Itération 3

Release notes

Planification du projet

Les objectifs de cette itération étaient les suivants :

- Histoire 5 sans implémenter d'avatar
- Ne pas implémenter la possibilité pour un user de modifier ses informations
- Interface de modification visible directement dans le popup
- Barre d'option qui apparaît avec un bouton, comme sur interfaces mobiles
- Un visiteur peut tout utiliser en local, et se synchronise si il décide de créer un compte. Cependant il ne peut pas voter sur les marqueurs
- Pas de gestion des conflits de marqueurs
- Mise à jour des informations au lancement du programme seulement
- Compte non utilisable avant la confirmation de l'utilisateur par e-mail

Typiquement, l'ensemble de ces points ont été respectés.

La *burnchart* du projet

Structure du code et choix techniques

La séparation du projet en *client*/*serveur* (ainsi que *common*, la partie du code partagée) implique également l'apparition de nouvelles classes. En effet les classes situées dans *common* et dont les instances sont échangées à travers le réseau doivent à présent implémenter une *interface* correspondante, et un système d'héritage est parfois requis pour séparer des fonctionnalités qui elles ne sont pas communes. Les classes situées dans *common* sont qualifiées de *sendable model* et leurs interfaces sont dites des *query model*. La plupart des contrôleurs font le lien entre un modèle et une vue et se trouvent du côté client, mais des fonctionnalités plus complexes que de simples requêtes existent également côté serveur, comme l'envoi d'e-mails.

La séparation entre "Modèle", "Vue" et "Contrôleur" existe toujours et les choix effectués à propos de leurs utilités et portées restent vrais. Chaque type de classe se trouvant

désormais dans un *package* propre à ce type, il est redondant de reprendre la liste extensive dans un tableau. Depuis le début de l'itération deux, des efforts ont été faits pour que les classes contenant des données se trouvent bel et bien dans la catégorie "Modèle".

Difficultés techniques rencontrées

Initialisation de communications RESTFul

De grosses difficultés ont été éprouvées à mettre sur pied le système de communications et à décider quelle librairie utiliser. Au final, notre choix s'est porté sur Jersey. La compréhension des mécanismes de fonctionnement de celle-ci a demandé du temps. Et son déploiement encore plus.

Refactoring : séparation des classes entre client et serveur

L'introduction d'un système client-serveur survenant tardivement dans le projet, le refactoring ainsi que la séparation claire et précise des classes entre le client et le serveur a été long, laborieux et a demandé une concentration ainsi qu'une organisation toute particulière.

Interface d'édition de pokémon

L'ajout de cette option a demandé un peu de temps mais s'est vite faite via le refactoring de classes déjà existantes et grâce à un héritage de contrôleurs et de vues.

Interface de login/logon et panel

Cela s'est fait sans gros problèmes et a permis de *refactorer* le *main* et d'autres parties du code pour le rendre (de notre point de vue) plus lisible et léger.

Système de réputation : finitions

Le système de login complet s'est accompagné d'une restructuration des tables de la base de donnée, permettant de stocker la relation entre un "vote" (une appréciation d'un utilisateur sur un marqueur) et un utilisateur, et évitant ainsi toute fraude (votes multiples entre le même utilisateur et un marqueur, entre un visiteur et un marqueur…)