Architecture sécurisée pour la gouvernance et la gestion de données industrielles en contexte de recherche

par

Charbel NASR

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE
LA MAÎTRISE AVEC MÉMOIRE EN GÉNIE DES TECHNOLOGIES DE L'INFORMATION
M. Sc. A.

MONTRÉAL, LE 23 AVRIL 2026

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

 Charbel Nasr 2026

Cette licence Creative Commons CC BY-NC-ND 4.0 signifie qu'il est permis de diffuser, d'imprimer ou de sauvegarder sur un autre support une partie ou la totalité de cette œuvre à condition de créditer l'auteur, que ces utilisations soient faites à des fins non commerciales et que le contenu de l'œuvre n'ait pas été modifié.

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Antoine Tahan, directeur de mémoire
Département de génie mécanique à l'École de technologie supérieure

M. Pavel Guerra Côté, codirecteur de mémoire
Innergex

M. Lucas Hof, président du jury
Département de génie mécanique à l'École de technologie supérieure

M. Alain April, membre du jury
Département de génie logiciel et TI à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 13 AVRIL 2026

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

AVANT-PROPOS

Le projet présenté dans ce mémoire s'inscrit dans le cadre de la recherche appliquée en technologies de l'information et porte sur la gestion et le stockage de données industrielles. Il est né d'un besoin spécifique observé dans un contexte réel, où la gestion, l'organisation et le partage d'ensembles de données volumineux issus de différents partenaires représentent un défi pour les équipes de recherche.

La plateforme a été développée en tenant compte des contraintes opérationnelles de l'équipe de recherche, notamment en termes de sécurité des données, de centralisation, de facilité de maintenance et d'intégration avec les outils déjà utilisés par les chercheurs.

Architecture sécurisée pour la gouvernance et la gestion de données industrielles en contexte de recherche

Charbel NASR

RÉSUMÉ

Les systèmes industriels modernes génèrent d'importantes quantités de données issues de capteurs et d'équipements automatisés, largement exploitées dans les travaux de recherche en apprentissage automatique. Toutefois, la gestion de ces données — en particulier leur organisation, leur documentation et leur partage dans le cadre d'un projet de recherche avec plusieurs partenaires — demeure une tâche complexe. L'absence de pratiques de gestion homogènes peut compromettre la reproductibilité des expérimentations lorsque les sources de données ou les transformations appliquées ne sont pas clairement documentées.

Ce mémoire présente la conception et la mise en œuvre d'une architecture sécurisée destinée à centraliser et gouverner des ensembles de données industrielles à l'aide de métadonnées structurées. Les données ciblées proviennent principalement de systèmes de supervision et d'acquisition en temps réel utilisés dans le domaine de l'énergie renouvelable. La solution proposée repose sur une base de données relationnelle, des interfaces applicatives sécurisées pour le contrôle des accès, ainsi que l'intégration d'un environnement JupyterHub dédié à la préparation et à l'exploration des données.

L'objectif de cette architecture est de faciliter l'organisation, le partage, la traçabilité et la reproductibilité des données industrielles utilisées dans les travaux d'analyse et de recherches, tout en limitant les besoins de maintenance opérationnelle.

Mots-clés : Systèmes SCADA, Gouvernance des données, Métadonnées, Base de données relationnelle, Sécurité des données, Contrôle d'accès, Interfaces applicatives (API), JupyterHub

Secure architecture for governance and management of industrial data in a research context

Charbel NASR

ABSTRACT

Modern industrial systems generate large volumes of data from sensors and automated equipment, which are widely used in machine learning research. However, managing these datasets — particularly in terms of organization, documentation, and sharing — remains a significant challenge for research teams. The lack of consistent data management practices can hinder the reproducibility of experiments when data sources or applied transformations are not clearly documented.

This thesis presents the design and implementation of secure architecture aimed at centralizing and governing industrial datasets using structured metadata. The data considered primarily originates from real-time supervision and acquisition systems deployed in the renewable energy sector. The proposed solution relies on a relational database, secure application programming interfaces for access control, and the integration of a JupyterHub environment dedicated to data preparation and exploratory analysis.

The objective of this architecture is to improve the organization, sharing, and reproducibility of industrial data used in research, while minimizing operational maintenance requirements.

Keywords: SCADA systems, Data governance, Metadata, Relational database, Data security, Access control, Application programming interfaces (API), JupyterHub

TABLE DES MATIÈRES

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

API	interface de programmation applicative (*Application Programming Interface*)
ASP.NET	cadriciel web Microsoft (*Active Server Pages .NET*)
CSV	format de fichier texte à valeurs séparées par des virgules (*Comma-Separated Values*)
DB	base de données (*Database*)
DBO	*Database Owner*
EF Core	cadriciel de mapping objet-relationnel pour .NET (*Entity Framework Core*)
ÉTS	École de technologie supérieure
HDD	Disque dur (*Hard Disk Drive*)
HTTP	protocole de transfert hypertexte (*HyperText Transfer Protocol*)
HTTPS	protocole sécurisé de transfert hypertexte (*HyperText Transfer Protocol Secure*)
IETF	*Internet Engineering Task Force*
IIS	serveur web Microsoft pour l'hébergement d'applications web (*Internet Information Services*)
JSON	format d'échange de données texte structuré (*JavaScript Object Notation*)
JWT	jeton d'authentification sécurisé (*JSON Web Token*)
MFA	*multi-factor authentication*
NIST	*National Institute of Standards and Technology*
Parquet	format de stockage de données colonne optimisée pour l'analyse (*Apache Parquet Columnar Storage Format*)
RBAC	contrôle d'accès basé sur les rôles (*Role-Based Access Control*)
REST	style d'architecture pour services web (*Representational State Transfer*)
SCADA	système de supervision et d'acquisition de données (*Supervisory Control and Data Acquisition*)
SQL	*Structured Query Language*

XX

SSD	*Solid State Drive*
WSL2	*Windows Subsystem for Linux 2*
ZIP	Format de compression ZIP

INTRODUCTION

Ce mémoire s'inscrit dans un contexte de recherche appliquée où des données industrielles sont exploitées pour des travaux d'analyse avancée et d'apprentissage automatique. Ces données proviennent généralement de systèmes industriels hétérogènes et sont produites en grande quantité. Dans un cadre de recherche, leur utilisation efficace repose non seulement sur leur disponibilité, mais aussi sur leur organisation et leur documentation. En pratique, ces exigences sont rarement satisfaites de manière systématique. Les ensembles de données sont souvent répartis entre plusieurs systèmes, décrits de façon incomplète et soumis à des règles d'accès implicites ou informelles. Cette situation complique la collaboration entre équipes, limite la réutilisation des jeux de données et rend difficile la reproduction des analyses lorsque les sources initiales ou les étapes de préparation ne sont pas clairement identifiées.

L'objectif est de concevoir une plateforme centralisée et sécurisée permettant le dépôt, l'indexation et l'accès contrôlé à des ensembles de données enrichis par des métadonnées structurées. Ces métadonnées constituent un élément central du dispositif, car elles permettent de documenter explicitement les sources, les transformations appliquées et les règles de visibilité associées à chaque ensemble. La solution développée repose sur une architecture applicative dans laquelle l'accès aux données s'effectue principalement via des interfaces programmatiques sécurisées. Cette approche vise à permettre l'exploitation directe des données dans des environnements d'analyse, sans nécessiter leur duplication ou leur téléchargement systématique. Elle contribue ainsi à renforcer la traçabilité des flux de traitement et à soutenir la reproductibilité des analyses, tout en maintenant un cadre de sécurité adapté à un environnement partagé.

Le périmètre d'utilisation de la plateforme est volontairement limité à un cadre interne dans le contexte de ce travail. L'architecture et le modèle de gouvernance ont toutefois été conçus de manière à supporter plusieurs niveaux de visibilité et des usages par différentes équipes.

Ce mémoire est structuré comme suit. Le premier chapitre présente le contexte scientifique et industriel ainsi que les enjeux liés à l'exploitation des données industrielles en recherche. Le

deuxième chapitre expose l'état de l'art des plateformes de gestion de données. Les chapitres suivants détaillent la conception architecturale, l'implémentation technique de la plateforme et son intégration dans les flux de travail analytiques. Enfin, le dernier chapitre discute des limites actuelles et des perspectives d'évolution.

CHAPITRE 1

CONTEXTE SCIENTIFIQUE ET INDUSTRIEL

1.1 Données industrielles dans les systèmes d'énergie renouvelable

Les installations de production d'énergie verte dépendent fortement des systèmes de contrôle industriels, en particulier des systèmes SCADA, pour surveiller en permanence le fonctionnement des équipements. Ces systèmes collectent diverses mesures, telles que la production d'électricité, la vitesse de rotation des composants, les niveaux de tension, la température des composants et l'état de fonctionnement des machines. Ces données sont complétées par des données météorologiques et environnementales : force et direction du vent, rayonnement solaire, température extérieure, ainsi que les précipitations (solides ou liquides), entre autres. Les systèmes industriels produisent également des fichiers journaux qui documentent les alertes, les changements d'état, les interventions de maintenance et les incidents affectant les opérations. Toutes ces informations sont enregistrées à différents intervalles, allant de quelques secondes à plusieurs minutes, en fonction des appareils et des exigences opérationnelles. Au fil du temps, cette collecte continue entraîne l'accumulation de quantités importantes de données couvrant de longues périodes d'activité.

1.2 Prédiction d'anomalies

La disponibilité croissante de ces données a favorisé le développement de méthodes d'analyse avancée, en particulier pour la détection d'anomalies. Dans le domaine de l'énergie renouvelable, cet enjeu est directement lié à des objectifs opérationnels concrets, tels que la réduction des coûts de maintenance, l'optimisation de la production ou l'allongement de la durée de vie des équipements.

Les approches actuelles reposent fréquemment sur des techniques d'apprentissage automatique exploitant des historiques de fonctionnement afin d'identifier des écarts par rapport à un comportement nominal. La qualité de ces historiques joue alors un rôle central : des données incomplètes, mal documentées ou peu représentatives des conditions réelles d'exploitation

peuvent compromettre la pertinence des modèles développés. L'accès à des ensembles fiables, clairement décrits et cohérents dans le temps constitue ainsi une condition essentielle à la validité des Méthodes développées.

1.3 Enjeux de gestion des données industrielle en recherche

Dans un contexte de recherche appliquée, la gestion des données industrielles ne se limite pas aux aspects de stockage. Elle implique également la capacité à documenter explicitement les ensembles de données à l'aide de métadonnées structurées décrivant leur origine, les capteurs ou systèmes d'acquisition, les périodes couvertes, les formats utilisés et, lorsque pertinent, des indicateurs de qualité.

En l'absence d'un cadre structuré, ces informations restent souvent dispersées entre fichiers de travail, notes techniques ou conventions locales propres aux équipes. Cette fragmentation entraîne une perte progressive du contexte d'acquisition et de préparation des données, rendant leur réutilisation plus complexe et fragilisant la reproductibilité des analyses. Ces difficultés sont accentuées lorsque plusieurs équipes ou projets exploitent des ensembles similaires sans mécanisme formel de gouvernance.

Par ailleurs, les données industrielles peuvent inclure des informations sensibles nécessitant des mécanismes explicites de sécurité et de contrôle d'accès. La gestion de la visibilité des ensembles, combinée à des règles d'authentification adaptées, constitue un enjeu central pour concilier collaboration scientifique, conformité organisationnelle et sécurité opérationnelle. Ces considérations soulignent la nécessité de solutions structurées permettant une gestion cohérente, documentée et sécurisée des données industrielles dans un cadre de recherche.

CHAPITRE 2

ÉTAT DE L'ART

2.1 Plateformes de gestion et de partage de données scientifiques

Les plateformes de gestion de données scientifiques occupent une place centrale dans les pratiques de recherche actuelles. Elles visent principalement à faciliter l'accès aux ensembles de données, à standardiser leur description et à encourager leur réutilisation au sein de la communauté scientifique. Elles répondent également à des exigences de conformité croissantes, notamment en lien avec les politiques de publication, les obligations institutionnelles et certaines contraintes réglementaires.

Des plateformes largement utilisées dans le milieu académique illustrent cette approche généraliste. Elles sont conçues pour accueillir des ensembles provenant de domaines variés et proposer des mécanismes communs de stockage, de description et de diffusion, indépendamment du contexte scientifique ou industriel d'origine. Cette généralité constitue un avantage en termes de mutualisation, mais implique également certains compromis lorsque les données présentent des contraintes spécifiques.

2.2 Fonctionnalités principales des plateformes

Malgré leur diversité, les plateformes de données scientifiques partagent généralement un ensemble de fonctionnalités fondamentales répondant aux besoins transversaux de la recherche.

Elles offrent tout d'abord des mécanismes de stockage centralisé, permettant de regrouper des ensembles structurés ou non structurés et d'en assurer la conservation sur le long terme. Certaines proposent des formes simples de versionnement ou d'indexation afin de faciliter la recherche et l'identification des ressources.

La gestion des métadonnées constitue un second pilier essentiel. Les ensembles sont généralement accompagnés d'informations descriptives portant sur leur origine, leurs conditions d'acquisition, leurs formats, ainsi que sur les restrictions d'usage ou de diffusion.

Les plateformes intègrent également des mécanismes de contrôle d'accès, souvent basés sur des rôles ou des groupes d'utilisateurs. La granularité de ces contrôles varie toutefois selon les solutions et reste parfois limitée dans des contextes ouverts ou publics.

De plus, un nombre croissant de plateformes propose un accès programmatique aux données, par l'intermédiaire d'interfaces applicatives. Cet accès permet l'intégration des ensembles dans des flux d'analyse automatisés, et, dans certains cas, l'utilisation directe d'environnements analytiques intégrés, tels que des notebooks interactifs.

Ces fonctionnalités, bien que communes à la majorité des plateformes, se manifestent de manière différente selon les choix architecturaux et les publics visés. Deux solutions particulièrement répandues permettent d'en saisir les contrastes :

2.2.1 Kaggle

Kaggle est une plateforme en ligne fondée en 2010, rachetée par Google en 2017. C'est un espace de référence pour le partage de jeux de données, l'organisation de compétitions analytiques et l'expérimentation de modèles. Au-delà du simple hébergement, elle intègre des environnements de calcul interactifs sous forme de notebooks, offrant ainsi aux utilisateurs la possibilité d'explorer et d'analyser les données directement en ligne. Son positionnement est résolument communautaire et tourné vers la pratique.

2.2.2 CKAN

CKAN est une plateforme libre de droits initialement développée par l'Open Knowledge Foundation. Elle est aujourd'hui largement adoptée pour la mise en place de portails de données ouvertes au public, en particulier par des administrations publiques et des organisations internationales.

2.3 Limites dans le contexte des données industrielles renouvelables

Si ces plateformes répondent efficacement aux besoins généraux de la recherche scientifique, leur utilisation dans des contextes industriels spécialisés soulève certaines limites. Les données issues des systèmes de production d'énergie renouvelable présentent en effet des caractéristiques particulières : elles sont souvent continues, fortement dépendantes du contexte opérationnel et produites à grande échelle sur de longues périodes.

Par ailleurs, ces données peuvent inclure des informations sensibles liées aux infrastructures industrielles, aux performances des équipements ou aux stratégies d'exploitation. Dans de nombreux cas, elles ne peuvent pas être diffusées publiquement et doivent rester hébergées sur des infrastructures locales sécurisées. Les plateformes généralistes, principalement orientées vers le partage ouvert ou interinstitutionnel, ne sont pas toujours adaptées à ces contraintes.

Ces limitations se traduisent notamment par une gouvernance parfois insuffisamment fine, une difficulté à imposer des règles de visibilité explicites ou une intégration limitée avec des environnements analytiques opérant dans des réseaux restreints.

2.4 Synthèse

Cette section a permis de présenter les principales plateformes de gestion et de partage de données scientifiques, ainsi que leurs fonctionnalités communes, notamment en matière de stockage, de gestion des métadonnées, de contrôle d'accès et d'intégration avec des environnements analytiques.

L'analyse a également mis en évidence les limites de ces solutions dans le contexte spécifique des données industrielles issues des systèmes d'énergie renouvelable, en particulier en ce qui concerne la gestion de la confidentialité, la gouvernance fine des accès et l'intégration dans des environnements sécurisés.

Ces constats justifient la nécessité de proposer une solution adaptée aux contraintes des environnements de recherche industrielle. Le chapitre suivant présente ainsi la problématique générale et l’analyse des besoins qui orientent la conception de l’architecture proposée.

CHAPITRE 3

PROBLÉMATIQUE ET ANALYSE DES BESOINS

3.1 Contexte opérationnel et problématique générale

Dans les projets de recherche appliquée exploitant des données industrielles, la gestion des ensembles expérimentaux ne se limite pas au stockage des fichiers. Les équipes doivent manipuler des volumes importants de données tout en respectant des contraintes liées à la sécurité, à la confidentialité et à la maîtrise des conditions d'accès.

Dans le domaine des systèmes énergétiques renouvelables, les données issues des infrastructures de supervision industrielle sont collectées de manière continue et restent étroitement associées à un contexte opérationnel précis. Pour pouvoir être exploitées dans des analyses scientifiques, ces données doivent être conservées sur la durée, documentées de façon explicite et rendues accessibles selon des règles clairement définies. Ces conditions sont nécessaires pour permettre des analyses reproductibles et comparables dans le temps.

Les plateformes généralistes présentées dans l'état de l'art permettent de répondre à certains de ces besoins, notamment en matière de stockage et de diffusion. En revanche, elles peinent à combiner simultanément la sensibilité des données industrielles, une gouvernance fine des accès et une structuration adaptée des métadonnées. Ce constat met en évidence la nécessité d'une approche plus spécialisée, capable de fournir un cadre cohérent pour la gestion, la visibilité et l'accès contrôlé aux données industrielles dans un contexte de recherche appliquée.

3.2 Problèmes rencontrés dans les environnements de recherche industrielle

3.2.1 Fragmentation des ressources et perte de contexte

Dans de nombreuses équipes de recherche, les ensembles expérimentaux sont répartis entre différents supports et systèmes, tels que des bases de données locales, des espaces de fichiers partagés, des supports externes ou des environnements analytiques individuels. Cette

dispersion complique la compréhension du contexte d'acquisition, des transformations appliquées et des conditions d'utilisation des données.

Avec le temps, le lien entre les ensembles et les informations nécessaires à leur interprétation tend à se dégrader. Cette perte progressive de contexte réduit la capacité à réutiliser les données dans de nouveaux travaux et limite leur valeur scientifique.

3.2.2 Difficultés de collaboration entre équipes

L'absence d'une infrastructure commune et sécurisée constitue un frein important à la collaboration entre équipes de recherche. Le partage des ensembles expérimentaux repose souvent sur des échanges manuels ou sur des mécanismes de contrôle d'accès peu granulaires, ce qui rend difficile un partage maîtrisé entre groupes ou organisations distinctes.

Dans ce contexte, les échanges par copie de fichiers restent fréquents. Cette pratique favorise la multiplication de versions divergentes d'un même ensemble et complique la coordination des travaux de recherche.

3.2.3 Contraintes de sécurité et de confidentialité

Les données industrielles peuvent contenir des informations sensibles relatives aux infrastructures physiques, aux performances des équipements ou aux processus de production. Pour cette raison, les solutions de diffusion publique ne sont généralement pas adaptées à leur gestion.

Certaines ressources doivent être conservées au sein d'infrastructures locales sécurisées et accessibles uniquement à des utilisateurs autorisés. Ces contraintes imposent des architectures compatibles avec des déploiements internes et intégrant des mécanismes explicites de contrôle d'accès et de gestion de la visibilité.

3.2.4 Contraintes opérationnelles et maintenance

Les équipes de recherche disposent rarement de ressources dédiées à l'administration de plateformes complexes. Les solutions nécessitant une configuration avancée ou une maintenance continue peuvent rapidement devenir difficiles à exploiter sur le long terme. Il est donc nécessaire de privilégier des systèmes stables, conçus pour un usage quotidien, et limitant autant que possible l'effort d'administration requis pour leur fonctionnement.

3.2.5 Absence de documentation minimale standardisée

Dans de nombreux environnements de recherche, aucun niveau minimal de documentation n'est imposé lors de la création d'un ensemble expérimental. La qualité et la complétude des informations associées dépendent alors fortement des pratiques individuelles des chercheurs.

Lorsque le contexte d'acquisition, les transformations appliquées ou les restrictions d'utilisation ne sont pas clairement documentés, la réutilisation des ensembles devient difficile, en particulier lorsque les personnes ayant produit les données ne sont plus impliquées dans le projet.

3.2.6 Volume des ensembles et difficulté de partage

Les ensembles issus de systèmes industriels peuvent atteindre des volumes importants, notamment lorsqu'ils couvrent des périodes d'acquisition continues sur plusieurs mois ou années. Le transfert complet de ces ensembles entre équipes ou environnements analytiques implique des temps de copie élevés et des coûts opérationnels significatifs.

Cette situation favorise la création de copies locales multiples, augmentant le risque d'incohérences entre différentes versions d'un même ensemble. Un accès programmatique permettant l'extraction ciblée de sous-ensembles, sans duplication complète des données, apparaît alors comme une approche plus adaptée dans un contexte de recherche industrielle.

3.3 Besoins fonctionnels

L'analyse des problèmes rencontrés dans les environnements de recherche industrielle permet d'identifier plusieurs besoins fonctionnels essentiels.

1. **Création et gestion d'ensembles expérimentaux structurés :** Le système doit permettre la création et la gestion d'ensembles expérimentaux clairement identifiés, associés à des métadonnées décrivant leur origine, leur contexte d'acquisition et leurs conditions d'utilisation.
2. **Gestion de différents types de ressources :** Le système doit être capable de gérer plusieurs types de ressources, incluant des ensembles de données structurées ainsi que des fichiers.
3. **Préparation et transformation des données :** Le système doit permettre la préparation des ensembles de données avant leur exploitation analytique. Cela inclut notamment des opérations de nettoyage, de transformation et de structuration des données, ainsi que, lorsque nécessaire, des mécanismes d'anonymisation, de normalisation ou de standardisation**.**
4. **Accès programmatique aux ensembles :** Le système doit offrir un accès programmatique aux ensembles expérimentaux afin de permettre leur intégration directe dans des flux de travaux analytiques, notamment dans des environnements de type carnet de note, sans duplication systématique des données.
5. **Gestion de plusieurs niveaux de visibilité :** Le système doit permettre de définir différents niveaux de visibilité et d'accès aux ensembles, incluant des accès publics, restreints aux utilisateurs authentifiés, organisationnels ou individuels, afin de répondre aux contraintes de confidentialité et de collaboration.

3.4 Besoins non fonctionnels

En complément des fonctionnalités attendues, le système doit répondre à un ensemble d'exigences non fonctionnelles liées à son exploitation et à son intégration dans un contexte de recherche industrielle.

1. **Sécurité et contrôle des accès :** Le système doit garantir un niveau de sécurité adapté aux données industrielles, incluant des mécanismes d'authentification et un contrôle d'accès basé sur des rôles et des règles de visibilité explicites.
2. **Compatibilité avec des déploiements internes :** Le système doit pouvoir être déployé sur des infrastructures locales sécurisées, lorsque les données ne peuvent pas être exposées via des solutions publiques.
3. **Intégration avec les environnements analytiques existants :** Le système doit permettre une intégration simple avec des environnements d'analyse déjà utilisés par les chercheurs, notamment par l'intermédiaire d'interfaces programmatiques sécurisées.
4. **Charge d'administration limitée :** Le système doit être conçu de manière à limiter les besoins d'administration manuelle et à permettre une exploitation stable sur le long terme, en adéquation avec les ressources généralement disponibles dans les équipes de recherche.
5. **Capacité d'évolution organisationnelle :** Le système doit pouvoir évoluer pour supporter des collaborations impliquant plusieurs équipes ou organisations, sans nécessiter de remise en cause majeure de l'architecture existante.

3.5 Synthèse des besoins

Ce chapitre a permis d'identifier les principaux problèmes rencontrés dans les environnements de recherche industrielle, notamment en matière de fragmentation des données, de collaboration entre équipes, de sécurité et de gestion des métadonnées. À partir de ces constats, un ensemble de besoins fonctionnels et non fonctionnels a été défini afin de guider la conception du système.

Ces besoins mettent en évidence la nécessité d'une solution capable de structurer les ensembles de données, de contrôler finement les accès, de s'intégrer aux environnements analytiques et de fonctionner dans un cadre sécurisé avec une charge d'administration limitée.

Le chapitre suivant présente l'architecture de la solution proposée, conçue pour répondre à ces exigences de manière cohérente et opérationnelle.

CHAPITRE 4

ARCHITECTURE DE LA SOLUTION

4.1 Vision globale de l'architecture

L'architecture du système est organisée de manière modulaire et repose sur plusieurs composants distincts assurant la gestion, le stockage et l'exploitation des données industrielles. Elle comprend une application web centrale, une base de données relationnelle, un stockage de fichiers, une couche d'accès programmatique via API et un environnement analytique basé sur JupyterHub.

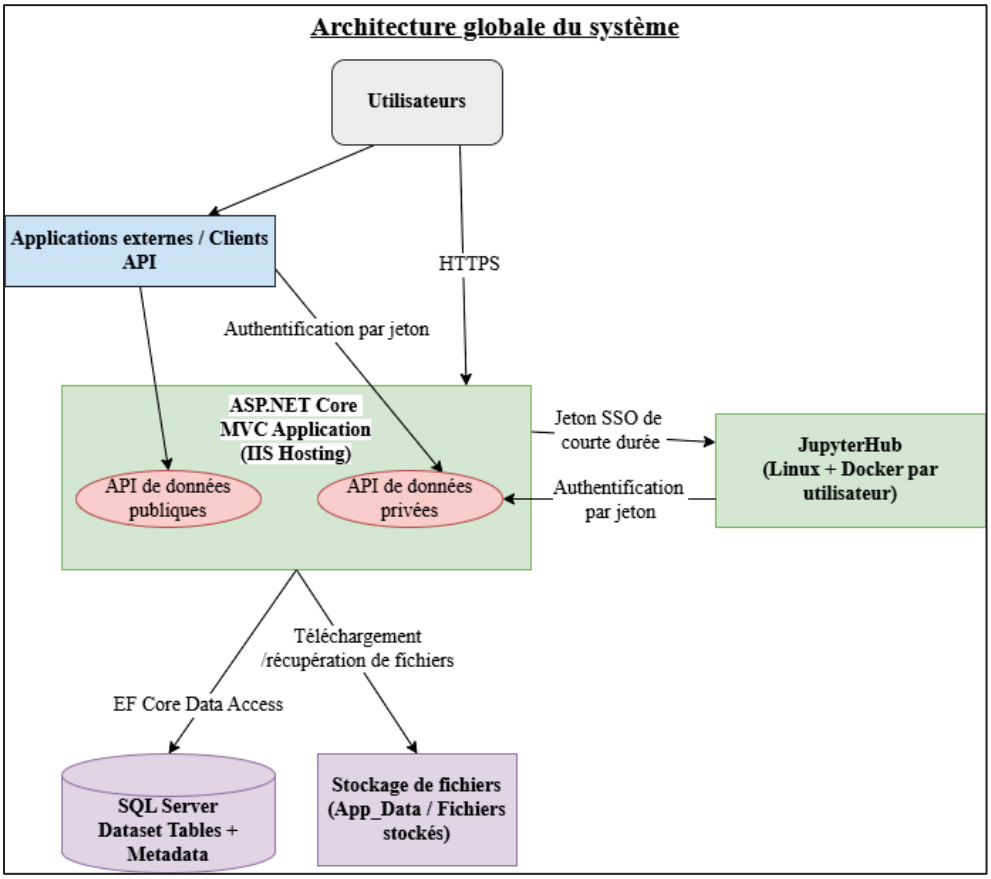

Figure 4.1: Architecture globale du système proposé montrant les principaux composants

L’application web constitue le point d’entrée du système. Elle est développée en ASP.NET Core selon le modèle MVC et assure la gestion des utilisateurs, des ensembles de données et des métadonnées. Elle orchestre également les opérations d’importation des données et la configuration des règles d’accès associées aux ensembles.

Le stockage principal des données est assuré par une base relationnelle SQL Server. Les ensembles de données structurées sont stockés dans des tables distinctes, organisées au sein de schémas séparés selon leur nature, tels que les données principales, les journaux d’événements ou les données contextuelles. Chaque ensemble est associé à une table dédiée. Les métadonnées décrivant les ensembles — incluant leur origine, leur visibilité, leur propriétaire et leurs paramètres d’utilisation — sont stockées dans des entités spécifiques, séparées des données brutes.

En complément du stockage relationnel, le système prend en charge le stockage de fichiers sur le système de fichiers du serveur. Les fichiers sont conservés physiquement sur disque, tandis que leurs informations descriptives et leurs règles d’accès sont gérées dans la base de données relationnelle, assurant une gestion cohérente des droits.

L’accès aux ensembles de données s’effectue par l’intermédiaire d’une couche API dédiée. Cette couche expose des points d’accès permettant la lecture contrôlée des données, sans accès direct aux tables de la base relationnelle. Les requêtes supportent des mécanismes de pagination par curseur, de filtrage et de sélection de colonnes. L’authentification repose sur l’utilisation de jetons temporaires.

L’environnement analytique est basé sur une intégration avec JupyterHub. Celui-ci est déployé sur un environnement Linux, avec une isolation par utilisateur à l’aide de conteneurs. Les environnements utilisateurs sont principalement utilisés pour l’exploration, la préparation et l’analyse des données accessibles via l’API.

La sécurité du système repose sur plusieurs mécanismes complémentaires. L’application web utilise une authentification par session. L’API repose sur des jetons d’accès temporaires, et

l'accès à JupyterHub utilise des jetons à durée de vie encore plus courte. Les autorisations sont déterminées à partir des rôles utilisateurs, des appartenances organisationnelles et des niveaux de visibilité définis pour chaque ensemble de données.

La Figure 4.1 présente une vue globale de l'architecture du système, illustrant les principaux composants, les couches de stockage et les interactions entre l'application web, les interfaces programmatiques et l'environnement analytique.

4.2 Choix technologiques et justification

L'architecture logicielle repose sur un ensemble de technologies sélectionnées afin de répondre aux contraintes d'un environnement de recherche industrielle, notamment en matière de sécurité, de stabilité et d'intégration avec des infrastructures existantes.

L'application centrale est développée en ASP.NET Core, un cadriciel libre de droits qui est destiné à la création d'applications web et d'API. Ce cadriciel offre une architecture modulaire, un modèle d'injection de dépendances intégré et un support à long terme. Dans le cadre de ce projet, son intégration naturelle avec l'écosystème Windows et sa compatibilité avec IIS facilitent les déploiements internes sur des infrastructures existantes. Par ailleurs, son modèle de sécurité mature permet la mise en place de mécanismes d'authentification et d'autorisation adaptés à des contextes industriels.

Le stockage des données structurées repose sur SQL Server, choisi pour sa robustesse transactionnelle et ses mécanismes avancés de gestion des accès. Ce système de gestion de base de données permet une organisation claire des ensembles expérimentaux, l'application de règles de sécurité fines et l'optimisation des performances par l'indexation. Il est également adapté à la gestion de données industrielles structurées et de séries temporelles, fréquemment rencontrées dans les systèmes de supervision.

L'accès aux données est organisé autour d'une architecture orientée API, permettant de dissocier le stockage physique des données de leur exploitation. Cette séparation limite l'exposition directe de la base de données et centralise les règles d'accès au sein de la couche

applicative. Elle facilite également l'intégration avec des outils analytiques externes et contribue à la reproductibilité des analyses en standardisant les modalités d'accès aux ensembles.

Les environnements analytiques sont fournis via une intégration avec JupyterHub, qui permet de déployer des environnements isolés par utilisateur. Cette approche est particulièrement adaptée aux workflows de recherche reposant sur des notebooks, en offrant un cadre contrôlé pour l'exploration et la préparation des données. L'authentification par jetons à durée de vie limitée renforce la sécurité des accès tout en évitant la persistance de privilèges non nécessaires.

Enfin, le système adopte une approche de stockage hybride, combinant une base de données relationnelle pour les données structurées et un stockage de fichiers pour les ressources volumineuses ou non structurées. Les métadonnées associées à l'ensemble des ressources sont centralisées dans la base relationnelle, ce qui permet d'assurer une gouvernance cohérente des accès et une gestion unifiée des ensembles expérimentaux.

4.3 Modèle de gestion des ensembles et métadonnées

La gestion des ensembles de données est au cœur de l'architecture. Le modèle choisi sépare clairement les données opérationnelles des informations descriptives associées. Chaque ensemble est considéré comme une entité logique indépendante.

Lors de l'importation, un ensemble structuré est stocké dans une table dédiée de la base relationnelle. Ce choix améliore les performances d'accès, simplifie l'indexation et réduit les interactions entre ensembles. Les tables sont réparties dans des schémas distincts selon la nature des informations : données principales, journaux d'événements et contexte environnemental.

Les métadonnées sont conservées dans des entités dédiées. Elles décrivent les caractéristiques essentielles de chaque ensemble : description fonctionnelle, visibilité, propriétaire, paramètres

d'activation de l'accès API et propriétés liées à la structure des données. Cette couche descriptive dissocie la gouvernance informationnelle du stockage physique.

Le modèle permet également un contrôle d'accès avancé fondé sur la combinaison de niveaux de visibilité, de rôles utilisateurs et d'appartenance organisationnelle, afin de gérer des ensembles publics, internes ou strictement personnels.

Enfin, la structuration relie systématiquement un ensemble principal à ses informations contextuelles (journaux d'événements, données environnementales), ce qui favorise l'interprétation scientifique. La séparation données/métadonnées facilite aussi l'évolution du système, en limitant les modifications nécessaires sur les structures existantes.

4.4 Architecture de sécurité

L'architecture de sécurité du système est organisée selon une approche multicouche. Elle sépare explicitement les mécanismes d'authentification, d'autorisation et de contrôle d'accès aux ressources, en fonction des points d'entrée et des usages.

4.4.1 Authentification des utilisateurs

L'authentification au portail applicatif repose sur un mécanisme local basé sur l'adresse courriel et un mot de passe. Les mots de passe sont stockés en base de données sous forme hachée. Les sessions utilisateurs sont maintenues à l'aide de témoins d'authentification sécurisés, générés par l'application web. Ces principes s'inscrivent dans les recommandations générales de sécurité applicative visant à limiter l'exposition des secrets d'authentification et à réduire l'impact d'une compromission de données.

4.4.2 Modèle d'autorisation et contrôle d'accès

L'autorisation d'accès aux ressources repose sur un modèle combinant trois dimensions :

- les rôles utilisateurs,
- l'appartenance organisationnelle,

- les niveaux de visibilité associés aux ensembles de données.

Les rôles définissent les capacités de consultation, de modification et d'administration. L'appartenance organisationnelle limite l'accès aux ensembles internes à un périmètre donné. Les niveaux de visibilité permettent de distinguer plusieurs catégories d'ensembles :

- ensembles publics accessibles sans authentification,
- ensembles privés accessibles aux utilisateurs authentifiés,
- ensembles internes accessibles aux membres d'une organisation,
- ensembles personnels accessibles uniquement à leur créateur et aux administrateurs.

Les décisions d'accès sont évaluées dynamiquement lors de chaque requête, à partir des métadonnées associées aux ensembles.

Cette architecture est basée sur le modèle de contrôle d'accès basé sur les rôles (RBAC), dans lequel les permissions sont attribuées en fonction des rôles associés aux utilisateurs. Ce modèle est largement documenté dans les recommandations du *National institute of standards and technology* (NIST), qui décrivent le RBAC comme une approche structurée permettant de simplifier la gestion des droits et de limiter les privilèges aux seules fonctions nécessaires.

4.4.3 Sécurité de l'accès programmatique via l'API

L'accès programmatique aux ensembles de données s'effectue par l'intermédiaire de jetons d'accès générés par les utilisateurs depuis le portail. Ces jetons sont stockés en base de données sous forme hachée et peuvent être révoqués à tout moment. Ils sont transmis via l'en-tête HTTP *Authorization* selon le schéma *Bearer*. Ce mécanisme correspond au modèle standard d'utilisation des jetons défini par la spécification OAuth 2.0 *Bearer Token Usage* (RFC 6750), largement adoptée pour l'authentification des API web.

Au niveau de l'implémentation, chaque requête API fait l'objet d'une extraction explicite du jeton depuis l'en-tête. Il est ensuite analysé et validé côté serveur afin de vérifier son existence, son état, ses permissions et sa période de validité. Cette vérification systématique garantit l'application effective des règles d'authentification définies pour l'accès programmatique.

Les autorisations associées à un jeton sont évaluées en fonction :

- de l'identité de l'utilisateur,
- de ses rôles,
- de son organisation,
- et des règles de visibilité définies au niveau des ensembles.

La documentation fournie aux utilisateurs inclut des exemples concrets d'utilisation de l'API, illustrant l'envoi du jeton d'accès via l'en-tête *Authorization: Bearer* dans les requêtes clientes. Ces exemples renforcent l'usage correct du mécanisme d'authentification et s'inscrivent dans les bonnes pratiques de sécurité associées à l'utilisation des jetons d'accès pour les API HTTP.

Les jetons sont configurés avec une durée de vie maximale de douze heures, au-delà de laquelle toute requête est explicitement rejetée par le serveur. Ce choix s'inscrit dans les recommandations du NIST SP 800-63B, qui préconisent l'utilisation d'informations d'identification à durée de vie limitée afin de réduire les risques liés à l'exposition prolongée ou à la réutilisation de jetons compromis.

4.4.4 Limitation des usages et protection de l'infrastructure

Des mécanismes de limitation de débit et de volume de données transférées sont appliqués aux accès via l'API public ou basé sur des jetons. Les limites sont appliquées au niveau de la couche applicative et visent à prévenir les usages abusifs ou non contrôlés, tout en maintenant un accès fonctionnel aux données.

Ces mécanismes s'appuient sur l'utilisation de codes de statut HTTP normalisés afin de signaler les situations de dépassement de seuil. En particulier, lorsque les limites de requêtes sont atteintes, l'API retourne un code 429 (*Too Many Requests*), accompagné d'un message d'erreur structuré. Ce comportement s'aligne sur la sémantique définie par la norme RFC 9110 pour l'utilisation des codes de statut HTTP, permettant aux clients d'identifier explicitement les conditions de limitation et d'adapter leur comportement en conséquence.

4.4.5 Intégration sécurisée avec l'environnement analytique

L'intégration avec l'environnement analytique repose sur l'utilisation de jetons JWT à durée de vie courte. Ces jetons sont générés par le portail applicatif et utilisés pour l'authentification auprès de l'environnement JupyterHub. Ils contiennent les informations minimales nécessaires à l'identification de l'utilisateur et expirent automatiquement après une courte durée. Aucun jeton à longue durée de vie n'est utilisé.

4.4.6 Traçabilité des accès via l'API

Les accès aux ensembles via l'API font l'objet d'une journalisation systématique. Les journaux incluent notamment :

- l'identifiant du jeton utilisé,
- l'utilisateur associé,
- les ressources consultées,
- les volumes de données transférées.

Ces informations permettent d'assurer une traçabilité des usages, d'identifier des comportements anormaux et de faciliter les analyses de sécurité ou d'audit.

4.5 Architecture d'accès aux données et API

L'accès aux ensembles de données est assuré exclusivement par une couche d'interfaces applicatives intégrée à l'application principale. Aucun accès direct aux systèmes de stockage, qu'il s'agisse de la base relationnelle ou du stockage de fichiers, n'est exposé aux utilisateurs, aux environnements analytiques ou aux outils externes. L'ensemble des interactions programmatiques transite par les API, qui constituent l'unique point d'entrée pour la consultation, l'extraction et l'exportation des données.

Les interfaces applicatives sont implémentées directement au sein de l'application ASP.NET Core. Elles s'appuient sur les mêmes mécanismes internes de validation et de traitement que l'application web, ce qui garantit une cohérence de comportement entre les accès via l'interface

utilisateur et les accès programmatiques. Chaque requête adressée à l'API est analysée et validée côté serveur avant toute interaction avec les systèmes de stockage.

L'accès analytique aux ensembles de données structurées repose sur un modèle de lecture en flux, utilisant une pagination par curseur. Les requêtes spécifient l'ensemble cible, les colonnes à extraire, les filtres applicables et une taille maximale de bloc. Le curseur est généré côté serveur à partir de la dernière clé primaire lue et encode l'état de progression de la lecture. Ce mécanisme permet une reprise déterministe de l'extraction et garantit un ordre de lecture stable, sans recourir à des mécanismes de décalage coûteux sur de grands volumes. Les données sont transmises sous forme de flux Parquet, avec un contrôle strict de la taille maximale des réponses afin d'éviter le chargement complet des ensembles en mémoire.

Les paramètres de requête font l'objet de vérifications systématiques, incluant la validité des colonnes demandées, la cohérence des filtres et le respect des contraintes définies pour l'ensemble ciblé. Des limites explicites sont appliquées sur le nombre de lignes retournées et sur le volume total de données transférées par requête. Ces contraintes sont évaluées dynamiquement et permettent de maintenir un fonctionnement stable de la plateforme lors d'accès concurrents ou prolongés à des ensembles volumineux.

Les informations associées à la pagination et à l'état des réponses sont communiquées au client au moyen d'en-têtes HTTP dédiés, tel que : X-Row-Count, X-Has-More et X-Next-Cursor. L'utilisation des en-têtes pour transporter ces métadonnées s'inscrit dans les principes de la sémantique HTTP, où les en-têtes servent à transmettre des informations structurées relatives à la réponse sans alourdir le corps du message. Cette approche est cohérente avec la norme RFC 9110, qui définit le rôle des en-têtes comme vecteurs de métadonnées pour les réponses HTTP.

Le système prend également en charge des opérations contrôlées de mise à jour ou de remplacement d'ensembles existants. Ces opérations sont réalisées via des processus applicatifs par étapes, reposant sur le transfert de blocs de données structurés, notamment au format Parquet. Les opérations de remplacement sont orchestrées par la couche applicative et

n’autorisent ni l’accès direct aux structures internes de stockage ni la modification des schémas existants au travers des interfaces programmatiques.

Les réponses d’erreur de l’API sont structurées selon le format *Problem Details* en JSON. Ce format correspond à la spécification définie par la norme RFC 7807, qui propose un schéma standardisé pour la représentation des erreurs dans les API HTTP. L’utilisation de ce modèle permet de fournir des messages d’erreur cohérents, lisibles par machine, et interprétables de manière uniforme.

En complément des accès analytiques, l’application propose un canal d’export destiné aux interfaces utilisateurs. Ce canal permet la génération et le téléchargement d’ensembles de données sous forme de fichiers CSV, regroupés et compressés au format ZIP. Les exports sont produits à la demande et suivent les mêmes règles de validation que les accès programmatiques, assurant un comportement cohérent entre les différents modes d’accès aux données.

L’ensemble des accès, qu’ils soient analytiques, programmatiques ou orientés interface utilisateur, repose sur une logique unifiée d’évaluation des règles définies au niveau des métadonnées associées aux ensembles. Chaque requête est traitée indépendamment et validée dynamiquement, garantissant une cohérence fonctionnelle entre les différents usages et une exploitation adaptée aux contraintes des données industrielles issues des systèmes de supervision.

4.6 Synthèse

Ce chapitre a présenté l’architecture globale de la solution, incluant les différents composants du système, les choix technologiques retenus ainsi que les mécanismes de gestion des données et de sécurité. L’approche adoptée repose sur une séparation claire des responsabilités entre les différentes couches du système, favorisant la modularité, la sécurité et la maintenabilité.

Les décisions architecturales décrites permettent de répondre aux besoins identifiés en matière de gouvernance des données, de contrôle d'accès et d'intégration avec les environnements analytiques.

Le chapitre suivant détaille l'implémentation de cette architecture, en présentant l'infrastructure utilisée, l'organisation des données et le déploiement des différents composants du système.

CHAPITRE 5

IMPLÉMENTATION

5.1 Infrastructure et capacités du serveur

L'implémentation de la solution repose sur une infrastructure serveur institutionnelle existante, mise à disposition dans le cadre des ressources informatiques de l'École de technologie supérieure. L'objectif n'était pas de concevoir une infrastructure matérielle dédiée ou optimisée, mais de déployer une plateforme logicielle capable de fonctionner de manière stable et sécurisée dans un environnement contraint, représentatif d'un contexte académique réel.

Tableau 5.1 Caractéristiques de l'infrastructure serveur

Élément	**Description**
Nom du serveur	MEC049250
Environnement	Infrastructure institutionnelle ÉTS
Système d'exploitation	Windows 11 Enterprise (24H2)
Processeur	Intel Xeon E3-1245 v6 (4 cœurs / 8 threads)
Mémoire vive	16 Go
Virtualisation	Activée (WSL2)
Stockage applicatif	Disque local HDD SATA (D:)
Stockage système	SSD NVMe (C:)
SGBD	SQL Server Express Edition (64-bit)
Environnement analytique	JupyterHub (WSL2 + Docker)
Exposition réseau	Accès interne via IIS (HTTPS)

Le Tableau 5.1 présente une synthèse des caractéristiques principales de l'infrastructure serveur utilisée pour le déploiement de la solution.

Le serveur hôte est intégré au domaine institutionnel et fonctionne sous Windows 11 Enterprise (version 24H2). Il dispose de ressources matérielles limitées mais suffisantes pour supporter

simultanément l'application web, la base de données relationnelle et l'environnement analytique, dans un contexte d'utilisation modéré.

5.1.1 Organisation du stockage

Le stockage est réparti entre deux supports physiques distincts. Le disque système SSD NVMe (C:) est réservé au système d'exploitation et aux composants logiciels standards. Les données applicatives, incluant les ensembles importés et les fichiers associés, sont stockées sur un volume HDD SATA distinct (D:). Cette organisation reflète une contrainte de l'infrastructure existante et permet d'isoler les données applicatives du système, sans supposer de performances élevées en entrée/sortie.

5.1.2 Base de données et services applicatifs

La base de données relationnelle repose sur Microsoft SQL Server Express Edition (64-bit), déployée localement sur le serveur. Cette édition impose des limites connues en termes de taille et de ressources, mais fournit un socle transactionnel fiable et cohérent avec un usage de recherche. L'architecture applicative reste compatible avec une migration vers une édition supérieure de SQL Server sans modification structurelle.

L'application web est hébergée via Internet Information Services (IIS), configuré pour un accès interne sécurisé en HTTPS. IIS joue également le rôle de point d'entrée unique pour les services exposés par la plateforme.

5.1.3 Environnement analytique et virtualisation

L'environnement analytique est déployé sur le même serveur physique à l'aide de Windows Subsystem for Linux 2 (WSL2). Un environnement Linux (Ubuntu) est utilisé pour l'exécution de JupyterHub, lequel repose sur Docker pour fournir des environnements isolés par utilisateur. Cette configuration permet de séparer logiquement les environnements analytiques du système hôte tout en conservant un déploiement centralisé.

L'accès à JupyterHub n'est pas exposé directement sur le réseau. Il transite exclusivement par un proxy inverse IIS configuré sur le serveur, assurant une intégration contrôlée avec le reste de la plateforme.

5.1.4 Capacité et contraintes

L'infrastructure impose des contraintes réalistes en matière de ressources, de stockage et d'administration. La solution a été implémentée pour fonctionner de manière stable dans ce cadre, sans dépendre d'une infrastructure haute performance. Cette approche permet d'évaluer le comportement de la plateforme dans des conditions proches de celles rencontrées dans de nombreux environnements de recherche appliquée, tout en conservant la possibilité d'une montée en charge ultérieure sur des infrastructures plus puissantes.

5.2 Architecture logique et physique de la base de données

Cette section décrit l'organisation logique des ensembles de données et leur implémentation physique au sein du système de gestion de base de données relationnelle. Elle précise la séparation entre données et métadonnées, les relations entre les entités principales et la structuration physique retenue sous SQL Server.

5.2.1 Modèle logique des ensembles de données

L'unité logique centrale du système est l'ensemble de données, représenté par une entité de métadonnées. Cette entité constitue le point d'ancrage de la gouvernance et de l'exploitation, indépendamment du format physique des données stockées.

Le modèle distingue explicitement deux catégories d'informations :

- les **données**, correspondant aux mesures ou informations opérationnelles,
- les **métadonnées**, décrivant le contexte, la provenance, la visibilité et les conditions d'utilisation.

Les métadonnées sont centralisées dans l'entité Métadonnée. Celle-ci référence l'ensemble principal et, le cas échéant, ses compléments contextuels. Elle conserve également les liens

vers les entités de gouvernance, notamment le propriétaire, l'organisation, la visibilité et la licence associée.

Les données opérationnelles sont réparties en trois catégories logiques :

- **Donnees** : données principales associées à l'ensemble,
- **DonneesEventLogs** : journaux d'événements ou d'alarmes,
- **DonneesContexteEnvironnemental** : données environnementales ou contextuelles.

Chaque catégorie reste distincte sur le plan logique, tout en étant rattachée à la même entité de métadonnées. Cette organisation permet d'associer explicitement données principales et contexte, sans fusionner des contenus de nature différente.
Les relations clés du modèle sont les suivantes :

- Métadonnée → Utilisateur (propriétaire de l'ensemble),
- Métadonnée → Visibilite (règles d'accès),
- Métadonnée → Licence (conditions d'utilisation),
- Métadonnée → Donnees (*relation one-to-one*),
- Metadonnee → DonneesEventLogs *(relation one-to-one* optionnelle),
- Metadonnee → DonneesContexteEnvironnemental (*relation one-to-one* optionnelle).

Ce découplage permet de faire évoluer le contenu des ensembles sans remettre en cause leur gouvernance ni leur identité logique.

5.2.2 Implémentation physique des données sous SQL Server

Sur le plan physique, chaque ensemble de données est matérialisé par des tables relationnelles distinctes dans SQL Server. Il n'existe pas de table générique unique pour l'ensemble des données analytiques. Chaque importation conduit à la création d'une table dédiée dans le schéma approprié, ce qui facilite l'isolation logique, l'indexation ciblée et la maintenance.

Les tables analytiques incluent systématiquement une clé primaire unique, utilisée comme référence pour la navigation et la pagination par curseur dans les accès programmatiques. Des

champs dédiés à l'indexation sont présents afin de supporter différents scénarios d'accès analytique, notamment :

- colonne temporelle principale,
- identifiant d'inclusion ou de regroupement,
- type et état de l'index appliqué.

L'indexation reste optionnelle et configurable par ensemble, ce qui permet d'adapter les performances aux usages réels sans imposer de structure rigide à l'ensemble des données.

Les données sont stockées sur un support HDD local, conformément aux contraintes de l'infrastructure existante. Cette caractéristique n'impacte pas le modèle logique et n'introduit aucune dépendance forte au support physique, permettant une évolution ultérieure vers des solutions de stockage plus performantes.

5.2.3 Implémentation physique des fichiers sous SQL Server

Certaines ressources ne peuvent pas être représentées sous forme de tables relationnelles. Ces contenus sont gérés via une entité dédiée, FichierStocke, qui permet de traiter les fichiers comme des ressources de première classe au sein du système.

Physiquement, les fichiers sont stockés sur le système de fichiers du serveur, dans un espace contrôlé. Logiquement, toutes les informations nécessaires à leur gouvernance sont conservées en base relationnelle, notamment :

- nom du fichier stocké et nom d'origine,
- taille,
- propriétaire,
- visibilité
- licence associée.

Cette approche hybride permet d'intégrer des ressources hétérogènes tout en conservant un modèle de gouvernance homogène avec les ensembles tabulaires.

5.2.4 Organisation des schémas et tables applicatives

Le modèle physique repose sur une séparation explicite des schémas SQL afin de distinguer les responsabilités applicatives des contenus analytiques.Le schéma *dbo* regroupe les tables applicatives et de gouvernance, incluant notamment :

- Utilisateur, Role, Entreprise pour la gestion des identités,
- Metadonnee, Visibilite, Licence, Site, TypeEnergieRenouvelable pour la gouvernance des ensembles,
- NotebookApiToken et NotebookApiAccessLog pour l'accès programmatique et la journalisation,
- FichierStocke pour les ressources non tabulaires.

Les schémas analytiques (donnees, donnees_event_logs, donnees_contexte_environnemental) contiennent les tables dynamiques créées lors de l'importation des ensembles. Cette séparation permet de maintenir une frontière claire entre la logique applicative et les données analytiques, tout en facilitant l'administration et l'évolution du schéma global.

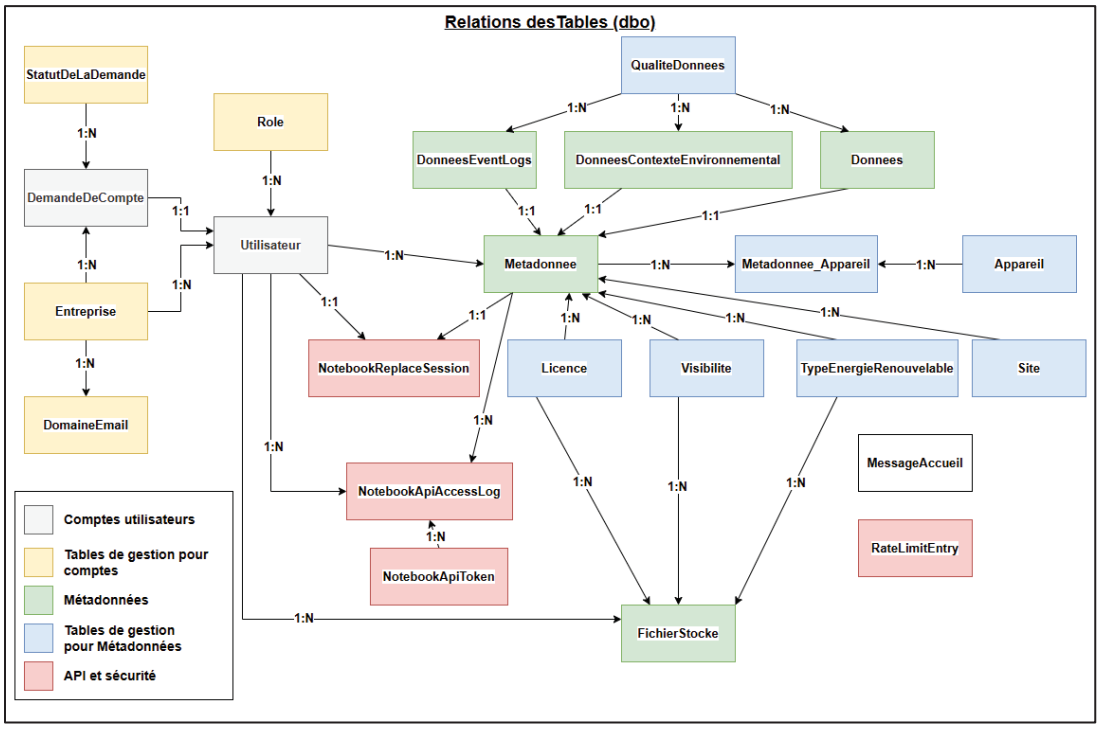

Figure 5.1 Relations entre les tables

La Figure 5.1 présente une vue synthétique des relations entre les tables, illustrant l'articulation entre gouvernance logique et stockage physique.

5.3 Déploiement applicatif

Le déploiement de la solution a été réalisé sur une infrastructure institutionnelle existante. Le mode de déploiement retenu vise la fiabilité, la reproductibilité et la simplicité opérationnelle, sans dépendre d'outils externes complexes ni de pipelines industriels automatisés. L'ensemble de la configuration est compatible avec un environnement de recherche appliquée soumis à des contraintes de sécurité et de maintenance limitées.

5.3.1 Hébergement de l'application sous IIS

L'application principale est hébergée sur le serveur MEC049250 via Internet Information Services (IIS). Un site IIS dédié, nommé Dataportal, est configuré avec un répertoire physique distinct situé sur le volume applicatif du serveur.

L'application est associée à un pool d'applications spécifique, exécuté sous une identité Windows dédiée (MEC049250\DataportalUser). L'exécution sous un compte explicite permet un contrôle précis des droits d'accès au système de fichiers et à la base de données.

L'exposition réseau du site est volontairement restreinte. Le portail est accessible exclusivement via HTTPS, à l'aide d'un certificat auto-signé, dans un périmètre interne au réseau de l'ÉTS. Aucun point d'accès HTTP non chiffré n'est activé.

5.3.2 Configuration applicative et gestion des paramètres

La configuration de l'application repose sur les mécanismes standards fournis par ASP.NET Core. Les paramètres sensibles, notamment la chaîne de connexion à la base de données, sont injectés au niveau du site IIS via des variables d'environnement. Cette approche évite toute présence de secrets dans le code source ou dans les fichiers de configuration versionnés.

La connexion à SQL Server utilise exclusivement l'authentification Windows intégrée (Integrated Security). Aucun identifiant SQL ni mot de passe n'est stocké ou manipulé par l'application. L'environnement d'exécution est configuré en mode Production, ce qui active les optimisations du cadriciel et désactive les fonctionnalités de diagnostic non nécessaires en exploitation.

5.3.3 Modèle de droits et séparation des responsabilités

Le modèle de déploiement repose sur une séparation stricte des responsabilités entre le système d'exploitation, l'application et le système de gestion de base de données.

L'identité Windows du pool d'applications est déclarée comme login dans SQL Server, puis mappée à un utilisateur de base de données disposant de droits explicitement limités. L'application est autorisée à effectuer des opérations de lecture et d'écriture sur les tables applicatives du schéma principal (dbo). Elle ne dispose d'aucun privilège global de modification structurelle.

Les opérations de création, de modification et de suppression de tables sont restreintes aux schémas dédiés aux ensembles importés. Cette séparation limite l'impact potentiel d'une défaillance applicative et empêche toute altération involontaire de la structure globale de la base.

5.3.4 Processus de déploiement et de mise à jour

Le cycle de déploiement suit une procédure volontairement simple et reproductible. Le code source est récupéré depuis le dépôt Git, puis compilé et publié à l'aide de la commande dotnet publish. Les fichiers générés sont placés dans un répertoire de build versionné par horodatage, permettant de conserver un historique des versions déployées.

Les évolutions du schéma de la base de données sont appliquées séparément à l'aide d'Entity Framework Core, soit via la ligne de commande, soit par l'exécution manuelle de scripts SQL. Cette étape explicite permet de contrôler précisément les modifications structurelles.

Le déploiement sur IIS consiste à mettre temporairement le site hors ligne, synchroniser les fichiers applicatifs vers le répertoire cible, puis réactiver le site. Les répertoires contenant les données persistantes et les journaux applicatifs sont exclus du processus afin d'éviter toute perte d'information.

5.3.5 Déploiement et intégration de l'environnement analytique

L'environnement analytique est déployé sur le même serveur physique à l'aide de Windows Subsystem for Linux 2 (WSL2). Un environnement Linux dédié est utilisé pour exécuter JupyterHub, lequel s'appuie sur Docker pour fournir des environnements isolés par utilisateur.

JupyterHub n'est jamais exposé directement sur le réseau. Il écoute uniquement sur l'interface locale et est accessible exclusivement via un proxy inverse IIS configuré en HTTPS.

L'authentification vers JupyterHub est entièrement orchestrée par l'application principale, qui agit comme fournisseur d'identité unique. Des jetons JWT à durée de vie courte sont générés dynamiquement pour initier les sessions analytiques, assurant une cohérence complète entre le portail applicatif et les environnements de notebooks.

5.4 Modèle des rôles et autorisations

Le modèle d'autorisations repose sur la combinaison de trois dimensions orthogonales : les rôles utilisateurs globaux, les niveaux de visibilité associés aux ressources, et la propriété des ensembles. Les décisions d'accès sont évaluées de manière centralisée dans la logique applicative et s'appliquent uniformément à l'interface web, aux accès programmatiques et aux mécanismes de téléchargement.

5.4.1 Rôles utilisateurs globaux

Chaque utilisateur est associé à un rôle unique à l'échelle de l'application. Ce rôle est défini lors de la création du compte et s'applique globalement, indépendamment des ensembles de données manipulés. Les rôles ne sont pas contextuels à une ressource particulière.
Les rôles définis sont les suivants :

- **Observateur** : accès en lecture aux ressources autorisées par leur niveau de visibilité.
- **Utilisateur** : accès en lecture étendu et possibilité d'interagir avec les interfaces analytiques.
- **Éditeur** : capacité de créer des ensembles et de modifier les ressources dont il est propriétaire.
- **Administrateur** : accès complet aux ressources et aux fonctions de gouvernance.

Ces rôles sont codifiés dans l'application et utilisés comme critère principal pour autoriser ou restreindre certaines opérations, notamment la création d'ensembles, la modification des métadonnées et l'administration globale.

5.4.2 Niveaux de visibilité des ensembles

L'accès aux ensembles de données est également conditionné par un niveau de visibilité explicitement associé à chaque ressource. Ces niveaux sont définis de manière déclarative dans les métadonnées de l'ensemble.
Quatre niveaux de visibilité sont supportés :

- **Public** : accessible sans authentification.
- **Privé** : accessible à tout utilisateur authentifié.
- **Interne** : accessible aux administrateurs et aux utilisateurs appartenant à la même organisation que le créateur.
- **Personnel** : accessible uniquement au créateur de l'ensemble et aux administrateurs.

Le niveau de visibilité est évalué dynamiquement lors de chaque requête et constitue un filtre préalable à toute opération d'accès.

5.4.3 Règles effectives d'accès aux ensembles

Les décisions d'accès aux ensembles sont prises par une fonction de validation centralisée, qui combine les trois dimensions suivantes :

- le rôle global de l'utilisateur,
- le niveau de visibilité de l'ensemble,
- l'appartenance organisationnelle et la propriété de la ressource.

Cette logique permet d'exprimer des règles d'accès précises sans multiplier les autorisations par ressource. Les règles sont évaluées de manière identique pour l'ensemble des points d'entrée du système, garantissant un comportement cohérent entre interface web, API analytiques et mécanismes d'export.

5.4.4 Gouvernance et modification des métadonnées

Les métadonnées associées à un ensemble constituent l'élément central de la gouvernance. Leur modification est strictement encadrée afin de préserver la cohérence descriptive des ressources.

Les règles de modification sont les suivantes :

- **Administrateur** : autorisé à modifier l'ensemble des métadonnées, quel que soit l'ensemble.
- **Propriétaire (créateur)** : autorisé à modifier les métadonnées des ensembles qu'il a créés.
- **Autres rôles** : aucun droit de modification des métadonnées.

Ce modèle garantit que les informations descriptives évoluent de manière contrôlée, tout en permettant des mises à jour lorsque cela est nécessaire.

5.4.5 Synthèse du modèle d'autorisations

Le modèle d'autorisations repose sur trois principes structurants :

1. Rôles globaux simples, appliqués uniformément à l'échelle de l'application.
2. Visibilité déclarative, attachée aux ressources plutôt qu'aux utilisateurs.

3. Propriété explicite, utilisée comme critère principal pour autoriser les opérations de modification.

Cette organisation permet de concilier sécurité, gouvernance et simplicité opérationnelle, tout en restant compatible avec une évolution vers des environnements multiorganisationnels plus larges.

5.5 Implémentation des mécanismes principaux

Cette section décrit les mécanismes fonctionnels centraux implémentés dans la solution, en mettant l'accent sur les workflows applicatifs et les traitements internes. L'objectif est d'illustrer comment l'architecture décrite précédemment se matérialise concrètement dans le fonctionnement du système, sans entrer dans le détail ligne par ligne du code source.

5.5.1 Parcours d'authentification et gestion des sessions

Le parcours d'authentification repose sur un mécanisme local entièrement géré par l'application.

5.5.1.1 Workflow de création de comptes

L'inscription débute par une demande de compte, soumise via l'interface applicative. Cette demande inclut les informations d'identification de base et déclenche la génération d'un jeton de vérification d'adresse courriel. La demande de compte est créée dans un état en attente, empêchant toute création tant que l'adresse n'a pas été validée. La validation s'effectue via un lien contenant le jeton temporaire. Une fois ce jeton vérifié, la demande passe à l'état actif. Un administrateur doit par la suite l'approuver pour que le compte soit créé et que l'utilisateur puisse se connecter.

5.5.1.2 Connexion et sécurité de session

La connexion s'effectue à l'aide d'une adresse courriel et d'un mot de passe, dont le code de hachage est stocké en base de données. Un mécanisme de verrouillage est appliqué après un nombre maximal d'échecs consécutifs. Le compte est alors bloqué pour une durée définie. Une

authentification multifactorielle optionnelle est supportée. Lorsqu'elle est activée, un code à usage unique est généré, stocké sous forme hachée avec une date d'expiration, puis transmis par courriel. La session n'est ouverte qu'après validation de ce code. La session utilisateur est maintenue via un témoin d'authentification sécurisé.

```
# Parcours d'authentification utilisateur
def authentifier(email, mot_de_passe):
    if compte_est_verrouille(email):
        return "REFUSE"

    if verifier_hash(email, mot_de_passe):
        if mfa_active(email):
            code = generer_code()
            envoyer_email(email, code)
            return "MFA_REQUIS"
        else:
            ouvrir_session(email)
            return "OK"

    incrementer_echecs(email)
    if seuil_atteint(email):
        verrouiller_compte(email)
    return "REFUSE"
```

5.5.2 Ingestion et création des ensembles de données

L'ingestion des ensembles constitue un mécanisme structurant du système.

5.5.2.1 Les formats pris en charge

- CSV,
- Excel (XLS/XLSX),
- Parquet,
- archives ZIP contenant des fichiers CSV homogènes.

5.5.2.2 Workflow d'importation

L'importation s'effectue par étapes successives :

1. Téléversement des fichiers vers un espace temporaire serveur.
2. Inférence de la structure : détection des colonnes, types et contraintes.
3. Validation utilisateur de la structure proposée.

4. Création dynamique d’une table SQL cible dans le schéma approprié.
5. Insertion massive des données via des mécanismes d'insertion en lot.
6. Création des métadonnées et association logique entre ensemble et table physique.

L’ensemble du traitement est réalisé de manière synchrone afin de simplifier la gestion des erreurs et le retour utilisateur. Une limite stricte de taille par étape est appliquée afin d’éviter les imports excessivement longs.

```
# Importation d’un ensemble tabulaire
def importer_ensemble(fichiers):
    colonnes = inferer_colonnes(fichiers)
    if colonnes_invalides(colonnes):
        return "ERREUR_SCHEMA"

    table = creer_table_sql(colonnes)
    inserer_donnees(table, fichiers)
    metadonnee = creer_metadonnee(table)
    return metadonnee
```

5.5.3 Indexation et optimisation des accès

Le système propose un mécanisme d’indexation configurable par ensemble. Lors de la préparation d’un ensemble, l’utilisateur peut indiquer les colonnes pertinentes pour l’analyse, notamment :

- colonne temporelle,
- colonne identifiant,
- colonne utilisée pour des filtres d’inclusion,
- colonne très utilisée.

La demande d’indexation est enregistrée avec un état (*pending*). La création effective des index est réalisée de manière différée par un service de maintenance exécuté hors période de charge.

Un service planifié analyse régulièrement les demandes en attente et déclenche la création des index SQL correspondants. Les états (*pending, running, completed, failed*) sont mis à jour afin de suivre l'évolution du traitement.

```
# Service de maintenance des index
def traiter_indexations():
    ensembles = recuperer_index_pending()
    for e in ensembles:
        if remplacement_en_cours(e):
            continue
        creer_index_sql(e)
        marquer_index_complet(e)
```

5.5.4 Accès analytique et API de lecture

L'accès analytique aux ensembles est réalisé exclusivement via des interfaces programmatiques. Les API de lecture fournissent les données sous forme de flux Parquet, adaptés aux traitements analytiques. La navigation repose sur une pagination par curseur basée sur la clé primaire, garantissant un ordre de lecture déterministe et évitant les limitations des approches par offset. Chaque requête :

1. valide les paramètres,
2. décode le curseur,
3. extrais un bloc de données,
4. encode le curseur suivant,
5. transmets les données en flux.

Des limites strictes sur le nombre de lignes et le volume transféré sont appliquées.

```
# Lecture analytique paginée
def lire_par_bloc(dataset, curseur, limite):
    cle = decoder_curseur(curseur)
    lignes = select_bloc(dataset, cle, limite + 1)
    prochain = encoder_curseur(lignes)
    return stream_parquet(lignes[:limite]), prochain
```

Ces API sont utilisées aussi bien par les notebooks JupyterHub que par des outils analytiques externes autorisés.

5.5.5 Mise à jour contrôlée du contenu des ensembles

La modification directe des données est volontairement restreinte.
Aucune mise à jour ligne par ligne ni modification de schéma n'est exposée via les API.
Les mises à jour s'effectuent exclusivement via un processus de remplacement contrôlé, organisé en sessions :

1. **Démarrage de session** : verrouillage de l'ensemble et création d'une table de *staging*.
2. **Téléversement par blocs** : validation du schéma et insertion progressive.
3. **Commit** : échange transactionnel entre table de *staging* et table active.
4. **Finalisation** : suppression de l'ancienne table et déverrouillage.

```
# Remplacement transactionnel d'un ensemble
def remplacer_ensemble(dataset, fichiers):
    verrouiller(dataset)
    staging = creer_table_staging()
    inserer_donnees(staging, fichiers)
    echanger_tables(dataset, staging)
    supprimer_ancienne()
    deverrouiller(dataset)
```

Les droits d'accès à ces opérations sont limités aux administrateurs et aux éditeurs propriétaires.

5.5.6 Export et interopérabilité

Un mécanisme d'export est fourni pour les usages non analytiques. Les ensembles peuvent être exportés sous forme de fichiers CSV découpés et compressés au format ZIP, accompagnés d'un manifeste descriptif. Les règles de visibilité et d'autorisation sont évaluées avant tout export.

5.5.7 Synthèse des mécanismes implémentés

Cette section a présenté l'implémentation des principaux mécanismes du système, incluant l'authentification, la gestion des ensembles de données, l'accès analytique via API ainsi que les processus d'importation, de mise à jour et d'exportation. Ces mécanismes traduisent concrètement les choix architecturaux définis précédemment.

L'ensemble de ces éléments permet d'assurer un fonctionnement cohérent du système, en respectant les contraintes de sécurité, de performance et de gouvernance des données.

Le chapitre suivant est consacré à la validation fonctionnelle de la solution, à travers différents scénarios d'utilisation visant à vérifier sa conformité aux besoins identifiés.

CHAPITRE 6

VALIDATION FONCTIONNELLE

6.1 Validation fonctionnelle par scénarios de rôles

La validation fonctionnelle du système s'appuie sur des scénarios d'utilisation structurés autour des rôles utilisateurs définis dans l'application. Quatre rôles sont considérés: observateur, utilisateur, éditeur et administrateur. Chacun correspond à un niveau distinct de privilèges et de fonctionnalités accessibles. Le Tableau 6.1 résume certaines des actions possibles disponibles pour chaque rôle.

Tableau 6.1 Matrice des rôles et des capacités

Action / Rôle	Observateur public	Observateur autorisé	Utilisateur	Éditeur	Administrateur
Afficher les ensembles de données publics	✓	✓	✓	✓	✓
Afficher les ensembles de données privés	×	✓	✓	✓	✓
Afficher les ensembles de données internes	×	×	✓	✓	✓
Afficher les ensembles de données personnels	×	×	✓(Propres)	✓(Propres)	✓
Créer un ensemble de données	×	×	✓(Personnels)	✓	✓
Modifier les métadonnées	×	×	✓(Propres)	✓(Propres)	✓
Accéder à JupyterHub	×	×	×	✓	✓
Gérer les utilisateurs	×	×	×	✓(Pas d'action)	✓

6.1.1 Scénarios associés au rôle d'observateur public

Le rôle d'observateur correspond à un utilisateur non authentifié. Il sert à valider la diffusion contrôlée des ensembles publics tout en garantissant l'isolement des ressources restreintes.

L'observateur peut consulter et filtrer les ensembles dont la visibilité est publique depuis l'interface de recherche. Il peut accéder à la fiche descriptive d'un ensemble public et visualiser ses informations descriptives. Les fichiers publics peuvent être consultés et, si les métadonnées l'autorisent, téléchargés. L'observateur peut également accéder aux points d'accès programmatiques publics lorsque l'ensemble est explicitement marqué comme accessible par API.
Toute tentative d'accès à un ensemble privé, interne ou personnel entraîne une demande d'authentification ou un refus explicite. L'observateur ne peut pas créer, modifier ou supprimer des ensembles ni des fichiers. Il n'a pas accès aux fonctionnalités d'importation, d'édition de métadonnées, d'indexation ni à l'environnement analytique.

6.1.2 Scénarios associés au rôle d'observateur authentifié

Le rôle d'observateur authentifié correspond à un utilisateur disposant d'un compte valide et connecté au portail applicatif. Ce rôle étend les capacités de l'observateur public en autorisant l'accès aux ensembles privés.

6.1.3 Scénarios associés au rôle d'utilisateur

Le rôle d'utilisateur correspond à un compte authentifié disposant de privilèges limités. Un utilisateur peut créer de nouveaux ensembles via le processus d'importation avec une visibilité personnelle. Il peut modifier et supprimer les ensembles dont il est propriétaire, ainsi que gérer les métadonnées associées dans les limites autorisées. Il peut également consulter et télécharger les ensembles accessibles selon les règles de visibilité, incluant les ensembles publics, privés, internes à son organisation et personnels.

Les fichiers peuvent être modifiés ou supprimés uniquement lorsqu'ils ont été déposés par l'utilisateur lui-même, la propriété étant vérifiée avant d'autoriser ces opérations. En revanche, l'utilisateur ne peut pas publier un ensemble avec une visibilité étendue ni accéder aux fonctions d'administration globale.

6.1.4 Scénarios associés au rôle d'éditeur

Le rôle d'éditeur correspond à un utilisateur disposant de privilèges étendus pour la préparation et la gestion des ensembles, sans accès aux fonctions d'administration du système.

Un éditeur peut créer des ensembles avec plusieurs niveaux de visibilité, configurer des options d'indexation lors de l'importation et gérer les métadonnées de ses propres ensembles. Il peut suivre les opérations d'indexation et gérer les fichiers associés. Il a également accès à l'environnement analytique, et peut interagir avec les ensembles via les interfaces programmatiques. Les opérations de remplacement de contenu sont permises pour ses propres ensembles, sous réserve du respect des règles de propriété et de visibilité.

L'éditeur ne peut pas approuver des comptes, modifier les rôles des utilisateurs ou administrer les organisations et domaines.

6.1.5 Scénarios associés au rôle d'administrateur

Le rôle d'administrateur correspond au niveau de privilège maximal. Un administrateur peut accéder à l'ensemble des ressources indépendamment de leur visibilité et de leur propriétaire. Il peut créer, modifier et supprimer des ensembles et des fichiers, superviser les opérations d'indexation et les mécanismes de remplacement de contenu. Il est responsable de la gestion des comptes : approbation des demandes, activation/désactivation, attribution des rôles. Il peut également administrer les entités organisationnelles et les domaines associés.

Certaines protections administratives restent actives afin d'éviter des incohérences (ex. restrictions sur la désactivation d'un administrateur actif ou la suppression d'une organisation encore utilisée).

6.2 Validation du workflow analytique

Cette section valide le fonctionnement du système à travers un workflow analytique complet, depuis la création d'un ensemble de données jusqu'à son exploitation dans un environnement de notebooks, puis sa mise à jour contrôlée. L'objectif est de démontrer que les mécanismes décrits aux chapitres précédents permettent de supporter des usages analytiques réels, tout en respectant les contraintes de gouvernance, de sécurité et de reproductibilité.

Le workflow présenté repose exclusivement sur des comportements implémentés dans l'application et se décompose en cinq étapes principales : importation des ensembles, indexation optionnelle, accès à l'environnement analytique, accès programmatique aux données et mise à jour contrôlée du contenu.

6.2.1 Importation et création des ensembles de données

La création d'un ensemble débute par un processus d'importation structuré sous la forme d'un assistant multiétapes. Les formats supportés incluent les fichiers CSV, Excel (XLS/XLSX), Parquet, ainsi que des archives ZIP contenant des fichiers CSV homogènes. Tous les fichiers associés à un même import doivent respecter un format identique.

La première étape consiste à définir les métadonnées générales de l'ensemble, notamment son nom, sa description, sa licence et son niveau de visibilité. Les options disponibles sont restreintes en fonction du rôle de l'utilisateur afin de garantir le respect des règles de gouvernance.

L'étape suivante concerne l'importation de la table principale de données, qui est obligatoire. Les fichiers soumis sont analysés afin d'inférer automatiquement la structure des colonnes et les types de données. L'utilisateur peut valider ou ajuster les types proposés. Des validations strictes sont appliquées, incluant l'unicité des en-têtes de colonnes, la cohérence des structures entre fichiers et la compatibilité des valeurs avec les types sélectionnés. Les erreurs de conversion sont détectées et consignées avant validation finale.

Deux étapes optionnelles permettent d'importer des ensembles complémentaires correspondant aux journaux d'événements et au contexte environnemental. Ces étapes suivent la même logique de validation et d'importation que la table principale.
Une fois l'import validé, une table SQL cible est créée dynamiquement dans le schéma approprié. Une clé primaire technique est ajoutée automatiquement afin de permettre la navigation par curseur. Les enregistrements de métadonnées sont ensuite créés et liés à la table physique, assurant la cohérence entre le stockage des données et leur gouvernance logique.

6.2.2 Indexation optionnelle des ensembles

Afin d'optimiser les accès analytiques, le système propose un mécanisme d'indexation configurable. Lors de la création de l'ensemble, les utilisateurs disposant des droits requis peuvent activer l'indexation et sélectionner des colonnes spécifiques, notamment une colonne temporelle, une colonne pour l'identifiant et, le cas échéant, une colonne d'inclusion utilisée pour des filtres analytiques.
L'activation de l'indexation n'entraîne pas immédiatement la création des structures physiques. Une demande d'indexation est enregistrée et associée à un statut indiquant qu'un traitement est requis. La création effective des index est prise en charge de manière différée par un service de maintenance dédié, qui traite les demandes en attente et met à jour leur état en fonction du résultat.

Ce fonctionnement permet de dissocier les opérations coûteuses de l'interaction utilisateur et de préserver la stabilité globale du système, tout en offrant des capacités d'optimisation adaptées aux analyses temporelles et séquentielles.

La Figure 6.2 illustre les différents points d'entrée permettant de demander une indexation, soit lors de la création ou de l'importation d'un jeu de données, soit via une requête manuelle depuis l'interface utilisateur. Les contrôles d'autorisation (rôle et propriété) ainsi que les conditions d'éligibilité (jeu de données de type séries temporelles, indexation activée) sont évalués avant l'enregistrement des métadonnées d'index et la mise à l'état en attente.

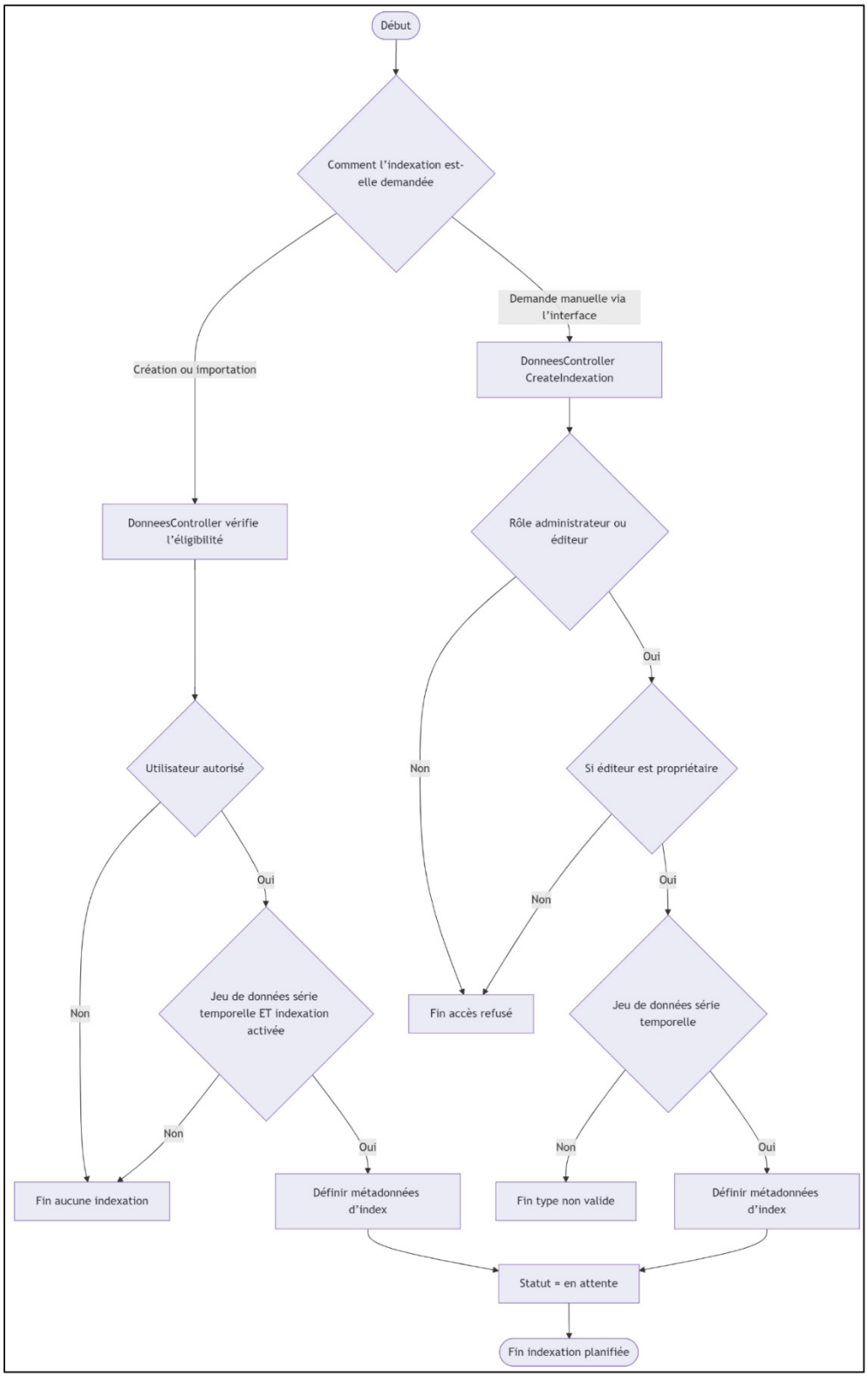

Figure 6.1 Processus de demande et de planification de l'indexation

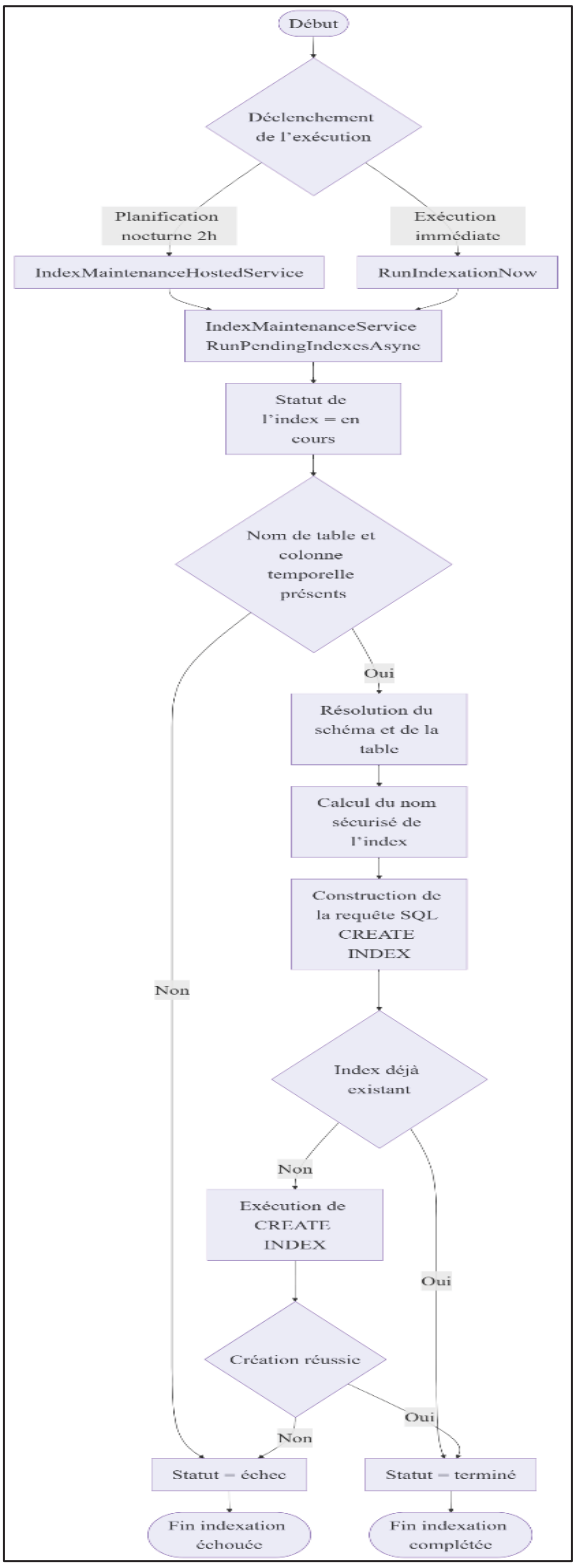

Figure 6.2 Processus d'exécution de l'indexation

Alors que la Figure 6.3 présente le processus d'exécution de l'indexation, déclenché soit automatiquement par un service planifié, soit manuellement par un utilisateur autorisé. Le service de maintenance applique successivement les validations nécessaires, génère la requête SQL de création d'index et met à jour l'état de l'indexation selon le résultat de l'exécution (terminée ou échouée). Ce mécanisme assure une exécution contrôlée et traçable de l'indexation.

6.2.3 Accès à l'environnement analytique

L'exploitation analytique des ensembles s'effectue via un environnement de notebooks basé sur JupyterHub. L'accès à cet environnement est réservé aux rôles disposant des privilèges requis.

Lorsqu'un utilisateur autorisé lance l'environnement analytique, l'application génère un jeton d'authentification à durée de vie courte et redirige le navigateur vers JupyterHub. Ce mécanisme de connexion unique permet d'authentifier l'utilisateur sans exposer d'identifiants persistants.

Une fois la session établie, l'environnement de notebooks est isolé par utilisateur et ne dispose d'aucun accès direct au système de gestion de base de données. Toutes les interactions avec les ensembles transitent exclusivement par les interfaces programmatiques de l'application, garantissant que les règles de visibilité et d'autorisation sont systématiquement appliquées, y compris dans le contexte analytique.

6.2.4 Accès programmatique aux données depuis les notebooks

L'accès aux ensembles de données depuis les notebooks repose sur des interfaces programmatiques fournissant les données sous forme de flux Parquet. Deux grandes familles d'API sont utilisées selon le contexte : des API génériques accessibles depuis des environnements externes et des API spécifiques à l'environnement JupyterHub.

Les API publiques permettent un accès en lecture aux ensembles dont la visibilité est publique et pour lesquels l'accès programmatique est explicitement autorisé. Les API privées

nécessitent l'utilisation d'un jeton d'accès et vérifient systématiquement les droits de l'utilisateur sur l'ensemble ciblé.

Dans le contexte de JupyterHub, des API dédiées peuvent être utilisées. Elles permettent un accès non indexé pour des lectures séquentielles simples, ainsi que des accès indexés lorsque l'indexation a été activée. Les accès indexés supportent des filtres temporels et, le cas échéant, des filtres d'inclusion supplémentaires.

La pagination repose sur un mécanisme par curseur basé sur la clé primaire, garantissant un ordre de lecture déterministe et évitant les limitations associées aux requêtes par décalage. Chaque réponse inclut les informations nécessaires pour poursuivre la lecture de manière contrôlée.

Ce modèle permet d'intégrer efficacement les ensembles dans des workflows analytiques tout en limitant les volumes transférés et en assurant la stabilité de l'infrastructure.

6.2.5 Mise à jour contrôlée du contenu des ensembles

La mise à jour du contenu des ensembles structurés s'effectue via un mécanisme de remplacement contrôlé, destiné aux usages analytiques avancés. Ce processus repose sur des sessions de remplacement gérées par l'application.

Lorsqu'une session est initiée, une table intermédiaire est créée et l'ensemble concerné est temporairement verrouillé afin d'éviter les accès concurrents. Les nouvelles données sont transférées par blocs successifs au format Parquet, validées puis insérées dans la table intermédiaire. Une fois l'ensemble des blocs reçus, l'utilisateur peut valider le remplacement, ce qui déclenche une permutation atomique des tables et la suppression de l'ancienne version.

Ce mécanisme garantit l'intégrité des données et permet des mises à jour complètes sans exposer de fonctionnalités de modification fine ou de changement de schéma. Les droits d'accès à ces opérations sont strictement limités aux administrateurs et aux éditeurs propriétaires des ensembles concernés.

6.2.6 Synthèse du workflow analytique

La validation du workflow analytique montre que le système couvre l'ensemble du cycle de vie des données, depuis leur ingestion jusqu'à leur exploitation et leur mise à jour, dans un cadre gouverné et contrôlé. Les mécanismes d'importation, d'indexation différée, d'accès via notebooks et d'accès programmatique s'articulent de manière cohérente pour soutenir des analyses reproductibles, tout en respectant les contraintes opérationnelles identifiées.

6.3 Validation de la couverture des besoins

Cette section met en correspondance les besoins fonctionnels et non fonctionnels identifiés au chapitre 3 avec les mécanismes effectivement implémentés et validés dans la solution développée.

L'analyse s'appuie exclusivement sur les scénarios d'usage observables (§6.1) et sur le workflow analytique réel mis en œuvre (§6.2). Aucun élément hypothétique ou non vérifié n'est pris en compte.

6.3.1 Validation des besoins fonctionnels

Le premier besoin fonctionnel portait sur la création et la gestion d'ensembles expérimentaux clairement identifiés, associés à des métadonnées décrivant leur origine, leur contexte d'acquisition et leurs conditions d'utilisation. Les scénarios par rôle (§6.1) et le workflow d'importation (§6.2) montrent que le système permet effectivement la création d'ensembles structurés à travers un processus contrôlé, intégrant la définition de métadonnées, l'importation de données tabulaires et l'association persistante entre données et description. Chaque ensemble constitue une entité cohérente et gouvernée, identifiable indépendamment de son contenu physique.

Le second besoin concernait la capacité à gérer différents types de ressources, incluant des ensembles de données structurées ainsi que des fichiers. Les usages validés confirment que le

système distingue clairement ces catégories de ressources, tout en leur appliquant des mécanismes homogènes de visibilité et d'autorisation. Les fichiers non convertibles en tables relationnelles sont gérés comme des ressources à part entière, disposant de métadonnées propres et intégrées au modèle de gouvernance global.

Le troisième besoin portait sur l'accès programmatique aux ensembles expérimentaux afin de permettre leur intégration directe dans des workflows analytiques, notamment dans des environnements de type notebook, sans duplication systématique des données. Le workflow analytique validé au §6.2 montre que cet accès est assuré via des interfaces programmatiques dédiées, utilisées concrètement depuis des environnements analytiques. Les mécanismes de streaming, de pagination par curseur et l'utilisation de formats analytiques adaptés permettent une consommation progressive des données, compatible avec des analyses reproductibles.

Le quatrième besoin fonctionnel identifiait la nécessité de gérer plusieurs niveaux de visibilité, incluant des accès publics, restreints aux utilisateurs authentifiés organisationnels ou individuels. Les scénarios par rôle (§6.1) démontrent que ces niveaux de visibilité sont effectivement appliqués de manière cohérente, tant pour les ensembles de données que pour les fichiers. Les règles de visibilité sont évaluées systématiquement lors des accès, garantissant une séparation effective des usages selon le contexte utilisateur et les contraintes de collaboration.

6.3.2 Validation des besoins non fonctionnels

Le premier besoin non fonctionnel concernait la sécurité et le contrôle des accès aux données industrielles. Les scénarios par rôle (§6.1) montrent que l'authentification, l'autorisation basée sur des rôles globaux et la gestion explicite de la visibilité sont appliquées à chaque interaction significative avec le système, y compris dans les accès analytiques programmatiques.

Le second besoin portait sur la capacité du système à être déployé sur des infrastructures locales sécurisées. Le déploiement décrit au chapitre 5 et les usages validés confirment que la solution fonctionne dans un environnement institutionnel contraint, sans dépendance à des

services externes, et reste compatible avec des politiques de sécurité internes et une administration centralisée.

Le troisième besoin non fonctionnel concernait l'intégration avec des environnements analytiques déjà utilisés par les chercheurs. Le workflow analytique réel (§6.2) montre que cette intégration est opérationnelle via des interfaces programmatiques sécurisées, permettant aux environnements de notebooks de consommer les données sans accès direct au système de stockage, conformément aux exigences de séparation des responsabilités.

Le quatrième besoin portait sur la limitation de la charge d'administration manuelle et la stabilité de l'exploitation sur le long terme. Les mécanismes observés montrent que la création, l'accès et l'exploitation des ensembles sont majoritairement pilotés par des règles applicatives, réduisant les interventions directes sur l'infrastructure et favorisant une exploitation reproductible adaptée aux contraintes des équipes de recherche.

Enfin, le besoin d'évolution organisationnelle visait la capacité du système à supporter des collaborations impliquant plusieurs équipes ou organisations. Les structures de gouvernance observées, notamment l'association des utilisateurs et des ensembles à des entités organisationnelles et à des règles de visibilité, montrent que l'architecture actuelle ne présente pas de dépendance structurelle empêchant cette évolution, sans nécessiter de remise en cause majeure du modèle existant.

6.3.3 Synthèse du Chapitre 6

L'analyse menée dans ce chapitre montre que la solution implémentée répond de manière cohérente à l'ensemble des besoins fonctionnels et non fonctionnels identifiés au chapitre 3, dans le périmètre effectivement couvert par l'implémentation actuelle. Les mécanismes validés reposent sur des usages réels, observables et reproductibles, tant du point de vue des utilisateurs que des environnements analytiques.

Cette validation met également en évidence certaines limites structurelles et des choix assumés liés au contexte d'implémentation, aux contraintes d'infrastructure et au périmètre du projet.

Ces éléments ne remettent pas en cause la validité de la solution proposée, mais ouvrent des perspectives d'amélioration et d'évolution.

Le chapitre suivant est consacré à l'analyse de ces limites et à la discussion des perspectives d'évolution possibles, tant sur le plan technique que fonctionnel.

CHAPITRE 7

LIMITES ET PERSPECTIVES

Cette section présente les principales limites identifiées dans le cadre de l'implémentation actuelle de la solution. Ces limites sont analysées à la lumière du périmètre du projet, des contraintes d'infrastructure et des choix architecturaux assumés. Elles n'invalident pas les mécanismes proposés, mais en précisent le domaine de validité et les conditions d'utilisation.

7.1 Limites de la solution

7.1.1 Limites liées à l'infrastructure et à l'environnement d'exécution

La solution a été conçue, déployée et validée sur une infrastructure institutionnelle existante, dont les ressources matérielles et logicielles sont limitées. En particulier, l'utilisation de SQL Server Express impose des contraintes explicites sur la taille maximale des bases de données et sur l'allocation des ressources, ce qui limite la capacité du système à supporter des volumes de données très importants ou des charges analytiques intensives sur le long terme.

Par ailleurs, le stockage des données applicatives sur un support HDD local constitue une contrainte significative en matière de performances d'entrées/sorties. Bien que l'architecture privilégie des accès séquentiels et paginés, cette limitation peut se traduire par des temps de réponse plus élevés lors des opérations d'importation volumineuses, de création d'index ou de lectures analytiques à fort débit.

Ces contraintes sont renforcées par l'application volontaire de mécanismes de limitation de débit et de volume de lecture au niveau des API. Ces mécanismes, indispensables pour garantir la stabilité de l'infrastructure et prévenir les usages abusifs, limitent néanmoins le débit maximal de lecture des ensembles, en particulier lors d'analyses nécessitant un balayage complet de jeux de données volumineux. Dans ce contexte, les performances observées reflètent un compromis assumé entre protection de l'infrastructure et capacité de traitement

Ces limites sont directement liées au contexte d'exécution retenu pour le projet et ne remettent pas en cause la validité des principes architecturaux, qui demeurent compatibles avec des infrastructures plus performantes.

7.1.2 Limites fonctionnelles et périmètre applicatif

Sur le plan fonctionnel, la solution se concentre volontairement sur la gestion, la gouvernance et l'exploitation analytique des ensembles de données. Elle ne couvre pas l'ensemble du cycle de vie des données industrielles. En particulier, les mécanismes de transformation avancée, de nettoyage automatique, de normalisation ou de validation scientifique approfondie des données ne font pas partie du périmètre implémenté.

Les mécanismes de mise à jour du contenu reposent sur un remplacement contrôlé des ensembles, sans support de modification fine, incrémentale ou collaborative à grande échelle. Ce choix favorise la traçabilité et la cohérence des données, mais limite les usages nécessitant des mises à jour fréquentes ou concurrentes sur des sous-ensembles restreints.

De même, les mécanismes d'indexation proposés sont orientés vers des usages analytiques ciblés (accès temporels, filtrage par identifiant ou inclusion) et ne constituent pas une solution d'optimisation universelle pour l'ensemble des requêtes possibles. Certains cas d'usage complexes peuvent nécessiter des structures ou des moteurs spécialisés non pris en charge dans l'implémentation actuelle.

7.1.3 Limites fonctionnelles et périmètre applicatif

La validation de la solution repose sur des scénarios fonctionnels et analytiques représentatifs, mais circonscrits au cadre du projet. Les usages ont été évalués avec un nombre restreint d'utilisateurs et sur des volumes de données compatibles avec l'infrastructure disponible.

En conséquence, les résultats observés ne permettent pas de tirer des conclusions générales sur le comportement du système dans des environnements à très grande échelle, soumis à une forte concurrence ou à des charges analytiques continues. Les mécanismes de limitation de débit, de

pagination et de contrôle d'accès, bien que validés fonctionnellement, n'ont pas été évalués dans des scénarios de saturation prolongée.

Cette limitation ne remet pas en cause la cohérence ni la pertinence de la solution proposée, mais restreint la portée des conclusions aux conditions expérimentales effectivement observées.

7.2 Perspectives d'évolution

Les limites identifiées à la section précédente ouvrent naturellement des perspectives d'évolution, tant sur le plan de l'infrastructure que sur celui des fonctionnalités et de la gouvernance. Ces perspectives ne constituent pas des résultats du présent travail, mais des prolongements cohérents rendus possibles par les choix architecturaux retenus.

7.2.1 Évolution de l'infrastructure et des capacités

Une première perspective concerne l'évolution de l'infrastructure sous-jacente. La migration vers une édition plus avancée du système de gestion de base de données permettrait de lever les contraintes liées à SQL Server Express, notamment en termes de taille maximale, de performances et de gestion concurrente des accès. Cette évolution pourrait être réalisée sans remise en cause de l'architecture logique actuelle, les mécanismes d'accès aux données étant déjà abstraits par des interfaces applicatives.

De même, l'utilisation de supports de stockage plus performants, tels que des SSD locaux ou des solutions de stockage distribuées, permettrait d'améliorer significativement les performances d'entrées/sorties. Ces améliorations seraient particulièrement bénéfiques pour les phases d'importation, d'indexation et de lecture analytique à fort débit, tout en conservant les mécanismes de limitation et de contrôle existants.

Enfin, un déploiement sur des infrastructures plus puissantes ou mutualisées (serveurs dédiés, clusters internes ou environnements virtualisés) offrirait la possibilité de supporter un nombre

accru d'utilisateurs simultanés et des charges analytiques plus importantes, sans modification majeure des composants applicatifs.

7.2.2 Enrichissement des fonctionnalités analytiques

Sur le plan fonctionnel, la solution pourrait être enrichie par l'introduction de mécanismes supplémentaires de préparation et de validation des données. L'intégration de pipelines de transformation automatisés, déclenchés après l'importation ou avant l'exploitation analytique, permettrait de standardiser certaines opérations récurrentes (nettoyage, normalisation, enrichissement).

Des mécanismes de contrôle de qualité plus avancés pourraient également être ajoutés, afin de détecter automatiquement des incohérences, des valeurs aberrantes ou des problèmes de complétude dans les ensembles importés. Ces extensions renforceraient la fiabilité scientifique des analyses tout en conservant la séparation actuelle entre données brutes et traitements analytiques.

Par ailleurs, l'introduction d'un versionnement explicite des ensembles de données constituerait une évolution naturelle. En complément du mécanisme actuel de remplacement contrôlé, un tel versionnement permettrait de conserver l'historique des évolutions d'un ensemble, facilitant la reproductibilité des analyses et la comparaison entre différentes itérations des données.

7.2.3 Ouverture vers des contextes multiorganisationnels étendus

Enfin, l'architecture actuelle offre une base solide pour une extension vers des environnements multiorganisationnels plus complexes. L'enrichissement des modèles de gouvernance, notamment en matière de délégation de droits ou de partage interorganisationnel, pourrait permettre d'élargir les cas d'usage à des contextes collaboratifs plus larges, tout en conservant un niveau de contrôle élevé.

Ces perspectives d'évolution restent compatibles avec les choix de conception initiaux et s'inscrivent dans une continuité logique du travail présenté.

7.2.4 Intégration de pipelines d'ingestion continue depuis les systèmes SCADA

Une perspective d'évolution majeure concerne l'intégration de pipelines d'ingestion continue permettant de transférer directement les données depuis les sites équipés de systèmes SCADA vers la plateforme. Dans l'implémentation actuelle, l'ingestion repose sur des fichiers importés manuellement ou préparés en amont. Bien que ce mode soit adapté à des usages expérimentaux et analytiques, il ne couvre pas les scénarios de collecte continue ou quasi-temps réel.

CONCLUSION

Ce travail a permis de concevoir, d'implémenter et de valider une solution de gestion et d'exploitation analytique d'ensembles de données industriels, intégrée dans un environnement institutionnel contraint. La solution proposée démontre qu'il est possible de concilier gouvernance des données, sécurité des accès et exploitation analytique, y compris dans des contextes où les infrastructures, les ressources et les politiques de sécurité sont fortement encadrées.

L'architecture mise en place répond aux besoins fonctionnels et non fonctionnels identifiés, en s'appuyant sur des mécanismes concrets et observables : gestion structurée des ensembles et des métadonnées, contrôle des accès par rôles et visibilité, intégration d'environnements analytiques par interfaces programmatiques, et workflows analytiques reproductibles validés par l'usage. La validation fonctionnelle et analytique réalisée confirme la cohérence de ces mécanismes dans des scénarios réalistes, du point de vue des utilisateurs comme des environnements de calcul.

Les limites identifiées reflètent principalement le périmètre du projet et les contraintes de l'infrastructure de déploiement, sans remettre en cause les principes architecturaux retenus. Les perspectives d'évolution discutées montrent que la solution constitue une base solide pour des extensions futures, tant en termes de capacités analytiques que d'intégration continue avec des systèmes industriels ou de collaboration multiorganisationnelle.

Dans ce cadre, le travail présenté apporte une contribution pragmatique à la problématique de la gestion et de l'exploitation analytique des données industrielles, en proposant une architecture réaliste, gouvernée et compatible avec les exigences des environnements de recherche appliquée.

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

ISO, (2019), ISO 8601-1:2019 — Date and time — Representations for information interchange. https://www.iso.org

Klyne, G., Newman, C., (2002), Date and Time on the Internet: Timestamps (RFC 3339). IETF. https://www.rfc-editor.org/rfc/rfc3339

Nottingham, M., Wilde, E., (2016), Problem Details for HTTP APIs (RFC 7807). IETF. https://www.rfc-editor.org/rfc/rfc7807

Fielding, R., Nottingham, M., Reschke, J., (2022), HTTP Semantics (RFC 9110). IETF. https://www.rfc-editor.org/rfc/rfc9110

Hardt, D., (2012), The OAuth 2.0 Authorization Framework (RFC 6749). IETF. https://www.rfc-editor.org/rfc/rfc6749

Jones, M., Bradley, J., Sakimura, N., (2015), JSON Web Token (JWT) (RFC 7519). IETF. https://www.rfc-editor.org/rfc/rfc7519

Ferraiolo, D., Kuhn, D., Chandramouli, R., (2003), Role-Based Access Control. NIST / Artech House.

NIST, (2017), Digital Identity Guidelines SP 800-63B. https://doi.org/10.6028/NIST.SP.800-63b

Fielding, R. T., (2000), Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation.

Microsoft, (2023), Web API design best practices. https://learn.microsoft.com/aspnet/core/web-api/

Microsoft, (2023), Enforce HTTPS in ASP.NET Core. https://learn.microsoft.com/aspnet/core/security/enforcing-ssl

Microsoft, (2023), Authentication and authorization in ASP.NET Core. https://learn.microsoft.com/aspnet/core/security/authentication/