

Rapport de projet :

Projet développement

Quentin Daveau, Rayene Limem,

Charles Paradot

Tuteurs : Jocelyn Niget, Mauricio Itturalde

Une image contenant signe, vert, pièce, tenant

Description générée automatiquement

A4 CESI Informatique de La Rochelle, Promotion 2019 – 2020

CESI La Rochelle, 17 000 La Rochelle, France

Table des matières

[2. Glossaire 4](#_Toc36888978)

[3. Introduction 5](#_Toc36888979)

[4. Management des risques 6](#_Toc36888980)

[5. Cyber sécurité 7](#_Toc36888981)

[6. Cryptographie 8](#_Toc36888982)

[7. Management de projet 9](#_Toc36888983)

[8. Conclusion 10](#_Toc36888984)

[9. Table des illustrations 11](#_Toc36888985)

# Glossaire

**Aucune entrée d'index n'a été trouvée.**

# Introduction

Dans le cadre de la quatrième année d’étude au CESI spécialité informatique afin d’obtenir le diplôme d’ingénieur informatique te réseaux, nous avons le choix de choisir parmi deux spécialités qui seront le sujet de deux mois d’étude à la fin de l’année scolaire. Ces deux spécialités (dominantes) sont Développement, et Réseaux. Chacune de ces spécialités propose, lors des dernières semaines d’étude, un projet à réaliser afin d’effectuer une évaluation finale des compétences acquises sous forme pratique.

Ce document est entièrement concerné au projet de la dominante développement, il va de soit que nous allons donc nous concentrer sur celui-ci.

Le projet final de la dominante développement, effectué par une équipe de trois personnes, débuté officiellement le 22 Juin 2020 et finissant par une soutenance orale de 20 minutes le 03 Juillet 2020 a pour objectif de capitaliser sur les connaissances accumulées tout au long de la dominante dans un dernier exercice pratique, de ce fait, il porte sur les thèmes des architectures orientées service, du développement Java reposent sur JEE et enfin du développement .NET et WCF.

Ce rapport va donc, par la suite, décrire l’ensemble des aspects du projet, que ce soient les aspects techniques ou bien la gestion de projet. Il est cependant bon de retenir que ce rapport est contraint par un ensemble de directives quant à son organisation ou son contenus, définies dans l’énoncé du projet et jouant un rôle intégral dans la notation de celui-ci. De ce fait il se peut que celui-ci semble inadapté à une lecture d’un parti externe au projet, étant-donné qu’il rentre dans nombre de détails techniques inhérents à celui-ci qui peuvent sembler peu pertinents, mais sont une nécessité afin de correctement répondre aux attentes.

# Besoin

Le contexte du projet se décrit comme suit : l’ensemble du projet et de ses directives nous a été fournis par le biais d’un document faisant office de cahier des charges décrivant le besoin du projet. Ainsi, pour résumer succinctement, il nous est demandé, sans aucune mise en contexte concrète, de créer trois applications (un client lourd développé en .NET, un serveur .NET et un serveur Java) dont le but étant, en travaillant collaborativement, de permettre :

1. à un utilisateur de s’identifier à partir d’un nom d’utilisateur et un mot de passe. Le serveur .NET assure la communication avec le client et gère l’identification. Si les identifiants sont corrects, le client lourd reçois un token utilisateur lui permettant d’accéder aux diverses fonctionnalités suivantes. Si l’utilisateur n’est pas identifié, il ne peut pas procéder à la suite ;
2. l’utilisateur doit pouvoir fournir un lot de fichiers cryptés par le biais du client lourd, qui seront envoyés vers le serveur .NET ;
3. le serveur .NET se doit de décrypter les fichiers donnés par le client en brute-force en y appliquant toutes les combinaisons de clé possible, la clé étant une série de 4 caractères de l’alphabet en majuscule. Ainsi pour chaque fichier, une multitude de jeux *fichier décrypté – clé* sera obtenue (un jeu par clé possible, soit approximativement 456 976 jeux);
4. pour chaque jeu *fichier décrypté – clé*, celui-ci doit être envoyé au serveur Java ;
5. le serveur Java se doit d’évaluer l’intelligibilité de chaque fichier décrypté reçus en comparant les « mots » s’y trouvant avec un dictionnaire. Il retire donc, pour chaque fichier décrypté reçus, un score estimant la probabilité que la clé utilisée pour décrypter le fichier soit la bonne ;
6. si le serveur Java estime qu’un fichier décrypté l’a été avec la bonne clé (c’est-à-dire, si le fichier décrypté obtiens un score supérieur à une limite définie), il se doit de rechercher dans celui-ci une information secrète (contenue dans un unique fichier du lot fourni par l’utilisateur) introduite par la phrase « l’information secrète est : ». Si il trouve l’information secrète, il se doit de contacter le serveur .NET afin de l’y avertir et de lui donner le jeu *fichier décrypté – clé* en question, ainsi que l’information secrète et le score du fichier. Malgré tout, si l’information secrète n’est pas trouvée dans le fichier mais que la clé de décryptage utilisée est considérée comme bonne, le serveur Java se doit de donner au serveur .NET le jeu *fichier décrypté – clé* en question et le score du fichier ;
7. si le serveur .NET reçois le message du serveur Java l’avertissant avoir trouvé la bonne clé pour le fichier en question, celui-ci se doit d’envoyer un Email à l’utilisateur ayant un rapport en .pdf contenant le nom du fichier, son contenus décrypté, son score, ainsi que la clé utilisée. Il se doit aussi de prévenir le client lourd si celui-ci est connecté. Si le serveur Java l’avertit en plus d’avoir trouvé l’information secrète, il doit l’ajouter aux données citées précédemment.

A noter que tout un ensemble de contraintes supplémentaires (protocoles utilisés, architecture des serveurs…) sont donnés pour la procédure citée ci-dessus, cependant nous n’y reviendrons pas dessus ici car ce ne serait pas pertinent. Il est juste bon de noter qu’en plus de ladite procédure, nous devons fournir de multiples diagrammes UML (diagramme de classe, diagramme de cas d’utilisation…).

# Découpage du projet

En se reposant sur l’énoncé précédemment cité, nous pouvons constater que le projet peut se diviser en, globalement, cinq tâches principales qui sont les suivantes :

1. création des UML ;
2. création du client lourd .NET ;
3. création du serveur .NET ;
4. création du serveur Java ;
5. rédaction du rapport.

Nous pouvons rajouter à ces tâches principales un ensemble de tâches secondaires pouvant découler directement ou indirectement de celles-ci :

* mise-à-jour des UML (suite à un retour de la part d’un expert) ;
* communication du client lourd et du serveur .NET ;
* communication du serveur .NET et du serveur Java ;
* recherche du message secret ;
* mise-à-jour des UML en post-développement ;
* création du diaporama pour la présentation.

Nous avons donc un ensemble de tâches principales particulièrement bien démarqué, chacune pouvant évoluer dans son propre environnement, mis-à-part potentiellement la création des UML, qui nécessite d’être partiellement effectué en tout premier et avec l’ensemble de l’équipe afin de correctement définir le fonctionnement général de l’architecture mise en place et ainsi assurer la bonne compréhension de la part de chaque membre. Les tâches secondaires peuvent être synonymes d’une faible durée de réalisation mais elles sont néanmoins tout aussi importantes que les tâches principales.

# Planification initiale et répartition des tâches

## Planification et répartition

A la vue des différentes tâches (principales et secondaires), il sembla évident de diviser le projet en une tâche par personne. L’équipe étant constituée de trois membres (Quentin Daveau, Limem Rayene et Charles Paradot), et la réalisation nécessitant trois applications (client lourd, serveur .NET, serveur Java), il fut décidé que chaque membre s’occuperai d’une application donnée :

* **Rayene Limem**:client lourd ;
* **Quentin Daveau (chef de projet)**: serveur .NET ;
* **Charles Paradot**: serveur Java.

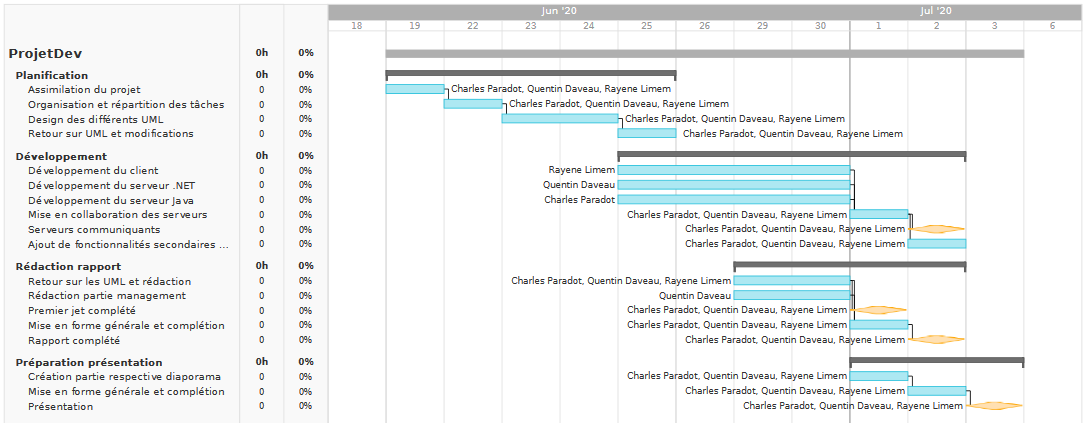
 Les deux tâches principales restantes (création des UML et rédaction du rapport) étant réalisées collectivement. Quentin étant désigné comme chef de projet, la tâche supplémentaire de la rédaction de la partie management du rapport lui fut attribuée. Ainsi, la planification et répartition des tâches initiale fut définie comme suit :

Figure 1: Le diagramme de Gantt initial

Nous pouvons remarquer plusieurs éléments notoires sur ce diagramme de Gantt. En un premier temps, nous pouvons remarquer que la planification commence à partir du 19 Juin 2020. En effet, l’équipe eu accès à l’énoncé du projet le 19 Juin, lui permettant d’en prendre connaissance tout au long du week-end afin de correctement l’aborder dès le lancement du projet, le 22 Juin.

En un second temps, on constatera qu’une durée importante est dédiée purement à l’assimilation du projet et à la mise en place des diagrammes UML (et ce collaborativement par l’ensemble des membres du groupe). L’énoncé du projet, et le projet même, étant relativement complexes et pouvant mener à nombre d’incertitude et de confusions, cette période prolongée d’analyse fut définie afin de s’assurer que tous les membres de l’équipe comprennent correctement les enjeux du projet. De même façon, la création collaborative des premiers UML eut un but similaire, c’est-à-dire de s’assurer que chaque membre de l’équipe comprenne parfaitement l’architecture qui allait être mise en place, ainsi que son fonctionnement (tout du moins sur un aspect macro, pour commencer), afin d’éviter toute confusion une fois que chaque membre commencera à développer l’application qui lui a été attribuée. Il est bon de noter qu’un retour de la part d’un expert en UML le mercredi 24 Juin étant planifié, il semblait judicieux d’attendre son opinion avant de se lancer sérieusement dans le développement des applications.

En un dernier temps, on pourra relever que la rédaction du rapport fut planifiée pour commencer relativement tôt. Ce choix fut effectué en vue de la semblable complexité du projet, et donc de l’incertitude de sa complétion sur l’aspect technique. Afin d’éviter d’impacter la rédaction du rapport à cause de difficultés (et donc de pertes de temps) dans le développement des applications, il fut choisi de privilégier le rapport lors de la seconde semaine, et ce même si l’application n’était pas complétée, pour assurer un minimum de qualité concernant celui-ci. Cependant, cela ne devait en aucun cas impacter directement le développement de l’application, juste s’assurer de la correcte rédaction du rapport.

## Organisation

Concernant l’organisation générale de l’équipe, il fut décidé de s’organiser à partir d’une méthode agile articulée autour du logiciel Microsoft Teams. Cette méthode impliquait donc les spécificités suivantes :

* Découpage du travail en tâches spécifiques, claires, et indépendantes afin que chaque membre de l’équipe puisse se focaliser sur son travail avec le moins de parasitage possible ;
* Réunions quotidiennes pour rapide mise au point du travail effectué, planification des tâches de la journée et potentielle entraide en cas de besoins ;
* Centralisation du travail effectué sur une plateforme commune (Microsoft Teams et GitHub) accessible par toute l’équipe à tout moment, permettant à chacun d’apporter un avis critique et constructif concernant le travail des autres, et ce peu importe l’heure de la journée (permettant une meilleure flexibilité, adaptée au télétravail).

Les outils exploités pour le bon déroulement du projet furent donc majoritairement Microsoft Teams et Github :

* Microsoft Teams est une plateforme de collaboration d’équipe en ligne offrant une multitude de fonctionnalités particulièrement adaptées au travail en distanciel. Ainsi Teams permet d’effectuer des réunions en ligne, des partages d’écran et de fichiers, offre un dépôt permettant de centraliser les documents et le travail effectué et de les éditer collaborativement et en temps réel. De ce fait, Teams se définit comme l’axe majeur de la collaboration au sein de l’équipe, et permit à l’équipe d’effectuer ses réunions journalières, d’échanger continuellement tout au long du projet et de travailler collaborativement sur des documents tel que les diagrammes UML ou ce même rapport ;
* Git est un outil de version control (contrôle de version) et GitHub est une plateforme d’hébergement de code source reposant sur Git. GitHub permit donc l’hébergement, la vérification et le contrôle du code développé par chaque membre du projet, et facilita grandement l’échange et l’intégration des ressources entre les différents membres.

Ainsi, l’organisation générale définie par l’équipe se voulait simple, mais efficace. Reposant sur l’ensemble des membres de celle-ci, elle comptait sur leur autonomie et prise d’initiative afin d’assurer une collaboration instinctive et volontaire. Supportée par Microsoft Teams et Git / GitHub, elle se voulait directe et peu contraignante pour chaque membre, leur permettant une large liberté d’approche tout en prévenant toute déviation grâce aux nombreuses réunions et à la communication facilitée, assurant un robuste suivi de l’avancement du projet auprès de chaque membre. De plus, l’utilisation d’outils comme GitHub et Microsoft Teams permit un archivage et une centralisation du code et de tout travail effectué, aspect primordial dans le développement.

# Modélisation

# Analyse des écarts

# Compétences acquises

# Conclusion

# Table des illustrations

**Aucune entrée de table d'illustration n'a été trouvée.**