



POLYTECH[®]
2011-2012

Mise en place d'un service SMS

Rapport de synthèse



Guillaume Leroy
Emmanuel Yagapen
Polytech' Montpellier
pour Synox représenté par
Jérôme Fenwick & Joffrey Verdier

Remerciements

Nous tenons à remercier toutes les personnes qui nous ont permis de mener à bien ce projet pendant ces deux mois :

- M. Jérôme Fenwick et M. Joffrey Verdier, de l'entreprise demandeuse Synox, pour leur disponibilité et leur aide au bon déroulement du projet
- Mme Michelle Cart, notre responsable pédagogique, qui nous a suivis régulièrement et conseillé tout au long de ce projet
- L'administration de Polytech'Montpellier pour nous avoir fourni le matériel nécessaire à la bonne réalisation de ce projet

Sommaire

1	Introduction.....	5
2	Présentation de l'environnement	6
2.1	Le groupe SYNOX.....	6
2.2	La plateforme Machine-to-Machine de gestion d'objets communicants.....	6
3	Présentation du projet	7
3.1	Le problème initial.....	7
3.2	Les besoins fonctionnels.....	7
3.3	La mission	7
3.4	Contraintes	7
3.4.1	Contraintes techniques	7
3.4.2	Contraintes temporelles.....	8
3.5	Ressources.....	8
4	Déroulement du projet.....	9
4.1	Gestion du projet.....	9
4.2	Démarche	9
4.2.1	Méthodes utilisées	9
4.2.2	Choix technologiques	9
4.3	Communication et synchronisation	10
5	Travail réalisé.....	11
5.1	Architecture du système d'envoi et de réception de SMS.....	11
5.1.1	Le service SMS	12
5.1.2	La base de données	12
5.1.3	Interface utilisateur.....	12
5.2	Conception	12
5.2.1	Diagrammes des cas d'utilisation.....	12
5.2.2	Diagramme de classe.....	13
5.2.3	Maquettage	15
5.3	Développement.....	15
5.3.1	Les commandes AT	15
5.3.2	Le mode PDU et la librairie ATSMS.....	15
5.3.3	Le service SMS	17
5.3.4	La base de données	18

5.3.5	L'interface graphique	19
5.4	Tests.....	21
5.5	Limites	22
6	Améliorations possibles.....	23
6.1	Permettre une utilisation multi-utilisateurs.....	23
6.2	Sécuriser l'accès à la base de données.....	23
6.3	Implémenter une gestion des contacts.....	23
7	Difficultés rencontrées	24
7.1	Apprentissage des technologies.....	24
7.2	Problèmes de compatibilité entre le Framework .Net et le JavaScript.....	24
7.3	Les accusés de réception.....	24
8	Conclusion	25
9	Annexes	26
9.1	Les commandes AT.....	26
9.1.1	Choix du format des messages.....	26
9.1.2	Choix de la mémoire.....	26
9.1.3	Affichage des messages.....	26
9.1.4	Envoi d'un message	26
9.2	Envoyer et recevoir des données	26
9.2.1	Connexion au modem	27
9.2.2	Envoi de données au modem.....	27
9.2.3	Réception de données.....	28
9.2.4	Déconnexion du modem	28
9.2.5	Exemple d'utilisation du protocole de communication	28
9.3	Diagramme de Gantt	30

Table des images

Figure 1 - Schéma d'architecture	11
Figure 2 - Diagramme des cas d'utilisation : Application	13
Figure 3 - Diagramme des cas d'utilisation : Utilisateur.....	13
Figure 4 - Diagramme des classes	14
Figure 5 - Informations nécessaires pour l'envoi d'un SMS en mode texte	15
Figure 6 – Informations nécessaires pour l'envoi d'un SMS en mode PDU	16
Figure 7 - SQL Server Management Studio : Structure de la table Message	19
Figure 8 - Requête LINQ : SMS en attente d'accusé.....	19
Figure 9 - Capture d'écran de l'interface d'envoi de SMS	20
Figure 10 - Capture d'écran de la boîte d'envoi	20
Figure 11 - Capture d'écran de la boîte de réception	21
Figure 12 - Extrait de l'algorithme de connexion au modem GSM	27
Figure 13 - Extrait de l'algorithme d'envoi de commandes au modem GSM	27
Figure 14 - Algorithme de réception des messages dyu modem GSM	28
Figure 15 - Protocole de communication entre le service SMS et le modem GSM	29
Figure 12 - Diagramme de Gantt (tableau)	30
Figure 13 - Diagramme de Gantt (schéma)	30

1 Introduction

Dans le cadre de notre formation de 5ème année en école d'ingénieurs à Polytech'Montpellier, nous devons effectuer un projet industriel en collaboration avec une entreprise. Ce projet a pour but de nous placer dans les conditions du monde professionnel.

Dans cette optique, nous avons choisi de réaliser le projet industriel proposé par M. Fenwick, représentant de l'entreprise SYNOX, société de prestation de services informatiques aux entreprises.

Synox possède une plateforme permettant à des applications d'échanger des informations avec des *objets distants*ⁱ à travers le réseau *GPRS*ⁱⁱ. Cependant ce réseau présente des problèmes de fiabilité compromettant le bon fonctionnement de la plateforme. Le réseau *GSM*ⁱⁱⁱ sur lequel s'appuie la technologie *SMS*^{iv} étant plus fiable, il est alors nécessaire de s'appuyer dessus afin d'assurer une continuité des échanges.

La mission qui nous a été confiée consiste donc à développer un mécanisme **d'envoi et de réception de SMS qui s'intégrera à la plateforme existante**.

Notre choix s'est porté sur ce projet car ce travail porte sur des technologies en fort développement que sont les objets mobiles et le Web. De plus, cette mission comprend des aspects dans le domaine du réseau que nous souhaitons approfondir.

Le projet a débuté le 5 décembre pour prendre fin le 9 février, soit une durée de 10 semaines.

Ce rapport vise à expliquer de manière synthétique le travail accompli durant ce projet. Pour plus de détails, nous vous invitons à vous reporter au rapport technique.

Dans un premier temps nous présenterons l'entreprise et le contexte dans lequel s'inscrit notre mission. Puis nous aborderons les objectifs attendus et le travail réalisé pour les atteindre. Nous décrirons ensuite la démarche suivie et les problèmes que nous avons rencontrés. Enfin, nous ferons un bilan sur le travail que nous avons effectué.


2 Présentation de l'environnement

2.1 Le groupe SYNOX

Le groupe SYNOX est une société informatique spécialisée dans les solutions mobiles et collaborative. Son activité porte essentiellement sur le développement spécifique et la mise en place d'infrastructures mobiles.

En 2010 son chiffre d'affaires a atteint plus 2,5 millions d'euros porté par la bonne santé du **cloud computing**.

2.2 La plateforme Machine-to-Machine de gestion d'objets communicants

Synox développe pour ses clients des applications communiquant avec des objets distants grâce au réseau GPRS. Cette communication est gérée par une plateforme Machine-to-Machine (**M2M**) centralisée qui **donne accès aux clients**  un suivi de leurs objets distants.

Schéma

On peut ainsi prendre l'exemple d'un autobus qui est équipé d'un capteur qui envoie un signal à la plateforme à chaque fois qu'il arrive à un arrêt. La plateforme se charge alors de retransmettre l'information à d'autres services comme par exemple une application mobile.

3 Présentation du projet

3.1 Le problème initial



Les objets de la plateforme M2M du groupe Synox **utilise** actuellement le réseau GPRS pour communiquer. Cependant, il se peut que ce réseau ne soit pas disponible à certains endroits et à certains moments, rendant impossible toute communication entre les objets distants et la plateforme.

On cherche donc à **résoudre un problème de fiabilité** lié au système actuel.

3.2 Les besoins fonctionnels

Pour pallier ce problème, Synox a choisi d'utiliser le réseau GSM, beaucoup plus fiable, en permettant aux objets distants de communiquer avec la plateforme par SMS.



Synox a donc besoin de fournir à ces programmes la possibilité d'envoyer des SMS afin d'assurer une continuité du service en cas d'une panne du réseau GPRS.

Il faut donc que les objets distants, dotés en conséquence d'un *modem*^v GSM, puissent envoyer et recevoir des SMS. De même, la plateforme M2M doit être dotée des mêmes capacités.

La solution sera associée à une base de données enregistrant tous les messages, ainsi qu'à une interface graphique de gestion en ligne pour permettre aux utilisateurs de communiquer par SMS avec les objets distants.

3.3 La mission

Notre mission consiste donc à concevoir et à développer les fonctionnalités suivantes du système d'envoi et de réception de SMS :

- Envoi et réception de SMS à travers un *service Windows*^{vi}
- Stockage des messages envoyés et reçus dans une base de données
- Associer un statut aux SMS envoyés afin de suivre leur état
- Ecriture et consultation des messages depuis une interface graphique

L'interface graphique a pour objectif de permettre à un utilisateur d'avoir un accès à l'ensemble des fonctionnalités proposées par le système.

3.4 Contraintes

3.4.1 Contraintes techniques

Pour mener à bien ce projet, nous devons respecter plusieurs contraintes techniques. Tout d'abord, le projet doit être développé à l'aide du *framework .Net*^{vii} en version 4.0 ou 3.5.

La base de données, afin de s'intégrer au framework .Net, doit être implémentée avec le système de gestion de base de données SQL Server 2008. Ces contraintes sur le framework sont dues au fait que la plateforme M2M déjà existante est développée avec cette technologie. Le Groupe Synox maintient ainsi une cohérence entre tous les éléments de cette plateforme.

Enfin, la liaison des données entre la base de données et le service ou l'interface web doit être effectuée en LINQ to SQL ou en ADO.NET.

3.4.2 Contraintes temporelles

Nous disposons pour réaliser ce projet d'un délai de 9 semaines à compter du 5 Décembre 2011. Le planning prévisionnel a été établi comme suit dans la lettre de mission :

Le projet débute le lundi 5 Décembre 2011 pour prendre fin le vendredi 3 Février 2012 avec la livraison de la solution.

De plus, une soutenance de projet est prévue entre le 8 et le 10 Février 2012.

3.5 Ressources

Pour mener à bien ce projet, nous avons constitué une équipe de deux élèves-ingénieur avec un planning de 30 jours.

Nous avons pu disposer d'une salle de projet fournie par Polytech' Montpellier et équipée de d'un ordinateur et de trois écrans. Nous avons également utilisé nos ordinateurs personnels.

M. Fenwick, notre demandeur, nous a également fourni un modem SMS muni d'une carte SIM.

4 Déroulement du projet

4.1 Gestion du projet

Le projet a commencé avec le premier entretien que nous avons eu avec le demandeur. À partir de cette discussion et des réponses qu'il nous a fournies, nous avons élaboré notre lettre de mission qui définit le champ de l'étude, le point d'arrivée et les modalités du projet.

Au cours de cet entretien, nous avons également découpé le projet en plusieurs phases.

La première phase consiste à mettre en œuvre l'envoi de SMS à l'aide du modem GSM. La communication avec le modem se faisant à l'aide d'instructions particulières (les **commandes AT**, aussi appelées **commandes Hayes**) via un port **COM**, il nous faut dans un premier temps étudier les possibilités de ce système et le prendre en main.

La deuxième phase consiste à traiter la réception des SMS et des accusés de réception.

Enfin, la troisième phase, la plus longue, consiste à concevoir et à réaliser le service Windows et l'interface graphique en ligne de gestion des messages.

Nous avons utilisé comme support de planification le logiciel Microsoft Project qui permet de planifier automatiquement les tâches en fonction des contraintes de début et de fin, qui prend en compte les jours fériés et les week-ends. Cet outil permet également de suivre le projet notamment grâce aux outils graphiques (diagramme de Gantt par exemple) qu'il met à disposition des utilisateurs.

Lors de l'évaluation des durées des tâches, nous avons dû prendre en compte le temps de prise en main du modem GSM et des commandes AT, ainsi que d'une productivité plus faible durant la période de Noël.

Vous trouverez en annexe le **diagramme de Gantt** du projet.

4.2 Démarche

4.2.1 Méthodes utilisées

Nous avons utilisé UML pour modéliser la base de données et écrire les cas d'utilisation. Nous n'avons pas utilisé d'autres méthodes particulières, le projet étant principalement concentré sur la conception et le développement.

4.2.2 Choix technologiques

Les choix technologiques que nous avons à effectuer sont relativement restreints pas les contraintes du projet.

Afin de mettre en place la liaison entre le service, l'interface graphique et la base données, nous avons le choix entre deux bibliothèques logicielles, **LINQ to SQL** et **ADO.Net**. Nous avons synthétisé les avantages et inconvénients dans le tableau suivant :

Bibliothèque Logicielle	LINQ to SQL	ADO.Net
Mise en place	Rapide	Plus longue
Utilisation	Relativement aisée avec le langage LINQ	Nécessite un temps d'apprentissage
Performance	Problème avec des bases de données complexes	Très bonne

Notre base de données n'étant pas complexe, nous avons décidé d'utiliser LINQ to SQL, plus rapide à mettre en place et plus facile à manipuler. De plus, nous avons déjà utilisé cette technologie dans un précédent projet.

4.3 Communication et synchronisation

Afin de pouvoir partager rapidement le résultat de nos travaux, nous avons décidé **de réaliser ce projet de la façon la plus collaborative possible**.

Nous avons ainsi utilisé les outils **Google Documents** pour le partage de la documentation, **Github** pour le partage du code et de documents, et **Remember The Milk** pour le partage des tâches.

De plus la possibilité de disposer d'une salle de projet a été une excellente occasion pour travailler ensemble et communiquer directement pendant ces six semaines.

Concernant notre propre communication et synchronisation au sein du groupe, elle a pu se faire facilement puisque nous étions tous deux présents aux mêmes horaires dans une même salle. De plus, nous nous entendons bien.

Ainsi, nous ne nous sommes pas attribués de rôles fixes mais avons alterné. Cela dans le but de manipuler l'ensemble des technologies et de pouvoir prendre du recul.

5 Travail réalisé

5.1 Architecture du système d'envoi et de réception de SMS

L'architecture du système d'envoi et de réception de SMS peut être représentée par le schéma suivant :

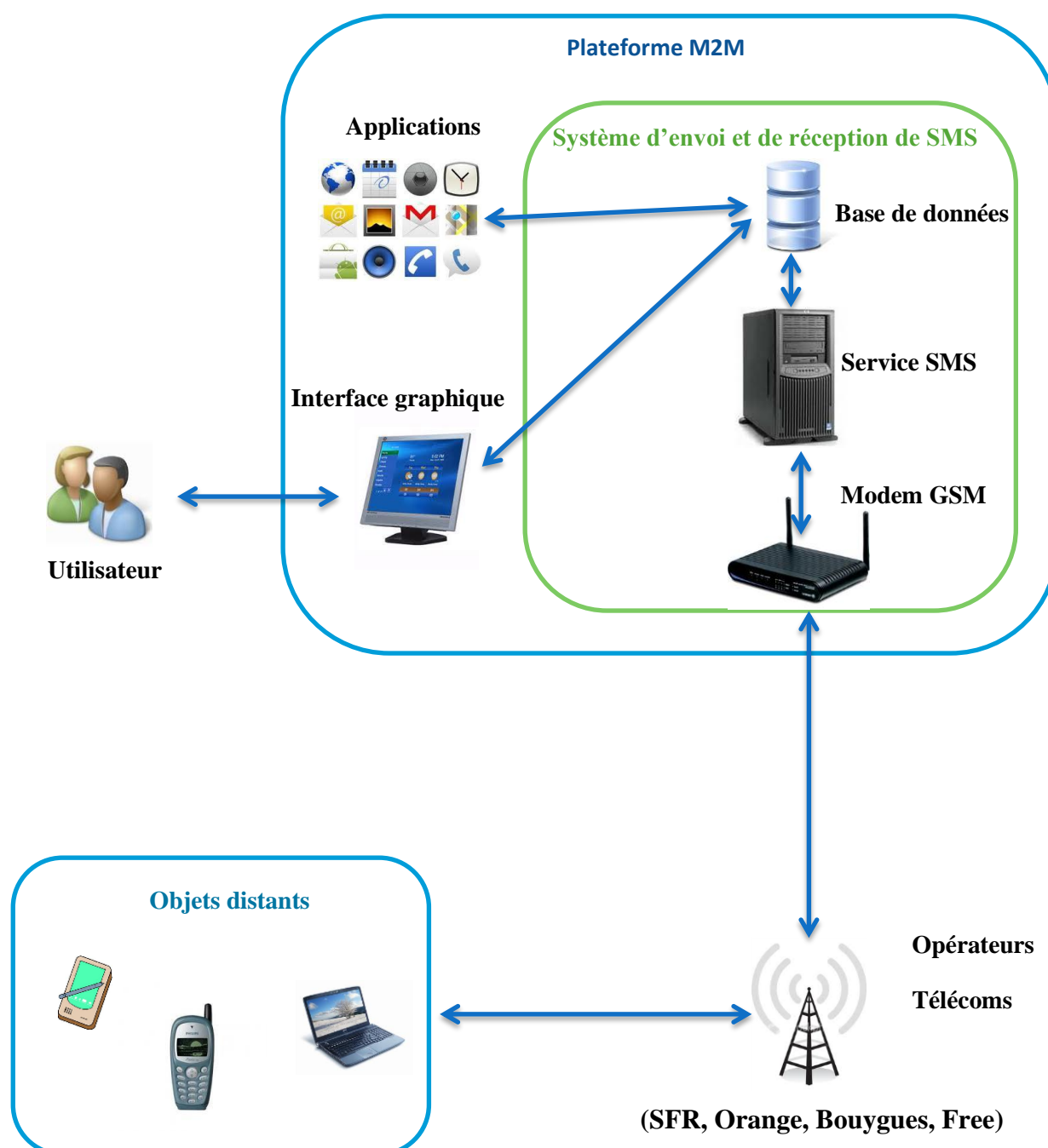


Figure 1 - Schéma d'architecture

Le système d'envoi et de réception de SMS est composé de trois parties : le service SMS, la base de données et interface graphique.

5.1.1 Le service SMS

Le service a pour but d'envoyer et de réceptionner des messages afin de satisfaire les demandes des clients. Pour cela, il va contrôler le modem SMS à travers des commandes AT afin que ce dernier puisse envoyer et recevoir correctement des messages.

De plus, pour des questions de compatibilité avec la plateforme M2M de Synox, il a été décidé que le service SMS prendra la forme d'un service Windows qui sera à l'écoute de la base de données afin de traiter les demandes de réception et d'envoi à travers un traitement par lot (batch).

5.1.2 La base de données

La communication entre les clients et le service SMS s'effectue par l'intermédiaire d'une base de données.

En effet, lorsqu'un utilisateur de la plateforme veut envoyer un SMS, les données du nouveau message sont insérées dans la base de données. Ainsi, le service, qui est configuré pour vérifier régulièrement la base de données, va lire les informations relatives au message afin de transmettre les bonnes commandes d'envoi au modem.

Au niveau de la réception des SMS, le service interroge régulièrement le modem pour y récupérer des nouveaux messages afin de les insérer dans la base de données.

5.1.3 Interface utilisateur

Cette interface permet aux clients d'envoyer et de recevoir des SMS en alimentant ou en consultant la base de données. Elle sera sous forme d'un site web d'administration qui permettra la visualisation des messages envoyés/reçus et elle inclura une interface pour envoyer le SMS avec les options possibles par le format PDU.

Il peut également être envisagé de créer un web Service pour que les applications Synox puissent s'y connecter simplement.

5.2 Conception

Afin de procéder à la phase de conception, nous avons réalisé des diagrammes de cas d'utilisation et un diagramme de classe.

5.2.1 Diagrammes des cas d'utilisation

5.2.1.1 Application

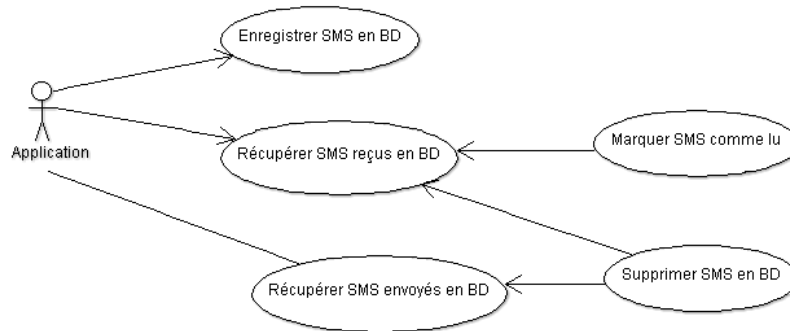


Figure 2 - Diagramme des cas d'utilisation : Application

La communication entre le service chargé de l'envoi et la réception des messages avec les différentes applications se font par l'intermédiaire d'une base de données.

Ainsi, lorsqu'une application veut envoyer un SMS, elle insère toutes les informations concernant le message en base de données. De même, lorsqu'elle a besoin de récupérer les SMS reçus ou envoyés, elle consulte la base de données. L'application peut alors supprimer ou marquer comme lu les messages reçus.

5.2.1.2 Utilisateur

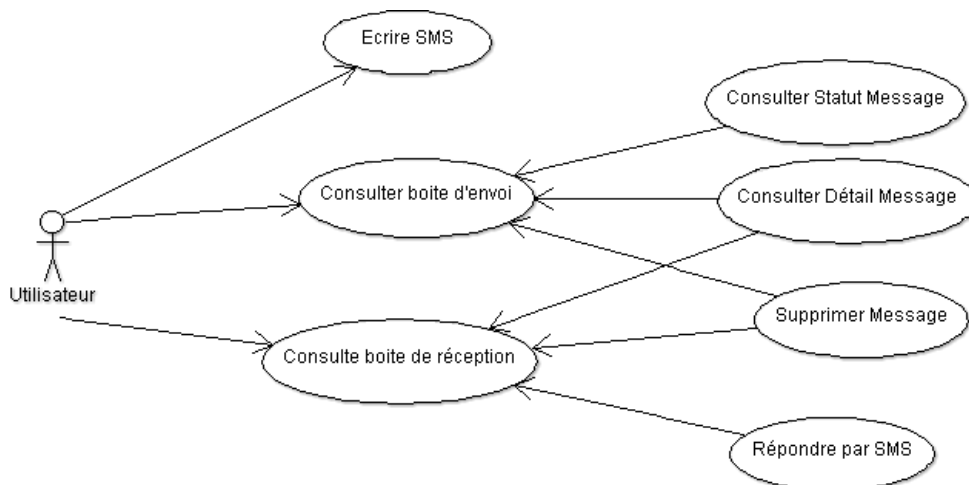


Figure 3 - Diagramme des cas d'utilisation : Utilisateur

Les utilisateurs auront la possibilité d'envoyer et de réceptionner des SMS par l'intermédiaire d'une interface graphique. La consultation des messages se compose de deux parties : les SMS envoyés et ceux reçus par le modem. L'utilisateur pourra les supprimer et marquer comme lu les messages réceptionnés.

5.2.2 Diagramme de classe

Après l'étude des besoins fonctionnels et des cas d'utilisation, nous avons pu déterminer de quelles informations nous avons réellement besoin pour le bon fonctionnement de notre application.

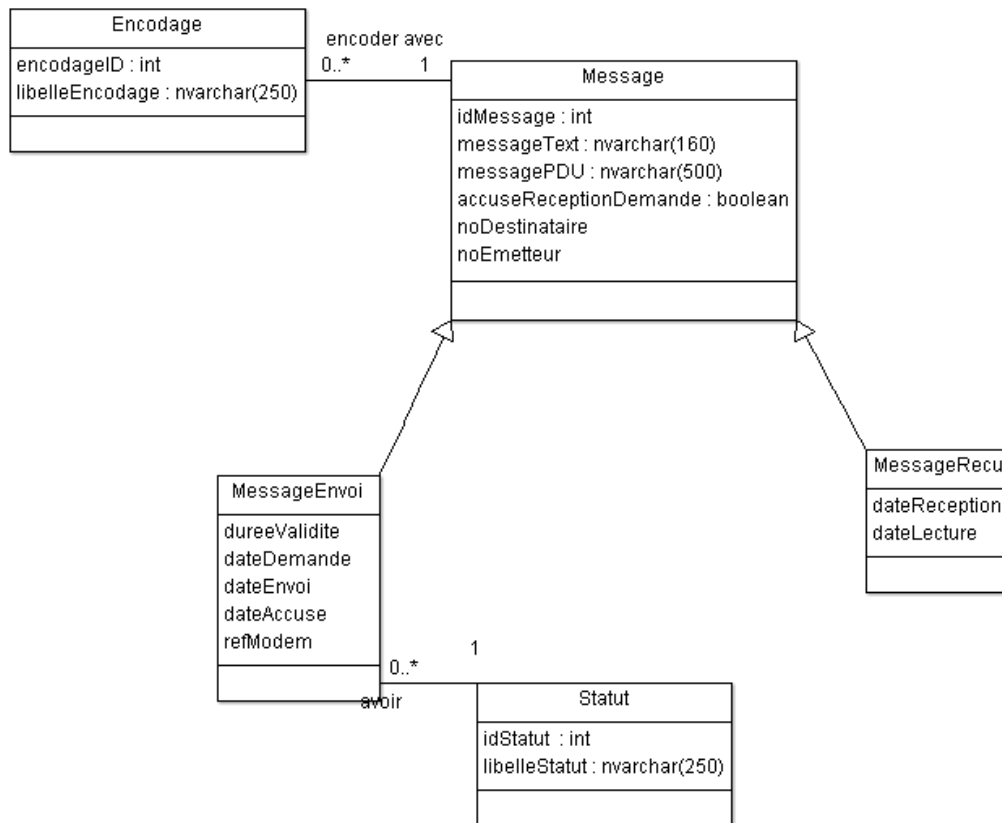


Figure 4 - Diagramme des classes

La classe **Message** regroupe les informations partagées par les classes **MessageEnvoi** et **MessageReçu** qui en héritent. En effet, chaque message est caractérisé par un identifiant unique, un destinataire, un émetteur et un message. L'attribut **accuseReceptionDemande** indique si le destinataire ou l'émetteur a demandé un accusé de réception du message au correspondant.

Le modem peut envoyer et recevoir des SMS au format **PDU**. Ceux-ci sont alors stockés dans le champ **messagePDU**. Si c'est un SMS au format texte, ce champ garde une valeur nulle.

Un **message envoyé** est caractérisé par une durée de validité. Un destinataire ne reçoit pas un SMS dont la période de validité est expirée, ce contrôle étant effectué par l'opérateur de télécommunication. De plus, le fonctionnement du service SMS étant basé sur du **traitement par lot**, les messages ne sont pas envoyés immédiatement. L'attribut **dateDemande** renseigne donc le moment où l'utilisateur a validé le SMS à envoyer et **dateEnvoi** correspond alors au moment où le message est réellement envoyé par le service. Lorsqu'un SMS est envoyé par le service, une **référence** lui est assignée par le modem. Cette identifiant est le seul moyen de faire correspondre le message envoyé avec un potentiel accusé réception.

Un **statut** est associé à chaque message envoyé. Cela permet de connaître le statut d'un SMS à tout moment, s'il est en attente d'envoi, s'il a été envoyé, s'il a été accusé ou s'il y a eu des erreurs.

Un **message reçu** est caractérisé par une date de réception et une date de lecture qui correspond au moment où le message est lu pour la première fois.

5.2.3 Maquettage

Nous avons modélisé toutes les interfaces graphiques durant la conception afin de valider l'enchaînement des pages Web et leur disposition. Cependant, celles-ci ont été amenées à évoluer au cours du développement suite aux remarques du client concernant l'ergonomie. Cela nous a obligé à revoir une partie de la conception de nos pages Web.

Pour plus de détails se référer au rapport technique

5.3 Développement

5.3.1 Les commandes AT

Les commandes AT, aussi appelées commandes Hayes, permettent de communiquer avec le modem GSM et de lui envoyer des instructions. La communication avec le modem se fait via le port série (port COM) sur lequel est branché le modem SMS. Le framework .Net possède nativement des classes pour gérer la connexion sur les ports COM (classe SerialPort notamment). Une fois la connexion créée, on peut y écrire et lire des données avec les fonctions classiques Write, Read, ReadExisting, etc.

Il existe de nombreuses commandes AT, avec des utilités diverses, comprenant toutes de nombreux paramètres, souvent optionnels. Afin de prendre connaissances de toutes ces informations, nous disposons d'un document détaillant toutes les commandes AT et leurs différents paramètres. De ce document, nous **en** avons extrait les éléments essentiels et utiles.

Tout en acquérant cette théorie, nous avons construit un logiciel d'expérimentation en C# pour mettre en œuvre l'envoi et la réception de SMS via les commandes AT. Au fur et à mesure, ce logiciel en console s'est étoffé, et nous en avons finalement gardé la plupart des fonctionnalités pour en faire la classe métier du futur service SMS.

Vous trouverez en annexe plus de détails sur les commandes AT et la manière de les utiliser.

5.3.2 Le mode PDU et la librairie ATSMS

Un modem GSM peut fonctionner selon le mode texte ou le mode **PDU** (Protocol Data Unit). Ce mode détermine la syntaxe avec laquelle certaines commandes AT sont envoyées. Cependant, là où les deux modes se différencient le plus c'est lors de l'envoi d'un SMS.

Le mode **texte** permet d'envoyer plus facilement des messages car il utilise des commandes AT simples. En effet, il faut juste fournir au modem le numéro du destinataire et le contenu du message à travers une commande AT.

"+85291234567"

C'est simple d'envoyer un message en mode TEXTE.

Figure 5 - Informations nécessaires pour l'envoi d'un SMS en mode texte

Cependant, bien que ce mode soit plus simple d'utilisation, il supporte moins de fonctionnalités que le mode PDU. Par exemple, le mode texte ne gère pas les accusés réception, les durées de validité et les caractères spéciaux.

Le mode **PDU** est un peu plus complexe car il utilise des commandes AT contenant de l'*hexadécimal*^{viii}. De plus, lors de l'envoi d'un SMS en mode PDU il faut absolument spécifier certains paramètres comme l'encodage du message ou encore la longueur de la chaîne hexadécimal (Pour plus de détails, se référer au **rapport technique**).

42

```
07915892000000F001000B915892214365F7000021493A283D0795C3F33C88FE06CDCB6E32885
EC6D341EDF27C1E3E97E72E
```

Figure 6 – Informations nécessaires pour l'envoi d'un SMS en mode PDU



Comme le système d'envoi et de réception de SMS doit permettre aux utilisateurs de choisir un encodage et une durée de validité et également gérer des accusés réception, nous avons décidé d'envoyer des messages uniquement en mode **PDU**.

Lors de l'analyse de la structure des commandes **PDU** nous nous sommes aperçus que la mise en place d'algorithmes de création de commandes PDU allaient nous prendre beaucoup de temps. En effet, pour créer de telles fonctions de conversion, il faut d'une part maîtriser parfaitement le protocole **TPDU** (Transport Protocol Data Unit) et d'autre part mettre en place plusieurs calculs de conversion. Ainsi, nous nous sommes mis à la recherche d'une librairie permettant une création simple de trames **PDU**.

Ainsi, nous avons intégré la librairie **ATSMS** à notre projet. Cette librairie **est** issue d'un projet open source trouvé sur le site <http://www.csharpfr.com> **et** permet d'encoder rapidement des messages en trames PDU tout en spécifiant un encodage, une durée de validité et des accusés réception. De plus, étant donné que le code source est accessible, la maintenance et les évolutions sont rendus plus faciles.



Pour plus de détails sur la librairie **ATSMS**, se référer au **rapport technique**.

5.3.3 Le service SMS

Un service Windows n'est pas un programme classique qui s'exécute à la demande au premier plan, mais est lancé au démarrage de l'ordinateur et s'exécute de manière invisible. Il effectue des traitements automatiquement, sans que l'utilisateur n'ait besoin d'intervenir.

Le service a donc accès au modem via le port COM. On a réutilisé ici la classe que nous avons créée pour gérer le modem SMS ainsi que la librairie ATSMS. Lorsque le service est démarré, deux *timers*^{ix} sont lancés, l'un pour la lecture des messages et les accusés reçus, l'autre pour l'envoi des messages en attente.

Quand le timer d'envoi lance un évènement, le service récupère les messages en attente en base de données, convertit le message en PDU si ce n'est pas déjà une trame PDU, et effectue l'envoi du message via le modem.

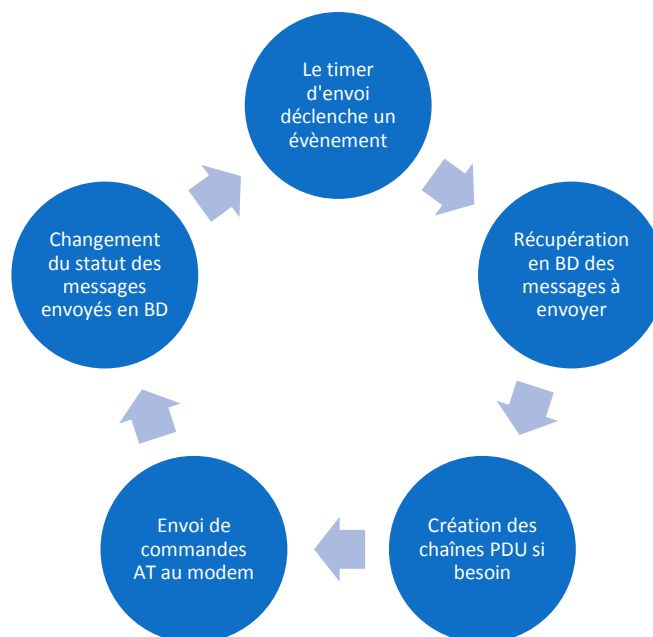


Figure 7 - Schéma descriptif de l'algorithme du timer de lecture

Quand le timer de lecture déclenche un évènement, on commence par récupérer les messages reçus par le modem. S'il y en a, on les insère dans la base de données. Ensuite, on vérifie dans la mémoire des accusés de réception si de nouveaux accusés sont arrivés. Si c'est le cas, on cherche dans la base de données les messages non accusés en ayant demandé un, et on compare les numéros de référence pour trouver le bon message et le mettre dans l'état « accusé reçu ». Enfin, le service vérifie le nombre de messages sur chaque mémoire, et vide celles qui sont presque pleines. En effet, si la mémoire est pleine, les nouveaux messages entrant ne seront pas enregistrés et seront donc perdus.

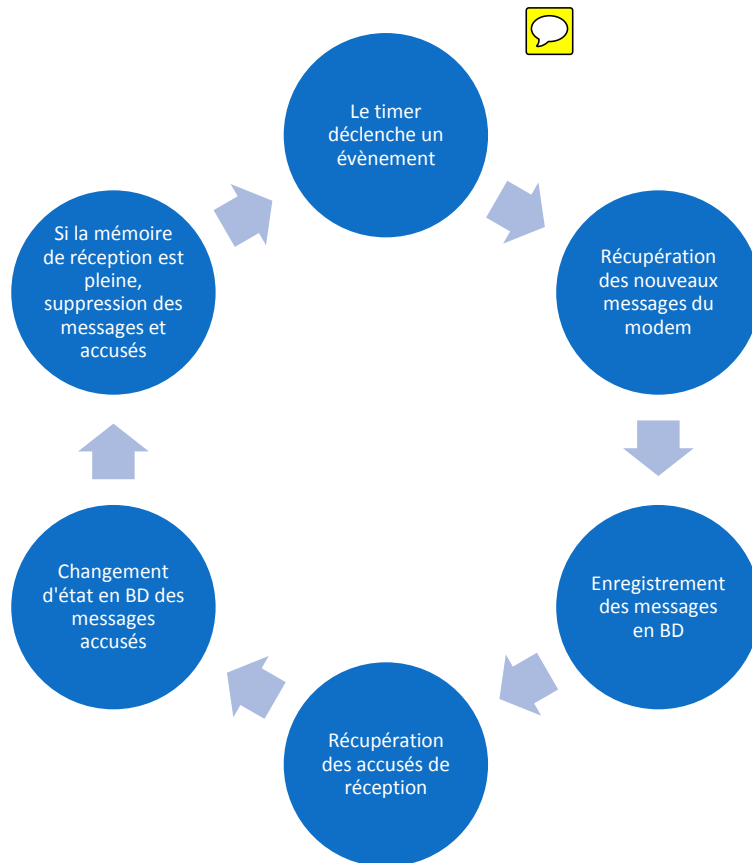


Figure 8 - Schéma descriptif de l'algorithme du timer de lecture

Un service ne peut pas se lancer comme un simple programme et doit être installé. Pour le déploiement, il faut donc ajouter au service un programme d'installation. Visual Studio 2008 permet de faire cela assez rapidement.

Cependant, le débogage n'est pas pratique de cette façon. Nous avons donc fait en sorte d'ouvrir en parallèle du service une console qui nous permettra de visualiser (en mode débogage uniquement) le déroulement du service.

5.3.4 La base de données

Nous avons installé la base de données sur le PC nous servant de serveur dans notre salle de projet. En effet, il fallait que celle-ci soit toujours accessible, même lorsque nous n'étions pas dans la salle. La configuration de SQL Server a pris un peu de temps, notamment pour le mode d'authentification. SQL Server propose par défaut s'identifier avec la session Windows, mais cela ne permet pas un accès à distance. Il a donc fallut paramétrer un compte SQL Server pour s'identifier à distance, à la place du compte Windows par défaut.

Une fois l'installation de SQL Server effectuée, nous avons installé le logiciel SQL Server Management Studio, permettant de créer plus rapidement qu'en SQL une base de données. Ce logiciel permet de gérer tous les aspects d'une base de données : sa mise en ligne hors ligne, sa structure, le contenu de ses tables, etc. L'installation n'est cependant pas intuitive est a donc pris un

certain temps. L'installation se déroule comme celle d'une nouvelle instance de SQL Server, ce qui nous a induit en erreurs.

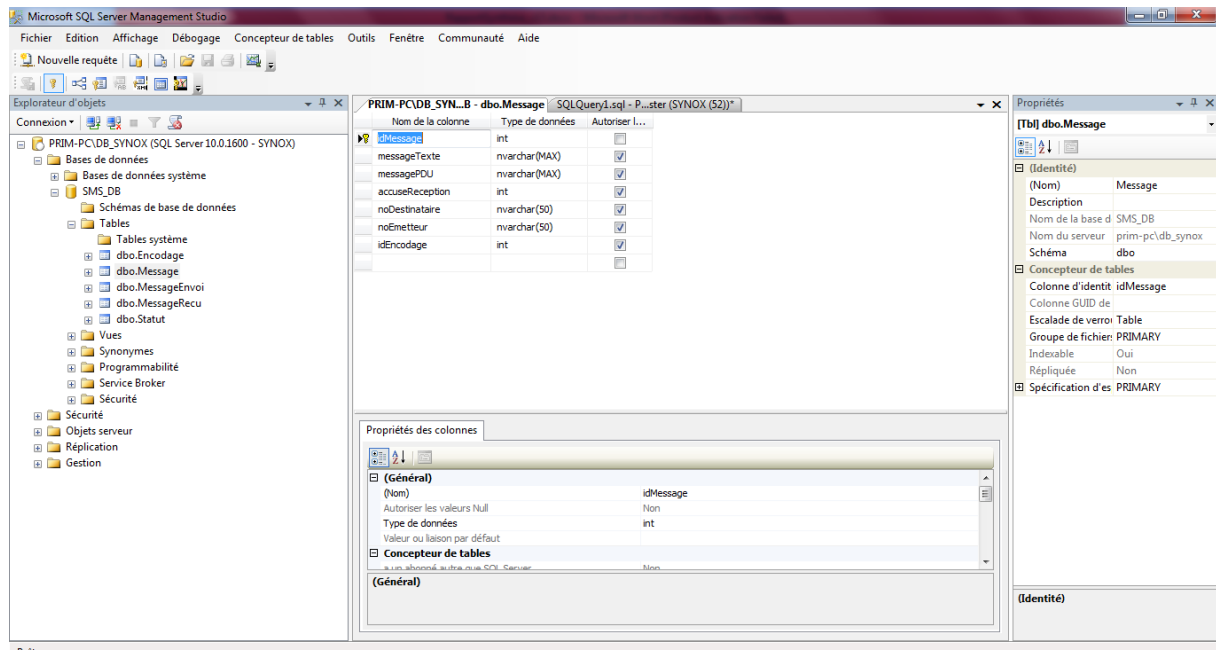


Figure 9 - SQL Server Management Studio : Structure de la table Message

Une fois la base de données créée, nous l'avons liée avec l'interface graphique et le service SMS avec LINQ to SQL. Ainsi, il est possible d'écrire directement dans le code C# des requêtes qui seront traduites et soumises à la base de données.

```
MessageEnvoi[] lesSMSAVerifier = (from msg in dbContext.MessageEnvoi
    where msg.Statut.libelleStatut == "Envoyé"
    && msg.Message.accuseReception == 1
    select msg).ToArray();
```

Figure 10 - Requête LINQ : SMS en attente d'accusé

LINQ to SQL a également l'avantage de créer automatiquement les classes métiers et d'y associer le résultat des requêtes, nous évitant ainsi d'avoir à instancier nous-mêmes les objets.

5.3.5 L'interface graphique

L'interface graphique doit permettre aux utilisateurs humains de la plateforme d'envoyer et de consulter des SMS. Elle communique avec le service SMS par l'intermédiaire de la base de données. Cette interface est sous forme d'un site web et se décompose en trois sous-interface : l'envoi des SMS, la consultation de la boîte d'envoi et la consultation des messages reçus.

5.3.5.1 L'envoi de SMS

L'interface d'envoi de SMS contient un formulaire permettant d'envoyer un message en lui remplissant simplement un formulaire.

La **date de validité** correspond à la période durant laquelle le message reste valide. Le destinataire ne reçoit pas le message dont la durée de validité est expirée.

Il est également possible d'envoyer directement une **trame PDU**.

Choix du mode SMS

Mode : Texte

Ecriture du SMS

Numéro du destinataire :

Encodage :

7bits ☐ Demander un accusé de réception

Date de validité :

jours heures minutes

Message à envoyer :

Valider

Figure 11 - Capture d'écran de l'interface d'envoi de SMS

Lorsque l'utilisateur clique sur le bouton « **Envoyer** », les informations du message à envoyer sont sauvegardées dans la base de données. Le service SMS, qui consulte régulièrement cette même base de données, crée une chaîne **PDU** d'après les informations sauvegardées en base de données puis envoie une commande AT au modem afin d'envoyer le SMS.

5.3.5.2 La boîte d'envoi

La boîte d'envoi permet de consulter les messages envoyés et leur statut. Ainsi, l'utilisateur peut connaître l'état des SMS, c'est-à-dire s'ils ont été envoyés ou s'ils ont été accusés.

SMS envoyés

Search

Destinataire	Message	Date Demande Envoi	Date Envoi	Statut	Action
0675610118	yre	24/01/2012 14:06:54	24/01/2012 14:09:43	Accuse	
0675610118	test accuse	24/01/2012 11:17:02	24/01/2012 11:17:10	Accuse	
0675610118	dftetghygghgh	23/01/2012 17:25:31	23/01/2012 17:25:45	Accuse	
0675610118	drtldhghgdgghgh	23/01/2012 17:23:46	23/01/2012 17:24:05	Envoye	
0625123338	\$^ú\$^^\$ûâ* % \$ *\$ +=<>	23/01/2012 17:23:42	23/01/2012 17:23:50	Accuse	
0625123338	jij jipv jvzp po à i ào iv ...	23/01/2012 17:21:57	23/01/2012 17:22:23	Accuse	
0625123338	jij jipv jvzp po à i ào iv ...	23/01/2012 17:21:56	23/01/2012 17:22:18	Envoye	
0625123338	spaaaaaaaaaaaaaaaaaaaaaaa...	23/01/2012 17:14:54	23/01/2012 17:15:04	Accuse	
	Trame PDU (Cliquer pour plus de details)	23/01/2012 17:05:34		Erreur	
	Trame PDU (Cliquer pour plus de details)	23/01/2012 17:03:32	23/01/2012 17:03:41	Envoye	

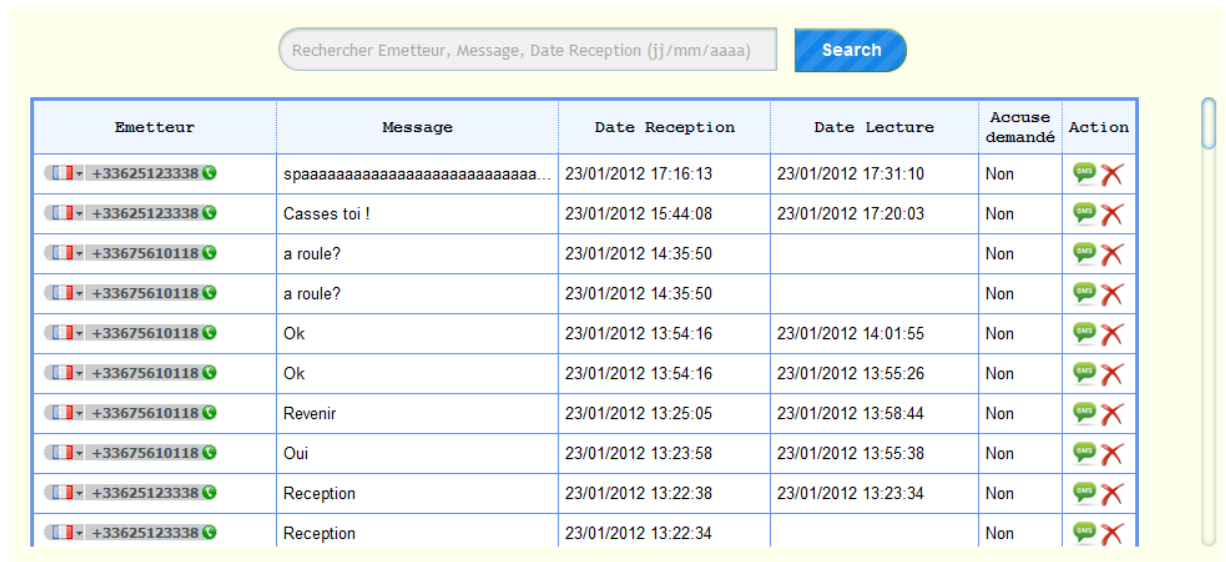
Figure 12 - Capture d'écran de la boîte d'envoi

La barre de recherche permet de filtrer les messages selon le destinataire, le contenu du message, la date d'envoi ou le statut du SMS. Les informations de cette interface et les résultats de la recherche sont directement issue de la base de données.

Le tableau contenant les messages se recharge automatiquement toutes les minutes afin de s'assurer que les données présentées aux utilisateurs soient à jour.

5.3.5.3 La boîte de réception

La boîte de réception permet de consulter les messages reçus par le service SMS en effectuant une lecture dans la base de données.



Rechercher Emetteur, Message, Date Reception (jj/mm/aaaa) **Search**

Emetteur	Message	Date Reception	Date Lecture	Accuse demandé	Action
+33625123338	spaaaaaaaaaaaaaaaaaaaaaaaaa...	23/01/2012 17:16:13	23/01/2012 17:31:10	Non	
+33625123338	Casses toi !	23/01/2012 15:44:08	23/01/2012 17:20:03	Non	
+33675610118	a roule?	23/01/2012 14:35:50		Non	
+33675610118	a roule?	23/01/2012 14:35:50		Non	
+33675610118	Ok	23/01/2012 13:54:16	23/01/2012 14:01:55	Non	
+33675610118	Ok	23/01/2012 13:54:16	23/01/2012 13:55:26	Non	
+33675610118	Revenir	23/01/2012 13:25:05	23/01/2012 13:58:44	Non	
+33675610118	Oui	23/01/2012 13:23:58	23/01/2012 13:55:38	Non	
+33625123338	Reception	23/01/2012 13:22:38	23/01/2012 13:23:34	Non	
+33625123338	Reception	23/01/2012 13:22:34		Non	


Figure 13 - Capture d'écran de la boîte de réception

L'utilisateur a la possibilité de supprimer des messages et de répondre à l'émetteur.

Tout comme l'interface de consultation des messages envoyés, nous avons implémenté une barre de recherche permettant de filtrer les messages et un système de rafraîchissement automatique de la boîte de réception.

5.4 Tests

Tout au long du développement, nous avons effectué **plusieurs niveaux de tests**. En premier lieu, chaque fonction a été testée individuellement, afin de vérifier qu'elle retourne les résultats attendus, et qu'elle traite correctement les cas d'erreurs et les cas limites.

Nous avons également fait d'importants **tests de fiabilité et d'efficacité** sur le service SMS et le modem GSM. En effet, le service SMS est amené à devoir s'exécuter sur de longues durées sans intervention humaine possible en cas de problème. Il faut donc que le service SMS ne s'arrête pas inopinément pour une raison ou une autre, et doit redémarrer en cas de problème. Nous avons donc laissé le script s'exécuter des jours  urant, ce qui nous a permis de mettre en évidence un problème récurrent : la communication **entre** la base de données est régulièrement rompue. Nous avons alors mis en place un **système relançant automatiquement la connexion et reprenant l'opération qui était en cours** quand le problème est survenu.

Nous avons réutilisé ce système de reprise sur erreurs pour tous les problèmes plus rares liés au réseau, comme les réponses incomplètes que renvoient parfois le modem.

Nous avons également voulu tester la capacité du modem à traiter de nombreux messages en même temps, que ce soit en réception ou en envoi, avec ou sans accusés de réception. C'est ainsi que nous avons mis en évidence les limites du modem, développées dans la partie 5.5.

L'interface graphique et les algorithmes liés ont aussi été longuement testés. Par exemple, en vérifiant le comportement de l'application lorsque l'utilisateur rentre des données aberrantes (numéro de téléphone contenant des lettres, durée de validité négative, etc.).

5.5 Limites



Cette solution, telle que nous l'avons conçu selon les demandes du client et les contraintes matérielles, possède cependant certaines limites.

La première concerne l'interface. Celle-ci ne possède en effet aucun système de session et est donc mono-utilisateur. Si cette interface est amenée à être utilisée par plusieurs personnes, il serait alors indispensable de mettre en place un tel système, en n'affichant que les messages destinées et envoyées à ou par une certaine personne.

La seconde limite se situe au niveau du modem, qui n'est pas toujours capable de gérer correctement une importante quantité de messages. Il arrive régulièrement que certains messages soient envoyés en double, ou même jamais transmis au destinataire. Mais les problèmes les plus récurrents concernent les accusés de réception, qui n'arrivent que rarement lorsqu'ils sont nombreux à être attendus.

6 Améliorations possibles

6.1 Permettre une utilisation multi-utilisateurs

Le service SMS et la base de données ne permettent pas actuellement une utilisation avec plusieurs utilisateurs.

Pour implémenter une telle fonctionnalité, il faut mettre en place un système d'identifiants unique dans la base de données afin de pouvoir identifier les utilisateurs.

Au niveau du service SMS, il faut que les messages reçus par le modem puissent être identifiés et être redirigés vers les utilisateurs concernés. Pour cela, il faudrait que l'identifiant unique de l'utilisateur figure dans le contenu du SMS reçu.

6.2 Sécuriser l'accès à la base de données

L'interface graphique communique directement avec la base de données pour y insérer les messages à envoyer et pour consulter les SMS. Pour rappel, l'interface créée n'est qu'un exemple afin de montrer les possibilités liées au service SMS. Cependant, il est aussi possible qu'une application puisse interagir directement avec la base de données pour envoyer ou recevoir des SMS.

Or, avec le modèle de fonctionnement actuel où les utilisateurs ont un accès total à la base de données, des problèmes peuvent survenir surtout en mode multi-utilisateurs. En effet, si les modifications de la base de données ne sont pas contrôlées cela peut créer des incohérences ou des pertes d'informations. Par exemple, actuellement, il n'y a rien qui empêche un utilisateur de supprimer tous les messages de la base de données, même ceux ne le concernant pas.

Afin d'éviter de telles problèmes, il faut mettre en place un **web service** qui garantit un accès à distance contrôlé à la base de données à travers des fonctions.

6.3 Implémenter une gestion des contacts

Afin de pouvoir identifier rapidement l'auteur d'un message reçu, il serait intéressant d'implémenter une gestion des contacts par rapport à chaque utilisateur. Ainsi, l'interface utilisateur affichera le nom du contact à la place de son numéro de téléphone. Cette implémentation concerne surtout l'interface graphique et la base de données.

7 Difficultés rencontrées

7.1 Apprentissage des technologies

Pendant ce projet, nous avons été amenés à manipuler des technologies que nous ne connaissions pas.

Tout d'abord, il a fallu que nous comprenions l'intérêt et la syntaxe des différentes commandes AT afin de cerner celles que nous devons utiliser afin de communiquer avec le modem GSM.

De plus, même si nous avons déjà réalisé des projets avec le Framework .Net, nous n'avions jamais mis en place un protocole de communication avec cette technologie. De même, l'aspect Web de ce Framework nous était inconnu.

L'utilisation de ces différentes technologies a induit que nous devions assimiler rapidement leur fonctionnement afin d'être productif le plus tôt possible.

7.2 Problèmes de compatibilité entre le Framework .Net et le JavaScript


Lors de la réalisation de l'interface graphique, nous avons été confrontés à plusieurs problèmes au niveau de la compatibilité entre la partie Web du Framework .Net et le *JavaScript*^x.

Ce problème de compatibilité vient surtout du fait que le Framework .Net possède ses propres composants Web, comme par exemple les champs de saisie ou les boutons. Leur utilisation facilite la gestion des événements et la récupération des données depuis les champs de saisie.

Cependant, les composants du Framework .Net n'ont pas la même structure que les composants HTML. Or, le JavaScript est conçu pour fonctionner avec la structure des objets HTML.

Ainsi, dans certains cas le code JavaScript que nous avons écrit ne marchait pas. Par exemple, lorsque nous avons voulu améliorer le rendu visuel du formulaire d'envoi avec la librairie externe JQuery écrite en JavaScript, le gestionnaire d'erreurs nous indiquait que des éléments de la page étaient introuvables.

7.3 Les accusés de réception

Pendant l'élaboration du service SMS, nous avons rencontré des  difficultés avec la réception des accusés. En effet, il arrivait que le modem GSM ne reçoive aucun **accusés** pendant quelques heures avant de reprendre un fonctionnement normal.

Après une longue série de tests nous nous sommes aperçus que le problème a deux origines distinctes. La première concerne une mauvaise configuration du modem GSM qui n'est alors pas configuré pour recevoir des accusés. La deuxième origine provient du fait que lorsque le modem reçoit un trop grand nombre d'accusés de réception, il arrive qu'il y ait une saturation et que certains accusés soient perdus.

8 Conclusion

Ce projet de deux mois a été **riche en expériences** sur plusieurs niveaux. Tout d'abord, cette mission était pour nous un certain défi sur le plan technique. Nous n'avions en effet que peu d'expérience dans le domaine du réseau, et nous avons également eu l'occasion d'approfondir nos connaissances du Framework .Net.

De même, nous avons pu tester notre aptitude à être responsables de projet, c'est-à-dire estimer les charges des différentes phases, gérer nos plannings et prendre des décisions. C'est réellement la meilleure façon de comprendre un tel métier et d'apprendre de nos erreurs.

Par la suite, ce projet nous a également permis de développer notre aptitude à travailler en équipe. Nous avons donc pu **mettre en application** nos capacités d'écoute, d'ouverture d'esprit, de sens relationnel ainsi que d'organisation et d'analyse.

Nous pouvons dire que nous avons **atteint les objectifs définis** dans la lettre de mission. En outre, nous avons identifié quelques améliorations qu'il serait intéressant de mettre en œuvre afin de perfectionner le système.

Les technologies mobiles sont très demandées aujourd'hui avec la forte présence d'objets mobiles communicants comme les smartphones ou les machines embarquées. Il nous semble que ce domaine prend une place de plus en plus importante dans l'informatique, et qu'il sera inévitable pour nous, en tant que futurs ingénieurs en informatique, de s'y intéresser afin de suivre l'évolution technologique.

Finalement, que ce soit sur nos réussites ou nos difficultés, ce projet industriel reste une étape enrichissante dans cette dernière année de cycle d'ingénieur.

9 Annexes

9.1 Les commandes AT

9.1.1 Choix du format des messages

La commande **AT+CMGF** permet d'indiquer au modem le format des messages. En effet, il existe 2 formats possibles : le format texte, utilisant les caractères ASCII, et le format PDU, en hexadécimal (caractères de 0 à 9 et de A à F).

En mode texte, les informations sont directement compréhensibles par l'utilisateur, alors que le mode PDU nécessite un décodage. Par ailleurs, le mode PDU offre plus de possibilités que le mode texte.

9.1.2 Choix de la mémoire

La commande **AT+CPMS** permet de choisir la mémoire avec laquelle le modem communique. Il y a 3 types de mémoires :

- SM : Les messages stockés sur la carte SIM
- SR : La mémoire des accusés de réceptions, sur la carte SIM, limité à 50 accusés
- ME : La mémoire Flash

9.1.3 Affichage des messages

La commande **AT+CMGL** permet d'afficher tous les messages stockés sur la mémoire choisit avec la commande AT+CPMS. Ainsi, si la mémoire SM est sélectionnée, la commande AT+CMGL affiche les messages de la carte SIM. Un paramètre permet de spécifier l'état des messages demandés (reçus, lus et/ou non lus, tous).

9.1.4 Envoi d'un message

La commande **AT+CMGS** permet d'envoyer un message. La procédure change selon le mode choisit avec la commande AT+CMGF.

En mode texte, il faut d'abord spécifier le numéro du destinataire, puis entrer le message en langage « humain ». En mode PDU, on spécifie d'abord la longueur de la trame PDU, puis la trame PDU elle-même. Le numéro du destinataire, entre autres informations, est déjà compris dans la trame et sera utilisé par le modem.

9.2 Envoyer et recevoir des données

Afin d'envoyer des commandes AT et de recevoir des réponses avec le modem, il faut mettre en place un protocole de communication.

Le modem SMS communique avec le service SMS par l'intermédiaire d'un **port série**. Nous avons ainsi utilisé des fonctions du Framework .Net qui nous permettent d'envoyer et de réceptionner des données simplement sur un port de communication.

Le protocole de communication que nous avons mis en place repose sur quatre fonctions :

- Connexion au modem
- Envoi de données
- Réception de données
- Déconnexion

Chaque commande AT envoyé au modem fait appel à ces quatre fonctions.

9.2.1 Connexion au modem

Avant de pouvoir envoyer des commandes AT au modem, il faut d'abord initialiser une connexion avec le modem. Pour cela, nous avons utilisé la classe **SerialPort** du Framework .Net qui nous permet d'instancier un objet représentant la liaison avec le modem.

```
PortCom = new SerialPort()  
{  
    PortName = serialPortName, //no de port, ex : COM11  
    BaudRate = 115200, //vitesse a laquelle on envoie/reçoit les donnees (bps)  
    DataBits = 8, //longueur standard des bits de données par octet  
};
```

Figure 14 - Extrait de l'algorithme de connexion au modem GSM

Lors de l'établissement de la connexion, nous devons spécifier le **numéro** du port utilisé, la **vitesse** à laquelle les données sont échangées et la **quantité** d'informations contenu dans chaque octet.

Une fois un objet de la classe **SerialPort** instancié, il suffit d'appeler la méthode «**Open()**» sur cette objet pour établir une connexion avec le modem.

9.2.2 Envoi de données au modem

L'envoi de données au modem se fait exclusivement à travers des **commandes AT**. Il faut qu'une connexion soit préalablement établie auprès du modem.

Pour transmettre des commandes, il suffit d'écrire des octets sur l'objet **SerialPort** à l'aide des données d'une **mémoire tampon**.

```
//buffer : Tableau d'octets qui contient les données à écrire sur le port.  
//offset : 0, index du buffer partir duquel commencer la copie des octets vers le port  
//buffer.length : Nombre d'octets à écrire sur le port  
PortCom.Write(buffer, 0, buffer.Length);
```

Figure 15 - Extrait de l'algorithme d'envoi de commandes au modem GSM

Cette mémoire tampon permet d'éviter de saturer le modem si la commande de sortie est trop longue. En effet, si le nombre d'octets dans la mémoire tampon de sortie est trop important l'objet **SerialPort** va attendre que le périphérique soit prêt à accepter plus de données.

9.2.3 Réception de données

Pour recevoir des données du modem, il faut préalablement avoir envoyé des données au périphérique GSM. En effet, les données envoyées par le modem correspondent uniquement à des réponses de **commandes AT**.

La réception de données s'effectue en lisant continuellement les données présentes sur la mémoire tampon du modem jusqu'à que le message reçu soit complet.

```
//si le signal comme quoi on a reçu un message est active dans les 10 prochaines secondes
if (receiveNow.WaitOne(10000, false))
{
    //on recupere les donnees presentes sur le port serie
    string message = PortCom.ReadExisting();

    //on l'ajoute a la reponse car on n'est pas sur que le message soit complet
    response += message;
}
else
{
    //message incomplet ...
}
}
while (!response.EndsWith("\r\nOK\r\n") && !response.EndsWith("\r\nERROR\r\n"));
```

Figure 16 - Algorithme de réception des messages du modem GSM

Un message est considéré comme complet lorsqu'il se termine par « **OK** » ou « **ERROR** ».

La fonction **receiveNow.Wait** retourne vrai si le modem reçoit des données avant que le **timeout** spécifié n'expire. Son utilisation a pour but d'éviter une attente active et également d'empêcher un blocage du programme en cas de réponse incomplète.

9.2.4 Déconnexion du modem

Lorsque le modem n'est plus utilisé par le service, il faut le déconnecter pour éventuellement permettre à d'autre programme d'utiliser cette ressource. Pour cela, il faut tout simplement appeler la fonction « **Close()** » sur l'objet **SerialPort** instancié lors de la connexion.

9.2.5 Exemple d'utilisation du protocole de communication

Le schéma ci-dessous montre un exemple de l'utilisation du protocole de communication lorsque la commande « **AT+CPMS ?** » est envoyée au modem. Cette commande permet de connaître le type de mémoire sur laquelle les messages sont stockés.

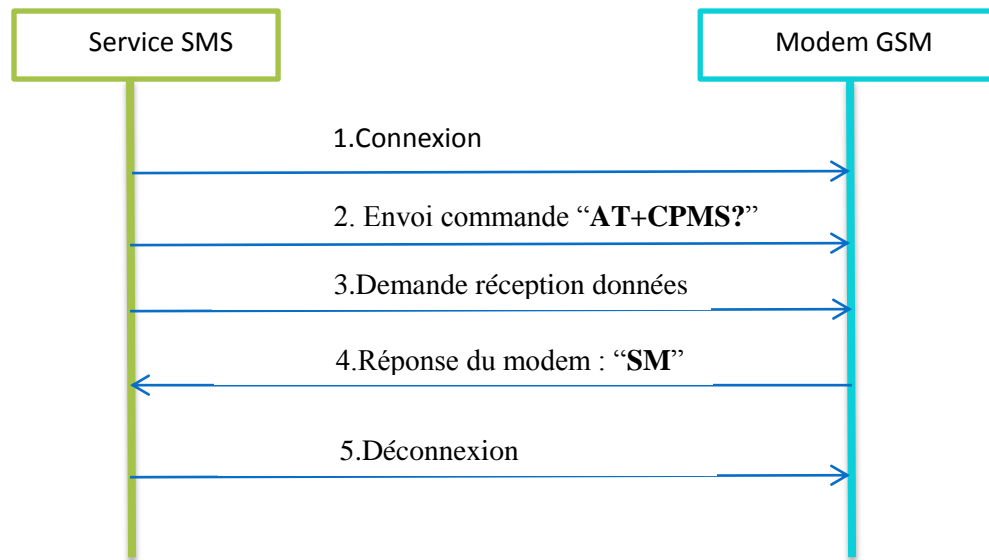


Figure 17 - Protocole de communication entre le service SMS et le modem GSM

« **SM** » signifie que les messages sont stockés sur la carte SIM du modem.

9.3 Diagramme de Gantt

N°	Mode Tâche	Nom de la tâche	Durée	Début	Fin	Prédécesseurs
1		Projet Industriel	30,36 jours	Lun 05/12/11	Ven 03/02/12	
2		Familiarisation aux technologies	4 jours	Lun 05/12/11	Ven 09/12/11	
3		Familiarisation avec les commandes AT	2 jours	Lun 05/12/11	Mar 06/12/11	
4		Tutoriel ASP.NET	1 jour	Mar 06/12/11	Jeu 08/12/11	3
5		Création d'une connexion entre .Net et le modem GSM	1 jour	Jeu 08/12/11	Ven 09/12/11	4
6		Analyse	1,5 jours	Ven 09/12/11	Mar 13/12/11	
7		Cas d'utilisation	1 jour	Ven 09/12/11	Lun 12/12/11	
8		Réalisation de maquettes	0,5 jour	Lun 12/12/11	Mar 13/12/11	7
9		Diagramme de classe métier	0,5 jour	Lun 12/12/11	Mar 13/12/11	7
10		Partie 1 : envoi de SMS	2 jours	Mar 13/12/11	Jeu 15/12/11	6
11		Sélection des commandes AT nécessaires	1 jour	Mar 13/12/11	Mer 14/12/11	
12		Création d'un petit logiciel d'envoi de SMS	1 jour	Mer 14/12/11	Jeu 15/12/11	11
13		Partie 2 : Réception des SMS	2 jours	Jeu 15/12/11	Lun 09/01/12	10
14		Sélection des commandes AT nécessaires	1 jour	Jeu 15/12/11	Ven 16/12/11	
15		Adaptation du logiciel d'envoi afin de réceptionner des SMS	1 jour	Ven 16/12/11	Lun 09/01/12	14
16		Partie 3 : Service Windows et interface graphique	12,93 jours	Mar 10/01/12	Mer 25/01/12	13
17		Service Windows	9,71 jours	Mar 10/01/12	Ven 20/01/12	
18		Création et intégration de la base de données	3,29 jours	Mar 10/01/12	Jeu 12/01/12	
19		Intégration des fonctions issues du logiciel d'envoi et de réception	3,29 jours	Mar 10/01/12	Jeu 12/01/12	
20		Mise en place d'un protocole de communication	6,5 jours	Ven 13/01/12	Ven 20/01/12	
21		Interface graphique	9,71 jours	Ven 13/01/12	Mer 25/01/12	
22		Elaboration de l'interface d'envoi	3,29 jours	Lun 16/01/12	Mer 18/01/12	
23		Elaboration de la boîte d'envoi	3,29 jours	Jeu 19/01/12	Lun 23/01/12	
24		Elaboration de la boîte de réception	2,21 jours	Mar 24/01/12	Mer 25/01/12	
25		Rédaction rapports	7,57 jours	Jeu 26/01/12	Ven 03/02/12	16

Figure 18 - Diagramme de Gantt (tableau)

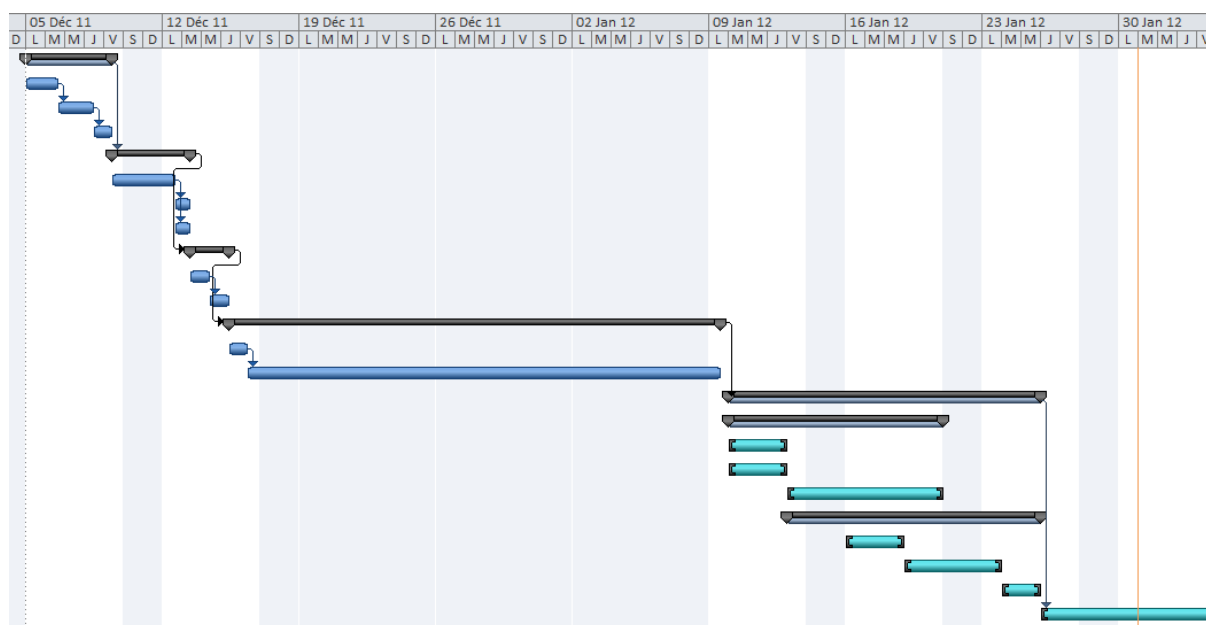


Figure 19 - Diagramme de Gantt (schéma)

Glossaire

ⁱ **Objets distants :**

Objets avec un système embarqué communiquant avec la plateforme. Ils envoient des informations qui seront interprétées par des applications Synox ou des utilisateurs.

ⁱⁱⁱ **GSM – Global System for Mobile Communications :**

Réseau de deuxième génération pour la téléphonie mobile.

^{iv} **SMS – Short Message Service :**

Service permettant de transmettre de courts messages sur le réseau GSM.

^v **Modem – Modulateur-démodulateur :**

Périphérique servant à communiquer avec des utilisateurs distants.

^{vi} **Service Windows :**

Logiciel s'exécutant au démarrage de l'ordinateur, en arrière-plan, ne nécessitant pas d'intervention de l'utilisateur.

^{vii} **Framework.Net :**

Environnement de travail Microsoft, fournissant des fonctions permettant de concevoir des applications lourdes ou légères. Cet environnement possède un compilateur intégré afin de pouvoir tester et déboguer ces applications.

^{viii} **Hexadécimal :**

Système de numération en base 16. Il utilise 16 symboles : les chiffres de 0 à 9 et les 6 premières lettres de l'alphabet, A à F.

^{ix} **Timer :**

Chronomètre déclenchant un évènement à chaque intervalle de temps (défini)

^x **JavaScript :**

Langage de script qui permet des interactions avec les différents objets d'une page Web