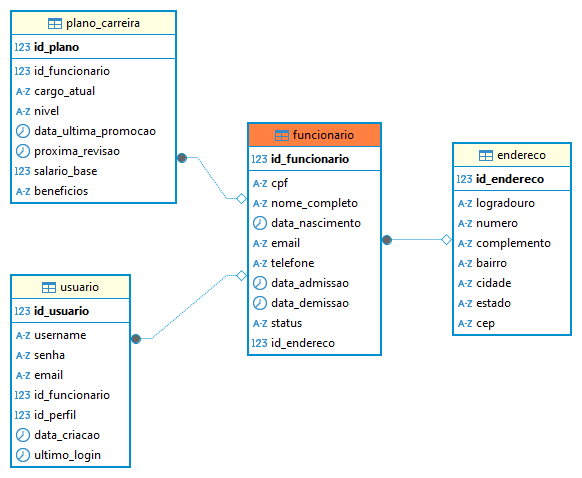
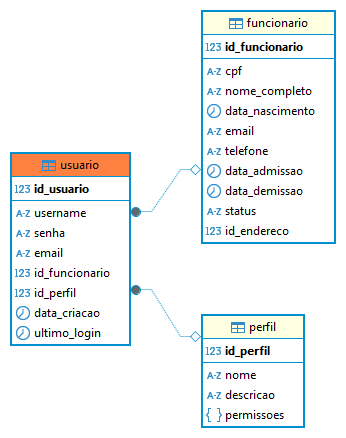
# Projeto de Banco de Dados - Gestão RH

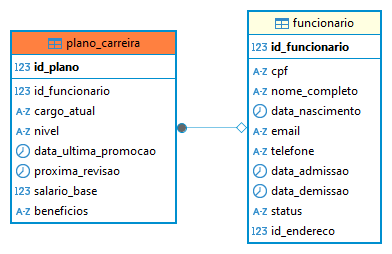
## 1. Visão Geral do Sistema

O sistema de Gestão RH tem como objetivo principal gerenciar todas as informações relacionadas aos colaboradores de uma empresa, desde cadastro básico até o plano de carreira, proporcionando uma ferramenta eficiente para o departamento de Recursos Humanos.

## 2. Modelagem Conceitual (Diagrama Entidade-Relacionamento)







[Funcionário] --possui--> [Endereço]

[Funcionário] --tem--> [Plano de Carreira]

[Usuário] --possui--> [Perfil]

[Usuário] --gerencia--> [Funcionário]

## 3. Modelagem Lógica (Esquema Relacional)

### 3.1 Tabelas Principais

#### FUNCIONARIO

| **Campo** | **Tipo** | **Tamanho** | **Descrição** | **Restrições** |
| --- | --- | --- | --- | --- |
| id\_funcionario | INT |  | Identificador único | PK, AUTO\_INCREMENT |
| cpf | VARCHAR | 11 | CPF do funcionário | UNIQUE, NOT NULL |
| nome\_completo | VARCHAR | 100 | Nome completo | NOT NULL |
| data\_nascimento | DATE |  | Data de nascimento | NOT NULL |
| email | VARCHAR | 100 | E-mail corporativo | NOT NULL |
| telefone | VARCHAR | 20 | Telefone para contato |  |
| data\_admissao | DATE |  | Data de admissão | NOT NULL |
| data\_demissao | DATE |  | Data de demissão (se aplicável) |  |
| status | ENUM |  | 'Ativo', 'Inativo', 'Afastado' | DEFAULT 'Ativo' |
| id\_endereco | INT |  | Chave estrangeira para endereço | FK |

#### ENDERECO

| **Campo** | **Tipo** | **Tamanho** | **Descrição** | **Restrições** |
| --- | --- | --- | --- | --- |
| id\_endereco | INT |  | Identificador único | PK, AUTO\_INCREMENT |
| logradouro | VARCHAR | 100 | Nome da rua/av | NOT NULL |
| numero | VARCHAR | 10 | Número | NOT NULL |
| complemento | VARCHAR | 50 | Complemento |  |
| bairro | VARCHAR | 50 | Bairro | NOT NULL |
| cidade | VARCHAR | 50 | Cidade | NOT NULL |
| estado | VARCHAR | 2 | UF | NOT NULL |
| cep | VARCHAR | 8 | CEP | NOT NULL |

#### PLANO\_CARREIRA

| **Campo** | **Tipo** | **Tamanho** | **Descrição** | **Restrições** |
| --- | --- | --- | --- | --- |
| id\_plano | INT |  | Identificador único | PK, AUTO\_INCREMENT |
| id\_funcionario | INT |  | Chave estrangeira para funcionário | FK, UNIQUE |
| cargo\_atual | VARCHAR | 50 | Cargo atual | NOT NULL |
| nivel | VARCHAR | 20 | Nível hierárquico | NOT NULL |
| data\_ultima\_promocao | DATE |  | Data da última promoção |  |
| proxima\_revisao | DATE |  | Data da próxima revisão | NOT NULL |
| salario\_base | DECIMAL | 10,2 | Salário base | NOT NULL |
| beneficios | TEXT |  | Benefícios adicionais |  |

#### USUARIO

| **Campo** | **Tipo** | **Tamanho** | **Descrição** | **Restrições** |
| --- | --- | --- | --- | --- |
| id\_usuario | INT |  | Identificador único | PK, AUTO\_INCREMENT |
| username | VARCHAR | 50 | Nome de usuário | UNIQUE, NOT NULL |
| senha | VARCHAR | 255 | Senha criptografada | NOT NULL |
| email | VARCHAR | 100 | E-mail | UNIQUE, NOT NULL |
| id\_funcionario | INT |  | Chave estrangeira para funcionário | FK, UNIQUE |
| id\_perfil | INT |  | Chave estrangeira para perfil | FK |
| data\_criacao | DATETIME |  | Data de criação | DEFAULT CURRENT\_TIMESTAMP |
| ultimo\_login | DATETIME |  | Data do último login |  |

#### PERFIL

| **Campo** | **Tipo** | **Tamanho** | **Descrição** | **Restrições** |
| --- | --- | --- | --- | --- |
| id\_perfil | INT |  | Identificador único | PK, AUTO\_INCREMENT |
| nome | VARCHAR | 50 | Nome do perfil | UNIQUE, NOT NULL |
| descricao | VARCHAR | 255 | Descrição do perfil |  |
| permissoes | TEXT |  | Permissões em JSON |  |

## 4. Script SQL de Criação do Banco de Dados

## 1. Script de Criação do Banco de Dados

-- Criação do banco de dados

CREATE DATABASE gestao\_rh;

\c gestao\_rh

-- Tabela ENDERECO

CREATE TABLE ENDERECO (

id\_endereco SERIAL PRIMARY KEY,

logradouro VARCHAR(100) NOT NULL,

numero VARCHAR(10) NOT NULL,

complemento VARCHAR(50),

bairro VARCHAR(50) NOT NULL,

cidade VARCHAR(50) NOT NULL,

estado VARCHAR(2) NOT NULL,

cep VARCHAR(8) NOT NULL

);

-- Tabela FUNCIONARIO

CREATE TABLE FUNCIONARIO (

id\_funcionario SERIAL PRIMARY KEY,

cpf VARCHAR(11) UNIQUE NOT NULL,

nome\_completo VARCHAR(100) NOT NULL,

data\_nascimento DATE NOT NULL,

email VARCHAR(100) NOT NULL,

telefone VARCHAR(20),

data\_admissao DATE NOT NULL,

data\_demissao DATE,

status VARCHAR(10) DEFAULT 'Ativo' CHECK (status IN ('Ativo', 'Inativo', 'Afastado')),

id\_endereco INT REFERENCES ENDERECO(id\_endereco)

);

-- Tabela PLANO\_CARREIRA

CREATE TABLE PLANO\_CARREIRA (

id\_plano SERIAL PRIMARY KEY,

id\_funcionario INT UNIQUE REFERENCES FUNCIONARIO(id\_funcionario),

cargo\_atual VARCHAR(50) NOT NULL,

nivel VARCHAR(20) NOT NULL,

data\_ultima\_promocao DATE,

proxima\_revisao DATE NOT NULL,

salario\_base DECIMAL(10,2) NOT NULL,

beneficios TEXT

);

-- Tabela PERFIL

CREATE TABLE PERFIL (

id\_perfil SERIAL PRIMARY KEY,

nome VARCHAR(50) UNIQUE NOT NULL,

descricao VARCHAR(255),

permissoes JSONB

);

-- Tabela USUARIO

CREATE TABLE USUARIO (

id\_usuario SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

senha VARCHAR(255) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

id\_funcionario INT UNIQUE REFERENCES FUNCIONARIO(id\_funcionario),

id\_perfil INT REFERENCES PERFIL(id\_perfil),

data\_criacao TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

ultimo\_login TIMESTAMP

);

## 2. Views

-- View para funcionários ativos com informações completas

CREATE OR REPLACE VIEW vw\_funcionarios\_ativos AS

SELECT

f.id\_funcionario,

f.cpf,

f.nome\_completo,

f.email,

f.telefone,

f.data\_admissao,

pc.cargo\_atual,

pc.nivel,

pc.salario\_base,

e.logradouro || ', ' || e.numero || ' - ' || e.bairro || ', ' || e.cidade || '/' || e.estado AS endereco\_completo

FROM FUNCIONARIO f

JOIN PLANO\_CARREIRA pc ON f.id\_funcionario = pc.id\_funcionario

JOIN ENDERECO e ON f.id\_endereco = e.id\_endereco

WHERE f.status = 'Ativo';

-- View para revisões de carreira próximas

CREATE OR REPLACE VIEW vw\_revisoes\_proximas AS

SELECT

f.id\_funcionario,

f.nome\_completo,

pc.cargo\_atual,

pc.nivel,

pc.proxima\_revisao,

(pc.proxima\_revisao - CURRENT\_DATE) AS dias\_para\_revisao

FROM FUNCIONARIO f

JOIN PLANO\_CARREIRA pc ON f.id\_funcionario = pc.id\_funcionario

WHERE f.status = 'Ativo'

AND pc.proxima\_revisao BETWEEN CURRENT\_DATE AND (CURRENT\_DATE + INTERVAL '30 days')

ORDER BY pc.proxima\_revisao;

-- View para aniversários de empresa por mês

CREATE OR REPLACE VIEW vw\_aniversarios\_empresa AS

SELECT

f.id\_funcionario,

f.nome\_completo,

f.data\_admissao,

EXTRACT(YEAR FROM AGE(CURRENT\_DATE, f.data\_admissao)) AS anos\_na\_empresa,

pc.cargo\_atual,

pc.nivel

FROM FUNCIONARIO f

JOIN PLANO\_CARREIRA pc ON f.id\_funcionario = pc.id\_funcionario

WHERE f.status = 'Ativo'

ORDER BY EXTRACT(MONTH FROM f.data\_admissao), EXTRACT(DAY FROM f.data\_admissao);

## 3. Funções

-- Função para promoção de funcionário

CREATE OR REPLACE FUNCTION promover\_funcionario(

p\_id\_funcionario INT,

p\_novo\_cargo VARCHAR(50),

p\_novo\_nivel VARCHAR(20),

p\_novo\_salario DECIMAL(10,2)

)

RETURNS VOID AS $$

BEGIN

UPDATE PLANO\_CARREIRA

SET cargo\_atual = p\_novo\_cargo,

nivel = p\_novo\_nivel,

salario\_base = p\_novo\_salario,

data\_ultima\_promocao = CURRENT\_DATE,

proxima\_revisao = CURRENT\_DATE + INTERVAL '1 year'

WHERE id\_funcionario = p\_id\_funcionario;

END;

$$ LANGUAGE plpgsql;

-- Função para calcular anos de empresa

CREATE OR REPLACE FUNCTION calcular\_anos\_empresa(p\_id\_funcionario INT)

RETURNS INT AS $$

DECLARE

v\_anos INT;

BEGIN

SELECT EXTRACT(YEAR FROM AGE(CURRENT\_DATE, data\_admissao)) INTO v\_anos

FROM FUNCIONARIO

WHERE id\_funcionario = p\_id\_funcionario;

RETURN v\_anos;

END;

$$ LANGUAGE plpgsql;

-- Função para gerar relatório de funcionários por cargo

CREATE OR REPLACE FUNCTION relatorio\_por\_cargo(p\_cargo VARCHAR(50))

RETURNS TABLE (

id\_funcionario INT,

nome\_completo VARCHAR(100),

cpf VARCHAR(11),

data\_admissao DATE,

anos\_na\_empresa INT,

salario\_base DECIMAL(10,2)

) AS $$

BEGIN

RETURN QUERY

SELECT

f.id\_funcionario,

f.nome\_completo,

f.cpf,

f.data\_admissao,

calcular\_anos\_empresa(f.id\_funcionario) AS anos\_na\_empresa,

pc.salario\_base

FROM FUNCIONARIO f

JOIN PLANO\_CARREIRA pc ON f.id\_funcionario = pc.id\_funcionario

WHERE pc.cargo\_atual = p\_cargo

AND f.status = 'Ativo'

ORDER BY f.nome\_completo;

END;

$$ LANGUAGE plpgsql;

## 

## 4. Gatilhos (Triggers)

-- Função de trigger para atualizar revisão de carreira

CREATE OR REPLACE FUNCTION tg\_atualizar\_revisao\_carreira()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.data\_admissao IS DISTINCT FROM OLD.data\_admissao THEN

UPDATE PLANO\_CARREIRA

SET proxima\_revisao = NEW.data\_admissao + INTERVAL '1 year'

WHERE id\_funcionario = NEW.id\_funcionario;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Trigger para atualizar revisão de carreira

CREATE TRIGGER tr\_atualizar\_revisao\_carreira

AFTER UPDATE OF data\_admissao ON FUNCIONARIO

FOR EACH ROW

EXECUTE FUNCTION tg\_atualizar\_revisao\_carreira();

-- Função de trigger para validar CPF único

CREATE OR REPLACE FUNCTION tg\_validar\_cpf\_unico()

RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT 1 FROM FUNCIONARIO WHERE cpf = NEW.cpf AND id\_funcionario != NEW.id\_funcionario) THEN

RAISE EXCEPTION 'CPF já cadastrado no sistema';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Trigger para validar CPF único

CREATE TRIGGER tr\_validar\_cpf\_unico

BEFORE INSERT OR UPDATE ON FUNCIONARIO

FOR EACH ROW

EXECUTE FUNCTION tg\_validar\_cpf\_unico();

-- Função de trigger para criar plano de carreira ao inserir funcionário

CREATE OR REPLACE FUNCTION tg\_criar\_plano\_carreira()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO PLANO\_CARREIRA (

id\_funcionario,

cargo\_atual,

nivel,

proxima\_revisao,

salario\_base

) VALUES (

NEW.id\_funcionario,

'Assistente',

'Júnior',

NEW.data\_admissao + INTERVAL '1 year',

2000.00

);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Trigger para criar plano de carreira

CREATE TRIGGER tr\_criar\_plano\_carreira

AFTER INSERT ON FUNCIONARIO

FOR EACH ROW

EXECUTE FUNCTION tg\_criar\_plano\_carreira();

## 5. Exemplos de Uso

-- Inserindo dados de exemplo

INSERT INTO ENDERECO (logradouro, numero, complemento, bairro, cidade, estado, cep)

VALUES ('Rua das Flores', '123', 'Apto 101', 'Centro', 'São Paulo', 'SP', '01001000');

INSERT INTO FUNCIONARIO (cpf, nome\_completo, data\_nascimento, email, telefone, data\_admissao, id\_endereco)

VALUES ('12345678901', 'João da Silva', '1980-05-15', 'joao.silva@empresa.com', '11987654321', '2020-01-10', 1);

-- Usando a função de promoção

SELECT promover\_funcionario(1, 'Analista', 'Pleno', 3500.00);

-- Consultando a view de funcionários ativos

SELECT \* FROM vw\_funcionarios\_ativos;

-- Usando a função de relatório por cargo

SELECT \* FROM relatorio\_por\_cargo('Analista');

-- Atualizando data de admissão (disparará o trigger)

UPDATE FUNCIONARIO SET data\_admissao = '2019-01-10' WHERE id\_funcionario = 1;

## 

## 6. Índices para Melhoria de Performance

-- Índices para consultas frequentes

CREATE INDEX idx\_funcionario\_status ON FUNCIONARIO(status);

CREATE INDEX idx\_plano\_carreira\_cargo ON PLANO\_CARREIRA(cargo\_atual);

CREATE INDEX idx\_plano\_carreira\_revisao ON PLANO\_CARREIRA(proxima\_revisao);

CREATE INDEX idx\_funcionario\_nome ON FUNCIONARIO(nome\_completo);

Este projeto implementado para PostgreSQL inclui:

1. Estrutura completa do banco de dados com todas as tabelas necessárias
2. Views para consultas frequentes e relatórios
3. Funções para operações complexas e reutilização de código
4. Triggers para automatizar validações e processos
5. Índices para otimização de consultas

As principais vantagens desta implementação:

* Garantia de integridade dos dados através de constraints e triggers
* Automatização de processos como atualização de planos de carreira
* Facilidade de geração de relatórios através das views e funções
* Boa performance graças aos índices estratégicos
* Flexibilidade com o tipo JSONB para armazenar permissões