



Formation professionnelle
Concepteur / Développeur

Développement de services web JAX-RS pour
l'import de données voirie et transport
permettant la constitution automatique de
réseaux de transports publics

Bertrand GUERRERO

Stage de 3 mois réalisé du 13.04.2015 au 13.07.2015 à Mobigis

Responsable de stage : Christophe LAPIERRE

Maître de stage : Julien LESBEGUERIES

Date de soutenance : 15.07.2015

Remerciements

Je tiens tout d'abord à remercier mon responsable de stage Christophe Lapierre qui m'a permis de réaliser mon stage dans les meilleures conditions possibles. Je le remercie pour ses conseils, le temps qu'il m'a consacré et pour ses commentaires sur mon travail.

Je remercie mon maître de stage Julien Lesbegueries pour m'avoir intégré dans son projet, m'avoir permis au cours du stage de découvrir des technologies telles que Dropwizard, merci pour sa pédagogie et pour toutes les bonnes pratiques de développement qu'il m'a transmises.

Merci à mes deux « super » formateurs Gilles Vanderstraeten (Java SE) et Laure Bouquety (Java EE), pour leurs cours et leur support. Ils m'ont appris et fait aimer la programmation orienté objet et plus spécialement le langage Java.

Merci à Florian pour les nombreuses pauses à discuter, et aussi à Wafi pour ses jolis dessins de réseaux ;-).

Enfin, je voudrais remercier particulièrement Frédéric Schettini de m'avoir donné cette opportunité de stage qui rentre pleinement dans mon projet professionnel, tant par les aspects techniques que par leurs domaines d'application.

Résumé

Mon sujet de stage porte sur le développement de web services permettant d'exposer des fonctionnalités d'import de données voirie et transport en commun, pour la constitution automatique de réseaux de transport multi-modaux.

L'objectif de mon travail est d'exposer des fonctionnalités codées initialement en langage SQL et/ou Python via une API REST (JAX-RS). J'ai donc dans un premier temps pris en main les chaînes de traitements (ArcPy/Python) et appris à manipuler les données (Postgis/GTFS) via le projet "DataWizard". En parallèle, afin d'implémenter ces web services REST (Java EE) j'ai intégré le projet « MobiSaaS » et ainsi découvert à développer avec le framework orienté micro-services « DropWizard ».

L'application développée dans le projet "MobiSaaS" présente une interface d'administration des fonctionnalités du progiciel MobiAnalyst¹ en mode SAAS. C'est dans ce cadre que je développe des fonctionnalités d'upload de données sur le serveur. Ainsi, j'ai réalisé tous mes développements dans un module Maven inclus dans un projet mutli-modules déjà existant. Mon développement a été réalisé en mode « programmation concurrente » afin de supporter plusieurs requêtes simultanées, et rendre ce service asynchrone. Enfin comme perspectives, les classes et méthodes utilitaires que j'ai développées, seront généralisées afin que le service "Upload" supporte plusieurs autres formats de données (données vectorielles « Shapefile » par exemple).

A l'occasion de ce stage j'ai pu travailler sur plusieurs projets : DataWizard, Crislab, MobiSaaS, etc... J'ai donc participé à différentes phases du cycle de vie d'un projet de développement logiciel : depuis la conception (R&D), au développement de code métier, jusqu'à la livraison au client.

1. <http://www.mobianalyst.fr/>

Abstract

My internship subject focuses on developing web services that expose the functionality of road and transit data import, for automatic build of multi-modal transport networks.

The objective of my work is to expose functionality originally coded in SQL and / or Python via a REST API (JAX-RS). So I initially took over the treatment of warps (ArcPy / Python) and learned to manipulate the data (PostGIS / GTFS) via the project "DataWizard". In parallel, in order to implement these REST web services (Java EE) I joined the project "MobiSaaS" and discovered to develop with the "DropWizard" framework oriented micro-services.

The application developed in the project "MobiSaaS" has an administration interface to expose features of MobiAnalyst package² like a SaaS. It is in this context that I develop features data upload to the server. So, I realized all my development in a Maven module included in an existing mutli-modules project. My development was carried out mode "concurrency" to support multiple concurrent requests, and make the asynchronous service. Finally as prospects, classes and utilities methods I developed will be generalized so that the "Upload" service supports several other data formats (vector data "Shapefile" for example).

During this internship I worked on several projects : DataWizard, Crislab, MobiSaaS, etc ... So I participated in various phases of of software development lifecycle : from the design (R D), to development of business code, until delivery to the customer.

2. <http://www.mobianalyst.fr/>

Sommaire

Sommaire	vi
Table des figures	1
1 Introduction	2
2 Présentation de Mobigis	4
2.1 Domaine d'application	4
2.2 Présentation de l'entreprise	5
2.3 Activités	6
2.4 Historique	6
2.5 Ressources de l'entreprise	8
3 Contexte du stage	10
3.1 Environnement de travail	10
3.2 Outils utilisés	10
3.3 Gestion de projets	11
4 Les Projets	13
4.1 MobiSAAS	13
4.1.1 Présentation	13
4.1.2 Eléments de spécifications fonctionnelles et techniques	13
4.1.3 Développements	15
4.1.4 Résultats obtenus / Difficultés rencontrées	16
4.1.5 Perspectives	16
4.2 DataWizard	17
4.2.1 Présentation	17
4.2.2 Eléments de spécifications fonctionnelles et techniques	18
4.2.3 Développements	18
4.2.4 Résultats obtenus / Difficultés rencontrées	19
4.2.5 Perspectives	20
5 Bilan	22
5.1 Bilan professionnel	22
5.2 Bilan personnel	22

6	Conclusion	23
7	Glossaire et définitions	24
8	Annexes	27
8.1	Apache Maven	27
8.2	DropWizard	28
8.3	Redmine	29
8.4	Extraits de codes	30

Table des figures

2.1	Offres de services Mobigis	5
2.2	Exemples de clients Mobigis	6
3.1	Interface et exemple de requête "GET" SoapUI	11
4.1	Offre du produit MobiAnalyst	13
4.2	Architecture globale.	14
4.3	Table des uploads sur le SGBD PostgreSQL	16
4.4	Exemple d'abstraction et d'héritage de méthode	17
4.5	Schéma des tables de métadonnées	19

Introduction

Ce stage s'inscrit dans le cadre de la formation "Concepteur / Développeur Informatique" délivrée par BGE Haute-Garonne et s'est déroulé durant 3 mois au sein de l'entreprise Mobigis. Les objectifs à l'issue de cette formation sont de savoir concevoir et développer des applications informatiques en utilisant le langage Java/JavaEE, dans un environnement professionnel.

Dans ce mémoire sera présenté le travail que j'ai réalisé, avec par exemple les résultats directs de mon action. Mais aussi, les tâches que j'ai effectuées et les moyens utilisés pour les accomplir : langages, frameworks, logiciels,...

Cette présentation s'organise de la façon suivante :

- Une première partie sera dédiée à la présentation du contexte du stage. Dans cette partie seront présentés le domaine d'application de ce stage : les Systèmes d'Informations Géographiques (SIG), et l'entreprise MobiGIS.
- Dans la deuxième partie, l'environnement de travail du stage sera expliqué : projets, équipes, outils, produits,...
- Une troisième partie présentera la méthodologie de mon travail (conception/développement) et des exemples de réalisations (codes)...
- Enfin, dans la dernière partie de ce mémoire seront présentés un bilan professionnel et un bilan personnel de cette nouvelle expérience.

Par la suite, les termes en **gras** seront définis dans le glossaire en fin du mémoire.

Première partie

Présentation de Mobigis

2.1 Domaine d'application

Les Systèmes d'Informations Géographiques (SIG) sont des outils informatiques permettant de représenter et d'analyser toutes les choses qui existent sur terre ainsi que tous les événements qui s'y produisent. Les SIG offrent toutes les possibilités des bases de données (telles que requêtes et analyses statistiques) et ce, au travers d'une visualisation unique et d'analyse géographique propres aux cartes. Ces capacités spécifiques font du SIG un outil unique, accessible à un public très large et s'adressant à une très grande variété d'applications. Les enjeux majeurs auxquels nous avons à faire face aujourd'hui (environnement, démographie, santé publique...) ont tous un lien étroit avec la géographie. De nombreux autres domaines tels que la recherche et le développement de nouveaux marchés, l'étude d'impact d'une construction, l'organisation du territoire, la gestion de réseaux, le suivi en temps réel de véhicules, la protection civile... sont aussi directement concernés par la puissance des SIG pour créer des cartes, pour intégrer tout type d'information, pour mieux visualiser les différents scénarios, pour mieux présenter les idées et pour mieux appréhender l'étendue des solutions possibles. Les SIG sont utilisés par tous ; collectivités territoriales, secteur public, entreprise, écoles, administrations, états utilisent les SIG. La création de cartes et l'analyse géographique ne sont pas des procédés nouveaux, mais les SIG procurent une plus grande vitesse et proposent des outils sans cesse innovant dans l'analyse, la compréhension et la résolution des problèmes. L'avènement des SIG a également permis un accès à l'information à un public beaucoup plus large (ex : Google Maps). Aujourd'hui, les SIG représentent un marché de plusieurs milliards d'euros dans le monde et emploient plusieurs centaines de milliers de personnes. Le SIG appliqué aux transports, peut être utilisé pour gérer et analyser certaines informations essentielles :

- Planification et analyse des itinéraires
- Localisation et suivi automatiques des véhicules
- Inventaire des arrêts de bus et des infrastructures, gestion des installations ferrées
- Maintenance des voies, du système d'alimentation électrique, des communications et des signaux
- Planification et modélisation des transports

2.2 Présentation de l'entreprise

MobiGIS¹ se présente comme une entreprise au cœur de l'innovation, dont l'activité est d'apporter des réponses sur-mesure aux besoins de ses clients en matière de **géomatique**, notamment lorsqu'elle est appliquée aux enjeux de transport et de mobilité. MobiGIS est une société innovante éditrice de solutions dans le domaine des Systèmes d'Information Géographique (SIG). Elle intervient dans les thématiques : de l'environnement et du développement durable ; de la mobilité des personnes ; du transport et de la logistique (Fig. 2.1).

Ses équipes font de la conception, mise en œuvre et déploiement d'architectures SIG, développement d'applications de cartographie web et mobiles, études de transports, solutions de prévention des risques, etc. L'entreprise est compétente dans l'édition de logiciels SIG, le conseil et les services en SIG, la R&D liée aux NTIC².



FIGURE 2.1 – Offres de services Mobigis

1. site de l'entreprise <http://www.mobigis.fr/>

2. exemples de réalisations <http://www.mobigis.fr/realisations/>

2.3 Activités

L'activité de MobiGIS est centrée sur les services SIG et l'édition de logiciels. Les clients de MobiGIS sont par exemples des industries, la grande distribution, les collectivités locales, et les intégrateurs et société de services en ingénierie informatique (SSII) (Fig. 2.2). L'entreprise développe en particulier une activité d'édition de logiciels. Ceux-ci permettent de faire des analyses multiples : territoire, pollutions, démographie, etc., d'élaborer des plans de déplacement urbain et entreprise, d'étudier et de préparer la réorganisation des réseaux multimodaux et la création de nouvelles infrastructures de transport. Elle compte parmi ses clients des Autorités Organisatrices de Transports (ex :Tisseo), des bureaux d'études, des sociétés de consulting immobilier et des agences d'urbanisme.



FIGURE 2.2 – Exemples de clients Mobigis

2.4 Historique

- 2007 - Création de la société MobiGIS par Frédéric SCHETTINI
 - Le projet de création de société a obtenu le titre de « projet de création d'entreprises innovantes » par le pôle innovation de la Chambre de Commerce et d'Industrie de Toulouse
 - La société MobiGIS est accompagnée par Oséo Midi-Pyrénées
 - MobiGIS obtient le statut de Jeune Entreprise Innovante (JEI)
- 2008 - Mise en place d'une démarche active de développement des activités de MobiGIS en Chine

- Obtention du Prix TOTAL de l'Innovation IT
 - Phase intensive de Recherche Développement
 - Début de collaboration avec le CNRS/LAAS
 - Labellisation du projet POTIMART par la PREDIM
- 2009 - Emménagement sur le site de Grenade-sur-Garonne (31)
 - MobiGIS intègre le groupement CECILE, composé de PME Toulousaines spécialisées dans la géolocalisation
 - MobiGIS rejoint l'association JEInnov
- 2010 - MobiGIS accompagne le ministre des transports en Chine
 - Labellisation par la Plateforme de Recherche et d'Expérimentation pour le Développement de l'Information Multimodale (PREDIM) du projet CAMERA
 - MobiGIS recrute un Volontaire International en Entreprise (VIE) pour intensifier son développement en Chine
 - Soutien de la Région de Midi-Pyrénées pour un Contrat d'appui
- 2011 - Commercialisation du progiciel MobiAnalyst
 - Soutien de la CCIT et d'HGI Tech pour intensifier son développement commercial
 - Recrutement d'un chef de projets SI et d'un ingénieur commercial
- 2012 - L'équipe MobiGIS s'étoffe et compte désormais plus de 10 collaborateurs
 - Prix de l'innovation au Toulouse Space Show et lauréat du concours Open Data « Défi numérique Toulouse Métropole »
 - Ouverture d'un bureau à Paris
 - Adhésion à l'Aerospace Valley
 - Participation à 10 congrès dont l'ITS World à Vienne
- 2013 - Signature de nouveaux contrats avec le groupe Total, Carrefour China, et l'APEM
 - Lancement de la solution Anvio et de la V2.3 de MobiAnalyst
- 2014 - MobiAnalyst© remporte le prix de meilleure application de l'année !
 - Ouverture d'un bureau au Canada pour intensifier l'activité en Amérique du Nord
 - Premier Workshop MobiGIS organisé en septembre 2014

2.5 Ressources de l'entreprise

Les logiciels et les compétences humaines à l'œuvre à MobiGIS sont au cœur de son activité, et même de la réputation de l'entreprise. Les compétences humaines permettent à l'entreprise de mener à bien des projets SIG, l'implémentation d'architecture logicielle, de systèmes de gestion de bases de données spatiales, et enfin de développer des solutions bureautique, serveur, web et mobile.

L'entreprise héberge également les compétences nécessaires pour mener à bien des missions de conseil, d'audit et de consulting. Elle peut établir des états des lieux, des analyses et des préconisations. Enfin, MobiGIS propose des formations sur ses progiciels sur site ou à distance.

Ces ressources peuvent être externalisées auprès des clients qui le souhaitent, de manière à mettre à leur disposition des chefs de projets, des experts SIG, des géomaticiens au profil plus généraliste, ou encore des développeurs. De fait, les employés se déplacent régulièrement chez les clients, le temps de monter les projets, de créer et d'installer les solutions, d'apporter des retouches si nécessaire, et de former les clients à l'utilisation des produits.

Les technologies maîtrisées par MobiGIS sont les SIG propriétaires, les SIG libres, et les langages de programmation logiciel, web et mobile. Parmi les logiciels SIG propriétaires il y a en premier lieu ESRI ArcGIS, mais aussi MapInfo, GeoConcept et Google Maps sont également présents. Actuellement, l'entreprise s'inscrit dans la tendance de nombreux éditeurs, de ne plus seulement proposer des solutions desktop (qui imposent d'installer des logiciels sur son poste de travail, etc...) mais aussi de proposer des solutions à distance, plus souples (et moins onéreuses pour l'acheteur), notamment sur le modèle d'ArcGIS Online³. Les SIG libres sont également très présents dans les projets dont notamment les logiciels QGIS, PostGIS, OpenLayers, ou encore GeoServer.

Les langages "objets" maîtrisés par les développeurs et géomaticiens de l'entreprise sont divers dont notamment Java, C++, C, Python... Ils permettent de programmer les progiciels de l'entreprise. L'équipe MobiGIS pratique également les langages web du moment : HTML 5, JavaScript, PHP, CSS 3, etc.. Ils développent sur les principaux supports mobiles : IOS et Android, qui prennent aujourd'hui une importance croissante dans le monde des SIG en raison de l'évolution des pratiques liés à l'utilisation des smartphones et des tablettes.

3. http://www.esrifrance.fr/ArcGIS_Online_1.aspx

Deuxième partie

Contexte du stage

3.1 Environnement de travail

Durant le stage j'ai travaillé sur un pc de marque Dell, dont le système d'exploitation est Windows 8.1 Professionnal (machine hôte). L'entreprise travaille avec de nombreuses machines virtuelles hébergées ou distantes afin de disposer d'environnement de tests, de développements, et de production. Pour cela, j'avais à ma disposition une machine virtuelle de développement via VirtualBox avec le système d'exploitation Windows Server 2012 R2 Standard.

3.2 Outils utilisés

Quotidiennement j'ai utilisé 2 environnements de développement intégré (IDE) : **Liclipse** (pour développer en langage Python) dans le projet DataWizard, et **Eclipse** version Mars (pour développer en langage Java/Java EE) pour les projets MobiSaaS et Crislab.

J'ai manipulé plusieurs **SGBD** : MongoDB (MobiSaaS), Oracle (Crislab), mais j'ai principalement travaillé avec le couple **Postgresql/Postgis** afin de gérer les données géographiques dans les 2 projets "DataWizard" & "MobiSaaS".

Afin de tester les web services REST développés, j'ai utilisé fréquemment l'outil **SoapUI** (Fig. 3.1). Il permet de mettre en place une suite de tests qui peuvent être lancés d'une traite du côté client, permet de tester les services web en mode bouchon mais aussi d'effectuer des tests de charge. Il permet entre autre de fournir une hiérarchie des services web, de lister les différentes méthodes disponibles, les paramètres attendus, de réaliser des requêtes et de récupérer les réponses ... C'est l'un des meilleurs outils de test unitaires concernant les services web.

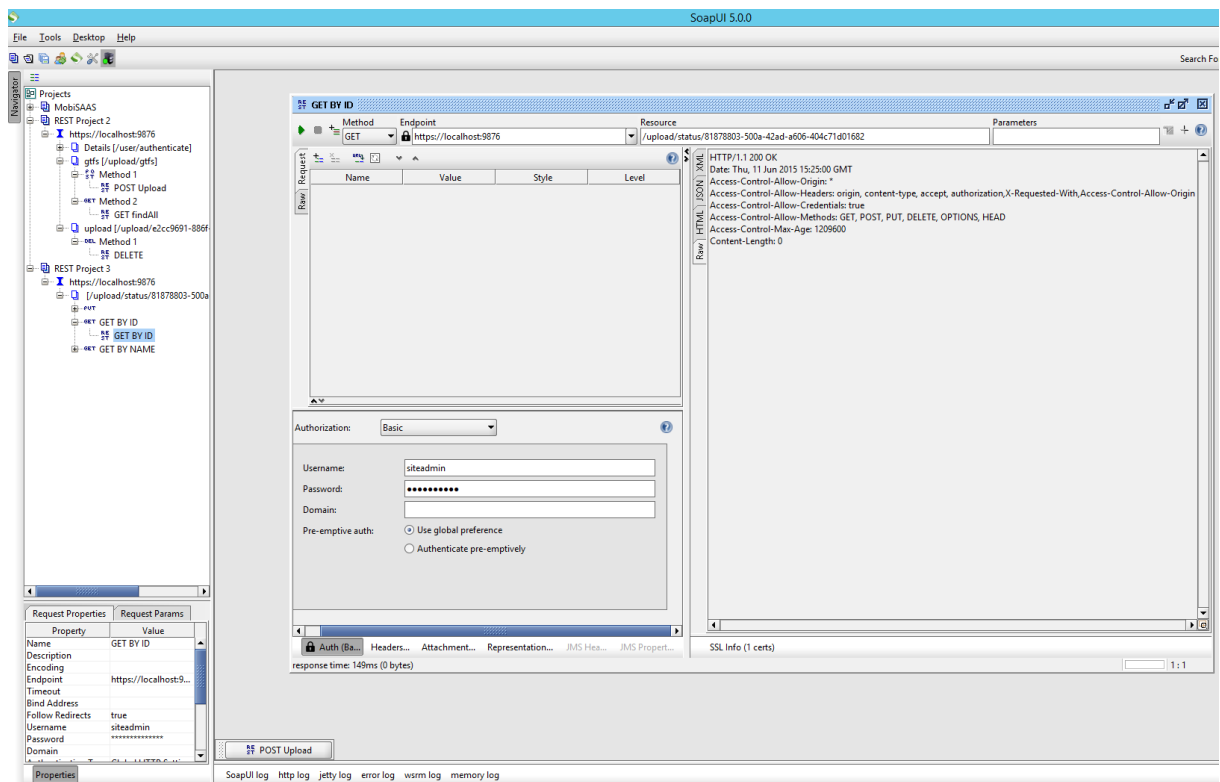


FIGURE 3.1 – Interface et exemple de requête "GET" SoapUI

3.3 Gestion de projets

Dans les projets auxquels j'ai participé, l'entreprise utilise des outils de gestion : planning, suivi de bugs, outils de mutualisation, gestion de versions. Ainsi, j'ai utilisé l'outil Trello pour la gestion des tâches, Redmine pour la gestion des projets (cf. Annexe 8.3), et SVN pour la gestion des codes sources. L'entreprise met à disposition de ses salariés un intranet avec de nombreux outils collaboratifs sous la plateforme eGroupware (Feuille de temps, etc...).

Un point de suivi informel était effectué plusieurs fois par semaine avec l'encadrant afin de présenter le travail effectué, les résultats intermédiaires, et le travail planifié pour la semaine suivante. Un bilan à la mi-stage a été effectué afin de réajuster les priorités, et arriver à produire un livrable satisfaisant en fin de stage. De plus, de nombreuses tâches de soutien aux équipes arrivaient au fil de l'eau : productions de réseaux de transport, tests, etc...

Troisième partie

Les Projets

4.1 MobiSAAS

4.1.1 Présentation

Le projet "MobiSAAS" : MobiAnalyst as a Service, s'inscrit dans la démarche d'entreprise de proposer des solutions en mode **SAAS**. Le projet consiste d'une manière générale à exposer les fonctionnalités de la solution desktop du produit "MobiAnalyst"(Fig. 4.1).

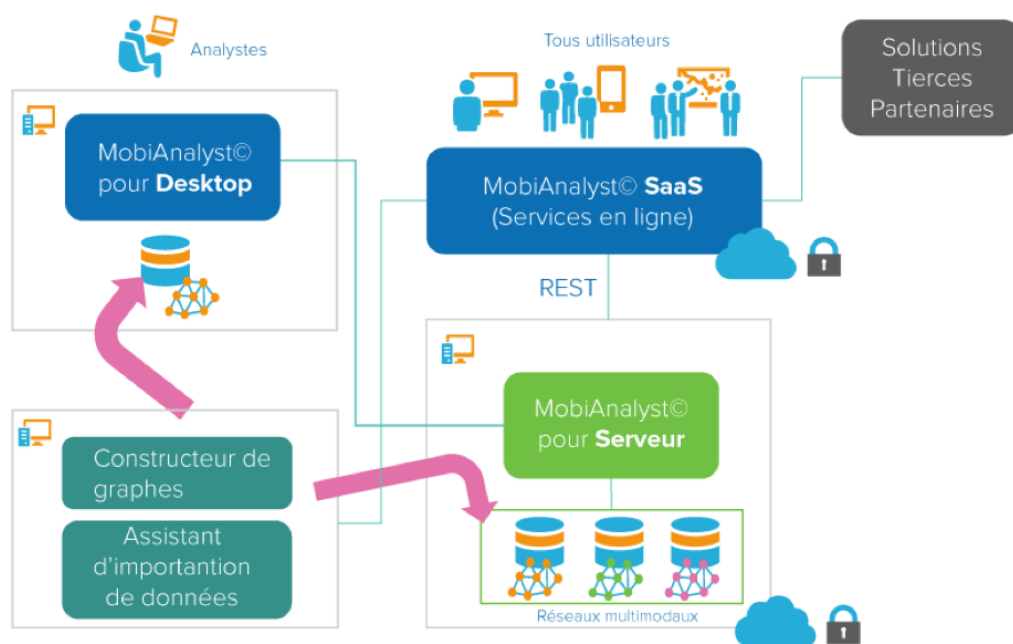


FIGURE 4.1 – Offre du produit MobiAnalyst

4.1.2 Eléments de spécifications fonctionnelles et techniques

Architecture actuelle

La figure 4.2 détaille l'ensemble des briques logicielles nécessaires au fonctionnement de la plate-forme MobiSAAS :

- **AGS** : ArcGIS Server. Le serveur de Système d'Information Géographique (SIG) permettant de publier et d'administrer des services cartographiques (MapService). Dans notre cas, les MapServices déployés sont des réseaux routiers et de transport en commun (TC), accompagnés de tables horaires (TimeTable) contenant les horaires des TC.
- **SOE** : Server Object Extension. Nous utilisons le mécanisme d'extension "SOE" pour déployer sur un MapService un service spécifique à MobiSAAS. Un SOE est une fonctionnalité qui se déploie sur un MapService et qui expose en Web Service (REST ou SOAP) les solveurs (algorithme de résolution d'itinéraires) de MobiAnalyst.
- **MobiAdmin** : Serveur REST d'administration MobiSAAS. Ce serveur (basé sur Java) est le point d'entrée des utilisateurs des services REST de MobiAnalyst. Il redirige les requêtes métiers vers le bon SOE déployé, il trace ces requêtes et enrichit la base de données client de MobiAnalyst. L'authentification qui est faite dans les requêtes utilise les comptes d'AGS créés au préalable.
- **Postgres** : Système de gestion de base de données. Cette base contient les données clients de MobiAnalyst : profil, traces d'utilisation de MobiSAAS.
- **MongoDB** : Système de gestion de base de données. Cette base contient les logs (statistiques mesurées en temps réel) correspondant aux services REST de MobiSAAS.

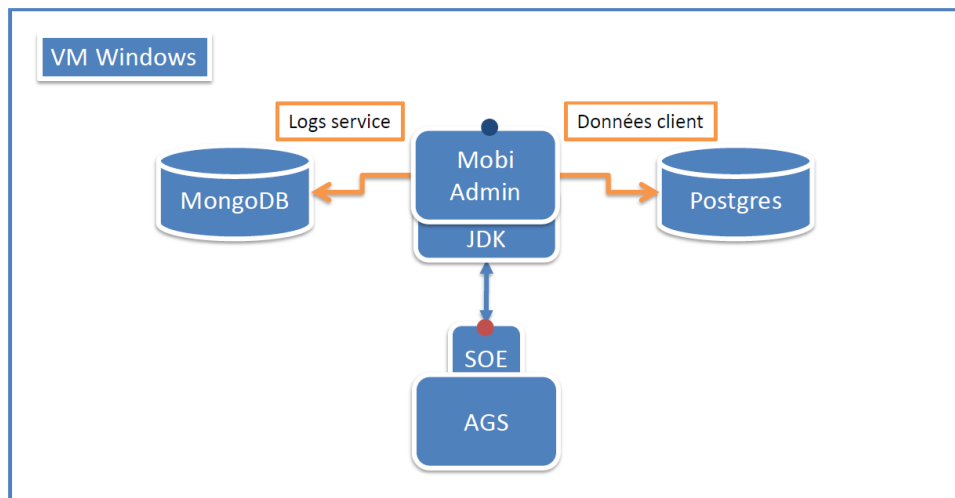


FIGURE 4.2 – Architecture globale.

Pour le composant "MobiAdmin" le choix technologique majeur est d'utiliser le framework DropWizard (cf. Annexe 8.2). Ce framework orienté micro-services nous permet de fournir notamment un serveur embarqué HTTP "Jetty". Jersey pour la partie webservice REST qui est l'implémentation de référence de la spécification JAX-RS. Ou encore Jackson pour la dé/sérialisation du JSON.

4.1.3 Développements

Mon stage et les développements demandés concernent le composant d'administration de l'infrastructure : "MobiAdmin". Dans l'objectif de gérer les données du client, je dois proposer un service de gestion des données de transport dans l'espace privatif du client. La fonctionnalité principale à développer est un "Upload de données" et plus particulièrement l'upload de données GTFS¹.

L'objectif de ce service d'import de données GTFS est de pouvoir manipuler ces données (fichier au format zip) (comme le fait le DataWizard) et ainsi pouvoir récupérer des métadonnées ex : l'extension géographique des données, le nom de l'agence, le nombre de lignes, le mode de transport,...

Une archive "Upload" peut contenir un seul ou plusieurs jeu de données GTFS (ensemble de fichiers .txt). Il faut donc récupérer et renseigner les métadonnées de chaque upload de fichier :

- Nom
- Date d'upload ou de la version du GTFS
- Licence (open data/ restreint/ privé)
- Url si open data
- Commentaire libre
- Chemin vers la données
- etc...

L'environnement de développement est donc un projet Java EE Maven composé des éléments de Dropwizard.

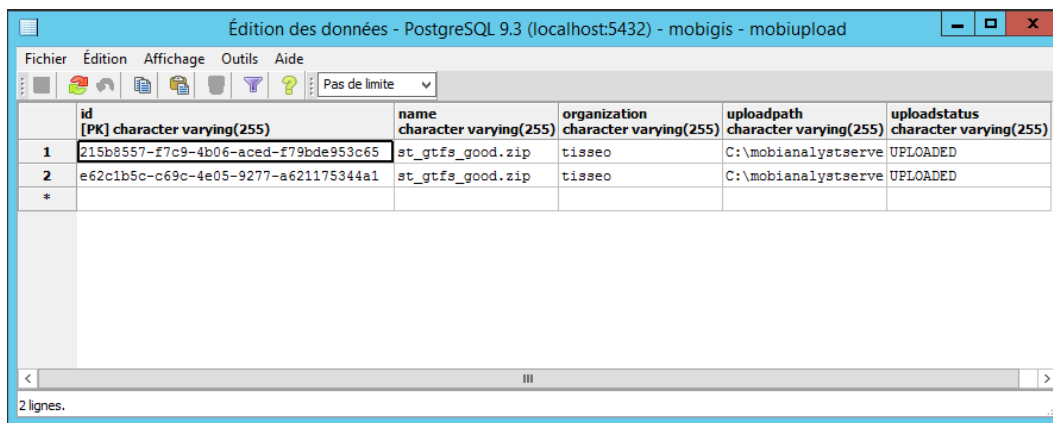
Ces WS sont à priori "lourds", sachant qu'un jeu de données peut faire jusqu'à plusieurs centaines de Mo, les opérations de téléchargement, validation, traitement, stockage dans en base,... vont être long à renvoyer une réponse au client après chaque requête. Les WS à développer sont donc asynchrones. De plus, une contrainte supplémentaire et que le service doit supporter plusieurs requêtes simultanées, il faudra donc s'orienter vers un développement en mode "programmation concurrente".

1. <https://developers.google.com/transit/gtfs/reference>

4.1.4 Résultats obtenus / Difficultés rencontrées

J'ai tout d'abord commencé par découvrir Maven (cf. Annexe 8.1), les projets, les modules, le désormais célèbre fichier **POM**, etc... Ensuite, je me suis documenté sur le code métier existant, et chercher des outils ou librairies de professionnels du domaine (cf. Outils métiers 8.4). Enfin, après le "Getting Started" de Dropwizard², une fois tout cela bien maîtrisé, j'ai pu commencé la conception et le développement de services web REST.

J'ai choisit de stocker les informations concernant ma partie de l'application dans une table PostgreSQL (Fig 4.3)



	id [PK] character varying(255)	name character varying(255)	organization character varying(255)	uploadpath character varying(255)	uploadstatus character varying(255)
1	215b8557-f7c9-4b06-aced-f79bde953c65	st_gtfs_good.zip	tisseo	C:\mobianalystserve	UPLOADED
2	e62c1b5c-c69c-4e05-9277-a621175344a1	st_gtfs_good.zip	tisseo	C:\mobianalystserve	UPLOADED

FIGURE 4.3 – Table des uploads sur le SGBD PostgreSQL

A chaque requête POST du client, je crée une instance d'objet Upload, les informations sont saisies dans la table de suivi, et je stocke les données envoyées sur le serveur. Pour chaque client ou utilisateur, pour chaque type de données, il y a un répertoire personnalisé, chaque upload est "taggé" d'un identifiant unique "UUID". Le code métier qui a été implémenté est pour le moment le test sur le type de données envoyées, la validation des données et la production d'un rapport d'un validation (JSON), enfin l'extraction (récursive) des données contenues dans l'archive(cf. Annexe 8.4).

4.1.5 Perspectives

L'idéal d'un programme développé dans un langage orienté objet, est la généricité, et la réutilisation d'un maximum de composants. Dans ce travail, l'objectif est clairement de produire un maximum de composant abstrait (classes et interfaces) et de méthodes réutilisables.

2. <https://dropwizard.github.io/dropwizard/getting-started.html>

Les perspectives du projet "MobiSAAS" sont nombreuses : gérer une architecture distribuée, augmenter les fonctionnalités SAAS notamment les principales du projet "DataWizard".

J'ai donc débuté le développement de ces composants abstraits (ex : Les ressources au sens DropWizard (Fig. 4.4). Grâce à cela on pourra donc "uploader" plusieurs types de données et réutiliser la méthode `manageUpload()`. Egalement, sur ce même principe j'ai développé une classe abstraite afin de faire proposer la gestion des threads et tous mes services en hérite, cela limite la duplication de codes dans l'application.

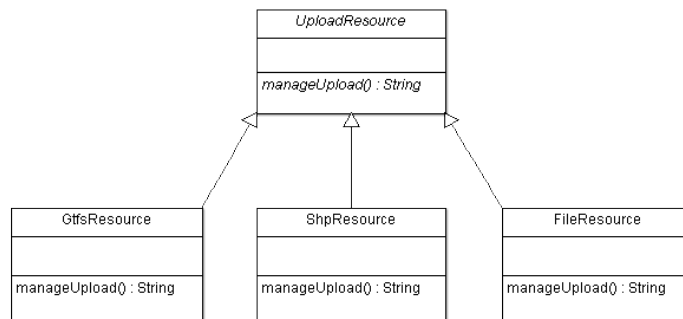


FIGURE 4.4 – Exemple d'abstraction et d'héritage de méthode

4.2 DataWizard

4.2.1 Présentation

Le projet "DataWizard", ensemble de scripts Python et SQL est une application dont le but est d'ingérer des données de transports, et de voirie dans une base de données spatiales ([PostgreSQL/PostGIS7](#)) afin de produire en sortie un réseau de transport (Network Dataset ou NDS).

4.2.2 Éléments de spécifications fonctionnelles et techniques

Le DataWizard permet de se lancer par étapes. Il y a 10 étapes successives, dont une étape préliminaire appelée étape 0, et une indépendante : l'étape 10.

- STEP 0 : Nettoyage et préparation de la base de données
- STEP 1 : Import des données transport en commun et conversion vers le schéma mobianalyst
- STEP 2 : Import des données de voirie
- STEP 3 : Import des données métro et conversion vers le schéma mobianalyst
- STEP 4 : Génération des données vitesses moyennes et horaires et connexion des réseaux de transport en commun et voirie
- STEP 5 : Calcul des horaires et des vitesses moyennes
- STEP 6 : Export des données vers des fichiers Shapefile (.shp)
- STEP 7 : Import des données dans la Géodatabase de sortie et changement du système de projection si nécessaire
- STEP 8 : Extraction des différents modes, génération du fichier xml servant à construire le réseau et de MobiNetwork.xml
- STEP 9 : Construction et compilation du réseau
- STEP 10 : Production de métadonnées pour le réseau

Afin de fournir, un réseau conforme aux attentes du logiciel MobiAnalyst "version 3.0" et des experts en analyse de réseaux de transports, l'équipe me fournit une spécification de métadonnées à extraire des données en entrée du DataWizard (Fig. 4.5).

4.2.3 Développements

J'intègre tout d'abord le projet en tant qu'utilisateur, j'exploite des données et produits des réseaux (Bordeaux, Champagne-Ardenne, Montreal, Melbourne, etc...) Ensuite, petit à petit je suis le plan de charge issu du projet Redmine et corrige des bugs, et anomalies.

Mon premier travail a été de développer l'étape 10 (STEP 10 : Export Metadata Tables) de l'exécution du DataWizard. Pour cela j'ai dû extraire le nombre de mode de transports en commun (variable selon chaque projet), leur code (variable selon chaque projet) et enfin extraire des listes de constantes pour documenter le réseau produit par le DataWizard.

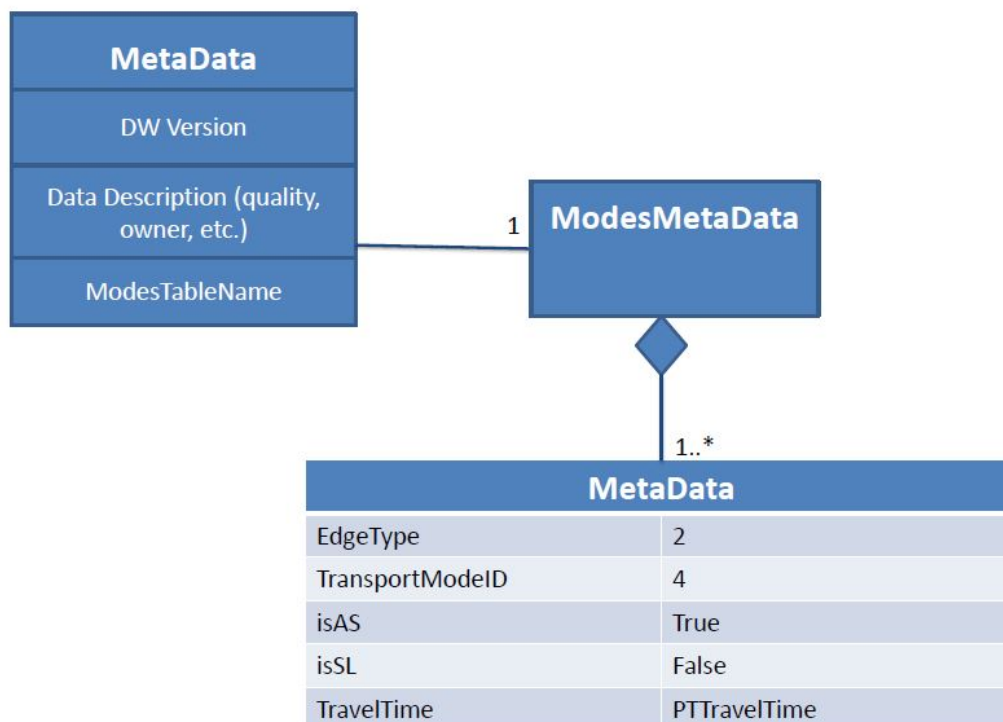


FIGURE 4.5 – Schéma des tables de métadonnées

Exemple ?

Un autre travail réalisé est le développement qui permet la gestion du système de projection des données géographiques pour les calculs du DataWizard. Cette modification impacte quasiment la totalité des requêtes SQL dites spatiales de l'application, et permet des résultats (calculs de distances) plus réalistes (par exemple pour Melbourne, Australie située dans l'hémisphère Sud).

4.2.4 Résultats obtenus / Difficultés rencontrées

Les principales difficultés concernent les données. Ces données sont complexes, les données GTFS ou les données de voirie (données Here (anciennement Navteq) ou OpenStreetMap³). La base de données "DataWizard" possède ainsi de nombreux schémas, et les requêtes SQL sont parfois très complexes mêlant fonctions, et requêtes géographiques.

3. <http://openstreetmap.fr/>

Les résultats obtenus sont la production de nombreux réseaux en version 3.0 (avec les métadonnées) pour nos analystes, et pour les projets nécessitant des réseaux récents (Moveazy, Mobilyse, etc...). Et une version de l'application qui je l'espère est plus stable.

Dans ce projet, j'ai également développé des fonctions utilitaires : BOMConverter.py, searchAndReplaceFiles.py, etc... Ces méthodes permettent notamment de formater les données avant leur import dans la base de données. Cela peut certainement éviter les mauvaises surprises d'encodage ou de caractères spéciaux fréquemment rencontrés lors de l'exploitation de données GTFS.

4.2.5 Perspectives

La liste des évolutions est longue, il y a en effet énormément de perspectives pour le projet... Tout d'abord permettre l'import de plusieurs formats de données, comme l'import de données Shapefile, mais aussi de données de transports en commun "Trident"... Mais aussi d'une manière générale optimiser les étapes, et développer des routines de simplification de données (afin de ne retenir que les données nécessaires),...

Quatrième partie

5.1 Bilan professionnel

Actuellement, la tendance dans le milieu de l'industrie informatique est le développement d'applications web. Grâce à ce stage au sein de l'entreprise MobiGIS, j'ai pu acquérir une expérience dans le développement de sites web dynamiques grâce à la puissance de la technologie Java EE. De plus ce langage étant très utilisé dans le monde de l'industrie « Web », cette expérience est pour moi très valorisante.

De plus, la pratique quotidienne du langage SQL m'a permis de combler certaines de mes lacunes dans le domaine de gestion des bases de données relationnelles.

5.2 Bilan personnel

Ces 3 mois de stage ont été très enrichissant car ils m'ont permis de découvrir le monde de l'industrie informatique, et d'appliquer mes compétences de géomaticien. En effet, après 7 années d'expériences dans le domaine de la recherche publique, j'ai pu intégré une équipe dynamique, des projets concrets, dans une entreprise privée.

Ce stage a été pour moi l'occasion de pratiquer la technologie Java EE, langage largement répandu dans le monde industriel. De réaliser ce rapport avec le langage \LaTeX .

Conclusion

Pour conclure ce rapport, et d'un point de vue "académique" les compétences suivantes peuvent être mises en avant suite à cette expérience :

- Pour l'activité «Développer des composants d'interface» :
 - Maquetter une application
 - Développer une interface utilisateur
 - Développer des composants d'accès aux données

- Pour l'activité «Développer la persistance des données» :
 - Utiliser l'anglais dans son activité professionnelle en informatique

- Pour l'activité «Développer une application n-tiers» :
 - Concevoir une application
 - Développer des composants métier
 - Construire une application organisée en couches

Glossaire et définitions

API : En informatique, une interface de programmation (souvent désignée par le terme API pour Application Programming Interface) est un ensemble normalisé de classes, de méthodes ou de fonctions qui sert de façade par laquelle un logiciel offre des services à d'autres logiciels. Elle est offerte par une bibliothèque logicielle ou un service web, le plus souvent accompagnée d'une description qui spécifie comment des programmes consommateurs peuvent se servir des fonctionnalités du programme fournisseur.

Des logiciels tels que les systèmes d'exploitation, les systèmes de gestion de base de données, les langages de programmation, ou les serveurs d'applications comportent une interface de programmation ¹.

Eclipse : L'environnement de programmation (IDE) en langage Java le plus connu est le projet "Eclipse" de la fondation Eclipse. Ce logiciel simplifie la programmation grâce à un certain nombre de raccourcis et notamment grâce à la possibilité d'intégrer de nombreuses extensions. Au fur et à mesure de l'avancement du code, Eclipse compile automatiquement le code et signale les problèmes qu'il détecte.

Géomatique : La géomatique est la combinaison syntaxique de deux mots : Géographie et Informatique. Le mot géomatique a été déterminé pour regrouper de façon cohérente l'ensemble des connaissances et technologies nécessaires à la production et au traitement des données numériques décrivant le territoire, ses ressources ou tout autre objet ou phénomène ayant une position géographique. La géomatique est un domaine qui fait appel aux sciences, aux technologies de mesure de la terre ainsi qu'aux technologies de l'information pour faciliter l'acquisition, le traitement et la diffusion des données sur le territoire (aussi appelées "données spatiales", "données géospatiales" ou "données géographiques"). La géomatique est étroitement liée à l'information géographique qui est la représentation d'un objet ou d'un phénomène localisé dans l'espace. Ainsi, la géomatique regroupe l'ensemble des outils et méthodes permettant de représenter, d'analyser et d'intégrer des données géographiques ².

1. http://fr.wikipedia.org/wiki/Interface_de_programmation

2. <http://www.sig-geomatique.fr/sig-geomatique.html>

Java : Java est un langage orienté objet, c'est-à-dire que le programme est vu comme un ensemble d'entités (de classes). Au cours de l'exécution du programme, les entités collaborent entre elles pour arriver à un but commun.

POM : Chaque projet ou sous-projet est configuré par un POM "Project Object Model" qui contient les informations nécessaires à Maven pour traiter le projet (nom du projet, numéro de version, dépendances vers d'autres projets, bibliothèques nécessaires à la compilation, noms des contributeurs etc.). Ce POM se matérialise par un fichier pom.xml à la racine du projet. Cette approche permet l'héritage des propriétés du projet parent. Si une propriété est redéfinie dans le POM du projet, elle recouvre celle qui est définie dans le projet parent. Ceci introduit le concept de réutilisation de configuration. Le fichier pom du projet principal est nommé pom parent. Il contient une description détaillée de votre projet, avec en particulier des informations concernant le versionnage et la gestion des configurations, les dépendances, les ressources de l'application, les tests, les membres de l'équipe, la structure et bien plus.

PostgreSQL : PostgreSQL est un système de gestion de bases de données relationnelles objet (Manuel PostgreSQL). PostgreSQL est un outil Open Source et disponible gratuitement, compatible avec les systèmes d'opérations les plus connus (Linux, Unix (Mac OSX, Solaris etc.) et Windows). PostgreSQL propose des interfaces de programmations pour des langages de programmation comme Java, C++, Python etc. Le développement de PostgreSQL a débuté en 1986 (appelé à l'époque Postgres). En 1995, les développeurs ajoutent un interpréteur de langage SQL à l'outil. A partir de 1996, l'outil s'appelle PostgreSQL afin de souligner le lien entre Postgres et le langage SQL. PostgreSQL peut être facilement étendu par l'utilisateur en ajoutant de nouvelles fonctions, de nouveaux opérateurs ou même de nouveaux langages de procédure.

PostGIS : PostGIS est une extension du système de gestion de base de données PostgreSQL qui permet de stocker des données (objets) géographiques dans la base de données. Cette extension permet d'utiliser une base de données PostgreSQL comme une base de données dans n'importe quel projet SIG. PostGIS est compatible avec de nombreux autres outils SIG comme par exemple QGIS, Mapserver, etc...

REST : REST "Representational State Transfer" (REST) est un style d'architecture logicielle comprenant des lignes directrices et des meilleures pratiques pour la création de services Web évolutifs. REST est un ensemble coordonné de contraintes appliqué à la conception de composants dans un système hypermédia distribué qui peut conduire à une architecture plus performante et maintenable.

REST a gagné sa réputation à travers le web comme une alternative plus simple à SOAP

et des services basés sur un WSDL. Les systèmes "RESTful" peuvent généralement, mais pas toujours, communiquer avec les verbes du protocole HTTP (GET, POST, PUT, DELETE, etc.) utilisés par les navigateurs Web pour récupérer des pages Web et envoyer des données à des serveurs distants.

SAAS : SAAS "Software as a Service" Le logiciel en tant que service est un modèle d'exploitation commerciale des logiciels dans lequel ceux-ci sont installés sur des serveurs distants plutôt que sur la machine de l'utilisateur. Les clients ne paient pas de licence d'utilisation pour une version, mais utilisent généralement gratuitement le service en ligne ou payent un abonnement récurrent³.

SoapUI : SoapUI est outil graphique qui permet de tester des services web basés sur diverses technologies. Il est disponible en deux versions : une version gratuite et open source et seconde version payante. Il est également disponible sous forme de plugin pour les IDE Netbeans, IntelliJ IDEA et Eclipse. SoapUI est développé entièrement en Java et utilise Java Swing pour son GUI, il fonctionne donc sur la plupart des systèmes d'exploitation et en plus il est disponible sous licence GNU. L'outil gère respectivement les services web basés sur les technologies telles que le HTTP (S), HTML, SOAP (WSDL), REST, AMF, JDBC et JMS.

SVN : SVN ou Subversion est un système de gestion de version, conçu pour remplacer CVS. Concrètement, ce système permet aux membres d'une équipe de développeur de modifier le code du projet quasiment en même temps. Le projet est en effet enregistré sur un serveur SVN et à tout moment, le développeur peut mettre à jour une classe avant de faire des modifications pour bénéficier de la dernière version et a la possibilité de comparer deux versions d'un même fichier.

WS : Acronyme de "Web Service" ou Service Web. Un WS est un programme informatique de la famille des technologies web 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, de manière synchrone ou asynchrone. Actuellement, le protocole de transport est essentiellement HTTP(S)⁴.

3. http://fr.wikipedia.org/wiki/Logiciel_en_tant_que_service

4. http://fr.wikipedia.org/wiki/Service_web

Annexes

8.1 Apache Maven

Apache Maven est 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.

Il est semblable à l'outil Ant, mais fournit des moyens de configuration plus simples, eux aussi basés sur le format XML. Maven est géré par l'organisation Apache Software Foundation. Précédemment Maven était une branche de l'organisation Jakarta Project.

Maven utilise un paradigme connu sous le nom de Project Object Model (POM) afin de décrire un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. Il est livré avec un grand nombre de tâches pré-définies, comme la compilation de code Java ou encore sa modularisation.

Un élément clé et relativement spécifique de Maven est son aptitude à fonctionner en réseau. Une des motivations historiques de cet outil est de fournir un moyen de synchroniser des projets indépendants : publication standardisée d'information, distribution automatique de modules « jar ». Ainsi en version de base, Maven peut dynamiquement télécharger du matériel sur des dépôts logiciels connus. Il propose ainsi la synchronisation transparente de modules nécessaires.

8.2 DropWizard

Dropwizard est un framework Java léger adapté au développement rapide de microservices REST et ne nécessitant pas de serveur d'application comme environnement d'exécution. Cela dit, au delà du framework, c'est surtout un assemblage habile de composants spécialisés parmi les meilleurs de l'écosystème Java :

- **Jetty**, un serveur HTTP et un moteur de servlet compacts et très performants
- **Jersey**, l'implémentation de référence de la spécification JAX-RS (web services REST)
- **Jackson**, une librairie de sérialisation/dé-sérialisation JSON
- **Hibernate Validator**, l'implémentation de référence de l'API Bean Validation (JSR 303)
- **SLF4J** et **Logback** pour la gestion des traces
- **Metrics** pour le monitoring
- **jDBI** pour l'interfaçage rapide à une base de données relationnelle. Cette librairie est de bien plus bas niveau que JPA ou Hibernate et présente peu d'abstraction ce qui rend sa prise en main aisée

On peut considérer ce projet comme une alternative crédible aux serveurs d'applications Java EE perçus comme lourds, compliqués et gourmands en ressources. Le champ des applications couvert par Dropwizard est en réalité plus vaste que celui des microservices (il est tout à fait possible de développer une IHM web) mais l'aisance avec laquelle on développe et déploie un service REST en fait une solution très adaptée à ce type d'usage (à l'instar de Spring Boot de Pivotal ou Spark avec lesquels il est en compétition). Packagée sous la forme d'un jar autonome contenant toutes ses dépendances, l'unité de déploiement n'a pas besoin de serveur d'application pour être exécutée (le conteneur Jetty est embarqué dans le jar). Avec ses 10 Mo tout au plus (dépendances comprises) l'empreinte mémoire d'une application Dropwizard est donc incomparablement plus faible qu'un Web Service SOAP déployé dans un serveur d'application Java EE (jusqu'à plusieurs centaines de Mo). En conséquences, le temps de démarrage d'une application Dropwizard est de quelques secondes quand il faut parfois plusieurs minutes pour un serveur d'application.

8.3 Redmine

Redmine est considéré comme l'un des outils de gestion de projets collaboratifs Open Source parmi les plus aboutis. Il recouvre un ensemble de fonctionnalités dont un aperçu est donné ci-dessous, comme la gestion multi-projets, la gestion des demandes d'évolution et des bugs, la gestion et l'indexation des documentations techniques, mais aussi la gestion des droits et des profils des différents intervenants. Il propose aussi une console de suivi de l'état d'avancement des projets, des tâches, des recettes... sous forme de diagramme de Gantt.

Redmine est une forge logicielle sous licence GPL. Ses principaux concurrents se nomment Trac, Retrospectiva, Django Projector ou encore InDefero. Notons que Jira ne trouve pas sa place ici, bien qu'il soit le concurrent le plus sérieux de Redmine, parce que Jira est sous Dual licence (l'utilisation commerciale nécessite une licence payante non-Open Source).

Redmine offre un ensemble de fonctionnalités comme par exemples :

Prise en charge de plusieurs projets ;

Contrôle d'accès avec un modèle flexible de rôles ;

Gestion avancée des tickets ;

Diagramme de Gantt et calendrier ;

Publication de news, documents et gestionnaire de fichiers ;

Notifications par emails et flux ATOM ;

Wiki et forums par projet ;

Outil de suivi du temps ;

Champs personnalisables pour les tickets, suivi de temps, projets et utilisateurs ;

Intégration avec plusieurs SCM : SVN, CVS, Git, Mercurial, Bazaar et Darcs ;

Une communauté active et un ensemble d'outils connexes itemsize

Plus de détails est disponible sur le wiki du projet : <http://www.redmine.org/projects/redmine/wiki>.

8.4 Extraits de codes

Instanciation d'un objet upload

```
// Instanciation
Upload up = new Upload();
up.setId(CreateUUID.randomUUID().toString());
up.setOrganization(user.getOrganization());
up.setName(fileDetails.getFileName());
```

La classe Upload.java associée

```
// Empty constructor
public Upload (){
}

// Constructor
public Upload(String id, String name, String uploadStatus, String
    organization, String uploadPath){
    super();
    this.id=id;
    this.name = name;
    this.uploadStatus = uploadStatus;
    this.organization = organization;
    this.uploadPath = uploadPath;
}

// Getters & Setters
public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}
```

```
public String getName() {  
    return name;  
}  
  
public void setName(String name) {  
    this.name = name;  
}  
  
public String getUploadStatus() {  
    return uploadStatus;  
}  
  
public void setUploadStatus(String uploadStatus) {  
    this.uploadStatus = uploadStatus;  
}  
  
public String getOrganization() {  
    return organization;  
}  
  
public void setOrganization(String organization) {  
    this.organization = organization;  
}  
  
public String getUploadPath() {  
    return uploadPath;  
}  
  
public void setUploadPath(String uploadPath) {  
    this.uploadPath = uploadPath;  
}  
  
    // suite des Getters & Setters
```

Classe UploadDAO.java

```

    // Create
    public String create(Upload up)
    {
        System.out.println("Upload instance created : "+up.getId()+" | "+up.getName()
            +" | "+up.getUploadStatus()+" | "+up.getOrganization()+" | "+up.
            getUploadPath());
        return persist(up).getId();
    }

    // Read one
    public Upload findByID(String id)
    {
        //return get(id);
        Query query = namedQuery("com.mobigis.dao.Upload.findByID");
        query.setParameter("id", id);
        List<Upload> res = list(query);
        if(res != null && res.size()>0)
        {
            return res.get(0);
        }
        return null;
    }

```

Extrait du log Eclipse

```

INFO [2015-06-11 15:31:52,883] com.mobigis.thread.AbstractUploadManagerThread:
    GTFS validation JSON report done !
file unzip : agency.txt
file unzip : calendar.txt
file unzip : calendar_dates.txt
file unzip : fare_attributes.txt
file unzip : fare_rules.txt
file unzip : frequencies.txt
file unzip : routes.txt
file unzip : shapes.txt
file unzip : stop_times.txt

```



```

file unzip : stops.txt
file unzip : trips.txt
Extract Done
Clean directory Done
upload status = UPLOADED
INFO [2015-06-11 15:31:52,920] com.mobigis.thread.AbstractUploadManagerThread:
    Process upload to Location = C:\mobianalystserver\download\tisseo\gtfs\215
    b8557-f7c9-4b06-aced-f79bde953c65
INFO [2015-06-11 15:31:52,920] com.mobigis.thread.AbstractUploadManagerThread:
    Upload file done ! Id = 215b8557-f7c9-4b06-aced-f79bde953c65 | Status =
    UPLOADED
C:\mobianalystserver\download\tisseo\gtfs\215b8557-f7c9-4b06-aced-
    f79bde953c65
INFO [2015-06-11 15:31:52,965] org.hibernate.engine.internal.
    StatisticalLoggingSessionEventListener: Session Metrics {
    36877 nanoseconds spent acquiring 1 JDBC connections;
    0 nanoseconds spent releasing 0 JDBC connections;
    358836 nanoseconds spent preparing 4 JDBC statements;
    6286839 nanoseconds spent executing 4 JDBC statements;
    0 nanoseconds spent executing 0 JDBC batches;
    0 nanoseconds spent performing 0 L2C puts;
    0 nanoseconds spent performing 0 L2C hits;
    0 nanoseconds spent performing 0 L2C misses;
    24185654 nanoseconds spent executing 1 flushes (flushing a total of 2
        entities and 2 collections);
    42650 nanoseconds spent executing 1 partial-flushes (flushing a total of 0
        entities and 0 collections)
    }
127.0.0.1 -- [11/Jun/2015:15:31:50 +0000] "POST /upload/gtfs HTTP/1.1" 200 61 "-" "
    Apache-HttpClient/4.1.1 (java 1.5)" 2254

```

Webographie

Sites sur les données/outils "métiers" :

<https://developers.google.com/transit/gtfs/examples/gtfs-feed>

<https://developers.google.com/transit/gtfs/>

<https://github.com/google/transitfeed/wiki/FeedValidator>

<http://onebusaway.org/developer-information/>

<https://github.com/OneBusAway/onebusaway/wiki/Importing-source-code-into-Eclipse>

Sites sur les frameworks et outils utilisés :

<http://www.objis.com/formation-java/tutoriel-formation-maven-2.html>

<http://mvnrepository.com/>

<https://github.com/bucharest-jug/dropwizard-todo>

<http://www.tutorialspoint.com/hibernate/>

<http://blog.xebia.fr/2011/11/14/rest-java-serveur/>

Et pour finir, un grand merci à <http://www.mkyong.com/>, tout simplement.