UNIVERSIDADE FEDERAL DE UBERLÂNDIA
MODELAGEM DE SOFTWARE

GUSTAVO MENDES SANTOS
HEITOR GUIMARÃES DA FONSECA FILHO
IGOR AUGUSTO REIS GOMES

Sistema de Gestão de Gravadora de Música

Uberlândia - MG
2023

Sumário

1 INTRODUÇÃO

A indústria da música é um campo dinâmico e apaixonante, permeado pela criatividade e expressão artística. No entanto, mesmo nesse cenário vibrante, a gestão eficiente dos processos envolvidos na produção musical é fundamental para o sucesso e sustentabilidade de uma gravadora. Este trabalho propõe um sistema abrangente de gestão para uma gravadora de música, enfrentando os desafios inerentes à organização de informações cruciais que permeiam o universo musical.

1.1 Contextualização

A gestão de uma gravadora envolve uma complexa rede de relações entre músicos, instrumentos, estilos musicais, bandas e álbuns. Atualmente, muitas gravadoras enfrentam desafios na organização eficiente dessas informações, o que pode resultar em perda de tempo, falta de transparência e, em última instância, impactar a qualidade da produção musical.

O propósito deste projeto é desenvolver um sistema de gestão de gravadora que abranja todas as fases do processo excluindo parte de venda, fornecedores e etapas após a produção do álbum, ou seja, abrange desde o cadastro detalhado de músicos até a produção de álbuns e a geração de relatórios analíticos. Ao fazer isso, visamos proporcionar à gravadora uma ferramenta robusta que otimize as operações diárias, promova a colaboração eficaz entre músicos e maximize o potencial criativo.

1.2 Metodologia de Modelagem

Este trabalho de modelagem de software emprega representações gráficas por meio de diagramas, dividindo-se em quatro seções principais. Inicialmente, tem-se a "Modelagem do Domínio do Problema", no qual percorremos as Etapas 1 a 4, abrangendo desde a modelagem estrutural e dinâmica em alto nível de abstração até o refinamento de casos de uso e a modelagem de estados associada à classe, respectivamente.

A seguir, na "Modelagem do Domínio da Solução" (Etapa 5), introduzimos elementos do domínio da solução computacional. Em seguida, na Etapa 6, destacamos situações especiais na modelagem com a modelagem de destaques na Especificação. A Etapa 7, “Modelagem de

Algoritmos de Métodos", concentra-se na representação dos algoritmos associados. A conclusão do processo se dá na Etapa 8, com a geração de código e o desenvolvimento iterativo.

2 ETAPA 1

Na etapa inicial de modelagem, focamos na representação **estrutural e dinâmica em alto nível de abstração** por meio de diversos diagramas, como os de classes, de casos de uso e de visão geral de interação. Na modelagem estrutural em alto nível de abstração, identificamos elementos essenciais do domínio do problema e, em seguida, procedemos à composição de diagrama(s) de classes com base nesses elementos. Já na esfera da modelagem dinâmica, identificamos as funcionalidades do software a serem desenvolvidos, construindo tanto o diagrama de casos de uso com base nessas funcionalidades quanto o diagrama de visão geral de interação.

É de suma importância ressaltar que inicialmente se trata de uma modelagem simplificada e abstrata, focando nos principais elementos e funcionalidades, servindo como base para o desenvolvimento posterior, isto é, fornecendo uma visão geral da estrutura e do comportamento do sistema. Portanto, nas próximas etapas, a modelagem será detalhada e refinada, levando em consideração requisitos específicos, restrições técnicas, entre outros.

2.1 Diagrama de Classes

Antes de construir o diagrama de classes, primeiro é necessário identificar as classes propriamente ditas utilizando como fonte neste caso, a especificação/descrição do projeto fornecida. Assim, a partir da mesma foi possível inicialmente a identificação das seguintes classes: **Músico**, **Endereço**, **Instrumento**, **Estilo Musical, Banda**, **Música** e **Álbum**.

Dado que nessa etapa não há a pretensão de identificar todas as classes ou todos os relacionamentos, é possível reconhecer a existência de algumas ligações entre as classes (mas não o tipo específico de relacionamento), como:

- Músico e Instrumento: um instrumento pode ser tocado por vários músicos e um músico pode tocar vários instrumentos.

- Músico e Endereço: um músico possui um único endereço e vários músicos podem morar no mesmo endereço.

- Músico e Banda: um músico pode estar apenas em uma banda por vez, enquanto uma banda pode conter vários músicos.

- Músico e Música: um músico pode participar de várias produções de músicas e uma música pode ser produzida por vários músicos

- Músico e Álbum: um álbum deve conter o músico produtor, e um músico pode produzir vários álbuns.

- Música e Instrumento: uma música pode ser gravada por vários instrumentos e um instrumento pode ser usado para produzir várias músicas.

- Música e Estilo Musical: uma música contém um estilo musical e um estilo musical pode estar contido em várias músicas.

- Música e Banda: uma banda pode produzir nenhuma ou várias músicas, enquanto uma música pode ser produzida sem banda ou apenas com uma banda.

- Música e Álbum: um álbum contém uma ou várias músicas, enquanto uma música pode estar contida em nenhum ou apenas em um único álbum.

2.2 Diagrama de Casos de Uso

Neste diagrama baseado nas funcionalidades identificadas, representa-se visualmente os diferentes casos de uso, descrevendo as interações entre os atores e o sistema. Dessa forma, ocorre a identificação das principais ações ou fluxos de eventos em cada caso de uso.

Dado isso, foram identificados dois atores: Administrador e Músico, além dos casos de uso: Cadastrar Músico, Cadastrar Estilo Musical, Gerenciar Instrumento, Gerar Relatório, Gerenciar Banda, Gravar Música e Produzir Álbum, os quais ambos atores e casos de uso serão detalhados a seguir no formato <CASO DE USO> (<ATOR(ES)>):

2.2.1 Cadastrar Músico (Administrador)

Permite ao administrador cadastrar novos músicos no sistema

2.2.2 Gerenciar Instrumento (Administrador e Músico)

Facilita a administração dos instrumentos disponíveis, permitindo adições, remoções e atualizações.

2.2.3 Gerar relatório (Administrador e Músico)

Oferece a capacidade de gerar relatórios relevantes para análise e tomada de decisões.

2.2.4 Cadastrar Estilo Musical (Administrador e Músico)

Possibilita a ambos os atores a inclusão ou modificação de novos estilos musicais no sistema

2.2.5 Gerenciar Banda (Músico)

Permite ao Músico administrar bandas, podendo criar ou alterara-las. Facilitando a adição ou remoção de membros.

2.2.6 Gravar Música (Músico)

Permite ao Músico registrar composições e gravações.

2.2.7 Produzir Álbum (Músico)

Permite facilitar o processo de produção de álbuns, organizando e incluindo músicas.

2.3 Diagrama de Visão Geral de Interação

Neste diagrama visualiza-se as interações entre os diferentes componentes do sistema, representando a sequência de mensagens trocadas, identificando os principais fluxos de comunicação e o comportamento geral do sistema em relação às funcionalidades identificadas.

No projeto em si não foram identificadas tantas sequências sobre os casos de uso, por exemplo, um músico pode gravar uma música e não necessariamente já inseri-la em um álbum, assim como um álbum pode ser criado sem nenhuma música inicialmente.

Assim, as sequências necessárias foram pensadas em caso um usuário entre no sistema ele possa selecionar as opções baseadas no tipo de ator que ele é. O fluxo identificado mostra as funcionalidades que cada ator tem acesso e após concluir alguma delas ele tem a opção de fazer outra ação ou sair do sistema. Dessa forma, o aprofundamento nos casos de uso e nas funcionalidades do sistema são feitos nas etapas seguintes.

3 ETAPA 2

Na Etapa 2, de **refinamento estrutural**, aprofundamos a modelagem de classes, identificando atributos específicos para cada uma e respeitando as ligações entre as classes. Esse refinamento é essencial para uma representação mais detalhada das entidades no sistema, contribuindo com as necessidades da gravadora.

A seguir especifica-se as classes, no formato <CLASSE>: [<ATRIBUTOS>]:

1. **Músico:** id_musico, nome, cpf, telefone, celular, email, instrumentos e endereco
2. **Endereço:** id_endereco, rua, bairro, número, cidade, complemento, estado e cep
3. **Instrumento:** id_instrumento, nome, modelo, tom_musical e marca
4. **Estilo Musical:** id_estilo_musical, nome e descrição
5. **Banda:** id_banda, nome, descrição, data_criação, líder e musicos
6. **Álbum:** id_album, titulo, cod_album, data_lancamento, formato, produtor, banda e musicas
7. **Música:** id_musica, nome, duração, data_criação, instrumentos, autores letra, autores musica, produtor, banda

4 ETAPA 3

Nesta terceira etapa, ocorre o **refinamento de casos de uso**, onde o objetivo é detalhar alguns casos de uso para melhorar a compreensão do sistema. Nesse caso foram usados os diagramas de atividades e sequência, envolvendo a participação de vários objetos que focam no caso de uso.

Foi visto a necessidade de revisar o caso de uso “Gerar relatório”, onde estava muito abstrato. Por isso dividimos esse caso de uso em outros dois, “Gerar relatório Música” e “Gerar relatório Álbum”, onde ambas as funcionalidades podem ser realizadas pelos atores Administrador e Músico.

Durante o processo, foi decidido também criar a classe Pessoa e a classe Administrador. A classe músico e administrador herdam a classe Pessoa. Afetando assim o relacionamento entre músico e endereço, já que agora o endereço se relaciona diretamente com a classe Pessoa. Por isso, o caso de uso “Cadastrar Músico” se transformou em “Cadastrar Pessoa” onde se pode escolher o tipo de pessoa a ser cadastrado.

4.1 Diagrama de Atividades

- **Adicionar Músico que gravou**
    - Durante a montagem de alguns diagramas, percebemos a necessidade de associar o instrumento utilizado na gravação da música com o músico que usou o instrumento. Por isso foi confeccionado um diagrama de atividades exclusivo para essa função. Com essa alteração adicionamos uma nova lógica, onde o instrumento só pode estar vinculado a um músico por vez, ou seja, a classe instrumento terá um atributo para descrever se aquele instrumento está em uso ou não.
    - O fluxo dessa atividade envolve primeiro verificar a existência do músico e a do instrumento, verificar se já estão associados, caso não estejam vinculados e o instrumento não esteja em uso por outro músico é feita a associação. Esses dados são salvos na classe Música, o que implica a criação de um novo atributo "musicoQueGravou" onde associa o id do músico com o id do instrumento em uma tabela Hash.

- **Gerar relatório Música**
    - Pensando em termos de desempenho, os dados da gravadora serão salvos em um banco de dados. Logo, para acessá-los é necessário fazer uma consulta SQL, pensando nisso não faria sentido que para gerar um relatório de todas as músicas de um álbum ou músicas de um determinado estilo musical fizéssemos diversas consultas.
    - Por isso pensamos que o caso de uso "Gerar relatório de Música" deveria ser uma filtragem de parâmetros SQL para fazer uma consulta única e retornar todos os dados necessários. Seguindo essa lógica o fluxo de atividade consiste em criar a consulta SQL com os parâmetros solicitados, buscar as músicas solicitadas, verificar a existência delas e formatar o relatório.

- **Gerar relatório Álbum**
    - Seguindo a mesma lógica do “Gerar relatório música”, esse caso de uso vai basicamente filtrar todas as músicas que são referentes aos álbuns, pegar o relatório de cada música existente no álbum e somar com as informações daquele álbum. Essa operação só é possível caso exista no mínimo uma música no álbum.

- **Adicionar Música no Álbum**
    - Dentro do caso de uso “Gerenciar álbum” foi especificado uma atividade de adicionar uma música no álbum. Por consequência adicionamos um atributo na classe Álbum que indica se aquele álbum está ou não finalizado. Seguindo o fluxo de execução, uma música só é adicionada ao álbum caso ela não pertença a ele incialmente e caso ele não esteja finalizado.

- **Adicionar Músico na Banda**
    - Esse diagrama visa mostrar o fluxo de execução ao tentar inserir um Músico em uma banda.

4.2 Diagrama de Sequência

No que tange o diagrama de sequência da Etapa 4, seu principal objetivo é mostrar a interação dos objetos ao longo do tempo em determinados métodos identificados. Assim sendo, foi possível a criação de 4 (quatro) diagramas de sequência.

O primeiro é o de cadastrarPessoa(), que é um método da classe Administrador e também é representado com um caso de uso. Basicamente é passado as informações necessárias (tipo de cadastro – administrador ou músico, endereço - id_endereço já existente ou dados para criar novo endereço, dados da pessoa) para uma instância da classe Administrador que realiza o cadastro da pessoa. Cabe ressaltar que o único atributo diferente entre músico e administrador, são os instrumentos tocados pelo músico no qual começa vazio.

Já o segundo é referente ao caso do uso “GravarMusica”, onde representa o registro de uma música que foi produzida por uma banda ou por músicos independentes, no qual mostra o registro dos dados básicos da música, verificação de uso dos instrumentos e alocação dos músicos participantes.

O terceiro e quarto, por sua vez representam os casos de uso “GerarRelatorioMusica” e “GerarRelatorioAlbum”, respectivamente, nos quais especifica-se algumas chamadas de métodos ao longo do processo de gerar algum tipo de relatório.

Por fim, nessa etapa não foi visto a necessidade de criar mais diagramas de sequência, visto que a maioria dos métodos do sistema não são complexos o suficiente para isso. Em seguida fica a relação das classes já identificadas até aqui com seus respectivos métodos:

1. **Pessoa:** getters/setters(), trocarEndereço(Endereço endereço)
2. **Administrador:** cadastrarPessoa(Pessoa, String tipo)
3. **Músico:** adicionarInstrumento(Instrumento), removerInstrumento(Instrumento)
4. **Endereço:** getResidentes()
5. **Instrumento:** getters/setters()
6. **Estilo Musical:** gerarRelatorioMusicas()
7. **Banda:** definirLider(Musico), removerLider(Musico), adicionarMusico(), removerMusico(), gerarRelatorioMusicasProduzidas()
8. **Album:** inserirMusica(Musica), removerMusica(), getMusicas(), gerarRelatorioAlbum(), finalizarAlbum()
9. **Música:** inserirMusico(Musico, String tipo), adicionarMusicoQueGravou(Musico, Instrumento), removerMusicoQueGravou(Musico, Instrumento), gerarRelatorioMusica(), gravarMusica()

5 ETAPA 4

Nesta última etapa da modelagem do domínio do problema, ocorre a **modelagem de estados associada à classe**, isto é, a modelagem dinâmica focada em uma única classe, que descreve a existência de um objeto desde a criação até a destruição e sua participação em vários casos de uso, fazendo-se assim, o uso do diagrama de máquina de estados.

5.1 Diagrama de Máquina de Estados

O principal objetivo do diagrama de máquina de estados é representar os estados de um objeto, bem como suas transições de estado, demonstrando todas as situações possíveis em que um objeto pode se encontrar ao longo de sua existência. Dessa forma, foram elaborados 4 diagramas de máquina de estados, os quais serão descorridos a seguir.

O primeiro é o de 'GravacaoMusica', que é um objeto da classe 'Musica' e também é representado como um caso de uso. Consideramos que a gravação está inicialmente no estado inativo e que, quando a gravação é solicitada, ela é iniciada. No entanto, durante o período em que está ativa, não é possível realizar outra gravação. Portanto, entendemos que o objeto 'GravacaoMusica' possui três estados: Pendente, Ativa e Inativa.

O segundo diagrama refere-se ao 'Relatório', um objeto genérico presente em mais de uma classe: Banda, Álbum, Música e EstiloMusical. Ele possui 3 estados: Pendente, Em Andamento e Concluído. As transições ocorrem da seguinte forma: inicialmente no estado Pendente, o relatório passa para Em Andamento quando é iniciado e, em seguida, de Em Andamento para Concluído quando finalizado.

O terceiro diagrama representa o objeto 'Instrumento', o qual, de acordo com a ênfase do problema, pode ser utilizado por vários músicos. Inicialmente, encontra-se sem uso. Quando um músico o utiliza, nenhum outro músico pode usá-lo até que o músico que o está utilizando termine.

Com base na premissa acima identificamos 4 estados: Sem uso, aguardar Instrumento, Em uso e Concluido. Assim, quando um músico deseja utilizar o instrumento, faz a solicitação. O sistema consulta o instrumento e verifica se está em uso por outro músico por meio do método 'ConsultarUsoInstrumento()' identificado nesta etapa. Se o instrumento estiver em uso, ele passa para o estado de Aguardar Instrumento; caso não esteja em uso, ou após o término do aguardo, passa para o estado Em Uso. Após o músico terminar de utilizar o instrumento, passa para o estado Concluído e o objeto é destruído.

O quarto objeto é o 'Álbum', que é criado pelo método 'CriarAlbum()', identificado nesta etapa, e começa vazio. Conforme inserimos músicas nesse álbum, ele permanece no estado Em Construção até que todas as músicas sejam inseridas, momento em que passa para o estado Concluído. Portanto, esse objeto possui 3 estados: Vazio, Em Construção e Concluído.

6 ETAPA 5

A partir da Etapa 5, ocorre a transição da modelagem do domínio do problema para a modelagem da solução do problema, mais especificamente, a **introdução de elementos do domínio da solução computacional**. O foco está na integração de elementos tecnológicos específicos para a solução computacional em desenvolvimento, por meio da identificação das demandas tecnológicas, e a partir disso definindo as opções dada as demandas identificadas.

O primeiro passo desta etapa foi selecionar os elementos mais adequados ao projeto e aos conhecimentos técnicos do grupo. Definimos que seria uma aplicação web usando o framework Spring Boot e a linguagem Java, com os dados armazenados em um banco de dados PostgreSQL (Supabase) por meio do Java Persistence API (JPA).

Além disso, optamos por adotar o modelo padrão de arquitetura Model-View-Controller (MVC), que oferece uma clara separação de responsabilidades. No que diz respeito ao front-end, optamos por utilizar HTML para estruturação, CSS e a biblioteca React (por meio da linguagem Typescript) para estilização, simplificando o processo de codificação.

6.1 Diagrama de Classes (com elementos)

Após as definições feitas anteriormente, partimos para a revisão do diagrama de classe para incluir os elementos do domínio computacional. Dado que nos situamos dentro do contexto do Spring Boot, o mesmo promove um modelo de arquitetura em camadas, no qual ajuda separar as responsabilidades da aplicação de maneira clara e estruturada, facilitando a manutenção, o teste e a escalabilidade do projeto. Portanto, temos três classes fundamentais, as quais serão introduzidas ao diagrama de classes: Controller, Service e Repository.

Inicialmente, têm-se a classe **Controller** (optamos pelo uso de uma única em contraposição do uso de múltiplas, com objetivo de simplificação), ela representa a camada responsável por receber as requisições do cliente (como o front-end) e direcioná-las para os devidos processamentos. Aqui, é onde ocorre a implementação dos métodos que lidam com uma interação direta com as operações/requisições HTTP (GET, POST, PUT, DELETE).

Portanto, a classe Controller relaciona-se com as Services por meio de instâncias dos Services pertinentes associados a cada rota/endpoint, por exemplo: administradorService: AdministradorService, musicoService: MusicoService, enderecoService: EnderecoService, entre outros, além disso, chamando os métodos correspondentes para processamento.

Já as classes **Service** por sua vez, concentram a lógica de negócios da aplicação, sendo responsáveis por implementar as regras de negócio, realizar operações de validação, processamento de dados e interações com o repositório (banco de dados). Além disso, ela abstrai as operações com o intuito de assegurar um acoplamento fraco entre a camada de controle e a de persistência.

Ademais, as classes Service relacionam-se com os Repositories relevantes por meio de instâncias dos mesmos (além de uma instância de sua entidade/classe correspondente) para acessar e manipular os dados do banco, realizando assim, operações intermediárias entre o Controller e o Repository. Por exemplo, têm-se a classe AdministradorService, na qual possui instâncias do Administrador (representando um administrador específico) e do AdministradorRepository, além de possuir métodos como:

- cadastrarPessoa(Musico, tipo: String): boolean ,
- getAdministrador(id_administrador: int): Administrador,
- getAdministradores(): Administrador[],
- trocarEndereco(id_endereco: int): boolean

Tais métodos lidam com as operações lógicas relacionadas ao administrador, como cadastrar uma pessoa (músico, administrador), obter um administrador pelo ID, listar todos os administradores, e trocar o endereço de um administrador. Esses métodos interagem com o AdministradorRepository para realizar operações no banco de dados, como buscar por ID, buscar todos os administradores ou realizar outras operações de persistência.

Por fim, implementamos os **Repositories**, os quais se tratam da camada responsável pela realização de operações de persistência de dados, interagindo diretamente com o banco de dados. Dessa forma, tratam-se interfaces (contendo apenas as assinaturas do métodos) que estendem (por se tratar de um contexto de Spring Boot) a interface JpaRepository fornecida pelo Spring Data JPA, na qual fornece métodos padrão para realizar operações de persistência comuns, como salvar, atualizar, excluir e buscar dados no banco de dados.

Desse modo, ao estender essa interface, tais métodos são herdados e torna também possível personalizar consultas adicionais, como na interface EnderecoRepository tem-se o método findByCep(String cep), que retorna um Endereço específico. Tal método pode ser utilizado em alguma operação da classe Service correspondente, dado que possui uma instância de seu Repository, sendo possível por meio do padrão de projeto de injeção de independência.

6.2 Diagrama de Sequência (com elementos)

Como resultado da Etapa 5, foi necessária uma remodelação dos diagramas de sequência. Nas fases anteriores, a solução computacional ainda não estava definida, então abstraímos as funcionalidades e chamadas de funções em uma classe denominada ‘Sistema’. Uma das principais mudanças é que, o usuário irá interagir com o sistema por meio de uma página web, na qual selecionará a funcionalidade desejada e preencherá os dados necessários.

No primeiro diagrama alterado, o de 'Cadastrar Pessoa', o administrador irá acessar a página web e preencher os dados necessários. Em seguida, ele irá confirmar a solicitação, e o frontend fará uma requisição para o backend. Este receberá os dados para cadastro, juntamente com o tipo de pessoa a ser registrada, por meio da classe 'AdministradorController'.

A classe 'AdministradorController' (também referida como 'Controller') irá criar uma instância do 'AdministradorService' (representado no diagrama como 'AtorAdministrador'), que, por sua vez, criará um objeto 'Administrador' para armazenar as informações do usuário que está fazendo a requisição.

7 ETAPA 6

Na sexta etapa destacamos as **situações especiais na modelagem**, identificamos situações que fogem da linearidade das etapas anteriores, seja por não terem terem uma correspondência direta com o paradigma de orientação a objetos ou por serem destaques julgados importantes pelo grupo.

Entendemos que era importante ter uma visão de fluxo do sistema e de como seria a navegação do usuário desde de a tela de login, passando pelas duas possíveis telas após o login, as quais podem ser:

- **Tela Administrador** com as opções disponíveis nesta tela (Cadastrar Pessoa, Gerenciar Instrumento, Cadastrar Estilo Musical, Gerar relatório).
- **Tela Musico** com as opções disponíveis nesta tela (Gravar Musica, Gerenciar Instrumento, Cadastrar Estilo Musical, Gerar relatório, Gerenciar Album e Gerenciar Banda)

Ao adentrar em outra tela, que apresenta uma complexidade maior, denominada 'Gerar Relatório', deparamo-nos com opções para diferentes tipos de relatórios: por Música, Estilo Musical, Álbum ou personalizado. Após a seleção e atribuição dos parâmetros necessários, um SQL dinâmico é gerado, o qual realiza a busca no servidor.

Posteriormente, essa solicitação é enviada ao banco de dados que, por sua vez, retorna o resultado final para o usuário. Esse relatório gerado é então exibido na tela de 'Mostrar Relatório'.

8 ETAPA 7

Na sétima etapa, ocorre a Modelagem de algoritmo de método, utilizando a modelagem dinâmica de classe para reunir as informações contidas na especificação e modelar os algoritmos dos métodos. Analisamos todos os diagramas criados até o momento e identificamos alguns métodos importantes a serem representados. Optamos por aqueles que têm uma ou mais mensagens originadas durante a execução do método, assim, foram elaborados 2 diagramas de atividade:

O primeiro é o método 'cadastrarPessoa()', pertencente à classe 'Administrador' e também representado como um caso de uso. Essencialmente, são passadas as informações necessárias (tipo de cadastro - administrador ou músico, endereço e dados da pessoa) para uma instância da classe 'Administrador', que dá início ao processo de cadastro da pessoa.

Após o envio dos parâmetros, verificamos se a pessoa já existe ou não. Caso não exista, retorna-se uma mensagem de erro e finaliza-se o processo. Se a pessoa não existir, então chamamos o método 'cadastrarEndereco()', que verifica se o endereço já está cadastrado. Caso exista, utiliza o método 'associaEndereco()' para associar o endereço existente à pessoa por meio do ID. Caso não exista, o método realiza o cadastro do novo endereço e, logo associa-o à pessoa.

A última verificação é referente ao tipo de pessoa (Administrador, Músico), onde utilizamos os métodos 'salvarAdministrador()' para gravar os dados da pessoa como um administrador, ou o método 'salvarMusico()' para gravar os dados da pessoa como um músico. Em seguida, retorna-se uma mensagem de conclusão para o usuário, caso a pessoa tenha sido cadastrada no sistema. Caso haja erro, retorna-se uma mensagem indicando a razão do erro.

Vale salientar um detalhe importante, no qual o único atributo diferente entre um músico e um administrador são os instrumentos tocados pelo músico, que começa vazio.

O segundo refere-se ao caso de uso 'Gravar Música', que representa o registro de uma música produzida por uma banda ou por músicos independentes. Esse caso de uso exibe o registro dos dados básicos da música, verifica o uso dos instrumentos e aloca os músicos participantes. O único parâmetro solicitado neste método é 'banda', que pode ser nulo ou preenchido com o nome da banda. Quando temos uma banda, entendemos que é necessário

utilizar o método 'reunirMusico()' para incluir todos os músicos na banda. Quando não há uma banda, incluímos apenas um músico.

Após a inclusão do(s) músico(s), é hora de associar os instrumentos que cada músico irá tocar e verificar se esses instrumentos estão disponíveis por meio do método 'getInstrumento()'. Caso o instrumento não esteja disponível, uma mensagem de erro é retornada ao usuário. Se todos os músicos já estiverem com seus instrumentos, é hora de realizar a produção da música por meio do método 'produzirMusica()'. Após a produção, a última etapa é salvar a música usando o método 'salvarMusica()'.

9 ETAPA 8

Com base em tudo que foi feito durante as etapas, com enfâse na 5, 6 e 7, dois métodos principais do sistema foram implementados, CadastrarPessoa() e GravarMusica(). Foi seguido o padrão definido, onde a classe Controller recebe as requisições e usa das classes Service para a criação de objetos e realização da lógica de negócios. No código em questão foi implementado apenas as duas funções no Controller, que por sua vez teve a necessidade de implementar 4 services(administrador, endereço, música, músico. Foi definido também que cada service apenas teria a instância de seu respectivo repository.

Na implementação do CadastrarPessoa() o controller faz uso do administradorService e enderecoService, no qual seguindo a lógica dos diagramas anteriores verifica a existência do endereço primeiro, e caso não exista ele o cadastra no sistema. Após isso encaminha os dados para o administradorService que verifica a existência da pessoa no sistema e caso não exista verifica o tipo de pessoa a ser cadastrado. Assim ele cria a instância do Músico ou Administrador e salva no banco de dados através do administradorRepository. Também mostramos o código do EndereçoService, EndereçoRepository e AdministradorRepository.

A implementação de GravarMusica() segue a mesma lógica explicada acima, dessa vez o Controller usa do musicoService e musicaService para executar o método. Para facilitar a leitura do código, a passagem de parametros foi feita abstraindo as listas em Strings. Após receber os parâmetros, as Strings autores_musica, autores_letra e instrumentos são transformados em listas enquanto musicoQueGravou é transformado em uma tabela Hash(como explicado nas etapas anteriores). Utilizando o musicoService ocorre uma verificação para saber se cada músico está associado ao instrumento utilizado na música e caso não esteja faz a associação. Após a verificação o musicaService grava a música e retorna o resultado.

Por não se tratar do código completo, as funções das classes service foram especificadas em um diagrama de classes. As classes AtorAdministrador e AtorMusico são abstrações da interface gráfica onde representa as funcionalidades que cada ator terá acesso. Também é mostrado a ligação entre as classes de controller, service e repository. A classe controller é mostrada apenas com os atributos e não com os métodos, porém, a classe service é mostrada contendo todos os métodos.

10 CONCLUSÃO

O processo de modelagem de software para o sistema de gestão de uma gravadora foi uma jornada crucial para compreender e estruturar de forma eficiente as necessidades do projeto. Desde a identificação dos requisitos até a seleção das tecnologias e a representação dos elementos do domínio, cada etapa foi fundamental para a criação de uma base sólida para o desenvolvimento.

Além disso, os diagramas foram peças-chave na representação visual da estrutura e interações dentro do sistema de gestão da gravadora. Eles desempenham um papel crucial na compreensão do design e na comunicação das relações entre entidades, métodos e processos.

Ademais, a escolha de tecnologias como Spring Boot, Java, React e PostgreSQL foi estratégica, alinhando-se não apenas com os conhecimentos da equipe, mas também com as demandas identificadas. A arquitetura Model-View-Controller (MVC) foi adotada visando a separação clara de responsabilidades, facilitando a manutenção e expansão do sistema no futuro.

A integração entre as classes Controller, Service e Repository proporciona uma estrutura organizada, onde o Controller gerencia as requisições, o Service implementa as regras de negócio e o Repository interage diretamente com o banco de dados. Esta abordagem em camadas oferece flexibilidade e facilita a adaptação do sistema às demandas em constante evolução do mercado.

Em suma, este relatório servirá como guia para a implementação do sistema, visando transformar conceitos em um sistema funcional.