Rapport de l'utilisation des méthodes Agile pour le projet ECOM
M2PGI 2016-2017
Groupe Ebooks

Membres du groupe : Sébastien OCHIER, Jérémy FERNANDES, Jordan BOUCHER, Clément SOULIER, Julien ARMAND

1 - Introduction

Dans le cadre de nos études, nous avons été amenés à réaliser le projet ECOM. Au cours de ce projet, nous avons utilisé des méthodes de développement agile pour faciliter notre travail en groupe. Voici le compte-rendu des méthodes que nous avons employées et notre analyse des résultats qu'elles ont produits.

2 - Présentation du projet

Le projet ECOM est un projet de 2ème année de Master Informatique consistant en la création et le déploiement d'un site de vente en ligne dont les produits sont libres.
Notre groupe a choisi de réaliser un site de vente en ligne de livres électroniques.
Ce projet vise à nous apprendre à utiliser les technologies et méthodes suivantes : Angular, développement d'IHM web, architecture de serveur, base de données, et déploiement cloud.
Mais il vise aussi à nous apprendre à utiliser la méthode agile pour organiser le développement d'un logiciel en groupe.

3 - Organisation du projet

Notre chef de projet était Julien Armand, mais tous les membres du groupe ont pris part au développement du site.

Après avoir réalisé un questionnaire en ligne pour récupérer les attentes des clients potentiels, nous avons réalisé la liste des US pour le projet. Ce questionnaire était basé sur l'analyse des sites de vente en ligne existants, et ces retours nous ont permis d'identifier les fonctionnalités les plus intéressantes pour les utilisateurs.

Lorsque nous avons tous ensemble réalisé les US, chacune d'entre elles a reçu une priorité, une cotation, et des critères d'acceptation plus ou moins bien définis en fonction de l'ambiguïté de l'US. Lors de l'attribution des cotations, si la cotation d'une US était trop élevée (4 ou plus), nous la divisions en plusieurs nouvelles US de taille plus réduite.

Une fois cette phase de conception terminée, nous avons pu passer à la phase de développement. Celle-ci a nécessité que nous nous organisions pour nous répartir les tâches. Pour cela, nous avons utilisé un kanban (https://kanbanflow.com/board/4abe8c66bb0c351c8aefaad2ec49ad54) et défini des sprints. Sur le kanban, les post-its dans la colonne "in progress" étaient marqués du nom des développeurs responsables pour leur complétion.

4 - Durée et contenu des sprints

Après un premier sprint d'installation et de recherche ayant pris environ un mois, notre groupe est parti sur des sprints d'une semaine, dédiant les journées du jeudi et vendredi au projet.

Ces sprints avaient donc besoin d'être équilibrés. Au début du projet, nous avons surestimé la difficulté des US, ce qui a décalé notre planning car les sprints se finissaient plus tôt que prévu.

Dû à notre correction dans l'autre sens, le problème opposé s'est présenté plus tard. De nombreuses tâches se sont révélées plus complexes que prévues, ce qui nous a forcés à adapter notre planning. Ces problèmes étaient dus à des technologies qu'il nous a été difficile de prendre en main (par exemple, l'intégration de la recherche en utilisant l'outil Elasticsearch).

A cause de cela, il a été courant pour des tâches de nécessiter que les développeurs consacre une partie du sprint suivant à leur complétion.

5 - Product Backlog

Lors du développement, nous avons rencontré de nombreux problèmes indépendants des US définies au commencement du projet. Le product backlog ne reflète donc pas bien le travail réalisé par les différents membres du groupe car il ne contient que les fonctionnalités attendues par le client.

De nombreuses US du product backlog n'ont pas été réalisées, parce qu'elles avaient une priorité faible et ont été abandonnées afin que nous puissions nous concentrer sur la livraison du produit à la date souhaitée.

6 - Gestion des sprints

Les sprints que nous avions défini au début du projet étaient organisés de manière trop thématique (exemple : gestion du panier). Il s'est avéré qu'il est plus efficace qu'une seule personne travaille sur ce thème pour qu'elle puisse se spécialiser et maîtriser cet élément du projet.

Quand un développeur avait fini la tâche sur laquelle il travaillait, il prenait une nouvelle tâche dans la colonne "sprint" du kanban et la déplacait dans la colonne « in progress » et la marquait de son nom. Ceci nous a permis de savoir en permanence qui travaillait sur quoi.

A cause de l'ambiguïté des tâches définies au début du projet, nous avons précisé les tâches sur le kanban en les définissant de manière plus technique. Des tâches imprévues sont également apparues au cours du développement, telles que le refactoring du code à plusieurs occasions et l'installation d'un serveur Jenkins pour le déploiement de la dernière version du site en continu.

7 - Démos

Tous les membres du groupe étaient équipés de machines capables de faire tourner le site en localhost. Cela nous a permis de savoir en permanence à quoi ressemblait le site dans sa version actuelle.

En plus du déploiement en local, nous avons régulièrement déployé le site sur le serveur cloud (les machines virtuelles mises à disposition par l'université). L'installation du serveur Jenkins a permis de s'assurer que cet instance du site était toujours la dernière version mise sur le répertoire git que nous utilisions pour le projet.

8 - Rétrospectives

Nous avons fait deux rétrospectives avec le professeur en début de projet. La première visait à nous

apprendre le fonctionnement d'une telle pratique. Elle nous a aussi permis d'identifier des problèmes d'organisation du groupe, tel que le fait que deux membres du groupe ne comprenaient pas l'architecture du site. Des problèmes de communication ont également été découverts, ce qui nous a permis de travailler plus efficacement, bien qu'ils n'aient pas été totalement résolus.
En dehors des rétrospectives prévues avec le professeur, nous n'avons pas fait de vraies rétrospectives. A la place, nous avons fait des réunions régulières pour se tenir informés de l'état du développement.

9 - Analyse des méthodes employées

Le product backlog est un outil utile pour le dialogue avec le client, car il permet de définir de façon claire les attentes de celui-ci. Mais une fois que le développement a commencé, il ne permet pas de suivre de façon précise l'avancement des différents travaux en cours.
C'est le kanban qui permet de suivre cela, et nous l'avons trouvé très utile. Cependant, un système de post-its physique aurait probablement été plus intuitif et fréquemment utilisé.

La courte durée du projet et le fait que nous n'avions que deux jours par semaine lui étant dédiées ont causé les rétrospectives et les stand-ups d'être trop coûteuses en temps pour valoir la peine de les faire. En revanche, dû au fait que nous travaillions tous dans les même salles, chacun était constamment au courant des travaux réalisés par les autres. Dans ces conditions, les rétrospectives ne nous semblaient pas utiles.

Nos problèmes de gestion des sprints étaient probablement dus à un manque de recherche concernant la difficulté de la réalisation technique des US au début du projet. Nous n'avons étudié la réalisation de ces fonctionnalités qu'une fois dans le sprint de leur développement. Pour certaines, cela n'a pas été un problème, mais pour d'autres, comme l'implémentation d'Elasticsearch, ce manque de prévoyance a entraîné des retards conséquents quand l'US s'est révélée plus complexe que ce à quoi nous nous attendions.

Au lieu d'avoir des sprints par thème, nous aurions dû répartir toutes les tâches et fonctionnalités par thème et divisé nos efforts sur chaque sprint entre ces thèmes. Cela nous aurait permis d'avancer plus rapidement et d'être chacun plus expérimenté sur le thème auquel nous aurions été assignés.