Projet Fédérateur

Système de Gouvernance et Protection des Données Sensibles

Architecture Microservices avec FastAPI + MongoDB

Table des matières

1 Introduction

1.1 Contexte du Projet

Dans un contexte où la protection des données personnelles est devenue une priorité réglementaire et éthique, ce projet vise à développer un système complet de gouvernance des données sensibles. Le système doit être conforme aux réglementations RGPD européennes et à la Loi 09-08 marocaine, tout en garantissant la qualité et la traçabilité des données.

1.2 Objectifs Stratégiques

— Développer une plateforme modulaire de gestion des données sensibles
— Implémenter des mécanismes automatisés de détection et protection
— Assurer la conformité RGPD et Loi 09-08 marocaine
— Garantir la qualité des données selon les normes ISO 8000 et ISO 25012
— Créer un système extensible basé sur une architecture microservices
— Intégrer un framework de masquage contextuel intelligent (EthiMask)

1.3 Stack Technique

— **Backend :** Python avec FastAPI
— **Base de données :** MongoDB (via MongoDB Compass)
— **ODM :** Motor + Beanie (async)
— **Orchestration :** Apache Airflow (workflows YAML)
— **Gouvernance Big Data :** Apache Atlas + Apache Ranger (HDP Sandbox)
— **ML/NLP :** HuggingFace Transformers, spaCy
— **Privacy :** Microsoft Presidio, TenSEAL

1.4 Architecture Microservices

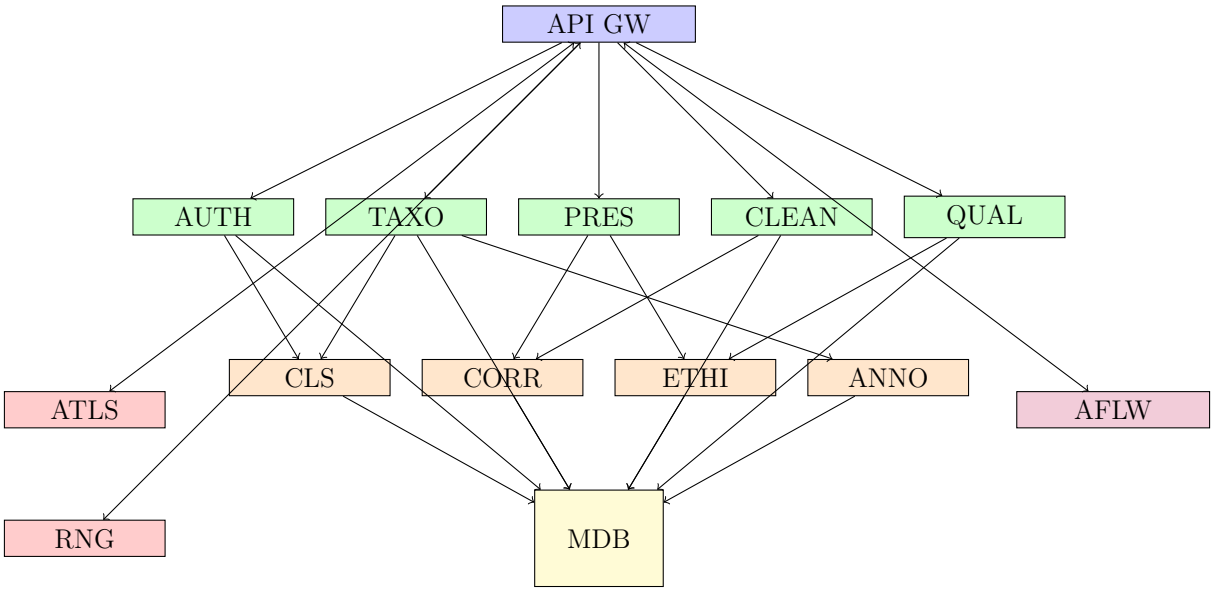

FIGURE 1 – Architecture Microservices du Système (avec abréviations)

Abréviations :

- **API GW** : API Gateway (FastAPI)
- **AUTH** : Authentification Service
- **TAXO** : Taxonomy Service
- **PRES** : Presidio Service
- **CLEAN** : Cleaning Service
- **QUAL** : Quality Service
- **CLS** : Classification Service
- **CORR** : Correction Service
- **ETHI** : EthiMask Service
- **ANNO** : Annotation Service
- **MDB** : MongoDB
- **AFLW** : Apache Airflow (Orchestration)
- **ATLS** : Apache Atlas (Metadata Governance)
- **RNG** : Apache Ranger (Policies / Security)

2 Orchestration des Workflows avec Apache Airflow

2.1 Pourquoi Apache Airflow ?

Apache Airflow est la solution d'orchestration recommandée pour ce projet car :

- **Workflows YAML/Python :** Configuration déclarative et programmable
- **Extensibilité :** Ajout de nouveaux modules sans modification du core
- **Monitoring :** Interface web pour surveiller l'exécution des pipelines
- **Retry & Alerting :** Gestion automatique des erreurs
- **Intégration :** Compatible avec FastAPI, MongoDB, et services externes

2.2 Installation et Configuration

Installation :

```
pip install apache-airflow
pip install apache-airflow-providers-http
pip install apache-airflow-providers-mongo
```

Configuration (airflow.cfg) :

```
[core]
dags_folder = /path/to/dags
executor = LocalExecutor
sql_alchemy_conn = sqlite:////path/to/airflow.db

[webserver]
base_url = http://localhost:8080
web_server_port = 8080
```

2.3 Architecture des DAGs

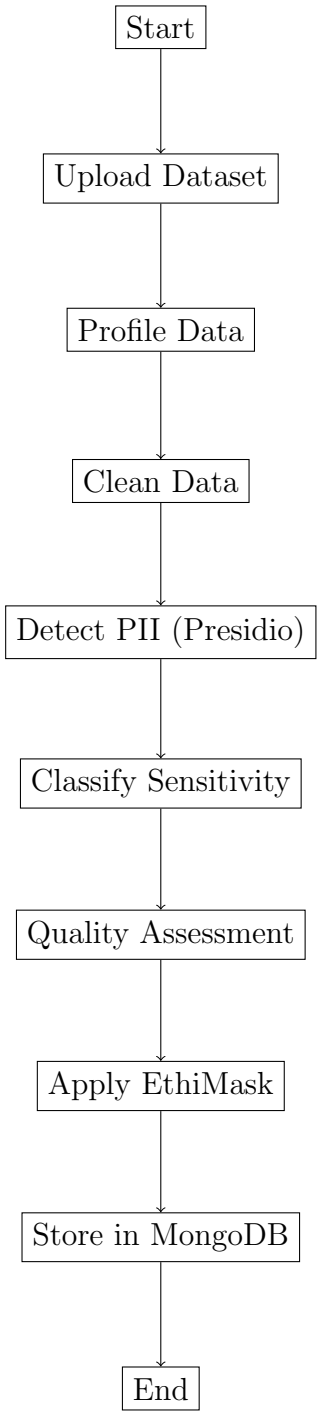

FIGURE 2 – Pipeline Principal de Traitement des Données

2.4 Exemple de DAG pour le Pipeline Complet

Fichier : dags/data_processing_pipeline.py
Note : Le code complet est fourni dans les ressources du projet. Voici la structure :

— Définition du DAG avec schedule et retry policy
— Task 1 : Upload et validation du dataset

— Task 2 : Profiling avec ydata-profiling
— Task 3 : Nettoyage (pipeline Python)
— Task 4 : Détection PII avec Presidio
— Task 5 : Classification fine-grained
— Task 6 : Évaluation qualité ISO
— Task 7 : Application EthiMask
— Task 8 : Stockage MongoDB et Atlas

2.5 Exemple de Configuration YAML

Fichier : config/pipeline_config.yaml

```yaml
pipeline:
  name: "data_processing_pipeline"
  schedule_interval: "0 2 * * *"  # Daily at 2AM

tasks:
  upload_dataset:
    service: "cleaning_service"
    endpoint: "/upload"
    retry: 3

  profile_data:
    service: "cleaning_service"
    endpoint: "/profile/{dataset_id}"
    depends_on: ["upload_dataset"]

  clean_data:
    service: "cleaning_service"
    endpoint: "/clean/{dataset_id}"
    params:
      remove_duplicates: true
      handle_missing: "mean"
      remove_outliers: true
    depends_on: ["profile_data"]

  detect_pii:
    service: "presidio_service"
    endpoint: "/analyze"
    depends_on: ["clean_data"]

  classify_sensitivity:
    service: "classification_service"
    endpoint: "/classify/{dataset_id}"
    depends_on: ["detect_pii"]

  quality_assessment:
```

```
    service: "quality_service"
    endpoint: "/evaluate/{dataset_id}"
    depends_on: ["clean_data"]

  apply_ethimask:
    service: "ethimask_service"
    endpoint: "/mask/{dataset_id}"
    depends_on: ["classify_sensitivity", "quality_assessment"]

  store_results:
    service: "storage_service"
    endpoint: "/store"
    depends_on: ["apply_ethimask"]
```

2.6 Intégration avec FastAPI

Pour que chaque microservice soit appelable par Airflow, chaque service FastAPI expose des endpoints standardisés :

— `POST /api/v1/{service}/execute` - Exécution de la tâche

— `GET /api/v1/{service}/status/{task_id}` - Statut de la tâche

— `GET /api/v1/{service}/result/{task_id}` - Résultat de la tâche

Communication Airflow → FastAPI :

— Utilisation de `SimpleHttpOperator` d'Airflow

— Authentification via JWT tokens

— Retry automatique en cas d'échec

— Logging centralisé

3 Tâche 1 : Module d'Authentification et Gestion des Rôles

3.1 Description

Développer un système d'authentification robuste avec gestion granulaire des rôles (Admin, Data Steward, Data Annotator, Data Labeler) utilisant FastAPI et JWT, intégré avec Apache Ranger pour le contrôle d'accès.

3.2 User Stories

1. **US-AUTH-01 :** En tant qu'utilisateur, je veux m'inscrire avec email/mot de passe pour accéder au système
2. **US-AUTH-02 :** En tant qu'utilisateur, je veux me connecter et recevoir un token JWT pour authentifier mes requêtes
3. **US-AUTH-03 :** En tant qu'Admin, je veux assigner des rôles aux utilisateurs pour gérer les permissions
4. **US-AUTH-04 :** En tant que système, je veux vérifier les permissions selon le rôle avant d'autoriser une action
5. **US-AUTH-05 :** En tant qu'Admin, je veux intégrer les politiques Apache Ranger pour contrôler l'accès aux données sensibles
6. **US-AUTH-06 :** En tant que Data Steward, je veux consulter les logs d'accès via Apache Atlas pour auditer les actions

3.3 Technologies et Bibliothèques

— **FastAPI :** Framework web asynchrone
— **python-jose :** Gestion JWT
— **passlib :** Hashing sécurisé des mots de passe
— **Motor + Beanie :** ODM MongoDB asynchrone
— **pydantic :** Validation des données

3.4 Documentation de Référence

— FastAPI Security : https://fastapi.tiangolo.com/tutorial/security/
— JWT Best Practices : https://datatracker.ietf.org/doc/html/rfc8725
— OWASP Auth Cheat Sheet : https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
— Apache Ranger : https://ranger.apache.org/

3.5 Algorithme de Vérification JWT

Algorithm 1 Vérification et Validation du Token JWT

Require: Token JWT, Secret Key, Algorithm
Ensure: User authentifié ou Exception

```
token ← Extract from Authorization Header
if token is None then
    raise UnauthorizedException
end if
payload ← jwt.decode(token, secret_key, algorithm)
email ← payload["sub"]
user ← Database.findOne({email : email})
if user is None OR NOT user.is_active then
    raise UnauthorizedException
end if
user.last_login ← Current UTC Time
Database.save(user)
return  user JWTError
raise UnauthorizedException
```

3.6 Intégration Apache Ranger

Apache Ranger permet de définir des politiques d'accès granulaires :

- **Resource-based policies :** Contrôle par dataset, table, colonne
- **Tag-based policies :** Contrôle par classification (PII, SPI, etc.)
- **Masking policies :** Masquage automatique selon le rôle
- **Row-level filtering :** Filtrage des données sensibles

Workflow d'intégration :

1. L'utilisateur s'authentifie via FastAPI → Reçoit JWT
2. Requête d'accès à une donnée sensible
3. FastAPI valide le JWT → Extrait le rôle
4. Appel à Ranger REST API pour vérifier la politique
5. Si autorisé, appliquer le niveau de masquage approprié
6. Retourner les données (masquées si nécessaire)

3.7 KPIs

- Temps de réponse authentification ¡ 100ms
- Taux de succès > 99.9%
- Zéro faille de sécurité (OWASP Top 10)
- Code coverage > 85%
- Intégration Ranger fonctionnelle

3.8 Livrables

1. Code source FastAPI avec structure microservice

2. Documentation API Swagger auto-générée
3. Tests unitaires et d'intégration (pytest)
4. Configuration Apache Ranger (policies JSON)
5. Guide de déploiement

4 Tâche 2 : Taxonomie Fine-Grained des Données Sensibles (PII/SPI)

4.1 Description

Créer une taxonomie détaillée des données sensibles incluant les PII (Personally Identifiable Information) et SPI (Sensitive Personal Information), conforme au RGPD et à la Loi 09-08 marocaine.

4.2 User Stories

1. **US-TAX-01 :** En tant que Data Steward, je veux consulter la taxonomie complète via l'API pour comprendre les classifications
2. **US-TAX-02 :** En tant qu'Admin, je veux ajouter/modifier des catégories PII/SPI dans MongoDB Compass
3. **US-TAX-03 :** En tant que système, je veux valider automatiquement les données selon la taxonomie
4. **US-TAX-04 :** En tant que développeur, je veux peupler la base avec le script de taxonomie
5. **US-TAX-05 :** En tant que Data Steward, je veux calculer le score de sensibilité d'un attribut
6. **US-TAX-06 :** En tant que système, je veux synchroniser la taxonomie avec Apache Atlas pour la gouvernance

4.3 Classification PII vs SPI

Catégorie	Exemples	Sensibilité
PII Direct	Nom, prénom, CIN, passeport	HIGH
PII Indirect	Adresse IP, cookies, identifiants	MEDIUM
SPI - Santé	Dossier médical, CNSS, groupe sanguin	CRITICAL
SPI - Finance	IBAN, RIB, carte bancaire, salaire	CRITICAL
SPI - Biométrie	Empreintes, iris, reconnaissance faciale	CRITICAL
SPI - Origine	Origine ethnique, religieuse, politique	CRITICAL
Quasi-identifiants	Code postal + âge + sexe	MEDIUM-HIGH

4.4 Formule de Score de Sensibilité

Le score de sensibilité est calculé selon la formule pondérée :

$$S_{total} = \alpha \cdot L_{legal} + \beta \cdot R_{risk} + \gamma \cdot I_{impact} \tag{1}$$

Où :

- L_{legal} : Niveau d'obligation légale (RGPD/Loi 09-08) $\in [0, 1]$
- R_{risk} : Risque d'atteinte à la vie privée $\in [0, 1]$
- I_{impact} : Impact en cas de fuite $\in [0, 1]$
- Poids suggérés : $\alpha = 0.4$, $\beta = 0.3$, $\gamma = 0.3$

Mapping Score → Niveau :

- $S \in [0, 0.3[$: LOW
- $S \in [0.3, 0.6[$: MEDIUM
- $S \in [0.6, 0.85[$: HIGH
- $S \in [0.85, 1.0]$: CRITICAL

4.5 Documentation de Référence

- RGPD (GDPR) : https://gdpr-info.eu/
- Loi 09-08 Maroc : https://www.dgssi.gov.ma/fr/loi-09-08-relative-la-protection-des-p
- NIST Privacy Framework : https://www.nist.gov/privacy-framework
- Apache Atlas REST API : https://atlas.apache.org/api/v2/index.html

4.6 Intégration Apache Atlas

Apache Atlas permet de créer un catalogue de données avec lignage et classification :

Synchronisation Taxonomie → Atlas :

1. Créer des types d'entités Atlas pour chaque catégorie PII/SPI
2. Définir des classifications (tags) : PII, SPI, CRITICAL, etc.
3. Associer les métadonnées de sensibilité
4. Créer le lignage des transformations de données
5. Activer les notifications d'événements

Exemple de création d'entité Atlas :

```
POST /api/atlas/v2/entity
{
  "entity": {
    "typeName": "data_attribute",
    "attributes": {
      "name": "CIN_Marocain",
      "qualifiedName": "PII-001::ID-CIN",
      "sensitivity_level": "HIGH",
      "legal_basis": ["RGPD Art.6", "Loi 09-08 Art.3"],
      "encryption_required": true
    },
    "classifications": [
      {"typeName": "PII"},
      {"typeName": "HIGH_SENSITIVITY"}
    ]
  }
}
```

4.7 Algorithme de Classification Automatique

Algorithm 2 Classification Automatique d'un Attribut

Require: Attribute A, Taxonomy T, Threshold θ
Ensure: Classification & Sensitivity Score

$candidates \leftarrow []$
for each *category* in T **do**
  for each *datatype* in *category.datatypes* **do**
    $score \leftarrow 0$
    if *A.name* matches *datatype.keywords* **then**
      $score \leftarrow score + 0.3$
    end if
    if *A.value* matches *datatype.regex_pattern* **then**
      $score \leftarrow score + 0.5$
    end if
    if *A.context* matches *datatype.context* **then**
      $score \leftarrow score + 0.2$
    end if
    if $score \geq \theta$ **then**
      $candidates.append(\{datatype, score\})$
    end if
  end for
end for
$best \leftarrow argmax(candidates, key = score)$
$sensitivity \leftarrow$ ComputeSensitivity($best$)
return *best.datatype*, *sensitivity*

4.8 KPIs

— Couverture taxonomie : > 50 types de données
— Précision classification : $> 90\%$
— Temps de recherche : < 50ms
— Synchronisation Atlas : < 2s par entité

4.9 Livrables

1. Code source FastAPI CRUD taxonomie
2. Script Python de population MongoDB
3. Collection MongoDB exportée (JSON)
4. Types et classifications Atlas (JSON)
5. Mapping RGPD/Loi 09-08 complet
6. Documentation API Swagger

5 Tâche 3 : Personnalisation Microsoft Presidio pour le Contexte Marocain

5.1 Description

Adapter Microsoft Presidio pour détecter les données sensibles spécifiques au contexte marocain (CIN, téléphones, IBAN MA, CNSS) et l'intégrer dans l'architecture microservice.

5.2 User Stories

1. **US-PRES-01 :** En tant que système, je veux détecter automatiquement les CIN marocains dans un texte
2. **US-PRES-02 :** En tant que Data Steward, je veux personnaliser les patterns de détection via configuration
3. **US-PRES-03 :** En tant qu'utilisateur API, je veux anonymiser un texte contenant des données sensibles
4. **US-PRES-04 :** En tant que développeur, je veux tester les recognizers avec des jeux de données
5. **US-PRES-05 :** En tant que système, je veux intégrer la détection avec le pipeline Airflow
6. **US-PRES-06 :** En tant que Data Annotator, je veux corriger les faux positifs détectés

5.3 Documentation de Référence

— Microsoft Presidio : https://microsoft.github.io/presidio/
— Presidio Analyzer API : https://microsoft.github.io/presidio/analyzer/
— Presidio Anonymizer API : https://microsoft.github.io/presidio/anonymizer/
— spaCy French Models : https://spacy.io/models/fr
— Regex101 (test patterns) : https://regex101.com/

5.4 Recognizers Personnalisés

Entités Marocaines Supportées :

— **CIN_MAROC :** Carte d'identité (format : 1-2 lettres + 6 chiffres)
— **PHONE_MA :** Téléphone mobile (+212 ou 0) [5-7]XXXXXXXX
— **IBAN_MA :** IBAN bancaire (MA + 24 chiffres)
— **CNSS_MA :** Numéro sécurité sociale (10 chiffres)
— **PASSPORT_MA :** Passeport marocain
— **PERMIS_MA :** Permis de conduire

5.5 Algorithme de Détection avec Score de Confiance

Algorithm 3 Détection Multi-Pattern avec Score Pondéré

Require: Text T, Patterns P, Context C, Threshold θ
Ensure: List of Recognized Entities

```
results ← []
nlp_artifacts ← spaCy.process(T)
for each pattern in P do
    matches ← Regex.findAll(T, pattern.regex)
    for each match in matches do
        score_pattern ← pattern.base_score
        score_context ← ComputeContextScore(match, C, nlp_artifacts)
        score_validation ← ValidatePattern(match.text, pattern.validator)
        score_final ← 0.5 · score_pattern + 0.3 · score_context + 0.2 · score_validation
        if score_final ≥ θ then
            results.append({entity : pattern.type, start : match.start, end :
            match.end, score : score_final})
        end if
    end for
end for
results ← RemoveOverlaps(results)
return results
```

5.6 Métriques d'Évaluation

Formules :

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

Objectifs :

- Precision $> 90\%$
- Recall $> 85\%$
- F1-Score $> 87\%$
- Temps traitement < 500ms/document

5.7 Datasets de Test

- **Dataset Piilo :** https://www.kaggle.com/datasets/lburleigh/piilo-dataset
- **Dataset Enron :** https://www.kaggle.com/datasets/wcukierski/enron-email-dataset - Emails pour test contexte professionnel
- **Dataset MAPA :** Créer un dataset synthétique marocain avec CIN, téléphones, IBAN

5.8 KPIs

— Précision détection CIN : $> 95\%$
— Taux faux positifs : $< 5\%$
— Couverture patterns : 100% formats marocains
— Performance API : < 200ms response time

5.9 Livrables

1. Code source recognizers personnalisés
2. Routes FastAPI pour analyse/anonymisation
3. Dataset de test annoté (500+ exemples)
4. Tests unitaires pytest (coverage $> 85\%$)
5. Rapport de performance avec métriques
6. Documentation technique recognizers

6 Tâche 4 : Data Cleaning et Profiling

6.1 Description

Développer un pipeline complet de nettoyage et profilage des données avec détection automatique des anomalies.

6.2 User Stories

1. **US-CLEAN-01 :** En tant que Data Annotator, je veux uploader un CSV pour le nettoyer
2. **US-CLEAN-02 :** En tant que Data Steward, je veux générer un rapport de profiling HTML
3. **US-CLEAN-03 :** En tant que système, je veux appliquer automatiquement le pipeline de cleaning
4. **US-CLEAN-04 :** En tant qu'utilisateur, je veux télécharger le dataset nettoyé
5. **US-CLEAN-05 :** En tant qu'Admin, je veux consulter les métadonnées dans MongoDB Compass
6. **US-CLEAN-06 :** En tant que système, je veux déclencher le cleaning via Airflow DAG

6.3 Documentation de Référence

— Pandas Documentation : https://pandas.pydata.org/docs/

— ydata-profiling : https://docs.profiling.ydata.ai/

— Scikit-learn Preprocessing : https://scikit-learn.org/stable/modules/preprocessing.html

6.4 Pipeline de Nettoyage

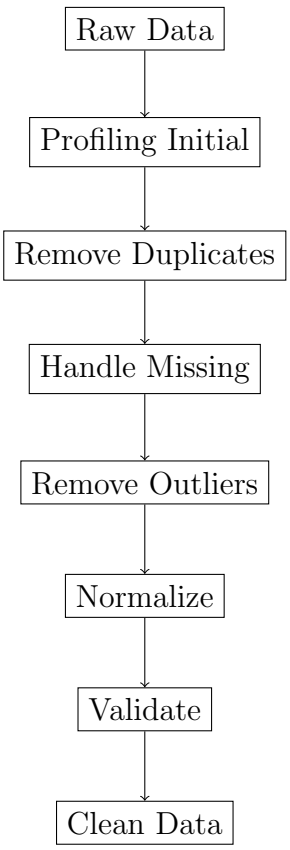

FIGURE 3 – Pipeline de Data Cleaning

6.5 Algorithme de Détection d'Outliers IQR

Algorithm 4 Détection et Suppression Outliers (Méthode IQR)

Require: DataFrame D, Columns C, Multiplier $m = 1.5$
Ensure: Cleaned DataFrame

for each col in C **do**
  $Q1 \leftarrow \text{Quantile}(D[col], 0.25)$
  $Q3 \leftarrow \text{Quantile}(D[col], 0.75)$
  $IQR \leftarrow Q3 - Q1$
  $lower\_bound \leftarrow Q1 - m \cdot IQR$
  $upper\_bound \leftarrow Q3 + m \cdot IQR$
  $D \leftarrow D[(D[col] \geq lower\_bound) \wedge (D[col] \leq upper\_bound)]$
end for
return D

6.6 KPIs

— Taux valeurs manquantes : $< 5\%$
— Taux doublons : $= 0\%$

— Taux outliers : $< 2\%$
— Score qualité global : $> 85/100$
— Temps traitement : < 10s pour 10k lignes

6.7 Livrables

1. Code Python pipeline cleaning
2. Routes FastAPI upload/clean/download
3. Rapport profiling HTML (ydata-profiling)
4. Tests unitaires pytest
5. Documentation technique
6. DAG Airflow pour automatisation

7 Tâche 5 : Système de Classification Fine-Grained

7.1 Description

Développer un système de classification multi-niveaux des données sensibles utilisant des modèles de Machine Learning et NLP.

7.2 User Stories

1. **US-CLASS-01 :** En tant que système, je veux classifier automatiquement chaque colonne d'un dataset
2. **US-CLASS-02 :** En tant que Data Steward, je veux ajuster les seuils de classification
3. **US-CLASS-03 :** En tant que développeur, je veux entraîner le modèle sur de nouvelles données
4. **US-CLASS-04 :** En tant qu'utilisateur, je veux visualiser la distribution des classes
5. **US-CLASS-05 :** En tant que système, je veux intégrer la classification avec Apache Atlas

7.3 Architecture du Classifieur

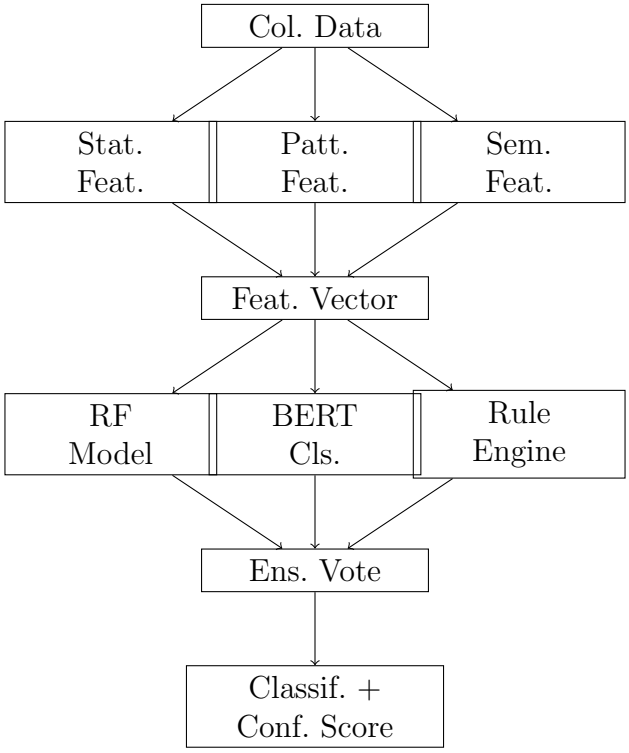

FIGURE 4 – Architecture compacte du classifieur avec abréviations

Abréviations :

— **Col. Data** : Column Data (données de colonne)

— **Stat. Feat.** : Statistical Features

— **Patt. Feat.** : Pattern Features
— **Sem. Feat.** : Semantic Features
— **Feat. Vector** : Feature Vector
— **RF Model** : Random Forest Model
— **BERT Cls.** : BERT Classifier
— **Rule Engine** : Classification par règles
— **Ens. Vote** : Ensemble Voting Mechanism
— **Conf. Score** : Confidence Score

7.4 Modèles HuggingFace Recommandés

— **CamemBERT :** https://huggingface.co/camembert-base - Modèle BERT français
— **FlauBERT :** https://huggingface.co/flaubert/flaubert_base_cased - Alternative française
— **DistilBERT Multilingual :** https://huggingface.co/distilbert-base-multilingual-case - Plus léger
— **RoBERTa-base :** Pour fine-tuning sur classification PII

Exemple de Fine-tuning :

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer, Trainer

model_name = "camembert-base"
model = AutoModelForSequenceClassification.from_pretrained(
    model_name,
    num_labels=len(SENSITIVITY_CLASSES)
)
tokenizer = AutoTokenizer.from_pretrained(model_name)

# Training avec dataset annoté
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=eval_dataset
)
trainer.train()
```

7.5 Algorithme de Classification Ensemble

7.6 Classes de Sensibilité

Niveau	Code	Description
0	PUBLIC	Données publiques, non sensibles

Niveau	Code	Description
1	INTERNAL	Données internes, usage limité
2	CONFIDENTIAL	Données confidentielles
3	PII	Identifiants personnels
4	SPI	Données personnelles sensibles
5	CRITICAL	Données critiques (santé, finance)

7.7 KPIs

— Accuracy globale : $> 92\%$

— F1-Score par classe : > 0.88

— Temps classification : < 2s par colonne

— Taux review manuel : $< 10\%$

7.8 Livrables

1. Code modèles ML (Random Forest, BERT)
2. Routes FastAPI classification
3. Modèles entraînés (pickle/HuggingFace)
4. Dataset d'entraînement annoté
5. Rapport d'évaluation avec métriques
6. Intégration Atlas pour tagging automatique

Algorithm 5 Classification Ensemble Multi-Modèles

Require: Column C, Models $M = \{M_1, M_2, ..., M_n\}$, Weights W
Ensure: Class Label, Confidence Score

$features \leftarrow$ ExtractFeatures(C)
$predictions \leftarrow []$
for each $model$ in M **do**
  $pred, conf \leftarrow model$.predict($features$)
  $predictions.append(\{pred, conf, weight : W[model]\})$
end for
$votes \leftarrow$ WeightedVote($predictions$)
$final\_class \leftarrow argmax(votes)$
$confidence \leftarrow$ ComputeEnsembleConfidence($predictions$, $final\_class$)
if $confidence < threshold$ **then**
  flag for manual review
end if
return $final\_class$, $confidence$

8 Tâche 6 : Correction Automatique des Incohérences (Data Quality V2)

8.1 Description

Développer un système intelligent de détection et correction automatique des incohérences dans les données, avec validation et apprentissage continu.

8.2 User Stories

1. **US-CORR-01 :** En tant que système, je veux détecter automatiquement les incohérences
2. **US-CORR-02 :** En tant que Data Steward, je veux définir des règles de correction personnalisées
3. **US-CORR-03 :** En tant que système, je veux proposer des corrections avec score de confiance
4. **US-CORR-04 :** En tant que Data Annotator, je veux valider ou rejeter les corrections proposées
5. **US-CORR-05 :** En tant que système, je veux apprendre des validations pour améliorer les futures corrections
6. **US-CORR-06 :** En tant qu'utilisateur, je veux générer un rapport de correction avec traçabilité

8.3 Types d'Incohérences Détectées

— **Incohérences de format :** Ex : date "32/13/2024", téléphone "06-12-34"
— **Incohérences de domaine :** Ex : âge = 250, température = -300°C
— **Incohérences référentielles :** Ex : ville="Paris" + pays="Maroc"
— **Incohérences temporelles :** Ex : date_fin ¡ date_debut

— **Incohérences statistiques :** Ex : valeurs aberrantes, distribution anormale
— **Incohérences sémantiques :** Ex : "email" contient un numéro de téléphone

8.4 Architecture du Système de Correction

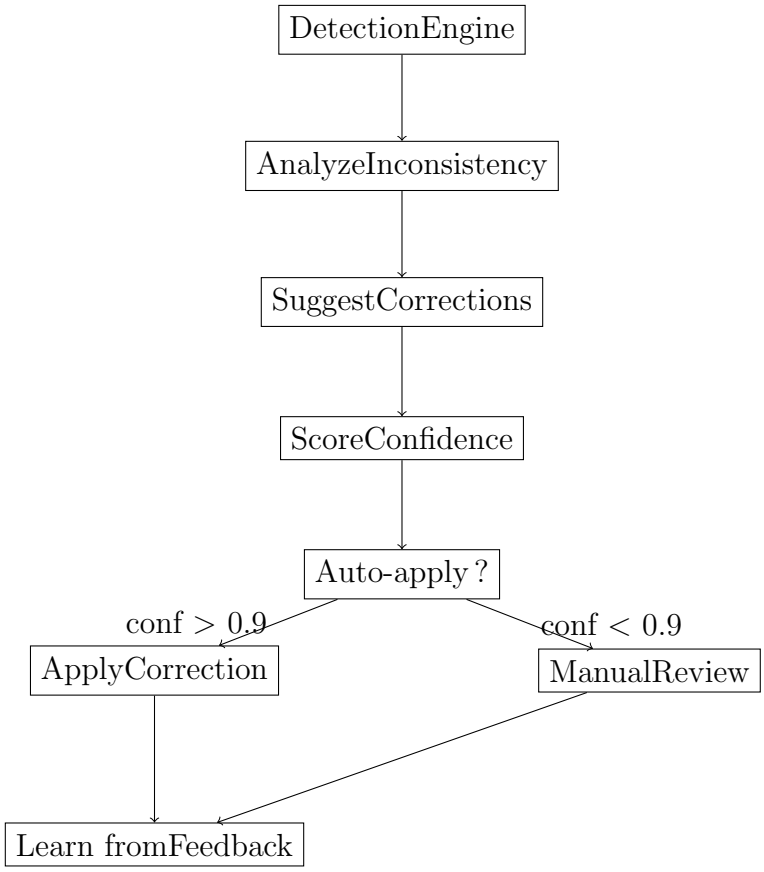

FIGURE 5 – Workflow de Correction Automatique

8.5 Algorithme de Correction avec ML

8.6 Modèles ML pour Correction

Approches recommandées :

— **Sequence-to-Sequence :** Pour correction de texte (T5, BART)
— **Classification :** Pour choisir la meilleure correction parmi candidates
— **Regression :** Pour imputation de valeurs numériques
— **Graph Neural Networks :** Pour corrections référentielles

Modèle HuggingFace recommandé :

```
from transformers import T5ForConditionalGeneration, T5Tokenizer

model = T5ForConditionalGeneration.from_pretrained("t5-base")
tokenizer = T5Tokenizer.from_pretrained("t5-base")
```

Algorithm 6 Correction Automatique Intelligente

Require: Row r, Inconsistencies I, ML_Model M, Rules R
Ensure: Corrected Row, Correction Log

$corrections \leftarrow []$
for each *inconsistency* in I **do**
  $candidates \leftarrow []$
  // Rule-based suggestions
  for each *rule* in $R[inconsistency.type]$ **do**
    if $rule$.matches($inconsistency$) **then**
      $suggestion \leftarrow rule$.apply($r$, $inconsistency$)
      $candidates.append(\{suggestion, score : rule.confidence\})$
    end if
  end for
  // ML-based suggestions
  $features \leftarrow$ ExtractFeatures(r, $inconsistency$)
  $ml\_suggestion, ml\_score \leftarrow M$.predict($features$)
  $candidates.append(\{ml\_suggestion, ml\_score\})$
  // Select best correction
  $best \leftarrow argmax(candidates, key = score)$
  if $best.score \geq 0.9$ **then**
    $r[inconsistency.field] \leftarrow best.suggestion$
    $corrections.append(\{field, old\_value, new\_value, confidence : best.score, auto : True\})$
  else
    $corrections.append(\{field, old\_value, candidates, auto : False, requires\_review : True\})$
  end if
end for
return r, $corrections$

```
# Format: "correct: <incorrect_value> context: <row_context>"
input_text = "correct: 32/13/2024 context: date_naissance"
input_ids = tokenizer(input_text, return_tensors="pt").input_ids
outputs = model.generate(input_ids, max_length=50)
corrected = tokenizer.decode(outputs[0], skip_special_tokens=True)
```

8.7 KPIs

— Taux détection incohérences : $> 95\%$

— Précision corrections auto : $> 90\%$

— Taux auto-correction : $> 70\%$

— Temps traitement : < 5s pour 1000 lignes

— Amélioration continue : $+5\%$ accuracy/mois

8.8 Livrables

1. Code Python système de correction
2. Routes FastAPI detect/correct/validate
3. Modèle ML entraîné pour corrections
4. Base de règles de correction (YAML)
5. Interface de validation pour Data Annotators
6. Rapport de traçabilité des corrections

9 Tâche 7 : Workflow de Validation Humaine

9.1 Description

Développer une interface et un workflow pour la validation humaine des annotations, classifications et corrections par les Data Labelers, Data Annotators et Data Stewards.

9.2 User Stories

1. **US-VALID-01 :** En tant que Data Labeler, je veux annoter les données détectées par Presidio
2. **US-VALID-02 :** En tant que Data Annotator, je veux valider les classifications automatiques
3. **US-VALID-03 :** En tant que Data Steward, je veux approuver les corrections proposées
4. **US-VALID-04 :** En tant qu'utilisateur, je veux voir ma file d'attente de tâches
5. **US-VALID-05 :** En tant que système, je veux tracker les métriques de qualité des annotateurs
6. **US-VALID-06 :** En tant qu'Admin, je veux assigner des tâches selon la charge de travail

9.3 Rôles et Responsabilités

Rôle	Responsabilités	Permissions
Data Labeler	- Annoter les données brutes - Confirmer/corriger détections PII - Labelliser la sensibilité	- Lecture datasets - Création annotations - Pas de modification structure
Data Annotator	- Valider classifications auto - Enrichir métadonnées - Corriger anomalies détectées	- Lecture + annotations - Modification métadonnées - Validation corrections
Data Steward	- Approuver corrections majeures - Définir règles de qualité - Gérer la taxonomie	- Toutes permissions annotation - Modification taxonomie - Approbation finale

9.4 Workflow de Validation

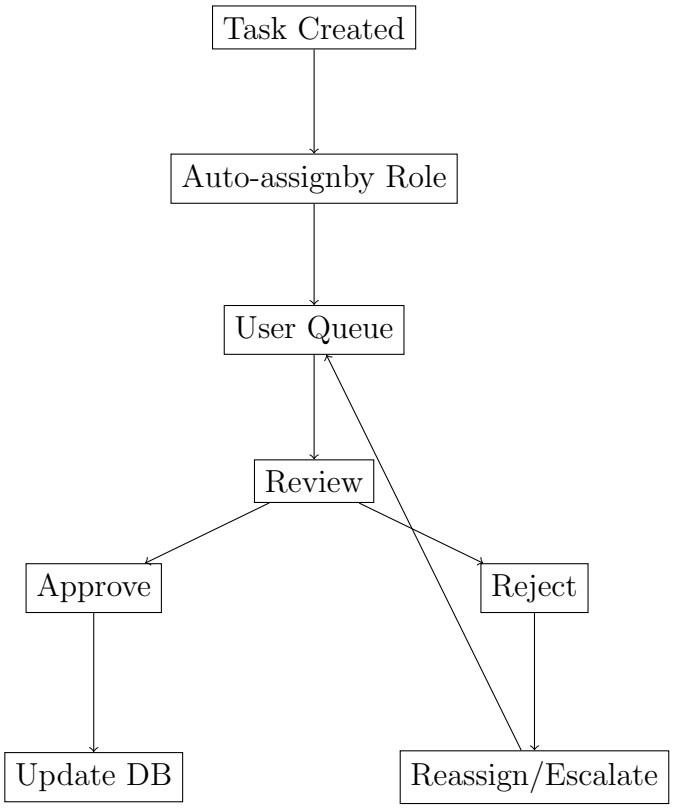

FIGURE 6 – Workflow de Validation Humaine

9.5 Algorithme d'Assignment Intelligent

9.6 Métriques de Qualité des Annotateurs

Formules de calcul :

$$\text{Inter-Annotator Agreement} = \frac{\text{Annotations concordantes}}{\text{Total annotations}} \tag{6}$$

$$\text{Cohen's Kappa} = \frac{p_o - p_e}{1 - p_e} \tag{7}$$

$$\text{Throughput} = \frac{\text{Tâches complétées}}{\text{Temps travaillé}} \tag{8}$$

$$\text{Quality Score} = 0.5 \cdot \text{Accuracy} + 0.3 \cdot \text{Kappa} + 0.2 \cdot \text{Speed} \tag{9}$$

9.7 Documentation de Référence

— Label Studio : https://labelstud.io/ (inspiration interface)

— Cohen's Kappa : https://en.wikipedia.org/wiki/Cohen%27s_kappa

Algorithm 7 Assignment Optimal des Tâches

Require: Task T, Users U, Workload W, Skills S
Ensure: Assigned User

$candidates \leftarrow \text{FilterByRole}(U, T.required\_role)$
$scores \leftarrow []$
for each $user$ in $candidates$ **do**
  $score \leftarrow 0$
  // Workload balance (lower is better)
  $score \leftarrow score - 0.4 \cdot (W[user]/max(W))$
  // Skill match
  $skill\_match \leftarrow S[user] \cap T.required\_skills$
  $score \leftarrow score + 0.3 \cdot (|skill\_match|/|T.required\_skills|)$
  // Historical performance
  $perf \leftarrow \text{GetPerformance}(user, T.type)$
  $score \leftarrow score + 0.3 \cdot perf$
  $scores.append(\{user, score\})$
end for
$best\_user \leftarrow argmax(scores, key = score)$
$W[best\_user] \leftarrow W[best\_user] + 1$
return $best\_user$

9.8 KPIs

— Inter-annotator agreement : > 0.85
— Temps moyen validation : < 30s/item
— Taux complétion tâches : $> 95\%$
— Quality score annotateurs : > 0.80

9.9 Livrables

1. Interface web de validation (FastAPI + HTML/JS)
2. API gestion des tâches et assignments
3. Système de métriques et reporting
4. Dashboard pour Data Stewards
5. Documentation utilisateur par rôle

10 Tâche 8 : Métriques et Normes ISO - Data Quality Scoring

10.1 Description

Implémenter un système complet d'évaluation de la qualité des données selon les normes ISO 8000, ISO 25012 et ISO/IEC 25024.

10.2 User Stories

1. **US-QUAL-01 :** En tant que Data Steward, je veux évaluer la qualité d'un dataset selon ISO 25012
2. **US-QUAL-02 :** En tant que système, je veux calculer automatiquement les 6 dimensions de qualité
3. **US-QUAL-03 :** En tant qu'utilisateur, je veux voir l'évolution de la qualité dans le temps
4. **US-QUAL-04 :** En tant qu'Admin, je veux définir les seuils de conformité ISO
5. **US-QUAL-05 :** En tant que système, je veux générer des recommandations d'amélioration
6. **US-QUAL-06 :** En tant qu'utilisateur, je veux exporter un rapport de qualité ISO

10.3 Dimensions ISO 25012

Dimension	Description	Poids	Formule
Exactitude (Accuracy)	Conformité aux valeurs réelles	25%	$\frac{Valeurs\_correctes}{Total}$
Complétude (Completeness)	Présence de toutes les données requises	20%	$\frac{Non\_null}{Total\_cellules}$
Cohérence (Consistency)	Absence de contradictions	20%	$\frac{Checks\_passed}{Total\_checks}$
Validité (Validity)	Conformité aux règles métier	15%	$\frac{Règles\_respectées}{Total\_règles}$
Unicité (Uniqueness)	Absence de doublons	10%	$1 - \frac{Doublons}{Total}$
Actualité (Timeliness)	Fraîcheur des données	10%	$\frac{Données\_récentes}{Total}$

10.4 Formule Globale de Qualité

$$Q_{global} = \sum_{i=1}^{6} w_i \cdot D_i \tag{10}$$

Où :

— D_i = Score de la dimension $i \in [0, 100]$

— w_i = Poids de la dimension i

— $\sum w_i = 1$

Conformité ISO :

— $Q_{global} \geq 85$: Conforme ISO

— $Q_{global} \in [70, 85[$: Partiellement conforme

— $Q_{global} < 70$: Non conforme

10.5 Algorithme de Calcul Multi-Dimensions

Algorithm 8 Évaluation Qualité ISO 25012

Require: DataFrame D, Reference R, Rules Ru, Weights W
Ensure: Quality Report

// 1. Complétude
$total\_cells \leftarrow |D.rows| \times |D.columns|$
$non\_null \leftarrow$ Count non-null values in D
$C_{completeness} \leftarrow (non\_null/total\_cells) \times 100$
// 2. Unicité
$duplicates \leftarrow$ Count duplicate rows in D
$C_{uniqueness} \leftarrow (1 - duplicates/|D.rows|) \times 100$
// 3. Validité
$rules\_passed \leftarrow 0$
for each $rule$ in Ru **do**
  if D satisfies $rule$ **then**
    $rules\_passed \leftarrow rules\_passed + 1$
  end if
end for
$C_{validity} \leftarrow (rules\_passed/|Ru|) \times 100$
// 4. Cohérence
$inconsistencies \leftarrow$ DetectInconsistencies(D)
$C_{consistency} \leftarrow (1 - |inconsistencies|/total\_cells) \times 100$
// 5. Exactitude (si référence disponible)
if R is not None **then**
  $matching \leftarrow$ Count matching values in D and R
  $C_{accuracy} \leftarrow (matching/total\_cells) \times 100$
else
  $C_{accuracy} \leftarrow 95.0$ // Score par défaut
end if
// 6. Actualité
$recent \leftarrow$ Count records with recent dates
$C_{timeliness} \leftarrow (recent/|D.rows|) \times 100$
// Score global
$Q_{global} \leftarrow \sum_i W_i \times C_i$
$iso\_compliant \leftarrow (Q_{global} \geq 85)$
// Générer recommandations
$recommendations \leftarrow$ GenerateRecommendations(C, threshold=70)
return {scores : C, global : Q_{global}, iso : $iso\_compliant$, recommendations}

10.6 Documentation de Référence

— ISO 8000 Data Quality : https://www.iso.org/standard/50798.html
— ISO/IEC 25012 :2022 : https://www.iso.org/standard/81745.html
— ISO/IEC 25024 :2015 : https://www.iso.org/standard/35747.html
— DAMA-DMBOK : https://www.dama.org/cpages/body-of-knowledge

10.7 KPIs

— Score global ¿ 85/100 pour conformité
— Temps évaluation ¡ 5s pour 10k lignes
— Toutes les 6 dimensions calculées
— Recommandations pertinentes générées
— Historique qualité conservé (évolution)

10.8 Livrables

1. Code Python évaluation ISO
2. Routes FastAPI evaluate/report
3. Dashboard visualisation qualité
4. Export PDF rapport ISO
5. Intégration MongoDB pour historique
6. Tests conformité ISO

11 Tâche 9 : EthiMask - Framework de Masquage Contextuel

11.1 Description

Développer un système intelligent de masquage adaptatif basé sur le contexte d'accès, le rôle utilisateur et la sensibilité des données, utilisant un perceptron multi-facteurs.

11.2 User Stories

1. **US-MASK-01 :** En tant que système, je veux calculer automatiquement le niveau de masquage approprié
2. **US-MASK-02 :** En tant qu'utilisateur, je veux accéder aux données avec le masquage adapté à mon rôle
3. **US-MASK-03 :** En tant qu'Admin, je veux configurer les poids du perceptron
4. **US-MASK-04 :** En tant que Data Steward, je veux auditer les décisions de masquage
5. **US-MASK-05 :** En tant que système, je veux appliquer le chiffrement homomorphique pour niveau 1
6. **US-MASK-06 :** En tant que système, je veux apprendre des patterns d'accès pour améliorer les décisions

11.3 Architecture du Perceptron V0.1

Le score de confiance normalisé $T'(u, p, s, h) \in [0, 1]$ est calculé selon :

$$T'(u,p,s,h) = \sigma\left(w_r R(u) + w_p P(p) + w_s S(s) + w_{hc} H_c(u) + w_{hf} H_f(u) + w_{hv} H_v(u) + b_T\right) \tag{11}$$

Composantes :

— $R(u)$: Score du rôle utilisateur $\in [0, 1]$

— $P(p)$: Légitimité de la finalité d'accès $\in [0, 1]$

— $S(s)$: Sensibilité de l'attribut $\in [0, 1]$

— $H_c(u)$: Conformité historique $\in [0, 1]$

— $H_f(u)$: Fréquence d'accès normalisée $\in [0, 1]$

— $H_v(u)$: Pénalité violations $\in [0, 1]$

— w_* : Poids configurables (somme = 1)

— b_T : Biais organisationnel

— σ : Fonction sigmoïde $\sigma(x) = \frac{1}{1+e^{-x}}$

11.4 Mapping Score → Niveau de Masquage

Score T'	Niveau	Action
$[0.85, 1.0]$	**Niveau 0 : Full Access**	Accès complet aux données non masquées
$[0.65, 0.85[$	**Niveau 1 : Anonymization**	Remplacement par identifiants chiffrés (HE)
$[0.45, 0.65[$	**Niveau 2 : Généralisation**	Agrégation (âge → tranche, ville → région)
$[0.25, 0.45[$	**Niveau 3 : Privacy Différentielle**	Ajout de bruit contrôlé (ϵ-DP)
$[0, 0.25[$	**Niveau 4 : Suppression**	Valeur remplacée par NULL ou "***"

11.5 Architecture Détaillée

11.6 Chiffrement Homomorphique (Niveau 1)

Pour le niveau Anonymization, utilisation de **TenSEAL** ou **Concrete-ML** :

Avantages :

- Calculs arithmétiques sur données chiffrées (addition, multiplication)
- Entraînement de modèles ML sans déchiffrement
- Conservation des propriétés statistiques
- Conformité RGPD renforcée

Librairies Python :

- TenSEAL : https://github.com/OpenMined/TenSEAL
- Concrete-ML : https://docs.zama.ai/concrete-ml
- PySyft : https://github.com/OpenMined/PySyft

Exemple d'implémentation TenSEAL :

```
import tenseal as ts

# Contexte homomorphique
context = ts.context(
    ts.SCHEME_TYPE.CKKS,
    poly_modulus_degree=8192,
    coeff_mod_bit_sizes=[60, 40, 40, 60]
)
context.global_scale = 2**40

# Chiffrement
value = 42.5
encrypted = ts.ckks_vector(context, [value])

# Calculs sur données chiffrées
result = encrypted * 2 + 10  # = 95.0

# Déchiffrement (uniquement autorisés)
decrypted = result.decrypt()[0]
```

11.7 Privacy Différentielle (Niveau 3)

Pour le niveau 3, application du mécanisme de Laplace :

$$\tilde{x} = x + \text{Lap}\left(\frac{\Delta f}{\epsilon}\right) \tag{12}$$

Où :

— x : Valeur originale

— $\tilde{x}$: Valeur bruitée

— Δf : Sensibilité de la fonction

— ϵ : Budget de privacy (typiquement $\epsilon \in [0.1, 1.0]$)

— Lap(λ) : Distribution de Laplace de paramètre λ

Librairie recommandée :

— Google DP : https://github.com/google/differential-privacy

— Opacus (PyTorch) : https://opacus.ai/

— Diffprivlib (IBM) : https://github.com/IBM/differential-privacy-library

11.8 Algorithme de Masquage Adaptatif

11.9 Évolution vers V1 (Réseau de Neurones)

Améliorations prévues :

1. Transformation de chaque facteur (R, P, S, H) en sous-réseau fully-connected
2. Architecture suggérée pour chaque facteur : Input → [8-16 neurones] → ReLU → Output
3. Réseau H(u) : 3 inputs (Hc, Hf, Hv) → 8 neurones cachés → ReLU → 1 output
4. Fonction de perte composite :

$$L = \alpha \cdot L_{privacy} + (1 - \alpha) \cdot L_{utility} \tag{13}$$

5. Entraînement par rétro-propagation avec dataset d'accès validés
6. Régularisation L2 pour éviter l'overfitting

Framework recommandé : PyTorch ou TensorFlow

11.10 Documentation de Référence

— Differential Privacy Book : https://www.cis.upenn.edu/~aaroth/Papers/privacybook.pdf

— Homomorphic Encryption : https://en.wikipedia.org/wiki/Homomorphic_encryption

— GDPR Anonymization Guide : https://www.cnil.fr/sites/default/files/atoms/files/cnil_guide_securite_des_donnees_personnelles-2023.pdf

— TenSEAL Tutorial : https://github.com/OpenMined/TenSEAL/tree/main/tutorials

Algorithm 9 Masquage Contextuel EthiMask

Require: User u, Purpose p, Attribute a, History H, Weights W
Ensure: Masked Value

// Calculer les facteurs
$R \leftarrow \text{ComputeRoleScore}(u.role)$
$P \leftarrow \text{ComputePurposeScore}(p, a.allowed\_purposes)$
$S \leftarrow \text{GetSensitivity}(a)$
$H_c \leftarrow \text{ComputeCompliance}(u, H)$
$H_f \leftarrow \text{ComputeFrequency}(u, a, H)$
$H_v \leftarrow \text{ComputeViolationPenalty}(u, H)$
// Calcul score de confiance
$linear \leftarrow W_r \cdot R + W_p \cdot P + W_s \cdot S$
$linear \leftarrow linear + W_{hc} \cdot H_c + W_{hf} \cdot H_f + W_{hv} \cdot H_v$
$linear \leftarrow linear + bias$
$T' \leftarrow \sigma(linear)$
// Déterminer niveau de masquage
if $T' \geq 0.85$ **then**
    $level \leftarrow 0$ // Full access
else if $T' \geq 0.65$ **then**
    $level \leftarrow 1$ // HE encryption
else if $T' \geq 0.45$ **then**
    $level \leftarrow 2$ // Generalization
else if $T' \geq 0.25$ **then**
    $level \leftarrow 3$ // Differential privacy
else
    $level \leftarrow 4$ // Suppression
end if
// Appliquer masquage
$masked\_value \leftarrow \text{ApplyMasking}(a.value, level)$
// Logger la décision
LogAccess(u, a, $level$, T', timestamp)
return $masked\_value$

11.11 KPIs

— Précision décisions : > 92%
— Temps calcul masquage : ¡ 50ms
— Taux de privacy breaches : = 0
— Satisfaction utilisateurs : > 80%
— Amélioration continue : +3% accuracy/trimestre

11.12 Livrables

1. Code Python EthiMask V0.1 et V1
2. Routes FastAPI pour masquage contextuel
3. Modèle PyTorch entraîné (V1)
4. Interface configuration poids
5. Dashboard audit décisions
6. Benchmark performance masquage
7. Documentation technique complète

12 Intégration Apache Atlas et Apache Ranger

12.1 Description

Intégrer les outils de gouvernance Big Data Apache Atlas (métadonnées et lignage) et Apache Ranger (contrôle d'accès) avec l'architecture microservices.

12.2 Architecture d'Intégration

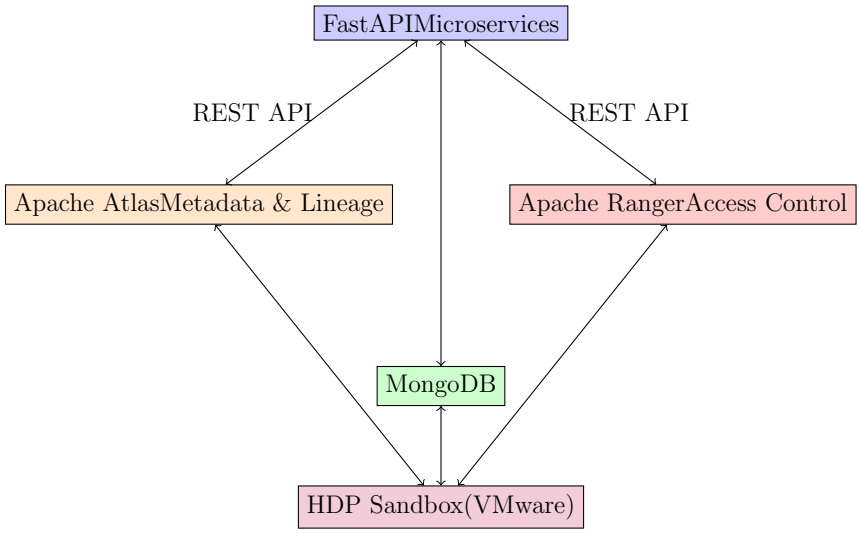

FIGURE 7 – Architecture d'Intégration avec HDP Sandbox

12.3 Configuration Apache Atlas

Cas d'usage :

- **Catalogage :** Enregistrer tous les datasets et leurs métadonnées
- **Lignage :** Tracer les transformations de données (pipeline Airflow)
- **Classification :** Appliquer tags PII/SPI automatiquement
- **Recherche :** Permettre la découverte de données par les Data Stewards
- **Audit :** Logger toutes les modifications de métadonnées

Entités Atlas à créer :

```
- data_source (MongoDB collections)
- dataset (fichiers CSV uploadés)
- data_attribute (colonnes/champs)
- pipeline (DAGs Airflow)
- process (transformations)
```

API Atlas REST :

- POST /api/atlas/v2/entity - Créer entité
- GET /api/atlas/v2/entity/guid/{guid} - Lire entité
- PUT /api/atlas/v2/entity - Mettre à jour
- POST /api/atlas/v2/search/basic - Rechercher
- GET /api/atlas/v2/lineage/{guid} - Obtenir lignage

12.4 Configuration Apache Ranger

Cas d'usage :

— **Policies par rôle :** Admin, Data Steward, Annotator, Labeler

— **Policies par tag :** PII → masquage, SPI → accès restreint

— **Masking dynamique :** Selon le niveau de sensibilité

— **Row-level filtering :** Filtrer les données sensibles

— **Audit trail :** Logger tous les accès

Types de policies :

1. **Access policies :** Allow/Deny sur ressources
2. **Masking policies :** Masquer colonnes (MASK, HASH, NULLIFY)
3. **Row filter policies :** SQL WHERE clause dynamique

Exemple de policy JSON :

```
{
  "policyType": "1",
  "name": "pii-masking-policy",
  "isEnabled": true,
  "resources": {
    "database": {"values": ["sensitive_data_db"]},
    "table": {"values": ["users"]},
    "column": {"values": ["cin", "phone", "email"]}
  },
  "policyItems": [
    {
      "accesses": [{"type": "select", "isAllowed": true}],
      "users": ["data_labeler"],
      "dataMaskInfo": {
        "dataMaskType": "MASK",
        "conditionExpr": "",
        "valueExpr": ""
      }
    }
  ]
}
```

12.5 Workflow d'Intégration

Scénario : Upload et Classification d'un Dataset

1. Utilisateur uploade CSV via FastAPI
2. FastAPI → Profiling et Cleaning
3. FastAPI → Presidio détection PII
4. FastAPI → Classification sensibilité
5. **FastAPI → Atlas :** Enregistrer dataset + métadonnées + tags
6. **FastAPI → Ranger :** Créer policies selon classifications

7. MongoDB : Stocker résultats
8. Airflow : Logger le workflow complet
9. **Atlas :** Créer lignage complet du pipeline

Scénario : Accès à une Donnée Sensible

1. Utilisateur requête donnée via FastAPI
2. FastAPI : Valider JWT → Extraire rôle
3. **FastAPI → Ranger :** Vérifier policy d'accès
4. **Ranger → FastAPI :** Retourner décision (allow + masking level)
5. FastAPI → EthiMask : Calculer masquage contextuel
6. FastAPI : Appliquer masquage
7. FastAPI → MongoDB : Logger accès
8. **FastAPI → Atlas :** Logger événement d'accès
9. FastAPI → Utilisateur : Retourner données (masquées)

12.6 Librairies Python

— **apache-atlas :** Client Python pour Atlas

```
pip install apache-atlas
```

— **requests :** Pour appels REST API Ranger
— **pyatlasclient :** Alternative pour Atlas https://github.com/jpoullet2000/pyatlasclient

12.7 Documentation de Référence

— Apache Atlas : https://atlas.apache.org/
— Atlas REST API : https://atlas.apache.org/api/v2/index.html
— Apache Ranger : https://ranger.apache.org/
— Ranger REST API : https://ranger.apache.org/apidocs/index.html
— HDP Sandbox : https://www.cloudera.com/downloads/hortonworks-sandbox.html

12.8 KPIs

— Temps réponse Atlas API : ¡ 200ms
— Temps réponse Ranger API : ¡ 150ms
— Taux de synchronisation : 100%
— Couverture lignage : 100% des pipelines
— Audit trail complet : 100% des accès

12.9 Livrables

1. Code intégration Atlas (Python)
2. Code intégration Ranger (Python)
3. Scripts création entités Atlas
4. Policies Ranger (JSON)
5. Documentation configuration HDP Sandbox
6. Dashboard visualisation lignage

12.10 Création du Dataset MAPA (Moroccan Personal Information)

Contenu recommandé (5000+ enregistrements) :

- **Identité :** CIN, nom, prénom, date naissance, lieu naissance
- **Contact :** Téléphone (mobile/fixe), email, adresse complète
- **Finance :** IBAN MA, RIB, numéro compte
- **Santé :** CNSS, groupe sanguin, allergies
- **Professionnel :** Employeur, salaire, fonction
- **Administratif :** Passeport, permis, carte RAMED

Génération avec Faker :

```
from faker import Faker
import pandas as pd

fake = Faker('fr_FR')
Faker.seed(42)

def generate_cin():
    letters = fake.random_uppercase_letter() +
              fake.random_uppercase_letter()
    numbers = fake.random_number(digits=6, fix_len=True)
    return f"{letters}{numbers}"

def generate_phone_ma():
    prefix = fake.random_element(['06', '07', '05'])
    number = fake.random_number(digits=8, fix_len=True)
    return f"{prefix}{number}"

def generate_iban_ma():
    return f"MA{fake.random_number(digits=24, fix_len=True)}"

# Générer 5000 enregistrements
data = []
for _ in range(5000):
    record = {
        'cin': generate_cin(),
        'nom': fake.last_name(),
        'prenom': fake.first_name(),
        'telephone': generate_phone_ma(),
        'email': fake.email(),
        'iban': generate_iban_ma(),
        'cnss': fake.random_number(digits=10, fix_len=True),
        # ... autres champs
    }
    data.append(record)
```

```
df = pd.DataFrame(data)
df.to_csv('mapa_dataset.csv', index=False)
```

12.11 Métriques d'Évaluation

Pour chaque tâche :

— Accuracy, Precision, Recall, F1-Score
— Temps de traitement (latency, throughput)
— Utilisation ressources (CPU, RAM)
— Taux d'erreur et types d'erreurs

Évaluation globale du système :

— End-to-end processing time
— Data quality score évolution
— Privacy preservation effectiveness
— User satisfaction (Data Stewards, Annotators)

12.12 Livrables

1. Dataset MAPA synthétique (CSV)
2. Scripts génération données
3. Annotations gold standard (500+ samples)
4. Rapports d'évaluation par tâche
5. Benchmark comparatif

13 Planning et Organisation

13.1 Répartition des Tâches

Phase 1 - Infrastructure (Semaines 1-3) :

— Setup environnement (Python, FastAPI, MongoDB, Airflow)
— Configuration HDP Sandbox (VMware)
— Architecture microservices de base
— Tâche 1 : Authentification et rôles

Phase 2 - Détection et Classification (Semaines 4-7) :

— Tâche 2 : Taxonomie fine-grained PII/SPI
— Tâche 3 : Personnalisation Presidio
— Tâche 5 : Classification fine-grained
— Intégration Apache Atlas

Phase 3 - Qualité et Correction (Semaines 8-11) :

— Tâche 4 : Data Cleaning pipeline
— Tâche 6 : Correction automatique incohérences
— Tâche 8 : Métriques ISO
— Intégration Apache Ranger

Phase 4 - Privacy et Validation (Semaines 12-15) :

— Tâche 9 : EthiMask framework
— Tâche 7 : Workflow validation humaine
— Orchestration Airflow complète
— Tests d'intégration

Phase 5 - Finalisation (Semaines 16-18) :

— Tests end-to-end
— Documentation complète
— Évaluation sur datasets
— Rapport final et présentation

13.2 Diagramme de Gantt

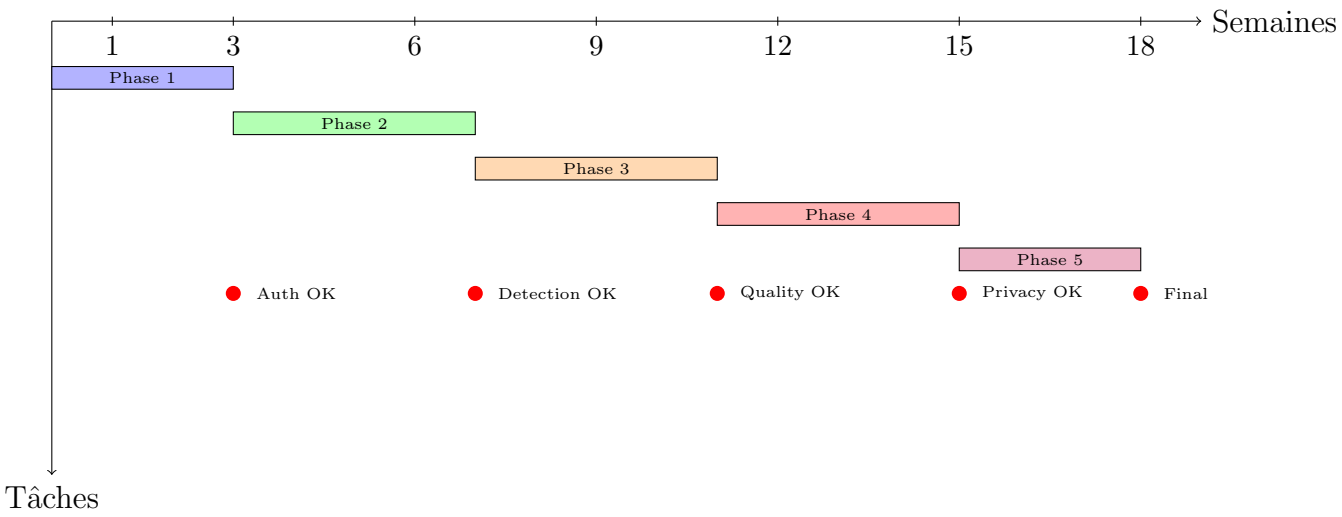

FIGURE 8 – Planning Projet (18 semaines)

13.3 Critères de Succès

Critères techniques :

- Toutes les 9 tâches complétées et fonctionnelles
- Architecture microservices opérationnelle
- Intégration Airflow + Atlas + Ranger réussie
- Tests unitaires coverage >80%
- Performance conforme aux KPIs définis

Critères qualité :

- Documentation technique complète
- Code commenté et structuré
- Rapports d'évaluation avec métriques
- Conformité RGPD et Loi 09-08 vérifiée
- Démonstration end-to-end réussie

13.4 Livrables Finaux

1. **Code source complet :**
   - Repository Git structuré
   - README avec instructions
   - Requirements.txt
   - Scripts de déploiement
2. **Documentation :**
   - Rapport technique (ce document)

- Documentation API (Swagger)
- Guide d'installation
- Guide utilisateur par rôle
- Documentation architecture

3. **Datasets et Modèles :**
   - Dataset MAPA synthétique
   - Modèles ML entraînés
   - Collections MongoDB exportées
   - Policies Atlas/Ranger
4. **Tests et Évaluation :**
   - Suite de tests (pytest)
   - Rapports de performance
   - Benchmark comparatif
   - Résultats évaluation ISO

14 Ressources et Références

14.1 Documentation Technique

Frameworks et Bibliothèques :

— FastAPI : https://fastapi.tiangolo.com/
— MongoDB : https://www.mongodb.com/docs/
— Beanie ODM : https://beanie-odm.dev/
— Apache Airflow : https://airflow.apache.org/docs/
— Presidio : https://microsoft.github.io/presidio/
— spaCy : https://spacy.io/
— Pandas : https://pandas.pydata.org/docs/
— Scikit-learn : https://scikit-learn.org/
— HuggingFace : https://huggingface.co/docs
— PyTorch : https://pytorch.org/docs/

Gouvernance et Privacy :

— Apache Atlas : https://atlas.apache.org/
— Apache Ranger : https://ranger.apache.org/
— TenSEAL : https://github.com/OpenMined/TenSEAL
— Opacus : https://opacus.ai/

14.2 Réglementations

RGPD et Privacy :

— RGPD officiel : https://gdpr-info.eu/
— Guide CNIL sécurité : https://www.cnil.fr/sites/default/files/atoms/files/cnil_guide_securite_des_donnees_personnelles-2023.pdf
— Loi 09-08 Maroc : https://www.dgssi.gov.ma/fr/loi-09-08-relative-la-protection-des-p
— NIST Privacy Framework : https://www.nist.gov/privacy-framework

Normes ISO :

— ISO 8000 Data Quality : https://www.iso.org/standard/50798.html
— ISO/IEC 25012 :2022 : https ://www.iso.org/standard/81745.html
— ISO/IEC 25024 :2015 : https://www.iso.org/standard/35747.html
— ISO/IEC 29100 :2024 https://www.iso.org/standard/85938.html

14.3 Tutoriels et Guides

FastAPI + MongoDB :

— FastAPI + Beanie Tutorial : https://github.com/BeanieODM/beanie
— Async Python : https://realpython.com/async-io-python/
— JWT Authentication : https://testdriven.io/blog/fastapi-jwt-auth/

Apache Airflow :

— Airflow Fundamentals : https://airflow.apache.org/docs/apache-airflow/stable/tutorial/fundamentals.html

— Building DAGs : https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html

— Custom Operators : https://airflow.apache.org/docs/apache-airflow/stable/howto/custom-operator.html

Machine Learning NLP :

— HuggingFace Course : https://huggingface.co/learn/nlp-course/chapter1/1

— Fine-tuning BERT : https://huggingface.co/docs/transformers/training

— spaCy Training : https://spacy.io/usage/training

14.4 Repositories GitHub Inspirants

— **Presidio Samples :** https://github.com/microsoft/presidio/tree/main/docs/samples

— **FastAPI Best Practices :** https://github.com/zhanymkanov/fastapi-best-practices

— **Differential Privacy :** https://github.com/google/differential-privacy

15 Annexes

15.1 Annexe A : Structure Complète du Projet

```
projet_federateur/
 README.md
 requirements.txt
 .env
 .gitignore
 setup.py

 app/
    main.py
    config.py
    database.py

    models/
       __init__.py
       user.py
       taxonomy.py
       dataset.py
       quality_metrics.py
       annotation.py
       audit_log.py

    schemas/
       __init__.py
       user.py
       taxonomy.py
       presidio.py
       cleaning.py
       quality.py

    routers/
       __init__.py
       auth.py
       taxonomy.py
       presidio.py
       data_cleaning.py
       classification.py
       correction.py
       quality.py
       ethimask.py
       annotation.py

    services/
       __init__.py
       auth_service.py
       taxonomy_service.py
```

```
      quality_service.py
      classification_service.py
      correction_service.py
      ethimask_service.py
      atlas_service.py
      ranger_service.py

   middleware/
      __init__.py
      auth.py
      logging.py

   utils/
       __init__.py
       validators.py
       helpers.py

presidio_custom/
   __init__.py
   recognizers/
      cin_recognizer.py
      phone_ma_recognizer.py
      iban_ma_recognizer.py
      cnss_recognizer.py
   analyzer_engine.py
   anonymizer_engine.py

data_cleaning/
   __init__.py
   pipeline.py
   validators.py
   profiler.py

ml_models/
   __init__.py
   classification/
      train.py
      predict.py
      model.py
   correction/
      correction_model.py
   ethimask/
       perceptron.py
       neural_network.py

airflow/
   dags/
      data_processing_pipeline.py
```

```
      quality_assessment_dag.py
      ethimask_application_dag.py
   config/
      pipeline_config.yaml
   plugins/
       custom_operators.py

atlas_integration/
   __init__.py
   entities.py
   lineage.py
   sync.py

ranger_integration/
   __init__.py
   policies.py
   access_control.py

tests/
   __init__.py
   test_auth.py
   test_taxonomy.py
   test_presidio.py
   test_cleaning.py
   test_classification.py
   test_correction.py
   test_quality.py
   test_ethimask.py
   test_integration.py

datasets/
   mapa_synthetic/
   test_data/
   annotations/

docs/
   architecture.md
   api_documentation.md
   user_guides/
   technical_specs/

scripts/
    populate_taxonomy.py
    generate_mapa_dataset.py
    setup_atlas.py
    setup_ranger.py
    deploy.sh
```

15.2 Annexe B : Configuration Airflow YAML

Exemple de configuration complète pour un DAG :

```yaml
dag_config:
  dag_id: "complete_data_processing"
  description: "Pipeline complet de traitement des données sensibles"
  schedule_interval: "0 2 * * *"  # Daily at 2AM
  start_date: "2024-01-01"
  catchup: false
  max_active_runs: 1

  default_args:
    owner: "data_governance_team"
    depends_on_past: false
    email_on_failure: true
    email_on_retry: false
    retries: 3
    retry_delay: 300  # 5 minutes

services:
  base_url: "http://localhost:8000/api/v1"
  timeout: 300
  verify_ssl: true

tasks:
  - task_id: "upload_dataset"
    service: "cleaning"
    endpoint: "/upload"
    method: "POST"
    dependencies: []
    params:
      file_type: "csv"
      validate: true
    retry_on_http_codes: [500, 502, 503]

  - task_id: "profile_data"
    service: "cleaning"
    endpoint: "/profile/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
    method: "GET"
    dependencies: ["upload_dataset"]

  - task_id: "clean_data"
    service: "cleaning"
    endpoint: "/clean/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
    method: "POST"
    dependencies: ["profile_data"]
    params:
      remove_duplicates: true
```

```
    handle_missing: "mean"
    remove_outliers: true
    outlier_method: "iqr"
    normalize: false

- task_id: "detect_pii"
  service: "presidio"
  endpoint: "/analyze"
  method: "POST"
  dependencies: ["clean_data"]
  params:
    entities: ["CIN_MAROC", "PHONE_MA", "IBAN_MA", "CNSS_MA"]
    score_threshold: 0.5

- task_id: "classify_sensitivity"
  service: "classification"
  endpoint: "/classify/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
  method: "POST"
  dependencies: ["detect_pii"]

- task_id: "quality_assessment"
  service: "quality"
  endpoint: "/evaluate/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
  method: "POST"
  dependencies: ["clean_data"]

- task_id: "detect_inconsistencies"
  service: "correction"
  endpoint: "/detect/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
  method: "POST"
  dependencies: ["quality_assessment"]

- task_id: "apply_corrections"
  service: "correction"
  endpoint: "/correct/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
  method: "POST"
  dependencies: ["detect_inconsistencies"]
  params:
    auto_apply_threshold: 0.9

- task_id: "apply_ethimask"
  service: "ethimask"
  endpoint: "/mask/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
  method: "POST"
  dependencies: ["classify_sensitivity", "apply_corrections"]

- task_id: "sync_atlas"
  service: "atlas"
```

```
    endpoint: "/sync/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
    method: "POST"
    dependencies: ["classify_sensitivity"]

  - task_id: "create_ranger_policies"
    service: "ranger"
    endpoint: "/create_policies"
    method: "POST"
    dependencies: ["classify_sensitivity"]
    params:
      dataset_id: "{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"

  - task_id: "final_quality_report"
    service: "quality"
    endpoint: "/report/{{ ti.xcom_pull(task_ids='upload_dataset')['dataset_id'] }}"
    method: "GET"
    dependencies: ["apply_ethimask", "sync_atlas", "create_ranger_policies"]

notifications:
  on_success:
    - type: "email"
      recipients: ["data-stewards@example.com"]
      subject: "Pipeline Success: {{ dag_run.dag_id }}"

  on_failure:
    - type: "email"
      recipients: ["admin@example.com"]
      subject: "Pipeline Failed: {{ dag_run.dag_id }}"
    - type: "slack"
      webhook_url: "${SLACK_WEBHOOK_URL}"

monitoring:
  sla: 3600  # 1 hour
  alert_on_sla_miss: true
  track_performance: true
```

15.3 Annexe C : Glossaire

PII Personally Identifiable Information - Données permettant d'identifier directement ou indirectement une personne

SPI Sensitive Personal Information - Données personnelles sensibles (santé, origine, religion, etc.)

RGPD Règlement Général sur la Protection des Données (GDPR en anglais)

ISO 25012 Norme internationale pour le modèle de qualité des données

HE Homomorphic Encryption - Chiffrement permettant des calculs sur données chiffrées

DP Differential Privacy - Technique de protection de la vie privée par ajout de bruit

DAG Directed Acyclic Graph - Graphe utilisé dans Airflow pour définir les workflows

ODM Object-Document Mapper - Interface entre code Python et MongoDB

JWT JSON Web Token - Standard pour les tokens d'authentification

REST Representational State Transfer - Architecture pour APIs web

NLP Natural Language Processing - Traitement automatique du langage naturel

ML Machine Learning - Apprentissage automatique

ETL Extract, Transform, Load - Processus de traitement de données

CRUD Create, Read, Update, Delete - Opérations de base sur données

16 Conclusion

Ce projet fédérateur propose un système complet et moderne de gouvernance des données sensibles, intégrant les meilleures pratiques en matière de protection de la vie privée, de qualité des données et de conformité réglementaire.

Points clés du projet :

- **Architecture extensible :** Microservices + Airflow permettant d'ajouter de nouveaux modules sans modification du core
- **Conformité réglementaire :** RGPD et Loi 09-08 marocaine intégrés dès la conception
- **Intelligence artificielle :** ML/NLP pour détection, classification et correction automatiques
- **Privacy by design :** EthiMask avec masquage contextuel et chiffrement homomorphique
- **Gouvernance Big Data :** Intégration Atlas/Ranger pour traçabilité et contrôle d'accès
- **Qualité ISO :** Évaluation selon normes ISO 8000 et ISO 25012
- **Validation humaine :** Workflow complet pour Data Labelers, Annotators et Stewards

Impact attendu :

- Réduction des risques de fuite de données sensibles
- Amélioration de la qualité des données ($> 85/100$)
- Automatisation des tâches de gouvernance (gain 60% de temps)
- Conformité démontrable aux audits RGPD
- Base solide pour extension future (nouveaux modules, nouvelles réglementations)