# DESCRIÇÃO

A tabela "rotas" armazena as rotas de fabricação (sequência de centros de trabalho) utilizadas para calcular o TPE (Tempo Padrão de Engenharia).

Uma rota define o caminho que uma peça percorre durante a fabricação, referenciandoas etapas de produção (da tabela "etapas") que compõem o processo.

Funções:

* Define sequências de fabricação (ex.: Corte → Dobra → Solda → Pintura)
* Referencia etapas de produção através de IDs
* Categoriza rotas por centro de produção
* Permite versionamento de rotas
* Parâmetros necessários são resolvidos dinamicamente a partir das etapas

# OBJETIVO

* Definir sequências de fabricação padronizadas
* Referenciar etapas de produção (tabela "etapas") na ordem correta
* Categorizar rotas por tipo de produto/processo e centro de produção
* Integrar com sistema de cálculo de TPE do widget ORBIS
* Parâmetros necessários são obtidos através das etapas configuradas

# 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 rotas (

id INT IDENTITY(1,1) PRIMARY KEY,

nome NVARCHAR(500) NOT NULL UNIQUE,

descricao NVARCHAR(MAX),

sequencia\_centros NVARCHAR(500) NOT NULL,

ativa BIT DEFAULT 1,

data\_criacao DATETIME2 DEFAULT GETDATE(),

data\_modificacao DATETIME2 DEFAULT GETDATE(),

centro\_prod NVARCHAR(40),

etapas NVARCHAR(MAX) NOT NULL

);

-- Índices para performance

CREATE INDEX idx\_rotas\_nome ON rotas(nome);

CREATE INDEX idx\_rotas\_ativa ON rotas(ativa);

CREATE INDEX idx\_rotas\_centro\_prod ON rotas(centro\_prod);

-- Constraint para validar JSON no campo etapas

ALTER TABLE rotas

ADD CONSTRAINT chk\_rotas\_etapas\_json

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

# FILTROS

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

* **Por nome**
* **Por descrição**
* **Por ativa (ativo/inativo)**
* **Por centro de produção**

# 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 rotas 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/*rotas*/:id`)

- Atualiza campos enviados no body

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

**DELETE (DELETE)**

- Quando usuário **excluir** via widget: `DELETE FROM rotas WHERE id = @id`

- Remove permanentemente o registro do banco

- Usado quando a rota foi criado por engano ou não é mais necessária

**DESATIVAR (PATCH)**

- Quando usuário desativar via widget: `UPDATE rotas SET ativo = 0 WHERE id = @id`

- Mantém o registro no banco com `ativo = 0`

- Usado quando a rota não deve mais aparecer, mas precisa ser mantida para histórico/auditoria

# API – DADOS DE ENTRADA

**POST "/orbis/rotas" - Criar nova rota**

* **Descrição**: Cria uma rota de fabricação no sistema.
* **Body (JSON):**

{

"nome": "CORTE LASER -> DOBRA -> MONTAGEM",

"descricao": "Rota para peças cortadas em laser com dobra e montagem",

"sequencia\_centros": "MA303,DA101,BC103",

"centro\_prod": "MP10",

"etapas": [1, 3, 4],

"ativa": true

}

**Retorno 201:**

{

"success": true,

"data": {

"id": 4,

"nome": "CORTE LASER -> DOBRA -> MONTAGEM",

"descricao": "Rota para peças cortadas em laser com dobra e montagem",

"sequencia\_centros": "MA303,DA101,BC103",

"centro\_prod": "MP10",

"etapas": [1, 3, 4],

"ativa": true,

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

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

}

}

**PUT "/orbis/rotas/:id" - Atualizar rota existente**

* **Descrição:** Atualiza dados de uma rota existente identificada pelo ID
* **URL Parameter:** "id" (obrigatório)
* **Body (JSON):**

{

"nome": "CORTE LASER -> DOBRA -> MONTAGEM (Atualizada)",

"descricao": "Rota atualizada com novas etapas",

"sequencia\_centros": "MA303,MA301,DE103,DA101,BC103",

"etapas": [1, 2, 3, 4]

}

* **Retorno 200:**

{

"success": true,

"data": {

"id": 4,

"nome": "CORTE LASER -> DOBRA -> MONTAGEM (Atualizada)",

"descricao": "Rota atualizada com novas etapas",

"sequencia\_centros": "MA303,MA301,DE103,DA101,BC103",

"centro\_prod": "MP10",

"etapas": [1, 2, 3, 4],

"ativa": true,

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

"data\_modificacao": "2025-10-10T10:30:00Z"

}

}

* **Retorno 404**: "{"success": false, "error": "Rota não encontrada"}"
* **Retorno** **409**: {"success": false, "error": "Já existe uma rota com este nome"}

**PATCH "/orbis/rotas/:id/desativar" - Desativar rota**

* **Descrição:** Soft delete - marca "ativo = 0" (mantém registro no banco)
* **URL Parameter:** "id" (obrigatório)
* **Retorno 200:**

{

"success": true,

"message": "Rota desativada com sucesso",

"data": {

"id": 25,

"ativo": false

}

}

**PATCH "/orbis/rotas/:id/ativar" - Reativar rota**

* **Descrição:** Marca "ativo = 1" (reativa parâmetro desativado)
* **URL Parameter:** "id" (obrigatório)
* **Retorno 200:**

{

"success": true,

"message": "Rota ativada com sucesso",

"data": {

"id": 25,

"ativo": true

}

}

**DELETE "/orbis/rotas/:id" - Excluir rota permanentemente**

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

{

"success": true,

"message": "Rota excluída permanentemente"

}

* **Retorno 404:** “{“success": false, "error": "Rota não encontrada”}"

# API – DADOS DE SAÍDA

**GET “/orbis/rotas” – Listar rotas**

* **Descrição:** Retorna lista de todas as rotas (com filtros opcionais de [codigo, nome, ativo, centro\_prod, descrição])
* **Parâmetros:** filtros opcionais
* **Retorno:** coleção de rotas.

**GET “/orbis/rotas/:id” – Buscar rota por ID**

* **Descrição:** Retorna uma rota específica pelo ID
* **Retorno 200**: Objeto único da rota
* **Retorno 404**: “{"success": false, "error": "Rota não encontrada"}”

**GET “/orbis/rotas/descricao/:descricao” – Buscar rota por descricao**

* **Descrição:** Retorna uma rota específica que contenha a descricao
* **Retorno 200**: Coleção de rotas que tenham a descrição
* **Retorno 404**: “{"success": false, "error": "Nenhuma rota encontrada"}”

# CATEGORIZAÇÃO DO DHIVE

# ÁREAS IMPACTADAS PELA INFORMAÇÃO

Engenharia de Processos

# VALOR AGREGADO PARA A COMPANHIA

* **Padronização**: Unifica rotas de fabricação em toda a empresa
* **Rastreabilidade**: Versionamento mantém histórico completo de alterações
* **Flexibilidade**: Permite criar e modificar rotas sem alteração de código
* **Integração**: Base para cálculos automatizados de TPE e planejamento
* **Qualidade**: Garante que processos sigam sequências aprovadas
* **Migração para 3DEXPERIENCE**: Remove dependência de planilhas Excel e sistemas legados
* **Otimização**: Facilita análise e otimização de processos produtivos

# 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)