# DESCRIÇÃO

A tabela “**parametros\_processo**” armazena todos os parâmetros de processo que podem ser utilizados nos cálculos de TPE (Tempo Padrão de Engenharia).

Esses parâmetros são campos dinâmicos que o usuário preenche ao realizar um cálculo (ex.: perímetro, área de superfície, espessura, quantidade de furos, etc.).

* Define quais parâmetros estão disponíveis para o cálculo
* Armazena tipo de dado, unidade de medida e validações
* Permite criar campos customizados sem alterar código
* Controla ordem de exibição e obrigatoriedade

# OBJETIVO

* Centralizar a definição de parâmetros de processo em uma única tabela configurável;
* Padronizar tipos de dados e unidades de medida;
* Permitir configuração sem necessidade de alteração de código (campos dinâmicos);
* Validar entradas do usuário com base em tipo e obrigatoriedade;
* Integrar com fórmulas de cálculo que referenciam esses parâmetros.

# LÓGICA TÉCNICA DE AQUISIÇÃO DE DADOS

**Fontes**

* Interface administrativa do widget (criação/edição de parâmetros);
* Chamadas de API diretamente (criação/edição de parâmetros).

**Script para criar tabela**

CREATE TABLE parametros\_processo (

id INT IDENTITY(1,1) PRIMARY KEY,

nome NVARCHAR(100) NOT NULL UNIQUE,

tipo NVARCHAR(50) NOT NULL,

unidade NVARCHAR(20),

descricao NVARCHAR(500),

opcoes NVARCHAR(MAX), --json

valor\_padrao NVARCHAR(200),

obrigatorio BIT DEFAULT 0,

imagem\_base64 NVARCHAR(MAX),

data\_criacao DATETIME2 DEFAULT GETDATE()

);

-- Índices para performance

CREATE INDEX idx\_parametros\_nome ON parametros\_processo(nome);

CREATE INDEX idx\_parametros\_tipo ON parametros\_processo(tipo);

CREATE INDEX idx\_parametros\_obrigatorio ON parametros\_processo(obrigatorio);

-- Constraint para validar JSON no campo opcoes (quando aplicável)

ALTER TABLE parametros\_processo

ADD CONSTRAINT chk\_parametros\_opcoes\_json

CHECK (opcoes IS NULL OR ISJSON(opcoes) = 1);

# FILTROS

Recomendações de filtros no consumo (API/relatórios/SQL):

* Por nome
* Por tipo
* Por descrição
* Busca textual em nome e descrição

# RECORRÊNCIA DE ATUALIZAÇÃO

* **Widget/API** → atualização **em tempo real** (transacional).

# CAMPO CHAVE PARA ATUALIZAÇÃO

**Chave primária**: `id` (INT IDENTITY)

**Chave natural (UNIQUE):** `nome` - garante que não existam parâmetros com nomes duplicados

**Estratégia de operações:**

**CREATE (POST)**

- Valida se `nome` já existe (UNIQUE constraint)

- Insere novo registro com ID auto-gerado

**UPDATE (PUT)**

- Identificação por `id` enviado na URL (`/orbis/parametros/:id`)

- Atualiza campos enviados no body

- Valida unicidade de `nome` se este campo for alterado

**DELETE (DELETE)**

- Quando usuário **excluir** via widget: `DELETE FROM parametros\_processo WHERE id = @id`

- Remove permanentemente o registro do banco

- Usado quando o parâmetro foi criado por engano ou não é mais necessário

# API – DADOS DE ENTRADA

**POST "/orbis/parametros" - Criar novo parâmetro (tipo numérico)**

* **Descrição**: Cria um parâmetro de processo no sistema.
* **Body (JSON):**

{

"nome": "quantidade\_dobras",

"tipo": "numero\_decimal",

"unidade": "unidade",

"descricao": "Quantidade de dobras na peça",

"opcoes": null,

"valor\_padrao": "0",

"obrigatorio": 1,

"imagem\_base64": "..."

}

* **Retorno 201:**

{

"success": true,

"data": {

"id": 15,

"nome": "quantidade\_dobras",

"tipo": "numero\_decimal",

"unidade": "unidade",

"descricao": "Quantidade de dobras na peça",

"opcoes": null,

"valor\_padrao": "0",

"obrigatorio": 1,

"imagem\_base64": "..."

"data\_criacao": "2025-10-09T14:30:00Z"

}

}

**POST "/orbis/parametros" - Criar novo parâmetro (tipo select com opções)**

* **Descrição**: Cria um parâmetro de processo no sistema.
* **Body (JSON):**

{

"nome": "maquina\_dobradeira",

"tipo": "select",

"unidade": null,

"descricao": "Tipo de máquina dobradeira",

"opcoes": [

"DOBRADEIRA PARA CHAPA FINA AÇO(< 3mm)",

"DOBRADEIRA PARA CHAPAS DE ALUMÍNIO",

"DOBRADEIRA AUTOMATIZADA",

"DOBRADEIRA P/ CHAPA GROSSA DE AÇO(>=3mm)",

"DOBRADEIRA PARA CHAPAS PAVILHAO POLTRO"

],

"valor\_padrao": "DOBRADEIRA AUTOMATIZADA",

"obrigatorio": 1,

"imagem\_base64": "..."

}

* **Retorno 201:**

{

"success": true,

"data": {

"id": 16,

"nome": "maquina\_dobradeira",

"tipo": "select",

"unidade": null,

"descricao": "Tipo de máquina dobradeira",

"opcoes": [

"DOBRADEIRA PARA CHAPA FINA AÇO(< 3mm)",

"DOBRADEIRA PARA CHAPAS DE ALUMÍNIO",

"DOBRADEIRA AUTOMATIZADA",

"DOBRADEIRA P/ CHAPA GROSSA DE AÇO(>=3mm)",

"DOBRADEIRA PARA CHAPAS PAVILHAO POLTRO"

],

"valor\_padrao": "DOBRADEIRA AUTOMATIZADA",

"obrigatorio": 1,

"imagem\_base64": "...",

"data\_criacao": "2025-10-09T14:30:00Z"

}

}

**PUT "/orbis/parametros/:id" - Atualizar parâmetro existente**

* **Descrição:** Atualiza dados de um parâmetro existente identificado pelo ID
* **URL Parameter:** "id" (obrigatório)
* **Body (JSON):**

{

"nome": "quantidade\_dobras",

"tipo": "numero\_decimal",

"unidade": "unidade",

"descricao": "Quantidade de dobras na peça (atualizado)",

"opcoes": null,

"valor\_padrao": "0",

"obrigatorio": 1,

"imagem\_base64": "..."

}

* **Retorno 200:**

{

"success": true,

"data": {

"id": 15,

"nome": "quantidade\_dobras",

"tipo": "numero\_decimal",

"unidade": "unidade",

"descricao": "Quantidade de dobras na peça (atualizado)",

"opcoes": null,

"valor\_padrao": "0",

"obrigatorio": 1,

"imagem\_base64": "...",

"data\_criacao": "2025-10-07T14:30:00Z"

}

}

* **Retorno 404**: "{"success": false, "error": "Parâmetro não encontrado"}"
* **Retorno 409**: "{"success": false, "error": "Já existe um parâmetro com este nome"}"

**DELETE "/orbis/parametros/:id" - Excluir parâmetro permanentemente**

* **Descrição:** DELETE físico - remove registro do banco definitivamente
* **URL Parameter:** "id" (obrigatório**)**
* **Retorno 200:**

{

"success": true,

"message": "Parâmetro excluído permanentemente"

}

* **Retorno 404:** “{“success": false, "error": "Parâmetro não encontrado”}"

# API – DADOS DE SAÍDA

**GET “/orbis/parametros” – Listar parâmetros**

* **Descrição:** Retorna lista de todos os parâmetros (com filtros opcionais de [nome, tipo, ativo, descrição])
* **Parâmetros:** filtros opcionais
* **Retorno:** coleção de parametros.

**GET “/orbis/parâmetros/:id” – Buscar parâmetro por ID**

* **Descrição:** Retorna um parâmetro específico pelo ID
* **Retorno 200**: Objeto único do parâmetro
* **Retorno 404**: “{"success": false, "error": "Parâmetro não encontrado"}”

**GET “/orbis/parametros/nome/:nome” – Buscar parâmetro por nome**

* **Descrição:** Retorna um parâmetro específico pelo nome
* **Retorno 200**: Objeto único do parâmetro
* **Retorno 404**: “{"success": false, "error": "Parâmetro não encontrado"}”

# CATEGORIZAÇÃO DO DHIVE

# ÁREAS IMPACTADAS PELA INFORMAÇÃO

Engenharia de Processos

# VALOR AGREGADO PARA A COMPANHIA

* Padronização: Unifica definição de parâmetros de processo em toda a empresa
* Flexibilidade: Permite criar parâmetros sem alteração de código
* Rastreabilidade: Histórico completo de alterações via auditoria
* Integração: Base para cálculos automatizados de TPE
* Qualidade: Validações impedem erros de entrada de dados
* Migração para 3DEXPERIENCE: Remove dependência de planilhas Excel e sistemas legados

# INTEGRAÇÃO COM SISTEMA DE AUDITORIA

Todas as operações de CREATE, UPDATE, DELETE, ACTIVATE e DEACTIVATE nesta tabela DEVEM registrar logs na tabela "logs\_auditoria".

1. RECEBER user\_id no Body

Todas as requisições POST, PUT, PATCH e DELETE devem receber o campo user\_id:

Body exemplo:

{

"nome": "...",

"descricao": "...",

"user\_id": "eduardo.silva" ← OBRIGATÓRIO

}

2. CHAMAR Função de Log

Após cada operação bem-sucedida, chamar:

registrarLog(

tabela: 'nome\_da\_tabela',

registro\_id: id\_do\_registro,

operacao: 'CREATE' | 'UPDATE' | 'DELETE' | 'ACTIVATE' | 'DEACTIVATE',

valores\_anteriores: objeto\_com\_valores\_antigos (NULL para CREATE),

valores\_novos: objeto\_com\_valores\_novos (NULL para DELETE),

user\_id: user\_id\_recebido\_no\_body

);

3. EXEMPLOS POR OPERAÇÃO

**CREATE** (POST):

* Capturar user\_id do body
* Executar INSERT na tabela
* Chamar: registrarLog('tabela', novo\_id, 'CREATE', null, dados\_inseridos, user\_id)

**UPDATE** (PUT):

* Capturar user\_id do body
* Buscar valores anteriores: SELECT \* FROM tabela WHERE id = @id
* Executar UPDATE na tabela
* Buscar valores novos: SELECT \* FROM tabela WHERE id = @id
* Chamar: registrarLog('tabela', id, 'UPDATE', valores\_anteriores, valores\_novos, user\_id)

**DELETE** (DELETE):

* Capturar user\_id do body
* Buscar valores anteriores: SELECT \* FROM tabela WHERE id = @id
* Executar DELETE na tabela
* Chamar: registrarLog('tabela', id, 'DELETE', valores\_anteriores, null, user\_id)

**ACTIVATE** (PATCH /:id/ativar):

* Capturar user\_id do body
* Buscar valor anterior: SELECT ativo FROM tabela WHERE id = @id
* Executar: UPDATE tabela SET ativo = 1 WHERE id = @id
* Chamar: registrarLog('tabela', id, 'ACTIVATE', {ativo: 0}, {ativo: 1}, user\_id)

**DEACTIVATE (PATCH /:id/desativar):**

* Capturar user\_id do body
* Buscar valor anterior: SELECT ativo FROM tabela WHERE id = @id
* Executar: UPDATE tabela SET ativo = 0 WHERE id = @id
* Chamar: registrarLog('tabela', id, 'DEACTIVATE', {ativo: 1}, {ativo: 0}, user\_id)