

**BACHARELADO EM**

**CIÊNCIA DA COMPUTAÇÃO**

**THIAGO BRUCHMANN CARNAIBA**

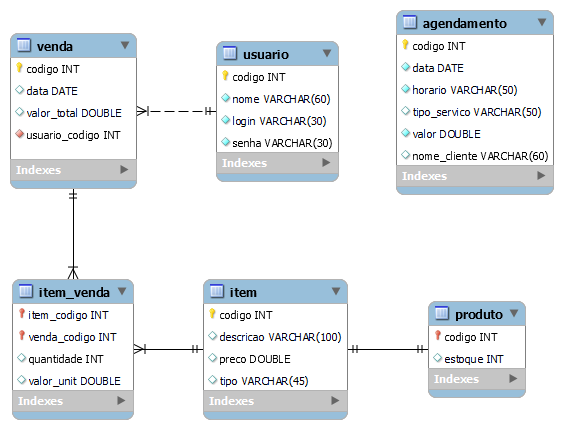
**PROJETO INTEGRADOR – BASE DE DADOS**

BarberSystem

Presidente Epitácio – SP

2021

Diagrama feito no MySQL



Criação das TABLE para PostgreSQL

CREATE TABLE IF NOT EXISTS item (

codigo SERIAL,

descricao VARCHAR(100) NULL,

preco DOUBLE PRECISION NULL,

tipo VARCHAR(45) NULL,

PRIMARY KEY (codigo));

CREATE TABLE IF NOT EXISTS produto (

codigo INT NOT NULL,

estoque INT NULL,

PRIMARY KEY (codigo),

FOREIGN KEY (codigo) REFERENCES item (codigo));

CREATE TABLE IF NOT EXISTS usuario (

codigo SERIAL,

nome VARCHAR(60) NOT NULL,

login VARCHAR(30) NOT NULL,

senha VARCHAR(30) NOT NULL,

PRIMARY KEY (codigo));

CREATE TABLE IF NOT EXISTS venda (

codigo SERIAL,

data DATE NULL,

valor\_total DOUBLE PRECISION NULL,

usuario\_codigo INT NOT NULL,

PRIMARY KEY (codigo),

FOREIGN KEY (usuario\_codigo) REFERENCES usuario (codigo));

CREATE TABLE IF NOT EXISTS agendamento (

codigo SERIAL,

data DATE NOT NULL,

horario VARCHAR(50) NOT NULL,

tipo\_servico VARCHAR(50) NULL DEFAULT NULL,

valor DOUBLE PRECISION NOT NULL,

nome\_cliente VARCHAR(60) NULL DEFAULT NULL,

PRIMARY KEY (codigo));

CREATE TABLE IF NOT EXISTS item\_venda (

item\_codigo INT NOT NULL,

venda\_codigo INT NOT NULL,

quantidade INT NULL,

valor\_unit DOUBLE PRECISION NULL,

PRIMARY KEY (item\_codigo, venda\_codigo),

FOREIGN KEY (item\_codigo) REFERENCES item (codigo),

FOREIGN KEY (venda\_codigo) REFERENCES venda (codigo));

Criação das TRIGGER para PostgreSQL

CREATE OR REPLACE FUNCTION atualiza\_estoque()

RETURNS trigger AS $$

BEGIN

IF (TG\_OP='INSERT') THEN

UPDATE produto SET estoque = estoque - new.quantidade WHERE codigo = new.item\_codigo;

RETURN NEW;

END IF;

IF (TG\_OP='UPDATE') THEN

UPDATE produto SET estoque = estoque + new.quantidade - old.quantidade WHERE codigo = new.item\_codigo;

RETURN NEW;

END IF;

if (TG\_OP='DELETE') THEN

UPDATE produto SET estoque = estoque + old.quantidade WHERE codigo = new.item\_codigo;

RETURN OLD;

END IF;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER atualiza\_estoque AFTER INSERT OR UPDATE OR DELETE ON item\_venda

FOR EACH ROW EXECUTE PROCEDURE atualiza\_estoque();

CREATE OR REPLACE FUNCTION atualiza\_venda()

RETURNS trigger AS $$

BEGIN

IF (TG\_OP='INSERT') THEN

UPDATE venda SET valor\_total = valor\_total + new.quantidade \* new.valor\_unit WHERE codigo = new.venda\_codigo;

RETURN NEW;

END IF;

IF (TG\_OP='UPDATE') THEN

UPDATE venda SET valor\_total = valor\_total + new.quantidade - old.quantidade \* new.valor\_unit - old.valor\_unit WHERE codigo = new.venda\_codigo;

RETURN NEW;

END IF;

if (TG\_OP='DELETE') THEN

UPDATE venda SET valor\_total = valor\_total - old.quantidade \* old.valor\_unit WHERE codigo = old.venda\_codigo;

RETURN OLD;

END IF;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER atualiza\_venda AFTER INSERT OR UPDATE OR DELETE ON item\_venda

FOR EACH ROW EXECUTE PROCEDURE atualiza\_venda();

CREATE OR REPLACE FUNCTION atualiza\_valor()

RETURNS trigger AS $$

BEGIN

IF (TG\_OP='UPDATE') THEN

UPDATE item\_venda SET valor\_unit = new.preco WHERE codigo = item\_codigo;

RETURN NEW;

END IF;

if (TG\_OP='DELETE') THEN

UPDATE item\_venda SET valor\_unit = 0 WHERE old.codigo = item\_codigo;

RETURN OLD;

END IF;

END;

$$ LANGUAGE PLPGSQL;

CREATE TRIGGER atualiza\_valor AFTER UPDATE OR DELETE ON item

FOR EACH ROW EXECUTE PROCEDURE atualiza\_valor();