REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'étude
pour l'obtention du diplôme d'ingénieur d'état en électronique

Système d'information géographique sur plate-forme Big Data dédié aux objets connectés

BEN CHABANE Fariza
BOUCHELAGHEM Marya Ikram

Sous la direction de M. SADOUN Rabah Prof. ENP
Présenté et soutenue publiquement le 30/06/2019

Composition du jury :

Président	**M. Mourad HADADI**	**Prof. ENP**
Promoteur	**M. Rabah SADOUN**	**Prof. ENP**
Promoteur	**Mme. Nour El Houda BENALIA**	**Dr. ENP**
Examinateur	**M. Cherif LARBES**	**Prof. ENP**

ENP 2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Electronique

Mémoire de projet de fin d'étude
pour l'obtention du diplôme d'ingénieur d'état en électronique

Système d'information géographique sur plate-forme Big Data dédié aux objets connectés

BEN CHABANE Fariza
BOUCHELAGHEM Marya Ikram

Sous la direction de M. SADOUN Rabah Prof. ENP
Présenté et soutenue publiquement le 30/06/2019

Composition du jury :

Président	**M. Mourad HADADI**	**Prof. ENP**
Promoteur	**M. Rabah SADOUN**	**Prof. ENP**
Promoteur	**Mme. Nour El Houda BENALIA**	**Dr. ENP**
Examinateur	**M. Cherif LARBES**	**Prof. ENP**

ENP 2019

Dédicaces

Nous dédions cet humble travail à nos chers parents,
Nos frères et soeurs,
ainsi qu'à toutes les personnes qui nous ont soutenues durant ce périple.

Remerciements

Nous remercions dieu tout puissant pour nous avoir donné le courage, la santé et la patience durant ce chapitre de notre vie.

On dédie un remerciement spécial à notre encadreur, **Mr Rabah SADOUN** qui nous a guidées et a répondu présent à tous nos appels afin de mener à bien ce projet. Son dévouement nous a motivées et encouragées à donner le meilleur de nous mêmes et enfin à comprendre que rien n'est impossible (l'unique limite étant le ciel)

Nous remercions **Mme Nour El Houda BENALIA** et **Mme Ouidad KARTOUS** pour le temps et les efforts qu'elles nous ont dédiés ainsi que Leur assistance à la correction du document .

On souhaite aussi remercier **Mr Amine KENDI** pour son aide, son support et l'expérience qu'il a partagé avec nous.

De façon générale, nous remercions L'école Nationale Polytechnique d'Alger de nous avoir hébergées et fourni le matériel nécessaire pour la réalisation de ce projet et particulièrement, le chef du département d'électronique, **Mr Mourad ADNANE** pour nous avoir facilité l'accès aux locaux de l'école à des heures tardives et durant les jours de congés.

Des remerciements chaleureux à nos parents pour leurs soutien, encouragement et patience durant toute cette période.

Pour finir, nous remercions notre famille, nos amis et chaque personne sans qui rien de tout cela n'aurait été possible.

ملخص

لقد أخذ نظام المعلومات الجغرافية نطاقا واسعا منذ رقمنة البيانات. لذا فقد نلجأ اليه في شتى المجالات, ذلك لجمع, معالجة و عرض البيانات المكانية أو الغير مكانية. معتبرة كأدوات قوية جدا, و التي, اذا وضعت في خدمة الأشياء المتصلة, تمكنها من صعود و تقوية نفوذها و تأتي بكل الطاقة لهذا المجال.

من خلال هده المذكرة, سنقوم بتوضيح كيفية تصميم و بناء نظام للمعلومات الجغرافية مخصص لأنترنت الأشياء قادر على التمكن من كميات ضخمة من البيانات في بيئة إنتاجية, وذلك الى على أساس الحلول المفتوحة المصدر. سنقوم بعرض هذا العمل فوق تطبيق على الإنترنت.

الكلمات المفتاحية : نظام المعلومات الجغرافية, انترنت الأشياء, البيانات الضخمة.

Abstract

Geographic information systems GIS have grown significantly since the digitization of data. They are thus used in different fields, for the collection, processing and visualization of spatial and non-spatial data. Highly performing tools, which, at the disposal of connected objects, can enhance the IoT processing and intelligent decision making. All of this will also provide this field with all the power and analysis it needs.

Through this document, we will present the design and implementation of a GIS dedicated to connected objects on a Big Data platform in a production environment, solely based on open-source solutions. This work will finally be illustrated by a web GIS client application.

Key words: Geographic information system, IoT, Big Data.

Résumé

Les systèmes d'information géographiques SIG ont pris une grande ampleur depuis la numérisation des données. Ainsi, on y a recours dans différents domaines pour la récolte, le traitement et la visualisation de données spatiales ou non spatiales. Des outils très performants, qui, mis à disposition des objets connectés peuvent faire gravir plusieurs marches au monde de l'IoT, et apporter à ce domaine toute la puissance et l'analyse dont il peut avoir besoin.

A travers ce document, nous allons expliciter la conception et réalisation d'un SIG dédié aux objets connectés sur plate-forme Big Data dans un environnement de production et basé uniquement sur des solutions open-source. Ce travail sera illustré au final par une application cliente Web GIS.

Mots Clés : Système d'information géographique, IoT, objets connectés, Big Data.

Table des matières

Table des figures

Liste des tableaux

Nomenclature

API : Application Programming Interface

DBMS : Data Base Managment System

ETL : Extract Transform Load

GIS : Geographic Information System

GPS : Global Positioning System

HTTP : Hypertext Transfer Protocol

IoT : Internet of Things

JSON : JavaScript Object Notation

M2M : Machine to machine

MPP : Massively Parallel Processing

MQTT : Message Queuing Telemetry Transport

NoSQL : Not Only Structured Query Langage

OGC : Open Geospatial Consortium

RFID : Radio-identification

SMTP : Simple Mail Transfer Protocol

SQL Structured Query Langage

URL : Uniform Resource Locator

WMS : Web Map Service

XML : Extensible Markup Language

Chapitre 1

1 Introduction Générale

1.1 Objectifs et Motivation

l'IoT, acronyme de Internet Of Things en anglais et qui signife Internet Des Objets, est un ensemble de technologies permettant à des capteurs disséminés sur le terrain de communiquer des données entre eux, ou avec des systèmes externes (ex. bases de données) selon le besoin d'utilisation. Il est considéré par tous les industriels et analystes comme la prochaine grande révolution technologique. On parle de 26.66 milliards d'objets connectés dans le monde en 2019, soit 75.44 milliards d'ici 2025 [2], d'où l'énorme potentiel d'usage.

Les GIS, acronyme de Geographic Information System en anglais, soit les systèmes d'information géographiques en français, quand à eux, suscitent un intérêt croissant chez les entreprises. En effet, il représente un outil de traitement de l'information ainsi qu'un formidable levier d'investigation qui permet de prévenir ainsi que de visualiser un certain nombre de situations dans la vie courante et dans les secteurs privés. Ils favorisent les prises de décision intelligentes qui aident à la bonne gestion et la meilleure utilisation de l'information. Couplés aux nouvelles technologies, ils représentent un puissant allié pour la maîtrise de l'information.

L'équation est vite posée, la combinaison de l'IoT et d'un GIS donne naissance à une solution idéale. Tout d'abord, un tel système va permettre de mettre la puissance offerte par les analyses spatiales dans un GIS à disposition des données engendrées par les objets connectés. En plus de cela, il va permettre de croiser les données récoltées d'objets connectés avec les données contextuelles qui se trouvent déjà dans un GIS. Enfin, le GIS va offrir un référentiel cartographique pour les capteurs IoTs, induisant alors une représentation de l'information avec de meilleures lisibilité, appréhension et exploitation de son parc IoT.

Nous voyons à travers un GIS orienté IoT un potentiel énorme, voire révolutionnaire. Un potentiel qui motive suffisamment pour en faire notre projet de graduation ; en effet, notre projet consiste à implémenter un système d'information géographique dédié aux objets connectés, capable de prendre en charge une quantité volumineuses de données.

Inspiré aussi du projet de fin d'étude "intégration d'un système d'information géographique à une solution Smart City". Présenté par le binôme BENLEFKI Yasmine et BOUCENNA Manal Maroua, elles ont implémenté une solution IoT - GIS dédiée aux villes intelligentes dans le cadre du projet Algiers Smart City. Pour cette réalisation, elles ont utilisé le projet open-source Sentilo, une plate-forme pour la gestion de capteurs et d'actionneurs. Cette dernière s'adresse à toute personne du monde de la Technologie de l'Information (IT) qui souhaite contribuer à l'expansion de l'IoT dans le contexte des villes intelligentes.

Ceci nous a motivées à imaginer une solution plus généraliste et plus performante qui posséderait plusieurs cas d'utilisations, ne se limitant pas aux villes intelligentes. Autrement dit, cela peut

s'étendre du domaine médical jusqu'à l'exploitation pétrolière. Cette motivation n'a fait que se raffermir par le fait que l'industrie des objets connectés soit en pleine effervescence, vu le nombre d'objets connectés qui ne fait que croître. Naturellement, cette croissance sans précédent s'accompagne d'une augmentation encore plus importante de production de données. Ces dernières auront de ce fait besoin d'un écosystème qui va permettre leur ingestion pour leur traitement ainsi que leur stockage. nous parlons alors du traitement Big Data.

Parmi les environnements open-source pour la gestion Big Data , on retrouve Apache Kafka. Bien qu'il ne soit pas destiné à l'origine aux dispositifs IoTs, Apache kafka s'est établi au fil des ans comme l'un des éléments de base des déploiements IoT à grande échelle. En effet, sa capacité à agir comme couche absorbante de données dans une infrastructure dédiée aux objets connectés est sans précédente. Beaucoup de grandes sociétés dans monde l'utilisent pour la gestion de leur données massives : Twitter utilise Kafka pour la gestion de données dans son cluster Storm, tandis que Netflix l'utilise pour la surveillance en temps réel et le traitement des événements.

Ainsi, un GIS dédié aux objets connectés à usage général, d'une grande performance et à une échelle de production, basé sur Big Data pour la gestion de données, représente la solution ultime.

1.2 Plan du document

Ce document contient six chapitres, et des annexes

— **Chapitre 2** (IoT-GIS sur plate-forme Big Data) est constitué de concepts clés pour l'implémentation d'un GIS dédié aux objets connectés à grande échelle. Il englobe des notions générales sur les GIS, l'IoT et le Big Data.

— **Chapitre 3** (Plate-forme d'IoT) commence par présenter les plate-formes IoT, suivi d'un exemple représentant la plate-forme Sentilo. Il enchaîne avec le développement de notre solution IoT composée d'Apache Kafka, Kafka connect ainsi que la base de données mongodb pour finir avec l'architecture générale de la solution IoT.

— **Chapitre 4** (Plate-forme GIS) commence par expliquer le fonctionnement des GIS, suivi des données traitées par les GIS ainsi que les différentes bases de données GIS. Puis, enchaîne par les services et les applications web cartographiques pour finir avec les choix de logiciel GIS open-source et l'architecture finale de notre GIS.

— **Chapitre 5** (Implémentation) présente l'architecture complète du système ainsi que le développement de l'environnement et de toute sa réalisation. Il est clôturé par une application web GIS, un géoportail, pour illustrer l'implementation du GIS d'IoT sur plate-forme Big Data.

— **Chapitre 6**(conclusion) résume notre travail, les problèmes rencontrés ainsi qu'une perspective future pour ce projet.

— **Annexe** contient les étapes d'installation de certains systèmes constituant l'environnement ainsi que des scripts, fichiers de configuration de données test.

Chapitre 2

2 IoT-GIS sur plate-forme Big Data

2.1 Introduction

Dans ce chapitre, nous allons introduire des notions de base, nécessaires à la conception de notre système d'information géographique dédié aux objets connectés, à savoir, ce qu'est une plate-forme IoT, une plate-forme GIS mais aussi des notions concernant le traitement massif de données qui est primordial pour une solution adaptée à un environnement de production.

2.2 Plate-forme IoT

2.2.1 Internet des objets

l'Internet des objets IoT [1] caractérise des objets physiques connectés ayant leur propre identité numérique et capables de communiquer les uns avec les autres. Ce réseau crée en quelque sorte une passerelle entre le monde physique et le monde virtuel.[3]

Techniquement, l'IoT désigne plutôt diverses solutions techniques (RFID [2], TCP/IP [3], technologies mobiles, etc.) qui permettent d'identifier des objets, de capter, stocker, traiter, et transférer des données dans les environnements physiques mais aussi entre des contextes physiques et des univers virtuels. L'enjeu majeur n'est pas tant d'inventer de nouvelles technologies que de perfectionner celles qui existent déjà, de les connecter, et de les intégrer.[4] En d'autres termes, ces objets sont caractérisés par un identifiant numérique normalisé (Addresse IP, Protocole SMTP [4], HTTP [5] ...) qui communiquent donc grâce à un réseau sans fil (WIFI [6], bluetooth [7], RFID).

Ces objets trouvent leur intérêt en logistique (pour ce qui concerne la traçabilité des biens, la gestion de stocks et l'acheminement), dans l'environnement (en tant que capteurs de température, humidité...), en domotique (pour tout ce qui concerne la gestion de l'électroménager, serrures et compteurs...), ou dans le domaine de la santé (nous citons les montres et bracelets connectés qui récoltent les données des signaux vitaux).

1. En anglais Internet Of Things
2. La radio-identification est une méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés « radio-étiquettes »
3. La suite TCP/IP est l'ensemble des protocoles utilisés pour le transfert des données sur Internet
4. Protocole de communication utilisé pour transférer le courrier électronique (courriel) vers les serveurs de messagerie électronique.
5. L'Hypertext Transfer Protocol est un protocole de communication client-serveur développé pour le World Wide Web.
6. est un ensemble de protocoles de communication sans fil régis par les normes du groupe IEEE 802.11 (ISO/CEI 8802-11).
7. Le bluetooth est une norme de communications permettant l'échange bidirectionnel de données à très courte distance en utilisant des ondes radio UHF sur une bande de fréquence de 2,4 GHz.

2.2.2 Composants d'un système IoT

Afin d'exploiter au mieux les données provenant de ces objets connectés, plusieurs éléments se sont immiscés dans l'environnement composant un système IoT.[5]

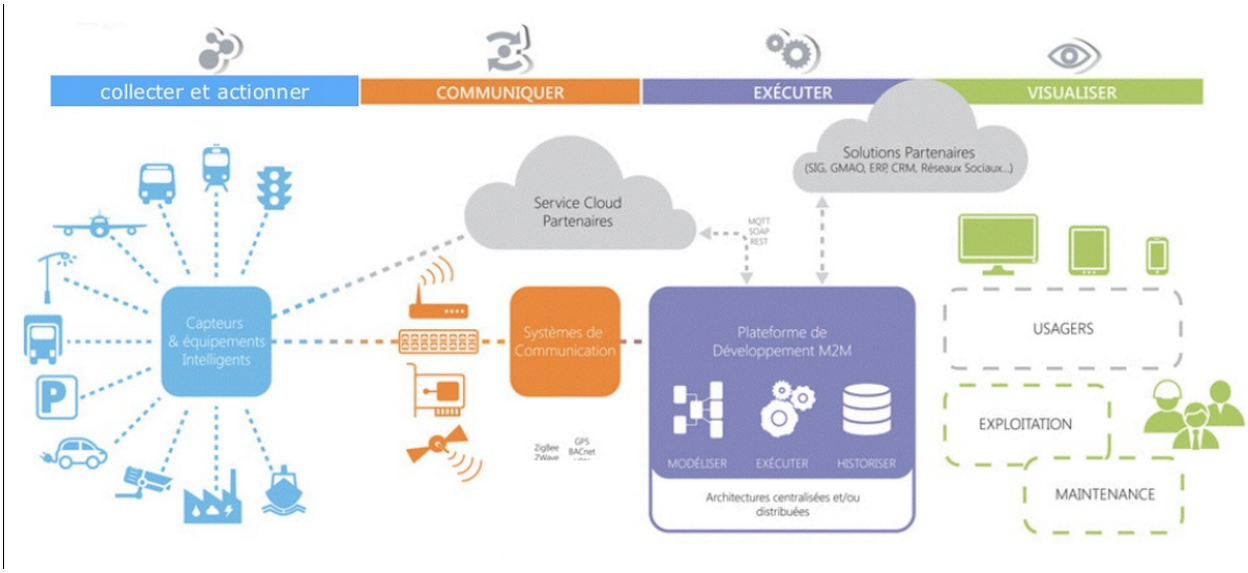


FIGURE 1 – Composants d'un système IoT

— **Les capteurs et radios émetteurs**
C'est les premiers éléments de la chaîne. Il sont composés d'un ou plusieurs modules afin de construire un objet connecté et exploitable.

— **Les plate-formes de gestion de capteurs**
Les plate-formes IoT ont rencontré une grande popularité dans le domaine sachant qu'elles facilitent la communication entre objets connectés, les échanges de données, la gestion des objets et le développement de fonctionnalités pour un objet, ce qui permet de sécuriser et de simplifier la gestion d'une infrastructure IoT. Elle représente l'élément central d'un système IoT.

— **La communication**
Même si la lettre "i" dans IoT désigne internet, il existe différentes sortes de réseaux pour la communication entre les capteurs et avec les plate-formes de gestion.

— **Les applications**
Elle permettent de traduire les données en informations utiles et aisément compréhensibles pour l'homme. Ceci peut se traduire à la création d'une interface utilisateur permettant la visualisation des données ou encore l'interaction avec les objets connectés

2.2.3 Plate-forme IOT

Une plate-forme IoT est un ensemble de services permettant de collecter, stocker, corréler, analyser et exploiter les données.[6]Une plate-forme est donc un ensemble d'APIs et pilotes de communication, de bases de données, de services de traitements et de calculs et bien souvent un service web pour générer des tableaux de bords de visualisation et d'exploitation.

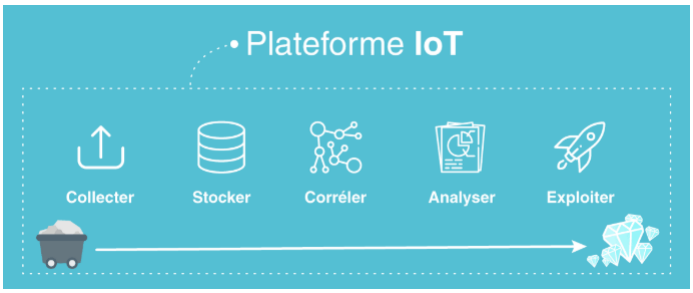


FIGURE 2 – Plate-forme IoT

2.3 Plate-forme Big Data

L'amas de données récolté par les systèmes IoTs nous oriente vers la notion de Big Data qui prend de plus en plus d'ampleur et qui est inévitable si nous souhaitons manipuler des objets connectés. Pour chaque architecture, nous retrouvons un traitement de données adapté.

2.3.1 Définition

Le Big Data ou mégadonnées désigne l'ensemble très volumineux de données numériques produites par l'utilisation des nouvelles technologies pour des fins personnelles ou professionnelles. Cela regroupe les données d'entreprise (courriers, documents, bases de données, historiques de processeurs métiers...) aussi bien que des données issues de capteurs, des contenus publiés sur le web (images, vidéos, sons, textes), des transactions de commerces électroniques, des échanges sur les réseaux sociaux, des données transmises par les objets connectés (étiquettes électroniques, compteurs intelligents, smartphones...), des données géolocalisées, etc...[7] Il s'agit d'un concept permettant de stocker un nombre indicible d'informations sur une base numérique.

L'expression « Big Data » date de 1997 selon l'Association for Computing Machinery. En 2001, l'analyste du cabinet Meta Group (devenu Gartner) Doug Laney décrivait les Big Data d'après le principe des "trois V" :[7]

— Le Volume de données de plus en plus massif ;

— La Variété de ces données qui peuvent être brutes, non structurées ou semi-structurées ;

— La Vélocité qui désigne le fait que ces données sont produites, récoltées et analysées en temps réel.

FIGURE 3 – Les 3V du Big Data

2.3.2 Base de données Big Data

Une base de données est un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données).[8] Il en existe plusieurs types, mais les plus populaires sont les base de données relationnelles.

Une base de données relationnelle (BDR) est une collection organisée de données structurées. Les BDRs mettent au premier plan les relations entre les données. Celles-ci sont organisées en tables à deux dimensions : on parle alors de lignes et de colonnes. Toutes les tables de la base de données peuvent être simplement rapprochées dès qu'elles partagent une même colonne (un attribut). Ces bases de données utilisent le langage SQL (Structured Query Language ou le langage de requête structuré), une interface de programmation applicative [8] (API) qui facilite l'interaction avec les bases de données.
Ces bases sont les plus utilisées et les mieux structurées pour le stockage de données, cependant ce n'est pas la taille ou le nombre de lignes d'une base qui fait d'elle une base de données Big Data ou pas.

Le Big Data est plutôt le fait de stocker des informations sur la base d'un modèle [Clef ; Valeur] mais sans structure de base de données forte. On peut donc insérer n'importe quel type de données dans une base de type Big Data sans avoir su au moment de la constitution de la base, quelles données allaient y être stockées. Le temps d'écriture est donc nettement abaissé, l'écriture se faisant sur le modèle d'une ligne simple sans tests de contrainte d'intégrité ni traitement d'index. On appelle globalement ça le NO SQL (puisque le Query n'est plus structré).[9]

Malgré leur non structuration, les bases de données No-SQL restent les plus puissantes. Cela est du au fait que plusieurs autres technologies entrent en jeu, comme la distribution du calcul et la parallélisation des blocs de calculs au lieu de la sérialisation (traitement vertical). De ce fait, ces bases sont rapides, flexibles et performantes.

2.4 Plate-forme GIS

Gérer un système de traitement IoT requiert énormément de données géolocalisées. Un GIS simplifie le traitement des données spatiales et améliore leur gestion.

2.4.1 Définitions

Tout d'abord, voyons ce qu'est un système d'information.

2.4.1.1 Un système d'information

Le système d'information (SI) est un ensemble de ressources (personnes, données, logiciels, processus, équipement informatiques ...) qui permettent la gestion de l'information comme la collecte,

8. Ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels.

le stockage, le traitement, l'archivage ainsi que le transport de l'information sous n'importe quelle forme (texte, image, son, vidéo ...).[10]

La conception et la gestion d'un système d'information combinent des tâches techniques liées au système informatique de traitement de l'information, composé d'ordinateurs et d'autres dispositifs matériels, de programmes et d'applications, de logiciels, de bases de données. Elle nécessite aussi l'analyser des besoins et des usages auxquels le système informatique doit répondre.

2.4.1.2 Un système d'information géographique GIS

Une information est géographique lorsqu'elle peut trouver une place sur une carte.

— **GIS** [9] **: un ensemble de données**
La définition française est due à l'économiste Michel Didier (1990), dans une étude réalisée à la demande du 16 conseil national de l'information géographique (CNIG) : Un système d'information géographique est un "ensemble de données repérées dans l'espace, structuré de façon à pouvoir en extraire commodément des synthèses utiles à la décision"

— **GIS : un système informatique**
La définition américaine émane du comité fédéral de coordination inter-agences pour la cartographie numérique (FICCDC, 1988) : Un système d'information géographique est un "système informatique de matériels, de logiciels, et de processus conçus pour permettre la collecte, la gestion, la manipulation, l'analyse, la modélisation et l'affichage de données à référence spatiale afin de résoudre des problèmes complexes d'aménagement et de gestion".

2.4.2 Composants d'un GIS

Un système GIS est basé sur 5 composants majeurs, a savoir :

FIGURE 4 – Les composants d'un GIS

— **Le logiciel :**
Ses fonctionnalités sont connues sous le nom des six A

9. En anglais Geographic Information System qui veut dire système d'information géographique

- *Acquisition* La numérisation des données géographiques
- *Archivage* La gestion des bases de données (DBMS [10])
- *Analyse* L'interaction et l'interrogation des données géographiques
- *Affichage* La mise en forme et la visualisation
- *Abstraction* Représentions du monde réel
- *Anticipation* La prospective

— **Les données :**
Les données sont géolocalisées et numérisées. leurs coordonnées sont définies par un système géodésique. Elles sont alors stockées dans des bases de données où elles sont organisées.

— **Le matériel informatique :**
Le plus généralement utilisé est l'ordinateur de bureau (sur lequel opère le logiciel GIS). Cela dit, ce type de traitement nécessite beaucoup de ressources en mémoire, ram et cpu. On utilise un serveur informatique centralisé pour la production.

— **Le savoir faire :**
Un concepteur de système GIS doit maîtriser des notions de systèmes de projection, informatique et géographie afin de réaliser un bon produit.

— **Les utilisateurs :**
Les utilisateurs de GIS sont des spécialistes techniques qui conçoivent et entretiennent le système, à ceux qui l'utilisent pour les aider à réaliser leur travail quotidien. Nous pouvons citer quelques domaines tel que :

- L'archéologie (localisation du mobilier/gestion de chantier)
- La biologie (étude des déplacements de populations animales)
- La géologie (prospection minière)
- Le tourisme (itinéraire et gestion de site)
- La gestion des forêts (étude des espaces, protection et écologie)
- La planification urbaine (aménagement du territoire/ étude des voiries)
- La sociologie (analyse spatiale sociologique)
- Le transport (étude des déplacement/analyse de réseaux)
- Les télécoms (implantation d'antennes.)
- . . . ETC

2.4.3 Catégories d'applications GIS

Il existe plusieurs catégories de GIS selon les besoins de l'utilisateur et des fonctionnalités offertes par le logiciel.[11]

— **Desktop GIS** est le logiciel GIS le plus utilisé et permet de d'afficher, éditer et analyser des données GIS. En d'autres termes, c'est une puissante machine qui intègre un système de cartographie. Exemple : ArcGis, QGIS.

10. En anglais Data Base Managment System

— **Système de Gestion de Base de Données (SGBD)** est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la base de données. Exemple : MySQL, Oracle Database.

— **Serveur Web Cartographique** est un logiciel qui permet de distribuer les informations ainsi que les cartes géographiques à travers internet. Il permet de faire des requêtes attributaire ou de localisation sur les données. Exemple : GeoServer, MapServer.

— **Client cartographique** est un logiciel qui permet d'afficher les données comme étant des images cartographiques, c'est-à-dire avec une légende, une échelle, la possibilité de changer d'échelle, de zoomer sur la carte et potentiellement d'interagir avec elle. Exemple : Openlayers, Leaflet.

— **GIS mobile** est un framework [11] intégré qui permet l'accès aux données spatiales et géographique au moyen d'appareils mobiles. à l'aide des technologies de communication GPS [12], Internet et sans fil, les GIS mobiles ont le grand potentiel de jouer un rôle important dans l'acquisition et la validation de données sur le terrain.

— **Les bibliothèques et extensions GIS** fournissent des fonctionnalités (d'analyse) supplémentaires qui ne font pas partie du logiciel de GIS de base, par exemple l'analyse du terrain, ou des fonctions de lecture d e formats de données spécifiques.

2.4.4 GIS pour l'IoT

La gestion d'objet connectées nécessite énormément de données géoréférencées. Un GIS a la capacité de traiter ces données afin de les exploiter pour la planification, la gestion et le développement de systèmes IoTs. Il peut par exemple, comme pour le projet "IoT-GIS for smart cities", servir pour le développement de solutions dans des villes intelligentes, à savoir : Parking des voitures, flux de voitures, enfreintes à la loi ... etc.

2.5 Architecture générale d'un IoT-GIS sur plate-forme Big Data

L'IoT d'une part et les GIS d'une autre part, voici l'architecture qu'on propose pour un systéme complet chargé de la gestion de données en provenances d'objets connectées et de leurs traitement par des GISs. L'architecture est composé de 4 couches majeurs :

1. **Passerelle IoT**
   Elle est responsable de la communication entre les capteurs chargé de récolter des données en temps réel avec la plate-forme IoT.
2. **Plate-forme IoT**
   Son but est de gérer les données IoT après leur ingestion pour une meilleure utilisation de ceux ci. Elle va permettre d'augmenter la performance et la fiabilité.
3. **Plate-forme Big Data**
   Ici, les données seront stocké, traité et analysé pour les mettre à dispostion de la couche suivante.

11. désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel
12. le Global Positioning System, un système de géolocalisation par satellite

4. **Plate-forme GIS**
   Elle exploite l'information géolocalisée pour détecter les besoins en temps réels et optimiser les ressources. Elle permet de visualiser des données, de partager des informations et d'appliquer des analyses et des prévisions sur des données selon leur position spatiale.

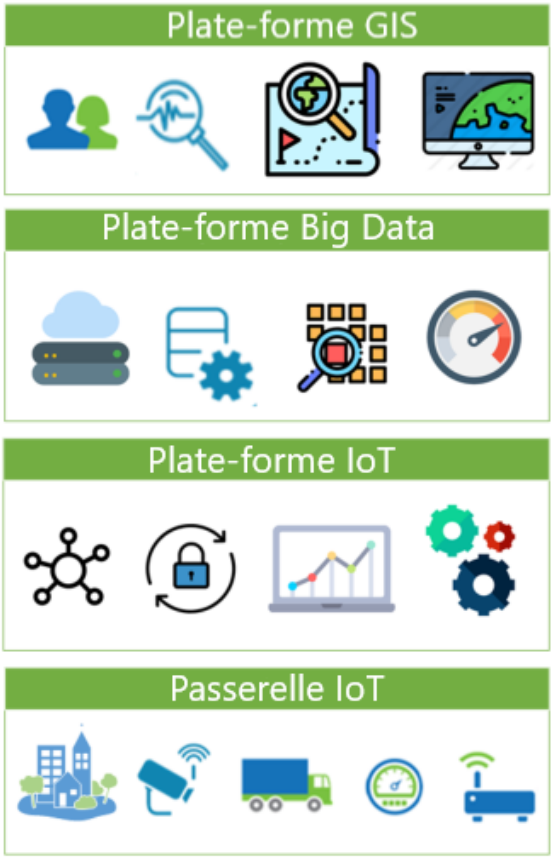


FIGURE 5 – Architecture générale de l'IoT-GIS sur plate-forme Big Data

2.6 Conclusion

Dans ce chapitre, nous avons introduit les concepts clés de départ, utiles à la compréhension du projet dans sa globalité. Par la suite nous allons expliquer les notions nécessaire à la conception d'une plateforme IoT ainsi que la solution que nous proposons.

Chapitre 3

3 Plate-forme IoT

3.1 Introduction

Souhaitant récolter et traiter des données en provenance d'objets connectés, nous allons, dans ce chapitre, parler des plate-formes IoTs, leur importance dans le traitement de ces données ainsi que la description de notre plate-forme, regroupant la solution Apache kafka et la base de stockage non relationnelle mongodb. Nous allons aussi parler du protocole le mieux adapté pour la communication avec des objets connectés qui n'est autre que le protocole MQTT.

3.2 Présentation des plate-formes IoTs

3.2.1 Architecture et fonctionnement d'une platforme IoT

Une platforme IoT rassemble différents types de protocoles et de données afin qu'ils soient gérés et analysés. Après avoir récolté les données grâce à une couche de connexion (interface de connexion), elle seront normalisées pour rejoindre la partie centrale de la plateforme IoT qui est la base de donnée (base de stockage Big Data). Ces données seront ensuite traitées selon le besoin ou directement affichées sur le tableau de bord de visualisation.

Le schéma suivant représente l'architecture global d'une platforme d'IoT

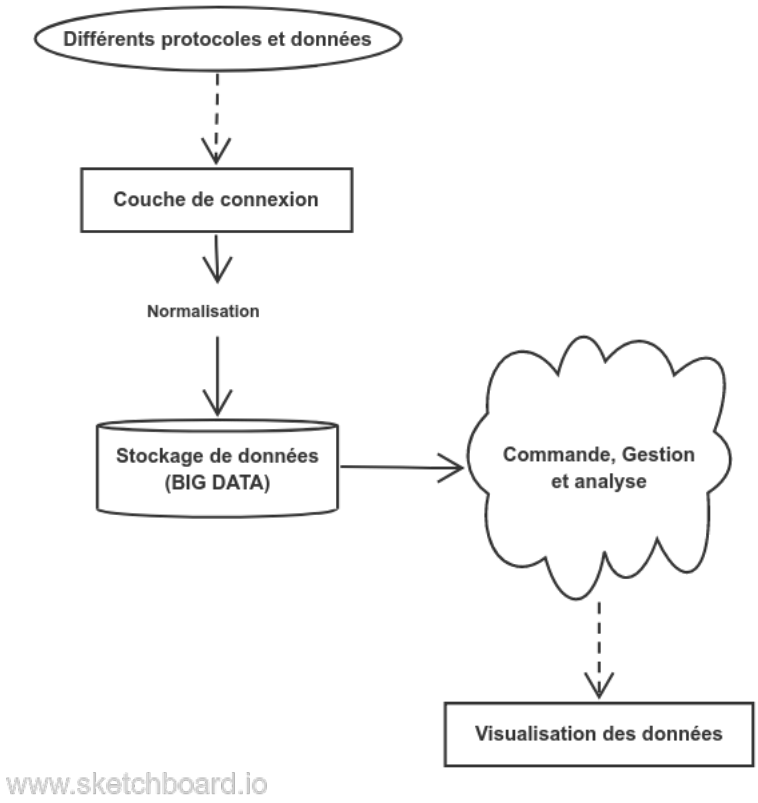


FIGURE 6 – Architecture et fonctionnement d'une plateforme d'IoT

3.2.2 Caractéristiques d'une plate-forme IoT

Il existe différentes plateformes avec différentes caractéristiques, les plus importantes d'entre elles sont celles-ci :[12]

1. Sécurité : Il est nécessaire que les données et les canaux de communication soit sécurisés.
2. Extensibilité : Il est nécessaire que la plate-forme IoT puisse se mettre à l'échelle de production de l'entreprise grandissante sans compromettre l'intégrité, la sécurité, la fiabilité et la fonctionnalité de la solution IoT.
3. Fiabilité : Il est nécessaire que la plate-forme IoT soit stable et qu'elle fournisse des solutions de dépannage des canaux et récupération des données en cas de panne.
4. Inter-opérabilité : La plate-forme IoT doit être accessible par chaque appareil peu importe son modèle, sa provenance ou son constructeur.
5. Modularité : Elle permet à la plateforme un déploiement rapide pour satisfaire les différentes requêtes des entreprises. C'est ce qui fera que la plateforme sera idéale pour un grand nombre de solution IoT et c'est ce qui lui permettra de résister à l'épreuve du temps.

3.2.3 Importance d'intégration d'une plate-forme IoT

De façon générale, les differents cas de figures de l'utilisation d'une pate-forme IoT se résument en 3 point :

— Intégration de systèmes IoTs dans des groupes (clusters) de serveurs analytiques.

— Communication bidirectionnelle pour le contrôle des appareils IoTs.

— Traitement de flux en temps réel provenant de capteurs IoTs en utilisant des algorithmes d'apprentissage automatiques (Machine Learning).

Nous voyons de ce fait l'enjeux de l'intégration d'une plateforme IoT dans notre architecture. Elle va permettre de collecter, gérer et analyser nos données de provenances de capteurs IoT. Un exemple de plateforme IoT est Sentilo, du projet de fin d'étude "Intégration d'un Système d'Information Géographique à une solution Smart City".

3.3 Exemple de Plate-forme IoT

Malgré le grand nombre de plate-forme IoT existantes, la majorité d'entre elles sont payantes ou privées, c'est à dire qu'elles permettent un accès gratuit ou utilisation gratuite à l'échange de fournir des informations.
Sentilo est parmi les rares plate-formes qui permettent aux développeurs de télécharger le code pour pouvoir le reproduire librement.

3.3.1 Présentation de Sentilo

Sentilo est une plate-forme pour la gestion des capteurs et des actionneurs, déployée dans la ville de Barcelone, en Espagne. La plateforme comprend de nombreuses fonctionnalités, telles que le Cloud Computing, une base de données NoSQL, une base de données mémoire et une interface RESTful [13] simple. Sentilo est conçu et développé en utilisant des composants open-source tels que les bases de données Redis, MySQL et MongoDB, hibernate, JSON ou Jquery [14].[13]

3.3.2 Cas d'utilisations de Sentilo

Sentilo s'adresse aux municipalités ou aux organisations qui ont besoin de traiter beaucoup d'informations provenant du terrain, générées par des matériels et logiciels hétérogènes (capteurs, etc...), et qui veulent d'une façon centralisée et homogène gérer et distribuer ces données dans leurs systèmes d'information.
Elle s'adresse également à toute personne du monde de la Technologie de l'Information (IT) qui souhaite contribuer à l'expansion de l'IoT et des villes intelligentes dans le but d'améliorer la qualité de vie des citoyens.

13. Etablissent une interopérabilité entre les ordinateurs sur Internet.

14. Bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web

3.3.3 Fonctionnement et architecture

Sentilo travaille de façon à isoler la couche qui fournit les données capteurs des applications qui les exploitent.
Ses principaux modules sont :[13]

- **Un système d'abonnement et de publication** qui permet de publier ou soustraire des informations (observations, alarmes ou ordres) ainsi que de s'abonner à un système d'évènements.
- **Une base de données de stockage** visant à effectuer un stockage de données en temps réel et réaliser des taux de performance élevés.
- **Un afficheur universel** fourni une démonstration publique qui peut être utilisé comme point de départ pour les afficheurs d'entreprises spécifiques.
- **Une console d'administration** pour configurer le système et gérer le catalogue.
- **Une base de données non-SQL** afin d'obtenir un système plus flexible et évolutif
- **Un module statistique de base** qui enregistre et affiche les indicateurs de rendement de base de la plateforme.
- **Une interface frontale** pour le traitement des messages, avec une API REST [15] simple afin d'envoyer et de recevoir les données, ordres et alarmes en provenance des capteurs.
- **Une architecture de composants extensible** pour agrandir la fonctionnalité de la plateforme sans modifier le système de base. On y retrouve un premier ensemble d'agents : un agent pour exporter des données vers des bases de données relationnelles et un autre pour traiter des alarmes internes basées sur des règles de base.

L'architecture de Sentilo

15. Une API **REST** (Representational State Transfer) est une API qui utilise des requêtes HTTP pour interagir avec les applications et manipuler les données.

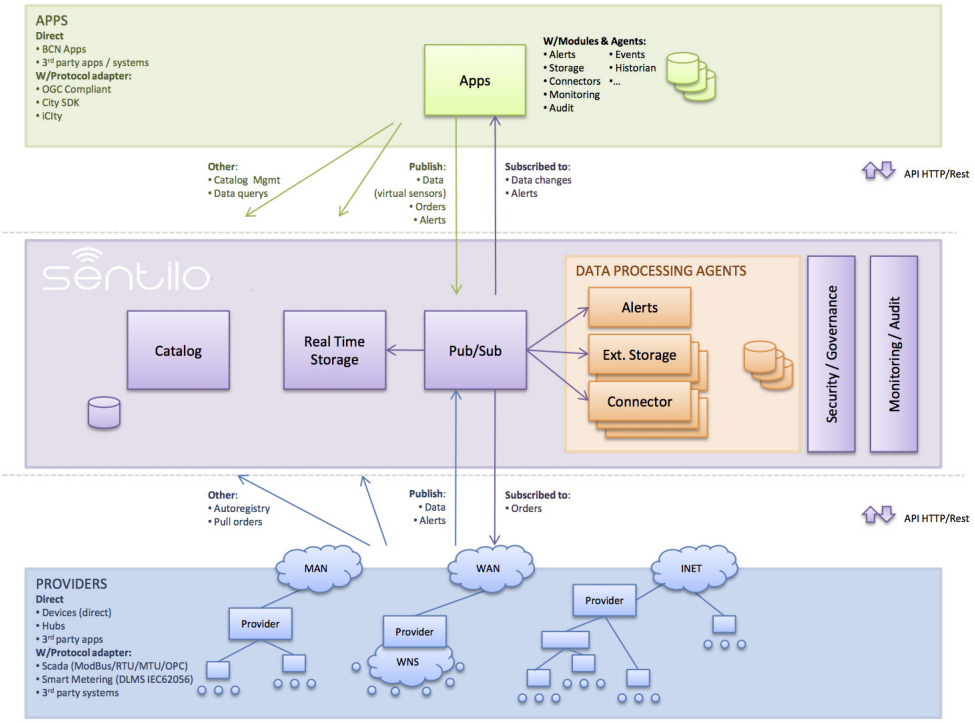


FIGURE 7 – Diagramme descriptif de l'architecture de sentilo

Sentilo charge les données capteurs, les publie et les transfère aux applications tierces selon le besoin. Pour se faire, il faut tout d'abord instancier un provider, un composant et un capteur depuis le catalogue.
Un provider est une entité qui gére les capteurs, il est caractérisé par un nom et une clé d'authentification qui lui permet de lire les données à partir du REST API.
Le catalogue est une application Web qui permet d'administrer, de régir et de surveiller les ressources et les activités de la plateforme Sentilo.[14]
Un composant est une carte qui contient un ou plusieurs capteurs (eg. Raspberry).

3.4 Proposition de solution IoT

Dans un soucis d'augmenter l'extensibilité de notre plate-forme pour des données Big Data ainsi que la gestion de ses composants, nous avons choisi de construire notre plate-forme IoT à partir de solutions open sources, à savoir, Apache Kafka, Le protocole MQTT ainsi que la base de données Mongodb.

3.4.1 Présentation de la plate-forme IoT proposée

Essentiellement, une plate-forme IoT à l'usage d'un GIS à besoin de :

1. **Bloc de connexion :**
   c'est la partie qui gère la connexion de la plate-forme IoT à l'environnement externe et entre les différents blocs composant la plate-forme. Ces connecteurs sont généralement des API (interface de programmation d'application) qui servent de façade par laquelle un logiciel offre

des services à d'autres logiciels.

Une API est une interface de programmation qui permet à deux programmes ou applications de communiquer en utilisant le même langage et de se partager des données.
Chaque API est construite selon un style d'architecture, c'est à dire un ensemble de contraintes à respecter. Les plus communes sont REST (Reprensentational State Transfer), et SOAP (Simple Object Access Protocol), basé sur XML.

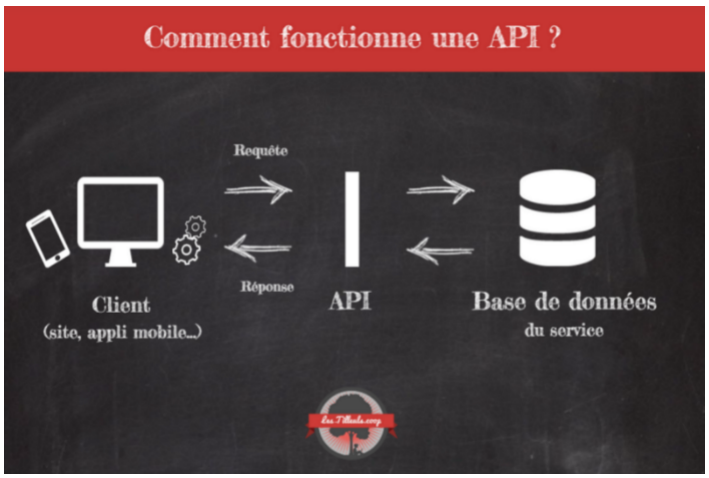


FIGURE 8 – Comment fonctionne une API

API REST sont basées sur HTTP, qui signifie Hypertext Transfer Protocol. C'est un protocole qui définit la communication entre les différentes parties du web. L'échange est basé sur des requêtes client et serveur. Un client lance une requête HTTP, et le serveur renvoie une réponse.

SOAP (Simple Objet Access Protocole) est un protocole de message qui permet la communication d'éléments distribué distants. SOAP peut être transporté sur une variété de protocoles de niveau inférieur, y compris le protocole de transfert hypertexte (HTTP). Il est le plus souvent utilisé comme API.

2. **Courtier de messages :**
C'est le bloc qui se charge de l'ingestion des données reçu grâce au connecteurs puis les redistribues vers d'autres blocs, selon l'utilisation. Les données ne stagnent pas dans ce bloc, il va servir en grande partie à la gestion du flux de ces données. En général, on parle de système de messagerie.

Système de messagerie est responsable du transfert des données d'une application à une autre, de sorte que les applications peuvent se concentrer sur les données, sans se soucier de la façon de les partager. La messagerie distribuée est basée sur le concept de la file d'attente de messages fiable. Les messages sont mis en attente de façon asynchrone entre les applications clientes et le système de messagerie.

Deux types de modèles de messagerie sont disponibles, l'un est de point à point et l'autre est un système de publication et souscription (pub-sub).

a. Système de messagerie point à point

Dans un système point à point, les messages sont maintenus dans une file d'attente. Un ou plusieurs consommateurs peuvent consommer les messages dans la file d'attente, mais un message particulier peut être consommé par un maximum d'un seul consommateur. Une fois qu'un consommateur lit un message dans la file d'attente, il disparaît. L'exemple typique de ce système est un système de traitement des commandes, où chaque commande sera traitée par un processeur de commandes, mais les processeurs de commandes multiples peuvent également fonctionner en même temps.

Le diagramme suivant illustre la structure :

FIGURE 9 – Système de messagerie point à point

b. Système de messagerie pub-sub

Dans le système de pub-sub, les messages sont persistants dans un sujet. Contrairement au système point à point, les consommateurs peuvent s'abonner à un ou plusieurs sujets dit "topics" et consommer tous les messages de ce sujet. Dans ce type de système, les producteurs de messages sont appelés éditeurs (publisher) et les consommateurs de messages sont appelés abonnés (subscriber).

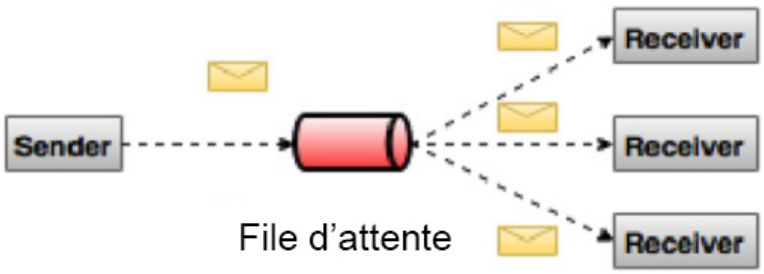


FIGURE 10 – Système de messagerie pub-sub

Chacun de ces deux systèmes a une force et une faiblesse. La force du système point à point est qu'il permet de diviser le traitement des données sur de multiples instances de consommation, ce qui permet d'échelonner le traitement. Malheureusement, les files d'attentes ne sont pas multisouscrites, une fois qu'un processus lit les données, il disparaît.
Le système Publish-subscribe permet de diffuser des données à de multiples processus, mais n'a aucun moyen de traitement d'échelle puisque chaque message va à chaque abonné.

3. **Base de stockage :**
   Finalement, une plate-forme IoT a besoin d'un bloc chargé du stockage des données récolté à partir des objets connectés. Cette base se doit d'être extensible et capable de gérer des données massives (Big Data). Celle ci sera le dernier port des données avant qu'elles ne soient envoyées à travers la couche de connecxion vers des applications tiers, à savoir, notre GIS.

3.4.2 Comparaison entre la solution IoT et Sentilo

Le choix de construire notre plate-forme ne vient forcément pas du ciel. Effectivement, bien que sentilo ait fait ses preuves pour la récolte de données IoTs, néanmoins son développement reste très restrictifs aux villes intelligentes. Aussi, on a aucun moyen de moduler son architecture pour comme par exemple ajouter une partie d'analyse de données, ce qui n'est pas présent dessus. Son extensibilité est limité, ce qui rend Sentilo peu adéquate pour un environnement de grande production.[15]
La meilleure solution s'offrant à nous est de créer une plate-forme, bloc par bloc, d'ou, la force de l'architecture. Combiner les meilleures solutions pour un systéme plus performant et évolutif capable de gérer des données de grandes échelle.

3.5 Éléments de construction de la solution IoT

Notre solution étant composé d'une plate-forme IoT et d'une passerelle IoT, nous présentons dans ce qui suit ces deux parties :

a. La plate-forme IoT

Afin de réaliser notre plate-forme, nous nous orientons vers des solutions hautement scalable et open-source, à savoir :

— Apache Kafka pour la gestion du flux de massages.
— Kafka connect pour la couche de connexion.
— Mongodb comme base de stockage IoT.

Nous allons expliciter tout cela plus en profondeurs dans les sections qui suivent.

3.5.1 Apache Kafka

3.5.1.1 Présentation d'Apache Kafka

Apache Kafka est un système de logs distribué. Un log est un système de messagerie durable. Tous les messages vont à la fin de ce log, de ce fait, on peut consommer les messages et les lire

séquentiellement. Cependant, en réalité, apache kafka est un peu plus complexe qu'un système de messagerie. Effectivement, il bénéficie de quelques fonctionnalités beaucoup plus développées.

En effet, les systèmes de messageries ont traditionnellement deux modèles : la file d'attente et la publication-abonnement. Dans une file d'attente, un groupe de consommateurs peut lire à partir d'un serveur et chaque enregistrement va à l'un d'entre eux ; dans la publication-abonnement le dossier est diffusé à tous les consommateurs. Chacun de ces deux modèles a une force et une faiblesse. La force de la file d'attente est qu'elle permet de diviser le traitement des données sur de multiples instances de consommation, ce qui permet d'échelonner le traitement. Malheureusement, les files d'attente ne sont pas multisouscrites — une fois qu'un processus lit les données, il disparaît. Publish-subscribe permet de diffuser des données à de multiples processus, mais n'a aucun moyen de traitement d'échelle puisque chaque message va à chaque abonné.

Le concept de groupe de consommateurs de Kafka généralise ces deux concepts. Comme pour une file d'attente, le groupe de consommateurs permet de diviser le traitement en un ensemble de processus (les membres du groupe de consommateurs). Comme pour la publication-abonnement, Kafka permet de diffuser des messages à plusieurs groupes de consommateurs.

Apache Kafka est en réalité une plateforme de streaming (diffusion) distribuée ce qui lui donne une qualification de gestion de données en temps réel.

Une plateforme de streaming en continu possède trois caractéristiques clés :

— Publier(Publish) et s'abonner(Subscribe) à des flux d'enregistrements, comme une file d'attente de messages ou un système de messagerie d'entreprise.
— Stocker les flux d'enregistrements d'une manière durable et résistante aux pannes.
— Traiter les flux d'enregistrements à mesure qu'ils se produisent.

Ces caractéristiques le rendent idéal pour la communication et l'intégration de différents composants de systèmes de données à grande échelle dans des systèmes de données du monde réel ou Big Data.

3.5.1.2 Histoire d'Apache Kafka

Auparavant, LinkedIn était confronté a un problème dans son architecture lambda qui était chargé de traiter des événements en temps réel. Le problème en question s'agissait de l'ingestion d'une énorme quantité de données provenant du site Web qui était de faible latence. Comme solution, Apache Kafka a été développé en 2010, puisqu'aucune solution n'était disponible pour faire face à cet inconvénient.
Cependant, des technologies étaient disponibles pour le traitement par lots, mais les détails de déploiement de ces technologies étaient partagés avec les utilisateurs en aval. Par conséquent, bien qu'il s'agisse du traitement en temps réel, ces technologies n'étaient pas suffisamment adaptées. En 2011, Kafka a été rendu public.

3.5.1.3 Composants d'Apache Kafka

Dans cette partie, nous présentons les composants majeurs d'Apache kafka, leurs rôles, ainsi que certains concepts nécessaire a son fonctionnement :

- **Les topics (sujets) :** Un topic ou un sujet est un flux de messages appartenant à une catégorie particulière. Il est divisé en partitions et garde une mini-maman d'une de ses partitions. Les topics peuvent avoir de nombreuses partitions, ils peuvent donc gérer une quantité arbitraire de données.
- **Les partitions :** Les partitions sont les éléments qui composent le topic. Chacune d'entre elles contient des messages dans une séquence ordonnée immuable. Elles sont implémentées comme un ensemble de fichiers de segments de tailles égales.
- **L'offset de partition :** Chaque message partitionnée a un identifiant de séquence unique appelée offset.
- **Les replicas de partitions :** Les replicas ne sont rien d'autre que des sauvegardes d'une partition. Ils ne sont pas utilisé pour l'écriture ou la lecture des données, ils sont utilisés pour prévenir la perte de données.
- **Les brokers (courtiers) :** Les brokers ou les courtiers sont un système simple responsable de la tenue des données publiées. Chaque broker peut avoir zéro ou plus de partitions par topic.
  1. Supposons que s'il y a N partitions dans un topic et N nombre de broker, chaque broker aura une partition.
  2. Supposons qu'il y ait N partitions dans un topic et plus de N brokers (n + m), le premier broker N aura une partition et le prochain broker M n'aura pas de partition pour ce topic particulier.
  3. Supposons qu'il y ait N partitions dans un topic et moins de N brokers (n-m), chaque broker aura une ou plusieurs partitions partagées entre eux.
- **Le cluster Kafka :** Si on dispose de plus d'un broker, on appel cet ensemble de brokers un cluster Kafka. Ce dernier peut être étendu sans temps d'arrêt. Les clusters sont utilisées pour gérer la persistance et la réplication des données des messages.
- **Les producers (producteurs) :** Les producers ou les producteurs publient des messages sur un ou plusieurs topics Kafka : Ils envoient des données aux brokers Kafka et chaque fois qu'un producer publie un message à un broker, celui-ci joint simplement le message au dernier fichier de segment.En fait, le message sera annexé à une partition. Le producer peut également envoyer des messages à une partition de son choix.
- **Les consumers (consommateurs) :** Les consumers ou les consommateurs lisent les données des brokers. Ils s'abonnent à un ou plusieurs topics et consomment les messages publiés en tirant des données des brokers.
- **Les leaders :** Le leader est le noeud responsable de toutes les lectures et écritures pour une partition donnée.Chaque partition a un serveur agissant comme un leader.
- **Les followers :** Les followers ou les suiveurs sont les noeuds qui suivent les instructions du leader. Si le leader est défaillant ou tombe en panne, l'un des followers deviendra automatiquement le nouveau leader. Un follower agit en tant que consommateur normal, tire des messages et met à jour son propre magasin de données.

3.5.1.4 Cluster Kafka

Une caractéristique très intéressante dans Apache kafka est son fonctionnement en cluster, c'est à dire en groupement de plusieurs brockers qui permet la parallélisation des traitements, plus d'efficacité et extensibilité. Dans ce qui suit, les composants principaux dans un cluster kafka ainsi que le rôle de chacun.[16]
Le diagramme ci-dessous montre le diagramme du cluster d'Apache Kafka :

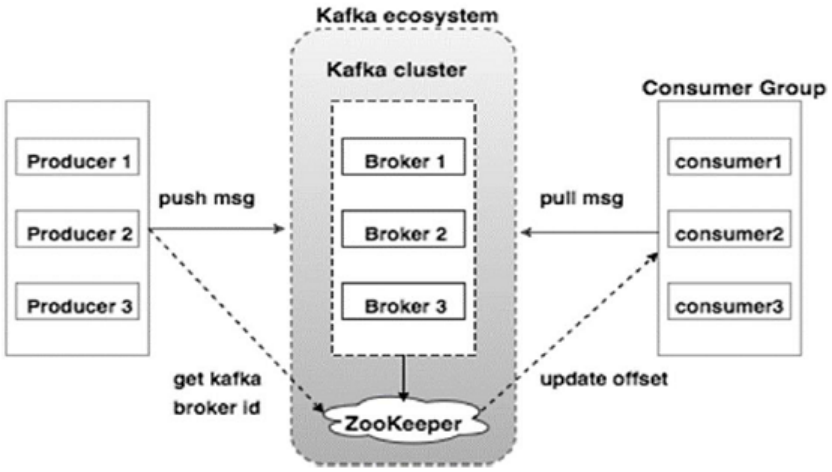


FIGURE 11 – Cluster Kafka

— **Le producer dans Kafka cluster**
Les producers transmettent les données aux brokers. Lorsque le nouveau broker est lancé, tous les producers font une recherche et envoient automatiquement un message à ce nouveau broker. Le producer de Kafka n'attend pas les accusés de réception du broker et envoie des messages le plus rapidement possible.

— **Le consumer dans Kafka cluster**
Puisque les brokers Kafka sont apatrides, ce qui signifie que le consumer doit maintenir combien de messages ont été consommés en utilisant la compensation de partition. Si le consumer reconnaît une compensation de message particulière, cela implique que le consumer a consommé tous les messages antérieurs. Le consumer émet une demande de tirage asynchrone au broker pour avoir un tampon d'octets prêt à consommer. Les consumers peuvent revenir en arrière ou sauter à n'importe quel point d'une partition simplement en fournissant une valeur de compensation. La valeur de compensation de consommation est notifiée par le zookeeper.

— **Le broker Kafka**
Kafka cluster se compose généralement de plusieurs brokers(courtiers) pour maintenir l'équilibre de la charge. Les brokers Kafka sont apatrides, donc ils utilisent un service nommé "zookeeper" pour maintenir l'état du cluster. Une instance de broker Kafka peut gérer des centaines de milliers de lectures et d'écritures par seconde et chaque broker peut gérer des Térabytes de messages sans impact de performance. L'élection du chef des brokers Kafka peut être faite par zookeeper.

— **Le coordinateur Zookeeper**
Zookeeper est utilisé pour gérer et coordonner les brokers Kafka. Le service zookeeper est principalement utilisé pour informer le producer et le consumer de la présence de tout nouveau broker dans le système Kafka ou de la défaillance du broker dans le système Kafka. Conformément à la notification reçue par le zookeeper concernant la présence ou l'échec du broker, le producer et le consumer prend une décision et commence à coordonner leur tâche avec un autre broker.

3.5.1.5 Orchestration du cluster kafka

L'Orchestration du cluster kafka se fait à l'aide de zookeeper.

a. Qu'est ce que Zookeeper ?
Zookeeper est un logiciel de haut niveau développé par Apache [16] qui agit comme un service centralisé et est utilisé pour maintenir les données de nommage et de configuration et pour fournir une synchronisation flexible et robuste dans les systèmes distribués. Zookeeper garde une trace de l'état des noeuds du cluster Kafka et garde également une trace des sujets (topics) Kafka, des partitions, etc.[17]

b. Comment fonctionne Zookeeper ?
Les données au sein du zookeeper sont divisées en plusieurs ensembles de nœuds et c'est ainsi qu'il atteint sa grande disponibilité et cohérence. En cas de défaillance d'un noeud, Zookeeper peut effectuer une migration de basculement instantanée ; ex : si un noeud maître échoue, un nouveau noeud est sélectionné en temps réel par sondage au sein d'un ensemble. Un client connecté au serveur peut interroger un noeud différent si le premier ne répond pas.

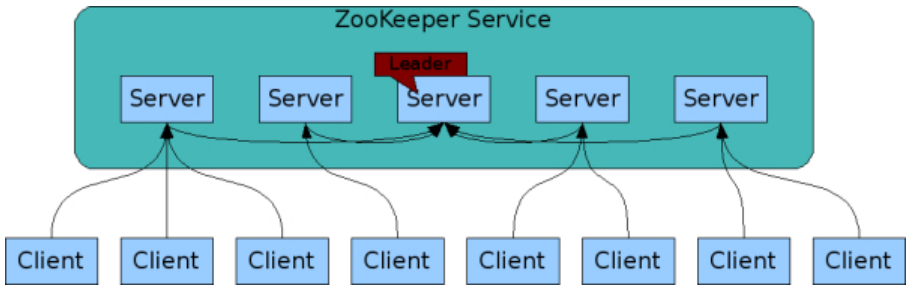


FIGURE 12 – Fonctionnement de Zookeeper

c. Pourquoi Zookeeper est nécessaire pour Apache Kafka ?

— **Choix du contrôleur**
Le contrôleur est l'une des entités de courtage les plus importantes dans un écosystème Kafka, il a la responsabilité de maintenir la relation leader-suiveur dans toutes les partitions. Si un noeud se ferme pour une raison quelconque, c'est la responsabilité du contrôleur de dire à toutes les répliques d'agir comme des chefs de partition afin de remplir les fonctions des chefs de partition sur le noeud qui est sur le point d'échouer. Ainsi, chaque fois qu'un noeud s'arrête, un nouveau contrôleur peut être élu et il peut également être assuré qu'à tout moment, il n'y a qu'un seul contrôleur et tous les noeuds suiveurs ont convenu de cela.

16. Une fondation créée en 1999 par huits développeurs qui ont créé le serveur web Apache.

— **Configuration des topics**
La configuration concernant tous les topics, y compris la liste des topics existants, le nombre de partitions pour chaque topic, l'emplacement de toutes les répliques, la liste des dérogations de configuration pour tous les topics et quel noeud est le leader préféré, etc.

— **Listes de contrôle d'accès**
Les listes de contrôle d'accès ou LCA pour tous les topics sont également tenues à jour au sein de zookeeper.

— **Composition du cluster**
Zookeeper tient également une liste de tous les courtiers qui fonctionnent à un moment donné et font partie du cluster.

Note : On ne peut pas exécuter les services Kafka sans d'abord installer Zookeeper.

3.5.1.6 Architecture et fonctionnement d'Apache Kafka

L'architecture d'Apache Kafka repose sur quatre API de base : l'API Producer, l'API Consumer, l'API Streams et l'API Connector.[18]

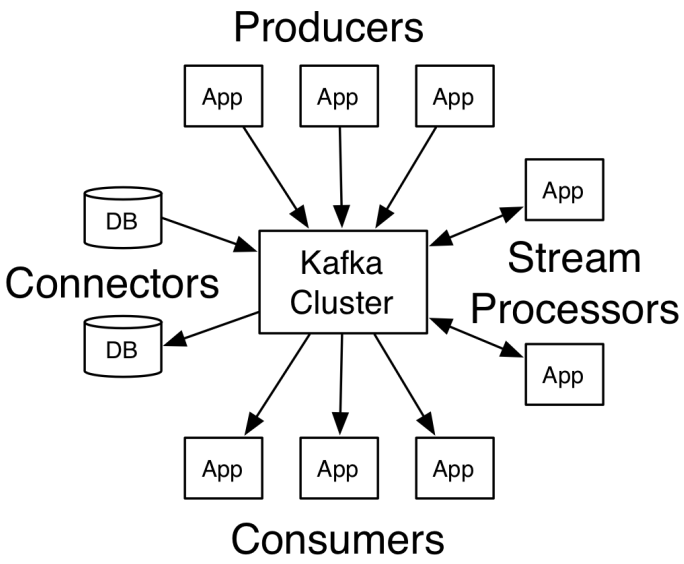


FIGURE 13 – Les APIs Kafka

— **L'API Producer :** L'API Producer permet a une application de publier un flux d'enregistrements vers un ou plusieurs topics Kafka.

— **L'API Consumer :** Cette API permet à une application de s'abonner à un ou plusieurs topics et de traiter le flux des documents qui lui sont produits.

— **L'API Streams :**l'API Stream permet a une application d'agir comme un processeur de flux, consommer un flux d'entrée d'un ou plusieurs topics et produire un flux de sortie à un ou plusieurs topics de sortie, transformant efficacement les flux d'entrée en flux de sortie.

— **L'API Connector :** Cette API est utilisé afin de construire et d'exécuter des producers ou des consumers réutilisables qui relient les topics Kafka aux applications ou systèmes de données existants.

3.5.2 Kafka connect

Kafka Connect est un framwork [17] permettant de connecter Kafka à des systèmes externes tels que des bases de données, des index de recherche et des systèmes de fichiers, en utilisant des connecteurs.

3.5.2.1 Types de connecteurs Kafka

Il en existe deux types :

— Les connecteurs **source** qui récoltent les données d'un système. Les systèmes sources peuvent être des bases de données, des tables de flux ou des brocker de messages dans les Topics de Kafka, rendant les données disponibles pour le traitement de flux avec une faible latence.

— Les connecteurs **Sink** qui transférent les données à partir de topics Kafka vers des systèmes externes comme par exemple, des bases de données.

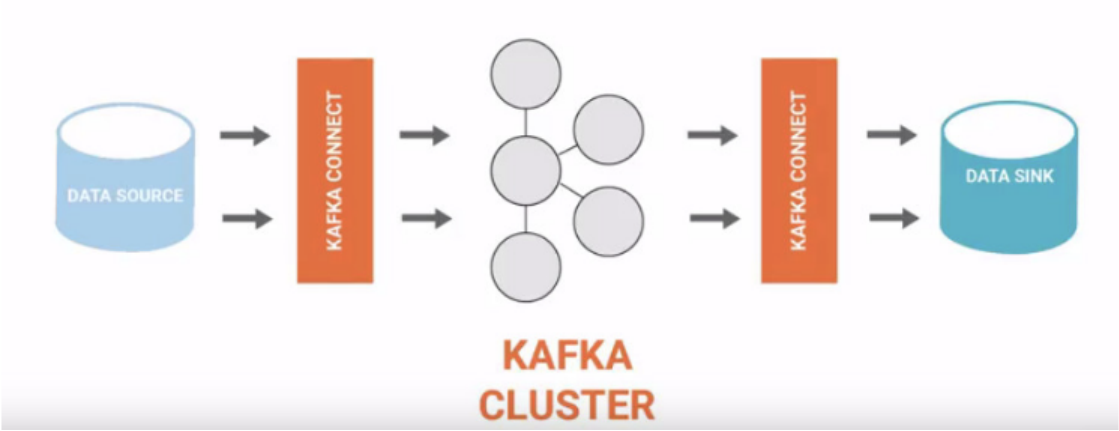


FIGURE 14 – Fonctionnement des connecteurs Kafka

3.5.2.2 Utilité des connecteurs Kafka

Les connecteurs Kafka apporte beaucoup d'ergonomie aux pipeline ETL (Extract, Load, Transform).
Un pipeline ETL implique l'extraction, la transformation et la charge. Tout moteur de traitement de flux comme Spark Streaming, Kafka Streams, etc est spécialisé pour la partie T (Transform), pas pour E (Extract) et L (Load). C'est là que Kafka cnnect entre en jeux. Kafka Connect s'occupe du E et du L, quel que soit le moteur de traitement, à condition que Kafka fasse partie du pipeline. Un avantage apparent est que le même code de E et L pour un connecteur source/sink peut être mis à

17. un farmework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel (architecture)

l'usage de différents moteurs de traitement. Cela assure le découplage et la réutilisation. Il bénéficie aussi d'un haut niveau de parallélisme des taches, la tolérance de fautes et l'ordonnancement.

3.5.3 Mongodb

Kafka étant un système de streaming de données, il n'est pas responsable de leur stockage. De ce fait, les données reçu grâce à kafka connect doivent êtres sauvegardé de façon durable afin de pouvoir les utiliser ultérieurement. Ces dernières représentent une quantité très volumineuse, ce que l'on appelle le Big Data.
Comme nous l'avons introduit précédemment, le Big Data est mieux pris en charge par les bases de données no-sql [18]. Ces bases de données fondées sur un stockage sous forme de documents nous évitent la structure à tableaux des bases de données relationnelles et nous permettent de regrouper les données de façon plus logique.

3.5.3.1 Caractéristiques de Mongodb

La base de données, Mongodb, a pris de l'avance sur le terrain pour plusieurs raisons. C'est une base open-source, elle a une compatibilité multiplateforme. elle a également des caractéristiques intégrées impressionnantes qui en font un excellent choix pour les compagnies qui ont besoin d'un accès rapide et flexible à leurs données, que ce soit pour prendre des décisions en temps réel, à la volée, ou pour créer sur mesure, des expériences axées sur les données pour les utilisateurs. elle est compatible avec les applications . NET, la plateforme Java, et plus encore. Elle a aussi été utilisé par des organisations comme Metlife, ADP, The Weather Channel, Bosch et Expedia.[19]

En outre, Mongodb nous permet de :[20][19]

Stocker de grands volumes de données qui ont souvent peu ou pas du tout structurés : Mongodb ne fixe aucune limite et permet d'ajouter différents types de données à mesure que les besoins changent. Elle est flexible et basé sur des documents, elle permet de stocker de type JSON (appelés BSON) dans un seul endroit sans avoir à définir au préalable quels « types » de données.

Tirez le meilleur parti de l'informatique en nuage (cloud computing) et du stockage : Le stockage en cloud (nuage) est une excellente solution économique, mais exige que les données soient facilement réparties sur plusieurs serveurs pour l'étendre. MongoDB peut charger un volume élevé de données et fournit beaucoup de flexibilité et de disponibilité dans un environnement cloud, avec des solutions de partage intégrées qui la rendent facile à partitionner et à répartir les données sur plusieurs serveurs.

Développer et produire de façon rapide : Avec les schémas de données dynamiques de MongoDB, nous pouvons procéder à des manipulations rapidement. Les données n'ont pas besoin d'être préparées à l'avance, elle offre la possibilité d'incorporer quoi que ce soit de nouveau, rapidement, et à un coût moindre.

Développer l'architecture de la base de données manière efficace et peu coûteuse : Elle permet de répartir les données sur le matériel de base, sur un site web ou sur le cloud sans avoir besoin de logiciel supplémentaire.

18. Not only structured query langage

3.5.3.1.1 Enfin Mongodb permet de tirer le meilleur parti des données capteurs et des appareils connectés, ou l'IoT. Des milliards de capteurs et de dispositifs connectés créent des millions de points de données, il est difficile pour les bases de données relationnelles d'absorber et d'analyser les processus d'ETL (extraction, transformation et chargement) qui prennent beaucoup de temps. MongoDB peut analyser des données de toutes sortes dans la base de données elle-même.

b. La passerelle IoT

3.5.4 Passerelle IoT

Une passerelle IoT est un dispositif physique ou un logiciel qui sert de point de connexion entre la plate-forme IoT et les contrôleurs, capteurs et dispositifs intelligents.

3.5.5 Communication IoT

Nous avons choisi pour la communication entre les objets connectés, le protocol MQTT. Dans ce qui suit, nous allons présenter ce protocole ainsi que notre motivation par rapport à son choix.

3.5.5.1 Protocole MQTT

MQTT(Message Queuing Telemetry Transport) est un protocole de messagerie de type Publish-Subscribe. Il fonctionne sur le protocole TCP/IP.Il a été développé en 1999 par Arlen Nipper (Arcom) et Andy Stanford-Clark (IBM). Il est principalement utilisé pour la surveillance depuis une région éloignée de l'IoT. La tache principale du MQTT est d'obtenir des données de divers appareils électroniques. Il les achemine également aux communications IT ou au infrastructure informatique. Une architecture en étoile est fondamentalement ordinaire pour le protocole IoT MQTT.

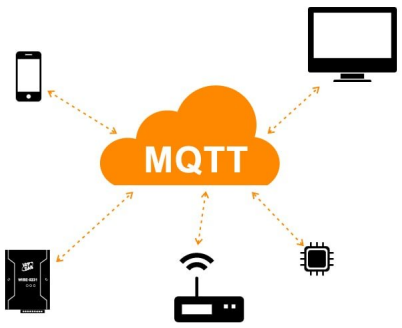


FIGURE 15 – Protocole MQTT

3.5.5.2 Broker MQTT

Le broker MQTT est au cœur du protocole de publication/souscription MQTT. En fonction de la mise en œuvre, un broker MQTT peut gérer jusqu'à des milliers de clients MQTT connectés simultanément.Il est responsable de recevoir tous les messages, de filtrer les messages, de déterminer qui est abonné à chaque message et d'envoyer le message à ces clients abonnés.En bref, le broker

MQTT est le carrefour central par lequel chaque message utilisant le protocole MQTT doit passer.

Il existe de nombreux brokers MQTT disponibles, ils varient dans leurs fonctionnalités et certains d'entre eux mettent en œuvre des fonctionnalités additionnelles.

Nous avons choisit d'utiliser pour notre projet Mosquitto qui un broker de message open source (sous licence EPL/EDL) qui implémente les versions 5.0, 3.1.1 et 3.1 du protocole MQTT.Il est léger et convient à une utilisation sur tous les appareils, depuis les ordinateurs à carte unique à faible puissance jusqu'aux serveurs complets. Le projet Mosquitto fournit également une bibliothèque C pour la mise en œuvre des clients de MQTT, et la très populaire mosquitto_pub et mosquitto_sub command line clients de MQTT.

3.5.6 Association Kafka et Mqtt

MQTT et Kafka, sur papier, sont des plates-formes de messagerie. Mais une distinction clé entre les deux concerne la façon dont ils sont utilisés. Contrairement à Kafka, MQTT est un protocole de messagerie de machine à machine (M2M). Cela signifie que si Kafka est responsable de la gestion des données et de l'ingestion dans les systèmes de calcul, MQTT se spécialise dans l'échange de paquets de données entre deux dispositifs physiques. le MQTT est uniquement destiné aux ensembles de données de petite taille, où la réactivité et l'efficacité énergétique sont prioritaires. Cela ne convient pas aux grands ensembles de données, le résultat étant une mauvaise mise à l'échelle. Kafka, d'autre part, est conçu spécifiquement avec des flux de données massives à l'esprit. Nous avons choisi d'utiliser pour notre solution IoT Kafka et MQTT en conjonction, coexistent séparément pour une performance et une évolutivité maximales.

A cet effet, deux solutions s'ouvrent à nous :

a. Brocker MQTT, Apache Kafka, Kafka connect et MQTT connect :

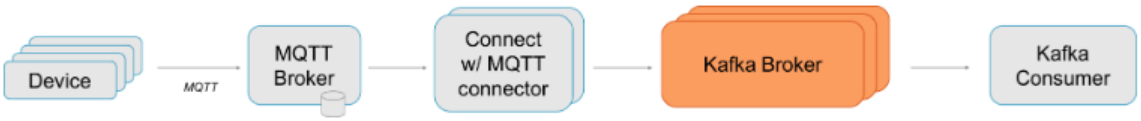


FIGURE 16 – Brocker MQTT et Apache Kafka

Dans cette approche, nous extrayons des données du brocker MQTT vers notre apache kafka grace au connecteur MQTT-Kafka, qui est un connecteur fournit par le groupe kafka. En effet, cette communauté bénéficie d'un grand nombre de connecteurs tous servis par kafka-connect. De ce fait, nous pouvons profiter des avantages qu'offre Kafka connect tel que la tolérance de défaut intégrée, équilibrage de charge, Convertisseurs, et Transformations de messages simples (SMT) pour le routage / filtrage / etc.

b. Proxy MQTT et Apache Kafka :

FIGURE 17 – Proxy MQTT et Apache Kafka

Dans ce cas de figure, nous contournons Kafka connect en destituant le brocker MQTT. Pour se faire, Nous faisons appel au Proxy [19] MQTT de la plate-forme confluent (qui est la plateforme oficielle de Kafka), c'est à dire que nous envoyons les données directement à kafka via le proxy MQTT

Les deux architectures ont leurs avantages et inconvénients. l'architecture b permet de passer outre kafka-connect. Mais vu que nous avons choisi d'utiliser mongodb en tant kafka consumer et que pour le connecter à kafka l'usage de kafka-connect est nécessaire, l'avantage de l'architecture b en est plus un. En plus, le broker MQTT mosquitto est open source, léger et convient à une utilisation sur tous les appareils. Cela nous a conduit à choisir l'architecture a.

19. Un proxy est un composant logiciel informatique qui joue le rôle d'intermédiaire en se plaçant entre deux hôtes pour faciliter ou surveiller leurs échanges.

3.6 Architecture général de la plate-forme IoT

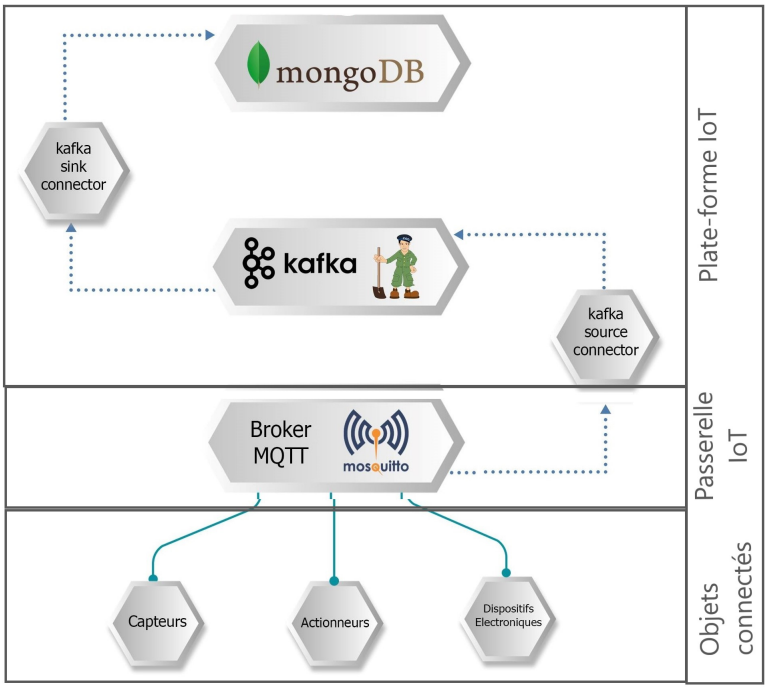


FIGURE 18 – Schéma général de la solution IoT

3.7 Conclusion

Dans ce chapitre, nous avons décrit les différentes briques constituant notre plate-forme IoT ainsi que son architecture finale. Dans ce qui suit, nous allons procéder à parler des plate-forme GIS, le traitement de données spatiales ainsi la solution GIS que nous avons selectionnée.

Chapitre 4

4 Plate-forme GIS

4.1 Introduction

Un GIS nous permet de représenter et d'analyser des objets présents sur terre ainsi que tous les événements qui s'y produisent. Dans ce chapitre, nous allons présenter en détail ce qu'est une plate-forme GIS ainsi que la solution proposé pour son implémentation.

4.2 Fonctionnalités du GIS

Premièrement, la fonctionnalité consiste essentiellement à prendre en charge la gestion des données géographiques, c'est dire à organiser l'information géographique, à en faire l'analyse et à la communiquer. Il permet de :[21]

- Visualiser
  - La cartographie générale ou thématique (données relatives à un ou plusieurs thèmes particuliers)
  - Des Atlas interactifs (Web mapping)
- Interroger
  - Des requêtes attributaires : interrogation de la table attributaire d'une couche, sur un ou plusieurs champs qui permettent de sélectionner des entités ayant des propriétés (attributs) communes.
  - Des requêtes spatiales : interrogation portant sur la géométrie et la position des entités d'une ou plusieurs couches et permettant de sélectionner des entités en fonction des entités d'une autre couche.
- Gérer
  - L'acquisition (importation, géoréférencement, digitalisation, ...)
  - La maintenance (mises à jour, corrections)
  - L'optimisation (compression, conversion, pyramides, cache, ...)
- Analyser
  - Les géotraitements
  - Les scripts

4.3 Principe de fonctionnement du GIS

On estime qu'environ 80 % de toute l'information a une composante "spatiale" ou géographique.[22] Autrement dit, la plupart des renseignements sont liés à un endroit. Ainsi, la géographie joue un rôle important. Un GIS combine des renseignements -auparavant non liés- en cartes ergonomiques faciles à comprendre. Il peut exécuter des fonctions analytiques complexes et présenter les résultats visuels sous forme de cartes, de tableaux ou de graphiques, ce qui permet aux décideurs de voir

virtuellement les questions dont ils sont saisis, puis de choisir le meilleur plan d'action.

Le GIS utilise des couches, appelées « thèmes », pour superposer différents types d'information, tout comme certaines cartes statiques utilisent des superpositions pour ajouter des niveaux d'information à un arrière-plan géographique. Chaque thème représente une catégorie d'information, comme les routes ou le couvert forestier. Comme pour les anciennes cartes, les couches qui sont en dessous restent visibles tandis que les thèmes supplémentaires sont placés au-dessus.[22]

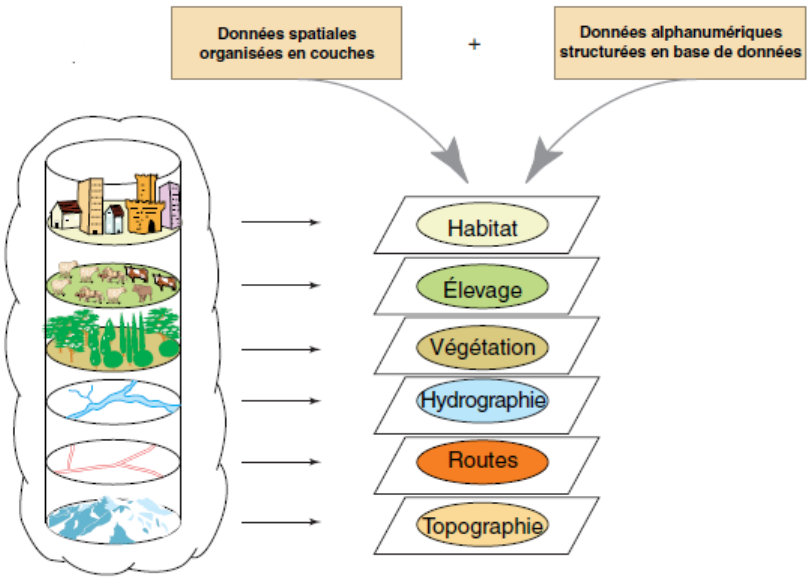


FIGURE 19 – Structuration de l'information géographique

4.4 Données du GIS

Les données géographiques représentent la partie la plus importante dans un GIS. Elles passent par certaines normalisations et traitements afin d'être utilisées par ces systèmes et standardisées pour créer d'immenses systèmes d'informations partagés ou privés. C'est à cet effet que nous commencerons par parler de latitude et longitude allant jusqu'aux modes de représentation des données et passant par les différentes normalisation que ces dernières doivent subir.

4.4.1 Latitude et longitude

Pour positionner un lieu sur le globe terrestre, nous utilisons les unités angulaires, la latitude et la longitude. La latitude mesure l'angle entre le plan équatorial et une ligne perpendiculaire à la surface passant par ce point ; tandis que la longitude mesure l'angle Ouest ou Est d'un méridien de référence (par exemple, celui passant par l'observatoire de Greenwich) à un autre méridien passant par ce point. Les mesures angulaires peuvent être exprimées en degrés décimaux ou en degrés, minutes et secondes.[23]

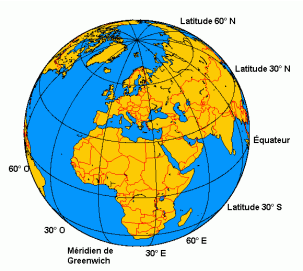


FIGURE 20 – Representation de la latitude

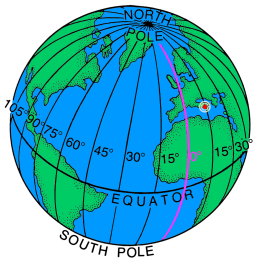


FIGURE 21 – Representation de la longitude

La planète terre n'étant pas une sphère parfaite, mais plutôt un géoïde [20]. Ce modèle est assez complexe à étudier et notamment pour des projections, nous approchons cette structure à un ellipsoïde (L'ellipsoïde est décrit par son demi-grand axe (rayon équatorial) et son aplatissement).

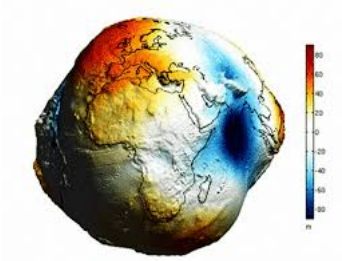


FIGURE 22 – Format géoïde de la terre

4.4.2 Projection d'une sphère sur un plan

Quand on épluche une orange, en considérant la peau de l'orange comme la surface terrestre et essayant tant bien que mal d'aplatir ces épluchures ; la distorsion de cette dernière est inévitable. C'est à se demander alors, comment les cartographes représentent-ils une surface courbe sur un plan ? Tout simplement grâce à une opération mathématique appelée "projection". Plusieurs projections ont été développées mais il n'y a pas de méthode mathématique pour transférer une sphère ou un ellipsoïde dans un espace à deux dimensions sans distorsion. Par conséquent, les projections modifient les données et incluent des déformations sur les longueurs, les surfaces ou les formes que nous pouvons observer et mesurer sur des cartes.[23]

Les projections cartographiques sont classifiées en fonction des propriétés géométriques qu'elles préservent :[23]

— La projection conforme, qui conserve les angles, tel que le méridien, est perpendiculaire sur toutes les parallèles (latitude) : de ce fait elle préserve les formes.

— La projection équivalente, qui préserve les proportions entre les surfaces.

— La projection équidistante, qui conserve la même échelle le long d'une ou plusieurs lignes. Elle conserve les distances sur les méridiens.

20. Le géoïde est une surface équipotentielle du champ de pesanteur coïncidant au mieux avec le niveau moyen des océans et qui se prolonge sous les continents.

Nous choisissons la projection qui nous est la plus adaptée selon nos besoins, étant donné qu'il n'y a pas de "meilleure" projection.

4.4.3 Système de coordonnées

Les systèmes de coordonnées utilisent les concept précédents (la surface de la terre et la projection) pour créer une zone de référence permettant de placer des objets à la surface de la Terre. On en distingue deux types :[23]

- **Les Systèmes de Coordonnées Géographiques SCG** utilisent la latitude et la longitude comme angles mesurés à partir du centre de la terre. Un SCG est défini essentiellement par l'ellipsoïde utilisé pour modéliser la Terre et par la position de l'ellipsoïde par rapport au centre de la Terre, appelée "datum". Un datum est créé sur l'ellipsoïde sélectionné et peut incorporer des variations locales d'altitude, étant donné que la forme réelle de la terre est un géoïde, plusieurs ellipsoïdes (à surface lisse) peuvent exister et donc plusieurs datums pour un seul lieu.
- **Systèmes de coordonnées projetés** sont définis sur une surface plate bidimensionnelle. Un système de coordonnées projetées est toujours basé sur un système de coordonnées géographiques ; par conséquent, il utilise un ellipsoïde et un datum.

De nos jours, il existe plusieurs systèmes de coordonnées, mais les plus utilisés restent :[23]

Transverse universelle de Mercator UTM qui est une projection cylindrique. Elle utilise un cylindre tangent à un méridien pour dérouler la surface de la Terre. Un maximum de 5° de distorsion par rapport au méridien central est acceptable. Cette projection divise la surface de la terre en 60 fuseaux de 6° en séparant l'hémisphère Nord et l'hémisphère Sud, un total de 120 zones (60 pour le Nord et 60 pour le Sud).

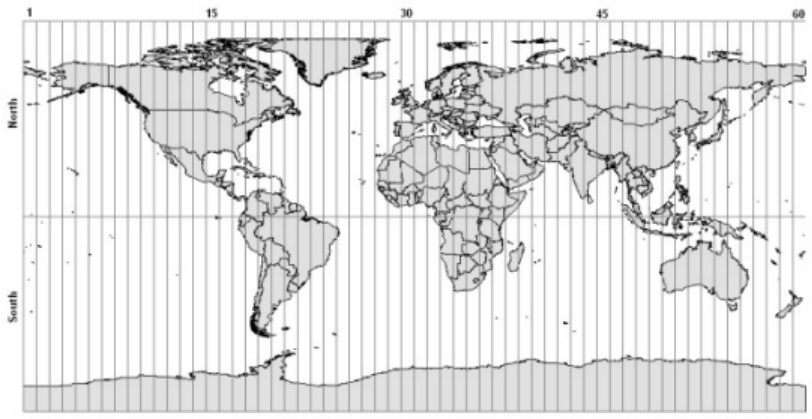


FIGURE 23 – La Transverse universelle de Mercator UTM

Pseudo-Mercator appelé aussi Web Mercator qui est une projection dérivée de La Transverse Mercator. Il cartographie les coordonnées (latitude et longitude) ellipsoïdales sur un plan en utilisant les équations de Mercator Sphérique (considère la terre comme une sphère et non

ellipsoïde). Cette projection a été popularisée par Google dans Google Maps ainsi qu'Open Street Map (OSM), et il est maintenant largement utilisé dans les systèmes de cartographie en ligne.

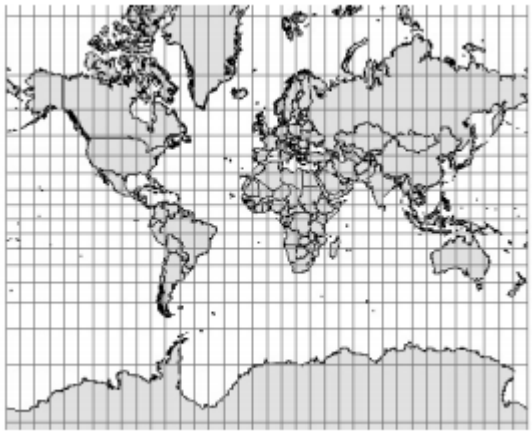


FIGURE 24 – La Pseudo-Mercator

4.4.4 Identifiant de Référence Spatiale (SRID)

Un Identifiant de Référence Spatiale (SRID, Spatial Référence Identifier) est un identifiant unique associé à un système de coordonnées. Il permet de consulter facilement un Système de Référence Spatiale (SRS). Un SRS contient des paramètres sur la projection, l'ellipsoïde et le datum. Il peut être défini à l'aide de la représentation du well-known text (WKT) de l'OGC [21].[23] Le SRS du système de référence géographique WGS84 est le suivant :

```
GEOGCS["WGS 84",
DATUM["WGS_1984",
SPHEROID["WGS 84",6378137,298.257223563,
        AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
UNIT["degree",0.01745329251994328,
        AUTHORITY["EPSG","9122"]],
        AUTHORITY["EPSG","4326"]]
```

FIGURE 25 – WGS84

Le chiffre 4326 de la dernière ligne est le SRID qui est l'unique identifiant de cette SRS. Il existe plusieurs SRID standard reconnus, tels que ceux définis par l'European Petroleum Survey Group (EPSG). Certaines bases de données et certains types spatiaux, tels que la géométrie PostGIS dans PostgreSQL ou le type de géographie dans SQL Server, utilisent un sous-ensemble prédéfini de codes EPSG, et seules les références spatiales dotées de ces SRID peuvent être utilisées.

21. L'Open Geospatial Consortium, ou OGC, est un consortium international pour développer et promouvoir des standards ouverts, les spécifications OpenGIS, afin de garantir l'interopérabilité des contenus, des services et des échanges dans les domaines de la géomatique et de l'information géographique.

Les codes EPSG constituent une liste des systèmes de coordonnées géoréférencées de projection utilisés tout d'abord par le groupe de producteurs de pétrole du même nom avant d'être utilisés par de nombreux logiciels de GIS. Ces codes sont notamment utilisés dans les standards de l'OGC (Open Geospatial Consortium).[24]

4.4.5 Modes de représentations de l'information géographique dans un GIS

Le niveau géométrique est la description de la position et de la forme des objets. La position peut s'exprimer par la latitude et les longitude des objets (ou des points qui composent ces objets) ou par des coordonnées x, y dans un système de projection.

— **Mode Vecteur :** Les objets peuvent être identifiés sous forme de points (villes, entreprises, exploitations agricoles,...), d'arcs ou de lignes (routes, chemins de fer,...) et de polygones ou de surfaces (communes, occupation du sol,...).[25]

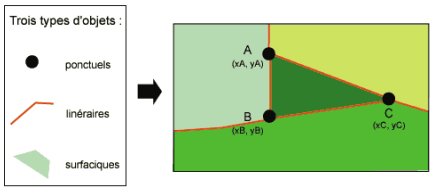


FIGURE 26 – Mode vecteur

— **Mode Raster :** Ce mode correspond à une division régulière de l'espace sous forme de cellules ou mailles généralement carrées appelées pixels, qui définissent la précision minimale de la structure. La juxtaposition des points recrée l'apparence visuelle du plan et de chaque information.[25]

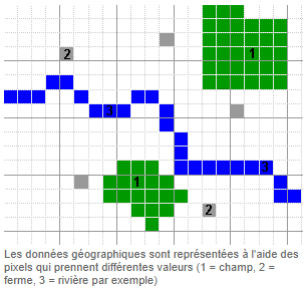


FIGURE 27 – Mode raster

La description d'un objet géométrique est appelé une **sémantique**. A chaque objet est attribuée une fiche contenant des informations de type alphanumérique. Ces informations décrivent l'objet (nom de la ville, numéro INSEE de la commune, type de l'occupation du sol,...). Ces fiches permettent de stocker des informations qui décrivent les objets : le contenu dépend des besoins du projet.

FIGURE 28 – Exemple de sémantique

C'est là où nous parlons de données attributaires : les données attributaires décrivent les objets graphiques de la carte. Il existe un lien dynamique dans le logiciel GIS entre les données géographiques, d'une part, et les données alphanumériques. Toutes ces données sont stockées dans des tables dites attributaires contenu dans des bases de données géographiques.

4.4.6 Bases de données géographiques

Le GIS actuel est utilisé pour afficher et analyser les données spatiales liées aux bases de données. Cette connexion représente la puissance au GIS : les cartes peuvent être tirées de la base de données et les données peuvent être référencées à partir des cartes. Lorsqu'une base de données est mise à jour, la carte associée peut également être mise à jour. Les bases de données GIS comprennent une grande variété d'informations, notamment géographiques, sociales, politiques, environnementales et démographiques.

Une base de donnée géographique est un ensemble de données spatiales et non spatiales structurées et organisées de manière à être interrogeables et analysables de façon interactive ou automatique. Une base de données géographique concerne habituellement une zone définie. Elle est gérée par un logiciel GIS. Elle intègre les données elles-mêmes ainsi que leurs métadonnées.

4.5 Serveur web géographique

4.5.1 Définition

Un serveur web géographique (Web Map Server en anglais) est un logiciel qui produit des cartes de données spatialement référencées d'une manière dynamique à partir d'information géographique.Il représente l'interface dédiée aux applications cartographiques basées sur le web.
Il fonctionne en produisant des requêtes afin de charger des couches désirées sous différents formats d'images (GIF, PNG ...etc), ou en format vectoriel (par exemple KML, GML).[26]

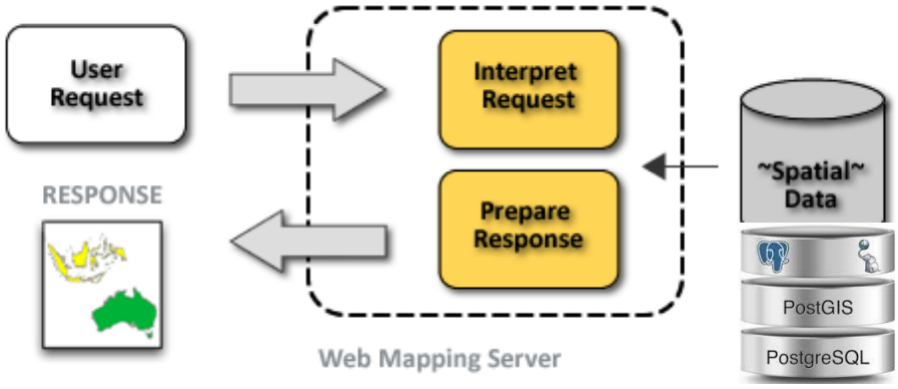


FIGURE 29 – Fonctionnement d'un serveur web cartographique

Maintenant, afin d'effectuer des requêtes sur les données géographiques, les services web cartographiques utilisent des protocoles de structures, aussi nommé "services de données spatiales", à l'exemple de :

- Web Map Service (WMS)
- Web Feature Service (WFS)
- Web Coverage Service (WCS)
- Web Processing Service (WPS)

Afin de mieux cerner ce que cela représente, nous allons parler plus en profondeur de ce qu'est un service de données spatiales.

4.5.2 Services de données spatiales

Un Service Web est un programme informatique permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur Internet ou sur un Intranet, par et pour des applications ou machines, sans intervention humaine, et en temps réel.[27]

Les services de données spatiales sont des services web permettant d'effectuer des traitements géomatiques ou géographiques (géocodage...), de renvoyer des cartes ou de donner accès à des données géographiques (débit d'un fleuve, altitude, nom d'une zone géographique...).[26]

Pourquoi utiliser un service web ?

L'intérêt vient principalement du besoin d'interopérabilité des GIS[1]

Le partage des données : Sert l'utilisation de données communes à l'ensemble des utilisateurs. En d'autres termes, ils permettent de combiner sur un poste utilisateur des données venant de plusieurs sources distantes ou locales, et de les traiter comme si elles étaient stockées sur le poste. Il est également possible de mettre en ligne sur un outil GIS Internet/Intranet des données provenant de plusieurs sources sans avoir à les transférer et les convertir périodiquement.

La mutualisation des ressources : Permet la décharge des ressources sur un serveur commun pour éviter d'acheter une machine très puissante et souvent onéreuse pour chaque géomaticien.

La Mutualisation des compétences : Aide au travail collaboratif et le developpement des GIS.

Quels sont les formats normalisés par l'OGC ?[1][28]

Par définition, le Web Map Service (WMS), le Web Map Tile Service (WMTS), le Web Coverage Service (WCS), le Web Feature Service (WFS) et le Web Feature Service Transactional (WFS-T) sont des protocoles HTTP fournissant des données géoréfencées par des URL [22] selon les normes internationales et standardisées de l'OGC. On parle souvent de flux de données. Ces différents services sont mis à disposition par des serveurs cartographiques tels les serveurs open-source Geoserver et Mapserver. Les données spatiales diffusées peuvent être lues directement sur un navigateur web, à partir de bibliothèques tel que Openlayers ou sur un logiciel client comme QGIS.

Web Map Service (WMS)

Le WMS définit un service permettant d'extraire une carte (ou une image) de données géoréférencées pour lesquelles plusieurs opérations sont disponibles (table 1). La donnée spatiale transmise par le serveur cartographique peut être à l'origine un vecteur ou un raster.

Opération	Description
Exceptions	Si une exception apparaît
GetCapabilities	Fournit les métadonnées du service avec les opérations, les paramètres et la liste des couches disponibles.
GetMap	Affiche une image géoréférencée pour un périmètre et une donnée spécifiés.
GetFeatureInfo (optionnel)	Affiche une donnée avec la géométrie et les valeurs correspondantes, pour un pixel sur la carte.
DescribeLayer (optionnel)	Indique le WFS ou WCS afin de fournir des informations supplémentaires sur la couche.
GetLegendGraphic (optionnel)	Affiche une légende liée à la couche.

TABLE 1 – Les opérations sur les requêtes WMS

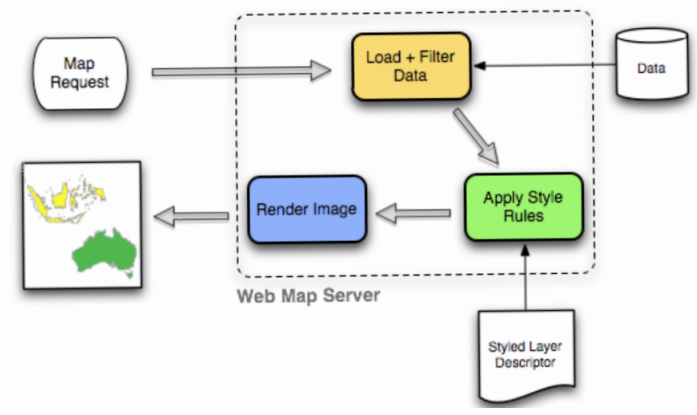


FIGURE 30 – Comment le WMS transforme des données en une carte

22. Uniform Resource Locator en anglais, littéralement "localisateur uniforme de ressource", désigne le nommage uniforme d'une ressource localisée.

Web Map Tile Service (WMTS)

Le WMTS est comme son nom l'indique un service HTTP qui fournit des images géoréfencées tuilées. A la différence du WMS, ce service charge des données calculées à priori "côté serveur" et ainsi il permet d'alléger les temps de chargement des données "côté client".

Web Feature Service (WFS)

Le WFS est un protocole d'échange de données vecteurs qui permet « côté client » de connaître les structures et les sources de la donnée spatiale. C'est un protocole d'échange de données vecteurs. Les opérations proposées par toutes les versions du WFS sont répertoriées dans la table 2.

Opération	Description
GetCapabilities	Génère les metadonnées qui décrit le service WFS avec les paramètres et opérations disponibles.
DescribeFeatureType	Décrit les types de données supportés par le service WFS.
GetFeature	Fournit une sélection d'objets géographiques (valeurs attributaires, géométrie) depuis une couche spatiale.
LockFeature	Empêche l'édition d'un objet géographique.
Transaction	Permet de créer, modifier et supprimer un objet géographique d'un vecteur.

TABLE 2 – Les opérations sur les requêtes WFS

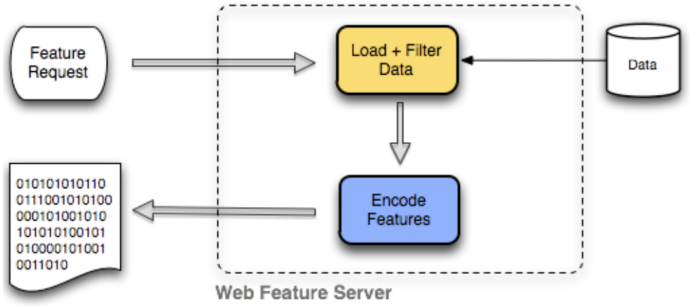


FIGURE 31 – Comment un WFS transforme une requête en une réponse

Web Feature Service Transactionnel (WFS-T)

Un service WFS de base ne permet que l'extraction d'entités, tandis qu'un service WFS de transaction offre en outre la possibilité de manipuler des entités, il permet la création, la suppression et la mise à jour des fonctionnalités.

Web Coverage Service (WCS)

Le WCS charge uniquement des données rasters et permet de les traiter de manière plus approfondie que le WMS. A travers ce protocole, la donnée peut être traitée à la base avec la possibilité de réaliser des requêtes plus précises. Il fournit une couverture, c'est-à-dire de l'information géographique numérique représentant des phénomènes variant dans le temps et dans l'espace (Modèle Numérique de Terrain, image satellite, etc...).

Opération	Description
GetCapabilities	Affiche une liste des données du serveur qui ont des paramètres et opérations liés au WCS valides.
DescribeCoverage	Produit un document XML qui décrit entièrement les requêtes des couvertures de la donnée.
GetCoverage	Fournit une image géoréférencée comme la requête GetMap pour le WMS dans des différents formats .

TABLE 3 – Les opérations sur les requêtes WCS

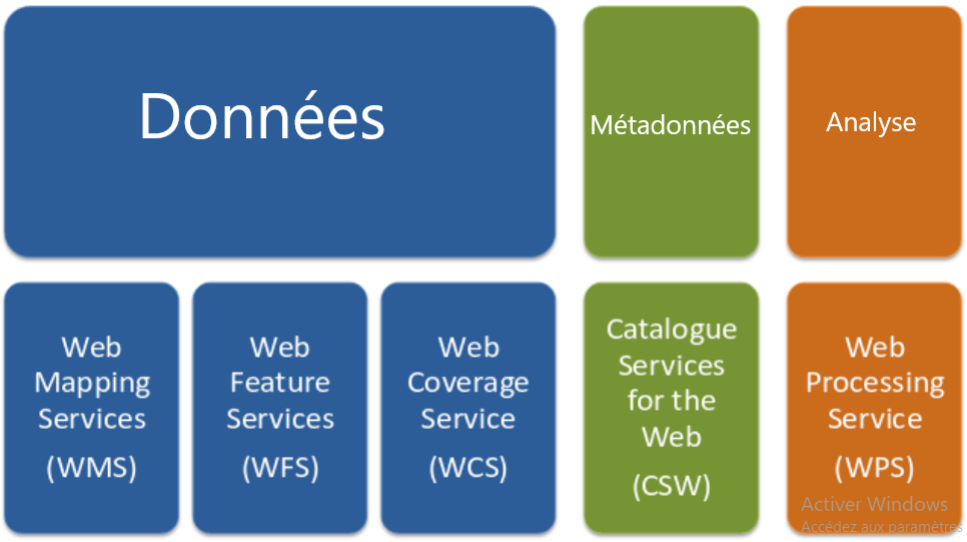


FIGURE 32 – Les services de données spatiales

Voici un tableau (Table 4) résumant les services de données spatiales les plus utilisés.

WMS Web Map Service	Fournit une carte au format image, pouvant correspondre à plusieurs couches de données
WFS Web Feature Service	Accés aux données vectorielles (géométrie et attributs)
WFS-T Web Feature Service Transactionnel	Création, modification et suppression d'objets
WCS Web Coverage Service	Visualisation de données raster Accès aux données (attributs)
CSW Catalog Service Web	Accès aux metadonnées Recherche parmi les données disponibles
WPS Web Processing Service	Géotraitement de données sur des serveurs distants
CT Coordinate Transformation	Changement de système de coordonnées
OpenLS Location Service	Service de base pour les applications mobiles : affichage de carte, géocodage, calcul d'itinéraire ...
SOS Sensor Observation Service	Gestion de capteurs et collecte de données de ces capteurs
SPS Sensor Planning Service	Service de planification de l'interrogation de capteurs (et récupération de données associées)

TABLE 4 – Principaux standards de services de l'OGC.[1]

Lorsqu'un client envoi une requête à un serveur, tout d'abord, c'est une requête pour connaître les possibilités du serveur (GetCapabilities), il peut demander une description supplémentaire pour une couche particulière puis demande les données, le traitement, (GetMap ou GetFeatureInfo dans le cas d'un WMS)... Après chaque requête le client reçoit une réponse sous forme de fichier XML ou image dans le cas des services web de l'OGC.

4.6 Application cliente cartographique

Au final, tout le système GIS sera visualisé et mis à disposition de l'utilisateur grâce à une application cliente dédiée.
Une manière de rendre un système GIS utilisable par un groupe de personnes n'ayant pas de grandes capacités techniques est de le mettre sur une page web. A cet effet, nous avons choisi comme application cliente une application web cartographique.
Pour mieux comprendre ce que c'est, nous allons introduire certaines notions sur la cartographie

web ou plus communément appelé en anglais le "web mapping", ainsi que les bibliothèques web cartographiques.

4.6.1 Cartographie web

La cartographie web ou "web mapping" est le fait d'utiliser des systèmes d'informations géographiques (GIS) sur le world wide web.

Une carte géographique sur le web est à la fois servis et consommés par les clients. c'est à dire qu'un utilisateur a la possibilité de choisir ce que la carte lui fournis comme informations ainsi que d'y contribuer avec ses propres données géographiques.

Ces cartes sont étroitement liées aux aspects de la conception du système qui se cache derrière elles comme l'acquisition et le stockage de données, l'architecture logicielle des serveurs, les algorithmes employés, jusqu'aux rapports de l'utilisateur final lui-même.[29]

Les applications web cartographiques se doivent d'être rapides, précises et faciles à utiliser. Puisqu'elles sont en ligne, on s'attend à ce qu'elles soient accessibles de n'importe où et sur presque n'importe quelle plateforme. Il n'y a que quelques outils qui répondent à toutes ces attentes, parmi eux OpenLayers et Leaflet.

4.6.1.1 OpenLayers

Une bibliothèque javascript opensource pour charger, afficher et récupérer des cartes à partir de plusieurs sources sur des pages Web. Développée à l'origine par MetaCarta en réponse, en partie, à Google Maps, la série 2.x de la bibliothèque est devenue un cadre mature et populaire avec de nombreux développeurs passionnés et une communauté très utile.[30]

4.6.1.2 Leaflet

Leaflet est une bibliothèque JavaScript open source pour les cartes interactives mobiles. Il est développé par Vladimir Agafonkin de Mapbox avec une équipe de contributeurs dédiés. Pesant environ 30 KB de code JS compressé, il possède toutes les fonctionnalités dont la plupart des développeurs ont besoin pour les cartes en ligne.[31]

4.7 Architecture classique d'un GIS

Un systéme d'information géographique repose sur 3 éléments, les données, les services et l'application cliente. Comme nous nous penchons vers un WebGIS, celui-ci sera forcément déployé sur internet et reposer sur l'architecture client-serveur. Dans ce qui suit, les 3 éléments constituant l'architecture d'un WebGIS :

Un GIS a besoin au minimum d'une source de données géographique. Nous pouvons représenter celà par une base de donnée géographique qui va se charger, traiter, stocker et gérer les données spatiales.
La base de donnée peut être hébergée par le même serveur web cartographique ou totalement dissociée de ce dernier, et cela selon l'architecture désirée.

Les serveurs sont des ordinateurs équipés de logiciels particuliers capables de répondre à des requêtes émises par les clients si elle fait partie du type de service proposé. Ils sont toujours en attente des demandes des clients. On dit que le serveur retourne une page Web au client, en réponse à une requête HTTP.

Les clients sont des programmes logiciels qui envoient des demandes à un serveur. Nous appelons client aussi bien l'ordinateur depuis lequel les demandes sont envoyées que le logiciel qui contient les instructions relatives à la formulation des demandes et la personne qui opère les demandes. L'ordinateur client est généralement un ordinateur personnel ordinaire, équipé de logiciels relatifs aux différents types de demandes qui vont être envoyées.

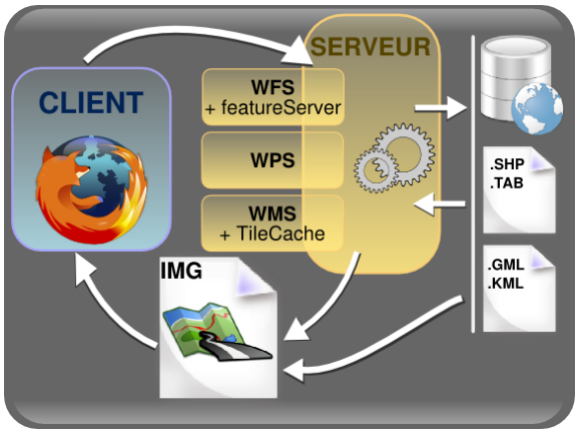


FIGURE 33 – Schéma simplifié d'une systéme GIS classique

4.8 Choix de la plate-forme GIS

En suivant l'architecture classique d'un GIS, nous avons choisi de réaliser une plate-forme qui suit le même raisonnement. Elle sera totalement constitué de solution open-sources, que nous présentons dans ce qui suit :

— Base de données géographique : PostgreSQL/PostGIS et Greenplum.

— Serveur web géographique : GeoServer

— Application cliente cartographique : MapStore

Nous allons détailler tout cela dans les sections à venir.

4.8.1 Choix de la base de données géographique

Les données géospatiales doivent être stockées et traitées dans une base de donnée adaptée à cet effet. Ce qui rend le choix de la base de donnée très important. Nous présentons dans ce qui suit le meilleur choix en tant que base de données géographique "PostGIS" .

4.8.1.1 PostgreSQL/PostGIS

Présentation PostGIS

PostgreSQL est un Système de Gestion de Base de Données (SBGD) [23]relationnelle libre disponible sous licence BSD. Ce système multi-plateforme est largement connu et réputé à travers le monde, notamment pour son comportement stable et pour leur respect aux normes de l'Institut de Normalisation Américaine (ANSI) SQL. Ce projet libre n'est pas géré par une entreprise mais par une communauté de développeurs.[32]
Parmi ses avantages :

- Fiabilité et stabilité légendaires
- Conçu pour une grande capacité
- Outils graphiques de modélisation et d'administration
- Très bonne compatibilité SQL
- Économies significatives sur les coûts de personnel
- Excellent support
- Déploiement illimité

Son avantage majeur pour nous est sa capacité sans égale de traitement des données géographiques grâce à l'extension PostGIS.

Présentation PostGIS

PostGIS est une extension de base de données spatiales pour la base de données relationnelle-objet PostgreSQL. Elle ajoute le support pour les objets géographiques permettant d'exécuter des requêtes de localisation en SQL. Les bases de données spatiales permettent le stockage de la géométrie de dossiers au sein d'une base de données et fournissent également une fonctionnalité pour interroger et retourner les dossiers utilisant ces géométries.
PostGIS présente de nombreux avantages tel qu'elle permet de : [33]

- Stocker et administrer de manière sécurisée les données à référence spatiale.
- effectuer des opérations spatiales (calcul de longueurs, de surfaces, unions et intersections de géométrie, etc...) grâce à ses fonctions spatiales avancées.
- assurer des connexions simultanées sur les données géographiques.
- visualiser les données géographiques grâce à un logiciel SIG comme QGIS.
- diffuser les données géographiques à travers des serveurs web cartographiques comme GeoServer.

4.8.1.2 GreenPlum

En cherchant une solution à la fois adaptée au traitement de données géographiques et au données massives (Big Data), Greenplum s'avère être le meilleur choix.
Greenplum était une société d'analyse de données volumineuses dont le siège est à San Mateo , en Californie . Elle a été acquise par EMC Corporation [24] en juillet 2010.
Depuis 2012, son logiciel de système de gestion de base de données est devenu la base de données

23. La licence BSD (Berkeley Software Distribution License) est une licence libre utilisée pour la distribution de logiciels. Elle permet de réutiliser tout ou une partie du logiciel sans restriction, qu'il soit intégré dans un logiciel libre ou propriétaire.

24. Une multinationale américaine dont le siège social est à Hopkinton, Massachusetts, États-Unis.

Pivotal Greenplum vendue par l'intermédiaire de Pivotal Software et est actuellement activement développée par la communauté Open Source [25] Greenplum Database et Pivotal.[34]
Le produit de base de données Greenplum de Pivotal est une plate-forme de données open source avancée et complète qui est basé sur la technologie Open-Source PostgreSQL. Il s'agit essentiellement de plusieurs instances de base de données orientées disque PostgreSQL agissant ensemble comme un système de gestion de base de données cohésif (SGBD).

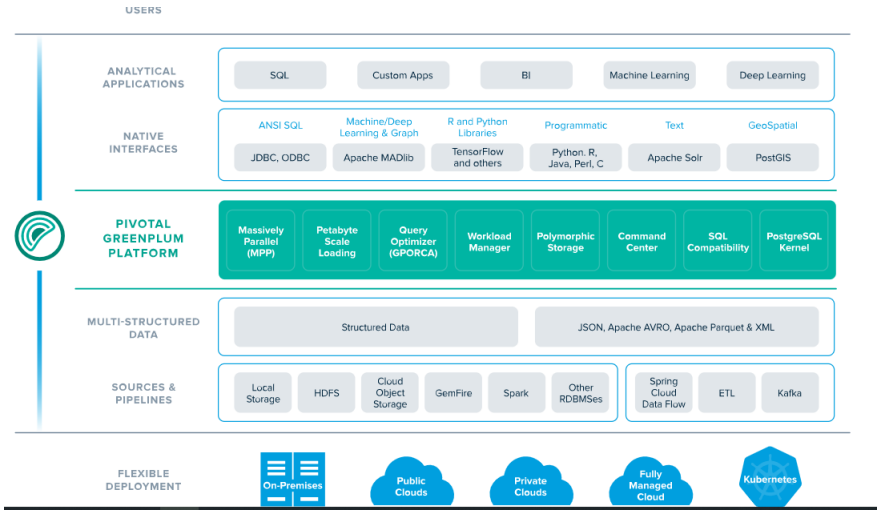


FIGURE 34 – Architecture de GreenPlum

Elle utilise des techniques de traitement massivement parallèle (MPP). MPP désigne les systèmes avec deux processeurs ou plus qui coopèrent pour effectuer une opération, chaque processeur avec sa propre mémoire, système d'exploitation et disques. Pivotal Greenplum s'adosse à une architecture de système haute performance dite "share-nothing" qui automatise le traitement en parallèle des données et des requêtes.
Au final, l'entrepôt de données Pivotal Greenplum propose un haut de niveau de performance en matière de requêtage et de rendement. Une technologie adaptée pour l'analyse Big Data.

Les données stockées dans ces bases de données géographiques, ainsi que dans d'autres sources sont distribuées à travers les GIS grâce aux serveurs web cartographiques.

4.8.2 Choix du serveur web géographique

Parmi les serveurs web cartographiques open-source qui offrent des services de cartographie, nous retrouvons MapServer et GeoServer.

4.8.2.1 MapServer

MapServer est un moteur de cartographie Web : il s'installe sur un serveur Web et se connecte aux sources de données géospatiales afin de servir des cartes aux applications client via le Web. Il supporte entre autres les normes WMS, WFS et WCS de services Web de l'OGC, et est reconnu en

25. Code source ouvert s'applique aux logiciels qui ont des possibilités de libre redistribution, d'accès au code source et de création de travaux dérivés.

tant que chef de file à l'échelle mondiale pour sa performance, sa robustesse, et la qualité du rendu cartographique qu'il produit. Il rivalise à ces niveaux avec les solutions propriétaires les grands éditeurs de logiciels du milieu de la géomatique.[35]

4.8.2.2 Geoserver

GeoServer est un serveur informatique open source et libre écrit en Java basé sur la librairie GeoTools qui permet aux utilisateurs de partager et modifier des données géographiques. Conçu pour l'interopérabilité, il publie les données de toutes les sources principales de données spatiales utilisant des normes ouvertes.[36]
C'est un serveur cartographique qui permet la gestion des informations géographiques. Il implémente de nombreuses spécifications service web de l'OGC (WMS, WFS-T, WCS, SLD...) et est, de plus, une application de référence pour l'OGC pour le support de la norme WFS. Geoserver gère de nombreux formats de sortie pour les données (PNG, SVG, KML, JPEG, PDF, GeoJSON...).

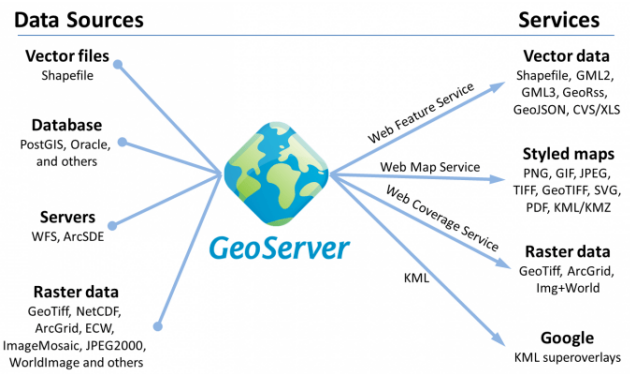


FIGURE 35 – Architecture de Geoserver

GeoServer offre bien plus de services que MapStore et s'est beaucoup mieux développé dernièrement, le choix de GeoServer se fait de lui même.

4.8.3 Choix de l'application cliente cartographique

Ayant choisi de réaliser une application web cartographique pour notre GIS, deux solutions généralistes s'ouvraient à nous : réaliser un tableau de bord ou un Géoportail dédié à notre GIS. MapStore est l'une des rares applications open-source permettant la construction de solution web cartographique.

4.8.3.1 MapStore

MapStore est un framework open source pour la construction d'applications web de cartographie utilisant des bibliothèques de cartographie standard, telles que OpenLayers et Leaflet. En tant que produit géoportail standard, c'est un produit Web pour la visualisation et l'analyse de cartes qui permet de construire un site Web interactif géospatial ou un service Web. Les avantages de son utilisation sont multiples :[37]

— Fournit un accès direct et en temps réel aux entrepôts de données géospatiales de tous les formats géospatiaux pris en charge.

— Fournit également toutes les fonctionnalités d'analyse spatiale.

— Permet la construction d'application géospatiale puissante, dynamique et ouverte, précédemment disponible uniquement dans une application de bureau.

— Ne dépend pas explicitement de n'importe quel moteur de cartographie, mais il peut fonctionner avec OpenLayers, LeafletJS et Cesium 3D viewer assurant la plus grande flexibilité quand on veut l'utiliser comme un framework.

4.9 Représentation de l'architecture finale de la plate-forme GIS-IOT

Au final nous pouvons résumer notre architecture IoT-GIS dans le schéma suivant :

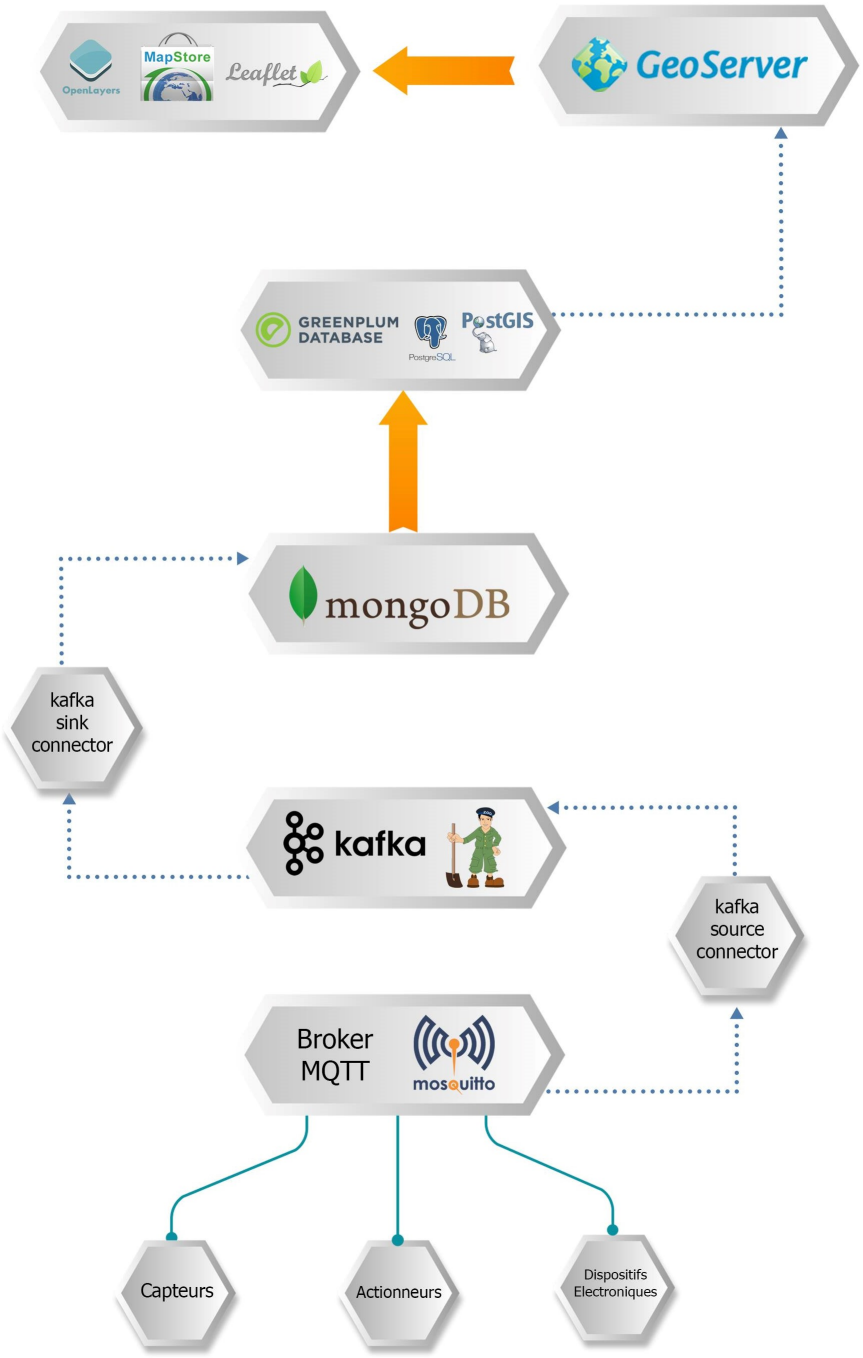


FIGURE 36 – Schéma de l'architecture IoT-GIS

4.10 Conclusion

Dans ce chapitre nous avons abordé les aspects majeurs d'une plate-forme GIS, à savoir, son fonctionnement, les normes associées aux données et services géographiques ainsi les principaux choix pour la réalisation de cette plate-forme. Dans ce qui suit, nous allons procéder à l'implémentation du systéme IoT-GIS dans sa globalité.

Chapitre 5

5 Implementation

5.1 Introduction

Dans cette partie, nous allons expliciter le cheminement de notre travail qui va de la récoltes des données capteurs IoT jusqu'à l'application cliente dédiée aux données géolocalisées.

5.2 Platforme OpenStack

Notre travail est déployé sur le Cloud [26] de l'Ecole Nationale Polytechnique d'Alger (ENP) grâce aux équipements dispensé par l'établissement (serveurs, VPN). Ces appareils fournissent de hauts niveaux de performances pour le déploiement d'architectures à grande échelle.
Afin d'implémenter notre architecture sur cloud, nous avons tout d'abord installé la platforme adéquate, OpenStack, qui est un système d'exploitation cloud. Il gère les ressources de calculs et de stockage ainsi que la configuration réseaux du centre de données.
Openstack est muni d'un tableau de bord qui donne aux administrateurs le contrôle et permet aux utilisateurs de s'approvisionner en ressources via une interface web.

26. Un cloud (« nuage ») est un ensemble de matériels, de raccordements réseau et de logiciels fournissant des services qu'individus et collectivités peuvent exploiter depuis n'importe où dans le monde.

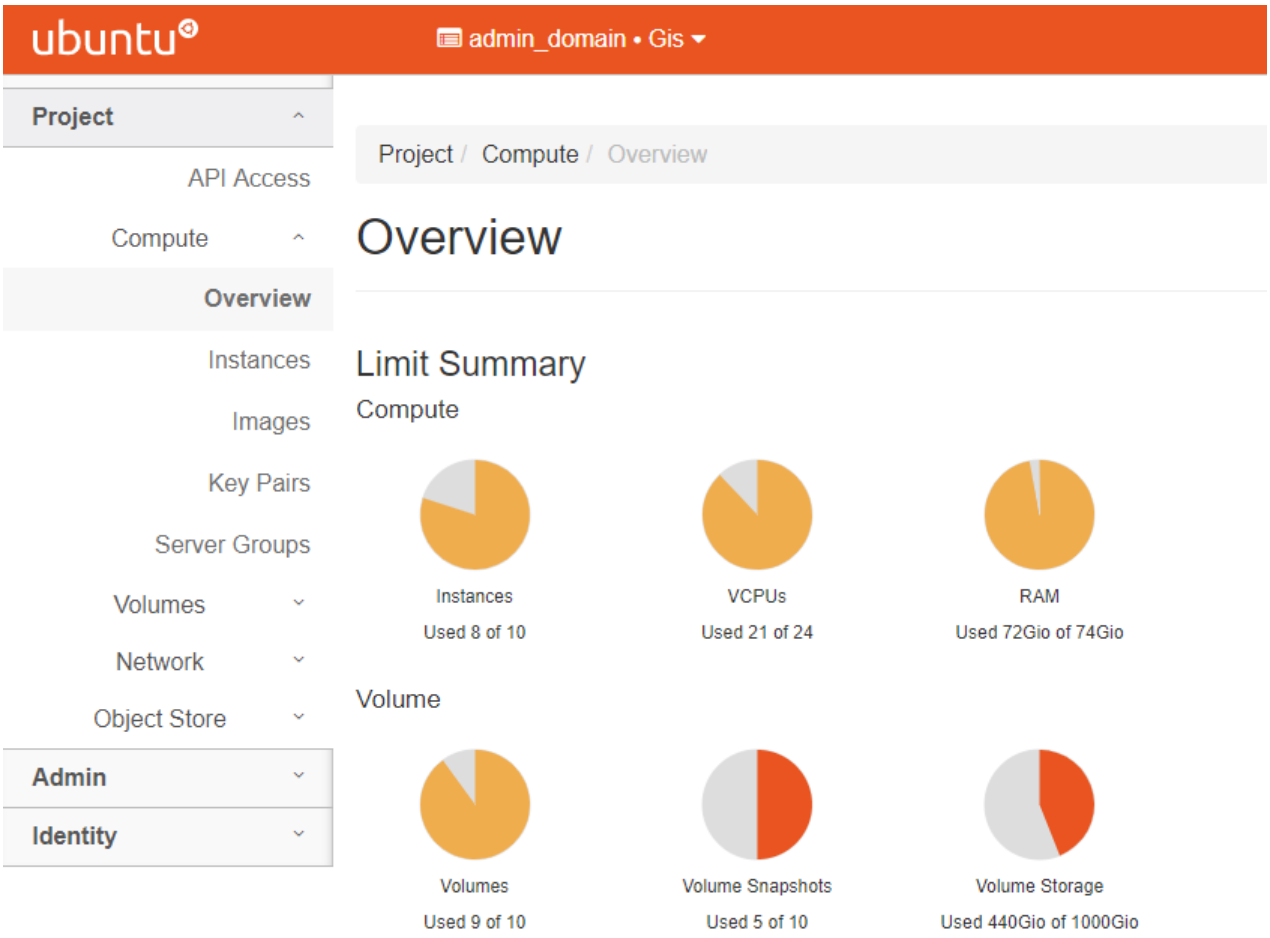


FIGURE 37 – Tableau de bord OpenStack

Au-delà de la fonctionnalité standard de l'infrastructure en tant que service, des composants supplémentaires assurent l'orchestration, la gestion des pannes et la gestion des services pour assurer une disponibilité élevée des applications utilisateur.

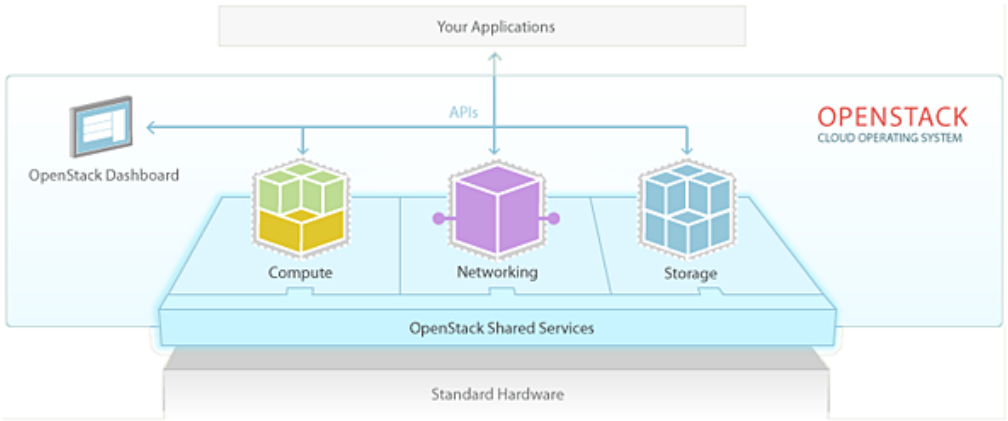


FIGURE 38 – Architecture de OpenStack

5.3 Architecture d'implementation finale

Notre architecture suit ce cheminement :

1. Les données récoltés de nos capteurs (simulés) sont envoyé à travers notre brocker MQTT (mosquitto)
2. Le brocker MQTT qui se charge de la collecte de données en provenance de capteur IoT va faire en sorte de transférer les données vers la plate-forme de streaming Apache Kafka.
3. De Kafka, la partie maîtresse de la plate-forme IoT, les données sont prises en charge et envoyées vers la base de données Mongodb afin de les stocker et les utiliser prochainement.
4. Dans le but de pouvoir exploiter les données, nous les déplaçons vers notre entrepôt de données Greenplum. On procède à un traitement spatial des données pour qu'elles trouvent leur intéret en tant que données géolocalisées.
5. A l'aide de geoserver, on distribue nos données capteurs géo-localisés ainsi que d'autre données spatiales.
6. Nous développons une application cliente, un géoportail, afin d'illustrer un exemple d'utilisation et le rendu de notre système.

Dans le diagramme suivant, notre architecture d'implémentation compléte :

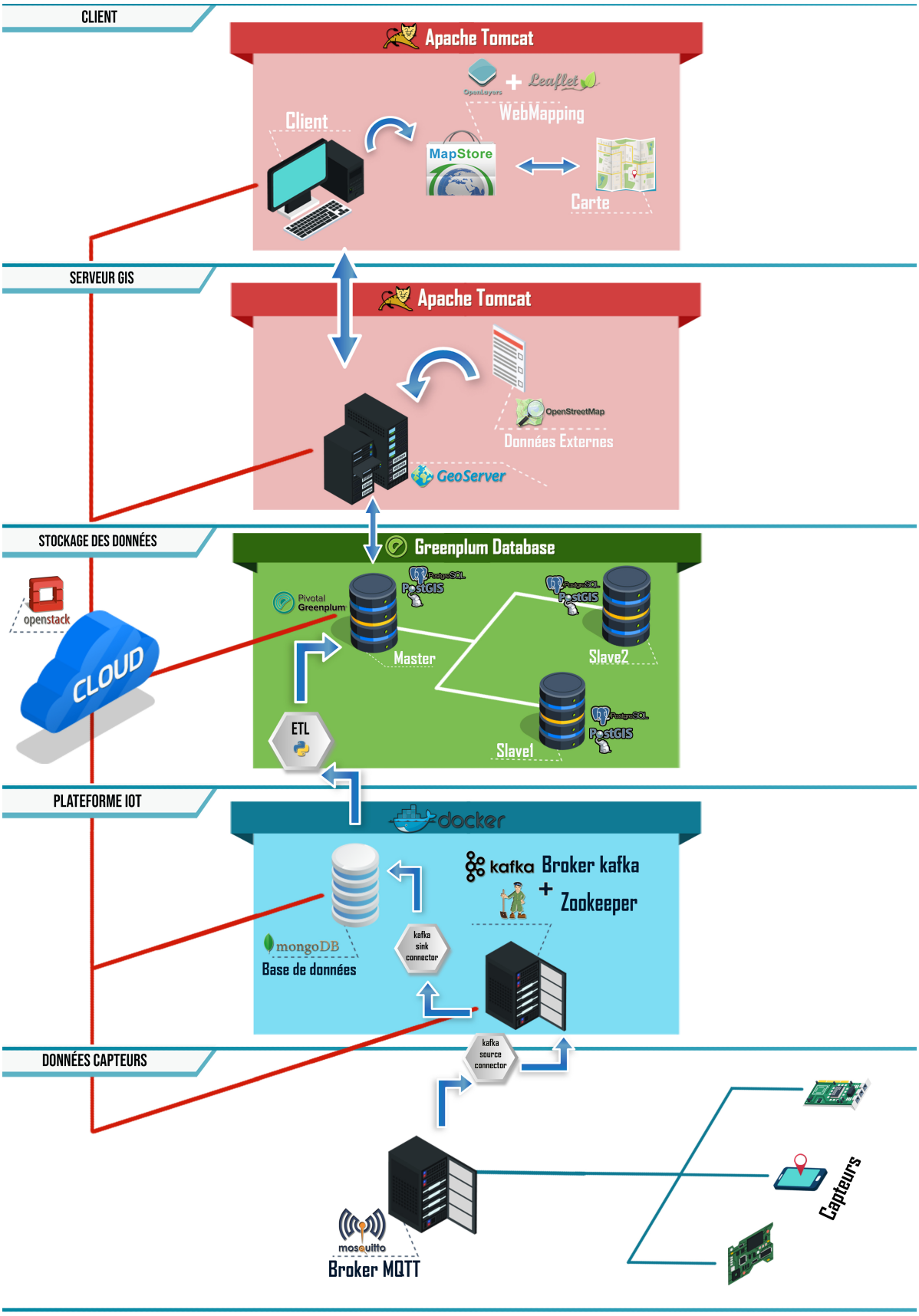


FIGURE 39 – Architecture de l'implémentation finale

5.4 Noeud Capteur et plate-forme IoT

Premièrement, on commence par récolter les données des capteurs vers la plate-forme IoT. Mais avant d'entamer cette partie, nous parlerons d'abord de la notion de conteneurisation utilisé dans cette implémentation.

5.4.1 Conteneurisation

Dans le développement de notre plate-forme, nous avons opté pour le déploiement de nos éléments constituant dans des conteneurs.

Un conteneur est un ensemble de processus logiciels léger et indépendant, regroupant tous les fichiers nécessaires à l'exécution des processus : code, outils système, bibliothèque et paramètres. Ils peuvent être utilisés pour exécuter des applications Linux ou Windows. Les containers se partagent le même noyau de système d'exploitation et isolent les processus de l'application du reste du système.[38]

Docker est une plateforme logicielle open source permettant de créer, de déployer et de gérer des containers d'applications virtualisées sur un système d'exploitation.[38] Celle-ci permet en effet d'assembler et de gérer efficacement une application ainsi que toutes ses dépendances dans un paquet unique qui peut être placé dans un conteneur, lui-même placé sur n'importe quel serveur Linux. Le docker apporte cette capacité de probabilité et de flexibilité au niveau de l'application.

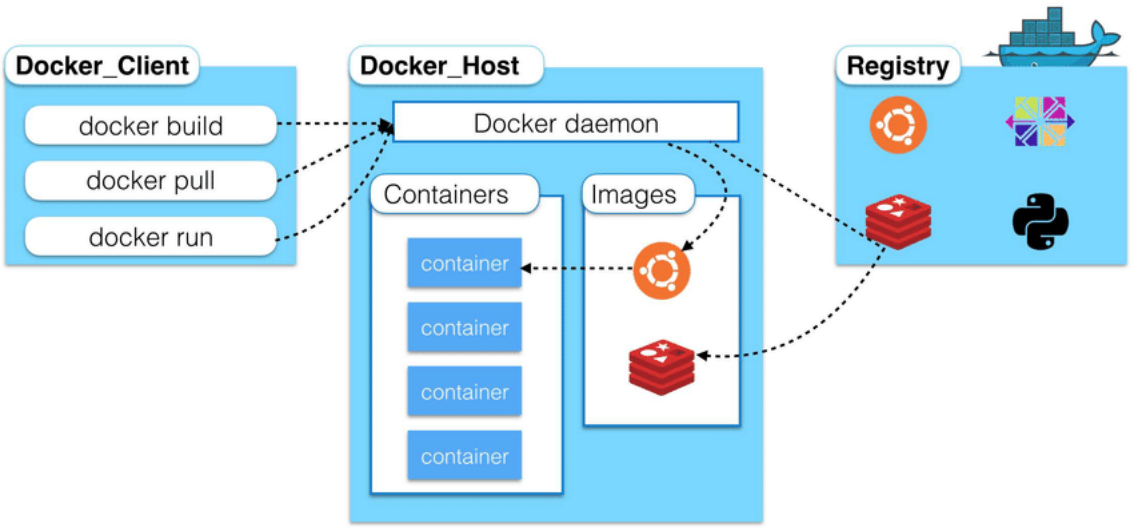


FIGURE 40 – Architecture d'un système docker

5.4.2 Implémentation de la solution

Maintenant que la notion de conteneurisation est expliquer nous pouvons procéder à l'explication de notre solution.

Nous avons choisi d'implémenter notre plateforme IoT dans des conteneurs docker.Notre plateforme se compose du broker kafka connecté à la base de donnée mongodb.

Les données capteurs quand à eux avant d'être transmise à la plateforme IoT doivent d'abord passer par une passerelle IoT.Cette passerelle n'est autre que le broker MQTT Mosquitto.

5.4.2.1 Broker MQTT Mosquitto

Mosquitto est le broker MQTT que nous avons choisi d'utiliser. Nous l'avons implémenté en installant le paquet mosquitto sur une raspberry pi et en y créant des topics auxquels les consommateurs peuvent s'abonner et les éditeurs peuvent y publier des données. Mais, pour tester notre architecture nous avons créer un conteneur mosquitto en utilisant l'image eclipse-eclipse-mosquitto :1.5.5 du docker hub.

```
 mosquitto :
image : eclipse-mosquitto :1.5.5
hostname : mosquitto
container_name : mosquitto
expose :
  - "1883"
```

5.4.2.2 Broker Apache Kafka

Le Broker kafka est l'un des composants phare de notre plateforme IoT. Nous avons choisi de l'implémenter en créant un conteneur docker à l'aide l'image confluentinc/cp-kafka :5.1.0 du docker hub.

```
 kafka :
image : confluentinc/cp-kafka :5.1.0
hostname : kafka
container_name : kafka
ports :
  - "9092 :9092"
```

5.4.2.3 Base de données cache Mongodb

La base de données cache Mongodb est l'autre composant essentiel de notre plateforme IoT.Dans le but de l'implémenter nous avons opter pour la conteneurisation et cela à travers l'image mongo :4.0.5 du docker hub.

```
 mongo-db :
image : mongo :4.0.5
hostname : mongo-db
container_name : mongo-db
expose :
  - "27017"
```

5.4.2.4 Connexion et validation

Le développement de notre solution ne sera pas uniquement composé de trois conteneurs (mosquitto, kafka et mongodb), il nous faudra rajouter un conteneur de connexion appelons le kafka-connect pour les relier.

Il faut savoir que pour le conteneur de connexion, nous aurons besoin d'un connecteur kafka-MQTT et d'un connecteur Kafka-mongodb. Le connecteur MQTT-Kafka va représenter un connecteur Source et celui de Kafka-Mongodb un connecteur Sink.
Comme ces connecteurs ne viennent pas par défaut avec le conteneur créer avec l'image Kafka-connect, nous les téléchargeons du site officiel des connecteurs de kafka https ://www.confluent.io/connector/
Une fois reçu, nous devons déplacer les fichiers contenus dans ces dossiers et dotés de l'extension .jar vers un dossier de notre choix que nous nommons : /tmp/custom/jars.

Nous utiliserons aussi deux autres conteneurs aux noms de zookeeper et mongoclient. Zookeeper est chargé de la coordination de kafka-brocker et mongoclient déploie une interface graphique pour la gestion de mongodb.

Nous obtiendrons au final six conteneurs, les six étant :

— Mosquitto : Représente le brocker mqtt ;
— zookeeper : Pour la coordination de plusieurs brocker Kafka ;
— kafka : Représente le brocker Kafka ;
— kafka-connect : Pour les différentes connexions à kafka ;
— mongo-db : Représente notre base de données ;
— mongoclient : Déploie une interface graphique pour la gestion de la base de données sur le web à l'adresse : http ://192.168.210.105 :3000/

La connexion entre les six conteneurs est illustrée par le schéma ci-dessus :

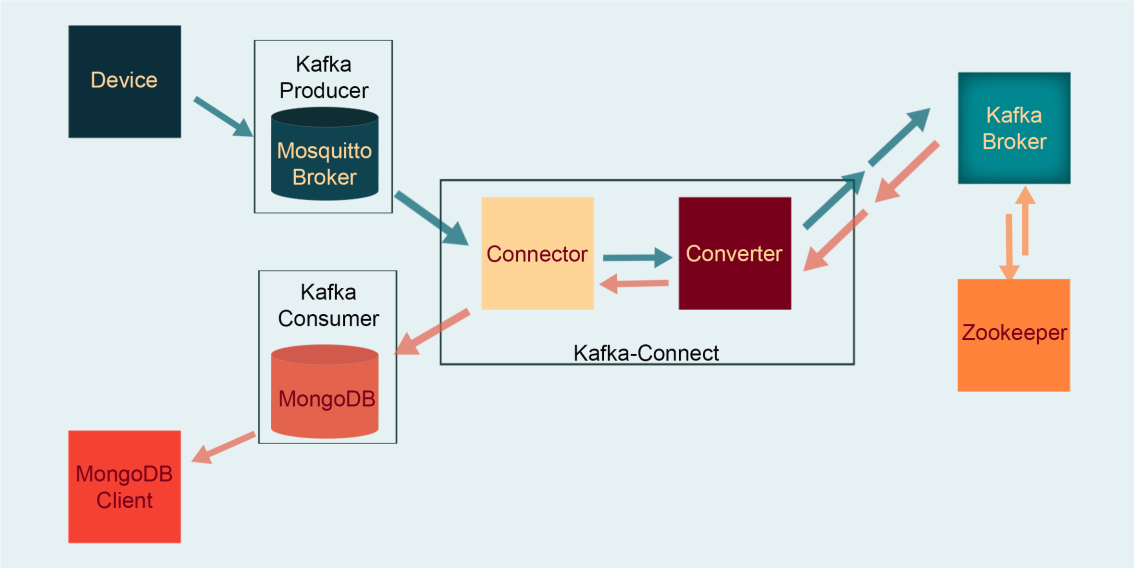


FIGURE 41 – Système de traitement de données du capteur vers la base de donnée

Nous allons par la suite spécifier ce chemin d'accès dans le fichier de configuration général qui contient les détails sur la composition de chaque conteneur. C'est le fichier docker-compose.yml qui va servir à la mise en place de notre système.

Le fichier docker-compose.yml qui permet de créer les six conteneurs se trouve dans l'annexe A.

Afin de lancer le script, sur l'invité de commande, nous nous plaçons dans le répertoire ou se trouve le fichier docker-compose.yml puis nous saisissons les commandes suivantes :

```
sudo docker-compose up -d
```

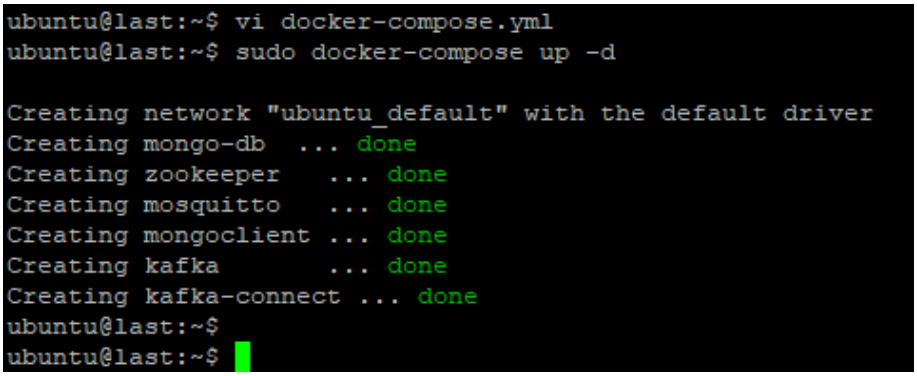


FIGURE 42 – Docker-compose up

Nous pouvons vérifier ensuite l'état de nos conteneurs avec :

```
sudo docker-compose ps
```

Nous voyons que tous les conteneurs sont en mode Up c'est à dire, fonctionnels.

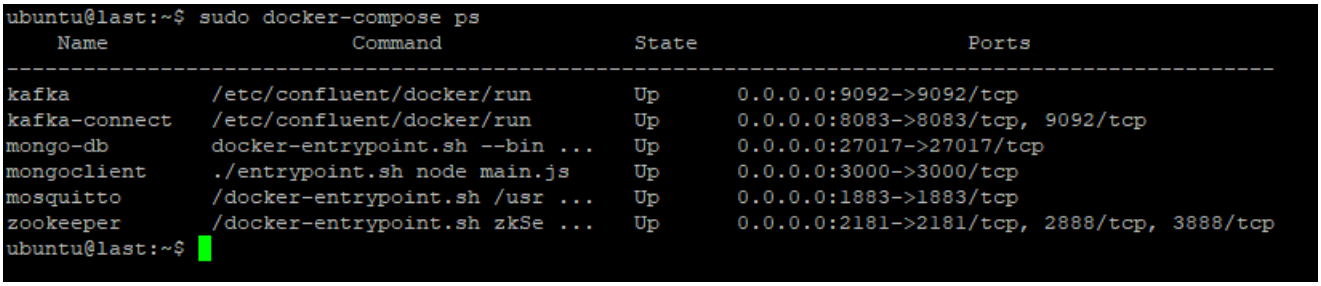


FIGURE 43 – Docker-compose ps

Ou, pour plus de détails, comme voir l'ID (identifiant unique) de chaque conteneur :

```
sudo docker ps
```

Grâce à cet identifiant, nous pouvons accéder à l'intérieur de chaque conteneur et y apporter les modifications que nous souhaitons, si l'envie ou l'obligation se pose.

5.4.3 Test de l'architecture

1. Afin de tester notre implémentation, nous simulons dix capteurs de température dans la wilaya d'Alger. Nous enregistrons dans un fichier au format JSON afin de nous rapprocher du cas réel de la récolte de données.

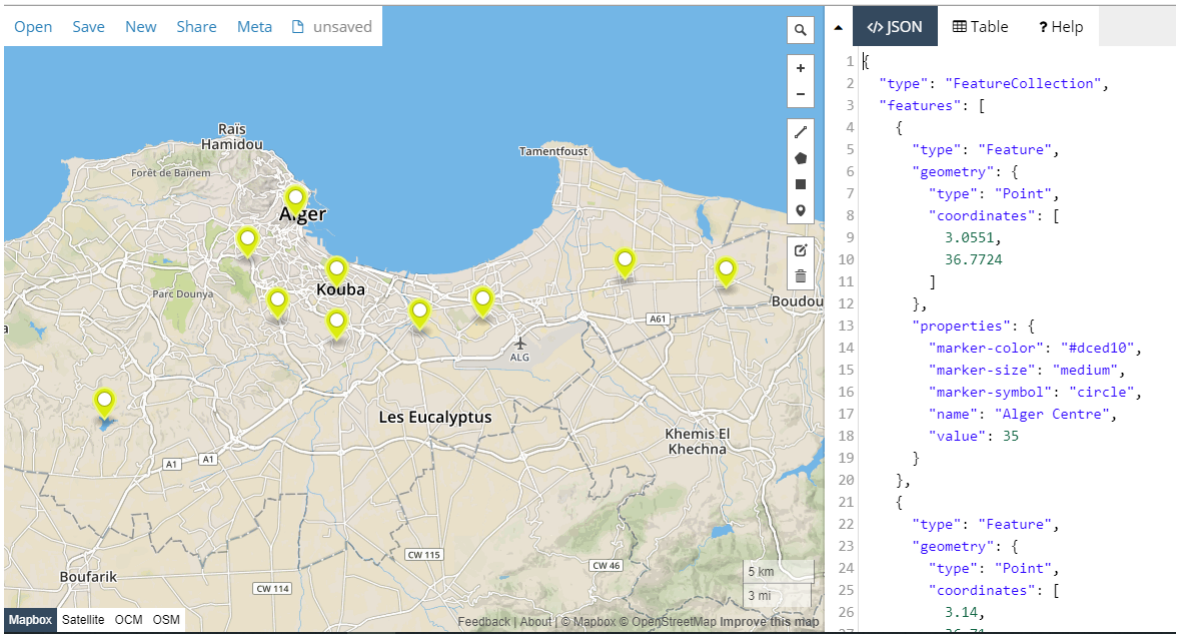


FIGURE 44 – Données au format JSON

Le format d'un capteur dans le fichier test est comme ce qui suit :

```
{
    "type" : "Feature",
    "geometry" : {
        "type" : "Point",
        "coordinates" : [
          3.0551,
          36.7724
        ]
    },
    "properties" : {
        "marker-color" : "dced10",
        "marker-size" : "medium",
        "marker-symbol" : "circle",
        "name" : "Alger Centre",
        "value" : 35
    }
},
```

Chaque capteur est caractérisé par sa position, longitude et latitude (coordinates) ; le nom

de la ville (name) ; et la valeur de la température en °c (value).
Le fichier complet est joint dans l'annexe A.

Par la suit, il nous faut configurer les connecteur **source mqtt** et **sink mongodb** contenus dans kafka-connect.

2. Nous configurons le **connecteur source mqtt** qui écoute sur le port 8083 à l'adresse 192.168.210.105 du conteneur Kafka-connect :

```
curl -X POST 'http ://192.168.210.105 :8083/connectors' -H 'Content-Type :
application/json' -d '{
   "name" : "mqtt-connect-kafka",
   "config" : {
     "connector.class" : "io.confluent.connect.mqtt.
     MqttSourceConnector",
     "tasks.max" : 1,
     "mqtt.server.uri" : "tcp ://mosquitto :1883",
     "mqtt.topics" : "mqtt-topic",
     "kafka.topic" : "kafka",
     "value.converter" :
     "org.apache.kafka.connect.converters.ByteArrayConverter",
     "confluent.topic.bootstrap.servers" : "kafka :9092",
     "confluent.topic.replication.factor" : 1
   }
}'
```

Nous y précisons l'adresse de la machine mqtt et kafka ainsi que les différents topics où nous comptons nous inscrire.

3. Nous ouvrons une fenêtre mqtt_subscriber et une autre kafka_consumer en écoute pour la réception du message mqtt. Il faudra s'inscrire dans les topics respectifs, tout en précisant le nom du réseaux ubuntu_default :

```
sudo docker run -it --rm --name mqtt-subscriber --network ubuntu_default
efrecon/mqtt-client sub -h mosquitto -t "mqtt-topic"
```

```
sudo docker run --network ubuntu_default --rm confluentinc/
cp-kafka :5.1.0 kafka-console-consumer --bootstrap-server kafka :9092 --topic
kafka-topic --from-beginning
```

4. Nous envoyons notre fichier test à l'aide de mqtt_publisher en nous inscrivons dans le topic mqtt-topic.

```
sudo docker run -it --rm --name mqtt-publisher --network ubuntu_default
efrecon/mqtt-client pub -h mosquitto -t "mqtt-topic" -m "message.json"
```

5. Une fois le message reçu dans kafka_consumer, nous avons la preuve que la connexion fonctionne. Les captures suivantes montrent les données reçues dans kafka-consumer et dans

mqtt-subscriber.

6. Il reste à configurer le connecteur **sink mongodb** afin de recevoir les données directement sur notre base de stockage.

```
curl -X POST 'http ://192.168.210.105 :8083/connectors' -H 'Content-Type :
application/json' -d '{
    "name" : "mongodb-kafka-mqtt",
    "config" : {
      "connector.class" :
     "at.grahsl.kafka.connect.mongodb.MongoDbSinkConnector",
      "tasks.max" : 1,
      "topics" : "kafka",
      "mongodb.connection.uri" :
     "mongodb ://mongo-db/data ?retryWrites=true",
      "mongodb.collection" : "DataCollection",
      "key.converter" :
      "org.apache.kafka.connect.json.JsonConverter",
      "key.converter.schemas.enable" : false,
      "value.converter" :
      "org.apache.kafka.connect.json.JsonConverter",
      "value.converter.schemas.enable" : false
    }
}'
```

De cette manière, les informations contenues dans le topic "kafka" sont directement placé dans notre base de données, Mongodb.

Grace au conteneur client mongodb, nous pouvons visualiser notre document stocké sur mongodb :

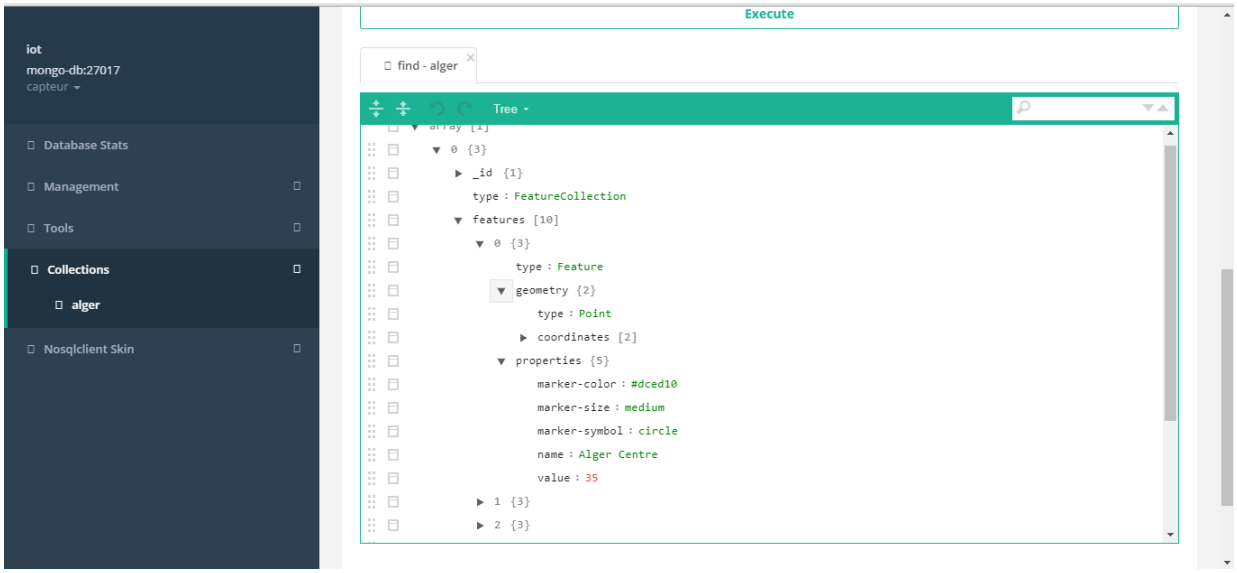


FIGURE 45 – Client mongodb

Les données capteurs étant sur notre base de données mongodb, elles sont prêtes à être utilisées par d'autre systèmes, dans notre cas, servis par un serveur cartographique en tant que données géolocalisées, ce que nous verrons par la suite.

5.5 Big Data et serveur GIS

5.5.1 Solution geoserver et base de données

Maintenant que les données capteurs sont sur la base de stockage de notre plateforme IoT, la question qui se pose est : Est il possible de publier ces données directement grace au serveur cartographique, ou bien il reste un traitement sur ces données avant de pouvoir les servir ?

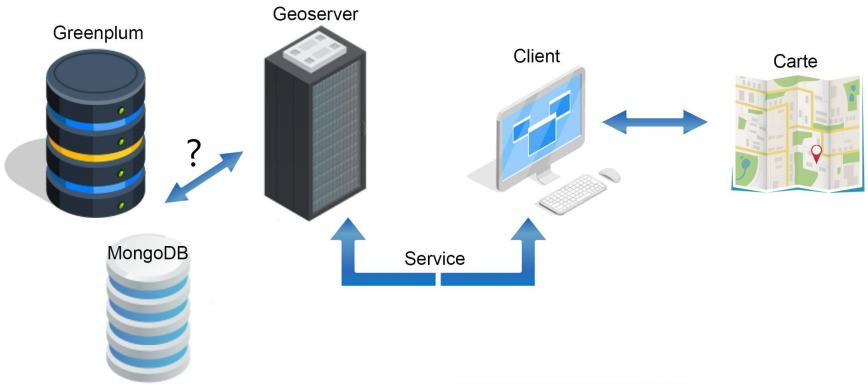


FIGURE 46 – Quelle solution de stockage ?

La réponse est, effectivement, ces données étant sans support géogra-phique, le fait de les diriger directement vers le serveur cartographique ne donnera aucun résultat. Parce que, et avant toute

chose, un serveur cartographique offre des services sur des données "géographiques". De ce fait, l'entrepôt de données GreenPlum trouve tout son intérêt. Cette base offre la possibilité de bénéficier des avantages de PostGIS, qui est, comme nous l'avons introduit précédemment, la meilleure base de données spatiales, mais en plus, elle va nous permettre de gagner en puissance et extensibilité, étant à la fois une base de données relationnelle et Big Data.

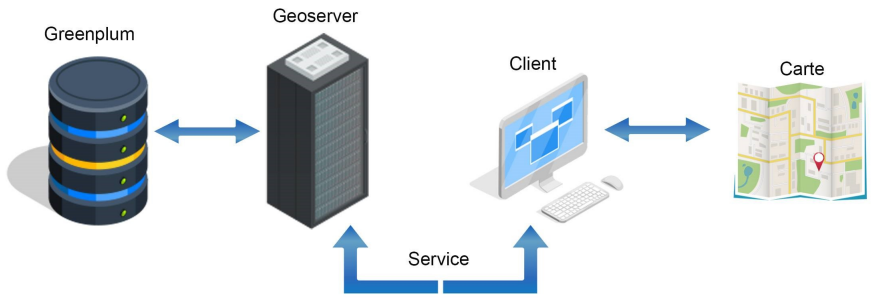


FIGURE 47 – Stockage de données géospatiales

Nous choisissons alors d'utiliser mongodb, la base de stockage de notre plate-forme IoT, comme base de données cache, c'est à dire qu'elle sera réinitialisé périodiquement, aprés avoir envoyé les données stockés dessus vers Greenplum, qui sera notre entrepôt de données finale, et où les données subirons un traitement géospatial.

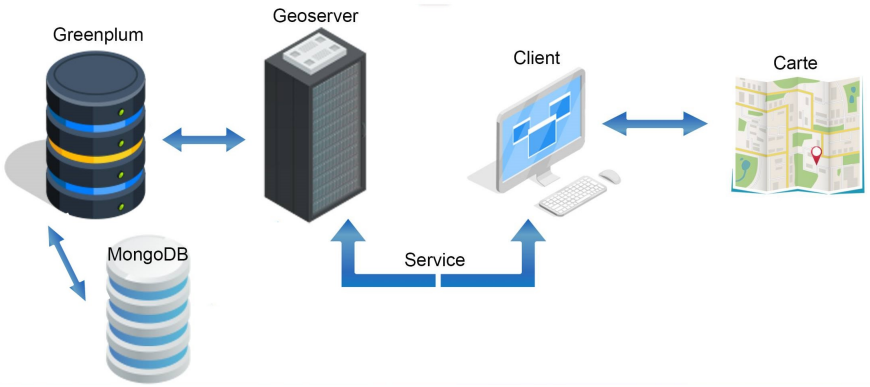


FIGURE 48 – Schéma du chemin de données IoT vers Geoserver

5.5.2 Entrepôt de données Greenplum

Suite aux ressources qui nous sont offertes pas les serveurs de notre école, nous avons choisis d'installer la base de données Greenplum avec une architecture d'un master (maitre) et de deux slaves (esclaves). La méthode de l'installation se trouve en annexe.

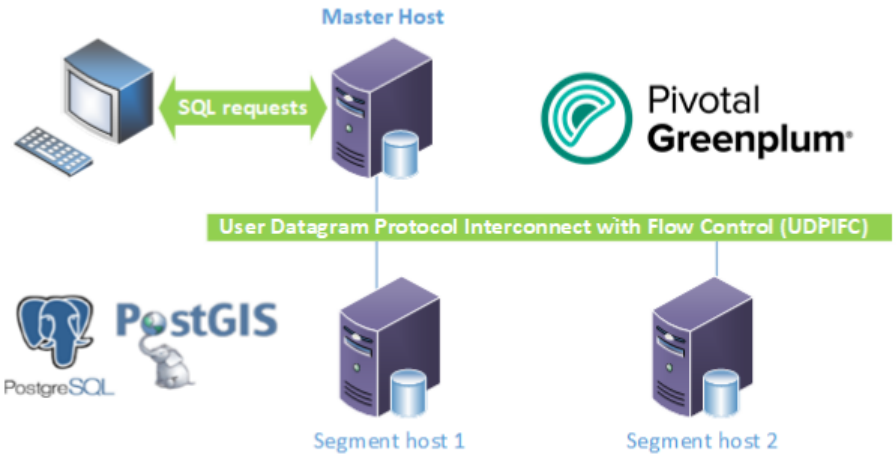


FIGURE 49 – Architecture de notre base de stockage greenplum

Nous transférons les données de mongodb vers l'entrepôt de données Greenplum. Pour cela, nous avons commencé par créer une table pour recevoir les données capteurs sur greenplum :

```
 CREATE TABLE mongodb.features (
id VARCHAR(100) REFERENCES mongodb.alger,
type VARCHAR(100),
geometry_type VARCHAR(100),
geometry_coord1 VARCHAR(100),
geometry_coord2 VARCHAR(100),
properties_marker_color VARCHAR(255),
properties_marker_size VARCHAR(100),
properties_marker_symbol VARCHAR(100),
properties_name VARCHAR(255),
       properties_value VARCHAR(255)
) ;
```

Par la suite, comme toutes les APIs chargées du transfert de données mongodb-greenplum étaient payantes, nous avons écrit un script ETL (Extract, Transform, Load) avec le langage de programmation python qui extrait la donnée de mongodb, la traite et puis l'écrit sur greenplum. L'intégralité du script se trouve dans annexe A.

```
Total Record for the collection: 1
first step okey.
CSV file successfully created: ./alger.csv
CSV file successfully created: ./features.csv
Connecting to Database
Truncated mongodb.alger
Loaded data into mongodb.alger
DB connection closed.
Connecting to Database
Truncated mongodb.features
Loaded data into mongodb.features
DB connection closed.
```

FIGURE 50 – Envoie de mongodb vers greenplum

Une fois les données reçu, il faudra avoir recours à certains traitement afin de les rendre exploitable par un serveur GIS, étant donné qu'à ce point, elles ne possèdent aucun support pour des opérations spatiales.

```
test_db=# SELECT * FROM mongodb.features
;
            id            |  type   | geometry_type | geometry_coord1 | geometry_coord2
properties_name | properties_value
--------------------------+---------+---------------+-----------------+-----------------
----------------+-----------------
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.0551          | 36.7724
Alger Centre    | 35
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.14            | 36.71
El Harrach      | 36
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.1833          | 36.7167
Bab ezzouar     | 35
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.28079         | 36.73829
Rouiba          | 36
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.35            | 36.7333
Reghaia         | 37
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.0833333       | 36.7333333
Ruisseau        | 35
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.0827          | 36.7045
Ain Naadja      | 37
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.0426          | 36.7163
bir khadem      | 36
 5cf79e66e569999a61bf66ef | Feature | Point         | 3.0222          | 36.75
Benaknoun       | 38
 5cf79e66e569999a61bf66ef | Feature | Point         | 2.9238          | 36.6613
Douera          | 38
(10 rows)
```

FIGURE 51 – Tableau de données sur greenplum

La suite de traitement sur les données est decrite ci-dessous :

1. Créer une colonne géométrie sur le tableau de données :

```
SELECT AddGeometryColumn( 'mongodb','features' , 'geom', 4326, 'POINT' , 2) ;
```

```
test_db=# SELECT AddGeometryColumn( 'mongodb','features' , 'geom', 4326, 'POINT' , 2);
                addgeometrycolumn
-------------------------------------------------
 mongodb.features.geom SRID:4326 TYPE:POINT DIMS:2
(1 row)
```

FIGURE 52 – AddGeometryColumn

2. Créer un Indexe spatial sur une colonne géométrie :

```
CREATE INDEX index_gis ON mongodb.features USING GIST (geom) ;
```

```
test_db=# CREATE INDEX index_gis ON mongodb.features USING GIST (geom);
CREATE INDEX
```

FIGURE 53 – CREATE INDEX

3. Créer une règle qui met à jour le tableau à chaque nouvelle insersion et transforme la colonne coordinates en type géométrique et le met dans la colonne geom :

```
CREATE RULE test as on INSERT to mongodb.features do also
UPDATE mongodb.features
SET geom= (CASE WHEN (geometry_coord1 IS NOT NULL)
THEN ST_PointFromText('POINT('||geometry_coord1||'
'||geometry_coord2||')',4326)
ELSE geom end) ;
```

```
test_db=# CREATE RULE test as on INSERT to mongodb.features do also
test_db-# UPDATE mongodb.features
test_db-# SET geom= (CASE WHEN (geometry_coord1 IS NOT NULL)
test_db(# THEN ST_PointFromText('POINT('||geometry_coord1||' '||geometry_coord2||')',4326)
test_db(# ELSE geom end);
CREATE RULE
test_db=#
```

FIGURE 54 – CREATE RULE

```
ies_name,properties_value,geom FROM mongodb.features

d2 | properties_name | properties_value |                        geom
---+-----------------+------------------+-----------------------------------------------------
   | Alger Centre    | 35               | 0101000020E6100000A913D044D870084017B7D100DE624240
   | El Harrach      | 36               | 0101000020E61000001F85EB51B81E09407B14AE47E15A4240
   | Bab ezzouar     | 35               | 0101000020E6100000BADA8AFD6577094088855AD3BC5B4240
   | Rouiba          | 36               | 0101000020E6100000C959D8D30E3F0A405F7B6649805E4240
   | Reghaia         | 37               | 0101000020E6100000CDCCCCCCCCCC0A4012143FC6DC5D4240
   | Ruisseau        | 35               | 0101000020E6100000915731A6AAAA0840AC4896DDDD5D4240
   | Ain Naadja      | 37               | 0101000020E61000009A081B9E5EA908401904560E2D5A4240
   | bir khadem      | 36               | 0101000020E6100000107A36AB3E570840F90FE9B7AF5B4240
   | Benaknoun       | 38               | 0101000020E610000005C58F31772D08400000000000604240
   | Douera          | 38               | 0101000020E6100000C05B2041E1630740226C787AA5544240
   | Draria          | 32               | 0101000020E610000000000000000084000000000000404240
```


FIGURE 55 – Tableau de données avec support spatial sur greenplum

```
godb.features

  |                          geom
--+----------------------------------------------------
  | 0101000020E6100000A913D044D870084017B7D100DE624240
  | 0101000020E61000001F85EB51B81E09407B14AE47E15A4240
  | 0101000020E6100000BADA8AFD6577094088855AD3BC5B4240
  | 0101000020E6100000C959D8D30E3F0A405F7B6649805E4240
  | 0101000020E6100000CDCCCCCCCCCC0A4012143FC6DC5D4240
  | 0101000020E6100000915731A6AAAA0840AC4896DDDD5D4240
  | 0101000020E61000009A081B9E5EA908401904560E2D5A4240
  | 0101000020E6100000107A36AB3E570840F90FE9B7AF5B4240
  | 0101000020E610000005C58F31772D08400000000000604240
  | 0101000020E6100000C05B2041E1630740226C787AA5544240
  | 0101000020E610000000000000000084000000000000404240
```


FIGURE 56 – La colonne géometrie

Ainsi les données sont prêtent à être exploité et publié par notre serveur GIS Geoserver.

5.5.3 Publication des données sur GeoServer

Nous entamons cette étape en créant une connexion entre geoserver et greenplum.

1. Nous ouvrons un nouvel espace de travail :

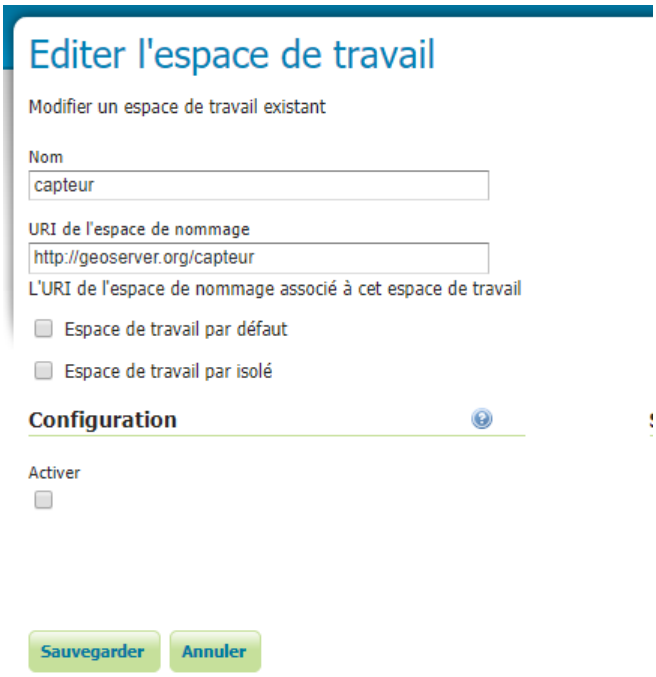


FIGURE 57 – Espace de travail geoserver

2. Puis, un nouvel entrepôt PostGIS :

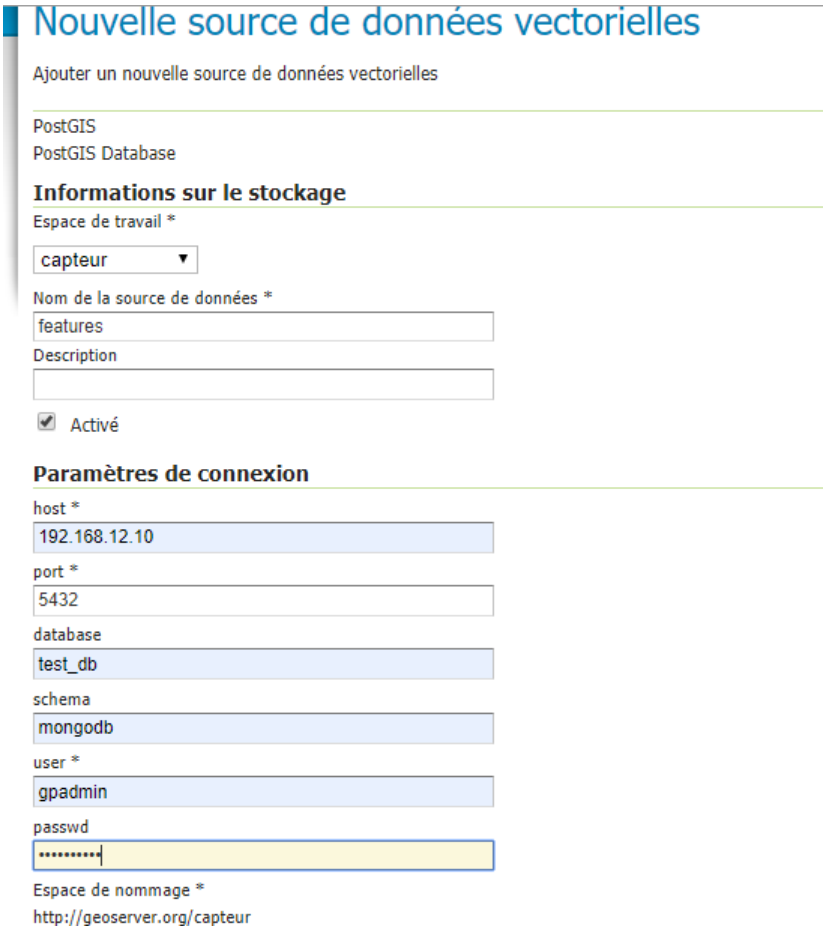


FIGURE 58 – Nouvel entrepôt PostGIS

Ensuite, nous introduisons les information nécessaires à la connexion geoserver-greenplum ; comme le nom de l'espace de travail, l'adresse de la machine contenant GreenPlum, son port, le nom de la base de données, le schéma ainsi que le mot de passe.

3. Une fois les données chargées, nous les publions en précisant l'emprise géographique à calculer à partir des données :

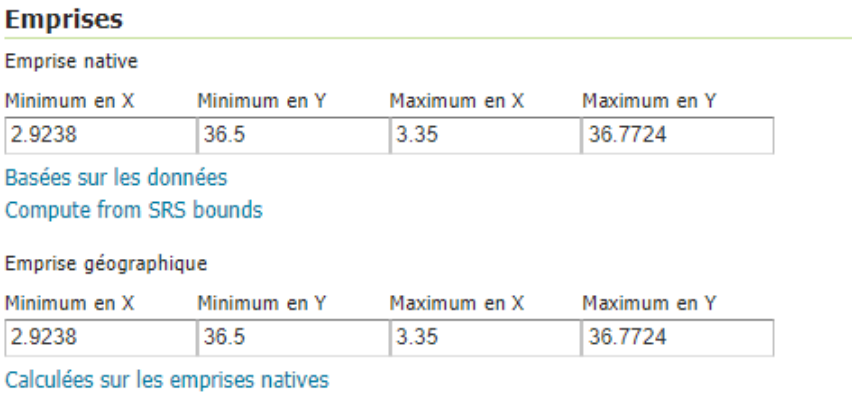


FIGURE 59 – Publication des données

4. Les données publiées, nous voyons bel et bien le type géométrique "point".

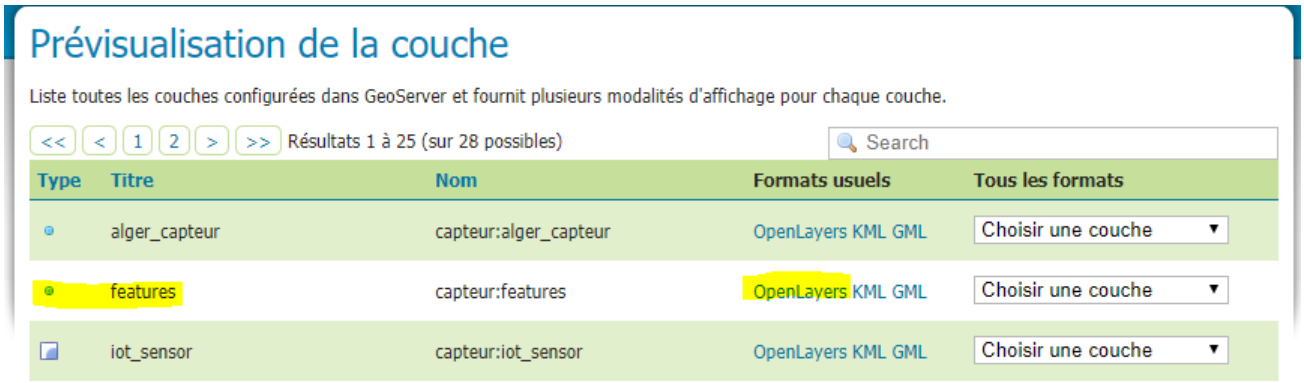
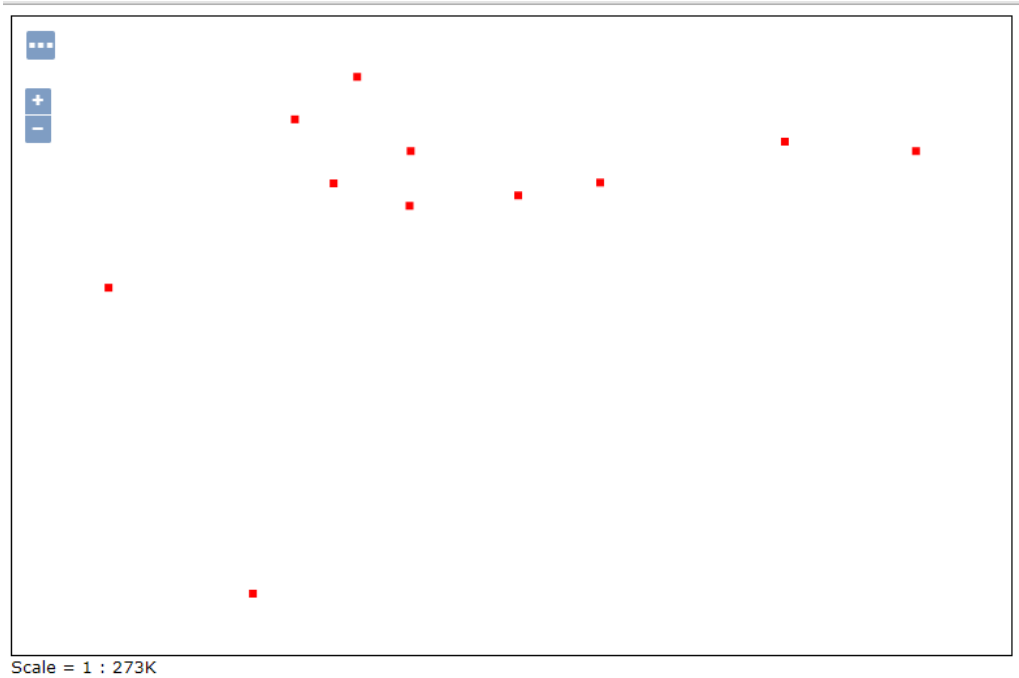


FIGURE 60 – Type de données publiées

Grâce à l'option de pré-visualisation de couche, incluse sur geoserver, nous pouvons bénéficier d'une première vue sur les données et faire des requêtes WMS pour charger les informations sur les capteurs de notre base de données.



Scale = 1 : 273K

features

fid	id	type	geometry_type	geometry_coord1	geometry_coord2	properties_marke
features.fid--4eeebf82_16ad53c65dc_-7e88	5cf79e66e569999a61bf66ef	Feature	Point	3.0551	36.7724	#dced10

FIGURE 61 – Prévisualisation de la couche

5.6 Application cliente

La dernière étape consiste à développer l'application cliente pour permettre la visualisation et l'utilisation de notre GIS-IoT par d'autres utilisateurs.
Notre choix d'application est un géoportail, un type d'application web cartographique.
Le géoportail est un portail Web public permettant l'accès à des services de recherche et de visualisation de données géographiques ou géolocalisées. De ce fait, il va permettre une meilleure illustration de notre travail.
Afin de le réaliser, nous utilisons une infrastructure logicielle de création d'application web cartographique, MapStore.

5.6.1 Devellopement de l'application

Nous avons choisis d'utiliser MapStore2 afin de créer notre carte.
Après son installation (qui se trouve dans l'annexe D), nous nous positionnons en création de nouvelle carte. Puis, nous commençons par charger au catalogue les couches servies par notre serveur GIS, à savoir nos capteur :

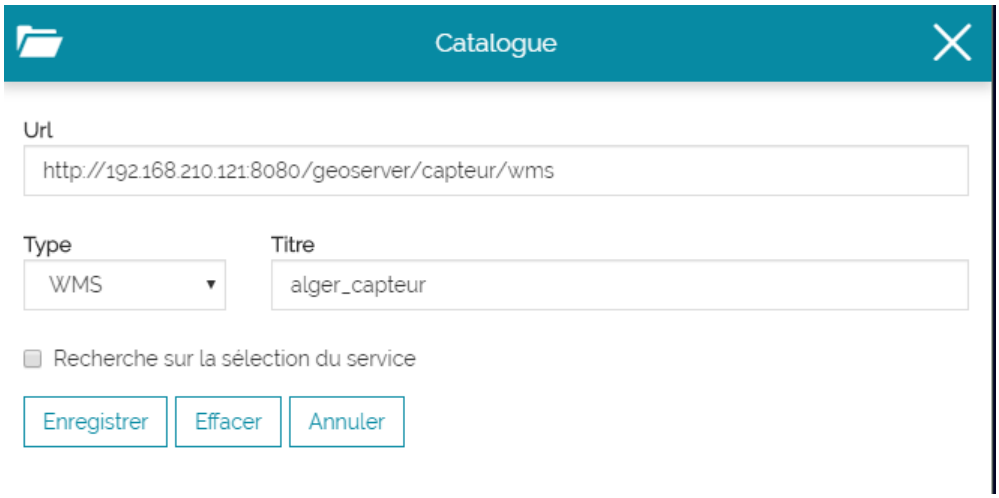


FIGURE 62 – Ajout des couches au catalogue

Cette connexion va directement charger les couches contenues dans l'espace de travail nommé. L'utilisateur n'aura plus qu'a ajouter à la carte la couche qu'il souhaite.

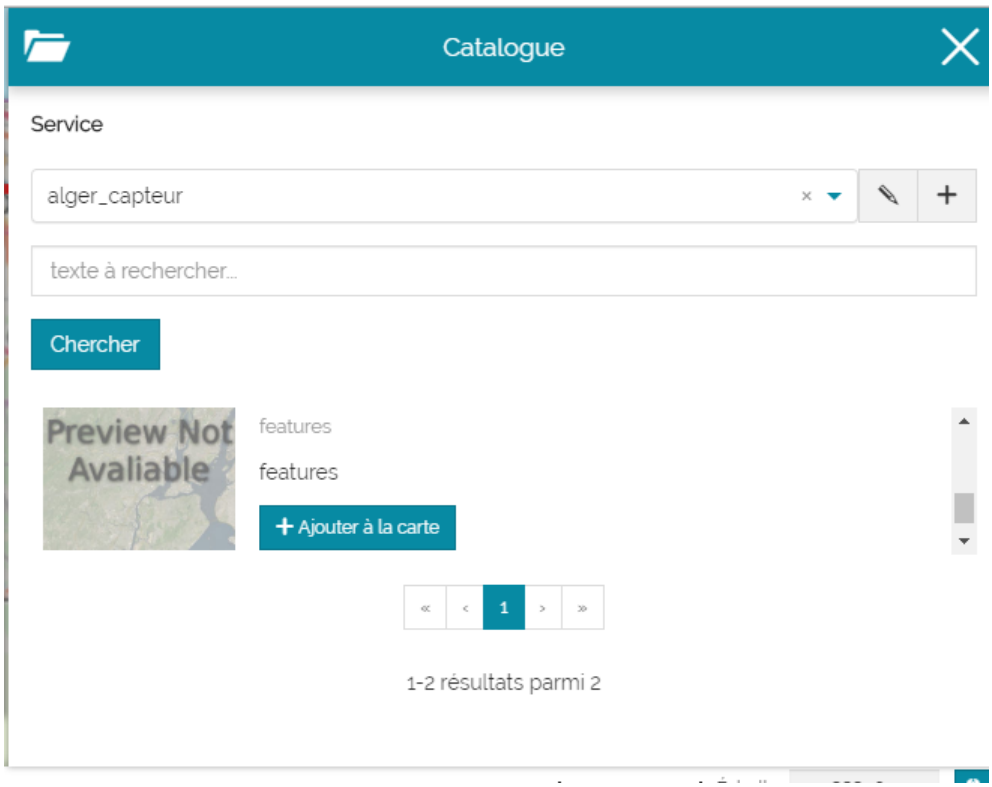


FIGURE 63 – Ajout des couches souhaité à la carte

Il peut par la suite, sélectionner ou non la couche de données qu'il souhaite voir.

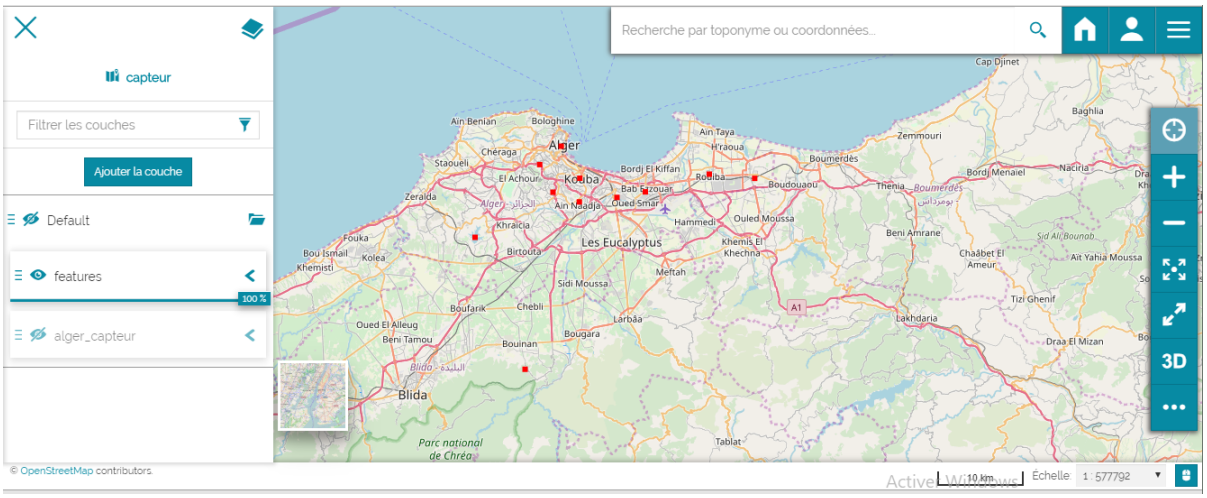


FIGURE 64 – Visualisation de couches

Dans la cartographie numérique, comme géoportail, une carte consiste en une superposition de couches de données géographiques et leurs styles dans des bases de données, qu'on appel des "overlay". Elle contient divers éléments de carte comme une légende et une barre d'échelle.
L'interface de portail webGIS est constitué en grande partie de :

— La table des matières comprend le contenu de la carte, comme les groupes de couches, les calques et les fonctions de gestion des calques ajoutés.

— La barre de menus comprend une barre de recherche, des connexions à la page d'accueil et à la page des comptes, une liste d'options contenant plusieurs fonctions.

— La barre latérale est un panneau de navigation.

— Le pied de page comprend une barre d'échelle, les coordonnées et les systèmes de coordonnées.

— le panneau de visualisation ou les cartes sont affichés.

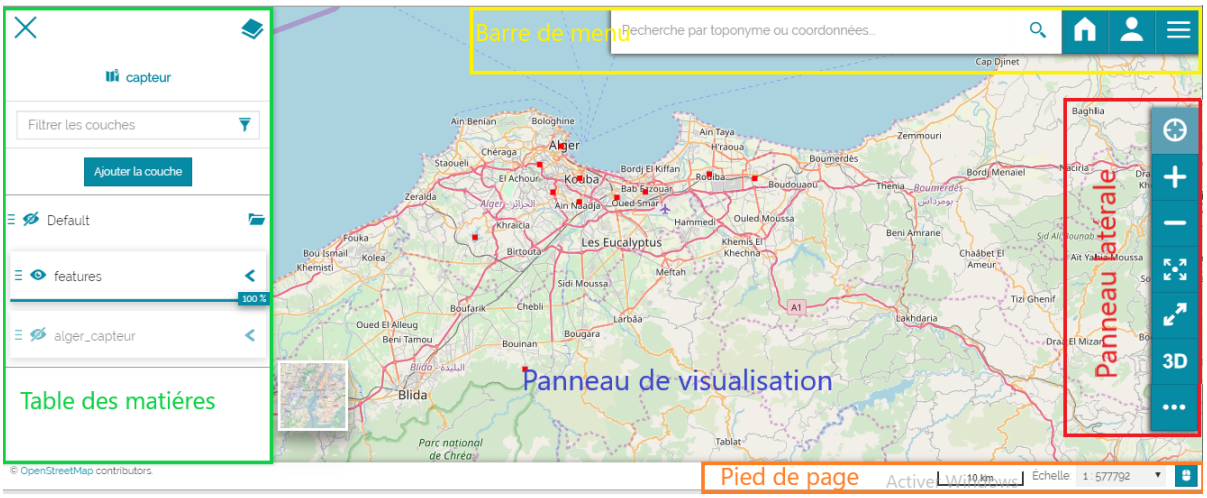


FIGURE 65 – Description de l'interface de la carte

La fonctionnalité la plus important encore, reste le chargement des informations stocké dans l'entrepôt de données liées au capteur grâce aux requêtes WMS servi par le serveur cartographique Geoserver.

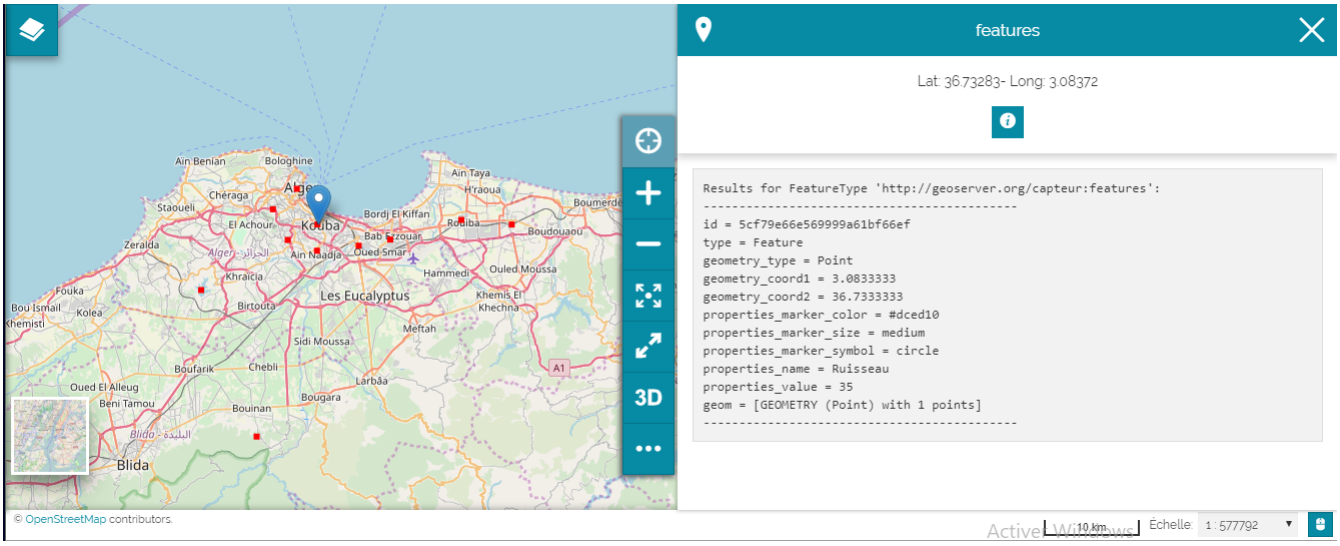


FIGURE 66 – Informations sur les données

Nous pouvons par la suite la sauvegarder afin de pouvoir la partager avec d'autres clients notre carte. La carte est accessible à l'adresse http ://192.168.210.125 :8082/mapstore sur le réseau de l'école (par mesure de sécurité).

5.7 Conclusion

Nous avons vu dans ce qui précéde, l'implémentation d'un GIS orienté IoT dans un environnement de production, et totalement construit avec des solutions open source, à savoir : kafka, mongodb, greenplum, geoserver et mapstore.

Chapitre 6

6 Conclusion Générale

Dans un avenir très proche, les objets connectés deviendront incontournable, leurs nombre ne fera que croître et leurs usage ne cessera de se diversifier, il est alors essentiel de mettre au point une architecture évolutive à usage général qui sera en mesure de se mettre à l'échelle des avancés technologiques qui font et feront leurs apparitions dans le monde.

Ce travail contribue à la mise en oeuvre d'un GIS orienté IoT dont l'architecture est évolutive.

Dans les premiers chapitres de notre travail, nous avons introduit les notions de base nécessaires à la conception de notre système d'information géographique sur des objets connectés, à savoir, ce qu'est l'internet des objet, un système d'information géographique mais aussi des notions concernant le traitement massif de données qui est primordial pour une solution adaptée à un environnement de production.

Ensuite, nous avons décrit et justifié le choix des différents blocs constituant notre architecture finale à sa voir : Apache Kafka, Mongodb, Greenplum, Geoserver, Mapstore.

— Apache kafka : Plateforme d'ingestion de données

— Mongodb : Base de donnée cache.

— Greenplum : Base de donnée Big Data.

— Geoserver : Serveur SIG.

— Mapstore : Application cliente.

Finalement, nous avons présenté les étapes suivies pour la mise en place de chaque élement de notre architecture tout en explicitant le cheminement des données qui va de leurs acquisition à travers capteurs IoT jusqu'à leurs affichage sur l'application cliente, représenté par un géoportail dédiées aux données géolocalisées.

6.1 Perspectives

Diverses améliorations pourraient être apportées à ce projet pour une plus grande efficacité :

— Ajouter un bloc de traitement de flux de donnée intermédiaire entre la file d'attente et la base de donnée cache.

— Utiliser des méthodes d'exploration de données (Data mining) pour obtenir plus d'information à partir des données spatiales acquises grâce à la bibliothèque MADlib.

— Appliquer des tests de performance plus approfondis sur les différents blocs de notre architecture.

— Assurer l'intégrité et la confidentialité des données.

Annexes

A Fichier de configuration et script

A.1 Ficher de configuration

La totalité du fichier Docker compose de configuration docker-compose.yml pour l'installation de la plate-forme IoT contenant six conteneurs

```
version : '3.3'

services :
 mosquitto :
    image : eclipse-mosquitto :1.5.5
    hostname : mosquitto
    container_name : mosquitto
    expose :
     - "1883"
    ports :
     - "1883 :1883"
 zookeeper :
    image : zookeeper :3.4.9
    restart : unless-stopped
    hostname : zookeeper
    container_name : zookeeper
    ports :
     - "2181 :2181"
    environment :
     ZOO_MY_ID : 1
     ZOO_PORT : 2181
     ZOO_SERVERS : server.1=zookeeper :2888 :3888
    volumes :
     - ./zookeeper/data :/data
     - ./zookeeper/datalog :/datalog
 kafka :
    image : confluentinc/cp-kafka :5.1.0
    hostname : kafka
    container_name : kafka
    ports :
     - "9092 :9092"
    environment :
    KAFKA_LISTENER_SECURITY_PROTOCOL_MAP :PLAINTEXT :
    PLAINTEXT,PLAINTEXT
    KAFKA_ADVERTISED_LISTENERS :PLAINTEXT ://kafka :9092,
    PLAINTEXT_HOST :localhost :29092
    KAFKA_ZOOKEEPER_CONNECT : "zookeeper :2181"
    KAFKA_BROKER_ID : 1
```

```
    KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR : 1
    volumes :
     - ./kafka/data :/var/lib/kafka/data
    depends_on :
     - zookeeper
  kafka-connect :
    image : confluentinc/cp-kafka-connect :5.1.0
    hostname : kafka-connect
    container_name : kafka-connect
    ports :
     - "8083 :8083"
    environment :
     CONNECT_BOOTSTRAP_SERVERS : "kafka :9092"
     CONNECT_REST_ADVERTISED_HOST_NAME : connect
     CONNECT_REST_PORT : 8083
     CONNECT_GROUP_ID : compose-connect-group
     CONNECT_CONFIG_STORAGE_TOPIC : docker-connect-configs
     CONNECT_OFFSET_STORAGE_TOPIC : docker-connect-offsets
     CONNECT_STATUS_STORAGE_TOPIC : docker-connect-status
     CONNECT_KEY_CONVERTER : org.apache.kafka.connect.
     json.JsonConverter
     CONNECT_VALUE_CONVERTER : org.apache.kafka.connect.json.
     JsonConverter        CONNECT_INTERNAL_KEY_CONVERTER :
    "org.apache.kafka.connect.json.JsonConverter"
     CONNECT_INTERNAL_VALUE_CONVERTER :
    "org.apache.kafka.connect.json.JsonConverter"
     CONNECT_CONFIG_STORAGE_REPLICATION_FACTOR : "1"
     CONNECT_OFFSET_STORAGE_REPLICATION_FACTOR : "1"
     CONNECT_STATUS_STORAGE_REPLICATION_FACTOR : "1"
     CONNECT_PLUGIN_PATH : '/usr/share/java,/etc/kafka-connect/jars'
     CONNECT_CONFLUENT_TOPIC_REPLICATION_FACTOR : 1
    volumes :
     - /tmp/custom/jars :/etc/kafka-connect/jars
    depends_on :
     - zookeeper
     - kafka
     - mosquitto
  mongo-db :
    image : mongo :4.0.5
    hostname : mongo-db
    container_name : mongo-db
    expose :
     - "27017"
    ports :
     - "27017 :27017"
```

```
  command : --bind_ip_all --smallfiles
  volumes :
   - ./mongo-db :/data
mongoclient :
  image : mongoclient/mongoclient :2.2.0
  container_name : mongoclient
  hostname : mongoclient
  depends_on :
   - mongo-db
  ports :
   - 3000 :3000
  environment :
   MONGO_URL : "mongodb ://mongo-db :27017"
   PORT : 3000
  expose :
   - "3000"
```

A.2 Fichier de données test

Dans ce qui suit, le fichier contenant les données tests, écrit sous un format GeoJSON.
Le GeoJSON (de l'anglais Geographic JSON, signifiant littéralement JSON géographique) est un format ouvert d'encodage d'ensemble de données géospatiales simples utilisant la norme JSON (JavaScript Object Notation).
[39] Il permet de décrire des données de type point, ligne, chaîne de caractères, polygone, ainsi que des ensembles et sous-ensembles de ces types de données et d'y ajouter des attributs d'information qui ne sont pas spatiales.


```
{
   "type" : "FeatureCollection",
   "features" : [
     {
      "type" : "Feature",
      "geometry" : {
      "type" : "Point",
      "coordinates" : [
       3.0551,
       36.7724
      ]
     },
       "properties" : {
       "marker-color" : "dced10",
       "marker-size" : "medium",
       "marker-symbol" : "circle",
       "name" : "Alger Centre",
       "value" : 35
      }
     },
     {
      "type" : "Feature",
      "geometry" : {
      "type" : "Point",
      "coordinates" : [
       3.14,
       36.71
      ]
     },
       "properties" : {
       "marker-color" : "#dced10",
       "marker-size" : "medium",
       "marker-symbol" : "circle",
       "name" : "El Harrach",
       "value" : 36
```

```
 }
},
{
 "type" : "Feature",
 "geometry" : {
 "type" : "Point",
 "coordinates" : [
  3.1833,
  36.7167
 ]
 },
  "properties" : {
  "marker-color" : "#dced10",
  "marker-size" : "medium",
  "marker-symbol" : "circle",
  "name" : "Bab ezzouar",
  "value" : 35
 }
},
{
 "type" : "Feature",
 "geometry" : {
 "type" : "Point",
 "coordinates" : [
  3.28079,
  36.73829
 ]
},
  "properties" : {
  "marker-color" : "#dced10",
  "marker-size" : "medium",
  "marker-symbol" : "circle",
  "name" : "Rouiba",
  "value" : 36
 }
},
{
 "type" : "Feature",
 "geometry" : {
 "type" : "Point",
 "coordinates" : [
  3.35,
  36.7333
 ]
 },
```

```
  "properties" : {
  "marker-color" : "#dced10",
  "marker-size" : "medium",
  "marker-symbol" : "circle",
  "name" : "Reghaia",
  "value" : 37
 }
},
{
 "type" : "Feature",
 "geometry" : {
 "type" : "Point",
 "coordinates" : [
  3.0833333,
  36.7333333
 ]
 },
  "properties" : {
  "marker-color" : "#dced10",
  "marker-size" : "medium",
  "marker-symbol" : "circle",
  "name" : "Ruisseau",
  "value" : 35
 }
},
{
 "type" : "Feature",
 "geometry" : {
 "type" : "Point",
 "coordinates" : [
  3.0827,
  36.7045
 ]
 },
  "properties" : {
  "marker-color" : "#dced10",
  "marker-size" : "medium",
  "marker-symbol" : "circle",
  "name" : "Ain Naadja",
  "value" : 37
 }
},
{
 "type" : "Feature",
 "geometry" : {
```

```
 "type" : "Point",
 "coordinates" : [
  3.0426,
  36.7163
 ]
 },
 "properties" : {
  "marker-color" : "#dced10",
  "marker-size" : "medium",
  "marker-symbol" : "circle",
  "name" : "bir khadem",
  "value" : 36
 }
},
{
 "type" : "Feature",
 "geometry" : {
 "type" : "Point",
 "coordinates" : [
  3.0222,
  36.75
 ]
},
 "properties" : {
  "marker-color" : "#dced10",
  "marker-size" : "medium",
  "marker-symbol" : "circle",
  "name" : "Benaknoun",
  "value" : 38
 }
},
{
 "type" : "Feature",
 "geometry" : {
 "type" : "Point",
 "coordinates" : [
  2.9238,
  36.6613
 ]
},
"properties" : {
 "marker-color" : "#dced10",
 "marker-size" : "medium",
 "marker-symbol" : "circle",
 "name" : "Douera",
```

```
          "value" : 38
        }
      }
   ]
}
```

A.3 Script ETL

Ce script ETL [27] permet d'extraire, traiter et charger les données de notre base de données mongodb vers Greenplum.

```
   import pymongo
from pymongo import MongoClient
import psycopg2
def create_alger(results) :
   '''
   Take the query outcome and convert to a list and return it.
   It also defines headers and return it.
   '''
   records = []
   headers = ['id','type']
   for record in results :
     tmp = []
     tmp.append(str(record['_id']).split("(")[0])
     tmp.append(record['type'])
     records.append(tmp)
   print('first step okey.')
   return records, headers
def create_features(results) :
   '''
   Take the query outcome and convert to a list and return it.
   It also defines headers and return it.
   '''
   records = []
   headers = ['id','type','geometry_type','geometry_coordinates1',
     'geometry_coordinates2', 'properties_marker-color',
     'properties_marker-size','properties_marker-symbol',
     'properties_name','properties_value']
   for record in results :
     for feature in record['features'] :
       tmp = []
       tmp.append(str(record['_id']).split("(")[0])
       tmp.append(feature['type'])
```

27. En anglais Extract Transform Load

```
        geometry = feature['geometry']
        tmp.append(geometry['type'])
        "try :"
        tmp.append(str(geometry['coordinates'][0]))
        "except IndexError :"
        "tmp.append('NA')"
        "try :"
        tmp.append(str(geometry['coordinates'][1]))
        "except IndexError :"
        "tmp.append('NA')"
        properties = feature['properties']
        tmp.append(str(properties['marker-color']))
        tmp.append(properties['marker-size'])
        tmp.append(properties['marker-symbol'])
        tmp.append(properties['name'])
        tmp.append(str(properties['value']))
        records.append(tmp)
    return records, headers
def create_table(records, headers, file_path) :
    "Take a list of records and headers and generate csv"
    f = open(file_path, 'w', encoding='utf-8')
    row_len = len(headers)
    f.write(format_list(headers, row_len, ',', '"'))
    for record in records :
     f.write(format_list(record, row_len, ',', '"'))
    f.close()
    print('CSV file successfully created : '.format(file_path))
def format_list(list, length, delimiter, quote) :
    counter = 1
    string = ''
    for record in list :
     if counter == length :
      string += quote + record + quote + ''
    else :
      string += quote + record + quote + delimiter
    counter += 1
    return string
def pg_load_table(file_path, table_name) :
      try :
        conn = psycopg2.connect(dbname="test_db",
               user="gpadmin",
               host="192.168.12.10",
               port="5432",
               password="marya21000")
        print("Connecting to Database")
```

```
        cur = conn.cursor()
        f = open(file_path, "r", encoding='utf-8')
        cur.execute("Truncate  Cascade ;".format(table_name))
        print("Truncated ".format(table_name))
        cur.copy_expert("copy  from STDIN CSV HEADER".
        format(table_name),f) cur.execute("commit ;")
        print("Loaded data into ".format(table_name))
        conn.close()
        print("DB connection closed.")
      except Exception as e :
        print("Error : ".format(str(e)))
def main() :
   # (1) Connect to MongoDB instance
     client = MongoClient('mongodb ://192.168.210.105 :
     27017/capteur')
   # (2) Choose database and create the db object
     db = client.capteur
   # (3) Choose the collection and create Collection Object
     alger = db.alger
     print('Total Record for the collection :
     ' + str(alger.count()))
   # (4) Retrieve all data from the restaurants collection
     results=alger.find()
   # (4-1) Transform data into a structured tables
     alger_tup = create_alger(results)
     alger_records = alger_tup[0]
     alger_headers = alger_tup[1]
     create_table(alger_records, alger_headers, './alger.csv')
   # (5) Retrive all data again for grades
     results = alger.find()
   # (5-1) Transform data for grades
     features_tup = create_features(results)
     features_records = features_tup[0]
     features_headers = features_tup[1]
     create_table(features_records, features_headers,
     './features.csv')
   # (6) Load data into Postgres
         pg_load_table('./alger.csv', 'mongodb.alger')
     pg_load_table('./features.csv', 'mongodb.features')

   if __name__ == "__main__" :
        main()
```

B Installation Greenplum

Nous avons choisi d'installer la version 5.17.0 de GreenPlum, disponible sur le site Pivotal, sur le système d'exploitation CentOS 7.x 64-bit .

B.1 Pré requis du système hôte

l'installation de GreenPlum 5.17 en mode production exige que serveurs se munissent de certaines spécifications systèmes afin de le prendre en charge.

Système d'exploitation	CentOS 64-bit 6.x ou 7.x
Fichier système	De type XFS
Processeur minimum	Pentium Pro
Mémoire RAM minimum	16 GB de RAM par instance
Exigence du disque	- 150 MB par hote pour l'installation de Greenplum - 300 MB (approximatifs) par segments pour les méta données - Espace libre approprié pour les données sur disques d'une capacité maximale de 70 %. - Haut débit, Stockage local
Exigence réseaux	10 Gigabit Ethernet. Un switch dédié, non-bloqué.
Logiciel et commodités	Librairie de compression zlib, bash shell, GNU tars, GNU zip,GNU sed et perl.

Table 5: Pré-requis système

Etant données les ressources offertes à nous, nous avons choisis d'installer sur un master et deux slaves.
L'installation se fait sur le segment master et puis ce dernier se charge de l'installation sur les segments.

Il faut aussi verifier que le SELinux est désactivé :

```
# sestatus
SELinuxstatus : disabled
```

si ce n'est pas le cas, il faut le faire en mofiant le fichier /etc/selinux/config file :

```
SELINUX=disabled
```

Il faut aussi désactiver tout pare-feu, ainsi que les iptables (ou firewalld)
cette commance vérifie l'état des iptables :

```
# /sbin/chkconfig --list iptables
```

Si ceux ci sont désactivé, on aura un retour :

```
iptables 0 :off 1 :off 2 :off 3 :off 4 :off 5 :off 6 :off
```
pour les désactiver :
```
/sbin/chkconfig iptables off
```

Pour vérifier l'état du firewalld :
```
# systemctl status firewalld
```
Si il est désactivé nous aurons une réponse :
```
* firewalld.service - firewalld - dynamic firewall daemon
   Loaded : loaded (/usr/lib/systemd/system/firewalld.service ; disabled ; vendor preset :
enabled)
   Active : inactive (dead)
```
Sinon, on execute ces commandes pour les désactiver :
```
# systemctl stop firewalld.service
# systemctl disable firewalld.service
```

B.2 Édition des paramètres du système d'exploitation

B.2.1 Paramètre du système linux

Dans le fichier Dans /etc/hosts nous précisions les adresses ip des instances, ainsi que les noms que nous souhaitons leur atribuer :
```
192.168.12.5 greenplum
192.168.12.21 slave1
192.168.12.4 slave2
```

Et dans dans /etc/sysctl.conf :
```
xfs_mount_options = rw,noatime,inode64,allocsize=16m
kernel.shmmax = 500000000
kernel.shmmni = 4096
kernel.shmall = 4000000000
kernel.sem = 500 1024000 200 4096
kernel.sysrq = 1
kernel.core_uses_pid = 1
kernel.msgmnb = 65536
kernel.msgmax = 65536
kernel.msgmni = 2048
net.ipv4.ip_forward = 0
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_max_syn_backlog = 4096
net.ipv4.conf.all.arp_filter = 1
net.ipv4.ip_local_port_range = 1025 65535
net.core.netdev_max_backlog = 10000
```

```
net.core.rmem_max = 2097152
net.core.wmem_max = 2097152
vm.overcommit_memory = 2
vm.swappiness = 10
vm.dirty_expire_centisecs = 500
vm.dirty_writeback_centisecs = 100
vm.dirty_background_ratio = 0
vm.dirty_ratio=0
vm.dirty_background_bytes = 1610612736
vm.dirty_bytes = 4294967296
```

Dans /etc/security/limits.conf :

```
* soft nofile 65536
* hard nofile 65536
* soft nproc 131072
* hard nproc 131072
```

Dans le fichier /etc/fstab :

```
/dev/vda /vda xfs nodev,noatime,nobarrier,inode64 0 0
```

Les étapes suivantes sont à refaire à chaque redémarrage :

1. `# echo deadline > /sys/block/vda/queue/scheduler`
2. `# /sbin/blockdev --setra 16385 /dev/vda`

Ensuite, dans le fichier /etc/systemd/logind.conf :

```
RemoveIPC=no
```

ceci prend effet une fois le service redémarré :

```
service systemd-logind restart
```

Et dans /etc/ssh/sshd_config :

```
Max Startups 10 :30 :200
```

Pour redémarrer le service :

```
$ service sshd restart
```

B.2.2 Installation de Greenplum v5.17.0

Configuration à faire

Sur le master, il faut créer un utilisateur avec un mot de passe, pour nous :

```
# groupadd gpadmin
# useradd gpadmin -g gpadmin
# passwd gpadmin
notre_mp
#passwd centos
```

```
notre_mp
#passwd root
notre_mp
```

Installation

1. Pour l'installation de la distribution RPM :
   Nous téléchargeons sur Pivotal la version v5.17.0 de greenplum, puis nous lançons l'installation :
   ```
   rpm -Uvh ./greenplum-db-5.17.0-rhel7-x86_64.rpm
   # chown -R gpadmin /usr/local/greenplum*
   # chgrp -R gpadmin /usr/local/greenplum*
   ```

2. Installation de Greenplum sur tous les hôtes restants :

   (a) Mettre le chemin d'accès au dossier d'installation en source :
   ```
   # source /usr/local/greenplum-db/greenplum_path.sh
   ```

   (b) Créer un fichier hostfile_exkeys qui contient les noms du master et des slaves (comme nommé précédemment dans /etc/hosts), et mettre autant de noms qu'il y'a de slave avec l'extension (-1, -2, ... n-1 nombre de slaves) :
   ```
   127.0.0.1 localhost localhost.localdomain localhost4
   localhost4.localdomain4
   : :1 localhost localhost.localdomain localhost6
   localhost6.localdomain6
   greenplum-master
   greenplum-master-1
   greenplum-salve1
   greenplum-slave1-1
   greenplum-salve2
   greenplum-salve2-1
   ```

   (c) Lancer l'installation avec la commande :
   ```
   # gpseginstall -f hostfile_exkeys -u gpadmin -p notre_mp
   ```

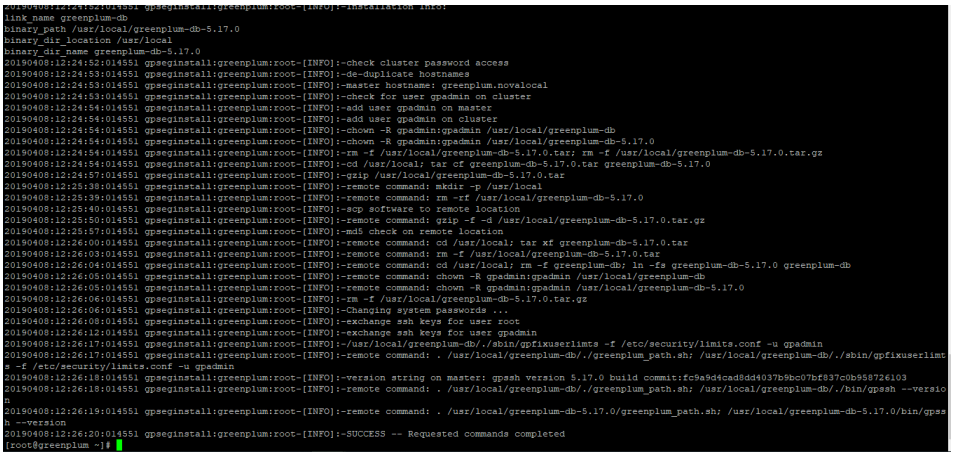


FIGURE 67 – Installation de greenplum

3. Pour confirmer que l'installation est réussi :
   `$ su - gpadmin`
   `# source /usr/local/greenplum -db/greenplum_path.sh`
   `$ gpssh -f hostfile_exkeys -e ls -l $GPHOME`
   Cette commande permet l'accès aux hôtes par ssh, sans mot de passe, si son retour est un succès, donc l'installation est réussit.

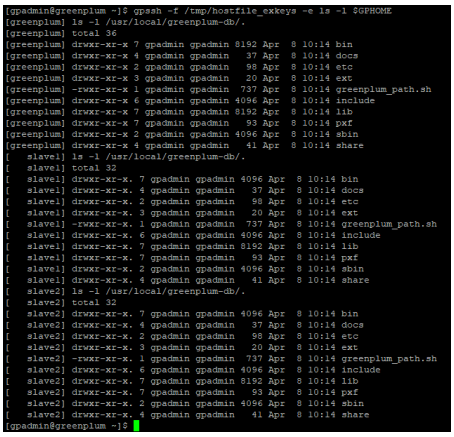


FIGURE 68 – Validation de l'installation

B.2.3 Installation des extensions

Installation de l'extension

Afin d'installer l'extension PostGIS, Grennplum a besoin de ces composants :

— PostGIS 2.1.5

— Proj 4.8.0

— Geos 3.4.2

— GDAL 1.11.1

— Json 0.12

— Expat 2.1.0

```
source /usr/local/greenplum-db/greenplum_path.sh
sudo yum install json-c-devel -y
* sudo yum install geos-devel -y
* sudo yum install -y proj-devel
sudo yum install git
git clone https ://github.com/greenplum-db/geospatial.git
sudo yum install gcc.x86_64
* sudo yum install libpqxx.x86_64 libpqxx-devel.x86_64
sudo yum install libxml2-devel.x86_64
./configure --with-pgconfig=/usr/local/greenplum -db-5.7.0/bin/ pg_config --without
-raster --without -topology --prefix=$GPHOME
sudo yum install byacc -- enablerepo = base
sudo yum groupinstall " Development tool "
```

Enfin nous installons l'extension PostGIS v2.1.5

```
gppkg -i postgis-ossv2.1.5_pv2.1_gpdb4.4-rhel5-x86_64.gppkg
```

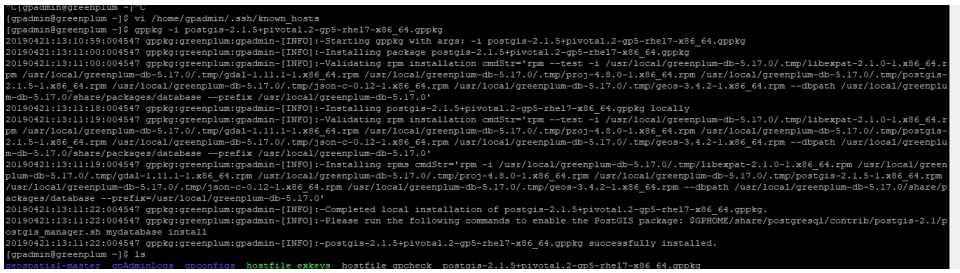


FIGURE 69 – Installation de PostGIS

Activation du support PostGIS

Dans le fichier ∼/bashrc :

```
MASTER_DATA_DIRECTORY=/usr/local/data/master/gpseg-1
export MASTER_DATA_DIRECTORY
export PGPORT=5432
#export PGUSER=gpadmin export PGDATABASE=default_login_database_name
```

```
   at the end : export LD_PRELOAD=/lib64/libz.so.1 ps
Ce fichier s'exécute en arrière plan à chaque démarrage.
```

```
   source ∼/.bashrc
```

Par la suite, nous activons la base de données ainsi que l'extension PostGIS sur la base de donnée de notre choix :

1. `su - gpadmin`
2. `gpstart`
3. `gpstat`
4. `psql -U gpadmin postgres`
5. `=>CREATE DATABASE mydatabase`
6. `=>\q`
7. `cd /usr/local/greenplum-db-5.17.0/share/postgresql/contrib/ postgis-2.1/ ./postgis_manager.sh mydatabase install`
8. `raster2pgsql -G`
9. `export POSTGIS_GDAL_ENABLED_DRIVERS="GTiff PNG JPEG GIF"`
10. `=>SELECT short_name, long_name FROM ST_GDALDrivers() ;`
11. `=>SELECT PostGIS_full_version() ;`

C Installation GeoServer

L'installation de Geoserver requiert :

1. L'environnement Java.
2. Apache TomCat.

C.1 Installation de java sur ubuntu server

Pour de meilleurs performances, il est conseillé d'installer java 8 comme environnement pour Geoserver.

1. Mise à jour du systéme et installation de paquets :
   ```
   apt-get update  apt-get upgrade
   apt-get install software-properties-common
   ```
2. Ajouter un repertoir Java et refaire une mise à jour :
   ```
   add-apt-repository ppa :webupd8team/java
   apt-get update
   ```
3. Installation de java 8
   ```
   apt-get install oracle-java8-installer
   ```

C.2 Installation de Apache tomcat

Apache Tomcat est un serveur web et conteneur de servlet qui est utilisé pour servir les applications Java. Tomcat est une implémentation open source des technologies Java servlet et JavaServer Pages, crée par Apache Software Foundation.

1. Tout d'abord, il faut créer un utilisateur qui va activer le service tomcat :
   ```
   sudo groupadd tomcat
   sudo useradd -s /bin/false -g tomcat -d /opt/tomcat tomcat
   ```

2. Installation de TomCat :
   (a) changer de repertoire pour le telechargement :
   `cd /tmp`
   (b) telecharger la version désiré (pour nous 9.0.17) :
   `curl -O http ://www-eu.apache.org/dist/tomcat/`
   `tomcat-9/v9.0.17/bin/apache-tomcat-9.0.17.tar.gz`
   (c) Créer le dossier d'installation :
   `sudo mkdir /opt/tomcat tar xvf`
   `apache-tomcat-9.0.17.tar.gz -C`
   `/opt/tomcat --strip-components=1`
3. Mettre à jour les permissions :
```
cd /opt/tomcat
sudo chgrp -R tomcat /opt/tomcat
sudo chmod -R g+r conf
sudo chmod g+x conf
sudo chown -R tomcat webapps/ work/ temp/ logs/
```
4. Créer un fichier de service systemd :

   (a) définir le chemin du répertoire java dans le fichier `/etc/environment` :
   `JAVA_HOME="/usr/lib/jvm/java-8-oracle/jre"`
   `source /etc/environment`
   (b) Ouvrir le fichier su service tomcat :
   `sudo vi /etc/systemd/system/tomcat.service`
   et y copier cette configuration :

```
[Unit]
Description=Apache Tomcat Web Application Container
After=network.target

[Service]
Type=forking

Environment=JAVA_HOME=/usr/lib/jvm/java-8-oracle/jre
Environment=CATALINA_PID=/opt/tomcat/temp/tomcat.pid
Environment=CATALINA_HOME=/opt/tomcat
Environment=CATALINA_BASE=/opt/tomcat
Environment='CATALINA_OPTS=-Xms512M -Xmx1024M
-server -XX :+UseParallelGC'
Environment='JAVA_OPTS=-Djava.awt.headless=true
-Djava.security.egd=file :/dev/./urandom'

ExecStart=/opt/tomcat/bin/startup.sh
ExecStop=/opt/tomcat/bin/shutdown.sh
```

```
User=tomcat
Group=tomcat
UMask=0007
RestartSec=10
Restart=always

[Install]
WantedBy=multi-user.target
```

(c) Il faudra ensuite relancer le service :

```
sudo systemctl daemon-reload
sudo systemctl start tomcat
sudo systemctl status tomcat
```

5. Ajuster le pare-feu et tester le serveur :

   (a) `sudo ufw allow 8080`

   (b) se connecter à l'addresse :
   http ://server_domain_or_IP :8080

   (c) Activer tomcat :
   `sudo systemctl enable tomcat`

6. Configuration de l'interface web management :

   (a) Editer le fichier tomcat-users.xml :
   `sudo vi /opt/tomcat/conf/tomcat-users.xml`

   (b) Ajouter la ligne :

```
<user username="marya" password="Pa$word" roles="manager
-gui,admin-gui"/>
```

   (c) changer les restrictions sur les adresses IP dans les deux fichier :

```
- sudo vi /opt/tomcat/webapps/manager/META-INF/context.xml
- sudo vi /opt/tomcat/webapps/host-manager/META-INF/
context.xml
```

   (d)

```
<Context antiResourceLocking="false" privileged="true" >
< !--<Valve className="org.apache.catalina.valves.
RemoteAddrValve"
allow="127\.\d+\.\d+\.\d+| : :1|0 :0 :0 :0 :0 :0 :0 :1" />-->
</Context>
```

   (e) relancer le service :
   `sudo systemctl restart tomcat`

7. Nous pouvons ainsi voir :

   (a) sur http ://server_domain_or_IP :8080

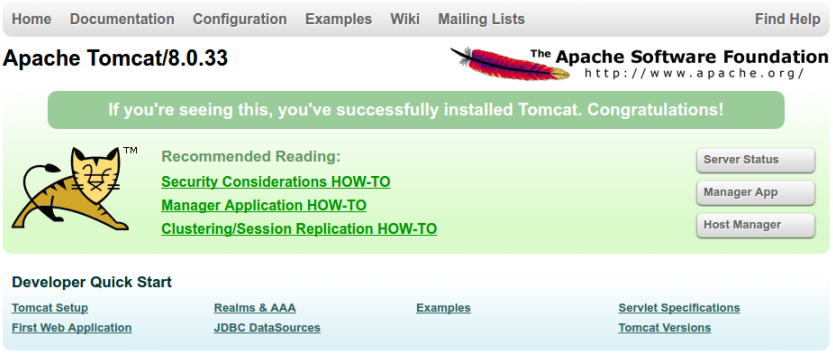


FIGURE 70 – Interface web

(b) sur http ://server_domain_or_IP :8080/manager/html

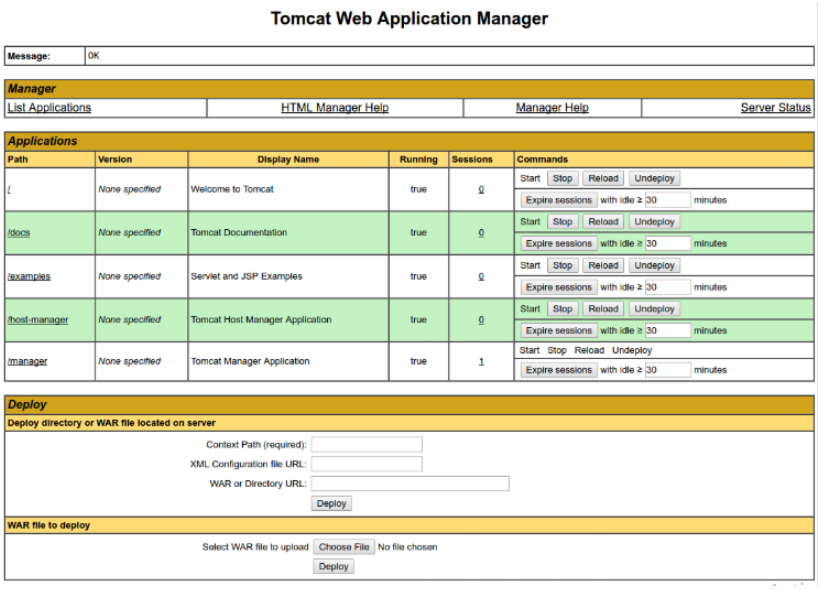


FIGURE 71 – Manager Application

(c) sur http ://server_domain_or_IP :8080/host-manager/html/

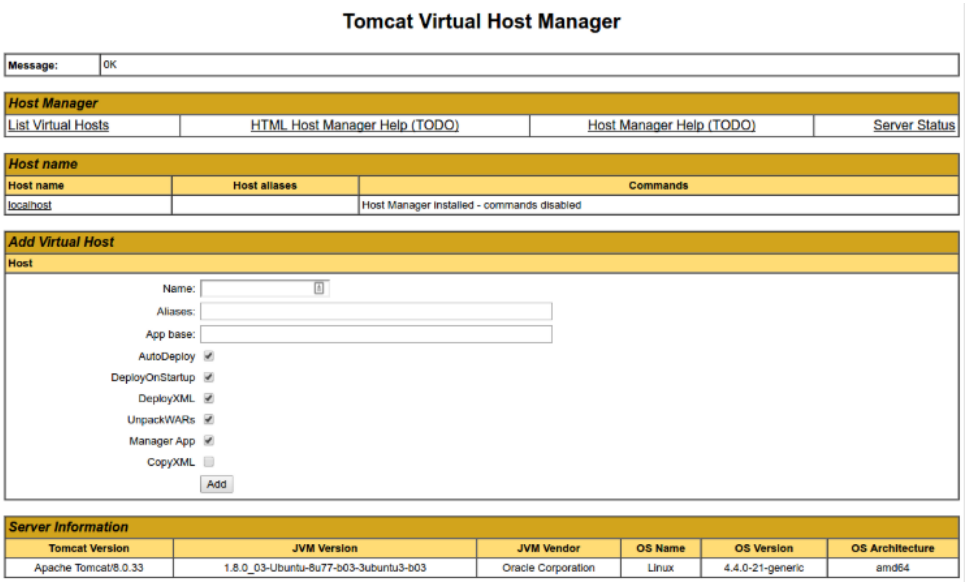


FIGURE 72 – Host Manager

C.3 Déploiement de GeoServer

1. Télécharger la version .war de GeoServer désiré, à partir de la page officielle :
   `wget https ://sourceforge.net/projects/geoserver/files/GeoServer/2.15.0/geoserver-2.15.0-war.zip/download`
   Dans notre cas v2.15.0
2. extraire les fichier :
   `unzip geoserver-2.15.0-war.zip`
3. Le fichier war de GeoServer est assez volumineux (>52 Mo). Sur Tomcat Manager il y a une limite par défaut pour l'application déployable qui est à 50 Mo que nous devrons modifier.
   `vi /opt/tomcat/webapps/manager/WEB-INF/web.xml`
   ```
   <multipart-config>
   < !-- 50MB max -->
   <max-file-size>62914560</max-file-size>
   <max-request-size>62914560</max-request-size>
   <file-size-threshold>0</file-size-threshold>
   </multipart-config>
   ```
4. relancer le service :
   `systemctl restart tomcat`
5. Et puis finalement deplacer le fichier de geoserver sur tomcat :
   `mv geoserver-war /opt/tomcat/webapps/`
6. sur la page http ://server_domain_or_IP :8080/geoserver

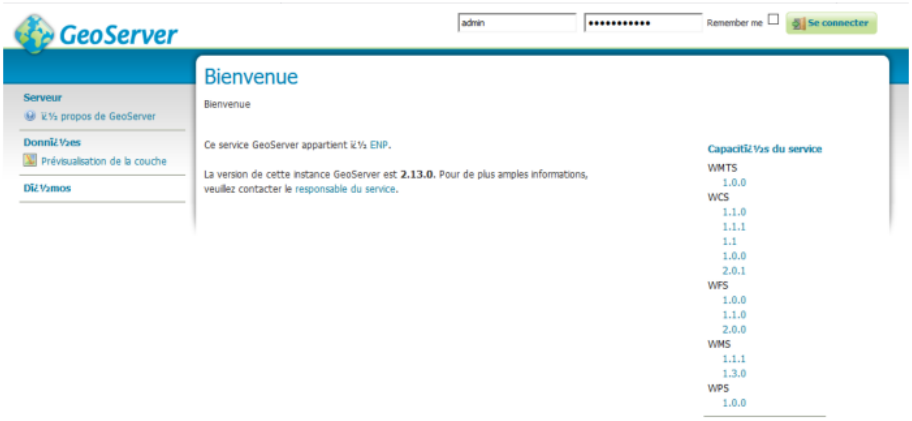


FIGURE 73 – Page de Geoserver

le mot de passe et nom d'utilisateur étant : geoserver, admin.

D Installation MapStore

1. télécharger la version désiré de mapstore sur leur site officiel :
   `wget https ://github.com/geosolutions-it/MapStore2/releases/download/v2019.01.01/mapstore2-2019.01.01-bin.zip`
2. Extraire les fichiers :
   `unzip mapstore2-2019.01.01-bin.zip`
3. changer de repertoire :
   `cd mapstore2`

4. Lancer l'application
   `./mapstore2_startup.sh`
5. Ouvrire la page web : http ://localhost :8800/mapstore
   Le dossier contient :
   - MapStore
   - Tomcat7
   - Java JRE (Win and Linux)

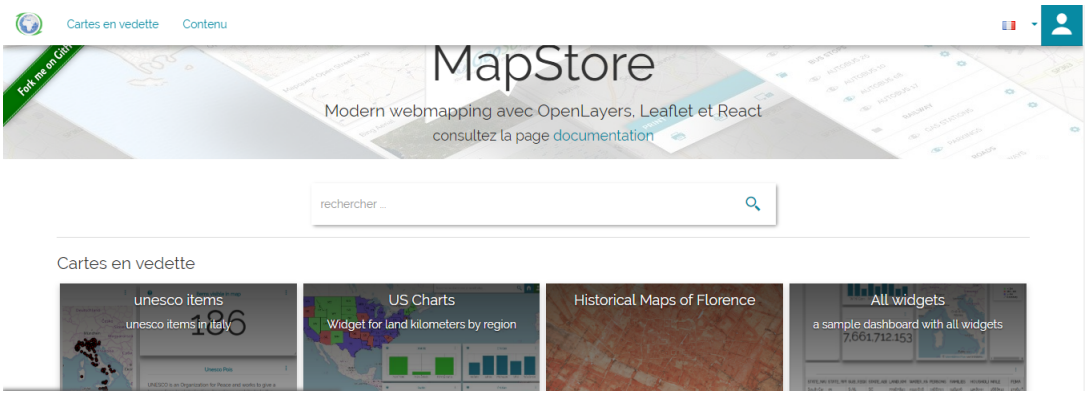


FIGURE 74 – Page de MapStore

Le mot de passe et nom d'utilisateur étant : admin, admin.

Références

[1] Elise Pelegris Henri Pornon, Pierrick Yalamas. Services web géographiques, état de l'art et perspectives. Octobre-novembre 2008.

[2] Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions).[en ligne].[consulté le 17 mai 2019].disponible sur https ://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[3] Internet des objets.[en ligne].[consulté le 02 avril 2019].disponible sur https ://www.futura-sciences.com/tech/definitions/internet-internet-objets-15158/.

[4] Françoise Massit-Folléa Pierre-Jean Benghozi, Sylvain Bureau. Internet des objets.[en ligne].[consulté le 09 mai 2019].disponible sur https ://-books.openedition.org/editionsmsh/84 ?lang=frftn3.

[5] Basic components to design a successful iot project.[en ligne].[consulté le 08 mai 2019].disponible sur http ://www.iotsens.com/basic-components-to-deisgn-successful-iot-project/.

[6] Qu'est-ce qu'une plateforme d'iot ?.[en ligne].[consulté le 11 mai 2019].disponible sur https ://www.digora.com/fr/blog/plateforme-iot-comment-choisir.

[7] Big data.[en ligne].[consulté le 08 avril 2019].disponible sur https ://www.siway.fr/blog/definitions-digital/definition/base-de-donnees-big-data-241.

[8] Bases de données et langage sql.[en ligne].[consulté le 03 mai 2019].disponible sur https ://laurent-audibert.developpez.com/cours-bd/ ?page=introduction-bases-de-donnees.

[9] Base de données big data.[en ligne].[consulté le 02 avril 2019].disponible sur https ://www.futura-sciences.com/tech/definitions/informatique-big-data-15028/.

[10] Systèmes informatiques.[en ligne].[consulté le 19 février 2019].disponible sur https ://www.universalis.fr/encyclopedie/systemes-informatiques-conception-architecture-et-urbanisation-des-systemes-d-information/.

[11] Kauri Kiiman. Introduction of open source software for gis education course project of a nordnatur intensive course opensource gis, gps and crowd sourcing in university of copenhagen, skovskolen. Avril 2013.

[12] Five most important iot platform characteristics.[en ligne].[consulté le 11 mai 2019]. disponible sur https ://wolkabout.com/blog/five-most-important-iot-platform-characteristics/.

[13] P. Charalampidis M.Dimitrakis-G. Drossis A. Fragkiadakis I. Fundulaki A.Makrogiannakis G. Margetis A. Panousopoulou S. Papadakis V.Papakonstantinou N. Partara M. Kalaitzakis, M. Bouloukakis. Building a smart city ecosystem for third party innovation in the city of heraklion. 2019.

[14] sentilo.[en ligne].[consulté le 09 mai 2019].disponible sur https ://sentilo-docs.readthedocs.io/en/latest/quickstart.html.

[15] Pavlos Charalampidis Manos Dimitrakis Giannis Drossis Alexandros Fragkiadakis Irini Fundulaki Katerina Karagiannaki Antonis Makrogiannakis Georgios Margetis Athanasia Panousopoulou Stefanos Papadakis Vassilis Papakonstantinou Nikolaos Partarakis Stylianos Roubakis Elias Tragos Elisjana Ymeralli Panagiotis Tsakalides Dimitris Plexousakis Manos Kalaitzakis, Manousos Bouloukakis and Constantine Stephanidis. Building a smart city ecosystem for third party innovation in the city of heraklion. 2019.

[16] Apache kafka - cluster architecture .[en ligne].[consulté le 17 mai 2019].disponible sur https ://data-flair.training/blogs/kafka-architecture/.

[17] What is zookeeper and why is it needed for apache kafka ? .[en ligne].[consulté le 26 mai 2019].disponible sur https ://www.cloudkarafka.com/blog/2018-07-04-cloudkarafka_what_is_zookeeper.html.

[18] Kafka architecture and its fundamental concepts.[en ligne].[consulté le 15 mai 2019].disponible sur https ://www.tutorialspoint.com/apache_kafka/apache_kafka_cluster_architecture.htm.

[19] CAREY WODEHOUSE. Should you use mongodb ? a look at the leading nosql database.[en ligne].[consulté le 15 mars 2019].disponible sur https ://www.upwork.com/hiring/data/should-you-use-mongodb-a-look-at-the-leading-nosql-database/.

[20] Meta S.Brown. Get to know relational and nosql databases that power big data analytics.[en ligne].[consulté le 18 avril 2019].disponible sur https ://www.forbes.com/sites/metabrown/2018/03/31/get-to-know-relational-and-nosql-databases-that-power-big-data-analytics/63d1859f1943. Mar 31, 2018.

[21] Alexandre Hirzel. Introduction aux systèmes d'information géographique.[en ligne].[consulté le 12 février 2019].disponible sur https ://www.iumsp.ch/sites/default/files/pdf/introduction_aux_sig_iumsp.pdf. 2014.

[22] What is gis ?.[en ligne].[consulté le 19 février 2019].disponible sur https ://mapcruzin.com/what-is-gis.htm.

[23] Stefano Iacovella. Geoserver beginner's guide. October 2017.

[24] Georezo le portail francophone de la géomatique.[en ligne].[consulté le 14 mars 2019].disponible sur https ://georezo.net/wiki/main/dico/epsg.

[25] Système d'information géographique (sig).[en ligne].[consulté le 07 mars 2019].disponible sur https ://www.notre-planete.info/terre/outils/sig.php.

[26] Mark de Blois. Serveurs cartographiques web. Septembre 2016.

[27] Service web.[en ligne].[consulté le 10 mars 2019].disponible sur https ://fr.wikipedia.org/wiki/service_web.

[28] Introduction aux flux wm(t)s, wfs(-t) et wcs .[en ligne].[consulté le 05 mars 2019].disponible sur https ://www.geomatick.com/2016/12/21/introduction-aux-flux-wms-wfs-et-wcs/.

[29] Web mapping.[en ligne].[consulté le 20 avril 2019].disponible sur https ://en.wikipedia.org/wiki/web_mappingcite_note-kraaksettingsandneeds-2.

[30] Erik hazzard Tomas Gratier, Paul spenser. Open layers 3 beginner's guide. 2015.

[31] Leaflet vs mapbox vs openlayers.[en ligne].[consulté le 03 mars 2019].disponible sur https ://stackshare.io/stackups/leaflet-vs-mapbox-vs-openlayers.

[32] Sql.sh.[en ligne].[consulté le 16 mars 2019].disponible sur https ://sql.sh/sgbd/postgresql.

[33] Djamila Mechbouh Omar El Kharki. Bases de données à référence spatiale postgis : Open source au service de l'information géospatiale.[en ligne].[consulté le 19 février 2019].disponible sur https ://www.researchgate.net/publication/. Octobre 2015.

[34] Pivotal greenplum.[en ligne].[consulté le 19 février 2019].disponible sur https ://en.wikipedia.org/wiki/greenplum.

[35] Mapserver.[en ligne].[consulté le 21 février 2019].disponible sur http ://www.mapgears.com/fr/technologies/mapserver/.

[36] Geoserver.[en ligne].[consulté le 18 février 2019].disponible sur https ://fr.wikipedia.org/wiki/geoserver.

[37] Mapstore.[en ligne].[consulté le 13 avril 2019].disponible sur https ://mapstore.readthedocs.io/en/latest/.

[38] Docker : tout savoir sur la plateforme de containérisation.[en ligne].[consulté le 20 mai 2019].disponible sur https ://www.lebigdata.fr/docker-definition.

[39] Geojson
sur https ://fr.wikipedia.org/wiki/geojson.