Compte-rendu - audit - définition du plan d'action

Agora - équipe jeux

29 mars 2024

Participants

Nom	Prénom	Groupe
Abdellah Godart	Karim	Professeur gestion de projet
Binginot	Etienne	1
Cauchois	Niels	1
Ducroq	Yohann	1
Khabouri	Izana	1
Mazuy	Axelle	1
Montagne	Erwann	1
Thiberville	Malvina	1
Van Liedekerke	Florian	1

Ordres du jour

— Point sur l'avancement général
— Fin de sprint et livrables
— Utilisation de Jira et graphiques d'avancement
— Les rétrospectives
— Suivi des risques
— Organisation des branches Git

1 Point sur l'avancement général

1.1 Mise en contexte

L'audit a commencé par une mise au point sur l'avancement de l'équipe concernant le projet.

Il a donc été mis en avant le fait que la partie jeux était actuellement à son 5ème sprint et avait déjà livré 3 jeux sur les 4 demandés, le dernier étant déjà avancé.

Pour l'heure, la date estimée de fin du dernier sprint serait le 21 avril.

1.2 Points à revoir

Toutefois, M. Godart a tenu à repréciser les tenants et aboutissants des réunions en méthode Agile, notamment en explicitant la différence entre sprint review et sprint planning, dans les deux cas le client étant le point essentiel.

2 Fin de sprint et livrables

2.1 Fin de sprint

M. Godart nous a rappelé qu'un sprint étant une sorte de contrat entre l'équipe et le client. Le contenu du contrat étant représenté par les User Stories du sprint. Un sprint ne prend donc fin que lorsque le client a validé chacune de ces User Stories. S'il manque un des points, le sprint n'est pas fini.

Nous devons donc faire valider, chaque sprint et chaque User Story par le client de manière officielle, par un document écrit ou un mail, par exemple.

2.2 Livrables attendus en fin de sprint

M. Godart nous a également indiqué que pour valider un sprint, il fallait non seulement que l'ensemble du code soit validé (au travers des User Stories) mais aussi présenter au client l'avancement des documents et de l'aspect qualité du projet. Cet aspect est démontrable de par :

- un rapport Sonarqube, dévoilant le pourcentage de code factorisable, la qualité du code et l'aspect sécurité ;
- un rapport démontrant de la couverture de tests ;
- le cahier de test ;
- une explication quant à l'automatisation des tests en indiquant lesquels sont automatisés à 100% et lesquels se font à la main.

2.3 Validation technique

Le client n'étant pas spécialisé dans l'aspect technique et donc du code, cette partie là est plutôt à valider du côté de M. Buresi, notre référent technique. Nous

lui enverrons donc un mail très prochainement afin de convenir d'un entretien. Préalablement, nous nous assurerons qu'il ait un accès à notre dépôt GitLab afin qu'il puisse analyser notre code.

3 Utilisation de Jira et graphiques d'avancement

3.1 Automatisation des graphiques d'avancement

Suite à la présentation des graphiques d'avancement réalisés à la main, M. Godart a tenu à nous montrer le caractère contraignant et rébarbatif de cela, étant donné qu'il fallait donc tenir à la main le compte pour les charges estimées, le temps passé et la charge restante.

L'équipe s'est donc engagée à automatiser ses graphiques, notamment par l'intermédiaire de Jira, en faisant entrer sur l'outil les temps et charges cités précédemment.

3.2 Saisie des temps

Dans cette continuité, il faut donc mettre un point d'honneur à la saisie des temps.

De même, comme abordé, chacun des membres doit pouvoir faire les choses de soi même, et non quelques membres uniquement.

Il est donc important de se renseigner davantage sur les outils possibles pour la gestion des temps (voir pour des plug-ins sur navigateur pour avoir un compteur de temps passé sur une tâche), et apprendre à maîtriser les outils déjà mis en place comme Jira.

4 Les rétrospectives

Durant l'audit, il a été rappelé l'importance et l'utilité d'une rétrospective de sprint. Elle a pour intérêt de lister les problèmes survenus au cours du sprint afin de trouver des solutions potentielles. Ces solutions permettront de s'améliorer lors du sprint prochain et de, surtout, ne pas reproduire les mêmes erreurs qu'au sprint précédent.

L'équipe en charge du projet a indiqué qu'elle pratiquait déjà ce genre de rituels, une fois à chaque fin de sprint. Les ressentis de chaque développeur étaient récoltés et les problèmes étaient soulevés, des solutions potentielles étaient aussi envisagées, ce qui est une bonne pratique.

Il a cependant été remarqué qu'il serait pertinent de mettre en place un outil, visant à répertorier les solutions potentielles où seraient indiquées les informations sur qui, quoi et quand. Ainsi, chaque développeur saurait plus aisément

ce qu'il doit améliorer. De plus, cet outil permettrait de vérifier de manière rigoureuse si les actions envisagées ont été mises en pratique et si elles ont été utiles.

5 Suivi des risques

5.1 Livrable attendu

A l'origine, l'équipe en charge du projet avait rédigé son suivi des risques dans une seconde partie du document de l'Analyse Des Risques.

Il est toutefois préférable d'avoir 2 documents distincts, une Analyse Des Risques, préalable au projet ainsi qu'un document de suivi des risques, rédigé tout au long du projet. Ce document répertoriera tous les changements concernant les projets comme :

— la modification d'une probabilité ;
— l'apparition d'un nouveau risque.

5.2 Réévaluation des risques

Durant cet audit, 2 risques principaux ont été évoqués.

5.2.1 Interfaçage plate-forme / jeux

Tout d'abord, le risque concernant l'interfaçage entre le groupe chargé de la plate-forme et le notre peut être revu. En effet, nous avions déployé une boîte noire, système d'authentification basique avec possibilité de lancer des parties, dans l'attente de l'autre groupe. Cette action nous a permis de pouvoir tester nos jeux et ainsi, ne pas pouvoir prendre de retard.

Cependant, la communication entre les 2 équipes s'étant grandement améliorée, il est maintenant possible de lancer l'un de nos jeux depuis leur plateforme, sans donc passer par notre boîte noire. Cette possibilité a été confirmée lors d'une réunion globale impliquant les 2 équipes ainsi que le client. Ce dernier ayant pu attester que l'interfaçage était réussi, nous pouvons donc revoir à la baisse la probabilité concernant ce risque.

5.2.2 Factorisation du code

Enfin, M. Godart a évoqué le fait que la factorisation du code puisse soulever un risque. Rappelons que le fait de développer une application fonctionnelle est tout aussi important que d'avoir une bibliothèque de code commune aux jeux. Cela permettra aux futurs intervenants d'implanter un jeu facilement.

Comme dit lors de l'audit, nous avons déjà factorisé un maximum du point de vue des entités et plus particulièrement des DTO. L'ensemble des jeux possédant des caractéristiques communes, nous avons créé des DTO rassemblant ces caractéristiques. Nous pouvons prendre pour exemple une carte et sa valeur, le concept de cartes est présent dans beaucoup de jeux, nous avons donc créé une

DTO pour ce concept. Ainsi, pour les jeux possédant des cartes, nous n'aurons donc qu'à créer une entité étendant ce concept, nous aurons juste simplement à rajouter le comportement spécifique du jeu en cours d'implantation.

Concernant, la structure des fichiers en front-end, elle est la même pour chaque jeu. Une norme a été mise en place afin de faciliter justement la conception d'un jeu. Les futurs utilisateurs aurons donc à suivre cette architecture uniforme.

Nous avons donc déjà bien réfléchi à cette factorisation de manière générale, cependant, nous allons quand même inscrire ce risque dans notre suivi des risques. Pour éviter que ce risque ne survienne et n'ait un trop grand impact, nous allons aussi recontacter M. Buresi, notre référent technique. Cela sera fait très prochainement afin de lui demander s'il voit des éventuels morceaux de code à factoriser, d'un oeil extérieur.

6 Organisation des branches Git

M. Godart nous a demandé la manière dont nous avons géré notre dépôt GitLab. Nous lui avons indiqué que nous avions 2 branches principales : main et preprod, correspondant aux serveurs de production et de recette respectivement.

Enfin, une séparation en 2 branches est faite : une pour l'équipe plateforme et une autre pour l'équipe jeux. Cette branche de développement est, par ailleurs, reliée à un serveur, comme les 2 branches principales citées précédemment. Pour déployer notre code sur notre serveur de développement, nous avons mis en place une CI/CD permettant d'assurer un déploiement automatique et en continu. Le script de déploiement fonctionne pour n'importe quel serveur, nous l'avons donc partagé au groupe chargé de la plate-forme.

Pour revenir à la branche de développement, nous la séparons ensuite de la manière suivante : une branche par tâche et fonctionnalité. Nous suivons également une norme de codage et nommage, facilitant l'accès et la manipulation de ces branches.

M. Godart nous a demandé si nous connaissions l'outil Git Flow et nous lui avons répondu que c'était justement l'outil que nous utilisions.