Rapport projet vote électronique

Sheila ”YoungFolk” Barron
Dylan ”Shyrame” Durand
Gaël ”Funkywi” Hollows-Pellier
Jean-Clair ”Abaca” Granier

10 janvier 2018

Remerciements

Nous tenons à remercier notre enseignant tuteur M. Lebreton qui nous a été d'une aide précieuse et qui nous a consacré du temps pour concevoir ce projet. De plus, nous tenons à remercier Mme. Messaoui de nous avoir aidés dans la rédaction de notre rapport.

Table des matières

Table des figures

Glossaire

Arithmétique modulaire Ensemble de méthodes permettant la résolution de problèmes sur les nombres entiers. Ces méthodes dérivent de l'étude du reste obtenu par une division euclidienne. 8

Garbage collector (GC) Entité de Java gérant la mémoire et libérant les ressources inutilisées. 18

Protocole SSL/TLS protocole de sécurisation des échanges sur Internet.. 7

Acronymes

GUI *Graphical User Interface*. 12

LIRMM Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier. 6

MVC Modèle-Vue-Contrôleur. 12

Introduction

L'informatique a de nos jours envahi l'ensemble des strates de notre société : de la sphère familiale à la vie professionnelle, les outils numériques se sont imposés en un temps record, et sont devenus indispensables à l'organisation de nos structures sociales. Des dossiers médicaux aux actes de naissance en passant par les formulaires administratifs, notre vie s'est entièrement structurée autour de ces outils informatiques. Les raisons en semblent évidentes : un ordinateur, froid et impartial, se contente de faire ce qui lui est demandé ; et aucun facteur extérieur se saurait le détourner de la tâche qui lui a été assignée.

Informatiser le système de scrutin, dans une démocratie comme la nôtre, semblait donc être une réponse évidente à un besoin devenu quasi primaire : confier à nos machines la lourde tâche de permettre à chacun de donner son avis, et ainsi s'assurer que l'impartialité du résultat sera respectée. Le vote sera ainsi rendu aussi impersonnel que possible, puisque ne nécessitant plus l'acte social d'aller voter ; et chacun pourra se fondre dans la masse, et voir son vote, porteur d'une opinion et représentant d'un pouvoir personnel, rejoindre ceux des autres, avec la certitude qu'il ne vaudra ni plus, ni moins.

L'application que nous allons vous présenter a donc pour objectif de proposer un système de scrutin sécurisé et confidentiel. L'idée directrice de notre projet est la protection des votes individuels ; notre travail s'est donc essentiellement axé sur les méthodes de chiffrement de l'information, pour que chaque vote soit individuellement illisible, et que le résultat global ne soit lisible que dans le contexte approprié.

Ce rapport vient clôturer notre projet, et nous permet d'en décrypter les enjeux. Nous allons donc analyser les différents besoins auxquels nous avons été confrontés et les contraintes inhérentes à ces besoins, définis dans le cahier des charges. Une fois les objectifs définis, nous entrerons dans la partie technique : nous verrons les différentes solutions choisies et suivrons leurs mises en place. Nous proposerons ensuite une analyse critique de ces solutions : leurs forces, éventuellement leurs faiblesses, et leur résultat final. Enfin, nous parlerons de l'organisation du travail au sein de notre équipe, tout au long des différentes phases du développement, et ferons un bilan de notre projet.

Nous espérons que nous prendrez autant de plaisir à découvrir notre projet que nous en avons eu à travailler dessus. Merci de votre attention.

En vous souhaitant une agréable lecture,

Sheila Barron
Dylan Durand
Gaël Hollows-Pellier
Jean-Clair Granier

1 Cahier des charges

1.1 Présentation du sujet et analyse du contexte

1.1.1 Présentation du sujet

FIGURE 1 – Système de vote traditionnel

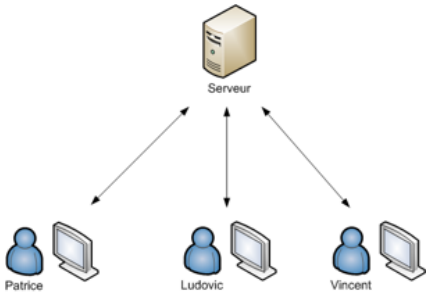


FIGURE 2 – Système d'application client-serveur

Lors de toutes les grandes élections, le votant doit se présenter lui même au lieu du scrutin et ainsi voter une unique fois. Chaque vote est alors anonyme et déposé dans une urne scellée. Le dépouillement s'effectue lorsque tout le monde a voté ou lorsqu'un organisateur le décide.

Le but du projet est de pouvoir modéliser ce système au travers d'une application et surtout de pouvoir tout automatiser. Chaque votant aurait alors un compte sur l'application et pourrait alors voter une unique fois. Le vote serait alors transmis au serveur qui ferait office d'urne en stockant les votes.

Le plus gros problème reste la sécurité car tous les votes sont anonymes. Il va donc falloir chiffrer les votes selon une méthode particulière telle que si un pirate intercepte un vote alors ce vote ne peut pas être lu. Contrairement au vote traditionnel, le vote ne sera pas anonyme car celui-ci sera chiffré. C'est pourquoi dans le but de montrer le fonctionnement de l'application, il sera intéressant que les utilisateurs puissent voir leurs votes chiffrés.

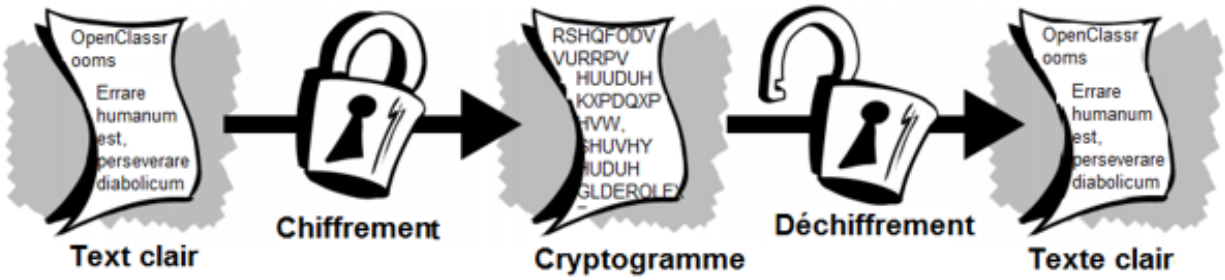


FIGURE 3 – Modélisation d'un système de chiffrement

Il existe déjà de nombreux systèmes de vote électronique. Certains sont très sécurisés et d'autres le sont moins mais il est certain que c'est une technologie déjà présente sur le marché. Par exemple un employeur peut recourir au vote électronique pour les élections professionnelles dans le respect de certaines conditions et formalités préalables.

On trouve des avantages et des motivations pour adopter la démarche d'un système de vote électronique :

- La rapidité du dépouillement de votes, vu que les calculs sont faits plus rapidement par des ordinateurs, au contraire d'un dépouillement humain qui serait plus lent car il demanderait plus d'effort de la part des intervenants humains.
- La fiabilité, vu que l'application de vote électronique fait un calcul systématique des votes, au contraire d'un dépouillement humain qui pourrait avoir des défaillances.
- La réduction de mobilisation de votants : imaginons l'extension du vote électronique sur notre application pouvant s'exécuter depuis l'ordinateur du votant, ceci renforce sa participation vu qu'il pourrait voter depuis son ordinateur et n'aurait pas besoin de se déplacer.
- Réduction des frais de matériel électoral et des besoins des ressources humaines, sachant qu'il est parfois difficile de trouver des volontés pour devenir des assesseurs, faire voter et dépouiller.

Cependant, on se retrouve aussi avec des inconvénients liés à l'application de vote électronique :

- Le code source de l'application de vote n'est pas partagé. Donc, les votants ne peuvent pas vérifier qu'il n'y a pas une faille à l'intérieur de l'application. La sécurité est gérée par le serveur de vote auquel les votants doivent faire confiance aveuglément.
- Le risque de fraude est non négligeable, il est impossible à l'électeur d'être sûr qu'il n'y a pas eu de fraude ; par exemple, il ne pourrait pas savoir si le

serveur n'a pas autogénéré des votes ou s'il n'a pas effectué un vote à la place d'un votant.

— Des risques de sécurité sont présents : il faut employer des méthodes qui garantissent le chiffrement des votes pour qu'il ne soit pas connu, des méthodes pour vérifier le fait qu'un vote provient bien d'un votant qui n'a pas encore voté, donc garantir que les seules personnes qui peuvent voter, sont bien des personnes registrées qui n'ont pas encore voté. Il est aussi nécessaire d'employer des méthodes contre les attaques qui compromettent l'intégrité du système, par exemple, se protéger contre l'attaque de l'homme au milieu aussi appelée attaque de l'intercepteur*, cet attaque a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles à été compromis. Dans ce cas, l'attaque aurait lieu entre les votants et le serveur ; de cette manière, l'authenticité des votes ne pourrait pas être garantie.

1.1.2 Analyse du contexte

Dans notre contexte, un utilisateur doit répondre à un scrutin, celui-ci après authentification par un mot de passe, il ne peut voter que par oui ou non, son choix doit être limité à deux réponses possibles. Le résultat de son vote ne peut pas être connu par quiconque intercepte le message, il souhaite garder la confidentialité de son vote. Quand le serveur reçoit une réponse, il ne peut pas connaître le verdict, il sait seulement que l'utilisateur a voté. Enfin l'administrateur souhait aussi connaître le résultat du scrutin.

Dans notre projet, nous cherchons à créer une application de vote électronique apportant les avantages mentionnées d'un système de ce type et garantissant la confidentialité des votes et donc leur chiffrement. Or, nous n'envisageons pas de faire face aux autres risques de sécurité que nous venons de mentionner dans la partie « Présentation du sujet ».

1.1.3 Analyse de l'existant

Nous avons découvert un logiciel très semblable à ce que nous souhaitons faire du nom de Belenios. En effet, le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM) aurait utilisé ce logiciel pour des élections lorsque nous y sommes allés pour avoir des réunions avec notre tuteur.

Belenios est une application de vote en ligne disponible sur tout navigateur qui garantit la sécurité et fiabilité d'un scrutin :

— Sécurité : le vote est secret, personne ne peut connaître le vote d'un votant.

— Fiabilité : Chaque votant peut voir si son vote à bien été chiffré et peut aussi recompter les votes. De plus chaque vote est bien celui d'un votant car un procédé de signature a été mis en place.

Pour son utilisation Belenios demande une authentification par login et mot de passe spécifique à une élection particulière tenu par un administrateur. Ce couple login et mot de passe est directement envoyé aux votants via leurs adresses mails.

De plus Belenios utilise la méthode de chiffrement asymétrique d'ElGamal que nous souhaitons également implémenter dans notre application. Afin de vérifier la validité d'un vote, l'application se sert de la Preuve à divulgation nulle de connaissance (zero knowledge proof). C'est un protocole qui permet de prouver qu'une proposition est vraie sans toutefois révéler d'autres informations.

Belenios est application de vote électronique fiable, cependant elle est loin d'être la seule sur le marché. En effet il existe de nombreuses autres applications de vote en ligne avec chacune leurs spécificité et leurs ressemblances. En voici quelques exemples :

— Helios : logiciel libre où l'électeur vote « oui » ou « non ».
— Neovote : cryptage des communications via l'utilisation du Protocole SSL/TLS.
— Balotilo : simple, les bulletins ne sont pas liés à un électeur à la fin de l'élection.

Finalement à l'aide de nos connaissances, de notre expérience et de notre persévérance nous devons réaliser une application de vote électronique en ligne originale. Cette application devra être suffisamment sécurisée et fiable, s'inspirant du vote traditionnelle mais également d'applications déjà existantes telle que *Belenios*.

Le vote en ligne possède différents avantages (rapidité et réduction de frais) et inconvénients (sécurité et fraudes) que nous devrons prendre en compte.

Le vote ne sera pas anonyme mais chiffré permettant ainsi aux utilisateurs de voir leurs votes assurant la fiabilité, de même on utilisera la méthode de chiffrement asymétrique d'ElGamal pour assurer la sécurité.

C'est alors que le projet prend tout son sens puisque son but est de nous initier aux protocoles d'échange de messages confidentiels et de comprendre comment fonctionnent les méthodes de chiffrement.

1.2 Analyse des besoins

1.2.1 Besoins fonctionnels

Les besoins fonctionnels du client sont détaillés au travers du Product Backlog sur lequel nous avons travaillé. Les besoins fonctionnels s'expriment au travers des user stories, artifices propres à la méthodologie agile SCRUM utilisée pour la gestion de notre projet, cette méthodologie et l'utilisation du Product Backlog sont expliquées plus en détail dans la partie « Gestion de projet » du rapport.

Les besoins fonctionnels suivants expriment ce que les différents acteurs interagissant avec l'application veulent faire et pourquoi. Nous avons identifié trois types d'utilisateurs de l'application : l'utilisateur général qui représente le concept abstrait d'un utilisateur ayant des besoins généraux, le votant et l'administrateur du scrutin, ces derniers sont des utilisateurs concrets qui ont les mêmes besoins que l'utilisateur général, mais aussi des besoins spécifiques.

Par exemple, l'administrateur veut pouvoir gérer le scrutin et le votant veut pouvoir voter. De cette manière, nous avons analysé et défini les besoins fonctionnels au travers des user stories, qui représentent des incréments fonctionnels de l'application. Pour déterminer qui était le demandeur, quelle était la demande et quels étaient les objectifs de la demande, nous avons utilisé la phrase connue des adeptes des méthodes agiles : « En tant que... je veux... afin de... ».

Nous présentons donc les besoins fonctionnels comme suivent :

En tant que	Je veux	Afin de
Utilisateur général	Avoir une interface graphique complète pour l'application	Pouvoir interagir visuellement avec l'application
	Avoir un formulaire de connexion	Pouvoir m'authentifier et me connecter
	Avoir les sections de l'interface graphique organisées dans un menu	Pouvoir accéder facilement aux différentes fonctionnalités de l'application
	Consulter les votes chiffrés qui ont été enregistrés jusqu'au moment	Vérifier que les votes enregistrés ont été bien chiffrés
	Avoir le résultat déchiffré à la fin de la votation	Connaître le résultat du scrutin à la fin de la votation
Administrateur	Disposer d'une base de données ayant un lien avec le serveur du système	Intéragir avec la base de données de l'application
	Disposer des données des votants	Savoir qui sont les votants
	Avoir la possibilité de mettre à jour la liste de votants du scrutin	Gérer les votants participant à la votation
	Avoir un flag indiquant si le votant a voté ou pas	Savoir quels votants ont voté
	Calculer le résultat de la votation	Mettre fin à la votation
	Réinitialiser le referendum du scrutin	Disposer d'un nouveau referendum
Votant	Voter depuis une application cliente	Voter depuis n'importe où
	Voter sans que le vote soit connu : avoir le vote chiffré	Garder la confidentialité de mon vote

FIGURE 4 – *Backlog product* : besoins fonctionnels

1.2.2 Besoins non fonctionnels

Les besoins non fonctionnels concernent les besoins de l'équipe de développement pour concevoir et créer l'application.

En outre, la même phrase utilisée pour définir les besoins fonctionnels a été utilisée pour définir les besoins non fonctionnels, appelés technical stories dans le

cadre de la méthodologie SCRUM, qui ne représentent pas un incrément fonctionnel pour le client mais qui sont nécessaires pour satisfaire ses besoins et, dans notre cas, réaliser le projet. Par exemple, pour garantir le critère de la confidentialité du vote, nous avons mis en place la technique de chiffrement à clé publique, expliquée dans la partie « Rapport Technique », et pour mettre en place cette technique il a fallu que nous développions des méthodes utilisant l'Arithmétique modulaire au préalable. Nous présentons donc les besoins non fonctionnels comme suivent :

En tant que	Je veux	Afin de
Développeur	Créer des diagrammes pour la conception du projet	Analyser et concevoir le projet
	Mettre en place un gestionnaire de sources pour le code du programme (SVN)	Pouvoir collaborer avec des autres développeurs
	Disposer d'un système d'organisation de tâches (Trello)	Attribuer les tâches à réaliser aux membres de l'équipe
	Se renseigner sur les différents types de chiffrement (symétrique, asymétrique)	Savoir quelle méthode implémenter pour le chiffrement des votes
	Disposer d'un diagramme de Gantt pour le projet	Visualiser les dates limites et les dépendances des tâches
	Implémenter la structure de classes du système de votation	Utiliser la programmation orientée objets
	Mettre en place une interface client-serveur	Utiliser des sockets pour la communication de l'application client-serveur
	S'initier à la manipulation d'une bibliothèque graphique (Swing)	Mettre en place une interface graphique
	Disposer d'une classe pour l'utilisation des grands entiers et des entiers modulaires	Pouvoir manipuler des grands entiers et des entiers modulaires
	Implémenter des méthodes opérant des grands entiers et des entiers modulaires	Réaliser les opérations modulaires nécessaires au chiffrement des votes
	Implémenter des méthodes nécessaires au chiffrement	Pouvoir implémenter le chiffrement du vote
	Rédiger les parties introduction, cahier de charges et rapport technique du rapport du projet	Avoir un compte rendu du projet
	Rédiger les parties résultats, gestion de projet, conclusion et ajouter des annexes au rapport du projet	Avoir un compte rendu du projet

FIGURE 5 – *Backlog product* : besoins non fonctionnels

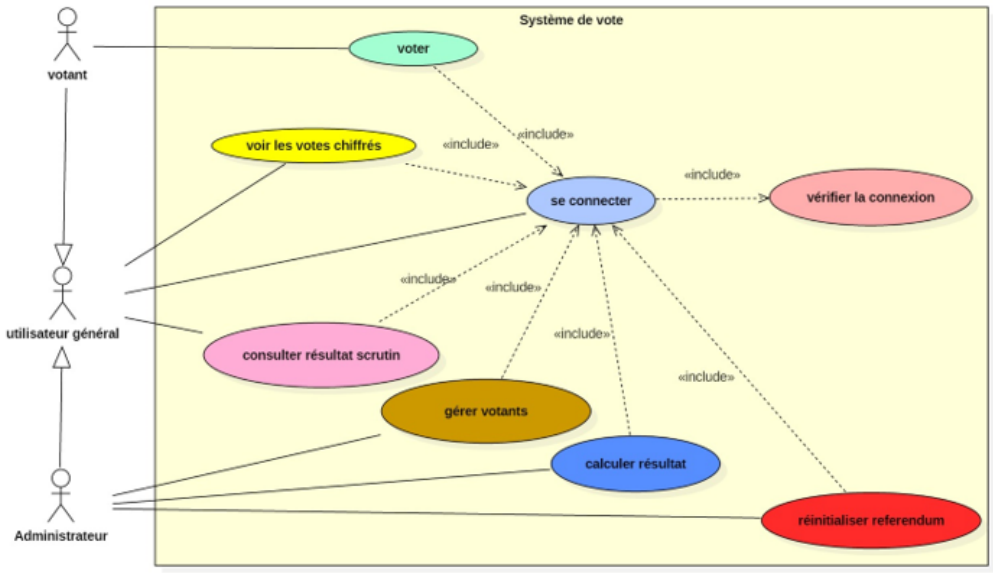


FIGURE 6 – Diagramme de cas d'utilisation final

Lors de l'utilisation de l'application de vote, trois acteurs seront nécessaires. Tout d'abord, un administrateur qui ne peut pas voter et qui gère les scrutins ainsi

que les votants. Ensuite de multiples votants qui représenteront le coté client. Enfin les actions qui pourront être réalisées par ces deux derniers seront réalisées par l'utilisateurs général.

Détaillons les différentes actions que peuvent réaliser ces acteurs :

Administrateur :

- *Gérer les votants* : suppression ou création d'un nouvel votant et mise à jour la base de données.
- *Calculer résultat* : clos le scrutin et permet aux votants de voir les résultats.
- *Réinitialiser referendum* : permet de réinitialiser les votes de tous les votants et de créer un nouveau scrutin.

Votant :

- *Voter* : le votant fait un choix entre deux réponses souvent oui ou non (pour nous 1 ou 0) afin de répondre à la question du scrutin. Ce vote est ensuite chiffré côté client et est stocké dans la base de données en attendant le calcul du résultat.

Utilisateur :

- *Voir les votes chiffrés* : pour montrer le fonctionnement de l'application et assurer sa fiabilité, les utilisateurs pourront voir comment ont été enregistrés les votes dans la base de données.
- *Consulter résultats* : Le résultat final est déchiffré. Après cette action les votants ont accès au résultat du scrutin.
- *Se connecter* : l'utilisateur envoie ses identifiants afin de pouvoir accéder aux différentes fonctionnalités de l'application, selon son rôle : votant ou administrateur.
- *Vérifier connexion* : protocole réalisé par le serveur afin de vérifier les identifiants du votant ou de vérifier l'authentification de l'administrateur.

Comme vous pourrez voir en annexe, le diagramme de cas d'utilisation final a un peu changé en comparaison avec le prototype initial. Au fur et à mesure que le projet avançait, des modifications ont été effectuées :

- Au début l'administrateur était aussi un votant mais pour bien séparer les rôles nous avons opté pour un unique administrateur avec des tâches bien précises.
- Nous pensions aussi réaliser des statistiques ou des graphiques illustrant des différents aspects de la votation, or avec le temps nous nous sommes rendus compte que ce n'était pas un point central du projet donc nous avons décidé de ne pas implémenter une telle fonctionnalité.

— Enfin nous voulions aussi permettre aux votants de changer leurs adresses mails ou leurs mot de passe, cependant après réflexion nous avons trouvé très peu d’utilité à nous concentrer dessus.

2 Rapport technique

2.1 Formalisation mathématique

Pour travailler la cryptographie, nous allons travailler dans un ensemble d'entiers modulaires.

$\mathbb{Z}/p\mathbb{Z}$ qui correspond à l'ensemble $\{0, 1, ..p-1\}$

2.1.1 Opérations modulaires

La cryptographie va également nécessiter la manipulation des entiers modulaires à travers des opérations modulaires. À travers cette partie, nous allons voir comment ces opérations fonctionnent.

p premier, a, b où a et b sont positifs $\in \mathbb{Z}/p\mathbb{Z}$

Addition modulaire :

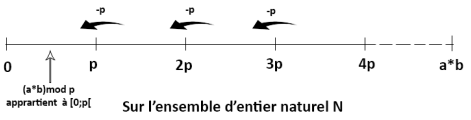
$$a + b \pmod p = \begin{cases} a + b & \text{si } a + b < p \\ a + b - p & \text{si } a + b \geq p \end{cases}$$

Soustraction modulaire :

$$-a \pmod p = \begin{cases} p - a & \text{si } a \neq 0 \\ a & \text{si } a = 0 \end{cases}$$

Multiplication modulaire :

$a \times b \pmod p = a \times b - kp$ de telle sorte que $0 \leq a \times b \pmod p \leq p$



Puissance modulaire :

$$a^b \pmod p = \Big\{ \quad (a \times \cdots a) \pmod p \quad \text{b fois}$$

Exemples : p (nombre premier) = 7 ; a = 3 ; b = 5

$$a + b \pmod p = 8 - 7 = 1$$

$$-a \pmod p = 7 - 3 = 4$$

$$a \times b \pmod p = 15 - 2 \times 7 = 1$$

2.1.2 Définitions : générateur et petit théorème de Fermat

Soit $G = \mathbb{Z}/p\mathbb{Z}^* = \mathbb{Z}/p\mathbb{Z}$ où 0 n'est pas inclus.

Générateur.
- $\exists$ a $\in$ G, 1, a, a^2, a^3, ..., a^{p-2} (nous avons p - 1 éléments), soient 2 à 2 distincts : a est un générateur de G
- Si a générateur, G = $\{$ 1,a^2, a^3, ..., a^{p-2} $\}$
- Taille de boucle quand a est générateur : p-1 = |G| (nombre d'éléments de G)

Petit théorème de Fermat. $a^{p-1} \equiv 1 \pmod p \ \forall a \in \mathbb{Z}/p\mathbb{Z}$

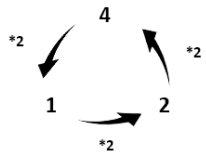
Exemple : p = 7 ; a = 2 (a générateur ?) ;

x		0	1	2	3	4	5
a^x		1	2	4	8	16	32
a^x	$\pmod p$	1	2	4	1	2	4

Notre boucle est de taille 3, c'est à dire quand nous avons $a^x \pmod p$ = 1 soit $a^0 = a^3 = 1$.

Comme notre taille de boucle est inférieur à p - 1, a = 2 n'est pas générateur.

p = 7 ; a = 3 (a générateur ?)

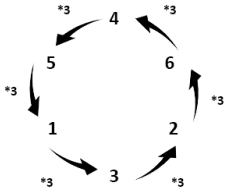


Taille de la boucle : 3

x		0	1	2	3	4	5	6
a^x		1	3	9	27	81	243	729
a^x	$(\mathrm{mod}\ p)$	1	3	2	6	4	5	1

Notre boucle est de taille 6, c'est à dire quand nous avons $a^x \pmod p = 1$ soit $a^0 = a^6 = 1$.

Comme notre taille de boucle est égale à p - 1, a = 3 est générateur.



Taille de la boucle : 6 = p-1

2.1.3 Relation entre la taille de la boucle et le générateur

Soient p un nombre premier et a un nombre appartenant à l'ensemble des nombres modulo p ($a \in \mathbb{Z}/p\mathbb{Z}$).

De plus, on peut dire que la taille de la boucle de a générée par p est :

— égale à $p-1$ si a est générateur de p;

— diviseur de $p-1$ si a n'est pas générateur de p.

D'après le petit théorème de Fermat, on peut écrire :

$$a^{p-1} \equiv 1 \pmod p \ \ \forall a \in \mathbb{Z}/p\mathbb{Z}$$

Proposition 1. *Soit $tb(a)$ la taille de la boucle de a.*

On peut donc poser que $a^{tb(a)} = 1 \pmod p = a^{2*tb(a)} = a^{3*tb(a)} = \ldots$

De plus, $tb(a)|(p-1)$ ($tb(a)$ divise $(p-1)$).

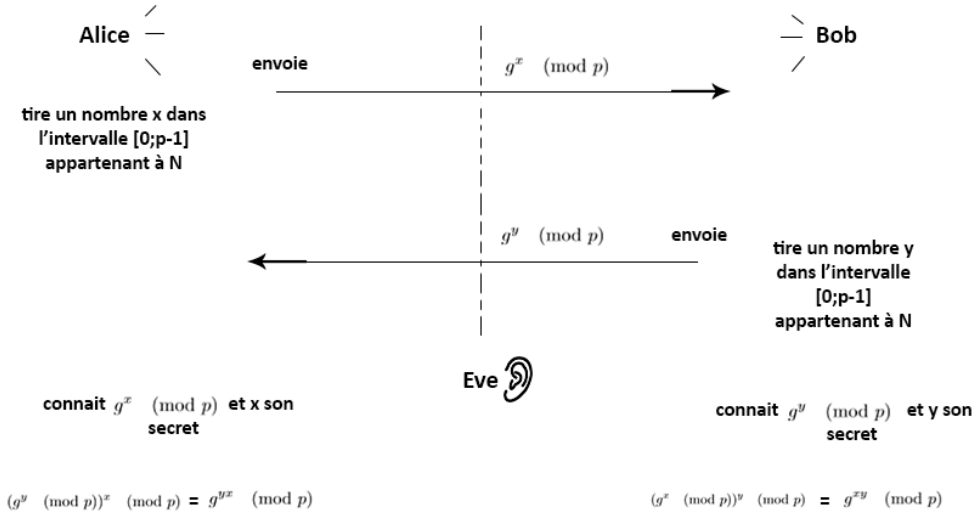
Cas particulier : p = 2q + 1

Posons q un nombre tel que $p = 2q + 1$.
Puisque $tb(a)|(p-1)$, on peut écrire : $tb(a)|2q$.
Les diviseurs possibles de $2q$ sont $2q$, 2, q et 1. On a alors :

a générateur $\iff tb(a) = 2q = p-1 \iff \begin{cases} a^q \not\equiv 1 \pmod p & \Rightarrow tb(a) \neq q \\ a^2 \not\equiv 1 \pmod p & \Rightarrow tb(a) \neq 2 \end{cases}$

2.1.4 Protocole d'échange de clé

Alice et Bob se mettent d'accord sur un secret sans que leur communication ne révèle rien du secret.



donc K = g$^{xy} \pmod p$ est un secret partagé dans G Eve écoute les communications. Eve connaît g$^x \pmod p$ Question : Est-ce qu'Eve peut calculer g$^{xy} \pmod p$? Eve pourrait vouloir calculer (g$^x \pmod p$)$^x \pmod p$ mais elle ne connait pas x et trouver g^x est trop coûteux.

Cette dernière assertion est prouvée dans la partie 2.2.4, à l'aide du log.

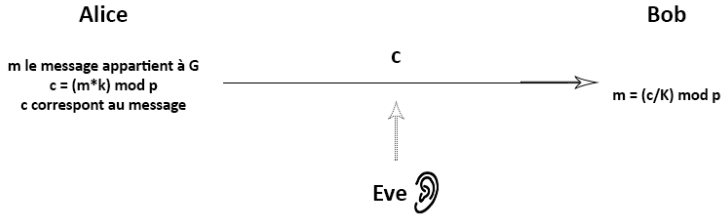
2.1.5 Chiffrement symétrique

Le chiffrement symétrique est la technique la plus ancienne et connue. Une clé secrète, qui peut être un numéro ou simplement une chaîne de lettres dans le désordre, est appliquée au texte d'un message pour modifier le contenu d'une

certaine manière. Cela pourrait être aussi simple que de décaler chaque lettre d'un certain nombre d'emplacements dans l'alphabet.

Supposons qu'Alice et Bob possèdent une clé secrète.

$$K = g^{xy} \pmod{p} \in G \text{ (le groupe secret qu'ils partagent)}$$



Eve connait C mais pas n et K.

Tout message m' peut être chiffré en c (avec une autre clé K').

2.1.6 Chiffrement asymétrique de El-Gamal

Le problème avec les clés secrètes est de les échanger sur un réseau de grande taille tout en évitant qu'elles ne tombent pas dans de mauvaises mains. Quiconque connaissant la clé secrète peut décrypter le message.

Une réponse est un chiffrement asymétrique, dans lequel il y a deux clés liées. Une clé publique est librement disponible pour quiconque voudrait nous envoyer un message. Une seconde, une clé privée est gardée secrète que vous sommes le seul à connaître.

Tous les messages (texte, fichiers binaires ou documents) cryptés à l'aide de la clé publique peuvent uniquement être décryptés en utilisant la clé privée correspondante.

Ce qui signifie que nous ne devons pas nous inquiéter sur la circulation des clés publiques sur Internet. Toutefois, le problème peut résider dans le fait que le chiffrement asymétrique est plus lent que le chiffrement symétrique (plus de traitement).

L'algorithme ElGamal est un chiffrement asymétrique fondé sur les logarithmes discrets. Il a été créé par Taher Elgamal.

G = $\{a^0, a^1, ..., a^{q-1}\}$ (avec $a^q \equiv 1 \pmod{p}$

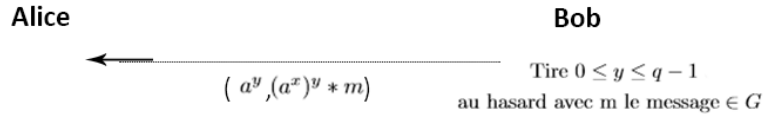
Clé publique : (a^x) que tout le monde connait

Clé privée : (x) avec 0 $\leq x \leq q-1$ Seulement Alice la connait

(c1, c2) : message chiffré

c1 : $a^y \in G$

c2 : $(a^x)^y * m \in G$



Alice déchiffre avec l'opération :

$$c2/c1^x = \frac{(a^x)^y * m}{a^y} = m$$

N'importe qui peut envoyer un message à Alice mais seule Alice peut le déchiffrer.

2.2 Conception

2.2.1 Choix technologiques : bibliothèques et API

L'application est entièrement codée en Java, et utilise un certain nombre de bibliothèques particulières, nécessaires à son bon fonctionnement. Les plus importantes sont présentées ici.

BigInteger : gestion de grands nombres La classe BigInteger de Java (java.math.BigInteger) permet de travailler sur de grands nombres entiers. Ces nombres sont codés sur un nombre théorique infini de bits, puisque la classe alloue dynamiquement autant de mémoire que nécessaire pour stocker la valeur demandée. La seule limite possible est donc une limite pratique, dictée par la quantité de mémoire disponible sur le système (plus d'informations sont disponibles dans la documentation Java de la classe, dont le lien se trouve dans la bibliographie).

La classe BigInteger est donc l'outil idéal pour stocker et travailler sur les valeurs des votes, une fois chiffrés. La conversion depuis un int est très simple (un constructeur de BigInteger est prévu à cet effet). De plus, outre les opérations les plus classiques (additions, multiplications, puissances…), la classe propose un certain nombre de fonctions permettant des opérations utiles pour ce projet. Par exemple, on peut y retrouver des opérations modulaires, comme modInverse ou modPow.

Socket : communication client-serveur La classe Socket de Java est, dans son utilisation, similaire à celles que l'on peut trouver dans d'autres langages de programmation : elle propose une interface de communication entre deux entités (généralement deux machines, un client et un serveur). Les fonctions que propose la classe simplifient son utilisation. Par exemple, le simple appel au constructeur crée un socket sur le numéro de port demandé, sans avoir besoin de gérer manuellement le bind du port ou créer un fichier de communication. De même, un simple appel à la fonction close() fermera les flux d'entrée et de sortie ouverts sur ce socket.

Cette classe est donc utile pour créer tout l'aspect client-serveur de l'application. Elle facilite la création de la partie réseau, et propose toutes les fonctions nécessaires.

API SQL : communication avec la base de données Java propose une API SQL (java.sql.*) permettant de communiquer avec des bases de données via des requêtes SQL, directement depuis le code. Un pilote, à ajouter au compilateur Java (ajout d'un *dependency*), permet de se connecter à la base de données gère la

connexion à la base de données : l'adresse de la base, un identifiant et un mot de passe suffisent.

L'application fonctionne avec une base de données Oracle, entièrement gérable via SQL. L'utilisation de cette API permet de gérer facilement la base de données, et est donc idéale pour le projet.

Swing : Bibliothèque graphique Java La bibliothèque Swing offre la possibilité de créer des interfaces graphiques pour n'importe quel système d'exploitation sous-jacent, au prix de performances moindres qu'en utilisant *Abstract Window Toolkit* (AWT). Il utilise le principe Modèle-Vue-Contrôleur (MVC) (les composants Swing jouent en fait le rôle de la vue au sens du MVC), et dispose de plusieurs choix d'apparence pour chacun des composants standards.

De plus, l'IDE *Netbeans* contient un *Graphical User Interface* (GUI) *Builder* qui permet de créer une interface avec un système de *drag and drop*, permettant de concevoir l'interface en temps réel en générant du code ; ainsi, la personnalisation de l'interface graphique se fait de manière plus intuitive.

2.2.2 Diagramme de classes

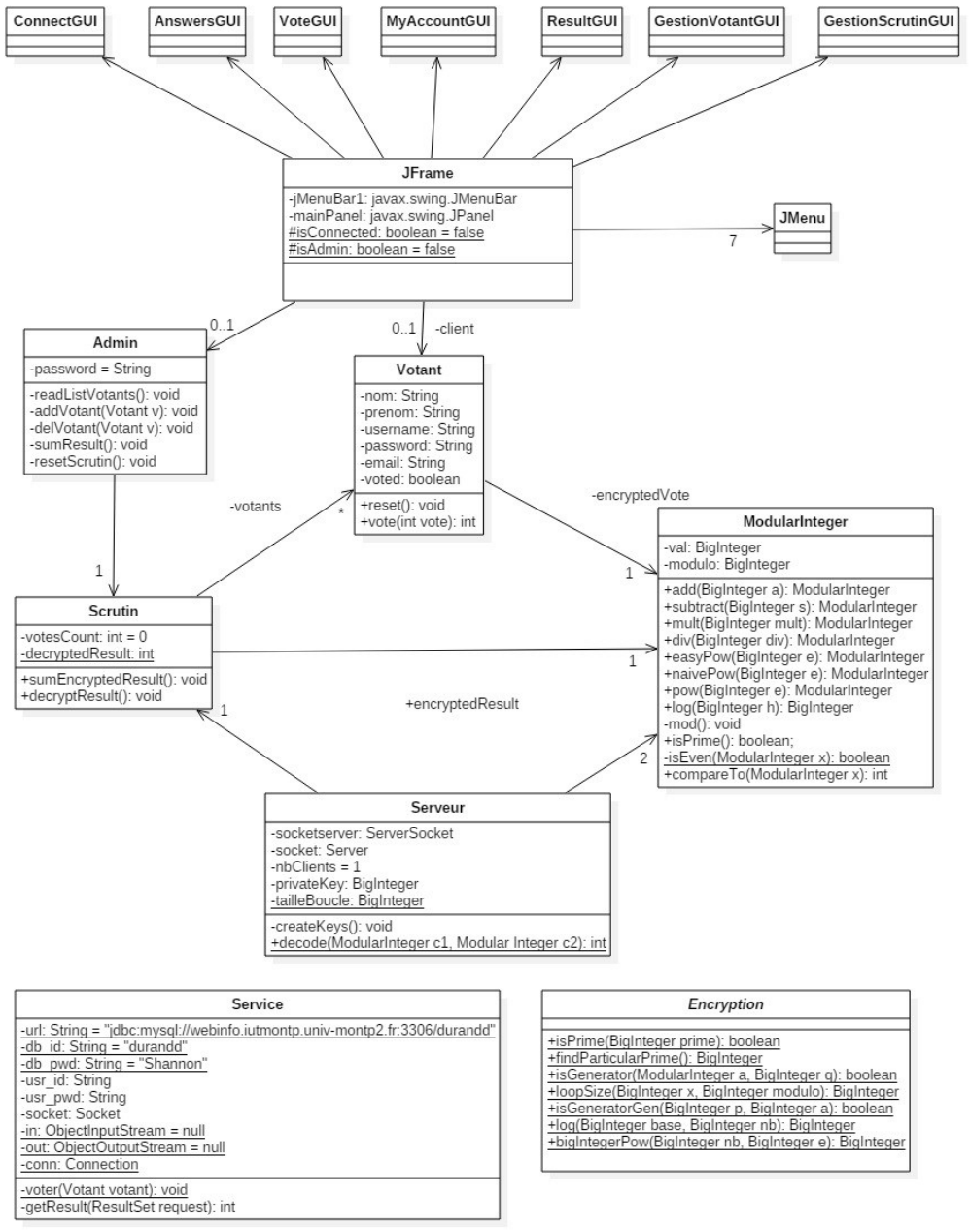


FIGURE 7 – Diagramme de classes de l'application

Ce diagramme permet de présenter les différentes classes contenues dans notre application. Notre programme fonctionne avec un système client-serveur.

Ici, notre client est JFrame soit l'interface graphique. Notre interface graphique fonctionne avec un système de cardLayout, comme un paquet de carte, on change la carte qui se trouve au sommet du paquet soit celle que l'on veut voir. Les cartes ici sont les panneaux. Le panneau principal (MainPanel) dans JFrame, est ici utilisé comme un conteneur afin de gérer les différents panneaux définis par les classes terminant par GUI. Chaque panneau contient des fonctionnalités qui sont actionnées par des boutons. Pour passer d'un panneau à un autre, nous utilisons JMenuBar qui comme son nom l'indique est la barre de Menu. Par conséquent, nous avons 7 classes JMenu qui correspondent chacune à une classe de type panneau.

La classe JFrame permet d'accéder à différentes fonctionnalités. Mais certaines fonctionnalités ne sont disponibles qu'en fonction de l'utilisateur en question. Au démarrage du client, trois menus sont accessibles par l'utilisateur comme il n'est pas connecté. Il a accès à ConnectGUI, ResultGUI et AnswersGUI. Si notre utilisateur se connecte et qu'il est un votant, il a également accès à VoterGUI et MyAccountGUI. Et si un admin se connecte celui-ci aura accès à GestionVotantGUI et GestionScrutinGUI mais pas accès aux panneaux VoterGUI et MyAccountGUI.

La classe Admin possède comme attribut, une instance de la classe Scrutin qui permet d'accéder à tous les votants et de voir tous les votes chiffrés. Il peut modifier la liste des votants. Il a également la possibilité de sommer tous les votes chiffrés afin d'obtenir le résultat du scrutin jusqu'à présent.

La classe ModularInteger permet de manipuler des grands nombres (avec BigInteger) et de les mettre sous modulo. Cette classe utilise donc l'arithmétique modulaire. Le serveur de classe Server possède la clé privée ainsi que l'accès au scrutin et tous les votes chiffrés. Avec l'aide de la classe Encryption, nous pouvons obtenir des nombres premiers ainsi que des générateurs et vérifier si un nombre BigInteger donné fait partie de ces catégories . A partir de Encryption nous pourrons également générer la clé privée et clé publique du serveur pour chiffrer un vote par la suite.

Pour finir, la classe Service permet au client de se connecter à sa base de données.

Ce diagramme de classe détient beaucoup plus de classes que celui réalisé au tout début de notre projet. Beaucoup de classes concernant l'interface graphique et le chiffrage se sont ajoutés au fil de l'avancement dans le projet. De plus, nous avons décidé d'avoir Admin et Votant comme deux classes distinctes.

2.2.3 Diagrammes de séquences

Les diagrammes présentés ici sont ceux des fonctionnalités principales de l'application. Dans une optique de simplification, on supposera que le serveur a

été lancé en amont, et qu'exception faite du premier diagramme , l'exécution des fonctions nécessite préalablement de s'être connecté au service.

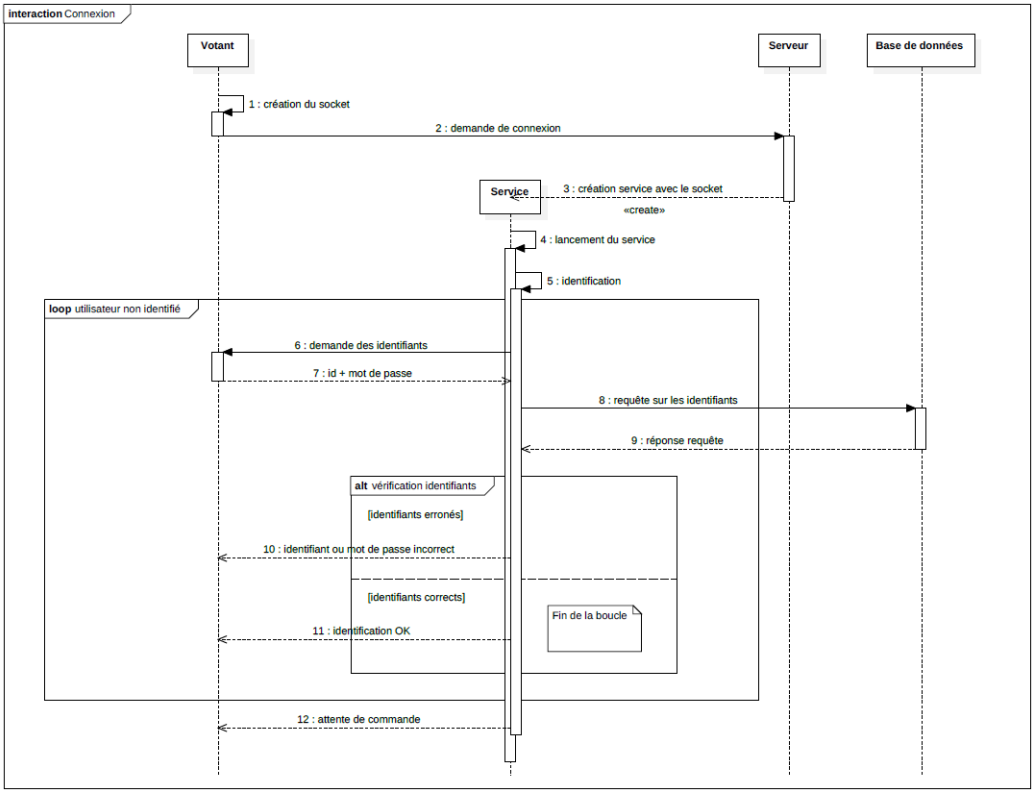


FIGURE 8 – Diagramme de séquence : connexion

Connexion Ce premier diagramme explicite les différentes étapes de la connexion d'un client (classe Votant) au système. La communication entre le serveur et le client s'effectue via une instance de la classe Socket, décrite plus haut dans ce chapitre. Le client commence donc par créer son propre socket, puis se connecte au serveur via ce socket. Le serveur est dit « multi-threadé » : pour chaque votant qui se connecte, il va créer un processus (ou *thread*) avec lequel interagira le votant. Ce *thread* est modélisé ici par la classe Service. Le serveur crée donc le service, en lui passant en argument le socket avec lequel s'est connecté le votant.

Une fois créé, le service lance le processus d'identification, correspondant à la boucle sur le diagramme. Le service demande au votant un identifiant et un mot de passe, puis vérifie dans la base de données si un couple identifiant / mot de passe correspond à celui fourni par l'utilisateur. Tant qu'il ne trouve aucune correspondance, le service continue de demander au votant ses identifiants. Une

fois une correspondance trouvée, le service met fin à la boucle et se met en attente d'ordres de la part du votant.

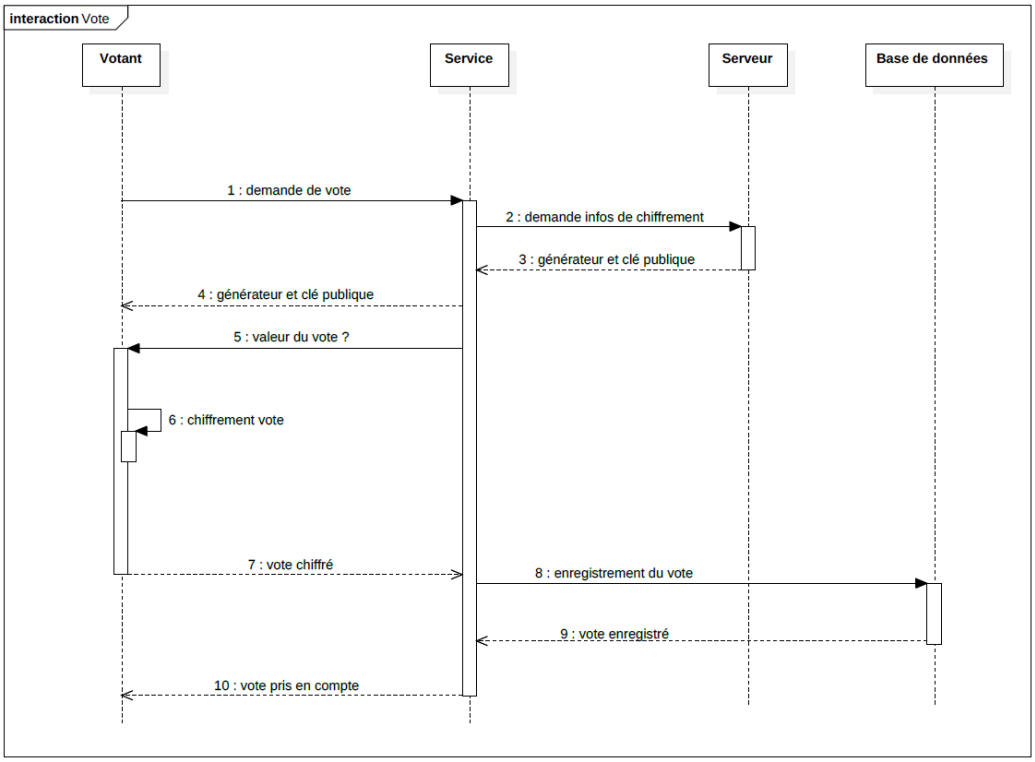


FIGURE 9 – Diagramme de séquence : vote

Vote Ce diagramme détaille le processus de vote. Comme toujours, tout part d'une demande de l'utilisateur : la classe Votant envoie une demande de vote au service via son socket. Le service récupère alors auprès du serveur le générateur et la clé publique, nécessaires au chiffrement du vote, et les transmet au votant. Ces informations sont stockées dans le serveur, puisqu'elles ne varient pas. Le votant chiffre alors son vote et le transmet au service, qui va se charger de mettre à jour la base de données avec le nouveau vote.

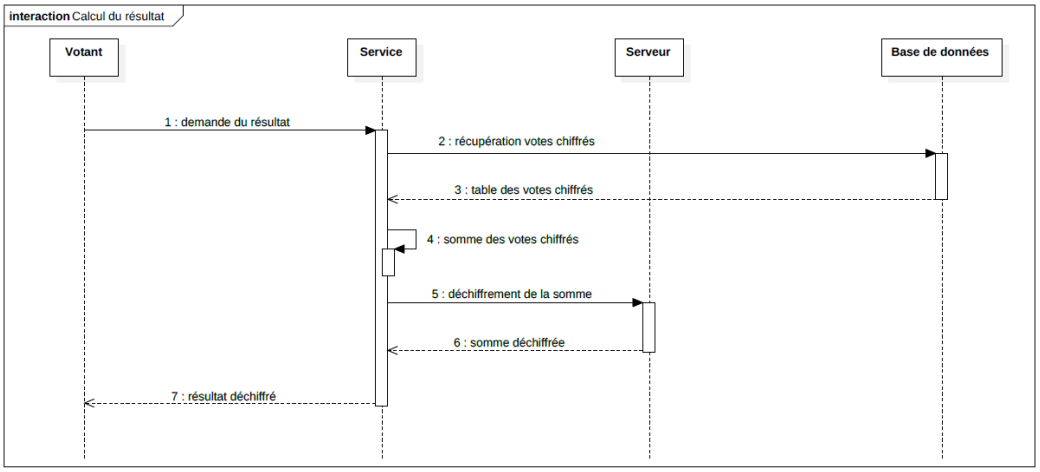


FIGURE 10 – Diagramme de séquence : calcul du résultat

Calcul du résultat On décrit ici le calcul du résultat. À la demande du votant, le service récupère les votes chiffrés dans la base de données, puis les somme (cette opération ne nécessitant pas de déchiffrage préalable, comme expliqué dans la partie mathématique du présent rapport). Le service transmet alors le résultat de cette somme au serveur, qui se chargera de le déchiffrer avant de renvoyer le résultat au service. Ce dernier pourra alors transmettre le résultat au client.

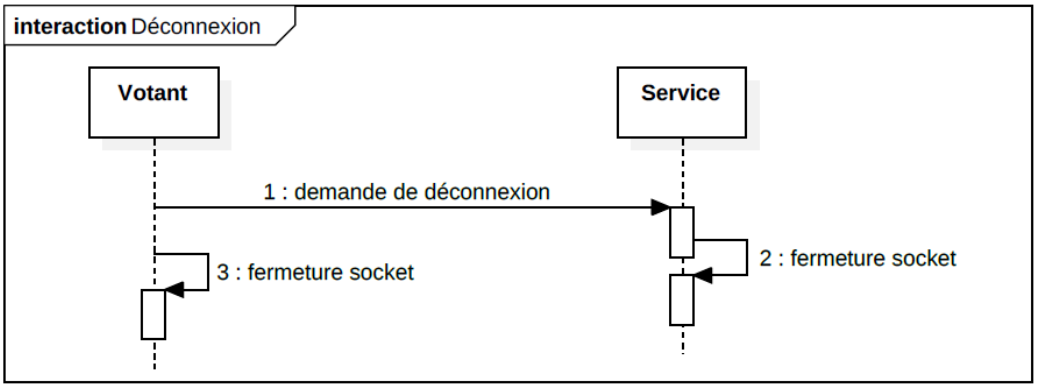


FIGURE 11 – Diagramme de séquence : déconnexion

Déconnexion Reste enfin à présenter le processus de déconnexion. Ce processus est très simple : il consiste en la fermeture des deux sockets, celui du client (ici, le votant) et celui du serveur (ici, le service). Cette fermeture est indispensable pour libérer les ressources utilisées par le socket. Une fois son socket fermé, le service (qui, ne l'oublions pas, est un *thread*) va mettre fin à son exécution. La magie

de Java et de son Garbage collector (GC) fera le reste, et libèrera les ressources utilisées par ce *thread*.

2.2.4 Algorithmes fondamentaux

En algorithmique, il existe plusieurs manières de résoudre un même problème, chacune ayant des avantages et des inconvénients. Ainsi, les algorithmes et les calculs utilisés dans le développement de l'application n'ont pas été choisis au hasard.

Les méthodes de chiffrement, en particulier, amènent à travailler sur de très grands nombres. Les questions de complexité algorithmique se posent donc ; et les choix effectués auront un grand impact sur les temps de calculs observés.

Complexité : exemple de la taille de la boucle

Trouver la taille de la boucle du générateur d'un nombre premier se fait en temps linéaire. Cela signifie que pour un nombre premier $p \approx 2^{1024}$, 2^{1024} opérations seront nécessaires.

Considérons une unité de calcul de 2GHz, qui sera donc capable d'effectuer environ 258 opérations par an. Une telle unité aura besoin d'environ 2966 années pour trouver la taille de la boucle d'un tel nombre. Un tel temps de calcul est, bien sûr, impensable, et justifie de trouver une méthode alternative pour déterminer si un nombre est générateur d'un autre nombre premier.

Des exemples de temps d'exécution sont donnés dans la partie 2.3.2 du présent rapport, avec l'analyse de la fonction *isGenerator()*. On peut aussi trouver en annexe un exemple d'algorithme (*loopSize()*) permettant de retrouver itérativement la taille d'une boucle.

Méthodes alternatives : exemple du calcul d'exponentiations

Le calcul d'un exposant est une opération assez instinctive, et se fait assez simplement : on cherche à multiplier un nombre par lui-même autant de fois que nécessaire, ce qui est un algorithme de complexité linéaire. Cependant, sur de grands nombres, le problème soulevé précédemment se pose : le temps de calcul devient bien trop important pour que l'algorithme soit exploitable.

On cherche alors à diminuer le nombre d'opérations à effectuer. Il existe un algorithme, connu sous le nom d'exponentiation rapide (ou *square-and-multiply* en anglais, parfois aussi appelé *double-and-odd*), permettant de contourner ce problème. Le principe est simple : au lieu de multiplier le nombre de manière linéaire, on va multiplier des expressions de ce nombre ayant un exposant plus petit entre elles.

Ainsi, pour calculer x^n (que l'on pourra aussi écrire *pow(x, n)*), on va calculer *pow(*x^2*,* $\frac{n}{2}$*)* si n est pair, ou $x*$ *pow(*x^2*,* $\frac{n-1}{2}$*)* si n est impair. En répétant récursivement cette écriture, on obtient un algorithme de complexité logarithmique, permettant d'éviter de trop grands temps de calcul.

Une comparaison des temps de calculs peut être trouvée dans la partie 2.3.2 de ce rapport, lors de l'analyse de la fonction *pow()*.

Utilisation de la complexité : exemple du logarithme

Il est possible de se servir de ces principes de complexité pour parvenir à ses fins. Ainsi, la méthode de chiffrement utilisée dans ce projet se fonde sur un postulat très simple : s'il est facile de calculer des exposants de grands nombres (comme montré ci-dessus), le temps de calcul d'un logarithme sera toujours bilinéaire (l'opération *log(*2^{1024}*)* nécessitera $2 * 2^{1024} = 2^{1025}$ opérations). Ainsi, s'il est impossible de calculer le logarithme d'un trop grand nombre, ce genre d'opérations est tout à fait réalisable sur des grandeurs plus raisonnables.

La méthode de chiffrement utilisée au cours de ce projet se fonde, elle aussi, sur ce principe : rappelons qu'un vote, une fois chiffré, se présente sous la forme $g^{x*y} * g^d$, avec d la grandeur à décoder, au maximum égale au nombre de votants (puisque chaque participant ne peut voter que zéro ou un) ; g le générateur, x la clé privée du serveur, et y la clé privée du client ; g, x et y sont tous trois de très grands nombres. Un vote chiffré est donc représenté par une grandeur bien trop importante pour être déchiffrée par un logarithme. Lors du déchiffrement, le vote chiffré est divisé par la valeur g^{x*y}. Il ne reste alors plus qu'à effectuer un logarithme sur g^d pour retrouver d et la valeur finale du vote. Cette opération devrait être rapide, puisque d représente un nombre à échelle humaine.

Supposons par exemple que l'ensemble de la population française, soit approximativement $64 * 10^6 \approx 2^{26}$ personnes, soit amené à voter via l'application ; il y aurait donc environ 227 opérations à effectuer. Reprenons notre unité de calcul de 2GHz : à raison de $2 * 10^9 \approx 2^{31}$ opérations par seconde, les 226 opérations seraient effectuées en $\frac{1}{16}$ de seconde environ. Ainsi, un algorithme de complexité apparemment inexploitable peut se révéler être un outil tout à fait utilisable, dans un contexte approprié.

Un exemple d'écriture de la fonction *log()*, ainsi que des tests de son temps d'exécution en fonctions de ses paramètres, est disponible dans la partie 2.3.2 de ce rapport.

2.3 Réalisation

Les fonctions présentées dans cette partie représentent le cœur du programme, et sont l'objet de tests ou d'explication. Certains compléments utiles peuvent être trouvés en annexe.

2.3.1 Application client-serveur

Côté serveur :

```
public static void main(String[] zero)
{
    Server server = new Server(2009); // creation d'un serveur le client doit connaitre le numero de port
    server.start(); // start le thread herite de Runable
    System.out.println("Server is on.");
}
```

FIGURE 12 – Fonction *main()* du serveur

```
@Override
public void run()
{

    this.createKeys();

    // try catch necessaire car connexion
    try
    {
        while(true)
        {
            // Un client se connecte on l'accepte : la connexion est etabli
            this.socket = this.socketserver.accept();
            System.out.println("Le client numero "+ nbClients + " est connecte !");
            /* après une connexion un service prend en charge le client.
            tandis que le serveur repart attendre un autre client */
            Thread t = new Thread(new Service(socket));
            // lancemet d'un thread en parallele
            t.start();

            // pour nous : on compte le nombre de clients qui se sont connecté au serveur
            nbClients++;
        }

    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

FIGURE 13 – Fonction *run()* du serveur

On voit ici la mise en application de la connexion d'un client au serveur. Lorsqu'un client tente de se connecter, le serveur accepte sa connexion (*this.socketserver.accept()*), puis crée un processus (ou *thread*) qui gèrera les interactions entre

le client et le serveur. Lors de la création du *thread*, on lui fournit le socket via lequel le serveur communique avec le client ; ainsi, le client pourra communiquer avec le service sans voir la différence entre le serveur lui-même et l'un de ses *threads*.

Côté client :

```
public static void main(String[] zero)  {

    Socket socket;
    ObjectInputStream in;
    ObjectOutputStream out;

    try {

        socket = new Socket(InetAddress.getLocalHost(),2009);// adresse du serveur : ici en local
            System.out.println("Demande de connexion");

            in = new ObjectInputStream(socket.getInputStream());
            out = new ObjectOutputStream(socket.getOutputStream());
            Scanner keyboard = new Scanner(System.in); // se connecte

            // début des communications avec le serveur
```

FIGURE 14 – Fonction *main()* du client

2.3.2 Chiffrement

Génération des clés privées et publiques :

Afin de générer la clé publique et la clé secrète nécessaires au chiffrement des votes, nous avons créé une fonction adéquate. La fonction *createKeys()* se trouve dans la classe Serveur et permet aussi de stocker le générateur utilisé.

A chaque démarrage du serveur, si aucune clé n'est instanciée, cette fonction est appelée. La fonction *findParticularPrime()* permet de retrouver un nombre premier de 128 bits de la forme $2q+1$ tel que q est aussi premier. Ensuite on appelle *findGenerator(p)* qui retourne un entier modulaire censé être un générateur de p. De plus on sait que $p-1$ est la taille de la boucle.

Finalement la clé privé et la clé publique sont créées respectivement à l'aide du générateur et de la taille de la boucle.

```java
private void createKeys()
{
    // chercher un p premier de la forme p = 2q+1 avec q premier
    BigInteger p = Encryption.findParticularPrime();

    // on trouve un generateur
    generator = Encryption.findGenerator(p);

    // on recupere la taille de la boucle, ici p-1
    tailleBoucle = p.subtract(BigInteger.ONE);

    privateKey = new BigInteger(128, new Random()).mod(tailleBoucle);
    publicKey = generator.pow(privateKey);
}
```

FIGURE 15 – Fonction de création des clés

Recherche d'un générateur :

Afin de trouver un générateur, nous utilisons la méthodologie mathématique vue dans les spécifications techniques du rapport.

Dans ce code le générateur est représenté par la variable a. Le générateur est instancié à deux puisque ni zéro ni un modulo p ne peuvent être générateur.

On incrémente a de un tant que celui-ci n'est pas générateur. La fonction retourne le générateur sous la forme d'un entier modulaire. S'il ne trouve pas de générateur, il renvoie $0[p]$.

```java
/**
 * @param p est big integer premier tel que p = 2q+1 avec q premier
 * @return un ModularInteger generateur dont la boucle est un nombre premier egal a (p-1)/2.
 * Si retourne 0[p] alors la fonction n'a pas trouve de generateur.
 */
public static ModularInteger findGenerator(BigInteger p)
{
    BigInteger two = new BigInteger("2");

    //essayer avec BigInteger p = new BigInteger("13");
    BigInteger q = (p.subtract(BigInteger.ONE)).divide(two);

    // a = 2[p]
    ModularInteger a = new ModularInteger(two, p);

    // tant que a n'est pas generateur et a <= p-1 alors a++
    while(!Encryption.isGenerator(a, q) && !a.getVal().equals(BigInteger.ZERO))
    {
        a = a.add(BigInteger.ONE);
    }

    // a la fin de la boucle a est generateur
    return a;

}
```

FIGURE 16 – Fonction de recherche d'un générateur

Test d'un générateur :

Nous allons maintenant détailler la fonction qui vérifie si un nombre modulaire est générateur d'un autre nombre. On considèrera ici que le fait que x soit générateur implique que la taille de la boucle soit égale à $p-1$.

On sait que la taille de la boucle soit est p-1 (dans ce cas, x est générateur), soit divise p-1. Puisque $p-1=2q$ avec p et q premier, $p-1$ divise 2, q, $2q$ et q1.

D'après les prérequis de la fonction, $a^1 > 1[p]$ implique que 1 n'est pas la taille de la boucle.

La fonction teste alors 2 possibilités :

— $a^q \neq 1[p] \Rightarrow q$ n'est pas la taille de la boucle.

— $a^2 \neq 1[p] \Rightarrow 2$ n'est pas la taille de la boucle.

Si les deux possibilité sont vraies, alors la taille de la boucle ne peut qu'être p-1 ; x est donc générateur.

```java
/**
 *
 * @param a l'element a tester et a>1
 * @param q tel que a.mod = 2q +1
 * @return true si l'element est generateur, false sinon.
 */
public static boolean isGenerator(ModularInteger a, BigInteger q)
{
        // si a exposant q != 1
        boolean prerequis1 = !a.pow(q).getVal().equals(BigInteger.ONE);

        BigInteger two = new BigInteger("2");
        // et si a exposant 2 != 1
        boolean prerequis2 = !a.pow(two).getVal().equals(BigInteger.ONE);

        // donc a generateur
        return (prerequis1 && prerequis2);
}
```

FIGURE 17 – Fonction de test d'un générateur

Il était aussi possible de trouver un algorithme générique pour savoir si un nombre est générateur sans avoir à utiliser un p tel que p=2q+1 ; cependant, sont temps d'exécution aurait été bien plus long. Cet algorithme est disponible en annexe.

Pour un nombre codé sur 17 bits, le temps d'exécution de la fonction générique est supérieur à 17 secondes tandis que celui de la fonction avec $p = 2q + 1$ ne dure même pas une milliseconde.

On peut imaginer maintenant le temps que cela prendrait si la fonction était appelée avec un entier codé sur 128 bits, comme dans l'application.

La complexité de cet algorithme est détaillée en 2.2.3, rubrique « Complexité ».

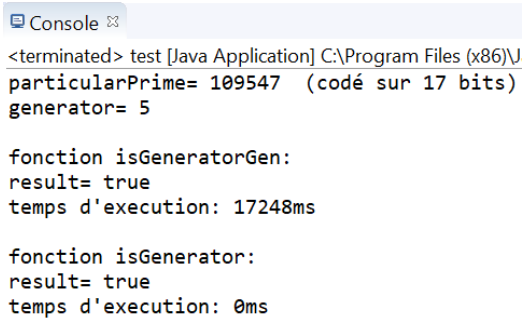


FIGURE 18 – Temps d'exécution des fonctions de test de générateurs

Au-delà du fait de savoir si un nombre est générateur ou pas, on utilise dans *isGenerator(p)* la fonction *pow(e)* des entiers modulaires. Cela permet notamment de réaliser le calcul $a^q[p]$.

Cette fonction est particulièrement intéressante, puisqu'elle utilise l'exponentiation rapide. On l'a donc réalisée sans réutiliser la fonction modPow proposée par la classe BigInteger.

On peut distinguer cinq cas :

— e < 0 : on lève une exception
— e == 0 : on renvoie 1
— e < 0 : on renvoie *this*
— e[2] == 0 : on renvoie *this.val*$^{e/2}$ * *this.val*$^{e/2}$
— sinon : on renvoie *this.val* * *this.val*$^{(e-1)/2}$ * *this.val*$^{(e-1)/2}$.

Cet algorithme est donc récursif.

Exponentiation rapide :

```
/**
 * Empowers this with a BigInteger, then applies modulo
 * @param e the BigInteger to use to empower this
 * @return this ^ e mod m
 */
public ModularInteger pow(BigInteger e) throws IllegalArgumentException
{
    // if e < 0 : throw exception
    if (e.compareTo(BigInteger.ZERO) < 0)
        throw new IllegalArgumentException("exponent must be > 0");

    // if e == 0
    if (e.compareTo(BigInteger.ZERO) == 0)
        return new ModularInteger(BigInteger.ONE, this.modulo);

    // if e == 1
    if (e.compareTo(BigInteger.ONE) == 0)
        return this;


    // two = 2 ; useful for upcoming calculations
    BigInteger two = new BigInteger("2");
    // this.val
    ModularInteger square = mult(this.val);

    // if e is even (e % 2 == 0)
    if (e.mod(two).compareTo(BigInteger.ZERO) == 0)
        // return this.valÂ²^(e/2)
        return square.pow(e.divide(two));

    // else : return this.val^((e-1)/2) * this.val
    // e = e-1
    e = e.subtract(BigInteger.ONE);
    // return this.valÂ²^((e-1)/2) * this.val
    return square.pow(e.divide(two)).mult(this.val);
}
```

FIGURE 19 – Fonction d'exponentiation rapide

Pour des grands nombres les algorithmes simples ne suffiront pas comme on peut observer sur l'image, le temps d'exécution de la fonction simple disponible en annexe est supérieur à une minute et dix secondes tandis que celui de la fonction avec exponentiation rapide dure à peu près trois millisecondes.

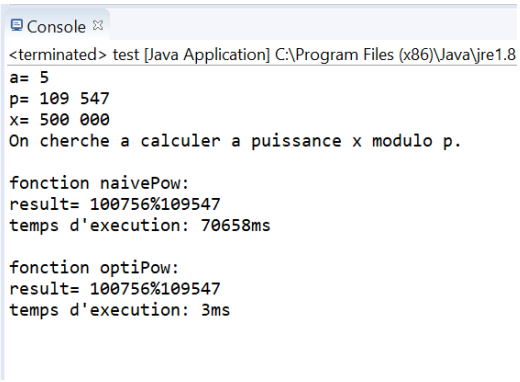


FIGURE 20 – Temps d'exécution des fonctions d'exponentiation

On peut maintenant imaginer le temps que cela prendrait si la fonction était appelée avec un entier codé sur 128 bits comme dans l'application.

La complexité de cet algorithme est détaillée en 2.2.3, rubrique « Méthodes alternatives ».

Décodage :

On a donc vu quels algorithmes nous utilisons pour trouver les clés ainsi que le générateur afin de chiffrer les votes ; nous allons maintenant voir comment le résultat est déchiffré avec la fonction *decode()*.

Tout d'abord, nous récupérons en argument les messages *c1* et *c2*, qui représentent le résultat chiffré.

On récupère ensuite la clé secrète. Puis on range dans la variable *temp* le calcul *c2/secretKey.val*. Pour finir, on retourne le résultat : $log_{generator}(temp)$.

Le résultat est :

— positif si *result>nbrParticipants/2*

— négatif sinon.

```
/**
 * decodes the given coded number
 * @param c1 the client's public key
 * @param c2 the number to decode
 * @return the decoded number
 */
public static int decode(ModularInteger c1, ModularInteger c2)
{
    /*  MATHEMATICAL EXPLANATION
        x: server's private key; y: client's private key
        g: generator; d: final result to return
        c1 = g^y; c2 = (g^(x*y) * g^d)
    */

    System.out.println("Started the decoding process");

    // K = (g^y)^x  i.e  g^(x*y)
    ModularInteger secretKey = c1.pow(privateKey);

    System.out.println("Calculated the secret key");

    // temporary result: c2 / K,  i.e  (g^(x*y) * g^d) / g^(x*y),  i.e  g^d
    ModularInteger temp = c2.div(secretKey.getVal());

    System.out.println("Calculated the temporary result");

    // final result : log_g(g^d) = d
    int result = Encryption.log(generator.getVal(), temp.getVal()).intValue();
    System.out.println(result);
    return result;
}
```

FIGURE 21 – Fonction de décodage

Calculs de logarithmes :

Finalement on peut parler d'une dernière fonction qui est la fonction log(base, nombre) qui est utilisée pour calculer le résultat final.

Le fonctionnement est simple : on instancie la variable test à un et la variable result à zéro. Puis tant que test<nb, on multiplie test par base et on incrémente result. Lorsqu'on sort de la boucle, on retourne la variable result.

```
/**
 *
 * @param base: the BigInteger used as base of the log
 * @param nb: the BigInteger such as nb = base^result
 * @return the BigInteger result such as nb = base^result
 */
public static BigInteger log(BigInteger base, BigInteger nb)
{
    // test: current power of base to compare with nb
    BigInteger test = BigInteger.ONE;
    // result: counter of test's power (test = base^result)
    BigInteger result = BigInteger.ZERO;

    // while base^result < nb
    while(test.compareTo(nb) == -1)
    {
        // test = test * base (i.e test = base^(result+1))
        test = test.multiply(base);
        // counter's incrementation
        result = result.add(BigInteger.ONE);
    }
    return result;
}
```

FIGURE 22 – Fonction *log()*

Le problème de cet algorithme réside dans sa complexité bilinéaire (2n). En effet, plus le résultat recherché est grand, plus le temps d'exécution sera long.

Ici le premier résultat est obtenu quasiment instantanément tandis que le deuxième nécessite plus de treize secondes de calcul. De plus si on recherche $log_2(1\,000\,000)$, le temps d'exécution s'allonge à plusieurs minutes.

Comme avant, on peut aussi imaginer le temps que cela prendrait si la fonction était appelée avec un entier codé sur 128 bits comme dans l'application.

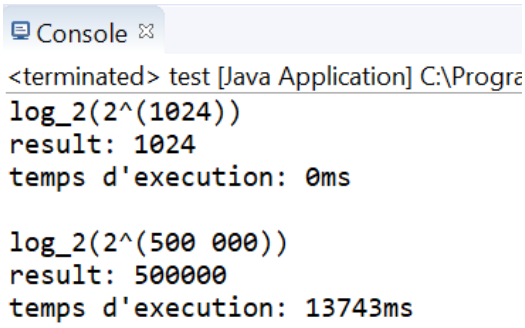


FIGURE 23 – Temps d'exécution des fonctions d'exponentiation

Remarque : ce serait aussi le temps que prendrait le plus puissant des ordinateurs pour déchiffrer la clé privée.

3 Résultats et manuel d'utilisation

3.1 Interface graphique commune à tous les utilisateurs

Dans un premier temps, nous démarrons l'interface graphique avec le serveur également en fonctionnement. Chaque panneau de l'interface graphique est accessible à travers la barre de menu se situant en haut de l'interface. Il suffit de cliquer dessus ou de passer la souris dessus pour que le panneau correspondant s'affiche.

Panneau *connect*

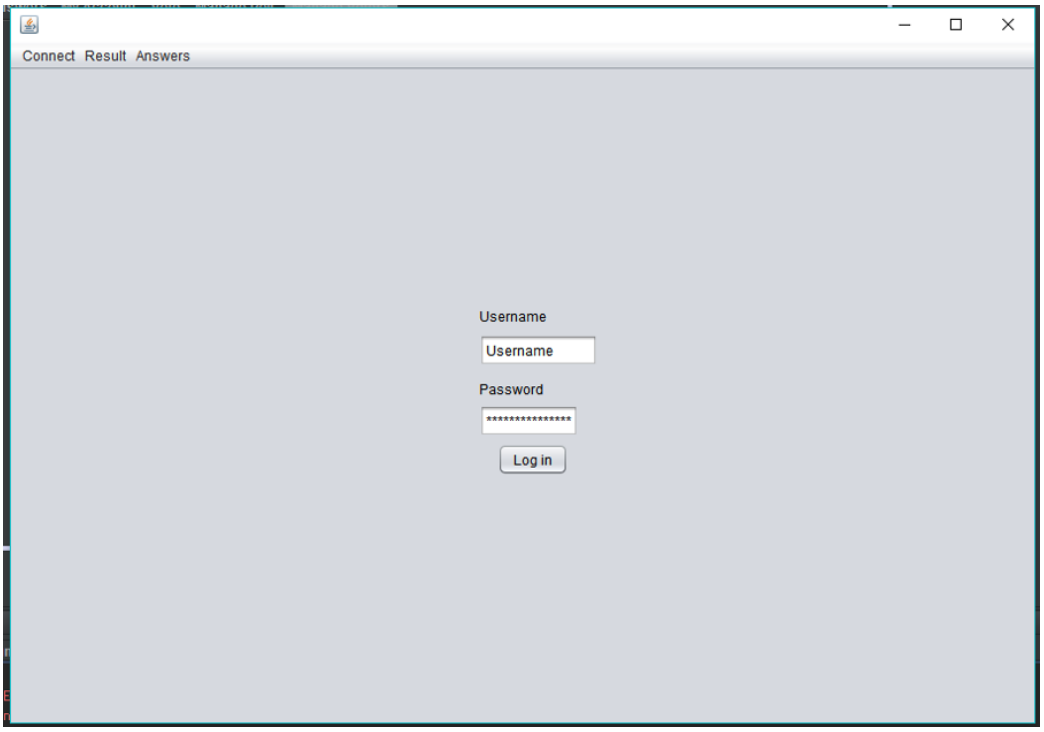


FIGURE 24 – Panneau *connect*

Quand nous avons sélectionné le menu Connect, nous accédons à l'interface suivante, il permet à l'utilisateur de se connecter s'il est votant ou admin. Qu'il soit connecté ou non, il peut quand même accéder aux panneaux Result et Answers. Pour se connecter il faut que l'utilisateur rentre son pseudo et son mot de passe dans les champs dédiés. Ensuite, il appuie sur le bouton "Log in" pour se connecter. Pour un admin, il se connectera avec le pseudo "admin" ainsi qu'avec son mot de passe.

Panneau result

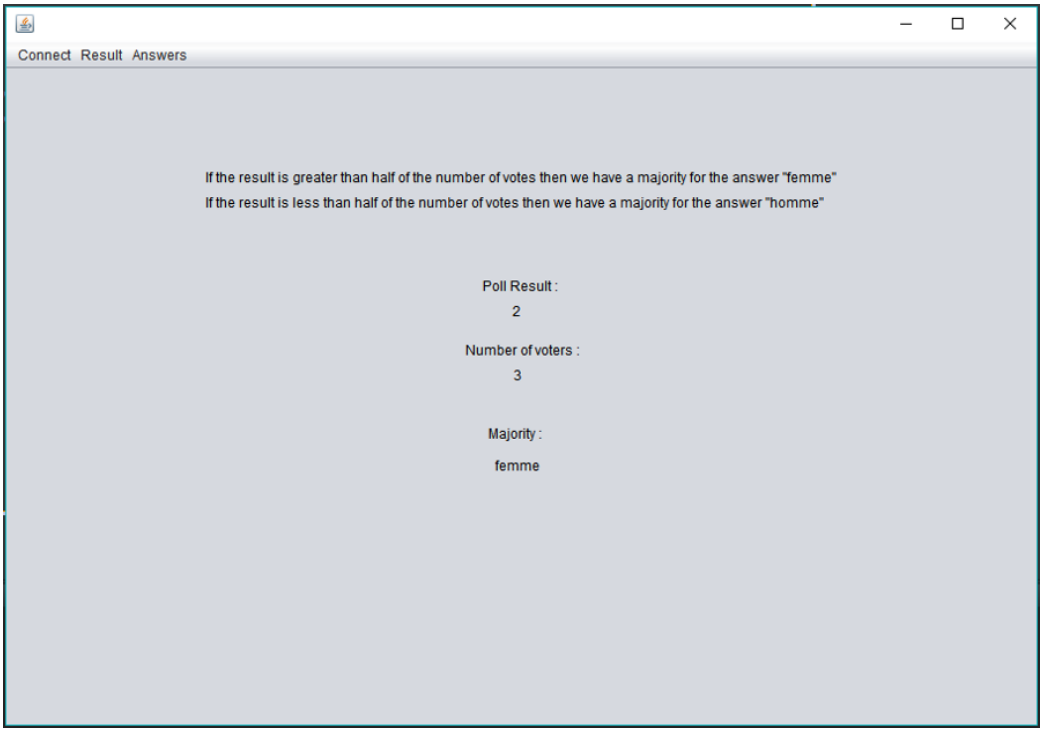


FIGURE 25 – Écran de résultats

Lorsque nous avons sélectionné le panneau Result, nous pouvons voir les résultats du scrutin. Quelques explications sont données pour comprendre comment on détermine la majorité de vote.

Panneau answers

Quand nous avons sélectionné le panneau Answers, nous pouvons observer les différents votes chiffrés du scrutin sans pour autant connaître à qui ils appartiennent.

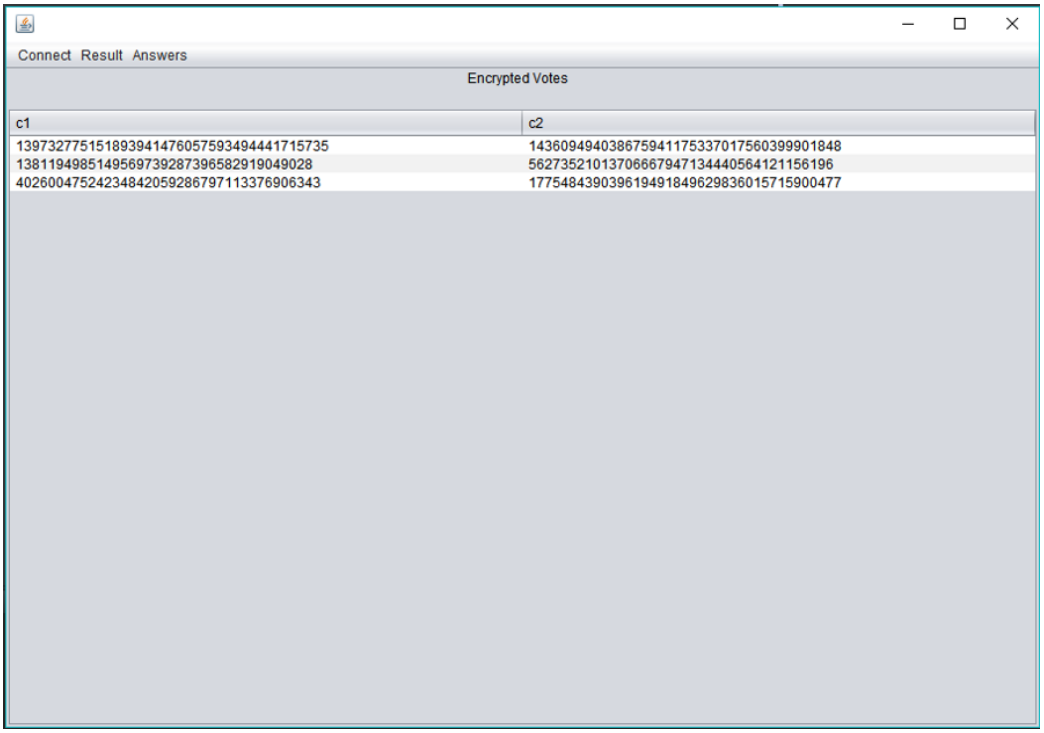


FIGURE 26 – Affichage des réponses

3.2 Interface graphique d'un votant

Panneau connect

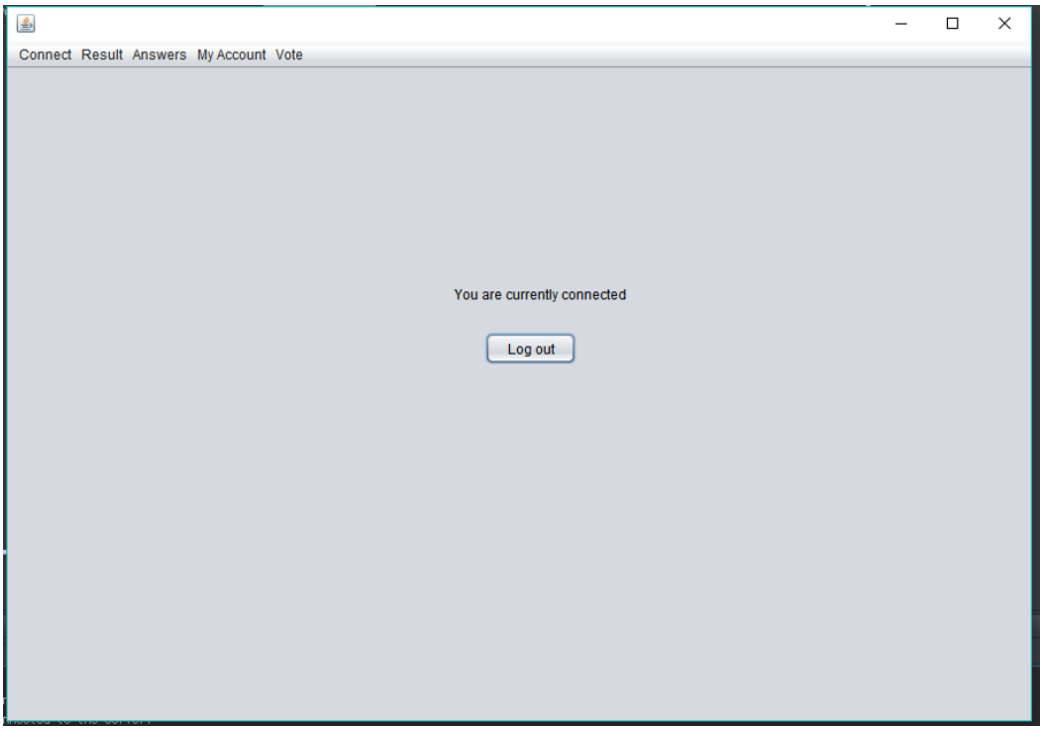


FIGURE 27 – Panneau *connect* après la connexion

Une fois le votant connecté, le panneau Connect change d'apparence et possède un nouveau bouton. Ce bouton permet au votant de se déconnecter et de revenir sur l'apparence d'une interface d'un utilisateur général. Lorsque le votant est connecté, on remarque que de nouveaux menus apparaissent, on expliquera leur panneau par la suite.

Panneau vote

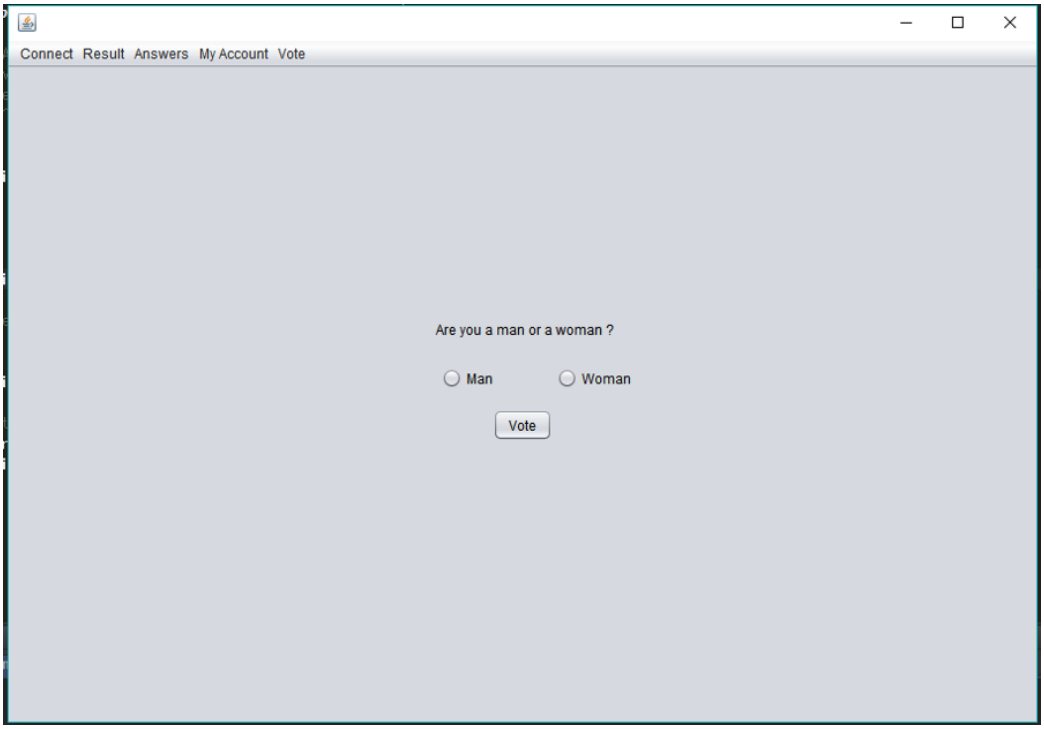


FIGURE 28 – Écran de vote

Si l'utilisateur n'a pas encore voté et qu'il sélectionne le menu Vote, il accèdera au panneau vote comme ci-dessus. Pour voter, il peut sélectionner soit Man ou Woman ; une fois son choix fait, il doit appuyer sur le bouton Vote pour confirmer son vote.

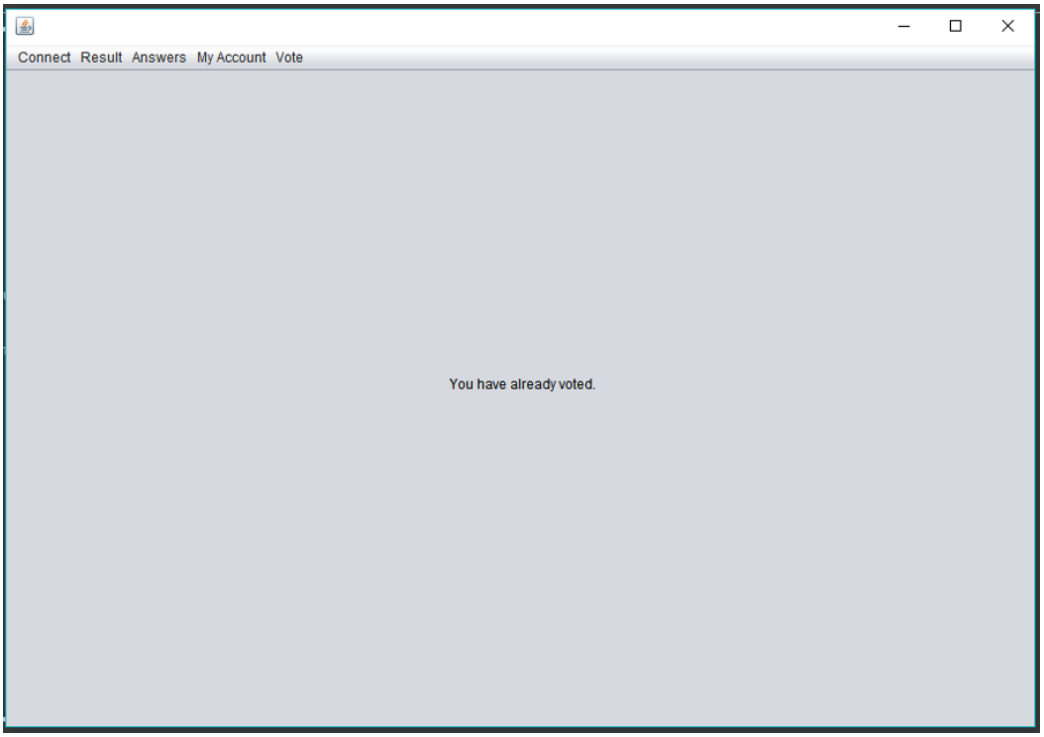


FIGURE 29 – Écran de vote après avoir voté

Panneau my account

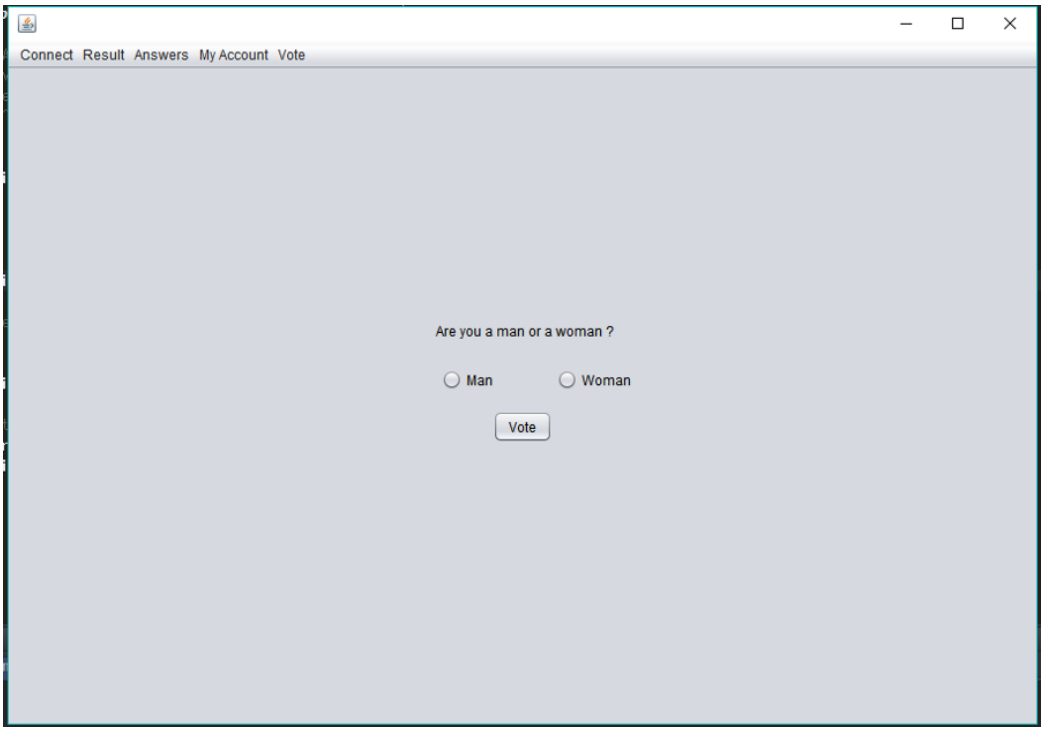


FIGURE 30 – Panneau *connect*

En sélectionnant le menu My Account, le votant peut voir les différentes informations concernant son compte ainsi que la possibilité de changer les informations le concernant. Pour changer une de ses informations, il suffit qu'il mette la nouvelle information dans le champ dédié à l'information (à droite de celle-ci) et de cliquer sur le bouton Change se situant à droite du champ.

Le votant peut également voir son vote chiffré et il a la possibilité de supprimer son vote. Cela lui permettra de voter de nouveau dans le panneau "Vote".

4 Gestion de projet

Afin d'éviter l'effet tunnel de l'organisation traditionnelle d'un projet informatique et de pouvoir avoir des retours fréquents de la part de notre client, nous avons jugé convenable d'utiliser la méthodologie agile SCRUM pour la gestion de notre projet.

Notre client et le commanditaire, appelé Product Owner dans le cadre de la méthodologie SCRUM, était représenté par M. Lebreton qui était au même temps notre tuteur de projet. Le but de ce projet étant de développer une application de vote électronique, l'un des critères les plus importants était le critère garantissant la confidentialité du vote. Nous avons donc estimé avec le client que les fonctionnalités les plus importantes de l'application étaient : la votation et l'affichage du résultat à la fin de la votation, or des fonctionnalités importantes ont été développées pour garantir la confidentialité du vote : le chiffrement des votes et le déchiffrement du résultat à la fin de la votation.

Dans cette partie du rapport, nous présenterons la démarche personnelle que nous avons suivi dans le cadre de la gestion de notre projet pour développer les fonctionnalités nécessaires à celui-ci ; nous décrirons ensuite la planification des tâches, et en dernier lieu, nous ferons un bilan critique dans lequel nous analyserons les résultats de la méthodologie mise en pratique pour atteindre les objectifs que nous avions fixé au départ, dans le cahier des charges du projet.

4.1 Démarche personnelle

Au cours de ce semestre, pendant environ deux mois et demi, nous avons dû nous organiser pour avancer dans notre projet, en parallèle de nos semaines de cours. Le projet a été découpé en six sprints, donc en six périodes limitées de temps destinées à l'avancement du projet, dont deux ont été dédiés à jeter les bases nécessaires pour le développement de l'application. Chaque sprint a eu la durée d'une semaine environ ; les durées ont pu varier en fonction du calendrier académique : partiels, vacances, jours fériés. Par conséquent, quelques sprints, notamment le premier et le dernier, ont eu une durée plus longue.

4.1.1 Méthodologie SCRUM

Dans le cadre de la méthodologie SCRUM que nous avions adoptée, l'équipe de développement chargée de créer des nouvelles fonctionnalités pour l'application, a été représentée par tous les membres de l'équipe : Dylan Durand, Sheila Barron Huerta, Jean-Clair Granier et Gaël Hollows-Pellier.

L'équipe étant auto-organisée, nous avons choisi la façon d'accomplir notre travail, sans que ce soit imposé par une personne externe. Il n'y a pas eu de notion

de hiérarchie interne : toutes les décisions ont été prises ensemble. Ce mode d'organisation a eu pour objectif d'augmenter l'efficacité de travail de l'équipe.

En outre, Dylan Durand a tenu le rôle du Scrum Master, il a donc eu pour rôle de veiller au bon fonctionnement de l'équipe et au respect des méthodes agiles. Il s'est également occupé de l'organisation, c'est-à-dire veiller à se réunir pour avancer et discuter sur le projet. Il a donc eu un rôle important, en étant garant du respect et de l'application de la méthodologie SCRUM et en facilitant la communication au sein de l'équipe.

Nous avons tous participé aux réunions tenues avec le client hebdomadairement afin d'avoir des retours fréquents de sa part, dans ces réunions nous faisons aussi le sprint review et donc un récapitulatif et des remarques sur l'avancement du projet, nous définissions aussi les objectifs et les points sur lesquels se concentrer pour le prochain sprint (cf. Annexes : Comptes Rendus des Réunions). De plus, ces réunions nous ont permis de vérifier si les tâches avaient été bien réalisées, de proposer des solutions et éventuellement de faire monter des difficultés rencontrés auprès de notre tuteur. En effet, le fait d'avoir une communication constante avec le commanditaire nous a permis d'avoir une vision globale de l'application et de prioriser le Product Backlog.

4.1.2 Product backlog

Comme expliqué précédemment dans la partie « Analyse de besoins » du rapport, nous avons utilisé le Product Backlog, aussi appelé carnet de produit, un outil propre à la méthodologie SCRUM, pour lister et décrire les besoins fonctionnels et non fonctionnels du projet, dans cette partie nous le décrirons plus en détail.

Le Product Backlog comportant six sprints, nous avons fixé les tâches à réaliser dans chaque sprint en définissant l'ordre d'exécution des technical stories à partir de leur priorité, définie à partir de leur importance et de la dépendance des besoins non fonctionnels. De même, nous avons défini l'ordre d'exécution des user stories à partir de leur estimation en termes de valeur client et d'effort. Nous avons choisi de faire d'abord les tâches ayant les valeurs client les plus importantes et les points d'effort les plus faibles.

Pour rendre l'attribution de points plus simple, nous avons décidé que les points d'effort iraient de zéro à dix. La valeur dix attribuée à la tâche la plus « difficile » à réaliser en termes d'effort et temps demandé. Nous avons encouragé une discussion ouverte sur les estimations pour finalement se mettre d'accord sur les points à attribuer à chaque story.

La valeur client a aussi été établie en utilisant une échelle, cette fois-ci allant de un à dix. Or, nous avons décidé que les stories les plus prioritaires avaient les valeurs les plus petites ; par conséquent, les stories les plus importantes à réaliser ont eu une valeur de un en termes de priorité.

Remarques

Il faut remarquer que pour satisfaire les besoins client, et donc réaliser les *user stories* ayant une valeur client, nous avons dû nous focaliser sur les besoins non fonctionnels, appelés *technical stories*, n'ayant pas de valeur client mais nous permettant de nous initier sur des techniques ou des technologies qui ont été nouvelles pour nous, par exemple : l'apprentissage de la cryptographie et des techniques de chiffrement et l'initiation à la manipulation d'une bibliothèque graphique nous permettant d'implémenter l'interface graphique de l'application. En conséquence, pendant les premiers sprints, nous nous sommes focalisés sur la réalisation des *technical stories*.

Il est aussi à noter que le *Product Backlog* a évolué au travers le temps car des nouveaux besoins se sont manifestés. En mettant ensemble les différentes *stories* et leurs estimations, nous avons établi le *Product Backlog* suivant où les besoins fonctionnels concernent la section jaune et les besoins non fonctionnels concernent la section rouge.

En tant que	Je veux	Afin de	Valeur	Effort	Priorité
Utilisateur général	Avoir une interface graphique complète pour l'application	Pouvoir interagir visuellement avec l'application	7	6	6
	Avoir un formulaire de connexion	Pouvoir m'authentifier et me connecter	5	2	5
	Avoir les sections de l'interface graphique organisées dans un menu	Pouvoir accéder facilement aux différentes fonctionnalités de l'application	4	1	7
	Consulter les votes chiffrés qui ont été enregistrés jusqu'au moment	Vérifier que les votes enregistrés ont été bien chiffrés	8	3	5
	Avoir le résultat déchiffré à la fin de la votation	Connaître le résultat du scrutin à la fin de la votation	10	4	5
Administrateur	Disposer d'une base de données ayant un lien avec le serveur du système	Intéragir avec la base de données de l'application	5	3	3
	Disposer des données des votants	Savoir qui sont les votants	5	2	4
	Avoir la possibilité de mettre à jour la liste de votants du scrutin	Gérer les votants participant à la votation	3	4	7
	Avoir un flag indiquant si le votant a voté ou pas	Savoir quels votants ont voté	6	1	7
	Calculer le résultat de la votation	Mettre fin à la votation	7	1	7
	Réinitialiser le referendum du scrutin	Disposer d'un nouveau referendum	3	2	7
Votant	Voter depuis une application cliente	Voter depuis n'importe où	8	5	5
	Voter sans que le vote soit connu : avoir le vote chiffré	Garder la confidentialité de mon vote	10	5	5
Développeur	Créer des diagrammes pour la conception du projet	Analyser et concevoir le projet	0	5	1
	Mettre en place un gestionnaire de sources pour le code du programme (SVN)	Pouvoir collaborer avec des autres développeurs	0	1	1
	Disposer d'un système d'organisation de tâches (Trello)	Attribuer les tâches à réaliser aux membres de l'équipe	0	1	1
	Se renseigner sur les différents types de chiffrement (symétrique, asymétrique)	Savoir quelle méthode implémenter pour le chiffrement des votes	0	5	2
	Disposer d'un diagramme de Gantt pour le projet	Visualiser les dates limites et les dépendances des tâches	0	2	3
	Implémenter la structure de classes du système de votation	Utiliser la programmation orientée objets	0	2	1
	Mettre en place une interface client-serveur	Utiliser des sockets pour la communication de l'application client-serveur	0	6	2
	S'initier à la manipulation d'une bibliothèque graphique (Swing)	Mettre en place une interface graphique	0	5	3
	Disposer d'une classe pour l'utilisation des grands entiers et des entiers modulaires	Pouvoir manipuler des grands entiers et des entiers modulaires	0	2	2
	Implémenter des méthodes opérant des grands entiers et des entiers modulaires	Réaliser les opérations modulaires nécessaires au chiffrement des votes	0	6	3
	Implémenter des méthodes nécessaires au chiffrement	Pouvoir implémenter le chiffrement du vote	0	7	3
	Rédiger les parties introduction, cahier de charges et rapport technique du rapport du projet	Avoir un compte rendu du projet	0	8	5
	Rédiger les parties résultats, gestion de projet, conclusion et ajouter des annexes au rapport du projet	Avoir un compte rendu du projet	0	6	6

FIGURE 31 – *Product backlog*

En ce qui concerne la répartition des tâches, au début de chaque sprint les membres de l'équipe choisissaient les tâches qui allaient le plus en accord avec ce

qu'ils voulaient faire. Nous nous mettions d'accord pour l'attribution des tâches ; de cette manière, nous nous sommes trouvés motivés à effectuer les tâches qui nous ont été attribuées.

4.1.3 Outils de communication et de travail

Pour communiquer durant le sprint, plusieurs options se sont offertes à nous. Premièrement, nous avons eu l'occasion d'avoir des réunions quotidiennes, appelés daily meetings ou mêlées quotidiennes. Nous avons pu les établir durant les journées de cours, durant nos pauses, et plus généralement durant les pauses cafés. Lors de ces réunions, chacun disait où il en était dans son travail et communiquait aux autres les éventuels problèmes rencontrés.

De plus, nous nous sommes parfois réunis pour travailler le weekend. Ces réunions nous ont permis d'avancer plus rapidement, de pouvoir nous entraider et de communiquer plus facilement. Nous avons aussi pu discuter en utilisant les différents réseaux sociaux, pour nous organiser et effectuer plus rapidement les tâches les plus urgents. D'ailleurs, nous avons mis en place des outils de collaboration en ligne pour l'avancement de l'édition du rapport, par exemple, Google Docs, Google Sheets et pour la partie « rapport technique » où nous avons détaillé les définitions et les explications mathématiques, nous avons utilisé l'outil informatique LaTeX.

En outre, nous avons utilisé SourceSup*, la plateforme d'hébergement de projets informatiques proposée par RENATER*, pour établir un système de subversion qui nous a permis d'organiser l'arborescence et les branches de développement contenant le code de l'application de vote électronique.

Le gestionnaire SourceSup utilise Subversion, logiciel de gestion de versions, distribué sous licence Apache et BSD (SVN), qui a permis aux différents membres de l'équipe de travailler en local, donc dans ses propres machines, et de transmettre les modifications du code au serveur, auquel tous les membres ont eu accès. Ainsi, nous avons pu développer le code en tenant en compte les modifications effectuées par les autres développeurs.

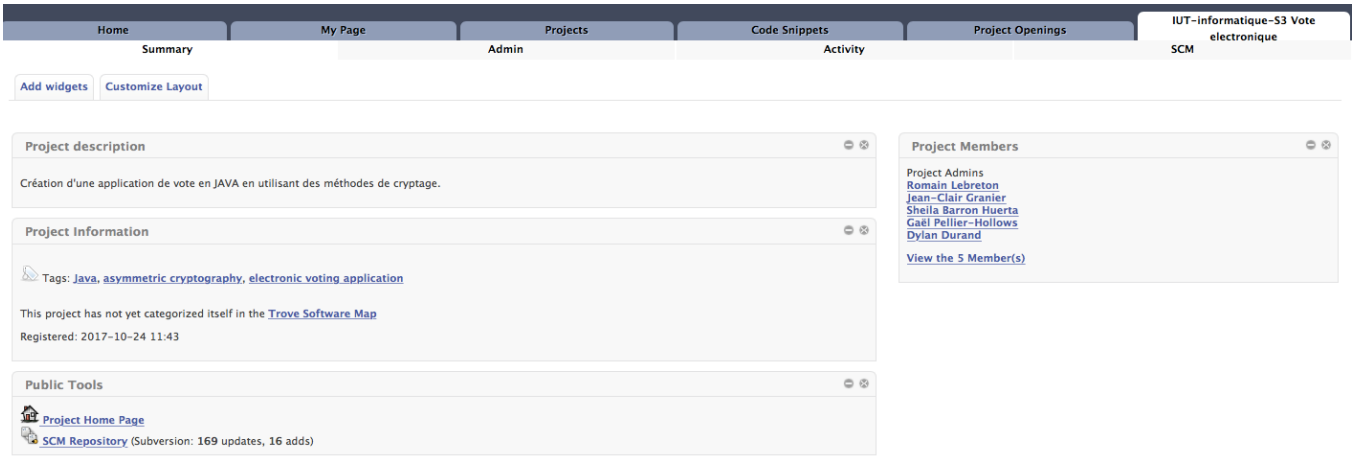


FIGURE 32 – Aperçu de la plateforme SourceSup par Renater

4.2 Planification des tâches

4.2.1 Diagramme de Gantt

Nous nous sommes servis du diagramme de Gantt, outil utilisé en gestion de projet informatique, pour représenter visuellement les différentes activités et tâches que nous avons dû mettre en place pour le développement de notre projet, ainsi comme le calendrier du projet. Cet outil a été important pour nous repérer dans le temps et pour prévoir le temps nécessaire aux différentes activités d'une manière prévisionnelle.

La colonne gauche du diagramme énumère les tâches que nous avons dû effectuer et les ressources disponibles au sein de l'équipe pour la réalisation du projet. Tandis que la ligne d'en-tête à droite représente les unités de temps, définissant le temps en jours, qui a été nécessaire pour la réalisation du projet. La ligne de temps commence en Octobre 2017, date où le projet nous a été affecté, et finit avec la soutenance du projet en janvier 2018.

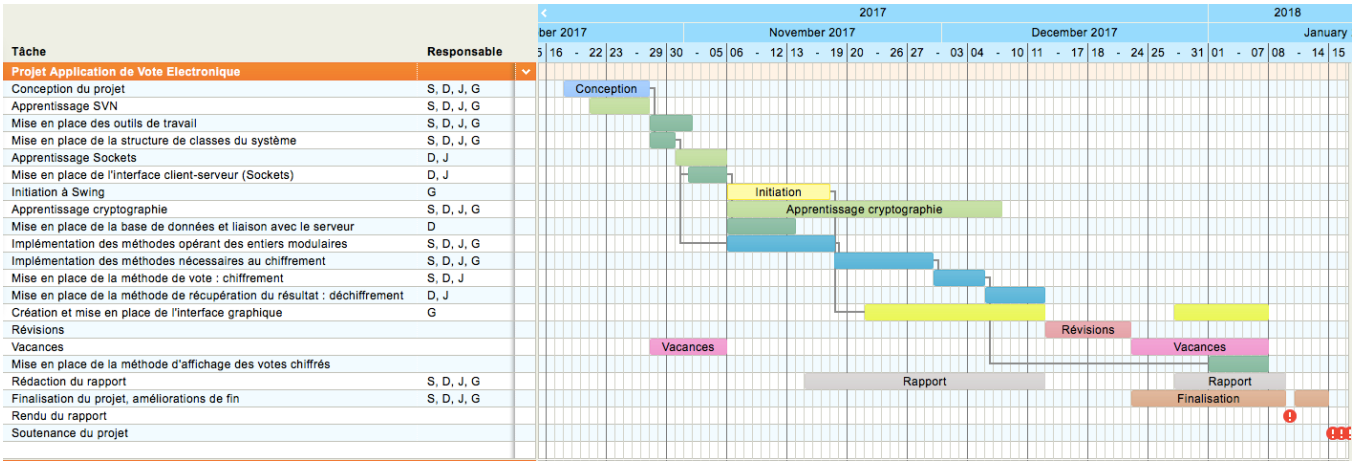


FIGURE 33 – Aperçu du diagramme de Gantt

Légende des responsables des tâches (ressources humaines au sein de l'équipe) :

— S : Sheila Barron Huerta
— D : Dylan Durand
— J : Jean-Clair Granier
— G : Gaël Hollows-Pellier

4.2.2 Sprint backlog

En début de chaque sprint, un but était décidé ; pour atteindre cet objectif, l'équipe de développement choisissait lors des réunions de planification de sprint quels éléments du Product Backlog devaient être réalisés. Ces éléments ont alors été groupés dans le Sprint Backlog, aussi appelé Carnet de Sprint, du sprint correspondant. Il faut remarquer que dans les deux premiers sprints nous n'avons pas livré des incréments fonctionnels ayant une valeur client car nous n'avons réalisé que des technical stories, ces sprints sont donc appelés des sprints 0.

Lors du développement du projet, nous avons établi six Sprints Backlogs qui ont été mis à jour régulièrement au cours des itérations afin que ceux-ci puissent donner une vision la plus précise possible de ce que l'équipe prévoyait de réaliser pour atteindre l'objectif du sprint.

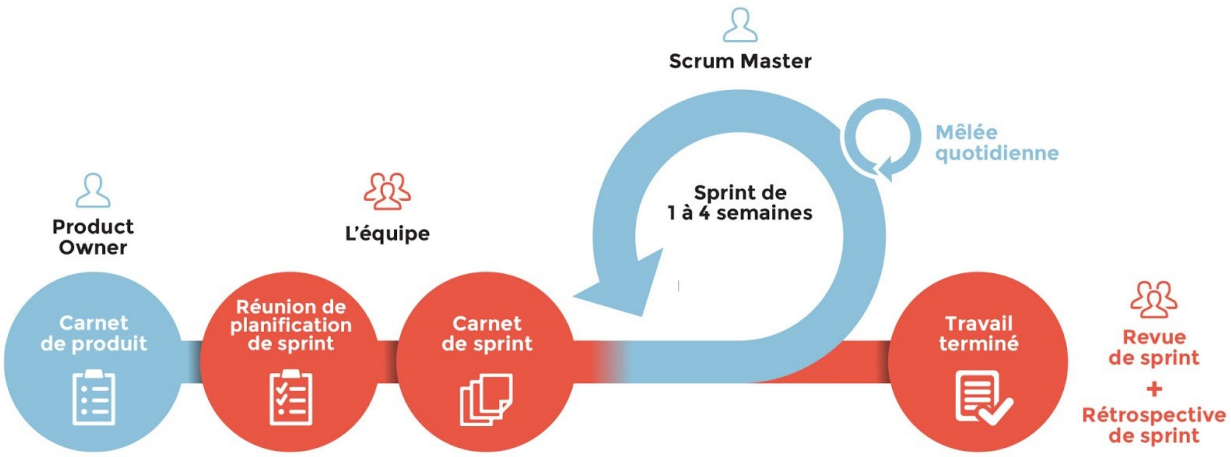


FIGURE 34 – Méthodologie utilisée

Premier sprint zéro

Dans un premier temps, nous avons dû faire la conception de l'application, nous avons donc réalisé des diagrammes UML nous permettant d'analyser des différents aspects : un diagramme de classes, décrivant la structure de classes que nous avons mis en place pour coder l'application ; un diagramme d'objets, illustrant le diagramme de classes ; un diagramme de séquence, décrivant la méthode de calcul du résultat du scrutin à la fin de la votation, l'une des méthodes les plus importantes à implémenter pour notre application ; et un diagramme de cas d'utilisation, donnant une vision globale du comportement fonctionnel de l'application et présentant les différents acteurs qui interagissent avec le système de vote électronique.

Nous avons ensuite implémenté la structure de classes du système en définissant les attributs principaux et les signatures des méthodes auxquelles nous avions pensé au départ. De plus, nous avons mis en place le gestionnaire de sources SourceSup , dont nous avons parlé dans la section « Démarche Personnelle », pour de gérer les mises à jour du code.

Finalement, nous avons élaboré un système d'organisation de tâches au travers l'application Trello, dont nous parlerons par la suite, qui nous a permis de répartir et organiser les tâches à effectuer pour la réalisation du projet. Vu qu'il s'agitait d'un sprint où nous avons fait la conception de l'application et la mise en place des outils nécessaires à son développement, donc nous avons jeté les bases pour le développement de l'application, nous appelons ce sprint : le premier sprint 0. Nous avons complété tout ce qui était prévu pour ce sprint.

Sprint	Stories prévues	Effort	Valeur pour le client	Réalisé
0	Créer des diagrammes pour la conception du projet	5	0	X
	Implémenter la structure de classes du système de votation	2	0	X
	Mettre en place un gestionnaire de sources pour le code du programme (SVN)	1	0	X
	Disposer d'un système d'organisation de tâches (Trello)	1	0	X

FIGURE 35 – Premier sprint zéro

Second sprint zéro

Dans ce sprint nous continuons toujours à réaliser des technical stories qui ont été essentielles pour le développement de l'application et pour pouvoir livrer des incréments fonctionnels par la suite. Nous avons donc procédé à nous renseigner sur les différentes méthodes et techniques de chiffrement lors que notre tuteur nous expliquait à quoi elles consistaient et les méthodes nécessaires pour leur implémentation, nous avons finalement opté par le chiffrement asymétrique, décrit dans la partie « Rapport Technique », pour chiffrer les votes. Il faut remarquer que l'apprentissage de la cryptographie a dépassé ce sprint, comprenant en fait plusieurs sprints, mais nous avons décidé de ne pas contempler cet apprentissage dans les sprints suivants à cause de sa continuité, donc nous ne considérons ici que l'initiation à la cryptographie.

Nous avons aussi mis en place une classe qui a été nécessaire à la manipulation des grands entiers et des entiers issus de l'arithmétique modulaire, la manipulation de ces entiers à été nécessaire pour développer la méthode de chiffrement, comme expliqué dans la partie « Rapport Technique » du rapport. Il est à noter qu'au début nous n'avions pas considéré ce technical story, ni le besoin des méthodes opérant le type d'entiers que nous venons de décrire, dans le Product Backlog car nous nous initions à la cryptographie et nous ne connaissions pas l'importance de la réalisation de ces technical stories. Enfin, pendant le deuxième sprint 0 nous avons mis en place l'interface client-serveur nécessaire pour que les votants puissent, par la suite, voter depuis leur propre application. Nous avons réussi à compléter toutes les technical stories prévues pour ce sprint.

Sprint	Stories prévues	Effort	Valeur pour le client	Réalisé
0	Se renseigner sur les différentes méthodes de chiffrement (symétrique, asymétrique)	5	0	X
	Disposer d'une classe pour manipuler des grands entiers et des entiers modulaires	2	0	X
	Mettre en place une interface client-serveur	5	0	X

FIGURE 36 – Second sprint zéro

Sprint 1

Il s'agit du premier sprint où nous avons répondu à un besoin fonctionnel, et donc livré un incrément fonctionnel. Le user story que nous avons accompli dans ce sprint est la mise en place d'une base de données liée au serveur du système de votation.

En outre, nous avons continué à développer des technical stories. Nous avons donc établi un diagramme de Gantt, ce technical story n'était pas contemplé au début dans le Product Backlog mais il a été nécessaire de l'ajouter pour avoir un outil nous permettant d'organiser le calendrier du projet. De plus, nous avons dû nous initier à la manipulation d'une bibliothèque graphique propre au langage de programmation Java, en l'occurrence Swing, pour avoir les connaissances nécessaires à la création d'une interface graphique pour l'application.

Cependant, nous n'avons pas pu accomplir tout ce que nous avions prévu. Nous pensions être prêts à implémenter les différentes méthodes opérant des entiers issus à partir de l'arithmétique modulaire, voire les méthodes nécessaires au chiffrement des votes, expliquées dans la partie « Rapport Technique » du rapport . Or, nous n'avons pas rapidement assimilé les fondements mathématiques de la cryptographie et nous nous sommes rencontrés avec des problèmes de conception de certaines classes, par conséquent, n'avons pas fini de implémenter les méthodes prévues.

Sprint	Stories prévues	Effort	Valeur pour le client	Réalisé
1	Disposer d'un diagramme de Gantt pour le projet	2	0	X
	S'initier à la manipulation d'une bibliothèque graphique (Swing)	5	0	X
	Disposer d'une base de données ayant un lien avec le serveur du système	3	5	X
	Implémenter des méthodes opérant des grands entiers et des entiers modulaires	6	0	
	Implémenter des méthodes nécessaires au chiffrement	7	0	

FIGURE 37 – Premier sprint

Sprint 2

Le fait de ne pas avoir fini de réaliser des stories prévus pour le sprint précédent a eu de répercussions sur ce sprint. Nous avons dédié cette période à discuter avec notre tuteur, à lui poser des questions, à bien comprendre et à implémenter les méthodes de l'arithmétique modulaire pour opérer des grands entiers ainsi que les méthodes nécessaires au chiffrement asymétrique. Nous avons aussi mis en place les artifacts nécessaires pour stocker les données des votants dans la base de données de l'application, ce qui a eu une valeur client pour l'administrateur du scrutin. Cette fois-ci nous avons réussi à finir ce qui était prévu.

Sprint	Stories prévues	Effort	Valeur pour le client	Réalisé
2	Implémenter des méthodes opérant des grands entiers et des entiers modulaires	6	0	X
	Implémenter des méthodes nécessaires au chiffrement	7	0	X
	Disposer des données des votants	2	5	X

FIGURE 38 – Second sprint

Sprint 3

Dans ce sprint nous avons mis en place une interface de connexion pour pouvoir authentifier l'utilisateur souhaitant se connecter à l'application. Ensuite, nous avons implémenté la méthode permettant au votant d'émettre son vote en utilisant le chiffrement de ce dernier.

De plus, une fonctionnalité a été développée pour pouvoir consulter les votes chiffrés enregistrés jusqu'au moment, ce user story n'était pas initialement prévu mais il a été ajouté au Product Backlog car il représentait une valeur client importante vu que la fonctionnalité permet aux utilisateurs de l'application de vérifier que les votes reçus par le serveur ont été bien chiffrés. Nous avons aussi développé l'interface client-serveur que nous avions mis en place dans le deuxième sprint 0 pour que le votant puisse voter depuis une application cliente. En outre, nous avons créé une fonctionnalité permettant de récupérer le résultat déchiffré du scrutin à la fin de la votation, ce qui était essentiel pour le client vu que le but d'une votation est de connaitre un résultat final.

Par ailleurs, nous nous sommes proposés de rédiger les premières parties du rapport, un technical story important pour avoir un compte rendu du projet. Néanmoins, nous n'avons pas fini de rédiger toutes les parties que nous pensions réaliser pour ce sprint à cause d'une manque du temps, vu que la fin du semestre s'approchait et nous avions un emploi du temps scolaire chargé et des travaux à rendre dans des autres cours, ce qui nous a empêché de nous consacrer pleinement au projet.

Sprint	Stories prévues	Effort	Valeur pour le client	Réalisé
3	Avoir un formulaire de connexion	2	5	X
	Voter sans que le vote soit connu : avoir le vote chiffré	5	10	X
	Consulter les votes chiffrés qui ont été enregistrés jusqu'au moment	3	8	X
	Voter depuis une application cliente	5	8	X
	Avoir le résultat déchiffré à la fin de la votation	4	10	X
	Rédiger les parties introduction, cahier de charges et rapport technique du rapport du projet	8	0	

FIGURE 39 – Troisième sprint

Sprint 4

Le dernier sprint a compris la finalisation et donc la mise en place d'une interface graphique complète pour l'application, ayant ses différentes fonctionnalités organisées par des sections dans un menu s'affichant une fois l'utilisateur connecté. Nous avons aussi créé une fonctionnalité pour que l'administrateur du scrutin puisse mettre fin à la votation en calculant le résultat de celle-ci. En outre, nous avons créé des fonctionnalités pour qu'il puisse savoir si un votant a voté ou pas, ajouter ou supprimer des votants du scrutin et réinitialiser le référendum de ce dernier. Parallèlement à l'accomplissement de ces user stories, nous avons fini de rédiger les parties du rapport qui n'avaient pas été finies dans le sprint précédent et les autres parties manquantes pour la finalisation du rapport.

Sprint	Stories prévues	Effort	Valeur pour le client	Réalisé
4	Avoir une interface graphique complète pour l'application	6	7	X
	Avoir les sections de l'interface graphique organisées dans un menu	1	4	X
	Calculer le résultat de la votation	1	7	X
	Rédiger les parties introduction, cahier de charges et rapport technique du rapport du projet	8	0	X
	Rédiger les parties résultats, gestion de projet et conclusion du rapport du projet	6	0	X
	Avoir un flag indiquant si le votant a voté ou pas	1	6	X
	Avoir la possibilité de mettre à jour la liste de votants du scrutin	4	3	X
	Réinitialiser le referendum du scrutin	2	3	X

FIGURE 40 – Quatrième sprint

Outil d'aide à la répartition des tâches

Au cours du projet, nous nous sommes servis d'un outil informatique nommé Trello pour organiser les tâches à réaliser dans les différents sprints. Il s'agit d'un outil de gestion de projet en ligne basé sur une organisation en planches listant des cartes, chacune représentant des tâches. Les cartes ont été assignées aux différents membres de l'équipe SCRUM et ont été mobiles d'une planche à l'autre, traduisant leur avancement : la panche TO DO contenait les tâches encore à réaliser, la planche DOING comprenait les tâches qui étaient en train d'être réalisés par les membres de l'équipe et la planche DONE contenait les tâches qui avaient déjà été réalisés. De même, nous avons eu la possibilité de fixer des dates limites pour les différentes tâches. Cette application nous a permis de suivre facilement l'évolution des sprints, de organiser l'avancement des tâches de notre projet et faciliter la répartition de ces dernières.

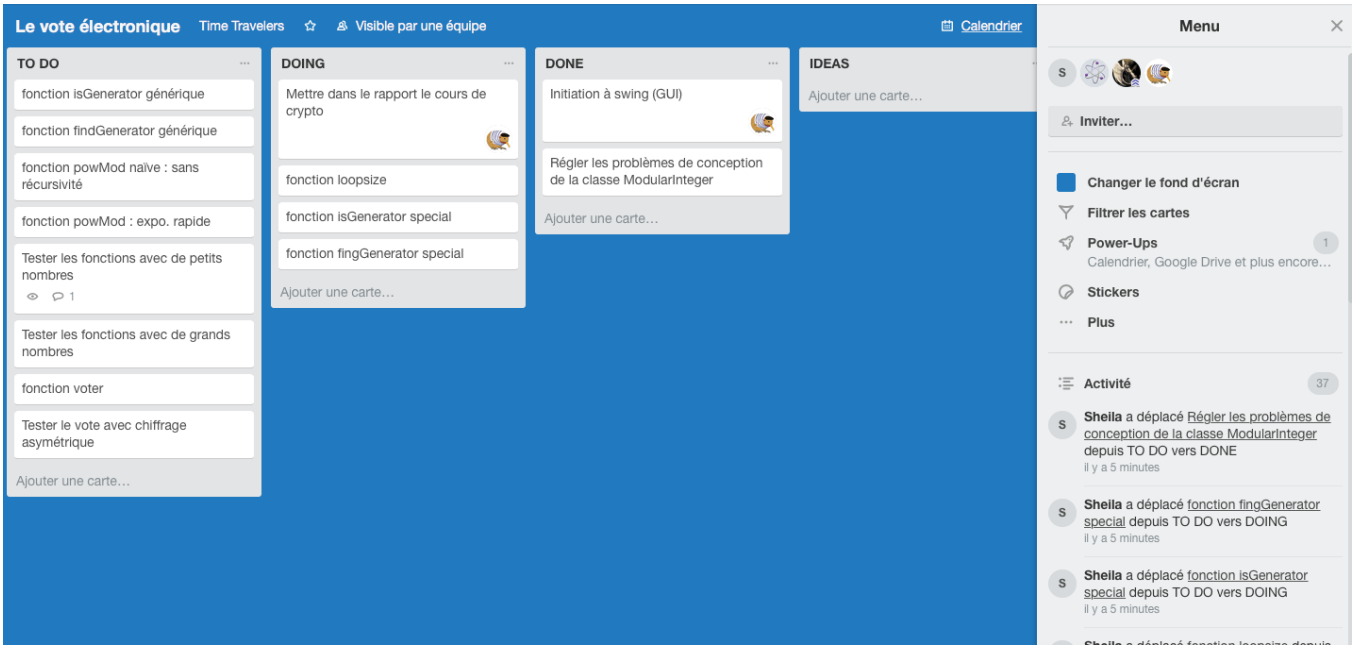


FIGURE 41 – Aperçu du Trello durant le second sprint

4.3 Bilan critique par rapport au cahier des charges

Nous avons développé notre projet tout en mettant en place les principes et les valeurs agiles de la méthodologie SCRUM. En effet, ces principes sont aussi utiles pour le client, qui pourra profiter d'un produit qui lui convient en fin de projet, que pour l'équipe de développement, qui peut affiner son travail sur les tâches les plus importantes et se remettre en question à la fin de chaque sprint afin d'optimiser au maximum son travail. Cette expérience a donc pu se montrer enrichissante car nous avons pu communiquer constamment au sein de l'équipe et avec le commanditaire en lui montrant l'avancement du projet à chaque itération. En ce qui concerne le changement et l'évolutivité, nous avons su accepter des stories, donc des besoins fonctionnels et non fonctionnels, surgissant en cours de sprint en l'adaptant.

La création de cette application nous a parfois confronté aux difficultés de la gestion de projet. On pourrait dire que la principale difficulté que nous avons eue est l'initialisation à la cryptographie, nous avons dû employer des fondements mathématiques, au début, étrangers à notre connaissance, nous avons donc pris du temps pour nous initier et pour pouvoir réaliser un sprint donnant une valeur cliente, mais, faisant partie des besoins non-fonctionnels, cela a fait partie de notre apprentissage, ce qui a été nécessaire pour le développement de l'application. Or, nous n'avons pas réussi à bien estimer le temps pour cet apprentissage ce qui a

eu un impact sur l'avancement de la troisième itération (sprint 1). De même, le fait d'avoir eu un emploi de temps académique assez chargé à la fin du semestre nous a difficulté l'accomplissement des tâches prévues pour l'avant dernier sprint, nous avons donc dû les réaliser dans la dernière itération.

Cependant, une fois que les bases de la cryptographie ont été bien assimilées et que nous avons pu développer les méthodes nécessaires au chiffrement, la vélocité de l'équipe a augmenté considérablement ce qui nous a permis d'effectuer plus de tâches dans les sprints suivants. Donc, nous avons finalement réussi à développer une application de vote électronique qui répondait aux besoins du client.

Graphiques de rapports d'activité

Burn Up chart :

Ce graphique, suit la progression vers la réalisation d'un projet, il y a deux lignes sur le graphique : le Burn Up Chart montre clairement à la fois le travail accompli et la portée du projet, donc les points d'effort nécessaires à la complétion du Product Backlog. L'axe vertical est la quantité de travail, mesurée en points d'effort, l'axe horizontal est le temps mesuré en itérations, donc en sprints.

À chaque sprint, nous illustrons la quantité de travail accomplie (courbe orange) et la quantité totale de travail (courbe bleue). Le projet est terminé lorsque les lignes se rencontrent.

Une ligne «idéale» est incluse dans le graphique (courbe grise). Cela montre l'achèvement nécessaire à chaque sprint pour respecter la date limite. Nous pouvons donc observer que le projet a été en retard par rapport à la ligne idéale. Or, même avec un Product Backlog évoluant, donc avec plus de points d'effort venant de nouvelles fonctionnalités à développer, une fois les fondements de la cryptographie assimilés et les méthodes de chiffrement implémentées (sprint 3), nous avons pu rencontrer la ligne représentant les points d'effort du Product Backlog, ainsi nous avons réussi à effectuer l'effort nécessaire au projet et donc à accomplir ce dernier.

Graphique de vélocité :

Nous avons pu constater la même conclusion obtenue à partir du Burn Up Chart grâce au graphique de vélocité, représentant les points d'effort estimés (barre orange) et effectivement complétés (barre grise) de chaque itération, ce qui illustre la vélocité de l'équipe au cours des sprints. Nous pouvons ainsi observer graphiquement le retard eu dans le sprint 1 et dans le sprint 3. Ces retards sont expliqués en détail dans la partie « Sprint Backlog ». Il est à noter que, même si nous n'avons pas réalisé la totalité d'un story dans un sprint spécifique, nous avons considéré l'avancement sur ce story dans le graphique de vélocité. Pour illustrer,

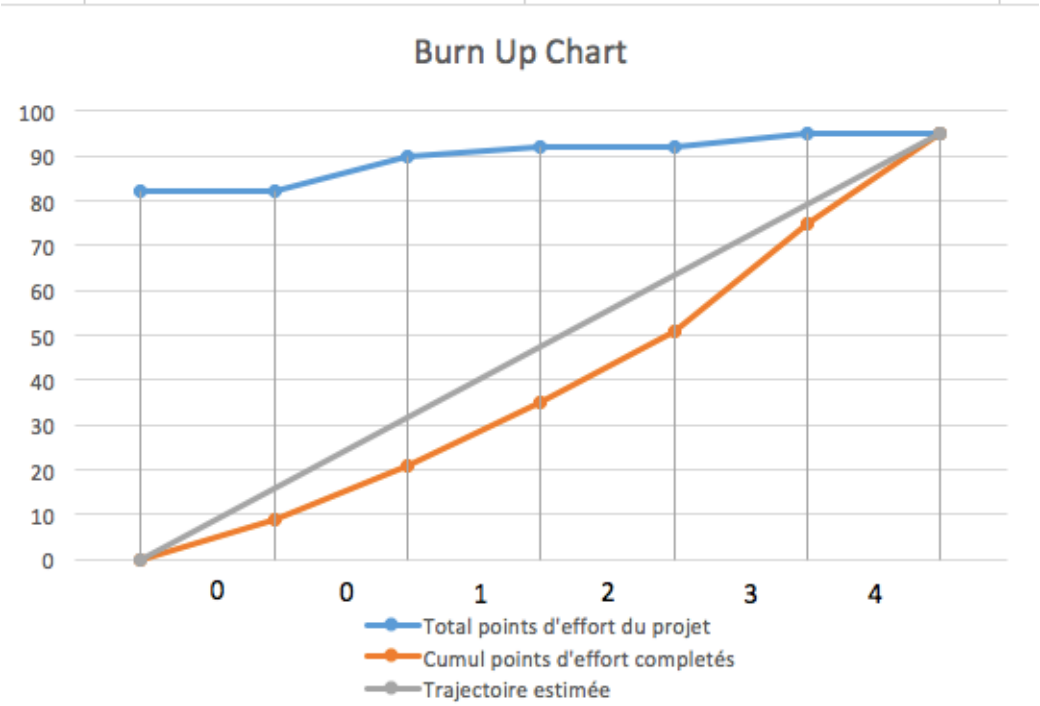


FIGURE 42 – Burn Up chart

nous avons considéré avoir avancé avec trois points d'effort des six points d'effort correspondants au technical story «implémentation des méthodes opérant des grands entiers et des entiers issus de l'arithmétique modulaire » (cf. sprint 1) donc, en fait nous avons accompli la moitié de ce story dans le sprint 1, et l'autre moitié dans le sprint 2. Une logique similaire a été appliquée pour les autres stories qui n'ont pas été réalisées dans le sprint prévu.

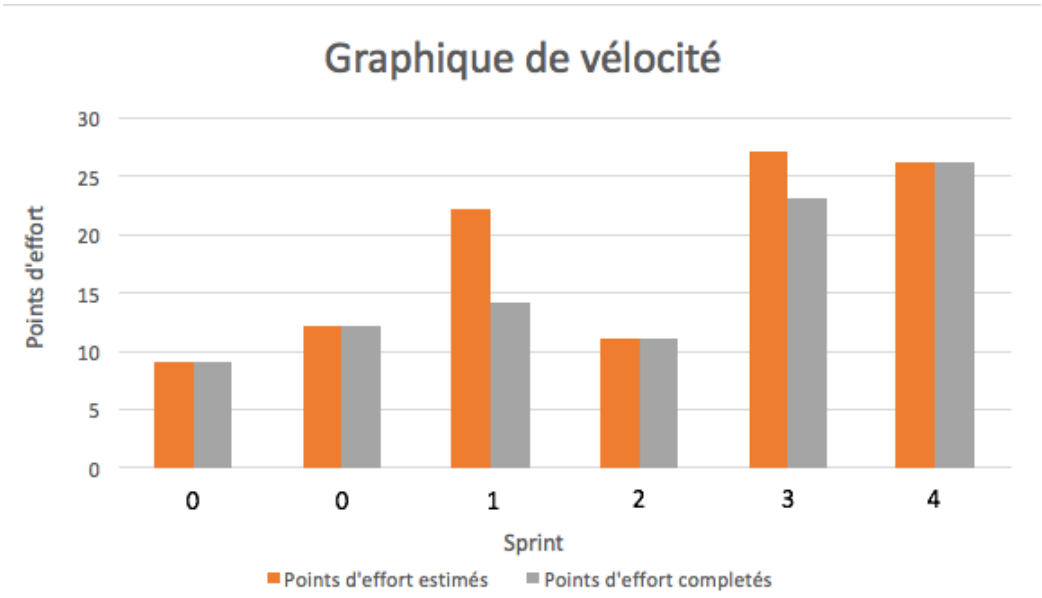


FIGURE 43 – Graphique de vélocité

5 Conclusions

5.1 Conclusion du groupe

Le jour où on nous a demandé de réaliser une application de vote électronique, aucun des membres de l'équipe n'aurait imaginé investir autant de temps, autant de soirées à travailler sur ce projet. Nous nous sommes tous investis au maximum dans ce projet allant même jusqu'à faire des réunions très tard le soir afin de consolider les acquis de chacun.

Ce projet paraissait relativement facile dans sa forme ; cependant, les mécanismes de chiffrement ont été difficile à assimiler. En effet, nous avons découvert ensemble des notions complexes qui dépassaient le cadre général de notre formation. La relation client-serveur, la connexion à une base de données, la complexité algorithmique, la création d'une interface graphique et par-dessus tout la conception d'algorithme de chiffrement sont autant de notions qui nous ont permis de gagner en expérience au cours de ce semestre.

Finalement nous nous sommes surpris à plus apprécier ce projet que prévu, mais nous sommes tout de même déçu de ne pas avoir pu en faire plus.

5.2 Conclusions personnelles

Je pense que ce qui m'a le plus motivé dans ce projet c'est que pour la première fois dans le cadre de ma formation, le projet qui m'a été proposé doit être codé en Java. En effet ce que je préfère c'est coder des applications. De plus depuis que nous sommes à l'Institut Universitaire et Technologique (IUT) nous avions réalisé uniquement des projets web.

Pour moi ce projet aura notamment été une occasion de découvrir les mécanismes de sécurité des applications client-serveur dont le chiffrement ElGamal. J'ai bien aimé toutes les notions de complexité algorithmique car j'ai pu comparer la théorie avec la pratique en montrant qu'il ne suffit pas qu'un algorithme marche pour être fonctionnelle, il doit être adapté à tout type de situation.

Pour finir je dirais que je suis très satisfait de ce que nous avons accompli et j'ai hâte de retravailler avec cette équipe.

Dylan Durand

Ce projet a été notamment une occasion de m'intéresser davantage au chiffrement et de pouvoir à coder en Java. Les notions de chiffrement ont été assez difficiles à assimiler mais elles permettent de comprendre son fonctionnement et je pense que ce sont des connaissances nécessaires pour un informaticien. Tout au cours de notre projet, nous avons pu utiliser des notions apprises en cours comme les sockets et la complexité algorithmique mais nous avons eu également la chance de pouvoir apprendre des nouvelles notions en dehors du cursus scolaire, chercher par nous même pour coder et être curieux. Pouvoir coder une interface graphique en utilisant les algorithmes de chiffrage a également été un réel divertissement et intéressante à mettre en place en comprenant son fonctionnement. Mon seul regret, c'est de ne pas avoir investi autant de temps que ce que je souhaitais.

Je souhaite conclure sur le fait que ce projet a été très enrichissant pour moi et c'est avec plaisir que je travaillerais de nouveau avec l'équipe "Time Travellers".

Gaël Hollows-Pellier

Ce projet m'a donné une base en cryptographie en m'apprenant à coder une méthode de chiffrement considérée puissante et utilisée dans plusieurs applications de nos jours, tout en me faisant réfléchir à l'importance de la complexité des algorithmes pour garantir des solutions efficaces qui s'effectuent dans un temps raisonnable. Je considère que les mécanismes de sécurité tels que celui du chiffrement à clé publique sont importants à mettre en place pour garantir la sécurité des applications, ce qui est un critère important de l'informatique. Ce projet m'a d'ailleurs donné l'envie d'en apprendre plus sur les différents systèmes de chiffrement, et la manière dont ils fonctionnent. En outre, j'ai pu en profiter pour en apprendre plus sur la communication entre sockets (ce que je n'avais pas eu l'occasion d'appliquer en Java jusqu'alors), la gestion de projet, l'importance de l'organisation de tâches, la mise en place d'une méthodologie agile, et pour découvrir de nouveaux utiles de collaboration, tels que SVN. Je me souviendrai donc de ce projet comme d'une expérience riche en apprentissages et en découvertes.

Sheila Barron

Nous voici donc à la fin de ce travail de groupe. À l'heure où j'écris ces lignes, il est dix heures du matin, et j'ai passé la nuit à finir d'écrire et mettre en forme le rapport. C'est qu'il en aura fallu, du temps et de l'énergie, pour venir à bout de ce projet… Le projet en lui-même ne nous aura pas demandé une somme de travail astronomique. Nous l'avons travaillé à intervalles aussi réguliers que possibles, et il nous a donné un certain nombre de choses à faire ; mais rien de bien insurmontable dans l'ensemble. Alors où est parti le temps qui nous a manqué pour aller au bout de tout ce que nous voulions en faire ? Pourquoi arriver jusqu'ici nous a demandé autant d'efforts ?

La véritable réponse se situe ailleurs. Un certain nombre de facteurs extérieurs au projet sont venus nous déranger dans notre travail, notamment les différents travaux de groupe demandés au sein de l'IUT. Ces travaux, bien que moins intéressants et censés être de moindre importance vis-à-vis de ce projet (qui doit sanctionner l'ensemble du semestre), nous ont tout simplement empêché d'avancer : de nouvelles échéances arrivaient pratiquement tous les jours ou tous les deux jours, nous obligeant à revoir sans cesse nos priorités, et à repousser les échéances les plus lointaines. Cela explique le regret partagé par chacun des membres du groupe : celui de n'avoir pas eu la possibilité de s'investir à la hauteur de ses attentes ; celui de n'avoir pas pu donner le meilleur de soi-même ; celui de n'avoir pas eu l'opportunité de prouver ce que l'on vaut, et de montrer ce que l'on sait faire.

J'ai la certitude que l'IUT, en m'obligeant à m'investir dans des matières et des travaux qui ne valaient vraiment pas autant de peine, m'a empêché de me donner à fond et de m'investir dans un travail qui, en plus de me tenir à cœur, aurait pu m'apprendre bien plus que ce que j'ai appris en un an et demie de cours. Cette certitude est, je pense, suffisamment importante et grave pour mériter sa place dans cette conclusion. D'un point de vue plus humain, l'accumulation du travail m'a appris à tenir compte de mes limites, et à me reposer sur les autres, ou parfois même à laisser les autres se reposer sur moi. C'est pourquoi je tiens à finir en remerciant du fond du cœur chacun des membres de ce groupe, qui ont tous su m'aider quand il le fallait, et à qui j'espère avoir pu apporter quelque chose en retour.

Jean-Clair Granier

6 Bibliographie

7 Annexes

7.1 Annexes techniques

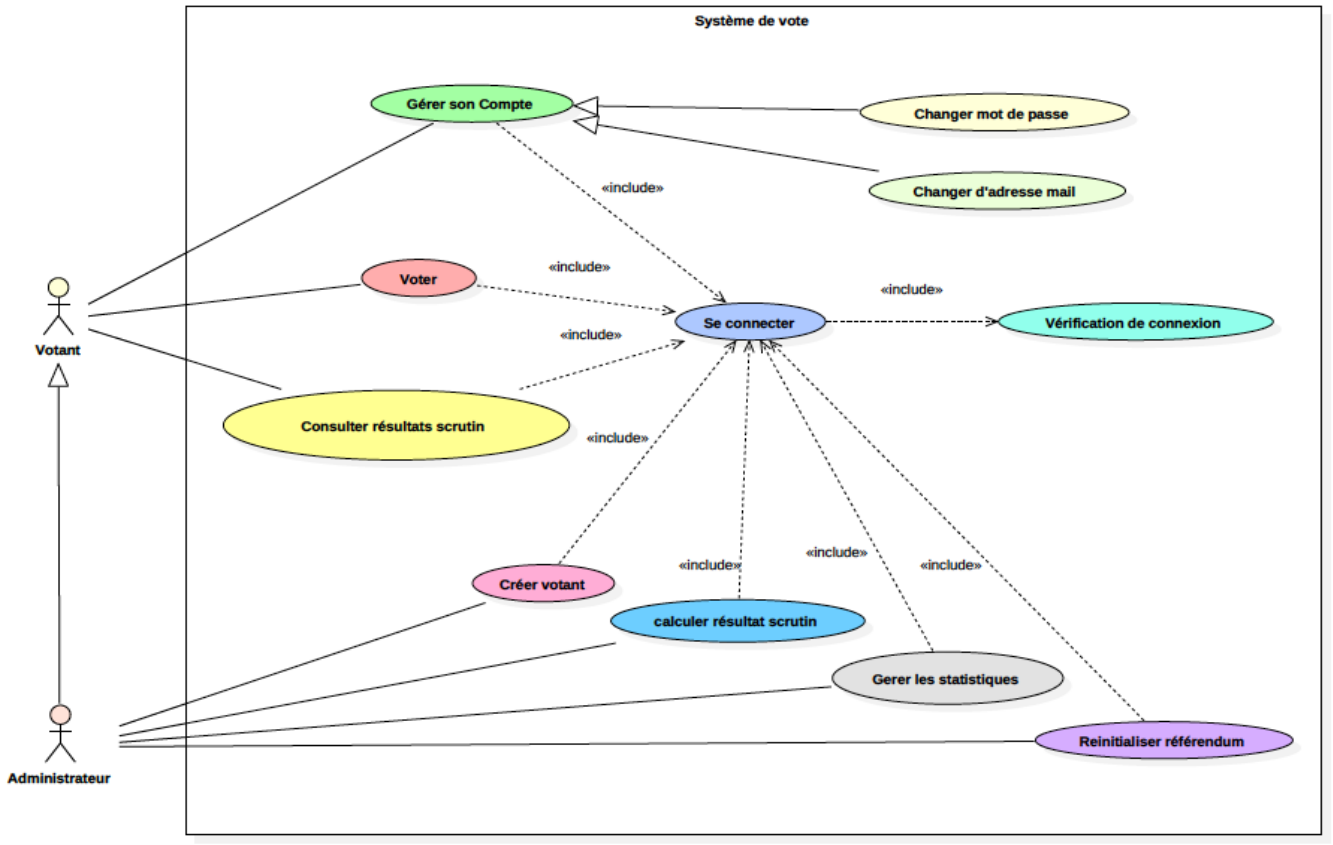


FIGURE 44 – Diagramme de cas d'utilisation initial

User story	Tâche	Estimation	Priorité
L'administrateur veut pouvoir stocker toutes les données du système de votation	Mettre en place une base de données	2	3
Un votant veut pouvoir voter depuis sa propre application	Mettre en place une connexion client-serveur	6	3
Un votant veut pouvoir voter	Créer une méthode de votation	3	1
Un votant ne veut pas que son vote soit connu	Créer les méthodes de chiffrement du vote (chiffrement asymétrique à clé publique)	9	1
Un votant veut posséder une interface pour se connecter	Créer une interface de connexion utilisateur (identifiant + mot de passe)	3	4
Un utilisateur veut pouvoir voir les résultat de la votation	Créer une méthode pour récupérer le résultat de la votation déchiffré	5	1
Un votant veut pouvoir interragir visuellement avec l'application	Mettre en place une interface graphique permettant au votant d'acceder aux différentes fonctionnalités de l'application	6	5
Un votant veut pouvoir consulter les votes chiffrés qui ont été enregistrés jusqu'au moment	Créer une interface et des méthodes pour afficher les votes chiffrés enregistrés jusqu'au moment	3	4

FIGURE 45 – Backlog product initial : besoins fonctionnels

```
ModularInteger a = new ModularInteger("5", "109547" );
BigInteger x = new BigInteger("500000");
System.out.println("a= 5");
System.out.println("p= 109 547");
System.out.println("x= 500 000");
System.out.println("On cherche a calculer a puissance x modulo p.");
System.out.println("");

System.out.println("fonction naivePow: ");
long debut = System.currentTimeMillis();
ModularInteger result = a.naivePow(x);
long duree = System.currentTimeMillis()-debut;
System.out.println("result= " + result);
System.out.println("temps d'execution: " + duree +"ms");
System.out.println("");

System.out.println("fonction optiPow: ");
debut = System.currentTimeMillis();
result = a.pow(x);
duree = System.currentTimeMillis()-debut;
System.out.println("result= " + result);
System.out.println("temps d'execution: " + duree +"ms");
```

FIGURE 46 – Tests sur les ModularInteger

```
//Traitements...
BigInteger p = new BigInteger("109547");
System.out.println("particularPrime= " + p + "  (codé sur 17 bits)");
BigInteger a = new BigInteger("5");
System.out.println("generator= " + a);
System.out.println();

System.out.println("fonction isGeneratorGen:");
long debut = System.currentTimeMillis();
boolean bool = Encryption.isGeneratorGen(p, a);
long duree = System.currentTimeMillis()-debut;
System.out.println("result= " + bool);
System.out.println("temps d'execution: " + duree +"ms");
System.out.println("");

System.out.println("fonction isGenerator:");
ModularInteger gen = new ModularInteger(a,p);
BigInteger q = (p.subtract(BigInteger.ONE)).divide(new BigInteger("2"));
debut = System.currentTimeMillis();
bool = Encryption.isGenerator(gen, q);
duree = System.currentTimeMillis()-debut;
System.out.println("result= " + bool);
System.out.println("temps d'execution: " + duree +"ms");
```

FIGURE 47 – Tests sur isGenerator()

```java
/**
 *
 * @param e the exponent for pow
 * @return (this.val^e) mod this.modulo
 */

public ModularInteger naivePow(BigInteger e)
{ // multiplication avec mod

    BigInteger result = BigInteger.ONE;
    BigInteger counter = BigInteger.ZERO;

    while (counter.compareTo(e) == -1) // counter < e
    {
        result = result.multiply(this.val);
        counter  = counter.add(BigInteger.ONE);
    }

    return new ModularInteger(result, this.modulo);

}
```

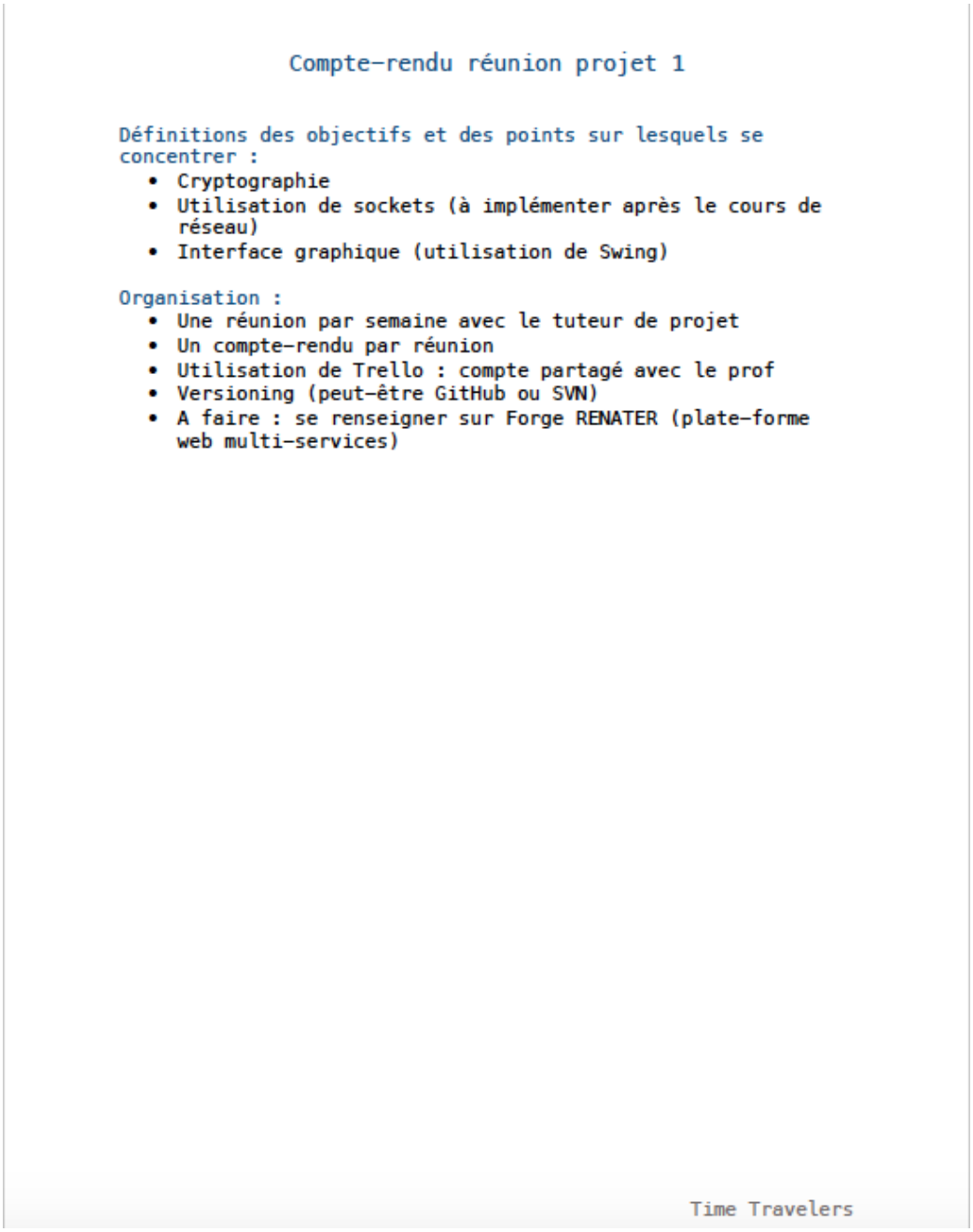
FIGURE 48 – Fonction naivePow()

```java
    System.out.println("log_2(2^(1024))");
    BigInteger z = new BigInteger("179769313486231590772930519078902473361797697894230657273430081"
            + "157732675805500963132708477322407536021120113879871393357658789768814416622492847430"
            + "063947412437776789342486548527630221960124609411945308295208500576883815068234246280"
            + "814739131105408272371633505106845862982399472459384797163048353563296242241372160");
    long debut = System.currentTimeMillis();
    BigInteger result = Encryption.log(new BigInteger("2"), z);
    long duree = System.currentTimeMillis()-debut;
    System.out.println("result: "+ result);
    System.out.println("temps d'execution: " + duree +"ms");
    System.out.println("");

    BigInteger z2 = Encryption.bigIntegerPow(new BigInteger("2"), new BigInteger("500000"));
    System.out.println("log_2(2^(500 000))");
    debut = System.currentTimeMillis();
    result = Encryption.log(new BigInteger("2"), z2);
    duree = System.currentTimeMillis()-debut;
    System.out.println("result: "+ result);
    System.out.println("temps d'execution: " + duree +"ms");
}
```

FIGURE 49 – Tests sur le log

7.2 Compte-rendu des réunions



Compte-rendu réunion projet 1

Définitions des objectifs et des points sur lesquels se concentrer :

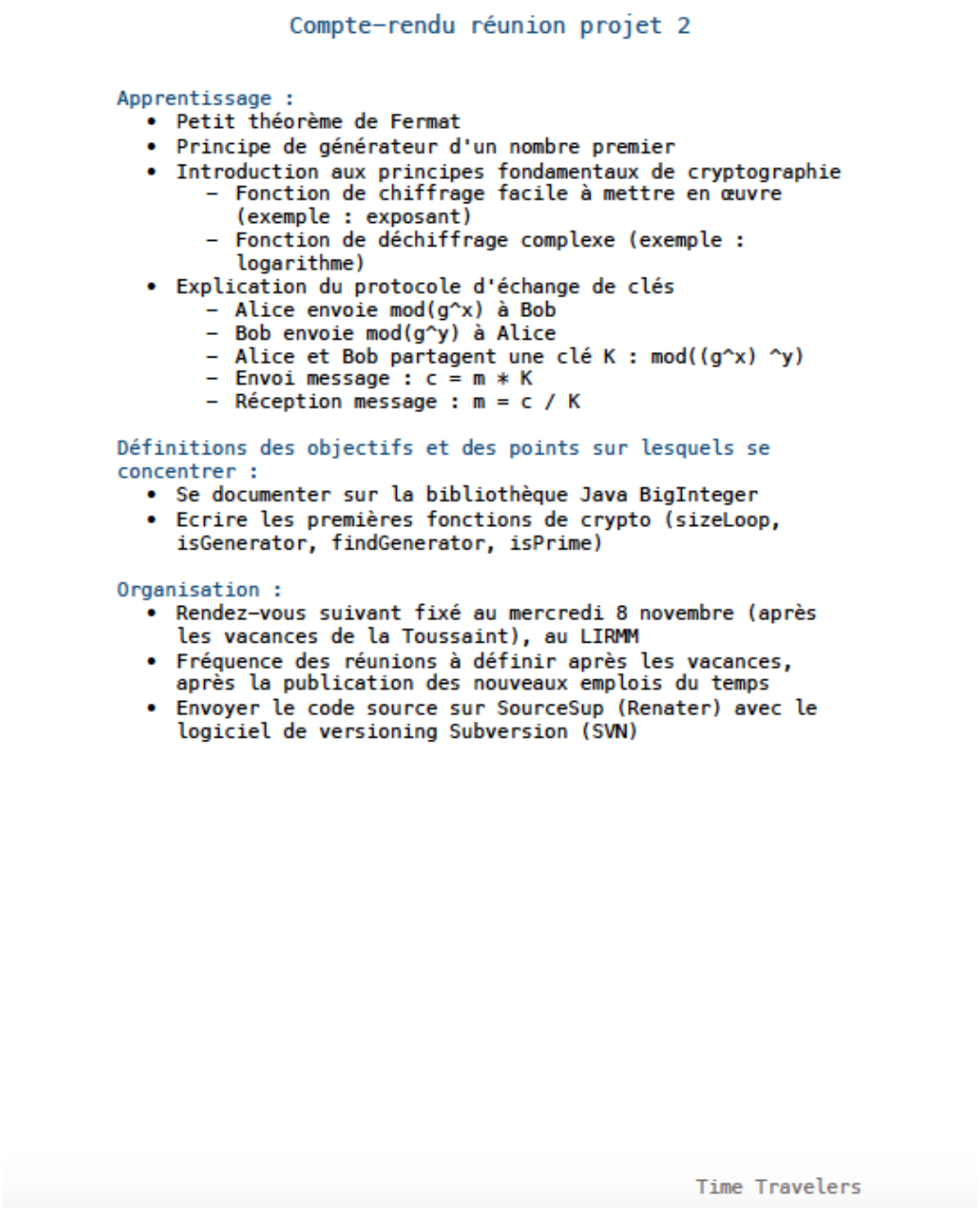
- Cryptographie
- Utilisation de sockets (à implémenter après le cours de réseau)
- Interface graphique (utilisation de Swing)

Organisation :

- Une réunion par semaine avec le tuteur de projet
- Un compte-rendu par réunion
- Utilisation de Trello : compte partagé avec le prof
- Versioning (peut-être GitHub ou SVN)
- A faire : se renseigner sur Forge RENATER (plate-forme web multi-services)

Time Travelers

FIGURE 50 – Compte-rendu de la réunion 1



Compte-rendu réunion projet 2

Apprentissage :

- Petit théorème de Fermat
- Principe de générateur d'un nombre premier
- Introduction aux principes fondamentaux de cryptographie
  - Fonction de chiffrage facile à mettre en œuvre (exemple : exposant)
  - Fonction de déchiffrage complexe (exemple : logarithme)
- Explication du protocole d'échange de clés
  - Alice envoie mod(g^x) à Bob
  - Bob envoie mod(g^y) à Alice
  - Alice et Bob partagent une clé K : mod((g^x) ^y)
  - Envoi message : c = m * K
  - Réception message : m = c / K

Définitions des objectifs et des points sur lesquels se concentrer :

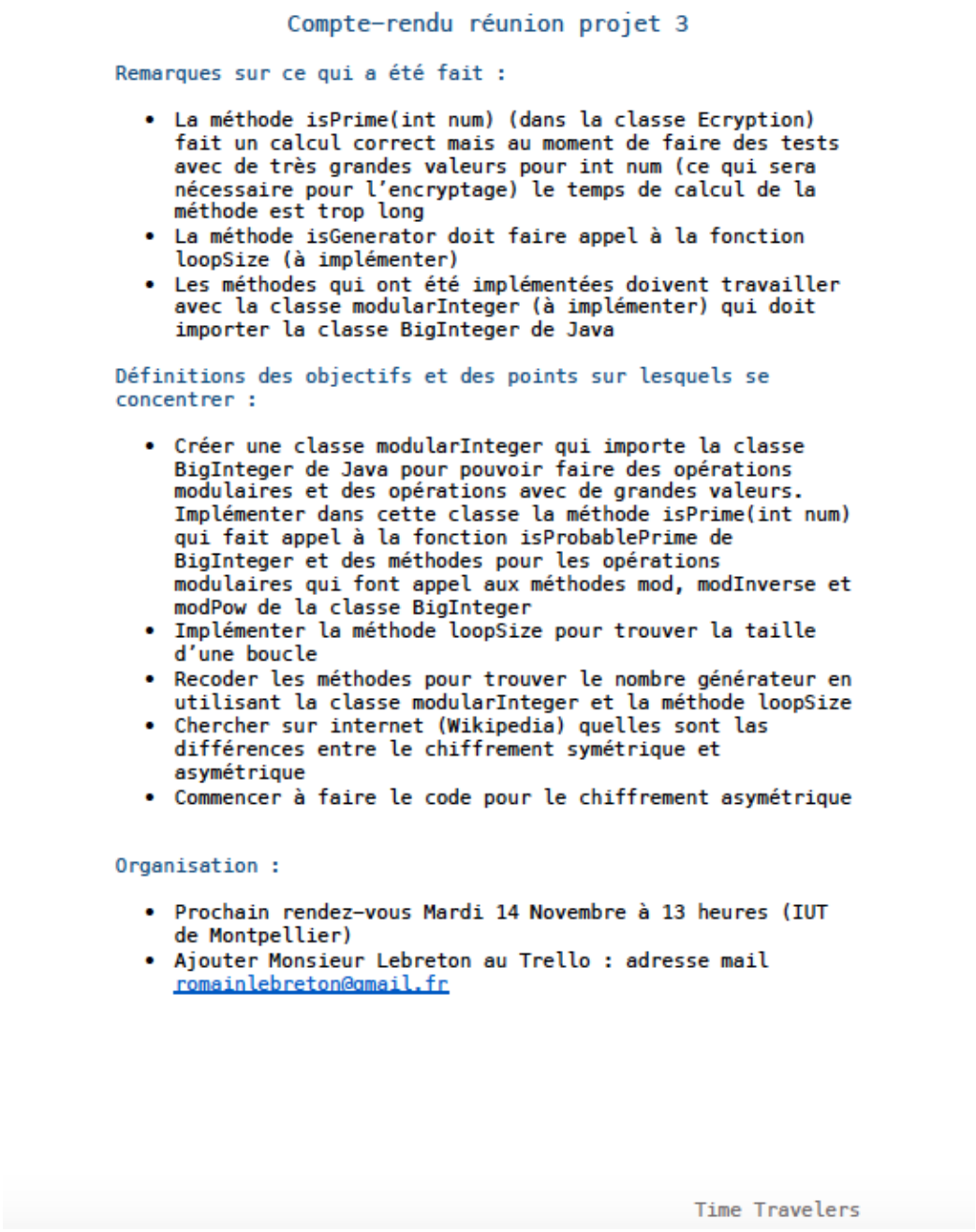
- Se documenter sur la bibliothèque Java BigInteger
- Ecrire les premières fonctions de crypto (sizeLoop, isGenerator, findGenerator, isPrime)

Organisation :

- Rendez-vous suivant fixé au mercredi 8 novembre (après les vacances de la Toussaint), au LIRMM
- Fréquence des réunions à définir après les vacances, après la publication des nouveaux emplois du temps
- Envoyer le code source sur SourceSup (Renater) avec le logiciel de versioning Subversion (SVN)

Time Travelers

FIGURE 51 – Compte-rendu de la réunion 2



Compte-rendu réunion projet 3

Remarques sur ce qui a été fait :

- La méthode isPrime(int num) (dans la classe Ecryption) fait un calcul correct mais au moment de faire des tests avec de très grandes valeurs pour int num (ce qui sera nécessaire pour l'encryptage) le temps de calcul de la méthode est trop long
- La méthode isGenerator doit faire appel à la fonction loopSize (à implémenter)
- Les méthodes qui ont été implémentées doivent travailler avec la classe modularInteger (à implémenter) qui doit importer la classe BigInteger de Java

Définitions des objectifs et des points sur lesquels se concentrer :

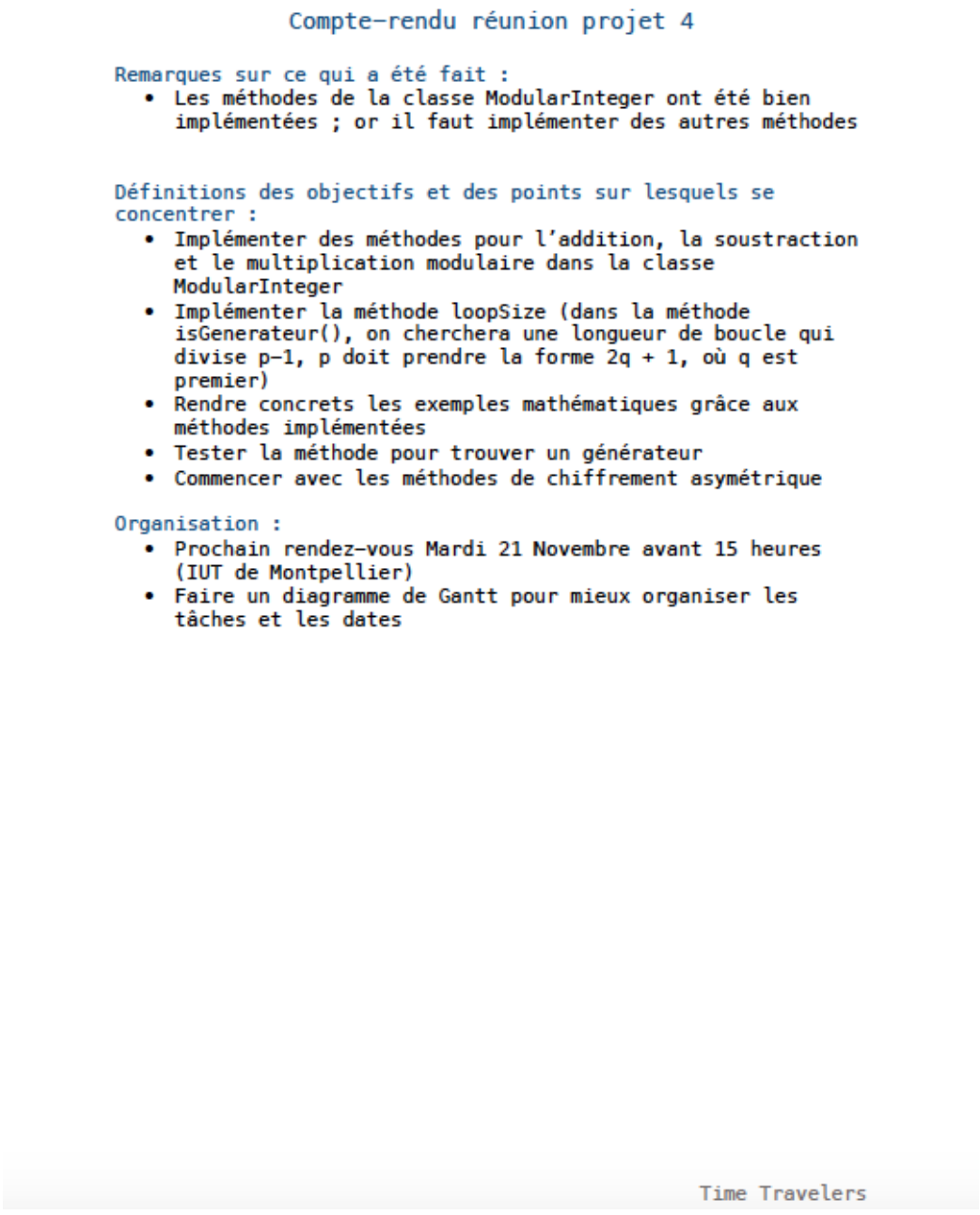
- Créer une classe modularInteger qui importe la classe BigInteger de Java pour pouvoir faire des opérations modulaires et des opérations avec de grandes valeurs. Implémenter dans cette classe la méthode isPrime(int num) qui fait appel à la fonction isProbablePrime de BigInteger et des méthodes pour les opérations modulaires qui font appel aux méthodes mod, modInverse et modPow de la classe BigInteger
- Implémenter la méthode loopSize pour trouver la taille d'une boucle
- Recoder les méthodes pour trouver le nombre générateur en utilisant la classe modularInteger et la méthode loopSize
- Chercher sur internet (Wikipedia) quelles sont las différences entre le chiffrement symétrique et asymétrique
- Commencer à faire le code pour le chiffrement asymétrique

Organisation :

- Prochain rendez-vous Mardi 14 Novembre à 13 heures (IUT de Montpellier)
- Ajouter Monsieur Lebreton au Trello : adresse mail romainlebreton@gmail.fr

Time Travelers

FIGURE 52 – Compte-rendu de la réunion 3



Compte-rendu réunion projet 4

Remarques sur ce qui a été fait :

- Les méthodes de la classe ModularInteger ont été bien implémentées ; or il faut implémenter des autres méthodes

Définitions des objectifs et des points sur lesquels se concentrer :

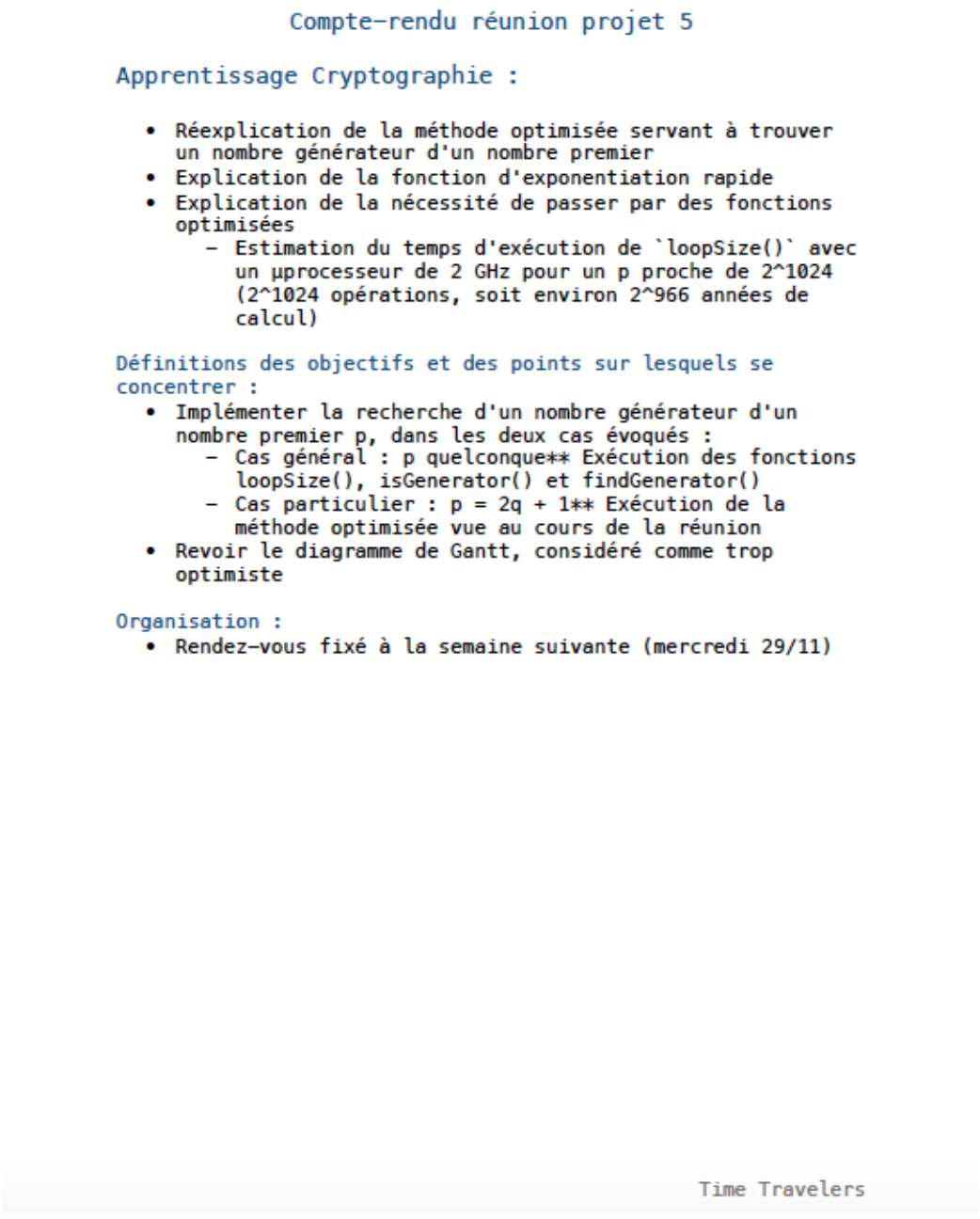
- Implémenter des méthodes pour l'addition, la soustraction et le multiplication modulaire dans la classe ModularInteger
- Implémenter la méthode loopSize (dans la méthode isGenerateur(), on cherchera une longueur de boucle qui divise p-1, p doit prendre la forme 2q + 1, où q est premier)
- Rendre concrets les exemples mathématiques grâce aux méthodes implémentées
- Tester la méthode pour trouver un générateur
- Commencer avec les méthodes de chiffrement asymétrique

Organisation :

- Prochain rendez-vous Mardi 21 Novembre avant 15 heures (IUT de Montpellier)
- Faire un diagramme de Gantt pour mieux organiser les tâches et les dates

Time Travelers

FIGURE 53 – Compte-rendu de la réunion 4



Compte-rendu réunion projet 5

Apprentissage Cryptographie :

- Réexplication de la méthode optimisée servant à trouver un nombre générateur d'un nombre premier
- Explication de la fonction d'exponentiation rapide
- Explication de la nécessité de passer par des fonctions optimisées
  - Estimation du temps d'exécution de `loopSize()` avec un µprocesseur de 2 GHz pour un p proche de 2^1024 (2^1024 opérations, soit environ 2^966 années de calcul)

Définitions des objectifs et des points sur lesquels se concentrer :

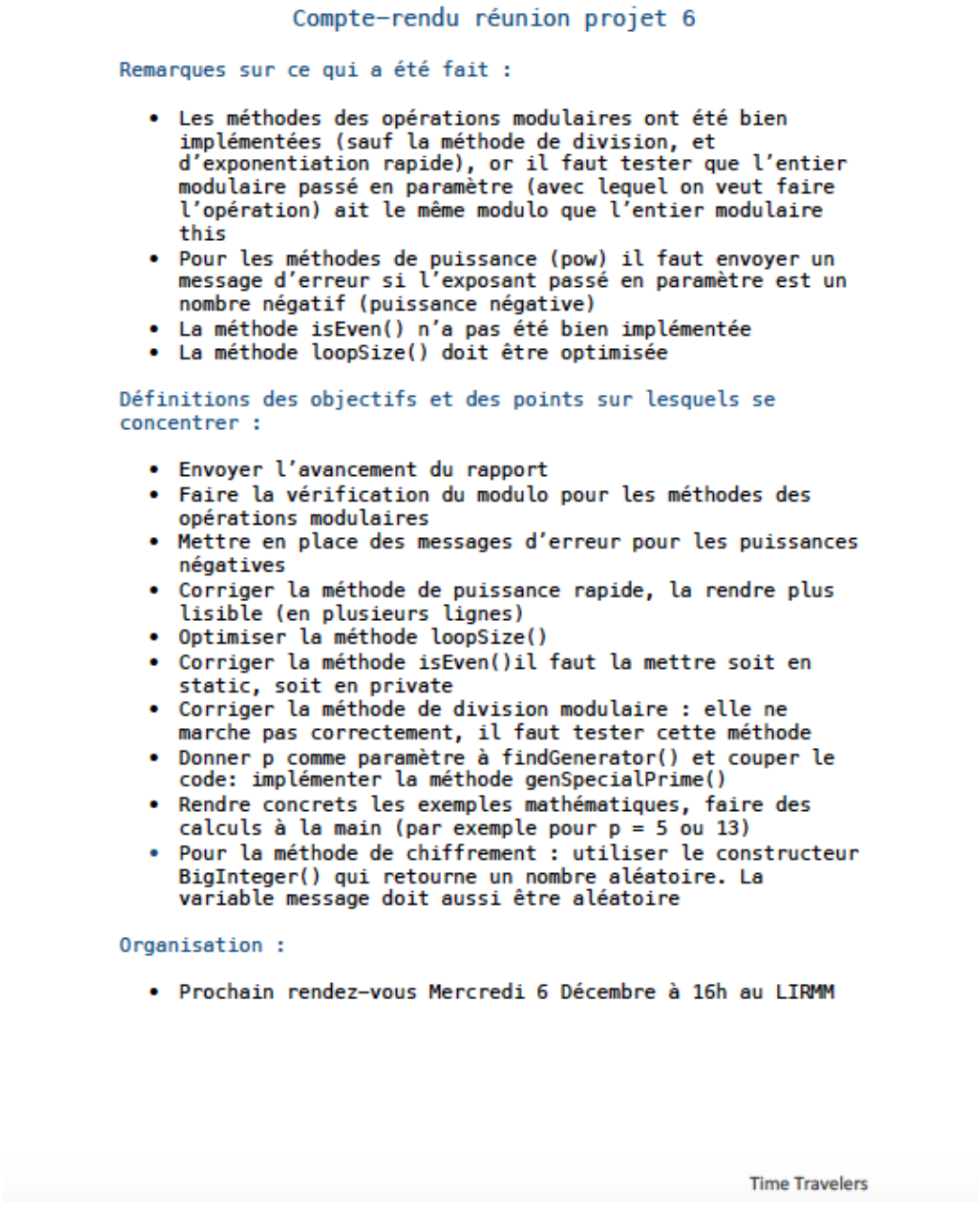
- Implémenter la recherche d'un nombre générateur d'un nombre premier p, dans les deux cas évoqués :
  - Cas général : p quelconque** Exécution des fonctions loopSize(), isGenerator() et findGenerator()
  - Cas particulier : p = 2q + 1** Exécution de la méthode optimisée vue au cours de la réunion
- Revoir le diagramme de Gantt, considéré comme trop optimiste

Organisation :

- Rendez-vous fixé à la semaine suivante (mercredi 29/11)

Time Travelers

FIGURE 54 – Compte-rendu de la réunion 5



Compte-rendu réunion projet 6

Remarques sur ce qui a été fait :

- Les méthodes des opérations modulaires ont été bien implémentées (sauf la méthode de division, et d'exponentiation rapide), or il faut tester que l'entier modulaire passé en paramètre (avec lequel on veut faire l'opération) ait le même modulo que l'entier modulaire this
- Pour les méthodes de puissance (pow) il faut envoyer un message d'erreur si l'exposant passé en paramètre est un nombre négatif (puissance négative)
- La méthode isEven() n'a pas été bien implémentée
- La méthode loopSize() doit être optimisée

Définitions des objectifs et des points sur lesquels se concentrer :

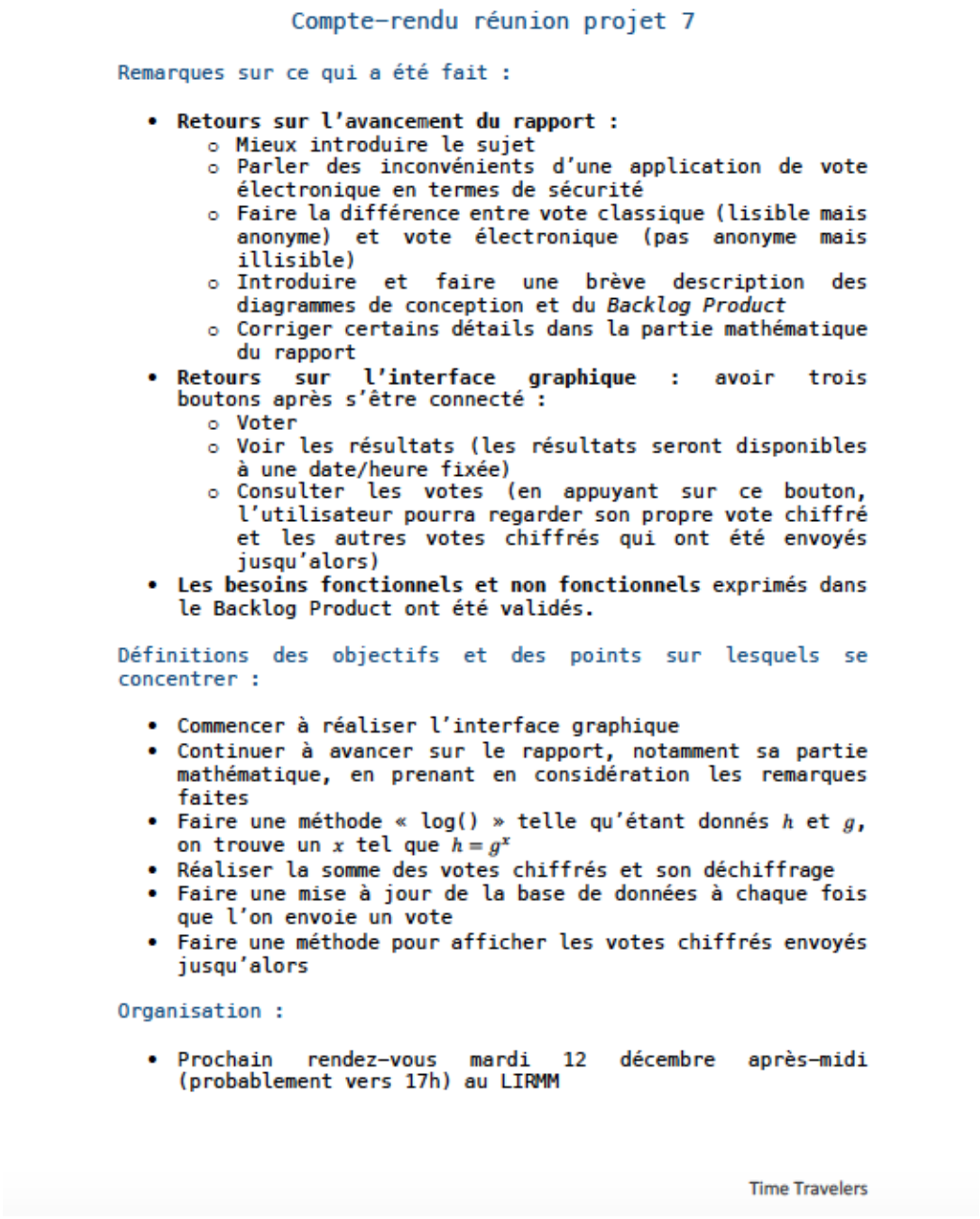
- Envoyer l'avancement du rapport
- Faire la vérification du modulo pour les méthodes des opérations modulaires
- Mettre en place des messages d'erreur pour les puissances négatives
- Corriger la méthode de puissance rapide, la rendre plus lisible (en plusieurs lignes)
- Optimiser la méthode loopSize()
- Corriger la méthode isEven()il faut la mettre soit en static, soit en private
- Corriger la méthode de division modulaire : elle ne marche pas correctement, il faut tester cette méthode
- Donner p comme paramètre à findGenerator() et couper le code: implémenter la méthode genSpecialPrime()
- Rendre concrets les exemples mathématiques, faire des calculs à la main (par exemple pour p = 5 ou 13)
- Pour la méthode de chiffrement : utiliser le constructeur BigInteger() qui retourne un nombre aléatoire. La variable message doit aussi être aléatoire

Organisation :

- Prochain rendez-vous Mercredi 6 Décembre à 16h au LIRMM

Time Travelers

FIGURE 55 – Compte-rendu de la réunion 6



Compte-rendu réunion projet 7

Remarques sur ce qui a été fait :

- **Retours sur l'avancement du rapport :**
  - Mieux introduire le sujet
  - Parler des inconvénients d'une application de vote électronique en termes de sécurité
  - Faire la différence entre vote classique (lisible mais anonyme) et vote électronique (pas anonyme mais illisible)
  - Introduire et faire une brève description des diagrammes de conception et du *Backlog Product*
  - Corriger certains détails dans la partie mathématique du rapport
- **Retours sur l'interface graphique** : avoir trois boutons après s'être connecté :
  - Voter
  - Voir les résultats (les résultats seront disponibles à une date/heure fixée)
  - Consulter les votes (en appuyant sur ce bouton, l'utilisateur pourra regarder son propre vote chiffré et les autres votes chiffrés qui ont été envoyés jusqu'alors)
- **Les besoins fonctionnels et non fonctionnels** exprimés dans le Backlog Product ont été validés.

Définitions des objectifs et des points sur lesquels se concentrer :

- Commencer à réaliser l'interface graphique
- Continuer à avancer sur le rapport, notamment sa partie mathématique, en prenant en considération les remarques faites
- Faire une méthode « log() » telle qu'étant donnés h et g, on trouve un x tel que $h = g^x$
- Réaliser la somme des votes chiffrés et son déchiffrage
- Faire une mise à jour de la base de données à chaque fois que l'on envoie un vote
- Faire une méthode pour afficher les votes chiffrés envoyés jusqu'alors

Organisation :

- Prochain rendez-vous mardi 12 décembre après-midi (probablement vers 17h) au LIRMM

Time Travelers

FIGURE 56 – Compte-rendu de la réunion 7

Résumé en français

Ce rapport de projet présente l'aboutissement à un système de vote électronique. A travers celui-ci, nous expliquons comment un système de vote électronique fonctionne, comment nous réalisons le nôtre, quels sont nos résultats obtenus et comment nous avons travaillé en équipe. Ce projet met en avant l'utilisation du chiffrement asymétrique et explique les processus nécessaires pour chiffrer un vote. De plus, le chiffrement est retranscrit en code Java, par cette transformation, on voit la nécessité d'avoir de bons algorithmes pour chiffrer et déchiffrer. L'occasion pour revoir des notions de complexité algorithmique vues au cours de notre DUT. Le système de vote électronique marche avec une interface graphique et applique un système de sockets pour client/serveur pour communiquer les votes.

Mots clés

Vote électronique, chiffrement asymétrique, travail d'équipe, Java, complexité algorithmique, interface graphique, sockets, client/serveur.

English summary

This project report shows the culmination to a electronic vote system. Through this report, we will explain how a electronic vote system works, how we made ours, what results we have obtained and how we have worked as a team to make this system. This project will showcase the use of asymmetric encryption and explains the processes required to encrypt a vote. Furthermore, the encryption is used in Java code, through this transformation, it also allows us to understand the necessity to have good algorithms to encrypt and decrypt. This is the opportunity to use notions we have learned through our institute of technology. The electronic vote system works with a graphical user interface and uses a client/server socket system to communicate the votes.

Key words

Electronic vote system, teamwork, asymmetric encryption, Java, graphical user interface, client/server, socket.