
            // grava a musica
            musicaService.gravarMusica(id_estiloMusical, nome, duracao, data_Criacao, listaAutoresLetra, listaAutoresMusica,
                listaInstrumentos, produtor, listaMusicoQueGravou, id_banda);

            return new ResponseEntity<>("Musica cadastrada com sucesso", HttpStatus.OK);
        } catch (RuntimeException e) {
            return new ResponseEntity<>("Erro: " + e.getMessage(), HttpStatus.BAD_REQUEST);
        } catch (Exception e) {
            return new ResponseEntity<>("Erro no banco de dados: " + e.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }
```

MUSICO SERVICE

- verificaInstrumento
- associarInstrumento

```java
public class MusicoService {

    @Autowired
    private MusicoRepository musicoRepository;

    public boolean verificaInstrumento(Long id_musico, Long id_instrumento) {

        // verifica no banco de dados se o musico tem aquele instrumento na sua lista de instrumentos
        boolean resposta = musicoRepository.findById(id_musico).get().getInstrumentos().contains(id_instrumento);
        if (resposta) {
            return true;
        } else {
            return false;
        }
    }

    // associa um instrumento a um musico
    public boolean associarInstrumento(Long id_musico, Long id_instrumento) {
        // verifica se o musico ja tem aquele instrumento
        if (verificaInstrumento(id_musico, id_instrumento)) {
            return false;
        } else {
            // adiciona o instrumento na lista de instrumentos do musico
            musicoRepository.findById(id_musico).get().getInstrumentos().add(id_instrumento);
            return true;
        }
    }
}
```

OBRIGADO!

Tem alguma dúvida?