1. Objetivo

Desenvolver uma aplicação Java backend orientada a objetos que utilize JDBC para realizar **todas as operações de persistência** (CRUD) com o banco de dados empresa. O sistema deverá ser funcional, atender a **10 regras de negócio obrigatórias** com uso de **condicionais**, e seguir boas práticas de codificação e organização.

2. Contexto da Atividade

Você é responsável por desenvolver a aplicação de gestão para a base de dados de uma empresa, composta por pessoas, funcionários e projetos. A aplicação deverá permitir o cadastro, atualização, consulta e exclusão de registros dessas três entidades, respeitando as regras de negócio listadas abaixo.

3. Regras de Negócio (Obrigatórias - 30% da nota)

As regras a seguir devem ser implementadas utilizando **condições (if, else, switch, etc.)**:

1. Ao cadastrar um funcionário, verificar se o ID da pessoa existe na tabela pessoa.
2. Um projeto não pode ser criado sem vínculo com um funcionário existente.
3. Proibir a exclusão de um funcionário que esteja vinculado a algum projeto.
4. Exibir mensagem de erro clara sempre que uma operação falhar por descumprimento das regras.
5. Toda operação bem-sucedida deve imprimir uma mensagem de confirmação no console.

4. Requisitos Técnicos e Funcionais

O sistema deverá:

- Utilizar Java com JDBC (sem frameworks ORM).
- Criar todas as operações CRUD para:
  - Pessoa
  - Funcionário
  - Projeto

- Incluir classes bem definidas com encapsulamento e responsabilidade única.
- Conectar-se ao banco MySQL com as configurações que criou no banco de dados.

5. Definição Descritiva dos Componentes do Sistema (Classes)

a) Pessoa

A classe Pessoa representa qualquer indivíduo registrado na empresa. É a base para outras entidades mais específicas, como Funcionario. Cada objeto do tipo Pessoa contém as seguintes informações:

- **id**: identificador único da pessoa (gerado automaticamente).
- **nome**: nome completo da pessoa.
- **email**: endereço de e-mail válido, necessário para contato.

b) Funcionario

A classe Funcionario representa os colaboradores da empresa que estão ativamente alocados em algum setor. Cada funcionário é também uma pessoa, portanto essa classe pode **estender Pessoa** (herança).

- **id**: chave estrangeira que faz referência a uma Pessoa.
- **matricula**: código único de identificação do funcionário no formato "F" + três números (ex: F007).
- **departamento**: área da empresa à qual o funcionário pertence (TI, RH, Marketing, etc.).

Essa classe é responsável por **comportamentos específicos de funcionários**, como regras de alocação em projetos, controle de quantidade de projetos ativos e validação da matrícula.

c) Projeto

A classe Projeto representa uma iniciativa interna ou externa desenvolvida pela empresa. Cada projeto está vinculado obrigatoriamente a um funcionário responsável.

- **id**: identificador único do projeto.
- **nome**: título do projeto.
- **descricao**: descrição detalhada do que o projeto propõe ou executa.

- **id_funcionario**: chave estrangeira que vincula o projeto a um funcionário específico.

6. **Entrega Obrigatória**

- Criar repositório no **GitHub** contendo:
  - Projeto Java completo com pastas organizadas
  - Arquivos .java, .sql, e configurações de conexão
  - Export do banco de dados (empresa.sql)
- Postar no **AVA**:
  - Um arquivo .txt com o **link do repositório GitHub**
  -

7. **CRITÉRIOS DE AVALIAÇÃO – PROVA PRÁTICA JAVA BACKEND (100%)**

Categoria	Critério Avaliado	Peso (%)
1. Regras de Negócio (com uso de condicionais)	a) Implementação das 5 regras de negócio obrigatórias utilizando comandos condicionais (if, else, switch)	**30%**
2. Estrutura e Funcionalidade do Código	a) Código executa sem erros (compila, conecta ao banco e executa as operações)	**10%**
	b) Todas as operações de persistência implementadas para as 3 entidades (insert, update, delete, select por ID, select lista)	**10%**
3. Boas Práticas de Codificação	a) Nome das variáveis e métodos seguem padrão camelCase e são semânticos	**5%**
	a) Código devidamente identado e organizado em classes separadas	**5%**
	b) Documentação básica presente nos métodos principais (comentários explicando o propósito de cada um)	**5%**
4. Banco de Dados	a) Criação correta do banco empresa, usuário devuser, tabelas com chaves primárias e estrangeiras conforme o modelo	**10%**

Categoria	Critério Avaliado	Peso (%)
	b) Banco populado com dados adequados conforme o enunciado (mínimo de 10 registros por tabela)	5%
5. Entrega e Organização	a) Projeto postado corretamente no **GitHub (pasta do proejto completa + export do banco de dados**	5%
	b)Arquivo .txt com o link do repositório enviado no **AVA** dentro do prazo	5%
	c) Repositório contém o **export do banco (.sql)**	5%
	d)Instruções mínimas de execução do sistema (comentário no código ou readme)	5%

⚠ **PENALIDADES IMPORTANTES**

- **Código não executa (não compila ou não conecta ao banco):** nota **0 (zero)**
- **Não entrega no GitHub ou não envia link no AVA:** nota **0 (zero)**
- **Regras de negócio incompletas ou ausentes:** nota proporcional à quantidade atendida
- **Plágio identificado:** nota **0 (zero)**, com encaminhamento à coordenação.
- **Atenção**: A **não execução do código funcional ou a ausência da entrega no GitHub ou AVA implica em nota 0 (zero)**.