Projet PGP

Etude et implantation d'un outil graphique de gestion de clé PGP

Plan de Développement 0.3

Janvier 2015

Auteur(s): Olivier Thibault

Version	Date	Changelog
0.1	14/12/2014	Vertion initiale
0.2	01/01/2015	Ajout de la partie planing
0.3	07/01/2015	Finalisation des parties 7, 8 et 9

Table des matières

1 Contexte du projet

1.1 Origine du projet

GnuPG est un logiciel libre en ligne de commandes, qui permet de réaliser un certain nombre d'opérations cryptographiques, sur des messages ou documents, ainsi que de gérer un trousseau de clés. Cet outil peut paraître assez complexe à appréhender pour un utilisateur novice, car il existe un grand nombre de commandes et d'options différentes.

La solution serait d'utiliser une interface graphique, qui présenterait de manière plus claire les possibilités de GnuPG. Il en existe quelques unes comme Kgpg, Seahorse ou Gpa, mais malheureusement, ces interfaces ne permettent pas d'exploiter pleinement GnuPG. En effet, certaintes commandes n'y sont pas implantées, ce qui oblige les utilisateurs à revenir en ligne commandes pour pouvoir effectuer certaines opérations.

Le projet consistera donc, en partie, à proposer une interface répondant à ces problèmes.

1.2 Contexte de développement

Ce projet s'inscrit dans le cadre de la formation du master 1 en sécurité des systèmes informatiques de l'UFR des sciences et techniques de Rouen. La période de réalisation s'étend de fin janvier à fin mai 2015.

Il est demandé de développer ce projet de manière "Agile". Cela se caractérise donc par un style de conduite de projet itératif incrémental, centré sur l'autonomie des ressources humaines impliquées dans la spécification, la production et la validation d'une application intégrée et testée en continu. L'équipe en charge du projet et la cliente doivent travailler le plus possible ensemble, de manière à permettre une meilleure réactivité des développeurs face à l'évolution éventuelle du besoin. Il sera nécessaire de livrer fréquemment une version opérationnelle avec des cycles de quelques semaines, afin de s'assurer de la satisfaction de la cliente.

Des audits sont prévus en cours d'année afin de contôler la qualité de gestion du projet.

1.3 Principaux acteurs

Ce projet est proposé par Mme. Magali BARDET (enseignante et responsable du master SSI à l'UFR des sciences et techniques de Rouen) afin d'apporter une solution au problème décrit au point 1.1. La cliente pourra aussi soutenir techniquement l'équipe en charge du projet du fait de ses nombreuses connaissances en cryptographie.

M. Karim Abdellah GODARD est un intervenant et assure le rôle de consultant en gestion de projet.

L'équipe en charge du développement est constituée de sept étudiants actuellement en première année du master SSI de Rouen :

- Bertille BOUILLIE	Reponsable client
- Guillaume LEROY	Architecte
- Ibrahima Sory BARRY	Chargé client
- Lucas BARBAY	Testeurs
- Matthieu FIN	Responsable technique
- Pierre BALMELLE	Responsable qualité
- Olivier THIBAULT	Chef de projet

1.4 Objectifs poursuivis

Dans un premier temps, il est demandé de réaliser une étude complète de GnuPG ainsi que du standard sur lequel il repose (OpenPGP), en vue de rédiger un rapport illustrant toutes les fonctionnalités. Ce rapport doit permettre à un utilisateur novice en PGP de comprendre cet outil, il doit donc être rédigé dans un esprit pédagogique. De ce fait, le premier objectif est d'aquérir une connaissance pointue du sujet, tout en se rappelant les questions que nous nous sommes posées au fil de cette étude. Nous devrons être capable de prendre assez de recul de manière a pouvoir expliquer les choses comme nous l'aurions souhaité en tant que novice.

Ensuite, une interface graphique permettant d'exploiter pleinement les fonctionnalités de GnuPG de manière conviviale et pédagogique doit être proposée. Les technologies choisies pour la développer sont C++ et Qt5. L'objectif est donc de maîtriser au mieux ces technologies afin de fournir une application de qualité, qui satisfera pleinement la cliente. Le défi est aussi de faire un effort particulier quand à l'ergonomie et l'intuitivité de l'interface, surtout en ce qui concerne la partie toile de confiance. L'application devra être accompagnée d'un manuel d'utilisation et inclure une aide pour guider l'utilisateur. Ces deux points nécessiteront un certain recul sur nos travaux pour qu'ils puissent être parfaitement compréhensibles par un novice en PGP.

Enfin, il est demandé d'effectuer une étude des limites cryptographiques de PGP et de produire un document qui décrit ces limites. À travers ce document, l'objectif est de renforcer notre sens de l'analyse en tant que futur professionnel de la sécurité informatique.

1.5 Documents de référence

Documents utiles à létude du sujet :

- Définitions du standard OpenPGP RFC 4880.
- Le manuel de GnuPG : https://www.gnupg.org/documentation/manuals/gnupg/.
- Le guide utilisateur de GnuPG : https://www.gnupg.org/gph/fr/manual.html.
- Exemple de visualisation d'une toile de confiance : https://www.archlinux.org/master-keys/#visualization.

Documents de gestion de projet :

- La spécification technique du besoin : Ce document traduit les besoins du client et recence les engagements de notre équipe dans le cadre du projet.
- Le cahier de recette : Ce document présente les moyens mis en oeuvre afin de vérifier que le produit est conforme aux attentes du client.
- Le document d'architecture logiciel : Ce document décrit les solutions techniques proposées et notre démarche de développement.
- Le document d'analyse des risques : Ce document établit les risques qui pourraient nuire au bon déroulement du projet ainsi que les solutions associées.

2 Méthodologie de développement

Notre méthodologie de développement s'inspirera majoritairement de la méthode agile Scrum et quelque peu d'Extreme Programming en ce qui concerne les tests. Le projet sera donc découpé en boîtes de temps (appelées "sprints"), de quatre semaines environ, et comportant différentes étapes :

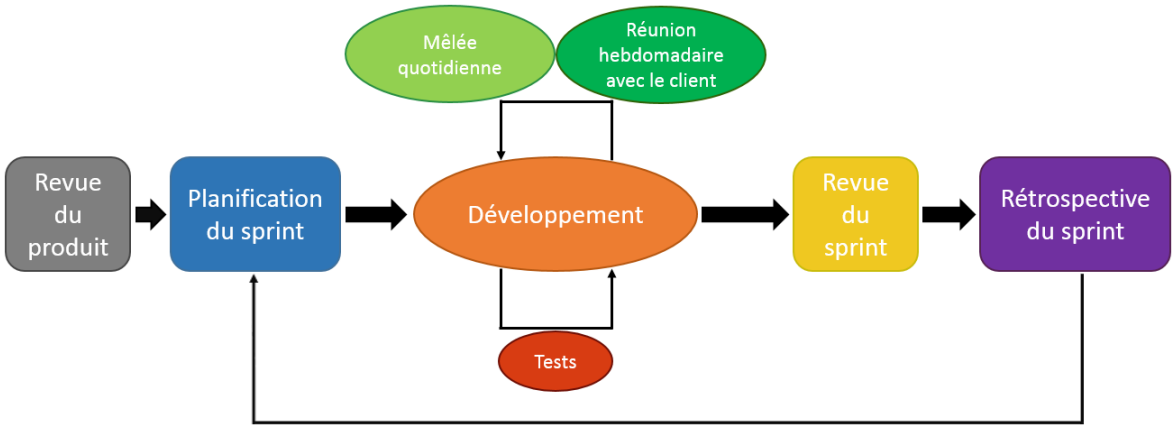

Figure 1 : Schéma d'un sprint

2.1 Revue du produit :

Cette étape a déjà été effectuée pendant la phase de l'avant projet et est materialisée par la spécification technique du besoin. Cependant, à chaque début de sprint, une réunion avec la cliente permettra de mettre à jour le document en fonction de l'évolution du besoin si nécessaire.

2.2 Planification du sprint :

Cette étape est déjà en partie réalisé pour les sprints prévus mais sera peut-être amené à évoluer au cours du projet. C'est pourquoi, en début de sprint, un but est décidé en fonction de l'état d'avancement du projet. Pour atteindre cet objectif, toute l'équipe devra définir avec la cliente, lors d'une réunion de planification de sprint, quels éléments du carnet du produit seront réalisés.

2.3 Développement :

Cette étape est constituée de plusieurs évènements :

Avant chaque journée de développement aura lieu une réunion (la mêlée quotidienne). Cet évènement permet aux développeurs de faire un point de coordination sur les tâches en cours et sur les difficultés rencontrées. Cette réunion dure environ quinze minutes et ne concerne que l'équipe des développeurs.

Chaque semaine l'équipe se réunira avec la cliente afin de faire le point sur l'avancement, ainsi que pour échanger d'éventuelles questions ou faire part de changement.

Chaque composant développé devra passer des tests unitaire, d'intégration et validation. Ces tests seront écrits par les testeurs et mis à disposition des autres développeurs. Ils devront éventuellement être modifier ou compléter si besoin. La cliente pourra aussi demander d'effectuer ces tests lors d'une réunion afin de vérifier la qualité du produit.

2.4 Revue du sprint :

À la fin d'un sprint, l'équipe et la cliente se réuniront pour effectuer la revue du sprint. L'objectif de la revue est de valider l'incrément de produit qui a été réalisé pendant le sprint. L'équipe présentera les tâches réalisées et fournira au préalable le livrable au client. Ensuite, une discussion sur l'état courant du projet amènera ou non à un ajustement du produit.

2.5 Rétrospective du sprint :

Cette étape se fait avec l'ensemble de l'équipe Scrum (Cliente, Scrum master et développeurs). Elle a pour but l'adaptation aux changements qui surviennent au cours du projet et l'amélioration continue du processus de réalisation. L'objectif est donc de relever d'après l'itération précédente, les éléments du processus qui ont bien fonctionnné et ceux qui sont à améliorer. Il s'en déduira un plan d'actions d'améliorations à mettre en place lors de l'itération suivante.

3 Organisation et responsabilités

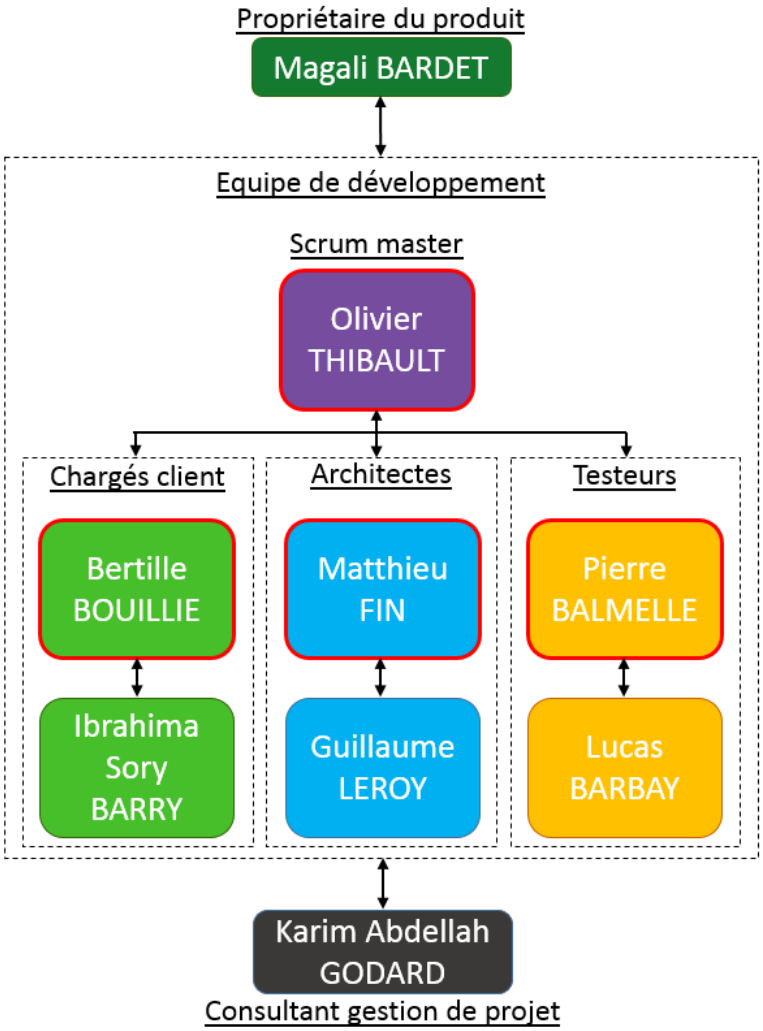

Figure 2 : Organigramme des participants au projet

Propriétaire du produit :

La propriétaire du produit est la cliente mais représente aussi les utilisateurs du produit. Son rôle est de définir les éléments du produit, l'ordre dans lequel les fonctionnalités seront développées et doit s'assurer que le produit est compris de l'équipe de développement. C'est également la cliente qui fixe les objectif d'un sprint, et peut à tout moment l'interrompre en cours.

Il est aussi important que la cliente reste disponible pour répondre aux questions de l'équipe ou pour lui faire part de ses retours. La cliente fait partie intégrante de l'équipe Scrum.

Equipe de développement :

L'équipe de développement est constituée de sept personnes. Chaque membre de l'équipe possède un second rôle spécifique. L'équipe a pour résponsabilité de livrer à chaque fin d'itération une nouvelle version de l'application enrichie de nouvelles fonctionnalités et potentiellement utilisable.

Scrum master :

Le Scrum master est responsable de la compréhension, de l'adhésion et de la mise en oeuvre de la méthode. C'est un "leader au service de l'équipe". Il assiste chaque rôle de l'équipe Scrum dans son activité. Il définit la durée des itérations, des modalités et de l'odre du jour des réunions.

Il a aussi d'autres attributions commes :
— communiquer la vision et les objectifs à l'équipe ;
— faciliter les rituels de Scrum ;
— coacher l'équipe de développement ;
— écarter les éléments pouvant perturber l'équipe ;
— aider à la coordination.

Chargés client :

Les chargés clients représente l'équipe de développement auprès du client. Ils sont là pour faciliter la communication entre l'équipe et le propirétaitaire du produit. Ils doivent être capable d'avoir une vision à la fois général et technique du produit. Les chargés client doivent donc être en mesure de reformuler le besoin et de le spécifier afin d'en assurer la compréhension par l'équipe en tenant compte de l'approbation du propirétaitaire du produit.

Architectes :

Les architectes ont pour mission de concevoir et de modéliser les différents composants de l'application à développer à partir de la spécification technique de besoin. C'est eux qui définissent l'environnement technique de développement (langages, outils...). Ils ont aussi un rôle de support technique si besoin pour les autres membres de l'équipe.

Testeurs :

Les testeurs sont chargés de concevoir et écrire les tests permettant la validation du code de l'application. Ils sont les responsables de la qualité du produit. Leurs tests doit donc non seulement vérifirer la validité du code et son bon fonctionnement, mais aussi la validité vis à vis des exigences du propriétaire du produit.

Consultant gestion de projet :

Le consultant veille à l'aquisition des connaissances nécessaires et guide l'équipe dans la succession des étapes. Il est présent pour former l'équipe et contrôler le bon avancement du projet, ainsi que pour répondre à d'éventuelle question des membres de l'équipe de développement. Il réalisera deux audits en cours de réalisation.

4 Organigramme des tâches

Voici l'organigramme des tâches élémentaires définit pour les trois sprints prévus :

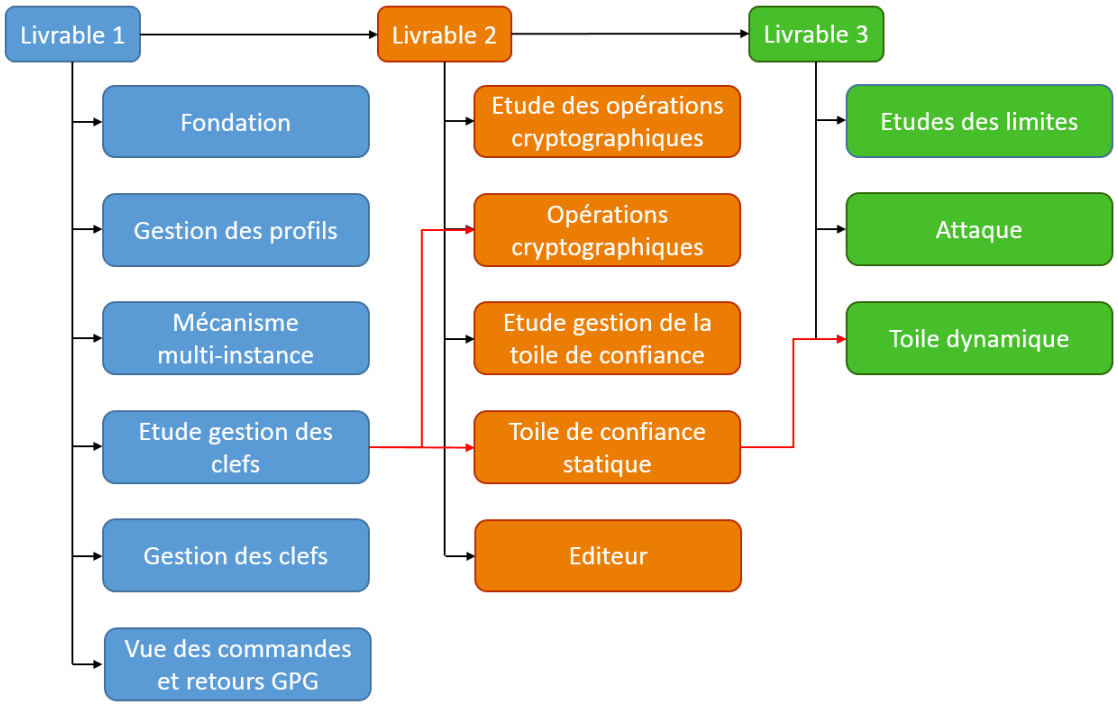

Figure 3 : Organigramme des tâches

À l'intérieur de ces taches, des sous-tâches seront définis et réalisés soit par un membre soit en binôme. Les flèches rouges indiquent une dépendance entre tâches élémentaires.

5 Évaluation du projet et dimensionnement des moyens

5.1 Evaluation globale de la charge

Le tableau ci dessous à été obtenue selon la technique d'estimation POKER. La charge exprimée en pourcent correspond donc à une moyenne des estimations de chaque membre de l'équipe de développement.

En bleu sont réprésentés les tâches du premier sprint, en orange celles du second, et en vert celles du derniers.

Catégories	Charge (%)
Fondation	5
Mécanisme multi-instance	2
Gestion des profils	6
Étude gestion des clefs	6
Gestion des clefs	9
Vue des commandes et retours GPG	2
Étude des opérations cryptographiques	6
Opérations cryptographiques	7
Étude gestion de la toile de confiance	8
Toile de confiance statique	11
Éditeur	4
Étude des limites	12
Attaque	12
Toile dynamique	10
Total	100

Figure 4 : Tableau d'évaluation de la charge par tâche

Le premier sprint a donc une charge de 30%, le second 36%, et le dernier 34%.

5.2 Besoin en moyens et en ressources

Il a été décidé que l'application sera codé en C++ et Qt5. Pour cela, chaque membre de l'équipe de développement devra installer sur son ordinateur QtCreator 3.3, qui est un EDI permettant de développer ce type de projet. Pour les tests, le module Qt test ainsi que le framework Cppunit seront utilisés.

Tout le contenu des livrables sera sur notre dépot Gitlab, le même que celui utilisé pour gérer les documents d'avant projet.

Pour la communication à distance, l'espace Slack utilisé lors de l'avant projet sera toujous à disposition de l'équipe.

6 Planning général

Entre la date de début et de fin de réalisation nous avons 16 semaine de disponible. Pour des raisons de départ prévu, des révisions et d'autres projet scolaire à venir, nous avons décidé de décompter les 3 semaines de vacances scolaires, ce qui nous donne 13 semaines. Une semaine de marge est prévue et nous ramène ainsi à 12 semaines de développement.

Le projet sera découpé en trois sprint de 4 semaines. Une semaine comptera 5 jours de développement, du lundi au vendredi. Le projet comptera donc 60 jours de travail, soit 20 jours par sprint.

Nous estimons une moyenne de disponibilité par membre à une 1h30 par jour. Soit pour 7 développeurs, 52h30 de temps de travail par semaine, donc 210h par sprint pour réaliser un certain nombre de tâches.

Le premier jour de chaque sprint aura lieu une réunion permettant de mettre au claire la planification de celui ci. En cours de sprint, les réunions hebdomadaires qui ne figure pas sur le planing devront être prévue avec le propriétaire du produit. À la fin du sprint, aura lieu la ou les réunion(s) de présentation du livrable, et de rétrospective.

Vous trouverez en annexe le planing général ainsi que celui du sprint 1.

7 Procédés de gestion

7.1 Gestion de la documentation

Au cours du projet plusieurs documents devrons être rédigés :
— Un rapport sur l'étude de GnuPG/OpenPGP ;
— Un rapport sur l'étude des limites de PGP ;
— Un manuel d'utilisation de notre application.

En ce qui concerne le rapport sur l'étude de GnuPG/OpenPGP ainsi que le manuel d'utilisation, chacun aura pour résponsabilité de rédiger une partie du document en rapport avec les tâches qu'il aura choisi de développer. Pour l'études des limites la question n'a pas encore été réfléchie. Chacun des membres devra relire les parties qu'il n'a pas rédigé afin d'apporter des suggestions éventuelles. Les documents devront ensuite être approuvés par la cliente.

En ce qui concerne leur gestion, il seront stockés sur notre dépot Gitlab, au sein du répertoire "documents" et d'un sous répertoire du même nom que le document à produire. Ils seront rédiger en LaTeX selon la demande du propirétaitaire du produit.

7.2 Gestion des configurations

Le code produit sera stockés sur notre dépot Gitlab. Voici le workflow qui sera mis en place pour gérer l'évolution et les différentes version produites :

Figure 5 : Workflow mis en place pour le développement

Les branches principales, fixes et immuables :

Master est la branche où tout est stable. Chaque commit correspond à une version stable du projet (release) qui peut être déployée en production et taguée en conséquence (vX.Y.Z).

Develop est la branche sur laquelle s'effectue le développement proprement dit. On y prépare les changements en vue de la prochaine release dans master.

Les branches secondaires qui se font et se défont avec le temps :

Feature part de develop et se merge dans develop. On crée une branche feature/xxx lorsque l'on travaille sur une fonctionnalité en particulier. Lorsqu'elle est terminée, on la merge dans develop pour ajouter la feature stable dans le scope de la prochaine release.

Release part de develop et se merge dans master et develop. On crée une branche release/xxx à partir de develop lorsque celle-ci reflète l'état désiré de la release (l'ensemble des fonctionnalités du scope ont été mergées). Ainsi, on peut préparer la prochaine release tranquillement, corriger d'éventuels bugs et poursuivre le développement en parallèle. Une fois que la release est prête (stable) on merge alors la branche dans master, mais aussi dans develop pour mettre à jour les modifications apportées.

Hotfix part de master et se merge dans master et develop / release. On crée une branche hotfix/xxx lorsque l'on veut résoudre un bug critique en production rapidement. C'est un peu comme une release non planifiée. Lorsque le correctif est développé, on le merge dans master avec le numéro de version qui convient, ainsi que dans develop (ou la branche release en cours, le cas échéant) pour mettre à jour les modifications apportées.

Les bonnes pratiques :
— Une feature = une partie bien spécifique du code ;
— Une feature = une branche ;
— Spliter des changements en petites étapes = commit ;
— Commiter le plus souvent possible ;
— Faire une review du code avant commit

À ne pas faire :
— Developper sur la branche master
— Developper sur la branche de developpement
— Faire un rebase sur une branche impliquant plusieurs développeurs
— Supprimer des branches non mergées
— Utiliser git rebase interactive sur des commit déjà poussés sur le dêpot central

8 Revue et points clef

Voir en section 2 les points 2.1, 2.2, 2.4 et 2.5, ainsi que le planing général en annexe.

9 Procédure de suivi et d'avancement

La mélée quotidienne durera environ quinze minutes et chaque membre devra s'exprimer sur trois points :
— Ce qu'il a fait la veille ;
— Ce qu'il prévoit de faire aujourd'hui ;
— Ses problèmes, blocages ou remarques.

Un fichier excel sera créer par le suivre l'avancement du sprint. Il listera toute les taches à réaliser, à qui elles sont attribués, les date de début et de fin prévue et un statut.

Un fichier excel sera créer pour suivre les différentes actions prévues à l'issue des réunions.

10 Annexes

10.1 Planing général

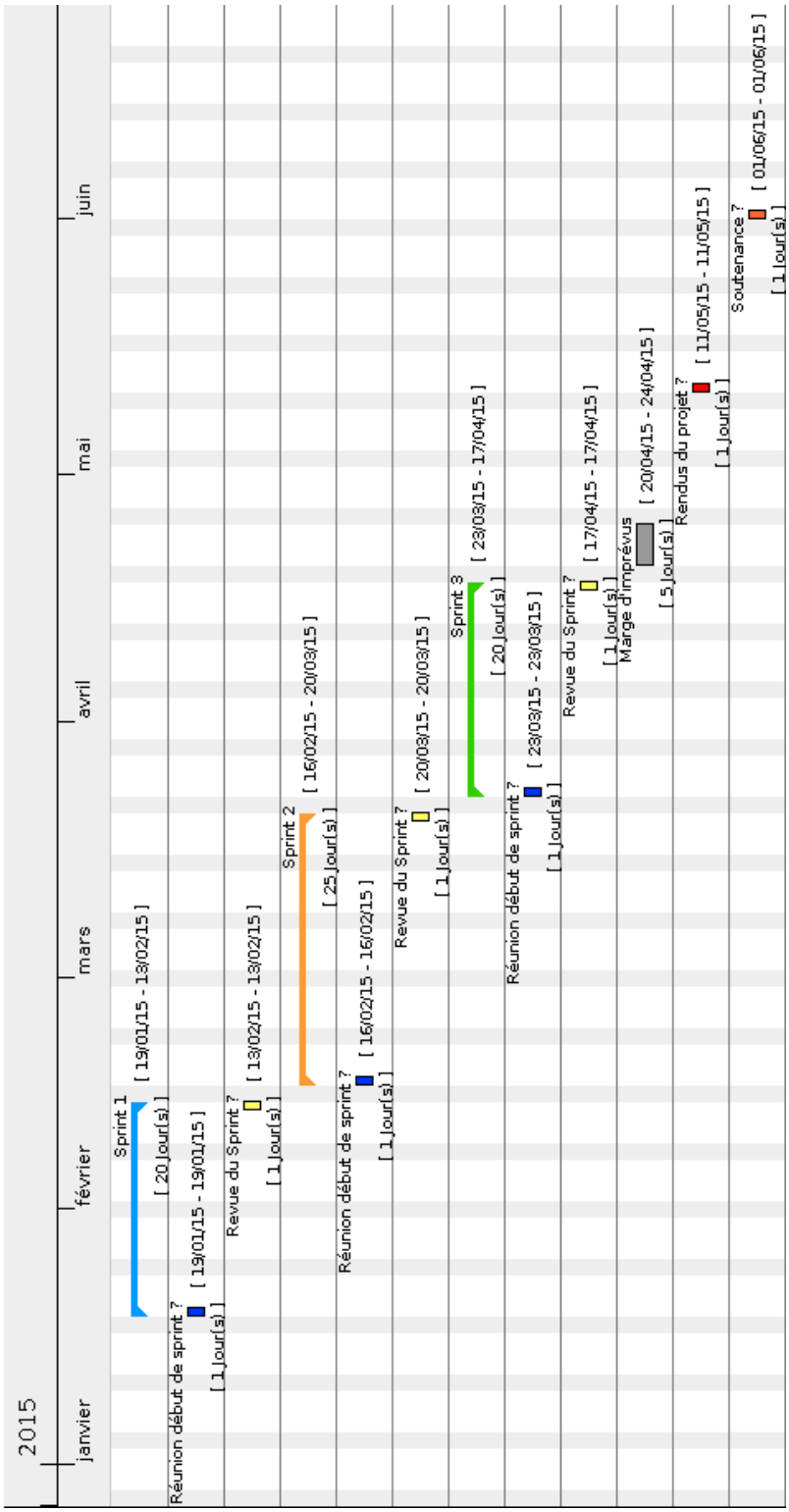

Figure 6 : Planing général

10.2 Planing du sprint 1

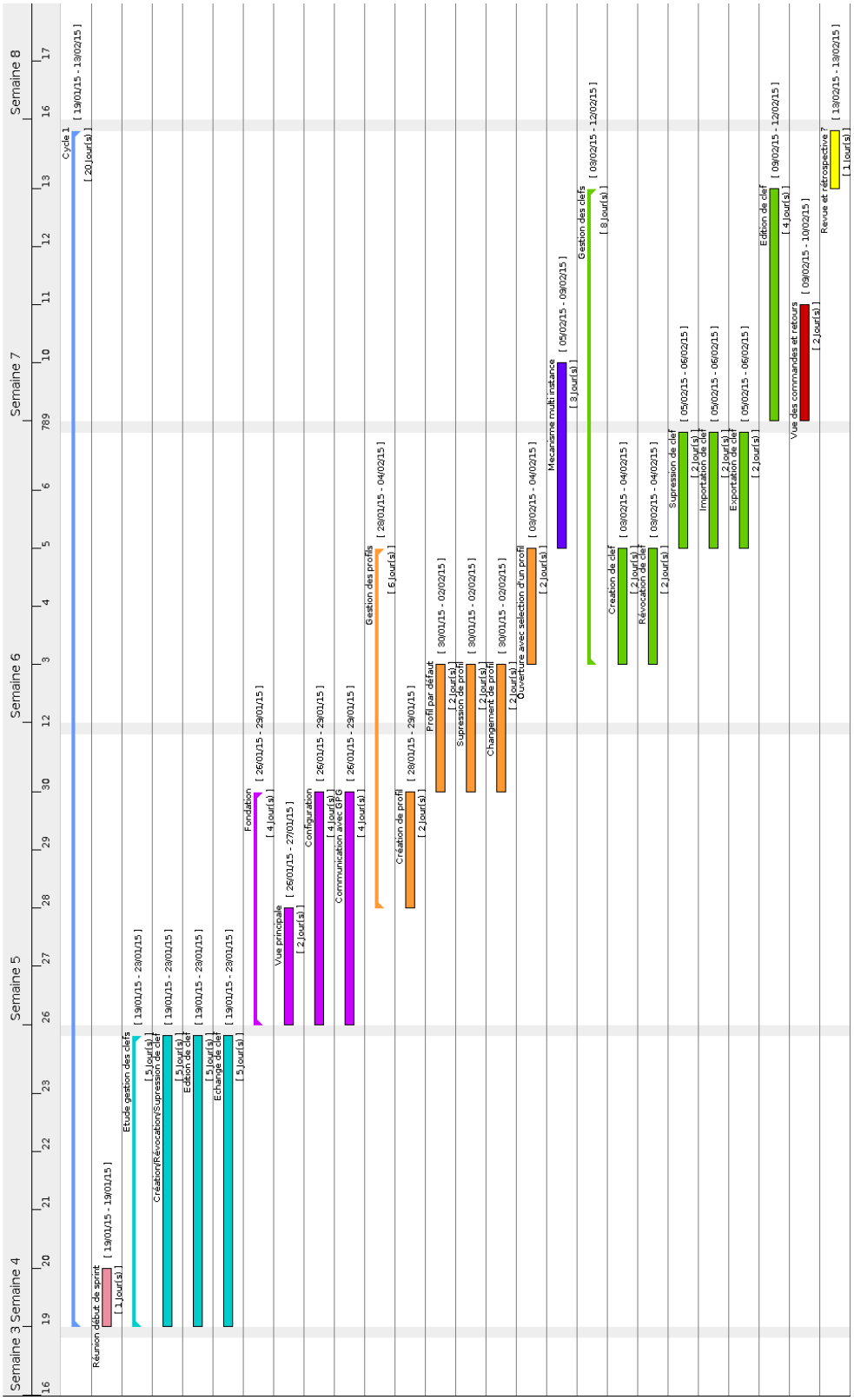

Figure 7 : Planing du sprint 1 (1j/h = 1h30)