LOG2990 – Projet de logiciel d'application Web

Hiver 2019

Rapport final

Groupe 2

Équipe 205

Stéphane Michaud [1904016]

Annie Rochette [1896142]

Antoine Lamontagne [1907605]

Jonathan Hélias [1893565]

Rossane Larocque [1872448]

Soumis à : Lévis Thériault

Remis : vendredi 19 avril 2019

Table des matières

Introduction

Parmi tous les cours de notre cheminement à l'école Polytechnique, les plus formateurs sont assurément nos cours de projets. Ces cours amènent non seulement les étudiants à appliquer leurs connaissances acquises dans d'autres cours, mais leur permettent également de développer davantage de connaissances de manière autonome. De plus, effectués en équipe, ces projets donnent aux étudiants une expérience concrète de travail collectif dans un environnement de conception technologique. Les programmes de génie informatique et de génie logiciel proposent notamment la conception d'une application web comme projet de deuxième année. Ce projet sera donc le sujet du présent rapport, en s'intéressant plus particulièrement au cheminement de notre équipe: l'équipe 205. Ainsi, nous discuterons de notre gestion d'équipe, de nos différentes méthodes de travail et des résultats obtenus suite à ces approches. Étant donné que le projet se divise en quatre sprints, des discussions sur le déroulement de chaque sprint seront d'abord présentées. Ces discussions se baseront sur des données accumulées lors du déroulement du projet. Ensuite, nous évaluerons le déroulement du projet de manière globale.

Sprint 1

Tâches et répartitions

Le sprint 1 consistait d'abord en une familiarisation avec les technologies utilisées lors du projet. Les premières heures consacrées au projet ont donc pour tous été passées à suivre des tutoriels en ligne de manière plutôt individuelle. On nous a ensuite présenté les tâches concrètes reliées à ce sprint. Ces dernières mentionnaient entre autres de créer l'interface de base de notre application web (page d'accueil, page d'administration et liste des parties). La première partie de logique y était également présente et nous devions implémenter l'algorithme de détection de différences sur une image BMP.

Pour répartir les tâches, nous avons choisi de subdiviser l'équipe en trois parties, dont deux binômes et une personne seule. Les tâches les plus simples ont été attribuées à la personne travaillant seule et les autres tâches ont été réparties par préférence parmi les binômes.

Approche

Puisqu'il s'agissait de notre première expérience de travail ensemble, nous avons établi des normes quant aux médias de communication et au codage. Parmi ces normes se trouvent notamment des règles concernant le format des *commits*, la langue utilisée pour le codage et le format des divers éléments du code (fonctions, variables, constantes, etc.). Nous avons également tenu à ce que des tours de table soient effectués à chaque rencontre pour s'assurer de l'avancement des tâches et pour faciliter la mise en commun. Étant donné que nous avions de la difficulté à trouver une plage horaire commune à l'extérieur des heures de cours, nous avons suggéré de communiquer en ligne dans le cas où nous manquerions de temps pour terminer nos tâches. Nous avons également mentionné que lorsqu'une personne termine avec une tâche, elle se doit de demander si quelqu'un a besoin d'aide avec la sienne. Si personne ne se manifeste, cette personne pourra ensuite débuter avec une nouvelle tâche. Ainsi, aucune mesure n'a été prise pour valider que les tâches soient terminées dans les temps. Il s'agissait donc de la responsabilité de chacun de valider son propre travail et d'établir son échéancier. Finalement, pour ce sprint nous n'avons pas adopté de règles quant aux tags d'activités de développement, de tests ou de linting, par simple oubli.

Résultats

Tout d'abord, notre manque de discipline par rapport à la taille des *commits* est apparent lorsque nous regardons les graphiques (figures 10 et 12 pour le sprint 1). La moyenne d'insertion et de suppression par commit était en général trop grosse pour quatre des membres de l'équipe. Cependant, il est à noter que puisque nous n'avions pas commencé à mettre des balises ([R], [D], [T], [A], etc.) pour tous nos *commits*, ces données pourraient être légèrement majorées dû aux *commits* de restructuration ([R]) qui implique généralement un grand nombre de suppressions et d'insertions par commit. Par la suite, nous observons que le nombre d'insertions, de suppressions et de *commits* était

plutôt élevé par rapport aux autres sprints et plus particulièrement pour SM et AL (figures 1, 4 et 7 pour le sprint 1). Ceci est dû au fait que ces deux membres avaient déjà travaillé avec certaines des technologies utilisées comme les sockets par exemple ce qui leur permettait de finir leurs tâches plus vite avec moins de recherche nécessaire. Ils pouvaient donc par la suite travailler sur les tâches restantes augmentant donc leur nombre de suppressions et d'insertions. En ce qui concerne l'ensemble des membres, ces données élevées s'expliquent par la création de multiples fichiers. En effet, puisqu'aucun fichier de développement n'existait avant le premier sprint, ce fut un sprint dans lequel énormément de fichiers ont été créés.

Les nombres d'heures présentés dans la figure 14 sont explicables par notre approche décentralisée où chaque personne faisait de l'autogestion en ce qui concernait leur temps et leurs tâches entreprises. Ceci explique pourquoi au final certaines personnes ont travaillé plus longtemps sur le projet au final. Bien que nous n'ayons pas de métriques quant aux types activités de développement, nous évaluons que la majeure partie du temps consacré à ce sprint était en lien avec la recherche et le développement. En effet, puisque les tâches concernaient surtout la vue, il y avait moins de temps alloué au débogage et aux tests. Toutefois, notre manque de discipline avec le *linting* s'est fait ressentir vers la fin, car nous avons dû passer beaucoup d'heures sur l'assurance qualité quelque temps avant la remise. Pour finir, en regardant la figure 14 nous pouvons voir qu'une grande partie des heures mises sur le sprint se passe peu avant la remise. Ceci est dû au fait qu'en général, nous avions travaillé moins lors des deux premières semaines.

Pour conclure, ce premier sprint à une approche moins stricte et moins formelle pour son développement ce qui mener à des résultats indésirables surtout en ce qui concerne la moyenne d'insertions et de suppression par *commits*.

Sprint 2

Tâches et répartition

Pour le sprint 2 nous avons commencé à utiliser ThreeJS pour l'implémentation des jeux en point de vue libre, nous avons créé notre base de données ainsi que la vue du jeu et nous avons ajouté la détection des différences pour les jeux en point de vue simple. Nous avons aussi profité du sprint 2 pour faire des ajustements à notre sprint 1 suite aux rétroactions que nous avons reçues. Puisque plusieurs tâches ressemblaient au sprint précédent, les binômes sont restés semblables, mais pour les tâches restantes elles ont été divisées selon les préférences de chacun.

Approche

Pour donner suite au sprint 1, nous avons amélioré notre façon d'utiliser notre répertoire git. Nous avons créé davantage de branches, une pour chaque tâche ou composante qui pourrait impacter le développement d'une autre. Nous avons aussi créé une branche « dev » sur laquelle se trouve les tâches complétées, c'est-à-dire sans bogue et correctement testée, pour que nous puissions les intégrer dans nos branches respectives au cours de notre développement. Nous avons aussi décidé de faire des *commits* plus régulièrement, donc de plus petits *commits*, car cela nous permet d'avoir un meilleur suivi de ce qui est modifié et de revenir en arrière plus facilement si un bogue est introduit. De plus, pour ce sprint, nous avons aussi mis davantage d'emphase sur les tests de nos fonctionnalités ainsi que sur le code de qualité. Pour ce faire nous avons utilisé des outils pour le *coverage* de nos tests et nous avons installé un outil pour la détection des erreurs de *linting,* ce qui nous permet de les corriger au fur et à mesure que nous écrivons du code.

Résultats

Par rapport au sprint 1, nous pouvons observer que la quantité de *commits* par personne est devenue plus uniforme à travers l'équipe (figure 1 pour le sprint 2). Une chose que nous pourrions en conclure est que les personnes qui faisaient de trop gros *commits* (donc généralement moins de *commits*) ont su mieux les diviser donc notre objectif a été atteint. Aussi, nous pouvons observer que notre temps mis sur le projet a bien été réparti tout au long de ce sprint comparativement au sprint 1 et 3 (figure 15). Cela est surtout dû au fait que nous avons commencé à tester nos fonctionnalités au fur et à mesure que nous les développions. L'outil de détection d'erreurs de *linting* nous a aussi permis d'avoir moins d'erreurs de dernières minutes à corriger contrairement à ce qui s'était produit pour le sprint 1. Comme nous avions aussi moins de nouvelles choses à apprendre comparativement au sprint 1, nous avons eu un début de sprint avec un peu moins d'heures mis sur le projet qu'au premier sprint (figure 15).

De plus, comme au sprint 2 nous n'avons pas eu besoin de changer à nouveau de cadriciel et que plusieurs des composantes nécessaires avaient déjà été créées, nous

avons moins d’insertions et de suppressions comparativement au sprint 1 (figures 10 et 12 pour le sprint 2).

Pour conclure, comme le sprint 1 avait surtout été un sprint pour l’apprentissage des technologies, le sprint 2 a surtout été un sprint où nous avons fait des ajustements sur notre façon de travailler et d’utiliser les différents outils à notre disposition. Ce sprint nous a tout de même permis de faire des ajouts majeurs dans notre projet notamment avec la scène en point de vue libre géométrique et avec la base de données.

Sprint 3

Tâches et répartition

Le sprint 3 était l'un des sprints les plus importants du développement de notre application. De ce fait même il s'agissait du sprint le plus difficile et gros. En effet, ce sprint était celui dans lequel notre jeu devenait vraiment jouable, à la fois en 2D et en 3D, et il s'agissait aussi de celui dans lequel on créait la scène thématique 3D. Pour que notre jeu devienne jouable nous avions multiples tâches, tel que la gestion du début, du déroulement et de la fin de la partie. La gestion des erreurs de détection, du chronomètre, des messages de jeu et des tableaux des temps étaient aussi à faire. De plus un gros travail était présent pour le jeu en mode libre, c'est à dire en 3D. Nous devions aussi changer nos anciennes fonctionnalités à la suite du retour du sprint 3.

Ainsi, devant toutes les tâches, nous avons décidé de ne pas répartir toutes les tâches dans l'immédiat, mais plutôt commencer à faire certaines, puis en faire d'autres une fois celles-ci terminées. Il y a quand même eu, au final, une division des tâches selon leur type, car il est plus productif pour une personne de continuer sur ce qu'il a commencé que de reprendre une tâche complètement différente. Alors, le travail s'est réparti facilement, c'est-à-dire que deux personnes s'occupaient du 3D au complet, avec une des deux focalisées principalement sur la scène thématique. Deux autres s'occupaient du déroulement de la partie en général, avec chacun leurs tâches qui se complétaient. La dernière personne était entièrement dédiée à la gestion des messages. C'est à la fin de ces tâches que tous se préoccupent d'avoir suffisamment de tests, vérifiaient le code des autres, et corrigent les erreurs du sprint précédent.

Approche

Au moment du sprint 3, nous avions déjà acquis une bonne expérience avec les technologies utilisées et avions moins de recherche à faire. Ainsi, comme nous n'étions pas stressés par les technologies, nous avons approché ce sprint, pourtant très important, de manière relaxe. Même si nous n'avions pas eu de notes exceptionnelles aux deux précédents sprints nous n'avons pas changé notre manière de fonctionner par rapport au sprint précédent. Chacun faisait ainsi ses tâches de son bord durant environ les deux premières semaines du sprint. C'est dans les dernières semaines de ce sprint, lorsque nous avions mis notre code en commun, que nous avons vraiment plus travaillé ensemble par rapport au sprint précédent. Des journées entières ont été passées en équipe, que ça soit à l'école ou en ligne. L'équipe entière n'était pas toujours présente, mais savoir qu'au moins un autre membre travaille aussi fort que soi aide à la productivité et surtout, à approuver le code.

Résultats

Tout d'abord, comme mentionné plus haut, le sprint 3 était le plus gros du développement de notre logiciel, et cela se remarque notamment dans les modifications au code ainsi qu'au temps passé dessus.

En effet, en comparaison avec les autres sprints (figure 3) on remarque que ce sprint est celui avec le plus de *commits*, 255. Cela montre aussi que nous ne faisions pas des commits trop gros. Si l'on regarde les graphiques des insertions et des suppressions (figures 4 et 7) par sprint on remarque aussi, si l'on ne prend pas en compte le sprint 1 où tous les fichiers ont été créés à partir de 0, qu'il s'agit du sprint avec le plus d'insertions et de suppressions. Deux membres de l'équipe, SM et RL, (figures 1 et 4) se remarquent notamment par leur haut nombre de *commits* et d'insertions. En analysant les balises et les descriptions de leurs *commits*, on arrive à comprendre pourquoi. En effet, on remarque que le développement sur la scène thématique a demandé beaucoup d'insertion de modèles et beaucoup de modifications sur le code existant déjà pour la scène géométrique. On remarque aussi que le changement de format du service *render* prend une part importante du sprint et a été fait à plusieurs reprises au cours du développement.

Non seulement le sprint 3 est le plus important au niveau des *commits* mais il s'agit aussi du sprint le plus long à terminer avec environ 260 heures totales, soit 60 de plus que les autres sprints. En regardant le travail par jour, on remarque que la scène thématique a été la partie du sprint la plus longue à réaliser, avec RL (figure 16) commençant à travailler dessus dès le début du sprint. On remarque aussi que dans les jours avant la remise toute l'équipe a passé de nombreuses heures sur le projet, environ 9 heures par jour par personne dans les quatre derniers jours. En analysant le graphique de la tendance des heures par jour, (figure 16) on comprend que peu de travail a été effectué dans les premières semaines du sprint. Il y a donc eu un travail massif à faire dans la dernière semaine, ce qui a été un peu stressant, mais aussi nous a permis d'être très productifs. Plus d'heures ont été passées sur les tests et sur l'assurance qualité par rapport aux sprints précédents, ce qui nous a permis d'avoir un meilleur code.

Pour terminer sur le sprint 3, il s'agissait du sprint le plus important et le plus long du développement. Malgré un gros travail un peu tardif, nous avons conservé notre méthode de répartition des tâches et d'approche du sprint pour réussir à le compléter. Nos nombreuses heures de travail paraissent dans notre code et dans le résultat que nous avons obtenu.

Sprint 4

Tâche et répartition

Dans le cadre du sprint 4, nous devions implémenter une façon de jouer au jeu des différences en mode multijoueur et donner une rétroaction aux joueurs à la fin d'une partie. Étant donné que l'architecture de l'application permettait déjà à deux joueurs de jouer au même jeu en même temps, c'est le jumelage des joueurs qui a pris le plus de temps lors de l'implémentation du mode multijoueur. En ce qui concerne les jeux en 3D, nous devions aussi gérer les collisions de façon qu'un utilisateur ne puisse pas passer à travers des objets.

Lors de ce sprint, deux personnes ont pris en charge le mode multijoueur, une personne s'est occupée de la rétroaction en fin de partie, une autre personne a implémenté la gestion des collisions dans les jeux 3D et la dernière personne s'est chargée de régler des bogues dans l'application. Lorsque la correction du sprint 3 fût disponible, l'équipe a travaillé sur les points faibles de notre logiciel que la correction a révélés.

Approche

Étant donné que nous avions déjà fait 3 sprints, nous étions habiles avec les méthodes de travail précédemment établies, donc nous n'avons rien changé quant à notre manière de travailler en équipe, d'utiliser Trello et d'utiliser Git. Ce sprint était plus facile que les autres sprints puisqu'il n'y avait presque pas de nouveaux développements à faire. L'élément principal de ce sprint était le mode multijoueur et selon l'architecture que nous avions au sprint 3, il restait simplement à jumeler les joueurs pour que ceux-ci puissent jouer ensemble. Lors de ce sprint, la quasi-totalité des tâches était terminée lors de la deuxième semaine de développement, donc l'équipe a mis plus d'effort sur la correction de bogues et sur l'assurance qualité de notre application. L'équipe a aussi mis des efforts sur l'interface utilisateur de l'application, car c'est à ce sprint-ci qu'elle sera corrigée.

Résultat

Le sprint 4 fut le sprint qui a demandé le moins d'effort de développement de la part de l'équipe. Tel qu'on le voit dans les figures 3, 6 et 9 des annexes, c'est le sprint où il y a eu le moins de *commits*, le moins d'insertions de lignes et de suppressions de lignes. Cela s'explique grâce à l'architecture de l'application qui a permis de facilement intégrer le mode multijoueur. C'est donc le sprint qui a nécessité le moins d'heures afin qu'il soit complété (figure 18). Puisque le développement s'est terminé plusieurs jours avant la remise finale, nous avons pu prendre plus de temps consacré à l'assurance qualité et à la recherche de bogues en naviguant dans l'application. Si l'on regarde les graphiques de moyenne d'insertions et de suppressions (figure 10 et 12), on remarque que la taille des *commits* est restée relativement semblable à celle du sprint précédent.

Bref, grâce aux efforts que l'équipe a mis sur l'ensemble des sprints précédents, le sprint 4 s'est déroulé sans accroche tout en conservant les méthodes de travail que nous avons accumulées depuis le tout début du projet.

Total et tendances

Au départ, nous étions très peu à l'aise avec les différentes technologies du cours. En conséquence, nos efforts étaient plus portés sur le développement et la recherche que sur la gestion de version et la prise de métriques. Au fur et à mesure que nous nous sommes adaptés aux outils du cours, on peut observer que nous avons mis plus d'effort dans l'assurance qualité. Ainsi, nous faisions des plus petits *commits* et ces derniers étaient plus précis (figures 10 et 12). Par exemple, nous nous sommes habitués à mettre des tags de développement et à signer tous nos *commits*, ce qui nous a permis d'obtenir des métriques. De plus, nous avons été de plus en plus rigoureux dans nos tests. Nous avons d'ailleurs été récompensés académiquement grâce à ces améliorations. En effet, nos notes ont graduellement augmenté du sprint 1 au sprint 3 et nous espérons que cela continuera pour le sprint 4. Lors des sprints, nous avions tendance à faire la majorité du travail lors de la dernière semaine comme présenté dans les graphiques du nombre d'heures mises sur le projet (figure 18). Cette habitude est restée assez constante pour tous les sprints. Cette méthode de travail a bien fonctionné au final, mais n'est pas la meilleure, car si nous avions rencontré des obstacles majeurs, nous n'aurions peut-être pas pu finir les sprints à temps. Comme amélioration pour de futur hypothétique sprint, il serait avantageux de mieux répartir notre temps ce qui n'était pas souvent possible avec la charge de travail des autres cours. Également, lors d'un nouveau projet, nous devrions nous attarder davantage sur les normes de code et de *commits*. Ceci nous ferait perdre moins de temps et nous permettrait d'analyser, dans le cas des balises de *commits*, comment nous pourrions changer notre temps mis sur les différents aspects comme le développement et les tests. Un échéancier plus efficace pourrait par la suite être mis en place.

Il est clair en analysant la figure 18 des annexes que le sprint 3 fut le plus difficile, probablement à cause de la nouveauté des technologies comme THREE.JS et la charge de travail. À l'inverse, le sprint 4 fut le plus facile, car l'architecture de certains services avait déjà été créée de façon à permettre d'implémenter le mode multijoueur avec facilité. Cette façon de travailler en ajustant notre architecture pour les sprints suivants n'est pas une stratégie avec laquelle nous avions commencé et est venue au fur et à mesure que le projet avançait. Lors de futur projet, il serait donc judicieux au début de chaque sprint d'avoir une discussion d'équipe sur l'architecture des nouveaux services et components à implémenter afin de ne pas trop refaire du code existant.

Conclusion

Finalement, nous pouvons définitivement constater l'importance de garder une bonne discipline dans les processus de gestion de développement. En effet, de sprint en sprint, nous avons notamment tenté d'être de plus en plus rigoureux face aux tests, à la grosseur des *commits* et à l'importance du respect des normes d'assurance qualité de manière générale. On peut donc voir une belle évolution entre le sprint 1, dans lequel nous avons visiblement manqué de rigueur sous plusieurs angles et le sprint 3 dans lequel nous obtenons un beau résultat. Ce n'est pas sans mettre de temps que nous avons atteint nos objectifs, car on remarque que notre sprint 3, le mieux réussi, a également été notre sprint le plus demandant en termes d'heures. Heureusement, tous nos efforts ont été récompensés. Ces efforts sont d'ailleurs illustrés par l'évolution des chiffres des graphiques comparant la grosseur des *commits* et le nombre d'insertions/suppressions. De plus, bien que nous manquions de métriques à ce niveau, nous avons bel et bien ressenti que l'assurance qualité est devenue pour nous un facteur d'importance capitale. Toutefois, on remarque dans la majorité des sprints que nous avions tendance à passer énormément d'heures sur notre projet dans les derniers jours précédents la remise. Dans notre cas, nous avons été chanceux de ne pas faire face à d'énormes imprévus qui auraient pu causer des problèmes majeurs trop peu de temps avant la remise. Cependant, il aurait été plus judicieux de s'assurer d'être prêts au moins deux jours avant la remise, afin de prévoir du temps pour résoudre un imprévu.

Annexes

Diagrammes concernant le nombre de *commits*

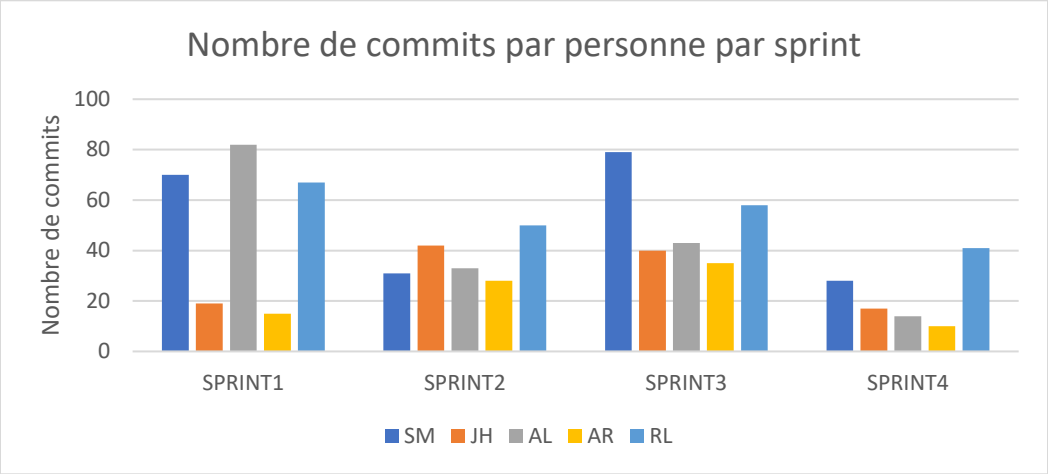

Figure 1 Nombre de commits par personne par sprint

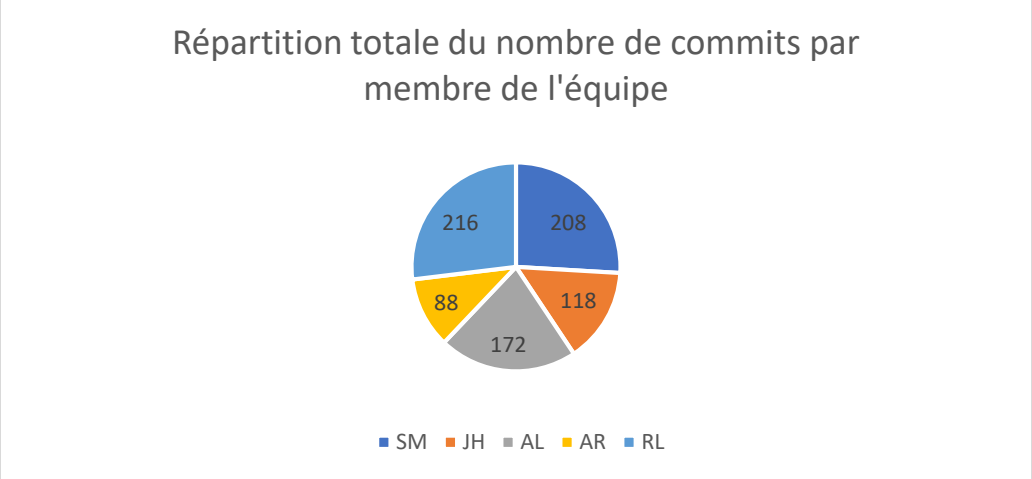

Figure 2 Répartition totale du nombre de commits par membre de l'équipe

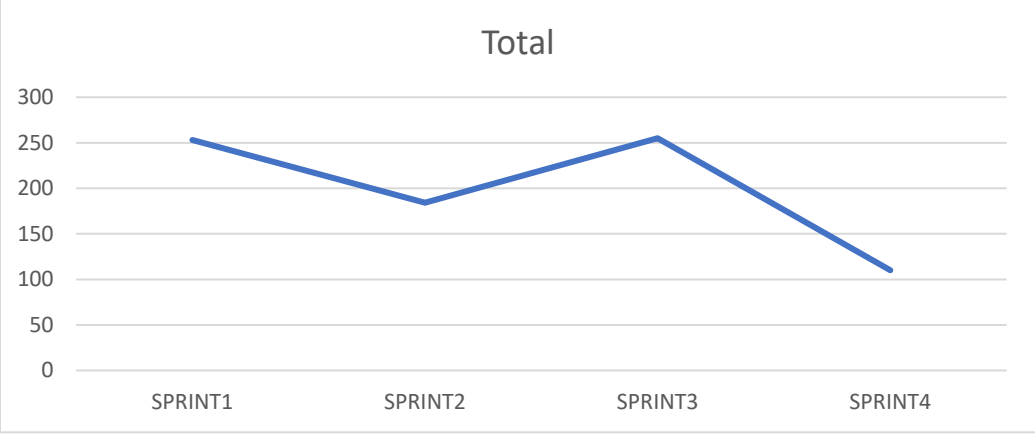

Figure 3 Total du nombre de commits par sprint

Diagrammes concernant le nombre d'insertions

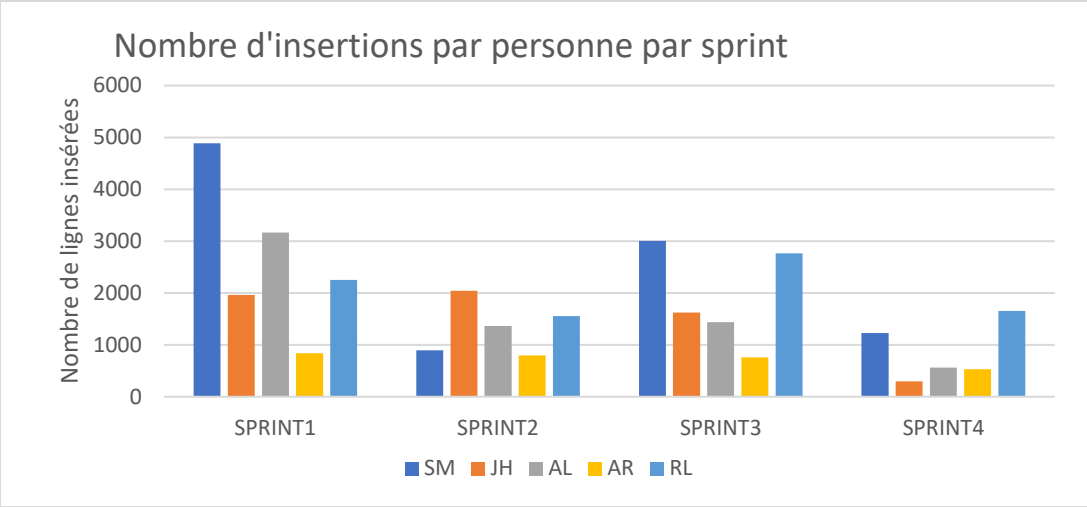

Figure 4 Nombre d'insertions par personne par sprint

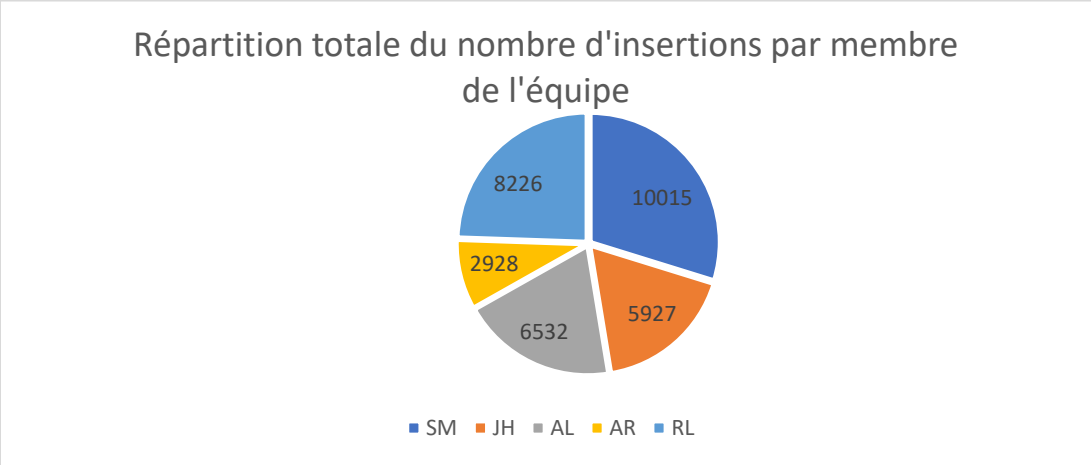

Figure 5 Répartition totale du nombre d'insertions par membre de l'équipe

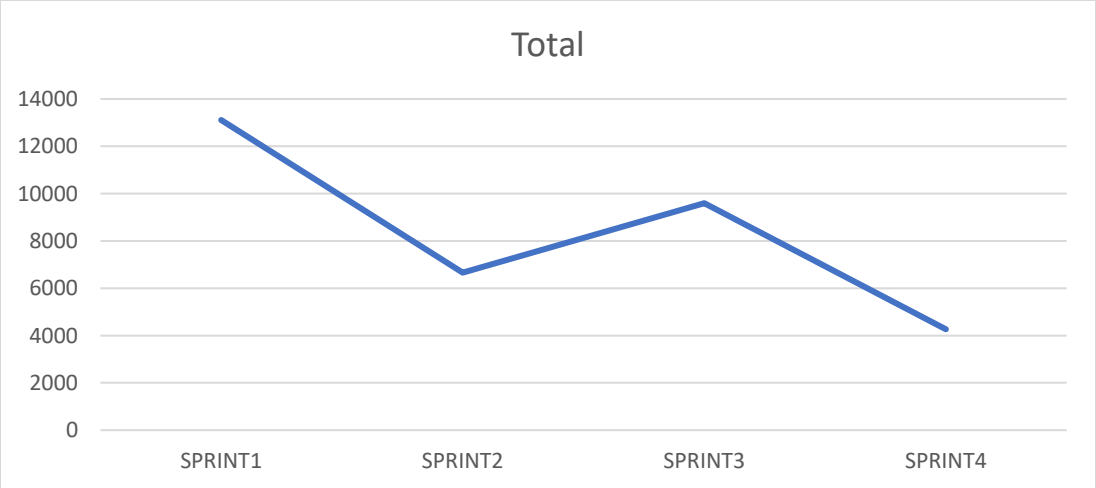

Figure 6 Total du nombre d'insertions par sprint

Diagrammes concernant le nombre de suppressions

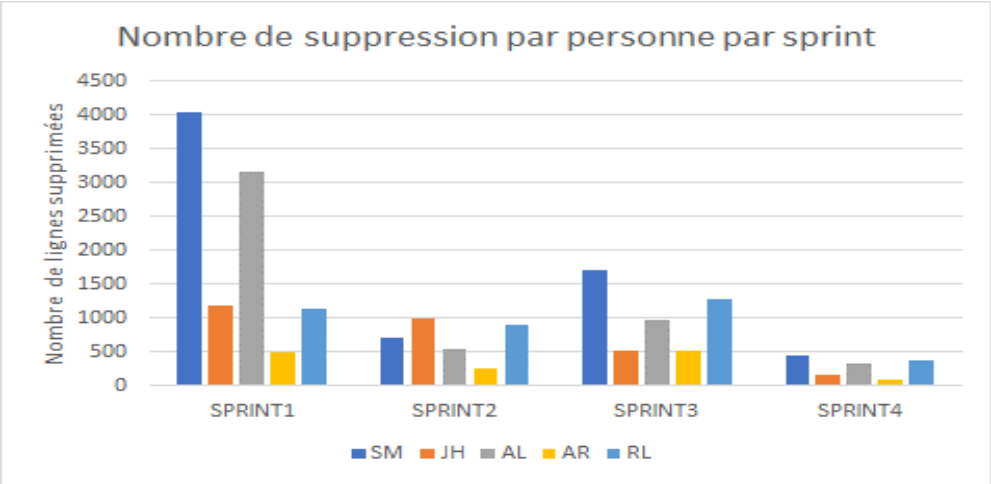

Figure 7 Nombre de suppression par personne par sprint

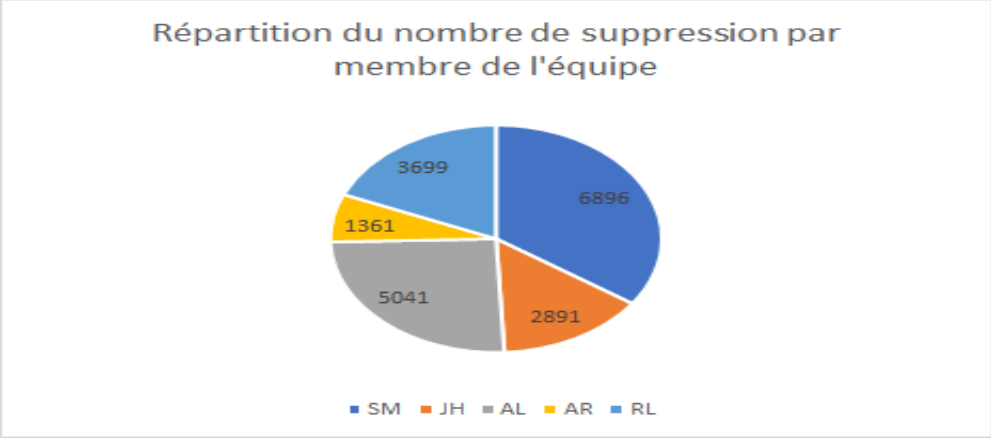

Figure 8 Répartition du nombre de suppressions par membre de l'équipe

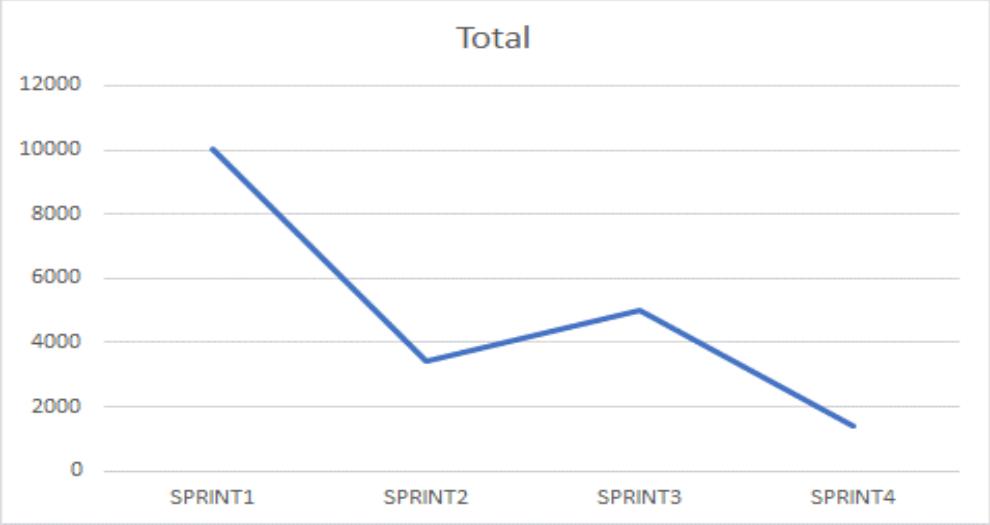

Figure 9 Total du nombre de suppression par sprint

Diagramme concernant la moyenne d’insertions

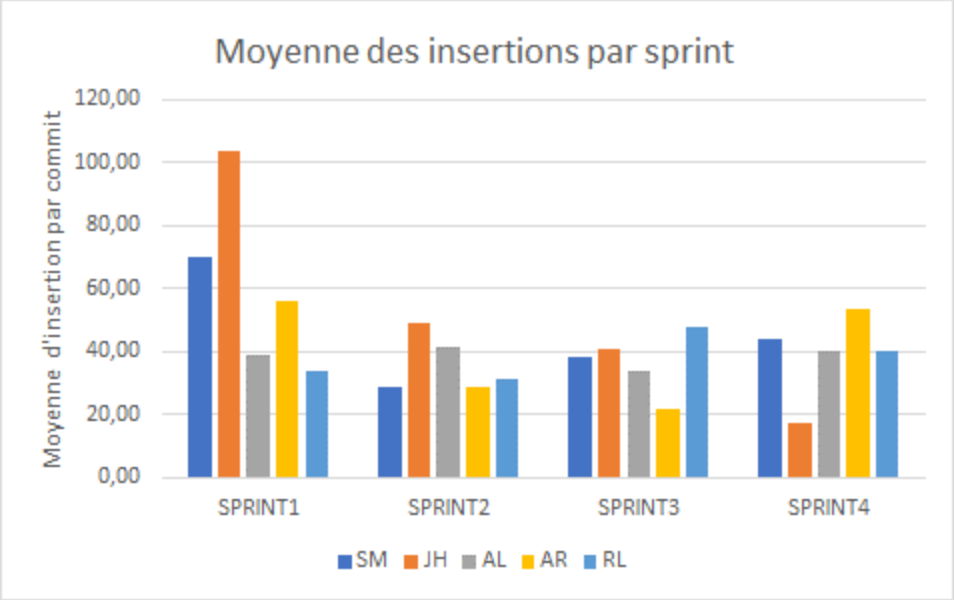

Figure 10 Moyenne des insertions par sprint

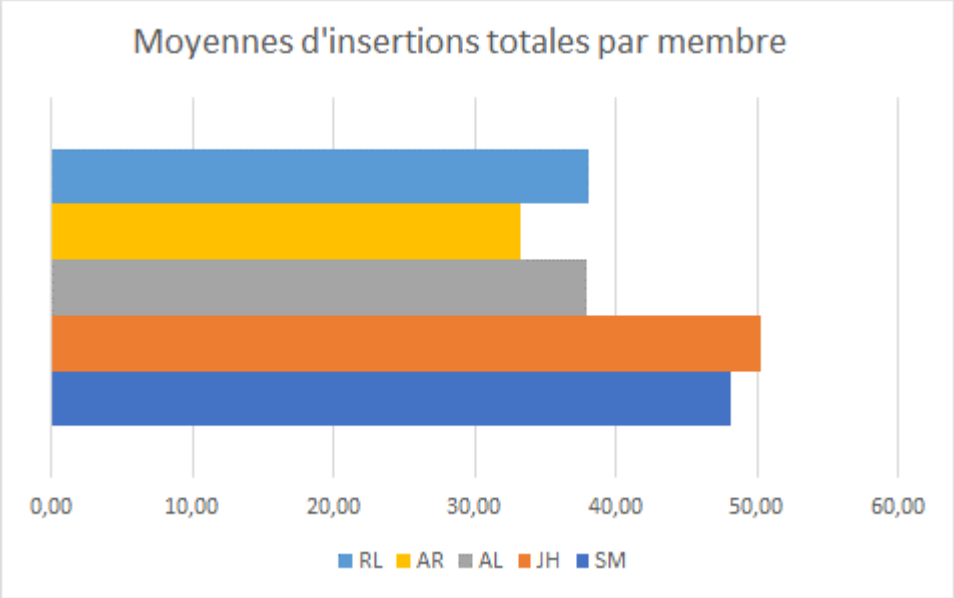

Figure 11 Moyenne d'insertions totales par membre

Diagramme concernant la moyenne de suppressions

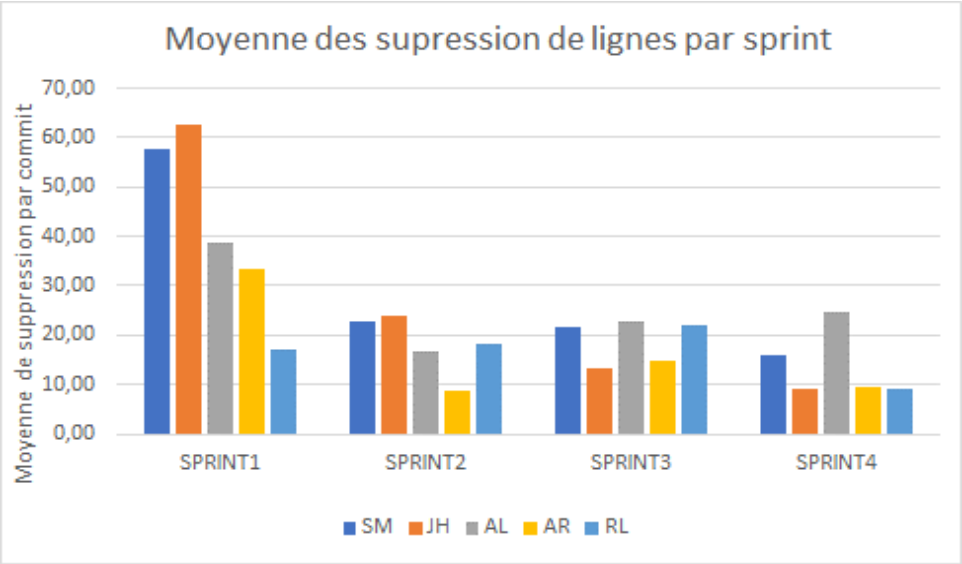

Figure 12 Moyenne des suppressions de lignes par sprint

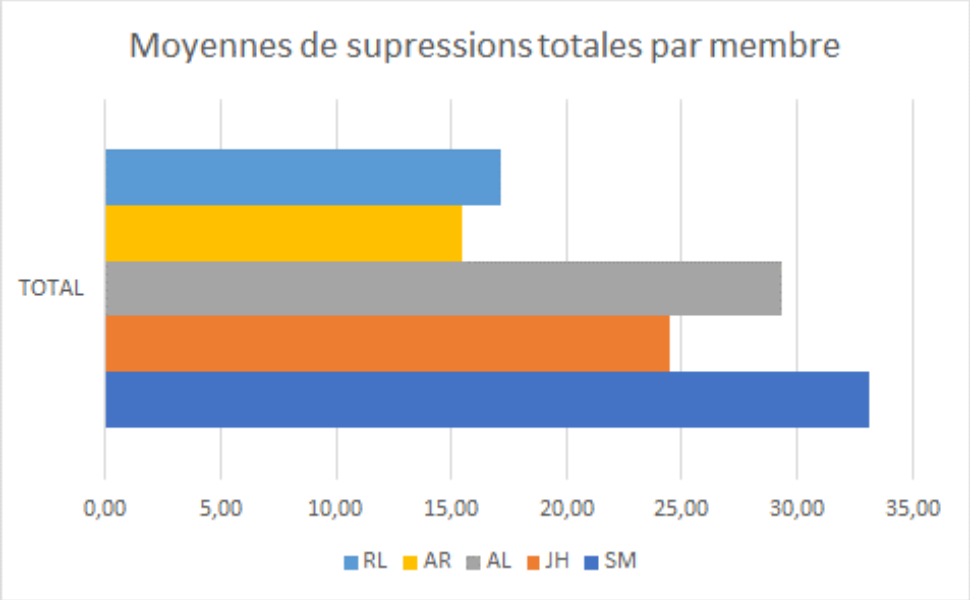

Figure 13 Moyenne de suppressions totales par membre

Diagrammes du nombre d’heures mises sur le projet

Sprint 1

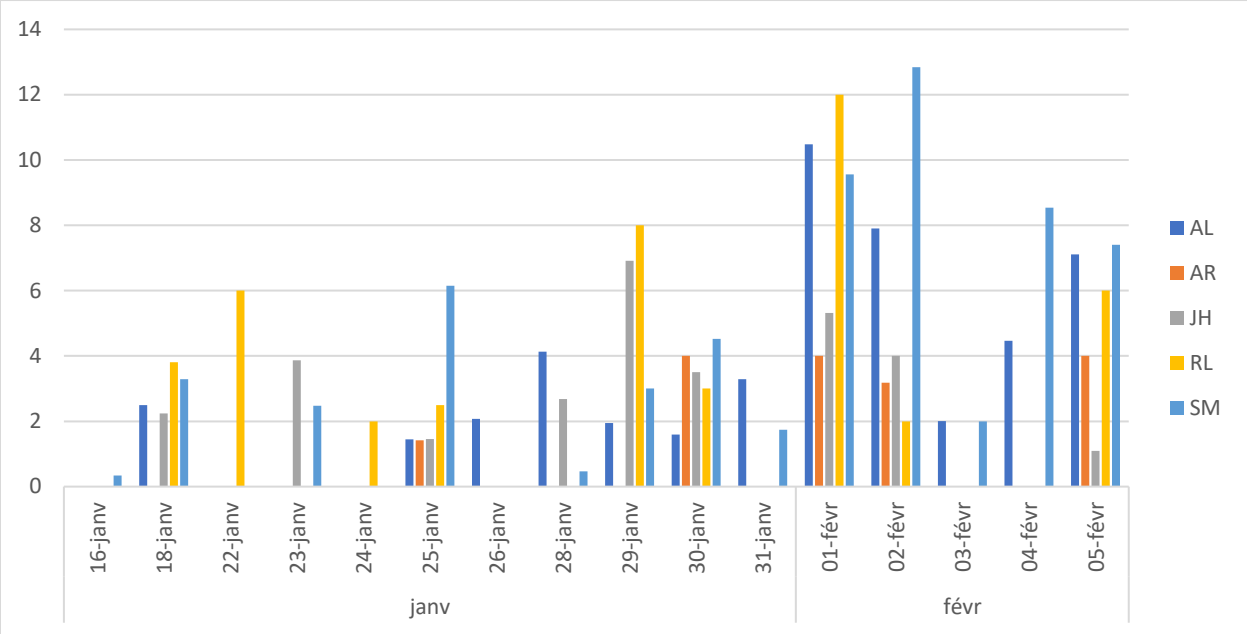

Figure 14 Nombre d'heures par jour par membre pour le sprint 1

Sprint 2

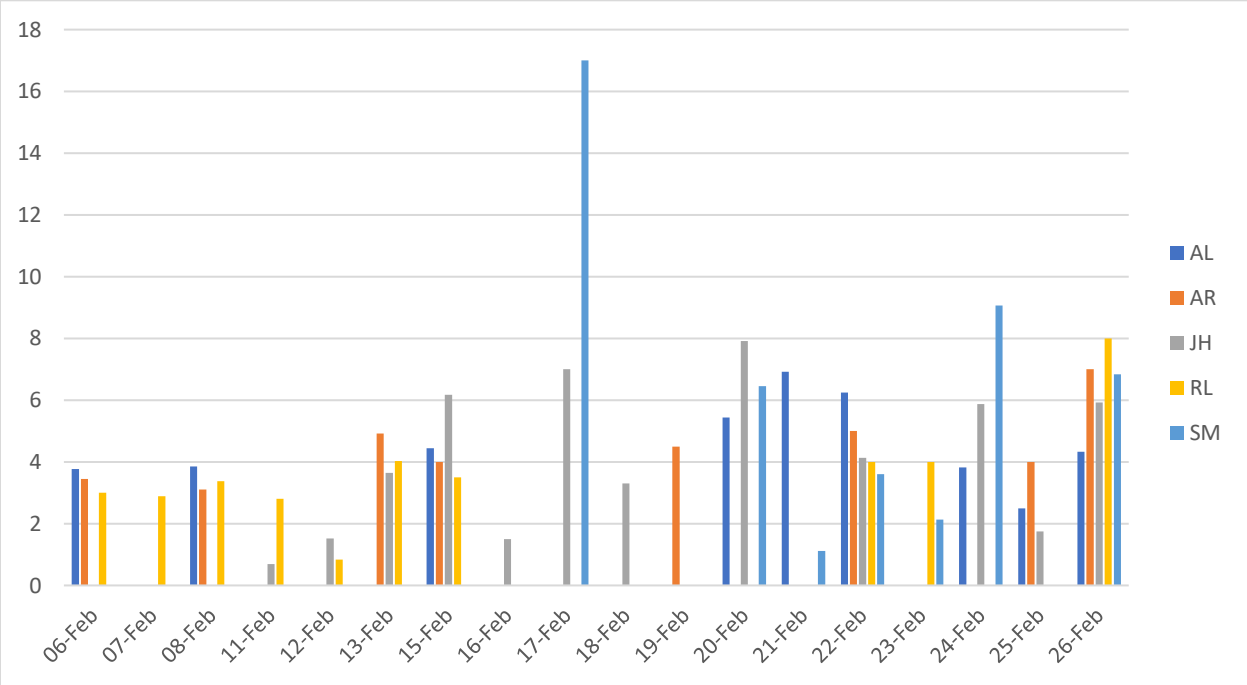

Figure 15 Nombre d'heures par jour par membre pour le sprint 2

Sprint 3

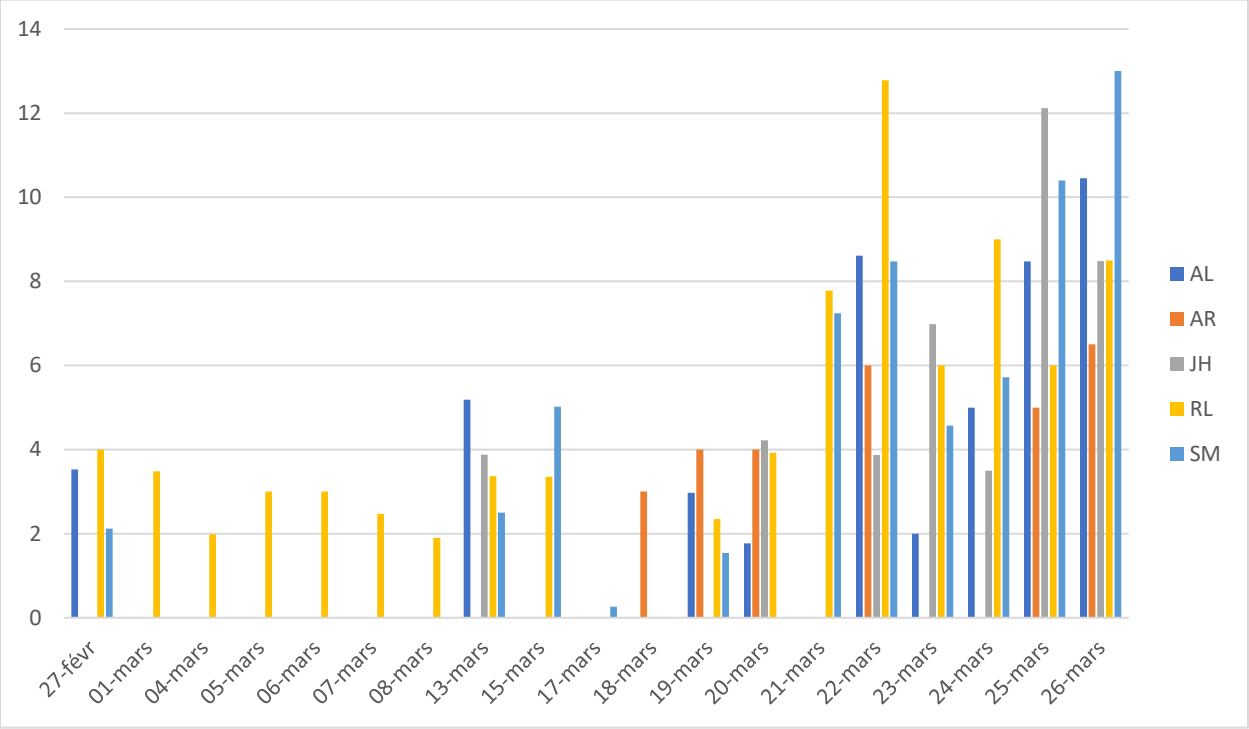

Figure 16 Nombre d'heures par jour par membre pour le sprint 3

Sprint 4

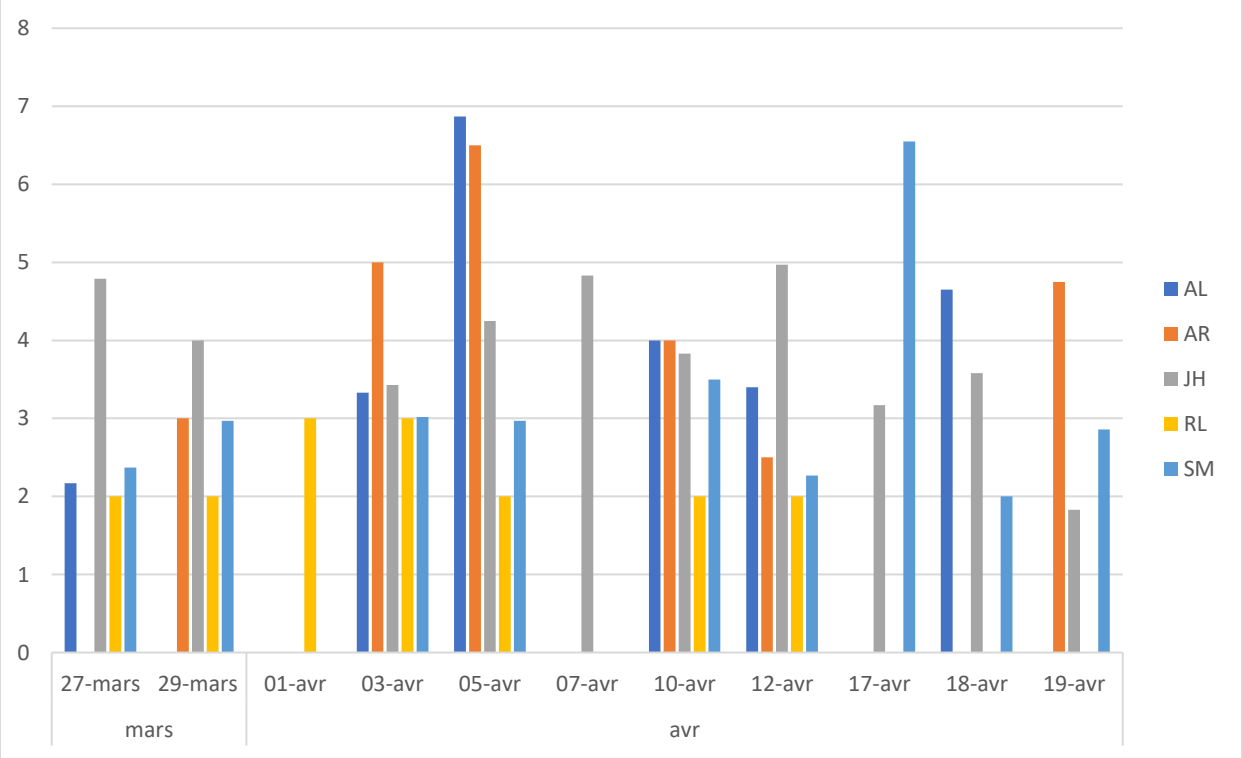

Figure 17 Nombre d'heures par jour par membre pour le sprint 4

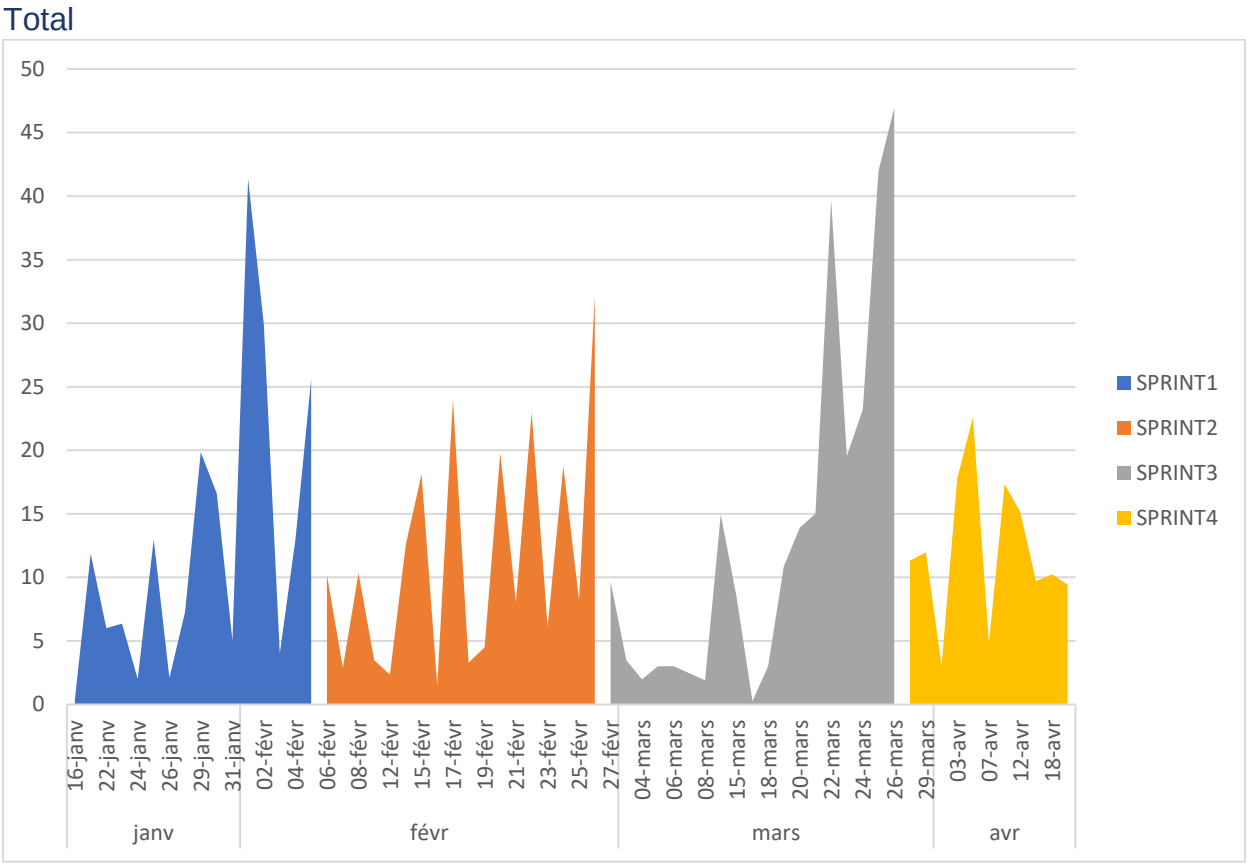

Figure 18 Nombre d'heures total par jour