Formation : **Concepteur Développeur Informatique**

Session 2015 / 2016

RAPPORT DE STAGE

CONSEIL DÉPARTEMENTAL

Sommaire :

1. REMERCIEMENTS

Je tiens tout d'abord à remercier :

Madame **Alexandra** HERY, Madame **Sophie** LETY, Madame **Magali** FROUILLOU, Monsieur **Alaa** BOUHATAR, Monsieur **Laurent** CORTIAL, Monsieur **Laurent** THAON, Monsieur **Yann** DELANAUX, Monsieur **Ludovic** PETERHANSEL, Monsieur **Florent** THEVENET & Monsieur **Yohann** LACAZE qui furent mes collègues de formation qui m'ont suivi et soutenu tout au long de la formation.

Monsieur Bruno **COLLIN**, mon formateur AFPA, pour son enseignement et son investissement.

Monsieur **BEON** Frédéric, mon maître de stage qui m'a fait confiance dès mon arrivée.

Monsieur **JOLY** Sébastien & Madame **SANSONNETTE** Géraldine ainsi que tous les collègues avec qui j'ai pu échanger pendant ma Période de d'Application en Entreprise.

2. INTRODUCTION

Etant passé par la comptabilité, la grande distribution et la couverture-zinguerie, j'ai découvert le métier de Développeur par le biais de mon entourage.

Après m'être renseigné sur les métiers de l'informatique et les formations qui existaient dans ce domaine, j'ai trouvé cette formation de **Concepteur Développeur Informatique** qui avait tout à fait l'air de correspondre à mes capacités et à ma personnalité.
Je me suis donc inscrit à Balma dans cette formation pour pouvoir décrocher le titre professionnel de niveau II de CDI.
Pour valider cette formation, j'ai dû mettre en pratique les connaissances et compétences apprises tout au long de la formation au cours d'une Période d'Application en Entreprise.

J'ai choisi de rejoindre le Conseil Départemental du Gers pour la période du 11 janvier 2016 au 01 Avril 2016, au sein du service Logiciel Métier pour reprendre le développement d'une application servant à la saisie des frais de déplacements des élus du Département ainsi qu'au traitement de ces frais par le service des Assemblées.

Ce projet regroupe les trois compétences générales qui couvrent cette formation, à savoir :

- Développer des composants d'interface.
- Développer la persistance des données.
- Développer une application N-TIERS (à plusieurs couches).

Ce rapport fait la synthèse du travail effectué pendant cette dite période.
Je vous présenterai donc la structure du Conseil Départemental du Gers, et surtout le service Logiciel Métier dans lequel j'ai travaillé.

Je décrirai ensuite le projet et son contexte, les objectifs de développement.

Je dévoilerai la conception du projet ainsi que les deux métiers que j'ai eu la chance de pouvoir exercer pendant cette période : celui de chef de projet ainsi que celui de développeur.

Enfin je ferai le bilan de cette Période d'Application en Entreprise en exposant les difficultés rencontrées, l'état d'avancement du projet à la fin de cette période ainsi que mes perspectives d'avenir.

3. ABSTRACT

To validate my professional title, I had to do a three-month internship to practice the three parts of my training, namely:

- Interface component development
- Data persistence development
- N-tier application architecture development

The French Department of the Gers started a paperless process.
By contrast, the elected representatives are still using paper to capture their expenses.
Therefore a software application for the elected representatives' travel expense refunds was needed, as well as to developing the processing of these expenses by the Assembly service.
This application named AppEgers was started by a previous trainee.
It was developed in JEE with JAVA language, using STRUTS2 Framework, JSP technology and MySQL for the database.
It was extremely time-consuming for me to dive in an architecture developed by another person.

Fortunately, I could take on the project manager role, in addition to developer role, and thus I planned my own work, I set up SCRUM Agile method process.
I had to report to my chief to be given an additional human resource, to transfer the knowledge to finish the development and maintain the application.

This experience was very meaningful; it got me work experience, and development skills.
My future prospects are subcontracting agreements to develop websites for a press agency.

4. PRÉSENTATION DE L'ENTREPRISE

4.1. Le Département

Le département du Gers situé dans la région LANGUEDOC ROUSSILLON MIDI PYRENEES.

CHIFFRES CLÉS :

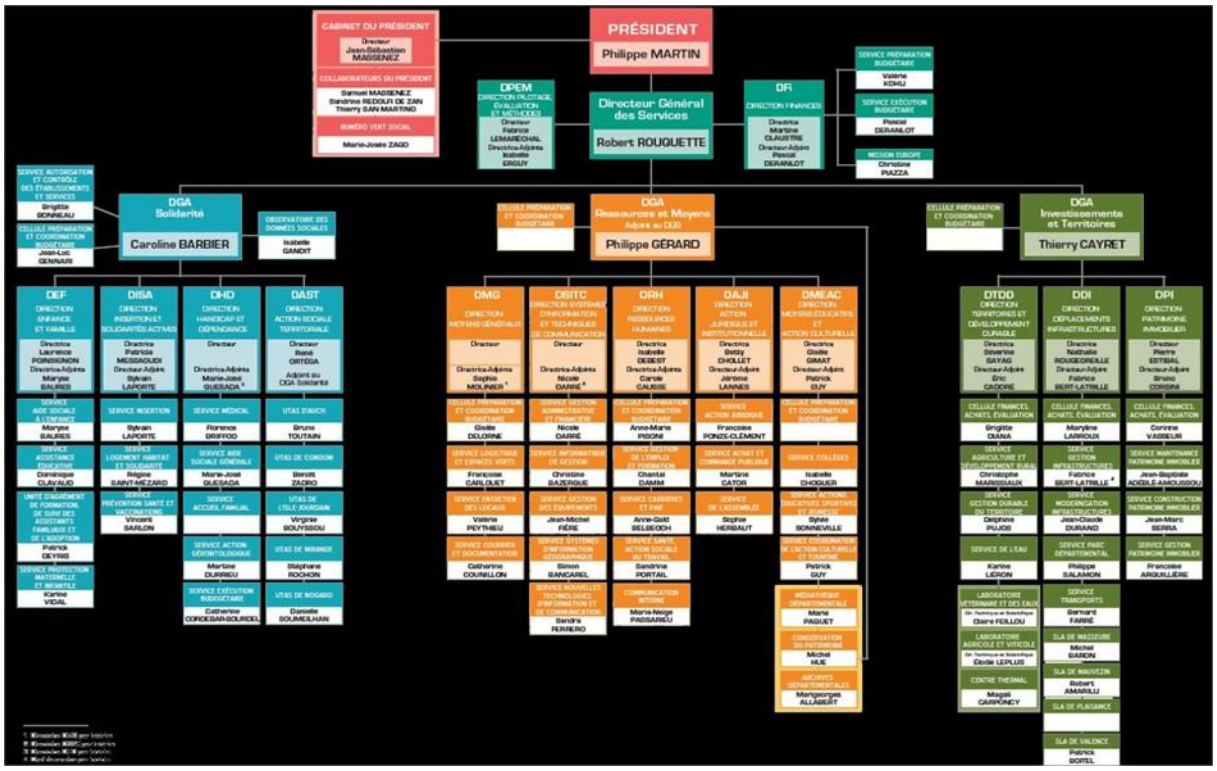
- ✓ Population : 188200 habitants
- ✓ 35% de la population active sont des retraités.
- ✓ Taux de chômage : 8,5%
- ✓ Préfecture : Auch
- ✓ Nombre de communes : 462
- ✓ Nombre de cantons : 31

4.2. Le conseil départemental

Le conseil départemental du Gers est l'assemblée délibérante du département français du Gers, collectivité territoriale décentralisée.
Son siège se situe à Auch.
Il est présidé par Philippe Martin, ancien ministre de l'Ecologie, du Développement Durable et de l'Energie.

Le conseil départemental est régi par une assemblée élue au suffrage universel direct.
Les conseillers généraux sont élus à raison d'un conseiller par canton.
Le conseil départemental élie une commission permanente qui représente l'organe exécutif du département.
Lors des assemblées, les élus du département prennent les décisions en rapport à leurs domaines de compétence.

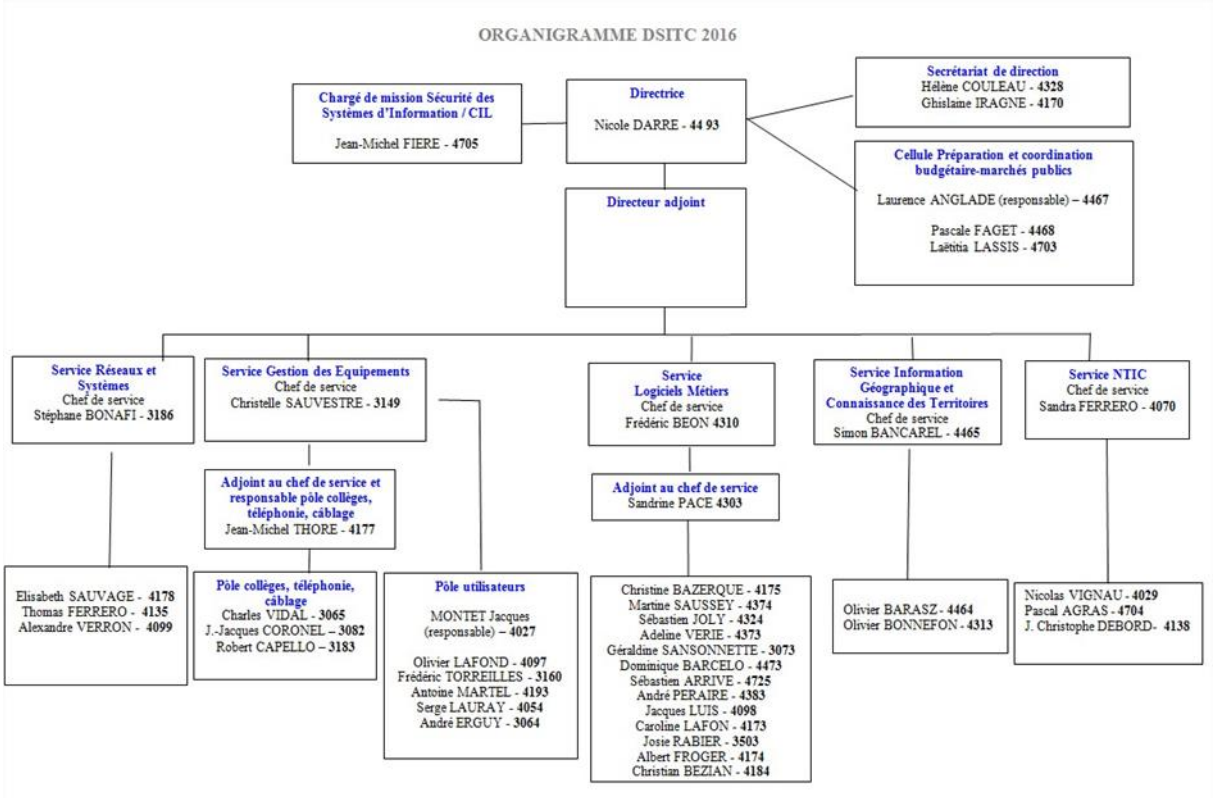
ORGANIGRAMME :



Les domaines de compétences du conseil départemental sont très diversifiés :

- La solidarité
- Le social
- L'éducation
- Les transports
- L'aménagement du territoire
- L'agriculture
- L'environnement
- La culture
- Le patrimoine
- Le sport
- Le tourisme

4.3. La DSITC et ses missions



Plusieurs services œuvrent à la Direction des Systèmes d'Information et Techniques de Communication (DSITC) pour offrir aux usagers du Conseil Départemental un service de qualité :

- Le service Réseau et Systèmes est chargé de la sécurité et de la pérennité des réseaux et systèmes.
- Le service Gestion des Equipements assure la gestion du parc informatique et téléphonique.
- Le service Logiciel Métier, dont j'ai fait partie pendant ma Période d'Application en Entreprise, assure la mise en place des logiciels de gestion des différentes applications métiers à la demande et en collaboration avec les différentes directions.

- Le service Information Géographique et Connaissance des Territoires réalise des travaux de cartographie, des Atlas et la numérisation de documents d'urbanisme.

- Le service Nouvelles Technologies d'Information et de Communication (NTIC) assure des réalisations graphiques (affiches, plaquettes, plateformes Web, sites internet & intranet, …).
  Il réalise aussi les prise de vue photo pour la collectivité et travaille à la création d'une photothèque.

5. PRÉSENTATION DU PROJET

5.1. Description :

Le projet dont le chef de service Logiciel Métier m'a chargé est un projet qui a été entamé par un de mes prédécesseurs.

Il s'agit de développer une application dont le but est la saisie des frais de déplacements des élus du Département ainsi que le traitement de ces frais par le service des Assemblées.

5.2. Contexte & contraintes

Les élus disposent chacun d'une tablette de type iPad.

Dans un mouvement de dématérialisation et de digitalisation, la DSITC a décidé de développer des applications à destination des élus.

Il s'agit dont de développé une application de type Client Léger pour la saisie des frais de déplacements des élus, accessible sur tablette coté Front et pour le traitement de ces frais par le servie de l'Assemblée coté Back, accessible sur PC.

L'application étant déjà commencé à mon arrivée, j'ai été contraint par l'environnement, l'application (AppE-Gers) est développée en langage JAVA basée sur la spécification JEE avec le Framework STRUTS2 sur l'IDE Eclipse.

5.3. Objectifs

L'application doit être facilement évolutive et maintenable.

Elle doit être consultable depuis l'extérieur, sur tablette comme sur PC, destinée aux élus pour la saisie de leurs frais de déplacements.

Le service de l'Assemblée doit pouvoir gérer les utilisateurs, leurs véhicules ainsi que la partie gestion des frais de déplacement se basant sur les règles d'éligibilité des déplacements et du calcul des remboursements qu'elles engendrent.

6. CAHIER DES CHARGES

L'application doit permettre la saisie des frais de déplacements des élus.

Effectivement, côté FRONT OFFICE l'utilisateur doit pouvoir créer un état mensuel (par mois et par véhicule) dans lequel il pourra saisir ces frais de déplacement concernant le mois de l'état mensuel.

Sur son profil, il ne doit pouvoir modifier que son mot de passe.

Côté BACK OFFICE, le service des Assemblées doit pouvoir créer, modifier, supprimer de nouveaux utilisateurs et de nouveaux véhicules pour pouvoir attribuer à chaque utilisateurs leur véhicule. C'est la puissance fiscale des véhicules qui est à l'origine du calcul du remboursement des frais.

Il doit aussi pouvoir consulter les différents états mensuels, valider ou rejeter les déplacements de chaque état avant de valider les états mensuels de frais pour qu'ils puissent être envoyé au Mandatement (Paiement).

L'utilisateur doit pouvoir :

- Se connecter à l'application.
- Créer un état mensuel du mois en cours à M-6.
- Pouvoir choisir son véhicule parmi une liste s'il en a plusieurs.
- Créer autant de déplacements que nécessaires avec
- Pour chaque déplacement : - saisir de l'objet du déplacement, qui en fonction des domaines de compétences de l'élu.
  - Saisir la date du déplacement.

- Saisir le lieu d'arrivée du déplacement (le lieu de départ étant toujours le lieu de résidence administrative de l'élu.

- Envoyer l'état mensuel au service des Assemblées pour traitement.

Le service des Assemblées est composé de trois personnes qui gèrent le traitement des frais de déplacement.

La première personne (Mme BAYLAC : Profil1) s'occupe de l'éligibilité de chaque déplacement.

La seconde (Mme HERBAULT : Profil2) s'occupe de vérifier le travail de Mme BAYLAC et de valider l'état mensuel.

La troisième personne (Mme CARRIE : Profil3) assure le mandatement des états mensuels validés par la seconde personne.

Le profil1 doit pouvoir :

- Se connecter à l'application.
- Créer, consulter, modifier ou supprimer des utilisateurs.
- Créer, consulter, modifier ou supprimer des véhicules.
- Attribuer des véhicules à des utilisateurs.
- Consulter tous les états mensuels envoyés par les élus.
- Valider ou rejeter les déplacements de chaque état mensuel
- Envoyer les états mensuels à profil2 pour validation.
- Renvoyer un état mensuel à l'élu concerné en cas de trop nombreuses erreurs.

Le profil2 doit pouvoir :

- Avoir les mêmes droits que le profil1.
- Valider un état mensuel qui sera envoyé à profil3.
- Rejeter un état mensuel qui sera renvoyé à profil1.

Le profil3 doit pouvoir :

- Se connecter à l'application.
- Consulter les états mensuels.
- Editer & Mandater les états mensuels.

7. CONCEPTION

7.1. Outils utilisés

- Planification : **GANTT Project**

GANTT Project est un logiciel libre de gestion de projet. Il permet la planification d'un projet à travers un diagramme de GANTT.

- Modélisation UML : **DIA**

DIA est un logiciel open source de création de diagramme d'activité, de classes, d'état, de transition, et de USE CASES.

- Modélisation Base de Données : **JMERISE**

JMERISE est un logiciel open source dédié à la modélisation des modèles conceptuels de données pour MERISE.
Il génère aussi le Modèle Logique de Données ainsi que le script SQL de création de la base de données.

- Environnement de développement (IDE) : **ECLIPSE MARS**

ECLIPSE MARS est une IDE extensible, universelle et polyvalente s'appuyant principalement sur JAVA.

- SGBDR (Système de Gestion de Base de Données Relationnelles : **MySQL**

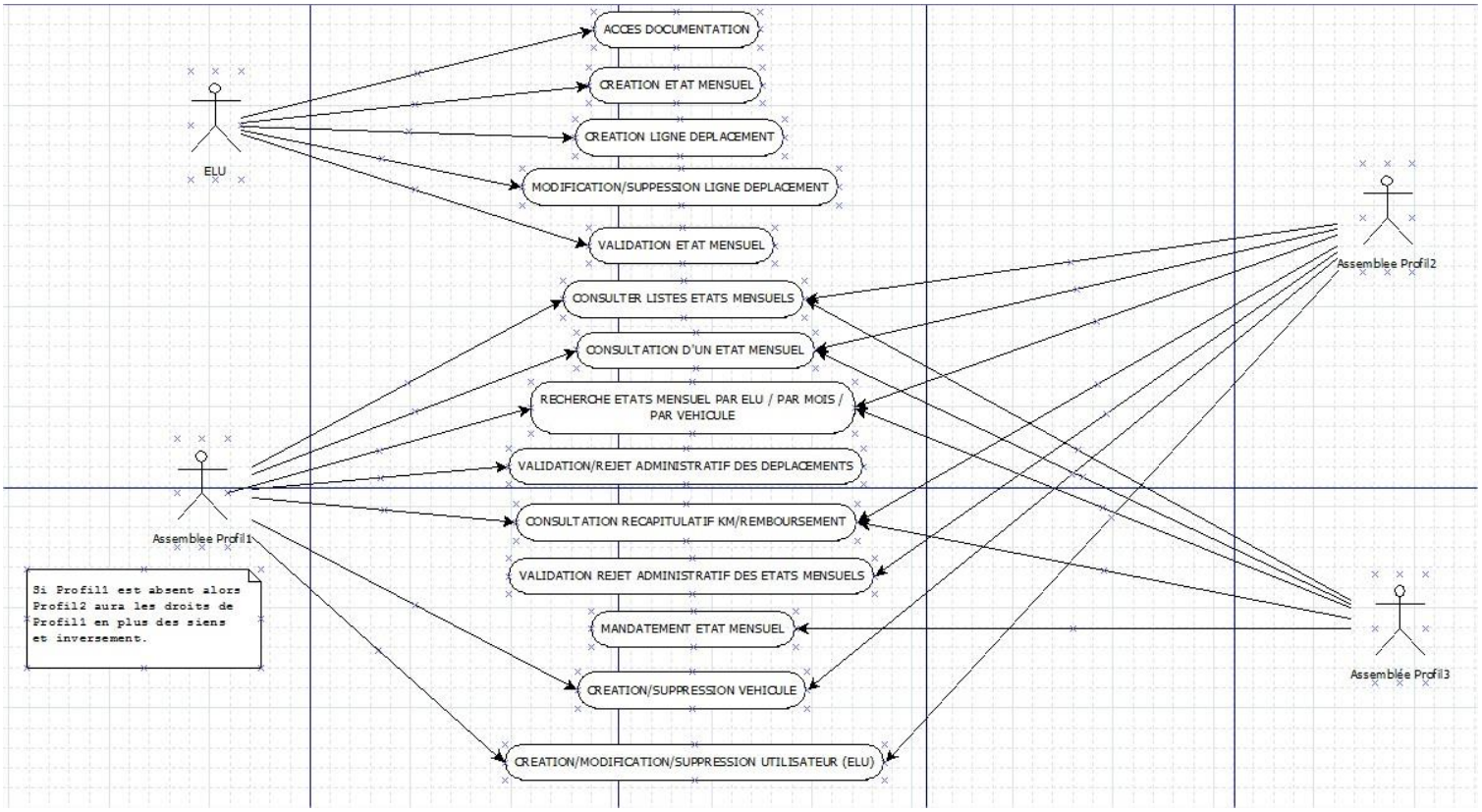
MySQL est un serveur de base de données Relationnelles SQL développé dans un souci de performances élevées en lecture.

C'est un logiciel libre, open source.

- Logiciel de travail collaboratif & gestion des versions : **SUBVERSION**

Subversion est un logiciel de gestion de versions fonctionnant sur le mode Client-Serveur avec un serveur informatique centralisé et unique où se situe les fichiers constituant la référence (repository) et des postes clients sur lesquels se trouvent les fichiers recopiés depuis le serveur (éventuellement modifiés localement depuis), et un logiciel client (TORTOISE SVN) permettant la synchronisation entre chaque client et le serveur de référence.

7.2. Cas d'utilisation



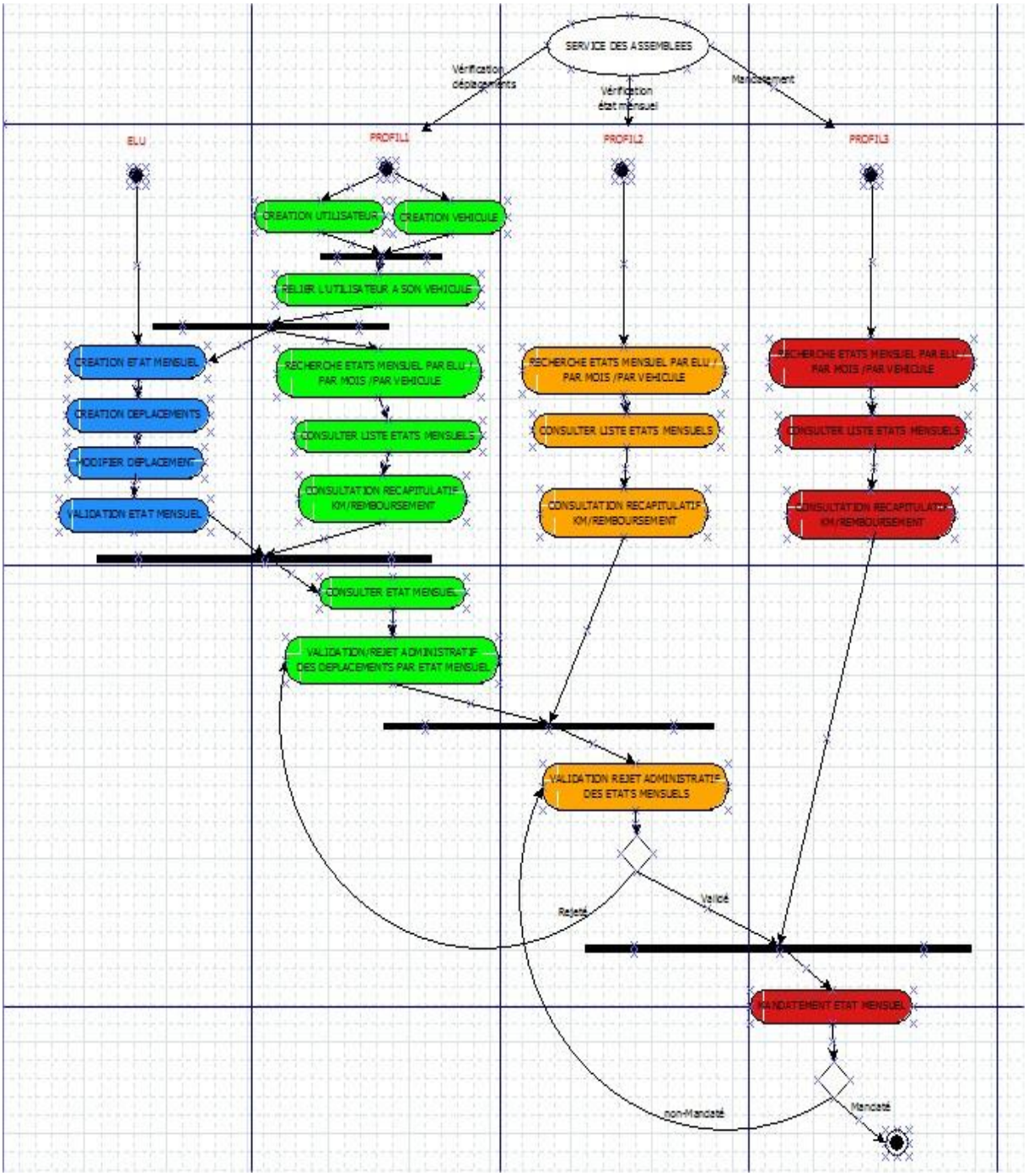
Comme expliqué dans le cahiers des charges, l'élu doit pouvoir créer des états mensuels, créer des lignes de déplacements dans les états mensuels, modifier ou supprimer des lignes de déplacements et valider les états mensuels pour traitement par le service des Assemblées.

Le profil1 et le profil2 doivent pouvoir se remplacer mutuellement sauf pour la validation des états mensuels qui reste un droit réservé au profil2.

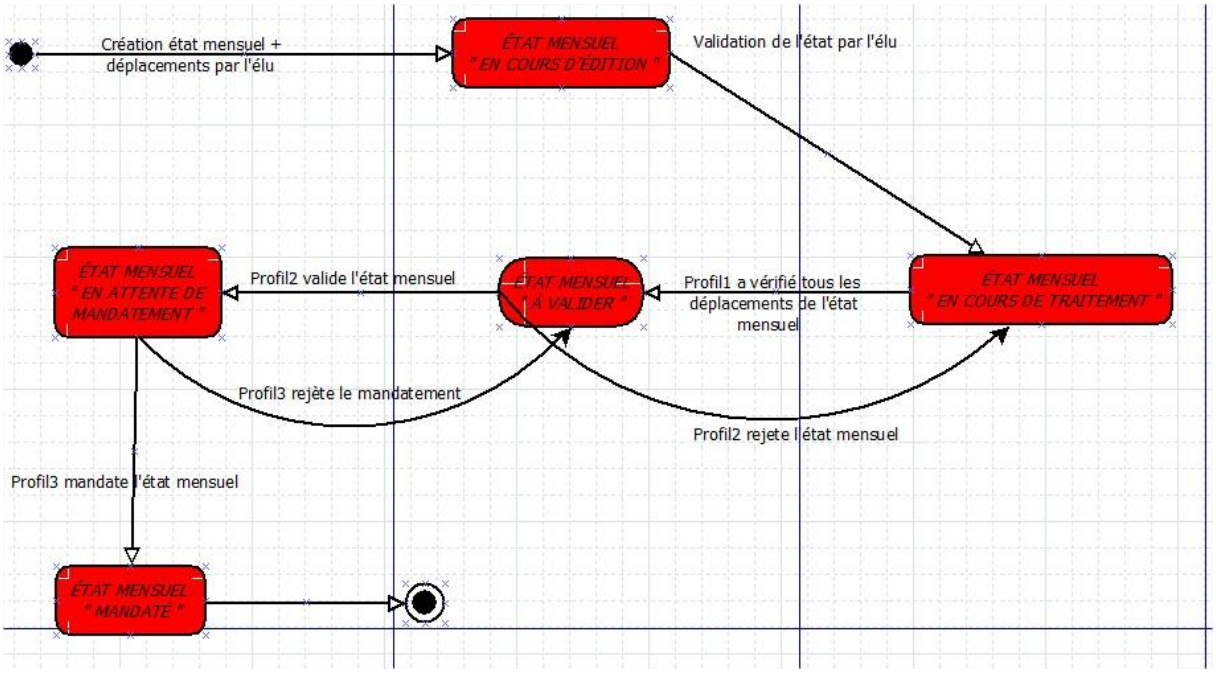
Le profil3 doit pouvoir consulter les états mensuels envoyés par les élus et mandater les états validés.

7.3. Diagrammes UML

- Diagramme d'Activité :



- Diagramme d'état-transitions :



- ♦ A la création d'un état mensuel, le statut de d'état est « en cours d'édition par l'élu ».
- ♦ Lorsque l'élu valide l'état mensuel pour envoi au service des Assemblées, le statut devient « état mensuel en cours de traitement par l'Assemblée ».
- ♦ Quand le profil1 a traité chaque déplacement et qu'elle enregistre le traitement des déplacements, le statut de l'état devient « état à valider par l'Assemblée ».
- ♦ Lorsque le profil2 a vérifié le traitement du profil1 et qu'il a validé l'état mensuel, celui-ci prend le statut : « état en attente de Mandatement ».
- ♦ Quand le profil3 a mandaté l'état mensuel, le statut de l'état devient « mandaté ».

7.4. Modèle de données

Pour décrire de façon formelle les données qui font partie du Système d'Information qui concerne l'application, on a pour habitude d'utiliser le **Modèle Conceptuel des Données**.

Il s'agit d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'**entités** (objets ou chose concrète), d'**attributs** (caractéristiques des entités) dont un ou un groupe d'attribut seront l'**identifiant** de l'entité (doit avoir une valeur unique pour chaque entité), de **relations** ou **associations** (relation ou lien sémantique entre les entités) et de **cardinalités** (nombre de fois minimum et maximum qu'une entité peut intervenir dans une relation).

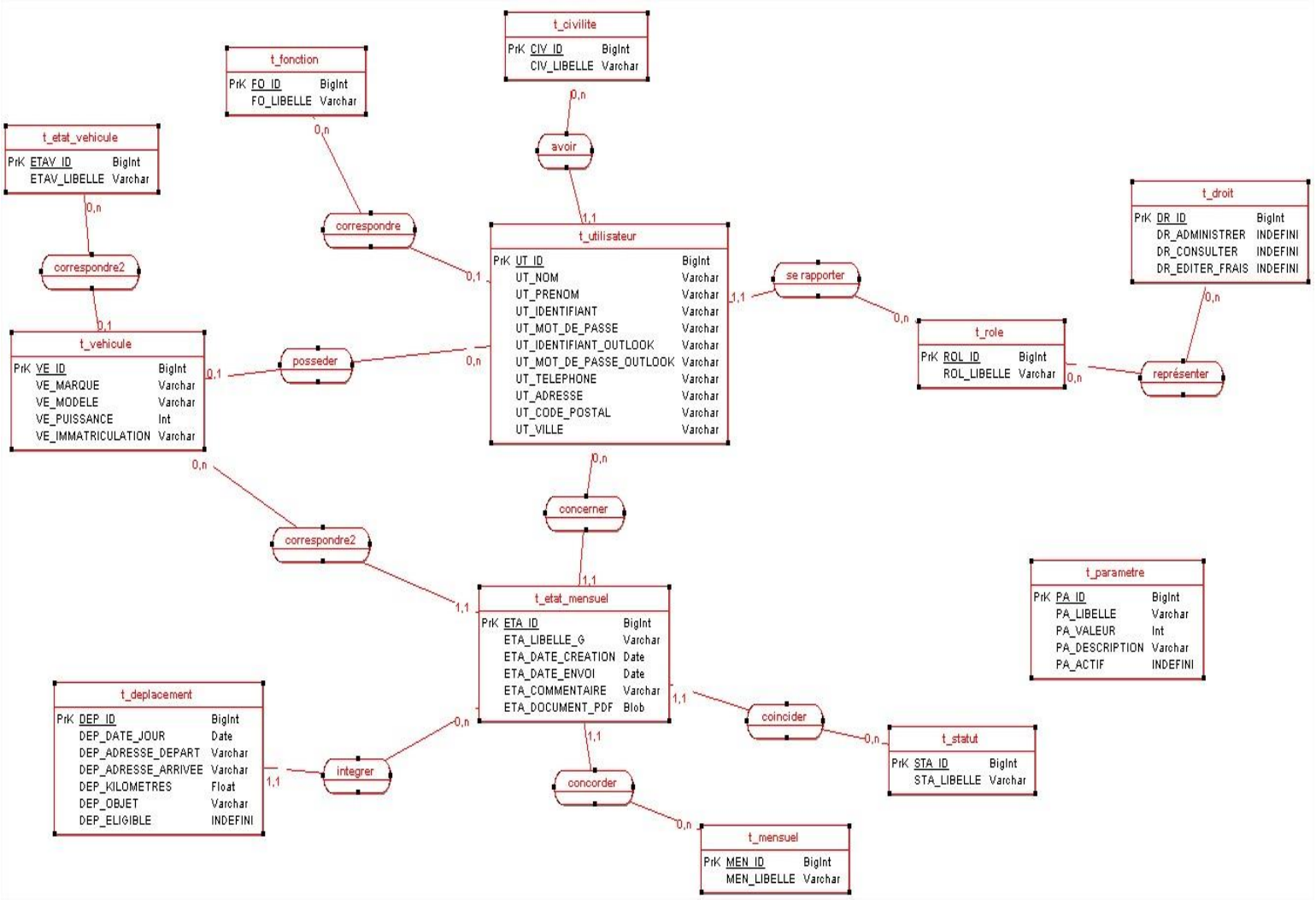
Voici la liste des entités du Système d'Information ainsi que leurs attributs :

- **UTILISATEUR :**
  - Son nom
  - Son prénom
  - Son identifiant de login.
  - Son mot de passe de login
  - Son adresse mail outlook
  - Son mot de passe outlook
  - Son numéro de téléphone
  - Son adresse
  - Son code postal
  - Sa ville de résidence

- **CIVILITE :**
  - Son libellé
- **FONCTION :**
  - Son libellé
- **VEHICULE :**
  - Sa marque
  - Son modèle
  - Sa puissance
  - Son immatriculation
- **ETAT_VEHICULE :**
  - Son libellé
- **ETAT_MENSUEL :**
  - Son libellé
  - Sa date de création
  - Sa date d'envoi au service des Assemblée
  - Son commentaire
  - Son document PDF
- **DEPLACEMENT :**
  - Sa date
  - Son adresse de départ
  - Son adresse d'arrivée
  - Sa distance en kilomètres
  - Son objet
  - Son éligibilité
- **STATUT :**
  - Son libellé
- **MENSUEL :**
  - Son libellé
- **ROLE :**
  - Son libellé

- **DROIT :**
  - Administrer
  - Consulter
  - Editer frais
- **PARAMETRE :**
  - Son libellé
  - Sa valeur
  - Sa description
  - Actif ?

Modèle Conceptuel des Données :

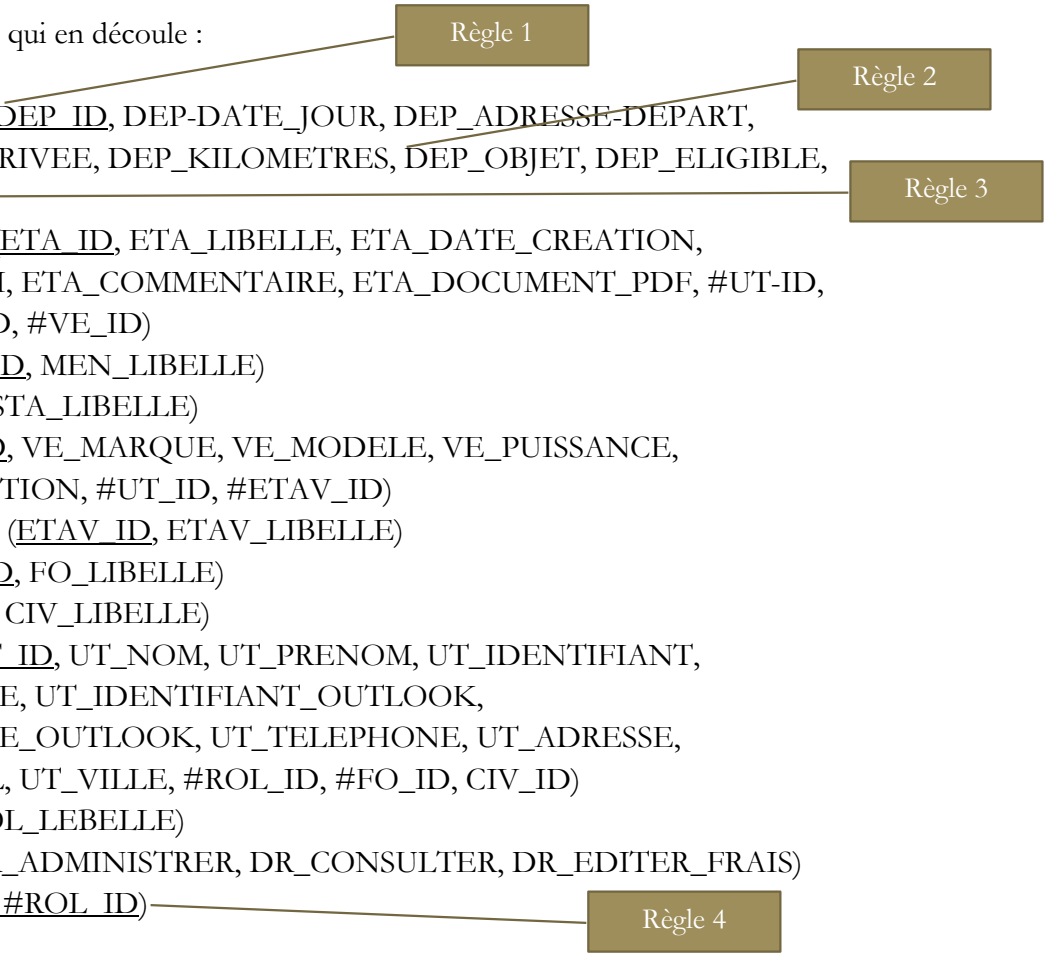


Passage du MCD au Modèle Relationnel :

Le **modèle relationnel** est une manière de modéliser les relations existantes entre les données.
Il existe des règles de passage du MCD au Modèle Relationnel :

1. Toutes les entités deviennent des relations (ou tables) ayant pour clé primaire leur identifiant.
2. Les propriétés d'une entité deviennent les attributs de la table.
3. Toute association hiérarchique (de type [1 : n]) se traduit par une clé étrangère.
4. Toute association non hiérarchique (de type [n : n]) devient une relation.

Le modèle Relationnel qui en découle :



DEPLACEMENT (DEP_ID, DEP-DATE_JOUR, DEP_ADRESSE-DEPART, DEP_ADRESSE_ARRIVEE, DEP_KILOMETRES, DEP_OBJET, DEP_ELIGIBLE, #ETA_ID)
ETAT-MENSUEL (ETA_ID, ETA_LIBELLE, ETA_DATE_CREATION, ETA_DATE_ENVOI, ETA_COMMENTAIRE, ETA_DOCUMENT_PDF, #UT-ID, #STA_ID, #MEN_ID, #VE_ID)
MENSUEL (MEN_ID, MEN_LIBELLE)
STATUT (STA_ID, STA_LIBELLE)
VEHICULE (VE_ID, VE_MARQUE, VE_MODELE, VE_PUISSANCE, VE_IMMATRICULATION, #UT_ID, #ETAV_ID)
ETAT_VEHICULE (ETAV_ID, ETAV_LIBELLE)
FONCTION (FO_ID, FO_LIBELLE)
CIVILITE (CIV_ID, CIV_LIBELLE)
UTILISATEUR (UT_ID, UT_NOM, UT_PRENOM, UT_IDENTIFIANT, UT_MOT_DE_PASSE, UT_IDENTIFIANT_OUTLOOK, UT_MOT_DE_PASSE_OUTLOOK, UT_TELEPHONE, UT_ADRESSE, UT_CODE_POSTAL, UT_VILLE, #ROL_ID, #FO_ID, CIV_ID)
ROLE (ROL_ID, ROL_LEBELLE)
DROIT (DR_ID, DR_ADMINISTRER, DR_CONSULTER, DR_EDITER_FRAIS)
ROL_DR (#DR_ID, #ROL_ID)

8. MANAGEMENT DU PROJET

8.1. Planification

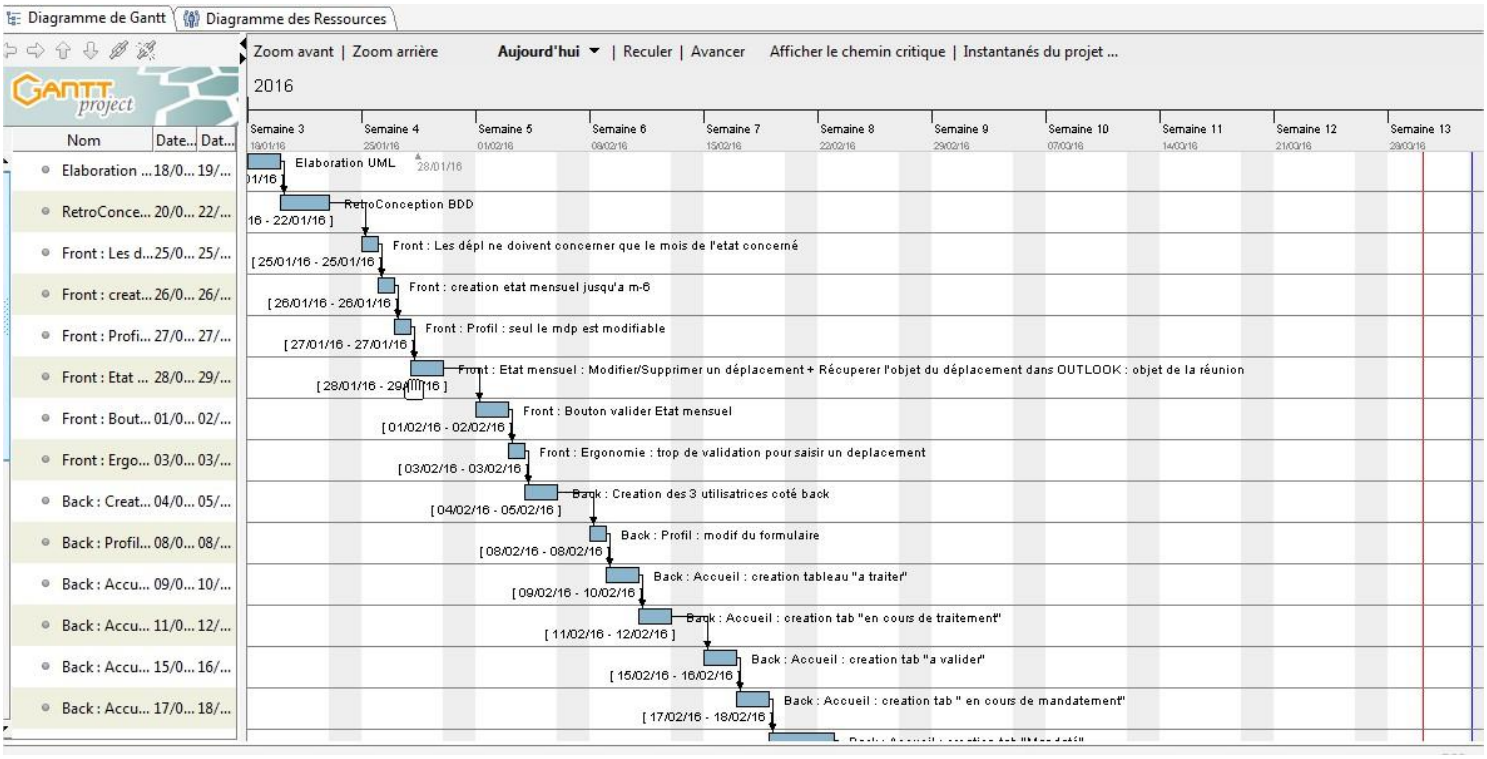
J'ai commencé ma Période d'Application en Entreprise par faire une planification de mon projet.

J'ai d'abord découpé et déterminer les tâches à effectuer et à les ordonnancer.

Ensuite il a fallu estimer leurs charges et le temps pessimiste nécessaire pour les développer.
J'ai donc réalisé le tableau suivant :

	Etapes	Tâches à effectuer	Durée en 1/2 journée	
			réel	pessimiste
CONCEPTION	A	Elaboration UML	3	4
	B	Retro-Conception BDD	3	4
FRONT	C	Création bouton Valider Etat Mensuel + Action Struts	1,5	2
	D	L'état mensuel ne doit comporté que des déplacements de	1,5	2
	E	l'élu ne doit pouvoir modifier que son mdp	1	2
	F	vehicule : liste + creation nouveau	2	3
	G	Suppression d'un deplacement	3	4
	H	Trop de fenetres de validation, a supprimer	2	3
		SOUS-TOTAL	**17**	**24**
		SOUS-TOTAL(en jour)	**8,5**	**12**
BACK	I	Création 3 profils back office + droits	4	6
	J	Profil: modification du formulaire	2	3
	K	Accueil : liste etat à traiter	3	4
		Accueil : liste etats en cours de taitement	3	4
		Accueil : liste etats à valider	2	3
		Accueil : liste etats avant Mandatement	2	3
		Accueil : liste etats à Mandatés	2	3
	L	Profil 1 : A traiter : choix et affichage de létat	3	4
	M	A traiter : valider/rejeter déplacement	3	4
	N	Frais : recherche par elu/ par mois	4	5
	O	Bouton validation du traitement des déplacements	2	3
	P	Calcul du remboursement	2	3
	Q	rajout date d'envoi de l'etat mensuel + calcul cumul km	2	3

Grâce à ce tableau, le diagramme de GANTT apparait ainsi :



8.2. Reporting

Après avoir réalisé mon diagramme de GANTT, je me suis vite rendu compte que je ne pourrai pas finir de développer dans le temps imparti.
J'ai donc alerté mon chef de service et demandé une ressource supplémentaire pour avancer au maximum et pour faciliter la maintenance de la future application.

J'ai obtenu ½ ressource (une personne quelques heures par semaine) qui avait des connaissance en langage objet (C) mais ne connaissait pas le langage JAVA, ni le Framework STRUTS2, ni JavaScript, ni JQUERY, ni Ajax.
J'ai donc dû la former à ces technologies petit à petit, j'ai dû la laisser développer des parties simples de l'application mais qui prenaient en compte toutes les facettes de l'application (DAO, classe Action, JSP, appel Ajax…) tout en vérifiant son travail.

8.3. Mise en place de la méthode Agile SCRUM

Comme indiqué précédemment, l'application avait déjà été commencée par un ancien stagiaire de l'AFPA.
Malheureusement, il n'a pas assez communiqué et n'a présenté le résultat de son travail qu'à la fin de sa Période d'Application en Entreprise.

L'application ne correspondait pas aux besoins réels des utilisateurs.

Pour éviter de reproduire ses erreurs, j'ai décidé de mettre en place la méthode Agile SCRUM. En effet, j'ai planifié des sprints de deux semaines et des revues de sprint avec le PRODUCT OWNER de la partie back qui n'avait pas été commencé.

Le but recherché était de mettre le client de l'application au cœur du projet pour que les avancements soient basés sur le concret et pour pouvoir rester flexible aux changements.

Je me suis donc rendu compte que la communication était la clé d'un projet réussi et que cette méthode Agile faisait écho à la simplicité, l'efficacité et à la qualité de la gestion d'un projet informatique.

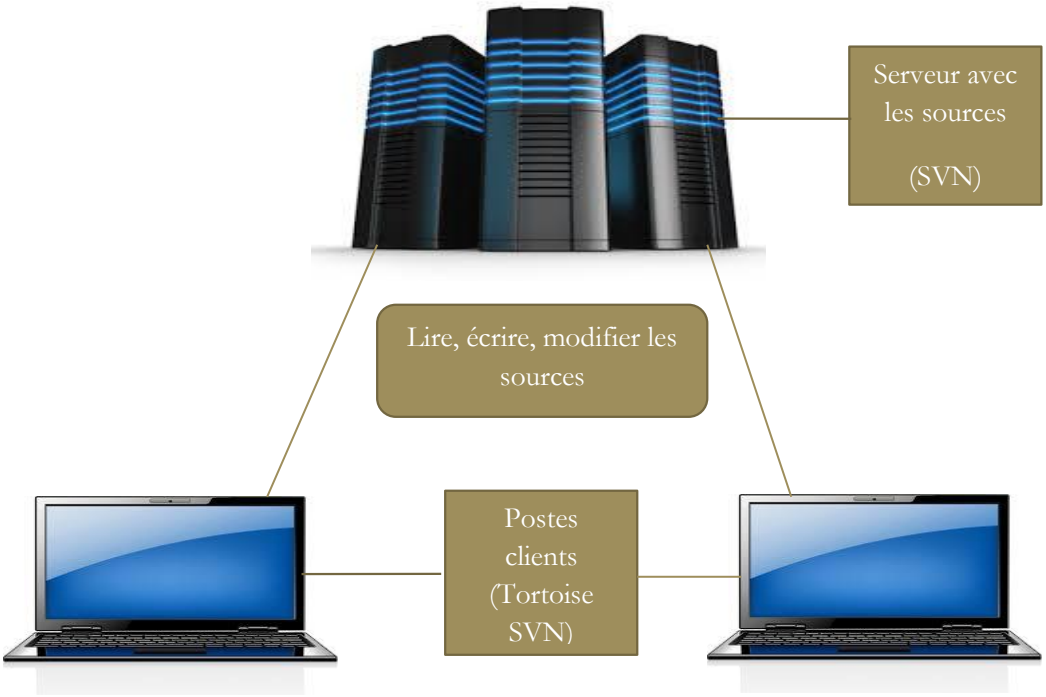
8.4. Travail collaboratif & gestion du versioning

J'ai choisi d'utiliser et d'installer SUBVERSION (SVN) pour gérer le développement collaboratif ainsi que la gestion du versioning de l'application.

J'ai d'abord installé SUBVERSION sur un serveur virtualisé du Conseil Départemental qui est hébergé par un de ses sous-traitants (DRI).

Ensuite il a fallu installer sur chaque poste, le logiciel client (TORTOISE SVN) permettant la synchronisation entre chaque client et le serveur de référence.

Voici un schéma simple qui permet de mieux visualisé la structure de SVN :

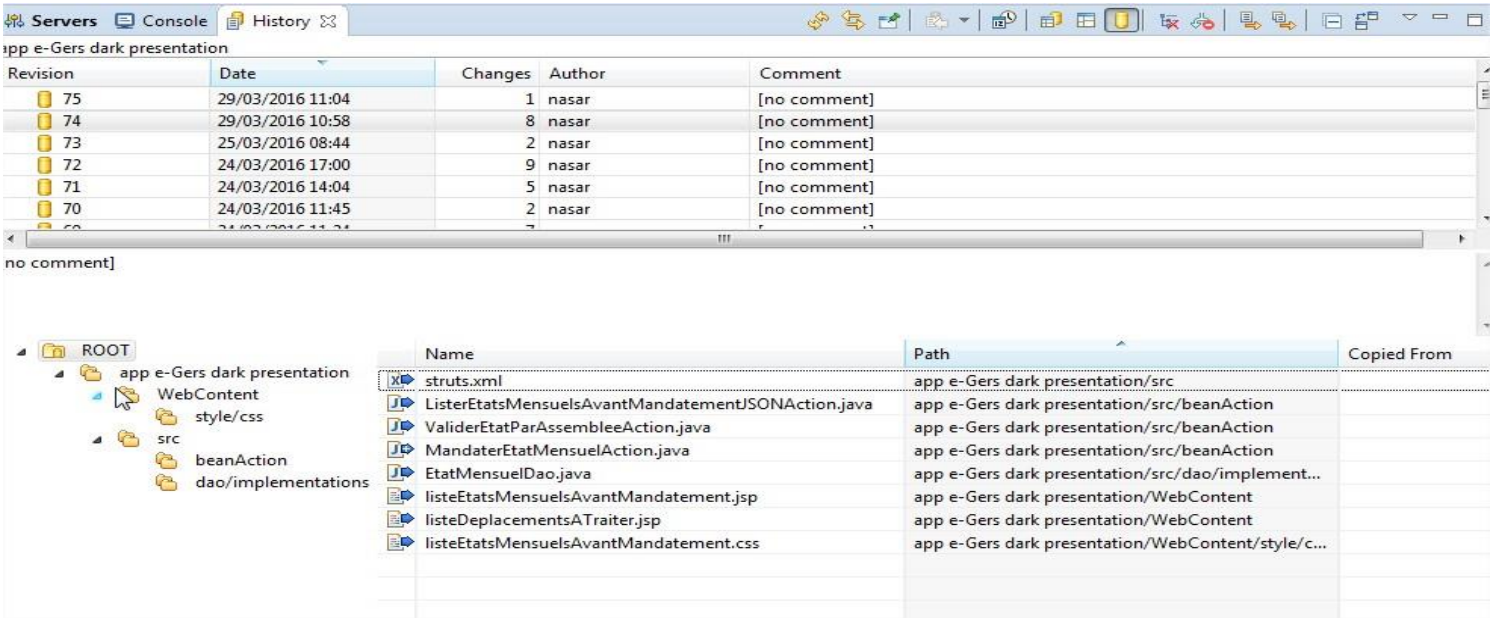


- Pour rapatrier toutes les sources du serveur sur le poste client il faut utiliser la commande SVN CheckOut.

- Pour mettre à jours les sources déjà présentes sur le poste client, on utilisera la commande SVN Update.

- Pour exporter sur le serveur, des modifications que l'on a apporté sur le poste client, il faudra utiliser la commande SVN Commit, à chaque Commit, SVN créera une nouvelle version de la source et conservera les précédentes.
  On pourra donc récupérer n'importe quelle version de la source.

Exemple : Historique et détail de la version 74 :

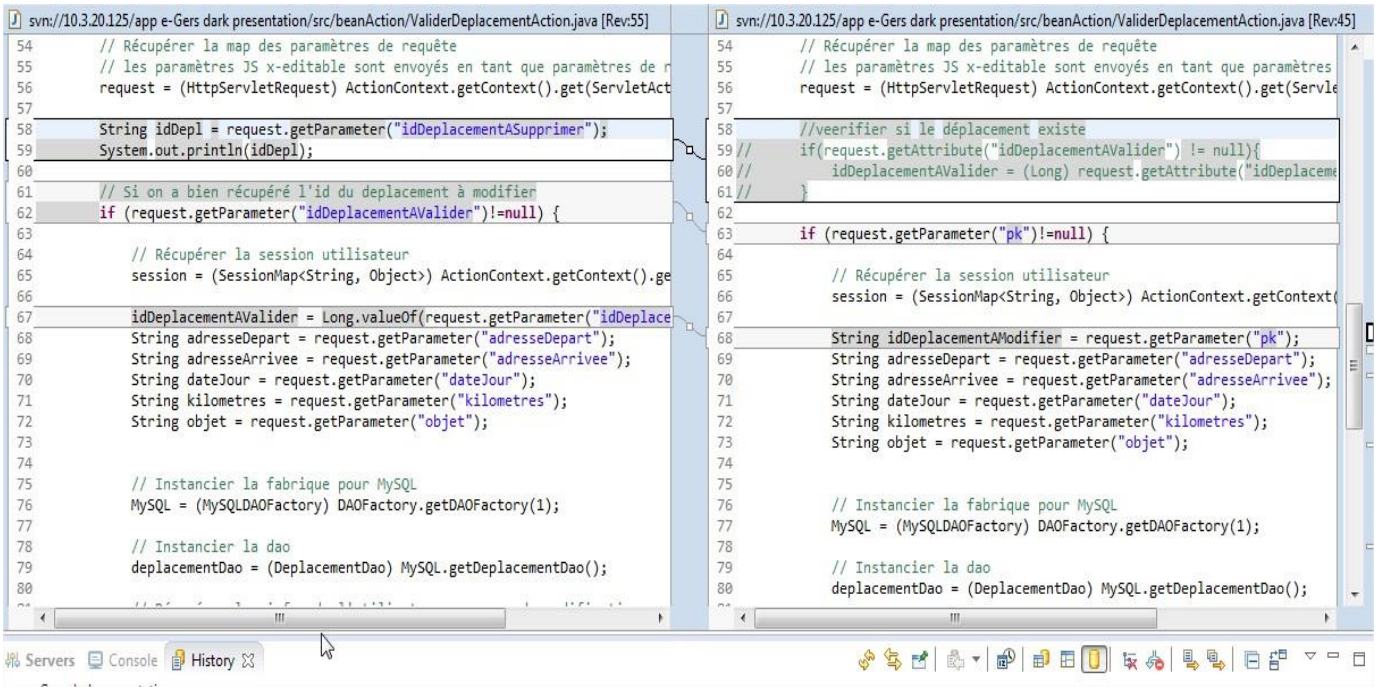


- L'architecture de la gestion des sources sur le serveur est par défaut un dossier appelé TRONC.
  A chaque fois qu'un poste client fait un nouveau Commit, une version supplémentaire du Tronc est créée.

- Il est aussi possible de créer des Branches pour qu'un développeur puisse développer dans son coin et que ses modifications ne soient pas prises en compte dans le Tronc.

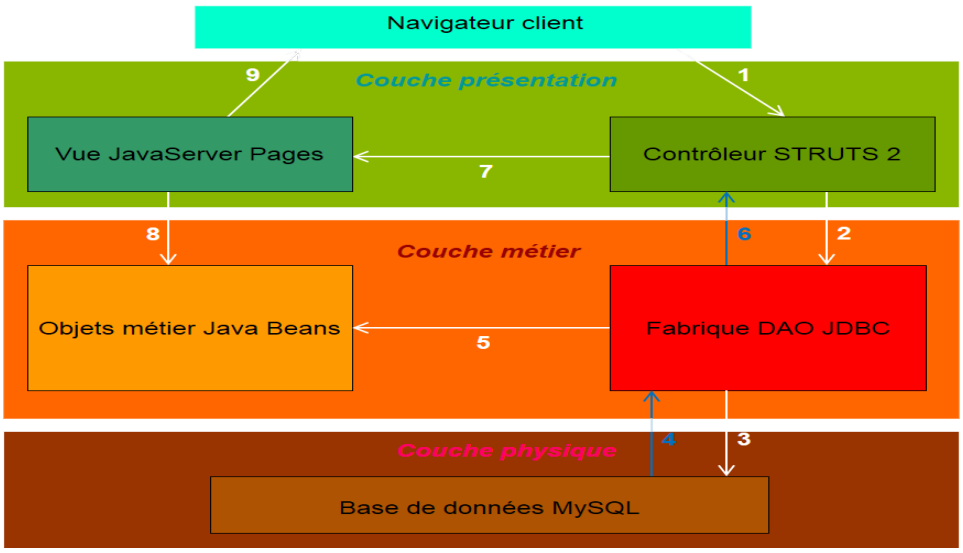
Une équipe peut par exemple travailler sur le tronc pour corriger des bugs pendant qu'une autre équipe puisse travailler sur une nouvelle fonctionnalité sur une branche.

Correction des conflits possibles entre les différentes versions :

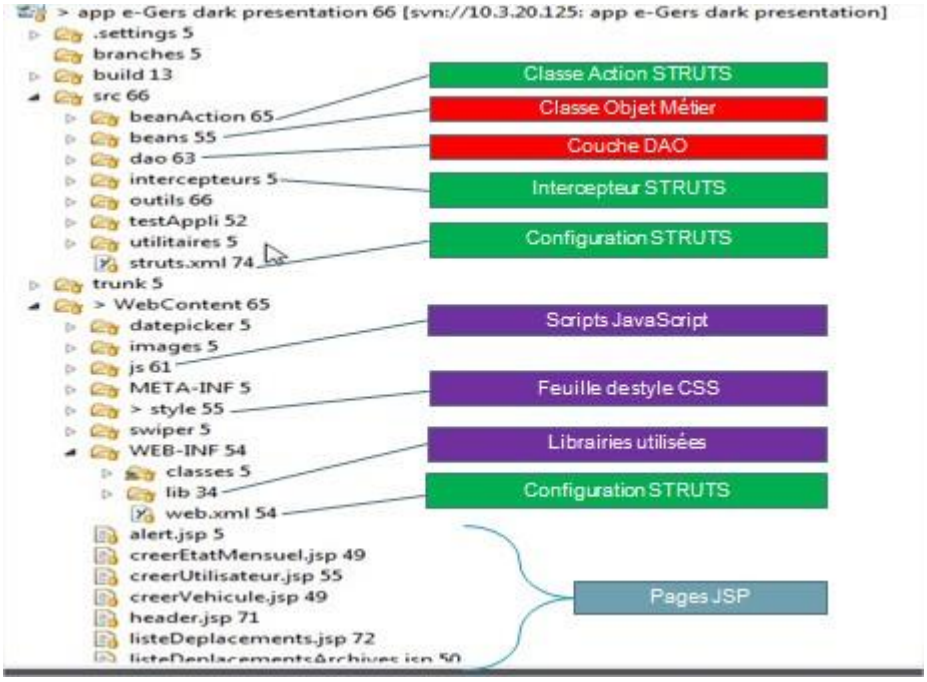


9. RÉALISATION DU PROJET

9.1. Architecture 3 Tiers



Structure de l'application :



Modèle

Vue

Contrôleur

9.2. ENVIRONNEMENT DE DÉVELOPPEMENT

Serveur d'application :

Le conteneur Web **APACHE TOMCAT** est un **conteneur de servlets et de JSP** qui permet d'exécuter des applications web.

Il est également un serveur http (serveur WEB).

TOMCAT est écrit en langage JAVA, il peut donc s'exécuter via la machine virtuelle Java sur n'importe quel système d'exploitation la supportant.

Ce serveur était déjà en place à mon arrivée et l'analyse des besoins a montré que l'application n'aurait pas besoin d'inclure des EJB donc ce choix de serveur d'application reste judicieux.

Environnement de développement :

Le package WAMP (Windows Apache MySQL PHP) est une plateforme de développement sous Windows, très utile pour développer un site ou application web dynamique.
Il est composé d'Apache (serveur http), de MySQL (SGBD) et PHP (plugin Apache qui permet de traiter des pages web dynamiques en PHP).
Le choix de cette plateforme est approprié car elle possède également PHPMyAdmin pour gérer plus facilement les bases de données et elle est régulièrement mis à jour et disponible en français.

9.3. LE PARADIGME MVC

Le **Modèle-Vue-Contrôleur** est une architecture logicielle.
MVC impose la séparation entre les données, les traitements et la présentation.
C'est pour cette raison que l'application est divisée en trois parties fondamentales : le modèle, la vue et le contrôleur.

9.3.1. MODÈLE ORM : MAPPAGE DES DONNÉES

Le modèle représente le comportement de l'application : traitements des données, interactions avec la base de données…
Il décrit ou contient les données manipulées par l'application.
Il assure la gestion de ces données et garantit leur intégrité.
Dans le cas typique d'une base de données, c'est le modèle qui la contient.

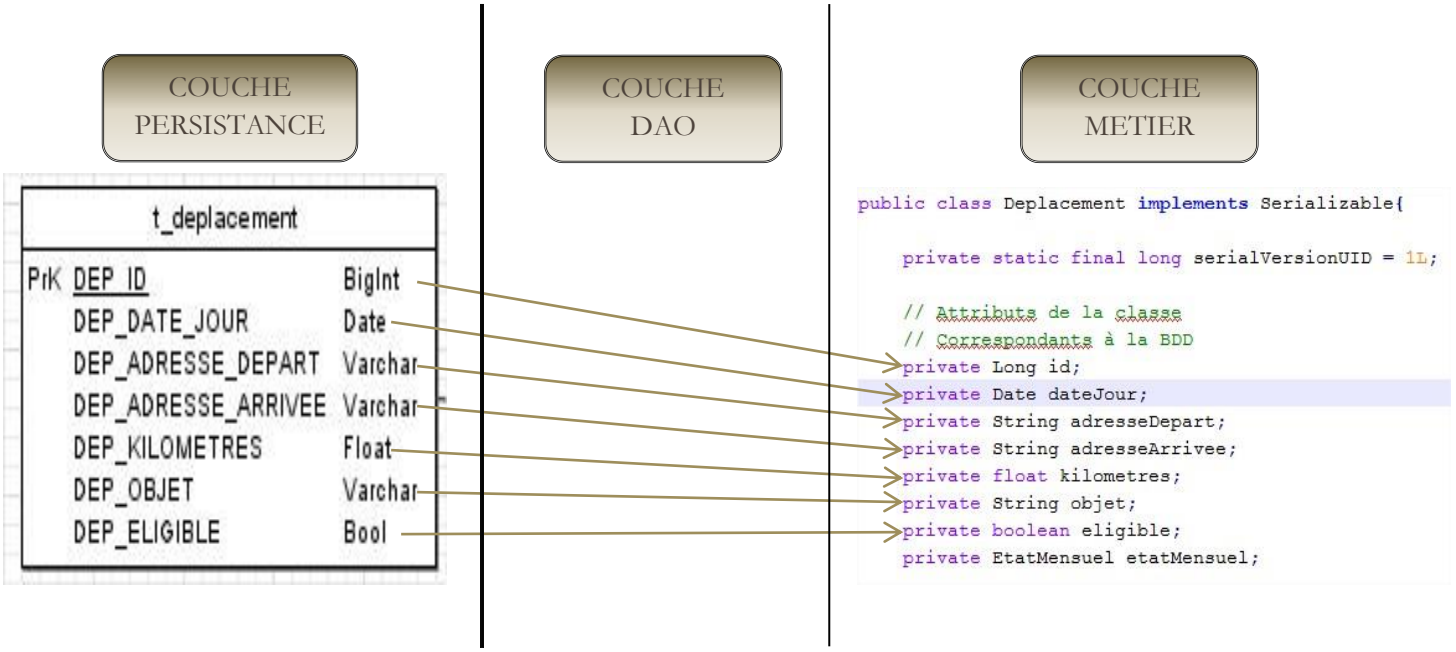
L'**ORM** (Object Relational Mapping) est une technique de programmation qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle en définissant des correspondances entre cette base de données et les objets du langage utilisé (JAVA).

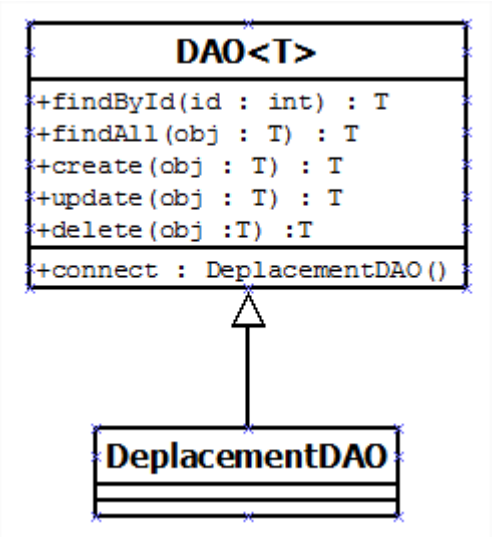
Dans le cas de l'application que j'ai eu à développer, le Mappage des données est effectué avec le Pattern DAO.

Le **pattern DAO** (Data Access Object) permet de faire le lien entre la couche métier et la couche persistance, ceci afin de centraliser les mécanismes de mapping entre notre système de stockage et nos objets Java.
La couche persistance correspond à notre système de stockage et la couche métier correspond à nos objets Java.
Le pattern DAO consiste à ajouter un ensemble d'objets dont le rôle sera d'aller :

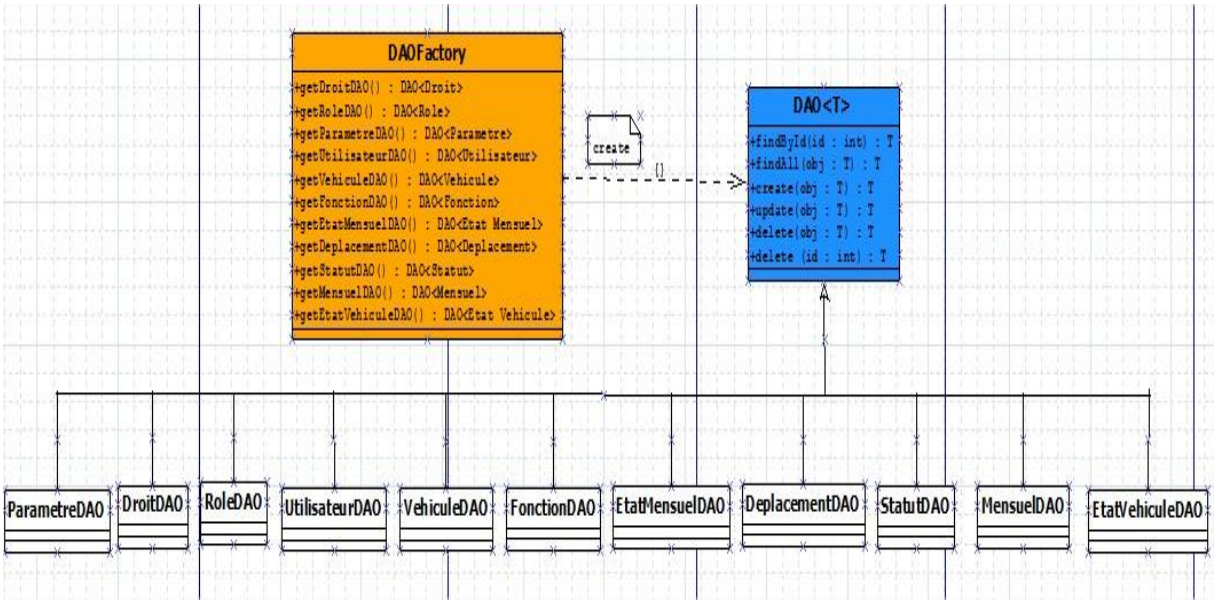
- Lire
- Ecrire
- Modifier
- Supprimer

Ces quatre tâches basiques sont souvent raccourcies à l'anglaise : **CRUD** (Create, Read, Update, Delete).





Pour plus de souplesse, le pattern DAO a été couplé avec le pattern Factory.
Le **pattern Factory** permet d'encapsuler l'instanciation de nos objets dans la classe.
Cela permet de prévenir des modifications sur la façon de créer les objets car l'instanciation sera centralisée dans un seul objet.



9.3.2. VUE / LES PAGES JSP

La vue correspond à l'interface avec laquelle l'utilisateur interagit.
Sa première tâche est de présenter les résultats renvoyés par le modèle.
Sa seconde tâche est de recevoir toutes les actions de l'utilisateur.
La vue n'effectue aucun traitement, elle se contente d'afficher les résultats des traitements effectués au Modèle.

Dans l'application que j'ai eu à développer, ce sont des pages **JSP (Java Serveur Pages)** qui servent de vue.
Les pages JSP sont des pages web mais contrairement aux pages HTML, les pages JSP peuvent inclure du code Java.
Elles permettent aussi de manipuler des **Beans** (objet Java réutilisable, paramétrable, sérialisable).

Voici un exemple développé dans l'application :

- Une classe action passe un Bean en attribut de requête :

```
// Ajouter bean en parametre de requete
request.setAttribute("utilisateur", utilisateur);
```

- On récupère l'attribut de requête dans la page JSP grâce à la balise useBean :

```
<jsp:useBean id="utilisateur" scope="request" class="beans.Utilisateur" />
```

- On peut enfin utiliser le Bean Utilisateur dans la page JSP :

```
<div id="message_header" class="panel-heading">
    <h4>Profil utilisateur de <strong><%=utilisateur.getPrenom()%> <%=utilisateur.getNom()%></strong>
</div>
```

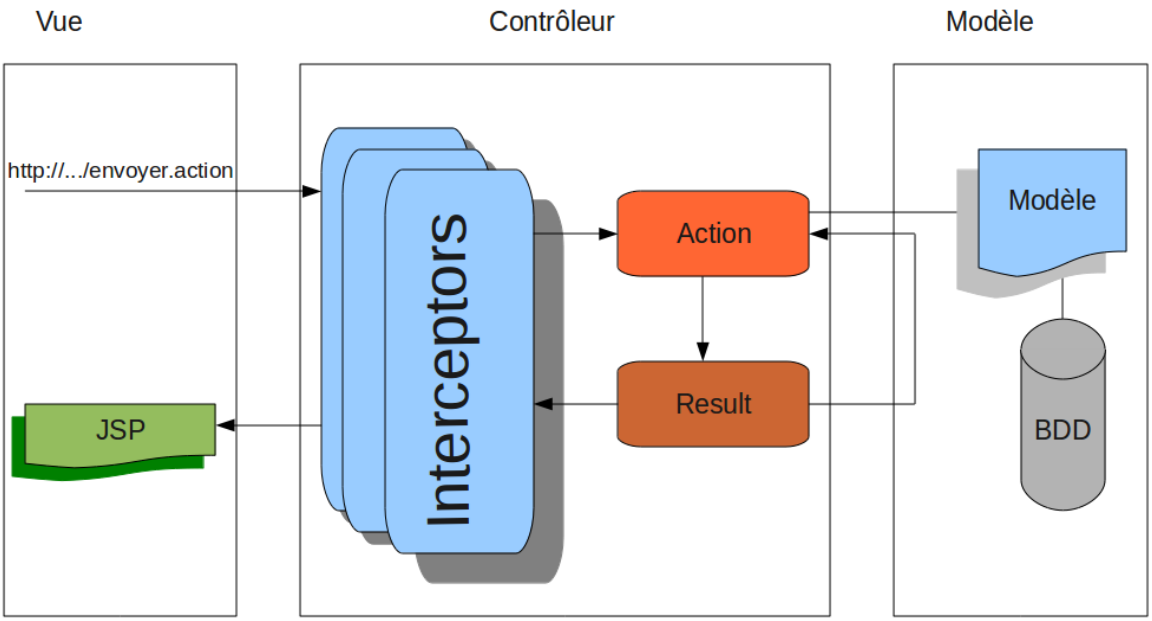
9.3.3. CONTRÔLEUR : LE FRAMEWORK STRUTS2

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchronise.
Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer.
Si une action nécessite un changement de données, le contrôleur demande la modification de ces données au Modèle et ensuite avertit la vue que les données ont changé pour qu'elle se remette à jour.
Certains événements ne concernent pas les données mais la vue, dans ce cas le contrôleur demande à la vue de se modifier.
On remarque bien que le contrôleur n'effectue aucun traitement ni ne modifie aucune données, il sert simplement d'aiguilleur.

L'application est développée avec le **Framework STRUTS2**.
Le choix de cette technologie repose sur :

- sa **fiabilité** : excellente réputation.
- sa **flexibilité** : chaque action peut être personnalisée, les fichiers de configuration sont simples d'utilisation.
- sa **performance** : elle est particulièrement performante et maintenable grâce à la séparation en couches.

C'est un Framework qui offre des outils de validation des entrées utilisateurs, des bibliothèques de balises JSP pour la création rapide de pages, une technique de routage pour les pages et accès web.
Il permet de gagner beaucoup de temps sur le développement de la partie Contrôleur.



9.3.3.1. Les fichiers de configuration

- **Web.xml :**

Le premier fichier de configuration du Framework STRUTS2 est le fichier **web.xml**
Il est généré automatiquement à la création d'un projet STRUTS.

On le trouvera dans le fichier WEB-INF, lui-même dans le fichier Webcontent.

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">

  <display-name>Cartable mobile</display-name>

  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
  </filter-mapping>

  <session-config>
    <session-timeout>30</session-timeout>
  </session-config>

</web-app>
```

- La balise <display-name> permet de définir le nom de l'application à afficher dans le navigateur.

- La balise <filter> définit un filtre implémenté par la classe dispatcher de STRUTS2. C'est cette classe qui jouera le rôle de contrôleur du modèle MVC.

- La balise <fliter-mapping> définit une liaison entre un modèle d'URL et le filtre qui doit traiter les URL qui suivent ce modèle.

- La balise <session-config> définit la durée d'une session utilisateur, ici 30 minutes.

- **Struts.xml :**

C'est le fichier de configuration de STRUTS2.
Ce fichier est à la racine du projet dans le fichier SRC.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

<!--     <constant name="struts.action.extension" value="" /> -->
    <constant name="struts.enable.DynamicMethodInvocation" value="false" />
    <constant name="struts.devMode" value="false" />
    <constant name="struts.custom.i18n.resources" value="package"  />

    <package name="beanAction" namespace="/" extends="struts-default, json-default">

        <!--  action par defaut qui envoie sur la page login -->
        <default-action-ref name="login" />

        <action name="login">
            <result>login.jsp</result>
        </action>

        <!--  Identification -->
        <!--  Appel de la méthode identifierUtilisateurs() de la classe AuthentificationAction  -->
        <action name="authentification" class="beanAction.AuthentificationAction" method="authentifierUtilisateur" >
            <result name="success">welcome.jsp</result>
            <result name="error">login.jsp</result>
            <result name="input">login.jsp</result>
        </action>

        <!--  Page accueil -->
        <action name="pageAccueil" class="beanAction.PageAccueilAction" method="pageAccueil">
            <interceptor-ref name="maPileIntercepteurs"/>
            <result name="success">welcome.jsp</result>
            <result name="login">login.jsp</result>
        </action>

        <!--  Page délibération -->
        <!-- Je dois passer par une action pour placer le nom de la page dans la session -->
        <action name="pageDelib" class="beanAction.PageDelibAction" method="pageDelib">
            <result name="success">recherche_delib.jsp</result>
        </action>
```

- Les balises <constant> permettent de définir des configurations constantes à l'application (struts.devMode définit si on utilise le mode développeur pour afficher les erreurs).

- Les balises <action> déclarent les actions correspondant aux actions de l'utilisateur de l'application.
  On définit d'abord le nom de l'action, la classe qui l'interprète ainsi que du nom de la méthode de la classe d'action et enfin le nom des JSP ou seront redirigés les résultats.

9.3.3.2. Les actions :

Une fois les actions déclarées dans le fichier de configuration struts.xml, il faut créer une classeAction par action.

Une classeAction correspond à une classe qui va encapsuler les traitements nécessaires aux actions et qui en fonction du résultat (result) qu'elle renverra, l'application aura telle réaction.

Bean.Action : Méthode envoyerEtatMensuelAssemblee () de la classeAction envoyerEtatMensuelAssembleeAction

```
public String EnvoyerEtatMensuelAssemblee() throws NumberFormatException, DAOException, ParseException{

    // Récupérer la map des paramètres de requête
    // les paramètres JS x-editable sont envoyés en tant que paramètres de requête
    request = (HttpServletRequest) ActionContext.getContext().get(ServletActionContext.HTTP_REQUEST);

    // Si le parametre de requete idUtilisateurCorrespondant existe
    if (request.getAttribute("idEtatMensuelAModifier")!=null) {
        // Le stocker dans une variable
        idEtatMensuelAModifier = (Long) request.getAttribute("idEtatMensuelAModifier");
        // Positionner le boolean a false
        profilEtatMensuelExistant=false;
    }
        // Récupérer la session utilisateur
        session = (SessionMap<String, Object>) ActionContext.getContext().getSession();

        // Le stocker dans une variable
        String idEtatMensuel = (String) request.getParameter("pk");
        System.out.println("id de l'etat : "+idEtatMensuel);

            // Instancier la fabrique pour MySQL
            MySQL = (MySQLDAOFactory) DAOFactory.getDAOFactory(1);

            // Instancier la dao
            etatMensuelDao = (EtatMensuelDao) MySQL.getEtatMensuelDao();

            // Récupérer les infos de l'etat mensuel en cours de modification:
            etatMensuel = etatMensuelDao.findByID(Long.parseLong(idEtatMensuel));

            // Positionner le statut par défaut (en cours de traitement par l'assemblee, id=2):
                    //instancier la dao
                    statutDao = (StatutDao) MySQL.getStatutDao();
                    // Récupérer le statut (id=2)
                    statut = statutDao.findByID(2L);
                    // Positionner le statut sur l'etat mensuel
                    etatMensuel.setStatut(statut);

            // Positionner la date d'envoi de l'etat mensuel à aujourd'hui (le jour de l'envoi)(par défaut):
                    Date datedEnvoi = new Date(Calendar.getInstance().getTime().getTime());
                    etatMensuel.setDateCreation(datedEnvoi);

            //mettre a jour l'etat mensuel:
            etatMensuel = etatMensuelDao.updateEnvoiEtatParElu(etatMensuel);

            // Si l'état mensuel renvoyé n'est pas null
    if(etatMensuel != null){
        return SUCCESS;
    }//sinon
    else{
        return ERROR;
    }
}
```

Voici la déclaration de l'action dans struts.xml :

```
<!-- Envoi de l'etat mensuel au service de l'Assemblee  -->
<action name="EnvoyerEtatMensuelAssemblee" class="beanAction.EnvoyerEtatMensuelAssembleeAction" method="EnvoyerEtatMensuelAssemblee">
    <result name="success">listeEtatMensuels.jsp</result>
    <result name="error">listeDeplacements.jsp</result>
</action>
```

Lorsque la classeAction retourne SUCCESS alors l'utilisateur est redirigé vers la page JSP listeEtatMensuels.jsp mais si elle retourne ERROR alors l'utilisateur se retrouvera sur la page listeDeplacements.jsp

9.4. LES LIBRAIRIES JAVASCRIPT UTILISÉES

9.4.1. JQUERY

JQUERY est une bibliothèque JavaScript donc un fichier d'extension .js.

JavaScript est un langage de programmation de scripts employé pour dynamiser des pages WEB. JQUERY comparé à JavaScript est beaucoup moins verbeux et donc il est moins susceptible de commettre des erreurs. Il tient compte également des navigateurs présents sur le marché, de leurs multiples versions et de leur comptabilité avec les instructions des langages JavaScript & Ajax.

Pour comprendre le fonctionnement de JQUERY, il faut tout d'abord faire référence au **DOM (Document Object Model)** qui est la structure de données qui représente un document HTML comme une arborescence.
La racine de cet arbre est un **nœud** nommé « document ».
Les balises HTML du DOM correspondent aux nœuds de l'arbre DOM et en constitue la structure.
Le langage JQUERY permet d'interroger le DOM pour connaitre les caractéristiques des balises ou de modifier ces éléments.

Pour utiliser JQUERY dans une page HTML ou JSP, dans le cas de notre application, il suffit d'y faire référence en utilisant la balise <script> :

```
<script type="text/javascript" src="js/jquery.js"></script>
```

Exemple d'utilisation dans l'application :

```
//selection de la ligne:
 $('#tableDeplacements tbody').on( 'click', 'tr', function () {
        if ( $(this).hasClass('selected') ) {
            $(this).removeClass('selected');
        }
        else {
            table.$('tr.selected').removeClass('selected');
            $(this).addClass('selected');
        }
 });
```

- Ligne 1 : A chaque fois que l'utilisateur clique (« click ») sur une ligne (« tr ») du tableau dont l'identifiant est « tableDeplacements » alors la fonction définit dans « function(){} » se lancera.

- Ligne 2 à 3 : si celle-ci (« this ») (qui correspond à la ligne cliquée) a la classe « sélectionné » alors elle prendra la classe « désélectionné ».

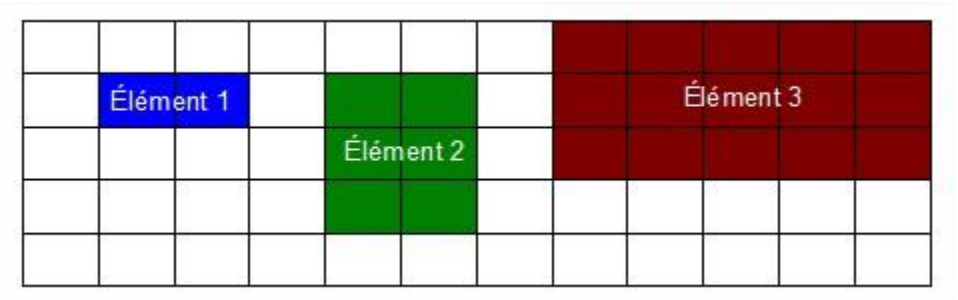
- Ligne suivants : sinon elle prendra la classe « sélectionné ».

9.4.2. BOOTSTRAP

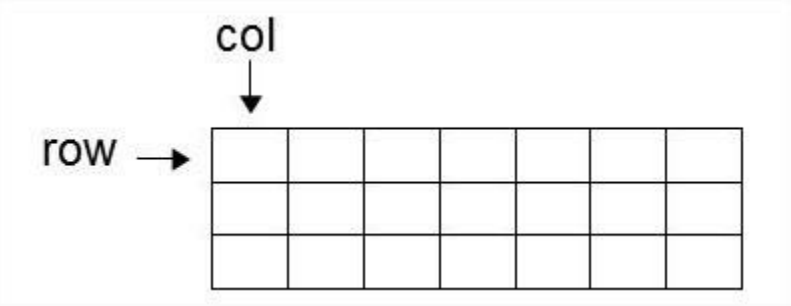
Bootstrap est une collection d'outils utile à la création de sites et d'applications web.
C'est un ensemble qui contient du code HTML & CSS.

Cette librairie permet de rendre une application **web responsive**, c'est-à-dire que l'affichage de l'application **s'adaptera à la taille de l'écran** utilisé en utilisant le **redimensionnement** & l'**empilement des éléments alignés**.

Bootstrap fonctionne avec un système de grille.
On peut alors organiser du contenu en utilisant pour chaque composant une ou plusieurs cases de la grille :



Une grille est découpée en lignes (« row ») et en colonnes (« col ») :



Il faut donc définir le nombre de colonnes pour chaque élément sachant qu'au maximum il y en a 12.

Bootstrap considère 4 types de tailles d'écrans :

	Petit écran (smartphone)	Écran réduit (tablette)	Écran moyen (desktop)	Grand écran (desktop)
				
Comportement	Redimensionnement	Redimensionnement	Redimensionnement	Empilage puis redimensionnement
Classe	col-xs-*	col-sm-*	col-md-*	col-lg-*
Valeur de référence	< 768 px	>= 768 px	>= 992 px	>= 1200 px

Voici un petit exemple qui permet de mieux visualiser le fonctionnement de Bootstrap :

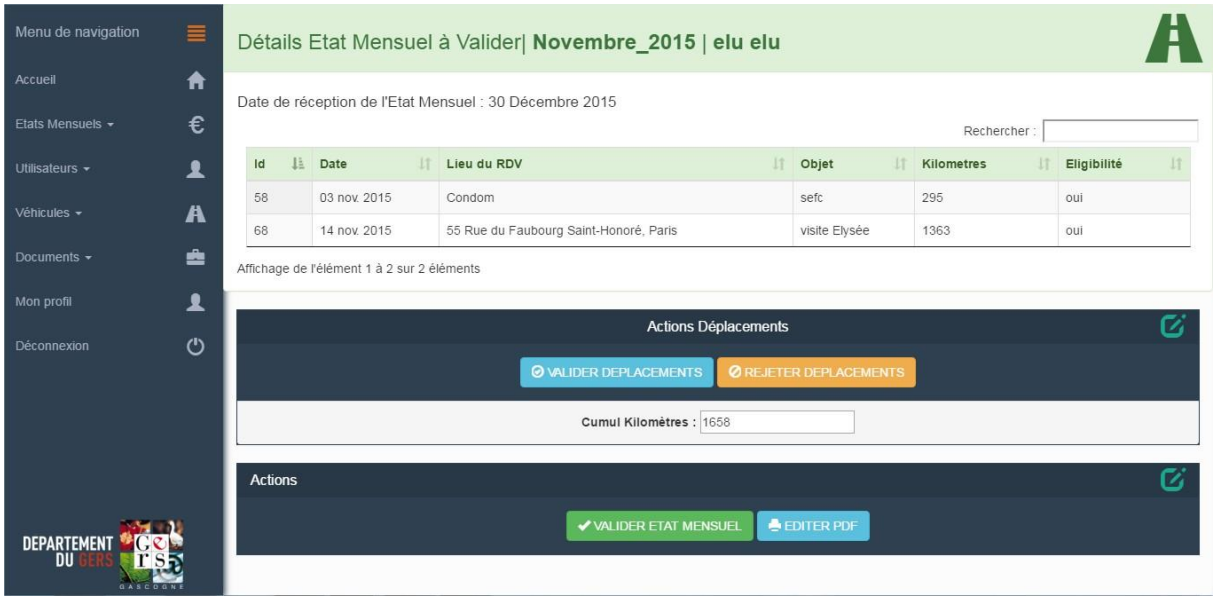
```html
<!DOCTYPE html>
<html>
  <head>
    <link href="assets/css/bootstrap.css" rel="stylesheet">
    <link href="assets/css/tuto.css" rel="stylesheet">
  </head>
  <body>
    <div class="container">
      <header class="row">
        <div class="col-lg-12">
          Entete
        </div>
      </header>
      <div class="row">
        <nav class="col-lg-2">
          Menu
        </nav>
        <section class="col-lg-10">
          Section
        </section>
      </div>
      <footer class="row">
          Pied de page
      </footer>
    </div>
  </body>
</html>
```

Le résultat se trouve à la figure suivante.

Voici maintenant le résultat sur l'application :

- Affichage sur grand écran (PC) :

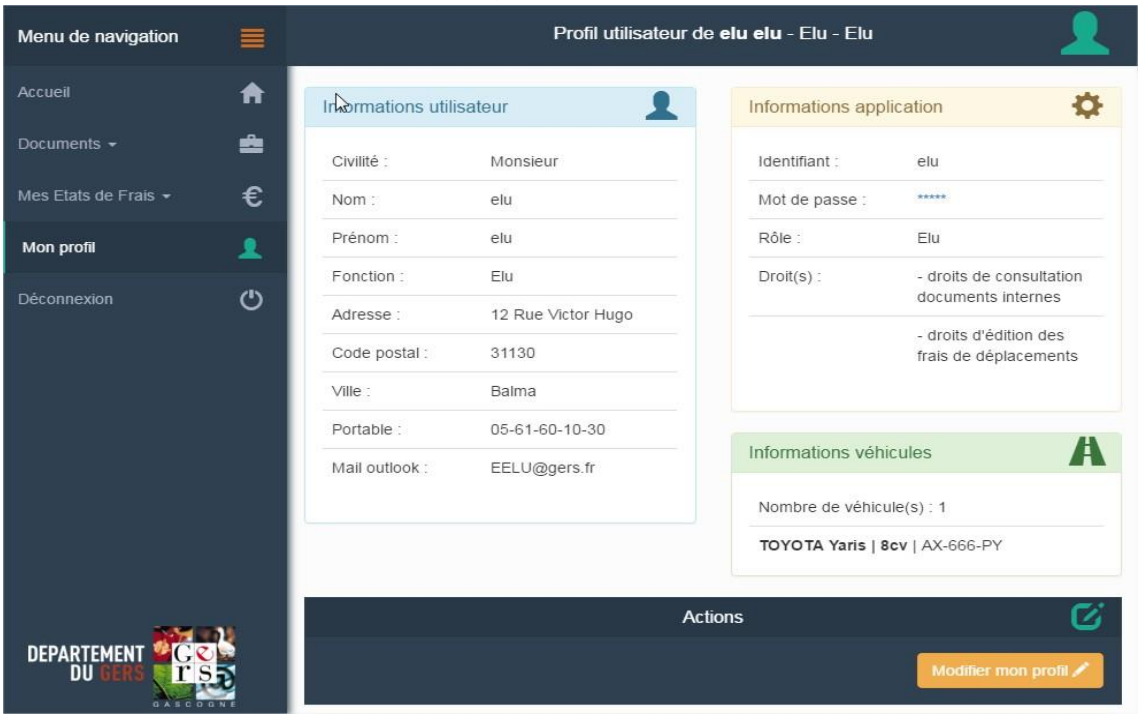
  Ce mode d'affichage sera utilisé pour la partie BACK OFFICE car les personnes travaillant au traitement des frais de déplacements utilisent des ordinateurs fixes avec des écrans de grande taille.



- Affichage sur écran moyen :

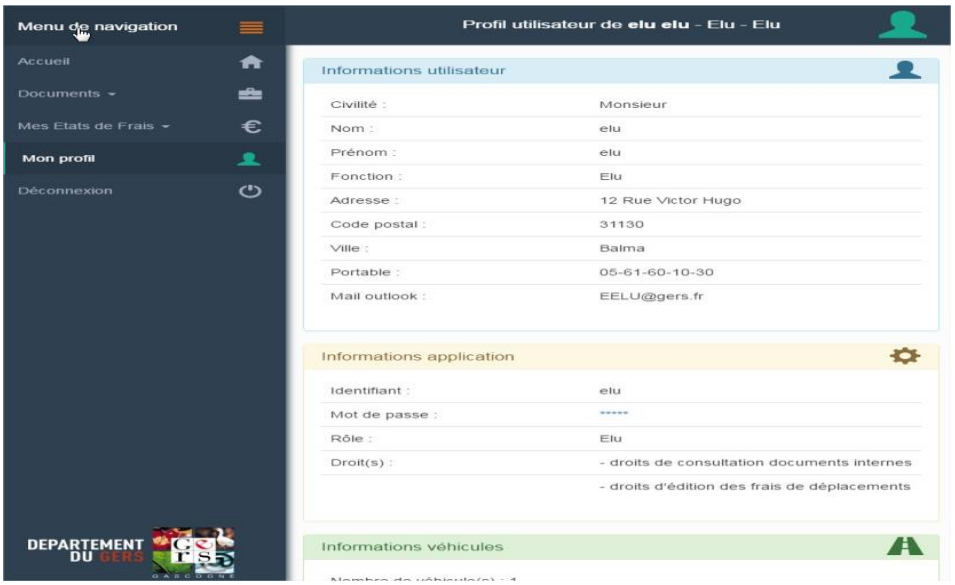
J'ai pris en compte la possibilité que les élus (utilisateur de la partie FRONT OFFICE) puissent accéder à l'application de chez eux sur un ordinateur portable.

Voici le résultat à l'affichage : on remarque l'empilement du bloc « Action »



- Affichage sur tablette :

Chaque utilisateur de l'application (les élus) disposent d'une tablette, c'est naturellement que j'ai pris en compte ce cas d'utilisation.
Dans cette capture d'écran on remarque l'empilement des blocs d'informations pour que l'affichage reste fluide.



- L'affichage sur Smartphone n'a pas été pris en compte pour des raisons techniques, l'affichage complet de tableaux ou de formulaires nécessite un minimum de place.

Bootstrap offre aussi une multitude de composants :

- Des icônes : appelés GLYPHICONS, gratuits et d'un large choix.

- Des boutons :

Pour obtenir un résultat visuel adapté aux attentes des utilisateurs, j'ai couplé ces deux composants :

9.4.3. AJAX

Ajax (Asynchronous JavaScript And XML) est un concept de programmation Web reposant sur plusieurs technologies comme JavaScript et XML ou parfois JSON.

Le principe de son fonctionnement est de faire communiquer une page Web avec un serveur Web sans occasionner le rechargement de la page (asynchrone).
C'est pour cette raison que JavaScript est utilisé, car c'est lui qui va se charger d'établir la connexion entre la page Web et le serveur.

- Exemple d'utilisation dans l'application :

```
// Fonction qui récupére la liste des utilisateurs :
function getUtilisateurs() {
    return $.ajax({
        type:  'GET',
        async: true,
        url:   "listerUtilisateursJSON.action?chargementSelect=true",
        dataType: 'json'
    });
}
```

La méthode getUtilisateurs () renvoi un appel Ajax pour nous renvoyer la liste des utilisateurs sans avoir à recharger la page entière.

- GET : c'est le type utilisé pour récupérer des données.
- Async : permet de définir si la requête est asynchrone ou pas.
- url : adresse à laquelle la requête sera envoyée au serveur
- dataType : définit de type de données à récupérer

Dans cet exemple l'adresse à laquelle est envoyée la requête est une classe Action STRUTS nommée ListerDeplacementJSONAction :

```
public class ListerUtilisateursJSONAction extends ActionSupport {

    @SuppressWarnings("unchecked")
    public void listerUtilisateursJSON() throws DAOException, IOException {

        // Récupérer la session utilisateur
        session = (SessionMap<String, Object>) ActionContext.getContext().getSession();
        // Récupérer la map des paramètres de requête
        request = (HttpServletRequest) ActionContext.getContext()
                .get(ServletActionContext.HTTP_REQUEST);

        // Si l'id de l'utilisateur connecté n'est pas null
        if (session.get("idUtilisateur")!=null)
            // Stocker cet id dans une variable
            idUtilisateur = (Long) session.get("idUtilisateur");
        // Instancier la fabrique pour MySQL
        MySQL = (MySQLDAOFactory) DAOFactory.getDAOFactory(1);
        // Instancier la dao
        utilisateurDao = (UtilisateurDao) MySQL.getUtilisateurDao();
        vehiculeDao = (VehiculeDao) MySQL.getVehiculeDao();

        // Récupérer la liste de tous les utilisateurs
        listeUtilisateurs = utilisateurDao.findAll();

        // Si l'utilisateur modifie un véhicule (on reçoit un paramètre boolean pour aiguiller)
        // Chargement de la liste des utilisateurs avec seulement leur nom et prénom (pour le select des pages modifierVehiculer et creerVehicule)
        if (request.getParameter("chargementSelect") !=null ) {

            // Pour chaque utilisateur dans la liste d'utilisateurs
            for (Utilisateur utilisateur : listeUtilisateurs) {

                // Créer un Objet JSON
                utilisateurJSON = new JSONObject();
                // Attribuer les propriétés de chaque utilisateur
                // ici on a besoin de l'id, du nom et du prénom
                utilisateurJSON.put("value", utilisateur.getId());
                utilisateurJSON.put("text", utilisateur.getNom()+" "+utilisateur.getPrenom());
                // Ajouter chaque objet JSON dans le tableau JSON
                tableauUtilisateursJSON.add(utilisateurJSON);
            }

            // Définir le type MIME (format JSon)
            response.addHeader("Content-Type", "application/json");
            // Créer flux d'écriture
            PrintWriter flux = response.getWriter();
            // Ecrire wrapper JSON
            tableauUtilisateursJSON.writeJSONString(flux);
            // Fermer flux d'écriture
            flux.close();
        }
```

Cette classe instancie la DAO et va chercher toutes les fonctions existantes en base de données (le cas échéant findAll ()).

Ensuite pour chaque objet Utilisateur remonté par la fonction findAll, on crée un objet JSON dans lequel on injecte l'id et le nom-prénom de l'utilisateur.
Chaque objet JSON sera intégrer à un tableau JSON.

Enfin on écrit la réponse du traitement de la classe au format JSON pour que la requête AJAX puisse la récupérer et l'utiliser.

9.4.4. X-EDITABLE

Cette librairie permet de créer des éléments modifiables sur une page WEB.
Il doit être utilisé avec BOOTSTRAP et JQUERY pour fonctionner.

- Voici donc les appels aux librairies BOOTSTRAP & JQUERY :

```
<link href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css" rel="stylesheet">
<script src="http://code.jquery.com/jquery-2.0.3.min.js"></script>
<script src="//netdna.bootstrapcdn.com/bootstrap/3.0.0/js/bootstrap.min.js"></script>
```

- Appel de la librairie X-EDITABLE :

```
<link href="bootstrap-editable/css/bootstrap-editable.css" rel="stylesheet">
<script src="bootstrap-editable/js/bootstrap-editable.js"></script>
```

Cette librairie fonctionne selon deux modes :

- Le mode inline : édition des éléments à modifier directement sur l'élément.
- Le mode popup : édition des éléments à modifier dans une popup.

```
// Champs editables intégrés à la ligne
$.fn.editable.defaults.mode = 'inline';
```

X-EDITABLE ne peut être utilisé que sur des éléments de balisage.
Habituellement, on l'utilise sur l'élément <A> avec des attributs d'option nommés data-*.

```
<a href="#" id="objet" class="myeditable" data-type="text" data-name="objet"></a>
```

Ensuite il suffit d'appliquer la méthode editable () sur cet élément.

```
$('#objet').editable({
    onblur : 'submit',
    clear : false,
    tpl: "<input type='text' style='width: 29vw'>",
    emptytext : "Veuillez saisir l'objet de votre déplacement"
});
```

On peut y associer d'autres options telles que la vérification de la validation du champ en question.

```
// Validation de l'objet du deplacement
$('#objet').editable('option', 'validate', function(v) {
    if(!v.trim()) return "L'objet doit être renseigné";
    else if(v==="Veuillez saisir l'objet de votre déplacement") return "L'objet est erroné";
    else if(v.length<3) return "L'objet est erroné";
});
```

- Si l'objet modifié est vide alors le message d'alerte « L'objet doit être renseigné » apparait.

- Si l'objet n'a pas été modifier ou que sa taille est inférieure à trois caractères alors un message d'alerte « L'objet est erroné » apparait.

9.4.5. DATATABLE

Le plugin DataTable est une librairie JQUERY très flexible qui permet d'améliorer l'accessibilité des données dans une table HTML.

Il utilise une multitude d'options qui peuvent être utilisées pour configurer la façon dont les données sont obtenues et comment les traiter.
Il prend en charge presque toutes les sortes de données (DOM, JavaScript, Ajax, et le traitement côté serveur).

- Pour l'utiliser dans une page WEB, il faut commencer par appeler les librairies JavaScript nécessaire au fonctionnement de DataTable :

```
<script type="text/javascript" charset="utf8" src="js/jquery.dataTables.js"></script>
<link rel="stylesheet" type="text/css" href="style/css/jquery.dataTables.css">
```

- Ensuite il faut déclarer un tableau HTML :

```
<table id="tableDeplacements" class="display table-bordered table-striped select" >
    <thead>
        <tr>
            <th >Date</th>
            <th >Ville d'arrivée</th>
            <th >Objet</th>
            <th >Kilomètres</th>
        </tr>
    </thead>
    <tbody>
    </tbody>
</table>
```

- Enfin, on appelle la fonction $().DataTable :

```
function listerDeplacements() {
        $.ajax({
            // A cette url (action) avec parametre idEtatMensuel
            url : 'listerDeplacementsJSON.action?idEtatMensuel='+'<%=request.getParameter("idEtatMensuel")%>',
            // Format JSON
            dataType : 'json',
            contentType: "application/json",
            // En cas de succès
            success : function(data) {
            // Remplir la liste de deplacements avec les donnees
            listeDeplacements = data;

                //creation de la table:
                var table = $('#tableDeplacements').DataTable({
                     dom: "Bfrtip",
                     stateSave: true,
                     async : true,
                     //où va chercher les données pour remplir le tableau:
                     "aaData" : listeDeplacements.records,
                     "DT_RowId": "#",
                     serverSide: false,
                     "aoColumns": [
                                 {"mDataProp" : "date"},
                                 {"mDataProp" : "arrivée"},
                                 {"mDataProp" : "objet"},
                                 {"mDataProp" : "kilomètres"}
                                 ],

                     //option qui rend les lignes sélectionnables:
                     select : {
                            style:    'os',
                            selector: 'td:first-child'
                        },

                     //internationalisation(traduction):
                     "oLanguage" : {
                            "sProcessing":     "Traitement en cours...",
                            "sSearch":         "Rechercher :",
                            "sLengthMenu":     "Afficher _MENU_ &eacute;l&eacute;ments",
                            "sInfo":           "Affichage de l'&eacute;l&eacute;ment _START_ &agrave; _END_ sur _TOT
                            "sInfoEmpty":      "Affichage de l'&eacute;l&eacute;ment 0 &agrave; 0 sur 0 &eacute;l&ea
                            "sInfoFiltered":   "(filtr&eacute; de _MAX_ &eacute;l&eacute;ments au total)",
                            "sInfoPostFix":    "",
                            "sLoadingRecords": "Chargement en cours...",
                            "sZeroRecords":    "Aucun &eacute;l&eacute;ment &agrave; afficher",
                            "sEmptyTable":     "Aucune donn&eacute;e disponible dans le tableau",
                            "oPaginate": {
                                "sFirst":      "Premier",
                                "sPrevious":   "Pr&eacute;c&eacute;dent",
                                "sNext":       "Suivant",
                                "sLast":       "Dernier"
                            },
                            "oAria": {
                                "sSortAscending":  ": activer pour trier la colonne par ordre croissant",
                                "sSortDescending": ": activer pour trier la colonne par ordre d&eacute;croissant"
                            }
                        }
                 });
```

Dans cet exemple qui fait partie de l'application, j'ai fait un appel Ajax pour récupérer la liste des déplacements qui servira à peupler mon tableau.

Les données renvoyées par l'appel Ajax sont de type JSON.
Les options :

- DOM : définit les éléments de commande de la table (B : boutons, f : filtres, r : éléments d'affichage de traitement, t : table, i : information du sommaire, p : pagination)
- STATE SAVE : restaure l'état de la table au rechargement de la page.
- ASYNC : rend l'appel asynchrone.
- aaData : permet de récupérer les données JSON à afficher.
- DT_RowId : « # » : permet de mapper l'id du tableau avec l'id des objets à afficher.
- SERVER SIDE : traitement côté serveur.
- aoColumn : permet de mapper les colonnes du tableau aux attributs de l'objet à afficher.
- SELECT : permet de rendre les lignes du tableau sélectionnable.
- oLangage : permet de faire de l'i18n (internationalisation).

Résultat à l'affichage :

9.4.6. BOOTBOX

Bootbox est une bibliothèque JavaScript qui permet de créer des boîtes de dialogue de programmation utilisant des boites modales Bootstrap.

Les options sont variées :

On peut modifier le titre, la taille (ClassName), les animations à l'affichage, le message d'elle inclue, le ou les boutons et le retour une fois le bouton cliqué.

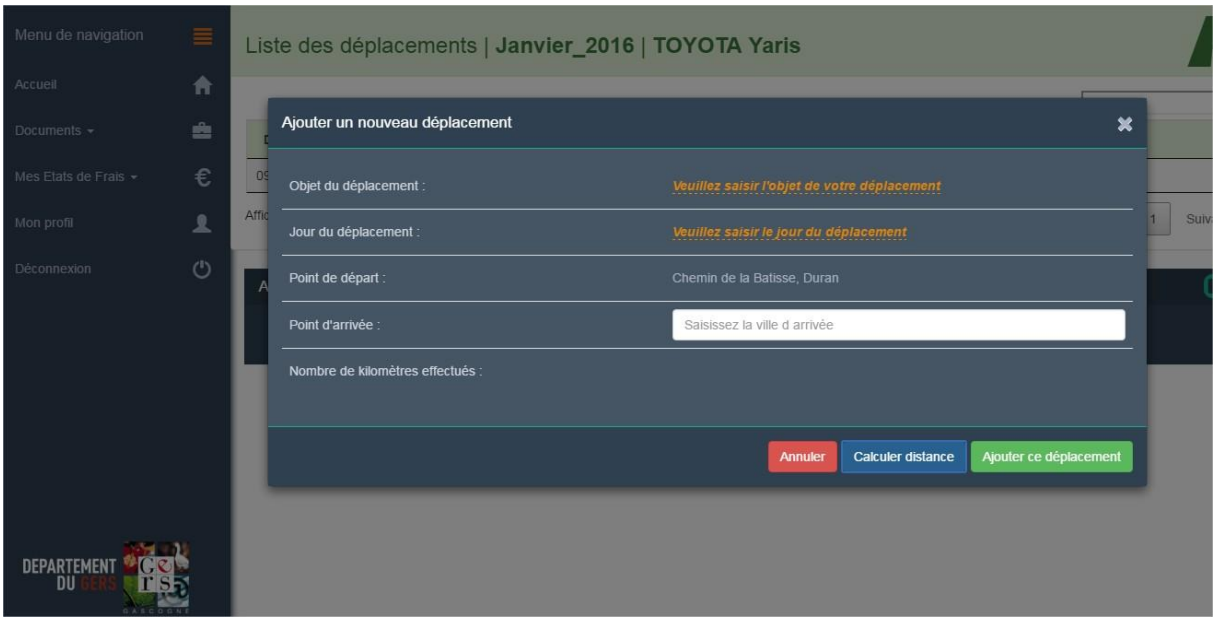
Pour des raisons de rapidité d'exécution dans le développement, de rendu satisfaisant et du fait que les librairies BOOTSTRAP et JQUERY nécessaires à son fonctionnement, j'ai utilisé la librairie BOOTBOX pour les popup de confirmation, d'alerte et même pour afficher une popup pour la saisie des frais de déplacements par les élus.

```
// Fonction qui instancie la popup de saisie des deplacements
function instancierPopUp() {

        // Instancier popup
        bootbox.dialog({
            title: "Ajouter un nouveau déplacement",
            className: "x-large",
            animate : true,
            // Formulaire de saisie sous forme de table HTML
            message: formulaireHTML,
            // 3 cas possibles
            buttons: {
                // 1er cas : Annuler
                danger: {
                    label: "Annuler",
                    className: "btn-danger",
                    callback: function () {}
                },
                // 2eme cas : Calculer la distance (apres avoir saisi la destination)
                calcul : {
                    label: "Calculer distance",
                    className: "btn-primary",
                    callback: function() {
                        calculerDistance();
                        return false; // pour ne pas fermer la popup
                    }
                },
                // 3eme cas - Ajouter le deplacement (apres avoir calculé le nombre de kilomètres)
                success: {
                    label: "Ajouter ce déplacement",
                    className: "btn-success",
                    callback: function () {
                        calculerDistance();
                        submitDeplacement();

                    }
                }
            }
        });
```

Résultat à l'affichage :



9.4.7. GOOGLE PLACE AUTOCOMPLETE API

L'API GOOGLE PLACE AUTOCOMPLETE est un service retournant des informations sur les emplacements.
Elle permet de faciliter l'expérience des utilisateurs mais sa gratuité est limitée à 25.000 requêtes par jour, ce qui est largement nécessaire pour notre cas d'utilisation.
Dans l'application, j'ai utilisé cette API, pour que lorsque l'utilisateur saisie un déplacement, quand il commence à taper l'adresse d'arrivée du déplacement, l'auto complétion lui propose des destinations.

Le champ dans lequel l'utilisateur tape une adresse retourne une liste d'adresses possibles en fonction des quelques caractères déjà tapés.
Lorsque l'adresse est sélectionnée, les données qui la composent sont parsées et injectées dans les champs de la base de données.

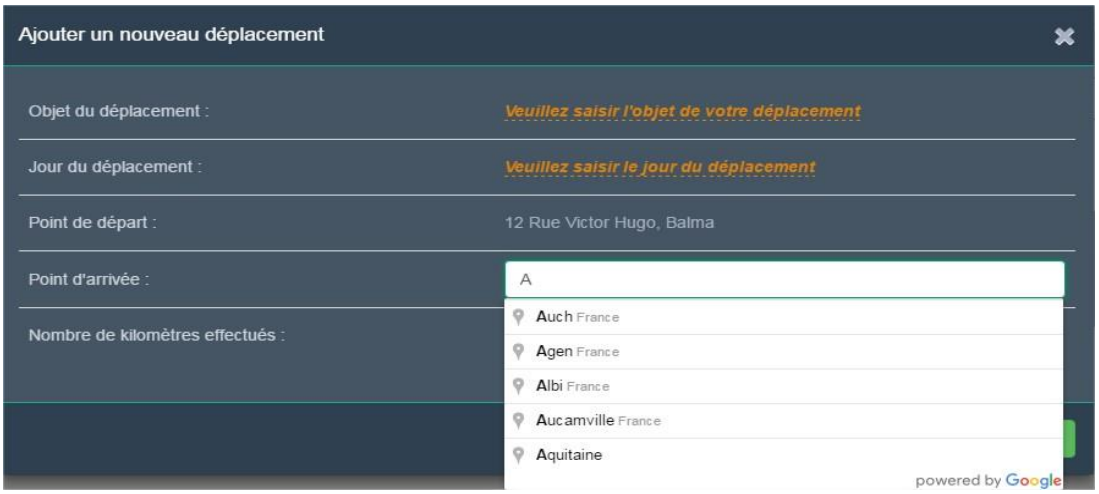
- Appel de l'API Google Place :

```
<script type="text/javascript" src="http://maps.googleapis.com/maps/api/js?key=AIzaSyCX9fNXA3tIABLmW2vOB61Mth5UQquyWRq&signed_in=true&libraries=places"></script>
```

- Méthode qui active l'auto complétion Google Place sur l'input « adresseArrivee » :

```
function activerAutoCompletionAdresse() {
        // Récupérer l'input pour activer l'autocompletion
        var input = document.getElementById('adresseArrivee');
        // Définir options d'autocompletion
        var options = {
                    type:['(cities)'],
                    // restriction uniquement villes françaises
                    componentRestrictions: {country: 'FR'}
                };
        // Activer l'autocompletion sur l'input
        autocomplete = new google.maps.places.Autocomplete(input, options);
}
```

Résultat à l'affichage :



9.4.8. GOOGLE DISTANCE MATRIX API

L'API Google Distance Matrix est un service fournissant la distance et le temps de parcours pour chaque couple de point [départ, arrivée] contenu dans une matrice.
Dans notre cas, j'ai juste eu besoin de récupérer la distance entre deux points.

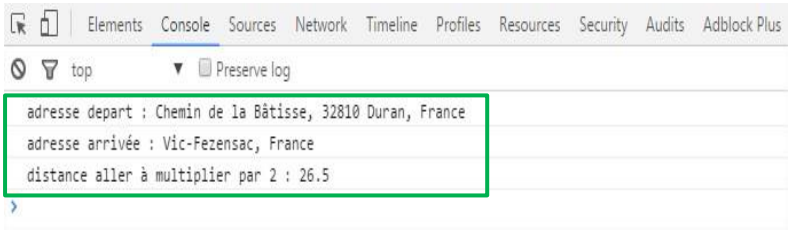
Le nombre quotidien de requêtes maximum est limité à 2.500 pour la version gratuite ce qui est suffisant pour l'utilisation que l'on en aura dans l'application.

Cette librairie a servi dans l'application App_e-Gers pour calculer la distance aller-retour des déplacements déclarés par les élus pour calculer le remboursement des frais engendrés par ces déplacements.

Utilisation dans l'application :

```
// Instancier l'api matrice (calcul de distance)
var service = new google.maps.DistanceMatrixService();
// Calcul de la distance (en voiture)
service.getDistanceMatrix ({
    // Point de depart
    origins: [departTexte],
    // Point d'arrivée
    destinations: [adresseArrivee],
    // Trajet de type voiture
    travelMode: google.maps.TravelMode.DRIVING,
    // Unite de distance
    unitSystem: google.maps.UnitSystem.METRIC,
    // Eviter autoroutes
    avoidHighways: false,
    // Eviter peages
    avoidTolls: false,
},
function(response, status) {
    // Si le serveur renvoie erreur
    if (status !== google.maps.DistanceMatrixStatus.OK) {
      // Afficher erreur
      bootbox.alert('Erreur : ' + status);
    // Sinon
    } else {
        // Afficher le nombre de kilomètres dans la popup
        $('#kilometres').text(((response.rows[0].elements[0].distance.value/1000)*2).toFixed(0)+" Kilomètres depuis votre résidence (Aller et retour
        compris).");
        // Stocker la distance dans une variable (en mètres)
        kilometres = ((response.rows[0].elements[0].distance.value/1000)*2).toFixed(0);
        var originList = response.originAddresses;
        var destinationList = response.destinationAddresses;
        console.log("adresse depart : "+originList);
        console.log("adresse arrivée : "+destinationList);
        console.log("distance aller à multiplier par 2 : "+kilometres/2);
```

Affichage de la console d'Eclipse :



Résultat à l'affichage :

10. BILAN

10.1. Difficultés rencontrées :

La première difficulté rencontrée a été de reprendre une application qui avait déjà été commencé par un ancien stagiaire.
J'ai donc passé beaucoup de temps à m'approprier l'architecture du projet et certaines librairies que je n'avais jamais utilisé.

La seconde difficulté en reprenant l'application, a été qu'un pan de l'activité n'avait pas été pris en compte.
J'ai relancé une phase de consultation pour identifier les problèmes d'exploitation pour remettre en adéquation l'application et les besoins et conditions d'utilisation.
J'ai donc mis en place une nouvelle organisation de travail : des réunions régulières et j'ai essayé de communiquer au maximum avec les différents acteurs qui m'entouraient.

Enfin j'ai eu quelques difficultés à installer et à paramétrer correctement le système de gestion du versioning SUBVERSION sur lequel j'ai passé l'équivalent d'une semaine.

10.2. Etat d'avancement du projet

L'état d'avancement du projet atteint 75% environ.
Les parties manquantes sont :

- La possibilité d'éditer au format PDF un état mensuel validé par le service des Assemblées (j'ai quand même prévu l'API que j'aurais pu utiliser : PDFBOX qui aurait parfaitement répondu aux besoins de l'application) pour qu'il parte au paiement.
- Un tableau de bord pour le service des Assemblées qui regroupe toutes les informations statistiques concernant les utilisateurs, leur véhicule, les déplacements & les états mensuels.

La reprise de l'existant et la mise en place du nouveau management du projet ont convergé vers l'obligation de faire des choix, qui ont été validé par le porteur du projet.
Je me suis donc consacré à l'achèvement des autres fonctionnalités de l'application.

La recette a été mise en exploitation pour une présentation au service RH du Conseil Départemental qui a été séduit par la facilité d'utilisation, l'ergonomie et l'efficacité de cette application et qui risque donc de généraliser cette application au remboursement des frais de tous les agents du département.

La présentation a donc validé la conformité de l'application aux attentes des utilisateurs.

10.3. Bilan et perspectives

Cette Période d'Application en Entreprise m'a permis de découvrir le métier de développeur mais surtout l'organisation et la gestion de Projet, ce qui a été une expérience extrêmement enrichissante.

En effet mon chef de service m'a laissé les rennes du projet et de sa gestion.
En plus des outils j'ai mis en place pour gérer le projet dans son ensemble, mes capacités, mes aptitudes et mon investissement ont pu servir au mieux le projet.
D'un point de vue technique, j'ai pu affûter mes connaissances et prendre de l'expérience et du recul sur le développement en utilisant des technologies que nous avions touché du doigt pendant la formation.

Mon travail et mon investissement ont été salué par le porteur du projet mais la conjoncture actuelle dans les collectivités territoriales a eu raison de la possibilité d'embauche qui s'offrait à moi.
Les autres perspectives qui se dégagent seraient de m'installer à mon compte dans le département du Gers où beaucoup de choses restent à faire dans le milieu de la conception et du développement informatique.