Client : Communauté d'Agglomération de La Rochelle	Numéro de marché : 12TN161
Maitre d'Ouvrage : Communauté d'Agglomération de La Rochelle	Agglomération de La Rochelle Direction de la Transformation Numérique 6, rue Saint-Michel 17000 LA ROCHELLE

DAT (Document d'Architecture Technique)
Plateforme de données LRTZC
LOT 1

Rev	Date	Préparé	Vérifié	Approuvé	Modifications	Statut
3.3	18/08/2022	Adrien Audemar Philippe Thamié Farid Mansouri	Farid Mansouri	David Berthiaud		Validé le 18/08/2022

Groupement Entreprises : **CITEOS solutions digitales**	**SYSTEME:**	Plateforme de données LRTZC
	SOUS-ENSEMBLE :	Infrastructure
	LOT :	Lot 1 Contributions Lot 2 et Lot 3

Format	Nom fichier
A4	*LRTZC7.1.1_Livrable_DAT Lot 1_V3.3.docx*

Privé	Libre	Interne	Restreint	Confidentiel
	X			

HISTORIQUE DES MODIFICATIONS

Rév.	Date	Resp.	Objet de la modification
1	23/03/22	Lot 1	Première revue DAT
2	Avril 2022	Lot 1	Premier partage DAT et retours de la CdA de La Rochelle
3.2	20/06/22	Lot 1	Réponses aux retours de la CdA
3.3	18/08/22	Lot 1	Revue avec la CdA de La Rochelle pour validation du DAT

1 Table des matières

2 Références documentaires

Libellé	Référence	Rév.	Date
Cahier des clauses contractuelles et techniques	21TN162_CCTP.pdf	-	**-**
Plan Assurance Qualité	20220609 - PAQ - Plateforme Territoriale de Données v2.docx	2	**16/06/22**
Spécifications Fonctionnelles Générales	LRTZC_SFG_v7.docx	7	**17/06/22**
Guide de l'AFAQ			
Référentiel général d'écoconception de services numériques (RGESN)			
Directive INSPIRE			
RGPD			
CMIS			
Open Géospatial Consortium			
RGI V2.0			
RGS			
Recommandations ANSSI			
RGAA 4.1			

3 Terminologie, abréviations et pictogrammes

Référence	Définition
GES	Gaz A Effet De Serre
LRTZC	La Rochelle Territoire Zéro Carbone
DAT	Dossier D'architecture Technique
MVP	Minimum Viable Product
ETL	Extract Transform Load
RGESN	Référentiel Général D'écoconception De Services Numériques
MCD	Modèle Conceptuel De Données
JDD	Jeu De Données
API	Application Programming Interface
FTP/SFTP	Ssh File Transfer Protocol
STI	Spécifications Techniques D'interface
SIG	Système D'information Géographique
BDD / SGDBR	Base De Données / Système De Gestion De Base De Données Relationnelles
IOT	Internet Des Objets
SAE	Systèmes D'archives Electroniques
IHM	Interface Homme-Machine
IGN	Institut National De L'information Géographique Et Forestière
GPS	Global Positioning System
AIPD	Analyse D'impact Sur La Protection Des Données
CNIL	Commission Nationale De L'informatique Et Des Libertés
SLA	Service Level Agreement

4 Contexte

De la conviction que la lutte contre les gaz à effet de serre (GES) est avant tout un défi collectif qui nécessite une meilleure mutualisation et coordination des acteurs et des ressources, y compris numériques, existantes et à venir, est née l'ambition de mettre à disposition de tous les acteurs du territoire, engagés dans le projet La Rochelle Territoire Zéro Carbone, une méta-plateforme comme outil technique transversal d'aide à la réponse à des problématiques de résilience et de transition écologique.

Elle répond à 3 objectifs :

- Répondre aux attentes des leviers du projet LRTZC, par le biais de la mise en place de cas d'usages leur permettant de mobiliser le plein potentiel des données du territoire pour atteindre leurs objectifs.
- Collecter les éléments permettant de produire les indicateurs de suivi du projet LRTZC.
- Permettre à La Rochelle d'évoluer vers une gouvernance territoriale des données qui vienne appuyer la mise en place des politiques publiques du territoire dans la continuité des premières pierres posées par l'Open Data.

La construction de cette plateforme constitue l'objet de ce DAT. Le DAT (Document d'Architecture Technique) rassemble les éléments techniques d'infrastructure, d'hébergement, de réseau et d'applicatif de l'ensemble de la plateforme de données LRTZC. Le DAT évoluera avec les choix techniques réalisés au fil du projet (ex : choix de l'hébergeur pour la plateforme de production).

Ainsi :

- Les architectures fonctionnelles et applicatives présentées sont celles de la plateforme cible complète du projet.
- Les architectures techniques sont détaillées pour la phase MVP (fin 2022) en attendant que l'étude de l'hébergement de production soit réalisée.

5 Philosophie de la plateforme

Qu'est-ce qu'une plateforme dite « Big Data » ?

- Capacité d'ingérer et de stocker des données en continu, et en croissance (volumétrie, grande quantité), et en variété (formats)
- Agilité de la collecte, l'indexation et le partage des données : les données sont soit générées en continu dans le temps avec un fort besoin en temps réel pour garantir l'historique ou via l'import ponctuel de Jeu de données plus ou moins volumineux.
- Variété des formats : les types de données sont de plus en plus diversifiés et avec des niveaux de structuration différents (bruts, semi-structurés ou structurés)
- Faire des traitements batch sur les données (séparer les données en petits lots de données où chaque serveur calcule un morceau du résultat en volumétrie)

La plateforme de données LRTZC est orientée big data par la mise en place de lots de collecte et de catalogues de données qui stockent les données en volumétrie et de façon brute. La volumétrie des données qui seront in fine ingérées par la plateforme permettra de définir si la plateforme est dite Big data et si des fonctionnalités spécifiques à la gestion en très forte volumétrie seront attendues.

Agilité de la collecte :
Avec un ETL classique, les informations sont extraites, transformées et chargées à partir de systèmes transactionnels à sources multiples dans un seul espace, comme un data Warehouse d'entreprise sans prendre en compte une intégration des données répondant aux attentes du Big Data.
Notre architecture comprend une partie ETL avec l'intégration pour gérer la qualité de la donnée, préparer un catalogue de données basé sur des métadonnées, la gestion des données de référence et une capacité de traitement s'adaptant au volume de données par un système de clusters soit pour chaque étape d'un pipeline ETL soit pour réaliser des calculs en mode distribué aussi bien en temps réel que par lots.

Stockage de la plateforme :
Afin de répondre au stockage volumineux et varié de données, plusieurs types de bases de données ont été sélectionnées selon les besoins.

Une base de données orientée NoSQL autorisera à la fois la scalabilité verticale et horizontale grâce à son mode distribué. Ce type de base est pertinent pour stocker de la donnée brute dans la phase d'acquisition de la plateforme. Une base de données relationnelles permettra d'organiser l'information pour des besoins de traitements métiers autour de la mise à disposition de données structurées.

Les fonctionnalités de calculs avancés… une innovation de la plateforme La Rochelle :

- Intégration d'outils de mutualisation et d'optimisation de recherche / calcul sur la donnée ; évolutivité forte de ces outils (possibilité d'en intégrer au fur et à mesure, en fonction de la volumétrie et des usages sur les données) (ex : solutions MongoDB pour le catalogue de données)
- Intégration d'outils de data visualisation (ex : Grafana)
- Implémentation d'algorithmes simples et experts qui peuvent être dimensionnés pour le big data
- Développement du bureau virtuel pour une création d'analyses à la demande, par les data experts

Livrée avec un socle d'outils d'analyse et de traitement des données, la plateforme pourra s'enrichir au besoin au fil du temps => elle est évolutive (ex : utilisation de SOLR (équivalent Elastic Search en Apache 2.0) dans le projet Dataverse). Par ailleurs, la migration d'outils vers des plus modernes ou efficaces pourra être envisagée.

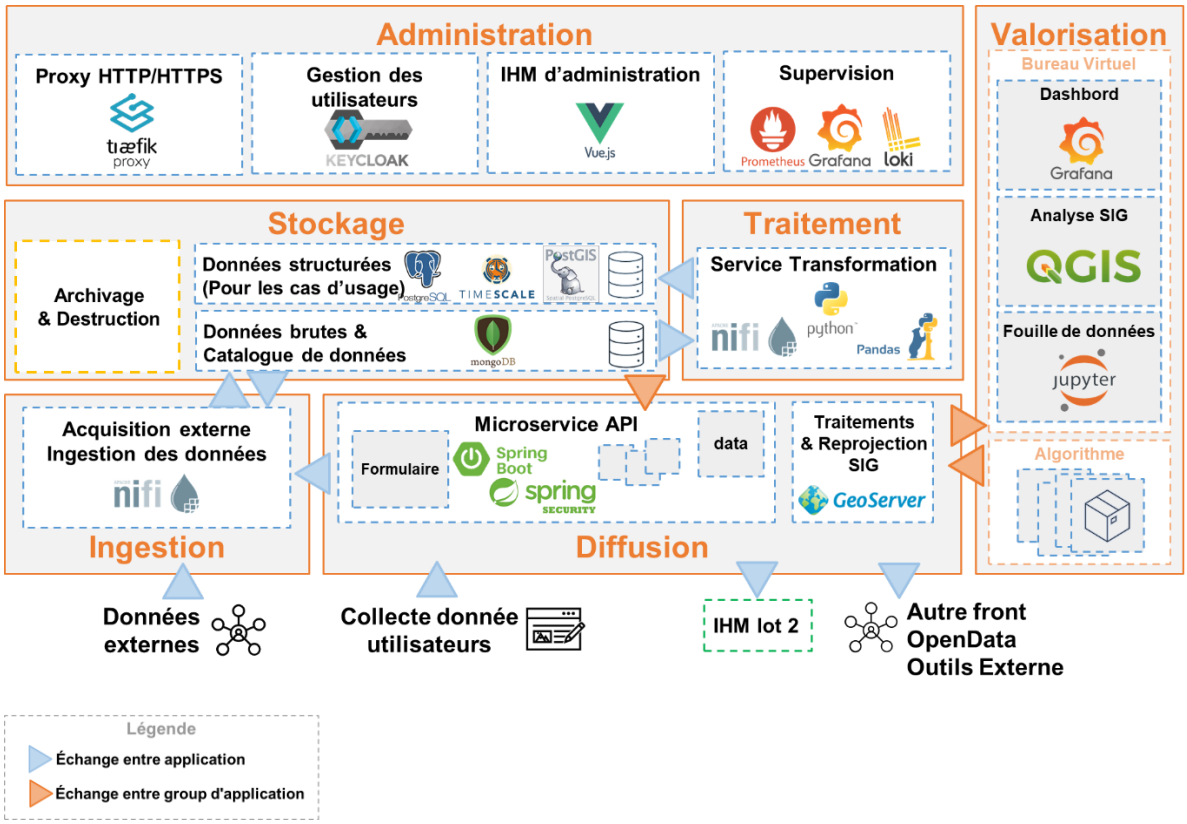
Démarche d'éco-conception

Le développement de la plateforme territoriale de données respectera une démarche éco-conception prenant en compte l'ensemble des impacts environnementaux aux différentes étapes de conception et d'usage du produit, jusqu'à sa fin de vie.

Le développement des services numériques proposés par la plateforme respectera le Référentiel général d'écoconception de services numériques (RGESN). Les objectifs sont de réduire la consommation de ressources informatiques et énergétiques et la contribution à l'obsolescence des équipements, qu'il s'agisse des équipements utilisateurs ou des équipements réseau ou serveur.

6 Représentation fonctionnelle

6.1 Architecture applicative



6.2 Architecture fonctionnelle macroscopique

L'architecture technico-fonctionnelle de la plateforme de données est la suivante. Elle intègre les éléments du lot 1, du lot 2 et du lot 3 :

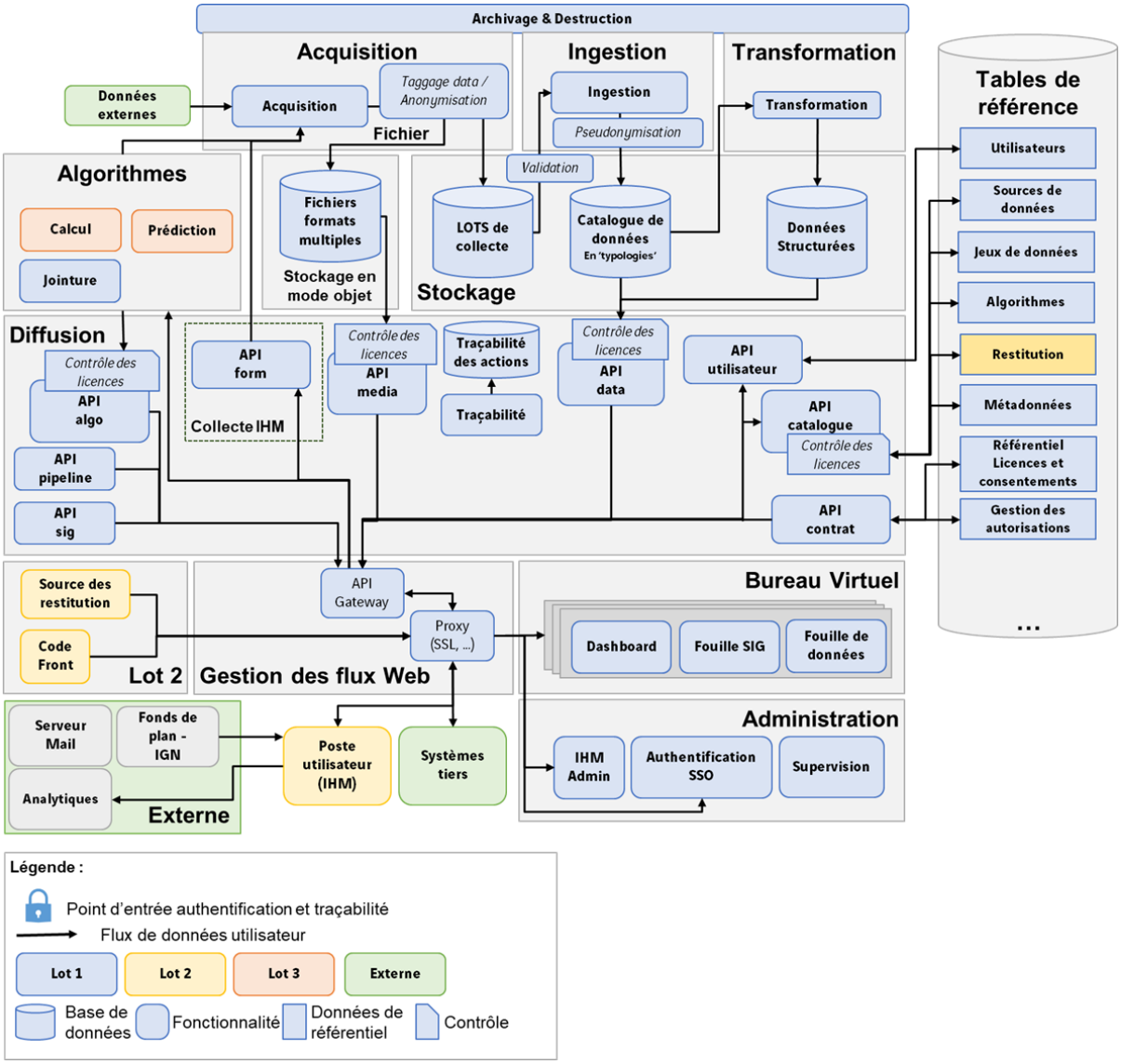


Figure 1 - Architecture fonctionnelle macroscopique

6.3 Flux de données – Externes à la plateforme

La plateforme échange des données (en tant que récepteur et diffuseur) avec des systèmes tiers. Une première vision de ces échanges est représentée sur le schéma ci-dessous.

Une étude protocole / réseau approfondie sera réalisée en phase d'exécution pour qualifier de manière exhaustive ces différents flux.

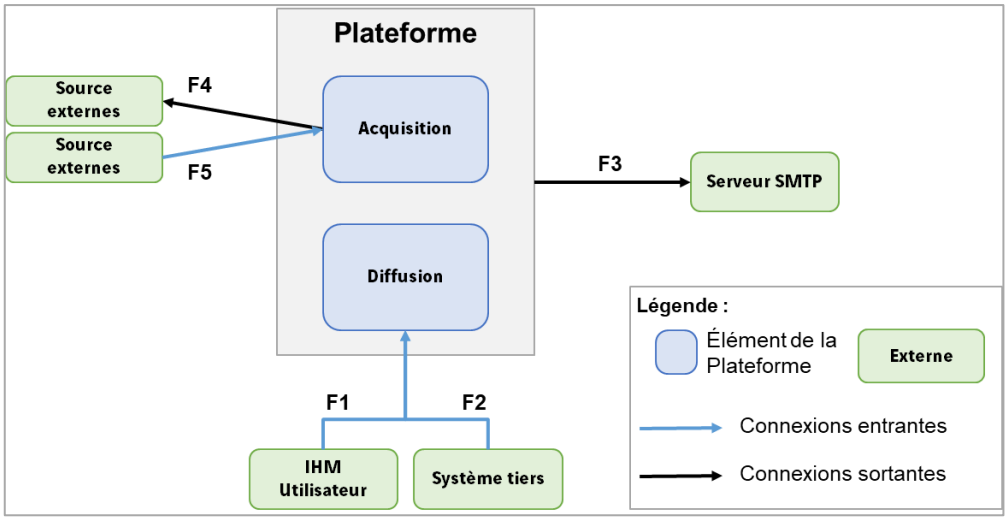


Figure 2 - Schéma des entrées / sorties de la plateforme

Ci-dessous, une matrice des principaux flux d'entrées et de sorties de la plateforme :

Référence	*Serveur*	*Client*	*Protocole applicatif*	*TCP ou UDP*	*Port*	*Objectif*
F1	Diffusion Plateforme	IHM Utilisateur	HTTP	TCP	80	Transfère les demandes vers HTTPS
F1	Diffusion Plateforme	IHM Utilisateur	HTTPS	TCP	443	Accès code front Accès aux APIs
F2	Diffusion Plateforme	Systèmes tiers	HTTPS	TCP	443	Consommation des API
F3	Serveur SMTP	Plateforme	SMTPS	TCP	465	Envoi des mails de notification et pour l'authentification
F4	Source externes	Acquisition Plateforme	HTTPS	TCP	443	Collecte de données via API Externe
F4	Source externes	Acquisition Plateforme	SFTP	TCP	22	Collecte de données via SFTP Externe
F5	Acquisition Plateforme	Source externe	HTTPS	TCP	443	Envoi de données par des système externe

La matrice des flux de données sera mise à jour sur chaque nouvelle évolution potentielle de la plateforme durant la phase de réalisation.

6.4 Modèle de données

6.4.1 Tables de référence

Les **tables de référence** représentent toutes les informations nécessaires au fonctionnement de la plateforme qui ne soient pas spécifiquement des lots de collectes, des données (/datas) ou des algorithmes. Par exemple :

- Gestion des utilisateurs (préférences, etc.),
- Liste des sources de données (connecteurs),
- Métadonnées associées aux Catalogue,
- Consentement, Licence et Smart contract,
- Gestion des autorisations (entre utilisateurs et objets),
- Catalogue d'algos de traitements,
- Catalogue de modèles de restitutions (dataviz : graphiques, cartes …),
- Catalogue d'indicateur/restitution
- Catalogue de jeux de données,
- Catalogue des Cas d'usage (titre, description, etc.),
- Etc..

Les **métadonnées** sont des informations descriptives permettant de caractériser et de retrouver les informations stockées dans les catalogues. Elles sont composées par exemple des informations : auteur, date de création, origine, informations sur le type de la données, les contraintes liées aux licences, informations de classification, etc.

Les catalogues comme les jeux de données, algorithmes, restitutions posséderont des métadonnées mais il peut être envisagé de créer un catalogue des métadonnées permettant d'alimenter un moteur de recherche depuis l'IHM de la plateforme.

6.4.2 Modèle de métadonnées

La définition des métadonnées est en cours et disponible depuis un fichier en ligne. Pour la vision MVP, les métadonnées définies sont :

- Champs de métadonnées obligatoires :
  - Titre
  - Description
  - Contact
  - Langue
  - Thématique
  - Schéma (Alphanumérique / Géographique)
- Champs Métadonnées optionnelles
  - Licences d'utilisation
  - Territoire

Pour aller plus loin, on pourrait standardiser le modèle de métadonnées de la plateforme dans un objectif d'interopérabilité.
Un thésaurus ou dictionnaire analogique (synonymes et antonymes) permet d'organiser les mots par champ lexical et de d'identifier des liens entre les métadonnées afin d'optimiser les analyses.

En informatique et en science de l'information, une ontologie est l'ensemble structuré des termes et concepts représentant le sens d'un champ d'informations, que ce soit par les métadonnées d'un espace de noms, ou les éléments d'un domaine de connaissances. L'ontologie constitue en soi un modèle de données représentatif d'un ensemble de concepts dans un domaine, ainsi que des relations entre ces concepts. Elle est employée pour raisonner à propos des objets du domaine concerné. Plus simplement, on peut aussi dire que l'ontologie est aux données ce que la grammaire est au langage ».

Remarque
L'ontologie est un sujet à part entière qui demande beaucoup de travail et qui n'avait pas été prévue initialement. Il serait intéressant en effet pour apporter de la sémantique aux données d'étudier cette partie peut être à travers un partenariat avec l'Université de La Rochelle sous forme d'une thèse, à voir ensemble.

6.4.3 Gestion des données de référence

Plusieurs sources de données existent et l'objectif est de fédérer la donnée pour avoir un unique référentiel, une seule source de données homogène.

Les référentiels pourront être dédiés par cas d'usage.

Il est projeté d'avoir deux types principaux de référentiel :

- Référentiel générique : données statiques et que l'on peut utiliser partout comme par exemple, une liste de code postal
- Référentiel métier : données pouvant évoluer dans le temps et spécifique au domaine métier de la plateforme, par exemple, la liste des thématiques.

Exemple : un référentiel homogène (lié à tous les bâtiments quelques soit leur type) doit être généré à partir de plusieurs référentiels dédiés à différents types de bâtiment (Université, Locaux d'affaires, etc.)
Cet exercice est souvent réalisé en passant par une application dédiée (géré par un référent humain manuellement).

6.4.4 Moissonnage

Il faut pouvoir être moissonné et moissonner les métadonnées d'autres plateformes de données via un catalogue standardisé.

Moissonner permet d'avoir des liens vers les producteurs, de stocker des références, de centraliser les liens vers les données (comme dans data.gouv) pour réorienter les utilisateurs vers les plateformes stockant les jeux de données.

Les deux fonctionnalités principales que le moissonnage pourrait apporter sont :

1. Afficher des jeux de données externes disponibles suite au résultat d'une recherche sur des métadonnées par l'utilisateur dans le catalogue de la plateforme (affichage de l'intitulé, de la description et du lien vers la plateforme externe)
2. Utiliser des jeux de données externes pour alimenter un cas d'usage dont la référence vers la source et la structure sont stockées dans les métadonnées moissonnées :
    a. soit par stockage d'un échantillon
    b. soit par stockage dans un cache temporaire afin de pas dupliquer la donnée)

Quelques définitions :

https://doc.data.gouv.fr/moissonnage/ckan/ : Le moissonneur utilise l'API de CKAN pour récupérer les métadonnées. https://doc.data.gouv.fr/moissonnage/dcat/ : DCAT est une ontologie RDF pour décrire des jeux de données. Ce moissonneur attend l'URL d'un catalogue DCAT (dcat:Catalog).

6.4.5 Modèle Conceptuel de données

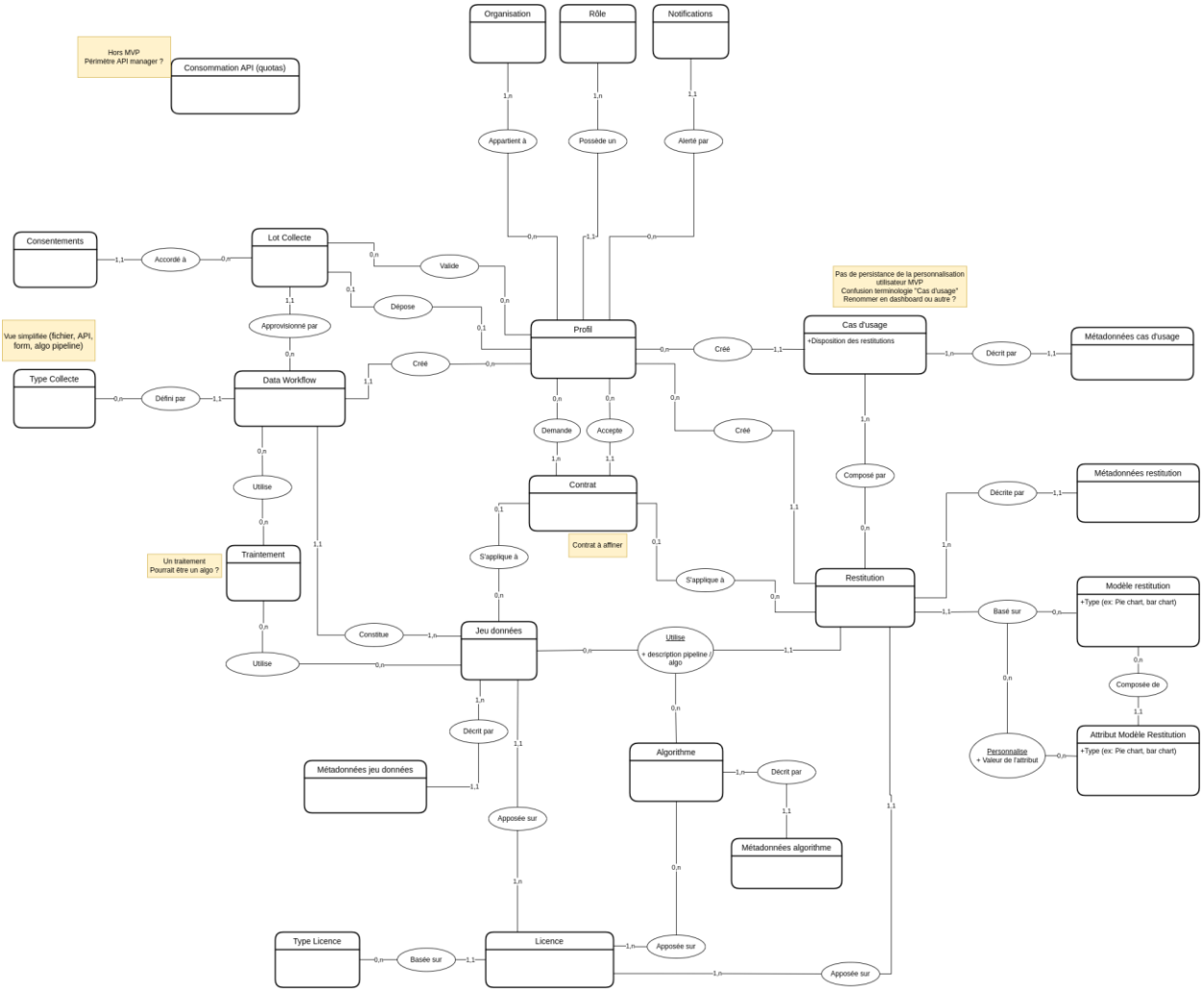


Figure 3 - Modèle Conceptuel de données

Une initialisation du **MCD (Modèle Conceptuel de Données)** sur la base des tables de référence portées par le lot 1 a été réalisée. Ce modèle est amené à évoluer dans le cadre de la conception technique des lots 1 et 2.

6.5 Profils d'utilisateurs

Les profils utilisateurs définis sont les suivants :

Visiteur non connecté	• Internaute non authentifié consultant les informations publiques
Contributeur individuel	• Internaute authentifié consultant les informations publiques et protégées • Citoyen ou acteur de la collectivité fournissant des données à la plateforme • Accès API
Producteur Institutionnel	• Acteur d'une organisation (collectivité ou entreprise), en charge de la mise en œuvre et du déploiement de l'un des cas d'usage • Gestionnaire de l'organisation, valideur des données intégrées dans la plateforme • Accès API
Administrateur délégué	• Producteur institutionnel, en charge de gérer les membres de son organisation
Créateur / Ré-utilisateur	• Utilisateur du bureau virtuel (ie Data Analyst)
Administrateur fonctionnel	• Acteur de la collectivité en charge de publier des contenus et d'assurer la gestion du site • Assigne les droits & permissions • En charge de modérer / Gouverner la plateforme
Administrateur technique	• Implémentation technique de la collecte de la donnée • Maintient la plateforme en conditions opérationnelles

Figure 4 - Liste des profils d'utilisateurs

7 Conception applicative

7.1 Introduction technico-fonctionnelle

Le socle technique de la plateforme permettra l'implémentation des cas d'usage ainsi qu'aux fonctionnalités directement en contact avec l'utilisateur. Il intègre les composants répondant aux fonctions suivantes :

- Les dispositifs d'acquisition & connecteurs,
- L'ingestion des données et la qualité des données,
- La transformation des données,
- Le stockage et la sécurisation des jeux de données,
- La mise à disposition des données dans le respect des règles d'accès (publication et export),
- L'archivage des données,
- La destruction des données.

L'ensemble du processus doit pouvoir être automatisé et orchestré en tenant compte des limites de ressources techniques (CPU, réseau, mémoire).

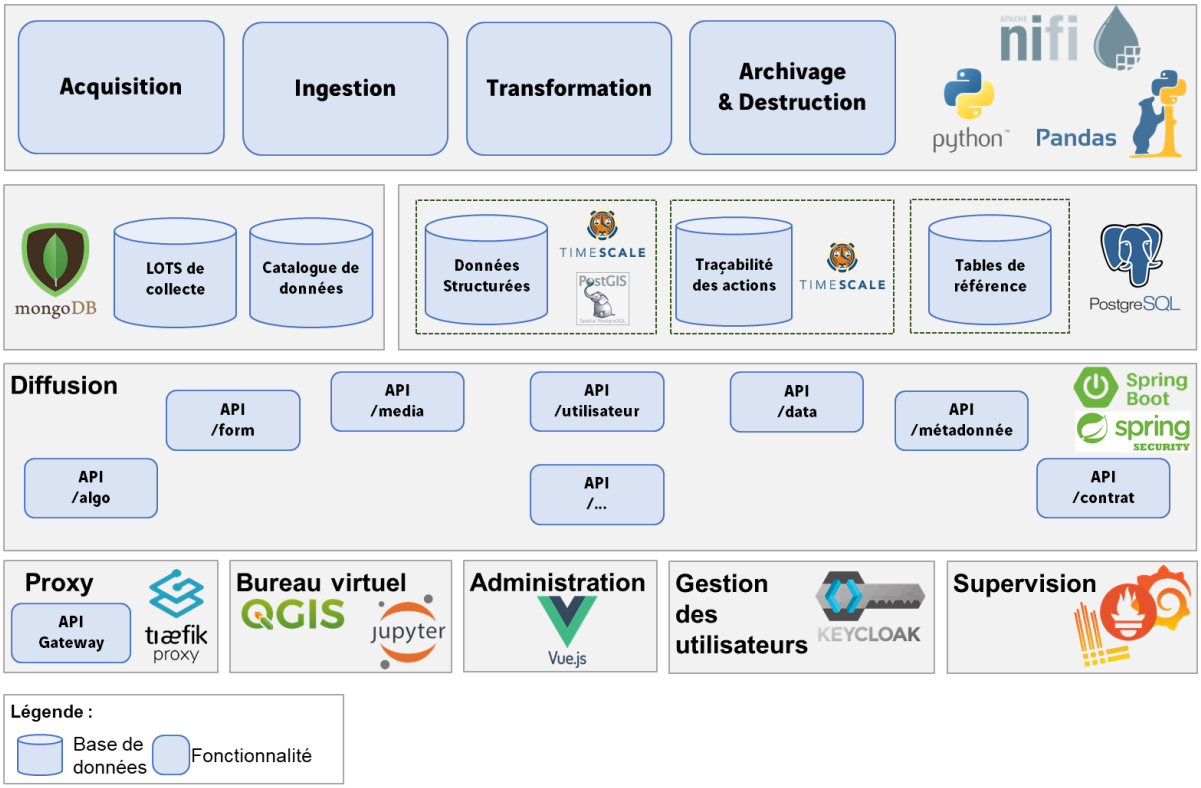


Figure 5 - Architecture applicative

7.2 Cycle de vie de la donnée

La solution logicielle NiFi va permettre de collecter en temps réel des données issues de sources très diverses. C'est dans la logique des applications Big Data dans lesquelles on cherche à avoir des données sources le plus rapidement possible.

Il offre l'avantage de cataloguer, créer et d'instancier un ensemble de briques fonctionnelles pouvant être utilisé lors de chaque étape du cycle de vie de la donnée. Cela permettra la réutilisation de chaque élément ou un ensemble complexe d'élément avec des variables et de minimiser les coups de déploiement de collecteur et de traitement sur des types de données similaires.

Il fournit également une API REST, ce qui permet de l'intégrer dans un écosystème complexe et de pouvoir créer des automatisations depuis d'autre élément de la plateforme.

Ci-dessous, la liste des briques techniques sélectionnées pour répondre aux besoins du cycle de vie de la donnée :

Composant / Framework	Type	Licence	Première version	Langage
Apache NiFi	Application	Apache-2.0 License	31 Août 2016	Java
Python	Langage	P.S.F. License Version 2	26 Janvier 1994	C
Pandas	Bibliothèque	Licence BSD	29 Janvier 2020	Python

Étude applicative réalisée :

Nous faisons le choix de la technologie NiFi par rapport à la technologie Pentaho Kettle, en ce qui concerne la gestion de la donnée.
La comparaison suivante entre NiFi et Pentaho Kettle a été réalisée :

- Besoin technique : traitement de la donnée entrante dès qu'elle arrive au niveau de la brique acquisition, ce qu'on appelle « au fil de l'eau ».
  - NiFi va automatiser les flux de données au fil de l'eau ou en mode batch.
  - Kettle est un ETL orienté batch, le traitement au fil de l'eau n'est pas son mode de fonctionnement natif.
- Besoin technique : clustering, c'est-à-dire un outil pensé pour être utilisé dans un cluster : déploiement de plusieurs instances de traitement de données entrantes en parallèle, pour suivre la montée en charge
  - NiFi a un clustering natif
  - Kettle est clusterisable
- Besoin technique : interopérabilité technologique et facilité de délégation de paramétrage
  - Kettle est pensé pour s'intégrer à la suite Pentaho (interopérabilité limitée avec d'autres outils qui ne font pas partie de la suite) et configurable au travers d'une application lourde. NiFi offre une interface API REST complète et paramétrable au travers d'une interface web.
- Besoin technique : pouvoir configurer à chaud l'outil sans perte de données
  - NiFi facilite la mise en mémoire tampon des données en file d'attente entre chaque étape pour éviter la perte de donnée lors de la modification d'un élément du flux. Il n'est pas nécessaire de compiler et de déployer les modifications du flux.

7.2.1 Acquisition

7.2.1.1 Visée fonctionnelle

La collecte des données est la première étape du traitement des données. Les données proviennent de toutes les sources externes disponibles (données de la ville et de la CdA, données géographiques, données externes type open data, etc.). Il est important que les sources de données disponibles soient fiables et correctement structurées pour que les données importées soient de la meilleure qualité possible.

Les dispositifs d'acquisition doivent permettre d'adresser une large diversité de technologies et de formats de données (fichiers plats, API, ftp, sftp, etc.).

La brique acquisition permet de gérer et d'automatiser des flux de données entre plusieurs systèmes informatiques afin de :

- Programmer la fréquence de récupération des données,
- Acquérir des données de systèmes tiers en respectant les exigences de sécurité (antivirus sur les fichiers) et de gestion de licences,
- Assurer un premier contrôle sur le format des données fournies,
- Proposer des connecteurs paramétrables respectant les contraintes d'interopérabilité,
- Gérer la scalabilité lors de l'acquisition,
- Stocker les données de manière temporaire avant leur intégration,
- Structurer les données acquises en lots de collecte pouvant par la suite être ingérés par la plateforme.

7.2.1.2 Visée technique applicative

La brique acquisition permet de collecter des données externes grâce à des connecteurs dédiés puis de normaliser les données afin de préparer leur traitement ultérieur. Cela se traduit par :

- Un catalogue de connecteur paramétrable,
- Un stockage de la donnée avant traitement,
- Un ajout de tags pour permettre la traçabilité,
- Gestion des droits d'accès lors de la collecte.

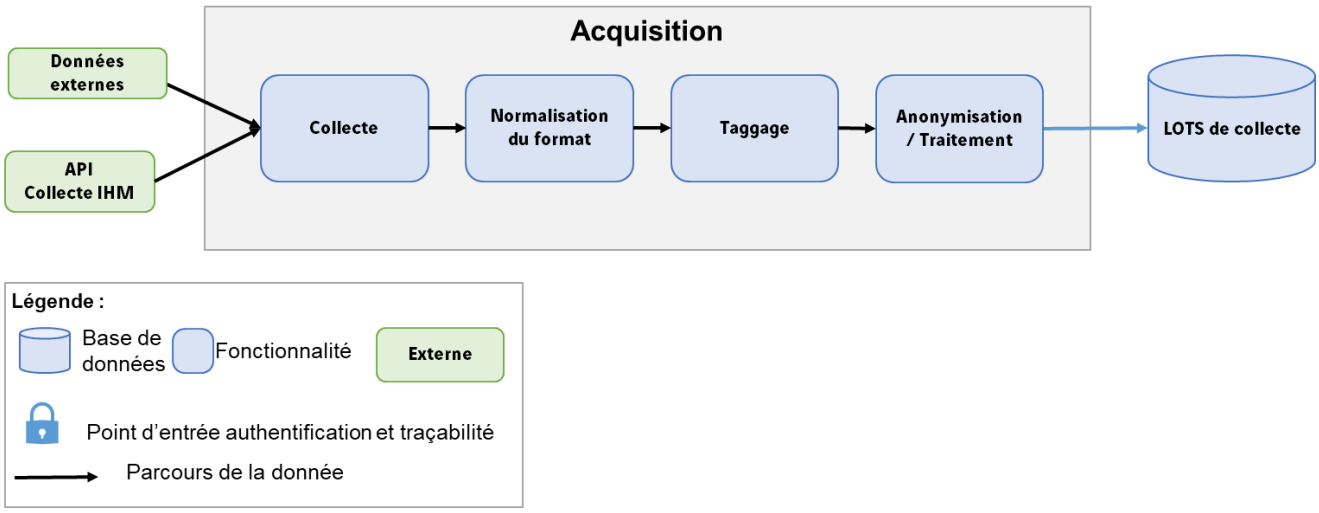


Figure 6 - Visée technique acquisition

7.2.1.2.1 Collecte

Notre démarche d'intégration inclut la définition d'une banque de connecteurs afin de capitaliser sur des paramétrages communs (fréquence, informations de source, etc.) et permettre la réutilisation de protocoles plus complexes. Cela se traduit par la mise en place d'un catalogue de connecteurs regroupés par type de traitement.

Un développement de connecteur spécifique peut être envisagé pour répondre aux nouveaux besoins et optimiser la configuration en phase d'exploitation.

7.2.1.2.2 Normalisation

La normalisation standardise les types de données vers un format propre à l'application pour permettre d'exploiter des briques fonctionnelles standards, configurables et réplicables.

7.2.1.2.3 Taggage

Le taggage consiste à ajouter des champs techniques pour chaque ligne de donnée afin de permettre la traçabilité de la donnée tout au long de sa vie dans la plateforme, elle se compose en autre des informations sur le producteur de la donnée, l'identifiant du catalogue de donnée, …. Les valeurs pourront être récupérées depuis la donnée source ou bien rajoutées lors de l'acquisition.

7.2.1.2.4 Anonymisation et Traitement

Des traitements complémentaires pourront être ajoutés pour respecter les contrats et licences d'échange avec les fournisseurs, comme par exemple :

- La limitation des champs collectés définie par le périmètre de la licence d'accès,
- Le respect aux exigences RGPD en anonymisant certaines données à caractère personnel.

7.2.1.3 Choix applicatif

La solution NiFi nous permet de gérer et d'instancier une large variété de connecteurs vers de multiples systèmes d'information grâce à :

- Une interface de paramétrage pouvant couvrir la majorité des cas d'usages,
- Un large catalogue de connecteurs vers des systèmes standards,
- La possibilité de rajouter et partager des connecteurs,
- Le paramétrage et la modification sans interruption de service.

7.2.2 Ingestion

7.2.2.1 Visée fonctionnelle

Après la collecte des données, suit la préparation des données. La préparation des données est l'étape de pré-traitement pendant laquelle les données brutes sont nettoyées en vue de l'étape suivante du traitement des données. Pendant cette phase de préparation, les données brutes sont vérifiées afin de déceler d'éventuelles erreurs. L'objectif est d'éliminer les données de mauvaise qualité (redondantes, incomplètes ou incorrectes) et de créer un catalogue de données qualifié qui peut être par la suite transformé pour alimenter les cas d'usages.

L'ingestion des données va réaliser un premier niveau de traitement grâce à l'application de règles et de scripts de traitement pour le contrôle et la qualité des données avant le stockage. Cela se traduit par :

- La mise en place d'un catalogue de règles de contrôle, correction et rejet pour la qualité des données,
- La mise en place d'un catalogue de règles de premier niveau de transformation de la donnée (déduplication, etc.) pour faciliter l'exploitation par les cas d'usage,
- La mise en place d'un catalogue de règles de gestion permettant d'intégrer de nouvelles données à des jeux de données existants (mode d'ingestion),
- Des traitements RGPD (pseudonymisation, masquage, etc.).
    - En cas de pseudonymisation, il faudrait que la clé permettant de réidentifier une personne soit sécurisée (par exemple en étant stockée dans une instance de base de données dédiée).
    - Selon les outils et technologies utilisés, ce besoin sera pris en compte dans la phase de conception détaillée.

7.2.2.2 Visée technique applicative

La brique ingestion permet de contrôler la qualité des données, de réaliser des opérations de correction, d'intégration et de transformation pour alimenter le catalogue de données. Cela se traduit par :

- Un catalogue de règles et de scripts de traitements paramétrables,
- La disponibilité d'une IHM d'administration,
- La gestion de la qualité de la donnée (règles et alertes).

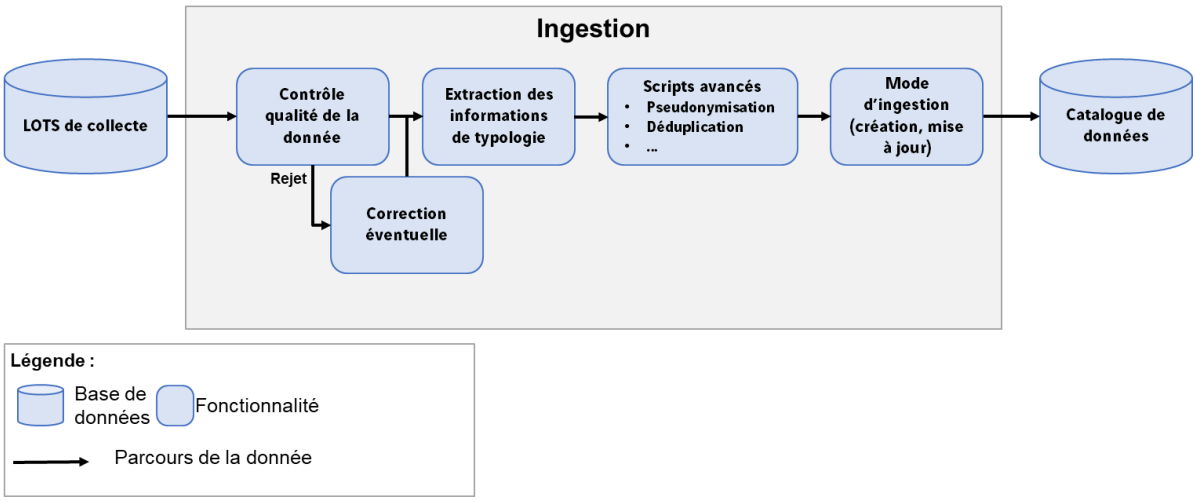


Figure 7 - Visée technique ingestion

7.2.2.2.1 Contrôle qualité de la donnée

Le format des données est vérifié et traité selon un schéma de référence afin de valider des règles de qualité et qui peut faire appel à une stratégie de correction sur des erreurs rencontrées. La gestion du suivi des rejets et leurs conservations pendant une période de temps sera définie lors de la conception détaillée.

Les indicateurs de qualité et d'erreur pourront être restitués sur l'outil de supervision qui pourra déclencher des alertes (ex : sur flux de données cassé, jeux de données vides, etc.).

Les familles de critère de qualité de la donnée sont :

- Indice de qualité (ou confiance, moyenne des indicateurs de qualité) pouvant être représenté sous forme d'un diagramme radar avec les axes suivants :
    1. Exhaustivité (données attendues présentes)
    2. Fraicheur (délai création et mise à disposition)
    3. Unicité (pas de doublons)
    4. Complétude (toutes les informations sont présentes)
    5. Exactitude (conforme aux schéma, format)
- Problème (rejet ou erreur lors d'un traitement)

On aura des objectifs qualité par jeu de données et transverses à définir selon un schéma cible de qualité pouvant être configuré manuellement et avec des valeurs accessibles depuis les métadonnées.

L'homogénéisation peut passer par l'utilisation des 5 mêmes axes de qualité pour n'importe quel jeu de données.

Les indicateurs de qualité et d'erreur pourront être restitués sur l'outil de supervision qui pourra déclencher des alertes (ex : sur flux de données cassé, jeux de données vides, etc.).

7.2.2.2.2 Extraction des informations de typologie

Les données vérifiées vont être analysées pour en extraire une typologie permettant un premier niveau de structuration pour déterminer le stockage idoine puis faciliter les traitements ultérieurs. Ci-dessous, des exemples de typologie :

- Géographique,
- Temporelle,
- Versionné,
- Clé unique.

7.2.2.2.3 Scripts avancés

Des traitements spécifiques peuvent compléter l'ingestion pour améliorer la garantie de la qualité, comme par exemple :

- La déduplication pour nettoyer les doublons,
- Le respect des exigences RGPD et contractuelles (ex : pseudonymisation ou obfuscation),
- Réduction de la précision des données (gardé l'année d'une date de naissance, réduire les adresses aux quartiers, …)
- Etc..

Les modalités de traitements seront définies lors de la contractualisation des données avec le producteur.

7.2.2.2.4 Mode d'ingestion

Plusieurs modes d'ingestion sont possibles pour créer, mettre à jour, compléter ou synchroniser le jeu de données cible qui complétera le catalogue de données.

7.2.2.3 Choix applicatif

Dans le cadre de l'ingestion, NiFi permet de gérer et d'automatiser les traitements via la création de pipeline d'enchaînement de briques fonctionnelles avec comme fonctionnalités clés :

- Une interface de paramétrage avancée,
- Un large catalogue de briques fonctionnelles paramétrables existantes,
- La possibilité de rajouter et partager des briques fonctionnelles,
- Le paramétrage et la modification sans interruption de service.

7.2.3 Transformation

7.2.3.1 Visée fonctionnelle

Suite à l'ingestion et à la création du catalogue de données, on pourra si besoin consommer un ou plusieurs jeux de données afin de transformer les données qui seront utilisées pour construire des restitutions de cas d'usage.

Cette phase va être dépendante des acteurs qui conceptualisera les différents cas d'usage et pourra évoluer avec le temps.

Plusieurs types de transformation sont disponibles :

- Transformation via agrégation de jeux de données,
- Transformation via combinaison de jeux de données,
- Transformation via la construction d'un schéma relationnel.

7.2.3.2 Visée technique applicative

La brique transformation consiste à construire des données structurées pour permettre l'exploitation au travers des restitutions liées aux cas d'usage. On peut déjà faire ressortir trois types de transformation, qui pourront évoluer en fonction des besoins :

Transformation via agrégation de jeux de données

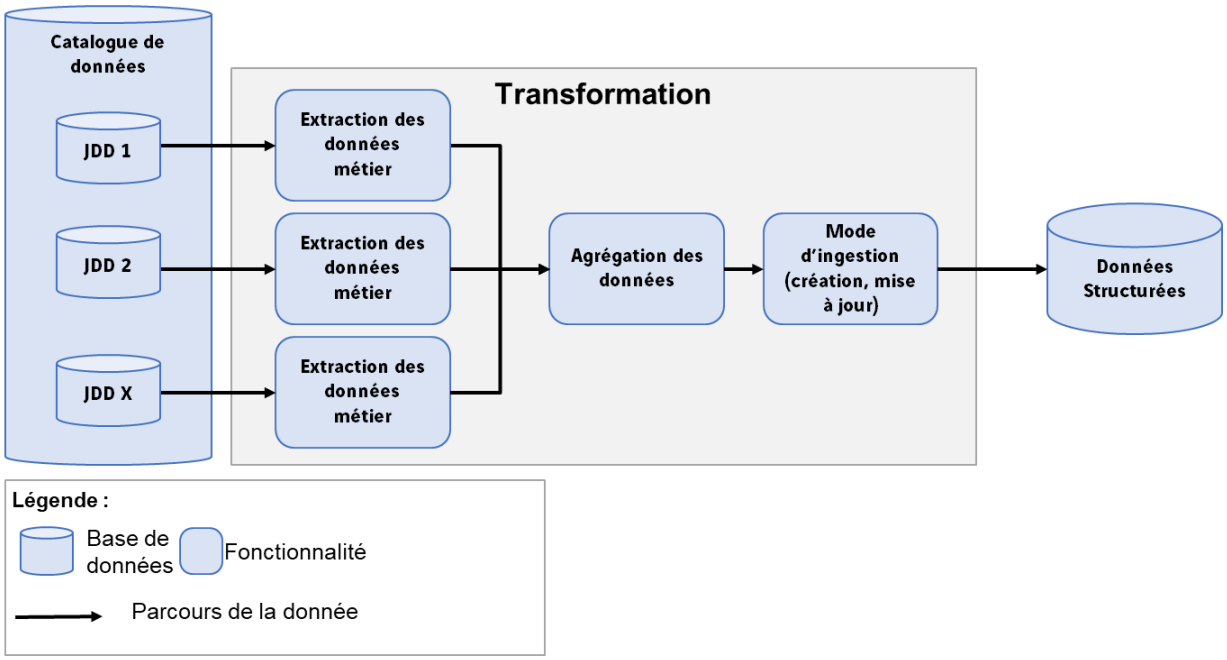


Figure 8 - Visée technique transformation 1

Transformation via fusion de jeux de données

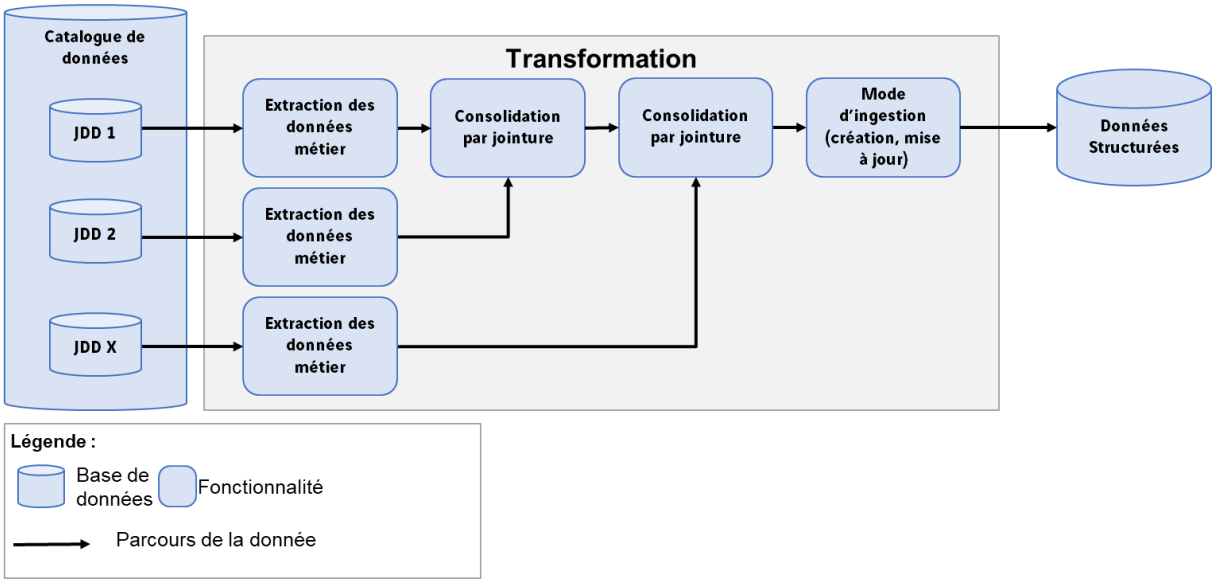


Figure 9 - Visée technique transformation 2

Transformation via la construction d'un schéma relationnel

- Construction de type relation 1-n

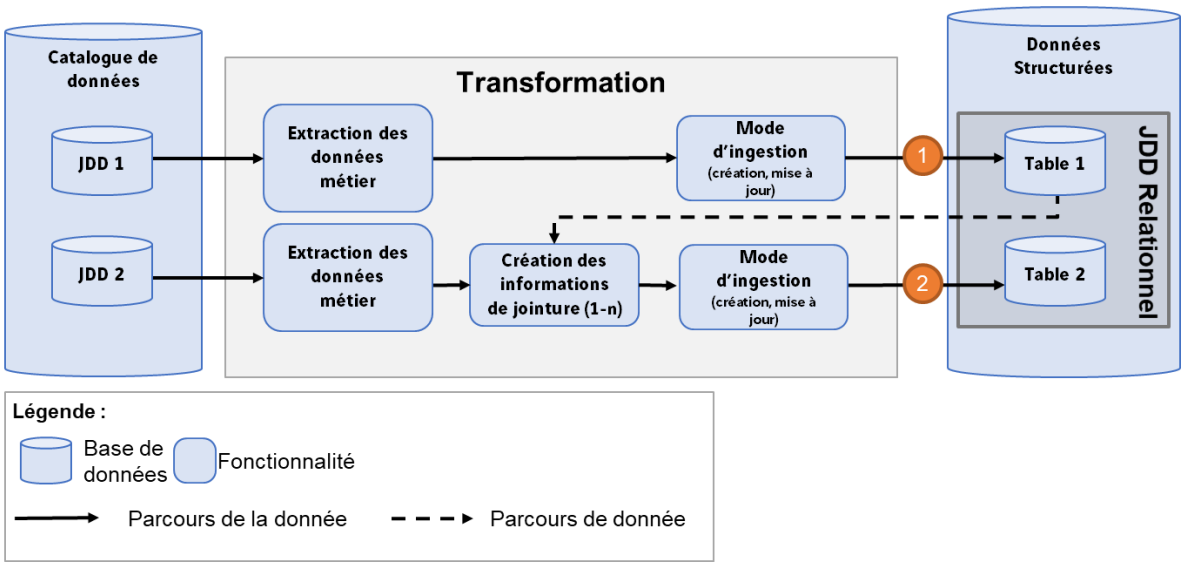


Figure 10 - Visée technique transformation 3

- Construction relation n-n

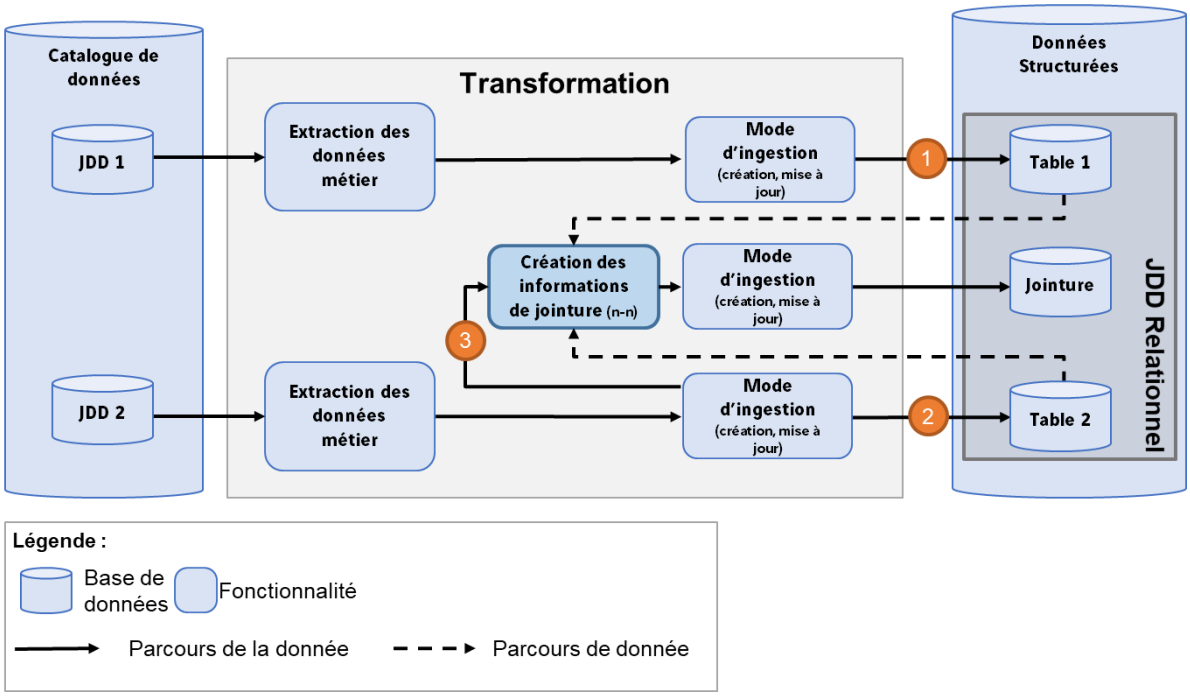


Figure 11 - Visée technique transformation 4

7.2.3.3 Choix applicatif

Dans le cadre de la transformation, NiFi permet de gérer et d'automatiser les traitements via la création de pipeline d'enchaînement de briques fonctionnelles avec comme option :

- Une interface de paramétrage avancée,
- Un large catalogue de briques fonctionnelles paramétrables existantes,
- La possibilité de rajouter et partager des briques fonctionnelles,
- L'intégration de scripts externes,
- Le paramétrage et la modification sans interruption de service.

La solution logicielle NiFi peut appeler des services scriptés en Python et Pandas (déjà disponibles ou à développer) ou des bibliothèques open source spécialisées dans la gestion en volume de données afin d'intégrer un enchaînement de règles sur le jeu de données à traiter demandant un niveau de traitement de complexité élevé.

7.2.3.4 Généalogie et traitements

Le suivi des traitements sur la donnée pourrait potentiellement être affiché sur le portail de supervision. Les STI (Spécifications Techniques d'Interface) définiront tous les traitements qui seront réalisés par et consultables sur l'outil NiFi.

Les métadonnées peuvent stocker toutes les informations de traitements sur la donnée :

- Producteur / Source
- Date

- Type de traitement
- Etc.

Les traitements pourraient être de plusieurs types avec des cibles utilisateurs différentes.

Ces informations doivent être disponibles pour l'utilisateur final à des fins de transparence.

La création de ces métadonnées pourrait être générée manuellement (à travers un formulaire en Markdown par exemple) et/ou automatiquement en partie.

Les métadonnées peuvent être liées à un jeu de données ou à un enregistrement de données (ex : Crowdsourcing avec un JDD avec des métadonnées sur le JDD et par ligne du JDD).

On pourrait implémenter un premier niveau de traçabilité niveau macro (vulgarisation du traitement) et un second niveau plus détaillé pour lister l'enchaînement des traitements réalisés (stockés dans un fichier, une bdd, etc.) sur demande utilisateur (ex : par formulaire).

Prendre exemple sur la généalogie SIG comme GeoSource (qui implémente la directive INSPIRE).

7.2.4 Stockage de la donnée

7.2.4.1 Visée fonctionnelle

L'étape de stockage intervient sur trois types de données : Lots de collecte (données brutes), catalogue de données (données vérifiées et structurées) et données structurées (transformées au plus près du cas d'usage).

Le stockage correct des données est généralement une exigence de conformité des directives telles que le Règlement général sur la protection des données (RGPD) imposé par l'Union européenne et pourra faire appel au chiffrement ou au traitement des données sensibles.

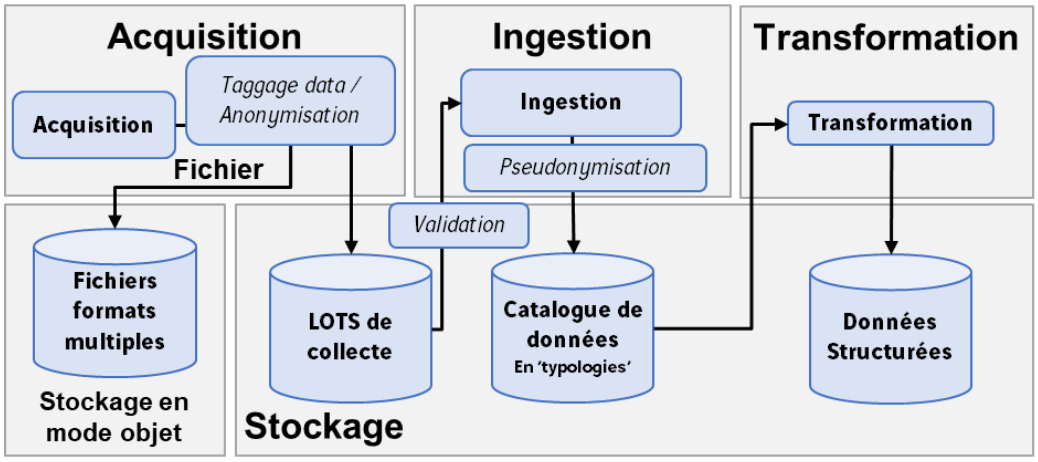
Le stockage doit aussi être optimisé au niveau des performances de mise à disposition pour l'utilisateur autour de la partie cas d'usage.

7.2.4.2 Visée technique applicative

L'objectif de la brique Stockage est de stocker les données en volume et de les mettre à disposition de manière sécurisée. Différentes typologies de données sont identifiées :

- Données référentielles, géolocalisées,
- Données temps-réel, IOT, industrielles,
- Données textuelles clé/valeur, formulaires,
- Données document type image.

On va retrouver quatre familles de stockage dans la plateforme :



7.2.4.2.1 Stockage des lots de collecte

La phase de stockage des données au plus près des producteurs reste en cours de définition sur les périmètres de la conservation, l'historisation et la méthodologie de stockage de ces données. La validation des choix applicatifs sera appuyée par les validations des définitions fonctionnelles.

Il n'y a pas d'accès en lecture ouverte à cette donnée à l'exception de procédure de validation des données et de la brique ingestion qui extrait un lot de collecte vers le catalogue de données.

Les lots de collecte dont la donnée :

- Est stockée et issue de connecteurs ou de la plateforme,
- Est dans un format brut normalisé, non transformé,
- Est identifiée et horodatée,
- N'est pas accessible en lecture ouverte.

Si la donnée récupérée respecte suffisamment les critères de qualité et ne nécessite pas de retraitement en vue de son ingestion, on peut éviter le stockage en lots de collecte et transmettre directement à la brique ingestion en conservant la phase de validation.

7.2.4.2.2 Stockage du catalogue de données

Les données nettoyées et validées passent par la création de tables en mode document pour permettre la création du catalogue des jeux de données. Elle représente le premier niveau de structuration de la donnée, orienté pour l'utilisation par la fouille de données et dédié aux scientifiques de la donnée.

La brique de stockage du catalogue de données :

- Constitue le catalogue de données de la plateforme avec un premier niveau de structuration,
- Fournit des typologies de stockage,
- Garantit la résilience de la donnée,
- Permet la fouille de données, la restitution et la transformation pour les cas d'usage le cas échéant.

7.2.4.2.3 Stockage des données Structurées

Création d'un ensemble de tables structurées en fonction des besoins des cas d'usage. Elle doit pouvoir fournir le support des typologies complexes pour fluidifier les utilisations dans les cas d'usage.

La brique de stockage des données structurées :

- Stocke les données consolidées pour les cas d'usage,
- Fournit les outils d'indexation complexes,
- Supporte les typologies de données,
- Garantit la disponibilité des données.

7.2.4.2.4 Stockage en mode **objet**

Certains jeux de données peuvent être rattachés à des données de formats type objet comme des documents PDF ou des images qui seront stockés dans un espace dédié. Le but est d'associer aux différents fichiers des métadonnées et de placer la référence des données de type objet dans le jeu de donnée stocké en base.

7.2.4.2.5 Stockage des données de référence

Plusieurs sources de données existent et l'objectif est de fédérer la donnée pour avoir un unique référentiel, une seule source de données homogène. Les référentiels pourront être dédiés par cas d'usage.

Les données de références seront stockées dans un SGBDR.

7.2.4.3 Choix applicatif

Dans un objectif de simplifier la maintenance et de minimiser le nombre de solutions applicatives déployées, on va chercher à capitaliser au maximum sur les solutions MongoDB et PostgreSQL.

Pour PostgreSQL, nous allons pouvoir étendre son périmètre en utilisant plusieurs modules, comme par exemple :

- PostGis pour le support des données géographiques.
- TimeScale pour la gestion des données temporelles.

Stockage des lots de collecte

Pour la partie lots de collecte, une base de données NoSQL est utilisé pour la souplesse et leur capacité de croissance horizontale. La solution MongoDB répond aux besoins et fournit les capacités de monter en charge pour supporter les flux de données entrants via la brique acquisition.

Stockage du catalogue de données

Ce type de stockage doit supporter des tables multi format pour couvrir toutes les typologies de données identifiées, avoir la capacité de supporter des données en volumétrie et fournir des

outils pour parcourir cette donnée. La solution MongoDB sera réutilisée pour couvrir ce périmètre et les besoins associés.

Stockage des données structurées

Ce type de stockage nécessite une structuration relationnelle et multi-typologique que PostgreSQL peut gérer pour permettre une disponibilité et une réplicabilité la plus adaptée au besoin de cas d'usage.

Ci-dessous, la liste des briques techniques répondant aux besoins de stockage de la donnée :

Composant / Framework	Type	Licence	Première version	Langage
PostgreSQL	Application	Licence PostgreSQL	9 Juillet 1996	C
PostGis	Extension PgSGL	GNU General	19 Avril 2005	C/C++
TimeScale	Extension PgSGL	Apache-2.0 License	30 Octobre 2018	C
MongoDB	Application	SSPL v1	18 Octobre 2011	Java

Étude applicative réalisée :

Nous faisons le choix de la technologie MongoDB par rapport à la technologie Cassandra, en ce qui concerne le stockage des lots de collecte et des catalogues de données.
La comparaison suivante entre Cassandra et MongoDB a été réalisée :

- Besoin technique : Souplesse pour accueillir le maximum de formats de données
  - MongoDB est une base de données orientée 'document' qui offre la souplesse nécessaire pour le multi format et le dynamisme requis par la plateforme. Cassandra demande la création des schémas avant ingestion.
- Besoin technique : Utilisation de protocoles standards (RGI 2.0) pour les requêtes en base de données
  - MongoDB utilise des requêtes sous le format standard JSON. Cassandra utilise son propre langage de requête CQL.
- *Autres avantages / inconvénients : MongoDB a l'avantage sur la gestion des requêtes, les agrégations, l'indexation, l'équilibrage de charge, la réplication. Cassandra propose des avantages sur les journaux des commits (enregistrement d'une transaction), des tables en mémoire et le support multi-master, ce ne sont pas des critères de choix pour la visée de la plateforme LRTZC.*

7.2.5 Stockage des objets métiers

7.2.5.1 Stockage de l'inventaire des objets

Plusieurs types d'objets seront stockés :

- Algorithmes,
- Modèle de restitution (dont la configuration),
- Restitution,
- Tableaux de bord,

- Formulaires.

La définition et les métadonnées de ces objets seront stockées dans un SGDBR.

7.2.5.2 Stockage des **logs**

La gestion des logs systèmes sont décrits dans la partie « Représentation technique – Infrastructure ».

Les logs liés aux actions utilisateurs pour de l'analyse web seront gérés par Matomo (fourni par la CdA de La Rochelle.

7.2.5.3 Stockage des consentements

Les consentements (ex : politique de gestion des données) seront stockés en base de données (ex : un booléen, true ou false).

7.2.6 Orchestration

7.2.6.1 Visée fonctionnelle

Les fonctionnalités d'orchestration doivent permettre de planifier, programmer, superviser et de conditionner l'ensemble des fonctionnalités relatives au cycle de vie des données sous la forme de processus techniques. L'objectif est de pouvoir au maximum automatiser les tâches d'administration et d'avoir un cycle de vie de la donnée automatisé de bout en bout. L'administrateur de la plateforme peut être amené à intervenir de manière ponctuelle dans ces processus dans le seul cas où une prise de décision humaine est nécessaire.

L'ensemble des opérations automatisées dans ce cadre produisent des rapports permettant d'en assurer l'exploitation et la traçabilité.

7.2.6.2 Visée applicative

La solution NiFi est utilisée pour gérer les briques d'acquisition, d'ingestion et de transformation autour de la gestion de chaîne de traitement de la donnée via des fonctions paramétrables et réplicables.

NiFi fournit nativement la majorité des fonctionnalités d'orchestration et de suivi des processus attendus.

Il existe au sein de NiFi une API disponible qui sera utilisée pour intégrer dans l'IHM d'administration des métriques pour la gestion de la qualité de la donnée, des états de pipelines, des contrôles quand une décision humaine est requise, etc.

La surface des règles éditables et configurables sera définie lors de la phase d'étude afin d'assurer l'optimum entre le paramétrage sans arrêt de service et la complexité des traitements à mettre en œuvre.

7.2.7 Archivage et Destruction

7.2.7.1 Visée fonctionnelle

Archivage

Les jeux de données présents sur la plateforme doivent pouvoir être archivés dans des systèmes d'archives électroniques (SAE). L'archivage peut porter sur tout ou partie d'un jeu de données (notamment dans le cas de données historisées) dont on souhaite archiver les valeurs les plus anciennes. L'archivage est conditionné à la durée de conservation définie pour chaque jeu de données. A défaut d'archivage, les données sont détruites.

La plateforme doit prévoir des connecteurs standards (exemple : CMIS) pour pouvoir exporter facilement ses données vers un logiciel tiers de type SAE. Après avoir été exportées, et avoir reçu la confirmation du système SAE en retour, les données sont détruites.

Les données archivées sont donc par définition inaccessibles. Les jeux de données entièrement archivés sont indiqués comme tel au niveau du catalogue (voir Lot 2).

Destruction

A défaut d'archivage, les données sont détruites en respectant des règles de conservation définies pour chaque jeu de données.

L'objet d'une destruction est divers :

- Le RGPD qui impose des délais de rétention sur des données à caractère personnel,
- La suppression de données à la suite de la demande de l'utilisateur,
- Les données avec un âge excédant la durée de conservation définie,
- Etc..

La destruction peut porter sur tout ou partie d'un jeu de données (en principe les données avec un âge excédant la durée de conservation). Elle consiste en la suppression logique des données concernées.

La définition des briques destruction et archivage répondent à des exigences légales qu'il faudra prendre en compte (ex: Données à caractère personnel).

La stratégie découlera aussi du type de jeu de données ou de la contractualisation validée en amont.

7.2.7.2 Visée applicative

La solution d'ETL NiFi déployée est à étudier dans le cadre de la création de règles d'archivage et de destruction planifiées avec des règles plus ou moins complexes.

Pour la suppression de données à la suite d'une demande utilisateur, un développement spécifique est probable si la solution NiFi ne nous permet pas assez de souplesse.

7.2.8 Focus données SIG

La création d'un tableau Excel est envisagé pour lister toutes les fonctionnalités de la partie SIG et définir :

- Les responsabilités,
- Le périmètre par système (plateforme LRTZC, SIG de La Rochelle, Geo17, IGN pour les fonds de plan, etc.),
- Les cas d'usage associés,
- Les indicateurs quantité pour aider le dimensionnement,
- etc.

Ci-dessous, les travaux préliminaires sur les périmètres SIG.

7.2.8.1 Visée fonctionnelle

Le référentiel géographique peut être constitué de :

- **Couches vectorielles**
- **Photos** : satellitaire, aérienne, etc.
- **Fonds de plans** vectoriel ou raster

Deux modes de récupération pour les fonds de plans standard du SIG sont envisagés :

- Via le service de l'IGN
- Par l'API Manager de la CdA de La Rochelle

Les données géographiques sont décrites selon 3 niveaux de description :

- Le **niveau géométrique,** qui décrit la forme et la localisation de l'objet : ce sont les données géométriques,
- Le **niveau sémantique**, qui décrit les informations permettant de caractériser l'objet géographique : il s'agit de sa définition et de ses données attributaires (nom, surface, type, nombre d'habitant ...),
- Le **niveau topologique** qui décrit les relations de l'objet avec ses voisins

7.2.8.2 Visée technique

On stockera principalement des données de type vectorielles et orthophotographies sur lesquelles on pourra faire des calculs orientés data (pas de fond de plans).

Une première orientation technique porterait sur la collecte et le stockage d'un référentiel vectoriel dans la plateforme pour des calculs légers comme la visualisation.

L'objectif principal est de constituer un référentiel géographique :

- Données vectorielles

- Orthophotographies / Ortho-images
- Raster

Les méthodes d'échanges pressenties pour la plateforme sont :

- La récupération de données vectorielles via API Rest ou via le format GeoJSON
- La récupération de données type raster et Orthophotographie via un service WMTS (Web Map Tile Service)

7.2.8.3 Choix applicatif

7.2.8.3.1 Collecte

Pour les collectes des données vectorielles disponibles dans des formats d'échanges "type export GeoJSON", l'utilisation de la solution NiFi est pressentie. Pour des besoins plus spécifiques d'échange avancé sur des formats géospatiaux, une phase de définition des besoins et de conception sera effectuée.

7.2.8.3.2 Stockage

Pour le stockage des données vectorielles dans le lot de collecte, la solution MongoDB propose des outils d'indexation et de recherche via la gestion des types de formes définies par les spécifications GeoJSON.

Pour le stockage avancé des données transformées ou complexse, on utilisera la solution PostGIS. PostGIS est une extension du SGBD PostgreSQL, qui active la manipulation d'informations géographiques sous forme de géométries (points, lignes, polygones), conformément aux standards établis par l'Open Géospatial Consortium. Il permet à PostgreSQL d'être un SGBD spatial (SGBDs) pour pouvoir être utilisé par les systèmes d'informations géographiques.

Les caractéristiques principales de PostGIS sont :

- S'appuient sur GDAL (raster), OGR (vecteur), PROJ.4 (projections), GEOS (géométrie)...
- Robustes et fiables (contrôle de cohérence)

7.2.8.3.3 Traitements et calculs

Le traitement des données rasters et celui des données vectorielles sont très différents et ne font pas appel aux mêmes outils, ni aux mêmes compétences. Le traitement des rasters et des données qu'ils contiennent est complexe. Il réclame des outils spécialisés. C'est pourquoi dans une majorité de cas ils ne servent que de fond de plan en tant que support à des couches vectorielles.

A contrario, les données vectorielles sont plus simples à manipuler, notamment à des fins d'analyse. Un algorithme permettra de faire le lien et de vérifier la cohérence topologique de données géographiques, notamment vis-à-vis de la superposition d'objets dans un même espace.

Deux niveaux de calculs sont pressentis :

- Calculs simples sur donnée type vecteur (voir PostGIS)
- Calculs experts avec des besoins qui apparaitront avec les Cas d'usage (voir Bureau virtuel et QGIS)

7.2.8.3.4 Restitution

De nombreux cas d'usage prévoient des restitutions sous forme de cartographie navigable du territoire. Le site web doit ainsi prévoir une fonctionnalité permettant de reprendre les éléments calculés au niveau de la brique logicielle SIG et de restituer ces informations sur le site sous la forme de cartographies interactives.

L'interface associée doit correspondre aux standards de navigation actuellement utilisés par les applications web. Il doit être ainsi possible de :

- Positionner manuellement la perspective par cliquer-tirer
- Zoomer et dézoomer via la molette de souris
- Cliquer sur les éléments de la carte pour afficher des informations relatives à cet élément
- Proposer d'éventuelles actions spécifiques par clic droit

Elle doit également permettre de représenter des objets standards tels que des bâtiments, des axes de communication, des éléments d'aménagement urbain, le patrimoine arboré, les sites industriels etc. L'ensemble de ces éléments doit être restitué grâce à des forme sur lesquels il est possible d'appliquer des filtres de couleur en fonction des propriétés de l'objet en question.

Chaque objet ou pin est cliquable et doit permettre d'afficher une fiche d'information complète s'y rapportant.

La restitution sera principalement orientée géo-services web avec l'utilisation de format standards :

- Données en GeoJSON
- Des flux en WMTS et WFS

Pour des besoins de respect de norme, la plateforme devra supporter la mise à disposition des données géospatiales dans les projections françaises **Lambert 93** ou **3946** (zone de La Rochelle) en plus du format GPS.

Pour la restitution et les reprojection des données géospatiales, l'outil GeoServer est pressenti.

7.3 Diffusion

7.3.1 Visée fonctionnelle

La diffusion doit pouvoir s'adapter automatiquement à la demande et répondre à des enjeux de charge.

Les données devront être mises à disposition de manière sécurisée, c'est à dire en respectant un système d'authentification, d'habilitations (droits) et de licences pour des utilisateurs humains ou des services techniques (typiquement un processus de Cas d'Usage).

La diffusion de données regroupe les fonctionnalités suivantes :

- Transmettre vers des systèmes tiers tout ou partie d'un jeu de données dans leur intégralité.
- Exporter les données via des formats standards (json/yaml/csv/…). La consommation de l'Api peut être réalisée par un système tiers ou par téléchargement disponible par un bouton sur une page web de la plateforme. Le téléchargement des données peut être soumis à plusieurs règles.
- Mettre à disposition des interfaces et connecteurs standardisés permettant à un système tiers de s'abonner à la publication de nouvelles informations.

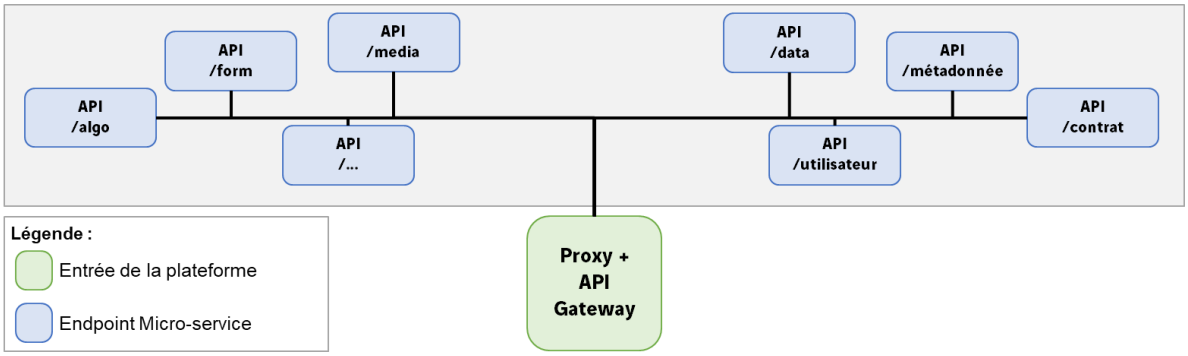
Un système externe peut consommer la donnée à partir de la brique de diffusion et sera soumis à la création d'un compte pour l'identification et la gestion des droits.

Plusieurs types d'abonnement existent, comme par exemple :

- Webhook (Push) : url d'un système tiers appelée lors d'un évènement généré par la plateforme (mise à jour des données).
- Abonnement à des points d'intérêt dans la plateforme pour les utilisateurs (par exemple pour notifier un utilisateur par e-mail de nouveaux cas d'usage ou de restitutions, d'indicateurs voire de news sur un métier).
- Abonnement à des services API (Pull) par des systèmes tiers ou des développeurs via la création de compte.

La gestion des abonnements aux APIs avec plans d'API pour la contractualisation et la gouvernance des données est à définir à la suite du MVP.

7.3.2 Visée technique applicative



Toute connexion aux données passe par un proxy qui va gérer le SSL et une API Gateway qui va se charger du routage des demandes utilisateurs et de la répartition de charge automatique sur les micro-services.

Les points d'accès fonctionnels sont regroupés dans des micros-services séparés pour permettre une scalabilité plus fine sur les services les plus consommateurs en ressource et optimiser la consommation des ressources.

La majorité des services d'APIs sont StateLess, ce qui signifie qu'ils ne contiennent aucune donnée de contexte sur disque. On peut donc avoir plusieurs instances logicielles de ce service opérationnelles en même temps, n'importe laquelle de ces instances sera en mesure de traiter une requête qui lui sera envoyée.

Les accès aux différents types de données (jeu de données, BDD, fichiers, etc.) sont contrôlés. Cette action sera déléguée à une API dédiée à la gestion des contrats et licences, afin de vérifier les droits d'accès selon les contrats et licences.

Les choix techniques s'orientent vers :

- Le protocole REST et des API dites "techniques" pour favoriser l'interopérabilité,
- La mutualisation des fonctionnalités communes,
- La possibilité d'intégrer des API dites spécifiques pour s'adapter au cas par cas,
- Des sous API "de description" permettant aux systèmes tiers d'obtenir le détail des modèles de données stockés.

7.3.2.1 Interfaces & APIs

Les contrats d'interface sont amenés à évoluer, donc ci-dessous seront présentés les principaux Endpoints.

Global aux APIs

Authentification et Sécurisation

L'authentification de l'utilisateur est réalisée au travers d'un token d'accès créé par l'application de gestion des utilisateurs et des groupes. À chaque appel à l'api, l'application devra fournir le token qui sera validé au travers de la clé de l'autorité d'authentification.

Traçabilité

Chaque API interagissant avec les données devra journaliser les actions effectuées sur la donnée via un collecteur. Elle a pour but de garantir l'intégrité et la traçabilité de la donnée interservices de la plateforme.

Ce mécanisme est fondé sur un identifiant unique associé à chaque jeu de données dans les tables de référence. Chaque action est historisée dans la base de données "Traçabilité des actions" sous un format normalisé (sans stockage du contenu) :

- Horodatage de l'action,
- Service associé / Action,
- Identifiant unique du lot,
- Identifiant du demandeur.

Données et Algorithmes

data

Tous les accès à la donnée sont contraints à une validation des droits et des consentements. Cela se traduit par :

- L'accès au Endpoint /data ne se fait qu'en mode impersonnification. L'impersonnification consiste à utiliser l'identifiant du consommateur pour effectuer les requêtes de sélection de données.
- Toutes les données, accessibles via l'Endpoint /data, sont modélisées en base en portant une colonne 'IdentifiantProducteur'. Cet identifiant permet de retrouver la source qui a produit cette donnée.
- Les requêtes de sélection de données utilisent le champ 'IdentifiantProducteur' pour cibler le résultat sur la liste des producteurs auquel le consommateur à droit.

algo

Les accès aux différents algorithmes se font au travers de l'Endpoint /algo, qui va se charger de vérifier si l'utilisateur a les autorisations pour accéder à l'algorithme puis de transmettre la demande à l'algorithme concerné en préservant l'identifiant du consommateur.

Lorsqu'un algorithme doit travailler avec des données de la plateforme, deux modes de connexion à /data existent :

- Utiliser l'impersonnification pour accéder aux données au nom du consommateur
- Utiliser son propre identifiant de consommateur

De plus, un algorithme peut être statefull (mis en cache et réutilisé). Cela permet, par exemple, d'exécuter des calculs périodiques et de garder les résultats dans un contexte de sauvegarde propre à l'algorithme.

A chaque fois qu'une nouvelle fonctionnalité touche à des données à caractère personnel, une analyse d'impact et des solutions techniques seront à définir (ex : durée de mise en cache).

Les outils de cache devront prendre en charge la gestion de la durée de vie du cache.

pipeline

Lorsque le besoin de traitement sur la donnée fait intervenir une chaine d'algorithmes, le traitement se fera par la délégation à un service de traitement de pipeline qui se chargera d'effectuer les actions au nom de l'utilisateur pour conserver la traçabilité à chaque étape.

Elle sera utilisée principalement dans la création de restitution complexe. Le formalisme des paramètres d'entrée sera construit lors de la phase de réalisation en lien avec le lot 2.

Synthèse des différences entre data et algo :

	Impersonnification	***Identifiant de consommateur en propre***
/data	*Oui (obligatoire)*	*Non*
/algo	*Oui (possible)*	*Oui (possible)*

Un algorithme peut prendre en paramètre un /data comme argument (par exemple pour appliquer un calcul de moyenne sur un jeu de données).

Des algorithmes seront dédiés pour gérer les pipelines permettant la construction et l'enchaînement d'appels d'algorithmes pour le traitement des données ayant pour avantage la parallélisation et la gestion des process complexes.

Afin de soulager la couche présentation pour l'utilisateur, l'accès aux données fera appel à un mécanisme de pagination. Si le besoin est exprimé, la mise en place d'un accès en mode stream aux données pourra être implémentée.

Accès aux descriptions

La plateforme se structure autour d'un ensemble de tables de références qui vont permettre aux systèmes tiers d'obtenir le détail des modèles de données stockés. Elles vont entre autres être constituées des référentiels suivants :

- Utilisateurs
- Autorisations
- Sources de données
- Jeux de données
- Algorithmes
- Métadonnées
- Licences et consentements
- …

Gestion des contrats

La gestion des contrats regroupe les trois fonctionnalités suivantes :

- Smart-contract
- Consentement
- Licences

Formulaire de crowdsourcing

Un autre point d'entrée de la donnée dans la plateforme est le crowdsourcing qui permet à un utilisateur de contribuer à des jeux de données par l'intermédiaire de formulaires ayant été préalablement construits par un administrateur de la plateforme.

Suivant le type de donnée et la licence du jeu de donnée concernée, il faudra demander le consentement de la personne contributrice et tracer la donnée qui lui est liée pour garantir la bonne exécution de :

- son droit de suppression des données
- sa durée de conservation des données.

Un connecteur sera instancié pour chaque type de jeu de données collecté depuis les formulaires (csv, json).

7.3.2.2 Stabilité de l'API

La compatibilité ascendante ajoutera de la stabilité à l'API, cela signifie que le code développé pour une version continuera à fonctionner avec les publications futures.

Entre autres, la stabilité de l'Api devra respecter les principes suivants :

- Les Endpoints API publiques ne seront ni déplacés ni renommés sans mettre à disposition des alias rétro-compatibles.
- Des nouvelles fonctionnalités ajoutées à un Endpoint API ne vont ni casser ni modifier le fonctionnement des méthodes existantes.
- Si une API doit être supprimée ou remplacée, elle sera déclarée obsolète mais restera dans l'API durant au moins deux publications principales. Des avertissements seront émis lors des appels aux méthodes obsolètes. Pour garantir la rétro-compatibilité, on ne supprime pas les APIs immédiatement pour laisser le temps aux développeurs de prendre en compte les mises à jour d'API

7.3.2.3 Abonnement à l'API

L'accès aux APIs est contrôlé par une solution d'authentification de type SSO qui va permettre de déterminer le niveau d'accès et d'habilitation des demandes. De plus, la plateforme ayant un fort enjeu de contrôle d'accès sur les données et de l'exploitation de smart contract, la plateforme privilégie les accès nominatifs.

Sur les endpoint API gérés par la plateforme, des règles de limite de flux seront à définir pour éviter l'utilisation par des robots agressifs qui pourraient nuire la plateforme.

A la suite de la phase de MPV, des besoins d'accès spécialisés pour les robots moissonneurs légitimes apparaitront avec des modes d'authentification adaptés à valider. Une des possibilités disponibles à l'étude serait de se reposer sur la solution d'API Manager de La Rochelle pour les plans d'abonnement sur les endpoint liés au moissonnage.

7.3.3 Choix applicatif

Notre solution capitalise sur l'expérience de Citeos Paris Solutions Digitales avec la mise en œuvre de plateforme technique de données se basant sur une architecture REST développée sur un serveur Springboot. Les API de mise à disposition sont normalisées techniquement permettant de facilement retrouver la donnée en fonction de métadonnées telles que la source et la catégorie de la donnée.

Cette plateforme intègre :

- Spring Boot pour la création des API ;
- Le module Spring Security pour supporter l'authentification portée par l'outil Keycloak.
- Un développement spécifique pour la gestion des licences.

Composant / Framework	Type	Licence	Première version	Langage
Spring	Framework	Apache-2.0 License	25 juin 2003	Java
Spring Boot	Framework	Apache-2.0 License	1 Avril 2014	Java
Spring Security	Framework	Apache-2.0 License	30 Mai 2006	Java

Étude applicative réalisée :

- Besoins techniques :
  - Échanger des données/informations entre des modules du back-end et des consommateurs (autres modules du back-end, front-end, consommateurs externes…) de ces données/informations. Le format privilégié est celui du REST HTTPS.
  - S'adapter à la demande et répondre à des enjeux de charge (montée en charge dynamique des services et répartition de charge)
  - Mettre à disposition de manière sécurisée des données, c'est à dire en respectant un système d'authentification, d'habilitations (droits) et de licences : Ces phases de contrôles seront effectuées dans chaque micro-service pour répondre aux exigences de sécurité et de traçabilité fonctionnel.
  - Consommation par des système externes de la donnée à partir de la brique de diffusion et sera soumis à la création d'un compte : La nécessité de tracer les actions et la vérification des droits et licences impose de suivre le mode d'authentification de la plateforme.

Notre choix se porte sur :

- L'utilisation d'un gestionnaire de trafic : routing des demandes type 'API Gateway' utilisant la technologie open source Traefik https://doc.traefik.io/traefik/
- Le Framework open source SpringBoot pour le développement des Endpoint
- L'orchestrateur Kubernetes pour assurer la scalabilité et la montée en charge de la brique Diffusion
- Le gestionnaire open source d'authentifications et de droits KeyCloak

Ces choix garantissent la couverture des besoins techniques et des besoins spécifiques au projet de traçabilité forte des accès à la donnée, de la validation des licences et consentements.
L'utilisation d'un API manager ne permet pas à ce stade de garantir la réponse à ces enjeux technico-fonctionnels spécifiques de la plateforme.

Pour autant, la solution proposée est évolutive et non bloquante pour l'intégration d'autres technologies dans le futur : le besoin d'abonnement aux APIs par des systèmes tiers pourrait passer par l'utilisation d'un API Manager produit open source (ex : composant Gravitee.io tel que celui actuellement utilisé par la CDA).

7.4 Gestion des utilisateurs

7.4.1 Visée fonctionnelle

7.4.1.1 Définition

D'une manière générale, on différencie 3 principaux types d'utilisateurs :

- Utilisateur non authentifié de la plateforme
- Personne physique disposant d'un compte personnel sur la plateforme :
  - Contributeur individuel
  - Producteur institutionnel
  - Administrateur délégué
  - Créateur / Réutilisateur
  - Administrateur fonctionnel
  - Administrateur technique
- Consommateur d'une API mise à disposition par la plateforme

7.4.1.2 Caractéristiques

Les utilisateurs authentifiés de la plateforme sont caractérisés par :

- **Profil** : attributs de l'utilisateur (nom, email, coordonnées, …)
- **Rôle** : ensemble de permissions d'accès à des fonctionnalités de la plateforme
- **Groupe** : ensemble d'utilisateurs auquel peut être associé un rôle et/ou un smart contract
- **Organisation** : certains utilisateurs sont rattachés à une organisation (ou personne morale) et peuvent être administrateurs délégués de la plateforme au sein de leur organisation
- **Consentements** : valider ou résilier l'accès à ses données personnelles ; gestion des licences apposées à ses production intellectuelles ; gestion de ses Smart Contract
- **Consommations d'API** : assignation d'autorisations spécifiques pour l'utilisation des API portail (volumétrie d'appels, etc.)

Les droits des utilisateurs sur la plateforme sont gérés par un mécanisme de groupe à travers d'une IHM d'administration.

7.4.1.3 Compte utilisateur

Un utilisateur pourra non seulement créer un compte en mode self pour accéder à la plateforme mais aussi passer par une authentification déléguée pour utiliser par exemple un compte France Connect. Il est envisagé d'intégrer dans un second temps LaRochelleConnect.

Selon une durée (à définir, par exemple : un an), si aucune nouvelle connexion n'est détectée, le compte pourrait devenir inactif (ou directement supprimé ou répondre à un workflow).

Il faudrait aussi détecter et afficher les informations des comptes inactifs sur un outil dédié.

La suppression d'un compte inactif entraîne la suppression des données personnelles (même si ces données personnelles sont actuellement utilisées dans un cas d'usage).

7.4.2 Visée Technique

7.4.2.1 Gestion des droits

La gestion des utilisateurs sera gérée par un système de contrôle d'accès aux ressources de la plateforme via les groupes utilisateurs.

Chaque appel d'Endpoint d'API à travers son verbe HTTP associé (consulter, modifier, supprimer ou créer), sera contrôlé par une gestion de droits d'accès via une table de référence associant les groupes, les Endpoint, verbes http et les droits associés.

7.4.2.2 Authentification unique

Chaque nouveau type de connexion via des systèmes tiers (ex : France Connect) passera par une contractualisation avec le fournisseur pour une intégration dans la plateforme. L'affichage des champs (pouvant contenir des données personnelles) sera spécifique à chaque service.

Le scope fonctionnel porté par l'outils de gestion d'authentification unique pourra concerner le contrôle des ACLs, Contrôles des accès API, etc.

Les pages de création de compte, d'authentification et de changement d'informations de compte (ex : mot de passe) pourront passer par des nouvelles pages chargées ou dans une intégration complète avec une personnalisation qu'il faudra définir.

Nous mettrons en priorité le SSO par FranceConnect dans un esprit de généricité, puisque la solution sera disponible en open source. Il est envisagé d'intégrer dans un second temps LaRochelleConnect.

7.4.3 Choix applicatif

Keycloak est une application permettant de déléguer la phase d'authentification sécurisée de n'importe quelle application web moderne avec un apport minimum en termes de code. Il apporte en autre les fonctionnalités suivantes :

- Inscription des utilisateurs
- Authentification unique
- Authentification en deux étapes
- Délégation d'authentification

- Intégration Lightweight Directory Access Protocol (LDAP)
- Customisation des interfaces utilisateurs par un système de thème
- API pour piloter les fonctionnalités

Pour la partie support de la solution étatique d'authentification France Connect, l'Insee a déjà construit une extension pour Keycloak qui ajoute le fournisseur d'identité France Connect disponible sous la licence MIT : https://github.com/InseeFr/Keycloak-FranceConnect

Il est aussi envisagé d'implémenter LaRochelleConnect.

	Type	Licence	Première version	Langage
Keycloak	Application	Apache-2.0 License	10 Septembre 2014	Java
Keycloak-FranceConnect	Extension Keycloak	MIT License	19 Décembre 2018	Java

7.5 Algorithme

Les algorithmes sont des outils développés en open source pour effectuer des traitements complexes sur les données de la plateforme. Chaque algorithme doit être construit dans un conteneur docker autoporteur et respecté les conventions de la plateforme (Sécurité, contrat, interopérabilité, …).

Le "fournisseur d'algorithme" ne met pas à jour ses algorithmes via la plateforme mais via les mécanismes de déploiement continue mis en œuvre par la vision DevOps. Les algorithmes seront validés par les administrateurs technique et fonctionnel avant d'être ajoutés dans la bibliothèque des algorithmes. Il se découpe en deux groupes :

- Les algorithmes producteurs de données autonomes
- Les algorithmes de calculs utilisés via API REST

Les traitements algorithmiques seront à renseigner dans un registre de la collectivité avec la description de son fonctionnement et des traitements réalisés sur les données. Par la suite, une analyse de besoin AIPD sera réalisée pour chaque traitement touchant à des DCP.

Cas particulier des contribution externes

Les processus de contribution externe par les équipes techniques avant intégration dans la plateforme depuis par exemple GitLab seront décrit dans le fichier CONTRIBUTING.md de la forge NAOS.

Ci-dessous, un exemple d'arborescence sur laquelle on pourrait s'inspirer :

CONTRIBUTING.md

Objectifs
Guide de contribution, comment s'impliquer et identification du processus de contribution et des licences associées.

Rôle et contribution
Rédaction par Lot 1, Lot 2 / Relecture et validation par CdA

Format : Markdown
Vision : Mise à jour sur chaque évolution du service numérique

PLAN DU FICHIER

1. How to contribute?
   1. See README.md
   2. Get the required software
   3. Set up for development on Windows
   4. Configure Git for contributing
   5. Work with a development container
   6. Run tests and test documentation
2. Topics
3. Reporting security issues (lien vers @email)
4. Quick contribution tips and guidelines
   1. Pull requests are always welcome
   2. Design and cleanup proposals
   3. Conventions
   4. Successful Changes
   5. Commit Messages
   6. Review
   7. Merge approval
   8. Sign your work
   9. How can I become a maintainer?
5. Community guidelines (see GOVERNANCE.md)
6. Coding Style

7.5.1 Algorithme producteur de données

Les algorithmes producteurs permettent de construire de nouvelles données via la consommation de données disponibles dans la plateforme. Chaque algorithme producteur utilise un identifiant de consommateur qui lui est propre et est soumis aux mêmes règles de contractualisation sur la donnée qu'un utilisateur.

Chaque algorithme de production de données sera attaché à une procédure d'acquisition et d'ingestion. Plusieurs algorithmes peuvent alimenter le même jeu de données.

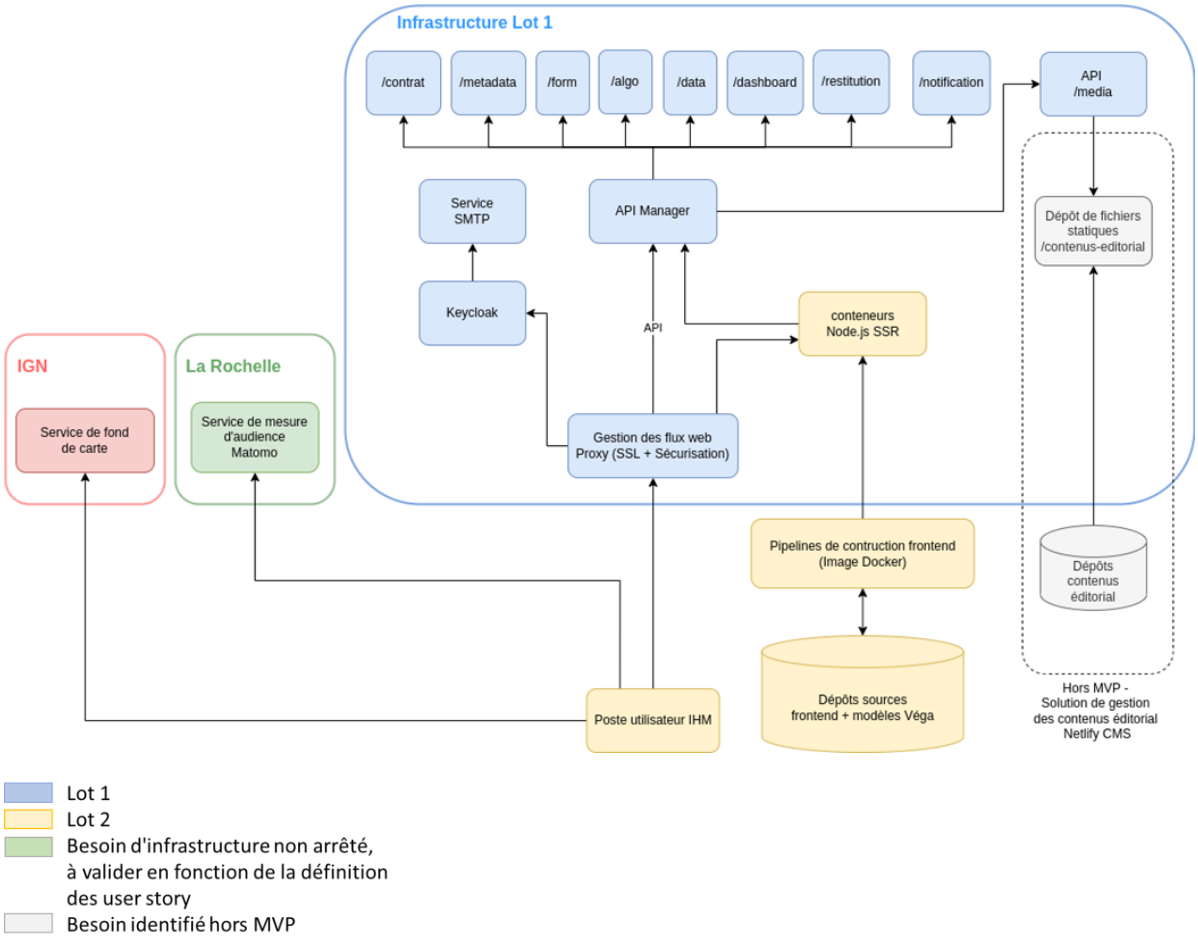
7.5.2 Algorithme de calcul (API)

Les Algorithme de calcul sont des outils disponibles via des accès API pour effectuer des traitements et calculs à la volée sur les données sans création d'un nouveau jeu de donnée.

Lorsque les algorithmes de calcul effectuent leur premier démarrage, ils s'enregistrent sur l'endpoint /algo afin de stocker leurs références dans le catalogue d'algorithmes et aussi permettre le suivi de leurs statuts. Ces algorithmes une fois référencés dans la base de données pourront être appelés ultérieurement à travers la brique de diffusion.

7.6 Restitution et lot 2 (back -> front)

7.6.1 Architecture (perspective front)



L'application web se présente sous la forme d'une application à page unique. Un service Node.js déployé sur l'infrastructure du lot 1.
L'application consomme les services hébergés par l'infrastructure du lot 1 ainsi que les services de mesure d'audience de La Rochelle et le service de fond de carte IGN.

Nous prévoyons de définir dans le proxy une route vers le serveur node.js du lot 2 pour permettre aux utilisateurs de récupérer le front et les modèles de restitution.

7.6.2 Mesure d'audience

Le service de mesure d'audience est basé sur la solution Matomo. Le service est exposé par la Communauté d'Agglomération de La Rochelle.

7.6.3 Valorisation / Restitution

7.6.3.1 Restitution diagramme

Des modèles prédéfinis sont stockés dans le front (métamodèle JSON et fichiers Javascript porté par la bibliothèque VEGA dans un docker). Pour la définition des restitutions, les éléments suivants seront utilisés.

- Liste de choix de depuis une table de référence > Endpoint de gestion réalisé par le Lot 1 appelé par le Lot 2 (stockage via un champ type JSON dans le PostgreSQL)
- Catalogue de restitution depuis une table de restitution > Endpoint de catalogue réalisé par le Lot 1 appelé par le Lot 2 (Relation avec algo/jdd/modèles/… et stockage des paramètres spécifiques dans un champ type JSON)
- Liste des préférences utilisateurs depuis une table de référence > Endpoint de gestion réalisé par le Lot 1 appelé par le Lot 2 (stockage via un champ type JSON dans le PostgreSQL)

7.6.3.2 Restitution cartographique

Notre solution SIG exposera des API cartographiques consommables par le lot 3 et le lot 2 via l'API Diffusion pour intégrer les problématiques :

- D'authentification et d'identification (traçabilité)
- De vérification des habilitations
- De vérification des licences
- De scalabilité

Le lot 2 utilise la technologie OpenLayers pour la partie frontale.

La reconstitution peut se faire côté front ou back (notamment la gestion des tuiles et des vecteurs), OpenLayers sait faire les deux.

Les caractéristiques principales des SIG sont :

- Représentation matricielle : rasters
  - Les Rasters sont des images (plans scannés, photographies aériennes, images satellitaires) repérées dans l'espace
  - Une matrice divise le territoire avec une grille régulière de cellules pour former une matrice de lignes et de colonnes sur plusieurs niveaux.
- Représentation vectorielle : vecteurs
  - Le format vectoriel utilise le concept d'objets géométriques (points, lignes, polygones) pour représenter les entités géographiques et gérer leur topologie.
  - Ces objets géométriques sont définis par leurs coordonnées dans un système de projection :
    - Localisation précise des objets
    - Possibilités de modélisation plus poussée
    - Facilité de manipulation des objets

- Les autres informations décrivant la donnée géographique sont portées par ses attributs (données attributaires).

Les caractéristiques liées à la représentation cartographique dépendront de l'usage utilisateur :

- Grand public : usage simple
  - Afficher et interagir avec une carte (zoom / déplacement) afin de consulter une représentation cartographique des données
  - Sélection du fond de carte
  - Clic sur un PI de la carte afin d'afficher plus de détails
  - Clic sur le graphique afin d'afficher les données brutes associées
- Cas d'usages LRTZC : à définir lors des spécifications du lot 3

7.6.4 Conventionnement

Le système de « smart contracts » est destiné à établir et automatiser un conventionnement entre producteur et réutilisateur : gestion du contrôle d'accès apposé aux jeux de données en shared data et gestion des licences de propriété intellectuelle. Ce système permettra d'expliciter les conditions d'utilisation liées aux jeux de données et restitutions. Une traçabilité des réutilisations permettra au producteur de s'assurer de la bonne utilisation de ses données.

La gestion des contrats regroupe les trois fonctionnalités suivantes :

- Smart-contract
- Consentement
- Licences

La conception technique sera réalisée durant le développement de cette fonctionnalité.

Les spécifications fonctionnelles des « smart contracts » n'étant pas encore rédigées, cette partie est susceptible d'être complétée à l'issue de ces travaux.

7.7 Bureau virtuel

Notre solution est la mise en œuvre d'un **ensemble de solutions standards Open Source** intégrées dans un service de gestion de valorisation permettant de gérer les accès sécurisés à ces solutions standards et **d'intégrer des services tiers conteneurisés**. Les outils pressentis sont les suivants :

- SIG : QGIS
- Dashboarding : Grafana
- Fouille de données : Jupyter

Ces solutions sont décrites par la suite. Elles seront confrontées au besoin émis par la CDA.

Pour capitaliser sur la mise en œuvre de ces solutions, le bureau virtuel consistera à la création des **sessions d'utilisation** à partir de ces outils. L'utilisateur aura accès à une application web présentant les produits et services mis à sa disposition : Grafana, QGIS, Jupyter, etc.

Deux approches sont possibles :

- Création de sessions sur des outils mutualisés
- Déploiement d'un écosystème applicatif par session utilisateur

Les spécifications fonctionnelles du « bureau virtuel » n'étant pas encore rédigées, cette partie est susceptible d'être complétée à l'issue de ces travaux.

7.7.1 Brique Dashboard

Pour les outils de Dashboarding, notre solution intègrera l'outils **Grafana**. Grafana est une solution open-source de **data visualisation**, fortement orientée timeseries mais permettant d'intégrer des **données de tous types**, notamment géographiques simples. Par ailleurs, Grafana est conçu autour d'un **système de banque de plugins développée par une communauté d'utilisateurs** permettant de développer et d'intégrer de nouveaux plugins spécifiques pour des rendus plus complexes (cartographiques, heatmap, ...)

Composant / Framework	Type	Licence	Première version	Langage
Grafana	Application	Apache-2.0 License	2014	Go et TypeScript

Grafana intègre l'authentification SSO et notre **gestion d'accès sécurisé** KeyCloak. Fortement configurable, cette solution permet une **forte acculturation des utilisateurs à la manipulation de premier niveau de la donnée** et permet une gestion de partage de Dashboard entre utilisateurs.

- ✓ **Scalabilité :** Cluster Grafana
- ✓ **Sécurité : I**ntégration KeyCloak, OpenLDAP.
- ✓ **Interopérabilité** : compatible Safari, Firefox, Chrome. Intégration de plugins (standard ou à façon). Interopérabilité native à Grafana en sources de données (PostgreSQL, Influx DB, ...)
- ✓ **Réplicabilité** : outil standard Open Source, fortement paramétrable par l'utilisateur

7.7.2 Brique SIG

QGIS est un système d'information géographique permettant de valoriser la donnée à forte complexité géographique. Conçue pour s'intégrer avec PostgreSQL et PostGis, notre solution de stockage, QGIS permet de créer des couches métiers à forte valeur ajoutée.

Notre solution intègrera un client web avec des fonctions d'exploration allégées pour permettre une première analyse de la donnée par des experts SIG, au sein du bureau virtuel.

Composant / Framework	Type	Licence	Première version	Langage
QGIS	Application	GNU-2.0 License	Juillet 2012	C++ et Pytho

- ✓ **Scalabilité :** déploiement conteneurisé
- ✓ **Sécurité :** Accès sécurisé sur périmètre adapté aux bases de stockage, authentification utilisateurs intégrée (plugin)
- ✓ **Interopérabilité** : Outil standard SIG pour le Lot 3 et Lot 2 + Respect du standard INSPIRE
- ✓ **Réplicabilité** : Solutions open-source

7.7.3 Brique Fouille de données

Notre solution intègrera un **serveur Jupyter Hub** permettant de gérer des environnements de ressources et de calculs liés à la plateforme pour accéder à des Notebook Jupyter : Un notebook donne la possibilité à un utilisateur de programmer et tester dans des environnements adaptés.

Cette solution est très utilisée dans le domaine de la recherche et dans les universités pour explorer des données et mettre en place des algorithmes de démonstration lisibles et documentés. Il permet de donner accès à des outils de modélisations et de visualisations classiques en Machine Learning dans le langage Python et les bibliothèques associées.

Jupyter Hub intègre l'authentification SSO et notre gestion d'accès sécurisé KeyCloak.

Composant / Framework	Type	Licence	Première version	Langage
Jupyter hub	Application	BSD-3 License	2014	Python

- ✓ **Scalabilité :** Cluster Jupyter
- ✓ **Sécurité :** intégration KeyCloak, accès sécurisé à la base
- ✓ **Interopérabilité** : compatible Safari, Firefox, Chrome. Compatible Python, R.
- ✓ **Réplicabilité** : outil standard Open Source

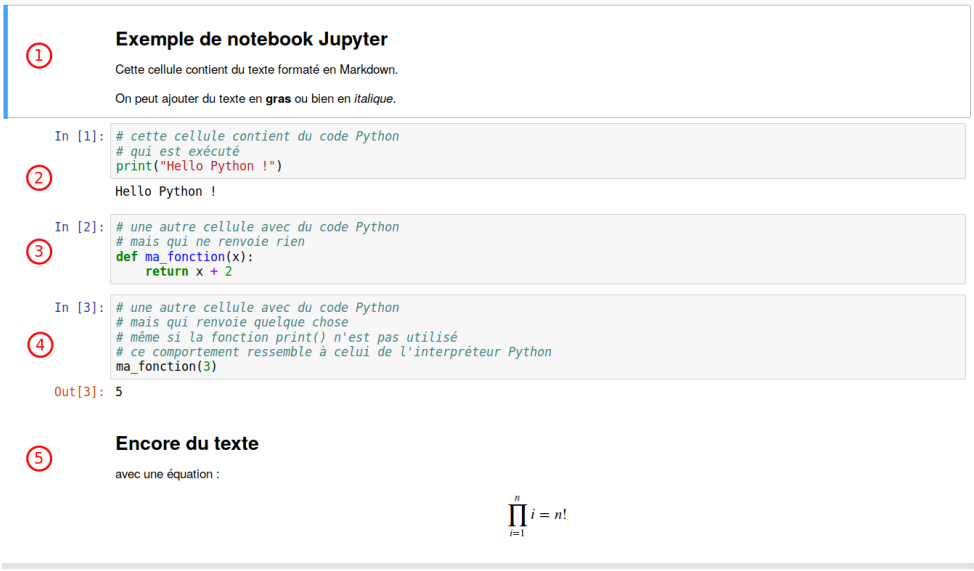


Figure 12 - Exemple de Notebook Jupyter

7.8 Conteneurisation et orchestration

7.8.1 Conteneurisation

Kubernetes, k8s (pour k, 8 caractères, s) ou encore « kube », est une plateforme Open Source qui automatise l'exploitation des conteneurs Linux. Elle permet d'éliminer de nombreux processus manuels associés au déploiement et à la mise à l'échelle des applications conteneurisées.

7.8.2 Scalabilité (applicative)

L'architecture est instanciée au travers de l'orchestrateur Kubernetes qui propose des outils pour simplifier les processus de montée en charge (load-balancing) et de résilience des applications par l'intermédiaire d'un mécanisme de basculement (failover).

7.9 Focus sur les IHMs

7.9.1 Authentification unifiée

LaRochelleConnect / FranceConnect

Nous mettrons en priorité le SSO par FranceConnect dans un esprit de généricité, puisque la solution sera disponible en open source. Il est envisagé d'intégrer dans un second temps LaRochelleConnect.

Double authentification

Une double authentification (à deux facteurs / A2F) pour accéder à des fonctions sensibles, comme l'accès au Back Office technique, peut être envisagée par l'envoi d'un SMS, d'un email ou d'une authentification via client OTP par smartphone.

7.9.2 Description générale

Une authentification unifiée peut être envisagée avec sur authentification, une redirection vers le portail idoine.

Trois accès sont pressentis :

1. Front Office grand public
2. Back Office fonctionnel
3. Back Office technique

Il y aura des URLs différentes (en utilisant des sous domaines, par ex : admintech.lrtzc) :

- Une seule URL pour Front office et Back office administration fonctionnelle avec gestion de droits d'accès et double authentification pour admin).
- Une URL pour le Back Office dit technique

Un outil spécifique supplémentaire dit « Bureau virtuel » pourra s'ouvrir depuis un bouton disponible dans le Front Office grand public (si accès autorisé, voir rôle utilisateur contributeur).

Remarque : Il est à noter aussi une volonté de ne pas recréer la roue, le but n'est pas de se substituer ou de développer des briques existantes. Par exemple, pour NiFi, on ne va pas sur le Back Office Technique proposer une fonctionnalité de gestion de pipeline (cœur de NiFi), mais plutôt récupérer des indicateurs permettant d'afficher entre autres le nombre de processus en cours.

7.9.3 Fonctionnalités pressenties

Exemple de Back Office technique

- Supervision de la donnée
    - Connecteur, rejets, qualité de la donnée, etc.
    - Gestion du cycle de vie de la donnée (ex : seuil et conditions par brique, suppression et archivage)
    - Monitoring de la donnée : mesures, alarmes, etc.
    - Gestion des métadonnées

    - Traçabilité des données
- Supervision plateforme
    - Applicative : disponibilité services applicatifs, applications et processus métier (exemple outil métier Nifi : nombre de processeurs) (Kubernetes, serveurs web, bdd, etc.)
    - Infrastructure, réseaux (disponibilité service, débit goulot étranglement, sécurité, gestion des flux, etc.) et machines physiques et VM (ressources, processeur, mémoire, stockage, etc.)
    - Journalisation (logs techniques)
- Administration technique
    - Gestion des habilitations
        - Gestion des rôles
        - Gestion des comptes techniques et fonctionnels
    - Voir partie « Autres » (car peu utilisé et par peu de personnes)
        - Gestion des droits RGPD
        - Gestion des accès au bureau virtuel
        - Gestion des notifications
        - Gestion des licences
        - Gestion des smarts contracts
        - Gestion des données de référence (ex: liste type objet disponible dans catalogue)

Exemple de Bureau virtuel (Fonctions avancées / Utilisateur expert)

- SIG
- Fouille de données (exploration)
- Dashboarding : Grafana
- Jupyter

Exemple de Back Office fonctionnel (utilisé par exemple par Alix)

- Gestion des comptes utilisateurs et invitation
- Gestion des organisations
- Gestion des workflows
- Modération
- Support utilisateurs
- Etc.

Exemple de Front Office Grand Public

- Story Mapping
- Capacité de créer une restitution, indicateurs et tableaux de bords > Espace de travail fonctions de base

8 Environnement

8.1 Environnements et rôles

Plusieurs environnements sont déployés pour le MVP. Un environnement correspond à une plateforme dédiée à des usages précis, que ce soit pour les développeurs ou les testeurs.

Le tableau ci-dessous récapitule les différents environnements mis en place par Citeos dans le cadre du MVP et à terme dans le cadre de la production.

Une phase intermédiaire (fin 2022) consiste à avoir :

- Une plateforme de développement, qui sera constante pour toute la durée du projet et hébergée sur les serveurs gérés par Citeos
- Une plateforme d'intégration qui soit indépendante des choix d'hébergement en production, afin de pouvoir lancer les développements et leur recettage fonctionnel rapidement
- Dans une méthode Agile, une plateforme d'intégration dite « MVP 2022 » permettra de récupérer les retours des utilisateurs finaux

Une phase secondaire (mi 2023) consiste à avoir :

- En complément, une plateforme de préproduction technique montée une fois l'environnement de production défini, dans un mode d'hébergement proche. Cette plateforme servira à compléter la recette fonctionnelle par une recette technique (tolérance de panne, volumétrie, scalabilité, redondance).
- Une plateforme de production au sein de l'hébergement choisi, qui accueillera la migration

Une phase tertiaire (post mi 2023) consiste à avoir :

- 3 plateformes : Développement, Préproduction, Production
- Des plateformes complémentaires créées au besoin

Plateformes pour le MVP :	Développement	Intégration	MVP 2022
Visée	Plateforme sur laquelle le développeur réalise des tests de validation technique sur les dernières versions de ses développements	Plateforme sur laquelle sont exécutés les tests fonctionnels et de non-régression	Plateforme sur laquelle un échantillon d'utilisateurs finaux pourra se connecter et réaliser des retours sur les fonctionnalités
Usagers	Développeur	Développeurs, Testeurs fonctionnels	Utilisateurs finaux
Configuration, Format	Infrastructure flexible pouvant être recréée au besoin, à mise à jour automatique	Version applicative stable sur de l'infrastructure qui peut être légère	Version applicative stable, mise à jour sur demande
Hébergement	Serveur OVH administré par Citeos	Serveur OVH administré par Citeos	Serveur OVH administré par Citeos
Mise à jour	Les mises à jour son continues sur la base du	Mise à jour via les merge requests effectuées dans	Mise à jour via les merge requests effectuées dans

	code poussé par le GIT (gestionnaire de sources et de configuration). Pour les images Docker, les mises à jour se basent sur le taguage des images docker pour suivre la dernière image générée par les outils de CI/CD.	le GIT. Le déploiement des nouvelles versions se font automatiquement, une fois spécifiées, pour permettre les tests fonctionnels.	le GIT. Le déploiement des nouvelles versions se font automatiquement, une fois spécifiées, pour permettre les tests fonctionnels

Plateformes à prévoir pour la production :	Préprod technique	Production
Visée	Plateforme de Recette technique (validation redondance ou autre)	Usages fonctionnels grand public (plateforme « visible »)
Usagers	Développeurs, Testeurs techniques	Grand public
Configuration, Format	Equivalent en termes d'infrastructure et de ressources matérielles à la plateforme de production	Infrastructure cible à déterminer
Hébergement	Emplacement dans l'hébergement cible de production	Hébergement cible de production
Mise à jour	Création et mise à jour à la demande	Infrastructure orientée immuable à mise à jour règlementée après validation en phase de préproduction

Remarques sur l'aspect AIPD :

- La CNIL recommande d'anonymiser les données à caractère personnel dans des serveurs hors production car plus facilement accessibles (accès développeurs, règles de sécurité moins fortes, etc.).
- Dans la mesure du possible, nous manipulerons dans la plateforme de préprod des données de tests fictives (données non personnelles) afin de simuler des fonctionnalités comme la gestion des consentements.

8.2 Déploiement et intégration des services

8.2.1 Définitions des notions utilisées

8.2.1.1 Les notions de CI/CD et DevOps

Le DevOps est un mouvement qui vise à concilier deux corps de métier : le développeur logiciel (dev) d'une part, l'administrateur de systèmes et d'architectures (Ops) d'autre part.

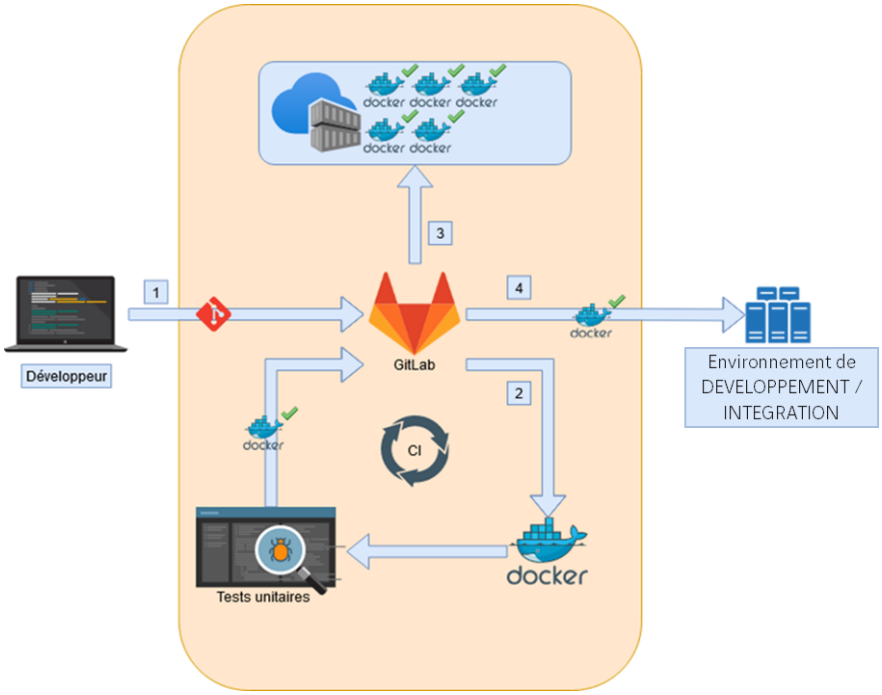
L'enjeu est de rendre en considération les contraintes de déploiement dès la phase de programmation. Avec pour objectif d'appliquer la logique des méthodes agiles à l'ensemble de l'activité informatique, le DevOps se concrétise par la mise en place de pipelines d'intégration et de livraison continues (CI/CD) courts.

Issues du monde DevOps, les pipelines d'intégration et de distribution continues (CI/CD) désignent une pratique qui consiste à améliorer la distribution de logiciels à l'aide de l'approche DevOps. Cela se matérialise notamment par une suite d'étapes à réaliser afin de distribuer une nouvelle version de logiciel. Cela sera mis en place via l'outil open source en ligne GitLab CI/CD :

- **Intégration continue :** compiler et tester le code en continu lors des développements (~ chaque jour, plusieurs fois par jour) afin de réduire l'introduction d'erreurs dans les services en développement.
- **Livraison continue :** packager les services pour qu'ils soient prêts à être déployés. Cela vient à la suite d'une l'intégration continue réussie.
- **Déploiement continu :** les déploiements demandent peu ou aucune intervention humaine

Application et illustration de ces principes ci-dessous :

Pipeline CI/CD	Notre approche LRTZC
1) Le développeur envoie ses modifications sur la forge interne.	Revue de code et validation par un/des développeurs expérimentés. Cela permet d'intégrer du code de la communauté open-source.
2) Une série de tests du code et des failles de sécurité connues est effectuée	Définition d'un pipeline de tests adapté : - Interopérabilité OS - Failles de sécurité et des images docker utilisées - Qualité de code (périmètre de règles adapté au projet) - Tests unitaires qui seront définis en phase projet
3) Si le commit est validé, l'image du code sera rajoutée aux listes d'images	Image versionnée déployée sur un répertoire d'images
4) Si le commit fait partie de la branche d'intégration, une routine de déploiement est exécutée.	Déploiement automatisé sur la plateforme d'intégration. Le déploiement en production se fera après passages des recettes et validation.



8.2.1.2 Conteneur et orchestrateur

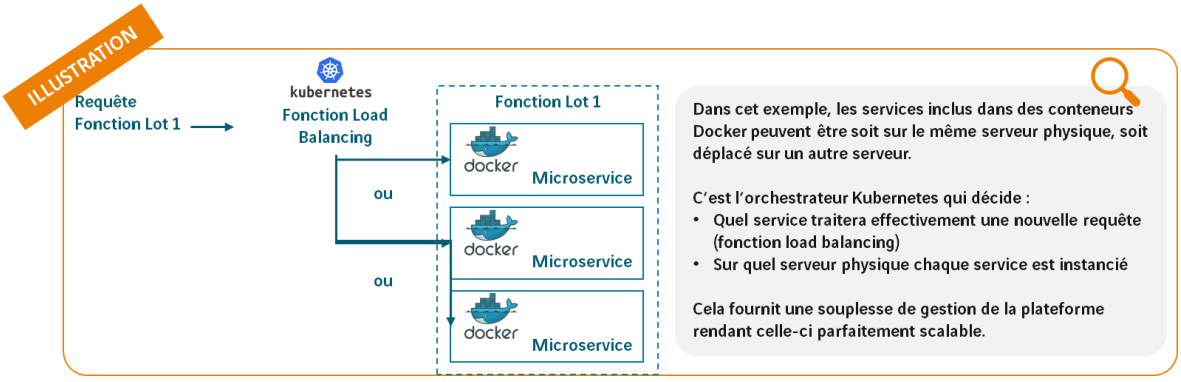
Deux notions clés du monde du DevOps sont utilisées au sein du projet : la conteneurisation et l'orchestration.

Conteneurisation des briques logicielles avec l'outil open source Docker :

- La conteneurisation consiste à isoler une brique logicielle dans un environnement indépendant tant niveau système d'exploitation que hardware.

Orchestration avec l'outil open source Kubernetes :

- L'orchestrateur permet la gestion des services des différentes briques logicielles conteneurisées. Il permet de gérer la disponibilité et scalabilité de ces services.



8.2.1.3 Usine logicielle

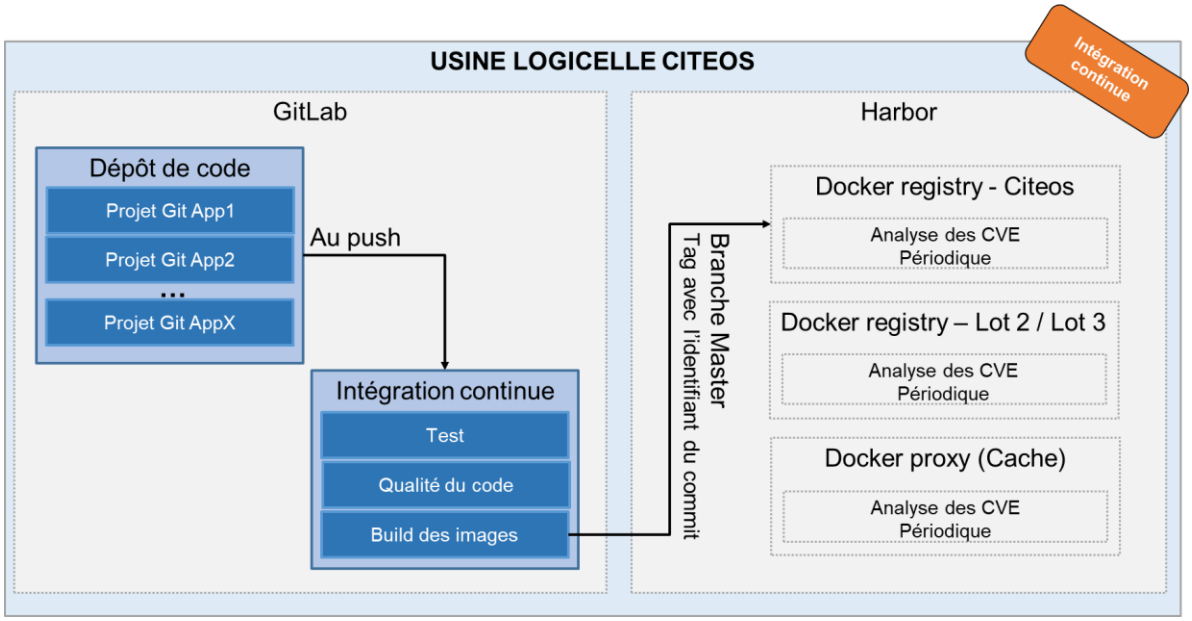
Une usine logicielle (ou software factory en anglais) couvre un ensemble d'outils et de procédures permettant, comme dans une chaîne de production, d'industrialiser le

développement informatique, comme le lancement régulier de la compilation, des tests unitaires, du déploiement. Elle permet de favoriser la collaboration des développeurs sur des projets et d'améliorer ainsi la qualité et la fluidité dans les phases de développement mais aussi d'exploitation.

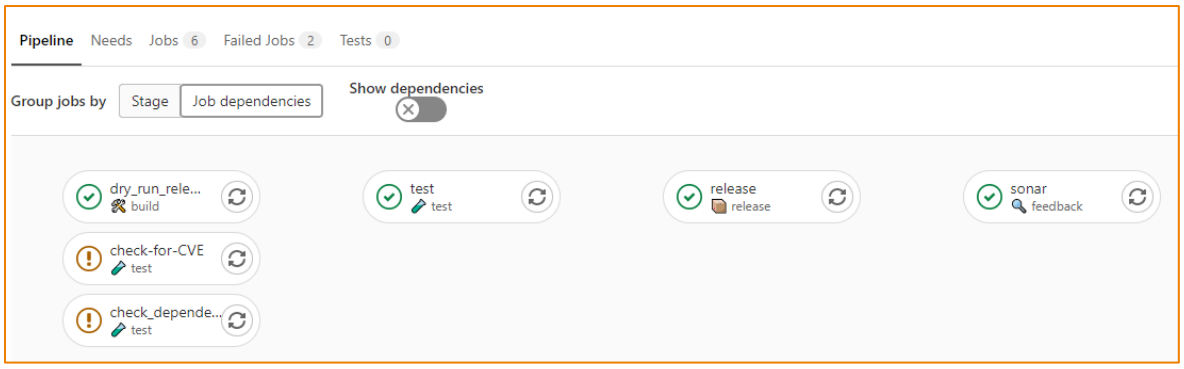
8.2.2 Intégration continue

Tous les développements spécifiques réalisés dans le cadre du projet de la Rochelle seront hébergés au sein de l'usine logicielle de Citeos. Ils profiteront de toutes les procédures de test et de suivi de qualité de code mis en place dans les bonnes pratiques de Citeos. Nous avons pour objectif d'implémenter des validations de bonne pratique GreenIT dans les procédures de suivi de qualité de code (notamment via le hackathon plugin Sonarqube « GreenIT »).

L'intégration continue se schématise de la manière suivante :



Les différentes étapes de validation sont visibles au sein de GitLab :

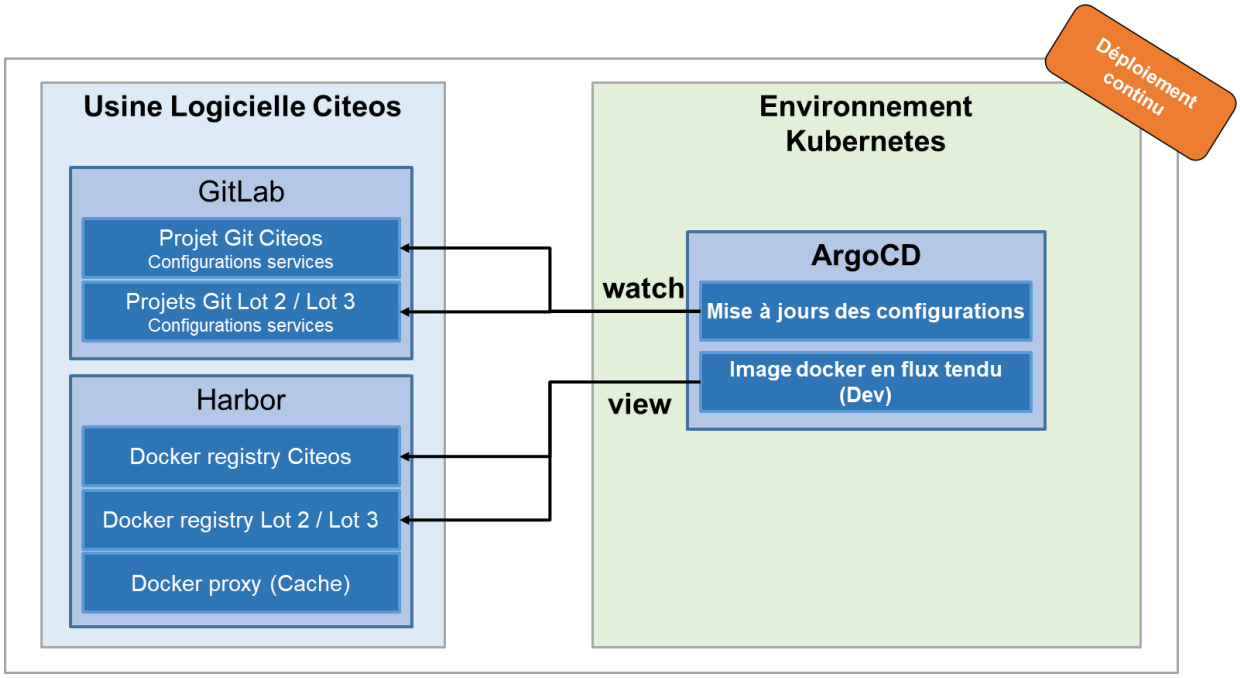


8.2.3 Déploiement continu

Nous exploitons les processus de déploiement continu GitOps (open source), en utilisant l'outil ArgoCD (open source), qui se charge des mises à jour et d'exploiter les capacités de l'orchestrateur Kubernetes :

- Chaque lot met à jour la configuration des services dans le GitLab.
- ArgoCD surveille la branche du GitLab et applique toute nouvelle version à la configuration du Kubernetes.

Le déploiement continu est limité aux applications et aux développements spécifiques, là où pour les composants open source sélectionnés (ex : PostgreSQL, NiFi, …), nous utilisons les déploiements existants pour Kubernetes disponibles en open source (lorsqu'ils existent).

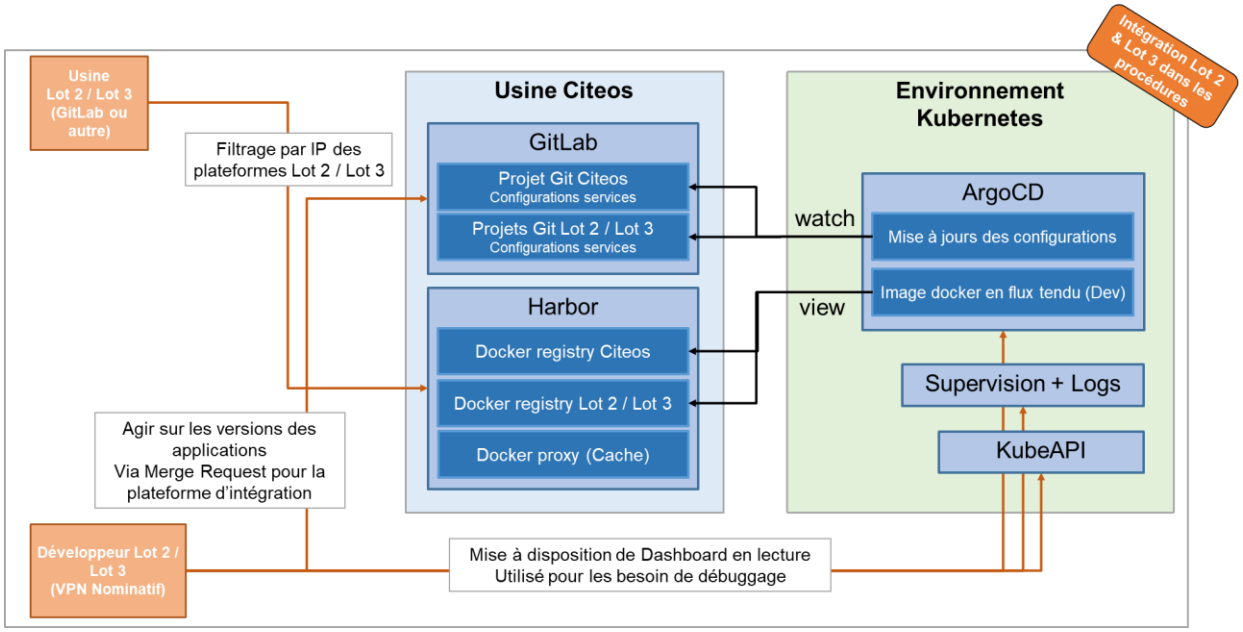


8.2.4 Intégration des lots 2 et 3

L'usine Citeos contient la description de ce qui doit être mis en œuvre dans Kubernetes par ArgoCD et plus précisément :

- Les fichiers de configuration des services
- Le registre des images docker

L'outil ArgoCD qui se trouve dans l'environnement Kubernetes, surveille les mises à jour des fichiers de configuration de l'usine Citeos et va déployer en continu les images dockers pour construire l'environnement cible.



8.2.5 Forge NAOS

8.2.5.1 Présentation des outils

- Bacula > outil de sauvegarde open source
- GitLab Cloud Native Helm Chart > Déploiement en mode Cloud Native

8.2.5.2 Présentation du processus et des bonnes pratiques de publication

- Distinction des contributions > @email pour identifier les auteurs
- Entête des fichiers source > chaque code source doit avoir un auteur, identifiant, licence SPDX et une copie de la licence dans le dépôt local
- Architecture fichiers > fichiers README, CONTRIBUTING et LICENCE dans le premier niveau de l'arborescence
- Documentation : publié sous Licence Ouverte 2.0, manuel utilisateur et guide du contributeur (sous Markdown conseillé)

8.2.5.3 Cycle de vie d'un projet

- Planification
- Création
- Vérification
- Conditionnement (GitLab registries)
- Vérification des failles de sécurité (GitLab DAST et SAST)
- Livraison (GitLab CI/CD) > Configuration (CI/CD)
- Protection (GitLab Mirroring)
- Monitoring (GitLab Monitoring)

8.2.5.4 Intégration dans la forge NAOS

Trois modes d'intégration sont possibles :

- Intégration du projet GitLab existant > Ouverture d'un compte sur NAOS et on s'appuie sur la forge
- Déploiement d'un GitLab spécifique > Monter une propre instance GitLab
- Synchronisation d'un GitLab extérieur >Script qui se synchronise avec l'exterieur

Remarques :

Nous allons publier tous les mois le code validé LRTZC vers NAOS où la communauté pourra travailler dessus.

Attention, il conviendra d'anticiper les problématiques de gouvernance. Dans l'idéal, il serait préférable de sélectionner une plateforme pour modifier le code et une autre juste en read-only.

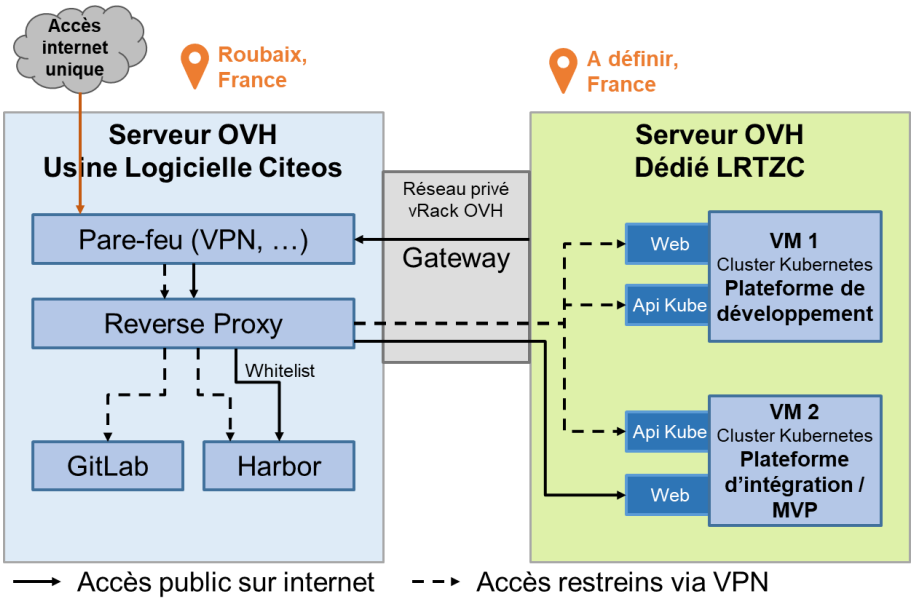
Dans un premier temps, nous mettrons à disposition le code source sur la forge NAOS qui est une exigence du projet en read only. Il conviendra dans un second temps de gérer l'animation et la gouvernance.

9 Représentation technique - Infrastructure

9.1 Infrastructure MVP

9.1.1 Hébergement

L'infrastructure pour les environnements de développement et d'intégration sont projetés dans sur un serveur unique chez OVH. La gestion des accès et de la sécurité seront administrés par les équipes de Citeos. Des VPN nominatifs seront transmis aux intervenant des différents lots. Un registre des accès sera tenu à jour tout du long du projet.



9.1.2 Dimensionnement des machines / services

Un serveur Advence-1 de chez OVH est pressenti aux vues des besoins.

- Disque de données sous raid 1 pour garantir la pérennité de la donnée
- Capacité de suivre la charge pour trois plateformes (développement, intégration et MVP)
- Usage pour un nombre restreint d'utilisateur
- Une capacité de stockage des jeux de donnée cohérent par plateforme
- Hébergement dans un datacenter en France
- Une bande passante publique de 1 Gbit/s

Advence-1

- 6c/12t (3.5/4.7GHz)
- 64Go
- 2x4To HDD
- A Roubaix sous 72h

⇨ A valider en phase de conception détaillée

9.1.3 Niveaux de service / Disponibilité

Les niveaux de service proposés pour le MVP ont les caractéristiques suivantes :

- Disponibilité assurée en heures et jours ouvrées
- Rétablissement sous 24h en heurs et jours ouvrés
- Fréquence des sauvegardes : 1/jour
- Pas de redondance serveurs

9.2 Infrastructure production

9.2.1 Hébergement

[A définir]

9.2.2 Dimensionnement des machines / services

[A définir]

9.2.3 Niveaux de service / Disponibilité

Le niveau de service de La Rochelle Université présente les caractéristiques suivantes :

Disponibilité du support technique :

- Plage de disponibilité : Lundi – vendredi ; 8h – 18h
- Délai de prise en compte d'un incident : 4h

Niveau de performance attendu

- Disponibilité de l'application : 97% dans la plage de disponibilité
- Temps de réponse bureau virtuel : < 3 sec
- Temps de réponse site Web : < 1 sec

Persistance des données

- Fréquence des sauvegardes : Min 1/jour
- Durée de conservation : 1 an
- Niveau de redondance des données : mécanisme type RAID pour la sécurisation du stockage.

9.3 Architecture réseau - Infrastructure

9.3.1 Architecture réseau (projet et environnement tiers)

[A définir]

9.3.2 Gestion des utilisateurs et autorisations

[A définir]

9.3.3 Matrice des flux (et sécurisation)

[A définir]

9.3.4 Intégrité

[A définir]

10 Exploitation et maintenance

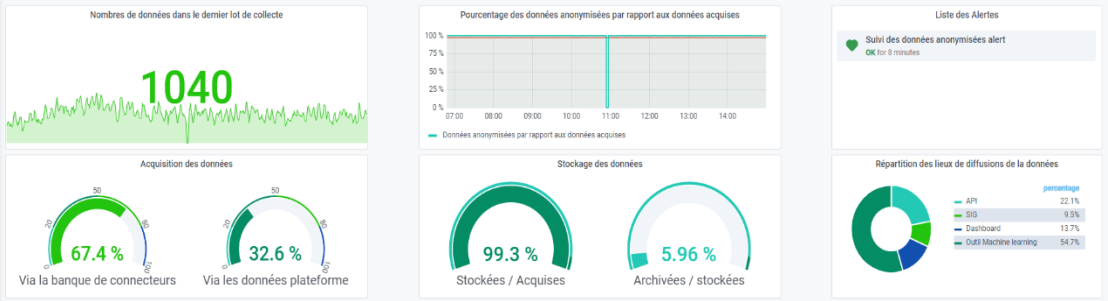
10.1 Supervision et Traçabilité

La supervision technique intègre :

- Une brique de supervision applicative et des infrastructures physiques (stockage, accès et ressources serveur) sur lesquelles est hébergé la plateforme
- Une brique de journalisation centralisée des évènements applicatifs pour une historisation des données et permettre l'investigation à la suite d'alertes remontées

Ces briques génèrent des données datées et classées par ordre chronologique.

Notre solution fondée sur une configuration de Prometheus couplée à un dashboarding Grafana permet de superviser un environnement orchestré complexe à travers un Dashboard d'indicateurs de supervision d'exploitation.



10.2 Maintenance

Les plages de maintenance et niveaux de services dégradés seront définis ultérieurement avant la mise en production.

Un outil de ticketing sera déployé pour le suivi des anomalies remontées afin de garantir le maintien en conditions opérationnelles, l'assistance et la maintenance de la plateforme (N2) à disposition de LRTZC, du lot 2, du lot 3 et de l'hébergeur.

Le niveau de criticité et délais de résolution associés seront ceux décrits dans le CCTP :

- Bloquant
- Majeur
- Mineur

10.3 Sauvegarde/Backup

[A définir]

10.4 Plans de Continuité d'Activité et de Reprise d'Activité

[A définir]

11 Exigences du projet LRTZC

11.1 Ecoconception

L'éco-conception correspond à une démarche de développement de biens ou de services prenant en compte l'ensemble des impacts environnementaux aux différentes étapes de conception et d'usage du produit, jusqu'à sa fin de vie.

Les principaux objectifs de cette exigence sont de :

- Intégrer l'éco-conception dans le processus de réalisation de la solution
- Mettre en place une démarche d'amélioration continue à travers notamment la réalisation d'ACV du produit
- Obtenir l'évaluation AFAQ éco-conception de l'AFNOR niveau « exemplaire »

Ci-dessous, la liste des principaux critères à prendre en compte dans la réalisation du service numérique :

- Architecture
  - Choix composants minimisant leurs impacts environnementaux
  - Adaptation des ressources selon l'usage
  - Exigence d'évolutivité technique (obsolescence)
- Contenus
  - Niveau compression adapté (image, vidéo, PDF, …)
  - Format fichier adapté au contexte (map sur mobile)
  - Stratégie de suppression des contenus
  - Stratégie d'accès aux données (périodicité, import dans la plateforme ou accès via API ...)
- Frontend
  - Poids maximum par écran à ne pas dépasser
  - Limitation du nombre de requêtes par écran
  - Charger que le nécessaire dans les librairies
  - Chargement des ressources à la demande
- Backend
  - Utilisation de cache serveur
  - Compression des contenus depuis le serveur
  - Stratégie de suppression des données
- Hébergement
  - Hébergement avec gestion durable équipements
  - Hébergement avec indicateurs d'impacts environnementaux
  - PUE (réduire ou limiter la consommation d'énergie nécessaire au bon fonctionnement et au refroidissement des serveurs) < à 1,5 (plus le PUE est proche de 1, mieux c'est) > Lot 1 et Univ.
  - WUE (réduire ou limiter la consommation d'eau nécessaire au refroidissement des serveurs) < 1 L/KWh

Dans l'optique de l'amélioration continu, des outils, registres et indicateurs seront mis en place afin de mesurer les impacts environnementaux de la plateforme de manière manuelle et automatisée quand cela est possible.

On intègrera dans les STI (Spécifications Techniques d'Interface) une analyse orientée éco-conception. Pour rappel, les objectifs d'une STI sont de décrire le périmètre fonctionnel et technique d'un connecteur de données (principalement : objectifs, flux de données, fréquence et volumétrie, interfaçage de connexion et modèle de données).

11.2 Interopérabilité

Cette exigence est directement liée à la vocation de la plateforme qui est de pouvoir importer ou exporter n'importe quel type de données. La solution proposée retiendra des standards techniques existants pour réduire au maximum les coûts de redéveloppement et de maintenance associés à des connecteurs, tant pour les flux entrants que sortants.

Ainsi, nous respecterons à minima les dispositions de standard d'interopérabilité proposées par le référentiel général d'interopérabilité de l'Etat (RGI V2.0). Ci-dessous, les principaux standards sélectionnés :

- Entrant
  - Import : fichiers csv, json et xml
  - Connecteurs entrants : SFTP et Rest HTTPS
- Sortant
  - Export : fichiers .csv .json
  - Connecteurs sortants : API Rest HTTPS
- Inter-lot : API Rest HTTPS
- Authentification : Open ID Connect (à préciser)
- Administration : VPN
- Exploitation : DNS
- Messagerie : SMTP/S
- Géomatique :
  - Collecte : WFS, WMTS et GeoJSON
  - Restitution : SLD
  - Reprojections supportées : Lambert 93 ou 3946 (zone de La Rochelle) et GPS
- Description d'API : YAML
- Image : PNG, SVG, GeoTIFF (à valider)
- Données (à affiner lors de la conception)
  - Catalogues de données et métadonnées
  - Modèles de données standardisés
  - Ontologies
  - Modèles de métadonnées standardisés
  - Export des données : standards dcat, inspire …

11.3 Scalabilité

Les variables à prendre en compte pour la croissance de la plateforme sont :

- Le nombre d'utilisateurs et le nombre de requêtes adressées par les utilisateurs ;
- Les volumes de données dans leur ensemble comme pour chaque jeu de données, susceptibles de s'enrichir dans le temps ;
- Le nombre de cas d'usage.

Pour s'adapter à ces besoins, notre architecture repose sur une structure micro-servicielle (c'est-à-dire comme un ensemble de services faiblement couplés), ce qui a l'avantage de :

- Pouvoir faire évoluer la plateforme fonctionnellement et techniquement (charge), et sans interruption de service
- Faciliter les déploiements grâce aux technologies de CI/CD
- Faciliter la maintenance

11.4 Privacy

Le traitement des données personnelles devra respecter le RGPD et entre autres permettre l'automatisation de l'exercice des droits des utilisateurs depuis l'IHM de la plateforme :

- Popup consentement (Intégrant information relative aux traitements algorithmiques)
- Page de profil avec la possibilité de modifier ou de supprimer ses informations personnelles voire d'en limiter l'utilisation
- Accès à la demande de restitution des données personnelles, etc.

Le Privacy by design intervient dès la phase de conception et prend en compte l'intégration des problématiques de protection de la vie privée grâce notamment à une analyse d'impact du traitement de la donnée (AIPD ; réalisée par la CDA de La Rochelle) sur la vie privée des utilisateurs.

L'application du privacy se traduit opérationnellement par le lot 1 par :

- Le remplissage d'un formulaire de traitement des données dont le modèle est fourni par la CdA de La Rochelle
- L'application potentielle de mécanismes d'anonymisation et/ou de pseudonymisation lors de l'acquisition et l'ingestion des données
- La mise en place de mécanismes d'exercice du droit RGPD dans le back-end

La gestion des accès par une authentification et la gestion de droits est réalisée.

La traçabilité des données passera par des systèmes de journalisation afin de permettre de détecter des incidents ou accès non autorisés à des données personnelles.

11.5 Réplicabilité et évolutivité

Un des objectifs clés est d'être en mesure de la répliquer sur d'autres territoires, ou sur d'autres problématiques puis de faciliter la réplication à différentes échelles, en France comme à l'international et dans différents contextes. Cela passera par :

- Choix techniques et fonctionnels pensés à la fois pour répondre aux besoins des cas d'usage mais également dans une logique de réplicabilité à grande échelle permettant de capitaliser dessus

- L'ensemble du code des composants de la plateforme soit disponible en Open Source pour pouvoir être librement adapté
- Implémentation des cas d'usage indépendant du socle technique, qui doit être par principe être « agnostique », c'est-à-dire pouvoir réutiliser dans des contextes différents de LRTZC
- Ensemble des opérations d'administration soient standardisées et puissent être réalisées via des interfaces dédiées

Le portage opérationnel de l'exigence Opensource (Apache 2.0) se traduit par les bonnes pratiques suivantes :

- **Distinction des contributions** > @email pour identifier les auteurs
- **Entête des fichiers source** > chaque code source doit avoir un auteur, identifiant, licence SPDX et une copie de la licence dans le dépôt local. L'ajout du stamp "Licence Open Source de type Apache Public License 2.0" dans le code source en entête sera automatisé
- **Architecture fichiers** > fichiers README, CONTRIBUTING et LICENCE dans le premier niveau de l'arborescence ainsi que les commentaires dans le code (permettant entre autres de générer de la documentation, par exemple d'API avec Swagger)
- **Documentation** : publié sous Licence Ouverte 2.0, manuel utilisateur et guide du contributeur (sous Markdown conseillé) et mise en place d'un Wiki
- **Veille open source** : tenir un registre des composants techniques avec le critère Open source et un Dashboard pour communiquer

La publication dans une forge de chez NAOS est aussi une exigence.

11.6 Sécurité des SI

Les exigences sécurité de la plateforme devront s'aligner avec le PAS en mode production avec pour principales exigences :

- Le respect des obligations légales ;
- Le respect des règles de sécurité de la CDA et de la Ville de La Rochelle ;
- L'auditabilité : la CDA doit pouvoir à tout moment réaliser des audits sûr, tout ou partie, des éléments composant le service mis en externalisation. Ce droit de regard doit pouvoir être réalisé par la collectivité elle-même ou par un tiers désigné par la collectivité (référencée par l'ANSSI);
- La réversibilité : On réunit sous le vocable « réversibilité » la réversibilité vers la collectivité et la transférabilité à un tiers désigné par le pouvoir adjudicateur. La réversibilité est due quelle que soit la cause ayant entrainé la fin du marché. Elle doit respecter les conditions qui suivent :
- La continuité de service ;
- Le plan de réversibilité ;
- La confidentialité des documents, contrats, etc.
- La gestion des comptes et des mots de passe (annuaire, force)
- Une architecture web trois tiers (https, chiffrement des données sensibles)

- La sécurité des serveurs (pare-feu, antivirus, gestion de l'obsolescence des composants) et gestion des traces
- Le respect les bonnes pratiques du TOP 10 OWASP (injection, cross-site scripting, etc.)

Le maintien de la propriété de la CDA sur :

- Tous les logiciels ou matériels dont la collectivité a payé les licences ;
- Tous les développements effectués pour le compte de la CDA ;
- Toutes les procédures manuelles ou électroniques liées à l'exécution du service ;
- Toutes les documentations produites par le prestataire dans le cadre de l'exécution du service.
- La protection contre les actions de contrefaçon ;
- La formation à la sécurité du personnel du prestataire ;
- La confidentialité du contrat et de ses annexes et de tous les documents, informations et données, quel qu'en soit le support, que les parties échangent à l'occasion de l'exécution du contrat ;
- L'obligation générale de conseil, d'information et de recommandation du prestataire en termes de qualité de service et mise à l'état de l'art.

11.7 Approche UI/UX

L'approche UX (User EXperience) ou expérience utilisateur, désigne la qualité de l'expérience vécue par l'utilisateur dans toute situation d'interaction. L'UX qualifie l'expérience globale ressentie par l'utilisateur lors de l'utilisation d'une interface, d'un appareil digital ou plus largement en interaction avec tout dispositif ou service.

Dans le cadre du projet LRTZC, cette approche prend son sens dans le fait que nous souhaitons nous mettre en capacité de mobiliser le maximum de citoyens et de partenaires qui puissent s'approprier le fonctionnement de la plateforme et s'engager dans la démarche en proposant leurs données et en contribuant aux cas d'usage qui le nécessitent.

De manière complémentaire, nous attendons également que la question de la protection des données personnelles soit intégrée de manière fluide et « sans couture » à l'expérience utilisateur. On peut à ce titre évoquer les bonnes pratiques mentionnées dans les cahiers IP du LINC.

Le détail de cette démarche est documenté dans le fichier : DémarcheUX-LTRZC_V1.0

La solution devra respecter le RGAA (Niveau WCAG AA) afin d'assurer l'accessibilité numérique pour rendre les services en ligne accessibles aux personnes en situation de handicap.

12 ANNEXE : Périmètre technico-fonctionnel du MVP

12.1 Authentification et gestion du profil

Fonctionnalités	Cadrage technico-fonctionnel
Accès	Accès Internet public / Pas d'usage grand public
Authentification	L'utilisateur devra s'authentifier pour accéder à la plateforme de données à travers un couple @email et mot de passe. La première connexion demandera une initialisation du mot de passe Type de déconnexion : clic sur un lien « déconnexion »
Rôles	Deux rôles pour les utilisateurs de la plateforme : - Administrateur fonctionnel - Contributeur individuel
Création de compte	- Création à l'initialisation de la plateforme du compte administrateur fonctionnel - Création d'organisations par l'administrateur fonctionnel (demande de création formalisée par mail par l'organisation) - Création des comptes utilisateur par invitation mail envoyée par l'administrateur fonctionnel avec un rôle unique par défaut « Contributeur individuel » L'utilisateur peut être lié ou pas à une organisation (pas de réaffectation ultérieure)
Gestion de compte	Informations personnelles collectées : nom, prénom et @mail (gestion des droits RGPD à définir avec Christian Plantin) 50 comptes utilisateurs maximum
Consentements	Pas de données personnelles => pas de consentements gérés par la plateforme
Traçabilité	Un utilisateur a accès dans son espace personnel à un moment M aux informations suivantes : - Liste des contrats de partage de données (ou conventionnement) de l'utilisateur vers des réutilisateurs qui sont en cours - Idem à l'inverse - Liste des jeux de données partagées à d'autres utilisateurs – en cours - Liste des jeux de données consommées depuis d'autres utilisateurs – en cours
RGPD	Impact de la suppression d'un compte utilisateur sur les données partagées et les données de l'utilisateur (nom / prénom / etc) => à définir avec une analyse RGPD (Christian Plantin) : Suppression de compte = suppression ou anonymisation des données Suppression des logs sauf si obligation de conservation

12.2 Consultation du catalogue de données

Fonctionnalités	Cadrage technico-fonctionnel

Recherche	La recherche dans le catalogue de données s'effectue sur les métadonnées (pas de recherche sur le contenu des jeux de données)
Modèle de restitution	3 modèles de Restitution : tableau, graphique et cartographique Les jeux de données du catalogue de données ont dans leur métadonnée l'indication des modes de restitution possibles pour le jeu de données.
Export	Export du jeu de données sous format csv / json Profondeur d'historique / volumétrie des données exportables à définir en conception détaillée et selon des critères de performance et d'éco-conception
Edition d'objets	Modification possible des métadonnées d'un jeu de données dont il est producteur par un utilisateur selon des modalités à définir en conception détaillée (nombre, format…)

12.3Collecte et création de jeux de données

Fonctionnalités	Cadrage technico-fonctionnel
Collecte de données	2 Protocoles principaux de connexion aux sources de données : 1. SFTP : lien, login, mot de passe, répertoire et nom de fichier 2. API Rest HTTPS : clé d'API, login, mot de passe et point de terminaison (endpoint) Format de données : csv, json
Sources de données	2 sources de données : 1. Sources de données externes à la plateforme 2. Contributions depuis le front de la plateforme : • Saisie d'indicateurs dans un formulaire structuré et dont la structure est statique (pas de modification / édition possible des champs et des formats attendus) - type Crowdsourcing • Définition : par l'administrateur fonctionnel • Création : par les équipes de développement • Crowdsourcing d'un jeu de données au format csv => analyse en cours des impacts sur la qualité de la donnée, les finalités d'usage, etc. sans porter atteinte à la gouvernance de la donnée
Lot de collecte	Conservation des lots de collecte : pas de limite de temps. Pas d'implémentation automatisée des fonctionnalités de destruction et d'archivage.
Types de licences	2 Types de licence intégrées : => En cours d'analyse avec Virginie Steiner • Open Data : licence libre et ODbL (orientée bases de données) pressenties • Copyright : 1 solution (ex : consultation) La restitution issue de plusieurs jeux de données va générer une nouvelle licence et un nouveau contrat (on se basera sur le tableau de croisement fourni par le cabinet juridique)

12.4Réutilisation et traitement des données

Fonctionnalités	Cadrage technico-fonctionnel
Smart Contract / Conventionnement	A la création d'un nouveau jeu de données dans la plateforme, nous saisissons dans la base de données de référence des Smart Contract, une matrice de partage entre le producteur de la donnée et un groupe d'utilisateurs. Cette saisie permet de faire le lien entre producteur et consommateur de données, et de tester dès le MVP les fonctionnalités de conventionnement / licence / droit d'accès à des jeux de données. L'évolution de la matrice en base de données pourra se faire manuellement sur demande pendant le MVP. Ce smart contract est statique : pas de gestion de workflow d'approbation et pas de gestion du cycle de vie du smart contract (ex : édition / suppression).
Algorithme	Les algorithmes de la plateforme sont ceux produits par le lot 1 et le lot 3 dans le cadre des cas d'usage. Une bibliothèque d'algorithmes est générée. La mise à disposition d'algorithmes « sur-étagère » auprès des utilisateurs sur le front pour qu'ils puissent les associer directement à des jeux de données et à des modèles de restitution n'est pas rendue possible au stade du MVP.

12.5Environnement et infrastructure

La plateforme dite MVP sera hébergée sous OVH et sera un environnement applicatif stable.

La plateforme dite d'intégration sera hébergée sous OVH, administrée par Citeos, et service de plateforme de recette.

Plus de détail est disponible au chapitre 8.1Environnements et rôles.

Le MVP s'appuie sur les ressources servicielles suivantes :

- Les fonds de carte SIG seront chargés depuis une API dédiée de l'IGN.
- Les analyses et statistiques d'utilisation de la partie Front seront réalisées depuis le serveur Matomo de la CdA afin de mutualiser les ressources.

Le périmètre de migration des données de la version MVP vers l'environnement de production (Université de La Rochelle a priori) comportera a minima :

- Tables de références
- Jeux de données
- Comptes utilisateurs

Le Service Level Agreement, ou SLA, de l'environnement MVP respectera les conditions suivantes :

- Heures et jours ouvrés (8h – 18h)
- GTR de 24h sur heures et jours ouvrés
- Sauvegarde des données : 1 x / jour