Integrador SR → SimpliRoute

Plataforma de integração entre banco IW/Oracle e SimpliRoute, composta por:

- **CLI**: scripts para gerar, inspecionar e enviar payloads manualmente ou em lote.
- **Serviço FastAPI**: executa polling periódico nas views Oracle e expõe endpoints para webhooks, automatizando a sincronização dos dados.
- **Persistência de retorno**: cada webhook recebido do SimpliRoute é salvo em arquivo e atualiza a tabela Oracle configurada (`TD_OTIMIZE_ALTSTAT` por padrão), preenchendo as colunas de ação, status e informações detalhadas.

Sumário

1. Arquitetura resumida

1. **Polling Oracle**: o serviço lê as views configuradas (visitas e entregas) usando `oracle_source.py`.
2. **Mapeamento**: cada registro é processado por `mapper.py`, que normaliza endereço, contatos, itens e define o tipo de visita/entrega.
3. **Envio ao SimpliRoute**: CLI ou serviço executam chamadas HTTP para a API SimpliRoute via `client.py`.
4. **Webhook**: SimpliRoute retorna status via `POST /webhook/simpliroute`. O JSON é salvo em `data/work/webhooks/` e o Oracle é atualizado por `oracle_status_sync.py`.
5. **Colunas atualizadas no Oracle**:

- `ACAO`: `A` (aguardando), `E` (entregue) ou `S` (suspensa).
- `STATUS`: valores numéricos distintos por `TPREGISTRO` (visitas ou entregas).
- `INFORMACAO`: JSON completo para auditoria.

2. Pré-requisitos

```
- Python 3.11+ para uso local do CLI e Docker/Docker Compose para execução
containerizada.
- Oracle Instant Client (Windows e Linux) armazenado em `settings/instantclient/`.
- Arquivo `settings/.env` ou `settings/config.yaml` com credenciais Oracle, tokens
SimpliRoute e parâmetros de polling.
- Acesso de **SELECT/UPDATE** na tabela de destino (padrão: `TD_OTIMIZE_ALTSTAT`),
nas colunas configuradas.
```

Estrutura do Instant Client

- `settings/instantclient/windows/instantclient_23_0/...`: executa localmente no Windows.
- `settings/instantclient/linux/*.zip`: utilizados no build do Docker e instalados em `/opt/oracle/instantclient`.
- `ORACLE_INSTANT_CLIENT`: permite apontar para outro diretório específico, se necessário.

3. Configuração inicial

1. **Clonar o repositório** e copiar um `settings/.env` seguro para a máquina.
2. **Instalar dependências (local)**:

   ```
   python -m venv .venv
   .\.venv\Scripts\Activate.ps1
   pip install -r requirements.txt
   ```

3. **Validar a conexão Oracle** ajustando `ORACLE_*` em `settings/.env` ou `settings/config.yaml` e executando `python -m src.cli.send_to_simpliroute preview --limit 1`.
4. **Configurar Docker (opcional)**: `docker compose build` já inclui o Instant Client quando os zips estão no diretório correto.

Estrutura de diretórios relevantes

- `data/input/`: arquivos de entrada opcionais para testes.
- `data/output/`: payloads gerados pelo CLI (`send_to_sr_*.json`).
- `data/work/`: logs do serviço (`service_events.log`) e webhooks recebidos (`data/work/webhooks/*.json`).
- `scripts/`: utilitários para checagem, envio e visualização de status.
- `src/`: código-fonte principal, incluindo integrações, mapeamento, cliente HTTP e sincronização de status.

- `settings/`: arquivos de configuração e dependências do Oracle Instant Client.
- `tests/`: scripts de teste, exemplos e validação de payloads.
- `tests/manual/webhook_sample.http`: requisição pronta para testar o webhook.

4. Variáveis de ambiente principais

Categoria	Variável	Descrição
Oracle	`ORACLE_HOST`, `ORACLE_PORT`, `ORACLE_SERVICE`, `ORACLE_USER`, `ORACLE_PASS`, `ORACLE_SCHEMA`	Configuração da conexão.
	`ORACLE_VIEW_VISITAS`, `ORACLE_VIEW_ENTREGAS`, `ORACLE_VIEWS`	Views lidas pelo polling/CLI.
	`ORACLE_POLL_WHERE`, `ORACLE_POLL_WHERE_VISITAS`, `ORACLE_POLL_WHERE_ENTREGAS`	Filtros padrão aplicados por view.
Serviço → Oracle	`SIMPLIROUTE_TARGET_TABLE` (padrão `TD_OTIMIZE_ALTSTAT`)	Tabela que recebe o retorno.
	`SIMPLIROUTE_TARGET_ACTION_COLUMN`, `SIMPLIROUTE_TARGET_STATUS_COLUMN`, `SIMPLIROUTE_TARGET_INFO_COLUMN`	Colunas atualizadas pelo webhook.
SimpliRoute	`SIMPLIR_ROUTE_TOKEN` (ou `SIMPLIROUTE_TOKEN`)	Token para chamadas REST.
	`SIMPLIR_ROUTE_WEBHOOK_TOKEN`	Token exigido no header `Authorization` dos webhooks.
Serviço	`POLLING_INTERVAL_MINUTES` (padrão 60)	Frequência da execução automática.
	`SIMPLIROUTE_POLL_WHERE`	Filtro explícito usado pelo serviço quando não é passado via CLI.
	`SIMPLIROUTE_POLLING_LIMIT`	Limite de registros por ciclo.
	`WEBHOOK_PORT`	Porta exposta pelo FastAPI (9000/8000 etc.).

> Consulte `settings/config.yaml` para valores padrão e exemplos completos.

5. CLI (`src/cli/send_to_simpliroute.py`)

Subcomandos

- `preview`: gera payloads sem enviar e salva em `data/output/` (use `--no-save` para imprimir em tela).
- `send`: gera payloads e, com `--send`, dispara o endpoint do SimpliRoute.
- `auto`: executa o comando definido em `SIMPLIROUTE_AUTO_COMMAND` (padrão `send --send`), ideal para Docker.

Argumentos frequentes

- `--limit 10`: controla quantos registros buscar em cada view (padrão `ORACLE_FETCH_LIMIT` ou 25).
- `--where "ID_ATENDIMENTO = 40367"`: aplica filtro adicional.
- `--view VWPACIENTES_COMVISITAS` / `--views view1 view2`: restringe as views consultadas.
- `--file caminho.json`: usa um arquivo local em vez do Oracle.
- `--send`: habilita o POST para o SimpliRoute (somente no subcomando `send`).

Exemplos rápidos (execução local)

```
# preview – 5 registros (visitas + entregas das views padrão)
python -m src.cli.send_to_simpliroute preview --limit 5

# preview – 1 visita de enfermagem
python -m src.cli.send_to_simpliroute preview `
        --limit 1 `
        --view VWPACIENTES_COMVISITAS `
        --where "(UPPER(TIPOVISITA) LIKE 'ENFER%' OR UPPER(ESPECIALIDADE) LIKE
'ENFER%')" `
        --no-save

# preview – 1 visita médica
python -m src.cli.send_to_simpliroute preview `
        --limit 1 `
        --view VWPACIENTES_COMVISITAS `
        --where "(UPPER(TIPOVISITA) LIKE 'MED%' OR UPPER(ESPECIALIDADE) LIKE
'MED%')" `
        --no-save

# preview – 1 entrega
python -m src.cli.send_to_simpliroute preview `
        --limit 1 `
        --view VWPACIENTES_ENTREGAS `
        --no-save

# preview – visitas e entregas simultâneas (2 de cada)
python -m src.cli.send_to_simpliroute preview `
        --views VWPACIENTES_COMVISITAS VWPACIENTES_ENTREGAS `
        --limit 4

# send – 5 registros (visitas + entregas)
python -m src.cli.send_to_simpliroute send --limit 5 --send

# send – 1 visita de enfermagem
python -m src.cli.send_to_simpliroute send `
        --limit 1 `
        --view VWPACIENTES_COMVISITAS `
        --where "(UPPER(TIPOVISITA) LIKE 'ENFER%' OR UPPER(ESPECIALIDADE) LIKE
'ENFER%')" `
        --send

# send – 1 visita médica
```

```
python -m src.cli.send_to_simpliroute send `
        --limit 1 `
        --view VWPACIENTES_COMVISITAS `
        --where "(UPPER(TIPOVISITA) LIKE 'MED%' OR UPPER(ESPECIALIDADE) LIKE
'MED%')" `
        --send

# send – 1 entrega
python -m src.cli.send_to_simpliroute send `
        --limit 1 `
        --view VWPACIENTES_ENTREGAS `
        --send

# send – reenviar um ID específico
python -m src.cli.send_to_simpliroute send `
        --view VWPACIENTES_COMVISITAS `
        --where "ID_ATENDIMENTO = 32668" `
        --limit 1 `
        --send

# diagnóstico rápido do SimpliRoute (token/endpoint)
python -m src.cli.send_to_simpliroute diagnose-sr --ping

# diagnóstico do Oracle (listar colunas da view)
python -m src.cli.send_to_simpliroute diagnose-db --limit 3 --view
VWPACIENTES_COMVISITAS
```

Exemplos equivalentes via Docker

> Substitua `docker compose` por `docker compose -f docker-compose.prod.yml` para ambiente de produção.

```
# preview – 5 registros (visitas + entregas)
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute preview --limit 5

# preview – 1 visita de enfermagem
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute preview `
        --limit 1 `
        --view VWPACIENTES_COMVISITAS `
        --where "(UPPER(TIPOVISITA) LIKE 'ENFER%' OR UPPER(ESPECIALIDADE) LIKE
'ENFER%')" `
        --no-save

# preview – 1 visita médica
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute preview `
        --limit 1 `
        --view VWPACIENTES_COMVISITAS `
        --where "(UPPER(TIPOVISITA) LIKE 'MED%' OR UPPER(ESPECIALIDADE) LIKE
```

```
'MED%')" `
        --no-save

# preview – 1 entrega
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute preview `
        --limit 1 `
        --view VWPACIENTES_ENTREGAS `
        --no-save

# preview – visitas e entregas simultâneas (2 de cada)
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute preview `
        --views VWPACIENTES_COMVISITAS VWPACIENTES_ENTREGAS `
        --limit 4

# send – 5 registros (visitas + entregas)
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute send --limit 5 --send

# send – 1 visita de enfermagem
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute send `
        --limit 1 `
        --view VWPACIENTES_COMVISITAS `
        --where "(UPPER(TIPOVISITA) LIKE 'ENFER%' OR UPPER(ESPECIALIDADE) LIKE
'ENFER%')" `
        --send

# send – 1 visita médica
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute send `
        --limit 1 `
        --view VWPACIENTES_COMVISITAS `
        --where "(UPPER(TIPOVISITA) LIKE 'MED%' OR UPPER(ESPECIALIDADE) LIKE
'MED%')" `
        --send

# send – 1 entrega
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute send `
        --limit 1 `
        --view VWPACIENTES_ENTREGAS `
        --send

# send – reenviar um ID específico
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute send `
        --view VWPACIENTES_COMVISITAS `
        --where "ID_ATENDIMENTO = 32668" `
        --limit 1 `
        --send

# diagnóstico rápido do SimpliRoute
```

```
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute diagnose-sr --ping

# diagnóstico do Oracle
docker compose run --rm simpliroute_cli `
    python -m src.cli.send_to_simpliroute diagnose-db --limit 3 --view
VWPACIENTES_COMVISITAS
```

Tipos de visita/entrega enviados

- `med_visit`: visitas médicas (ESPECIALIDADE/TIPOVISITA).
- `enf_visit`: visitas de enfermagem.
- `rota_log`: entregas em rota ou neutras (`TPREGISTRO = 2` ou views de entregas sem subtipo).
- `adm_log`: entregas de material para admissão (detectado por campos de tipo).
- `acr_log`: entregas por acréscimo de material.
- Tags de retirada (`ret_log`) e mudança de PAD (`pad_log`) podem ser ativadas conforme evolução da logística.
- Quando presente, a coluna `TP_ENTREGA` da view de entregas tem prioridade para definir essas tags.

6. Serviço FastAPI (`simpliroute_service` / `integrador_service`)

Como executar localmente

```
uvicorn src.integrations.simpliroute.app:app --reload --host 0.0.0.0 --port 8000
```

Endpoints

- `GET /health`, `/health/live`, `/health/ready`: endpoints de healthcheck para monitoramento e Docker.
- `POST /webhook/simpliroute`: recebe eventos do SimpliRoute, valida o token e dispara persistência no Oracle.

Webhook → Oracle

- Cada evento gera um arquivo em `data/work/webhooks/webhook_<timestamp>.json`.
- `ACAO` recebe `A`, `E` ou `S` conforme o status do SimpliRoute.
- `STATUS` usa códigos diferentes por `TPREGISTRO`:
  - `TPREGISTRO = 1` (visitas): `0` Planejada, `1` Programada, `2` Realizada.
  - `TPREGISTRO = 2` (entregas): `0` Em preparação, `2` Dispensação, `3` Em rota.
- `INFORMACAO` guarda o JSON completo para auditoria.

> Teste manualmente com `tests/manual/webhook_sample.http` (VS Code REST Client) ou adapte para `curl`.

7. Execução via Docker

Ambiente de desenvolvimento

```
docker compose build simpliroute_service
docker compose up simpliroute_service
docker compose up simpliroute_cli_limit1   # execução pontual do CLI
```

- Os containers montam `./settings` (somente leitura), `./data/output` e `./data/work` (leitura/escrita).
- Healthcheck padrão consulta `http://localhost:8000/health/live`.

Ambiente de produção (dry-run e real)

- `docker-compose.prod.yml` expõe o mesmo serviço com restart `unless-stopped` e logs limitados.
- Para subir apenas o serviço contínuo: `docker compose -f docker-compose.prod.yml up -d integrador_service`.
- Acompanhe os logs com `docker compose -f docker-compose.prod.yml logs -f integrador_service`.

8. Testes e validação

- Configure `PYTHONPATH` para a raiz (`set PYTHONPATH=%CD%` no Windows) e rode `pytest tests/test_mapper.py tests/test_mapper_fixed.py`.
- Para validar um ciclo completo manualmente:
  1. `python -m src.cli.send_to_simpliroute preview --limit 1`.
  2. `python -m src.cli.send_to_simpliroute send --limit 1 --send`.
  3. Envie um webhook de teste e confirme `ACAO/STATUS/INFORMACAO` no Oracle.

Testes em infraestrutura (modo seguro, sem envios)

Para que a equipe de infra execute testes sem enviar dados reais ao SimpliRoute, temos um procedimento seguro:

- Crie/uso do arquivo de ambiente: `settings/.env.test` (já incluído no repositório). Este arquivo deve conter, no mínimo:
  - `SIMPLIR_ROUTE_TOKEN=` (vazio)
  - `SIMPLIROUTE_POLL_WHERE=1=0`
  - `SIMPLIROUTE_DISABLE_SEND=1`
- Use um override do compose `docker-compose.test.yml` (força `preview` nos serviços CLI).
- Comando recomendado (PowerShell):

```
docker compose -f docker-compose.prod.yml -f docker-compose.test.yml --env-file
settings/.env.test up -d --build
```

- Verificações:
  - `docker compose -f docker-compose.prod.yml -f docker-compose.test.yml --env-file settings/.env.test logs --tail 200`
  - `Get-Content data/output/send_history.log -Tail 200`

- `dir data\output\send_to_sr_*.json`

Observação: o cliente HTTP respeita `SIMPLIROUTE_DISABLE_SEND=1` ou `SIMPLIROUTE_DRY_RUN=1` e retorna resposta simulada sem executar POSTs.

9. Troubleshooting

Sintoma	Causa provável	Ação sugerida
`ORA-01031: insufficient privileges` ao receber webhook	Usuário Oracle sem `UPDATE` na tabela de destino	Solicitar grant específico (UPDATE nas colunas usadas).
`ORA-00942: table or view does not exist`	`ORACLE_SCHEMA`/`SIMPLIROUTE_TARGET_TABLE` incorretos ou sem permissão	Ajustar `.env`/`config.yaml` e confirmar acesso.
`HTTP 400` com `Object with key=... does not exist`	`visit_type` inexistente no SimpliRoute	Manter `med_visit`, `enf_visit`, `rota_log` ou cadastrar o novo tipo.
Webhook `401`	Header `Authorization` ausente ou token incorreto	Definir `SIMPLIR_ROUTE_WEBHOOK_TOKEN` e enviar `Bearer <token>`.

10. Fluxo operacional sugerido

1. Verifique se o container `simpliroute_service` está saudável (`docker compose ps`).
2. Acompanhe `data/work/service_events.log` para confirmar execuções horárias.
3. Para ajustes pontuais (por exemplo, reenviar um atendimento), use o CLI com `--where` e `--limit`.
4. Sempre que o SimpliRoute alterar o catálogo de tipos, atualize a lógica em `mapper.py` antes do próximo envio.

11. Contato e autoria

Produzido por: **DAVID BARCELLOS CARDOSO TECNOLOGIA DA INFORMACAO LTDA - ME**

CNPJ: 57.929.932/0001-30

Telefone: +55 21 98605-8337