BACHARELADO EM
CIÊNCIA DA COMPUTAÇÃO

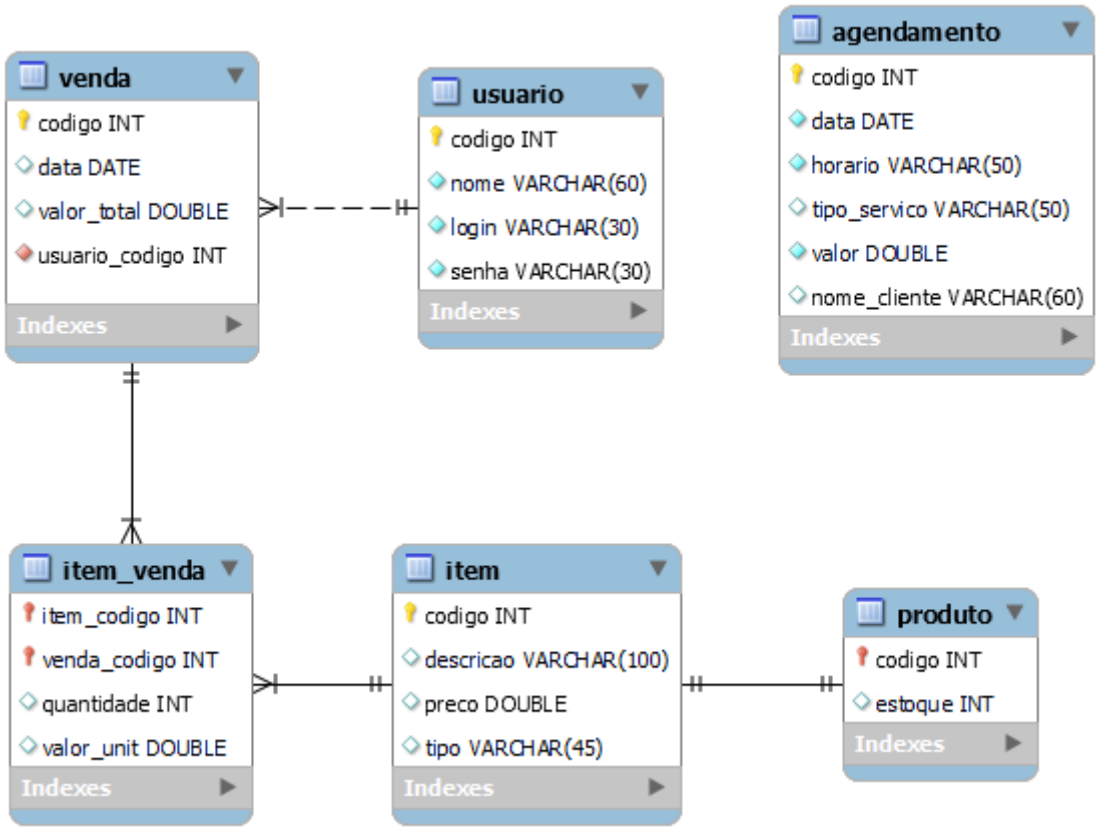
THIAGO BRUCHMANN CARNAIBA

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();
```