5 - RecipeProcessor

Iagro CLP Local — RecipeProcessor

Traduz um registro `work_order_recipe_plc_sync` (payload HTTP) em persistência local: um comando pai `create-recipe` e N linhas `write` filhas. **Não** abre sessão OPC; execução física é **7 - CommandProcessor**.

Código: `src/services/recipe_processor.py`
Mapeamento: `src/models/recipe_mapper.py` (`RecipeMapper`)
Persistência: `DatabaseManager.create_recipe_command` (`database.py` ~L1569)

📋 Resumo

Etapa	Onde
Validação negócio	`process_recipe`
Mapeamento JSON → tags	`RecipeMapper.map_recipe_to_variables`
Resolução nome → `variable_id`	`get_variable_id_by_name`
Transação DB	`create_recipe_command` (commit único)

🎯 Responsabilidades e limites

Faz:

- Dedup por sync id (`has_recipe_sync_in_progress`).
- Bloqueio se outra receita ativa (`get_active_recipe`).
- Exige `is_connected` True (gate CLP antes de enfileirar).
- Ignora mappings sem variável cadastrada no container.

Não faz:

- PATCH `NA_MAQUINA` / `BLOQUEADO` (CommandProcessor).
- Ordenação de execução OPC (CP usa campo `"order"`).
- Inserir `recipe-finalize` ou resets.

📁 API pública

Método	Assinatura	Retorno
`__init__`	`database_manager`, `api_service`, `container_id`, `smart_calda_id`, `container_variables`, `recipe_mapping_config?`, `branch_id?`	Configura `RecipeMapper`
`process_recipe`	`recipe_sync: dict`, `is_connected: bool`	`True` se `create_recipe_command` retornou id
`update_container_variables`	`list`	Recria mapper com novos nomes

Instanciado em `RecipeMonitorService._update_processor()` sempre que variáveis ou `configurationJson` mudam.

📁 Contratos de dados

Entrada `recipe_sync` (API)

Campo API	Uso interno
`id`	`recipe_id` no JSON do pai
`parcelId` / `parcel_id`	`parcel_id`
`recipeData` / `recipe_data`	Entrada do mapper (dict ou string JSON)
`status`	Esperado `NA_FILA` na captura
`createdAt`	Ordenação no monitor (não aqui)

Saída DB — pai `create-recipe`

```
{
  "recipe_data": { "...": "valores mapeados / brutos" },
```

```
  "parcel_id": "<uuid parcela>",
  "recipe_id": "<uuid work_order_recipe_plc_sync>"
}
```

Coluna pai	Valor
`command_type`	`create-recipe`
`variable_id`	NULL (constraint)
`status`	`pending`
`requested_by`	`recipe_{recipe_id}` implícito no fluxo monitor

Saída DB — filhos `write`

Coluna	Valor
`command_type`	`write`
`recipe_command_id`	id do pai
`order`	do mapping
`value`	jsonb `{ "value": <opc> }` ou estrutura do mapper
`status`	`pending`
`variable_id`	resolvido por nome

Importante: `requested_by` dos filhos na criação segue padrão DB (`recipe` batch) — por isso **não** entram em `get_pending_write_commands` (sem prefixo `reset_after_production_`).

`configurationJson.recipe_mapping`

Dict **caminho no recipe_data → nome da variável** na tabela `variables` do container.

Exemplo conceitual:

```
"SituacaoAtualSmart.Temperatura" → "TEMP_SETPOINT"
"OrdemServico.Produto"           → "PRODUTO_COD"
```

Caminhos inválidos ou variável ausente → entrada descartada (log debug).

🔄 Algoritmo `process_recipe` (passo a passo)

```
1. recipe_id, parcel_id, recipe_data ← recipe_sync (parse JSON se str)
2. SE has_recipe_sync_in_progress(container_id, recipe_id): RETURN Fals
3. SE NOT is_connected: RETURN False
4. active ← get_active_recipe(smart_calda_id, branch_id)
   SE active E active.id != recipe_id: RETURN False
5. mappings ← map_recipe_to_variables(recipe_data)
6. write_commands ← []
   PARA cada mapping:
     vid ← get_variable_id_by_name(container_id, variable_name)
     SE vid: append { variable_id, value, order }
7. SE write_commands vazio: RETURN False
8. cmd_id ← create_recipe_command(container_id, value_json, write_comma
9. RETURN cmd_id is not None
```

`create_recipe_command` **(DatabaseManager)**

Transação atômica:

1. Valida ≥1 filho com `variable_id`.
2. Revalida `has_recipe_sync_in_progress` (race).
3. INSERT pai `pending`.
4. INSERT filhos ordenados.
5. Rollback se zero filhos válidos.

🔄 Sequência após enfileiramento

```
RecipeMonitor captura NA_FILA
  → process_recipe True
  → CommandProcessor pega create-recipe em get_pending
  → _execute_recipe_command (loop filhos)
  → API NA_MAQUINA; pai executing
  → (produção + supervisório + resets)
  → check_executing_recipe_parents → pai completed
```

🎯 Integração

Origem	Comportamento
`RecipeMonitorService`	Chama `process_recipe` na primeira receita elegível da lista
`SequentialCoordinator._task_monitor_recipes`	`limit=1` na API; mesmo processor
`CLPContainer`	Vars/config propagam via `update_container_variables` no monitor

⚙️ Configuração

- `recipe_mapping`: subtree de `SmartCalda.configurationJson`.
- `branch_id`: header em `get_active_recipe`.
- **Variáveis:** devem existir em `variables` para o `container_id` (descoberta prévia).

⚠️ Matriz de resultados `process_recipe`

Condição	Retorno	Efeito colateral
Sync já com create aberto	False	Log warning DB
CLP desconectado	False	—
Outra receita ativa	False	—
Nenhum mapping / var	False	—
`create_recipe_command` None	False	Rollback já feito no DB
Exceção	False	Log stack

🔗 Referências

- **6 - RecipeMonitorService** (gates antes de chamar)
- **7 - CommandProcessor** (execução + ERP)
- **4 - ApiService** (`get_active_recipe`)
- **2 - Services — Visão geral** (diagrama ciclo)