1 - IAgro CLP Local — Visão geral

Iagro CLP Local

Cliente desktop **CustomTkinter** que opera SmartCaldas no chão de fábrica: sessão OPC UA dupla (monitor + realtime), fila transacional PostgreSQL (`write_commands`) e sincronização de status de receita com o backend NestJS IAgro.

📚 Mapa de páginas

#	Título	Conteúdo aprofundado
2	Services — Visão geral	Arquitetura, ciclo ERP ↔ fila ↔ CLP, modos sequencial/paralelo
3	AuthService	JWT, refresh ~90s, tenant extraction
4	ApiService	REST completo, headers, armadilhas `parcel_id`
5	RecipeProcessor	Mapper, transação `create_recipe_command`
6	RecipeMonitorService	Gates SQL/API, polling adaptativo
7	CommandProcessor	Fila, máquina de estados, finalize, sanitizer
8	ProductionMonitorService	Gatilhos OPC, resets, end/cancel desativados
9	CLPContainer	Connect lifecycle, `_sync_command_processor_thread`

10	SequentialCoordinator	read_worker + serial worker
11	UI — IAgroCLPAppCTk	Login, containers, loading persist
12	API — contrato OpenAPI	YAML, operationIds

🔄 Fluxo end-to-end (referência rápida)

```
[Operador] Login (3) → ApiService (4) + SessionManager
  → Seleciona branch / SmartCalda
  → CLPContainer.connect (9)

Modo sequencial (10) — padrão com monitor ON:
  read_worker: OPC → buffer → on_data_updated → PM (8)
  serial_worker: health → DB flush → CP.process_commands (7) → RM captu

Captura:
  RM gates → API NA_FILA → RP (5) → INSERT create-recipe + writes

Execução:
  CP _execute_recipe_command → OPC → ERP NA_MAQUINA
  pai create-recipe → executing

Produção:
  PM start/pause/fail → ERP EM_PRODUCAO / PAUSADO / …

Encerramento supervisório:
  Backend confirm-action → ERP terminal
  → recipe-finalize (DB CLP)
  → CP finalize → reset_after_production_* → pai completed
  → RM captura próxima NA_FILA
```

🗄️ Persistência local (PostgreSQL)

Tabela	Papel
`containers`	Instância SmartCalda + credenciais OPC
`variables`	Tags descobertas / cadastradas
`write_commands`	Fila de trabalho (tipos em **7**)

Schema: `config/schema.sql` · Migrações: `config/migrations/`.

🏗️ Modos de execução

Modo	Leitura OPC	Comandos	Receitas `NA_FILA`
Sequencial + monitor ON	`read_worker` (10)	`process_commands` no serial loop	`_task_monitor_recipes`
Paralelo	`VariableMonitorThread`	`CommandProcessor.run` thread	`RecipeMonitorService.run` thread

`CLPContainer._sync_command_processor_thread` evita CP em thread duplicada quando o coordenador assume o poll.

⚠️ Invariantes do sistema (leitura obrigatória)

1. Writes de receita inicial **não** competem na fila global — só loop interno do create-recipe.
2. `NA_MAQUINA` (ERP) ≠ `create-recipe completed` (DB) — resets pós-produção separados.
3. `has_pending_recipe_commands` ≠ “existe pai executing”.
4. Conclusão/cancel terminal: **supervisório** (finalize), não gatilhos CLP comentados no PM.

🔗 Repositório

Caminho	Conteúdo
`docs/confluence/pages/`	Fonte das páginas (Markdown)
`docs/services/`	Espelho técnico por serviço
`docs/api/backend-openapi.yaml`	Contrato HTTP
`.env.example`	Variáveis de ambiente
`docs/confluence/ESTILO-CONFLUENCE.md`	Títulos numerados, emojis H2