

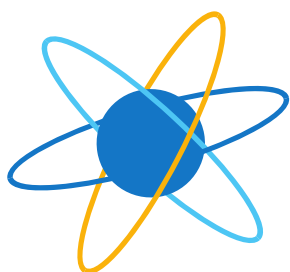
# SYSTÈME DE GESTION DES TRANSPORTS

---

Version 1.0

*27 mars 2014*

## Dossier d'Architecture Technique



Faculté  
des Sciences  
Aix\*Marseille Université





## Page d'informations

### CONTACTS

Prénom(s) & Nom	Adresse e-mail
Roland AGOPIAN	roland.agopian@univ-amu.fr
Ibrahima Sory BALDE	ibbaldes@yahoo.fr
Mehdi-Jonathan CADON	mehdi-jonathan@hotmail.fr
Lionel GAIROARD	lionel.gairoard@gmail.com
Anthony JULIEN	anthonyjulien2@gmail.com
Adrien LERICOLAIS	adrienmgs@gmail.com
Rémi MEZELLE	remi.mezelle@gmail.com
Ravi PACHY	pachy.ravi@gmail.com
Bien Aimé SUANGA WETO	maitreswing@gmail.com
Ahoua Khady TOURE	takamor91@yahoo.fr

### VERSIONS

Version	Date	Auteur(s)	Modification(s)
0.1	2014-03-06	Ahoua Khady TOURE	1
0.1	2014-03-06	Mehdi-Jonathan CADON	2
0.1	2014-03-06	Lionel GAIROARD, Adrien LERICOLAIS	3.3
0.1	2014-03-06	Anthony JULIEN, Rémi MEZELLE	3.4
0.1	2014-03-06	Ravi PACHY, Bien Aimé SUANGA WETO	3.5
0.1	2014-03-26	Rémi MEZELLE, Bien Aimé SUANGA WETO	Authentification et contrôle d'accès
1.0	2014-03-26	Mehdi-Jonathan CADON	Mise à jour et correction intégrale

### DIFFUSION

Version	Date	Approbateur(s)
1.0	2014-03-26	Mehdi-Jonathan CADON

### VALIDATION

Version	Date	Responsable(s)
1.0		Roland AGOPIAN



## Table des matières

<b>1</b>	<b>Présentation du projet</b>	<b>5</b>
1.1	Introduction	5
1.2	Calendrier du projet	5
1.2.1	Phase de réalisation du projet	5
1.3	Objectif du projet	5
1.4	Enjeux du projet	5
1.5	Exigences du projet	5
1.5.1	La localisation	5
1.5.2	L'interface utilisateur	6
1.5.3	Les cas d'utilisations	6
1.5.4	Les contraintes	6
<b>2</b>	<b>Architecture fonctionnelle</b>	<b>7</b>
2.1	Architecture fonctionnelle	7
2.2	Architecture physique	7
2.3	Fonctions de service et de contrainte	7
2.3.1	La localisation	7
2.3.2	Les cas d'utilisations	8
<b>3</b>	<b>Architecture technique</b>	<b>13</b>
3.1	Application	13
3.2	Authentification et contrôle d'accès	13
3.2.1	Pourquoi LDAP ?	13
3.2.2	Structure de l'annuaire	14
3.2.3	Architecture	15
3.2.4	Quand s'authentifier ?	15
3.2.5	Droits utilisateurs	15
3.2.6	Gestion des conflits	17
3.3	Données	18
3.3.1	Problématique	18
3.3.2	Le stockage	18
3.3.3	L'import/export de données	19
3.4	Sauvegarde	20
3.4.1	Problématique	20
3.4.2	Cluster	20
3.4.3	Sauvegarde en dur	21

<b>3.5</b>	<b>Synchronisation</b>	<b>22</b>
3.5.1	Introduction . . . . .	22
3.5.2	Gestion des priorités & des profils de synchronisation . . . . .	22
3.5.3	Mode connecté . . . . .	23
3.5.4	Mode hors-ligne . . . . .	23
3.5.5	Gestion des conflits . . . . .	25
<b>4</b>	<b>Exploitation et préparation . . . . .</b>	<b>29</b>
<b>4.1</b>	<b>Installation des librairies et drivers</b>	<b>29</b>
4.1.1	Création du pilote MySQL pour Qt5 . . . . .	29
<b>4.2</b>	<b>Installation et configuration de l'annuaire</b>	<b>31</b>
4.2.1	Serveurs . . . . .	31
4.2.2	Configuration des clients . . . . .	31
<b>4.3</b>	<b>Procédure de génération de la base de données</b>	<b>31</b>
4.3.1	Génération des tables SQL avec JMerise . . . . .	31
<b>4.4</b>	<b>Configuration du cluster</b>	<b>32</b>



# 1 — Présentation du projet

## 1.1 INTRODUCTION

La présentation du projet s'articule autour de plusieurs points :

- le calendrier du projet ;
- son objectif ;
- ses enjeux ;
- ses exigences.

## 1.2 CALENDRIER DU PROJET

Le calendrier du projet permet de suivre l'évolution du projet. C'est dans celui-ci qu'il faut indiquer les différentes phases du projet et les deadlines associées à ces phases.

### 1.2.1 PHASE DE RÉALISATION DU PROJET

Cette phase consiste à répondre au projet décrit par le cahier des charges. Les documents suivants en sont issus :

- code source ;
- documentation d'architecture (DAT, PTI, ...) et d'exploitation ;
- droits de propriété.

## 1.3 OBJECTIF DU PROJET

L'objectif est de fournir la solution logicielle de type Transport Management System (TMS) attendue, et sa documentation associée.

## 1.4 ENJEUX DU PROJET

Garmir Khatch espère améliorer la qualité de ses services grâce à l'assistance du TMS et de l'automatisation de certaines de ses tâches. Le TMS lui permettra non seulement de réduire les coûts relatifs à la logistique mais également de maintenir son professionnalisme, améliorant ainsi son image de marque par preuve de son efficacité et de son efficience sur le terrain.

## 1.5 EXIGENCES DU PROJET

Cette section précise les services attendus et les exigences du projet.

### 1.5.1 LA LOCALISATION

La suite logicielle doit proposer des versions traduites en plusieurs langues avec des alphabets et des sens de lecture différents. À minima, les traductions dans les langues suivantes sont fournies :

1. anglais ;
2. français ;
3. espagnol ;
4. arabe.

### 1.5.2 L'INTERFACE UTILISATEUR

La solution fournit au minimum, et pour chacun des groupes utilisateurs ci dessous, une interface propre permettant de réaliser les actions qui leur sont liées.

### 1.5.3 LES CAS D'UTILISATIONS

Le TMS devra permettre :

1. la gestion des niveaux de sécurité (cryptographie) ;
2. la gestion des droits d'accès ;
3. la gestion des sauvegardes et des préférences ;
4. la gestion de la synchronisation ;
5. la gestion des tableaux de bord ;
6. la gestion des statistiques ;
7. la gestion des réquisitions & waybills/delivery notes ;
8. la gestion des chauffeurs ;
9. la gestion des véhicules ;
10. la gestion des prestataires.

### 1.5.4 LES CONTRAINTES

Le présent document tiens compte des contraintes suivantes :

**Contrainte 1.5.1 — Contrainte de compatibilité.** La suite logicielle doit impérativement être compatible avec les matériels utilisés, les systèmes d'exploitation et, au minimum, les versions de logiciels.

**Contrainte 1.5.2 — Contrainte de coûts et de moyens de communications.** Il faut limiter les coûts et se plier aux préférences sur les moyens de communications définies dans le cahier des charges.

**Contrainte 1.5.3 — Contrainte de fonctionnalité.** La suite logicielle fournit un outil intégré permettant de numériser les documents, via une interface graphique claire permettant de fixer facilement les paramètres de numérisation.

**Contrainte 1.5.4 — Contrainte de délai.** AMU FSI 2014 s'engage à fournir des garanties contractuelles de délais, et à s'y tenir. Les éventuelles pénalités de retard sont fixées d'un accord commun avec Garmir Khatch.

**Contrainte 1.5.5 — Contrainte légale.** Les contraintes légales de chaque pays s'appliquant à Garmir Khatch lors de ses interventions, la solution peut s'adapter aux normes en vigueur.

**Contrainte 1.5.6 — Contrainte réglementaire.** Garmir Khatch a une image et une éthique mondialement connue découlant de ses activités. Il conviendra de la prendre en compte lors de la réalisation de la solution.



## 2 — Architecture fonctionnelle

### 2.1 ARCHITECTURE FONCTIONNELLE

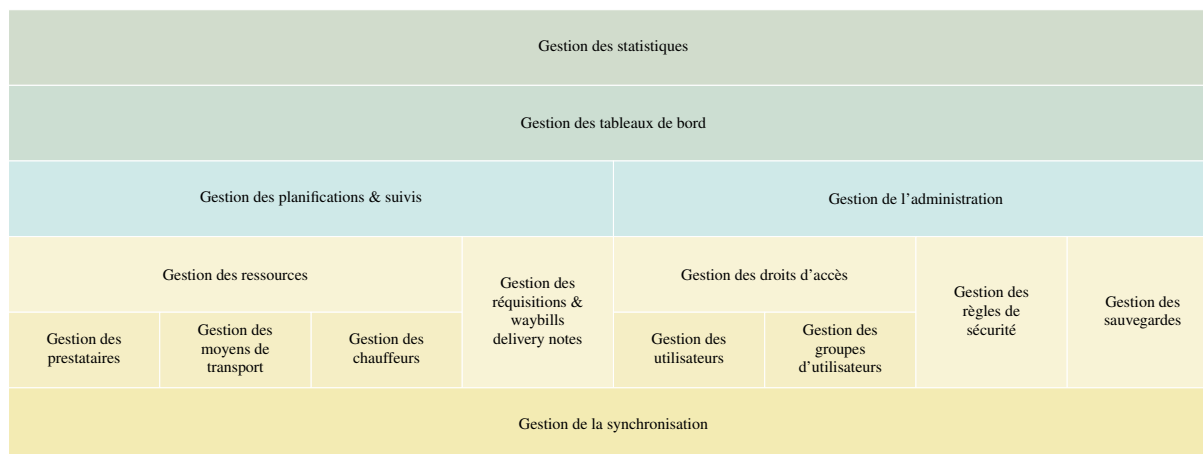


FIGURE 2.1 – Architecture fonctionnelle

### 2.2 ARCHITECTURE PHYSIQUE

La solution est prévue pour fonctionner sur l'architecture générale présentée dans la figure 2.2. Comme indiqué sur celle-ci, les utilisateurs sont en mesure d'interagir avec l'application par l'intermédiaire de différents matériels que sont ordinateurs portables, smart-phones et téléphones satellitaires. Les informations ainsi traitées sont mises en commun sur un serveur dit « local » à un lieu d'intervention. Si nécessaire, une synchronisation des informations peut avoir lieu entre le serveur « local » et le serveur « central » du siège.

### 2.3 FONCTIONS DE SERVICE ET DE CONTRAINTE

Cette section précise les services fonctionnels mis à disposition des utilisateurs et contraintes respectées.

#### 2.3.1 LA LOCALISATION

La suite logicielle propose des versions traduites en plusieurs langues, y compris avec des alphabets différents. Au minimum, les traductions dans les langues suivantes sont fournies :

- anglais ;
- arabe ;
- espagnol ;
- français.

En outre, la suite logicielle fournit la possibilité de passer *aisément* d'une langue à l'autre sans nécessiter de redémarrage.

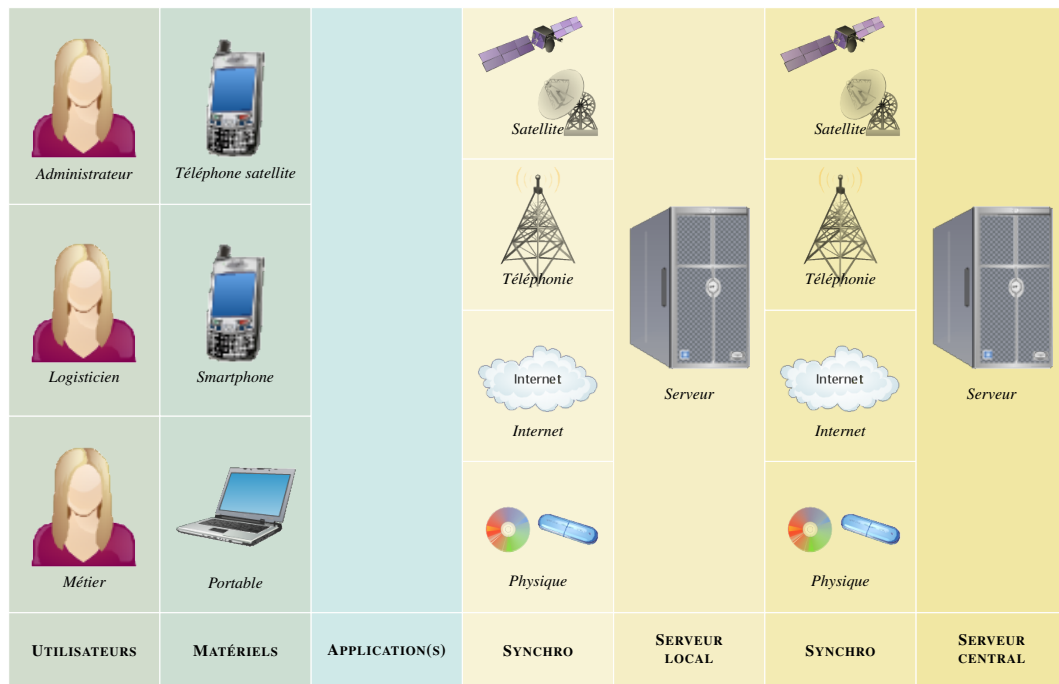


FIGURE 2.2 – Architecture générale

### 2.3.2 LES CAS D'UTILISATIONS

Cette section présente le diagramme des cas d'utilisation et précise pour chacun d'eux l'ensemble des actions possibles.

#### GESTION DES NIVEAUX DE SÉCURITÉ (CRYPTOGRAPHIE)

La suite logicielle fournit une interface permettant d'ajouter, supprimer et définir les règles de sécurités appliquées aux communications sur un lieu d'intervention, c'est à dire, les algorithmes de chiffrement utilisés lors des échanges de données sur le réseau.

Initialement, les algorithmes suivants sont fournis :

- primitives d'échange de clé :
  - NULL : i.e. aucun,
  - RSA : conformément à la RFC 3447 ;
- primitives de chiffrement :
  - NULL : i.e. aucun,
  - AES : conformément à la RFC 3394 ;
- primitives de hachage :
  - NULL : i.e. aucun,
  - MD5 : conformément à la RFC 1321,
  - SHA : conformément à la RFC 3174 ;

#### GESTION DES UTILISATEURS ET DES DROITS D'ACCÈS

La suite logicielle permet de gérer :

1. utilisateurs ;
2. groupes d'utilisateurs.

Un utilisateur peut appartenir à un ou plusieurs groupes d'utilisateurs.

La suite logicielle fournit à minima un groupe d'utilisateur appelé *administrateur* capable de :

1. ajouter, modifier et/ou supprimer utilisateurs et groupes d'utilisateurs ;
2. affecter des droits d'accès aux utilisateurs et groupes d'utilisateurs ;



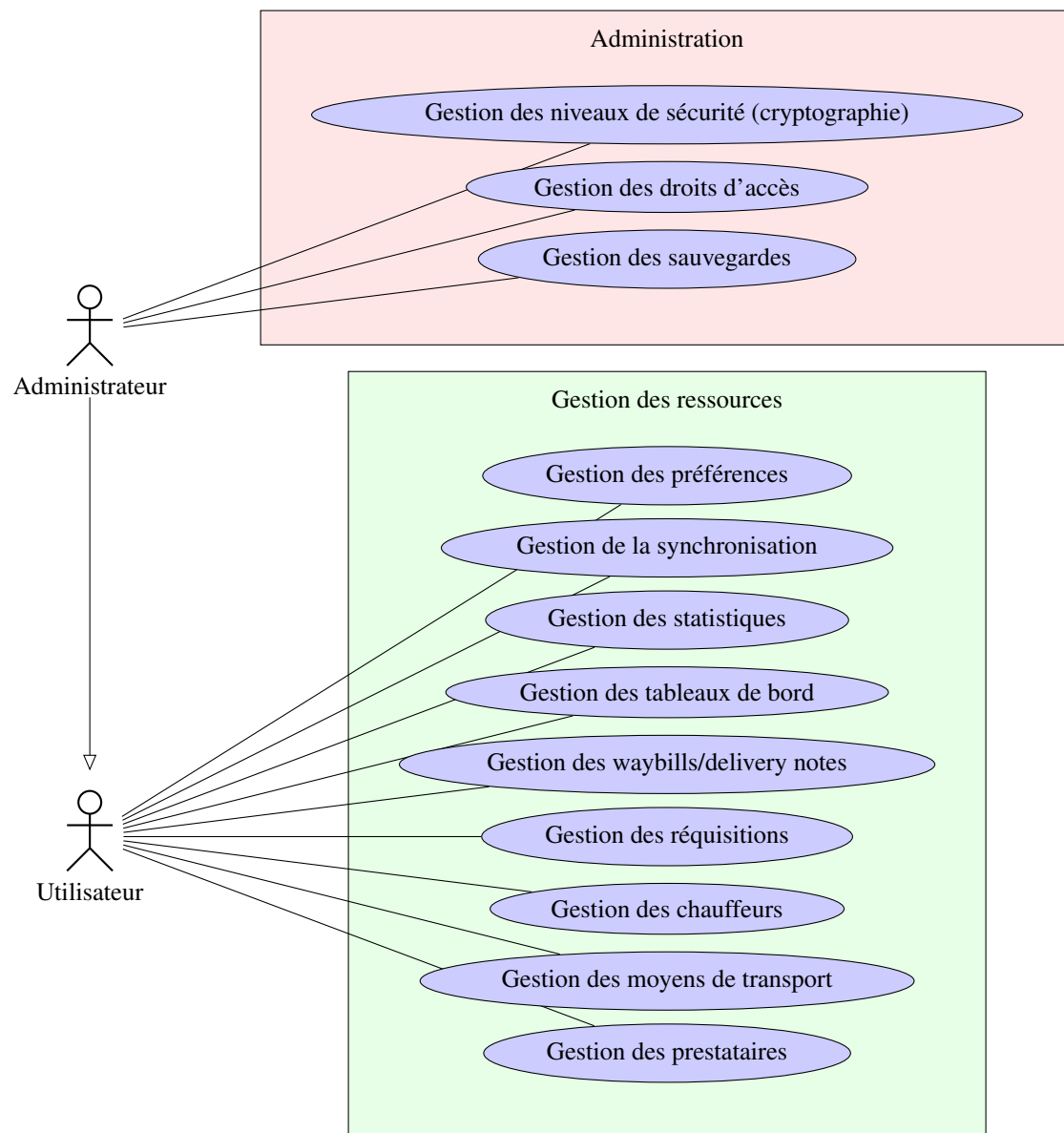


FIGURE 2.3 – Diagramme des cas d'utilisation

3. ranger les utilisateurs dans des groupes d'utilisateurs.

Les droits d'accès possibles sont les suivants :

1. autorisé ;
2. indéfini ;
3. interdit ;

Les conflits de droits résultant sont gérés par les 2 priorités suivantes :

1. les droits d'accès individuels sont prioritaires sur ceux inhérents aux groupes ;
2. les interdictions sont prioritaires sur les autorisations ;

#### **GESTION DES SAUVEGARDES**

La suite logicielle permet d'effectuer des sauvegardes externalisées. Les moyens techniques mis en place à cette fin sont décrits dans la section 3.4.

Ce système permet les manipulations suivantes :

1. lancer une sauvegarde (aussi bien distante que locale) ;
2. charger une sauvegarde existante (qu'elle soit distante ou locale) ;
3. ajouter, modifier, supprimer, importer et exporter une *configuration de sauvegarde*.

Une *configuration de sauvegarde* rassemble les paramètres suivants :

1. la liste exhaustive des données sauvegardées ;
2. la planification (date, heure, minute, ...) et ses répétitions (une seule fois, tous les jours - ou seulement certains, toutes les semaines, tous les mois, ...) ;
3. le support de réception qu'il soit distant ou local (serveur, client, baie de disque, disque dur, disque optique, clé USB, ...).

#### **GESTION DES PRÉFÉRENCES**

La solution est conçue de sorte à permettre l'intégration d'un module permettant aux utilisateurs de garder en mémoire leurs préférences, comme la langue utilisée. Néanmoins, ce module n'est pas fourni, mais pourra faire l'objet d'une mise à jour ultérieure.

#### **GESTION DE LA SYNCHRONISATION**

La suite logicielle est prévue pour fonctionner avec ou sans réseau. Afin de mettre à jour les données lorsque les moyens de communications sont rétablis, un système de synchronisation est mis en place. Les détails techniques de cette synchronisation sont présentés en section 3.5.

#### **GESTION DES TABLEAUX DE BORD**

La suite logicielle permet de générer des *tableaux de bord*, dont le contenu peut être défini, chargé et enregistré dans une *configuration de tableau de bord*.

#### **GESTION DES STATISTIQUES**

La suite logicielle est conçue de sorte à permettre l'intégration d'un module capable de produire un ensemble de statistiques à partir des informations enregistrées. Néanmoins, ce module n'est pas fourni par la solution proposée, mais pourra faire l'objet d'une mise à jour ultérieure.

#### **GESTION DES DONNÉES**

La suite logicielle permet de numériser des documents papier. Ces derniers peuvent être ensuite enregistrés, affichés, supprimés, importés, exportés et archivés.

En outre, la suite logicielle permet d'ajouter, modifier, et supprimer les informations qu'elle aura à traiter, et les liens qui les relient. Initialement, les informations suivantes sont déjà définies et peuvent être traitées :

1. planifications ;
2. réquisitions ;

3. waybills/delivery notes ;
4. prestataires ;
5. chauffeurs ;
6. véhicules.

#### **PLANIFICATIONS**

Une planification rassemble les informations suivantes :

1. planifications ;
2. réquisitions ;
3. waybills / delivery notes ;
4. prestataires ;
5. chauffeurs ;
6. véhicules.

#### **RÉQUISITIONS**

Une Réquisition rassemble les informations suivantes :

1. code du pays, et numéro unique de réquisition ;
2. origine, destination, date de la réquisition, date de livraison souhaitée ;
3. moyen de transport (air, mer, route, ...) ;
4. pour chaque article :
  - (a) code de l'article,
  - (b) numéro de compte,
  - (c) description de l'article,
  - (d) quantité,
  - (e) unité ;
  - (f) limite budgétaire (prix unitaire, prix total) ;
5. les accords :
  - (a) nom, et date de signature du demandeur,
  - (b) nom, et date de signature du chef de projet (en charge du budget),
  - (c) nom, et date de signature de l'agent financier,
  - (d) nom, et date de signature de la logistique ;
6. les détails de l'envoi :
  - (a) adresse du consignataire (adresse complète, téléphone, mobile, fax, nom du contact & adresse e-mail),
  - (b) adresse de livraison (si différente - adresse complète, téléphone, mobile, fax, nom du contact & adresse e-mail),
  - (c) agent de dédouanement (le cas échéant - adresse complète, tel, mobile, fax, contact & email),
  - (d) marquage de l'envoi (ex : no. CTN, logo Croix-Rouge, "en transit", "donation de..."),
  - (e) demandes spéciales ou remarques (ex : instructions d'emballage, incoterms, documents joints, etc.) ;

#### **WAYBILLS/DELIVERY NOTES**

Un Waybills/delivery notes rassemble les informations suivantes :

1. code du pays, et numéro unique de Waybills/delivery notes ;
2. date de la demande ;
3. type de transport( voix aérienne, maritime, routière ou ferroviaire) ;
4. code du pays et numéro de la réquisition associée à ce Waybills/delivery notes ;
5. numéro du véhicule utilisé ;
6. numéro du contrat.

**PRESTATAIRES**

Un prestataire rassemble les informations suivantes :

1. numéro d'identification dans la base ;
2. nom du prestataire ;
3. patente du prestataire.

**CHAUFFEURS**

Un chauffeur rassemble les informations suivantes :

1. numéro d'identification ;
2. numéro de la personne dont la fonction est chauffeur ;
3. type(s) de permis que détient ce chauffeur.

**VÉHICULES**

Un véhicule rassemble les informations suivantes :

1. numéro d'identification ;
2. poids maximum de charge ;
3. volume maximum de charge ;
4. autonomie maximum ;
5. numéro du type de véhicule ;
6. numéro du prestataire auquel le véhicule appartient.



## 3 — Architecture technique

### 3.1 APPLICATION

L'IHM est réalisé en langage C++ à l'aide du framework Qt.

**Notation 3.1.** *L'application doit être réalisée en client lourd. Il a été exclu l'éventualité d'utiliser un serveur en local (sur les postes client) c'est pourquoi les langages comme PHP ont été exclus. En outre le choix a été fait de préférer un langage qui soit :*

1. *orienté objet (ce qui exclu les langages comme C) ;*
2. *multiplateforme ;*
3. *suffisamment vieux pour avoir été durement éprouvé (ce qui exclu les langages comme Ruby) ;*
4. *facile à prendre en main (d'où le choix du framework Qt) ;*
5. *performant et peu coûteux en mémoire (ce qui exclu les langages comme Java) ;*
6. *compatible avec les divers autres éléments utilisés (SGBDR, LDAP, XML, ...) ;*

### 3.2 AUTHENTIFICATION ET CONTRÔLE D'ACCÈS

Dans le cadre de la mission confiée par Garmir Katch nous avons pour tâche de mettre en place un module d'authentification et de contrôle d'accès à intégrer à la solution finale. Pour cela nous nous basons sur une base de données d'annuaire afin de regrouper les informations sur les utilisateurs de la société (informations personnelles, droits, groupes).

Un annuaire est prévu pour être plus sollicité en lecture qu'en écriture. Cela signifie qu'un annuaire est conçu pour être plus souvent consulté que mis à jour. Dans notre cas les utilisateurs auront d'avantage besoin d'accéder à la base de données en lecture plutôt que pour y modifier des champs (d'où le choix d'une base de données d'annuaire plutôt qu'une relationnelle).

Cet annuaire est géré par le protocole LDAP (Lightweight Directory Acces Protocol).

#### 3.2.1 POURQUOI LDAP ?

Le protocole LDAP présente plusieurs avantages liés aux exigences de l'utilisation d'un annuaire plutôt qu'une base de données relationnelle :

- Les annuaires doivent être compacts et reposer sur un protocole réseau léger : comme son nom l'indique c'est un protocole allégé pour accéder aux annuaires.
- Un serveur d'annuaire doit comporter des mécanismes permettant de coopérer, c'est-à-dire d'étendre la recherche sur des serveurs tiers si jamais aucun enregistrement n'est trouvé : le protocole LDAP dispose d'un mécanisme de référencement (referral).
- Un annuaire doit être capable de gérer l'authentification des utilisateurs ainsi que les droits de ceux-ci pour la consultation ou la modification de données : LDAP fournit une méthode d'authentification (bind).
- Possibilité de sécuriser la connexion en TLS via un port spécifique utilisant le protocole LDAPS.
- Répartition possible de l'annuaire sur plusieurs points d'accès via un service de réplication fourni par LDAP.
- Ce même service permet une haute disponibilité des serveurs.

Ces fonctionnalités nous permettent de répondre à la demande d'optimisation des flux (protocole léger), à la sécurité (disponibilité, confidentialité)

L'intérêt principal de LDAP est la normalisation de l'authentification. Il est facile de programmer un module d'authentification utilisant LDAP à partir d'un langage possédant une API LDAP.

### 3.2.2 STRUCTURE DE L'ANNUAIRE

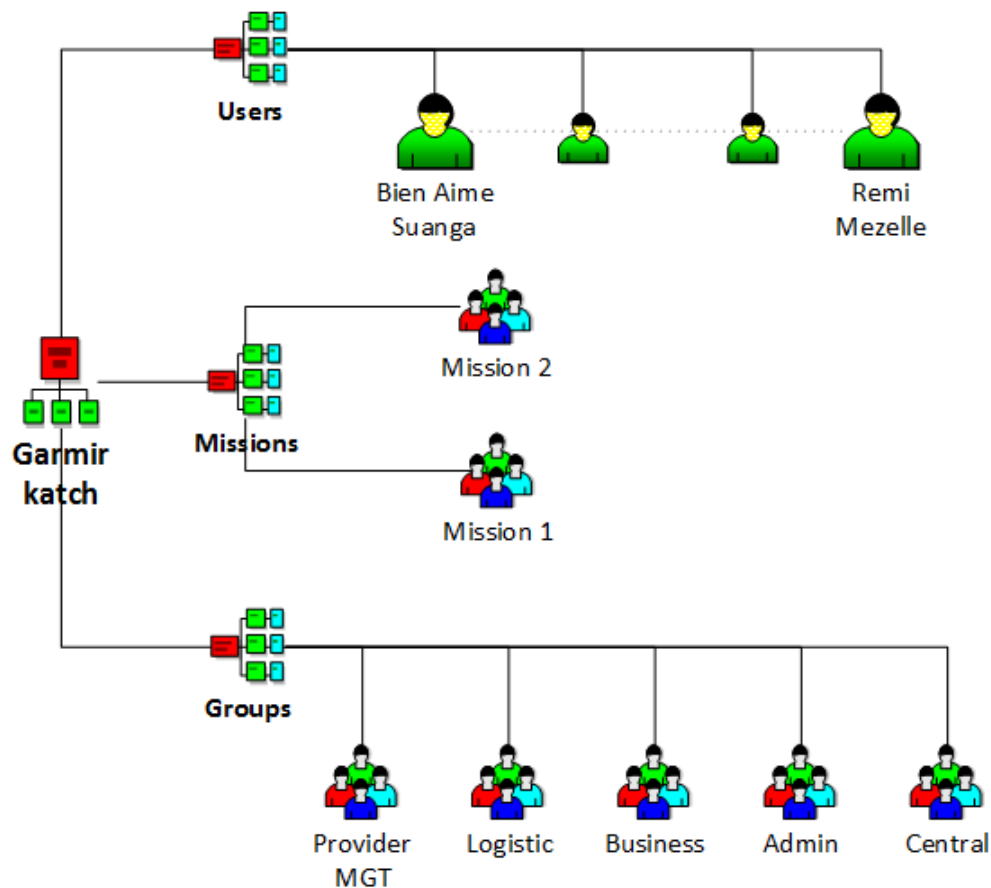


FIGURE 3.1 – Schéma de l'annuaire LDAP mis en place

L'annuaire est organisé en trois parties :

- la première recense toutes les personnes de la société et leurs droits,
- la deuxième concerne les différents groupes d'utilisateurs et les droits de chaque groupe,
- la troisième permet de rassembler les personnes présentes sur chaque mission.

Chaque utilisateur dispose d'un identifiant unique (uid) et d'un mot de passe (userPassword) et aussi différents champs représentant ses droits. Le mot de passe d'un utilisateur est stocké en haché SSHA. Enfin, il est possible de se connecter anonymement mais l'accès aux données de l'annuaire sera restreint.

Chaque groupe se voit attribuer des droits spécifiques ce qui peut entraîner des conflits. Par défaut nous avons fait primer les droits des utilisateurs avant ceux des groupes.

Grâce à l'unité d'organisation Missions on pourra déterminer rapidement quelles sont les personnes présentes sur chaque mission en cours.

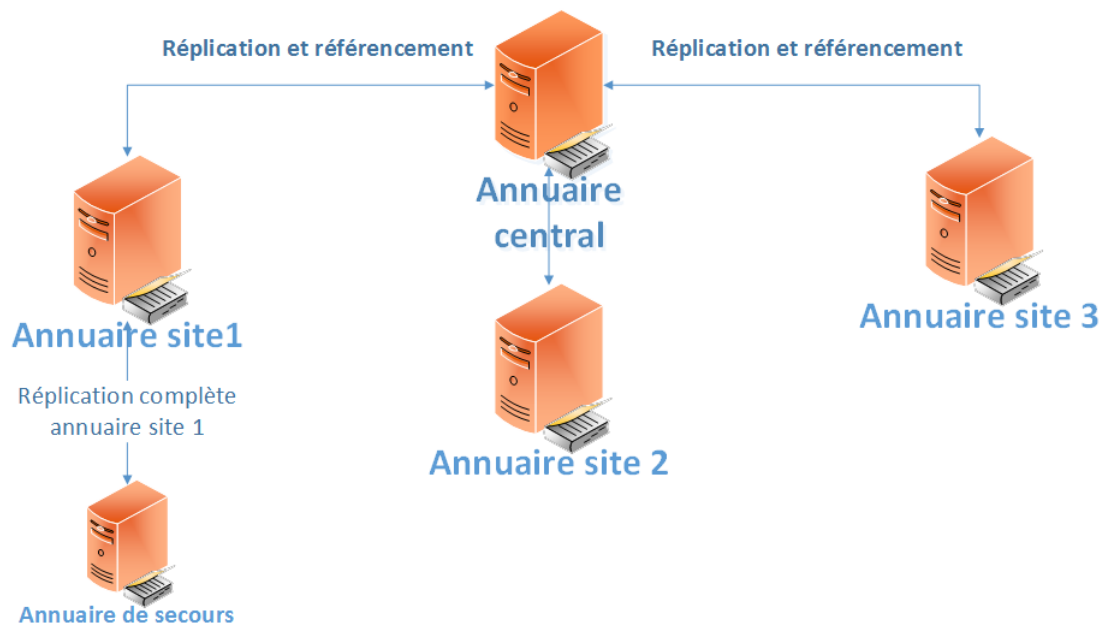


FIGURE 3.2 – Architecture globale

### 3.2.3 ARCHITECTURE

L'annuaire global, situé au siège de la société, rassemble toutes les informations et les annuaires locaux ne contiennent que les informations sur les utilisateurs présents sur chaque site. Ainsi, des mécanismes de référencement et de réplication entre les serveurs locaux et le serveur central pourront être mis en place. De plus, un serveur de secours pourra assurer une haute disponibilité de l'annuaire local avec une réplication périodique des données de ce dernier.

### 3.2.4 QUAND S'AUTENTIFIER ?

Tout d'abord, il est nécessaire de définir le cadre dans lequel le module sera utilisé : un utilisateur pourra effectuer toutes les modifications sur sa base de données interne, ce n'est que lors de la synchronisation des données que l'utilisateur devra s'authentifier.

L'authentification permettra de déterminer quelles sont les modifications que l'utilisateur a le droit d'apporter à la base de données. Ainsi, aucun annuaire ne sera installé sur les postes clients, évitant toute tentative frauduleuse d'accorder des droits différents à une personne.

Les serveurs mis en place prennent en charge le protocole non standard « LDAPS » (LDAP over SSL). Ce protocole utilise le port 10636 (ou 636). Le protocole LDAPS diffère du LDAP sur deux points :

- à la connexion, le client et le serveur établissent une connexion TLS avant que n'importe quelle autre commande LDAP ne soit envoyée,
- la connexion LDAPS doit être fermée lors de la clôture de TL

Ce protocole permet de ne pas faire transiter les mots de passe en clair sur le réseau et ainsi garantir leur confidentialité.

### 3.2.5 DROITS UTILISATEURS

Les droits des utilisateurs sont stockés dans des attributs LDAP ajoutés aux informations standards. Chaque utilisateur dispose de droits qui lui sont propres, mais aussi de droits liés au(x) groupe(s) auquel(s) il appartient. Voici la liste des champs créés pour définir les droits d'une personne :

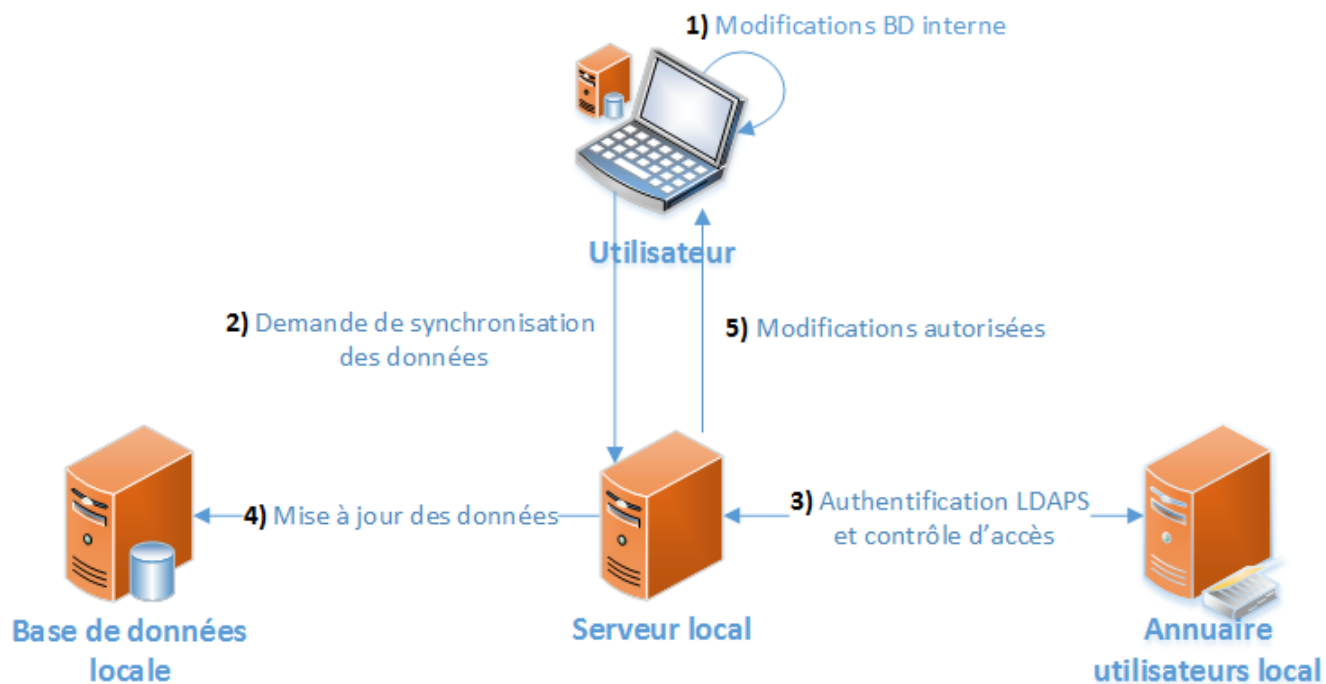


FIGURE 3.3 – Authentification pour la synchronisation

- aFieldRead : Autorisation de lecture d'un champ d'une table
- aFieldWrite : Autorisation d'écriture d'un champ d'une table
- aTableAdd : Autorisation d'ajout d'une ligne d'une table
- aTableDel : Autorisation en suppression d'une ligne d'une table
- rFieldRead : Refus en lecture d'un champ d'une table
- rFieldWrite : Refus en écriture d'un champ d'une table
- rTableAdd : Refus d'ajout d'une ligne d'une table
- rTableDel : Refus en suppression d'une ligne d'une table

Ces attributs permettent de définir soit l'ensemble des permissions, soit l'ensemble des refus : si un attribut autorise la lecture pour un champ cela implique que la lecture des autres champs sera interdite.

Ci-dessous deux exemples d'attribut et leur valeurs :

- Autorisation lecture des champs RequisitionID, VehicleID et TransportMean table Waybill :  
aFieldRead : Waybill(RequisitionID, VehicleID, TransportMean)
- Refus d'ajout d'un champ dans la table Requisition : rTableAdd : Requisition

On remarque pour le premier exemple qu'il ne sera pas nécessaire de remplir le champ rFieldRead pour la table Waybill car il est sous-entendu que l'accès en lecture aux autres champs de la table Waybill sera refusé.



### 3.2.6 GESTION DES CONFLITS

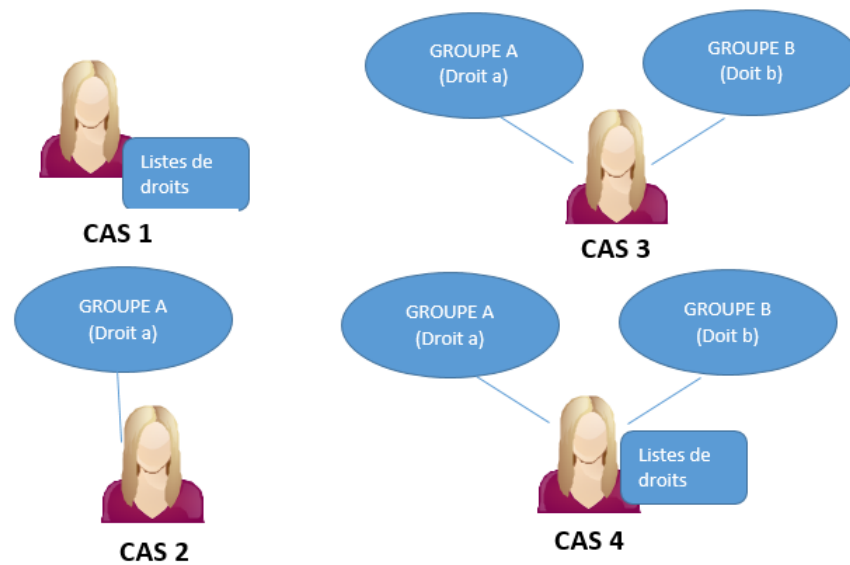


FIGURE 3.4 – Quelques exemples de demande de permission

Voici comment fonctionne notre outil lors des différents demandes des permissions

- **CAS 1** : L'utilisateur aura directement ses permissions
- **CAS 2** : L'utilisateur aura les permissions du groupe
- **CAS 3** : Priorité aux permissions autorisées entre les groupes
- **CAS 4** : Priorité aux permissions directement liées à l'utilisateur

### 3.3 DONNÉES

#### 3.3.1 PROBLÉMATIQUE

Ce chapitre traite du stockage et du formatage des données sur les différents postes, et dans les différentes situations d'utilisation précédemment décrites.

Cette problématique est divisée selon trois points clés :

- le stockage ;
- le transfert ;
- l'importation et l'exportation.

#### 3.3.2 LE STOCKAGE

Cette section développe en détail les différents type de stockage au sein de l'architecture clients/serveurs.

##### CÔTÉ SERVEUR

Le stockage des données côté serveur se fait grâce au SGDBR MySQL. Cette solution a été retenue pour plusieurs raisons :

- solution traitant rapidement les données ;
- solution compatible avec la plupart des systèmes d'exploitation (Windows, GNU/Linux, MacOS, ...);
- solution facile à utiliser ;
- solution pouvant facilement s'interfacer à l'aide d'API diverses ;
- solution maîtrisée par Garmir Khatch.

##### CÔTÉ CLIENT

Le stockage des données côté client se fait soit via deux représentations d'une base de données locale : une tournant sur MySQL et une autre utilisant des fichiers au format XML. S'il est possible d'effectuer la synchronisation avec la base de donnée centrale et que le périphérique le permet, la première représentation est choisi par défaut. Si la synchronisation au serveur central est indisponible, le stockage des données est fait par la deuxième représentation de la base au format XML.

Le choix de l'utilisation des fichiers XML est dû aux raisons suivantes :

- format universel, compatible à l'import/export avec la plupart des SGDBR (dont MySQL) ;
- taux de compression assez important sur ce format, ce qui peut favoriser le transfert des données dans des situations où la connexion est limitée ;
- exploitable sur les périphériques Android.

##### GÉNÉRATION DE LA BASE DE DONNÉES

Le bon fonctionnement de la base de données s'appuie sur les fichiers suivants :

1. *GkTms.ini* ;
2. *GkTms.mcd* ;
3. *GkTms.sql*.

##### GKTMS.INI

Ce fichier définit les éléments de configuration indispensables à la connexion avec la base de données. Il se présente sous la forme suivante :

```

1 dbDriver=QMYSQL
2 dbHostName=localhost
3 dbUserName=root
4 dbPassword=
5 dbName=GkLogistic

```

GkTms.ini

1. *Driver* est le driver Qt correspondant à la base de données considérée (dans le cas présent MySQL) ;
2. *HostName* est l'adresse URL de la base de données (ici localhost) ;
3. *UserName* est le nom d'utilisateur (ici root) ;
4. *Password* est le mot de passe d'utilisateur (ici aucun) ;
5. *Name* est le nom que l'on souhaite donner à la base de donnée considérée (ici GkLogistic). Ce dernier est optionnel, et ne présente de réel intérêt que dans le cas où plusieurs bases de données distinctes seraient utilisées.

Via Qt, l'utilisation de ces informations de connexion se fait, par exemple, de la façon suivante :

#### **GkTms.MCD**

Ce fichier définit le MCD (Modèle Conceptuel des Données) éditable par l'intermédiaire de l'outil de développement JMerise. Ce dernier permet, entre autres choses, de générer le fichier de script *GkTms.sql*.

#### **GkTms.SQL**

Ce fichier définit l'architecture relationnelle (en SQL) de la base de données. Ce script est exécuté quand la base de données n'existe pas, et la crée.

#### **GkTms.XML**

Ce fichier définit l'architecture relationnelle (en XML) de la base de données. Ce script est exécuté quand la base de données n'existe pas, et la crée.

### **3.3.3 L'IMPORT/EXPORT DE DONNÉES**

L'import et l'export de données se fait directement via une fonctionnalité de l'outil côté client, qui permet de sélectionner les données à importer dans le contexte spécifique, ainsi que les données à exporter vers le serveur central, toujours selon ce contexte. Comme décrit dans la partie traitant de ce sujet, l'import/export de données est réalisé selon différents supports (réseau, support physique) à la discrétion de l'exploitant.

### 3.4 SAUVEGARDE

#### 3.4.1 PROBLÉMATIQUE

La suite logicielle permet de synchroniser des données, à la fois entre postes clients et serveur local, mais également entre serveur local et serveur central, et ce, via les différentes architectures ci dessous.

#### 3.4.2 CLUSTER

Une première solution utilisant la technologie du *cluster* peut être envisagée. Cette technique permet de créer un groupe logique de serveurs qui s'exécutent simultanément tout en donnant l'impression aux utilisateurs de ne constituer qu'un seul serveur. En considérant le matériel existant, et le fait que l'achat de serveur dédié pour ce cluster serait coûteux au client, une solution de *clustering* peut être effectuée grâce à de simples postes clients qu'il faut configurer comme des serveurs. Dans l'architecture proposée (voir figure 3.5), deux postes clients (configurés comme des serveurs maître/esclave) sont reliés directement par un câble ethernet, ce qui permet de dupliquer au fur et à mesure toutes les données. Le serveur maître transmet les données au serveur esclave, et en cas d'incident sur le serveur maître, c'est le serveur esclave (ou de secours) qui reprend la main.

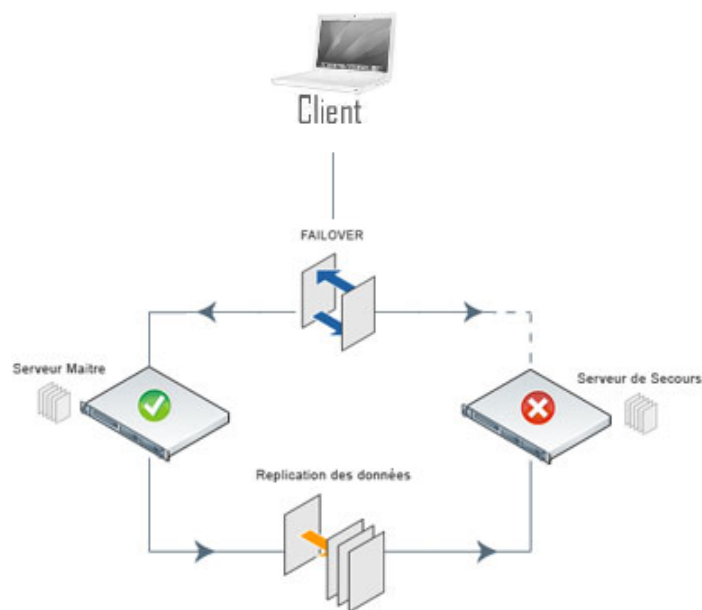


FIGURE 3.5 – Cluster, serveur maître et esclave directement connectés

Le serveur de secours est en *hot standby*, c'est-à-dire qu'il reste allumé en mode passif, attendant que des requêtes clients soient reçues (requêtes qui sont envoyées au maître tant que ce dernier est actif).

#### FAIL-OVER

Dans le cadre d'une architecture serveur haute disponibilité, le *fail-over* permet une reprise automatique inférieure à une minute sur le serveur de secours en cas de défaillance du serveur maître. Le *fail-over* est donc la capacité d'un équipement à basculer automatiquement vers un chemin réseau redondant ou en veille. Cette solution est préconisée sur les serveurs ou les architectures serveurs qui nécessitent une disponibilité permanente et un haut niveau de connectivité.

Il existe deux utilisations du protocole *fail-over* :

- mode statique : chaque client est configuré pour basculer sur une URI précise si la standard ne répond plus ;
- mode dynamique : le maître fournit dynamiquement l'URI de l'esclave aux clients qui peuvent la mettre à jour.

Le mode retenu est le mode statique, où chaque client a sa configuration et la conserve, aucune configuration spécifique n'est donc nécessaire pour le maître, et il faut simplement configurer sur l'esclave la connection avec le maître.

#### RÉPLICATION DES BASES DE DONNÉES EN TEMPS RÉEL

Les écritures sur le serveur maître sont répliquées en temps réel sur le serveur de secours, ne ralentissent pas le serveur maître et permettent de minimiser les pertes de données. La réplication se fait par recopie des journaux de transactions entre le maître et l'esclave.

Le mode de synchronisation choisi est le mode synchrone : une transaction émise par un client n'est validée que si l'écriture sur le maître ainsi que la synchronisation avec le serveur de secours sont validées. Ceci permet de s'assurer de la bonne duplication des données sur les deux serveurs.

#### LIMITATIONS

Seul un esclave à la fois peut être connecté au maître. Un maître hors ligne ne peut être réintroduit qu'à froid. Répliquer les données de l'esclave vers le maître une fois ce dernier remonté n'est pas automatique, cela nécessite une intervention manuelle.

#### LOGICIEL

Apache Active MQ est un agent de messages open source, écrit en Java et associé avec un client Java Message Service. Cette suite prend en charge divers protocoles de transport et précisément le protocole de fail-over qui permet de configurer les clients afin qu'ils basculent sur le serveur esclave en cas de non réponse du maître. Il permet aussi de charger les différentes configurations (clients, maître et esclave) conservées dans des fichiers XML.

### 3.4.3 SAUVEGARDE EN DUR

Une deuxième solution peut être mise en œuvre de manière complémentaire au *cluster* afin de mieux respecter la contrainte concernant l'intégrité des données. Un support mobile de stockage est utilisé afin de directement sauvegarder le contenu des serveurs. Le support peut être un disque dur, une clé USB, une carte mémoire... Cette sauvegarde devra être effectuée toutes les six heures afin de permettre de remonter les données en cas de panne des deux serveurs.

### 3.5 SYNCHRONISATION

#### 3.5.1 INTRODUCTION

L'objectif de cette section est de décrire les méthodes proposées afin de répondre au problème de synchronisation. Les échanges de données pouvant être limités (pas de réseau, liaison satellitaire uniquement), il convient de pouvoir choisir précisément les éléments que l'on souhaite synchroniser<sup>1</sup>.

Pour permettre la gestion de ces éléments, on introduit la notion de *priorité* ; à chaque catégorie d'éléments (e.g. planning des transports, liste des fournisseurs, informations sur les véhicules, ...) peut être associée une priorité de laquelle dépendra la synchronisation ou non. À partir de cette « hiérarchie d'importance », on associe des *profils de synchronisation* paramétrables qui - une fois activés - gère la synchronisation de façon transparente en fonction des choix de l'utilisateur. Pour résoudre les problèmes de connexion et assurer le fonctionnement de la suite logicielle indépendamment de la liaison réseau, deux modes sont prévus :

- connecté ;
- hors-ligne.

L'utilisation de ces deux modes est décrit dans la figure 3.6 : on utilise le mode connecté lorsque la liaison réseau est établie avec le serveur local ; et le mode hors-ligne quand il n'y a pas de liaison avec le serveur local. Dans les deux cas, nous faisons abstraction de la liaison entre le serveur local et le serveur central dans la mesure où l'utilisateur ne se synchronisera qu'avec le serveur local.

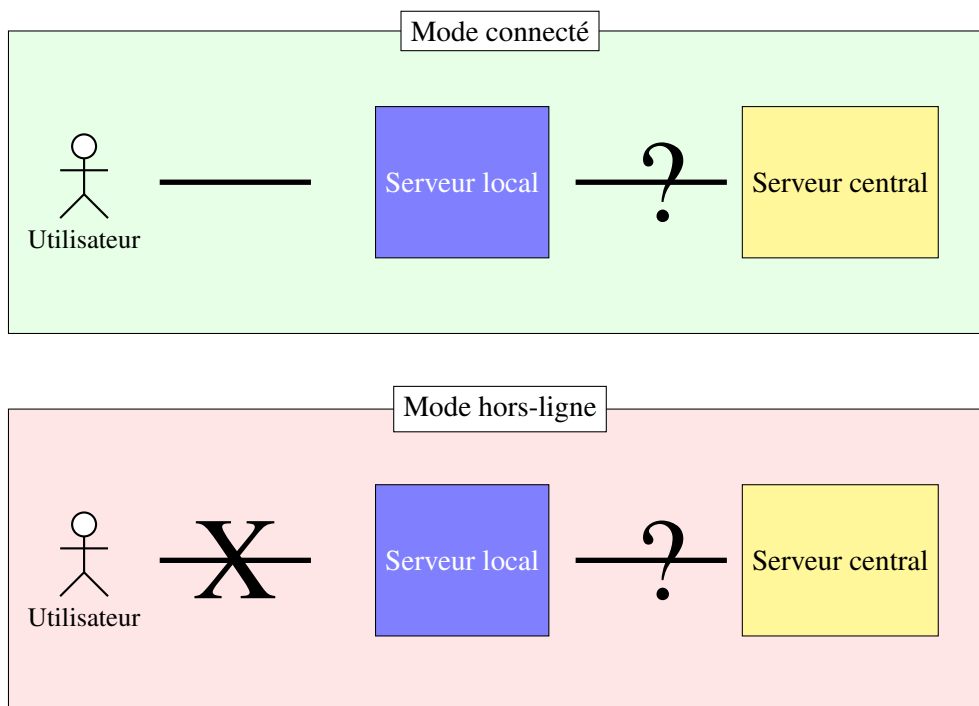


FIGURE 3.6 – Utilisation des modes connecté et hors-ligne

#### 3.5.2 GESTION DES PRIORITÉS & DES PROFILS DE SYNCHRONISATION

Une priorité permet à l'application de « choisir » ce qui transite sur le réseau. Cinq priorités sont définies (par ordre d'importance croissant) pour qualifier des éléments synchronisables :

1. Il est à noter que la synchronisation est bi-directionnelle : on reçoit les informations du serveur autant qu'on en envoie.

1. négligeable ;
2. secondaire ;
3. normal ;
4. important ;
5. crucial.

L'utilisateur peut gérer les priorités à synchroniser en les associant à un profil : lorsque l'utilisateur choisit un profil de synchronisation, seuls les éléments ayant une priorité incluse dans le profil sont synchronisés.

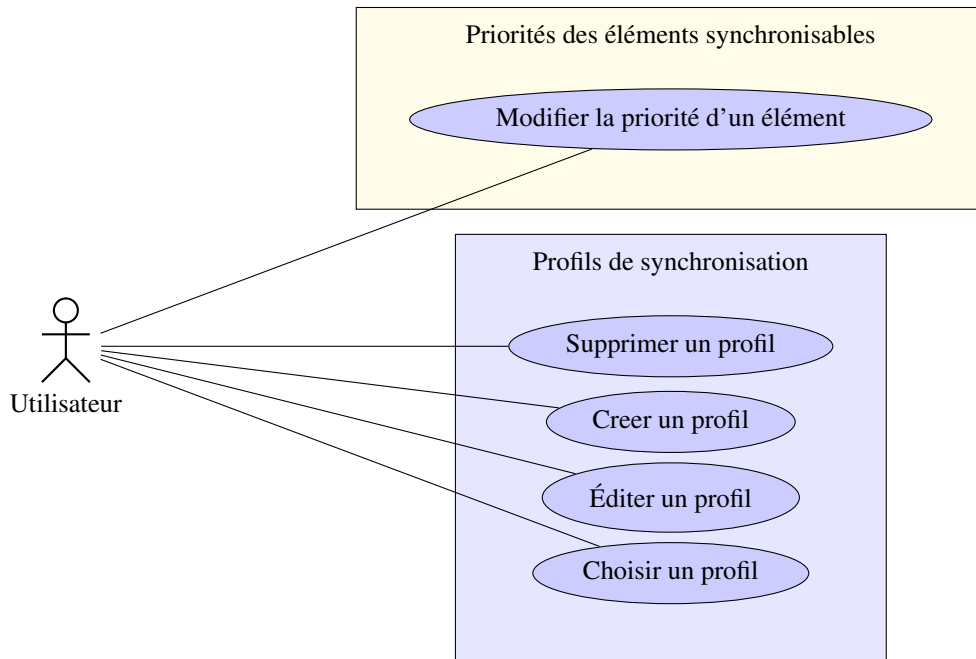


FIGURE 3.7 – Diagramme des cas d'utilisation pour la synchronisation

### 3.5.3 MODE CONNECTÉ

Dans ce mode, on suppose qu'il y a une liaison réseau entre l'utilisateur et le serveur local ; toutes les modifications sont effectuées en « temps réel<sup>2</sup> ». Pour ce mode, l'utilisateur doit :

1. se connecter ;
2. utiliser la suite logicielle ;
3. se déconnecter après utilisation.

L'utilisateur ne pourra effectuer que les modifications dont il a les *permissions*.

### 3.5.4 MODE HORS-LIGNE

Contrairement au mode connecté, le mode hors-ligne permet à l'utilisateur de modifier toutes les informations dont il dispose localement. Lorsqu'il souhaite synchroniser ses informations avec le serveur, il doit se connecter et c'est à ce moment là que le serveur vérifie que l'utilisateur n'outrepasse pas les droits qui lui sont accordés. Si c'est le cas, le serveur rejette les modifications locales.

2. Un écart de temps pourra être constaté dans le cas où le réseau n'offre qu'un faible débit.

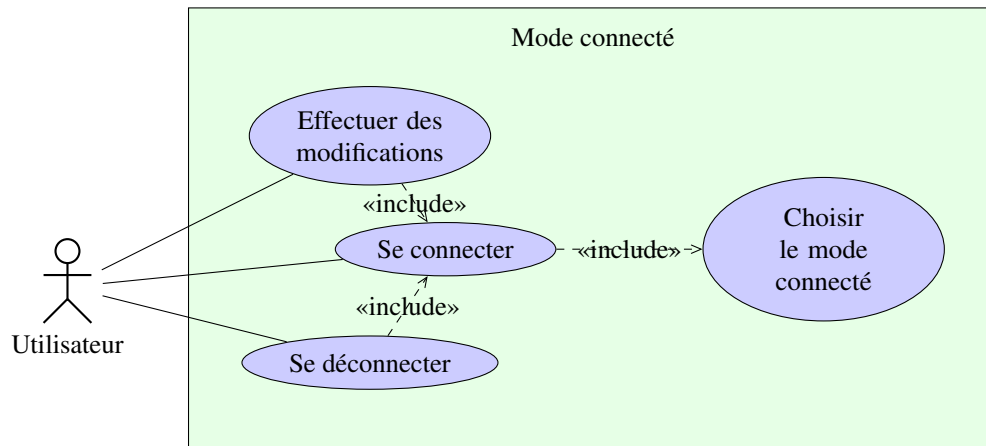


FIGURE 3.8 – Diagramme des cas d'utilisation pour le mode connecté

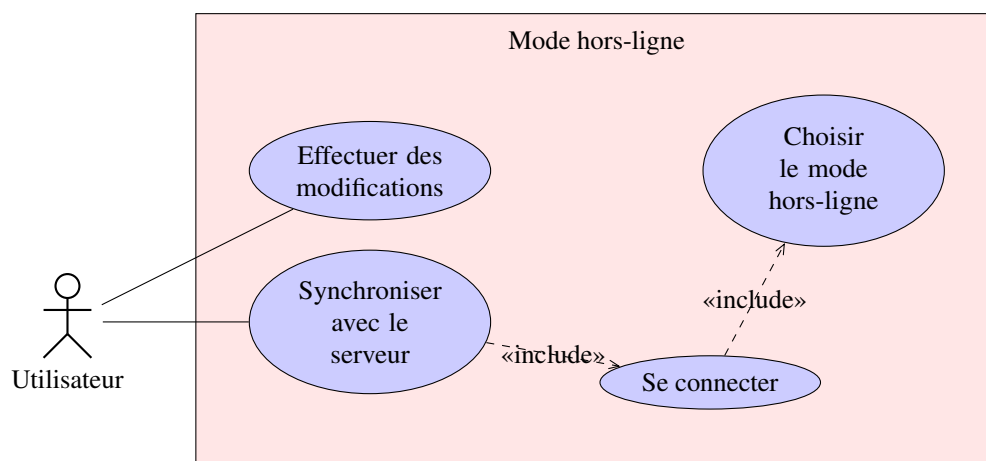


FIGURE 3.9 – Diagramme des cas d'utilisation pour le mode hors-ligne



### 3.5.5 GESTION DES CONFLITS

Durant la synchronisation, des conflits peuvent survenir au niveau du contenu des informations. Dans ce cas, l'utilisateur est averti du conflit et des informations qu'il touche et il lui revient le soin de les gérer en choisissant explicitement ce qui est correct et qui doit être enregistré sur le serveur.

Afin de savoir si la version des informations sur le serveur est plus récente (ou plus vieille) que la version locale, un horodatage est mis en place tel que :

- chaque modification locale est horodatée avec l'heure locale ;
- à la synchronisation, l'écart temporel entre l'heure locale et l'heure serveur est mesuré<sup>3</sup>.

Les activités "ajout d'une donnée" (Fig. 3.10), "modification d'une donnée" (Fig. 3.11), et "synchronisation" (Fig. 3.12) représentent les actions réalisées par le programme.

---

3. Par exemple, supposons que la machine locale retarde de deux heures : si une information a été modifiée à 17h (heure serveur) et qu'une modification est apportée à 16h le même jour en local, l'écart de temps entre les deux machines est mesuré lors de la synchronisation (i.e. deux heures) et permet de dater réellement la modification en local et ainsi déterminer laquelle succède à l'autre.

## AJOUT D'UNE DONNÉE

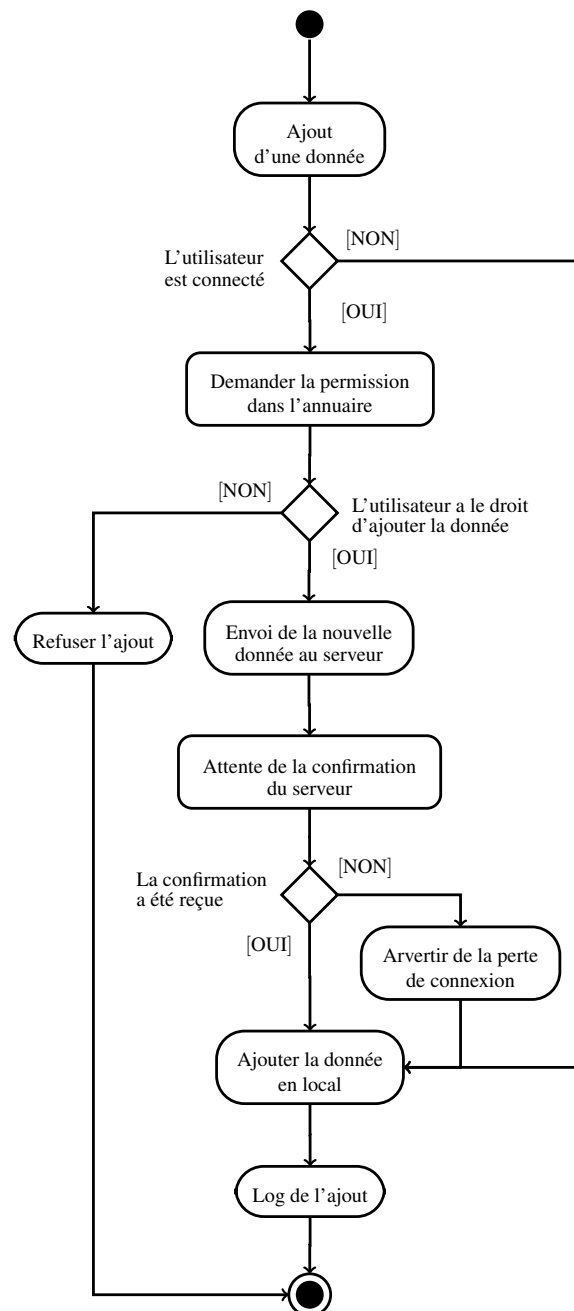


FIGURE 3.10 – Diagramme d'activité : Ajout d'une donnée

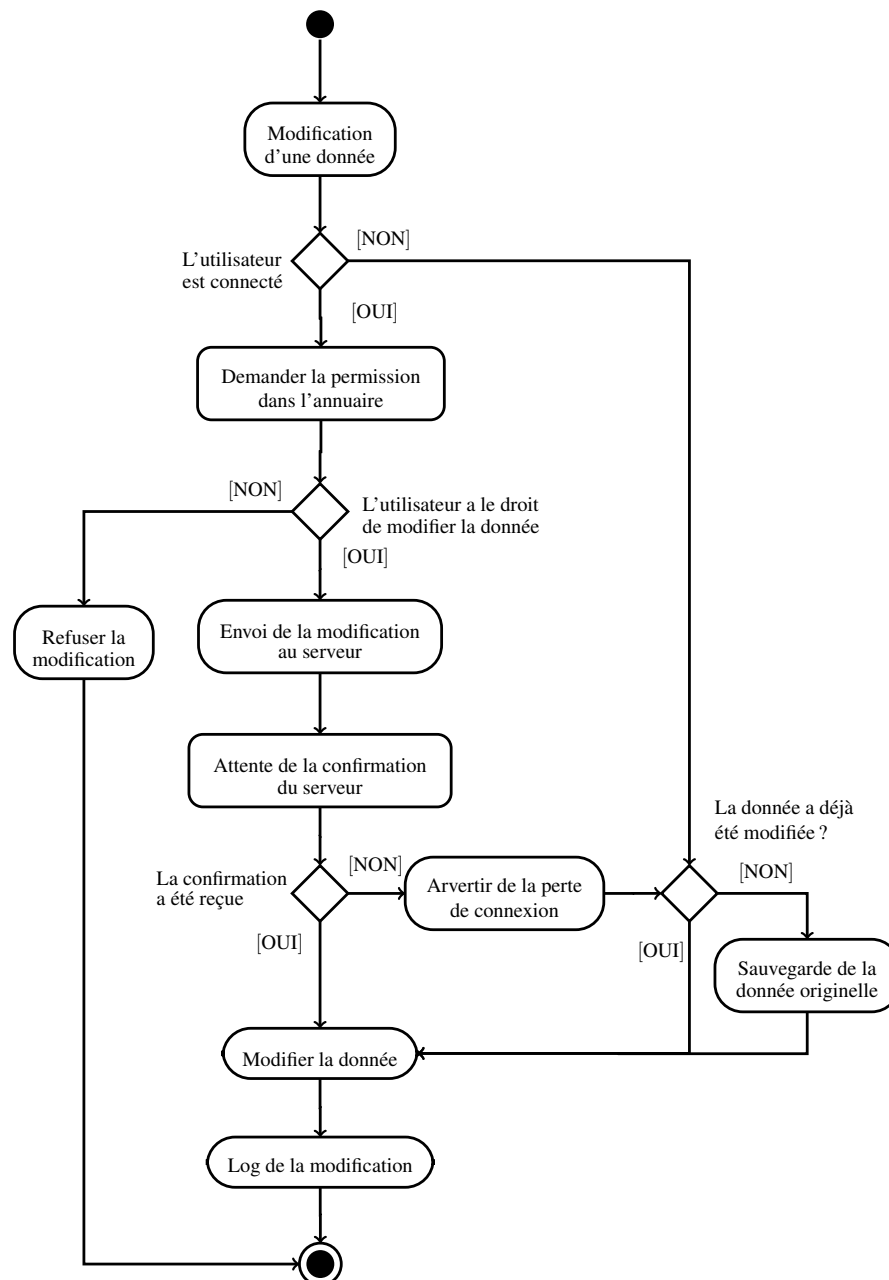
**MODIFICATION D'UNE DONNÉE**

FIGURE 3.11 – Diagramme d'activité : Modification d'une donnée

## SYNCHRONISATION DES DONNÉES

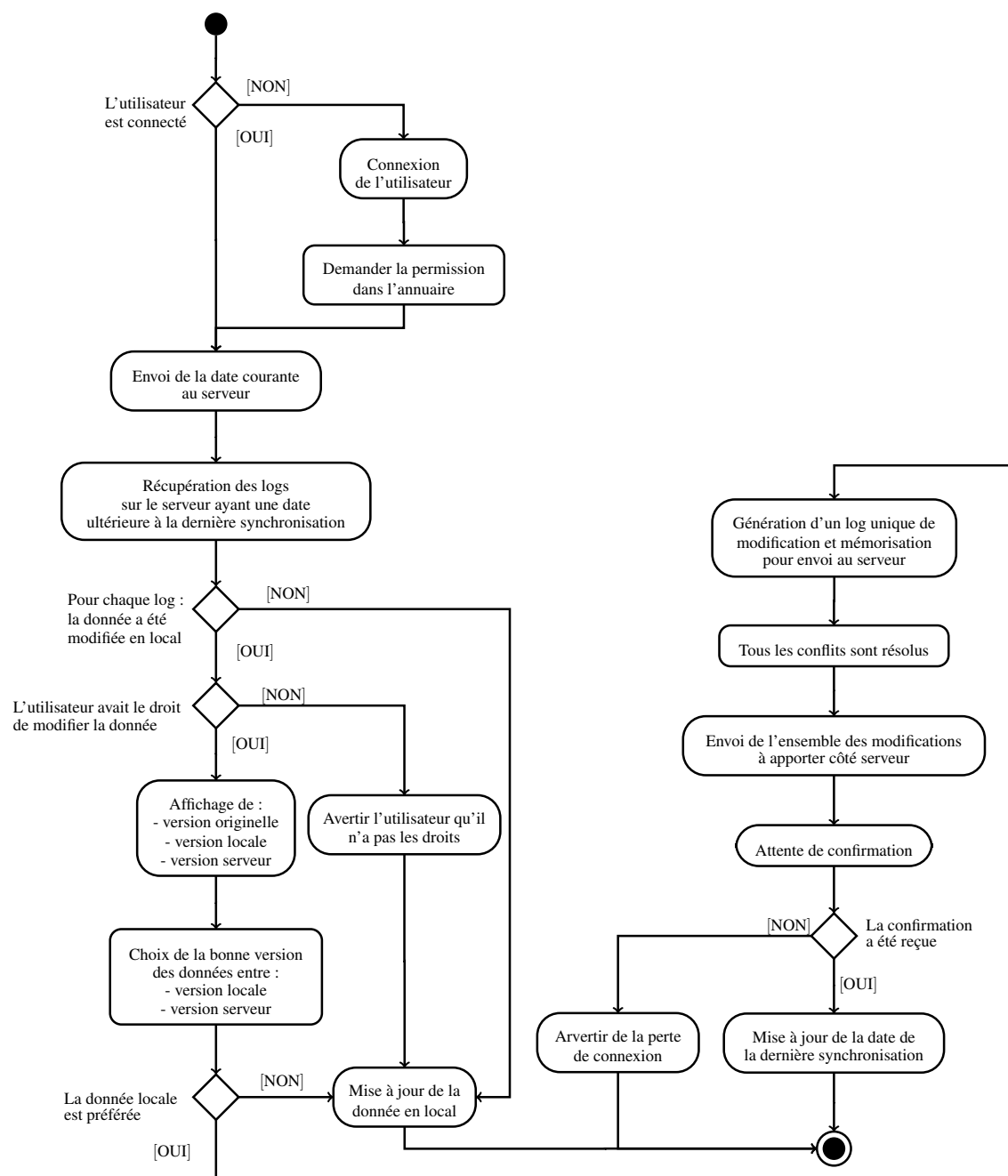


FIGURE 3.12 – Diagramme d'activité : Synchronisation des données



## 4 — Exploitation et préparation

### 4.1 INSTALLATION DES LIBRAIRIES ET DRIVERS

Afin d'assurer le bon fonctionnement de l'application, cette dernière requiert que soient présents les packages suivants (dans une version égale ou supérieure à celle précisée) : Les éléments nécessaires à la première mise en service sont fournis dans le livrable initial.

Microsoft Windows	GNU/Linux
Windows 7	
MinGW 4.8.1 (TMD)	
Qt 5.2.1	Qt 5.2.1
QMySQL 5.6.16	QMySQL 5.6.16

#### 4.1.1 CRÉATION DU PILOTE MYSQL POUR QT5

Cette section précise comment créer le pilote MySQL pour Qt5 en utilisant MinGW. La rédaction de cette dernière a été motivée par le fait que le pilote en question n'est malheureusement pas fourni par défaut dans le framework Qt.

##### INSTALLATION DU FRAMEWORK QT

L'installation des composants par défaut convient très bien, mais il est tout à fait possible de procéder à une installation complète. Sélectionner le répertoire racine d'installation du framework (ici C :). À terme, Qt est alors installé, ici dans le répertoire C :/Qt/Qt5.2.1.

Ensuite, extraire les sources Qt dans un répertoire temporaire de son choix, puis copier le dossier *qt-everywhere-opensource-src-5.2.1* ainsi obtenu dans le répertoire C :/Qt/Qt5.2.1/ et le renommer *sources*.

En fin de manipulation, le framework Qt est installé dans le répertoire C :/Qt/Qt5.2.1/ et les sources présentes dans C :/Qt/Qt5.2.1/sources/.

##### INSTALLATION DU DRIVER QMYSQL

**Installation sous Linux** Vous avez besoin des fichiers d'en-tête de MySQL ainsi que la bibliothèque partagée *libmysqlclient.so*. Selon votre distribution Linux, vous devez peut-être installer un paquet qui est généralement appelé "mysql-dev".

Dites à **qmake** où trouver les fichiers d'en-tête de MySQL et les bibliothèques partagées (ici, on suppose que MySQL est installé dans /usr/local) et lancez make :

```
1 cd $QTDIR/src/plugins/sqldrivers/mysql
2 qmake "INCLUDEPATH+=/usr/local/include" "LIBS+=-L/usr/local/lib_-lmysqlclient_r" mysql
   .pro
3 make
```

Après l'installation de Qt, vous devez également installer le plugin dans l'emplacement standard :

```
1 cd $QTDIR/src/plugins/sqldrivers/mysql
2 make install
```

**Installation sous Windows** Vous devez obtenir les fichiers d'installation de MySQL. Exécutez *Setup.exe* et sélectionnez "Installation personnalisée". Installez le module "Libs & Inclure les fichiers". Construire le plugin comme suit (ici on suppose que MySQL est installé dans C:\MySQL) :

```
1 cd %QTDIR%\src\plugins\sqldrivers\mysql
2 qmake "INCLUDEPATH+=C:/MySQL/include" "LIBS+=C:/MYSQL/MySQL_Server_5.6.16/lib/opt/
   libmysql.lib" mysql.pro
3 nmake
```

Si vous n'utilisez pas un compilateur Microsoft, remplacer **nmake** avec **make** dans la ligne ci-dessus.

### CONSTRUCTION & DÉPLOIEMENT

Ouvrir le terminal de commande de Qt. Ce dernier fonctionne comme n'importe quel terminal à ceci prêt qu'il définit par défaut toutes les variables d'environnement Qt nécessaires. Il s'ouvre sur le répertoire Qt.

Exécuter les commandes suivantes :

```
1 set mysql=C:\\PROGRA~2\\MySQL\\MYSQLS~1.6
2 cd C:\\Qt\\Qt5.2.1\\sources\\qtbase\\src\\plugins\\sqldrivers\\mysql\\
3 qmake "INCLUDEPATH+=%mysql%\\include" "LIBS+=%mysql%\\lib\\libmysql.lib" -o Makefile
   mysql.pro
4 mingw32-make
```

Après construction réussie (et sans erreur), des fichiers seront produits dans C:\Qt\Qt5.2.1/sources/qtbase/plugins/sqldrivers, que sont :

1. libqsqlmysql.a ;
2. libqsqlmysqld.a ;
3. qsqlmysql.dll ;
4. qsqlmysqld.dll.

```
1 For .dll files, move them to C:\Qt\QT5.1.1\5.1.1\mingw48_32\plugins\sqldrivers.
2
3 For .a files, move them to C:\Qt\Qt5.1.1\5.1.1\mingw48_32\lib
4
5 Also, copy libmysql.dll from %mysql%\lib to C:\Windows
6 Testing
7
8 To use the driver, don't forget to add QT += sql to project file, else it don't work.
9
10 Check which drivers are available by this code:
11
12 C++
13 #include <QtCore/QCoreApplication> #include <QtSql> int main() { QCoreApplication a(
   argc, argv); qDebug() << QSqlDatabase::drivers(); return a.exec(); }
14 1
15 2
16 3
17 4
18 5
19 6
20 7
21 8
22
23 #include <QtCore/QCoreApplication>
24 #include <QtSql>
25
```

```

26 int main() {
27     QCoreApplication a(argc, argv);
28     aDebug() << QSqlDatabase::drivers();
29     return a.exec();
30 }

```

## 4.2 INSTALLATION ET CONFIGURATION DE L'ANNUAIRE

L'installation du serveur ldap est faite sous le système windows 7/32 bits ( système utilisé actuellement selon le cahier des charges de consultation), mais aussi sous le système Linux ce qui est demandé pour une imigration future.

### 4.2.1 SERVEURS

#### WINDOWS

La technologie utilisée pour la mise en place des serveurs d'annuaires est Apache DS. Afin de mettre en place le serveur sur une machine Windows, télécharger et installer le logiciel Apache DS à l'adresse suivante <http://directory.apache.org/apacheds/> (ou bien chercher ApacheDS sur votre moteur de recherche préféré). Suite à l'installation, le service ApacheDS - default est démarré, le serveur écoute maintenant sur les ports 10389 et 10636.

Afin de configurer et gérer le serveur il est possible d'installer le logiciel Apache Directory Studio qui fournit une interface utilisateur simple et ergonomique. Pour cela il suffit de créer une connexion sur le serveur via son adresse et le port d'écoute.

Pour générer l'annuaire que nous avons conçu, créer une partition nommée GarmirKatch dont le suffixe sera dc=GarmirKatch,dc=fr. Importer via Apache Directory Studio les fichiers schemagarmirApache.ldif, Racine.ldif, Users.ldif, Groups.ldif et Missions.ldif dans le serveur.

#### LINUX

Pour configurer votre serveur linux voir cette adresse <sup>1</sup>, vous y trouverez toutes les explications pour installer et configurer votre serveur.

Toute fois, ci joint à documentation les fichiers Database.ldif, GarmirSchema.ldif, Racine.ldif, Users.ldif, Groups.ldif et Missions.ldif qui sont des fichiers de configurations.

### 4.2.2 CONFIGURATION DES CLIENTS

Chaque client devra être configuré avant de partir en mission pour intégrer l'adresse et le port d'écoute du serveur dans le fichier de consultation ldap.ini.

## 4.3 PROCÉDURE DE GÉNÉRATION DE LA BASE DE DONNÉES

### 4.3.1 GÉNÉRATION DES TABLES SQL AVEC JMERISE

#### PRÉSENTATION


JMerise est un logiciel dédié à la modélisation des modèles conceptuels de donnée pour Merise. Il permet aussi, les relations réflexives, la généralisation et la spécialisation des entités. Il génère le MLD et le script MySQL associé.


#### PROCÉDURE D'INSTALLATION DE JMERISE

Pour installer et utiliser JMerise, unzipper simplement le fichier "**JMerise.zip**" et exécuter le jar nommé "*JMerise.jar*".

1. <http://www.bgeek-france.ec0.fr/2011/10/23/admin/linux/installation-et-parametrage-de-%C2%AB-base-%C2%BB-d%E2%80%99openldap-ubuntu-11-04-et-ubuntu-11-10-mode-cnconfig.html>

**GÉNÉRATION DE TABLES SQL**

Une fois la MCD réalisée, JMerise propose une méthode de vérification qui permet de s'assurer que le modèle que l'utilisateur vient de créer est correct. Pour cela, il suffit de cliquer sur le bouton .

L'outil peut aussi générer la MPD associée ainsi que le code MySQL permettant la création des tables en cliquant sur le bouton .

**4.4 CONFIGURATION DU CLUSTER****PROCÉDURE DE FAIL-BACK (RÉTABLISSEMENT DU SERVEUR MAÎTRE)**

- éteindre le serveur esclave ;

**Notation 4.1.** *Les clients n'ont pas besoin d'être redémarrés. Étant préalablement configurés ils se reconnecteront au maître une fois ce dernier remis en ligne.*

- copier les données de l'esclave sur le maître ;
- redémarrer les deux serveurs.





## Table des figures

2.1	Architecture fonctionnelle . . . . .	7
2.2	Architecture générale . . . . .	8
2.3	Diagramme des cas d'utilisation . . . . .	9
3.1	Schéma de l'annuaire LDAP mis en place . . . . .	14
3.2	Architecture globale . . . . .	15
3.3	Authentification pour la synchronisation . . . . .	16
3.4	Quelques exemples de demande de permission . . . . .	17
3.5	Cluster, serveur maître et esclave directement connectés . . . . .	20
3.6	Utilisation des modes connecté et hors-ligne . . . . .	22
3.7	Diagramme des cas d'utilisation pour la synchronisation . . . . .	23
3.8	Diagramme des cas d'utilisation pour le mode connecté . . . . .	24
3.9	Diagramme des cas d'utilisation pour le mode hors-ligne . . . . .	24
3.10	Diagramme d'activité : Ajout d'une donnée . . . . .	26
3.11	Diagramme d'activité : Modification d'une donnée . . . . .	27
3.12	Diagramme d'activité : Synchronisation des données . . . . .	28