Projeto BD - para Nota de AV2

Segue abaixo um modelo do que deve ter em seu projeto. As especificações devem ser seguidas, mas você pode desenvolver seu projeto para qualquer área de negócios. Ex: Pet shop, Estacionamento, E-commerce e etc.

Projeto: Sistema de Gerenciamento - "TechManager"

Disciplina: Banco de Dados
Valor: Base 4,0 pontos(sem trigger) ou 5,0 pontos (com trigger)
Modalidade: Individual ou em equipe(até 5 integrantes)
Data de Entrega:

21/11 - CHAMADA 1

05/12 - CHAMADA 2

1. Objetivo

Este projeto tem como objetivo principal consolidar os conhecimentos adquiridos em sala de aula, integrando os conceitos de banco de dados relacional com o desenvolvimento de uma aplicação prática. Você irá projetar, implementar e povoar um banco de dados, e desenvolver uma interface front-end para interagir com os dados através de operações CRUD (Create, Read, Update, Delete).

2. O Desafio: Criando o TechManager

Você foi contratado para desenvolver um sistema interno para uma empresa de tecnologia, o TechManager. O sistema deve gerenciar informações cruciais sobre os Funcionários e os Projetos aos quais eles estão alocados.

O sistema deve permitir:

- Cadastrar novos funcionários e projetos.
- Visualizar uma lista de todos os funcionários e projetos.
- Editar as informações de qualquer funcionário ou projeto.
- Excluir (ou inativar) funcionários e projetos.

- Associar e desassociar funcionários a projetos.

3. Especificações do Banco de Dados

Você deve criar um banco de dados chamado `techmanager` com, pelo menos, as seguintes tabelas:

Tabela `funcionarios`:

- `id` (INT, PRIMARY KEY, AUTO_INCREMENT)
- `nome` (VARCHAR(100), NOT NULL)
- `cargo` (VARCHAR(50))
- `email` (VARCHAR(100), UNIQUE)
- `data_contratacao` (DATE)
- `salario` (DECIMAL(10,2))

Tabela `projetos`:

- `id` (INT, PRIMARY KEY, AUTO_INCREMENT)
- `nome` (VARCHAR(100), NOT NULL)
- `descricao` (TEXT)
- `data_inicio` (DATE)
- `data_prevista_termino` (DATE)
- `status` (VARCHAR(20)) -- Ex: 'Planejamento', 'Em Andamento', 'Concluído'

Tabela `alocacoes` (Tabela de Junção - N para N):

- `funcionario_id` (INT, FOREIGN KEY REFERENCES `funcionarios`(id))
- `projeto_id` (INT, FOREIGN KEY REFERENCES `projetos`(id))
- `data_alocacao` (DATE) -- Data em que o funcionário foi alocado ao projeto
- `horas_trabalhadas` (INT) -- Número de horas semanais dedicadas ao projeto
- PRIMARY KEY (`funcionario_id`, `projeto_id`)

4. Requisitos Obrigatórios (CRUD - 4,0 pontos) - O Crud deve ter obrigatoriamente no seu projeto independentemente da área de negócios escolhida.

Sua aplicação, com o front-end na linguagem de sua escolha (ex: Python/Flask, JavaScript/Node.js, PHP, Java/Spring Boot, C#/.NET, etc.), deve implementar as seguintes funcionalidades:

1. Para Funcionários:

    - Create: Formulário para cadastrar um novo funcionário.
    - Read: Página/listagem para visualizar todos os funcionários.
    - Update: Formulário para editar os dados de um funcionário existente.
    - Delete: Botão/Ação para remover um funcionário (com confirmação).
2. Para Projetos:
    - Create: Formulário para cadastrar um novo projeto.
    - Read: Página/listagem para visualizar todos os projetos.
    - Update: Formulário para editar os dados de um projeto existente.
    - Delete: Botão/Ação para remover um projeto (com confirmação).
3. Para Alocações:
    - Create: Uma forma de alocar um funcionário a um projeto (definindo `horas_trabalhadas`).
    - Read: Em uma página de detalhes do projeto, listar todos os funcionários alocados, e vice-versa.
    - Update: Possibilidade de alterar as `horas_trabalhadas` de uma alocação.
    - Delete: Remover um funcionário de um projeto (desalocar).

5. Requisito Bônus: TRIGGERS (+1,0 ponto) - Projetos apresentados sem Triggers concorrem por 9,0 pontos, com TRIGGERS concorrem a 10 pontos (pelo menos UMA trigger).

Para garantir a integridade dos dados e a qualidade da informação, implemente pelo menos UMA das triggers abaixo no seu banco de dados:

- Trigger 1: `trg_impedir_salario_negativo`
    - Objetivo: Impedir a inserção ou atualização de um salário com valor negativo na tabela `funcionarios`.
    - Ação: Se um valor negativo for tentado, a trigger deve disparar um erro e abortar a operação.
- Trigger 2: `trg_registrar_alteracao_salario`
    - Objetivo: Auditar mudanças salariais.
    - Ação: Criar uma tabela `auditoria_salarios` (`id`, `funcionario_id`, `salario_antigo`, `salario_novo`, `data_alteracao`). Sempre que o salário de um funcionário for atualizado, a trigger deve inserir automaticamente um registro nesta tabela de auditoria.
- Trigger 3: `trg_atualizar_status_projeto_auto`
    - Objetivo: Automatizar a lógica de negócio.

    - Ação: Criar uma trigger que, ao alocar o primeiro funcionário a um projeto (INSERT na tabela `alocacoes`), altere automaticamente o `status` do projeto de 'Planejamento' para 'Em Andamento'.

Observação: A trigger deve estar documentada no script SQL de criação do banco, que deve ser entregue junto com o projeto.

6. Critérios de Avaliação

- **Funcionamento (4,0 pts): O programa executa sem erros e todas as operações CRUD, no Banco de Dados, funcionam conforme especificado.**
- **Bônus - Trigger (1,0 pt): Implementação e funcionamento correto de pelo menos uma trigger, conforme descrito.**

7. Itens para Entrega

Para que o projeto seja avaliado, você deve apresentar o programa na data marcada. É obrigatória a presença de todos os integrantes da equipe no dia da apresentação.