


Projet Génie Logiciel

Charte d'équipe

ENSIMAG 2ème année - 2022-2023



Groupe 1 - équipe 3

Jorge Luri Vañó
Nils Depuille
Vianney Vouters
Virgile Henry
Logan Willem

I. Valeurs de l'équipe

Notre équipe repose sur les principes et valeurs suivantes suivant :

- Maintenir une bonne ambiance générale pendant l'entièreté du projet
- Engagement pour l'équipe
- S'entraider et progresser ensemble
- Accepter les forces et faiblesses de chacun
- Obligation de moyen et droit à l'erreur
- Communiquer un maximum sur ses avancées, réussites et difficultés
- Respect des horaires fixés et travail attendu en dehors
- Transparence dans l'organisation du travail

II. Respect de la charte d'équipe

La charte d'équipe ayant été validée et acceptée par tous en début de projet, chacun se doit de respecter la totalité des articles présentés. Toute infraction des articles de la charte ci-présente implique une sanction d'un esprit bon enfant. On mesure le degré d'une infraction par un nombre de points décidés par l'ensemble de l'équipe au moment des faits. Un barème indicatif des degrés d'infraction et des sanctions associées est donnée ci-dessous :

Echelle d'infraction

<u>Arriver entre 5 et 20 minutes en retard :</u>	1 point
<u>Arriver entre 20 et 45 minutes en retard :</u>	2 points
<u>Arriver entre 45 minutes et 3 heures en retard :</u>	3 points
<u>Arriver plus de 3 heures en retard :</u>	5 points
<u>Push un commit sur "develop" ou "master" qui ne compile :</u>	3 points
<u>Push un commit dans "develop" sans documenter :</u>	1 point



Echelle de sanction

Tous les points : un café ou un chocolat offert pour chaque membre

Tous les 3 points : un petit déjeuner offert pour chaque membre

Tous les 5 points : une tournée de bar offerte

Tous les 15 points : Menu Tacos offert à tous les membres

III. Utilisation de Git

1) Organisation des branches

Le Git repose sur l'utilisation de GitFlow et est composé de 3 branches principales :

- master
- develop
- features

Master comporte les versions livrées du projet tandis que develop est utilisée pour ajouter les fonctionnalités en cours de sprint pour le rendu suivant. La branche feature est utilisée pour développer les fonctionnalités.

2) Branches features

Chaque sous-branch de "features" correspond à une fonctionnalité précise et suit la nomenclature suivante : etape-<lettre etape>-<nom feature>

Chaque branche feature est propre à un membre qui en est le responsable. Il est nécessaire de demander l'accord du responsable d'une branche "feature" avant de la modifier.

3) Develop et Master

Des changements apportés à "master" ne peuvent survenir que depuis "develop". Les fonctionnalités apportées sur "master" doivent être complètes et sans erreur. Les modifications sur "master" n'arrivent qu'à la fin d'un sprint ou avant un suivi.

Tout commit dans "develop" et "master" doit pouvoir compiler. Les nouveaux commits doivent, dans la mesure du possible, ne pas perturber les fonctionnalités développées précédemment. Les tests nécessaires doivent être validés au préalable. Une fonctionnalité sur "develop" ou "master" perturbée suite à un commit doit immédiatement être corrigée.

4) Description des commits

Tout commit doit être accompagné d'une description exhaustive des modifications réalisées.

5) Tests fonctionnels avant commit dans master

Avant de réaliser un commit dans master, la totalité des membres devront réaliser une banque de tests permettant de vérifier le fonctionnement parfait des fonctionnalités implémentées et devront tous valider le commit avant de le réaliser.

6) Sauvegardes

À la fin de chaque journée de travail, une sauvegarde du Git est réalisée par le Git Master sur un ordinateur séparé et indépendant des machines utilisées pour travailler.

7) Numéro de version

Lorsqu'une feature est achevée, la branche est merge puis supprimée avec GitFlow. Le commit porte alors un tag suivant la nomenclature: Etape <lettre etape> - <nom feature> - <Numéro version>

Une modification majoritaire ou complète des fonctionnalités implémentées augmente le 1^{er} chiffre et remet à 0 les autres chiffres (ex. 1.1 -> 2.0). Un ajout de fonctionnalité ou de fonction augmente le 2^{ème} chiffre de la version et remet à 0 le troisième (ex. 1.0.1 -> 1.1). Une modification mineure (ex: résolution de bug) augmente le 3^{ème} chiffre de la version (ex. 1.0 -> 1.0.1).

I. Organisation de l'équipe

1) Lieu de travail

Chaque jour de la semaine, l'équipe se retrouve à l'Ensimag, la Bibliothèque Universitaire ou une coloc selon ce qui est convenu la veille. Un membre pourra télétravailler ponctuellement, si les autres membres l'acceptent. Par défaut, nous nous retrouvons en présentiel à l'ENSIMAG en E300.

2) Horaires

En semaine, nous travaillons ensemble de 9h à 12h30 et de 14h à 17h30. Les horaires peuvent être modifiés ponctuellement si les membres l'approuvent. En dehors de ces horaires de travail, les membres ont une obligation de moyen. Ils doivent avancer sur leurs tâches attribuées au préalable.

3) Daily meeting

Tous les matins, une réunion d'une quinzaine de minutes doit permettre à chacun de partager et se tenir informer des avancées de chacun. Une répartition du travail pour la journée ou pour les jours suivants est ensuite réalisée. Une liste des points à aborder doit être réalisée au préalable

par le secrétaire général et un compte rendu est dressé durant la réunion pour engager les membres sur les tâches attribuées.

4) Validation de la complétion d'une tâche

Tous les avancements sont validés avant d'être mis sur "develop" avant de passer à une autre tâche. Chaque membre doit tenir informé l'équipe de ses avancées. devront être validés au préalable.

5) Gestion des tensions et conflits

Lors d'un conflit ne concernant que 2 membres, les deux parties impliquées résolvent dans le respect de chacun et de manière informelle le conflit. S'il n'est pas résolu, une personne indépendante jouera le rôle de médiateur informel. Si un désaccord reste, l'équipe en discute et délibère en réunion. L'équipe peut finalement chercher la médiation d'un professeur.

II. Rôles

Les rôles suivants ont été attribués en considérant les envies de chacun ainsi que leurs forces et faiblesses. Les rôles sont définitifs sur la toute la durée du projet.

1) Chef de Projet – Nils Depuille

Le Chef de Projet est chargé d'organiser l'avancement global du projet. Il suit chaque membre individuellement pour s'assurer de leur avancement et apporter des réponses à leurs difficultés. Il doit pouvoir repérer les risques du projet, doit pouvoir repérer les écarts au planning et apporter des correctifs. Il a une connaissance globale du projet et est responsable du bon déroulement des réunions de suivi.

2) Secrétaire Général – Vianney Vouters

Le Secrétaire Général veille à la bonne utilisation des différents outils de communication, d'organisation et de travail (hors Git). Il maintient à jour les informations et est chargé de dresser le planning en accord avec le chef de projet. Il dresse un compte rendu de chaque réunion et veille à la bonne communication.

3) Git Master & Human Ressources – Jorge Luri Vañó

Le Git Master veille à la bonne utilisation de Git et fait en sorte que chaque membre travaille dans de bonnes conditions de ce point de vue. Il est chargé de résoudre les problèmes liés à Git et de mettre en place une stratégie de sauvegarde du code.

4) Test Master – Logan Willem

Le Test Master est chargé de l'exhaustivité et de la maintenance de la base de tests. Il veille à ce que les tests soient suffisants et validés pour chaque fonctionnalité sur "develop" et "master". Il est chargé d'écrire les scripts de tests de façon uniforme et de les ajouter au projet. Il se doit d'adopter une stratégie de test éco-responsable..

5) Code Master – Virgile Henry

Le Code Master est chargé de vérifier que le code est correct, propre et bien formaté. Il vérifie le respect des règles de documentation du code et s'assure de sa cohérence avec le code. Il est aussi responsable de l'architecture globale du logiciel et de son bon fonctionnement. Il peut travailler directement depuis les branches "develop" et "master".

III. Code

1) Langue de programmation

Seul l'anglais sera utilisé dans l'ensemble du code.

2) IDE

Seul Visual Studio Code est autorisé comme IDE.

3) Formatage

Les normes élémentaires de formatage devront être respectées. En particulier, grâce au point IV.2., un simple Ctrl+Shift+I permettra ce formatage adéquat.

4) Javadoc

Chaque méthode se doit d'être documentée de sorte à permettre la génération automatique de la documentation avec Javadoc.

5) Documentation

La documentation doit être réalisée au fur et à mesure de la programmation d'une fonctionnalité. L'objectif est de pouvoir informer directement du développement d'une fonctionnalité mais aussi de pouvoir prendre du recul. Cette documentation doit être claire, concise et tenue à jour. Les parties complexes du code doivent être commentées exhaustivement.

IV. Outils utilisés

Google Drive : Cloud de stockage des documents.

Google Docs : Bureautique et documentation.

Trello : Outil de gestion de projet.

Discord : Outil de télétravail, de communication et de partage de ressources.

Messenger : Moyen de communication informel.

Fork : Gestion de Git pour le Git Master.

Planner : Outils de planning et de création de diagrammes de Gantt.

Visual Studio Code : Éditeur de code imposé par la partie IV.2.

Signatures

Les cinq membres approuvent l'intégralité de la charte et s'engagent à respecter la totalité des articles et pénalités.

Signé à Grenoble le 04/01/2023

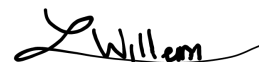
Jorge Luri Vañó



Virgile Henry



Logan Willem



Nils Depuille



Vianney Vouters

