**📘 PRD – Sistema de Ingestão NIST (S3 ↔ Banco MITRA)**

**1. Visão Geral**

O sistema **MITRA NIST** integra dados biométricos e biográficos oriundos de arquivos NIST armazenados em um sistema compatível com **S3 (MinIO)**, realizando leitura, extração e persistência em um **banco de dados relacional (PostgreSQL)** sob o schema findface.

A arquitetura deverá seguir **TDD raiz**, **DDD**, **SOLID**, **DRY** e **arquitetura hexagonal**, de modo que o domínio permaneça **independente da tecnologia**.

**2. Objetivo Funcional**

1. Conectar-se ao sistema S3 (endpoint configurável).
2. Ler arquivos .nst e extrair campos essenciais.
3. Persistir os dados em suas tabelas correspondentes.
4. Evitar duplicação por md5\_hash ou documento.
5. Movimentar os NISTs processados para <prefixo>-lidos/.
6. Registrar logs e exceções em tb\_log.
7. Permitir reprocessamentos seguros e reversíveis.

**3. Requisitos Funcionais**

| **Código** | **Requisito** | **Descrição** |
| --- | --- | --- |
| **RF01** | Upload de NIST | Enviar arquivos locais .nst para o S3 no formato nist/<campo\_1:008>/<nome\_arquivo>.nst. |
| **RF02** | Leitura no S3 | Ler objetos .nst diretamente do bucket configurado. |
| **RF03** | Extração de Campos | Extrair e interpretar campos NIST, especialmente 1:008 (origem), 2.030 (nome), 2.035 (data de nascimento), 2.038 (nacionalidade), 2.039 (sexo), etc. |
| **RF04** | Normalização | Padronizar acentuação, espaços e capitalização dos dados. |
| **RF05** | Persistência | Inserir ou atualizar registros em tb\_nist e tb\_base\_origem. |
| **RF06** | Deduplicação | Evitar duplicações com base em md5\_hash ou (nome, nascimento, mãe). |
| **RF07** | Log de Operações | Registrar todos os inserts, updates e erros em tb\_log. |
| **RF08** | Movimentação | Após processar, mover o arquivo para \*-lidos/. |
| **RF09** | Reprocessamento Seguro | Ignorar arquivos já existentes em \*-lidos/. |
| **RF10** | Testes Automatizados | Desenvolver todos os componentes por **TDD**. |

**4. Requisitos Não Funcionais**

| **Categoria** | **Especificação** |
| --- | --- |
| Linguagem | Python 3.11+ |
| Testes | Pytest com cobertura mínima de 95% |
| ORM | SQLAlchemy |
| Confiabilidade | Transações atômicas com rollback em falhas |
| Logging | Configurável via logging padrão |
| Desempenho | Capaz de processar 2.000 NISTs/lote |
| Escalabilidade | Adaptável a múltiplos buckets simultaneamente |
| Segurança | Validação de tipos, erros tratados e logs consistentes |

**5. Fluxo de Dados (Macro)**

1. **Listagem** — Identifica novos NISTs no bucket.
2. **Download** — Lê bytes e extrai 1:008.
3. **Parsing** — Converte dados brutos em entidades de domínio.
4. **Persistência** — Realiza upsert no banco.
5. **Movimentação** — Move o arquivo processado para -lidos/.
6. **Logging** — Cria log de sucesso ou falha.

**6. Estrutura da Chave S3 (Regra Fixa)**

Ao enviar um arquivo NIST, **a chave S3 deverá seguir rigorosamente o formato:**

nist/<valor do campo 1:008>/<nome do arquivo>.nst

Exemplo:

Campo 1:008 → "PF/RR"

Arquivo → 123456789.nst

Chave final → nist/PF/RR/123456789.nst

Se o campo 1:008 for inexistente ou ilegível, o diretório padrão será nist/desconhecido/.

**7. Entidades Principais (DDD)**

**7.1 Pessoa**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| name | str | Nome da pessoa |
| social\_name | str | Nome social |
| birth\_date | date | Data de nascimento |
| sex | str | M, F ou U |
| mother\_name | str | Nome da mãe |
| father\_name | str | Nome do pai |
| birthplace | str | Naturalidade |
| nationality | str | País |
| passport | str | Passaporte |
| document | str | Documento |
| uri\_nist | str | Caminho S3 |
| md5\_hash | str | Hash do arquivo |
| active | bool | Ativo/inativo |
| origin\_name | str | Origem (campo 1:008) |

**7.2 BaseOrigem**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id\_base\_origem | int | Identificador |
| name | str | Nome da base |
| active | bool | Status |

**7.3 Log**

| **Campo** | **Tipo** | **Descrição** |
| --- | --- | --- |
| id\_log | int | Identificador |
| id\_origem | int | Chave associada |
| cd\_tipo\_log | int | Código do log |
| ds\_log | str | Mensagem |
| dt\_log | datetime | Data/hora |

**8. Arquitetura e Pastas**

project/

│

├── domain/

│ ├── pessoa\_domain.py

│ └── nist\_domain.py

│

├── application/

│ ├── usecases/

│ │ ├── process\_nist\_usecase.py

│ │ ├── upload\_nist\_usecase.py

│ │ └── move\_processed\_usecase.py

│ └── services/

│ ├── nist\_parser\_service.py

│ ├── normalization\_service.py

│ └── checksum\_service.py

│

├── infra/

│ ├── s3/

│ │ ├── miniosdk.py

│ │ └── minio\_manager.py

│ ├── db/

│ │ ├── orm\_db.py

│ │ ├── orm\_models.py

│ │ ├── pessoa\_repository.py

│ │ └── orm\_handlers.py

│ └── sanitizers.py

│

├── cli/

│ ├── nist\_manager.py

│ └── uploader\_cli.py

│

└── tests/

├── unit/

├── integration/

└── e2e/

**9. Casos de Uso (Application Layer)**

**UC01 — Processar NIST**

1. Receber key de um NIST no S3.
2. Baixar bytes e extrair campos.
3. Criar Pessoa e BaseOrigem.
4. Persistir no banco (upsert).
5. Mover o arquivo para nist-lidos/<origem>/....
6. Criar log de operação.

**UC02 — Upload NIST**

1. Receber caminho local de um .nst.
2. Extrair 1:008 e montar chave S3.
3. Fazer upload.
4. Retornar URI completa.

**UC03 — Reset Banco**

1. Truncar tabelas e resetar sequences.
2. Logar ação.

**10. Testes (TDD Raiz)**

| **Teste** | **Objetivo** |
| --- | --- |
| test\_sanitizers.py | Normalização de texto, datas e sexo |
| test\_minio\_key\_structure.py | Garantir estrutura nist/<1:008>/arquivo.nst |
| test\_parser\_nist.py | Validação de extração de campos |
| test\_repository\_upsert.py | Upsert com deduplicação por md5/documento |
| test\_process\_nist\_usecase.py | Fluxo completo: leitura, persistência e movimentação |
| test\_move\_object.py | Confirma movimentação para -lidos/ |
| test\_logs.py | Verifica criação de logs em sucesso e erro |

**11. Critérios de Aceite**

* ✅ Todos os testes verdes.
* ✅ Estrutura de chave S3 conforme padrão.
* ✅ Logs consistentes.
* ✅ Domínio agnóstico de tecnologia.
* ✅ Transações com rollback.
* ✅ Movimentação segura para -lidos/.
* ✅ Cobertura mínima de 95%.

**12. Padrões de Código**

* Docstrings em **Português-BR**.
* **Type hints** obrigatórios.
* **Verificações de tipo** (isinstance).
* **Nomes em inglês** e sem abreviações.
* **Erros tratados** e logados.
* **Formato S3 fixo:** nist/<campo\_1:008>/<arquivo>.nst.

**13. Configuração (12-Factor)**

S3\_ENDPOINT=http://10.95.4.61:9001

S3\_BUCKET=teste

S3\_ACCESS=teste

S3\_SECRET=Nv30R1Yt

S3\_SECURE=false

PG\_HOST=127.0.0.1

PG\_PORT=5432

PG\_DB=mitra

PG\_USER=postgres

PG\_PASSWORD=\*\*\*\*\*

LOG\_LEVEL=INFO

**🧩 Diagrama Arquitetural (DDD + Hexagonal)**

+---------------------------------------------------------------+

| MITRA NIST |

|---------------------------------------------------------------|

| Interface (CLI / Scheduler) |

| ├── nist\_manager.py |

| └── uploader\_cli.py |

|---------------------------------------------------------------|

| Application Layer |

| ├── UseCases |

| │ ├── process\_nist\_usecase.py |

| │ ├── upload\_nist\_usecase.py |

| │ └── move\_processed\_usecase.py |

| ├── Services |

| │ ├── nist\_parser\_service.py |

| │ ├── normalization\_service.py |

| │ └── checksum\_service.py |

| ├── Ports (interfaces) |

| │ ├── s3\_port.py |

| │ └── repository\_port.py |

|---------------------------------------------------------------|

| Domain Layer |

| ├── pessoa\_domain.py |

| └── nist\_domain.py |

|---------------------------------------------------------------|

| Infrastructure Layer |

| ├── S3 Adapter |

| │ ├── miniosdk.py |

| │ └── minio\_manager.py |

| ├── DB Adapter |

| │ ├── orm\_models.py |

| │ ├── pessoa\_repository.py |

| │ ├── orm\_db.py |

| │ └── orm\_handlers.py |

| └── sanitizers.py |

+---------------------------------------------------------------+

**Sequência UC01 (Processar NIST)**

Operador → CLI → UseCase → Parser → Repository → S3

1. CLI invoca process\_nist\_usecase

2. UseCase requisita lista de objetos ao S3

3. S3 retorna chaves "nist/<1:008>/<arquivo>.nst"

4. UseCase lê bytes e envia ao Parser

5. Parser extrai campos e cria entidade Pessoa

6. Repository faz upsert em tb\_nist

7. Log é criado em tb\_log

8. Arquivo é movido para nist-lidos/<1:008>/arquivo.nst

9. Resultado é retornado

**🧪 Bloco de Testes de Contrato (TDD Inicial)**

import pytest

from pathlib import Path

from hashlib import md5

from infra.sanitizers import sanitize\_text, parse\_date, normalize\_sex

from infra.s3.minio\_manager import \_campo\_1\_008

# =======================

# Testes de Normalização

# =======================

def test\_sanitize\_text\_removes\_accents\_and\_whitespace():

assert sanitize\_text(" João da Silva ") == "JOAO DA SILVA"

def test\_parse\_date\_accepts\_multiple\_formats():

assert str(parse\_date("19870115")) == "1987-01-15"

assert str(parse\_date("15/01/1987")) == "1987-01-15"

assert parse\_date("INVALID") is None

def test\_normalize\_sex\_maps\_codes\_to\_letters():

assert normalize\_sex("1") == "M"

assert normalize\_sex("2") == "F"

assert normalize\_sex("3") == "U"

assert normalize\_sex("M") == "M"

assert normalize\_sex("F") == "F"

# =======================

# Testes de Chave S3

# =======================

def test\_s3\_key\_structure\_respects\_1\_008\_value(tmp\_path):

nist\_bytes = b"1.008=PF/RR\n2.030=JOAO DA SILVA"

campo = \_campo\_1\_008(nist\_bytes)

expected\_path = f"nist/{campo}/123456.nst"

assert expected\_path == "nist/PF/RR/123456.nst"

def test\_s3\_key\_structure\_unknown\_defaults\_to\_desconhecido():

campo = \_campo\_1\_008(b"sem campo valido")

assert campo == "desconhecido"

# =======================

# Testes de Deduplicação

# =======================

def test\_md5\_uniqueness\_same\_bytes\_same\_hash():

data = b"teste"

assert md5(data).hexdigest() == md5(b"teste").hexdigest()

def test\_md5\_difference\_for\_different\_files():

assert md5(b"A").hexdigest() != md5(b"B").hexdigest()