Arquitetura de Dados para a "Campo Inteligente"

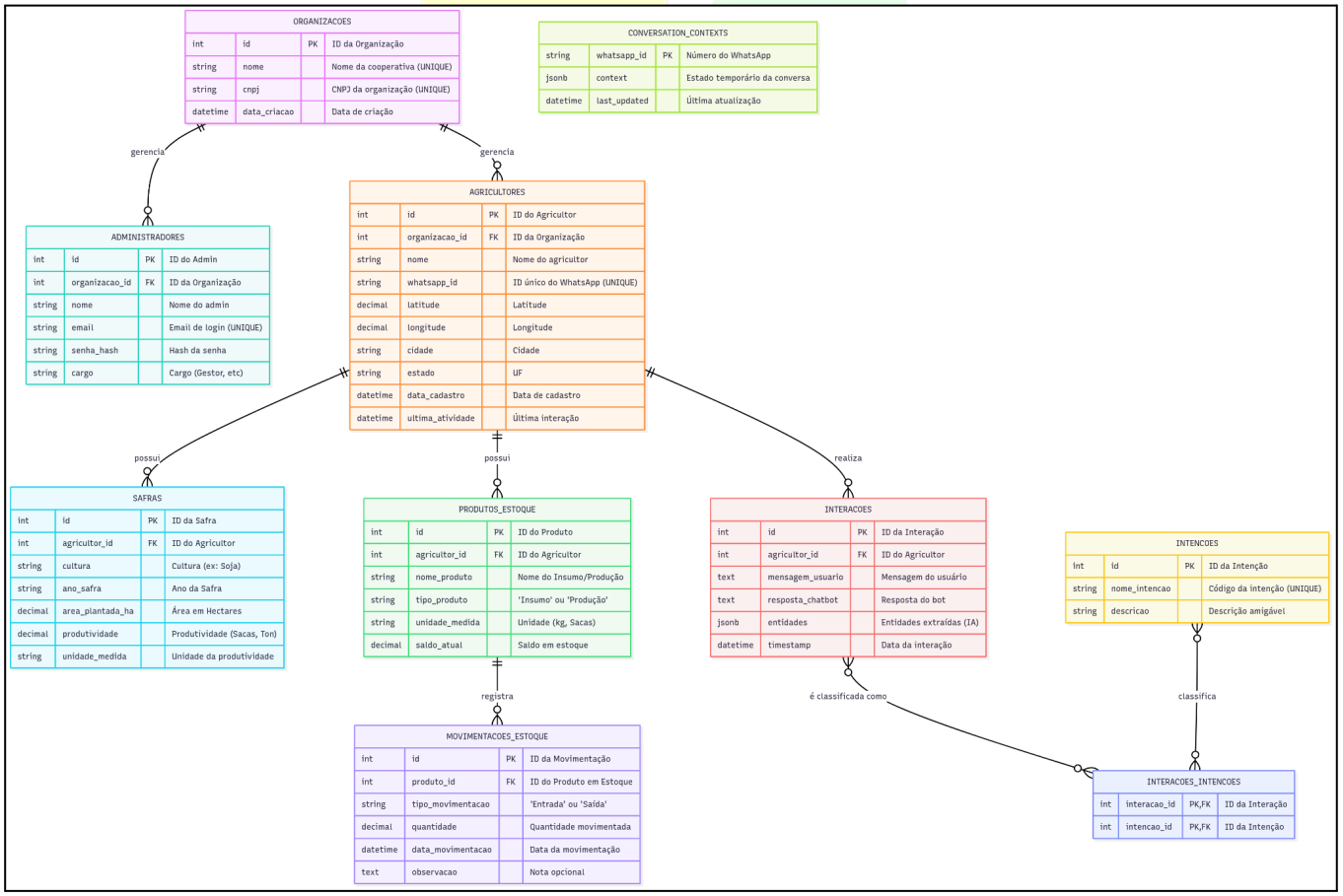
por: Arthur Lago Martins

Propósito: Este documento detalha a arquitetura final do banco de dados para a plataforma "Campo Inteligente". Ele deve ser usado pela equipe de desenvolvimento do chatbot para alinhar a lógica de captura, armazenamento e consulta de dados.

1. Hospedagem do Banco de Dados PostgreSQL

Conforme definido pela equipe, o banco de dados **PostgreSQL** será hospedado em um servidor próprio, gerenciado por Marcos.

2. Diagrama de Relacionamentos (ERD)



[Clique aqui para ter acesso ao diagrama]

3. Descrição Detalhada das Tabelas

Tabela: `Organizacoes`

Propósito: Funciona como a tabela central do sistema de "multi-tenancy". Cada registro aqui é uma "entidade" (cooperativa, empresa, etc.) que é dona de um conjunto de administradores e agricultores. Ela garante que os dados de uma cooperativa não se misturem com os de outra.

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`id`	`SERIAL PRIMARY KEY`	Identificador único da organização.	`1`
`nome`	`VARCHAR(255) UNIQUE`	Nome da cooperativa ou empresa.	`Cooperativa Agroforte`
`cnpj`	`VARCHAR(18) UNIQUE`	CNPJ da organização, se aplicável.	`12.345.678/0001-99`
`data_criacao`	`TIMESTAMPTZ`	Data de cadastro da organização na plataforma.	`2025-07-01...`

Relacionamentos:

- É a **tabela-mãe** principal da arquitetura de permissões. Ela não depende de nenhuma outra tabela.
- Possui um relacionamento de **um-para-muitos (1-N)** com a tabela `Administradores`. Isso significa que **uma** Organização pode ter **vários** Administradores.
- Possui um relacionamento de **um-para-muitos (1-N)** com a tabela `Agricultores`. Isso significa que **uma** Organização pode gerenciar **vários** Agricultores.

Tabela : `Administradores`

- **Propósito:** Armazena os usuários que podem fazer login no painel de controle. Com a alteração, cada administrador agora pertence obrigatoriamente a uma organização.

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`id`	`SERIAL PRIMARY KEY`	Identificador numérico único do administrador.	`1`
`organizacao_id`	`INTEGER REFERENCES Organizacoes(id)`	**Chave Estrangeira** que liga o agricultor à organização que o gerencia.	
`nome`	`VARCHAR(255)`	Nome completo do administrador.	`Artur Martins`
`email`	`VARCHAR(255) UNIQUE`	E-mail utilizado para o login no painel.	`artur.martins@email.com`
`senha_hash`	`VARCHAR(255)`	Senha criptografada (nunca salvar a senha em texto puro).	`$2b$10$....`
`cargo`	`VARCHAR(50)`	Papel do usuário dentro da plataforma.	`Gestor`

Relacionamentos:

- Esta tabela agora **depende diretamente** da tabela `Organizacoes`.
- A dependência é criada pela coluna `organizacao_id`, que é uma **Chave Estrangeira (Foreign Key)**.
- Este campo `organizacao_id` aponta para o `id` da tabela `Organizacoes`, criando a parte "muitos" de um relacionamento **muitos-para-um (N-1)**. Ou seja, **muitos** administradores podem pertencer à **mesma** organização.

Tabela : `Agricultores`

- **Propósito:** Tabela central que armazena os dados de cada agricultor cadastrado via chatbot.

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`id`	`SERIAL PRIMARY KEY`	Identificador numérico único do agricultor.	`101`
`organizacao_id`	`INTEGER REFERENCES Organizacoes(id)`	Chave Estrangeira que liga o agricultor à organização	`157`
`nome`	`VARCHAR(255)`	Nome completo do agricultor.	`Juan Pablo`
`whatsapp_id`	`VARCHAR(50) UNIQUE`	Número do WhatsApp com DDI (ex: 55739...).	`5573999998888`
`latitude`	`DECIMAL(10, 8)`	Latitude geográfica precisa enviada pelo WhatsApp.	`-13.85861111`
`longitude`	`DECIMAL(11, 8)`	Longitude geográfica precisa enviada pelo WhatsApp.	`-40.08583333`
`cidade`	`VARCHAR(100)`	Cidade obtida via Reverse Geocoding.	`Jequié`
`estado`	`VARCHAR(2)`	Sigla do estado (UF).	`BA`
`data_cadastro`	`TIMESTAMPTZ`	Data e hora do primeiro contato.	`2025-07-05 10:30:00-03`
`ultima_atividade`	`TIMESTAMPTZ`	Data e hora da última mensagem trocada.	`2025-07-05 11:00:00-03`

Relacionamentos: Tabela-mãe. As tabelas `Safras`, `Produtos_Estoque` e `Interacoes` dependem dela.

Tabela : `Safras`

- **Propósito:** Armazena o histórico de produção do agricultor, crucial para a simulação de safra e análise de crédito.

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`id`	`SERIAL PRIMARY KEY`	Identificador único do registro de safra.	`501`
`agricultor_id`	`INTEGER REFERENCES Agricultores(id)`	Chave estrangeira que conecta à tabela `Agricultores`.	`101`
`cultura`	`VARCHAR(100)`	O tipo de produto cultivado.	`Café Arábica`
`ano_safra`	`VARCHAR(10)`	Período da safra.	`2024/2025`
`area_plantada_ha`	`DECIMAL(10, 2)`	Área plantada em hectares.	`15.50`
`produtividade`	`DECIMAL(10, 2)`	Quantidade colhida (ex: em sacas, toneladas).	`80.00`
`unidade_medida`	`VARCHAR(20)`	Unidade da produtividade.	`Sacas`

Relacionamentos: Depende de `Agricultores` (Um agricultor pode ter muitas safras).

Tabela : `Interacoes`

- **Propósito:** Log bruto de cada mensagem trocada para fins de auditoria e análise qualitativa.

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`id`	`SERIAL PRIMARY KEY`	Identificador único da interação.	`1501`
`agricultor_id`	`INTEGER REFERENCES Agricultores(id)`	Chave estrangeira que conecta ao agricultor.	`101`
`mensagem_usuario`	`TEXT`	Texto exato enviado pelo usuário.	`Qual o preço da soja hoje?`
`resposta_chatbot`	`TEXT`	Texto exato respondido pelo bot.	`O preço da soja...`
`timestamp`	`TIMESTAMPTZ`	Data e hora da interação.	`2025-07-05 11:30:00-03`
`entidades`	`JSONB`	Um objeto com as entidades extraídas pela IA.	`{"produto": "soja", "localizacao": "Jequié"}`

Relacionamentos: Depende de `Agricultores`. É a tabela-mãe de `Interacoes_Intencoes`.

Tabela : `Intencoes`

- **Propósito:** Dicionário de funcionalidades do chatbot, baseado no menu principal, para categorizar as conversas.

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`id`	`SERIAL PRIMARY KEY`	Identificador único da intenção.	`1`
`nome_intencao`	`VARCHAR(100) UNIQUE`	Código da funcionalidade.	`previsao_clima`
`descricao`	`VARCHAR(255)`	Descrição amigável da funcionalidade.	`Previsão Climática`

Relacionamentos: Tabela independente que serve como um "dicionário" para `Interacoes_Intencoes`.

Tabela : `Interacoes_Intencoes`

- **Propósito:** Tabela de ligação que conecta uma única interação a uma ou várias intenções identificadas pela IA.

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`interacao_id`	`INTEGER REFERENCES Interacoes(id)`	Chave estrangeira que aponta para a conversa.	`1501`
`intencao_id`	`INTEGER REFERENCES Intencoes(id)`	Chave estrangeira que aponta para a intenção.	`6` (analise_mercado)

Relacionamentos: Depende de `Interacoes` e `Intencoes`, criando a relação Muitos-para-Muitos (M-N).

Tabela : `Produtos_Estoque`

- **Propósito:** Catálogo de insumos e produtos colhidos para a funcionalidade de "Controle de Estoque".

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`id`	`SERIAL PRIMARY KEY`	Identificador único do produto no estoque.	`701`
`agricultor_id`	`INTEGER REFERENCES Agricultores(id)`	Chave estrangeira que conecta ao dono do estoque.	`101`
`nome_produto`	`VARCHAR(255)`	Nome do item.	`Fertilizante NPK 10-10-10`
`tipo_produto`	`VARCHAR(50)`	Categoria do item: "Insumo" ou "Produção".	`Insumo`
`unidade_medida`	`VARCHAR(20)`	Unidade de medida do item.	`kg`
`saldo_atual`	`DECIMAL(10, 2)`	Quantidade disponível no estoque.	`250.00`

Relacionamentos: Depende de `Agricultores`. É a tabela-mãe de `Movimentacoes_Estoque`.

Tabela : `Movimentacoes_Estoque`

- **Propósito:** Registra todas as entradas e saídas de cada produto, formando um histórico completo para o "Controle de Estoque".

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`id`	`SERIAL PRIMARY KEY`	Identificador único da transação.	`901`
`produto_id`	`INTEGER REFERENCES Produtos_Estoque(id)`	Chave estrangeira que conecta ao produto.	`701`
`tipo_movimentacao`	`VARCHAR(10)`	"Entrada" ou "Saída".	`Saída`
`quantidade`	`DECIMAL(10, 2)`	Quantidade que entrou ou saiu.	`50.00`
`data_movimentacao`	`TIMESTAMPTZ`	Data e hora do registro.	`2025-07-05 11:15:00-03`
`observacao`	`TEXT`	Nota opcional sobre a movimentação.	`Aplicação na lavoura de café`

Relacionamentos: Depende de `Produtos_Estoque` (Um produto pode ter muitas movimentações).

Tabela: `conversation_contexts`

- **Propósito:** Tabela para armazenar o **estado temporário** da conversa, como no código Python atual. Ajuda o bot a lembrar em qual etapa de um fluxo de perguntas ele está. Os dados daqui devem ser movidos para as tabelas permanentes ao final do fluxo.

Nome da Coluna	Tipo de Dado (PostgreSQL)	Descrição	Exemplo
`whatsapp_id`	`VARCHAR(50) UNIQUE`	Número do WhatsApp com DDI (ex: 55739...).	`5573999998888`
`context`	`JSONB`	Objeto JSON com o estado da conversa.	
`last_updated`	`TIMESTAMPTZ`	Data da última atualização.	

4. Como o Chatbot Deve Interagir com o Banco de Dados

A equipe do chatbot deve usar esta documentação para direcionar o desenvolvimento. O fluxo principal é:

1. **Usar a tabela `conversation_contexts`** para gerenciar o estado da conversa passo a passo (como o código atual já faz).
2. **Ao final de um fluxo** (ex: cadastro de agricultor, registro de estoque), pegar os dados consolidados do `contexto` e chamar uma função para salvá-los de forma estruturada nas **tabelas permanentes** (`Agricultores`, `Safras`, `Produtos_Estoque`, etc.).
3. **A cada interação**, além de salvar o estado, o bot deve analisar a mensagem, identificar intenções e entidades, e salvar essa análise nas tabelas `Interacoes`, `Interacoes_Intencoes` e no campo `entidades` da tabela `Interacoes`.

5. Código SQL (PostgreSQL)

```
-- Tabela de Organizações (Inquilinos)
CREATE TABLE IF NOT EXISTS Organizacoes (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(255) UNIQUE NOT NULL,
    cnpj VARCHAR(18) UNIQUE,
    data_criacao TIMESTAMPTZ DEFAULT NOW()
);

-- Tabela de Administradores (Usuários do Painel)
CREATE TABLE IF NOT EXISTS Administradores (
    id SERIAL PRIMARY KEY,
    organizacao_id INTEGER NOT NULL REFERENCES Organizacoes(id),
    nome VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    senha_hash VARCHAR(255) NOT NULL,
    cargo VARCHAR(50)
);

-- Tabela de Agricultores (Usuários do Chatbot)
CREATE TABLE IF NOT EXISTS Agricultores (
    id SERIAL PRIMARY KEY,
    organizacao_id INTEGER NOT NULL REFERENCES Organizacoes(id),
    nome VARCHAR(255) NOT NULL,
    whatsapp_id VARCHAR(50) UNIQUE NOT NULL,
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    cidade VARCHAR(100),
    estado VARCHAR(2),
    data_cadastro TIMESTAMPTZ DEFAULT NOW(),
    ultima_atividade TIMESTAMPTZ
);

-- Tabela de Safras (Histórico de Produção)
CREATE TABLE IF NOT EXISTS Safras (
    id SERIAL PRIMARY KEY,
    agricultor_id INTEGER NOT NULL REFERENCES Agricultores(id),
    cultura VARCHAR(100) NOT NULL,
    ano_safra VARCHAR(10),
    area_plantada_ha DECIMAL(10, 2),
    produtividade DECIMAL(10, 2),
    unidade_medida VARCHAR(20)
);

-- Tabela de Produtos em Estoque (Insumos ou Produção)
```

```
CREATE TABLE IF NOT EXISTS Produtos_Estoque (
    id SERIAL PRIMARY KEY,
    agricultor_id INTEGER NOT NULL REFERENCES Agricultores(id),
    nome_produto VARCHAR(255) NOT NULL,
    tipo_produto VARCHAR(50) NOT NULL,
    unidade_medida VARCHAR(20) NOT NULL,
    saldo_atual DECIMAL(10, 2) NOT NULL
);

-- Tabela de Movimentações de Estoque (Entradas e Saídas)
CREATE TABLE IF NOT EXISTS Movimentacoes_Estoque (
    id SERIAL PRIMARY KEY,
    produto_id INTEGER NOT NULL REFERENCES Produtos_Estoque(id),
    tipo_movimentacao VARCHAR(10) NOT NULL,
    quantidade DECIMAL(10, 2) NOT NULL,
    data_movimentacao TIMESTAMPTZ DEFAULT NOW(),
    observacao TEXT
);

-- Tabela de Interações (Log de Conversas)
CREATE TABLE IF NOT EXISTS Interacoes (
    id SERIAL PRIMARY KEY,
    agricultor_id INTEGER NOT NULL REFERENCES Agricultores(id),
    mensagem_usuario TEXT,
    resposta_chatbot TEXT,
    entidades JSONB,
    timestamp TIMESTAMPTZ DEFAULT NOW()
);

-- Tabela Dicionário de Intenções
CREATE TABLE IF NOT EXISTS Intencoes (
    id SERIAL PRIMARY KEY,
    nome_intencao VARCHAR(100) UNIQUE NOT NULL,
    descricao VARCHAR(255)
);

-- Tabela de Ligação (Muitos-para-Muitos entre Interações e Intenções)
CREATE TABLE IF NOT EXISTS Interacoes_Intencoes (
    interacao_id INTEGER NOT NULL REFERENCES Interacoes(id) ON DELETE CASCADE,
    intencao_id INTEGER NOT NULL REFERENCES Intencoes(id) ON DELETE CASCADE,
    PRIMARY KEY (interacao_id, intencao_id)
);

-- Tabela para Estado Temporário da Conversa
CREATE TABLE IF NOT EXISTS conversation_contexts (
    whatsapp_id VARCHAR(50) PRIMARY KEY,
    context JSONB,
    last_updated TIMESTAMPTZ DEFAULT NOW()
```

) ;