AUDEX — Document Technique d'Architecture et de Conception (Phase MVP)

Version 1.0 — Octobre 2025

Ragnang-Newende Yanis Axe DABO
Développeur Logiciel, Passionné d'IA

Octobre 2025
Ouagadougou, Burkina Faso

Table des matières

1 Architecture Globale

1.1 Type et titre de l'architecture

L'architecture retenue pour AUDEX est de type **n-tiers** (client–serveur), avec une séparation claire des responsabilités entre les différentes couches : la couche de présentation (*Web app — React*), la couche de services applicatifs (*FastAPI*), la couche de données (*SQLite pour le MVP, Postgres en pilote*), la couche d'intelligence (pipelines IA de vision et de scoring) et la couche de sécurité/traçabilité (hachage et ancrage des rapports). Cette architecture modulaire facilite la maintenabilité, l'évolutivité et la clarté du code.

1.2 Architecture logique

La table suivante présente les principaux composants de l'architecture et les technologies associées :

Couche	Description	Technologies
Présentation (UI)	Application web réactive avec gestion locale des audits, cartes et génération de rapports.	React, HTML5, IndexedDB
API / Backend	Services REST pour l'ingestion, l'analyse, le scoring, la génération de rapport et la gestion des utilisateurs.	FastAPI (Python)
IA / Analyse	Pipelines de traitement combinant OCR, vision par ordinateur et règles heuristiques pour détecter et classifier les anomalies.	OpenCV, Scikit-learn, Tesseract
Persistance	Base de données relationnelle et stockage de médias.	SQLite (MVP), Postgres (Pilote)
Sécurité / Intégrité	Authentification, hachage des rapports et ancrage sur blockchain.	JWT, AES, Web3.py

Table 1 – Vue logique des composants principaux

1.3 Architecture physique

L'implantation varie selon les phases du projet :

— **MVP :** déploiement monolithique dans un conteneur Docker unique regroupant l'API et l'interface utilisateur, avec SQLite pour le stockage local des données et des médias.

— **Pilote :** déploiement distribué (API hébergée sur une plateforme cloud, base de données gérée et stockage d'objets externe), permettant la montée en charge et l'accès simultané de plusieurs auditeurs.

— **Production :** infrastructure scalable avec intégration continue, supervision, sauvegardes automatiques et montée en charge dynamique. Les audits sont ancrés sur le *mainnet* de la blockchain pour assurer la traçabilité.

2 Diagrammes techniques

2.1 Diagramme de cas d'usage

Ce diagramme illustre les interactions entre les acteurs (Auditeur, Superviseur, Administrateur, Lecteur) et les fonctions principales (import des médias, analyse et scoring, génération du rapport, visualisation de la carte, approbation de l'audit, gestion des utilisateurs, vérification d'intégrité).

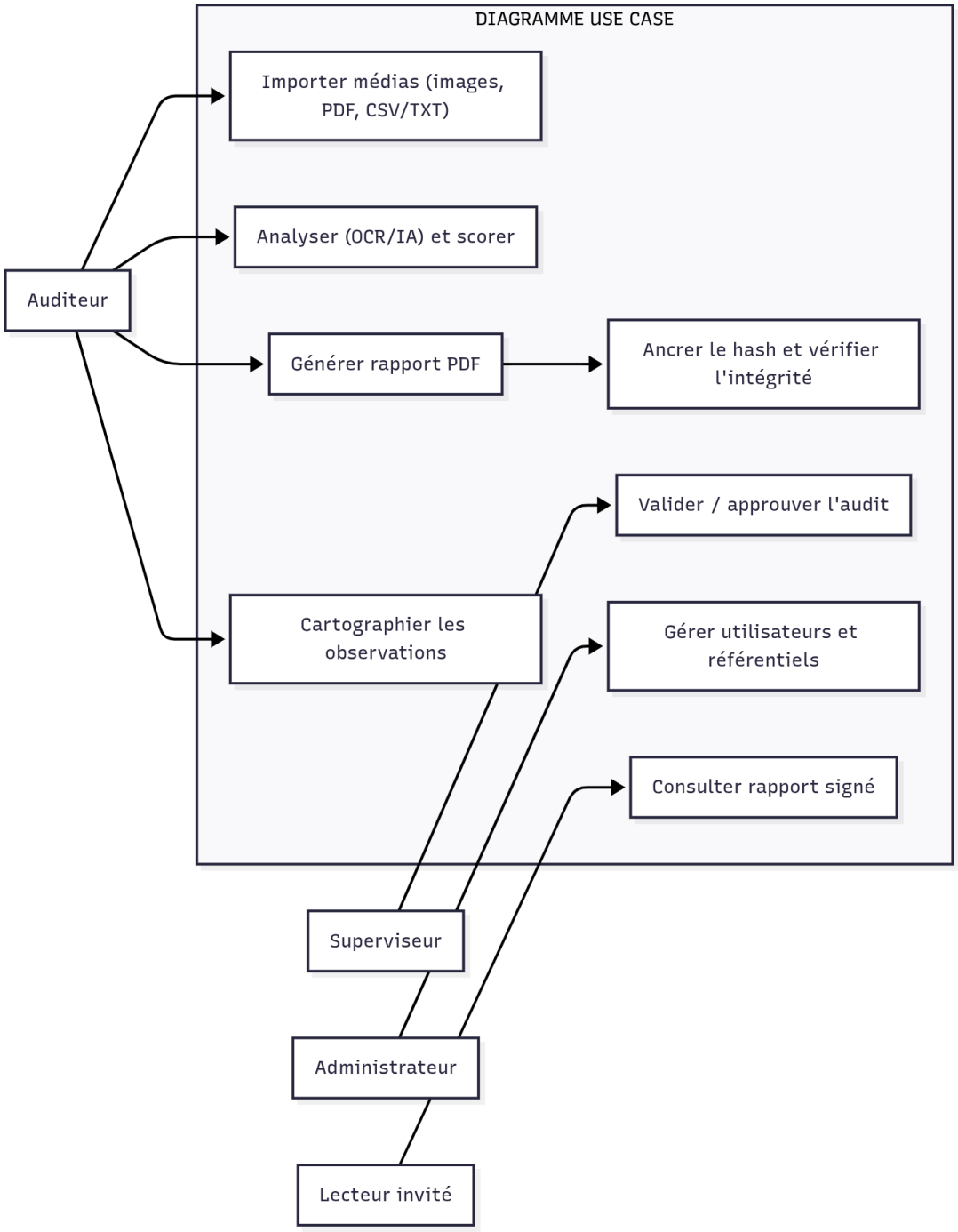

FIGURE 1 – Diagramme de cas d'usage

2.2 Diagramme de classes (noyau métier)

Le diagramme de classes suivant présente les entités principales d'AUDEX (`Site`, `Audit`, `Observation`, `Media`, `Recommandation`, `Utilisateur`, `Règle`) et leurs relations. Un audit est rattaché à un site et créé par un utilisateur ; il agrège plusieurs observations, chacune possédant des médias et des recommandations. Les règles locales valident ou pondèrent les observations.

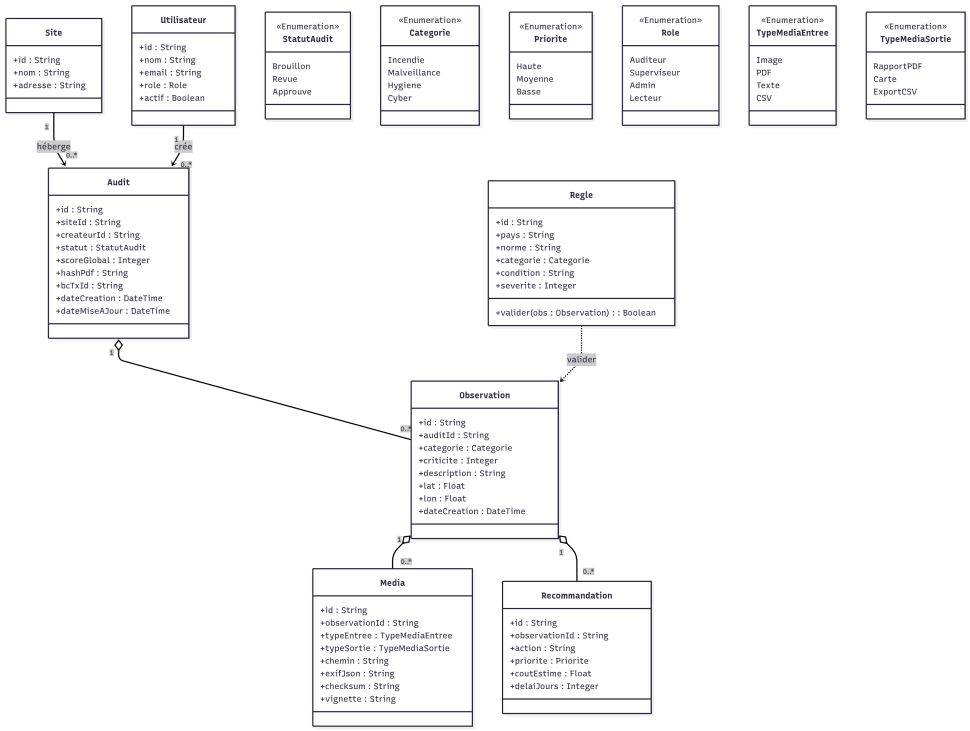

FIGURE 2 – Diagramme de classes du domaine AUDEX

2.3 Diagramme d'architecture système

Le schéma suivant résume les interactions entre la Web App côté client, les services du backend, les bases de données, le stockage de fichiers et la blockchain. Le client transmet les fichiers à l'API pour ingestion ; l'API délègue l'extraction aux pipelines IA, persiste les résultats et déclenche la génération de rapport et l'ancrage du hachage.

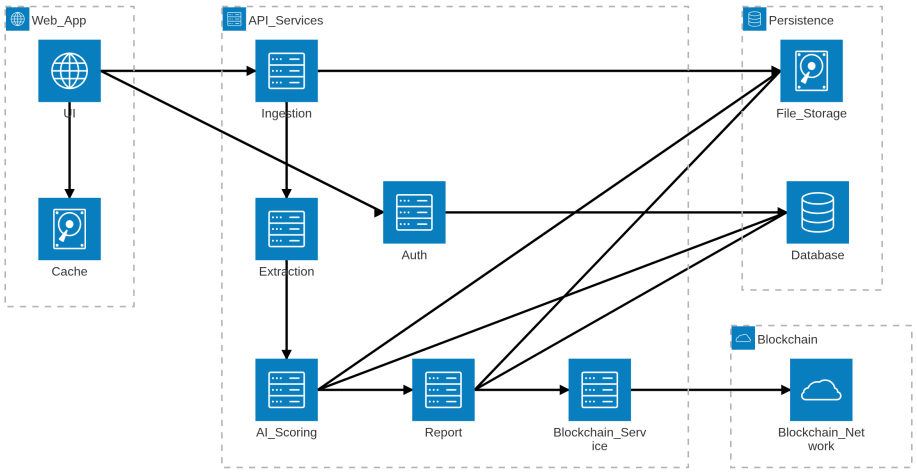

FIGURE 3 – Vue d'architecture logique des composants

2.4 Diagramme de séquence (flux principal)

Ce diagramme de séquence détaille les échanges entre l'auditeur, l'interface utilisateur, l'API, les composants d'IA, la base de données, le stockage de fichiers et la blockchain lors du traitement

d'un audit : import des fichiers, création de l'audit, analyse, sauvegarde des résultats, génération et ancrage du rapport.

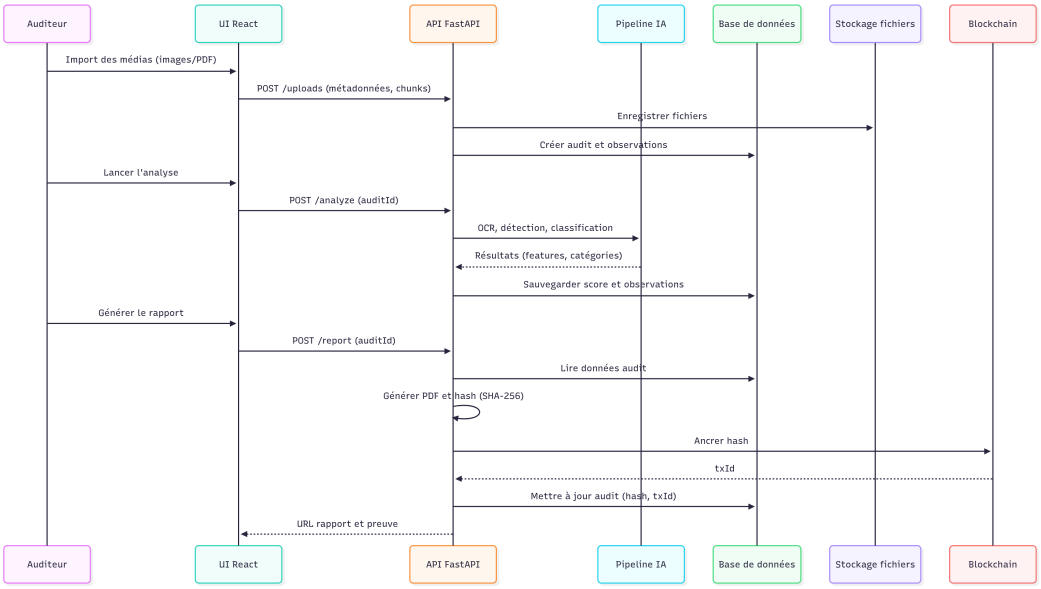

FIGURE 4 – Diagramme de séquence — flux principal d'un audit

3 Patterns et approches d'implémentation

3.1 Patterns utilisés

Les modèles de conception suivants structurent le code pour en faciliter l'évolution et la maintenabilité :

Pattern	**Rôle**	**Exemple**
Service Layer	Séparer la logique métier des contrôleurs API.	*ScoringService*, *ReportService*
Repository	Isoler la persistance et simplifier les requêtes.	*AuditRepository*, *UserRepository*
Adapter	Interfacer des services externes (OCR, blockchain).	*OCRAdapter*, *BlockchainAdapter*
Factory	Générer des objets complexes (rapports).	*ReportFactory*
Strategy	Gérer les variantes IA (pays, normes).	*ComplianceStrategy*
Observer	Déclencher des actions après certaines opérations.	*AuditObserver*

TABLE 2 – Principaux patterns de conception utilisés

3.2 Approches de développement

— **Modularité :** chaque service (ingestion, extraction, IA, scoring, rapport, blockchain) est développé de manière indépendante et testable.

— **Interfaces explicitement définies :** les dépôts et les services publient des interfaces stables, limitant le couplage et facilitant le remplacement ultérieur.

— **Tests unitaires et d'intégration :** les parsers et calculs de scoring sont testés isolément ; des tests de bout en bout valident les flux complets.

— **Intégration continue (à partir du MVP+) :** automatisation de la construction et du déploiement, avec exécution automatique des tests.

— **Code documenté et typé :** utilisation d'annotations de types Python et d'une documentation claire pour faciliter l'onboarding.

4 Mesures de sécurité

4.1 Contrôle d'accès et gestion d'identité

— Authentification par jetons JWT, avec expirations courtes et mécanismes de rafraîchissement.

— RBAC (contrôle d'accès basé sur les rôles) définissant les permissions des auditeurs, superviseurs, administrateurs et lecteurs.

— Journalisation des actions sensibles (création, modification, validation d'audit) pour assurer la traçabilité.

4.2 Protection des données

— Chiffrement en transit via HTTPS (TLS 1.3).

— Chiffrement local (AES) des caches et exports temporaires, selon les besoins.

— Nettoyage automatique des métadonnées EXIF non nécessaires pour limiter les fuites accidentelles.

4.3 Intégrité et traçabilité

— Hachage SHA-256 des rapports PDF générés et ancrage sur une blockchain (réseau *testnet* durant le MVP, *mainnet* en production).

— Vérification d'intégrité côté lecteur : recalcul du hachage lors de la consultation pour détecter toute modification.

4.4 Sécurité applicative

— Validation stricte des fichiers chargés (taille, type MIME, schéma) et sanitisation des PDF et images pour éliminer les contenus potentiellement dangereux.

— Limitation des volumes d'upload et analyse antivirus optionnelle.

— Mise en place d'en-têtes HTTP de sécurité (CSP, HSTS, X-Frame-Options, X-Content-Type-Options).

4.5 Sécurité opérationnelle

— Gestion sécurisée des secrets via des variables d'environnement et rotation régulière des clés.

— Surveillance des vulnérabilités des dépendances et mises à jour régulières.

— Sauvegardes automatiques des bases et médias, avec tests de restauration périodiques.

— Mise en place de la supervision et des alertes pour détecter les anomalies.

5 Résumé de l'architecture sécurisée

Le tableau ci-après synthétise les principales menaces identifiées et les mesures associées :

Composant	**Menace**	**Mesures de sécurité**
Auth API	Falsification de jetons	Vérification de la signature et des délais d'expiration des JWT
Données	Vol ou fuite	Chiffrement AES, utilisation systématique d'HTTPS
Rapport PDF	Modification du contenu	Hachage SHA-256 et ancrage sur blockchain
Upload	Fichiers malveillants	Vérification du type MIME, analyse antivirus, limitation de taille
Accès	Escalade de privilèges	RBAC strict, journalisation et audits réguliers

TABLE 3 – Résumé des menaces et des mesures de sécurité

6 Conclusion technique

L'architecture et le design du système AUDEX s'appuient sur une base modulaire et testable qui permettra une évolution maîtrisée du produit. La séparation claire des couches (présentation, services, données, IA et sécurité) et l'utilisation de patterns éprouvés (service layer, repository, adapter, etc.) assurent une lisibilité et une maintenabilité optimales.

Les mesures de sécurité intégrées (authentification forte, chiffrement, contrôle d'accès, journalisation et ancrage blockchain) contribuent à garantir la confidentialité, l'intégrité et la traçabilité des audits réalisés. Cette conception jette les bases d'un MVP fiable et évolutif, ouvrant la voie aux phases pilote et production sans refonte majeure.