12 - API — contrato OpenAPI

Iagro CLP Local — Contrato OpenAPI (cliente)

Especificação **derivada** do uso real em `AuthService` e `ApiService`. Documenta o que o desktop implementa; o NestJS pode expor rotas adicionais não listadas aqui.

Arquivo: `docs/api/backend-openapi.yaml` (OpenAPI 3.0)

> **Info**
> Ordem de correção em divergência: `src/services/*.py` → YAML → páginas Confluence **3**, **4**, **12**.

📋 Operações implementadas no desktop

Auth

HTTP	operationId	Python
POST `/auth/login`	`login`	`AuthService.login`
POST `/auth/refresh`	`refreshToken`	`AuthService.refresh_token`
GET `/auth/userinfo`	`getUserInfo`	`AuthService.get_user_info`

Negócio

HTTP	operationId	Python
GET `/branches`	`getBranches`	`ApiService.get_branches`
GET `/smart-caldas`	`getSmartCaldasByBranch`	`ApiService.get_smart_caldas_by_branch`
GET `/smart-caldas/{id}`	`getSmartCaldaById`	`ApiService.get_smart_calda_by_id`

GET `/smart-caldas/{id}/active-view-variables`	`getActiveViewVariables`	`ApiService.get_active_view_variables`
GET `/work-order-recipe-plc-sync`	`getWorkOrderRecipeSync`	`ApiService.get_work_order_recipe_sync`
GET `/work-order-recipe-plc-sync/{id}`	—	`ApiService.get_recipe_sync_by_id`
GET (múltiplos `status`)	`getActiveRecipe`	`ApiService.get_active_recipe`
PATCH `/work-order-recipe-plc-sync/{id}`	`updateParcelStatus`	`ApiService.update_parcel_status`
PATCH `…/process-sequence`	`updateParcelProcessSequence`	`ApiService.update_parcel_process_sequence`

📋 Rotas do ecossistema não chamadas pelo desktop

Rota / ação	Quem executa	Relação CLP
`POST …/recipe-modal/confirm-action`	Backend + frontend web	Escreve variável supervisório; ERP terminal
INSERT `recipe-finalize`	`ClpDatabaseService` (Nest)	PostgreSQL **local** CLP
Poll OPC / `write_commands`	Desktop **7**	Execução física

Documentar estas rotas no OpenAPI do backend; o YAML em `docs/api/` pode incluir stubs comentados para rastreabilidade.

🎯 Segurança (OpenAPI `components.securitySchemes`)

Esquema	Header	Obrigatoriedade
`bearerAuth`	`Authorization: Bearer {jwt}`	Rotas autenticadas
—	`X-Tenant-ID`	Quase todas
—	`X-Branch-ID`	SmartCalda, sync, PATCH
—	`x-supervisor-key`	`active-view-variables` (servidor local UI)

📁 Schemas e query — receita sync

GET lista

Query	Tipo	Exemplo
`smartCaldaId`	uuid	Calda conectada
`status`	string	`NA_FILA`
`limit`	int	50 (RM thread) / 1 (coordenador)
`page`	int	Paginação API

PATCH status (`updateParcelStatus`)

Body mínimo:

```
{ "status": "NA_MAQUINA" }
```

Opcional em cancel supervisório: `productsUsed` (estrutura definida no backend).

ID no path: `work_order_recipe_plc_sync.id` (no Python: parâmetro `parcel_id`).

Status ERP referenciados no cliente

`NA_FILA` , `NA_MAQUINA` , `EM_PRODUCAO` , `PAUSADO` , `PROCESSADA` , `CANCELADO` , `BLOQUEADO` .

📁 Respostas — tolerância do parser

`ApiService` normaliza:

- Lista na raiz
- `{ "data": [...] }`
- `{ "items": [...] }`, `{ "results": [...] }`, `{ "branches": [...] }`

No YAML preferir `oneOf` ou descrição livre — o cliente não valida com jsonschema em runtime.

🔄 Mapeamento ciclo supervisório (documentação cruzada)

```
OpenAPI confirm-action (backend)
  → não listado como chamada Python
  → efeito: recipe-finalize row (PostgreSQL CLP)
  → documentado em 7 - CommandProcessor
```

⚙️ Ferramentas locais

```
npx @redocly/cli preview-docs docs/api/backend-openapi.yaml
```

Import no Swagger Editor para diff com Swagger oficial NestJS.

🔗 Referências

- **3 - AuthService** · **4 - ApiService**
- **7 - CommandProcessor** (`recipe-finalize`)
- **2 - Services — Visão geral**