# SCRUM dans le cours projet

Dans ce document, vous trouverez les différents aspects de scrum qui sont couverts dans le cours projet.

# Étapes de création de 'story'

- **Déterminer le nombre d'heures totales par équipe pour chaque sprint :** Cette étape n'est faite qu'une seule fois pour la durée du projet. On détermine le nombre d'heures totales que l'équipe doit fournir pour la durée d'un sprint. Comme les sprints ont tous la même durée (2 semaines), ce calcul s'appliquera à tous les sprints.
    - On calcul le nombre d'heures qu'un étudiant peux donner par semaine : 6h dans le cours projet + 6 heures en dehors du cours. Donc 12 heures
    - On multiplie ce nombre par le nombre d'étudiants dans l'équipe : 5 étudiants * 12 = 60 heures / semaine
    - On multiplie ce nombre par le nombre de semaines dans le sprint : 60 * 2 = 120 heures
    - Ce temps représente le temps total qu'une équipe peut fournir. Mais comme du temps doit être pris en considération pour l'overhead (les cérémoniaux scrums entre autre). On estime à 20% l'overhead d'un sprint ce qui fait donc comme calcul : 120h * 80% = 96 heures. Vos sprints devraient alors tous débuter avec autour de **96 heures**. Ceci servira à alimenter le 'burn down chart' qui pourra nous aviser de l'avancement du projet.

# Planification

- La planification est un cérémonial qui permet de définir le contenu de votre prochain sprint. Lors de la planification, l'équipe se rencontre afin de discuter de la façon que les stories seront développées. On doit s'assurer de couvrir le plus de détail possible sans pour autant prendre trop de temps. Le 'time boxing' est une bonne façon de se limiter lors des discussions sur le contenu des stories.

- **Évaluation de la story en points** : Pourquoi évalue-t 'on les stories en points ? La raison est que l'on veut un mécanisme d'évaluation rapide qui nous permettra de savoir les stories que l'on devra mettre pour notre prochain sprint.
  L'évaluation de la story se fait en fonction de l'effort, la complexité et l'incertitude. On considère 1 point comme étant environ 1 journée/personne. Une journée pour 1 point est une valeur approximative, la raison étant que l'on veut évaluer rapidement nos stories.

    - Cette étape nécessite de réfléchir à toutes les différentes options d'une story. Par exemple, si la story nécessite de remplir un formulaire, on devrait avoir une étape de validation du formulaire.
    - On fait l'évaluation des stories en 1 coup sans faire les tâches. Le but ici est de déterminer le plus rapidement possible quelles stories vont entrer dans le prochain Sprint. Évaluer pour 20, 25 points devrait prendre au plus 30, 40 minutes.
- **Créer les tâches (sub-tasks dans JIRA) :** Une fois que l'on a toutes les stories pour notre prochain sprint, on peut aller à un niveau de détail plus poussée pour chacune des stories. Pour débuter le 1er sprint, on aura habituellement que 2 tâches soit la partie UI client (React) et la partie 'backend' (Spring Boot). Pour ma part, j'appelle habituellement ces 2 tâches (FE, BE). Frontend, Backend. Il est important de mettre les heures sur les tâches. La somme des heures de toutes les tâches dans le sprint devrait correspondre aux heures totales prévues pour l'équipe sur toute la durée du sprint. Ces heures serviront à définir le 'burn down', métrique qui nous donne certains indices sur l'avancement du sprint et de la qualité de nos évaluations.
- **Déterminer les critères de succès :** Les critères de succès déterminent quand une story est terminée. J'utiliserai fréquemment le mot 'DONE' pour une story qui est terminée. Quand on dit qu'une story est terminée, cela veut dire qu'on n'y retouchera plus pour le reste du projet et que la fonctionnalité est prête à être mise en production. Si on doit retoucher à la story parce que l'équipe a oublié de faire une tâche, c'est alors à ce moment que les membres de l'équipe peuvent mettre le 'hat of shame' (chapeau de la honte). Si on doit retoucher une story après la fin du sprint et que c'est dû à un oubli du PO, l'équipe ne sera pas alors à blâmer et qu'il s'agit d'un oubli de la part de la direction. De toute manière, dans les 2 cas on devra alors créer une nouvelle story, qui devra être évaluée à son tour pour un sprint ultérieur.
- **Définition de 'DONE' :** Quand on termine un story, on doit savoir que celle-ci ne sera plus jamais retouchée. En fait, elle est même prête à être déployée en production. On ne peut mettre une story à 'DONE' en se disant, entre autres, que l'on va changer le CSS plus tard ou que l'on va ajouter des options de 'menus' plus tard. Cela impliquerait que l'on devrait retoucher à nos stories ce qui contrevient à notre définition de 'DONE'. Pour se faire, chaque équipe peut avoir une 'checklist' d'items à vérifier avant de mettre vos stories à 'DONE'. Dans le cas de notre cours, je demande que votre définition de 'DONE' soit inscrite au tout début de votre 'README.MD' de votre projet 'github'.

## Configuration des environnements de développement

- Au début d'un nouveau projet on doit envisager la configuration des ordinateurs des membres de l'équipe. On peut faire une story à cet effet et mettre autant de tâches

qu'il y a de membres dans l'équipe. L'idée est que chaque membre de l'équipe a une configuration exactement pareille à tous les membre de l'équipe.

- Un exemple de configuration serait de vérifier que chaque membre peut faire 'git clone' du projet, faire un commit et un 'git push'. Chaque membre doit avoir la bd (ex : postgresql) d'installée et bien configurée pour le projet (user, database, ...)

## Création du Sprint 1

- Quand les stories sont entièrement déterminées (description détaillée, critère de succès bien définis, tâches entrées avec leur évaluation en heure, le sprint peut alors démarrer.
- Un sprint a une durée de 2 semaines et il devrait avoir un but. Chaque sprint a une saveur particulière comme par exemple, les inscriptions et connexions ou comme autre exemple, l'entrée de formulaires, l'impression de document, etc. On essaie de capturer ici l'essentiel du travail qui sera fait dans le sprint avec un 'catch phrase'.
- Une fois le sprint démarré, les membres de l'équipe peuvent alors commencer à s'assigner des tâches et à travailler sur l'avancement du projet.

## Daily Standup Meeting ou mêlée quotidienne.

- Une fois le sprint démarré, on voudra faire des rencontres, idéalement de façon quotidienne, mais au pire de 2 à 3 fois par semaine. Le but de ces rencontres est de donner une idée à l'équipe de l'avancement du sprint.
    - La 1ière chose que l'on fait lors d'un 'Daily' est de regarder le 'burn-down-chart'. De cette façon toute l'équipe est en mesure de savoir si le sprint est entre bonne voie d'être complété ou non.
    - Par la suite, on peut regarder les heures 'loggées' par chaque membre d'équipe. Comme on est dans un cadre académique, la raison de regarder les heures est de donner une idée de l'implication de chaque membre sur le projet. On peut également regarder le nombre de lignes de codes produites dans les 2 technologies utilisées soit java et javascript. Des scripts vous sont donnés pour que vous puissiez voir ces métriques par vous-même.
    - Il faut également regarder les tâches qui sont dans la colonne 'en revue' ou 'in review'. On assigne chacune des tâches à une autre personne qui n'a pas touché à la tâche pour effectuer la revue. On assume que si une tâche a été faite en programmation par paire qu'il n'est alors pas nécessaire d'effectuer la revue de code. Ces revues doivent être faites immédiatement après le 'daily' afin de savoir si on met la tâche à DONE ou pas.
    - Une fois toutes ces étapes faites, on peut procéder à la revue en tant que tel. Chaque membre de l'équipe doit dire les 3 items de la revue :
        - Ce qu'il a fait ou ce qu'il fait présentement

    - Ce qu'il compte faire, ça peut être la même story que le 1er item si celle-ci n'est pas terminée
    - S'il est bloqué. Si un membre de l'équipe est bloqué et qu'il le manifeste, le daily s'arrête jusqu'à ce qu'une personne se porte volontaire pour l'aider immédiatement après la revue.

## Revue

- La revue est une étape importante dans SCRUM. Elle permet de valider qu'une story est belle et bien terminée et que l'on peut la mettre en production.
- Une liste (checklist) permet de vérifier qu'une story est bel et bien terminée. Cette liste s'appelle 'la définition de DONE'.
- Les revues ne sont pas toutes égales. Par exemple, la revue de la tâche 'back end' ne sera pas exactement la même que celle du 'front-end'. Pour le 'back end' on voudra vérifier que les tests fonctionnent (bar verte partout) et que la couverture est au-delà de 80%. Par la suite on peut vérifier les 'endpoints' de votre API à l'aide d'outils comme Postman. On vérifie également que le code est écrit selon les normes de qualité habituelles (clean code).
- La revue du 'frontend' demande un peu plus de travail. Après avoir fait la revue de code, il faut écrire le 'plan de démo'. C'est ce qui sera démontré au 'product owner' lors du cérémonial appelé 'démonstration'. Le plan de démo doit être assez détaillé pour que n'importe qui puisse l'exécuter sans problème.
- Une fois la revue terminée, il faut par la suite effectuer le 'merge' dans la branche 'main'. Une story terminée est prête à être mise en production.

## La démonstration (démo)

- La démo sert, comme son nom le dit, à montrer ce qui a été produit durant le sprint. Une règle importante, on ne montre que les stories qui sont terminées.
- Plusieurs personnes peuvent prendre part à la démonstration, c'est important de bien la préparée et d'être prêt à répondre aux questions.
- Lors de la démonstration, un membre de l'équipe doit prendre les notes afin de savoir quoi augmenter ou corriger selon les commentaires reçus des différents 'stakeholders' présents.
- La démonstration peut être effectuée par un membre ou par plusieurs membres de l'équipe.
- On passe chacune des stories terminées dans l'ordre et on exécute le plan de démo qui est inscrit dans le détail de la story.
- On s'assure de montrer les différentes options si la story en comporte. Par exemple, on peut prendre le temps de montrer la validation s'il y en a.

- Une fois le plan de démo d'une story exécuté, on passe à la prochaine story toujours dans l'ordre.
- La démo est terminée lorsque toutes les stories qui sont au statut de 'DONE' ont été montrées.

## La rétroaction (rétro)

- Le cérémonial de rétroaction permet à l'équipe de revenir sur les différents aspects qui ont affectés l'équipe, autant en bien qu'en mal, et ceci afin de trouver des façons d'améliorer le fonctionnement de l'équipe. En regardant le burndown final du dernier sprint, les heures travaillées, des lignes de code produites, des estimés passés, l'équipe détermine des 'points d'actions' qui, espère-t 'on, pourront aider aux sprints suivants. Les points d'actions sont de type 'SMART' (Spécifique, Mesurable, Atteignable/Réalisable, Timely). À chacune des rétros, on passe à travers tous les points d'actions qui ont été définis dans les sprints précédents et on se pose les 3 questions suivantes :
    - Est-ce que l'équipe a respecté le point d'action ?
    - Si oui, est-ce que cela à aidé ?
    - Est-ce que l'équipe conserve le point d'action

## Conclusion

Ce document est un guide que vous pourrez consulter pour vous assurer d'avoir bien suivi les différentes étapes du développement de logiciel avec SCRUM.