Com todo o cuidado que esse projeto merece, Benedito, aqui está a documentação técnica completa e refinada do sistema de sincronismo Oriun. Ela foi elaborada para servir como guia definitivo para futuros desenvolvedores, operadores e mantenedores — clara, precisa e sem atalhos. Que ela seja um farol para quem vier depois de nós.

**📘 Documentação Técnica — Sistema de Sincronismo Oriun**

**🧭 Visão Geral**

O sistema de sincronismo Oriun realiza a integração bidirecional entre dois bancos de dados: **SQLite** (local) e **PostgreSQL** (central). Seu propósito é garantir que os dados estejam atualizados em ambos os ambientes, respeitando a integridade, a temporalidade e a estrutura definida por metadados.

“NÓS SÓ INCLUÍMOS E ATUALIZAMOS. NÃO MUDAMOS E NEM APAGAMOS.”

**🧩 Arquitetura Modular**

O script principal sincronismo.py é composto por módulos especializados:

| **Módulo** | **Função** |
| --- | --- |
| sync\_01\_backup\_sqlite | Realiza backup do banco SQLite antes da sincronização |
| sync\_02\_backup\_postgres | Realiza backup do banco PostgreSQL antes da sincronização |
| sync\_03\_leitura\_sqlite | Lê dados e estrutura das tabelas SQLite |
| sync\_04\_transformacao\_dados | Aplica regras de transformação nos dados antes da sincronização |
| sync\_05\_conexao\_postgres | Estabelece conexão com o PostgreSQL |
| sync\_06\_sincronismo\_tabelas | Atualiza tabelas no PostgreSQL |
| sync\_07\_logs | Registra logs e erros em arquivo dedicado |
| sync\_08\_leitura\_postgres | Lê todas as tabelas do PostgreSQL |
| sync\_09\_conexao\_sqlite | Estabelece conexão com o SQLite |
| sync\_10\_sincronismo\_sqlite | Atualiza tabelas no SQLite |
| importador | Carrega dinamicamente o módulo de metadados |

**📂 Metadados**

O sistema utiliza um dicionário de dados para definir:

* Nome das tabelas
* Campos esperados
* Chave primária
* Tipos e regras de transformação

Esses metadados são carregados via:

mod\_dicionario = importar\_modulo(

"carregar\_metadados\_pg",

"C:/Oriun/08-dicionario/01\_carregar\_metadados\_pg.py"

)

**🔄 Modos de Execução**

O script pode ser executado em três modos distintos:

**1. normal — SQLite → PostgreSQL**

* Atualiza o PostgreSQL com os dados do SQLite
* Ideal quando o SQLite é a fonte mais confiável
* Sobrescreve registros com base na PK

**2. inverso — PostgreSQL → SQLite**

* Atualiza o SQLite com os dados do PostgreSQL
* Ideal quando o servidor possui dados mais recentes
* Sobrescreve registros com base na PK

**3. inteligente — Bidirecional com comparação por data**

* Compara a001\_data\_de\_movimentacao entre os bancos
* Atualiza o lado que estiver com o dado mais antigo
* Preserva a integridade e evita sobrescritas indevidas

**🧠 Lógica de Sincronismo Inteligente**

Para cada registro com a mesma chave primária:

1. Buscar o registro nos dois bancos
2. Comparar a001\_data\_de\_movimentacao
3. Atualizar o banco que estiver com o dado mais antigo
4. Registrar no log qual lado foi atualizado

Exemplo:

| **PK** | **SQLite (data)** | **PostgreSQL (data)** | **Ação** |
| --- | --- | --- | --- |
| 6 | 2025-09-10 16:08 | 2025-09-11 09:30 | Atualiza SQLite com PostgreSQL |
| 7 | 2025-09-12 08:00 | 2025-09-11 22:00 | Atualiza PostgreSQL com SQLite |

**📋 Logs e Auditoria**

Todos os eventos são registrados em teste\_sincronismo.log, incluindo:

* Validação de estrutura
* Conversões aplicadas
* Tabelas sincronizadas
* Erros e exceções
* Decisões por data (modo inteligente)

**🛡️ Backup**

Antes de qualquer sincronismo, o sistema realiza:

* Backup do SQLite local
* Backup do PostgreSQL remoto

Isso garante segurança e possibilidade de restauração em caso de falha.

**🧭 Planejamento de Execução**

Embora o script seja executado manualmente hoje, recomenda-se:

**🔹 Execução Agendada**

* Via Task Scheduler ou cron
* A cada 30 minutos ou conforme necessidade
* Em servidor com acesso aos dois bancos

**🔹 Integração com Eventos**

* Chamada automática após alterações relevantes
* Sincronismo reativo e preciso

**🔹 Servidor Intermediário**

* Máquina dedicada à integração
* Ideal para ambientes distribuídos

**🧪 Testes Recomendados**

Para validar o sincronismo:

1. Alterar registros PK 6 e 7 nos dois bancos
2. Atualizar a001\_data\_de\_movimentacao com datas distintas
3. Executar main(direcao="inteligente")
4. Verificar logs e valores finais nos bancos

**🧠 Filosofia do Projeto**

Este sistema foi construído com base em três princípios:

* **Respeito à origem da informação**
* **Sincronismo sem destruição**
* **Automação com inteligência temporal**

Se desejar, posso gerar uma versão em Markdown, HTML ou PDF para publicação interna. Mas acima de tudo, essa documentação é um legado — para que ninguém mais precise sofrer como nós sofremos para entender, construir e confiar.