LOG2900 - Projet intégrateur 2

Hiver 2019

Rapport de gestion

Groupe 108

1896107 Jeremy Boulet

1848238 Tristan Cromer

1895657 Arthur Garnier

1878502 Duc-Thien Nguyen

1904804 Michael Sauget

1893058 Gabriel Houle-Violette

Soumis à : Michel Gagnon

Dimanche 19 avril 2019

Table des matières

Introduction

Dans le cadre du cours de LOG2900, nous avons eu pour mandat de bâtir une application web en MEAN stack. Il s'agit d'un projet en équipe de 6 qui nous introduit à la démarche AGILE et au développement en TDD. Il est divisé en quatre gros sprints et fait intervenir plusieurs technologies très prisées en informatique présentement. Nous avons eu à travailler avec Angular, Express, Node et ThreeJS pour construire un jeu de 7 différences pouvant être joué autant en mode solo qu'en mode multijoueur, en 2D comme en 3D.

Afin d'analyser notre démarche et notre progrès, nous avons récolté des données tout au long du projet. Ces données recensent le nombre d'heures travaillées par jour pour chacun des membres, les tâches couvertes lors de ces journées et une multitude d'informations sur les commits que nous avons effectués sur le projet. Grâce au script en python de Michel Gagnon, nous avons accès à un rapport sur les tâches que chaque commit représentait. Ces tâches sont représentées par des lettres suivant nos initiales dans chaque message de commit: *D* (développement), *T* (tests), *B* (*bugfix*), *R* (*refactor*/remaniement) et A (autre).

Dans ce rapport, nous aborderons les données que nous avons récoltées et essayerons de tirer des conclusions quant à notre évolution au cours de la session. Nous discuterons aussi de notre utilisation des outils GIT, Trello et Slack. Nous nous pencherons aussi sur la façon dont nous avons réussi ou non à incorporer la méthode AGILE et le TDD (*Test-Driven Development*) à notre développement. Finalement, nous détaillerons la manière dont nous avons géré le côté humain d'un projet d'envergure comme celui-ci en utilisant les outils acquis dans le cours de HPR.

Métriques

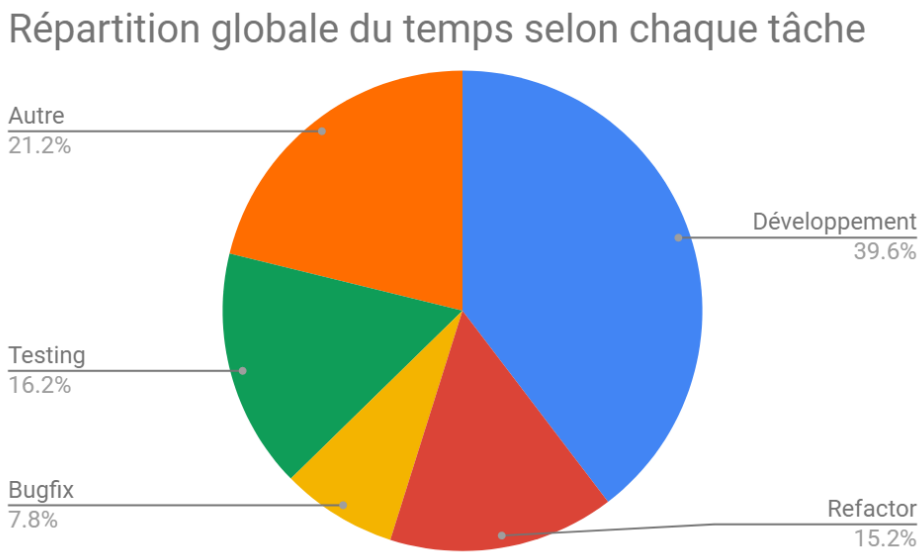
Métriques générales

Les données ont été recueillies sur une table Excel partagée sur un Google Drive. Chaque membre de l'équipe avait pour responsabilité de rentrer leurs heures travaillées sur le projet en fonction de la tâche et ce de façon quotidienne.

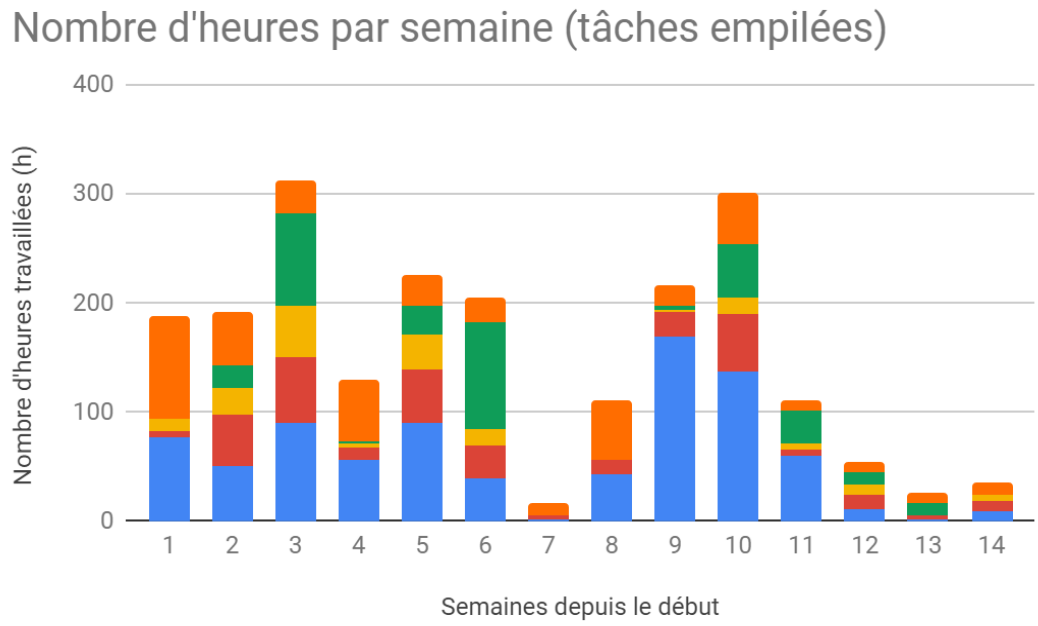
	SPRINT 1			SPRINT 2			SPRINT 3				SPRINT 4				Total
SEMAINE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
Développement	76	50	91	55	91	40	2	44	170	138	60	12	2	10	**829**
Refactor	7	48	59	13	49	29	4	12	22	53	6	13	4	10	**319**
Bugfix	10	25	49	3	31	16	0	1	2	15	5	9	0	6	**164**
Testing	1	20	85	3	27	97	0	0	4	49	30	11	11	0	**338**
Autre	95	50	29	56	28	23	11	54	20	46	11	10	10	11	**443**
TOTAL	**189**	**192**	**312**	**129**	**225**	**205**	**17**	**111**	**217**	**301**	**112**	**55**	**27**	**36**	**2092**

Tableau 1: Heures par semaine selon les tâches

Graphique 1: ensemble des graphiques regroupant la répartition des tâches à chaque semaine en termes de temps



Graphique 2: Répartition globale du temps selon chaque tâche

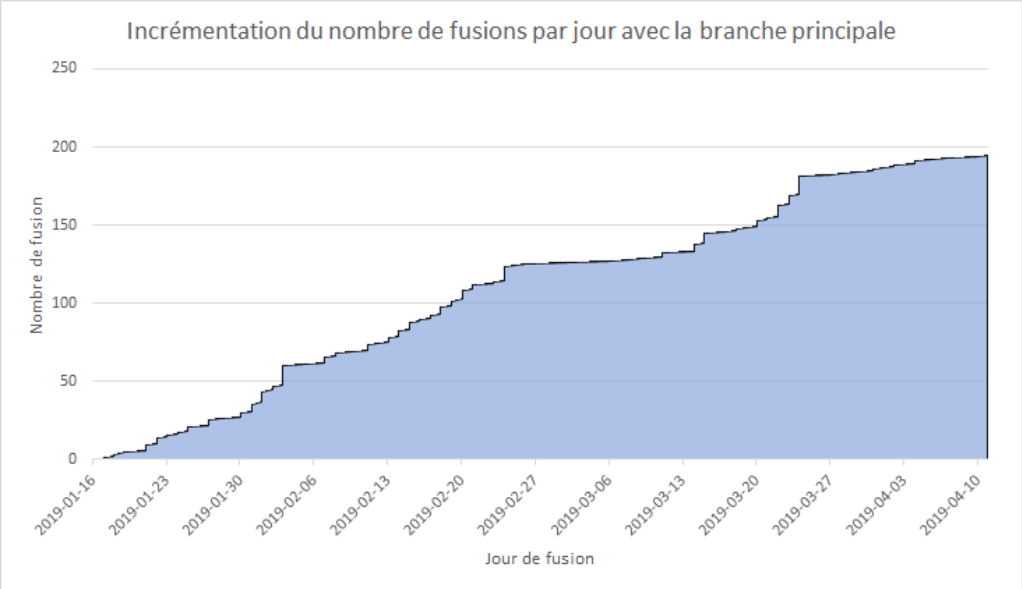


Graphique 3: Nombre d'heures par semaine selon chaque tâche

Globalement, en terme de nombres d'heures, c'est le développement qui a été le plus coûteux. Le graphique 2 présente que presque 40% du temps total y a été consacré contre 16% pour les tests. Le cours visant le TDD, on aurait dû observer une certaine linéarité entre ces deux tâches. En revanche, par le graphique 3, on remarque facilement que le temps passé sur les tests est principalement concentré durant les fins de sprints. C'est quelque chose que nous avons pu remarquer assez rapidement, toutefois cela n'a pas été réellement corrigé avant le dernier sprint. L'agilité est cependant restée au premier plan, car le temps de développement est plutôt constant tout au long de la durée du projet à toutes les semaines. Nous avons ainsi réussi à développer des fonctionnalités régulièrement. L'apport des scrums quotidiens nous a permis de rapidement cibler les problèmes et de redistribuer les efforts afin d'optimiser notre temps. Notre manquement au TDD est donc l'élément sur lequel l'équipe aurait le plus à retravailler.

De plus, avec l'intégration continue et les revues de code avant chaque fusion sur la branche principale, l'équipe a été en mesure d'enchaîner les fonctionnalités tout en gardant du code de qualité, mais aussi permettre à tous les membres d'avoir une idée générale de l'implémentation de chaque fonctionnalité. C'est ce qui a fait notre force et ce dont on est le plus fier.

Métriques sur les fonctionnalités



Les fonctionnalités de l'application sont définies dans un document duquel les différentes cartes du Trello les subdivisent en plus petites tâches au travers des quatre *sprints*. Le graphique ci-haut présente l'évolution du nombre de fusions à la branche principale à chaque jour depuis le début du projet. On remarque immédiatement certaines tendances.

Tout d'abord, on observe rapidement de fortes hausses dans le nombre de fusions lors des fins de *sprints*. Les fonctionnalités étaient implémentées graduellement au cours du *sprint*, mais les tests ont trop été souvent relégués à la fin pour les premières semaines. Ainsi, les pics de fusions sont principalement dus à l'ajout de tests supplémentaires et de corrections de bogues de dernière minute.

Ensuite, le 14 et 15 mars voient d'importants ajouts de fusions. Cela est expliqué par un remaniement majeur du code pour gérer le déroulement des parties de jeu. En préparation du mode multijoueur, l'équipe a décidé de retravailler certaines classes afin de facilement intégrer le mode multijoueur et d'éviter de la duplication de code majeure.

Finalement, on remarque quelques plateaux, en particulier en début de *sprints*. Pour le premier, du 16 janvier au 20, c'est dû à l'apprentissage des technologies comme Angular et ses composantes. Cela s'est aussi fait remarquer en début de *sprint* 2 avec l'intégration de ThreeJS. Les plateaux en fin de *sprint* sont aussi expliqués par le temps à effectuer des rétrospectives à la suite d'une remise, planifier la prochaine et concevoir l'architecture des classes pour les nouvelles fonctionnalités à développer. Deux plateaux sont plus prononcés: un commençant vers le 27 février et un autre le 27 mars. Le premier est expliqué par le période des examens intra et une semaine de relâche dans laquelle le focus a été la conception pour les fonctionnalités à la fois du *sprint* 3 et 4. Ainsi peu de code a été écrit durant ce temps. Le deuxième plateau coïncide avec le *sprint* 4. La majorité des fonctionnalités, en plus de leurs tests, étaient déjà implémentées, car nous avions mis l'effort nécessaire au *sprint* 3 pour garder un code générique, particulièrement pour l'intégration du mode multijoueur qui est un point majeur du *sprint* 4. Bref, l'effort placé dans un *sprint* précédent nous a permis de travailler sur le visuel et la robustesse de notre application.

Utilisation de GIT

Dès la première rencontre d'équipe, nous avons décidé d'adopter le système de *pull request* et de CI (*continuous integration*) sur GitHub. Processus laborieux au départ, il nous a fallu nous habituer à cette manière de gérer le code, mais nous nous sommes rapidement rendus compte des bénéfices énormes que cela apportait. Nous étions capables de détecter les erreurs dans le code avant qu'elles ne soient fusionnées à la branche principale. Le processus de révision du code par les pairs nous a permis de développer des normes de codage strictes qui fait en sorte qu'un programmeur extérieur ne pourrait distinguer le code écrit par l'un ou l'autre des membres de notre équipe.

Nous avons développé les fonctionnalités sur des branches distinctes pour les fusionner une à une sur la branche principale, ce qui nous a permis de paralléliser le développement. Vers la moitié de la session, nous avons reçu un script de la part du professeur nous permettant de récupérer l'information sur nos commits pour en faire des statistiques. Nous nous sommes aperçus à ce moment que notre moyenne de lignes par commit était beaucoup trop haute. Dans les semaines qui ont suivi, nous nous sommes donnés comme objectif de baisser ces chiffres et nous avons réussi à le faire. Voici donc en quelques tableaux les données recueillies à ce sujet :

Tableau 2: Statistiques globales sur nos habitudes de commits par sprints

Auteur	Nb de commits	Lignes insérées	Lignes supprimées	Moy. de lignes ajoutées	Moy. de lignes supprimées
Michael	1249	11043	7211	8,84	5,77
Arthur	502	6608	3363	13,16	6,70
Duc-Thien	894	19980	11202	22,35	12,53
Gabriel	660	5813	3616	8,81	5,48
Jeremy	957	12416	7576	12,97	7,92
Tristan	341	3859	2092	11,32	6,13
TOTAL	4603	59719	35060	12,90	7,42

Tableau 3: Statistiques sur nos habitudes de commits par sprints

Auteur	Nb de commits	Lignes insérées	Lignes supprimées	Moy. de lignes ajoutées	Moy. de lignes supprimées
Sprint 1					
Michael	69	1472	565	21,33	8,19
Arthur	38	1129	669	29,71	17,61
Duc-Thien	141	3789	2510	26,87	17,80
Gabriel	114	1365	943	11,97	8,27
Jeremy	86	3539	2483	41,15	28,87
Tristan	39	871	679	22,33	17,41
Sprint 2					
Michael	284	3984	2374	14,03	8,36
Arthur	59	888	518	15,05	8,78
Duc-Thien	195	5367	2911	27,52	14,93
Gabriel	132	1805	1183	13,67	8,96
Jeremy	219	2300	1489	10,50	6,80
Tristan	110	1279	580	11,63	5,27
Sprint 3					
Michael	738	4325	3099	5,86	4,20
Arthur	268	2907	1287	10,85	4,80
Duc-Thien	174	7649	2842	43,96	16,33
Gabriel	362	2461	1352	6,80	3,73
Jeremy	501	4978	2948	9,94	5,88
Tristan	81	980	275	12,10	3,40
Sprint 4					
Michael	163	1331	1207	8,17	7,40
Arthur	177	2281	945	12,89	5,34
Duc-Thien	392	3768	3125	9,61	7,97
Gabriel	82	268	188	3,27	2,29
Jeremy	176	2162	1299	12,28	7,38
Tristan	122	811	587	6,65	4,81

Nous avons donc un total de 24 659 lignes de code ajoutées au cadriciel initial, étalées sur 4603 commits. Notre moyenne par commit est de 12,90 lignes ajoutées et 7,42 retirées. Les commit plus petits ont été beaucoup plus faciles à faire lorsque nous avons découvert les outils comme *GitKraken* et *GitDesktop,* qui aident à effectuer les tâches de base de GIT grâce à une interface graphique ergonomique.

Nous aimerions souligner que ces statistiques ne sont pas représentatives du niveau d'implication des membres de l'équipe. Nous avons beaucoup travaillé en *pair programming* et n'avons pas été assez assidus à écrire les initiales des deux membres de la paire dans les messages de commit. Nous pouvons cependant constater que la moyenne de lignes par commit diminue drastiquement entre le premier et dernier sprint. Nous pouvons aussi remarquer que tous nos membres ont une moyenne de lignes supprimées très élevée. Nous faisons beaucoup de travail sur l'esthétique du code et sommes très souvent en train de le retravailler pour le rendre conforme à nos normes de programmation.

Gestion d'équipe

Méthodologie de travail

Un des objectifs principaux du cours est de permettre aux étudiants d'explorer la méthode AGILE qui est très répandue en entreprise. Elle place une emphase sur un développement rapide et réactif selon les requis du client. Afin de coordonner chaque membre de l'équipe face aux tâches à accomplir à chaque *sprint*, nous avons tenté d'établir une méthode de travail dès le premier *sprint*. Ce projet étant celui de plus grande envergure depuis le début de notre cursus, il nous a fallu quelques itérations avant d'atteindre une réelle agilité. Effectivement, à notre premier *sprint*, nous avions simplement assigné les différentes cartes du Trello à des pairs d'étudiants. Comme la courbe d'apprentissage en début de projet était plutôt à-pic, un temps important fut passé à apprivoiser les nouvelles technologies et ce n'est donc qu'à l'approche de la première remise que nous avons réalisé la rigidité de notre méthode de collaboration.

C'est lors du post mortem le lendemain de la remise qu'en équipe nous avons ajusté notre approche. À la suite d'une remise légèrement trop serrée dans les temps, nous avons ciblé les sources de difficultés, puis nous avons optimisé notre méthodologie. Les problèmes principaux étaient la gestion du temps et l'organisation des réunions d'équipe. Ainsi afin de corriger cela, dans l'esprit du *sprint*, nous avons établi que nous devions faire un scrum quotidien de 10 à 15 minutes afin que chaque membre de l'équipe puisse expliquer ce qu'il a accompli depuis la dernière rencontre, identifier ses bloquants, puis la prochaine étape. Par la suite, les bloquants sont adressés par les membres qui peuvent avoir eu une expérience similaire ou une rencontre de conception peut être tenue afin d'effectuer un remaniement de code. Nous avons également désigné la première heure de chaque période de projet à l'horaire comme une heure de scrum allongée dans laquelle nous révisions le Trello et nous attribuions les prochaines tâches.

En plus de la méthode agile, le cours place une forte emphase sur le développement dirigé par les tests (TDD). Idéalement, les tests devraient être écrits en premier, mais même après les quatre *sprints* et maintes tentatives, c'est un défi que nous n'avons su relever. Cela était d'autant plus difficile puisque notre conception visait à être le plus générique possible. De ce fait, nous devions avoir tests tout aussi génériques, ce qui s'est avéré être un défi d'envergure. Toutefois, nos tests ont tout de même été écrits en même temps ou plus généralement tout juste après l'implémentation des méthodes.

HPR

Dans un projet d'équipe à 6 personnes, il est important de prendre soin de la dynamique de groupe. Nous avons appliqué les acquis de nos cours de HPR et jugeons que cela a été grandement bénéfique à la productivité de l'équipe. Dès les premières semaines, nous avons rempli ensemble un contrat d'équipe, dans lequel nous recensions nos attentes, nos forces et nos faiblesses. La formulation de ce contrat a été un moment clé dans l'évolution de notre

cohésion d'équipe. Certains membres ont pu nous faire part de leurs inquiétudes face à l'ampleur du projet. Nous avons dès lors pu ajuster notre approche pour apaiser certaines de ces inquiétudes. De plus, nous avons pris le temps de sortir durant les premières semaines pour prendre une bière en guise d'activité de cohésion. Nous croyons que cette sortie a joué un rôle important dans la dynamique de l'équipe. Nous étions par la suite plus à l'aise d'échanger et de débattre entre nous.

Après le premier sprint, nous avons pris quelques heures pour faire un post-mortem et dégager tous les bons et moins bons aspects des trois premières semaines. Nous avons formulé quelques nouvelles normes d'équipe, la plupart concernant notre organisation lors des réunions. Nous avons continué de faire ces post-mortem après chaque remise de sprint.

Grâce à la rencontre avec la guide HPR, nous avons aussi donné à chacun des membres des rôles précis en concordance avec leurs forces.

Conclusion

La création d'une application web de 7 différences nous a permis de manipuler de nouvelles technologies et d'apprendre de nouvelles méthodologies de travail, notamment l'architecture MEAN, le *Test Driven Development* et la gestion de projet Agile. Outre les connaissances technologiques acquises, ce projet nous a permis de développer des capacités de gestion d'équipe importantes en milieu professionnel.

Cependant, ce projet ne s'est pas effectué sans difficultés. Les métriques concernant les heures passées sur le projet ainsi que le script d'extraction de statistiques des commits sont de bons indicateurs des différentes phases par lesquelles nous sommes passées. Les deux premiers sprints ont été des périodes intensives d'apprentissage. Nous devions explorer des technologies qui n'avaient jamais été couvertes lors de notre parcours académique. Le troisième sprint nous a permis de consolider les bases de notre application à l'aide une grosse semaine de conception durant la relâche. Finalement, le dernier sprint a été majoritairement du peaufinage durant lequel plus de 80% de fonctionnalités avaient déjà été complétées dès le début du sprint.

En conclusion, nous sommes fiers de notre livrable final et satisfaits de tout ce que nous avons pu apprendre lors de ce projet. Nous nous sentons beaucoup plus préparés au marché du travail en ingénierie logicielle que nous l'étions au début de la session. Cependant, nous jugeons que le nombre de crédits alloués au cours de LOG2990 n'est pas du tout représentatif de la charge de travail qui y est associée. Nous pensons qu'une meilleure communication au sein de l'équipe de chargés aurait été bénéfique à la réussite de tous les étudiants. Nous aurions aussi beaucoup apprécié avoir plus de formation et de soutien par rapport aux technologies qui nous étaient imposées. Nous avons l'impression ceci nous aurait permis de pousser encore plus loin notre apprentissage des technologies web.