# Centro Universitário UNISATC

Engenharia de Software 3a fase – Banco de Dados II – Prof. **CRISTIANE PAVEI MARTINELLO FERNANDES**

# TRABALHO FINAL COM BASE EM METODOLOGIAS ATIVAS DE APRENDIZAGEM

**Projeto de banco de dados para um sistema de Suprimentos ( PratoCheio.com )**


Gianluca Andrade Silvestre

Murilo Salvan

Roger Balcevicz

Contas Git

Roger-Balcevicz

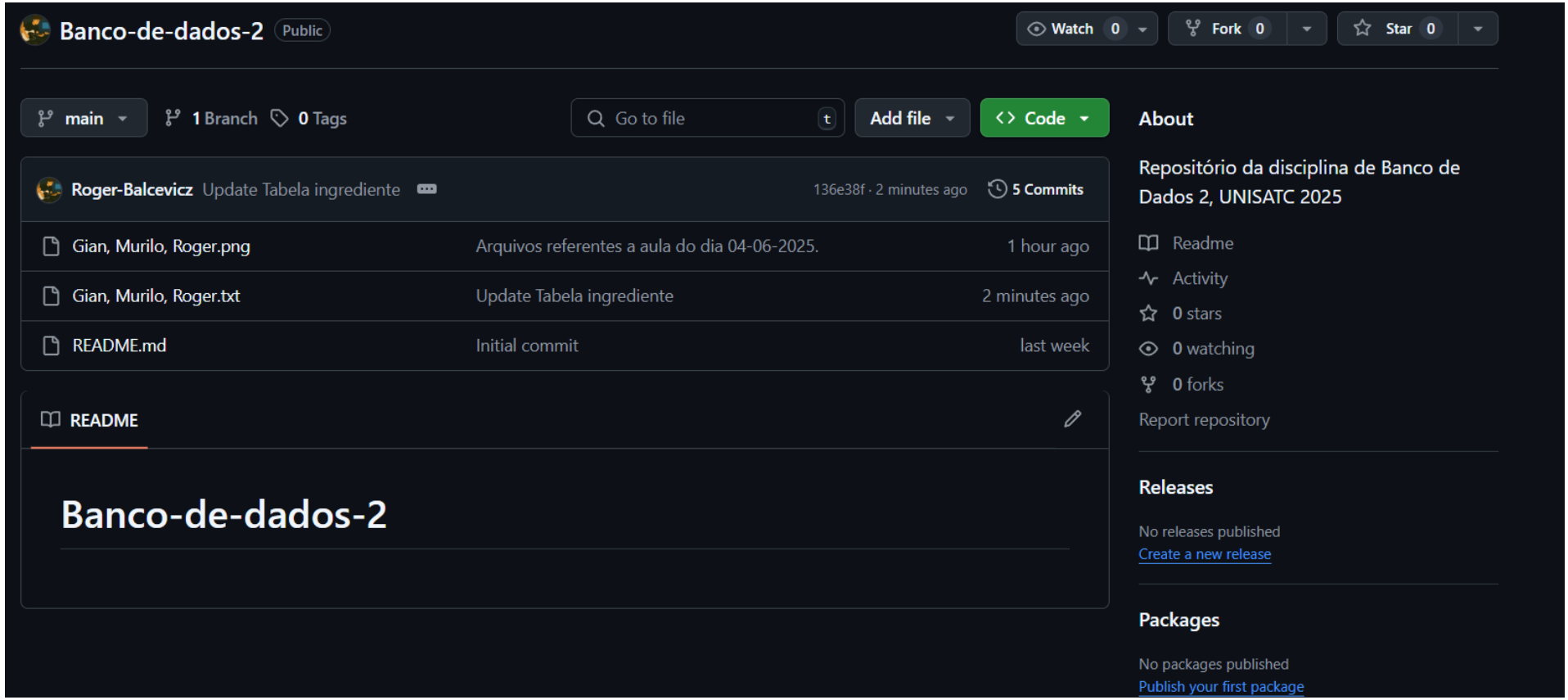
GiaNinWorld

omrl

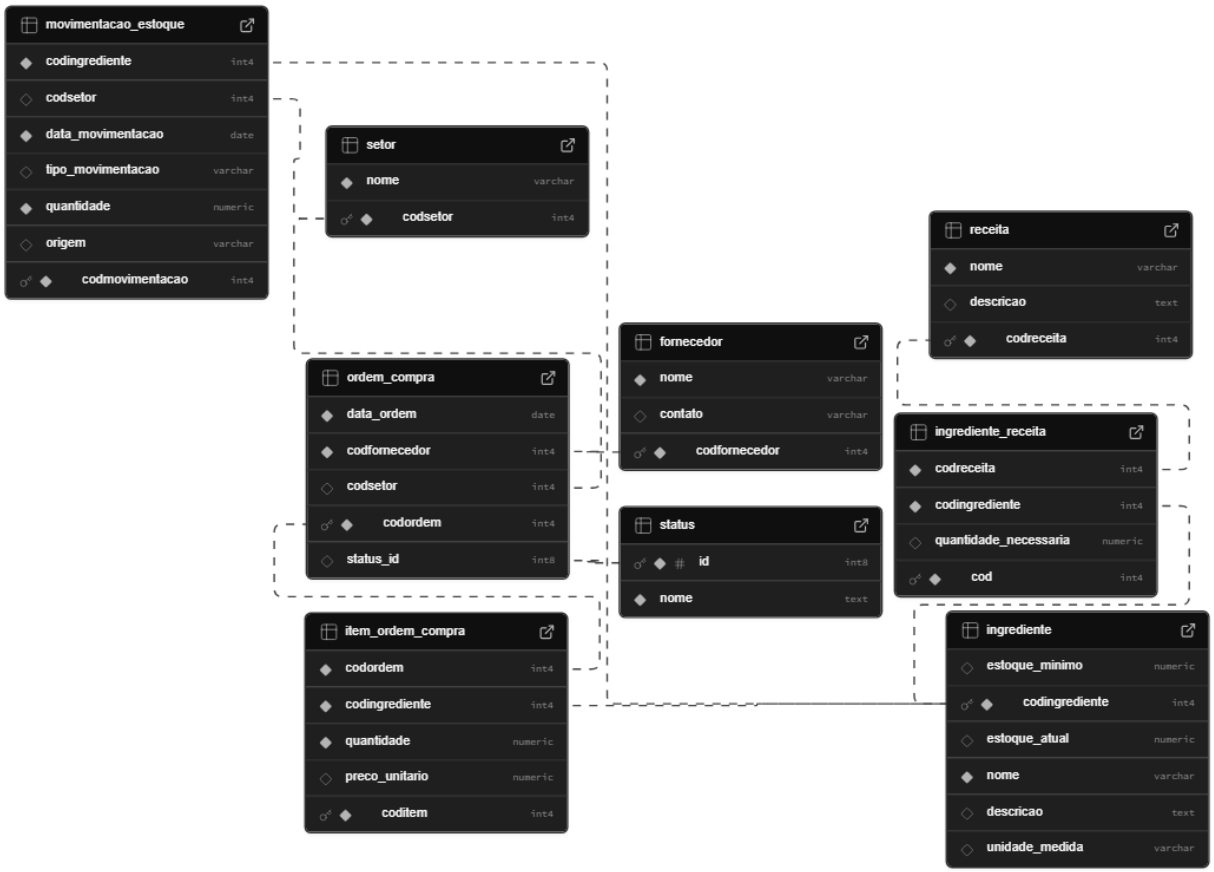

Criciúma, 11/06/2025

**URL do projeto no GitHub**

https://github.com/Roger-Balcevicz/Banco-de-dados-2



# Modelo ER Físico



# Dicionário de Dados

| **Tabela** | Ingrediente |
|---|---|
| **Descrição** | Tabela responsável por armazenar os dados dos ingredientes utilizados em receitas e compras. codingrediente |

| **Atributos** | | | | | | |
|---|---|---|---|---|---|---|
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| codingrediente | serial | 1 – sem limite | NOT NULL | X | | Código de identificador do ingrediente |
| nome | varchar(100) | 1 - 100 | NOT NULL | | | Nome do ingrediente |
| descricao | text | - | NULL | | | Descrição do ingrediente |
| unidade_medida | varchar(20) | - | NULL | | | Unidade de medida (kg, litro, etc.) |
| estoque_atual | decimal(10,2) | >= 0 | NULL | | | Quantidade atual em estoque |
| estoque_minimo | int | - | NULL | | | Quantidade mínima de estoque |
| | | | | | | |

| **Índice** | | | | |
|---|---|---|---|---|
| **Nome do índice** | **Clustered** | **NonClustered** | **Unique** | **Colunas** |
| idx_nome_ingrediente | | X | | nome |
| idx_estoque_atual | | X | | estoque_atual |

| **Tabela** | Fornecedor |
|---|---|
| **Descrição** | Tabela com os dados cadastrais dos fornecedores. |

| **Atributos** | | | | | | |
|---|---|---|---|---|---|---|
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| codfornecedor | serial | 1 – sem limite | NOT NULL | X | | Código do fornecedor |
| nome | varchar(100) | 1 - 100 | NOT NULL | | | Nome do fornecedor |
| contato | varchar(100) | - | NULL | | | Informação de contato do fornecedor |
| | | | | | | |

| **Índice** | | | | |
|---|---|---|---|---|
| **Nome do índice** | **Clustered** | **NonClustered** | **Unique** | **Colunas** |
| idx_nome_fornecedor | | X | | nome |

| **Tabela** | ordem_compra | | | | | |
|---|---|---|---|---|---|---|
| **Descrição** | Tabela que armazena as ordens de compra emitidas. | | | | | |
| **Atributos** | | | | | | |
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| codordem | serial | 1 – sem limite | NOT NULL | X | | Código da ordem de compra |
| data_ordem | date | - | NOT NULL | | | Data de emissão da ordem |
| codfornecedor | int | - | NOT NULL | | | Código do fornecedor |
| codsetor | int | - | NULL | | | Código do setor solicitante |
| status | varchar(50) | - | NULL | | | Status atual da ordem |
| | | | | | | |
| **Índice** | | | | | | |
| **Nome do índice** | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx_status_ordem | | X | | Data_ordem | | |
| idx_data_ordem | | X | | status | | |

| **Tabela** | item_ordem_compra | | | | | |
|---|---|---|---|---|---|---|
| **Descrição** | Itens associados a uma ordem de compra. | | | | | |
| **Atributos** | | | | | | |
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| coditem | serial | 1 – sem limite | NOT NULL | X | | Código do item |
| codordem | int | - | NOT NULL | | X | Código da ordem de compra |
| codingrediente | int | - | NOT NULL | | X | Ingrediente comprado |
| quantidade | decimal(10,2) | >=0 | NOT NULL | | | Quantidade adquirida |
| Preco_unitario | decimal(10,2) | >=0 | NULL | | | Preço por unidade |
| | | | | | | |
| **Índice** | | | | | | |
| **Nome do índice** | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx_ingrediente_item | | X | | codingrediente | | |

| Tabela | movimentacao_estoque | | | | | |
|---|---|---|---|---|---|---|
| Descrição | Registra entradas e saídas de ingredientes do estoque | | | | | |
| **Atributos** | | | | | | |
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| codmovimentacao | serial | - | NOT NULL | X | | Código da movimentação |
| codingrediente | int | - | NOT NULL | | X | Ingrediente movimentado |
| codsetor | int | - | NULL | | X | Setor que realizou a movimentação |
| data_movimentacao | date | - | NOT NULL | | | Data da movimentação |
| tipo_movimentacao | varchar(20) | - | NOT NULL | | | Tipo da movimentação |
| quantidade | decimal(10,2) | - | NOT NULL | | | Quantidade movimentada |
| origem | varchar(100) | - | NULL | | | Origem da movimentação (ex: produção, ajuste) |
| | | | | | | |
| **Índice** | | | | | | |
| **Nome do índice** | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx_tipo_data | | X | | tipo_movimentacao, data_movimentacao | | |

| Tabela | setor | | | | | |
|---|---|---|---|---|---|---|
| Descrição | Cadastro dos setores internos do restaurante. | | | | | |
| **Atributos** | | | | | | |
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| codsetor | serial | - | NOT NULL | X | | Código do setor |
| nome | varchar(100) | - | NOT NULL | | | Nome do setor |

| Tabela | receita | | | | | |
|---|---|---|---|---|---|---|
| Descrição | Cadastro de receitas do cardápio. | | | | | |
| **Atributos** | | | | | | |
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| codreceita | serial | - | NOT NULL | X | | Código da receita |
| nome | varchar(100) | - | NOT NULL | | | Nome do prato |
| descricao | Int | - | NULL | | | Descrição detalhada da receita |
| | | | | | | |
| **Índice** | | | | | | |
| **Nome do índice** | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx_nome_receita | | X | | nome | | |

| **Tabela** | ingrediente_receita | | | | | |
|---|---|---|---|---|---|---|
| **Descrição** | Tabela que relaciona ingredientes usados em cada receita. | | | | | |
| **Atributos** | | | | | | |
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| cod | serial | - | NOT NULL | X | | Código da associação |
| codreceita | int | - | NOT NULL | | | Receita relacionada |
| codingrediente | int | - | NOT NULL | | | Ingrediente utilizado |
| quantidade_necessaria | decimal(10,2) | >=0 | NOT NULL | | | Quantidade por porção |
| | | | | | | |
| **Índice** | | | | | | |
| **Nome do índice** | **Clustered** | **NonClustered** | **Unique** | **Colunas** | | |
| idx_ingrediente_receita | | X | | codreceita, codingrediente | | |

| **Tabela** | status | | | | | |
|---|---|---|---|---|---|---|
| **Descrição** | Status da ordem de compra | | | | | |
| **Atributos** | | | | | | |
| **Nome da Coluna** | **Tipo do Dado** | **Valor min e max** | **Nulidade** | **PK** | **FK** | **Descrição** |
| id | serial | - | NOT NULL | X | | Código do status |
| nome | text | - | NOT NULL | | | Nome do status |

## Script dos comandos DDL para criação do Banco de dados

```
CREATE TABLE setor (
    codsetor SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL
);

CREATE TABLE ingrediente (
    codingrediente SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    descricao TEXT,
    unidade_medida VARCHAR(20),
    estoque_atual DECIMAL(10,2) DEFAULT 0
);

CREATE TABLE fornecedor (
    codfornecedor SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    contato VARCHAR(100)
);

CREATE TABLE ordem_compra (
    codordem SERIAL PRIMARY KEY,
    data_ordem DATE NOT NULL,
    codfornecedor INTEGER NOT NULL,
    codsetor INTEGER, -- quem solicitou
    status VARCHAR(50) DEFAULT 'pendente',
    CONSTRAINT fk_ordem_setor FOREIGN KEY (codfornecedor) REFERENCES fornecedor(codfornecedor),
    CONSTRAINT fk_ordem_fornecedor FOREIGN KEY (codsetor) REFERENCES setor(codsetor)
);
```

```
CREATE TABLE item_ordem_compra (
    coditem SERIAL PRIMARY KEY,
    codordem INTEGER NOT NULL,
    codingrediente INTEGER NOT NULL,
    quantidade DECIMAL(10,2) NOT NULL,
    preco_unitario DECIMAL(10,2),
    CONSTRAINT fk_item_ordem FOREIGN KEY (codordem) REFERENCES ordem_compra(codordem),
    CONSTRAINT fk_item_ingrediente FOREIGN KEY (codingrediente) REFERENCES ingrediente(codingrediente)
);



CREATE TABLE movimentacao_estoque (
    codmovimentacao SERIAL PRIMARY KEY,
    codingrediente INTEGER NOT NULL,
    codsetor INTEGER,
    data_movimentacao DATE NOT NULL,
    tipo_movimentacao VARCHAR(20) CHECK (tipo_movimentacao IN ('entrada', 'saida', 'ajuste', 'produção')),
    quantidade DECIMAL(10,2) NOT NULL,
    origem VARCHAR(100),
    CONSTRAINT fk_movimentacao_ingrediente FOREIGN KEY (codingrediente) REFERENCES ingrediente(codingrediente),
    CONSTRAINT fk_movimentacao_setor FOREIGN KEY (codsetor) REFERENCES setor(codsetor)
);

CREATE TABLE receita (
    codreceita SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    descricao TEXT
);

CREATE TABLE ingrediente_receita (
    cod SERIAL PRIMARY KEY,
    codreceita INTEGER NOT NULL,
    codingrediente INTEGER NOT NULL,
    quantidade_necessaria DECIMAL(10,2),
    CONSTRAINT fk_receita_ingrediente_receita FOREIGN KEY (codreceita) REFERENCES receita(codreceita),
    CONSTRAINT fk_ingrediente_ingrediente_receita FOREIGN KEY (codingrediente) REFERENCES ingrediente(codingrediente)
);
```

```
CREATE TABLE status_ordem (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(50) NOT NULL
);
```

**-- Alteração na tabela ingrediente: adicionado campo estoque_minimo**

ALTER TABLE ingrediente

ADD COLUMN estoque_minimo NUMERIC DEFAULT 10;

**-- Adiciona a coluna codstatus com valor padrão 'pendente' (id = 1)**
ALTER TABLE ordem_compra
ADD COLUMN codstatus INT NOT NULL DEFAULT 1;

**-- Cria a chave estrangeira codstatus → status_ordem(id)**
ALTER TABLE ordem_compra
ADD CONSTRAINT fk_ordem_status
FOREIGN KEY (codstatus)
REFERENCES status_ordem(id);

## Script que popula as tabelas do Banco de dados

```
-- INSERTS PARA A TABELA SETOR

INSERT INTO setor (nome) VALUES

('Cozinha Principal'),

('Bar'),

('Confeitaria'),

('Churrasqueira'),

('Sushi Bar'),

('Delivery'),

('Salão'),

('Administração'),

('Estoque Seco'),

(‘Geladeira Central’);


-- INSERTS PARA A TABELA INGREDIENTE

INSERT INTO ingrediente (nome, descricao, unidade_medida, estoque_atual) VALUES

('Farinha de Trigo', 'Ingrediente de cozinha', 'kg', 50),

('Leite Integral', 'Ingrediente de cozinha', 'litro', 30),

('Ovo', 'Ingrediente de cozinha', 'unidade', 200),

('Carne Moída', 'Ingrediente de cozinha', 'kg', 20),

('Cebola', 'Ingrediente de cozinha', 'kg', 15),

('Queijo Mussarela', 'Ingrediente de cozinha', 'kg', 25),
```

('Tomate', 'Ingrediente de cozinha', 'kg', 18),

('Açúcar', 'Ingrediente de cozinha', 'kg', 40),

('Manteiga', 'Ingrediente de cozinha', 'kg', 12),

('Fermento Biológico', 'Ingrediente de cozinha', 'g', 5);

-- INSERTS PARA A TABELA FORNECEDOR

INSERT INTO fornecedor (nome, contato) VALUES

('Distribuidora Alimentos LTDA', 'distribuidoraalimento@gmail.com'),

('Laticínios Bom Leite', 'laticiniosbomleite@gmail.com'),

('Hortifruti Central', 'hortifruticentral@gmail.com'),

('Carnes Premium', 'carnespremium@gmail.com'),

('Ovos São Pedro', 'ovossaopedro@gmail.com'),

('Queijos Mineiros', 'queijosmineiros@gmail.com'),

('Panifício Real', 'panificioreal@gmail.com'),

('Doces da Vó', 'docesdavo@gmail.com'),

('Sabor e Qualidade', 'saborequalidade@gmail.com'),

('Leites Vale Verde', 'leitesvaleverde@gmail.com');

```
-- INSERTS PARA A TABELA ORDEM_COMPRA
INSERT INTO ordem_compra (data_ordem, codfornecedor, codsetor, codstatus) VALUES
('2025-06-01', 1, 1, 1),
('2025-06-02', 2, 2, 1),
('2025-06-03', 3, 3, 1),
('2025-06-04', 4, 1, 1),
('2025-06-05', 5, 2, 1),
('2025-06-06', 6, 4, 1),
('2025-06-07', 7, 5, 1),
('2025-06-08', 8, 6, 1),
('2025-06-09', 9, 7, 1),
('2025-06-10', 10, 1, 1);

-- INSERTS PARA A TABELA ITEM_ORDEM_COMPRA
INSERT INTO item_ordem_compra (codordem, codingrediente, quantidade, preco_unitario) VALUES
(1, 1, 10.0, 2.5),
(2, 2, 10.0, 3.0),
(3, 3, 10.0, 0.5),
(4, 4, 10.0, 12.0),
(5, 5, 10.0, 2.0),
(6, 6, 10.0, 18.0),
(7, 7, 10.0, 3.5),
```

```
(8, 8, 10.0, 4.0),
(9, 9, 10.0, 9.0),
(10, 10, 10.0, 0.05);

-- INSERTS PARA A TABELA MOVIMENTACAO_ESTOQUE
INSERT INTO movimentacao_estoque (codingrediente, codsetor, data_movimentacao, tipo_movimentacao, quantidade, origem) VALUES
(1, 1, '2025-06-05', 'entrada', 10.0, 'Ordem de Compra 1'),
(2, 1, '2025-06-06', 'entrada', 10.0, 'Ordem de Compra 2'),
(3, 1, '2025-06-07', 'entrada', 30.0, 'Ordem de Compra 3'),
(4, 2, '2025-06-08', 'produção', -3.0, 'Receita Lasanha'),
(5, 2, '2025-06-09', 'produção', -5.0, 'Receita Lasanha'),
(6, 3, '2025-06-10', 'entrada', 5.0, 'Ordem de Compra 6'),
(7, 3, '2025-06-11', 'produção', -2.0, 'Receita Pizza'),
(8, 4, '2025-06-12', 'produção', -1.0, 'Receita Doce'),
(9, 4, '2025-06-13', 'entrada', 7.0, 'Ordem de Compra 9'),
(10, 5, '2025-06-14', 'produção', -0.5, 'Receita Bolo');

-- INSERTS PARA A TABELA RECEITA
INSERT INTO receita (nome, descricao) VALUES
('Lasanha Bolonhesa', 'Camadas de massa e carne com molho'),
('Bolo de Leite', 'Bolo simples com leite integral'),
('Quiche de Alho-Poró', 'Torta salgada com alho-poró'),
```

('Sanduíche Natural', 'Sanduíche leve com frango'),

('Sopa de Legumes', 'Caldo com vegetais variados'),

('Torta de Frango', 'Torta recheada com frango desfiado'),

('Pizza Margherita', 'Pizza com tomate e manjericão'),

('Salada Tropical', 'Salada com frutas e folhas verdes'),

('Escondidinho', 'Prato de carne com purê de batata'),

('Brigadeiro', 'Doce de chocolate tradicional');

-- INSERTS PARA A TABELA INGREDIENTE_RECEITA

INSERT INTO ingrediente_receita (codreceita, codingrediente, quantidade_necessaria) VALUES

(1, 1, 2.0),

(1, 4, 1.5),

(2, 2, 1.0),

(2, 3, 2.0),

(3, 5, 0.5),

(4, 6, 2.5),

(5, 7, 1.2),

(6, 8, 1.8),

(7, 9, 0.4),

(10, 8, 2.0);

## Script de functions, triggers, views.

```
-- ================================
-- FUNCTIONS
-- ================================

-- 1. Calcular custo da receita
CREATE OR REPLACE FUNCTION fn_calcular_custo_receita(cod_receita INT)
RETURNS NUMERIC AS $$
DECLARE total NUMERIC := 0;
BEGIN
  SELECT SUM(ir.quantidade_necessaria * COALESCE(i.preco_unitario, 0))
  INTO total
  FROM ingrediente_receita ir
  JOIN ingrediente i ON ir.codingrediente = i.codingrediente
  WHERE ir.codreceita = cod_receita;
  RETURN COALESCE(total, 0);
END;
$$ LANGUAGE plpgsql;

-- 2. Estoque atual de um ingrediente
CREATE OR REPLACE FUNCTION fn_quantidade_disponivel(id INT)
RETURNS NUMERIC AS $$
DECLARE total NUMERIC := 0;
BEGIN
  SELECT estoque_atual INTO total FROM ingrediente WHERE codingrediente = id;
  RETURN COALESCE(total, 0);
END;
$$ LANGUAGE plpgsql;

-- 3. Inserir ordem de compra
CREATE OR REPLACE FUNCTION fn_inserir_ordem_compra(
  p_data DATE,
  p_codfornecedor INT,
```

```
  p_codsetor INT,
  p_codstatus INT,
  p_ingredientes INT[],
  p_quantidades NUMERIC[],
  p_precos NUMERIC[]
)
RETURNS INT AS $$
DECLARE nova_ordem_id INT;
BEGIN
  INSERT INTO ordem_compra (data_ordem, codfornecedor, codsetor, codstatus)
  VALUES (p_data, p_codfornecedor, p_codsetor, p_codstatus)
  RETURNING codordem INTO nova_ordem_id;

  FOR i IN 1..array_length(p_ingredientes, 1) LOOP
    INSERT INTO item_ordem_compra (codordem, codingrediente, quantidade, preco_unitario)
    VALUES (nova_ordem_id, p_ingredientes[i], p_quantidades[i], p_precos[i]);
  END LOOP;

  RETURN nova_ordem_id;
END;
$$ LANGUAGE plpgsql;

-- 4. Registrar movimentação
CREATE OR REPLACE FUNCTION fn_registrar_movimentacao(
  p_codingrediente INT,
  p_codsetor INT,
  p_data DATE,
  p_tipo VARCHAR,
  p_quantidade NUMERIC,
  p_origem VARCHAR
)
RETURNS TEXT AS $$
BEGIN
  INSERT INTO movimentacao_estoque (codingrediente, codsetor, data_movimentacao, tipo_movimentacao, quantidade,
origem)
  VALUES (p_codingrediente, p_codsetor, p_data, p_tipo, p_quantidade, p_origem);

  RETURN 'Movimentação registrada.';
END;
```

```
$$ LANGUAGE plpgsql;

-- 5. Ajustar estoque
CREATE OR REPLACE FUNCTION fn_ajustar_estoque(
  p_codingrediente INT,
  p_quantidade NUMERIC,
  p_observacao VARCHAR
)
RETURNS TEXT AS $$
BEGIN
  PERFORM fn_registrar_movimentacao(
    p_codingrediente, NULL, CURRENT_DATE, 'ajuste', p_quantidade, p_observacao
  );
  RETURN 'Estoque ajustado.';
END;
$$ LANGUAGE plpgsql;

--6. Previsão de reposição
CREATE OR REPLACE FUNCTION fn_previsao_reposicao(id INT)
RETURNS DATE AS $$
DECLARE
  consumo_medio NUMERIC;
  estoque NUMERIC;
  dias_restantes INT;
BEGIN
  SELECT AVG(ABS(quantidade)) INTO consumo_medio
  FROM movimentacao_estoque
  WHERE codingrediente = id AND tipo_movimentacao = 'saida';

  SELECT estoque_atual INTO estoque FROM ingrediente WHERE codingrediente = id;

  IF consumo_medio IS NULL OR consumo_medio = 0 THEN
    RETURN NULL;
  END IF;

  dias_restantes := estoque / consumo_medio;
  RETURN CURRENT_DATE + dias_restantes;
END;
$$ LANGUAGE plpgsql;
```

```
-- ================================
-- TRIGGERS
-- ================================

-- Atualiza estoque atual na tabela ingrediente sempre que houver movimentação
CREATE OR REPLACE FUNCTION fn_atualiza_estoque()
RETURNS TRIGGER AS $$
BEGIN
  UPDATE ingrediente
  SET estoque_atual = (
    SELECT COALESCE(SUM(quantidade), 0)
    FROM movimentacao_estoque
    WHERE codingrediente = NEW.codingrediente
  )
  WHERE codingrediente = NEW.codingrediente;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_recalcular_estoque_total
AFTER INSERT OR UPDATE OR DELETE ON movimentacao_estoque
FOR EACH ROW
EXECUTE FUNCTION fn_atualiza_estoque();

-- Dispara aviso quando estoque de um ingrediente ficar abaixo do mínimo
CREATE OR REPLACE FUNCTION fn_aviso_estoque_baixo()
RETURNS TRIGGER AS $$
DECLARE
  v_estoque NUMERIC;
  v_minimo NUMERIC;
  v_nome TEXT;
BEGIN
  SELECT estoque_atual, estoque_minimo, nome
  INTO v_estoque, v_minimo, v_nome
  FROM ingrediente
  WHERE codingrediente = NEW.codingrediente;

  IF v_estoque < v_minimo THEN
```

```
    RAISE NOTICE 'Estoque abaixo do mínimo para "%": atual = %, mínimo = %.',
                 v_nome, v_estoque, v_minimo;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_baixo_estoque
AFTER INSERT OR UPDATE ON movimentacao_estoque
FOR EACH ROW
EXECUTE FUNCTION fn_aviso_estoque_baixo();

-- Gera movimentação de entrada quando a ordem for marcada como recebida
CREATE OR REPLACE FUNCTION fn_receber_ordem_compra()
RETURNS TRIGGER AS $$
DECLARE item RECORD;
BEGIN
  IF NEW.codstatus = 2 AND OLD.codstatus IS DISTINCT FROM 2 THEN
    FOR item IN
      SELECT codingrediente, quantidade
      FROM item_ordem_compra
      WHERE codordem = NEW.codordem
    LOOP
      INSERT INTO movimentacao_estoque (
        codingrediente, codsetor, data_movimentacao, tipo_movimentacao, quantidade, origem
      )
      VALUES (
        item.codingrediente,
        NEW.codsetor,
        CURRENT_DATE,
        'entrada',
        item.quantidade,
        CONCAT('Recebimento da ordem ', NEW.codordem)
      );
    END LOOP;
  END IF;
  RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_recebimento_ordem_compra
AFTER UPDATE OF codstatus ON ordem_compra
FOR EACH ROW
EXECUTE FUNCTION fn_receber_ordem_compra();

-- ================================
-- VIEWS
-- ================================

-- 1. View: Estoque atual
CREATE OR REPLACE VIEW vw_estoque_atual AS
SELECT codingrediente, nome, unidade_medida, estoque_atual, estoque_minimo
FROM ingrediente;

-- 2. View: Ingredientes com estoque abaixo do mínimo
CREATE OR REPLACE VIEW vw_avisos_estoque_baixo AS
SELECT codingrediente, nome, estoque_atual, estoque_minimo
FROM ingrediente
WHERE estoque_atual < estoque_minimo;

-- 3. View: Total de compras por fornecedor
CREATE OR REPLACE VIEW vw_compras_por_fornecedor AS
SELECT f.nome AS fornecedor,
       SUM(ioc.quantidade * ioc.preco_unitario) AS total_compras
FROM item_ordem_compra ioc
JOIN ordem_compra oc ON ioc.codordem = oc.codordem
JOIN fornecedor f ON f.codfornecedor = oc.codfornecedor
GROUP BY f.nome
ORDER BY total_compras DESC;
```