SYSTÈME DE GESTION DES TRANSPORTS

Version 0.1

16 mars 2014

Dossier d'Architecture Technique

Faculté des Sciences
Aix*Marseille Université

Dossier d'Architecture Technique

Version 0.1

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	Mehdi-Jonathan CADON, Lionel GAIROARD, Anthony JULIEN, Adrien LERICOLAIS, Rémi MEZELLE, Ravi PACHY, Bien Aimé SUANGA WETO, Ahoua Khady TOURE	Rédaction

DIFFUSION

Version	Date	Approbateur(s)
1.0	-	-

VALIDATION

Version	Date	Responsable(s)
1.0	-	-

Dossier d'Architecture Technique

Version 0.1

Table des matières

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 au TMS, et le fonctionnement global de l'organisation grâce à l'assistance de l'outil et l'automatisation de certaines de ces 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, AMU FSI 2014 s'engage à fournir les langues traductions des langues suivantes :

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

AMU FSI 2014 s'est engagé à respecter les contraintes ci-dessous :

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 au 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

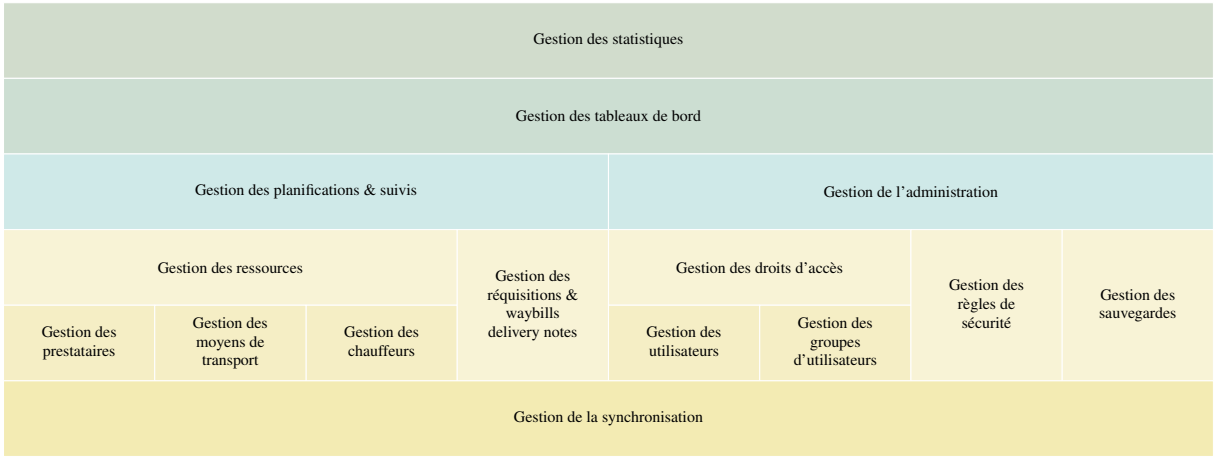

FIGURE 2.1 – Architecture fonctionnelle

2.2 Architecture physique

La solution est prévue pour fonctionner sur l'architecture générale présentée dans la figure 2.2.

Comme présenté en figure 2.2, 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 langages suivants sont fournis :

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

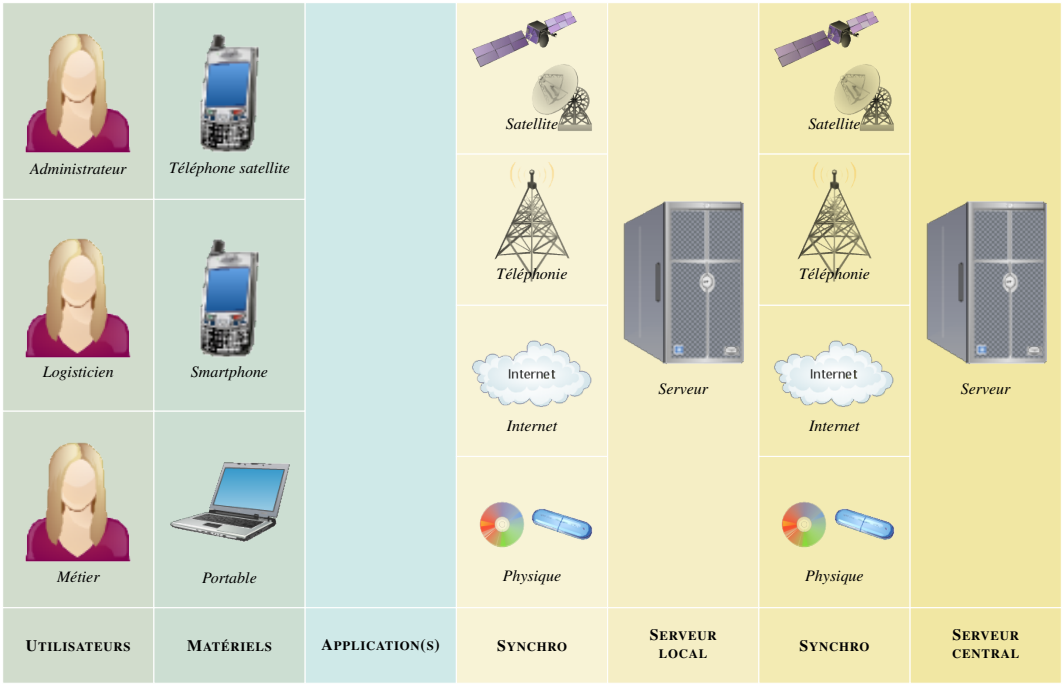

FIGURE 2.2 – Architecture générale

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

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.

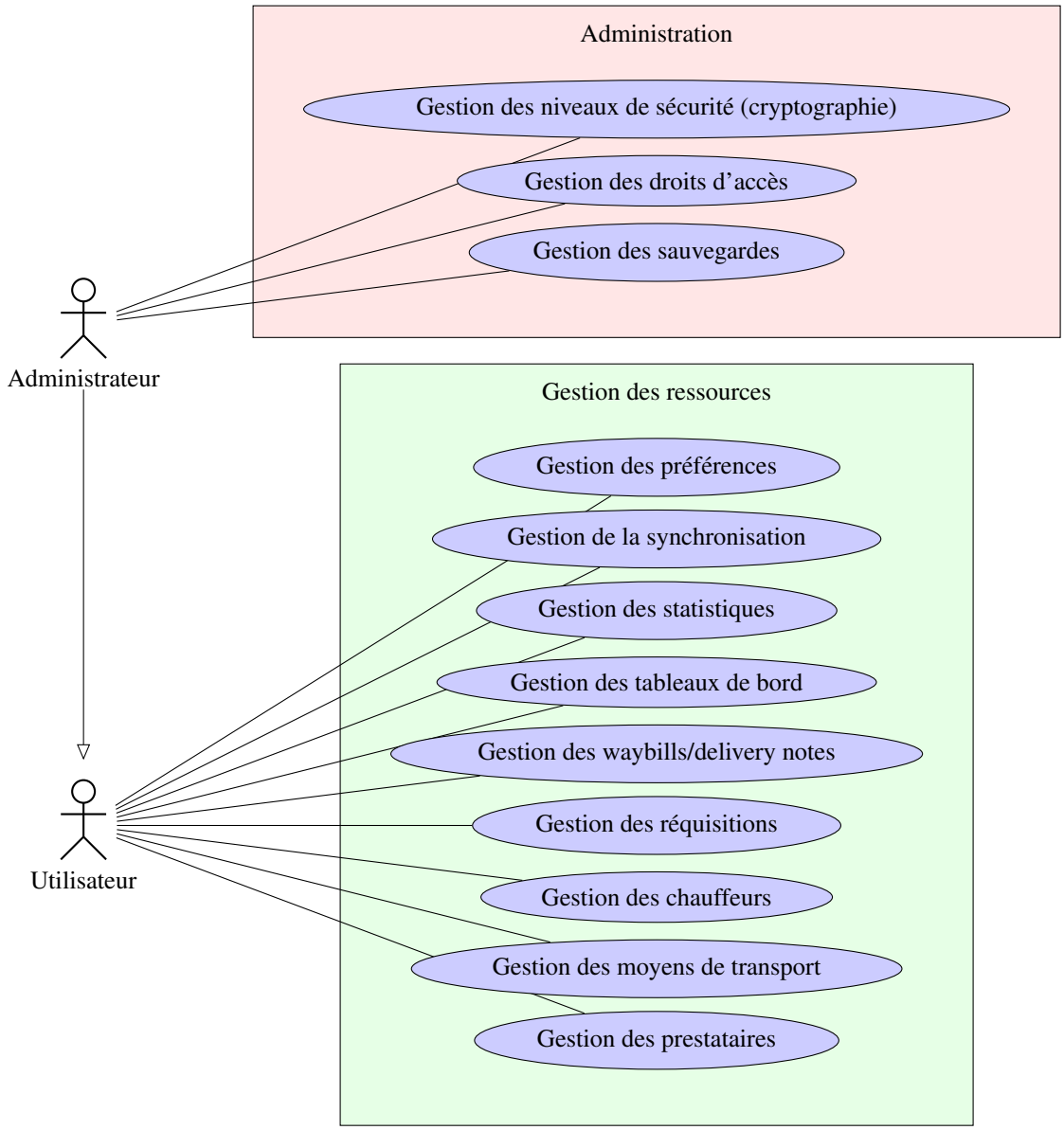

FIGURE 2.3 – Diagramme des cas d'utilisation

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 ;
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 décris 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 fournit, 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 définit, 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 fournit 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 relies. 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
PRESTATAIRES
CHAUFFEURS
VÉHICULES

2.3.3 CONTRAINTES

Cette section tient compte des contraintes respectées jusque là non évoquées.

Contrainte 2.3.1 — Contraintes de délai. AMU FSI 2014 s'engage à fournir des garanties contractuelles de délais, et à s'y tenir. Les éventuelles pénalités de retard seront fixées d'un accord commun avec Garmir Khatch.

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 réalisé 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. *optimisé et sécurisé (ce qui exclu les langages comme Java) ;*
6. *compatible avec les divers autres éléments utilisés (SGBDR, LDAP, XML, ...) ;*

3.2 Authentification

L'authentification est gérée par un annuaire LDAP actuellement en cours d'étude et de développement.

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 de prédilection de notre entreprise, ce qui garantit une interopérabilité maximale.

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 un 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.

Structuration des données

Génération de la base de données

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

1. *Database.properties* ;
2. *Database.sql*.

Database.properties

Ce fichier définit les propriétés 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
```

Database.properties

1. *dbDriver* est le driver Qt correspondant à la base de données considérée (dans le cas présent MySQL) ;
2. *dbHostName* est l'adresse URL de la base de données (ici localhost) ;
3. *dbUserName* est le nom d'utilisateur (ici root) ;
4. *dbPassword* est le mot de passe d'utilisateur (ici aucun) ;
5. *dbName* 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 :

```cpp
#include <QtGlobal>
#include <QtSql>

int main() {
    QSqlDatabase db = QSqlDatabase::addDatabase("QMYSQL");
    db.setHostName("localhost");
    db.setUserName("root");
    db.setPassword("");
    db.setDatabaseName("GkLogistic");
    if (!db.open()) {
        qDebug() << "Connection to " << db.hostName() << " is failed" << endl;
        qDebug() << "An error was encountered:" << endl;
        qDebug() << db.lastError().text() << endl;
        return EXIT_FAILURE;
    }
    qDebug() << "Connection to " << db.hostName() << " is successfull" << endl;
    // do something
    db.close();
    return EXIT_SUCCESS;
}
```

Database.example.cpp

DATABASE.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.

3.3.3 LE TRANSFERT

Le transfert des données quant à lui, est réalisé selon un format et une norme qui est préalablement défini.

FORMAT

Eu égard des solutions retenues pour le transfert des données, le transfert des données se fera grâce à des fichiers XML.
La possibilité est donnée de compresser ces fichiers, dans un premier temps au format *ZIP*, mais ce choix peut être remis en question selon les observations réalisées sur les données de test, afin de retenir une éventuelle meilleure solution.

NORME

3.3.4 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. Les étapes de cette configuration seront fournies dans le manuel de déploiement. Dans l'architecture proposée (voir figure 3.1), 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.

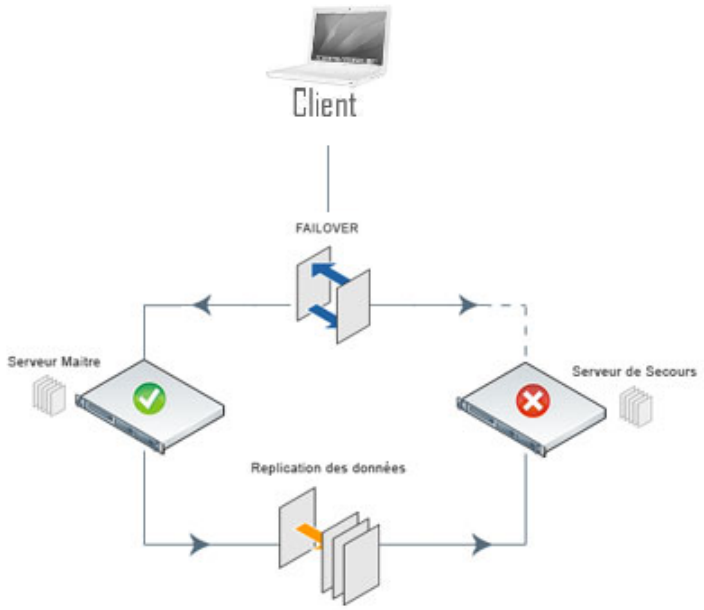

FIGURE 3.1 – Cluster, serveur maître et esclave directement connectés

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.

Procédure de fail-back (rétablissement du serveur maître)

- éteindre le serveur esclave ;

  Notation 3.2. *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.

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 [1].
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 Fig. 3.2 : 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.

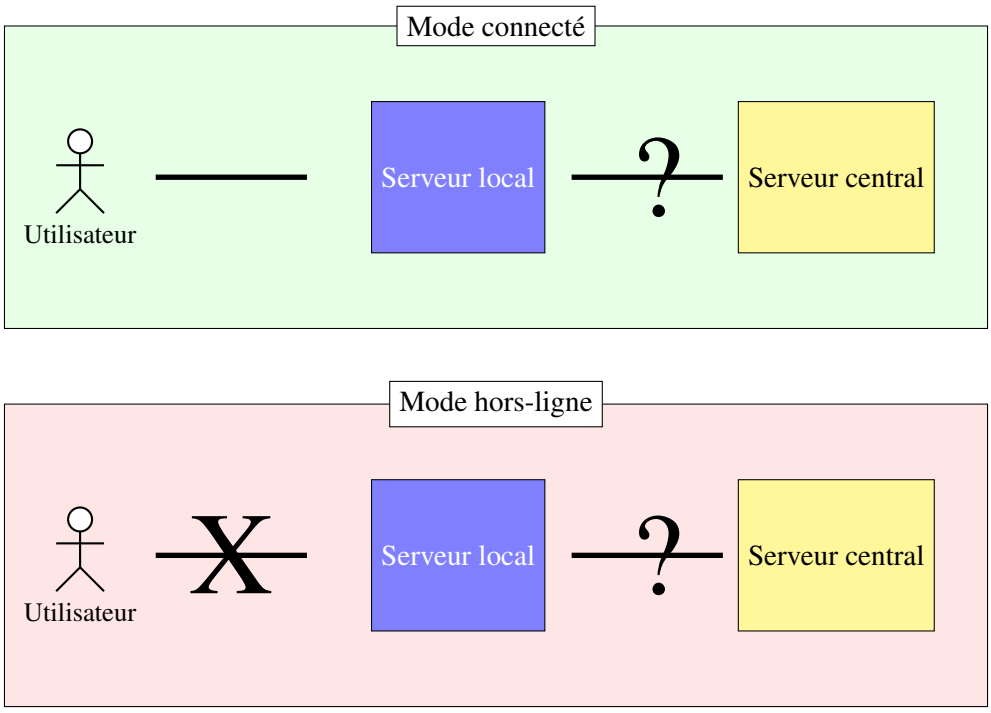

Figure 3.2 – Utilisation des modes connecté et hors-ligne

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 inclue dans le profil sont synchronisés.

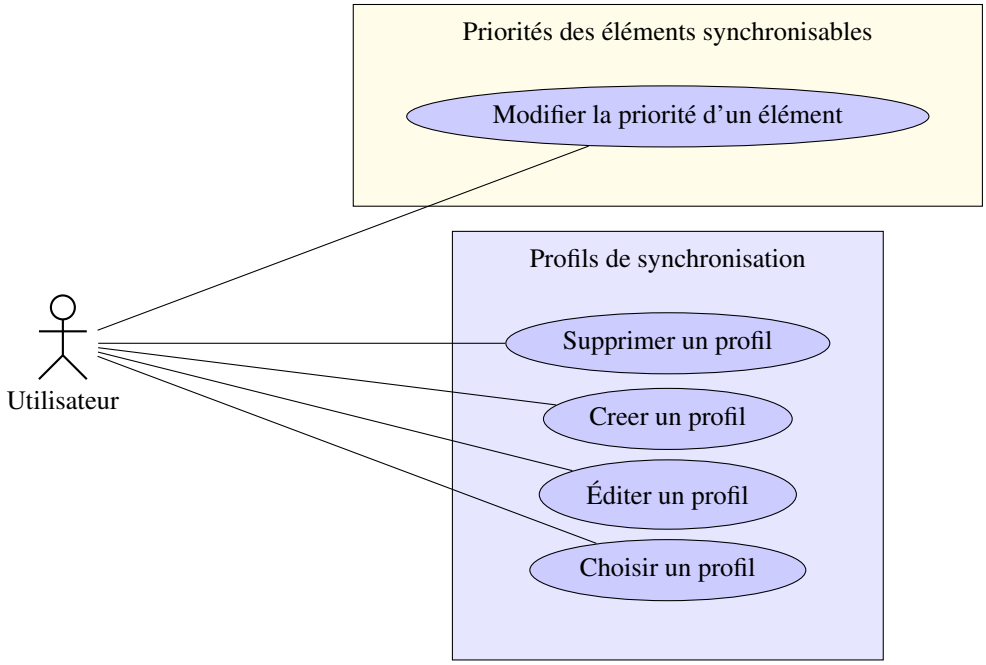

FIGURE 3.3 – Diagramme des cas d'utilisation pour la synchronisation

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 [2] ». 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.

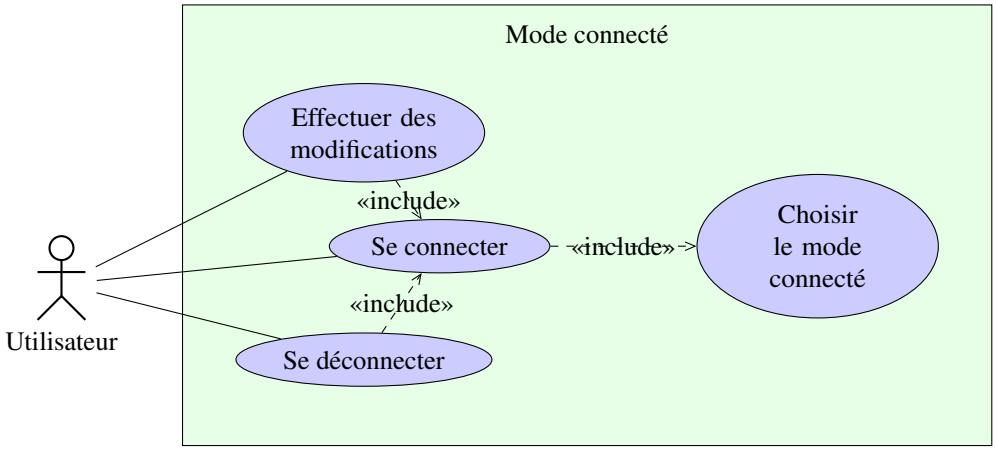

FIGURE 3.4 – Diagramme des cas d'utilisation pour le mode connecté

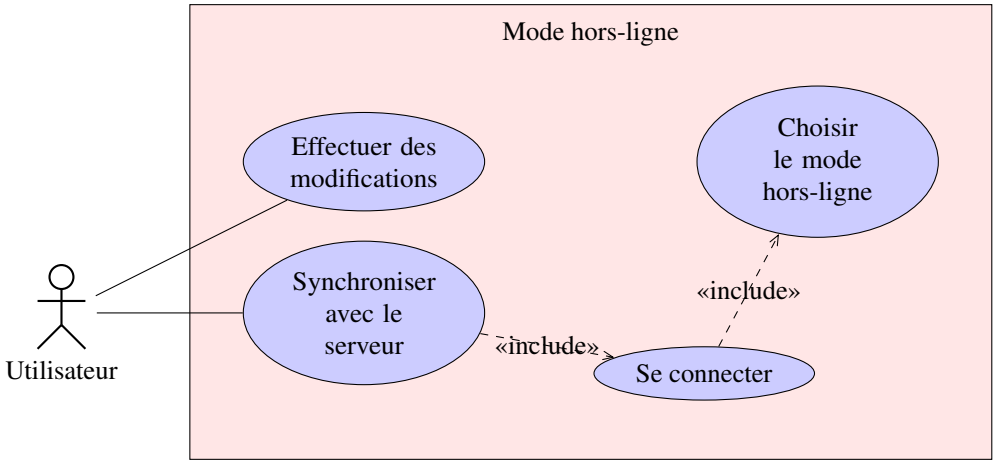

FIGURE 3.5 – Diagramme des cas d'utilisation pour le mode hors-ligne

3.5.5 Gestion des conflits

Durant la synchronisation, que ce soit en mode connecté ou hors-ligne, 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 [3].

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 deux heures 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. À cette fin, l'application enregistre les éventuels changements d'heure opérés en local pour notifier le serveur des écarts d'horodatage.

4 — Exploitation et préparation