PortNet HARPON

Cahier des Charges & Spécifications Techniques

Équipe Portshield

30 janvier 2026

OBJECTIF

Vision : PortNet HARPON est un DSS (Decision Support System) qui transforme un dossier de conteneur en décision actionnable. Il fournit un score de risque, des preuves visuelles et une recommandation avant même l'ouverture physique du conteneur.

1 OBJECTIFS & UTILISATEURS

1.1 La Promesse

Sécuriser davantage sans ralentir le flux portuaire grâce à un contrôle ciblé (*Smart Targeting*) plutôt qu'un contrôle massif aléatoire.

1.2 Utilisateurs Cibles

- **Agent Douanier (Opérationnel) :** Besoin critique : *"Dites-moi quoi contrôler en priorité et pourquoi, en moins de 10 secondes."*
- **Superviseur (Pilotage - Optionnel) :** Besoin : *"Vue globale des tendances, des filières suspectes et des points chauds."*

2 ARCHITECTURE DE DONNÉES (I/O)

2.1 Entrées (Inputs)

— **Dossier Conteneur (PDF/Images) :** Facture Commerciale, Packing List, Bill of Lading (B/L), Certificats d'origine.

— **Données Déclaratives :** Pays d'origine, HS Code, Valeur déclarée, Poids, Importateur, Transitaire.

— **Contrainte MVP (dégradation élégante) :**

  — Si métadonnées manquantes : le score est calculé sur les éléments disponibles et le système indique *"données incomplètes"*.

  — Si document illisible (scan) : extraction *mockée/fixture* pour la démo, et journalisation *EXTRACTION_PARTIAL*.

2.2 Sorties Décisionnelles (Outputs)

Pour chaque dossier analysé (Case), le système génère :

1. **Global Risk Score (0–100)** : avec statut visuel (Vert / Orange / Rouge).
2. **Risk Breakdown** : décomposition (Documentaire / Anomalies / Réseau).
3. **Top Reasons** : 3 à 7 explications en langage naturel.
4. **Evidence Pack** : preuves structurées (extraits de documents, graphiques Benford, incohérences surlignées).
5. **Recommended Action** : Libérer / Demander Pièces / Inspecter physiquement + checklist.
6. **Audit Log** : traçabilité des actions (*analyse, décision, feedback, export*).

3 PÉRIMÈTRE FONCTIONNEL

3.1 Fonctionnalités MVP

- ☐ **Création d'un Case :** Upload de dossiers PDF + saisie métadonnées de base.
- ☐ **Analyse Automatique :** Pipeline générant score, raisons, preuves et action recommandée.
- ☐ **Dashboard Agent :** Liste des dossiers triée par risque décroissant.
- ☐ **Page Détail :** Affichage ergonomique (Score → Raisons → Preuves → Action).
- ☐ **Traçabilité :** Journalisation simple des événements et décisions.
- ☐ **Vue Réseau (Louvain) :** Visualisation graphique des liens entre acteurs (importateur, transitaire, fournisseur).
- ☐ **Export PDF :** Génération d'un rapport officiel "Evidence Pack" pour archivage.
- ☐ **Feedback Loop :** Bouton pour valider ou infirmer la fraude (confirmé / faux positif).

3.2 Fonctionnalités Avancées (Bonus / Si Temps)

- ☐ **Template Reuse Detector :** Détection de gabarits de fraude identiques entre acteurs différents.

4 SPÉCIFICATIONS TECHNIQUES MINIMALES

4.1 Contrat d'Interface (API Endpoints)

L'architecture repose sur une API REST (Python/FastAPI) consommée par un client Web.

`POST /cases`
: Création d'un dossier, upload des fichiers et métadonnées.

`POST /cases/{id}/analyze`
: Déclenchement (asynchrone si possible) des moteurs d'analyse.

`GET /cases`
: Liste légère pour le Dashboard (ID, Statut, Score, Action, Timestamp).

`GET /cases/{id}`
: Détail complet pour l'écran de décision (Breakdown, Reasons, Evidence Pack, Audit).

Critère de performance : l'appel `GET /cases/{id}` doit fournir toutes les données nécessaires à l'affichage en une seule requête et viser < 200ms (sur dataset de démo).

4.2 Critères d'Acceptation (Validation avant développement)

- Le front peut afficher la page Détail avec **un seul appel** `GET /cases/{id}`.
- Chaque alerte affichée dans l'UI correspond à au moins un enregistrement **`signals`** avec une **preuve** (`evidence_json`).
- Le score global est explicable par la somme des **contributions** (`signals.score_contribution`).
- Les décisions agent (action + feedback) sont tracées dans **`audit_logs`**.

4.3 Modèle de Données (Schéma Relationnel Avancé)

L'architecture repose sur 9 tables interconnectées, conçues pour supporter l'analyse de graphes et la traçabilité forensique.

- **1. Cœur Décisionnel**
  - **`cases`** : entité centrale (`id, reference_id, status, global_score, risk_level, recommended_action, risk_breakdown_json`).
  - **`audit_logs`** : journal d'audit (`actor, event_type, payload_json, created_at`).
- **2. Flux Documentaire & Extraction (No-OCR)**
  - **`documents`** : métadonnées techniques (`filename, storage_path, file_hash, mime_type, size_bytes, page_count`).
  - **`extractions`** : cache des données extraites (`raw_text, key_value_json`) pour éviter le re-processing.
  - **`document_similarities`** : liens pour la détection de templates (`similarity_score, method`).
- **3. Moteur IA & Preuves (Evidence Pack)**
  - **`features`** : vecteurs d'entrée pour Isolation Forest (`feature_name, feature_value`).
  - **`signals`** : résultats des algorithmes (`algo_source, signal_type, severity, score_contributio evidence_json`).
- **4. Graphe & Réseau (Module Louvain)**
  - **`entities`** : acteurs normalisés (`name, normalized_name, entity_type, risk_score_history`).
  - **`case_entity_links`** : arêtes du graphe (`role`).

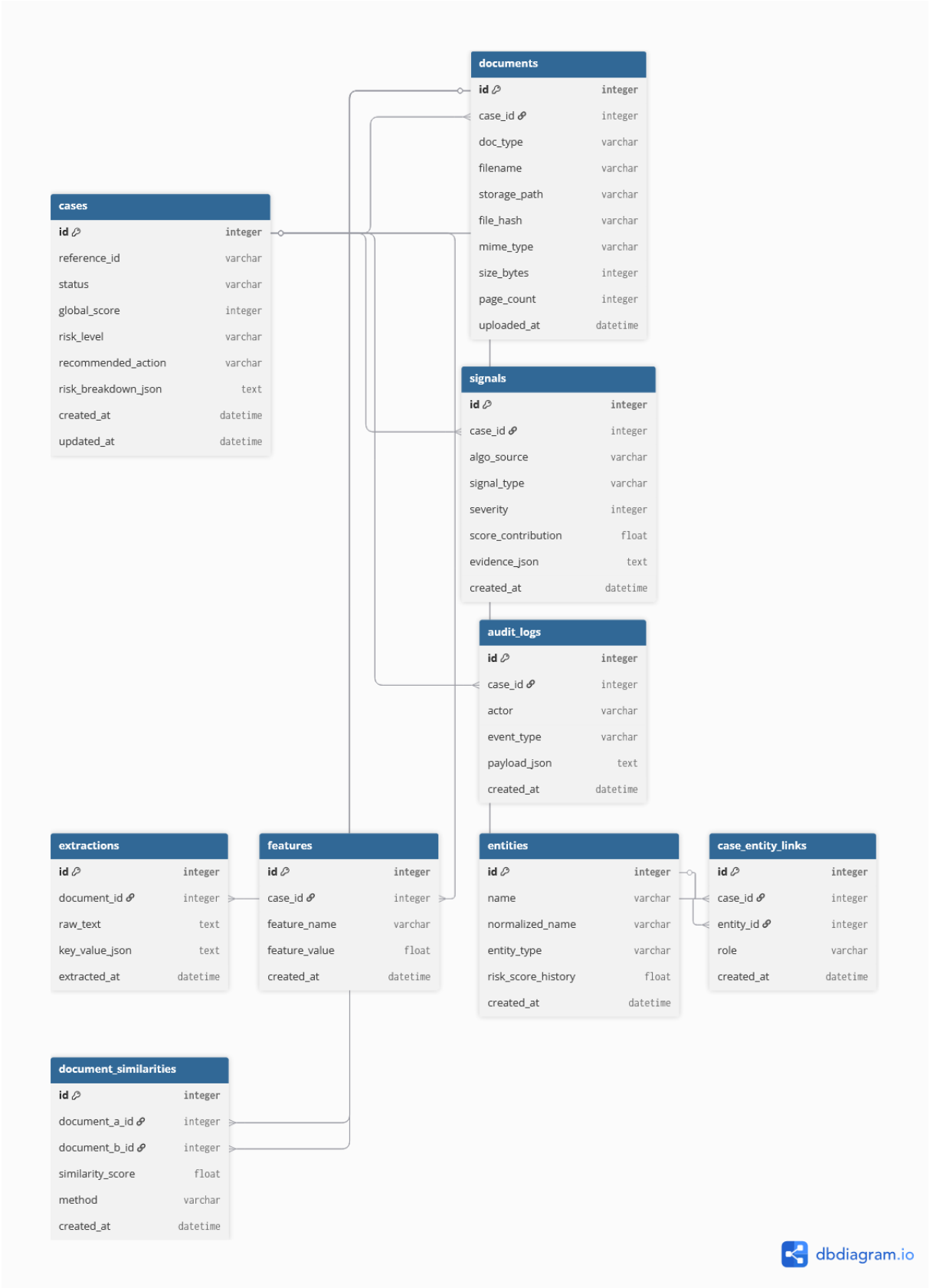

FIGURE 1 – Diagramme Entité-Relation (ERD) complet - PortNet HARPON

5 PLAN DE RÉALISATION

Cette roadmap privilégie une approche itérative *"Anti-Risque"* : garantir un historique (seed), sécuriser le flux de données avant d'ajouter la complexité algorithmique, et assurer un démonstrateur visuel fort (Graphe) pour la finale.

5.1 Phase 0 : Initialisation & Données Synthétiques (Seed)

Objectif : Garantir que les algorithmes disposent d'un historique dès le démarrage.

- Génération d'un script `seed_db.py` injectant 50 dossiers (45 sains, 5 frauduleux).
- Simulation d'un réseau de fraude (acteurs récurrents) pour tester Louvain.
- **Livrable :** Base de données pré-remplie et scénarios de démo prêts.

5.2 Phases 1 & 2 : Socle Technique & Contrat d'Interface

Objectif : Mettre en place un pipeline end-to-end fonctionnel.

- Définition stricte du contrat API (`GET /cases/{id}` retourne toutes les données d'affichage).
- Initialisation FastAPI + PostgreSQL (9 tables).
- Front-end React : Navigation Dashboard ↔ Page Détail.
- **Livrable :** Application navigable permettant de créer et visualiser un dossier.

5.3 Phases 3 & 4 : Ingestion & Extraction (Stratégie No-OCR)

Objectif : Fiabiliser l'entrée des données sans risque technique majeur.

- Upload multi-fichiers avec calcul de hash (SHA256) pour l'intégrité.
- Extraction de texte natif (via `pdfplumber`) pour éviter la lourdeur de l'OCR.
- Dégradation élégante : *EXTRACTION_PARTIAL* si document illisible.
- **Livrable :** Données textuelles stockées et affichables.

5.4 Phase 5 : Moteur Décisionnel v1 (Rule Engine)

Objectif : Première version du DSS explicable.

- Implémentation de règles métier simples (incohérences, outliers basiques).
- Génération automatique des *Top Reasons*, de l'*Evidence Pack* et de l'*Action recommandée*.
- **Livrable :** Fiche décisionnelle (Score + Raisons + Preuves + Action).

5.5 Phases 6 & 7 : IA (Benford & Isolation Forest)

Objectif : Déployer la détection statistique et les anomalies.

- **Benford :** distribution des chiffres sur montants facturés (preuve visuelle).
- **Isolation Forest :** détection des dossiers atypiques par rapport à l'historique (seed).
- **Livrable :** Alertes automatiques sur prix/poids anormaux.

5.6 Phase 8 : Module Réseau (Algorithme de Louvain)

Objectif : Effet "Wow" — détection de filières organisées.

- Normalisation des acteurs (Importateurs, Transitaires, Fournisseurs) dans `entities`.
- Construction du graphe des relations et détection de communautés (Louvain).
- **Livrable :** Visualisation graphique d'une filière suspecte.

FEUILLE DE ROUTE & CHECKLIST

SPRINT / PHASE	OBJECTIF OPÉRATIONNEL	FAIT ?
Sprint 1 : Socle *(Phases 0, 1 & 2)*	• Exécuter `seed_db.py` (50 cas générés). • API FastAPI connectée à la BDD. • Front-end : Navigation Dashboard ↔ Détail.	☐
Sprint 2 : Flux *(Phases 3 & 4)*	• Upload fonctionnel (stockage fichiers). • Extraction texte sans OCR (pdfplumber). • Affichage du texte brut dans la page Détail.	☐
Sprint 3 : Décision *(Phases 5 & 6)*	• Moteur de règles (Rule Engine). • Calcul Loi de Benford + Graphique. • Affichage "Evidence Pack" (Score + Raisons). • **Action recommandée** (Libérer / Demander pièces / Inspecter) + checklist. • **Audit log minimal** (CASE_CREATED, ANALYSIS_START/DONE, DECISION_MADE).	☐
Sprint 4 : Anomalies *(Phase 7)*	• Calcul des features (Prix/kg, Poids). • Intégration Isolation Forest (Scikit-learn). • Alertes automatiques sur dossiers atypiques.	☐
Sprint 5 : Réseau *(Phase 8)*	• Normalisation des entités (Importateurs, Transitaires, Fournisseurs). • Construction des liens (SQL) + graphe exploitable. • Visualisation Graphe + Algo Louvain (communautés).	☐
Sprint 6 : Final *(Phases 10 & 11)*	• Export PDF du rapport (Evidence Pack). • Optimisation Perf (`GET /cases/{id}` < 200ms). • **Feedback Loop** : Fraude confirmée / Faux positif (journalisé). • Scénario de démo répété 3 fois.	☐