**CENTRO PAULA SOUZA**

**FATEC SANTANA DE PARNAÍBA**

**Análise e Desenvolvimento de Sistemas**

**Erick Gomes Barbosa**

**Gustavo Silva Souza**

**Lucas Mateus Braga de Souza**

**PROJETO DE BANCO DE DADOS**

**IEQ Parque**

**Santana de Parnaíba**

**2024**

**Erick Gomes Barbosa**

**Gustavo Silva Souza**

**Lucas Mateus Braga de Souza**

**PROJETO DE BANCO DE DADOS**

**IEQ Parque**

Trabalho universitário da disciplina “Laboratório de Banco de Dados”, ministrada pelo professor especialista Everson Nunes de Almeida da FATEC Santana de Parnaíba, com o intuito de estabelecer nota parcial da média final dos alunos.

**Santana de Parnaíba**

**2024**

RESUMO

Este trabalho aborda sobre o projeto de desenvolvimento de um sistema ERP para o ministério de louvor da igreja quadrangular IEQ Parque, que necessita desse sistema para organizar, integrar e informatizar seus processos. Tal documento traz à tona os problemas enfrentados pela igreja e como eles podem ser resolvidos computacionalmente, de forma que os processos genéricos da engenharia de software, como o levantamento de requisitos, a modelagem de diagramas e etc. estejam inclusos e sendo utilizados para melhor compreensão da análise, porém com ênfase na modelagem dos dados.

**Palavras-chave:** sistema ERP, ministério de louvor e modelagem de dados.

ABSTRACT

This paper discusses the project to develop an ERP system for the worship ministry of a Foursquare church, called IEQ Parque, which needs this system to organize, integrate and computerize its processes. This document highlights the problems faced by the church and how they can be solved computationally, so that generic software engineering processes such as requirements gathering, diagram modeling and so on are included and used to better understand the analysis, but with an emphasis on data modeling.

**Keywords:** ERP system, worship ministry and data modeling.

**SUMÁRIO**

[1 INTRODUÇÃO 6](#_Toc184658058)

[1.1 OBJETIVO 8](#_Toc184658059)

[1.2 OBJETIVOS ESPECÍFICOS 8](#_Toc184658060)

[2 PÚBLICO-ALVO 9](#_Toc184658061)

[3 REQUISITOS 9](#_Toc184658062)

[3.1 REQUISITOS DE NEGÓCIO 9](#_Toc184658063)

[3.2 REQUISITOS NÃO FUNCIONAIS 10](#_Toc184658064)

[3.3 REQUISITOS FUNCIONAIS 12](#_Toc184658065)

[4 DIAGRAMA DE CASO DE USO 14](#_Toc184658066)

[5 PROTÓTIPOS 16](#_Toc184658067)

[6 MODELAGEM DO BANCO DE DADOS 16](#_Toc184658068)

[6.1 MODELO CONCEITUAL 17](#_Toc184658069)

[6.2 MODELO LÓGICO 18](#_Toc184658070)

[6.3 MODELO FÍSICO 18](#_Toc184658071)

[6.4 DICIONÁRIO DE DADOS 18](#_Toc184658072)

[6.5 INSTÂNCIAS DAS TABELAS 18](#_Toc184658073)

[6.6 INSTRUÇÕES SQL 18](#_Toc184658074)

[6.6.1 VIEWS 18](#_Toc184658075)

[6.6.2 STORED PROCEDURES 18](#_Toc184658076)

[6.6.3 FUNCTIONS 18](#_Toc184658077)

[6.6.4 TRANSACTIONS 18](#_Toc184658078)

# INTRODUÇÃO

O ministério de louvor da igreja quadrangular IEQ Parque é responsável por executar ministrações e louvores durante cultos e eventos. Tal ministério está presente em todos os cultos da semana, sejam eles na própria igreja ou nas casas dos membros, como ocorre nos Grupos de Comunhão (GCs). Contudo, são muitos cultos, eventos e GCs por semana, então o ministério de louvor se organiza para que todos eles tenham ministrações e louvores.

A maneira à qual o ministério de louvor trabalha segue a estrutura convencional de cultos, que é dividida em três partes: (1) ministração, louvor e adoração, (2) pregação e (3) oração. No caso, o ministério atua nessas três seções, contudo de formas diferentes. Na primeira seção do culto, os levitas, como são chamados os músicos na igreja, ministram e executam louvores, enquanto os membros da igreja os acompanham em adoração, seja cantando ou dançando; na segunda seção, os levitas apenas fazem um fundo musical pouco perceptível para a pregação; na terceira seção, os levitas voltam a ministrar e tocar louvores, enquanto a os membros da igreja os acompanham, assim como na primeira seção, entretanto não há ênfase em adoração, e sim em oração.

Entre os levitas, existem três principais classificações: (1) levita, (2) ministro e (3) líder. O primeiro termo, levita, é o mais genérico para se referir aos músicos, pois todo músico da igreja, independente de sua função, é um levita; o segundo termo, ministro, refere-se apenas aos levitas que podem ministrar em cultos, ou seja, realizar pregações enquanto cantam. Em síntese, eles são os principais vocalistas de determinado culto, e podem cantar e ministrar durante ele; por fim, o terceiro e último termo se refere ao líder, que é quem gerencia o ministério de louvor. Aliás, é válido lembrar que pode existir mais de um líder no ministério de louvor.

Os músicos que atuam como levitas costumam ser aqueles que tocam instrumentos ou cantam, mas que não ministram. Isso ocorre porque é raro encontrar instrumentistas que também sejam ministros, já que, para exercer esse papel, é necessário tocar, cantar e ministrar simultaneamente. Além disso, cantores iniciantes, com frequência, não ministram devido à falta de experiência e vivência no mundo espiritual, então são apenas levitas até que cresçam espiritualmente.

Entre os ministros, todos sempre são cantores e, normalmente, com exceção a eventos raros, só há um único ministro por louvor. Logo, mesmo que haja outras pessoas com cargo de ministro cantando no mesmo louvor, no louvor desse culto, eles são apenas levitas, enquanto outra pessoa exerce o cargo de ministro.

Em relação à liderança, ela é selecionada anualmente pelos pastores e por outros membros da administração da igreja, no início de todo ano e na presença de todos os membros da igreja. Com o passar dos anos, a liderança pode se manter a mesma ou não, a depender da vontade dos pastores e da administração da igreja. Por fim, é importante lembrar que os líderes devem ser levitas, mas que não precisam ser ministros.

Existem diversos tipos de culto, e cada tipo de culto possui louvores diferentes. Os principais cultos da igreja englobam o culto da família, o culto de missões, o culto de intercessão, os GCs e o culto de jovens. Logo, para cada um deles, há algumas diferenças. No caso, cada tipo de culto pode possuir quantidade de músicas diferentes por louvor, como também um repertório de músicas diferente e levitas diferentes para tocar, pois há levitas que tocam apenas em determinados cultos.

Todo mês, a liderança cria escalas de louvor. Essas escalas servem para indicar quais são as datas e horários de cada culto que haverá no mês, como também para indicar quais levitas tocarão em cada um desses cultos e qual papel eles desempenharão nele. Essa escala é criada de acordo com a disponibilidade dos levitas, e caso um levita não possa tocar em determinado dia, ele deve avisar com um mês de antecedência para que o líder crie a escala sem o colocar em determinada data à qual ele está ocupado. De qualquer forma, caso haja algum imprevisto, o levita deve encontrar alguém para substituí-lo e, caso não encontre, o louvor é feito com uma pessoa a menos, mesmo que só haja instrumentistas, cantores ou nenhum dos dois.

A seleção das músicas de um culto é feita pelo ministro responsável pela ministração de tal culto. Ele deve enviar as músicas selecionadas para os levitas com, no mínimo, cinco dias de antecedência para que eles possam estudar as músicas. Aliás, caso haja algum imprevisto, ele ainda pode enviar as músicas, mas é punido pelo líder. Por fim, para cada música selecionada, o ministro deve informar o nome, a tonalidade, o autor, a versão (com link) e o BPM da música, como também a ordem de execução dela e, em caso de medley - termo que se refere a união de duas ou mais músicas em uma só -, o ministro deve informar quais músicas devem ser unidas, como elas serão unidas e até qual parte cada música será executada.

Alguns cultos, como o culto da família, o culto dos jovens e etc., possuem ensaios musicais para que os levitas pratiquem as músicas, esses que geralmente acontecem duas horas e meia antes do culto. No caso, o ministro envia a música com, no mínimo, cinco dias de antecedência, os levitas aprendem as músicas e, no dia do culto, algumas horas antes do louvor, os levitas ensaiam as músicas, já as tocando na ordem que serão executadas no louvor do culto.

De tempos em tempos, a liderança convoca todos os levitas para reuniões. Essas reuniões acontecem de forma esporádica, sem padrão de intervalos de data. Elas servem para que os levitas tenham comunhão entre si, alinhem situações corriqueiras e recebam palavras bíblicas próprias e específicas para levitas, igual a um culto normal. Inclusive, é comum que façam tudo isso comendo, e cada um leva algum item para a comemoração.

A adoção de novos membros para o ministério de louvor é feita anualmente. No caso, no início de cada ano, é ofertado um curso sobre adoração a Deus, que qualquer membro da igreja pode fazer. Após esse curso, os que participaram do curso e que quiserem adentrar ao ministério de louvor podem solicitar participar da avaliação musical, onde costumam performar músicas selecionadas pelo líder de louvor em frente a alguns jurados, que na maioria das vezes são os próprios levitas do ministério de louvor. Logo, caso sejam aprovados, podem fazer parte do ministério.

A expulsão ou punição de membros do ministério de louvor pode ocorrer caso o levita quebre as doutrinas da igreja. As consequências, além da expulsão definitiva, incluem meses fora das escalas de louvor, ou seja, sem tocar nos cultos da igreja.

## OBJETIVO

O objetivo desse trabalho é contribuir com a disseminação de soluções tecnológicas para as instituições religiosas, sem intenções de se obter lucro e sem discriminação.

## OBJETIVOS ESPECÍFICOS

Os objetivos específicos são de contribuir com uma solução tecnológica para simplificar e aumentar a eficácia da comunicação do ministério de louvor da Igreja Quadrangular Evangélica IEQ Parque.

# PÚBLICO-ALVO

Em geral, os envolvidos no problema são todos os membros da igreja que realizam atividades no ministério do louvor. Contudo, de forma mais específica, seguem as classificações:

* **Patrocinadores:** pessoa ou grupo que financia o projeto e que aprova ou não as suas funcionalidades. Neste projeto, o patrocinador é a igreja, que é representada pelos pastores, que compõem o cargo mais alto da instituição;
* **Usuários principais:** são os usuários diretamente envolvidos no problema. Neste projeto, os usuários principais são os membros do ministério de louvor;;
* **Usuários secundários:** são usuários que não estão diretamente envolvidos no problema, mas que se beneficiam do software. Neste projeto, os usuários secundários são todos os membros da igreja que não fazem parte do ministério de louvor;
* **Usuários potenciais:** são os usuários que atualmente não são, mas que podem vir a ser futuros usuários principais. Neste projeto, os usuários potenciais são membros da igreja que não fazem parte do ministério do louvor, mas que podem vir a fazer parte dele.

# REQUISITOS

Requisitos são todas as funcionalidades, serviços e restrições de um sistema, que podem ser classificados como Requisitos de Negócio (RN) (1), que envolvem o conhecimento e os detalhes do negócio de um cliente; Requisitos Não Funcionais (RNF) (2), que envolvem aspectos relacionados à qualidade do software e, por fim, Requisitos Funcionais (RF) (3), que envolvem as funcionalidades do sistema computacional.

Nos tópicos a seguir, estão dispostos os requisitos de cada tipo do sistema de ministério de louvor.

## REQUISITOS DE NEGÓCIO

A seguir, estão os requisitos de negócio do sistema:

* RN01: o ministério de louvor está presente em todos os cultos da igreja;
* RN02: cada levita executa instrumentos diferentes e funções diferentes em cada culto;
* RN03: só há um ministro por culto;
* RN04: pode haver mais de um líder no ministério;
* RN05: eleições para a liderança ocorrem anualmente no início do ano;
* RN06: há vários tipos de culto;
* RN07: a liderança cria novas escalas de louvor mensalmente;
* RN08: se um levita não tem disponibilidade para tocar em algum dia, ele deve avisar com um mês de antecedência;
* RN09: se o levita avisar sobre sua indisponibilidade para uma escala de louvor com atraso, ele mesmo deve arranjar alguém para substituí-lo;
* RN10: são os ministros que selecionam as músicas de cada culto;
* RN11: os ministros devem enviar para os levitas o nome, tom, BPM, artista, link da versão e descrição de todas as músicas de cada louvor com, no mínimo, cinco dias de antecedência ao dia do culto de tal louvor;
* RN12: alguns louvores possuem ensaios;
* RN13: a liderança, esporadicamente, convoca os levitas para congregações;
* RN14: em toda congregação, o líder discute uma pauta;
* RN15: os levitas sempre levam algo para comer durante as congregações;
* RN16: a liderança aplica um curso no início de todo ano para a capacitação de pessoas para que, caso queiram, entrem no ministério de louvor;
* RN17: as pessoas que finalizarem o curso preparatório para adentrar ao ministério de louvor devem, caso queiram participar do ministério, fazer uma prova técnica de música, baseada no instrumento que a pessoa pretende tocar. Caso sejam aprovados por todos os jurados, iniciam suas atividades no ministério, e caso sejam reprovados, só podem tentar entrar no ministério novamente no próximo ano, após refazer o curso;
* RN18: os levitas podem ser punidos caso cometam heresias em relação à doutrina;
* RN19: os levitas podem ser expulsos do ministério caso cometam grandes heresias contra a doutrina.

## REQUISITOS NÃO FUNCIONAIS

A seguir, estão os requisitos não funcionais do sistema:

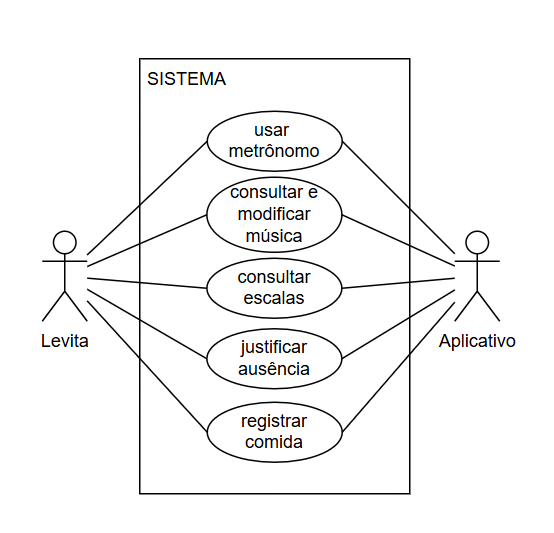
* RNF01: o sistema deve ser capaz de processar até 50 requisiçõessimultâneas de usuários sem degradação perceptível de desempenho;
* RNF02: o sistema deve permitir que as consultas ao banco de dados retornem resultados em até 3 segundos;
* RNF03: o sistema deve ser escalável para suportar o aumento do número de usuários, cultos, eventos e músicas, sem necessidade de reestruturação significativa da arquitetura.
* RNF04: o sistema deve estar disponível a todo tempo, com taxa de disponibilidade mínima de 99,5%;
* RNF05: o sistema deve garantir que os dados dos usuários estejam sempre salvos por meio de rotinas de backups;
* RNF06: o sistema deve permitir seu uso apenas para usuários autenticados;
* RNF07: o sistema deve criptografar as senhas dos usuários com algoritmos de hash;
* RNF08: o sistema deve implementar níveis de controle de acesso para os levitas, ministros e líderes, de forma que o líder tenha acesso às suas próprias funcionalidades, às funcionalidades de ministro e às funcionalidades de levita, enquanto o ministro possui acesso às suas próprias funcionalidades e às funcionalidades de levita, enquanto o levita só tem acesso às suas próprias funcionalidades;
* RNF09: o sistema deve permitir configurações opcionais de autenticação multifator (MFA) via e-mail ou SMS;
* RNF10: o sistema deve ter uma interface minimalista, intuitiva, responsiva, que considere a experiência do usuário e que seja de fácil navegação;
* RNF11: o sistema deve apresentar a visualização de escalas de louvor, calendários, dashboards e insights de forma clara e objetiva;
* RNF12: o sistema deve apresentar mensagens de confirmação sempre que o usuário realizar alguma ação;
* RNF13: o sistema deve ser projetado de forma modular para permitir manutenções mais simples e que não afetem o sistema como um todo;
* RNF14: o sistema deve ter um código-fonte bem documentado, como também documentações claras, objetivas, precisas, detalhadas e atualizadas;
* RNF15: o sistema deve garantir a confidencialidade e integridade dos dados;
* RNF16: o sistema deve estar alinhado às normas da LGPD;
* RNF17: o sistema deve ter mecanismos de reversão de ações para garantir que os usuários não negligenciem dados por descuido;
* RNF18: o sistema deve ser compatível com qualquer plataforma mobile;
* RNF19: o sistema deve estar disponível em português e seus sistemas de calendários e fuso-horários devem ser baseados na localização do Estado de São Paulo;
* RNF20: o sistema deve ser eficiente, utilizando-se de técnicas de cache, processamento paralelo e afins, entre outras boas práticas;
* RNF21: o sistema deve ser acessível para pessoas com deficiências visuais.

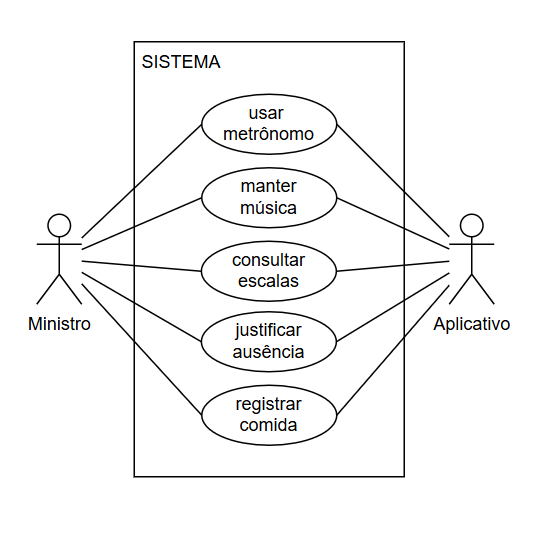
## REQUISITOS FUNCIONAIS

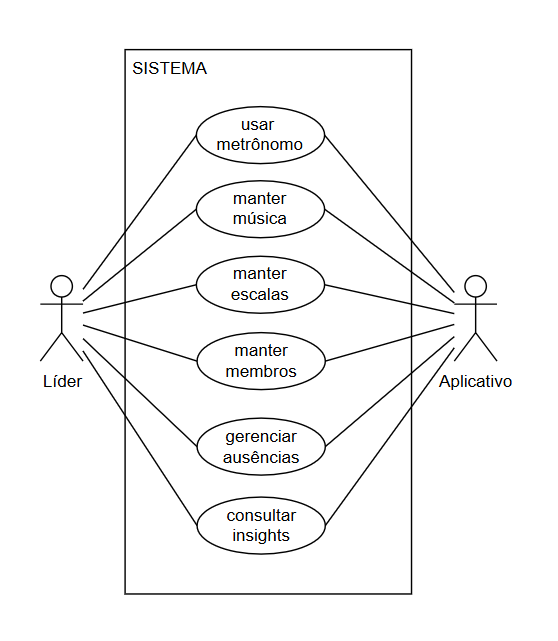
A seguir, seguem os requisitos funcionais do projeto:

* RF01: o sistema deve permitir que os levitas calculem a tonalidade de arquivos de áudio;
* RF02: o sistema deve permitir que os levitas alterem a tonalidade de arquivos de áudio;
* RF03: o sistema deve permitir que os levitas baixem arquivos de áudio com tonalidade alterada;
* RF04: o sistema deve permitir que os levitas calculem os BPMs de arquivos de áudio;
* RF05: o sistema deve permitir que os usuários utilizem a ferramenta de metrônomo;
* RF06: o sistema deve permitir que os levitas mantenham as datas de suas ausências em escalas de louvor;
* RF07: o sistema deve permitir que os levitas mantenham as justificativas de suas ausências em escalas de louvor;
* RF08: o sistema deve permitir que levitas solicitem substitutos quando justificarem a ausência em uma escala com atraso;
* RF09: o sistema deve permitir que levitas sejam substituídos em escalas quando justificarem a ausência em uma escala com atraso por meio de substituições;
* RF10: o sistema deve permitir que o líder de louvor verifique as datas das ausências dos levitas nas escalas de louvor;
* RF11: o sistema deve permitir que o líder de louvor verifique as justificativas de cada data de ausência dos levitas nas escalas de louvor;
* RF12: o sistema deve permitir que o líder de louvor mantenha escalas mensais de louvor;
* RF13: o sistema deve permitir que ministros mantenham perfis para músicas com suas características;
* RF14: o sistema deve permitir que os ministros mantenham o louvor de cada culto e suas características até cinco dias antes do culto;
* RF15: o sistema deve permitir que o líder de louvor agende reuniões e consagrações com os músicos;
* RF16: o sistema deve permitir que os levitas informem qual alimento levarão para as consagrações antes que ela ocorra;
* RF17: o sistema deve permitir o envio de notificações aos levitas quando escalas de louvor e consagrações são criadas;
* RF18: o sistema deve permitir que o levita ative ou desative configurações de notificações;
* RF19: o sistema deve permitir que a liderança cadastre novos membros ao ministério;
* RF20: o sistema deve permitir que a liderança desative a conta de membros que estão em processo de punição por quebra de doutrinas;
* RF21: o sistema deve permitir que a liderança reative contas desativadas;
* RF22: o sistema deve permitir que a liderança remova levitas do ministério;
* RF23: o sistema deve permitir que levitas solicitem alterações em seu perfil de usuário para a liderança;
* RF24: o sistema deve permitir que a liderança mantenha o perfil de usuário dos levitas toda vez que eles solicitarem por alterações.

# DIAGRAMA DE CASO DE USO







# PROTÓTIPOS

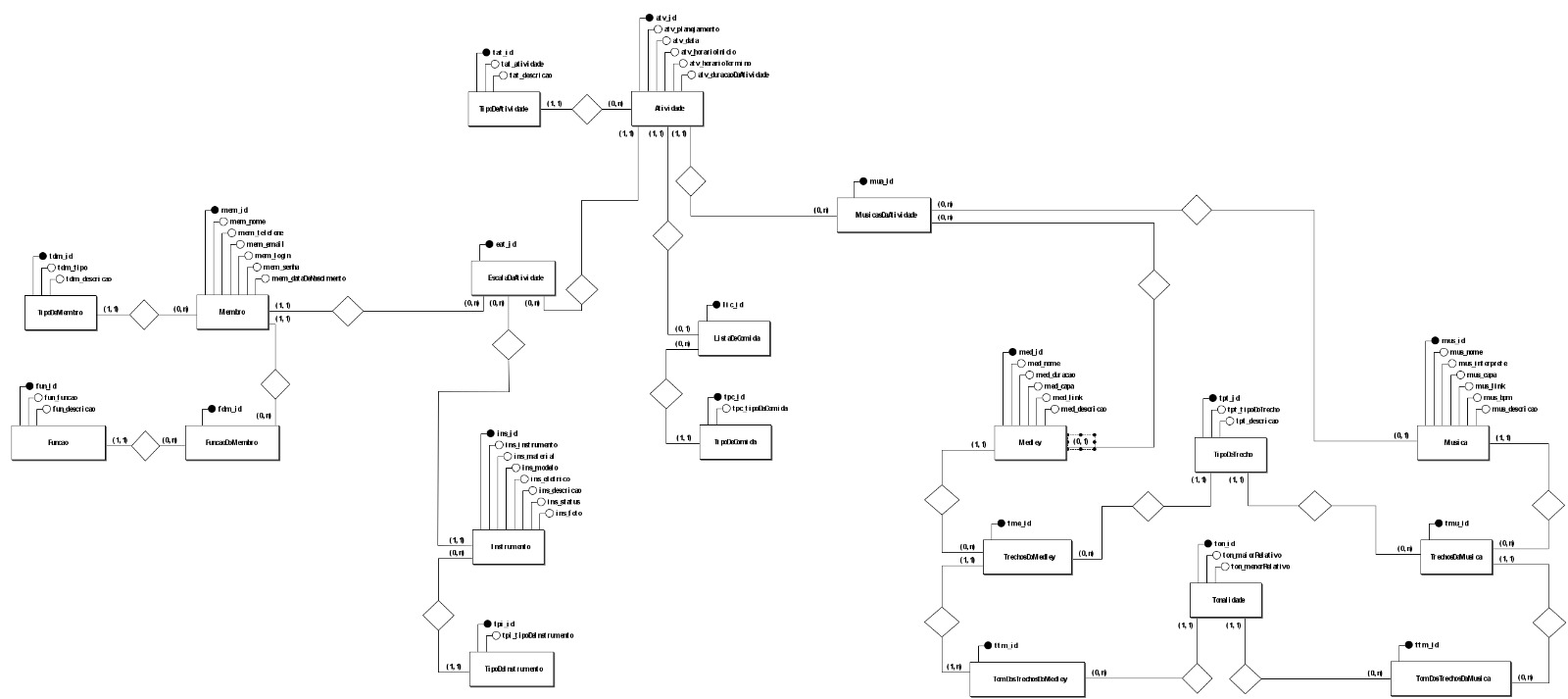
Pegar prints com (H)Eric(o) (Bucchi)

# MODELAGEM DO BANCO DE DADOS

A modelagem de dados refere-se à...

## MODELO CONCEITUAL

O Modelo Conceitual representa de forma abstrata e independente de tecnologia a estrutura dos dados de um sistema. Ele foca em descrever os principais elementos do negócio, suas entidades e relacionamentos, servindo como base inicial para a compreensão e o planejamento do banco de dados.



## MODELO LÓGICO

O Modelo Lógico é uma evolução do modelo conceitual, onde os conceitos abstratos são refinados e organizados com detalhes técnicos. Ele define tabelas, colunas, tipos de dados e relacionamentos no contexto de um SGBD, mas ainda não considera as limitações específicas de implementação.

## MODELO FÍSICO

O Modelo Físico, por sua vez, é a tradução prática do modelo lógico em estruturas concretas no banco de dados. Ele define as implementações específicas, como índices, tabelas físicas, partições e outras configurações relacionadas ao desempenho e armazenamento.

## DICIONÁRIO DE DADOS

O Dicionário de Dados documenta detalhadamente cada elemento do banco de dados, como tabelas, colunas, tipos, restrições e relações. Ele é essencial para manter a consistência e facilitar o entendimento dos dados por desenvolvedores e analistas.

### ENTIDADE “TIPODEMEMBRO”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **tipoDeMembro** | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| tdm\_id | SMALLINT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| tdm\_tipo | VARCHAR | 30 | NOT NULL |  | Campo que diz qual o tipo do membro. |
| tdm\_descricao | VARCHAR | 300 |  | “Sem descrição.” | Campo que descreve a classificação do membro. |

### ENTIDADE “FUNCAO”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **funcao** | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| fun\_id | SMALLINT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| fun\_tipo | VARCHAR | 30 | NOT NULL |  | Campo que diz qual a função do membro. |
| fun\_descricao | VARCHAR | 300 |  | “Sem descrição.” | Campo que descreve a função do membro. |

### ENTIDADE “MEMBRO”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **membro** | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| mem\_id | SMALLINT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| mem\_nome | VARCHAR | 80 | NOT NULL |  | Aponta qual a função do membro. |
| mem\_telefone | INT |  |  | NULL | Aponta quem é o membro que exerce essa função. |
| mem\_email | VARCHAR | 100 | NOT NULL |  |  |
| mem\_login | VARCHAR | 15 | NOT NULL |  |  |
| mem\_senha | TEXT |  | NOT NULL |  |  |
| mem\_dataDeNascimento | DATE |  |  | NULL |  |
| mem\_status | BOOL |  |  | TRUE |  |
| tdm\_id | INT |  | NOT NULL | 1 |  |

### ENTIDADE “FUNCAODOMEMBRO”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **funcaoDoMembro** | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| fdm\_id | SMALLINT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| fun\_id | SMALLINT |  | NOT NULL |  | Aponta qual a função do membro. |
| mem\_id | INT |  | NOT NULL |  | Aponta quem é o membro que exerce essa função. |

### ENTIDADE “TIPODEINSTRUMENTO”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **tipoDeInstrumento** | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| tdi\_id | SMALLINT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| tdi\_tipo | VARCHAR | 20 | NOT NULL |  | Aponta qual a função do membro. |
| tdi\_descricao | VARCHAR | 150 |  | “sem descrição.” | Aponta quem é o membro que exerce essa função. |

### ENTIDADE “INSTRUMENTO”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **instrumento** | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| ins\_id | SMALLINT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| ins\_instrumento | VARCHAR | 20 | NOT NULL |  | Aponta qual a função do membro. |
| ins\_material | VARCHAR | 150 |  | “sem descrição.” | Aponta quem é o membro que exerce essa função. |
| ins\_modelo | VARCHAR | 20 |  |  | Informa o modelo do instrumento. |
| ins\_eletrico | BOOLEAN |  |  |  | Diz se o instrumento é elétrico ou não. |
| ins\_descricao | VARCHAR | 100 |  |  | Traz informações básicas do instrumento. |
| ins\_status | BOOLEAN |  |  | TRUE | Diz se o instrumento está disponível ou não. |
| ins\_foto | VARCHAR | 255 |  |  | Armazena o caminho da imagem. |
| tdi\_id | INT |  | NOT NULL |  | Aponta para o tipo de instrumento. |
| mem\_id | INT |  | NOT NULL |  | Aponta para o dono do instrumento. |

### ENTIDADE “TIPODECOMIDA”

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **tipoDeComida** | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| tdc\_id | SMALLINT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| tdc\_tipo | VARCHAR | 20 | NOT NULL |  | Diz qual a classificação do alimento. |

### ENTIDADE “LISTADECOMIDA”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **listaDeComida** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| lic\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| lic\_quantidade | VARCHAR | 30 | NOT NULL |  | Informa a quantidade da comida. |
| tdc\_id | UNSIGNED SMALLINT |  | NOT NULL |  | Aponta para o tipo de comida. |

### ENTIDADE “TONALIDADE”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **tonalidade** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| ton\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| ton\_maiorRelativo | CHAR | 2 | NOT NULL |  | Diz qual o tom maior relativo. |
| ton\_menorRelativo | CHAR | 2 | NOT NULL |  | Diz qual o tom menor relativo. |

### ENTIDADE “TIPOTRECHO”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **tipoTrecho** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| tpt\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| tpt\_tipo | VARCHAR | 20 | NOT NULL |  | Diz qual o tipo do trecho. |
| ton\_descricao | VARCHAR | 100 | NOT NULL |  | Explicação sobre o tipo do trecho. |

### ENTIDADE “MUSICA”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **musica** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| mus\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| mus\_nome | VARCHAR | 20 | NOT NULL |  | Nome dado à música. |
| mus\_interprete | VARCHAR | 100 | NOT NULL |  | Cantor da música. |
| mus\_capa | VARCHAR | 255 |  |  | Link simbólico do arquivo da imagem de capa. |
| mus\_link | VARCHAR | 100 |  |  | Link da música em alguma plataforma que a tenha. |
| mus\_bpm | UNSIGNED  TINYINT |  |  |  | Batidas por minuto da música. |
| mus\_descricao | VARCHAR | 150 |  |  | Descrição da música. |

### ENTIDADE “TRECHOSMUSICA”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **trechosMusica** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| tmu\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| mus\_id | UNSIGNED INT |  | NOT NULL |  | Aponta para qual é a música. |
| tpt\_id | UNSIGNED INT |  | NOT NULL |  | Aponta para qual o tipo do trecho. |

### ENTIDADE “TOMTRECHOSMUSICA”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **tomTrechosMusica** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| ttm\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| tmu\_id | UNSIGNED INT |  | NOT NULL |  | Aponta para qual é o trecho da música. |
| ton\_id | UNSIGNED TINYINT |  | NOT NULL |  | Aponta para qual a tonalidade desse trecho. |

### ENTIDADE “MEDLEY”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **medley** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| med\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| med\_nome | VARCHAR | 30 | NOT NULL |  | Nome dado ao medley. |
| med\_duracao | VARCHAR | 8 | NOT NULL |  | Diz a duração total do medley. |
| med\_capa | VARCHAR | 255 |  |  | Link para a imagem de capa desse medley. |
| med\_link | VARCHAR | 100 |  |  | Link para alguma plataforma que tenha esse medley, caso já exista. |
| med\_descricao | VARCHAR | 100 |  |  | Descrição do medley. |

### ENTIDADE “TRECHOSMEDLEY”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **trechosMedley** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| tme\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| med\_id | UNSIGNED INT |  | NOT NULL |  | Aponta para qual medley esse trecho pertence. |
| tpt\_id | UNSIGNED INT |  | NOT NULL |  | Aponta para qual o tipo desse trecho. |

### ENTIDADE “TOMTRECHOSMEDLEY”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **tomTrechosMedley** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| ttm\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| tme\_id | UNSIGNED INT |  | FK, NOT NULL |  | Aponta para qual trecho pertence esse tom pertence. |
| ton\_id | UNSIGNED INT |  | FK, NOT NULL |  | Aponta para qual a tonalidade desse trecho. |

### ENTIDADE “MUSICAATIVIDADE”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **musicaAtividade** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| mua\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| atv\_id | UNSIGNED INT |  | FK |  | Aponta para qual atividade esse registro está associado. |
| mus\_id | UNSIGNED INT |  | FK |  | Aponta para a música vinculada a essa atividade. |
| med\_id | UNSIGNED INT |  | FK |  | Aponta para o medley vinculada a essa atividade. |

### ENTIDADE “TIPOATIVIDADE”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **tipoAtividade** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| tat\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| tat\_nome | VARCHAR | 30 |  |  | Informa o nome da atividade. |
| tat\_descricao | VARCHAR | 150 |  |  | Descreve o que é essa atividade. |

### ENTIDADE “ATIVIDADE”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **atividade** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| atv\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| atv\_planejamento | VARCHAR | 300 |  |  | Detalha como será o cronograma da atividade. |
| atv\_dataDaAtividade | DATETIME |  |  |  | Data de realização da atividade. |
| atv\_horarioInicio | VARCHAR | 10 |  |  | Diz quando iniciará a atividade. |
| atv\_horarioTermino | VARCHAR | 10 |  |  | Diz quando encerrará a atividade. |
| atv\_local | VARCHAR | 200 |  |  | Informa onde será realizada a atividade. |
| tat\_id | UNSIGNED INT |  | FK |  | Aponta para qual o tipo da atividade. |

### ENTIDADE “ESCALAATIVIDADE”

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **escalaAtividade** | | | | | | |
| Nome da coluna | Tipo de dado | Tamanho | Restrições | Valor padrão | Descrição |
| eat\_id | UNSIGNED INT |  | PRIMARY KEY, AUTO\_INCREMENT, NOT NULL |  | Campo para identificar os registros dessa tabela. |
| mem\_id | UNSIGNED INT |  |  |  | Aponta para o membro escalado. |
| ins\_id | UNSIGNED INT |  |  |  | Aponta para o instrumento ou equipamento que o membro usará. |
| atv\_id | UNSIGNED INT |  | FK |  | Aponta para qual o tipo da atividade. |

## INSTÂNCIAS DAS TABELAS

As Instâncias de Tabelas de Banco de Dados referem-se aos registros ou dados reais armazenados nas tabelas. São as informações concretas manipuladas pelas aplicações, representando o conteúdo do banco em um dado momento.

|  |  |  |
| --- | --- | --- |
| **ton\_id** | **ton\_maiorTom** | **ton\_menorTom** |
| 1 | C | Am |
| 2 | C# | A#m |
| 3 | D | Bm |
| 4 | D# | Cm |
| 5 | E | C#m |
| 6 | F | Dm |
| 7 | F# | D#m |
| 8 | G | Em |
| 9 | G# | Fm |
| 10 | A | F#m |
| 11 | A# | Gm |
| 12 | B | G#m |

## INSTRUÇÕES SQL

As Instruções SQL são comandos utilizados para interagir com bancos de dados, permitindo a realização de operações como criação e modificação de tabelas (DDL), manipulação de dados (DML), controle de acesso (DCL) e execução de transações (TCL). Elas são a base para consulta, inserção, atualização e exclusão de dados.

As instruções criadas pela equipe estarão dispostos nesse capítulo de maneira seccionada, onde cada etapa corresponderá a uma subseção.

### CRIAÇÃO DA ESTRUTURA

`create database IEQParque;

use IEQParque;

create table ieqparque.tb\_tipoMembro(

tpm\_id int auto\_increment,

tpm\_tipo varchar(20) not null unique,

tpm\_descricao varchar(50) not null unique,

primary key(tpm\_id)

);

create table ieqparque.tb\_membro(

mem\_id int auto\_increment unique,

mem\_nome varchar(50) not null,

mem\_telefone varchar(15) not null,

mem\_email varchar(50) not null,

mem\_login varchar(40) not null unique,

mem\_senha varchar(40) not null,

mem\_dataNasc date not null,

primary key(mem\_id),

tpm\_id int not null REFERENCES tb\_tipoMembro(tpm\_id)

);

create table ieqparque.tb\_funcoes(

fun\_id int auto\_increment unique,

fun\_nome varchar(50) not null unique,

fun\_descricao varchar(100) not null unique,

primary key(fun\_id)

);

create table tb\_funcoesMembro(

fmb\_id int AUTO\_INCREMENT,

mem\_id int not null REFERENCES tb\_membro(mem\_id),

fun\_id int not null REFERENCES tb\_funcoes(fun\_id),

primary key(fmb\_id)

);

create table tb\_tipoAtividade(

tat\_id int AUTO\_INCREMENT PRIMARY KEY,

tat\_nome varchar(50) not null unique,

tat\_descricao varchar(100) not null unique

);

create table tb\_atividade(

atv\_id int AUTO\_INCREMENT PRIMARY KEY,

atv\_planejamento int not null,

atv\_data date not null,

atv\_horarioInicio time not null,

atv\_horarioFim time not null,

tat\_id int not null REFERENCES tb\_tipoAtividade(tat\_id)

);

create table tb\_ausencia(

aus\_id int AUTO\_INCREMENT PRIMARY KEY,

aus\_justificativa text not null,

mem\_id int not null REFERENCES tb\_membro(mem\_id),

atv\_id int not null REFERENCES tb\_atividade(atv\_id)

);

create table tb\_tonalidade(

ton\_id int AUTO\_INCREMENT PRIMARY KEY,

ton\_maiorTom varchar(3) not null,

ton\_menorTom varchar(3) not null

);

create table tb\_musica(

mus\_id int AUTO\_INCREMENT PRIMARY KEY,

mus\_nome varchar(50) not null,

mus\_interprete varchar(60) not null,

mus\_capa varchar(100) not null,

mus\_link varchar(100) not null,

mus\_bpm int not null,

mus\_descricao text not null,

ton\_id int not null REFERENCES tb\_tonalidade(ton\_id)

);

create table tb\_tipoTrechoMusica(

ttm\_id int primary key auto\_increment,

ttm\_tipo varchar(20) not null,

ttm\_descricao text not null

);

create table tb\_trechosMusica(

trm\_id int primary key auto\_increment,

ttm\_id int REFERENCEs tb\_tipoTrechoMusica(ttm\_id),

mus\_id int REFERENCES tb\_musica(mus\_id)

);

create table tb\_tomTrechoMusica(

tdm\_id int primary key auto\_increment,

trm\_id int REFERENCES tb\_trechosMusica(trm\_int),

ton\_id int REFERENCES tb\_tonalidade(ton\_id)

);

create table tb\_medley(

med\_id int primary key auto\_increment,

med\_nome varchar(70) not null,

med\_capa varchar(100) not null,

med\_link varchar(100) not null,

med\_descricao text

);

create table tb\_trechosMedley(

tmd\_id int primary key auto\_increment,

med\_id int REFERENCES tb\_medley(med\_id),

mus\_id int REFERENCES tb\_musica(mus\_id),

ton\_id int REFERENCES tb\_tonalidade(ton\_id)

);`

### INSERTS

use ieqparque;

insert into tb\_tonalidade(ton\_maiorTom, ton\_menorTom)

values('C', 'Am'),

('C#', 'A#m'),

('D', 'Bm'),

('D#', 'Cm'),

('E', 'C#m'),

('F', 'Dm'),

('F#', 'D#m'),

('G', 'Em'),

('G#', 'Fm'),

('A', 'F#m'),

('A#', 'Gm'),

('B', 'G#m');

INSERT INTO tb\_tipoMembro (tpm\_tipo, tpm\_descricao) VALUES

('Líder', 'Membro que lidera um grupo ou ministério'),

('Membro', 'Membro regular da igreja'),

('Visitante', 'Pessoa que visita a igreja');

INSERT INTO tb\_membro (mem\_nome, mem\_telefone, mem\_email, mem\_login, mem\_senha, mem\_dataNasc, tpm\_id) VALUES

('João Silva', '11987654321', 'joao.silva@email.com', 'joaosilva', 'senha123', '1985-05-15', 1),

('Maria Oliveira', '11912345678', 'maria.oliveira@email.com', 'mariaoliveira', 'senha456', '1990-08-20', 2),

('Carlos Pereira', '11998765432', 'carlos.pereira@email.com', 'carlospereira', 'senha789', '2000-12-30', 3);

INSERT INTO tb\_funcoes (fun\_nome, fun\_descricao) VALUES

('Líder de Louvor', 'Responsável por coordenar o ministério de louvor e dirigir os ensaios'),

('Vocalista', 'Membro que canta em grupo ou solo durante os cultos e eventos'),

('Instrumentista', 'Músico que toca instrumentos durante os louvores'),

('Compositor', 'Membro que cria músicas originais para o ministério de louvor'),

('Técnico de Som', 'Responsável pela parte técnica de som e iluminação durante os cultos');

INSERT INTO tb\_funcoesmembro(mem\_id, fun\_id) VALUES

(1, 1),

(1, 2),

(2, 3),

(3, 5);

INSERT INTO tb\_tipoAtividade (tat\_nome, tat\_descricao) VALUES

('Culto de Louvor', 'Reunião semanal dedicada à adoração e louvor a Deus'),

('Ensaios de Louvor', 'Reuniões regulares para ensaiar músicas e preparar o ministério'),

('Apresentação Especial', 'Eventos especiais onde o ministério de louvor se apresenta'),

('Retiro Espiritual', 'Encontro para aprofundamento espiritual e musical do ministério'),

('Workshop de Música', 'Oficinas para aprimoramento de habilidades musicais e vocais');

INSERT INTO tb\_atividade (atv\_planejamento, atv\_data, atv\_horarioInicio, atv\_horarioFim, tat\_id) VALUES

(1, '2023-10-01', '18:00:00', '20:00:00', 1),

(2, '2023-10-03', '19:00:00', '21:00:00', 2),

(3, '2023-10-15', '19:00:00', '20:30:00', 3),

(4, '2023-11-05', '09:00:00', '17:00:00', 4),

(5, '2023-11-12', '14:00:00', '16:00:00', 5),

(6, '2023-10-08', '18:00:00', '20:00:00', 1),

(7, '2023-10-10', '19:00:00', '21:00:00', 2),

(8, '2023-10-22', '19:00:00', '20:30:00', 3);

INSERT INTO tb\_ausencia (aus\_justificativa, mem\_id, atv\_id) VALUES

('Doença', 1, 1),

('Viagem', 2, 2),

('Feriado', 3, 3),

('Compromisso pessoal', 1, 4),

('Doença', 2, 5),

('Viagem', 3, 6),

('Feriado', 1, 7);

INSERT INTO tb\_musica (mus\_nome, mus\_interprete, mus\_capa, mus\_link, mus\_bpm, mus\_descricao, ton\_id) VALUES

('Santo, Santo, Santo', 'Reginaldo Veloso', 'capa\_santo\_santo\_santo.jpg', 'https://link-para-musica.com/santo\_santo\_santo', 75, 'Uma canção de adoração que exalta a santidade de Deus.', 1), -- C

('Te Louvarei', 'Davi Sacer', 'capa\_te\_louvarei.jpg', 'https://link-para-musica.com/te\_louvarei', 80, 'Uma música que expressa gratidão e louvor a Deus.', 2), -- C#

('Apenas um Toque', 'Anderson Freire', 'capa\_apenas\_um\_toque.jpg', 'https://link-para-musica.com/apenas\_um\_toque', 85, 'Uma canção que fala sobre o poder do toque de Deus.', 3), -- D

('Eu Sou Teu', 'Leonardo Gonçalves', 'capa\_eu\_sou\_teu.jpg', 'https://link-para-musica.com/eu\_sou\_teu', 90, 'Uma declaração de entrega total a Deus.', 4), -- D#

('Ousado Amor', 'Hillsong', 'capa\_ousado\_amor.jpg', 'https://link-para-musica.com/ousado\_amor', 70, 'Uma canção que fala sobre o amor incondicional de Deus.', 5), -- E

('Como é Bom Te Adorar', 'Ana Paula Valadão', 'capa\_como\_e\_bom\_te\_adorar.jpg', 'https://link-para-musica.com/como\_e\_bom\_te\_adorar', 78, 'Uma música que celebra a alegria de adorar a Deus.', 6); -- F

INSERT INTO tb\_tipoTrechoMusica (ttm\_tipo, ttm\_descricao) VALUES

('Verso', 'Parte da música que apresenta a história ou mensagem.'),

('Refrão', 'Parte repetitiva e cativante da música.'),

('Ponte', 'Parte que conecta diferentes seções da música.'),

('Intro', 'Parte inicial da música que prepara o ouvinte.'),

('Final', 'Parte que encerra a música.');

INSERT INTO tb\_trechosMusica (ttm\_id, mus\_id) VALUES

(1, 1),

(2, 1),

(1, 2),

(2, 2),

(1, 3),

(2, 3),

(1, 4),

(2, 4),

(1, 5),

(2, 5),

(1, 6),

(2, 6);

INSERT INTO tb\_tomTrechoMusica (trm\_id, ton\_id) VALUES

(1, 1),

(2, 1),

(3, 2),

(4, 2),

(5, 3),

(6, 3),

(7, 4),

(8, 4),

(9, 5),

(10, 5),

(11, 6),

(12, 6);

INSERT INTO tb\_medley (med\_nome, med\_capa, med\_link, med\_descricao) VALUES

('Medley de Adoração', 'capa\_medley\_adoracao.jpg', 'https://link-para-medley.com/medley\_adoracao', 'Um medley que reúne canções de adoração.');

INSERT INTO tb\_trechosMedley (med\_id, mus\_id, ton\_id) VALUES

(1, 1, 1),

(1, 2, 2),

(1, 3, 3);

### VIEWS

As VIEWs são consultas pré-definidas e salvas no banco de dados que permitem visualizar dados de uma ou mais tabelas de forma simplificada ou filtrada. Elas ajudam a centralizar a lógica das consultas, facilitar o acesso a informações complexas e melhorar a segurança ao restringir dados expostos.

use ieqparque;

-- View que retorna todos os membros que possuem uma função e um tipo determinado

create view vw\_funcoesMembro as

select f.\*, m.\*, tm.tpm\_tipo, tm.tpm\_descricao

from tb\_funcoes f left join tb\_funcoesmembro fm on f.fun\_id = fm.fun\_id

right join tb\_membro m on m.mem\_id = fm.mem\_id

inner join tb\_tipomembro tm on tm.tpm\_id = m.tpm\_id;

-- View que retorna a ausência de um membro à alguma atividade da igreja

create view vw\_ausenciaMembro as

select au.aus\_id, m.mem\_nome, ta.tat\_nome, a.atv\_horarioInicio, a.atv\_horarioFim, au.aus\_justificativa

from tb\_tipoatividade ta inner join tb\_atividade a on ta.tat\_id = a.tat\_id

inner join tb\_ausencia au on a.atv\_id = au.atv\_id

inner join tb\_membro m on m.mem\_id = au.mem\_id

order by m.mem\_nome;

create view vw\_musicaTonalidade as

select m.mus\_id, m.mus\_nome, m.mus\_descricao, t.ton\_maiorTom, t.ton\_menorTom

from tb\_musica m inner join tb\_tonalidade t on m.ton\_id = t.ton\_id;

create view vw\_musicasMedley as

select md.med\_nome, m.mus\_nome, t.ton\_maiorTom, t.ton\_menorTom

from tb\_medley md join tb\_trechosmedley tm on md.med\_id = tm.med\_id

join tb\_musica m on m.mus\_id = tm.mus\_id

join tb\_tonalidade t on m.ton\_id = t.ton\_id

create view vw\_musicas\_trechos as

select m.mus\_nome, tt.ttm\_tipo

from tb\_musica m

join tb\_trechosMusica tm on m.mus\_id = tm.mus\_id

join tb\_tipoTrechoMusica tt on tm.ttm\_id = tt.ttm\_id;

### STORED PROCEDURES

As Stored Procedures são blocos de código SQL armazenados no banco de dados, que podem ser executados repetidamente. Elas permitem automatizar processos complexos, reduzir a redundância de código e melhorar o desempenho ao executar operações diretamente no servidor.

use ieqparque;

-- Procedure para adicionar um novo membro

CREATE PROCEDURE sp\_adicionar\_membro(

IN p\_nome VARCHAR(50),

IN p\_telefone VARCHAR(15),

IN p\_email VARCHAR(50),

IN p\_login VARCHAR(40),

IN p\_senha VARCHAR(40),

IN p\_dataNasc DATE,

IN p\_tpm\_id INT

)

BEGIN

DECLARE membro\_existente INT;

-- Verifica se o membro já existe

SELECT COUNT(\*) INTO membro\_existente

FROM tb\_membro

WHERE mem\_email = p\_email OR mem\_login = p\_login;

IF membro\_existente > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Email ou login já estão em uso.';

ELSE

INSERT INTO tb\_membro (mem\_nome, mem\_telefone, mem\_email, mem\_login, mem\_senha, mem\_dataNasc, tpm\_id)

VALUES (p\_nome, p\_telefone, p\_email, p\_login, p\_senha, p\_dataNasc, p\_tpm\_id);

END IF;

END

### FUNCTIONS

As Functions são similares às stored procedures, mas são projetadas para retornar um valor único ou um conjunto de resultados. Elas são utilizadas em consultas SQL para realizar cálculos ou operações específicas de forma reutilizável e eficiente.

use ieqparque;

CREATE FUNCTION fn\_calcular\_idade(data\_nasc DATE)

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE idade INT;

SET idade = YEAR(CURDATE()) - YEAR(data\_nasc);

IF MONTH(CURDATE()) < MONTH(data\_nasc) OR (MONTH(CURDATE()) = MONTH(data\_nasc) AND DAY(CURDATE()) < DAY(data\_nasc)) THEN

SET idade = idade - 1;

END IF;

RETURN idade;

### TRANSACTIONS

As Transactions agrupam operações SQL em blocos que garantem consistência e integridade dos dados. Elas seguem as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), assegurando que as alterações sejam completamente aplicadas ou revertidas em caso de falhas.

`use ieqparque;

-- Trigger para verificar a idade de um membro da igreja

CREATE TRIGGER trg\_verificar\_idade

BEFORE INSERT ON tb\_membro

FOR EACH ROW

BEGIN

IF fn\_calcular\_idade(NEW.mem\_dataNasc) < 18 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Membro deve ter pelo menos 18 anos.';

END IF;

END

-- Trigger para manter a integridade da tabela de funções

CREATE TRIGGER trg\_verificar\_funcao\_unica

BEFORE INSERT ON tb\_funcoesMembro

FOR EACH ROW

BEGIN

DECLARE funcao\_existente INT;

SELECT COUNT(\*) INTO funcao\_existente

FROM tb\_funcoesMembro

WHERE mem\_id = NEW.mem\_id AND fun\_id = NEW.fun\_id;

IF funcao\_existente > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Membro já possui esta função.';

END IF;

END`