



Stage de fin d'études

Cycle des Ingénieurs diplômés de l'ENSG 3ème année

# Développement d'une application de gestion des parcelles

**Chakir Ferdaousse**

Novembre 2015

Non confidentiel

Ecole Nationale des Sciences Géographiques
6-8 Avenue Blaise Pascal - Cité Descartes - 77420 Champs-sur-Marne
Téléphone 01 64 15 31 00 Télécopie 01 64 15 31 0

## Jury

**Président de jury :**

Pierre-Yvves Hardouin, directeur des enseignements de l'ENSG

**Commanditaire :**

Capgemini Technology Services SAS

**Encadrement de stage :**

Jean François Mony senior project manager Capgemini
Adrien Drouet Capgemini
Emmanuel Fritsch ENSG

**Responsable pédagogique du cycle Ingénieur :**

Emmanuel Fritsch, IGN/ENSG/DE/DCAIG

**Tuteur du stage pluridisciplinaire :**

Patricia Parisi, IGN/ENSG/DE/DSHI



## Stage de fin d'étude du 01/06/2015 au 27/11/2015

**Diffusion web :** ENSG

**Situation du document :**

Rapport de stage de fin d'études présenté en fin de 3ème année du cycle des Ingénieurs

**Nombres de pages :** 55 pages

**Système hôte :** LaTeX

**Modifications :**

| EDITION | REVISION | DATE | PAGES MODIFIEES |
|---|---|---|---|
| 1 | 0 | 10/2015 | Création |
| 1 | 1 | 11/2015 | 10-20 |
| 1 | 1 | 11/2015 | 15-46 |

# Sommaire

# Table des illustrations

# REMERCIMENT

*Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.*

*Je tiens à remercier dans un premier temps, toute l'équipe pédagogique de l'Ecole Nationale des sciences Géographiques (ENSG) et de l'école Hassania des Travaux Publics (EHTP).*

*Je tiens à remercier les intervenants professionnels responsables de la formation cycle d'ingénieur, mastère architecture des systèmes d'information géographiques à L'ENSG ainsi que les intervenants professionnels responsables de la formation d'ingénieur d'état, option SIG à l'EHTP.*

*Je tiens à remercier vivement mon maitre de stage, Mr Jean François Mony, Manager au sein du service CSD au sein de l'entreprise Capgemini, pour son accueil, sa disponibilité et le temps passé ensemble.*

*Je tiens à remercier, Mr Adrien Drouet, pour son partage de son expertise au quotidien. Grâce aussi à sa confiance j'ai pu m'accomplir totalement dans mes missions. Il fut d'une aide précieuse dans les moments les plus délicats.*

*Je remercie également toute l'équipe VERDI pour leur accueil, leur esprit d'équipe et en particulier Mr Yosri ERRAMI qui m'a beaucoup aidé à comprendre les problématiques des services cartographiques. Ainsi que Mme Lamiae ATTOUK, qui m'a aidé à m'intégrer dans l'équipe.*

*Mes vifs remerciements également à M Emmanuel Fritsch, mon maître de stage ENSG et professeur à l'Ecole Nationale des sciences Géographiques, pour sa disponibilité et sa coordination.*

*Enfin, je tiens à remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce rapport de stage : ma famille, mes camarades de promotion.*

# RESUME


Dans le cadre de ma formation de double diplôme entre l'école Hassania des Travaux Publics et l'école Nationale des Sciences Géographiques, j'ai été amenée à effectuer un stage de fin d'étude d'une durée de six mois au sein du service CSD Capgemini Nantes.

Ce stage a été une réelle opportunité pour m'intégrer dans le monde professionnel d'une grande structure telle que Capgemini.

Tout au long de mon stage, j'ai eu l'occasion de non seulement mettre en œuvre mes connaissances techniques mais aussi d'améliorer mes capacités de communication et mon relationnel.

Ce stage m'a aussi permis de m'améliorer techniquement en me poussant vers de nouvelles technologies et en m'exposant aux différentes problématiques.

L'organisme d'accueil m'a proposé un sujet, qui m'intéresse particulièrement dans un domaine dans lequel j'aimerai évoluer à savoir le développement d'application SIG. L'essentiel de mon stage s'inscrit dans ma participation au développement d'une application WEB SIG pour l'Agence de Services et de Paiement.

Cependant, j'ai aussi participé à des sujets qui ne manquent pas d'importance que j'évoquerai en détail dans ce document.

Le présent document représente un rapport sur les différents projets auxquels j'ai participé ainsi que sur ma mission sur chaque projet. Il contiendra aussi une description de mon environnement de travail et des différentes technologies utilisées.

Je présenterai aussi dans ce document un bilan des différentes connaissances que j'amélioré ou acquis lors de mon stage.

L'objectif de se stage était de me confronter aux différents challenges et contraintes d'un grand projet au sein d'une grande structure. Et ceci dans le domaine du développement informatique avec une extension des SIG.

# Abstract

During my last year of my engineering training in the National school of geographic sciences in collaboration with Hassania school of Public works, I had the opportunity to do an internship in CSD service Of Capgemini Nantes.

My internship which lasted six months and was a real opportunity to experience professional work environment especially in a big structure such as Capgemini.

During this internship I had not only the chance to improve my technical knowledge already acquired from my college years but also to work on my communication skill.

This internship also helped me to improve myself by exposing me to new challenges and to different problematics.

The main subject of my internship was to participate in the development of a web application with a GIS extension for ASP agency. I was particularly interested by this subject in which I wanted to improve my skills.

Moreover, I had the chance to take part indifferent projects that I will detail in the present document.

This report includes in detail my mission in every project as well as the technology that I used and the description of my work environment.

I will also make an assessment of the skills that i improved and those that I acquired thanks to my internship.

The main purpose of this internship is to confront new challenges and constraints that a big project can presents.

# Terminologies

- **MOA :** Maitre d'ouvrage
- **MOE :** Maitre d'œuvre
- **AMOE :** Assistance au Maitre d'oeuvre
- **ZONE PIAO :** Photo-interprétation des contrôles télédétection
- **Géométries de références :**
- **MXD :** Map Exchange Document
- **SD :** Service Definition
- **SGBD :** Système de Gestion de base de données
- **ETL :** Extract, Transform, Load
- **WMS :** Web Map Service
- **WFS :** Web  Feature Service
- **WFS-T :** Web Feature Service Transactional
- **ETCS :** European Train Control System
- **IHM :** Interface Homme Machine

# INTRODUCTION

Au cours de mon stage de fin d'étude à Capgemini Nantes, j'ai pu participer à trois projets de différentes natures dont le plus imposant est le projet VERDI. Ce projet, constitue les deux tiers de la durée de stage, une durée pendant laquelle j'ai pu intervenir sur plusieurs tâches.

Le projet VERDI a pour MOA l'agence des services de paiement (ASP) qui a pour mission d'apporter de l'accompagnement et de l'aide à la décision au près des décideurs du secteur public.

En tant qu'organisme payeur l'ASP se doit une certaine rigueur dans le fondement des aides versée, chose qu'elle acquiert à travers des opérations de contrôle.

Sur ce point, Capgemini met son expertise au service de l'ASP afin de leur apporter l'intelligence nécessaire pour améliorer, faciliter et automatiser leurs opérations.

En outre, vers la fin de mon stage j'ai participé à un projet d'ETCS au profit de la SNCF.

La SNCF se doit toujours de contrôler des ses données afin de garantir la sécurité de ses clients. Par conséquent, elle prévoit des unités de contrôles pour ses chaînes de production.

C'est dans ce cadre que Capgemini apporte son savoir faire et les ressources dont elle dispose pour accompagner la SNCF dans les opérations de contrôle afin de lui permettre de tenir son engagement envers ses clients.

- **Contexte du stage**

Le stage s'est déroulé à Nantes au sein de l'entreprise Capgemini et plus particulièrement au sein du service CSD. Il a duré six mois du premier juin 2015 au 27 novembre 2015. Pendant, la durée du stage j'ai pu participer à trois projets :

| Projet | Client | Durée |
|---|---|---|
| VERDI : Développent d'application WEB-SIG pour la gestion des parcelles . | ASP | 4 mois |
| Etat de l'art et comparaison de Qgis server et Mapserver | Capgemini | 1 mois |
| ETCS | SNCF | 1 mois |

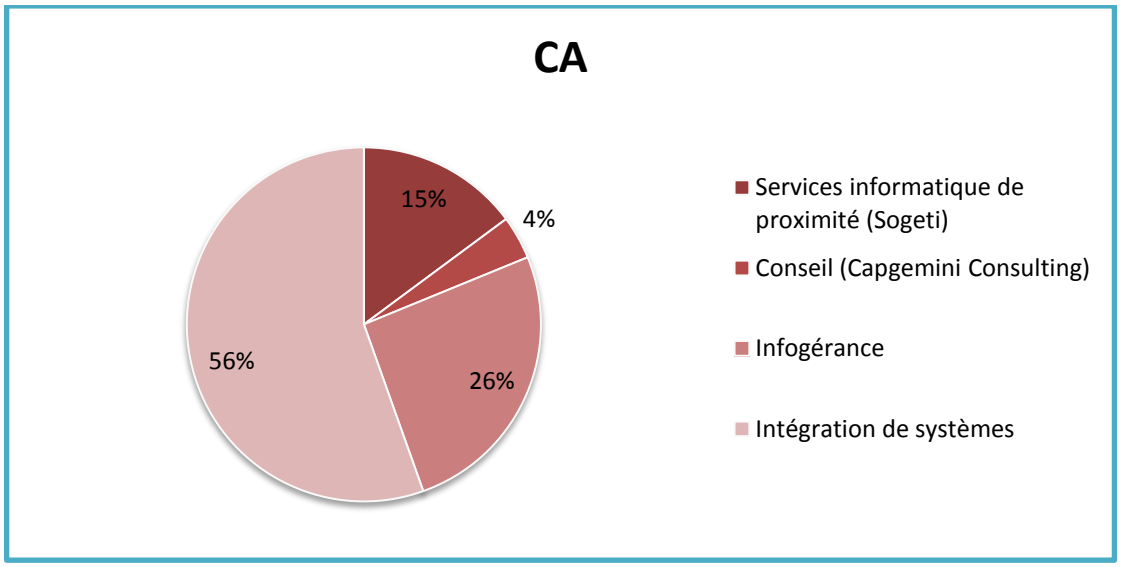
# 1. Présentation de l'organisme d'accueil

Créée par Serge Kampf en 1967 sous le nom de Sogeti, Capgemini fut la première entreprise de services numériques dans le secteur des services informatique en France.

Elle s'est agrandie au fil des années et à travers plusieurs acquisitions pour devenir :

- 1967-1975 : Sogeti,
- 1975-1991 : Expansion du groupe Cap Gemini Sogeti (CGS),
- 2000 : Cap Gemini Ernst & Young (CGEY),
- 15 avril 2004 : le groupe prend sa dénomination actuelle : Capgemini,
- 2005 : Création de la filiale Capgemini Consulting.

Capgemini est une groupe solide présent dans 40 pays et dispose de 10,5 million de chiffre d'affaire répartit comme suit :




Capgemini propose à ses clients un ensemble de prestations pivotant autour de quatre métiers :

- Conseil : Capgemini accompagne les entreprises pour améliorer et consolider leurs performances économiques.
- Intégration : Capgemini aide ses clients à intégrer des systèmes d'information et des applications informatiques qui leur permette d'automatiser, simplifier et moderniser leur travail.
- Services de proximité : qui proposent au sein de l'entreprise un accompagnement et un savoir-faire informatiques ("Local Professional Services" ou LPS).

Capgemini, représente maintenant une entreprise de 244 470 collaborateurs et est présente dans plus de 40 pays. Capgemini agit dans plusieurs secteurs dont nous trouvons :

- Services publics,
- Services financiers,
- Transport et logistique.
- Télécommunications, Média et divertissement

Le groupe Capgemini est fondé sur sept valeurs :

- Honnêteté
- Audace
- Confiance
- Liberté
- Solidarité
- Simplicité
- Plaisir

# 2. Missions

## *1. Mission générale : Concepteur/développeur d'application SIG*

La mission globale du stage consiste à participer dans le développement d'application avec une dimension cartographique. Afin de remplir cette mission Il faut être capable d'intégrer une équipe et de comprendre l'existant.

A travers cette mission j'ai intégré pendant mon stage deux équipes. J'ai été amenée à utiliser plusieurs outils techniques, mettre en œuvres mes compétences ainsi que d'acquérir de nouvelles connaissances.

## *2. Projets*

### 1. Projet 1 : VERDI

#### 1. MOA

L'agence de service de paiement (ASP) est un organisme payeur et un opérateur public. Elle contribue à la mise en œuvre de politiques publiques européennes, nationales et locales.

Elle intervient sur l'ensemble des fonds européens :

- Fonds européen agricole de garantie (FEAGA),
- Fonds européen agricole pour le développement rural (FEADER),
- Fonds européen pour les affaires maritimes et la pêche (FEAMP),
- Fonds européen de développement régional (FEDER),
- Fonds social européen (FSE).

Cette agence dispose d'un grand budget et est classée comme le premier organisme payeur européen d'aides agricoles et le principal organisme payeur français désigné pour gérer le FEAGA et le FEADER

Cet établissement est pluri-ministériel, il mène des missions pour 12 ministères.

L'ASP travaille pour les 26 Conseils régionaux et plus de 80 Conseils départementaux. L'agence a un vaste champ d'activités dont l'Agriculture, la pêche, l'habita…

## 2. Le contexte

L'Agence de Services de Paiement (ASP) est un organisme chargé du paiement des aides de la Politique Agricole Commune (PAC) aux exploitants agricoles français.

Pour cela, elle dispose du Service des Contrôles en Exploitation Surface et Animaux (SCESA) qui effectuent des opérations de contrôle en se basant sur cinq principes :

- la précision et complétude des contrôles,
- l'échange avec les exploitants,
- l'optimisation des campagnes de contrôle,
- la traçabilité des résultats de contrôle,
- l'optimisation des ressources allouées aux contrôles.

Afin de bien remplir sa mission, l'ASP dispose de plusieurs applications, dont les principales liées au projet sont :

- HORUS : orienté contrôle de surface par télédétection ;
- NOMADE : permettant les contrôles surface in situ, associé à ArpentGIS Mobile pour l'acquisition GPS ;
- PACo : pour la gestion des campagnes et des ressources ;
- ISIS : support de déclaration, puis de valorisation des aides.

Cependant, L'ASP a ressenti le besoin de moderniser l'outil du contrôle surface (HORUS,NOMADE,ArpenGIS) ainsi que les outils de contrôles d'animaux . C'est dans ce contexte, que s'inscrit le projet Verdi qui à travers lequel Capgemini accompagnerai L'ASP à satisfaire le besoin ressenti.[1]

### ➢ Utilisateurs

L'application VERDI aura pour utilisateurs :

[1] D'après le document de la conception techniques de VERDI « DCT »

- Les agents de la direction générale,
- Les agents du SCESA,
- Les agents des services de contrôles dans 26 Directions Régionales (DR) de l'ASP

## 3. L'équipe

Outre le chef de projet, position occupé par monsieur Jean-François MONY, l'équipe travaillant sur le projet est divisée en deux équipes :

i) L'équipe de Limoge :

Cette équipe est constituée de membres de profils fonctionnels capables de coordonner avec le client pour rédiger les spécifications du projet.

ii) L'équipe de Nantes :

Cette équipe est constituée de douze membres dont :

- 9 Développeurs
- 1 responsable technique
- 1 architecte
- 1 chef d'équipe

Les deux équipes sont chapotées par un chef de projet en la personne de monsieur Mony Jean François.

## 4. Architecture de VERDI

Le projet Verdi est constitué principalement de l'application client lourd « Verdi » avec un branchement sous forme d'application web.

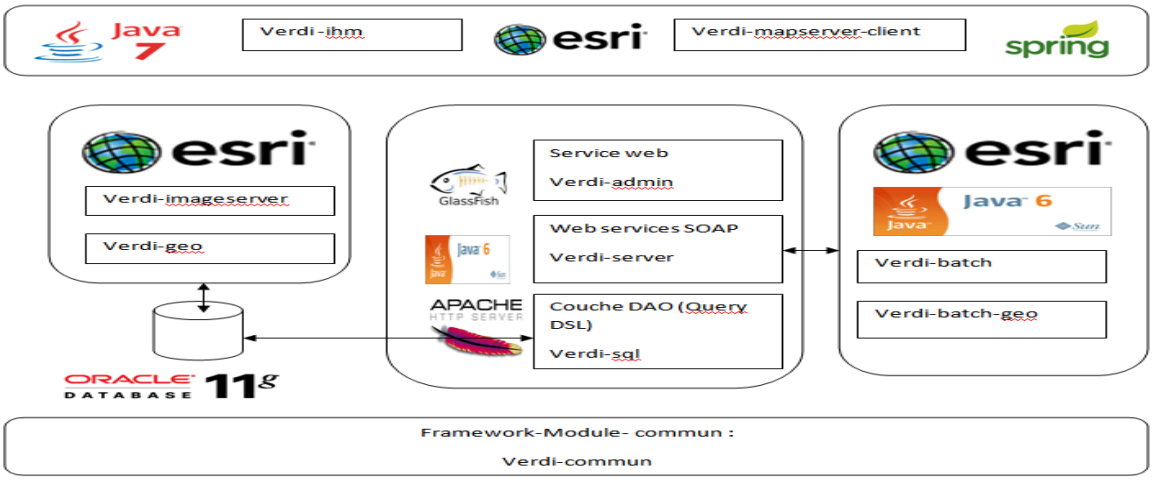



Figure 1-Architecture le l'application VERDI

Verdi est une imbrication de neuf modules :

- VERDI WEBSTART : Ce module contient le code nécessaire pour implémenter l'installation du client lourd en utilisant la technologie Java Web Start.
- VERDI IHM : c'est le module du déploiement de l'IHM du client lourd avec la technologie Java Swing.
- VERDI BATCH : c'est le module contenant tous les batchs du déploiement des services cartographiques de VERDI.
- VERDI SERVEUR : Dans ce module, nous trouvons tous les web services implémentés en SOAP du client lourd.
- VERDI SQL : Ce module représente la couche d'accès aux données stockées dans des bases de données ORACLE.
- VERDI GEO : Ce module contient tout ce qui concerne les services cartographiques de VERDI.
- VERDI ADMIN : Ce module représente l'application web client rapatriées à VERDI. Ce rapport se focalisera essentiellement sur ce module.
- VERDI COMMUN : Ce module contient le Framework du client lourd, les contrats de services (endpoints) et l'implémentation des services de règles.

# 5. Application Web « Gestion des géométries de références »

## 1. Introduction

L'application web de la gestion des géométries de références constitue un autre volet du projet VERDI. Cette application est dédiée à la gestion et à l'utilisation des géométries de références par les profils Superviseur, Superviseur Prestataires, et assistance utilisateur locale ou national (cf utilisateurs VERDI).

Cette application remplit deux fonctionnalités phares :

- Gestion du référentiel des géométries de référence PIAO et ceci à travers la création, la modification, la suppression des géométries de références ainsi que leurs attributs.
- Consultation des géométries de référence lors des contrôles PIAO.

L'application est uniquement accessible au sein du réseau de l'ASP.


Figure 2-Application de gestion de parcelles

## 2. Décomposition fonctionnelle

Fonctionnellement l'application est décomposée en seize cas d'utilisations décrivant l'expérience d'utilisation de l'application.

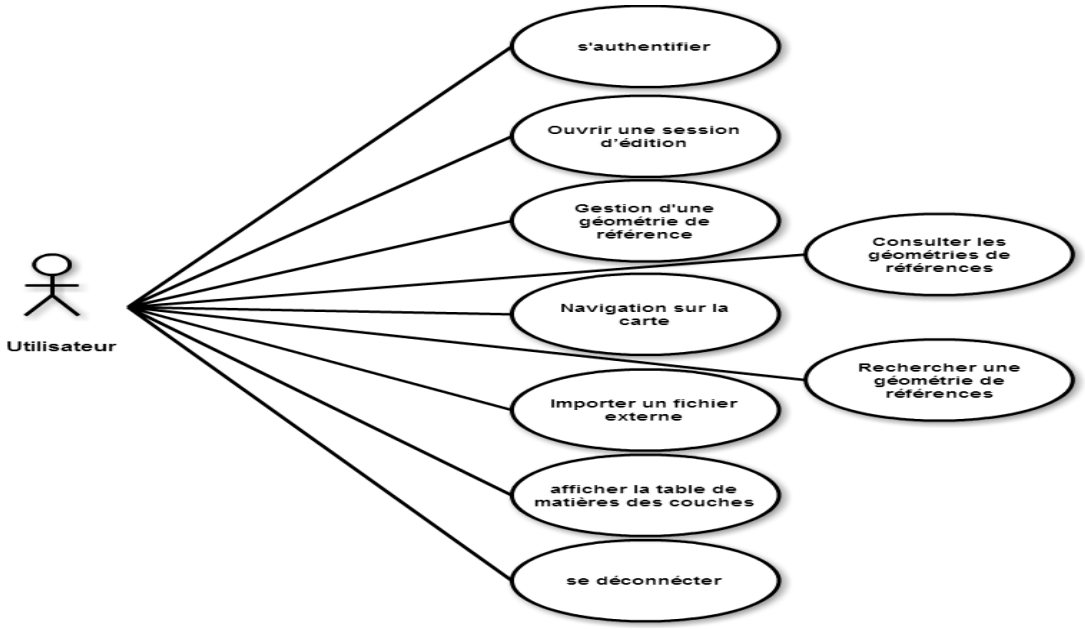



Figure 3-Diagramme des cas d'utilisation de l'application de gestion des parcelles

## 3. Outils

### 1. Techniques[2]

L'application web est essentiellement développée par les langages de programmation web, à savoir :

- **Javascript** : c'est un langage de programmation pour des pages web interactives. Javascript a été au début utilisé côté client, cependant nous le retrouvons maintenant côté serveur aussi avec des logiciels comme « NodeJs ».Ce langage est orienté objet. Ses objets sont équipés de constructeurs permettant de créer leurs propriétés.

- **HTML** : L'Hypertext Markup Language, c'est le format de données des pages web. Il est conçu d'un ensemble de balises permettant d'écrire de l'hypertexte. Il permet de structurer les pages web et d'y insérer différentes ressources multimédia. Il permet aussi de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web. Ce langage est souvent utilisé avec d'autre langage de programmation web notamment Javascript et PHP ainsi qu'avec des feuilles de style (CSS).

- **CSS** : Les feuilles de style en cascade, c'est un langage informatique qui décrit la présentation des documents HTML et XML. Les standards définissant CSS sont publiés par le World Wide Web Consortium (W3C). Introduit au milieu des années 1990, CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web dans les années 2000.

Cependant, Pour le développement des web services, les développeurs ont utilisé JAVA.
En particulier, l'application s'est reposée sur deux outils :

- **Dojo** : Dojo est une bibliothèque logicielle open source en JavaScript. Son but est le développement rapide d'applications en Javascript exécutées côté client et communiquant avec le serveur avec une granularité inférieure à la page grâce à Ajax. Dojo est sous double licence et peut donc être obtenue soit selon les termes de la licence BSD, soit sous ceux de l'Academic Free License.

  Il fournit un système de paquetage conçu pour faciliter la séparation des fonctionnalités en paquetages et sous-paquetages individuels. Le script de base pour le bootstrap de Dojo initialise un espace de nom hiérarchique appelé "dojo". Après son initialisation, une fonction utilitaire peut charger tout paquetage via XMLHttpRequest ou un autre système de transport de données. Pour le code développé avec Dojo, on peut initialiser des espaces de noms parallèles à "dojo".

- **ArcGIS Api for Javascript** : est une solution proposée par ESRI pour incorporer la fonctionnalité cartographique allégée dans une application Web. Elle est gérée par un API REST en arrière-plan, capable de récupérer des informations en mode sans état sur le serveur. Lorsque vous lancez l'application, le code est exécuté dans le navigateur au lieu de recourir au serveur. Cette particularité permet une utilisation client rapide et propre. L'ArcGIS API for JavaScript repose sur le kit JavaScript Dojo : vous n'avez donc pas à vous soucier de l'adaptation de particularismes de navigateur dans votre code ; l'infrastructure les gère automatiquement.

- **GlassFish** : est le nom du serveur d'applications Open Source Java EE 5 et désormais Java EE 7 avec la version 4.1 qui sert de socle au produit Oracle GlassFish Server1 (anciennement Sun Java System Application Server2 de Sun Microsystems). Sa partie Toplink persistence3

[2] Wikipédia « https://fr.wikipedia.org »

provient d'Oracle. C'est la réponse aux développeurs Java désireux d'accéder aux sources et de contribuer au développement des serveurs d'applications de nouvelle génération.

- **Maven** : un outil pour la gestion et l'automatisation de production des projets logiciels Java en général et Java EE en particulier. L'objectif recherché est comparable au système Make sous Unix : produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication.

En ce qui concerne les services cartographiques, nous utilisons les outils suivants :

- **ArcGis Serveur** : c'est le serveur cartographique d'ArcGIS. Ce serveur permet la publication des services web et supporte le WMS, WFS, WFS-T, WMTS. ArcGIS permet la gestion du serveur à travers l'ArcGIS Manager qui présente une interface des services publiés.
- **ArcMap** : l'application où vous affichez et explorez les jeux de données SIG pour votre zone d'étude, où vous attribuez des symboles et où vous créez des mises en page de carte en vue de l'impression ou de la publication. ArcMap est aussi l'application que vous utilisez pour créer et modifier des jeux de données.
- **Arcpy** : est un site-package Python qui offre un moyen utile et productif d'effectuer l'analyse, la conversion et la gestion de données géographiques, ainsi que l'automatisation de cartes avec Python.

En outre de ces outils, nous trouvons aussi pour l'environnement du développement :

- **Eclipse** : est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la Fondation Eclipse visant à développer un environnement de production de logiciels libre qui soit extensible, universel et polyvalent, en s'appuyant principalement sur Java.

Les données sont intégrées dans des bases ORACLE par l'outil :

- **Oracle SQL développeur** : est un environnement de développement intégré (EDI) multiplateforme, fourni gratuitement par Oracle Corporation et utilisant la technologie Java (Java Development Kit). C'est un outil graphique permettant d'interroger des bases de données Oracle à l'aide du langage SQL.

Pour assurer la collaboration entre les différents développeurs et gérer les différentes versions de l'application, L'équipe a opté pour l'utilisation de L'outil de versioning :

- **Subversion**: (en abrégé svn) est un logiciel de gestion de versions, distribué sous licence Apache et BSD. Il a été conçu pour remplacer CVS. Ses auteurs s'appuient volontairement sur les mêmes concepts (notamment sur le principe du dépôt centralisé et unique) et considèrent que le modèle de CVS est bon, seule son implémentation est perfectible. Subversion fonctionne donc sur le mode client-serveur, avec :
  - un serveur informatique centralisé et unique où se situent :
    - les fichiers constituant la référence (le « dépôt » ou « référentiel », ou « repository » en anglais),
    - un logiciel serveur Subversion tournant en « tâche de fond » ;
  - des postes clients sur lesquels se trouvent :
    - les fichiers recopiés depuis le serveur, éventuellement modifiés localement depuis,

un logiciel client, sous forme d'exécutable standalone (ex. : SmartSVN) ou de plug-in (ex. : TortoiseSVN, Eclipse Subversive) permettant la synchronisation, manuelle et/ou automatisée, entre chaque client et le serveur de référence

## 6. Architecture

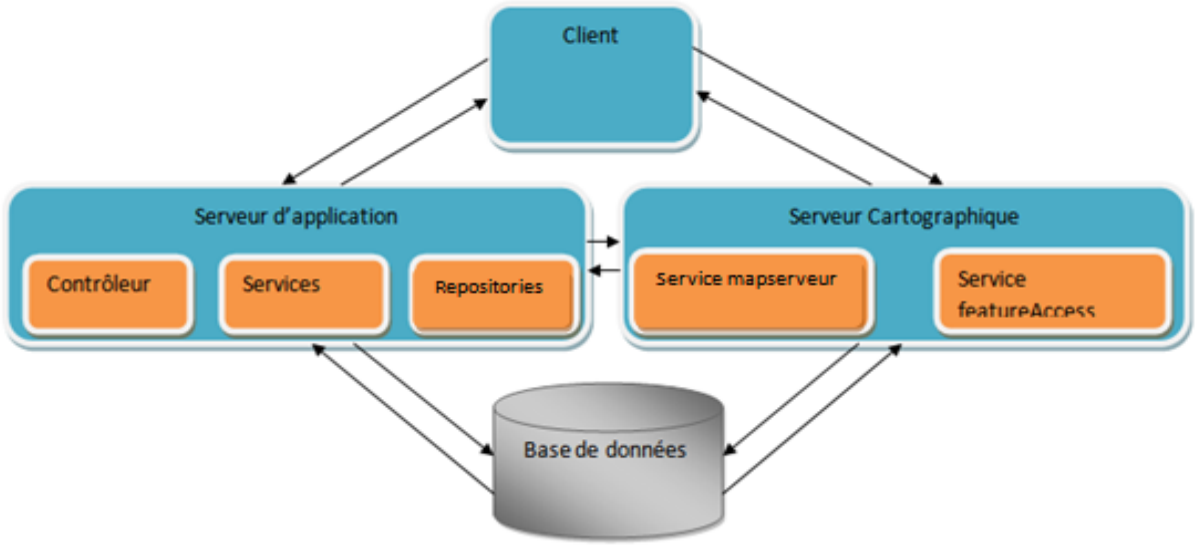



Figure 4-Architecture de l'application de gestion des parcelles

Pour certains navigateurs qui ne supportent pas « the cross call ajax domain », Il a été nécessaire d'utiliser un proxy.

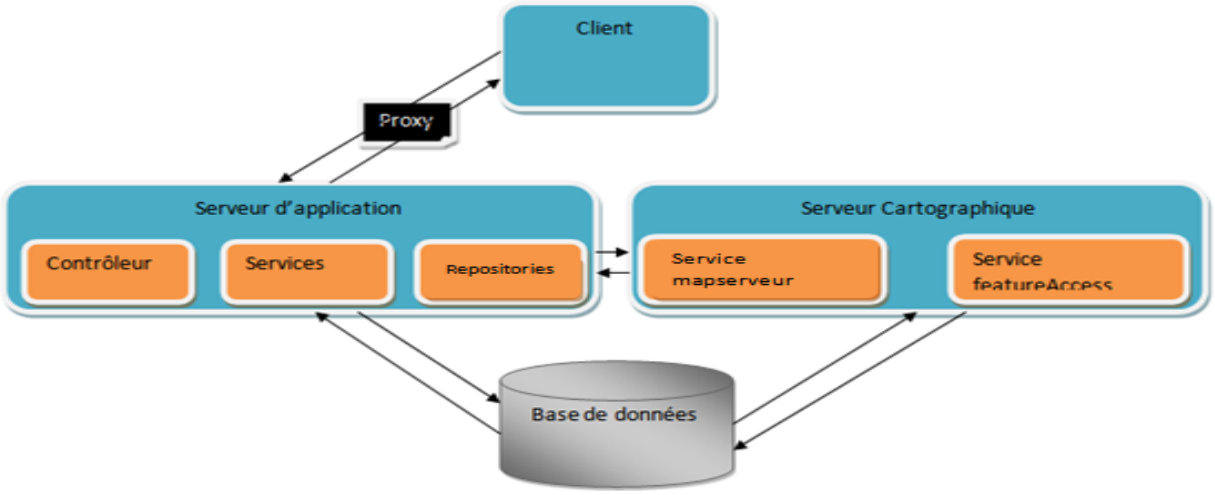


Figure 5-Architecture de l'application de gestion des parcelles

# 7. Gestion de projet

L'équipe fait un point chaque matin ou chaque développeur donne un compte rendu de ces activités en cours et celle à venir. Pendant ce point l'équipe peut également évoquer d'autre sujet comme par exemple le compte rendu des réunions avec la MOA ou bien les décisions prises concernant l'ensemble de l'équipe.

A fin de renforcer la cohésion entre l'équipe de Nantes et celle à limoges les développeurs reste connectés pendant toute la journée du travail en vidéo conférence grâce au logiciel Skype.

Il est aussi demandé de chaque développeur d'envoyer un mail du « Reste à faire » dans lequel il chiffre en journée le temps nécessaire pour achever ses activités.

La combinaison de ces méthodes permet d'avoir plus de visibilité sur le travail de chaque développeur et aide également le chef de projet à bien répartir les tâches sur les ressources du projet.

# 8. Mission

## 2. Déroulement de la mission


Ma mission se décompose principalement en quatre étapes **:**

- Découvrir l'existant :

A fin d'être le plus efficace possible sur le projet, Il m'a été primordial de découvrir l'existant. Ceci à été fait à travers la lecture des spécifications, ainsi que les différents diagrammes expliquant l'architecture de l'application, le modèle conceptuel des données ainsi que les différentes solutions techniques utilisées.

Lors de cette phase, j'ai pu me documenter sur toutes les solutions techniques qui ne m'étaient pas été familières. En outre, cela m'a donné l'occasion de découvrir la méthodologie du travail de l'équipe.

Cette étape à faciliter mon intégration et m'a permise d'appréhender plus facilement dans le code.

- Préparation des mxds et génération des SD :

Une fois la première étape dépassée, et grâce à ma formation en système d'information géographique, j'ai été dés le départ chargée des tâches cartographiques, à savoir la préparation des mxds afin de générer des fichiers « SD » pour la publication des services cartographique de l'application.

L'outil utilisé lors de cette étape est le logiciel ArcGIS, un logiciel que j'ai eu l'occasion de manipuler au cours de ma formation.

La génération des fichiers SD a été faite par des scripts python dans les quels j'ai utilisé en outre des fonctions proposées par la bibliothèque Arcpy, des fonctions développées par l'équipe du projet.

➢ Participation au développement :

Cette étape constitue la plus grande partie de mon stage, Durant cette phase j'ai pu développer différentes fonctionnalités :

- Blocage de la carte sur une extent d'une zone PIAO
- Outils de navigation : Boutons d'historique de navigation, bouton de pan, Liste déroulante d'échelle
- Table de matière des couches
- Filtrage de la carte
- Consultation d'une géométrie depuis l'onglet de recherche
- Calcul du milieu d'un polyline

### 3. Correction des anomalies :

Cette étape est la dernière phase de ma mission, et dans laquelle j'ai corrigé les anomalies de développement remontées par le client ou par l'équipe des testeurs, mon intervention ne se limite pas qu'aux fonctionnalités que j'ai développées mais aussi sur d'autres fonctionnalités développées par d'autres membres de l'équipe.

### 4. Services cartographiques

1. Les fichiers « .mxd » :

La mission principale de l'application web consiste à gérer les géométries de référence et leurs attributs que ce soit par création, modification, suppression ou tout simplement par consultation. Afin de remplir cette mission, Il fallait déployer des services cartographiques dans lesquels nous avons publié les couches des géométries de références ainsi que leurs attributs.

Le déploiement des services cartographiques commence par la préparation des mxds après la génération via un script python des SDs et finalement publier les SDs générés via ArcGis Manager

.

Ma tâche consistait à faire toute la chaîne décrite ci-dessus afin d'avoir des services cartographiques opérationnels pour l'application web.

➢ **Préparation des mxds :**

L'application web utilise deux services cartographiques :

- « GeometrieReference » : ce service est de type « feature Access » .Le choix d'activation de l'option « feature Access » pour ce service, vient du besoin de modifier les objets géographiques des couches publiées dans ce service. Dans ce service, nous avons publié deux couches :

  - COUCHE_GEOM_REF_POLYGONE : la couche des géométries de références de types polygones
  - COUCHE_GEOM_REF_POLYGNE : la couche des géométries de références de types ligne.

❖ « PhotoRéférences » : Ce service est de type « mapserver » et publiant les couches :

- COUCHE_PHOTO_GEOM_POINT : la couche représentant les photos associées aux géométries de références
- REF_COUCHE_ZONE_PIAO : La couche représentant les zones PIAO.

Pour des besoins fonctionnelles de l'application web, nous avons besoin d'avoir pour les photos à la fois la méta données sur la photo, la Géométrie référence à laquelle elle est associée ainsi que la géométrie du point représentant la photo sur la carte. Ces informations sont stockées dans trois différentes tables, par conséquent, il fallait faire une jointure entre les trois tables afin de pouvoir récupérer toutes ces informations par une simple requête sql.

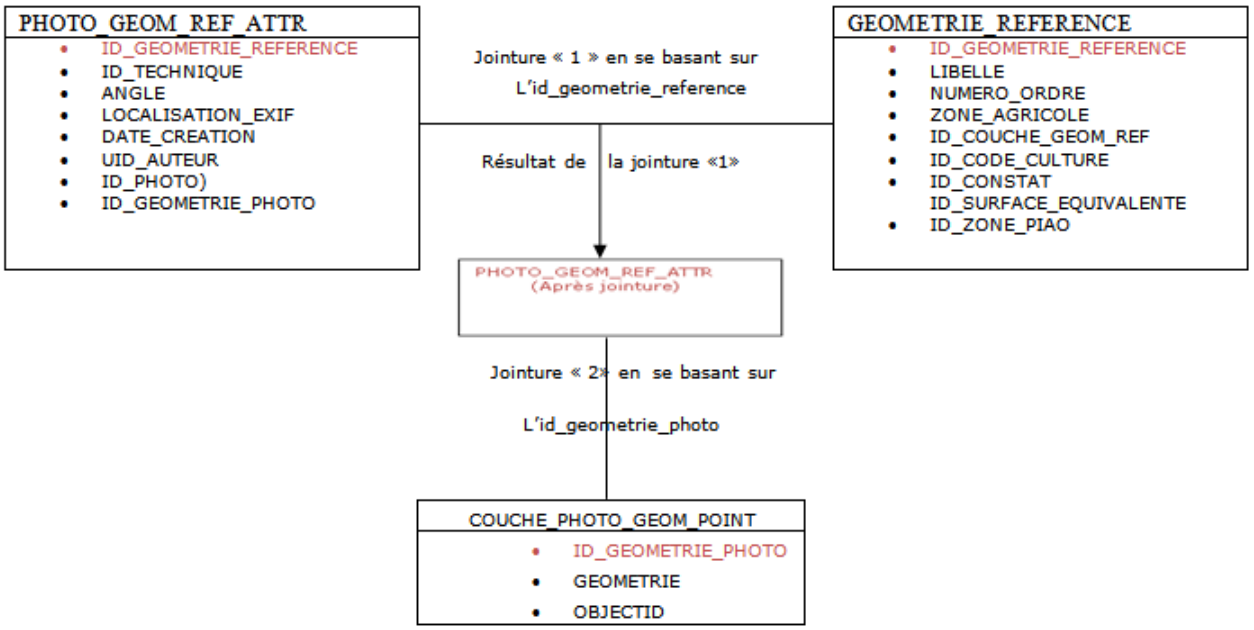



**Figure 6-Jointure de la couche des photos**[3]

Idem, Pour les Zones Piao nous avons besoin de récupérer à la fois la géométrie des zones Piao ainsi que leurs codes, libellés et d'autres informations. Ces informations, sont stockés dans deux tables et donc il faut faire une jointure entre ces deux tables la.

[3] Depuis le DCT de VERDI

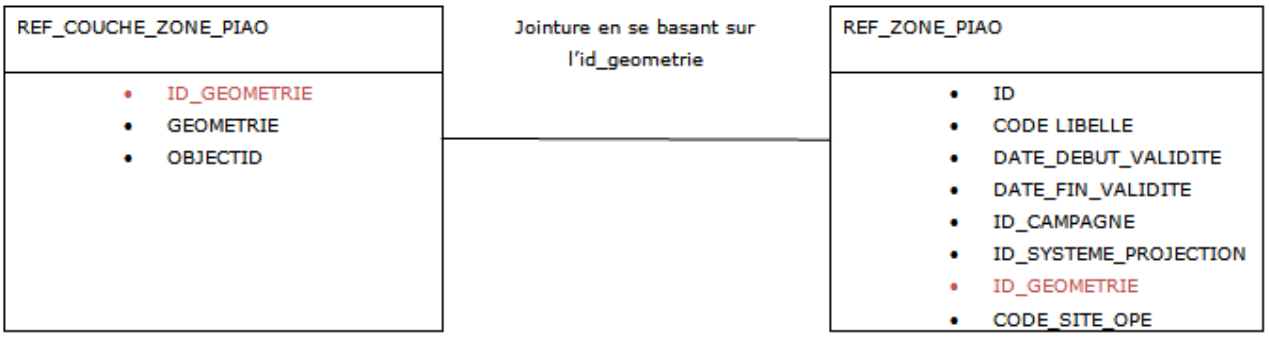



**Figure 7-jointure de la couche des zones PIAO**[4]

Ces jointures sont crées avec un script python. Avec le logiciel ArcMap, nous avons créé deux mxds, Le premier concernant le service « GeometrieReference » contient les couches des géométries références de type polygones et la couche des celles de type polyligne. Cependant, celui du service « PhotoRéférences » nous avons opté à le laisser vide et d'ajouter les couches à travers le script python.

### 2. Génération des fichiers « .sd »

Un fichier « .sd » est un fichier qui contient des informations sur les propriétés, les fonctionnalités et le type de service, qui sont encapsulées dans un fichier portable. Vous pouvez également configurer le fichier de définition de service pour y inclure les données référencées par votre ressource SIG.

Afin d'industrialiser le processus de publications des services cartographique, l'équipe de VERDI à mis en place des scripts python et des fichiers « .xml » et des fichiers geojson capables de communiquer entre eux afin de générer les fichiers « .sd » correspondants aux profils de connexion et l'environnement d'installation.

- Pom.xml : Ce sont des fichiers Maven dans lesquels, nous renseignons les paramètres de connexion à la base de données, les chemins vers les répertoires.
- .json : Il y a un fichier pour chaque service, et dans lesquels il y a les paramètres du service à savoir, le type de service.
- .py : ce sont les scripts pythons qui génèrent les fichiers « .mxd » et dans lesquels, nous créons les fichiers « .SD » grâce à la fonction « arcpy.mapping.AnalyzeForSD (sddraft) » qui prend en paramètre un fichier « .sddraft » créé par la fonction « arcpy.mapping.CreateMapSDDraft »

### 3. Symbologie

Nous distinguons trois types des photos affichées sur la carte de l'application web :

- Les photos non géolocalisées : qui n'ont pas de données « EXIF »,
- Les photos non orientées : qui n'ont pas de donnée « Angle »,

[4] Depuis le DCT de VERDI

❖ Les photos géolocalisées et orientées : qui ont des données « EXIF » et un « Angle » d'orientation.

Cette distinction est représentée par 3 types de symboles imposés par la MOA.

❖ Les photos non géolocalisées : ces photos sont représentées par un cercle jaune.
❖ Les photos non orientées : ces photos sont représentées par un losange vert
❖ Les photos géolocalisées et orientées : ces photos sont représentées par une flèche verte et dont l'orientation dépend de la valeur de l'angle de la photo.
❖

Comme expliqué dans la partie de la génération des SDs après changement des paramètres de la connexion nous perdions toute la symbologie définie dans le MXD. Pour remédier à ce problème, nous avons créé deux fichier « .lyr » ,le premier permet de changer le type de la symbologie des couches de « OTHER » qui est un type non supporté par ESRI à « UNIQUE VALUES » pour pouvoir appliquer la symbologie contenue dans le Deuxième fichier « .lyr » l'appliquons via la fonction « .. » de la bibliothèque Arcpy sur la couche résultante des jointures.

## 5. Participation au développement

### 1. Blocage de la carte sur une extent d'une zone PIAO

Après authentification, l'utilisateur choisit une zone PIAO parmi, les zones proposées dans une liste déroulantes.

Les zones PIAO proposées dans la liste dépendent du profil d'authentification. Pour ce fait, il fallait qu'une fois arrivé sur une zone PIAO le contrôleur ASP ne peut pas naviguer sur une autre zone qui peut ne pas lui être autorisée. Par conséquent, ce besoin fonctionnel s'est traduit par le blocage de la carte sur l'extent de la géométrie de la zone PIAO choisie après l'authentification.

Pour cela il fallait :

- Récupérer l'identifiant de la zone PIAO choisi
- Récupérer l'extent de cette zone PIAO
- Instancier la carte sur l'extent de cette zone
- Détecter tout événement causant le changement d'extent de la carte
- Traiter chaque changement de la carte pour l'autoriser ou le bloquer s'il cause de dépassement de l'extent permise.

En ce qui concerne la première étape, qui consistait à récupérer l'identifiant de la zone PIAO, est ceci à travers un web service qui lui interroge la base de données pour récupérer les informations concernant les zones PIAO autorisées à cet utilisateur.

Cet identifiant, je l'utilise par la suite dans la requête envoyée au serveur cartographique pour récupérer la géométrie de la zone PIAO correspondante à l'id.

La requête est envoyée au service cartographique « PhotoRéference » de type « Mapserveur » contenant la couche des zones PIAO en jointure avec la table contenant la méta-data sur la zone PIAO « cf p. service cartographique ». Cette requête est exécutée avec l'objet « QueryTask » qui prend comme

paramètre l'url du service cartographique. En instanciant cet objet avec l'url du service «PhotoRéference » j'utilise la méthode « executeForExtent(query, callback?, errback?) » qui prend comme paramètres :

- Requête : la requête SQL à exécuter sur la couche appelée depuis le WebService
- Callback : la fonction à appeler une fois l'événement « execute-for-count-complete » est terminée.
- Errback : la fonction à appeler si une erreur se produit.

Cette requête permet de récupérer directement un objet « extent » de la géométrie résultante de la requête passée en paramètres. cet objet a comme attributs (xmin,xmax, ymin,ymax) qui représentent les bornes délimitant la zone PIAO en question.

Une fois j'ai récupéré l'extent, Je l'utilise pour définir l'extent initiale de la carte.

A ce stade, l'utilisateur arrive bien sur la zone concernée mais il peut en sortir en utilisant les outils de navigation de la carte (Zoom, pan…).Donc, il fallait détecter tous ces événement qui peuvent causer le dépassement de la zone permise.

Pour cela j'ai utilisé les deux événements de la carte :

- Map.on (pan-end) : c'est l'événement détectant, la fin du « pan » de la carte, comme tout événement, il prend comme paramètre la fonction de « call back » , qui dans notre cas est la fonction qui contrôle l'extent de la carte.
- Map.on (zoom-end) : c'est l'événement détectant la fin du zoom sur la carte

Dans ces deux événements j'appelle une fonction nommée « valideExtent » qui prend comme paramètre l'extent résultante de l'événement. J'ai écris cette fonction afin de rétablir la carte à l'extent valide en cas de dépassement.

Le corps de cette fonction permet d'effectuer le traitement suivant :

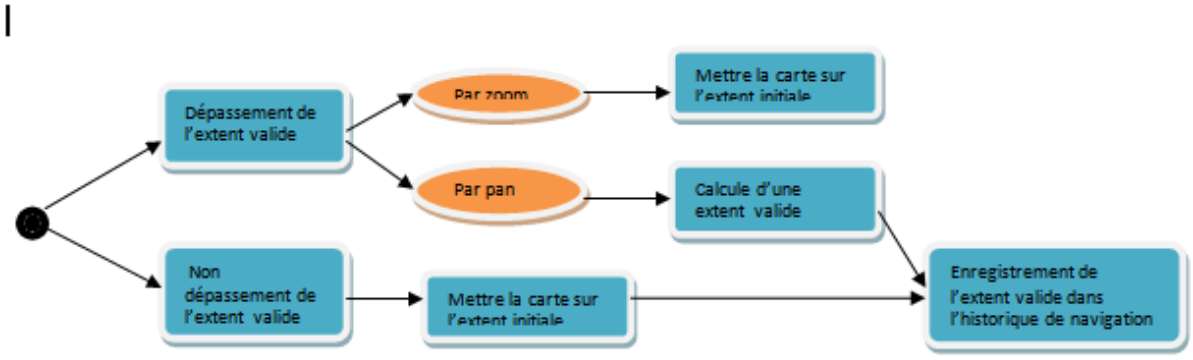



Figure 8-Gestion d'extent de la carte

- Extent initiale : c'est l'extent initiale de la carte c.-à-d. l'extent de la zone PIAO,
- Extent valide : c'est une extent qui est identique ou contenue dans l'extent initiale de la carte.

Le test du dépassement de l'extent valide, est fait grâce à la fonction « extent1.contain(extent2) »de l'objet « extent » qui renvoie un booléen égale à vrai si l'extent2 est incluse dans l'extent1.

La redéfinition de l'extent de la carte est possible avec la fonction « setExtent (extent) » de l'objet « Map ».

En ce qui concerne le calcul de la nouvelle extent valide, il est fait en deux étapes :

- Etape 1 : calcul des décalages dx et dy en faisant la différence entre les xmin,xmax,ymin,ymax des deux extents.
- Etape 2 : annulation de ce décalage grâce à la fonction « offset (dx,dy)) de l'objet « extent ». Cette fonction décale l'extent suivant l'axe X de dx et suivant l'axe Y de dy. Je passe comme paramètres à cette fonction l'inverse du décalage calculer pour compenser le dépassement

Une fois l'extent valide est calculée, je redéfinie l'extent de la carte à cette extent .Ensuite, j'enregistre cette extent dans l'historique des extents valides qui sera expliqué dans le paragraphe suivant.

### 2. Outils de navigation

Pour rendre l'utilisation de l'application plus agréable pour l'utilisateur, il a été nécessaire de mettre à sa disposition certains outils qui facilitent sa navigation sur la carte. Parmi ces outils :

- Les deux boutons précédent et suivant :

Ces boutons permettent à l'utilisateur de basculer en amont et en aval entre ses dix derniers déplacements de la carte et qui sont contenus dans sa zone PIAO.

Afin de réaliser ce fonctionnement, j'ai opté à enregistrer l'extent de chaque déplacement valide dans un tableau de taille égale à dix. J'enregistre en parallèle de cela dans une variable « index » l'indice de l'extent actuelle de la carte dans le tableau. Cette variable me permet de basculer entre les extents du tableau en fonction du bouton cliqué. Si l'utilisateur fait un onzième déplacement, je l'enregistre dans la dixième position dans le tableau et je supprime la première extent afin de garder toujours dans le tableau les dix plus récents déplacements de l'utilisateur.

Si l'utilisateur clique sur le bouton précédent jusqu'à atteindre la première extent dans le tableau, je désactive le bouton précédent. Dans le même logique s'il clique sur le bouton suivant jusqu'à atteindre la dernière extent dans le tableau, je désactive le bouton suivant.

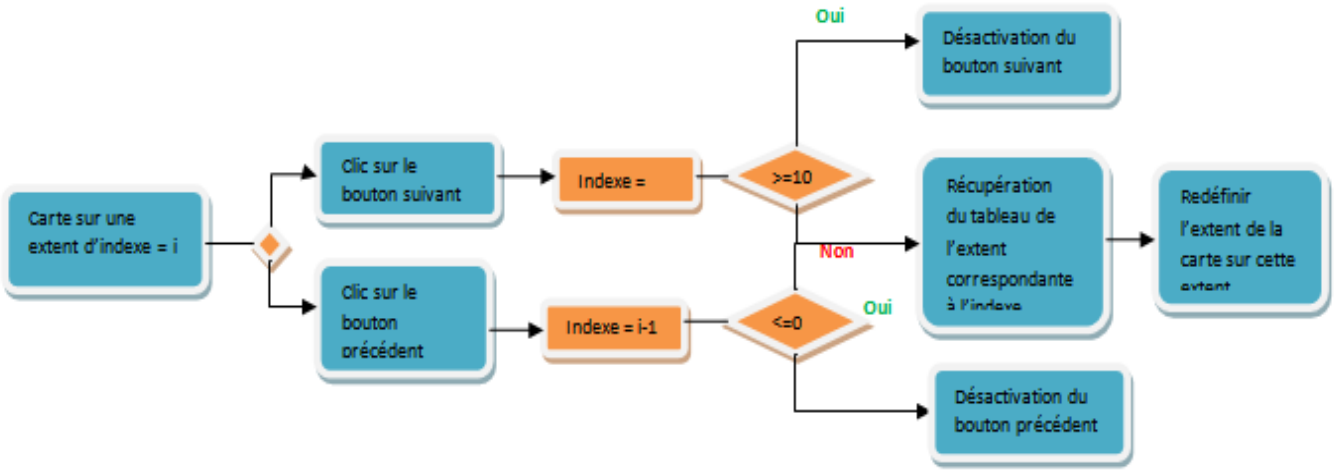



Figure 9-Historique de navigation

- Le bouton du pan :

L'application permet à l'utilisateur de créer et éditer des géométries .Par conséquent, le client a jugé préférable que lors de ses actions l'utilisateur puissent activer ou désactiver le pan sur la carte suivant ses besoins.

D'où la création dans boîte à outil de l'application le bouton du pan. Ce bouton permet d'activer ou de désactiver l'action du pan sur la carte. Ceci grâce à la fonction « disablePan () » de l'objet « Map » de l' «API ArcGIS for JavaScript » pour la désactivation et la fonction « enablePan () » pour l'activation du pan. Ces deux fonctions son appelées par l'événement « onClic () » sur le bouton.

- La liste déroulante des échelles :

Il s'agit d'une liste déroulante contenant les différentes échelles définies dans les « lods » de la carte. Cependant, la carte utilisée initialement dans l'application ne contenait pas des « lods ».Par conséquent, il fallait avant tout définir les niveaux de détail de la carte de l'application. Ceci a été possible avec « … » à qui j'ai passé comme paramètre un tableau contenant les différentes échelles imposées par le client.

La liste déroulante doit à tout moment afficher l'échelle actuelle de la carte. En définissant les « lods » de la carte, nous avons limité les échelles et les niveaux de zoom à ce qui a été défini.

### 3. Table de matière des couches

Cet outil sert à afficher toutes les couches de la carte à l'instant t que ce soit des couches de base ou importées par l'utilisateur.

Elle propose les fonctionnalités suivantes à l'utilisateur :

- Visibilité sur les couches présentes sur la carte
- Activer/ désactiver la visibilité d'une couche

Cependant, L'outil doit être en cohérence avec les autres outils de l'application qui manipule la visibilité des couches.

La Template utilisée pour l'application propose déjà un exemple de table de matière. Je me suis donc appuyée sur ce Template pour développer l'outil.

Dans un premier lieu, j'ai commencé par récupérer tous les noms des couches toujours présentes dans la carte depuis le fichier de configuration « default .js ».Une fois cela est fait, il fallait que je les affiche dans la fenêtre de l'outil en respectant l'ordre imposé par le client et en séparant les fonds de carte des autre couches.

La deuxième partie du développement de l'outil consistait à manipuler la visibilité des couches. L'information de la visibilité est contenue dans l'attribue booléen « visible » des couches. Cet attribut est en lecture / écriture. Par conséquent, il me suffisait de changer la valeur cet attribut en cochant ou en décochant la couche pour avoir le comportement voulu.

La troisième partie du développement consistait à intégrer les couches importées par le contrôleur dans la table de matière tout en respectant l'ordre imposé par le client. En effet, Il voulait qu'on a en premier les couches de type point, en suite, celles de type ligne suivies par celles type polygones et en dernière position les fonds de carte. A chaque fois que l'utilisateur importe une couche, il fallait savoir quel type de géométrie contienne-t-elle ? et actualiser la table de matière pour l'intégrer. Pour cela, j'ai écris une fonction qui teste le type de la géométrie, une information contenue dans l'attribut « geometryType » de l'objet « Layer » de l'API ArcGIS pour JavaScript. Pour l'insertion, de la couche j'utilise la même fonction que j'ai écrit pour insérer les couches de bases. (Explication du code de cette fonction).

L'outil de l'application qui permet la consultation d'une géométrie manipule la visibilité des couches lui aussi. Une fois cliquant sur une des miniatures de la géométrie, il fallait rendre le fond de carte associé visible et rendre invisible tous les autres fonds de carte. Pour assurer ce comportement, j'ai ajouté une fonction qui actualise la table de matière et qui est appelée par l'événement du clic sur la miniature.

### 4. Filtrage de la carte

Cette fonctionnalité permet de filtrer les géométries contenues dans la couche «Couches_GEOM_REF » pour ne garder que celle appartenant à la zone PIAO sur laquelle le contrôleur est. Cela est surtout pour augmenter la performance de l'application et augmenter sa réactivité.

Afin de filtrer la carte j'ai :

- Je récupère tous l'id des objets qui existent dans chaque couche, pour la zone PIAO en question, grâce une requête envoyer au web service,
- Je récupère les géométries des objets dont l'identifiant figure dans le tableau des identifiants récupérés précédemment,
- J'ajoute à la carte les géométries récupérées.

Cette solution quoiqu'elle améliore la performance de l'application, elle n'est pas la solution la plus optimale. Ce qui sera plus judicieux est de filtrer au moment du chargement de la couche depuis le WebService pour ne récupérer que les géométries appartenant à la zone PIAO du contrôleur .Ceci, est

possible grâce à l'option « ON_DEMAND » de l'appelle du WebService et qui filtre sur l'extent dans les requêtes envoyées au WebService.

### 5. Calcul du milieu d'un polyline

En associant une photo à une géométrie, il fallait la positionner au centre de cette géométrie .Dans le cas où la géométrie est un polygone

La localisation du centre est beaucoup plus facile grâce à la fonction « .3 » de l'objet « extent » de l'api d'ArcGIS. Cependant pour une ligne, L'api d'ArcGIS ne propose aucune fonction permettant de localiser le milieu d'une ligne. Dns, j'ai du écrire ma propre fonction.

La détermination du milieu se fait suivant le processus suivant :

- Calculer la longueur de la ligne, L
- Calculer la moitié de cette longueur pour savoir à quelle distance se situera le milieu ,d/2
- Je calcule la longueur entre chaque deux sommet de la ligne et je la compare avec la moitié de la distance.
- J'incrémente à chaque fois la distance du segment calculé précédemment avec la longueur de celui qui le suive, Tout en comparant toujours avec d/2
- A la sortie de cette boucle, je sais dans quel segment [AB], se trouve le milieu
- Je calcule les coordonnées du milieu C qui se situe sur le segment [AB], et :
  - **$Sqrt((Xc-Xb)^2+(Yc-Yb)^2)= L-d/2$**

## 6. Correction des anomalies

### 1. Logiciel

La gestion des anomalies se fait à travers le logiciel « mantis BUG TRACKER»

- **mantis BUG TRACKER** : Mantis est un système de suivi d'anomalies logicielles (bugs) basé sur une interface web. Il est écrit en PHP et requiert une base de données (MySQL, SQL Server, PostgreSQL ou DB2) supportée et un serveur web. Mantis peut être installé sur Microsoft Windows, Mac OS, OS/2 et sur de nombreux OS du type Unix.

Le principe de cet outil consiste à enregistrer la déclaration d'un bogue informatique, puis pour les techniciens de maintenance informatique concernés, à mettre à jour l'avancement de sa résolution, jusqu'à sa clôture. Le déclarant de l'anomalie peut s'informer à tout moment via le serveur Web de l'avancement du traitement de son problème. **

### 2. Environnement du travail

L'environnement du travail du projet Verdi se décompose en une chaîne de quatre nœuds :

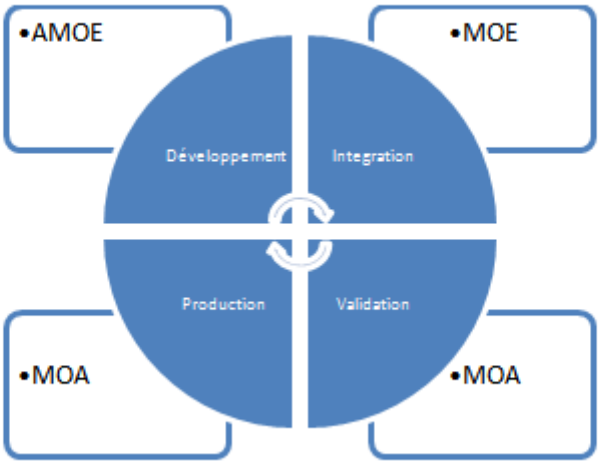



Figure 10-Environnement de travail

L'environnement du développement représentant l'assistance à la maitrise d'œuvre « AMOE » est composé principalement des développeurs qui développent les fonctionnalités de l'application et qui reçoivent la plupart des anomalies pour les corriger.

L'environnement de l'intégration représentant la maîtrise d'œuvre « MOE » est composé de développeurs mais principalement des ressources de profil fonctionnel. Cette équipe teste les fonctionnalités développées par l'équipe du développement ainsi que leurs corrections des anomalies. Elle a aussi l'habilité d'accepter ou de refuser les anomalies remontées par le client. La MOE a principalement pour rôle, assurer la cohérence de l'application avec le cahier des spécifications fonctionnelles.

L'environnement de validation est constitué d'une équipe de testeurs chez le client, qui testent les fonctionnalités développées et déjà testées par l'équipe de l'intégration. C'est l'avant dernier anneau de la chaîne de la production.

L'environnement de production c'est la dernière étape et constitue la version de l'application confirmée et approuvée par les différentes équipes du projet. Cependant, il est possible qu'on décèle de nouvelles anomalies ou de nouvelles évolutions et dans ce cas elles sont remontées à la MOE.

Cet outil permet d'affecter à chaque développeur une anomalie rapportée par le testeur avec un commentaire expliquant le scénario suivi pour reproduire l'anomalie. Une fois l'anomalie reçu le développeur procède comme suit :

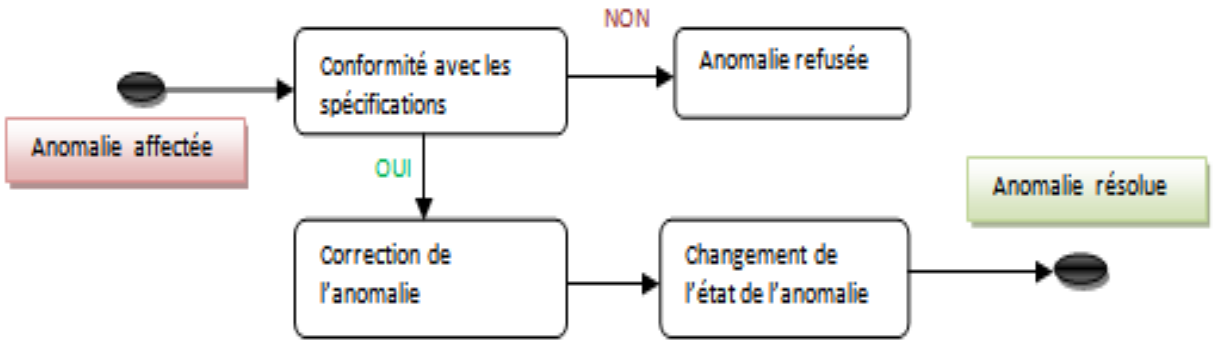



Figure 11-Gestion des anomalies en Développement

Une fois l'anomalie est corrigée par le développeur, elle est remontée encore une fois à l'équipe de testeurs pour une vérification de la correction apportée :

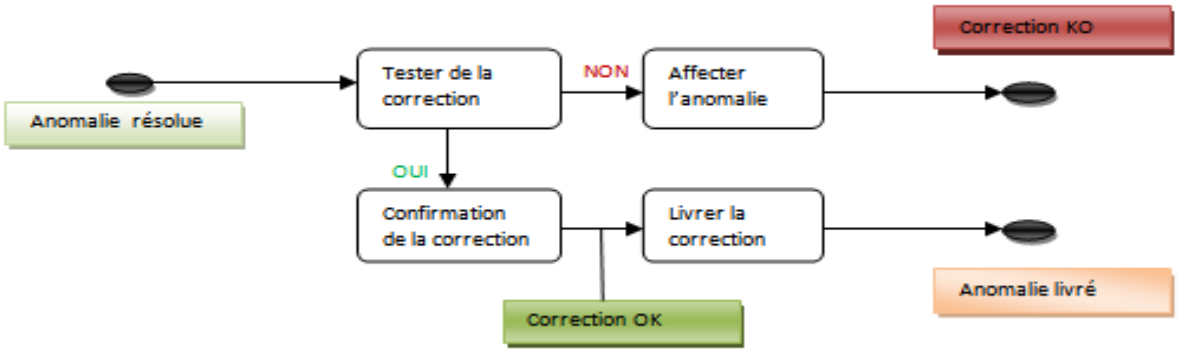



Figure 12-Gestion des anomalies en Intégration

Si la correction de l'anomalie est confirmée par les testeurs, elle passe à l'état confirmée et elle est livrée au client qui la teste encore une fois. Deux cas se présentent soit la correction est acceptée et dans ce cas on ferme l'anomalie, soit la correction est refusée et dans ce cas on remonte l'anomalie à l'équipe des testeurs.

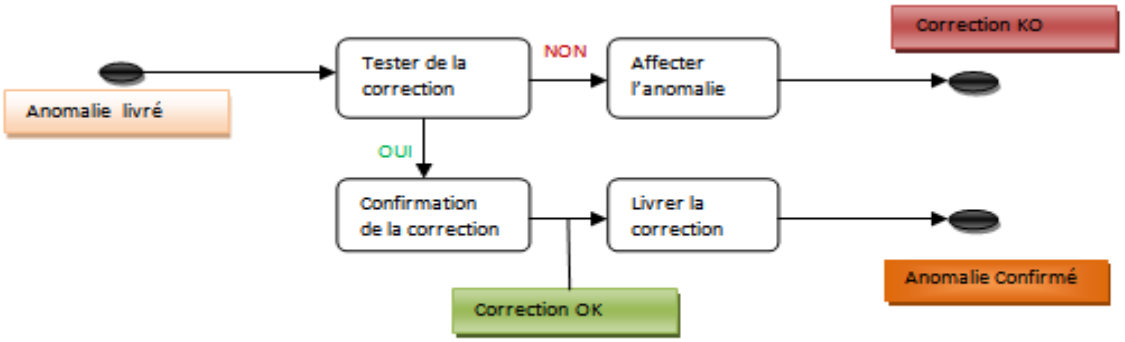



Figure 13-Gestion des anomalies en Validation

Cette méthode pour la résolution des anomalies offre une certaine transparence sur le déroulement du projet et permet de suivre le cycle de vie des anomalies. Elle permet aussi de favoriser la communication entre les différentes équipes du projet et de la maintenir surtout pour un projet d'une grande ampleur tel que VERDI.

### 9. Conclusion

Le fait d'intégrer un grand projet tel que VERDI , m'a permis de monter en compétence dans plusieurs nouvelles et de maîtriser d'avantage celles acquises lors de ma formation . Il m'a aussi aidé à m'être en œuvre mes deux profils de développeur et de géomaticien. En outre, étant confrontée à des problématiques de différentes natures, j'ai été amenée aussi à utiliser mon esprit de raisonnement et ma logique.

Le fait d'être aussi un membre d'une grande équipe m'a permis d'expérimenter le travail en équipe et m'a aidée à fortifier mon relationnel.

Cependant, le plus grand avantage que m'a apporté VERDI est de pouvoir découvrir les différentes étapes par lesquelles passe un projet informatique avant qu'il voie le jour.

## 2. Projet 2 : Etat d'art de QGIS Server

### 1. Présentation du projet

Avec la baisse de la charge sur le projet VERDI, j'ai participé en parallèle à une étude d'état de l'art du serveur cartographique du SIG QGIS ainsi que sa comparaison avec Mapserver.

Cette étude se concentrait principalement sur les performances en temps de réponse des services WMS et WFS fourni QGIS server.

Lors de cette étude nous avons pu tester la plupart des requêtes des différents services cartographiques et les comparer avec ceux du serveur cartographique Mapserver.

Le choix de comparer par rapport à Mapserver vient du fait que les deux serveurs sont des serveurs OpenSources.

Ce projet est de petite ampleur et il a pour finalité estimé les capacités de QGIS Serveur, afin de le proposer comme solution dans les projets à venir.

## 2. L'équipe du projet

L'équipe veillant sur cet état est composée principalement de:

- Mony Jean François : chef de projet
- Sylvain Barbier : chef de projet
- Drouet Adrien : tests et analyse
- CHAKIR Ferdaousse : tests et analyse

Vu que le projet a un périmètre restreint, il n'a pas nécessité l'implication de plusieurs ressources.

## 3. Outils techniques

Afin de réaliser les différents tests de performances nous avions eu recours au logiciel « Apache JMeter».

- **Apache JMeter**: est un projet de logiciel libre permettant d'effectuer des tests de performance d'applications et de serveurs selon différents protocoles ainsi que des tests fonctionnels. Il est développé au sein de la Fondation Apache (ASF).

JMeter est entièrement écrit en Java, ce qui lui permet d'être utilisé sur tout système d'exploitation supportant une machine virtuelle Java (JVM).

Il permet de simuler le comportement de plusieurs utilisateurs agissant de manière simultanée sur une application Web. Au fur et à mesure de ses développements, il a été étendu et permet de tester d'autres types d'applications : serveurs ftp, serveurs de services Web, bases de données accessible via jdbc, serveurs TCP/IP, etc.

Il mesure le temps de réponse de chaque requête et produit des statistiques de ces temps de réponse.

En outre, Nous avons utilisé le SIG QGIS principalement pour la production des fichiers « .qgs » dont nous verrons plus tard l'utilité.

## 4. Tests

Avant de se lancer dans les tests, nous avons préfixé une grille de tests qui fixera le périmètre de notre analyse.

| Type service | Test | | | | | | |
|---|---|---|---|---|---|---|---|
| | Description | | Données | | Type de données | | |
| | | | Shapfile | Postgis | Polygone | Ligne | Point |
| WMS | Symbologie complexe | | | x | x | x | x |
| | Symbologie simple | | | x | x | x | x |
| | Labeling | | | | | | |
| | Montée charge | | | x | x | x | x |
| | Type de requête | GetMap | x | x | x | x | x |
| | | GetCapabilities | x | x | x | x | x |
| | | GetFeatureInfo | x | x | x | x | x |
| | | DescribeLayer | x | x | x | x | x |
| | | GetLegendGraphic | x | x | x | x | x |
| WFS | Type de requête | DescribeFeatureType | x | x | x | x | x |
| | | GetFeature | x | x | x | x | x |
| | | Transaction | x | x | x | x | x |

Figure 14-Grille de Tests

## 1. Environnement des tests

Les tests ont été exécutés sur machine virtuelle sous « VirtualBox » avec 2 CPU et 2 Go de RAM. Sur cette machine nous retrouvons le système d'exploitation « Debian » de l'image « jessie64 ».

Nous avons également utilisé comme serveur web le logiciel « Apache 2.4.10 » ainsi que « Postgres 9.4.3 » comme SGBD. Pour supporter l'extension spatiale de nos données, nous avons installé et ajouté l'extension PostGIS 2.1.4 (GEOS 3.4.2, PROJ 4.8.0, GDAL 1.10.1, LIBXML 2.9.1).

Pour la production des projets « .qgs » nous avons utilisé le SIG QGIS 2.10.

Afin d'avoir une base comparative avec un autre serveur cartographique, nous avons choisi le serveur cartographique Mapserver 6.4.1.

Pour la création et la configuration des environnements de développement virtuels nous avons utilisé le logiciel « Vagrant ».

- **Vagrant :** est un logiciel libre et open-source pour la création et la configuration des environnements de développement virtuel. Il peut être considéré comme un wrapper autour du logiciel de virtualisation comme VirtualBox.

## 2. Données

Les données utilisées lors de cette étude proviennent de GEOFABRIK. Il s'agit des données OSM de la Corse. Pour ce qui est de la quantité de donnée :

- 12000 points
- 4500 lignes
- 129000 polygones

Nous avons aussi utilisé les données de la Base Adresse National Open source « BANO ».

## 3. Publications des services cartographiques

Afin de publier des web service de type WMS ou WFS, il suffit de créer dans QGIS Desktop un projet contenant les différentes couches à publier et l'enregistrer sous un fichier « .qgs » .Mettez ce dernier dans un répertoire contenant l'exécutable de QGIS server et le tour est joué !

Vous pouvez à priori accéder à vos couche à directement à travers un url ressemblant à l'exemple suivant :

http://localhost/cgi-bin/qgis_mapserv.fcgi?map=/shared/gis/corse.qgs&SERVICE=WMS&VERSION=1.1.1&REQUEST=GetMap&LAYER=corse_roads&STYLES=&SRS=EPSG%3A4326&BBOX=5.0526201171874,38.937553710938,13.281379882812,45.452446289063&WIDTH=749&HEIGHT=593&format=image%2FPNG

A fin de test le web service publié il suffit d'ajouter la couche WMS/WFS publiée dans QGIS Desktop à travers l'outil «  Ajout une couche WMS » .

Renseignez l'URL cité dessus et cliquez sur « connexion » pour avoir apparaitre toutes les couches contenues dans le service.

Le fichier « .qgs » sert comme sorte de Template pour le service web publié. Il contient les différentes caractéristiques des couches publiées  ainsi que des informations sur leurs symbologies.

Par conséquent, avec les services web publiés, nous avons le même aperçu des couches que dans QGIS Desktop.

Avec cette approche nous constatons une certaine similarité avec ArcGIS, qui permet de publier un web service de type WMS ou WFS à travers les fichiers « .mxd ».

## 5. Mission

Dans ce projet, je me suis occupée de tester les différentes possibilités offertes par QGIS dans la symbologie et faire une comparaison avec Mapserver.

Il m'est aussi affecté le test des services WFS et WFS-T de QGIS et leurs comparaison avec celui du Mapserver.

En outre, j'ai pu aussi tester la publication du raster avec QGIS server et comparer sa performance avec celle du Mapserver sur ce point.

### 4. Scénario de test

1. WMS

Création et enregistrement d'un projet .qgs contenant des couches postgis de la corse .Nous avons également utilisé des données open street map pour la corse.

- Couches utilisée:
  - corse_railways
  - corse_places
  - corse_landuse
  - planet_osm_point

Création d'une page html "index.html". Dans cette page via la bibliothèque Javascript "OpenLayers" nous ajoutons les couches WMS publiées par QGIS serveur depuis le projet crée.

- Conclusions
  - Rendu visuel identique à celui de QGIS Desktop.

- Nous arrivons bien à récupérer la carte ainsi que les métadonnées du service WMS avec les requêtes “GetMap” et “GetCapablities”.
- Récupération de la légende sous forme d’une image avec la requête “GetLegendGraphic”

- Test de performances

Comme cité ci-dessus, pour les tests de performances nous avons utilisé le Logiciel « JMeter ».Ce logiciel permet de simuler un trafic de réseau pour une requête donnée. Chose qui pour nous aider pour avoir sur le comportement des deux logiciels face à une montée de charge.

Les tests de performance de QGIS Server suivant trois axes:

- L’impact de la complexité de la symbologie

Dans un premier temps nous avons effectué des tests avec la symbologie la plus simple possible et pour trois différents types de géométries. Et tout en comparant toujours avec Mapserver, nous avons obtenu les résultats suivants :

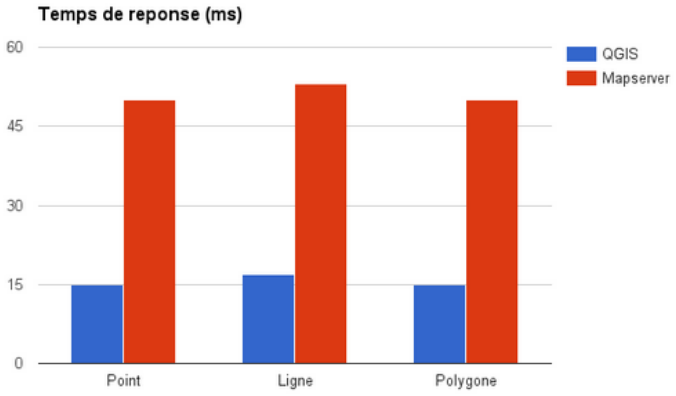



Figure 15-Impact de la symbologie

Ce test a montré que QGIS Server est beaucoup plus rapide que Mapserver ainsi que le type de géométrie affiché n’a presque pas d’influence sur le temps de réponse des deux serveurs.

A fin de tester l’impact de la complexité de la symbologie, nous avons commencé par créer des symboles complexe dans le fichier « Mapfile » et dans le projet « .qgs ».

- QGIS le symbole est une composition des différents symboles de la librairie fourni par QGIS Desktop
- Mapsserver, il s’agit de la description vectorielle. (cf p.Symbologie)

Une fois ces symboles créés nous renseignons la requête responsable de l’affichage de la couche dans « JMeter ».Le diagramme suivant présente le résultat obtenu :

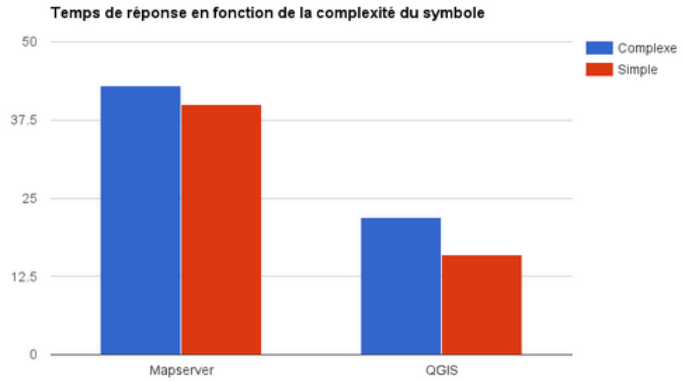



Figure 16-Impact de la symbologie

- L'impact du nombre des features affichés

Pour ces tests, J'ai ajouté dans « JMeter » vingt cinq requêtes, à chaque requête nous ajoutons une couche pour augmenter le nombre de features affichés à chaque fois.

Dans ce test j'ai utilisé les données de Bano().

J'ai effectué les tests pour 1, 5, 10 clients pour avoir une idée des limites des deux serveurs.

Je vous présente ci-dessous les résultats obtenus pour un et dix clients


Figure 17-Impact du nombres de features affichés

Nous constatons qu'avec l'augmentation des nombres des features, nous remarquons que la performance de Mapserver est largement meilleure que celle de QGIS Serveur. En effet, à partir de 140 Mo de données pour un seul utilisateur QGIS serveur ne renvoie aucune réponse ce qui est très modeste comme performance.

- La montée en charge

Le test de la montée en charge est primordial pour estimer les capacités d'un logiciel.

Sur une unique couche, nous avons effectué une vingtaine de requêtes en boucle, en ajoutant un utilisateur toutes les 15 secondes.

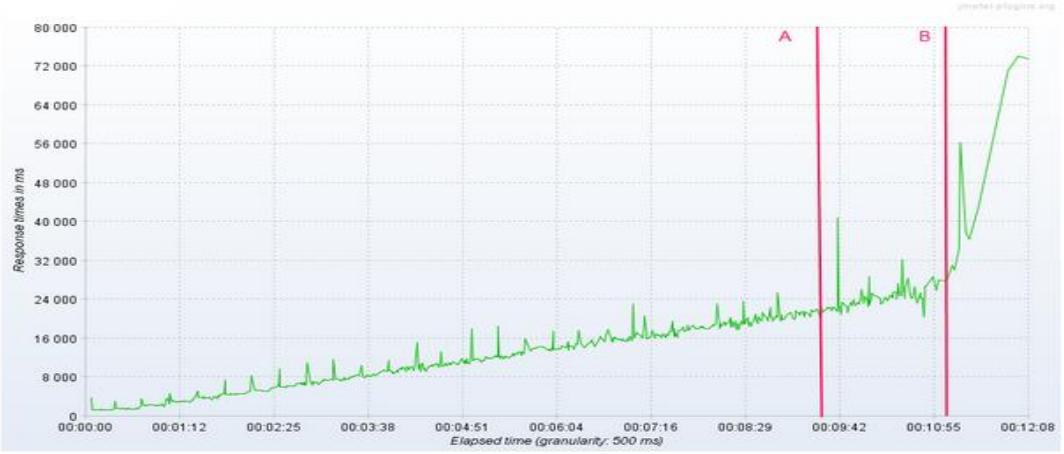



Figure 18-Tests de Performances

- A (40 utilisateurs): apparition des premières erreurs dans les réponses
- B (45 utilisateurs): plus de réponse du tout

**Analyse** :

Le temps de réponse augmente comme on a pu le voir précédemment linéairement en fonction du nombre d'utilisateurs. Le point de rupture se trouve à partir de 40 utilisateurs, les nouvelles requêtes ne peuvent plus être traitées. Sur le serveur, la cause est le manque de mémoire vive + swap. Des que ces mémoire n'ont plus d'espace libre, il est impossible pour QGIS de répondre.

Pour Mapserver, le résultat est similaire mais apparaît une minute plus tard, avec cinq nouveaux utilisateurs. Il semble donc que Mapserver utilise moins de mémoire pour fournir les mêmes données.

Les résultats sont tout de même à prendre avec des pincettes. Le plate forme sur laquelle est installée JMeter commence à souffrir énormément quand le nombre de thread (un utilisateur = un thread) augmente.

## 2. WFS

- Scénarios de test

En ce qui concerne le service web de type WFS nous étions plutôt intéressés par les requêtes transactionnelles qui permettent d'éditer des features layer .En s'orientant dans cette voie j'ai mis en place l'environnement de test suivant :

- Ajouter dans la carte créée auparavant dans la page “index.html”, une couche WFS “corse_landuse”
- Utilisation des contrôles Openlayers pour l’édition

- Conclusions

- Récupération et Affichage de la couche WFS
- Récupération des attributs de la couche WFS en filtrant sur une bbox avec la requête “GetFeature”
- Possibilité d’ajouter, modifier et supprimer des couches avec le protocole WFS-T.

Le service WFS fournit par QGIS Serveur permet comme toute service WFS la récupération des caractéristiques du service avec la requête “GetCapabilitie”, la récupération des données attributaire avec la requête “GetFeature”.En outre, des requêtes basiques des services WFS, Le service WFS fournit par QGIS Serveur supporte les requêtes transactionnelles. Ceci permet de modifier les couches publiées en WFS en ajoutant, supprimant ou modifiant ses objets.

## 3. Symbologie

### Création

#### QGIS Server

Afin de personnaliser la symbologie des couches WMS, on la personnalise dans le projet .qgs avec le QGIS Desktop.

QGIS Desktop permet de modifier le rendu des couches .Pour cela, on choisit un symbole parmi ceux proposés ou alors on crée un nouveau symbole personnalisé.

Pour ce qui est de la création d’un symbole, Il suffit de choisir dans ***Layer Properties->Style*** un symbole de départ et de lui en ajouter d’autres pour obtenir un symbole plus complexe. Nous pouvons aussi changer sa couleur, sa taille, et son orientation. Cette dernière peut être un angle fixe ou définie à partir d’un champ attributaire de la couche.

Après création du symbole nous pouvons le sauvegarder et exporter le style sous un des deux formats : sld ou Qgis Layer Style File (.qml).

#### MapServer

MapServer permet aussi de personnaliser la symbologie des couches. Pour cela, on crée le symbole souhaité dans le fichier Mapfile. Exemple:

```
SYMBOL
        NAME "complexe"
        TYPE vector
     POINTS
                0 4
                2 0
                4 4
                0 4
                -3 2
                -1 -2
                2 0
                -3 2
                0 4
                -1 -2
                -2 0
                -1.5 3
                1 2
                0.5 -1
                -2 0
        END
END
```

Figure 19-Symbologie Mapserver

Il est clair qu'avec l'augmentation du degré de complexité du symbole, la création de ce dernier devient plus simple dans le projet .qgs que dans le fichier mapfile.

Nous pouvons également définir l'orientation du symbole avec la propriété ANGLE qui peut prendre une valeur d'angle fixe ou un attribut de la couche. Exemple:

```
CLASS
        STYLE
        SYMBOL 'complexe'
                SIZE 2
                COLOR 0 0 255
                ANGLE [attribut]
        END
END
```

Figure 20-Symbologie Mapserver

Cependant, en ouvrant le fichier XML décrivant le symbole de QGIS nous constatons que sa compréhension par un développeur non expérimenté en QGIS n'est pas évidente .Ce qui n'est pas le cas dans le fichier Mapfile qui est beaucoup plus explicite.

- Import / Export

Comme est déjà mentionné, QGIS Desktop permet d'exporter ainsi qu'exporter des styles ou des symboles.

Pour les symboles, nous pouvons les importer depuis des fichiers xml décrivant le symbole comme nous pouvons les importer depuis une url. Nous pouvons également importer des symboles svg en ajoutant le chemin du fichier contenant les symboles dans les “SVG paths” dans ***Setting->Options***. Une fois cela est fait nous retrouvons nos symboles importés avec les symboles svg proposés par QGIS.
En ce qui concerne l’export Qgis permet d’exporter les symboles sous format xml ainsi que les styles sous le format .xml ou .qml.

- Agrégation des symboles

Qgis ne permet pas d'agréger les symboles en fonction du niveau de zoom. Cependant, il permet de graduer la taille des symboles en fonction de la valeur d’un champ. Ceci est possible en personnalisant la propriété “size” du symbole dans

***Layer Properties->Style->Size->Size Assistant*** ceci améliore le rendu visuel de la couche dans le projet qgs et par conséquence améliore le rendu visuel de la couche WMS.

- Qgis server “GetLegendGraphic”

Avec la requête “GetLegendGraphic” du service WMS déployé par Qgis server nous pouvons récupérer la légende des couches présentes dans le projet .qgs sous format d’image.

- Génération du fichier de configuration du service de carte

A la réalisation de ces tests, on s'aperçoit qu’un fichier mapfile est simple à comprendre. Toutes les lignes ont du sens et sont documentées par l'éditeur. L’écriture programmatique de ce fichier est aisée. Cela permet notamment de fournir des symbologies personnalisées par le client.

Le fichier de projet QGIS est lui beaucoup plus complexe. Le .qgs sert à deux choses : la configuration des couches et le sauvegarde de l'état de l’application QGIS Desktop. On se retrouve alors avec un fichier d’un minimum de 400 lignes à la création d’un projet. Le .qgs est un fichier XML non documenté. La DTD n’existe pas. En expérimentant, on réussi tout même à avoir un fichier avec le strict minimum (une vingtaine de lignes, exemple à suivre). En avançant à tâtons, on devrait être capable de générer ce fichier qgis mais l’absence de documentation est un frein à cette utilisation en dehors de QGIS Desktop.

```
<!DOCTYPE qgis PUBLIC 'http://mrcc.com/qgis.dtd' 'SYSTEM'>
<qgis projectname="" version="2.10.1-Pisa">
 <projectlayers layercount="1">
   <maplayer geometry="Point" type="vector">
     <id>corse_points</id>
     <datasource>dbname='dbuser-db' host=localhost port=5432
user='dbuser' password='dbuser' sslmode=disable key='ogc_fid' srid=4326
type=Point table="public"."corse_points" (wkb_geometry)
sql=</datasource>
     <layername>corse_points</layername>
     <provider encoding="System">postgres</provider>
     <renderer-v2 symbollevels="0" type="singleSymbol">
       <symbols>
         <symbol alpha="1" clip_to_extent="1" type="marker" name="0">
           <layer pass="0" class="SvgMarker" locked="0">
             <prop k="angle_dd_expression" v="osm_id"/>
             <prop k="angle_dd_useexpr" v="1"/>
             <prop k="color" v="#000000"/>
             <prop k="name" v="gpsicons/plane.svg"/>
             <prop k="size" v="5"/>
           </layer>
         </symbol>
       </symbols>
     </renderer-v2>
   </maplayer>
 </projectlayers>
</qgis>
```

Figure 21-Fichier .qgs

- Conclusion

QGIS serveur comme tout serveur cartographique fournit des services web cartographiques (WMS/WFS). Cependant, le fait que QGIS serveur utilise les mêmes bibliothèques que QGIS Desktop lui donne un grand avantage. Cela nous permet de profiter de toutes les fonctionnalités offertes par QGIS Desktop que se soit en termes de symbologie ou de traitement spatial. Le rendu obtenu dans QGIS sera le rendu obtenu par les clients du service WMS.

On peut regretter l'absence de fonctionnalités avancées pour la légende ou l'agrégation des features

## 6. Conclusion

Tout semble être à l'avantage de QGIS serveur à la lecture de ce document. QGIS a réussie à combiner la simplicité de déploiement d'un Mapserver (fichier Mapfile/Qgs) avec la puissance d'un client lourd pour la configuration des couches.

Du coté des performances, les tests sont également en faveur de QGIS. Mais les résultats sont à relativiser. Ils sont effectués sur des machines virtuelles hébergées sur des PC qui n'ont pas grand chose en commun avec un serveur de production d'une application cartographique.

QGIS serveur semble également manquer de documentation.

Il est difficile de comparer l'activité de développement entre Mapserver et QGIS, ce dernier étant à la fois un client lourd et un serveur mais les chiffres sont quand même significatifs. Dans le dernier mois (à l'écriture des ces lignes, le 10 septembre 2015), les dépôts Github de Mapserver ont reçu 42 contributions par 8 auteurs différents. Pendant ce même temps QGIS a reçu 602 contributions par 37 auteurs différents.

Avec le temps passé à étudier les deux solutions, nous avons des difficultés à identifier des contraintes projets qui nous feraient pencher pour Mapserver plutôt que QGIS Server, en dehors des couches Raster.

# 3. Projet 3 : Contrôle des données SNCF « ETCS"

## 1. Présentation du projet

- ETCS :

ETCS est un système de commandes à distance permettant de simplifier la conduite des trains et de la rendre plus intelligente. Ce système permet aussi d'augmenter la sécurité de la circulation des trains.

Vu l'énorme enjeu présent, la SNCF doit absolument faire un contrôle sur toutes ces données pour déterminer avec une grandes précision où placer les boîtiers d'ETCS.

Par conséquent, La SNCF a mis en place deux chaînes de production de données qui utilisent des technologies différentes afin de minimiser l'erreur sur les données. Ces dernières sont par la suite contrôlées par des unités de contrôle qui mettent en place des programmes pour apparier les données cohérentes des deux chaînes ou relever celles qui sont en anomalies.

Ce projet comme son nom l'indique vise à contrôler les données de la SNCF. Nous disposons des données produites de deux chaînes de production « SP2.1 » et « SP2.2 » avec respectivement deux différentes types de technologies « Exelis-Esri » et «Technet et Rail ». Ces données sont par la suite intégrées dans deux serveur de données ORACLE et SQL Server grâce à des ETLs.

Les données se constituent d'une couche de segments « TAV » et deux couches de points appelés respectivement « NAV » et « CRT ».

Les données issues des deux bases de données sont censées être identiques .Cependant, il y a toujours une différence entre la théorie et la pratique. Par conséquent, il a été nécessaire de mettre en place une unité de contrôle.

Le rôle de l'équipe est d'implémenter et de proposer des algorithmes au client afin à déceler les anomalies dans les données. Il faut aussi que l'équipe fournisse au client une interface IHM pour lancer l'opération de contrôle ainsi que des rapports cartographique présentant les données en anomalie.

L'architecture de l'unité de contrôle se présente comme suit :

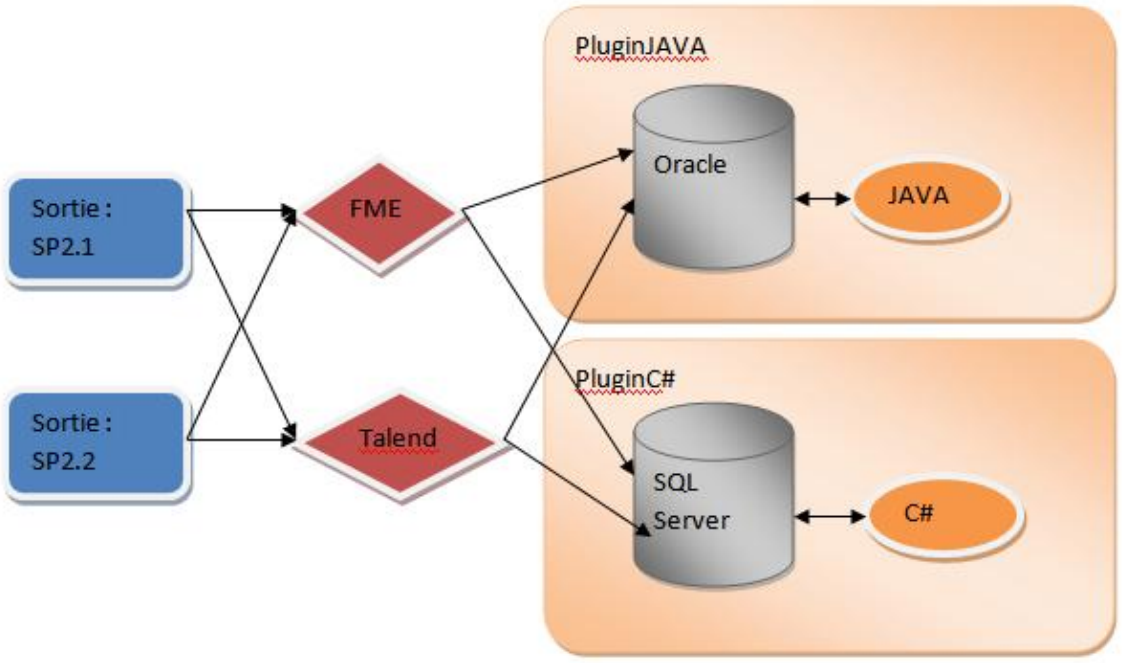



Figure 22-Architecture unité de contrôle ETCS

## 2. L'équipe

L'équipe de contrôles est constituée de cinq personnes :

- Chef de projet : Jean François Mony
- Responsable technique : Adrien Drouet
- Trois ingénieurs développeurs :
  - Nassim Dekkar : C#, Oracle spatial, ArcGIS
  - Ferdaousse CHAKIR : Java, PostGis, ArcGIS
  - Claire Lefeuvre : FME, Talend

## 3. Outils

Comme dans tout projet, chaque équipe s'arme d'un nombre d'outil que se soit technique ou managérial :

- Outil managérial :
  - CA Clarity PPM : c'est un outil de gestion de portefeuilles de projets (GPP) .Il est articulé autour de plusieurs modules complémentaires : Portfolio Manager, Project Manager, Ressource Manager.

Cet outil peut être utilisé simultanément par plus de 100 000 utilisateurs.

- Outil technique
    - PostgreSQL : est un système de gestion de base de données relationnelle et objet (SGBDRO). C'est un outil libre disponible selon les termes d'une licence de type BSD.

      Ce système est concurrent d'autres systèmes de gestion de base de données, qu'ils soient libres (comme MariaDB, MySQL et Firebird), ou propriétaires (comme Oracle, Sybase, DB2, Informix et Microsoft SQL Server). Comme les projets libres Apache et Linux, PostgreSQL n'est pas contrôlé par une seule entreprise, mais est fondé sur une communauté mondiale de développeurs et d'entreprises.
    - PostGIS : est une extension (plugin) du SGBD PostgreSQL, qui active la manipulation d'informations géographiques (spatiales) sous forme de géométries (points, lignes, polygones), conformément aux standards établis par l'Open Geospatial Consortium. Il permet à PostgreSQL d'être un SGBD spatial (SGBDs) pour pouvoir être utilisé par les systèmes d'informations géographiques.
    - ArcGIS
    - Eclipse
    - SVN

Pour l'écriture des algorithmes nous nous somme mis d'accord sur l'utilisation du langage PLPGSQL pour ce qui est de l'interaction avec la base données PostgreSQL. En ce qui concerne la base de données ORACLE nous avons opté à programmer les algorithmes en C#

- PL/PGSQL : (Procedural Language/PostgreSQL Structured Query Language) est un langage procédural géré par PostgreSQL. Ce langage est très similaire au PL/SQL d'Oracle, ce qui permet de porter des scripts de ou vers Oracle au prix de quelques adaptations1.
- C# : C sharp, est un langage de programmation orienté objet, créé par la société Microsoft, et notamment un de ses employés, Anders Hejlsberg, le créateur du langage Delphi.

Il a été créé afin que la plate-forme Microsoft .NET soit dotée d'un langage permettant d'utiliser toutes ses capacités. Il est très proche du Java dont il reprend la syntaxe générale ainsi que les concepts (la syntaxe reste cependant relativement semblable à celle de langages tels que le C++ et le C). Un ajout notable au C# est la possibilité de surcharge des opérateurs, inspirée du C++. Toutefois, l'implémentation de la redéfinition est plus proche de celle du Pascal Objet.

## 4. Modèle conceptuel des données

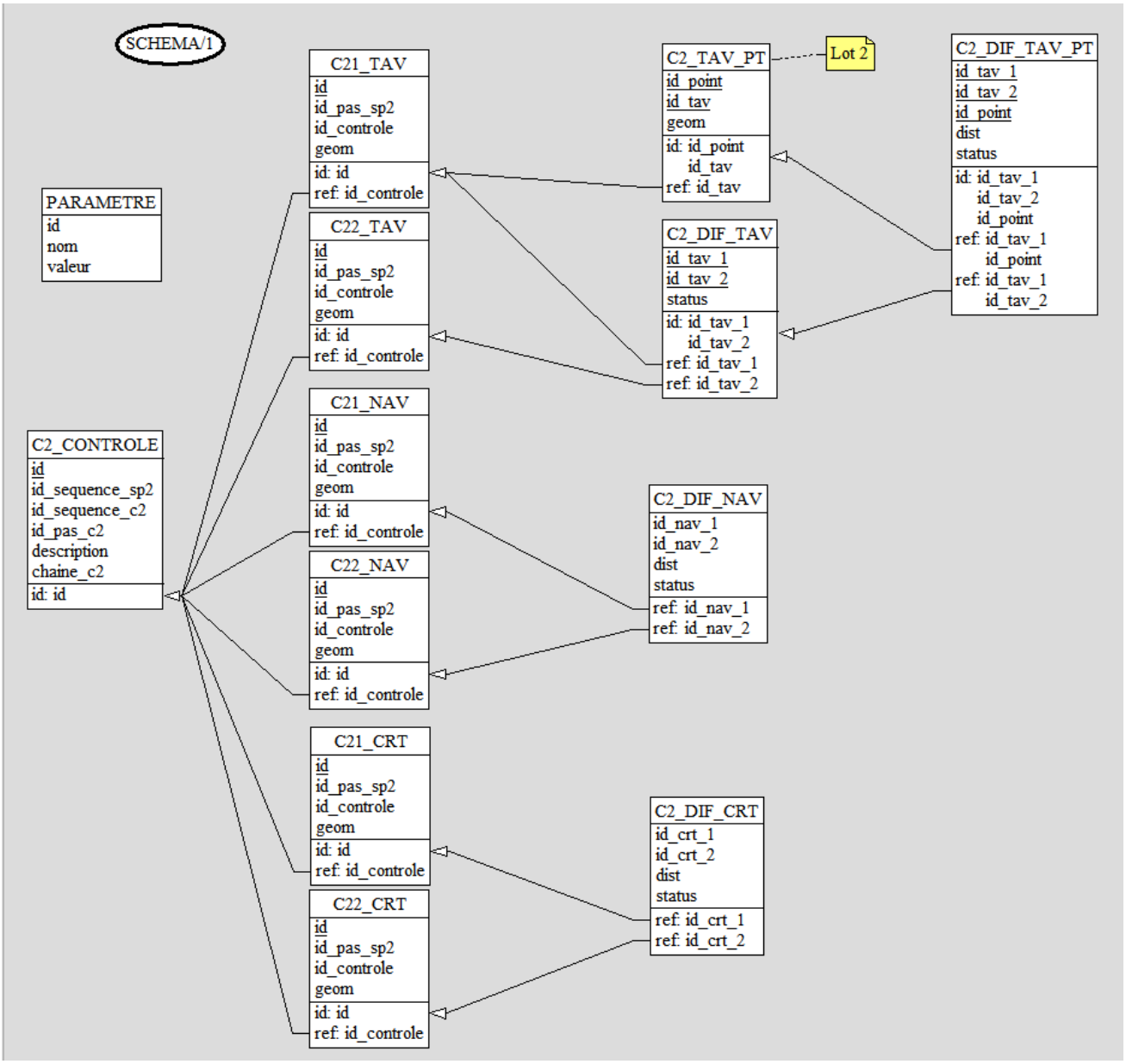



Figure 23-Modèle Conceptuel des données

## 5. Mission

### 1. Algorithmes

Comme mentionné ci-dessus, ma mission sur ce projet consistait à écrire les algorithmes en PLPGSQL en utilisant les différentes fonctions de PostGIS. Je m'occupais également des générations des fichiers « .mxd » dont nous se servons comme rapports cartographiques pour représenter les données en anomalie.

Comme nous disposons de trois types de données, le client nous a fourni trois algorithmes permettant d'apparier les données venant des deux chaines de production et de relever les points en anomalie.

En ce qui concerne l'algorithme traitant les données « NAV », Il décrit le processus suivant :

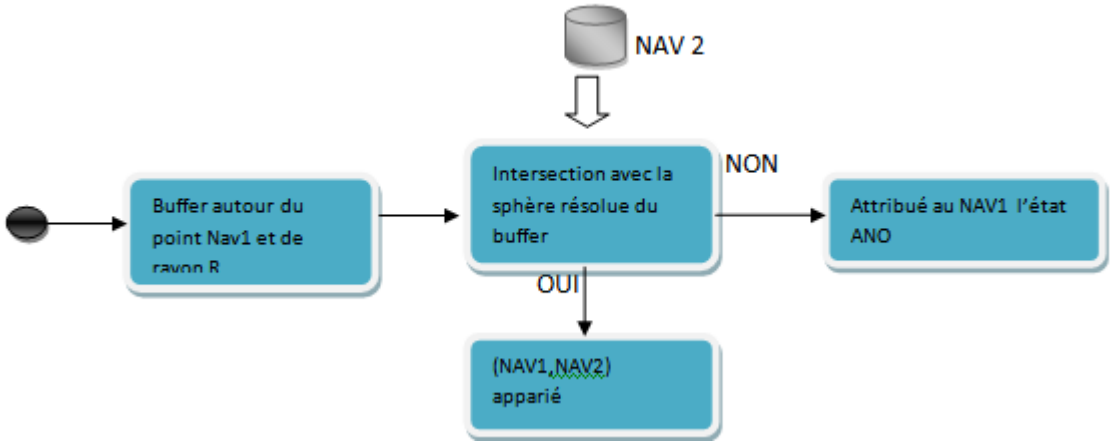



Figure 24-Algorithme appariement des points NAV

Après avoir parcouru toute la table des NAV1, Il faut attribuer les objets NAV2 n'ayant aucun jumeau dans NAV1


Figure 25-Algorithme appariement des points NAV

L'algorithme décrit ci-dessus a été traduit par la suite dans un script PLpgsql. Le choix d'utiliser ce langage se justifie par limitation des « va et vient » entre le plugin java de contrôle et la base de données.

En ce qui concerne, le géotraitemment du buffer sur les points des Nav1, revenait à calculer une distance 3D entre le point du Nav1 et les points des Nav2 grâce à la fonction « ST_3DDistance ».Cette fonction calcule la distance entre deux points sans perdre l'information sur le z. J'ai opté pour cette approche car la fonction « ST_buffer » proposée par PostGIS ne supporte pas les géométries 3D.

Pour ce qui est de la deuxième étape, qui consiste à trouver les points Nav2 contenus dans la sphère résultante du buffer sur le Nav1, Je l'ai traduit en ne récupérant que les points dont la distance 3D calculée dans la première étape et qu'elle doit être inférieure à R le rayon du buffer.

En suite pour les points qui satisfont le test de la distance je les insère par une requête SQL dans la table « C2_DIF_NAV » en renseignant les deux identifiants des deux points de Nav1 et Nav2 avec l'état « APPARIE ».

Dans le cas ou aucun point du Nav2 ne satisfait la condition sur la distance, J'insère l'identifiant du Nav1 dans la table « C2_DIF_NAV » en renseignant l'identifiant du Nav2 à « Null » et l'état à « ANO ».

A la fin de l'algorithme, Je parcours la table des Nav2 pour récupérer les points dont aucun jumeau de la table 1 n'est retrouvé et je les insère dans la table « C2_DIF_NAV » en renseignant l'identifiant du Nav1 à « Null » et l'état à « ANO ».

J'ai utilisé la même logique pour traiter les points « CRT ». Je relève de la même manière les points appariés et ceux qui sont en anomalie et j'insère de la même manière dans la table « C2_DIF_CRT ».

En ce qui concerne l'algorithme traitant les données « TAV », il se décompose en trois parties :

$$\frac{Long(Buffer(A) \bigcap B)}{min(Long(A), Long(B))} > 0.45$$

- Pour les tronçons ne vérifiant pas la condition ci-dessus, ils représentent une anomalie de niveau 1, il faut les stocker dans la table « C2_dif_TAV » avec :
  - l'ID objet(TAV1) = ID_TAVI
  - l'ID objet(TAV2) = Null
- Pour les tronçons vérifiant la condition, ils sont traités dans la deuxième partie.

- Deuxième partie :
  - Les tronçons vérifiant la condition ci-dessus, avant de les appariés avec le tronçon de Tav1 il faut que la valeur suivante :

$$\frac{Surf(Buffer(A) \bigcap Buffer(B))}{Surf(Buffer(A) \cup Buffer(B))}$$

Soit supérieure à une tolérance minimale fixée par le client est dans ce cas :

- Si la condition n'est pas vérifiée, les tronçons sont en anomalie de niveau 2 et nous stockons dans la table « C2_Tav_dif » :
  - l'ID objet(TAV1) = ID_TAVI
  - l'ID objet(TAV2) = ID_TAV2
  - Etat = ANO2
- Si la condition est vérifiée, nous vérifions que le même valeur est inférieur à une tolérance maximale fixé par le client .Si c'est le cas , nous insérons dans la table « C2_DIF_TAV » :

- l'ID objet(TAV1) = ID_TAVI
- l'ID objet(TAV2) = ID_TAV2
- Etat = APPARIE

Si non:

- l'ID objet(TAV1) = ID_TAVI
- l'ID objet(TAV2) = ID_TAV2
- Etat = ANO3

- Troisième partie :

  Dans cette partie nous générons pour chaque tronçon Tav1 une suite de points, nous calculons par la suite pour point sa distance doit être inférieur à une valeur paramétrable.

  Pour les points vérifiant cette condition, nous les stockons dans la table « C2_TAV_DIF_PT » avec un état « APPARIE »les autres points sont inséré avec un état «ANO».

Cependant, lors de la traduction de cet algorithme avec le langage PLpgsql, nous avons été confrontés par les limites de l'extension PostGIS concernant les géométries 3D. Par conséquent, nous avons contourné le problème en créant pour chaque tronçon dans Tav1 et Tav2 deux autres géométries et qui représentent sa projection sur le plan « YZ » et « XZ ». Et comme cela nous ramenons le traitement demandé sur une géométrie 3D au traitement de trois géométries 2D supportées par les fonctions de PostGIS.

Le schéma ci-dessous présente comment j'ai procédé pour répondre au besoin exprimé pas l'algorithme du client :

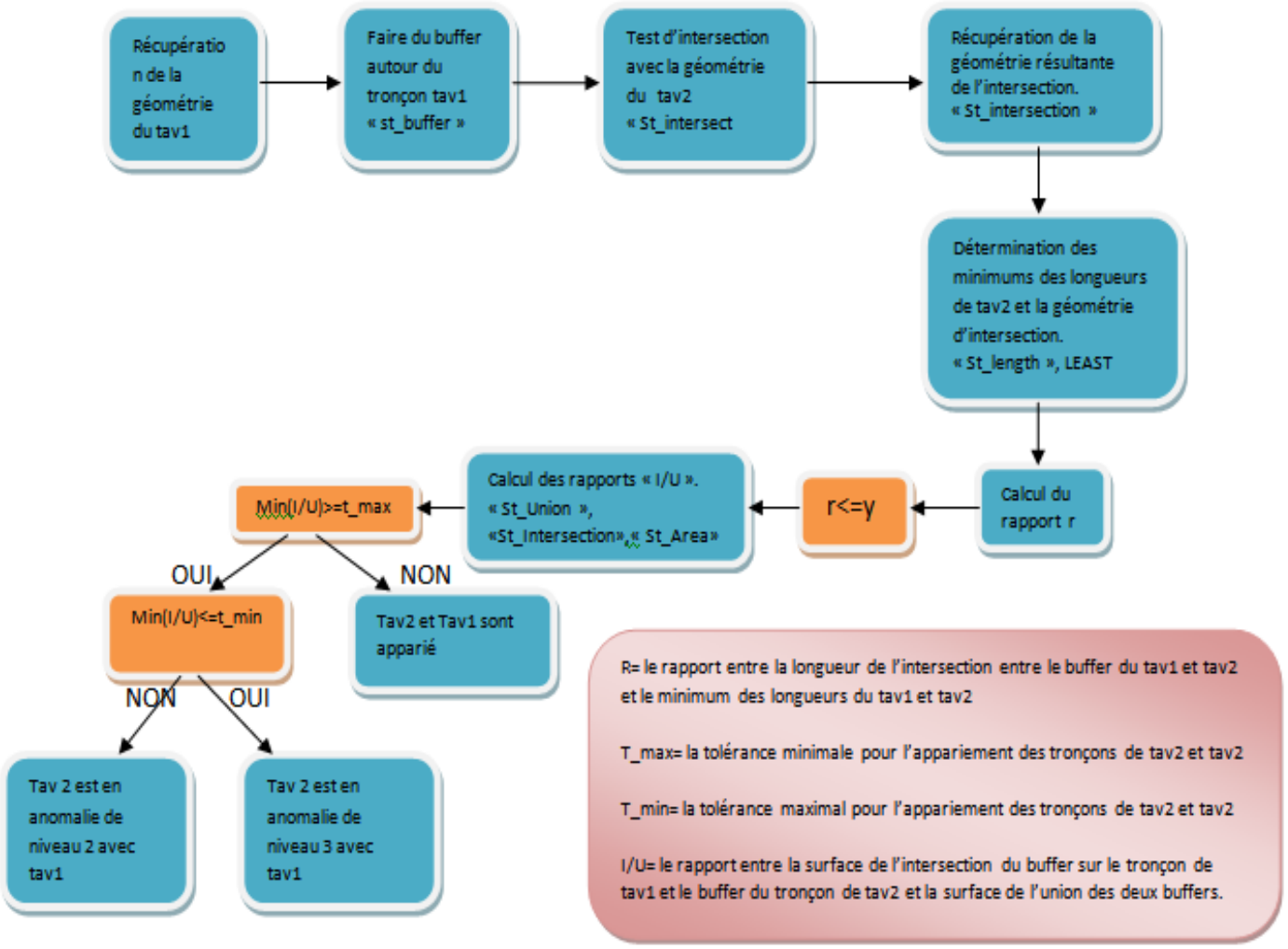



Figure 26-Algorithme appariement des lignes TAV

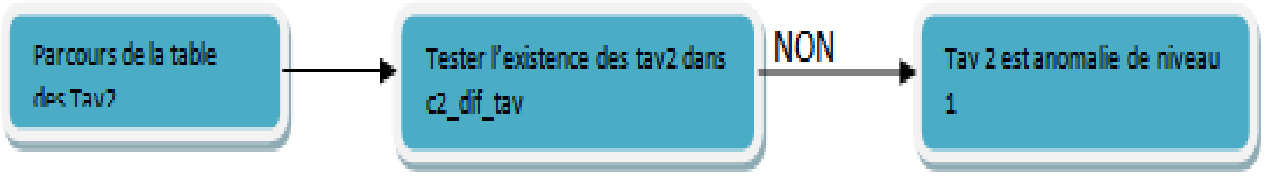



Figure 27-Algorithme appariement des lignes TAV

Pour la génération des points pour chaque tronçon de Tav1 :

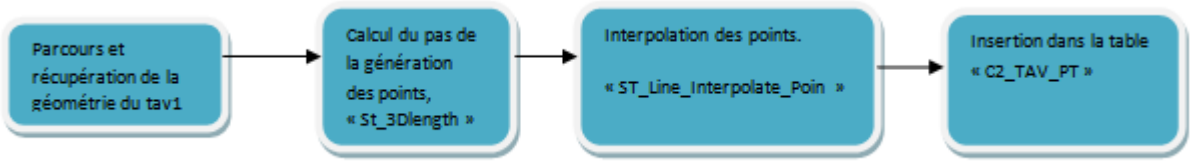



Figure 28-Algorithme appariement des lignes TAV

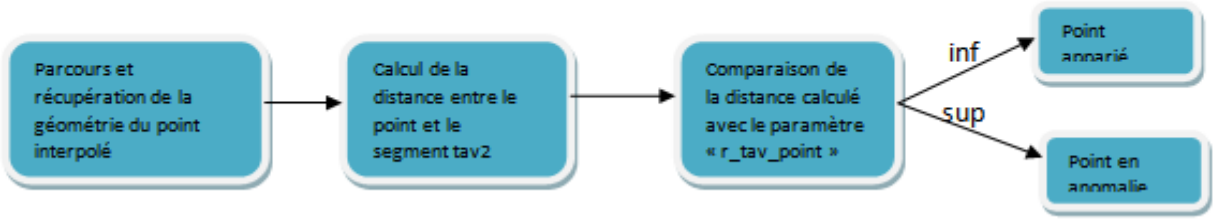



Figure 29-Algorithme appariement des lignes TAV

Les processus décrits ci-dessus sont enregistrés sous forme de procédure plpgsql cette procédure est appelé par la suite avec le plugin java.

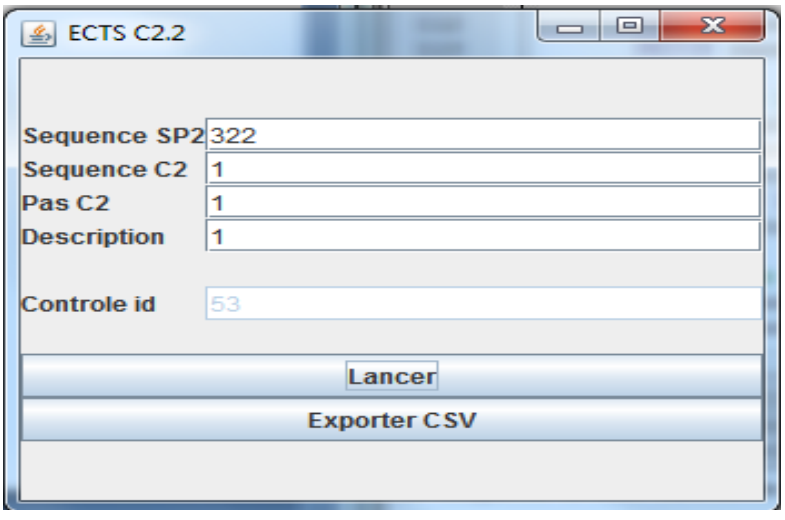



Figure 30-Interface plugin Java

## 2. Générations des fichiers « .mxd »

Nous générons pour chaque SGBD un rapport cartographique sous forme d'un fichier « mxd ».Pour cela et pour limiter le plus possible les opérations faites par le client dans le fichier mxd, j'ai mis en place un script python dans lequel je génère mon fichier de connexion « .sde » grâce à la fonction « CreateDatabaseConnection_management » de la bibliothèque Arcpy. En suite, j'ajoute mes couches à un mxd vide avec la fonction «MakeQueryLayer_management » et puis je leur ajoute des jointures grâce à la fonction « AddJoin_management ». Finalement j'applique la symbologie depuis un fichier « .lyr » que j'ai préparé précédemment.

Pour la génération des fichiers « .mxd » nous disposons d'un répertoire constitué des deux scripts pythons responsables de la génération des deux fichiers « .mxd » ainsi que trois dossiers :

- Template : contenant le mxd vide sur lequel je me base pour générer les deux fichiers « .mxd » contenant les couches
- SDE : contient les fichiers de connexion « .sde » générés depuis les scripts python.
- Symbology : contient les fichiers « .lyr » depuis lesquels je copie la symbologie.

En addition des fichiers « .mxd », il fallait que je mette en place un outil qui permet au client d'afficher les données concernant un identifiant d'un contrôle. Pour cela, j'ai utilisé la boîte à outil python, une fonctionnalité offerte par ArcGIS. Dans cette boîte à outil j'ai ajouté le script qui permet de lancer le traitement voulu.

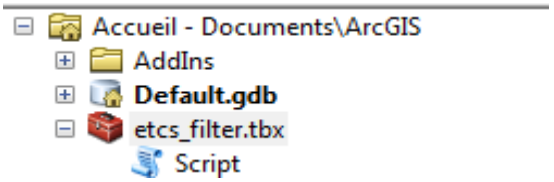



Figure 31-Boite à outils python

La boîte outil python me permet de définir les paramètres que l'utilisateur doit renseigner. Dans ce cas, l'utilisateur définit l'id contrôle en fonction duquel il souhaite afficher les données. Je récupère par la suite cet id de contrôle dans le script python grâce à la fonction « GetParameterAsText() » et puis afin de filtrer les données sur cet id de contrôle , j'ajoute à chaque couche présente dans le mxd un expression définition :

```
idcontrole   = arcpy.GetParameterAsText(0)
mxd = arcpy.mapping.MapDocument("CURRENT")
            for lyr in arcpy.mapping.ListLayers(mxd):
            lyr.definitionQuery ="id_controle ="+idcontrole
```

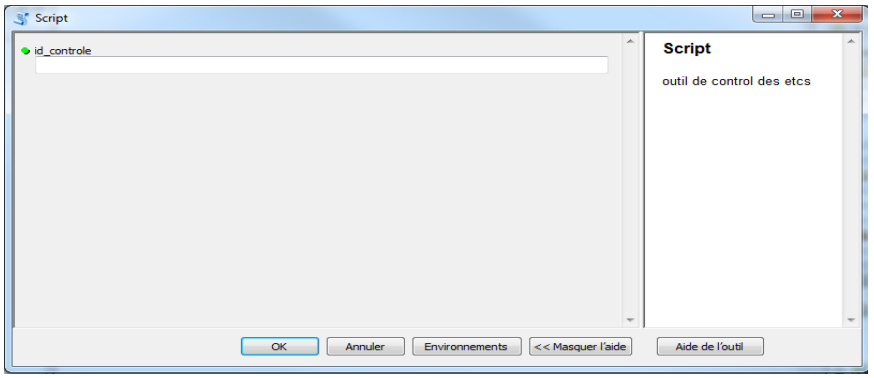



Figure 32-Outil python de contrôles

### 6. Conclusion

Ce Projet a été une très bonne occasion pour mettre en œuvre mes connaissances en PostGIS et SQL acquise lors de ma formation. Cela m'a aussi permis de réutiliser les connaissances acquises lors du projet VERDI en ce qui concerne la génération des mxd.

Ce projet m'a permis aussi de participer à l'étape prématurée d'un projet à savoir la définition des spécifications et la mise en place des ressources nécessaires.

## 3. Bilan

Lors de mon stage j'ai pu acquérir de nouvelles connaissances, ainsi que monter en compétence dans certaines déjà acquises lors de ma formation.

| Montée en compétence en : | Nouvelles compétences : | Nouvelles technologies : |
|---|---|---|
| Travail d'équipe | Programmation procédural avec Pl/pgsql | PL/pgsql |
| D développement Web | Programmer avec python | Api d'ArcGIS for script |
| SQL | | JMeter |
| Analyse et rigueur | | SVN |

# 4. Conclusion

Dans le présent document j'ai présenté les différents projets sur lesquels je suis intervenue ainsi que les différentes problématiques auxquelles j'ai été confrontée. Des problématiques qui m'ont aidé à me forger davantage techniquement et m'ont permis de faire appel à mon sens de raisonnement et ma logique.

Ce stage était pour moi une réelle opportunité pour m'épanouir dans une ambiance de travail d'équipe coopérative tout en m'exposant à de nouveaux défis techniques.

Concernant les objectives de mon stage, Le plus important concernant la participation au développement d'application Web SIG a été atteint à travers L'application des gestions de parcelles du projet VERDI .En outre, J'ai pu aussi atteindre d'autre objectifs tel que l'intégration d'une équipe de professionnel et de pouvoir s'adapter à leurs modes de fonctionnement et d'améliorer mon sens de relationnel.

En outre, ce stage m'a permis comme détaille ce rapport, d'expérimenter d'autre terrain comme celui du benchmarking et d'analyse à travers l'état de l'art porté sur QGIS.

# 5. Références Bibliographiques


le document de la conception techniques de VERDI « DCT » .....13

**Le site de Dojo - https://dojotoolkit.org/** .....24

Le site de QGIS - http://hub.qgis.org/ .....35

Le site ressources d'ArcGIS - http://resources.arcgis.com/ .....20

LeSite de PostGIS -http://www.postgis.org/docs/ST_GeomFromText.html .....43

**Site d'ArcGIS API - https://developers.arcgis.com/javascript/jsapi/** .....24

Wikipédia « https://fr.wikipedia.org » .....17