



**Universidade do Minho**  
Mestrado Integrado em Engenharia Informática

## **Unidade Curricular de Bases de Dados**

Ano Letivo de 2017/2018

## **MuDBa – Base de Dados Musical**

**Francisco José Moreira Oliveira, a78416**

**João Pedro de Moura Pereira e Ferreira Carvalho, a79073**

**Raul Vilas Boas, a79617**

**Vitor Emanuel Carvalho Peixoto, a79175**

Novembro, 2017

**BD**

Data de Recepção	
Responsável	
Avaliação	
Observações	

## **MuDBa – Base de Dados Musical**

**Francisco José Moreira Oliveira, a78416**

**João Pedro de Moura Pereira e Ferreira Carvalho, a79073**

**Raul Vilas Boas, a79617**

**Vitor Emanuel Carvalho Peixoto, a79175**

Novembro, 2017

## Resumo

Para este trabalho foi-nos pedido uma base de dados relacional de um tema à nossa escolha. Optamos então por escolher fazer um sistema de base de dados (SBD) de uma plataforma de armazenamento de informação acerca de músicas.

Inicialmente tivemos problemas debatendo a organização da nossa base de dados, tendo em conta a ideia inicialmente proposta e debatemos o quanto complexa seria a nossa implementação do SBD. Após assentarmos as ideias e chegarmos a uma conclusão do que queríamos fazer, começamos por contextualizar o problema numa situação do dia-a-dia, dando-lhe uma utilidade.

De seguida elaboramos um método de levantamento de requisitos para a elaboração do projeto. Após este passo prosseguimos para a construção do modelo conceitual, com as entidades, atributos e relações corretas. Esse modelo foi convertido para lógico e posteriormente físico, sendo sempre validado recorrendo aos métodos mais corretos.

Foram implementadas restrições para melhorar o funcionamento da base de dados, interrogações para comprovar o seu correto funcionamento e outros passos essenciais na construção de uma base de dados relacional.

No fim elaboramos uma breve conclusão acerca do trabalho realizado e daquilo que poderá ser o futuro desta base de dados.

**Área de Aplicação:** Catálogo de músicas online (“IMDb” da Música).

**Palavras-Chave:** Sistema de base de dados, Base de dados, Base de dados relacional, Música, Artista, Álbum, Gravação, Género, Banda, Entidade, Atributo, Relacionamento, Requisito, Modelo Conceitual, Modelo Lógico, Modelo Físico, Normalização, Trigger, Função, Vista, Transição, Interrogação, SQL, MySQL Workbench,

# Índice

1. Definição do Sistema	1
1.1. Contexto de aplicação do sistema	1
1.2. Fundamentação da implementação da base de dados	1
1.3. Análise da viabilidade do projeto	1
2. Levantamento e Análise de Requisitos	2
2.1. Método de levantamento e de análise de requisitos adotado	2
2.2. Requisitos levantados	2
2.2.1. Requisitos de descrição	2
2.2.1. Requisitos de exploração	3
2.2.3. Requisitos de controlo	3
2.3. Análise geral dos requisitos	4
3. Modelação Concetual	5
3.1. Abordagem de modelação realizada	5
3.2. Entidades	5
3.3. Relacionamentos	5
3.4. Associações de atributos com entidades e relacionamentos	6
3.5. Generalização ou especialização de entidades	7
3.6. Diagrama ER	7
3.7. Validação do modelo de dados com o utilizador	8
4. Modelação Lógica	9
4.1. Construção e validação do modelo de dados lógico	9
4.1.1. Entidades	9
4.1.2. Atributos	11
4.1.3. Relacionamentos	11
4.2. Desenho do modelo lógico	11
4.3. Validação do modelo através da normalização	12
4.4. Atributos redundantes	13
4.5. Validação com interrogações do utilizador	14
4.6. Validação do modelo com transações	16
4.7. Revisão do modelo lógico com o utilizador	17
5. Implementação Física	18
5.1. Seleção do sistema de gestão de bases de dados	18
5.2. Tradução do esquema lógico para o SGBD	18
5.3. Tradução das interrogações do utilizador para SQL	19
5.4. Tradução das transações estabelecidas para SQL	20
5.5. Escolha, definição e caracterização de índices em SQL	22
5.6. Espaço em disco da base de dados e taxa de crescimento	22

5.7. Definição e caracterização dos <i>triggers</i>	24
5.8. Definição e caracterização das vistas de utilização em SQL	25
5.9. Definição e caracterização dos mecanismos de segurança em SQL	26
5.10. Revisão do sistema implementado com o utilizador	27
6. Conclusões e Trabalho Futuro	28

## **Anexos**

I. Anexo 1 - Resultado do Levantamento de Requisitos	32
--	----

# Índice de Figuras

<b>Figura 1</b> – Diagrama ER.	7
<b>Figura 2</b> – Conversão para o modelo lógico das entidades fortes.	10
<b>Figura 3</b> – Conversão para o modelo lógico das entidades fracas.	10
<b>Figura 4</b> – Integridade referencial.	11
<b>Figura 5</b> – Modelo Lógico.	11
<b>Figura 6</b> – Atributo ‘duracao’ nas entidades Gravação e Faixa.	13
<b>Figura 7</b> – Árvore de consulta da <i>query extra</i> em álgebra relacional.	14
<b>Figura 8</b> – Árvore de consulta da 1 <sup>a</sup> <i>query</i> em álgebra relacional.	15
<b>Figura 9</b> – Árvore de consulta da 2 <sup>a</sup> <i>query</i> em álgebra relacional.	15
<b>Figura 10</b> – Árvore de consulta da 3 <sup>a</sup> <i>query</i> em álgebra relacional.	16
<b>Figura 11</b> – Trajetórias das transações no Diagrama.	17

# Índice de Tabelas

<b>Tabela 1</b> – Tabela das entidades.	5
<b>Tabela 2</b> – Tabela dos relacionamentos.	5
<b>Tabela 3</b> – Tabela dos atributos.	6
<b>Tabela 4</b> – 1 <sup>a</sup> Forma normal.	12
<b>Tabela 5</b> – 2 <sup>a</sup> Forma normal.	13
<b>Tabela 6</b> – Transações e as respetivas operações sobre tabelas	20
<b>Tabela 7</b> – Espaço ocupado pelo povoamento inicial.	23
<b>Tabela 8</b> – Estimativa de crescimento do espaço em disco.	23
<b>Tabela 9</b> – Privilégios dos utilizadores.	26

# **1. Definição do Sistema**

## **1.1. Contexto de aplicação do sistema**

Num mundo onde a digitalização da informação é um tópico que tem apresentado um crescimento exponencial nos últimos anos, a necessidade de ter toda a informação em todo o lado é cada vez mais imperativa.

Na sociedade de hoje, a Internet é utilizada maioritariamente por jovens, que são um público bastante exigente em termos de informação disponível na Internet.

Este público dedica bastantes horas do seu dia a ouvir música em todo o tipo diferente de plataformas, em qualquer sítio, a qualquer hora. Assim torna-se evidente que há uma necessidade deste público ter à sua disposição toda a informação que achem relevante acerca de uma música, artista ou álbum.

Para além do público mais jovem esta plataforma de informação musical atrai também o público mais adepto de música ou fã de algum artista específico e que sente necessidade de saber mais sobre determinada música, artista ou álbum.

## **1.2. Fundamentação da implementação da base de dados**

A plataforma terá disponível diversas informações relativas ao mundo da música, incluindo os álbuns, as faixas, os artistas, incluindo bandas e os seus integrantes e diversas outras informações.

A nossa ideia surgiu do facto de haver uma necessidade no mercado de uma plataforma de informações musicais que seja robusta, confiável e dedicada especificamente à música. Esta plataforma faria com que o público jovem interagisse mais com o que está por detrás de uma música: os seus autores, a sua história e que há mais do que apenas uma música. Há álbuns inteiros que merecem ser escutados e conhecidos e novos artistas por descobrir. Isto irá levar a um conhecimento mais abrangente do mundo da música na nossa perspetiva e possivelmente acabar com a pirataria no mundo da música.

## **1.3. Análise da viabilidade do projeto**

Esta base de dados teria uma aplicação, do nosso ponto de vista, mais rentável, numa plataforma online onde pudéssemos aceder aos dados a qualquer hora e em qualquer lugar, sendo que o lucro viria de publicidade ou de ferramentas *premium* da plataforma online. Esta base de dados será mantida pelos próprios artistas, onde podem dar a conhecer melhor as suas músicas, com mais informação e organizada de uma maneira intuitiva. O facto de serem os artistas a manterem as suas informações atualizadas irá levar à inscrição de novos artistas que terão uma maior facilidade em serem descobertos e lançados no mundo da música. Esta base de dados é mantida também por um administrador.

## 2. Levantamento e Análise de Requisitos

### 2.1. Método de levantamento e de análise de requisitos adotado

O nosso método de levantamento de requisitos adotado foi um inquérito online (Anexo 1) com várias questões acerca de como devemos estruturar e preencher uma base de dados de músicas. A esse inquérito obtivemos respostas de 20 pessoas espalhadas pelo país, de ambos os sexos e com a maioria das idades a variar entre os 16 e os 25 anos. Sendo esta faixa etária o principal público-alvo deste projeto, podemos concluir que este levantamento de requisitos é bastante fidedigno relativamente ao que os nossos clientes irão querer ver num projeto deste tipo.

### 2.2. Requisitos levantados

#### 2.2.1. Requisitos de descrição

Os requisitos de descrição estão ligados à Linguagem de Descrição de Dados (DDL) que é usada para definir a estrutura da base de dados.

Assim sendo, pelas opiniões obtidas no nosso questionário e pela nossa opinião pessoal, os requisitos de descrição recolhidos junto dos inquiridos foram os seguintes:

- Deverão ser identificados os Artistas, Gravações (álbuns) e Faixas (músicas), sendo caracterizados pelos seus respetivos IDs, características relativas às suas informações e outras informações que achamos coerente utilizar dado os resultados do questionário de levantamento de requisitos:
  1. Para **Artista**: Nome; Biografia; Localidade; Início e Fim;
  2. Para **Gravação**: Título; Ano; Descrição; Pontuação; Duração; Tipo e Género;
  3. Para **Faixa**: Nome; Duração; Número; Álbum e Artista.
- A base de dados terá uma estrutura mais coerente se houver uma separação das características distintas de uma banda e um artista *solo*, numa relação de especialização de Artista. A Banda e o Artista *Solo* serão caracterizados por:
  1. Para **Banda**: Nº de membros;
  2. Para **Artista Solo**: Nascimento; Falecimento; Sexo.
- Outro requisito verificado no questionário foi informação acerca dos membros das bandas, então será necessário caracterizar os membros com atributos relativos às suas informações:
  1. Para **Membro**: Nome; Nascimento; Falecimento; Sexo; Localidade e Instrumento.
- Deverá ser possível consultar as informações relativas a qualquer um dos objetos da base de dados e facilmente criar relações entre qualquer objeto da base de dados que esteja, obviamente, relacionado, tais como:
  1. Devem haver atributos suficientes para que seja possível a exploração desejada da base de dados, sendo possível relacionar qualquer entidade com outra e consultar informação, podendo esta ser suscetível a filtragem, seleção ou ordenação tendo em conta os atributos exigidos pelo utilizador.

## 2.2.2. Requisitos de exploração

Os requisitos de exploração envolvem a Linguagem de Manipulação de Dados (DML) cujos comandos são usados para a gestão da base de dados.

Neste caso os requisitos de exploração irão assumir o papel mais importante no SGBD visto esta ser uma base de dados cuja estrutura está desenhada para ser estática, numa fase inicial, mas o conteúdo dessa mesma estrutura está em constante alteração. Assim, verificando as opiniões obtidas no questionário, podemos elaborar a seguinte lista de requisitos:

- Sendo esta base de dados baseada na consulta de informações acerca do mundo da música é essencial ser possível **selecionar** e **visualizar** dados da base de dados. O sistema tem de estar preparado para serem efetuadas consultas personalizadas de informação feita pelo utilizador, dependendo do que o utilizador quer visualizar. Algumas das principais questões dos utilizadores relativamente à da base de dados serão:
  - a. Qual o álbum com maior pontuação de uma determinada banda?
  - b. Qual o artista que lançou determinada música?
  - c. Quais as gravações lançadas num determinado ano pela banda X e qual o nome do tipo de gravação?
  - d. Quais as músicas lançadas por todos os artistas de um determinado país, por ordem alfabética?
  - e. Quais as músicas das bandas cujo guitarrista tem o ID = X?

Estas são apenas algumas das questões que os utilizadores acham que o sistema deve ser capaz de responder.

- Visto que o mundo da música está em constante mudança, com músicas novas, bandas novas, novos membros, fins de bandas, etc., é também importante poder **inserir** e **alterar** dados da base de dados.

## 2.2.3. Requisitos de controlo

Os requisitos de controlo funcionam com base na Linguagem de Controlo de Dados (DCL) que serve para atribuir permissões e privilégios a objetos da base de dados.

Esta base de dados tem como propósito ser uma plataforma aberta a qualquer pessoa, no entanto, para evitar perda de dados por malícia ou engano, teremos de a proteger, bloqueando acessos a “desconhecidos” a determinadas funcionalidades na base de dados.

- Assim, um **visitante** terá liberdade total para visualizar todos os dados da base de dados, no entanto está impedido de adicionar, apagar ou modificar informação, que será atribuída aos artistas e ao administrador.
- O **administrador** terá acesso total à base de dados, seja apagar, adicionar, selecionar ou modificar informação.
- O **artista** será o principal responsável por atualizar a informação relativa a si, podendo adicionar álbuns ou faixas, remover ou adicionar membros, caso se trate de uma banda, modificar informações acerca do seu projeto, etc. No entanto estará impedido de remover álbuns, faixas e afins dado que esse tipo de conteúdo não pode ser “apagado da história” desse artista.

### **2.3. Análise geral dos requisitos**

Após especificar os requisitos para este SGBD obtidos através de um inquérito realizado, podemos concluir que estes requisitos serão bastante informativos acerca do propósito deste projeto e darão a informação que os utilizadores identificaram como essencial numa plataforma deste tipo, seja numa perspetiva de organização da base de dados, conteúdo presente, estruturação dos dados, ou permissões e privilégios dos mesmos.

Assim sendo, apresentamos os requisitos recolhidos junto do público inquirido, tendo o resultado sido aprovado. Podemos então avançar para a modelação conceitual do projeto.

### 3. Modelação Conceitual

#### 3.1. Abordagem de modelação realizada

Após termos recolhidos os requisitos abordados no capítulo anterior, chega a altura de começar a desenhar a estrutura da base de dados. Para isso vamos proceder à modelação conceitual. A modelação conceitual é um passo essencial para a criação do SBD, pois ajuda-nos a identificar entidades, atributos e relações.

#### 3.2. Entidades

Após recolher os objetos no levantamento de requisitos temos de verificar quais vão ser considerados entidades. Logicamente, dado o conceito deste projeto, as entidades principais serão: Artista; Gravação e Faixa. Optamos por criar novas entidades também para criar uma relação de especialização entre Artista e os seus derivados: Banda e Artista Solo, sendo que a banda pode ser composta por diversos membros, também caracterizados numa entidade.

Entidades	Descrição
Artista	Contém informação do artista e pode controlar informação relativa a si (adicionar álbuns, remover membros, etc.)
Banda	Contém toda a informação específica de uma banda.
Artista Solo	Contém toda a informação específica de um artista solo.
Gravação	Contém toda a informação relativa a uma gravação.
Faixa	Contém toda a informação relativa a uma faixa.
Membro	Contém toda a informação de um membro de uma banda.

Tabela 1 – Tabela das entidades.

#### 3.3. Relacionamentos

Para o nosso SBD funcionar temos de elaborar os relacionamentos entre as diferentes entidades abordadas no capítulo anterior, tendo em conta os requisitos levantados.

Entidade	Cardinalidade	Relação	Cardinalidade	Entidade
Gravação	1	Contém	N	Faixa
Artista	1	Compila	N	Gravação
Artista Solo	1	É	1	Artista
Banda	1	É	1	Artista
Banda	N	Composta por	N	Membros

Tabela 2 – Tabela dos relacionamentos.

### 3.4. Associações de atributos com entidades e relacionamentos

Após definir as entidades e os seus relacionamentos nesta BD, vamos abordar os atributos relativos a cada entidade ou relacionamento. Os atributos foram definidos pelos requisitos levantados no capítulo anterior.

Visto que nenhuma entidade apresenta um atributo único que seja capaz de a caracterizar, iremos atribuir um atributo “ID” a cada entidade (exceto Banda e Solo, cujas chaves primárias serão explicadas no capítulo seguinte) de modo a usá-lo como chave primária dessa entidade. No nosso modelo conceitual não iremos ter nenhum atributo composto ou multivalor.

Entidade	Atributo	Descrição	Tipo	PK	N
Faixa	ID	Número identificativo de uma faixa.	Inteiro	X	
	Nome	Nome da faixa.	Caracteres		
	Duração	Duração em tempo dessa faixa.	Tempo		
	Número	Número dessa faixa na gravação.	Inteiro		
Gravação	ID	Número identificativo da gravação.	Inteiro	X	
	Título	Título da gravação.	Caracteres		
	Ano	Ano de lançamento.	Ano		
	Descrição	Breve descrição da gravação.	Caracteres		
	Pontuação	Pontuação da crítica à gravação.	Inteiro		
	Duração	Duração em tempo da gravação.	Tempo		
	Tipo	Tipo de gravação (Álbum, EP, etc.).	Caracteres		
	Género	Género musical (Pop, Rock, Indie, etc.).	Caracteres		
Artista	ID	Número identificativo do artista.	Inteiro	X	
	Nome	Nome do artista.	Caracteres		
	Biografia	Curta biografia do artista.	Caracteres		
	Localidade	Local de nascimento/fundação do artista.	Caracteres		
	Início	Início de carreira/fundação da banda.	Ano		
	Fim	Fim de carreira/dissolução da banda.	Ano		X
Artista Solo	Sexo	Género da individualidade.	Caracter (M/F)		
	Nascimento	Data de nascimento do artista.	Data		
	Falecimento	Data de falecimento do artista.	Data		X
Banda	Nº membros	Número de membros da banda.	Inteiro		
Membro	ID	Número identificativo do membro.	Inteiro	X	
	Nome	Nome do membro da banda.	Caracteres		
	Instrumento	Instrumento que toca.	Caracteres		
	Sexo	Género do membro.	Caracter (M/F)		
	Localidade	Localidade onde nasceu.	Caracteres		
	Nascimento	Data de nascimento do membro.	Data		
	Falecimento	Data de falecimento do membro.	Data		X

PK – primary key  
N – (pode ser) nulo

**Tabela 3** – Tabela dos atributos.

### 3.5. Generalização ou especialização de entidades

O nosso SBD poderia estar simplesmente orientado por uma única entidade Artista. No entanto, como detetado no levantamento de requisitos, o nosso público-alvo prefere ter presentes informações detalhadas, que especifiquem o tipo de artista – banda ou solo – e informações acerca dos membros dessas bandas. Podemos assim estabelecer uma relação de classe–subclasse entre Artista-Banda e Artista-Artista Solo. Para trabalhar esta relação de especialização de Artista, as entidades especializadas vão herdar a chave primária da entidade principal (*Primary Key Inheritance*).

### 3.6. Diagrama ER

Após termos recolhido todos os requisitos para a base de dados e apontadas todas as entidades, relações e atributos, podemos prosseguir para o desenho do diagrama ER.

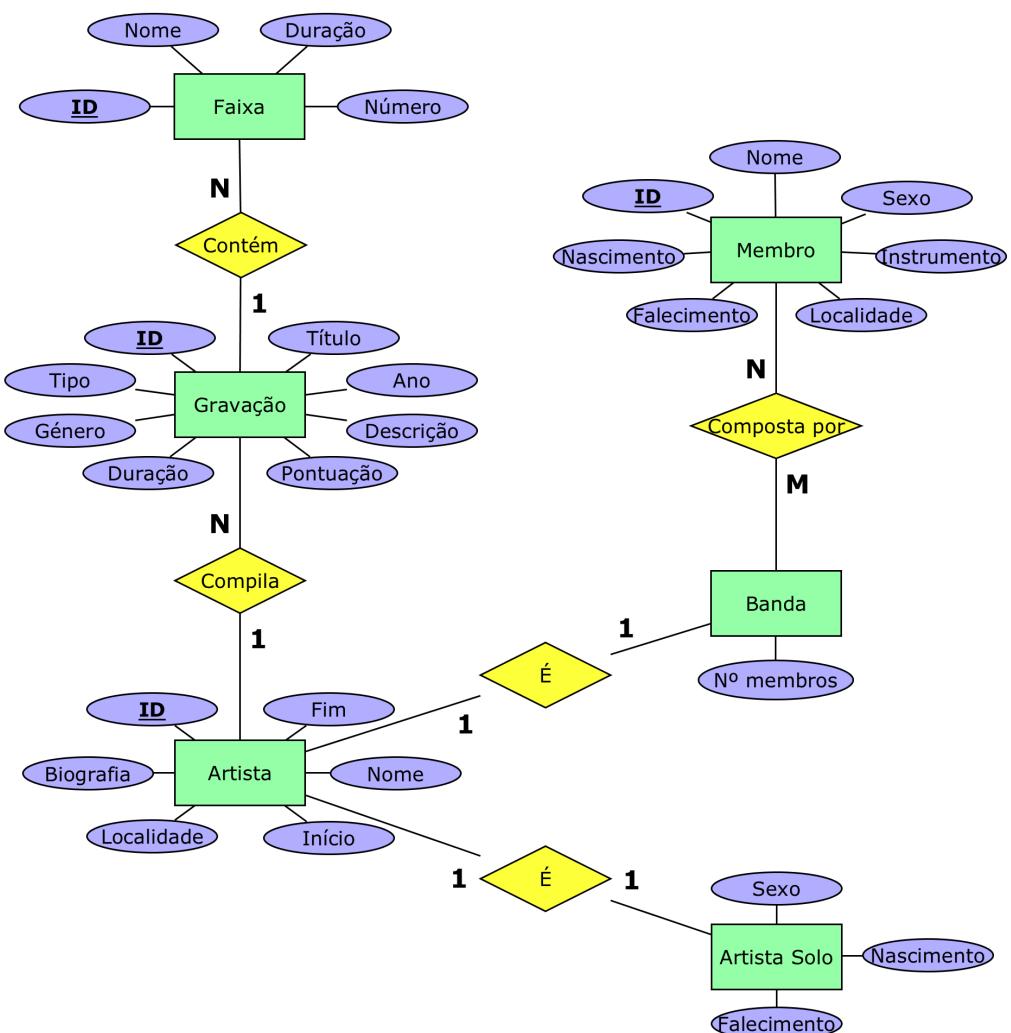


Figura 1 – Diagrama ER.

### **3.7. Validação do modelo de dados com o utilizador**

Após concluir a modelação do SBD, é necessário ver se o sistema desenvolvido é capaz de responder aos requisitos levantados inicialmente. Este passo é muito importante porque vai determinar a qualidade do modelo desenvolvido através da sua capacidade de resposta aos seus requisitos.

Analizando os requisitos levantados pelos utilizadores no inquérito e o nosso modelo conceitual do problema, somos capazes de concluir que o modelo é uma representação fidedigna dos requisitos recolhidos inicialmente, pelo que o apresentamos ao público-alvo, tendo este concordado com os resultados obtidos até esta fase, pelo que podemos avançar para a modelação lógica do problema.

## 4. Modelação Lógica

### 4.1. Construção e validação do modelo de dados lógico

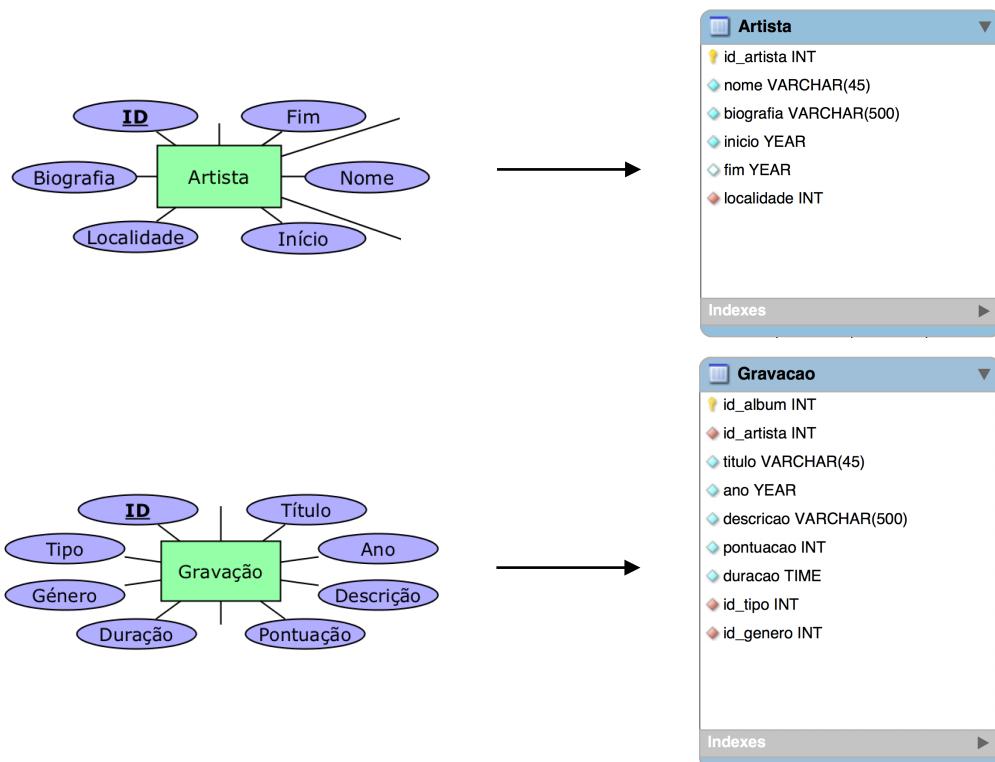
Após a construção e validação do modelo de dados conceitual, podemos avançar para a construção do modelo de dados lógico. Esta é a segunda fase da criação de uma base de dados. A modulação conceitual criada no capítulo anterior é adaptada e melhorada para um modelo lógico, orientado aos objetos.

Ao longo do desenvolvimento vamos validando o modelo de acordo com os requisitos de utilizador.

#### 4.1.1. Entidades

As entidades podem ser divididas em dois tipos, entidades fracas e fortes. Uma entidade é considerada forte se não depender da existência de outra entidade [1]. As ocorrências dessa entidade são unicamente identificáveis através da sua chave primária. Por outro lado, uma entidade fraca é dependente de uma outra entidade [1].

- **Entidades Fortes:** Na nossa base de dados há 4 entidades consideradas fortes, sendo elas: Artista; Gravação; Faixa e Membro. A seguir apresentamos a transformação dessas entidades em tabelas do modelo lógico:



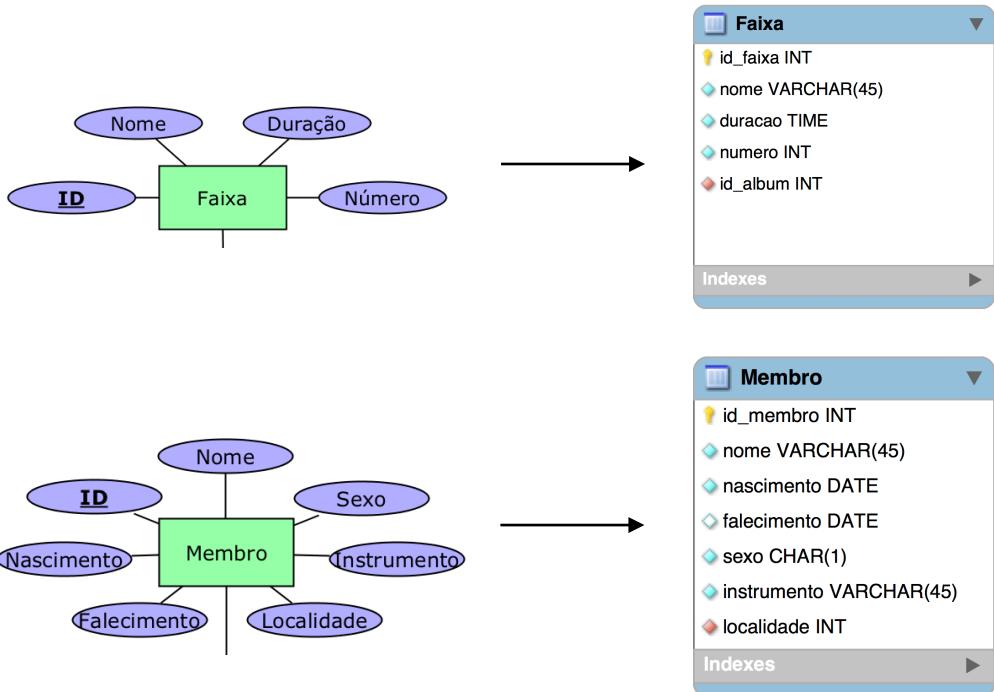


Figura 2 – Conversão para o modelo lógico das entidades fortes.

- Entidades Fracas:** Nesta base de dados há 2 entidades fracas: Banda e Artista Solo. Estas entidades, apesar de possuírem uma chave primária, são dependentes da entidade Artista que lhes herda a sua chave primária.

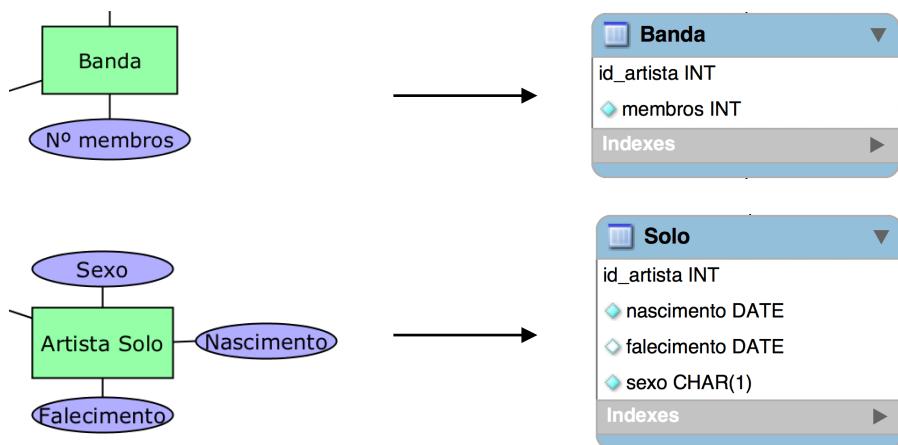


Figura 3 – Conversão para o modelo lógico das entidades fracas.

Um dos métodos usados para garantir exatidão e consistência de dados num SBD são as restrições de integridade. Essas restrições evitam repetição de ocorrências de um atributo causados por incoerências ou erros gramaticais (Ex: em duas ocorrências de Membro dizer que um tem localidade em “Lisboa, Portugal” e o outro em “Lisboa”). Para otimizar então a nossa base de dados, admitimos que os atributos Tipo e Género, da entidade Gravação, e o atributo Localidade, das entidades Membro e Artista, serão entidades, com uma chave primária (ID) e um nome que a descreve, sendo que esses atributos serão agora representados pela sua chave primária nas entidades onde antes estavam.

Este tipo de restrição de integridade é conhecido por Integridade Referencial [2].

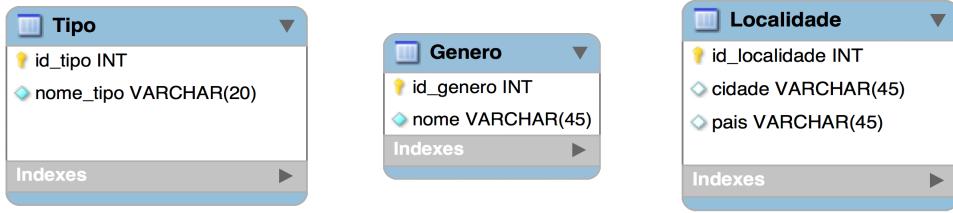


Figura 4 – Integridade referencial.

#### 4.1.2. Atributos

Os atributos de cada entidade foram os mesmos escolhidos para o modelo lógico, adaptando as respetivas chaves primárias e estrangeiras. Os tipos de cada atributo foram também definidos de modo a representar fidedignamente a informação, mantendo também a integridade dos mesmos<sup>1</sup>.

#### 4.1.3. Relacionamentos

É agora importante estabelecer os relacionamentos existentes entre tabelas. No caso das entidades geradas para haver integridade referencial, todos os seus relacionamentos serão de 1:N, tal como os relacionamentos entre Artista, Gravação e Faixa. Sendo Artista Solo e Banda especializações de Artista, os relacionamentos entre as subentidades e a entidade será 0..1:1 visto que Artista pode ser Solo ou Banda. Por fim, o relacionamento entre Membro e Banda será de N:M, visto que uma banda pode ter vários membros e um membro também pode pertencer a várias bandas.

### 4.2. Desenho do modelo lógico

Assim, concluída a conversão das entidades conceituais para as tabelas lógicas e criadas as relações entre entidades, podemos avançar com um modelo lógico:

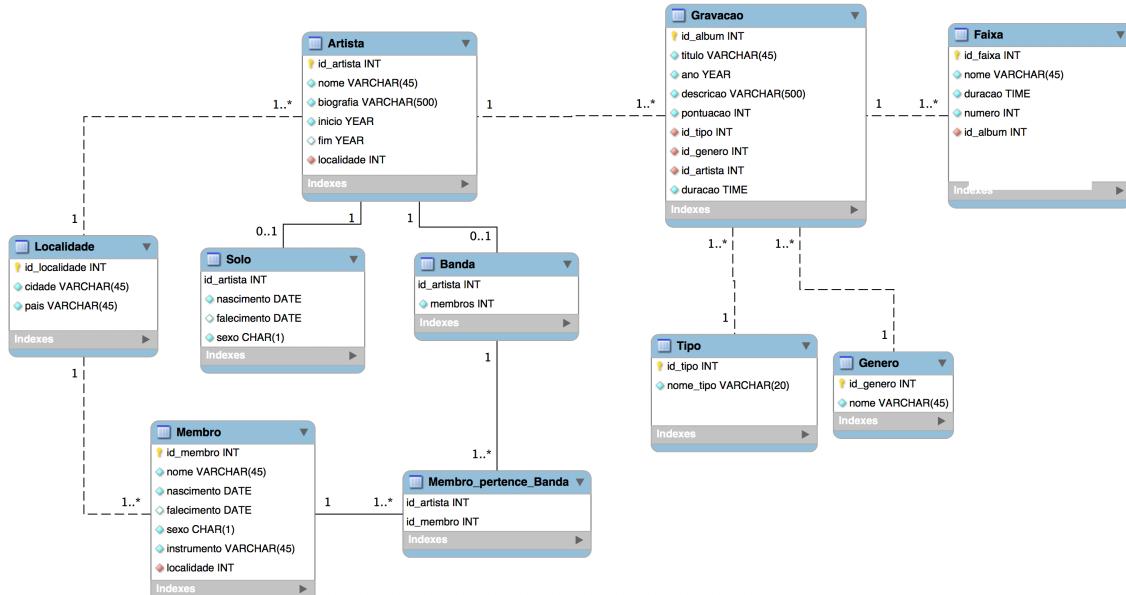


Figura 5 – Modelo Lógico.

<sup>1</sup> Atributos com textos grandes como por exemplo biografia em Artista ou descrição em Gravação foram definidos como varchar erroneamente, tendo este erro sido detetado numa fase terminal do trabalho, não podendo assim ser corrigido atempadamente.

### 4.3. Validação do modelo através da normalização

Normalização é um processo que serve para reduzir a redundância entre atributos e reduzir os atributos ao mínimo possível de modo a corresponder aos requisitos apenas.

Para validar o modelo através da normalização, verificamos o seguinte conjunto de requisitos: identificação das dependências funcionais (DF), 1<sup>a</sup> forma normal (1NF), 2<sup>a</sup> forma normal (2NF), 3<sup>a</sup> forma normal (3NF).

- **Dependências funcionais:** no nosso modelo lógico todas as tabelas apresentam uma única chave primária. Assim, podemos definir as seguintes dependências funcionais:

1. Artista:

id\_artista ® nome, biografia, inicio, fim, localidade

2. Solo:

id\_artista ® nascimento, falecimento, sexo

3. Banda:

id\_artista ® nome, biografia, inicio, fim, localidade

4. Membro:

id\_membro ® nome, nascimento, falecimento, sexo, instrumento, localidade

5. Localidade:

id\_localidade ® cidade, pais

6. Gravação:

id\_album ® titulo, ano, descricao, pontuacao, id\_tipo, id\_genero, id\_artista, duracao

7. Faixa:

id\_faixa ® nome, duracao, numero, id\_album

- **1<sup>a</sup> forma normal:** nesta regra, uma relação diz-se normalizada se não houverem grupos repetidos na tabela (se houver atomicidade na tabela). Um grupo repetido é um atributo, ou grupo de atributos, numa tabela que ocorrem com vários valores para uma única ocorrência da chave [2]. Assim, pode-se dizer que o modelo está normalizado segundo a 1<sup>a</sup> forma normal, dado que uma inserção de várias gravações, por exemplo, do mesmo artista irá criar várias ocorrências com a chave do artista sempre desagrupada.

ID Álbum	ID Artista	Título	Ano	...
9	8	Isn't Anything	1988	...
12	8	Loveless	1991	...
42	8	mbv	2013	...

**Tabela 4 – 1<sup>a</sup> Forma normal.**

- **2<sup>a</sup> forma normal:** na segunda forma normal, todos os atributos não chave devem depender unicamente da chave primária dessa tabela [2]. No nosso SBD, todos os atributos não chave dependem da chave primária dessa tabela e todo o tipo de informação que achamos relevante não dependente dessa chave estão noutras entidades, representadas pelas suas chaves primárias na entidade como chaves estrangeiras. Também não existem tabelas com chaves primárias compostas, logo o modelo está validado pela 2<sup>a</sup> forma normal.

No exemplo seguinte vemos como está organizada a tabela Gravação, relativamente à identificação dos artistas que compilaram gravações:

ID Álbum (PK)	ID Artista (FK)	Título	Ano	...
12	8	Loveless	1991	...
5	4	Parachutes	2000	...

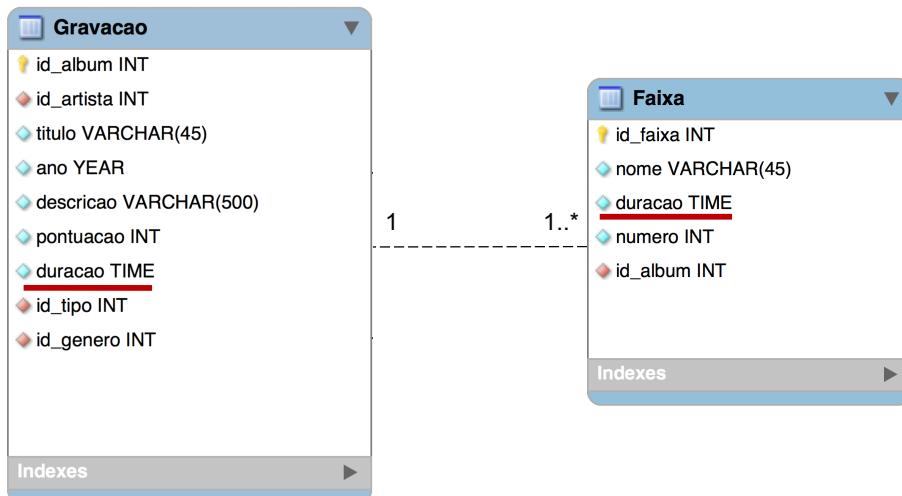
  

ID Artista (PK)	Título	...
4	Coldplay	...
8	My Bloody Valentine	...

**Tabela 5 – 2ª Forma normal.**

- **3ª forma normal:** na terceira forma normal, todos os atributos não chave de uma tabela são independentes entre si e nenhum pode ser obtido por uma equação entre atributos dependentes da mesma chave primária, eliminando assim quaisquer redundâncias existentes [2].  
Após uma análise da nossa base de dados encontramos uma dependência funcional que origina redundância na relação entre Gravação e Faixa.  
No nosso modelo lógico, poderia haver alguma dúvida relativamente ao atributo Falecimento em Solo e Fim de carreira em Artista, visto que ambos partilham a mesma chave primária e o fim de carreira depende do falecimento do artista solo, no entanto, o fim de carreira não depende do falecimento do artista.  
Assim sendo, conclui-se que o modelo se encontra normalizado até à 3ª forma normal.

#### 4.4. Atributos redundantes



**Figura 6 – Atributo ‘duracao’ nas entidades Gravação e Faixa.**

Ao prestarmos atenção aos atributos de ambas as entidades, reparamos que o atributo duração em Gravação é redundante, visto que pode ser obtido na tabela Faixa através da soma dos atributos 'duracao' de todas as faixas que têm o id\_album igual.

Apesar de não causar problemas na normalização, o atributo duração em Gravação é assim redundante e pode ser removido dessa entidade, tendo acabado por ser mesmo removido.

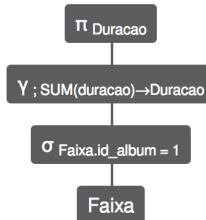
## 4.5. Validação com interrogações do utilizador

Aqui optamos por selecionar as interrogações propostas nos requisitos que consideramos mais exigentes em termos de exploração das tabelas, visto que estas irão comparar mais atributos de diferentes tabelas do que as interrogações não selecionadas. Se conseguirmos responder a estas interrogações selecionadas utilizando a álgebra relacional, pode-se admitir que o modelo está validado através das interrogações do utilizador. Optamos por adicionar uma interrogação (*query*) extra para provar a redundância presente na entidade Gravação, detetada no capítulo anterior.

- **Query extra: Qual a duração total do álbum com ID = 1?**

Esta query serve também para provar o que vimos no capítulo anterior, onde o atributo Duração em Gravação é redundante e pode ser obtido através da soma das durações das faixas que integram esse álbum. Esta query pode ser traduzida para álgebra relacional na seguinte expressão e respetiva árvore de consulta:

$$\pi_{\text{Duracao}} \gamma_{\text{Faixa}, \text{SUM}(\text{duracao}) \rightarrow \text{Duracao}} \sigma_{\text{Faixa.id\_album} = 1} (\text{FAIXA})$$

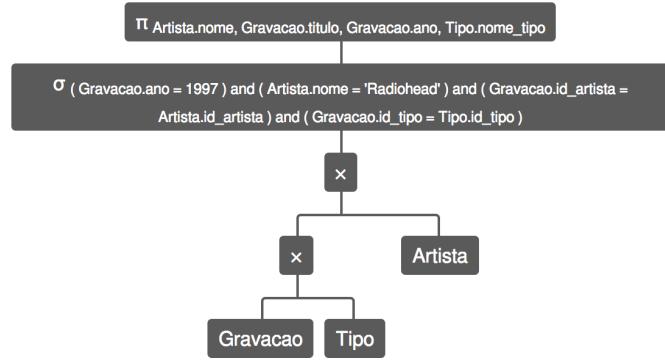


**Figura 7 – Árvore de consulta da query extra em álgebra relacional.**

- **1ª Query: Quais as gravações lançadas em 1997 pela banda Radiohead (sem conhecimento do seu ID) e qual o nome do tipo de gravação?**

Esta query pode ser traduzida para álgebra relacional na seguinte expressão e respetiva árvore de consulta:

$$\pi_{\text{Artista.nome, Gravacao.titulo, Gravacao.ano, Tipo.nome_tipo}} (\sigma_{\text{Gravacao.ano} = '1997' \wedge \text{Artista.nome} = 'Radiohead'} \wedge \text{Gravacao.id_artista} = \text{Artista.id_artista} \wedge (\text{Gravacao.id_tipo} = \text{Tipo.id_tipo})) (\text{GRAVACAO} \times \text{TIPO} \times \text{ARTISTA})$$

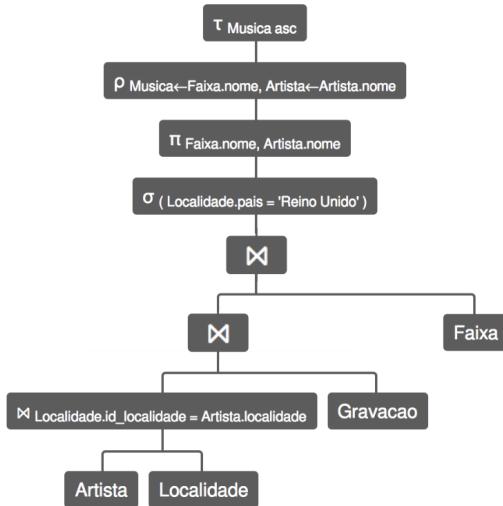


**Figura 8 – Árvore de consulta da 1ª query em álgebra relacional.**

- **2ª Query: Quais as músicas lançadas por todos os artistas de um determinado país, por ordem alfabética?**

Esta query pode ser traduzida para álgebra relacional na seguinte expressão e respetiva árvore de consulta:

$$\tau_{Musica} \rho_{Musica \leftarrow Faixa.nome, Artista \leftarrow Artista.nome} \pi_{Faixa.nome, Artista.nome} (\sigma_{Localidade.pais = 'Reino Unido'}$$

$$LOCALIDADE \bowtie_{Localidade.id_localidade = Artista.localidade} (ARTISTA \bowtie (GRAVACAO \bowtie FAIXA))$$


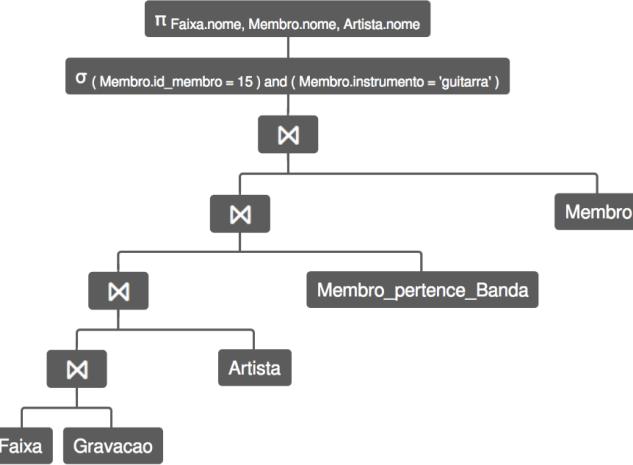
**Figura 9 – Árvore de consulta da 2ª query em álgebra relacional.**

- **3ª Query: Quais as músicas das bandas constituídas pelo guitarrista com ID = 15?**

Esta query pode ser traduzida para álgebra relacional na seguinte expressão e respetiva árvore de consulta:

$$\pi_{Faixa.nome, Membro.nome, Artista.nome} (\sigma_{Membro.id_membro = 15 \wedge Membro.instrumento = 'guitarra'} (FAIXA \bowtie$$

$$GRAVACAO \bowtie ARTISTA \bowtie MEMBRO_PERTENCE_BANDA \bowtie MEMBRO))$$



**Figura 10 – Árvore de consulta da 3<sup>a</sup> query em álgebra relacional.**

As expressões em álgebra linear foram revistas pela ferramenta *RelaX* [3], que também gerou automaticamente as árvores de consulta. Assim sendo, e verificada a capacidade de resposta do SBD às interrogações impostas pelo utilizador nos requisitos de exploração, podemos inferir que este é válido.

## 4.6. Validação do modelo com transações

Nesta validação vamos verificar se o modelo relacional responde às transações requisitadas, verificando no modelo lógico a sua capacidade. Para tal vamos definir três transações de três tipos diferentes: inserção, alteração e interrogações de dados.

- a) **Inserir detalhes de uma nova gravação:** esta transação de inserção de uma nova gravação inicia-se na sua respetiva tabela, inserindo o título do álbum, ano de lançamento, uma curta descrição, a sua pontuação, tipo de gravação, género musical e o artista que compilou a gravação.
- b) **Atualizar detalhes de um artista:** esta transação altera os dados do registo de um artista, presentes na tabela Artista. Pode alterar o seu nome, biografia, início de carreira, fim de carreira, ou a sua localidade.
- c) **Remover membro de banda quando este falece:** nesta transação, os detalhes do membro estão na tabela Membro. Verificada a informação de que este membro faleceu, o seu ID vai ser removido da tabela Membro\_pertence\_Banda.

Assim, podemos representar no diagrama o caminho de cada transação:

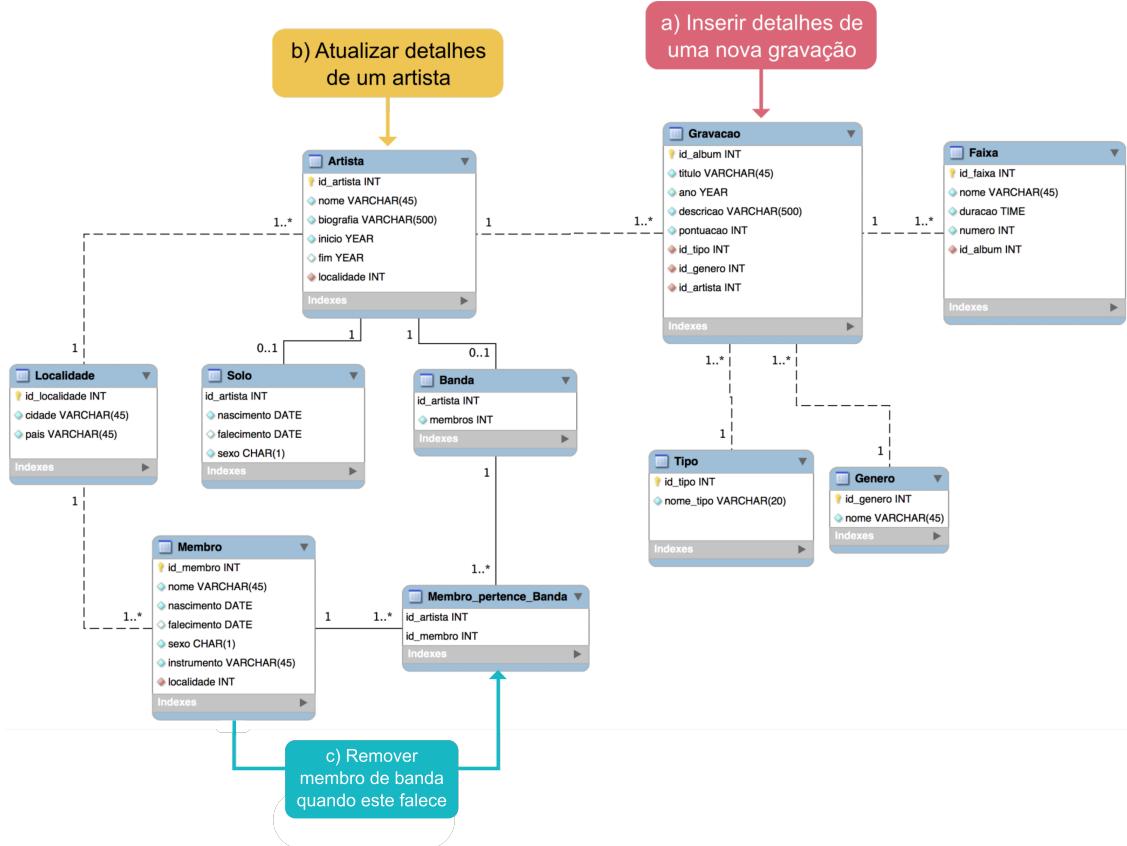


Figura 11 – Trajetórias das transações no Diagrama.

## 4.7. Revisão do modelo lógico com o utilizador

Após concluir a estruturação do modelo lógico relacional e a sua respetiva validação pelos três métodos diferentes, chegou a altura de confrontar o utilizador com a nossa estrutura. Esta etapa irá assegurar que o modelo construído está de acordo com os requisitos e não terá falhas que poderão causar problemas no futuro [1].

Assim sendo, após rever o modelo com o utilizador, este foi determinado correto, na medida em que responde aos requisitos do utilizador.

## 5. Implementação Física

### 5.1. Seleção do sistema de gestão de bases de dados

Temos agora de escolher um SGBD que se adapte às necessidades da nossa base de dados e que seja uma boa aposta na sua manutenção futura.

Optamos então pelo MySQL da Oracle. Esta escolha foi parcialmente influenciada por ser este o sistema de gestão de bases de dados relacional usado nas aulas da UC de Bases de Dados, facilitando a nossa adaptação e mais rápida aprendizagem do sistema.

Contudo elaboramos uma pesquisa para perceber se existiria por ventura um outro sistema que se adequasse melhor às nossas necessidades, tendo chegado à conclusão que o MySQL seria a escolha ideal, por diversos motivos.

Em primeiro lugar está desenhado e otimizado para aplicações Web, estando sempre atualizado com o crescimento da quantidade de dados na internet, segundo um *Whitepaper* da Oracle [4]. Esta característica joga a nosso favor, visto que este SBD tem como objetivo ser aplicado numa plataforma online, disponível 24 horas por dia, em qualquer lugar.

Outro fator que consideramos importante é o facto de ser um sistema cuja licença é gratuita e que não necessita de hardware avançado ao contrário do Microsoft SQL Server, segundo esse mesmo *Whitepaper*.

Tendo em conta que este é um projeto desenvolvido por estudantes numa fase ainda empreendedora, o facto de não ter custos elevados é um aspeto bastante importante na escolha do SGBD.

O facto de estar disponível nos três principais sistemas operativos (Microsoft Windows, Apple MacOS e Ubuntu Linux) é importante também nesta escolha.

No entanto, um dos principais fatores é a aplicação MySQL Workbench, uma ferramenta visual que permite alterar, desenvolver e administrar a base de dados, capaz de desenhar um diagrama ER e rapidamente o projetar numa base de dados física através do comando “Forward Engineer”, entre outras ferramentas que, se efetuadas manualmente, iriam necessitar de muito tempo e esforço [4]. Aspeto essencial, dado o facto de sermos iniciantes na criação e gestão de bases de dados.

Por estes motivos decidimos adotar o MySQL para a realização deste projeto. Não podemos afirmar que, num projeto futuro, este seria este o nosso SGBD, porque teríamos de analisar qual se adaptaria melhor ao caso específico, mas certamente seria uma das opções a considerar.

### 5.2. Tradução do esquema lógico para o SGBD

O sistema de gestão de bases de dados escolhido (MySQL) tem uma aplicação (MySQL Workbench) que permite desenhar o diagrama lógico relacional da base de dados e efetuar ações que em outros SGBD levariam algum tempo e esforço. Uma das ações disponíveis é o “Forward Engineer” (*Synchronize Model* para a versão MacOS). Esta opção cria automaticamente um script SQL, onde estão as criações das tabelas, com os seus atributos e definições das chaves, e executa-o, criando o modelo físico da base de dados.

### 5.3. Tradução das interrogações do utilizador para SQL

Para esta etapa, usaremos as *queries* propostas pelo utilizador que foram selecionadas e aprovadas recorrendo à álgebra relacional. Essas *queries* serão agora expressas na linguagem SQL:

- **Query extra: qual a duração total do álbum com ID = 1?**

Nesta query tivemos de usar as funções *time\_to\_sec* e *sec\_to\_time* dado que a soma de um dado do tipo *TIME* daria a soma desse número como um só (Ex: 00:01:30 + 00:00:45 = 0130 + 0045 = 175). As funções irão converter a duração em segundos, efetuar a soma e depois converter novamente para o formato *TIME*.

```
SELECT sec_to_time(SUM(time_to_sec(duracao))) AS Duração
FROM Faixa f
WHERE f.id_album = 1;
```

- **1<sup>a</sup> Query: quais as gravações lançadas em 1997 pela banda Radiohead (sem conhecimento do seu ID) e qual o nome do tipo de gravação?**

```
SELECT a.nome, g.titulo, g.ano, t.nome_tipo
FROM Gravacao g, Tipo t, Artista a
WHERE (g.ano = '1997')
AND (a.nome = 'Radiohead')
AND (g.id_artista = a.id_artista)
AND (g.id_tipo = t.id_tipo);
```

- **2<sup>a</sup> Query: quais as músicas lançadas por todos os artistas de um determinado país (seja Reino Unido, por exemplo), por ordem alfabética?**

```
SELECT f.nome AS Musica, a.nome AS Artista
FROM Artista a
INNER JOIN Localidade AS l ON l.id_localidade = a.localidade
INNER JOIN Gravacao AS g ON g.id_artista = a.id_artista
INNER JOIN Faixa AS f ON f.id_album = g.id_album
WHERE (l.pais = 'Reino Unido')
ORDER BY Musica;
```

- **3<sup>a</sup> Query: quais as músicas das bandas constituídas pelo guitarrista com ID = 15?**

```
SELECT f.nome, m.nome, a.nome
FROM Faixa f
INNER JOIN Gravacao AS g ON g.id_album = f.id_album
INNER JOIN Artista AS a ON a.id_artista = g.id_artista
INNER JOIN Membro_pertence_Banda AS mpb ON mpb.id_artista = a.id_artista
INNER JOIN Membro AS m ON mpb.id_membro = m.id_membro
WHERE (m.id_membro = 15)
AND (m.instrumento = 'guitarra');
```

Todas estas *queries* foram executadas no MySQL Workbench, tendo os resultados obtidos sido os esperados, pelo que podemos afirmar que tanto a implementação do sistema como a tradução das *queries* para a linguagem SQL estão corretos.

## 5.4. Tradução das transações estabelecidas para SQL

Uma transação é definida por um grupo de operações que apresentam as seguintes propriedades: atomicidade, consistência, isolamento e durabilidade (Propriedades ACID). A atomicidade de uma transação é essencial, pois irá impedir a transação caso haja um erro ocorrido numa fase seguinte. Assim ou todas ou nenhuma operação é aplicada [5].

As transações foram abordadas no capítulo 4.5, sendo elas:

- (A) Inserir detalhes de uma nova gravação**
- (B) Atualizar detalhes de um artista**
- (C) Remover membro de banda quando este falece**

Para analisar as transações, podemos dividir esta etapa em dois passos essenciais [1]:

- **Mapear as transações numa tabela de relações:**

	(A)				(B)				(C)			
	I	R	U	D	I	R	U	D	I	R	U	D
<b>Artista</b>							X					
<b>Gravação</b>	X											
<b>Membro</b>											X	
<b>M_p_B</b>												X
<b>Banda</b>											X*	

I – Insert; R – Read; U – Update; D – Delete; M\_p\_B – Tabela Membro\_pertence\_Banda;

\* – Este update é feito pelo trigger a)

**Tabela 6** – Transações e as respetivas operações sobre tabelas

- **Transações em SQL:**

### (A) Inserir detalhes de uma nova gravação

```
CREATE PROCEDURE inserirGravacao (IN id_album INT,
                                    IN titulo VARCHAR(45),
                                    IN ano YEAR,
                                    IN descricao VARCHAR(500),
                                    IN pontuacao INT,
                                    IN id_tipo INT,
                                    IN id_genero INT,
                                    IN id_artista INT)

BEGIN
    DECLARE erro BOOL DEFAULT 0;
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
    START TRANSACTION;
        INSERT INTO gravacao VALUES (id_album, titulo, ano, descricao, pontuacao,
                                       id_tipo, id_genero, id_artista);
    IF erro
        THEN ROLLBACK;
    ELSE COMMIT;
    END IF;
END
```

**(B) Atualizar detalhes de um artista**

```
CREATE PROCEDURE atualizaArtista (IN nID INT,
                                    IN nNome VARCHAR(45),
                                    IN nBiografia VARCHAR(500),
                                    IN nInicio YEAR,
                                    IN nFim YEAR,
                                    IN nLocalidade INT)

BEGIN
DECLARE erro BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
START TRANSACTION;
    UPDATE Artista a SET a.nome = nNome,
                        a.biografia = nBiografia,
                        a.inicio = nInicio,
                        a.fim = nFim,
                        a.localidade = nLocalidade
    WHERE a.id_artista = nID;
IF erro
THEN ROLLBACK;
ELSE COMMIT;
END IF;
END
```

**(C) Remover membro de banda quando este falece - O triggera** (capítulo 5.7) irá ser disparado

aqui, visto que vamos alterar o atributo falecimento de um membro, atualizando o número de membros da banda na tabela Banda.

```
CREATE PROCEDURE removeMembro (IN memID INT,
                                IN memFalecimento DATE)

BEGIN
DECLARE erro BOOL DEFAULT 0;
DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET erro = 1;
START TRANSACTION;
    UPDATE Membro m SET m.falecimento = memFalecimento
    WHERE id_membro = memID;
    DELETE FROM Membro_pertence_Banda
    WHERE id_membro = memID;
IF erro
THEN ROLLBACK;
ELSE COMMIT;
END IF;
END
```

Estas transações foram todas testadas, tendo sido comprovado o seu correto funcionamento.

## 5.5. Escolha, definição e caracterização de índices em SQL

As nossas tabelas irão estar definidas pelos índices já declarados na modelação lógica (chaves primárias) e não achamos necessária a definição de mais índices, dado que os que estão definidos são suficientes para executar qualquer interrogação proposta pelo utilizador.

## 5.6. Espaço em disco da base de dados e taxa de crescimento

Um dos aspectos essenciais é retribuir os requerimentos em termos de disco que esta base de dados irá exigir para o utilizador saber se o seu *hardware* consegue suportar a implementação física da base de dados e se conseguirá suportar tendo em conta o crescimento do número de informação na base de dados. Para este capítulo vamos estimar o espaço ocupado por cada linha de cada tabela existente nesta base de dados e o espaço total dado o povoamento inicial.

Tabela	Coluna	Tipo	Tamanho [6]	Tamanho p/ linha	Linhas	Espaço Total
Faixa	id_faixa	INT	4	62	51	3 162
	nome	VARCHAR(45)	45			
	duração	TIME	5			
	numero	INT	4			
	id_album	INT	4			
Gravação	id_album	INT	4	566	5	2 830
	titulo	VARCHAR(45)	45			
	ano	YEAR	1			
	descricao	VARCHAR(500)	500			
	pontuacao	INT	4			
	id_tipo	INT	4			
	id_genero	INT	4			
	id_artista	INT	4			
Artista	id_artista	INT	4	555	6	3 330
	nome	VARCHAR(45)	45			
	biografia	VARCHAR(500)	500			
	inicio	YEAR	1			
	fim	YEAR	1			
	localidade	INT	4			
Solo	id_artista	INT	4	11	1	11
	nascimento	DATE	3			
	falecimento	DATE	3			
	sexo	CHAR(1)	1			
Banda	id_artista	INT	4	8	5	40
	membros	INT	4			

Membro	id_membro	INT	4	105	21	2 205
	nome	VARCHAR(45)	45			
	nascimento	DATE	3			
	falecimento	DATE	3			
	sexo	CHAR(1)	1			
	instrumento	VARCHAR(45)	45			
	localidade	INT	4			
Membro pertence Banda	id_artista	INT	4	8	21	168
	id_membro	INT	4			
Tipo	id_tipo	INT	4	24	4	96
	nome_tipo	VARCHAR(20)	20			
Género	id_genero	INT	4	49	10	490
	nome	VARCHAR(45)	45			
Localidade	id_localidade	INT	4	94	17	1 598
	cidade	VARCHAR(45)	45			
	país	VARCHAR(45)	45			

**Tabela 7** – Espaço ocupado pelo povoamento inicial.

Assim, podemos concluir o espaço total ocupado pelo povoamento inicial, que é a soma da última coluna:

$$3162 + 2830 + 3330 + 11 + 40 + 2205 + 168 + 96 + 490 + 1598 = \mathbf{13\,930\ bytes}$$

Tendo em perspetiva as seguintes taxas de crescimento:

Tabela	Agora		Daqui a 1 ano			Daqui a 5 anos		
	Linhas	Tamanho	Cresc	Linhas	Tamanho	Cresc	Linhas	Tamanho
Faixa	51	3 162	6x	306	18 972	4x	1 224	75 888
Gravação	5	2 830	6x	30	16 980	4x	120	67 920
Artista	6	3 330	5x	30	16 650	2x	60	33 300
Solo	1	11	5x	5	55	2x	10	110
Banda	5	40	5x	25	200	2x	50	400
Membro	21	2 205	5x	105	11 025	2x	210	22 050
M_p_B	21	168	5x	105	840	2x	210	1 680
Tipo	4	96	1x	4	96	1x	4	96
Género	10	490	1.1x	11	539	1x	11	539
Localidade	17	1 598	2x	34	3 196	1.5x	51	4 794
<b>TOTAL (bytes)</b>	--	<b>13 930</b>	--	--	<b>68 553</b>	--	--	<b>108 507</b>

**Tabela 8** – Estimativa de crescimento do espaço em disco.

Podemos então admitir um resultado fixo de 13 930 Bytes para o espaço ocupado em disco pelo povoamento inicial e as estimativas de 68 553 Bytes daqui a 1 ano e 108 507 Bytes daqui a 5 anos.

## 5.7. Definição e caracterização dos triggers

Os *triggers* são essenciais para atualizar informações de colunas quando uma coluna da qual dependem é alterada ou para criar restrições na inserção de dados. No nosso caso podemos definir alguns *triggers* úteis:

- a) Atualizar o número de membros de uma banda quando um elemento falece:

```
DELIMITER $$  
CREATE TRIGGER atualizaMembros  
AFTER UPDATE ON Membro  
FOR EACH ROW  
BEGIN  
IF NEW.falecimento  
THEN  
UPDATE Banda, Membro_pertence_Banda  
SET membros = membros - 1  
WHERE (Banda.id_artista = Membro_pertence_Banda.id_artista)  
AND (Membro_pertence_Banda.id_membro = NEW.id_membro);  
END IF;  
END; $$
```

- b) Declarar fim de carreira quando o artista solo falece:

```
DELIMITER $$  
CREATE TRIGGER fimCarreiraSolo  
AFTER UPDATE ON Solo  
FOR EACH ROW  
BEGIN  
IF NEW.falecimento  
THEN  
UPDATE Artista SET Artista.fim = YEAR(NEW.falecimento)  
WHERE (NEW.id_artista = Artista.id_artista);  
END IF;  
END; $$
```

- c) Impedir que um Artista Solo seja também uma Banda:

```
DELIMITER $$  
CREATE TRIGGER limitaBanda  
BEFORE INSERT ON Banda  
FOR EACH ROW  
BEGIN  
DECLARE msg VARCHAR(200);  
IF (NEW.id_artista IN  
(SELECT id_artista FROM Solo  
WHERE Solo.id_artista = NEW.id_artista))  
THEN SET msg = 'Um Artista Solo não pode ser Banda!';  
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;  
END IF;  
END; $$
```

d) Impedir que uma Banda seja também Artista Solo:

```
DELIMITER $$  
CREATE TRIGGER limitaSolo  
BEFORE INSERT ON Solo  
FOR EACH ROW  
BEGIN  
DECLARE msg VARCHAR(200);  
IF (NEW.id_artista IN  
(SELECT id_artista FROM Banda  
WHERE Banda.id_artista = NEW.id_artista))  
THEN SET msg = 'Uma Banda não pode ser Artista Solo!';  
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = msg;  
END IF;  
END; $$
```

Todos os *triggers* foram testados no MySQL Workbench, onde foi comprovado o seu correto funcionamento.

## 5.8. Definição e caracterização das vistas de utilização em SQL

As vistas são importantes na otimização de consultas. Estas são muito úteis na otimização e/ou simplificação de consultas, visto que estas tabelas irão conter a junção de algumas tabelas.

Neste caso podemos definir algumas vistas bastante úteis. Uma delas é uma vista que relate as bandas e os seus membros numa única tabela. Esta vista irá auxiliar a 3ª query, poupano-lhe uma junção, tornando o código mais simples e de fácil entendimento.

```
CREATE VIEW membrosBanda AS  
SELECT m.id_membro,  
m.nome AS membro,  
m.nascimento,  
m.sexo,  
m.instrumento,  
a.id_artista,  
a.nome AS banda,  
a.biografia,  
a.inicio,  
a.fim  
FROM Artista a, Membro_pertence_Banda mpb, Membro m  
WHERE a.id_artista = mpb.id_artista  
AND mpb.id_membro = m.id_membro;
```

Podemos também definir uma vista para mostrar todos os álbuns Rock. Esta vista seria mais dedicada aos utilizadores da plataforma mais inclinados para certos estilos musicais.

```
CREATE VIEW albunsRock AS  
SELECT *  
FROM gravacao g  
WHERE g.id_genero=1;
```

Ambas as vistas foram testadas e provado o seu correto funcionamento. Mais vistas poderiam ser criadas futuramente, após ser estudada a vantagem da sua implementação.

## 5.9. Definição e caracterização dos mecanismos de segurança em SQL

Os mecanismos de segurança são um dos pontos mais importantes no que toca à boa manutenção de uma base de dados, pois uma falha neste sistema, poderia levar à perda da totalidade dos dados. Um dos mecanismos de segurança mais usados está relacionado com a DCL, Linguagem de Controlo de Dados. Esta linguagem permite atribuir **privilégios e permissões** a cada utilizador da base de dados dependendo do papel que este pode/deve efetuar na base de dados, seja de manutenção, administração ou apenas consulta de uma determinada tabela.

Para tal, definimos três utilizadores do sistema e as suas permissões de controlo de dados na seguinte tabela:

Tabela	Administrador				Artista				Visitante			
	S	I	U	D	S	I	U	D	S	I	U	D
Faixa	X	X	X	X	X	X			X			
Gravação	X	X	X	X	X	X			X			
Artista	X	X	X	X	X			X	X			
Solo	X	X	X	X	X			X	X			
Banda	X	X	X	X	X			X	X			
Membro	X	X	X	X	X	X	X	X	X			
M_p_B	X	X	X	X	X	X	X	X	X			
Tipo	X	X	X	X	X				X			
Género	X	X	X	X	X				X			
Localidade	X	X	X	X	X				X			

**Tabela 9 – Privilégios dos utilizadores.**

O administrador será o “dono” do sistema, tendo absoluto controlo em tudo.

Os artistas serão os responsáveis pela manutenção da informação da sua banda, tendo que adicionar álbuns sempre que os lança, registar novos membros, se for o caso, alterar dados do artista, etc.

O visitante é o utilizador comum da plataforma que quer efetuar uma consulta dos dados disponíveis na plataforma.

Assim sendo, podemos passar à implementação dos utilizadores e respetivas permissões em SQL, usando para isso uns pequenos exemplos. Neste primeiro iremos demonstrar a criação dos utilizadores:

```

CREATE USER 'admin'@'localhost';
      SET PASSWORD FOR 'admin'@'localhost' = 'mudbadmin';

CREATE USER 'artista1'@'localhost';
      SET PASSWORD FOR 'artista1'@'localhost' = 'artista1';
-- repetir para cada artista

CREATE USER 'visitante'@'localhost';
      SET PASSWORD FOR 'visitante'@'localhost' = 'visit';

```

Agora, temos de atribuir os respetivos privilégios para cada utilizador:

```
GRANT ALL PRIVILEGES ON mudba.* TO 'admin'@'localhost';

GRANT SELECT, UPDATE ON mudba.Artista TO 'artista1'@'localhost';
REVOKE INSERT, DELETE ON mudba.Artista FROM 'artista1'@'localhost';
-- repetir o processo de GRANT e REVOKE para as outras tabelas.

GRANT SELECT ON mudba.* TO 'visitante'@'localhost';
REVOKE UPDATE, INSERT, DELETE ON mudba.* FROM 'visitante'@'localhost';
```

Deste modo, temos um sistema com uma correta distribuição de privilégios.

Outro mecanismo de segurança essencial numa base de dados é o **backup de dados**. O backup deve ser efetuado periodicamente e/ou sempre que alguma alteração é efetuada. Este pode ser facilmente executado pelo MySQL Workbench, pois está habilitado com um botão que guarda um ficheiro .sql com a criação das tabelas, povoamento, triggers, vistas, etc.

## 5.10. Revisão do sistema implementado com o utilizador

Após concluirmos que a implementação do modelo físico da base de dados relacional respeita os requisitos impostos pelo utilizador, fizemos uma abordagem relativamente ao modelo físico.

O utilizador, considerou o modelo produzido uma representação fidedigna dos requisitos levantados inicialmente, pelo que o sistema foi aprovado pelo utilizador.

## 6. Conclusões e Trabalho Futuro

Concluída a validação do sistema podemos considerar finalizada a realização do projeto e analisar os aspectos positivos e negativos que retiramos da realização do trabalho.

Numa fase inicial observamos a viabilidade do projeto e levantamos os requisitos junto do utilizador. Esta fase inicial é uma fase muito importante visto ser aqui que analisámos a necessidade ou não da execução do projeto. Do nosso ponto de vista, a abordagem neste capítulo foi a mais correta visto que recolhemos os requisitos junto de pessoas reais e que se enquadram dentro da faixa etária para a qual esta base de dados está projetada.

O processo de construção do modelo conceitual também foi bem executado na nossa opinião, tendo as escolhas das entidades, atributos e relações sido as mais corretas na nossa opinião.

Na conversão para modelo lógico efetuamos todos os passos corretos, excetuando no tipo dos atributos definidos com 500 caracteres que deveriam ter sido considerados do tipo *text* e não *varchar* mas que não foram corrigidos a tempo por termos encontrado essa falha em cima da hora. Efetuamos as validações para o modelo lógico, tendo em conta as regras da normalização e concluído este passo prosseguimos para a modelação física usando o MySQL. Nesta etapa do projeto fizemos o povoamento da base de dados com informação útil para a mesma e criamos restrições para o correto funcionamento do sistema. As transações definidas pelo utilizador foram executadas, dando os resultados previstos. Definimos vistas para auxiliar consultas de dados e os mecanismos de segurança, tais como os utilizadores do sistema. Todos estes passos foram executados de uma maneira correta, do nosso ponto de vista. Calculamos então o espaço ocupado em disco pelo povoamento inicial e uma perspetiva de crescimento nos próximos anos, acertada no nosso ponto de vista.

Este modelo foi completamente revisto pelo utilizador que considerou ser o mais correto para os requisitos levantados inicialmente, pelo que concluímos o projeto.

Numa fase futura, poderão ser adicionadas mais tabelas para melhoria do sistema já existente, bem como novos atributos e relações para aumentar a quantidade de informação disponível. Uma adição interessante também no futuro seria a inserção de mais dados para o povoamento, visto que a base está pouco povoada atualmente.

Nesta fase final, podemos concluir como pontos positivos, que o projeto foi essencial na aprendizagem de novos conceitos que não abordamos nas aulas da UC e consolidação dos já adquiridos. Tendo os pontos negativos sido algumas pequenas falhas (como a troca dos tipos *text* e *varchar*) e outras que possam ter ocorrido na modelação do sistema.

No geral a nossa avaliação ao nosso desempenho no trabalho e ao resultado final é positiva.

## Referências

- [1] T. Connolly e C. Begg, *Database Systems: A Practical Approach to Design, Implementation and Management*, 6<sup>a</sup> ed., Essex: Pearson, 2015.
- [2] F. Gouveia, *Fundamentos de Bases de Dados*, 1<sup>a</sup> ed., Lisboa: FCA - Editora de Informática, 2014.
- [3] J. Kessler, "RelaX - Relational Algebra Calculator," [Online]. Available: <http://dbis-uibk.github.io/relax/>. [Acedido em 23 11 2017].
- [4] Oracle, "Top 10 Reasons to Choose MySQL for Web-based Applications," A MySQL Strategy Whitepaper, 2011.
- [5] Microsoft Corporation, "What is a Transaction," [Online]. Available: [https://msdn.microsoft.com/en-us/library/aa366402\(VS.85\).aspx](https://msdn.microsoft.com/en-us/library/aa366402(VS.85).aspx). [Acedido em 26 11 2017].
- [6] D. Anastasia e M. Goméz, *SI654: Database Application Design*, 1<sup>a</sup> ed., Michigan: Computer Aided Engineering Network, 2004.
- [7] MySQL, "Data Type Storage Requirements," [Online]. Available: <https://dev.mysql.com/doc/refman/5.7/en/storage-requirements.html#data-types-storage-reqs-innodb>. [Acedido em 26 11 2017].

## **Lista de Siglas e Acrónimos**

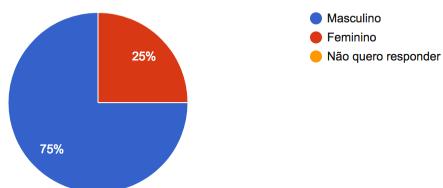
BD	Base de Dados
SGBD	Sistema de Gestão da Base de Dados
SBD	Sistema de Base de Dados
DDL	Linguagem de Descrição de Dados (Data Definition Language)
DML	Linguagem de Manipulação de Dados (Data Manipulation Language)
DCL	Linguagem de Controlo de Dados (Data Control Language)
PK	Chave Primária (Primary Key)
NN	Não Nulo (Not Null)
MV	Multivalor
1NF	1 <sup>a</sup> Forma Normal
2NF	2 <sup>a</sup> Forma Normal
3NF	3 <sup>a</sup> Forma Normal
DF	Dependência Funcional
UC	Unidade Curricular

## **Anexos**

# I. Anexo 1 - Resultado do Levantamento de Requisitos

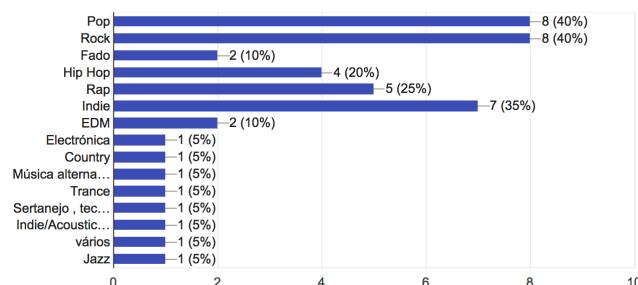
## Sexo

20 respostas



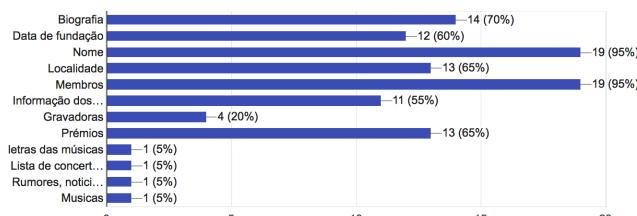
## De que tipo de música gostas?

20 respostas



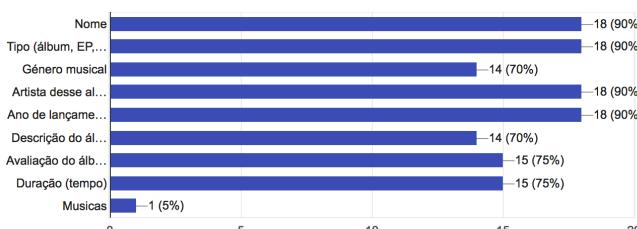
## Seleciona as informações acerca de uma BANDA que aches essencial a aplicação ter:

20 respostas



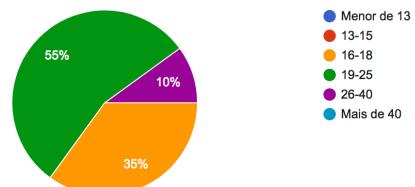
## Seleciona as informações acerca de um ÁLBUM que aches essencial a aplicação ter

20 respostas



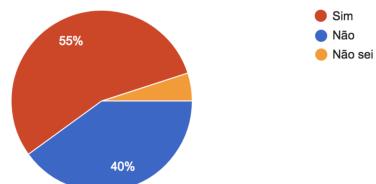
## Idade

20 respostas



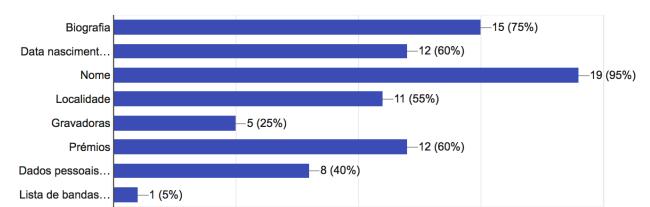
## Sentes falta de uma aplicação onde possas encontrar toda a informação de uma música, artista, álbum, etc?

20 respostas



## Seleciona as informações acerca de um ARTISTA INDIVIDUAL que aches essencial a aplicação ter:

20 respostas



## Seleciona as informações acerca de uma MÚSICA que aches essencial a aplicação ter:

20 respostas

