

Rapport de projet développement du
réalisé par : Brut Guillaume, Nunesse
William, Sorrenti Victor et Rochette
Martin.

RAPPORT DE PROJET DEVELOPPEMENT

CESI

Table des matières

Table des Illustrations	1
I. Introduction :.....	2
II. Rappel du besoin :.....	2
III. Découpage du projet :.....	3
IV. Planification initiale :.....	4
a. Répartition des tâches :.....	4
V. Modélisation de l'application distribuée :.....	5
VI. Analyse des écarts :.....	14
VII. Analyse des compétences acquises par étudiants :.....	16
VIII. Bilan :.....	16

Table des Illustrations (toutes les illustrations seront disponible dans le fichier de rendu)

Figure 1 - Découpage du Projet.....	3
Figure 2 - Planning prévisionnel	4
Figure 3 - Architecture logicielle globale.....	5
Figure 4 - Architecture physique globale	6
Figure 5 - Diagramme d'activité	6
Figure 6 - Diagramme des composants	7
Figure 7 - Diagramme de Classe Client Lourd.....	8
Figure 8 - Diagramme de classe Serveur .NET	8
Figure 9 - Diagramme de classe Serveur JEE	9
Figure 10 - Diagramme de séquence Client Lourd	10
Figure 11 - Diagramme de séquence Serveur .NET	11
Figure 12 - Diagramme de séquence de la librairie NMSLib	12
Figure 13 - Diagramme de séquence du serveur JEE	13
Figure 14 - Planning réel.....	14
Figure 15 - Tableau récapitulatif des écarts	15

I. Introduction :

Suite aux choix de la spécialisation développement de 4^{ème} année, les étudiants ont à réaliser un projet de développement traitant sur les notions vues en cours lors des différents blocs. Ce projet nécessite une équipe d'expert .NET (C#/WCF/SQL Server) et d'expert Java(EJB/JMS/web services/Oracles).

II. Rappel du besoin :

Ce projet implique la création d'une plateforme logiciel permettant à un utilisateur de s'authentifier, de déchiffrer des fichiers cryptés et d'obtenir le fichier cible en clair ainsi que la clé. La plateforme doit être composée et codée dans différents langages (.Net et Java).

La plateforme .Net se charge de :

- Permettre l'authentification d'un utilisateur
- Permettre la réception de fichier crypté
- Générer une série de clés utiles au déchiffrement
- Déchiffrer les fichiers cryptés avec les clés
- Envoyer les fichiers décryptés à la plateforme JEE
- Générer un PDF avec les retours de la plateforme JEE
- Envoyer un mail à l'utilisateur avec le PDF

La Plateforme JEE se charge de :

- Réceptionner les fichiers décryptés
- S'assurer que les fichiers sont correctement décryptés
- Chercher l'information secrète
- Faire un retour à l'application .NET (Nom du fichier, information secrète et clé)

III. Découpage du projet :

Ce projet peut être découpé de la manière suivante :

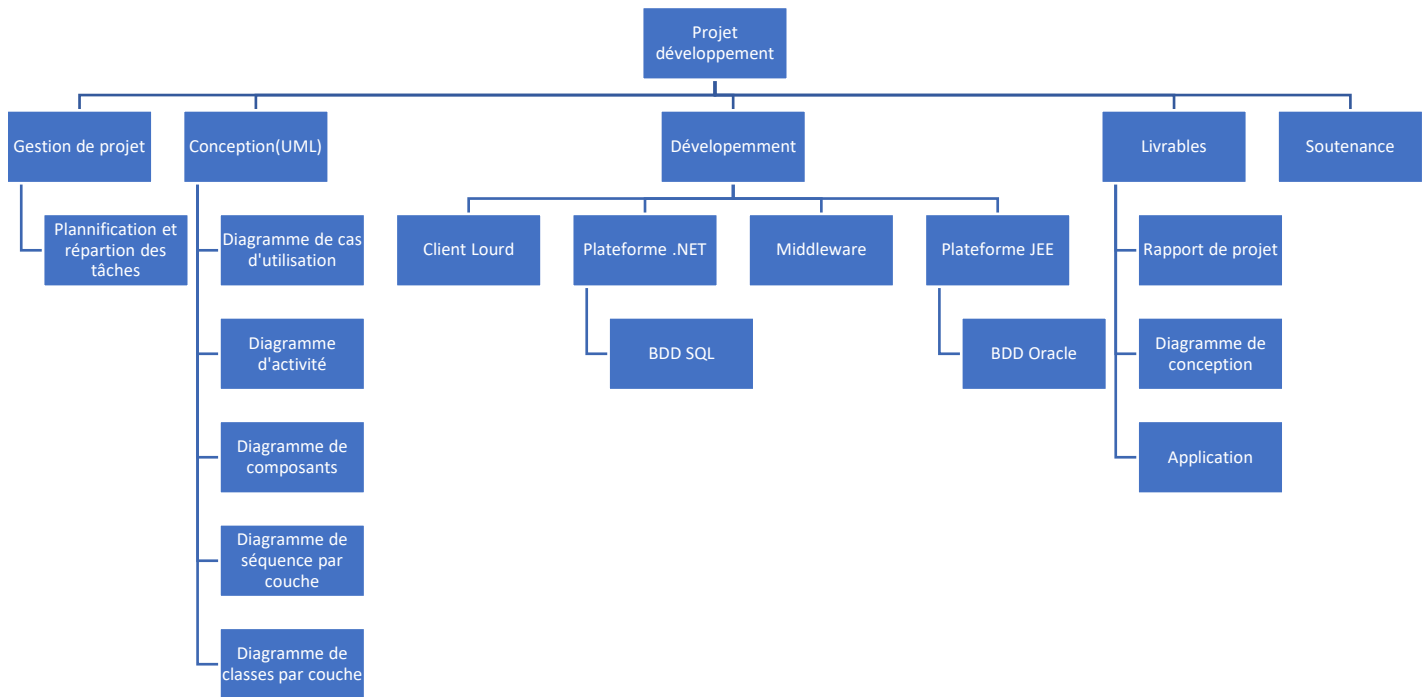


Figure 1 - Découpage du Projet

IV. Planification initiale :

Après un brainstorming nous avons conclu la planification prévisionnelle suivante :

CESI	Projet Dev du 22/06/2020 au 03/07/2020				Semaine 1								Semaine 2					
Tâche		Date de début	Date de fin	Durée	22	23	24	25	26	27	28	29	30	1	2	3		
Martin Rochette																		
1.UML		22-juin	24-juin	24h														
2. Serveur .NET																		
2a.BDD SQL																		
3. Serveur JEE		25-juin	01-juil	56h														
3a. BDD Oracle		25-juin	25-juin	4h														
4. Client Lourd																		
5. Middleware																		
6. Préparation Soutenance		02-juil	03-juil	12h														
William Nunesse																		
1.UML		22-juin	24-juin	24h														
2. Serveur .Net		25-juin	01-juil	56h														
2a.BDD SQL		25-juin	25-juin	4h														
3. Serveur JEE																		
3a.BDD Oracle																		
4. Client Lourd																		
5. Middleware																		
6. Préparation Soutenance		02-juil	03-juil	12h														
Guillaume Brute																		
1.UML		22-juin	24-juin	24h														
2. Serveur .Net																		
2a. BDD SQL																		
3. Serveur JEE																		
3a. BDD Oracle																		
4. Client Lourd																		
5. Middleware		25-juin	01-juil	56h														
6. Préparation Soutenance		02-juil	03-juil	12h														
Victor Sorrenti																		
1.UML		22-juin	24-juin	24h														
2. Serveur .Net																		
2a. BDD SQL																		
3. Serveur JEE																		
3a. BDD Oracle																		
4. Client Lourd		25-juin	30-juin	40h														
5. Middleware																		
6. Livrables		30-juin	03-juil	16h														
7. Préparation Soutenance		02-juil	03-juil	10h														

Figure 2 - Planning prévisionnel

a. Répartition des tâches :

En suivant la planification des tâches, nous avons donc réparti les tâches comme ceci :

- Conception du projet tous ensemble (Brainstorming, Diagrammes) ;
- Martin s'occupe du développement du serveur JEE ;
- William s'occupe du développement du serveur .NET ;
- Guillaume s'occupe du développement du Middleware ;
- Victor s'occupe du développement du client lourd ;

V. Modélisation de l'application distribuée :

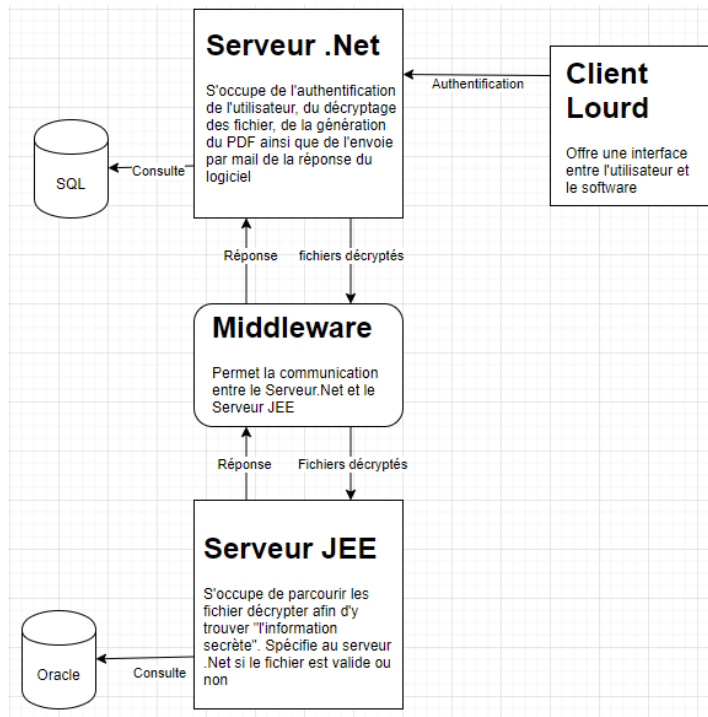


Figure 3 - Architecture logicielle globale

Ce diagramme représente l'architecture logiciel de notre plateforme. Elle permet de mettre en évidence l'architecture de cette dernière.

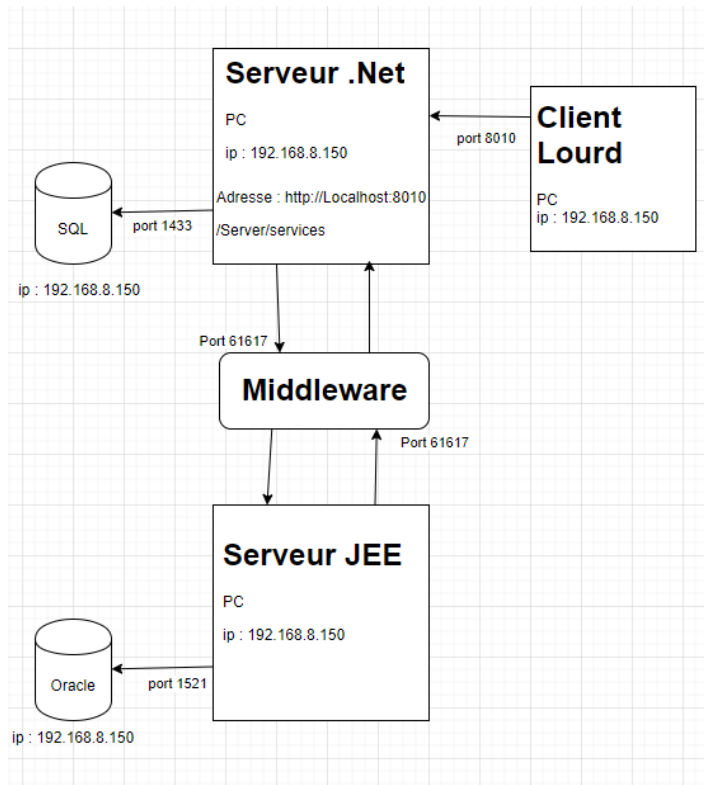


Figure 4 - Architecture physique globale

Ce diagramme représente l'architecture physique que notre plateforme. Il permet de spécifier comment sont répartie les serveurs sur les machines ainsi que leur manière de communiquer. On remarquera donc que l'architecture est bien une architecture 3tiers grâce au Serveur JEE déporté au client lourd déporté.

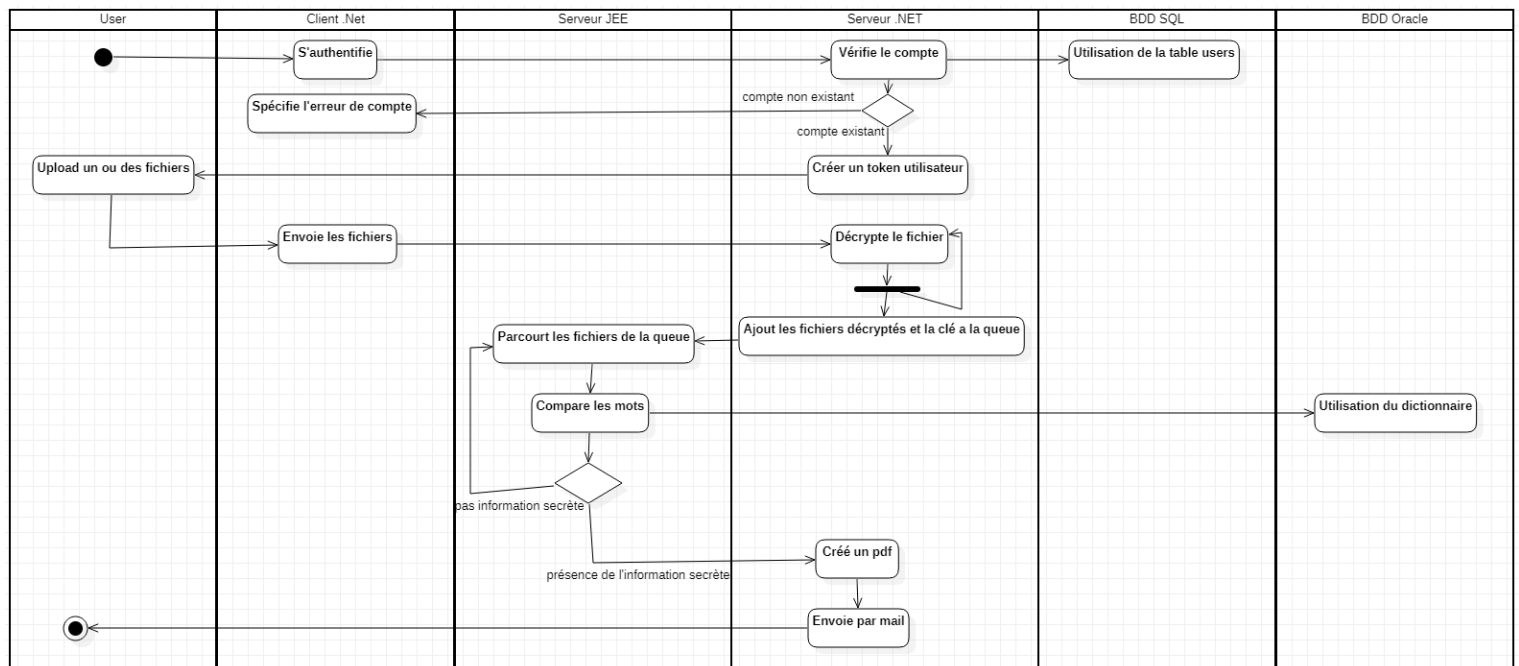


Figure 5 - Diagramme d'activité

Ce diagramme représente l'activité notre plateforme. Elle permet de mettre évidence les différentes interactions entre les différentes composantes de notre plateforme.

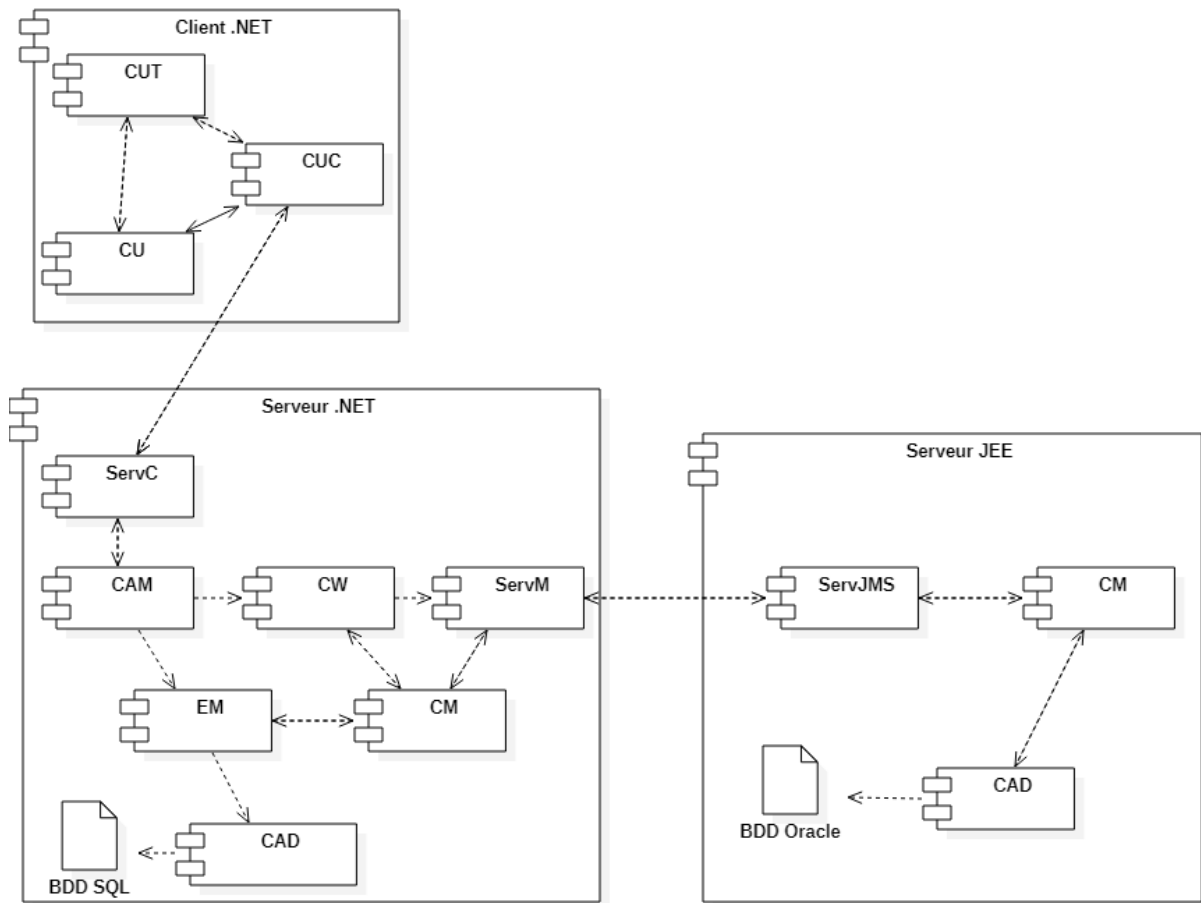


Figure 6 - Diagramme des composants

Ce diagramme des composants représente les différentes couches de notre plateforme. On remarque donc que le Serveur .Net est composé de 6 couches et le Client Lourd de 3 Couche, ce qui nous donne une architecture 9 couche. Il permet aussi au prévoir les différents package à créer lors du développement des plateformes.

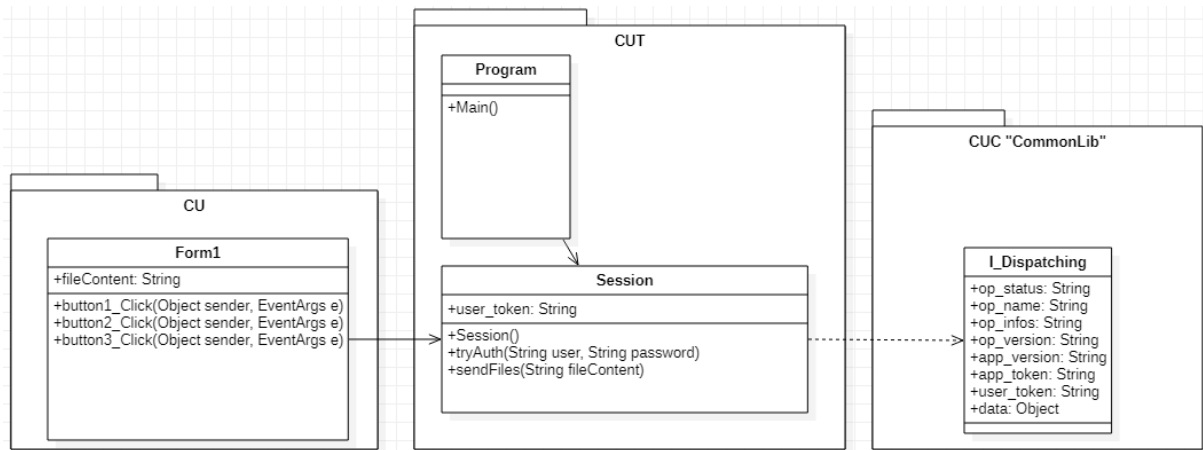


Figure 7 - Diagramme de Classe Client Lourd

Ce diagramme représente les classes de notre Client Lourd. Il permet de mettre en évidence le fonctionnement interne de cette application. On peut aussi remarquer que nous avons utilisé la librairie « CommonLib » afin de structurer nos messages et que nous avons utilisé Winform pour l'interface.

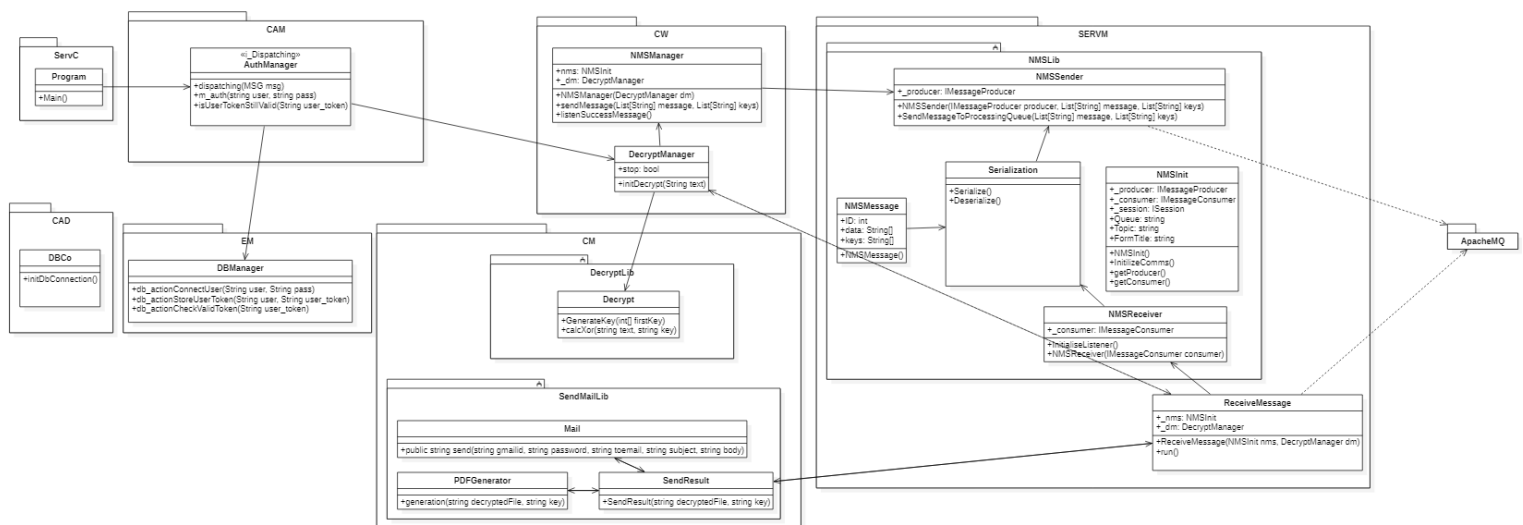


Figure 8 - Diagramme de classe Serveur .NET

Ce diagramme représente les classes de notre serveur .NET. Il permet de mettre en évidence le fonctionnement interne de la plateforme .NET. On remarque aussi que le serveur.Net construit et sérialise les messages en objet JSON (avec la librairie « NMSLib ») avant de les envoyer au Serveur JEE via le serveur de messagerie « Apache MQ ».

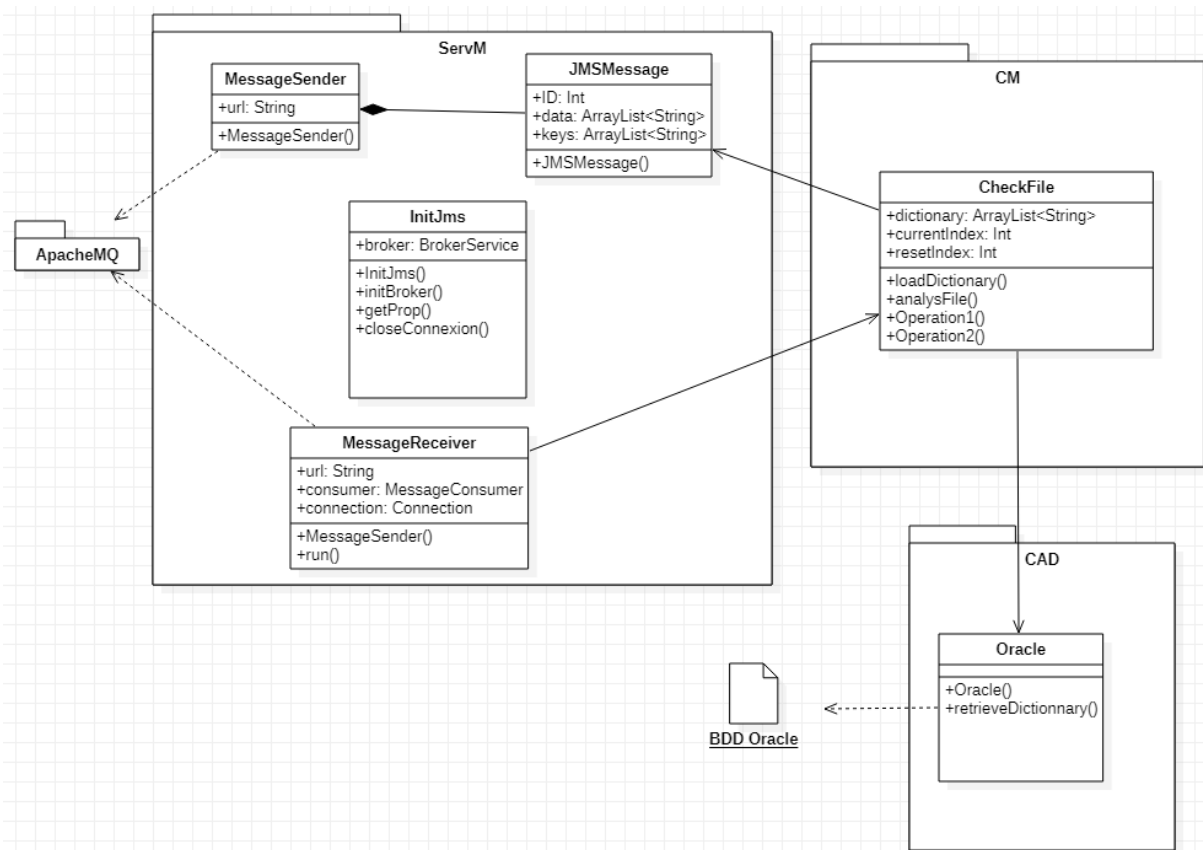


Figure 9 - Diagramme de classe Serveur JEE

Ce diagramme représente les classes de notre serveur JEE. Il permet de mettre en évidence le fonctionnement interne de la plateforme JEE. On remarque les messages sont gérés par un serveur de messagerie « Apache MQ » qui permet de faire transiter les messages entre les deux serveurs. Les messages sont préparés au préalable et stockés dans une queue par JMS.

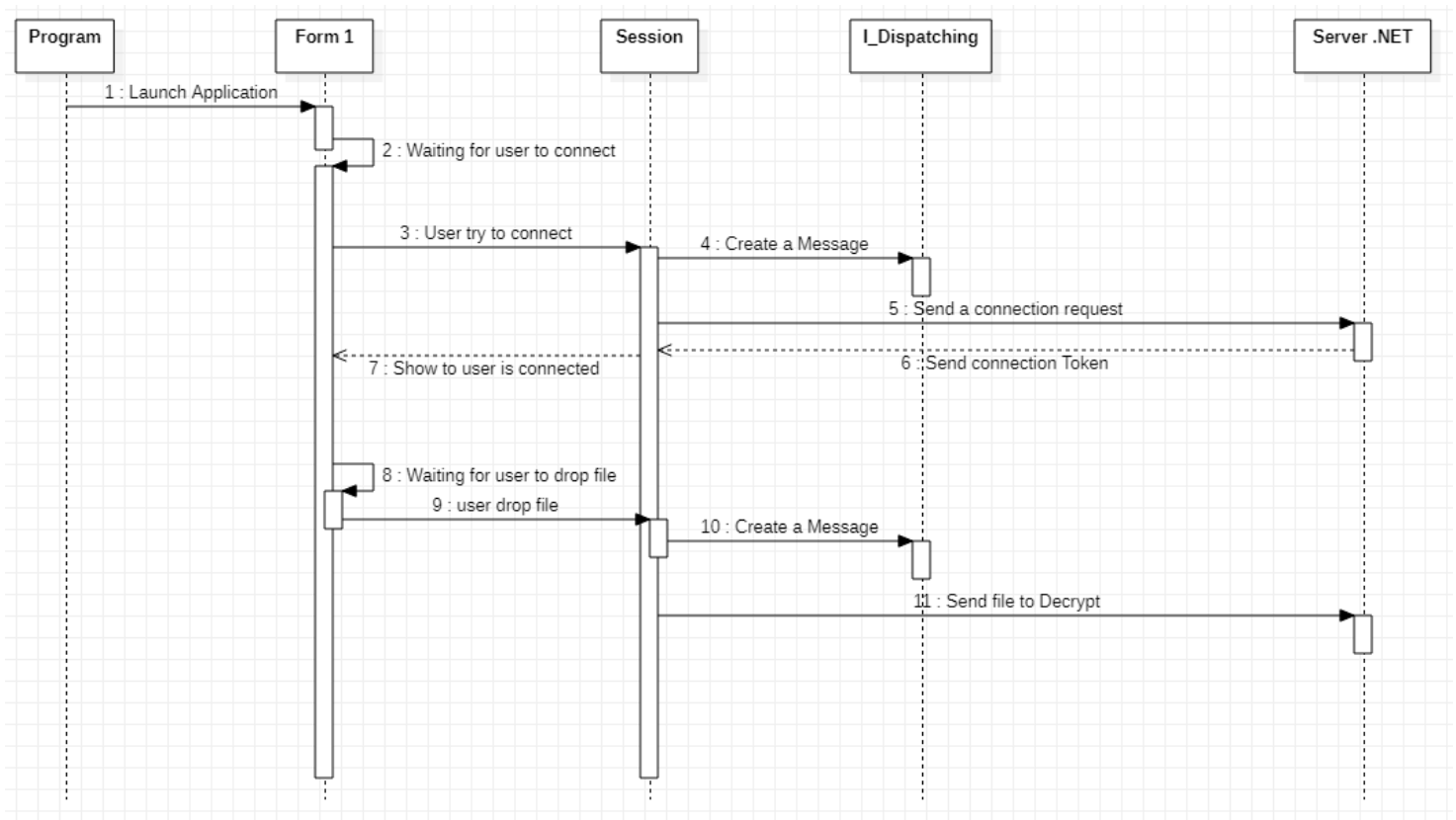


Figure 10 - Diagramme de séquence Client Lourd

Ce diagramme de séquence représente les différentes actions interne de notre Client lourd. On remarque que l'interface « I_Dispatching » de la librairie « Commonlib » s'occupe de structurer nos messages avec les informations spécifiées par l'utilisateur lors de l'utilisation de l'application (identifiant/password et dépôts de fichiers).

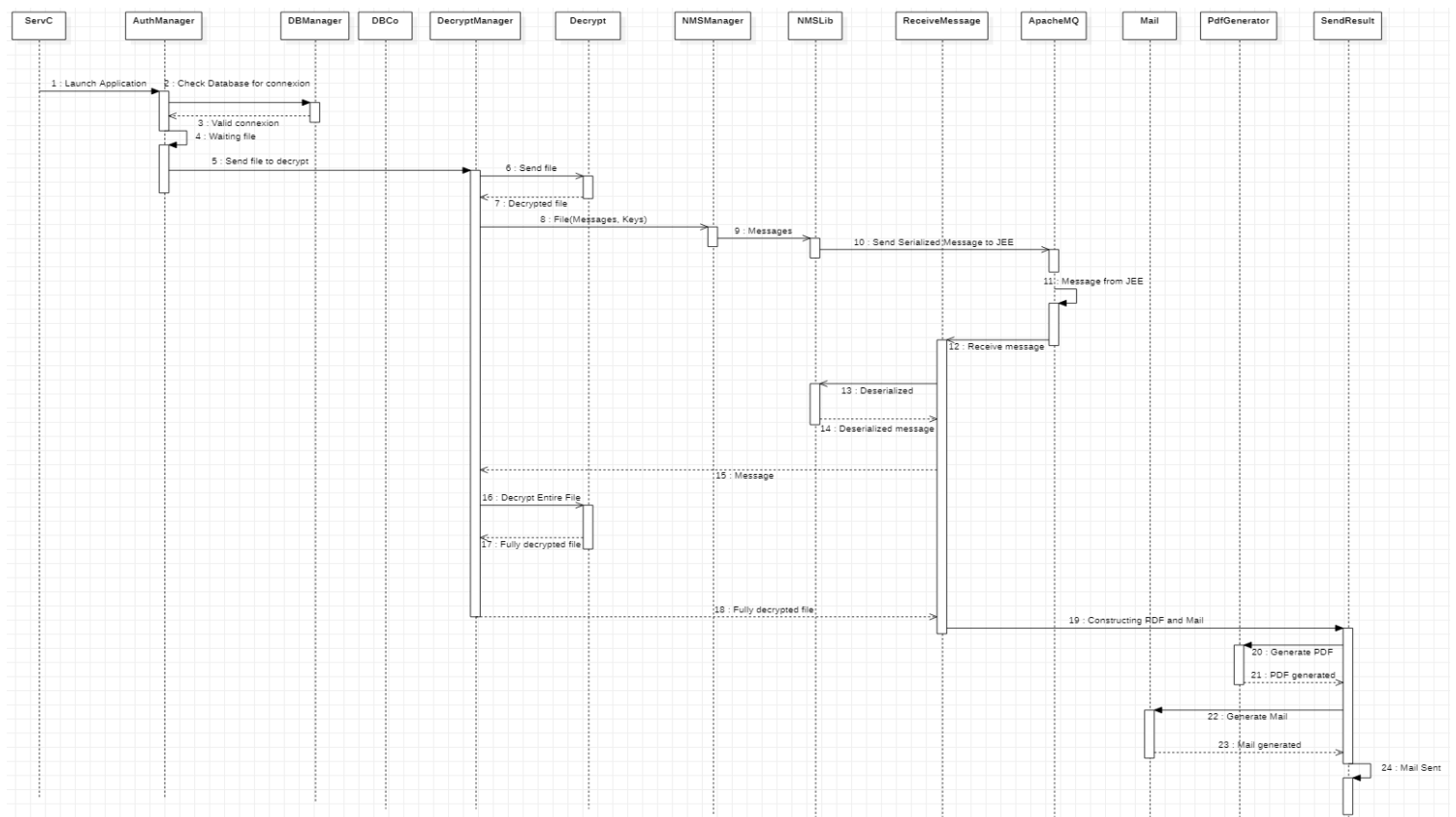


Figure 11 - Diagramme de séquence Serveur .NET

Ce diagramme de séquence représente les différentes actions internes de notre Serveur.net. On remarque que le décryptage se fait en plusieurs temps :

- On commence par décrypter les 1000 premiers caractères du fichier avec chaque clé,
- Puis ce fichier est envoyé au serveur JEE,
- Le serveur JEE parcourt les 1000 premiers caractères,
- Si le fichier correspond au pourcentage de mots français désirés alors le serveur JEE renvoi le fichier au serveur .NET,
- Le serveur .NET fini alors de décrypter totalement le fichier,
- Le renvoi au Serveur JEE qui finit de le parcourir,
- Si le fichier contient alors l'information secrète,
- Le serveur JEE renvoi le fichier décrypté, l'information secrète et la clé
- Enfin le serveur .NET génère un PDF et envoi un mail avec le PDF en pièce jointe.

Les messages contenant les fichiers décryptés sont construits et envoyer grâce à la librairie « NMSLib ». Le fonctionnement de la librairie « NMSLib » est détaillé ci-dessous.

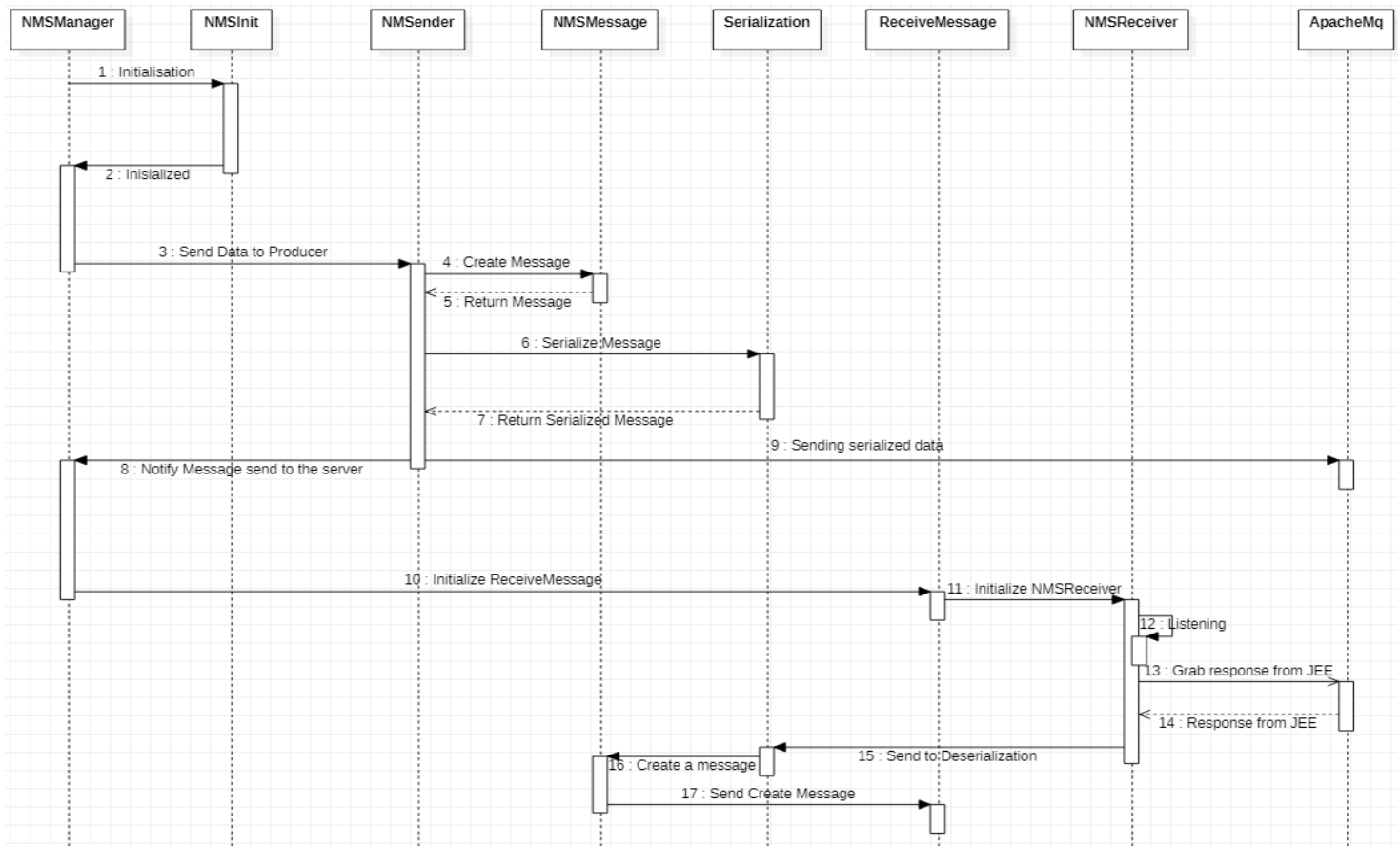


Figure 12 - Diagramme de séquence de la librairie NMSLib

Ce diagramme de séquence détail les différentes actions interne à la librairie « NMSLib ». Ici on comprend comment la librairie construit nos messages à partir des informations reçues (fichier décrypté), elle les sérialise en objets JSON et les fait transités vers le serveur JEE via le Serveur de messagerie « Apache MQ ». Lors d'une réception de message, la librairie effectue le travail inverse, c'est-à-dire qu'elle désérialise le message qui proviens de « Apache MQ ».

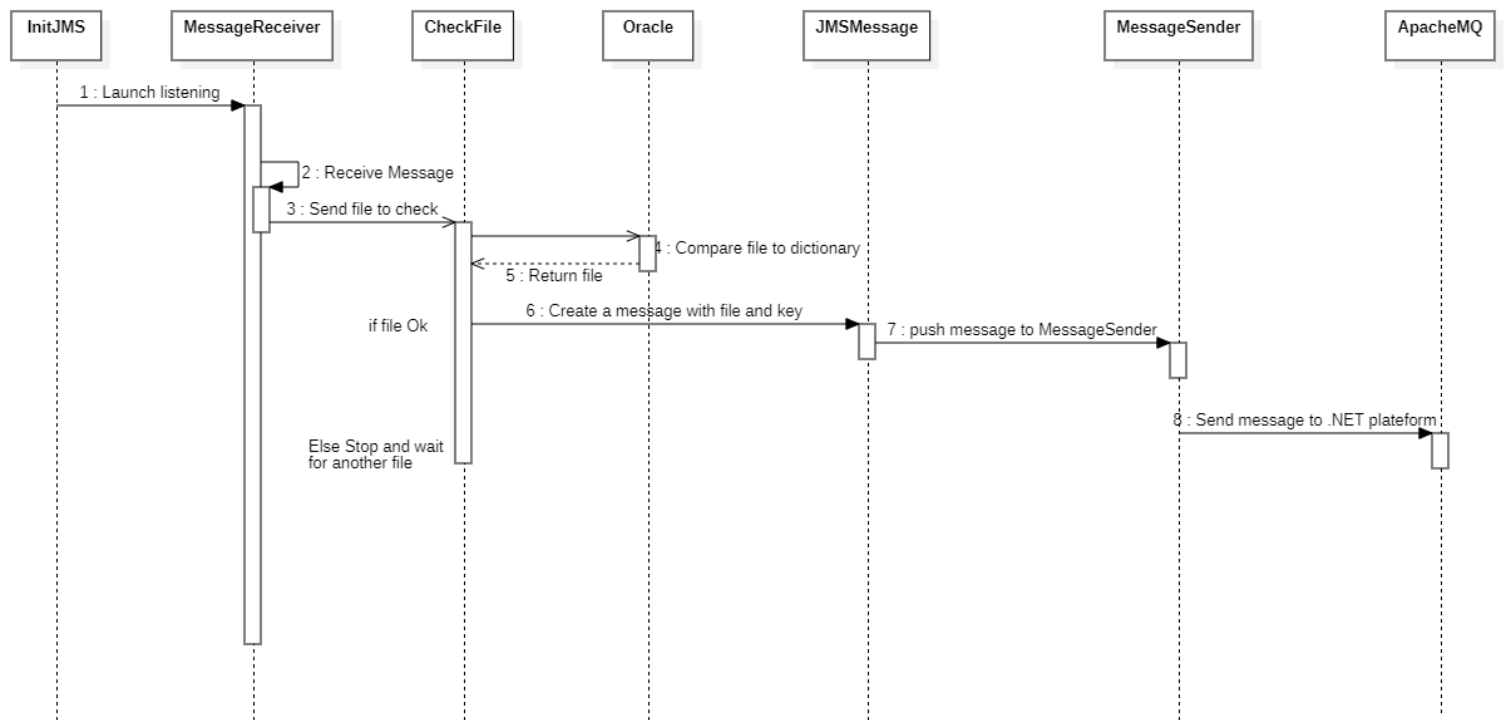


Figure 13 - Diagramme de séquence du serveur JEE

Ce diagramme de séquence représente les différentes actions interne à notre Serveur JEE. On remarque que le serveur vérifie si les fichiers respectent un pourcentage correct de mots français alors il envoie un message au serveur .NET, sinon il ne fait rien et passe au fichier suivant. Les messages sont construits grâce à JMS et envoyés au Serveur.NET via le serveur d'application « ApacheMQ ».

VI. Analyse des écarts :

Tout d'abord nous avons remarqué de gros écart à propos de la répartition des tâches. Finalement les tâches été réalisées comme ceci :

Projet Dev du 22/06/2020 au 03/07/2020					Semaine 1					Semaine 2				
Tâche	Date de début	Date de fin	Durée		22	23	24	25	26	29	30	1	2	3
Martin Rochette														
1.UML	22-juin	24-juin	28h											
2. Serveur .NET	29-juin	29-juin	8h											
2a. BDD SQL														
3. Serveur JEE	25-juin	26-juin	12h											
3a. BDD Oracle	26-juin	26-juin	6h											
4. Client Lourd														
5. Middleware	30-juin	01-juil	16h											
6. Préparation Soutenance	02-juil	03-juil	12h											
William Nuness														
1.UML	22-juin	24-juin	28h											
2. Serveur .Net	25-juin	26-juin	16h											
2a. BDD SQL														
3. Serveur JEE														
3a. BDD Oracle														
4. Client Lourd	26-juin	29-juin	4h											
5. Middleware	29-juin	01-juil	24h											
6. Préparation Soutenance	02-juil	03-juil	12h											
Guillaume Brute														
1.UML	22-juin	24-juin	22h											
2. Serveur .Net	24-juin	25-juin	10h											
2a. BDD SQL	25-juin	25-juin	2h											
3. Serveur JEE	26-juin	26-juin	12h											
3a. BDD Oracle	26-juin	26-juin	6h											
4. Client Lourd	29-juin	29-juin	8h											
5. Middleware	30-juin	01-juil	24h											
6. Préparation Soutenance	02-juil	03-juil	12h											
Victor Sorrenti														
1.UML	22-juin	24-juin	30h											
2. Serveur .Net	25-juin	29-juin	24h											
2a. BDD SQL														
3. Serveur JEE														
3a. BDD Oracle	26-juin	26-juin	2h											
4. Client Lourd														
5. Middleware														
6. Livrables	30-juin	03-juil	16h											
7. Préparation Soutenance	02-juil	03-juil	10h											

Figure 14 - Planning réel

Ces écarts de planning sont dû au fait que, lors du premier brainstorming nous n'avions pas vraiment toutes les données du projet en tête. Donc nous n'avions pas vraiment conscience du travail nécessaire, du temps et des technologies à utiliser. Nous nous sommes donc adaptés en fonction des compétences et de la volonté de chacun.

Cependant la répartition des tâches a pu être revue et nous a quand même permis de réaliser le projet en temps et en heures.

En ce qui concerne les écarts entre notre projet et le cahier des charges, voici un tableau récapitulatif et détaillé de notre avancement :

Cahier des charges	Écarts	Détails
Architecture : 3 tiers - SOA	Présent	Prouvé par le diagramme d'architecture logiciel
Framework : Microsoft Framework 4.0	Présent	
Langage : C# 4.0	Présent	
Solution : 9 couches	Présent	Prouvé par les diagrammes de classes (6 sur le serveur .NET et 3 Client Lourd)
Client		
A déployer sur les postes clients	Présent	peut être déployer sur n'importe quel poste
Lourd / asynchrone	Présent	Pas d'attente
3 couches (1 composant physique)	Présent	Prouvé par le diagramme de classe du client Lourd
Parallélisations des traitements issus du middleware (traitements lourds). Client non bloqué lorsqu'il attend les réponses de la plateforme.	Pas présent	Créer un thread principale pour le client puis créer un thread secondaire pour la connexion afin d'éviter de bloquer le premier thread
WPF possible, donc pas obligatoire	Pas présent	Utilisation de windows Form
Middleware		
A déployer sur un serveur	Présent	Serveur messagerie Apache MQ
Architecture de type service	Présent	Prouvé par le diagramme d'architecture logiciel
Point d'entrée unique	Présent	voir class Program.cs
Communication sécurisée basée sur le chiffrement de bout en bout	Pas Présent	Analyser les messages grâce à un logiciel type WireShark ou possibilité de rajouter une méthode qui crypte ou décrypte les messages sérialisé.
Structure des messages	Présent	Voir librairie "Commonlib" interface "I_Dispatching"
une seule méthode exposée sur le point d'entrée unique de signature : STG_m_service(STG msg) ?	Présent	voir class Program.cs
Distribué	Présent	car chaque partie du logiciel est capable de fonctionner seule / sur une autre machine
Journalisation de l'activité du CAM. (Logs)	Pas présent	Rajouter une méthode qui permettrait de créer un fichier.Txt log dans un dossier logs
Les contrôleurs de workflow et les composants métiers doivent supporter des charges de travail importantes.	Présent	beaucoup de requêtes en même temps
5 couches	Présent	voir diagramme des classes
1 composant physique pour les couches 5 – 4 – 3	Présent	voir diagramme des classes
1 composant physique pour la couche 2	Présent	voir diagramme des classes
1 composant physique pour la couche 1	Présent	voir diagramme des classes
Data		
SQL Server STD Ed 2008	Présent	Version plus récente
Mode d'authentification mixte	Présent	utilisateur windows + utilisateur spécial SQL
Mise en place de l'agent SQL Server dans un processus Windows distinct	Présent	
Mise en place du moteur SQL Server dans un processus Windows distinct	Présent	
Sauvegarde automatique journalière (sauvegarde complète)	Pas présent	Modifier les paramètre de la sauvegarde de la base pour passer les paramètre en mode sauvegarde complète et spécifier la localisation du fichier de stockage
Communication		
Liaison de type netTcp entre le client lourd et la plateforme	Pas présent	Mise en place faisable d'autant plus que WCF fourni un protocole réseaux basé sur net.TCP
Communication sécurisée avec un chiffrement à 128b	Pas présent	Mise en place faisable via une librairie ou paramétrage

Figure 15 - Tableau récapitulatif des écarts

On remarque de légers écarts avec le cahier des charges, nous pensons qu'avec un peu plus de temps nous pourrions tout mettre en place ces fonctionnalités d'autant plus que nous avons une base solide et que nous savons comment nous y prendre.

VII. Analyse des compétences acquises par étudiants :

Victor Sorrenti : Ce projet m'a permis de mettre en pratique toute la théorie que nous avons vue lors du bloc dominante Dev. Il m'a permis de développer mes compétences en développement (notamment avec la création de diagramme UML que nous n'avions pas vue depuis longtemps) C# et de découvrir plus en profondeur la cryptographie. Il m'a permis de mieux comprendre le fonctionnement des communication logiciel (WCF, JMS, Apache MQ ...) vue en cours. Il m'a permis aussi d'améliorer ma gestion de mon temps et de mon stress.

Martin Rochette : Ce projet a été très enrichissant en compétence informatique de par la mise en pratique de la théorie vu en prosit et l'utilisation des deux différents langages. J'ai pu aussi mieux comprendre l'intérêt de l'UML qui aide grandement lors de l'implémentation de l'application. J'ai aussi mieux compris l'asynchronisme et l'utilisation d'une "queue" afin de pouvoir gérer de plus ou moins grandes charges de données.

William Nunesse : Grâce à ce projet j'ai vraiment l'impression d'avoir gagné en expérience sur de nombreuses compétences. Comme je me suis principalement chargé du serveur .NET la plupart des compétences acquises sont en lien avec cette technologie. Des notions avant très théoriques (malgré les workshops car nous sommes guidés durant leurs réalisations) sont à présent de véritables compétences ingénieurs. Notamment la gestion des threads, l'utilisation de WCF ou encore la construction de librairie pour les messages (NMSLib) pour ne citer qu'eux. De même ce projet m'a permis de solidifier certains acquis comme la modélisation UML et la cryptographie (que nous avons vu dans le projet précédent). Enfin des compétences tels que la gestion de projet, du temps et le travail en équipe même si aujourd'hui maîtrisés sont bonnes à exercés en continue. D'une manière générale comme pour la plupart des projets précédents j'ai l'impression d'avoir acquis de nombreuses compétences.

Guillaume Brut : Ce projet m'a permis de mettre en pratique un grand nombre de compétences acquises lors des nombreux précédents prosits. Je peux citer en particulier JEE, WCF, les systèmes de thread, de synchronisation, les serveurs de message et bien plus. J'ai aussi pu apprendre à bien structurer un code important, ainsi qu'optimiser les performances en ne pas sacrifiant le fonctionnement du programme.

VIII. Bilan :

Ce projet fût un projet très intéressant mais aussi très stressant aux vues de la quantité de travail à fournir aux premiers abords. Cependant grâce à ce projet nous avons pu approfondir la plupart des compétences que nous avons vu lors de nos cours dans le bloc dominante Dev, c'est-à-dire toute la partie Java JEE (JMS, EJB, CDI etc ...) toute la partie C# (WCF, Thread etc). Ce projet nous a permis aussi de revoir toutes les notions de Conception de logiciel avec notamment les Diagrammes de composants, de classes et de séquence. Enfin ce projet nous a permis d'améliorer nos techniques de management de projet (gestion du temps, des tâches ...).