UNIVERSIDADE FEDERAL DE UBERLÂNDIA
CAMPUS MONTE CARMELO

Felipe Leme Dias
Luiz Martini Souza

Sistema de Lista de Tarefas

Sistema de organização de tarefas

Monte Carmelo, MG
2023

Sumário

1 INTRODUÇÃO

1.1 APRESENTAÇÃO

O presente projeto propõe a criação de um banco de dados para um sistema de lista de tarefas, com o objetivo de organizar e gerenciar as atividades diárias, tanto em âmbito pessoal quanto profissional. O problema abordado está relacionado à dificuldade de acompanhar e priorizar as tarefas de forma eficiente, resultando em perda de produtividade, desorganização e possíveis atrasos na conclusão das atividades. Diante desse contexto, a criação do banco de dados se torna fundamental para oferecer uma solução tecnológica que otimize o gerenciamento de tarefas, proporcionando uma visão clara das demandas, prazos e responsabilidades.

1.2 DESCRIÇÃO DO PROBLEMA

A eficiente gestão de tarefas é essencial para o bom funcionamento de equipes e projetos em qualquer organização. No entanto, muitas vezes, a falta de um sistema adequado para acompanhar e gerenciar as atividades pode levar a atrasos, perda de informações importantes e desorganização geral. É nesse contexto que um banco de dados de sistema de tarefas desempenha um papel fundamental, fornecendo uma plataforma centralizada para registrar, controlar e monitorar todas as tarefas em andamento.

1.3 IMPORTÂNCIA

A importância do banco de dados proposto reside nas tabelas e consultas que foram desenvolvidas para atender às necessidades de gerenciamento de tarefas. Cada tabela representa uma entidade específica do sistema, como usuários, tarefas, projetos, comentários, categorias, entre outras. O banco de dados proposto é de extrema importância para garantir a eficiência e organização no gerenciamento de tarefas. Ele permite a centralização, relacionamento e consulta dos dados, fornecendo uma base sólida para a tomada de decisões, acompanhamento das atividades e melhoria contínua do processo de gerenciamento de tarefas.

2 OBJETIVOS

O objetivo principal deste projeto é desenvolver um banco de dados que ofereça uma plataforma centralizada para o registro, acompanhamento e gerenciamento das tarefas, permitindo aos usuários organizar suas atividades de forma eficiente e garantindo a conclusão dentro dos prazos estabelecidos. Portanto, a implementação de um banco de dados de sistema de tarefas é fundamental para a

organização e o sucesso de projetos, fornecendo uma estrutura sólida para o gerenciamento eficiente das tarefas, o monitoramento do progresso e a facilitação da comunicação e colaboração entre os membros. Além disso, os objetivos específicos incluem:

- Permitir a criação e atualização de tarefas, incluindo informações como título, descrição, prazo de conclusão, prioridade e status.
- Possibilitar a associação de tarefas aos usuários responsáveis, fornecendo um sistema de atribuição e acompanhamento das responsabilidades.
- Oferecer a funcionalidade de adicionar comentários às tarefas, permitindo a comunicação e registro de informações relevantes relacionadas às atividades.
- Implementar um sistema de categorização das tarefas, possibilitando a organização por temas, projetos ou contextos específicos.
- Gerar notificações e alertas para os usuários, com base nos prazos das tarefas, garantindo que nenhum compromisso seja esquecido ou negligenciado.

3 DESENVOLVIMENTO

O sistema de lista de tarefas permite atribuir tarefas a usuários, definir categorias, prioridades e status, adicionar comentários e anexos, associar tarefas a projetos e equipes, enviar notificações e acompanhar o progresso das tarefas. Ele fornece uma estrutura organizada para gerenciar tarefas e promover a colaboração entre membros de equipes e usuários que desejam organizar sua tarefas.

3.1 REQUISITOS

Requisitos:

1. Cada tarefa possuirá: ID, TÍTULO, STATUS, CATEGORIA, DESCRIÇÃO, DATA DE VENCIMENTO E DATA DE CRIAÇÃO e cada tarefa deverá ter um tipo de prioridade.
2. Cada prioridade possuirá: ID, NOME e VALOR; também vai ter uma ou mais tipos de tarefas.
3. Cada tarefa deverá possuir uma categoria a qual tem: ID e NOME; e cada categoria pode ter uma ou mais tarefas.
4. Cada tarefa possuirá vários comentários no qual terá características como ID, DATA EM QUE FOI CRIADO E TEXTO DO COMENTÁRIO; a tarefa pode ter vários comentários e é possível fazer vários comentários em várias tarefas.

5 Cada tarefa possuirá vários anexos no qual terá características como ID, e NOME DO ARQUIVO; e anexos podem ser feitos em varias tarefas.

6. Cada Usuários possuirá: ID, NOME, EMAIL e SENHA; e realizam várias tarefas e as tarefas podem ser realizadas por vários usuários.

7. Usuários podem receber várias notificações que possuem: ID, DATA DE ENVIO e MENSAGEM; as notificações podem ser enviadas para vários usuários.

8. Cada projeto possuirá: ID, NOME, DATA DE INICIO, DATA FIM e DESCRIÇÃO; e será associado a várias tarefas e cada tarefa poderá ser associada a um projeto.

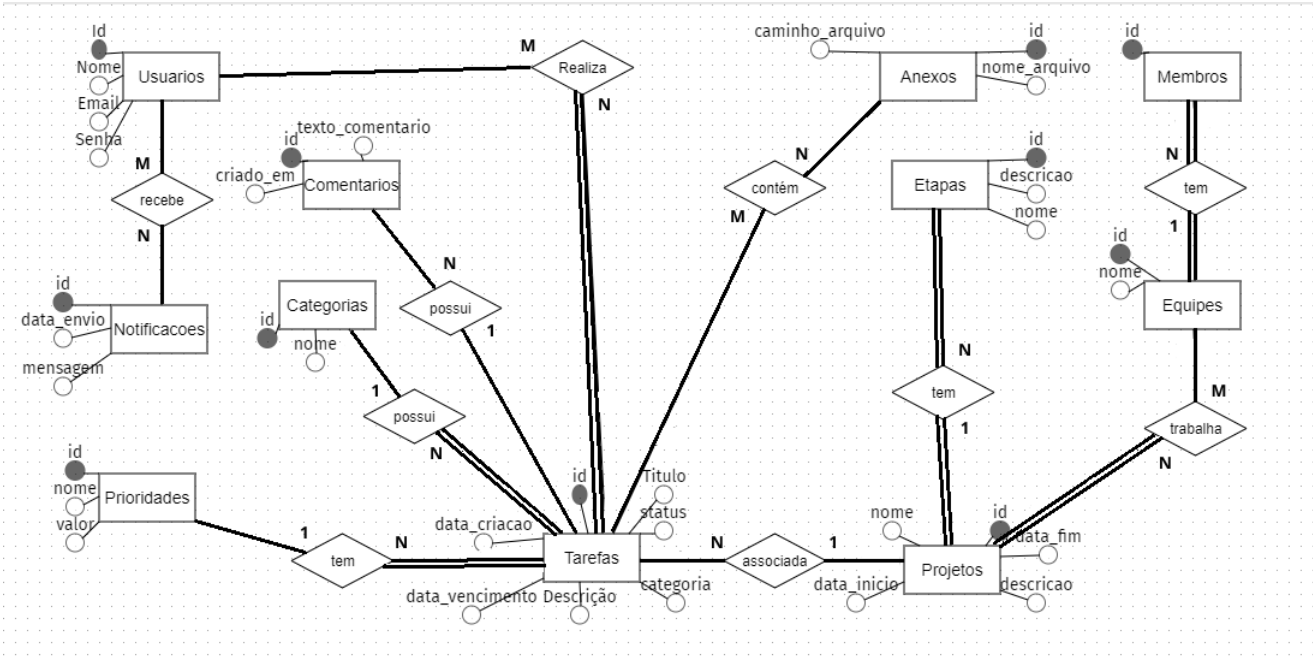
9. O projeto deverá ter várias etapas que possuí: ID, DESCRIÇÃO e NOME; e cada etapa deverá ser específica para um projeto.

10. O projeto deverá ser realizado por várias equipes que possuí: ID e NOME DA EQUIPE; e cada equipe pode ter vários projetos.

11. Cada equipe deverá ter vários membros.

12. Membros devem somente de uma equipe.

3.2 DIAGRAMA ENTIDADE-RELACIONAMENTO



3.3 MODELO RELACIONAL

Usuarios = {id_usuario(PK), nome, email, senha}

Tarefas = {id_tarefa(PK), titulo, descricao, data_criacao, data_vencimento, status, prioridade, id_categoria, id_projeto, usuario_id} prioridade chave estrangeira referencia id_prioridades da tabela Prioridades.

Comentarios = {id_comentario(PK), id_tarefa, texto_comentario, criado_em} id_tarefa chave estrangeira referencia id_tarefa da tabela Tarefas.

Categorias = {id_categoria(PK), nome}

Prioridades = {id_prioridades(PK), nome, valor}

Anexos = {id_anexos(PK), nome_arquivo, caminho_arquivo}

Equipes = {id_equipes(PK), nome}

Membros = {id_membros(PK), id_equipe} id_equipe chave estrangeira referencia id_equipes da tabela Equipes.

Projetos = {id_projeto(PK), nome, descricao, data_inicio, data_fim}

Notificacoes = {id_notificacao(PK), mensagem, data_envio}

Etapas = {id_etapas(PK), nome, descricao, projeto_id} projeto_id chave estrangeira referencia id_projeto da tabela Projetos.

Usuarios_recebeNotificacao = {id_usuario(PK), id_notificacao(PK)} id_usuario chave estrangeira referencia id_usuario da tabela Usuarios, id_notificacoes chave estrangeira referencia id_notificacao da tabela Notificacoes.

Tarefa_possuiComentarios = {id_comentario(PK), id_tarefa(PK)} id_comentario chave estrangeira referencia id_comentario da tabela Comentarios, id_tarefa chave estrangeira referencia id_tarefa da tabela Tarefas.

Usuarios_realizaTarefas = {id_usuario(PK), id_tarefas(PK)} id_usuario chave estrangeira referencia id_usuario da tabela Usuarios, id_tarefas chave estrangeira referencia id_tarefa da tabela Tarefas.

Tarefa_contemAnexos = {id_tarefa(PK), id_anexos(PK)} id_tarefa chave estrangeira referencia id_tarefa da tabela Tarefas, id_anexos chave estrangeira referencia id_anexos da tabela Anexos.

Equipes_trabalhamProjetos = {id_equipes(PK), id_prjeto(PK)} id_equipes chave estrangeira referencia id_equipes da tabela Equipes, id_projeto chave estrangeira referencia id_projeto da tabela Projetos.

4 CONSULTAS

Pelo menos 3 tuplas devem ser alteradas ao longo do script (UPDATE):

1. Alterar o nome do projeto com id_projeto = 1

```
UPDATE Projetos SET nome = 'Projeto␣dos␣Deuses'
WHERE id_projeto = 1;
```

2. Alterar o status da tarefa com id_tarefa = 2

```
UPDATE Tarefas SET status = 'Tarefa␣quase␣pronta'
WHERE id_tarefa = 2;
```

3. Alterar a descrição da tarefa com id_tarefa = 3

```
UPDATE Tarefas SET descricao = 'Fazer␣PDF'
WHERE id_tarefa = 3;
```

Pelo menos 3 tuplas devem ser removidas ao longo do script (DELETE).

1. Remover um usuário e todas as suas notificações recebidas:

```
DELETE FROM Usuarios_recebeNotificacao
WHERE id_usuario = 1;
```

2. Remover uma tarefa e todos os comentários e anexos relacionados:

```
DELETE FROM Tarefa_possuiComentarios
WHERE id_tarefa = 1;
```

3. Remover uma equipe e todos os projetos em que trabalha:

```
DELETE FROM Equipes_trabalhamProjetos
WHERE id_equipes = 1;
```

Pelo menos 4 das consultas devem utilizar duas ou mais tabelas como fonte.

1. Consulta para obter todas as tarefas com seus respectivos projetos:

```
SELECT T.*, P.nome AS nome_projeto
FROM Tarefas AS T
JOIN Projetos AS P ON T.id_projeto = P.id_projeto;
```

2. Consulta para buscar os membros de uma equipe específica:

```
SELECT U.*
FROM Usuarios AS U
JOIN Membros AS M ON U.id_usuario = M.id_membros
JOIN Equipes AS E ON M.id_equipe = E.id_equipes
WHERE E.nome = 'Equipe 1';
```

3. Selecionar todas as tarefas com suas respectivas categorias:

```
SELECT Tarefas.*, Categorias.nome AS nome_categoria
FROM Tarefas
INNER JOIN Categorias ON Tarefas.id_categoria = Categorias.
    ↪ id_categoria;
```

4. Selecionar todas as tarefas com seus respectivos comentários:

```
SELECT Tarefas.*, Comentarios.texto_comentario
FROM Tarefas
INNER JOIN Tarefa_possuiComentarios ON Tarefas.id_tarefa =
    ↪ Tarefa_possuiComentarios.id_tarefa
INNER JOIN Comentarios ON Tarefa_possuiComentarios.
    ↪ id_comentario = Comentarios.id_comentario;
```

Pelo menos 2 das consultas deve utilizar junção externa.

1. Nesta consulta, usamos a junção externa esquerda (LEFT JOIN) para garantir que todas as tarefas sejam retornadas, mesmo que não tenham um projeto associado. A coluna "nome_projeto"exibirá o nome do projeto correspondente, se existir.

```
SELECT T.*, P.nome AS nome_projeto
FROM Tarefas AS T
LEFT JOIN Projetos AS P ON T.id_projeto = P.id_projeto;
```

2. Nesta consulta, usamos a junção externa esquerda (LEFT JOIN) para garantir que todas as categorias sejam retornadas, mesmo que não tenham tarefas associadas. A coluna "titulo_tarefa"exibirá o título da tarefa correspondente, se existir.

```
SELECT C.*, T.titulo AS titulo_tarefa
FROM Categorias AS C
LEFT JOIN Tarefas AS T ON C.id_categoria = T.id_categoria;
```

Pelo menos 2 das consultas deve utilizar funções de agregação com agrupamento.

1. Nesta consulta, retorna o nome da categoria e o número total de tarefas associadas a cada categoria. A função de agregação COUNT é utilizada para contar o número de tarefas agrupadas por categoria.

```
SELECT C.nome AS nome_categoria, COUNT(T.id_tarefa) AS
    ↪ total_tarefas
FROM Categorias AS C
LEFT JOIN Tarefas AS T ON C.id_categoria = T.id_categoria
GROUP BY C.nome;
```

2. Nesta consulta, utilizamos as tabelas Tarefas, Tarefa_possuiComentarios e Comentarios para obter a contagem de comentários para cada tarefa. A função de agregação COUNT é utilizada para contar o número de comentários agrupados por tarefa.

```
SELECT T.id_tarefa, COUNT(C.id_comentario) AS
    ↪ total_comentarios
FROM Tarefas AS T
LEFT JOIN Tarefa_possuiComentarios AS TC ON T.id_tarefa =
    ↪ TC.id_tarefa
LEFT JOIN Comentarios AS C ON TC.id_comentario = C.
    ↪ id_comentario
GROUP BY T.id_tarefa;
```

Pelo menos 2 das consultas devem utilizar UNION, INTERCEPT ou EXCEPT.

1. Nessa consulta, utiliza o UNION para combinar duas consultas em uma única lista. A primeira parte da consulta retorna o título e o status das tarefas concluídas, enquanto a segunda parte retorna o título e o status das tarefas pendentes.

```
SELECT titulo, status FROM Tarefas WHERE status = '
    ↪ Concluida'
UNION
SELECT titulo, status FROM Tarefas WHERE status = 'Pendente
    ↪ ';
```

2. Nessa consulta, utiliza o UNION para combinar duas consultas em uma única lista. A primeira parte da consulta retorna o nome dos usuários que realizam tarefas, adicionando uma coluna 'tipo' com o valor 'Realiza tarefas'. A segunda parte retorna o nome dos usuários que recebem notificações, adicionando uma coluna 'tipo' com o valor 'Recebe notificações'.

```
SELECT nome, 'Realiza␣tarefas' AS tipo FROM Usuarios
INNER JOIN Usuarios_realizaTarefas ON Usuarios.id_usuario =
    ↪ Usuarios_realizaTarefas.id_usuario
UNION
SELECT nome, 'Recebe␣notificacoes' AS tipo FROM Usuarios
INNER JOIN Usuarios_recebeNotificacao ON Usuarios.id_usuario
    ↪ = Usuarios_recebeNotificacao.id_usuario;
```

Pelo menos 2 das consultas devem apresentar subconsultas, sendo que:

1. Utilizando a subconsulta IN:
   Nessa consulta, retorna todas as tarefas que possuem a categoria com nome "Categoria 1".

```
SELECT *
FROM Tarefas
WHERE id_categoria IN (
  SELECT id_categoria
  FROM Categorias
  WHERE nome = 'Categoria␣1'
);
```

2. Utilizando a subconsulta EXISTS:
   Nessa consulta, retorna todos os usuários que realizaram tarefas que possuem comentários.

```
SELECT *
FROM Usuarios
WHERE EXISTS (
  SELECT *
  FROM Usuarios_realizaTarefas
```

```
    JOIN Tarefa_possuiComentarios ON Usuarios_realizaTarefas.
        ↪ id_tarefas = Tarefa_possuiComentarios.id_tarefa
    WHERE Usuarios_realizaTarefas.id_usuario = Usuarios.
        ↪ id_usuario
);
```

5 CONCLUSÕES

Após fazer todo o processo da criação de um banco de dados, notamos que a parte principal da criação são a discussão dos requisitos e das necessidades do sistema, se os requisitos foram bem definidos, a modelagem torna-se fácil. Por fim, tivemos dificuldades na ordem de criação das tabelas e fazer com que o script, diagrama e modelo relacional se ligassem; sempre tinha que mudar alguma coisa de algum lugar.

6 REFERÊNCIAS

- Paulino, A. A. (Profa. Alessandra Aparecida). Aula 12 - Linguagem de Manipulação de Dados (DML). Slides cedidos pelo Prof. Bruno Augusto Nassif Travençolo.
- Paulino, A. A. (Profa. Alessandra Aparecida). Aula 13 - DML: Select. Slides cedidos pelo Prof. Bruno Augusto Nassif Travençolo.
- Paulino, A. A. (Profa. Alessandra Aparecida). Aula 13a - Sequências, valores padrão e restrições. Slides cedidos pelo Prof. Bruno Augusto Nassif Travençolo.
- Paulino, A. A. (Profa. Alessandra Aparecida). Aula 14 - DML: Select Join. Slides cedidos pelo Prof. Bruno Augusto Nassif Travençolo.
- Paulino, A. A. (Profa. Alessandra Aparecida). Aula 15 - DML: Select Subconsultas. Slides cedidos pelo Prof. Bruno Augusto Nassif Travençolo.
- OpenAI. ChatGPT (Versão GPT-3.5). 2021.