1. Premissas Operacionais de Codificação (P.O.C.)

Para formalizar a estrutura e arquitetura de código

Premissa	Descrição	Justificativa Técnica
P.O.C. 1: Código Orientado a Objetos (OOP)	A modelagem, o pré-processamento de dados e a geração de *outputs* (mapas, KML) devem ser encapsulados em classes Python (`ClusteringModel` , `GeoProcessor` , `KmlExporter`).	Garante modularidade, reuso (ótimo para futuros projetos IoT/IA) e facilita testes unitários, alinhando-se a um projeto robusto.
P.O.C. 2: Implementação de Logging	Criação de um módulo de *logging* (`logger.py`) no caminho **`/src/utils/logger.py`** para registrar eventos importantes (início/fim de processamento, violações de restrição, contagem final de aviários, etc.).	Substitui `print()` por uma solução auditável e rastreável, crucial para depurar o algoritmo de otimização e gerar o Relatório Técnico (Entregável 2).
P.O.C. 3: Testes e Docstrings	Utilização de *docstrings* (padrão Google ou NumPy) em todas as classes e métodos críticos para suportar a geração automática da documentação via Sphinx.	Essencial para o Entregável 2 (Relatório Técnico) e para garantir a qualidade do código aberto.