Compte rendu de réunion technique

Groupe 1 Agora V3-1 : Partie plateforme Web

02 Mai 2024

Nom	**Rôle**
Stéphane Buresi	Référent technique PHP / Symfony
Yacine Ben Ahmed	Secrétaire
Robin Sapin	Participant
Sid Ahmed Brahimi	Participant
Corentin Pille	Participant
Mohamed Cherfi	Participant
Michel Nassalang	Participant

Table des matières

1 Ordre du jour

La réunion, d'une durée de 30 min avait pour objectif de faire, dans un premier temps, un point technique sur le projet AGORA concernant les différents fichiers de la partie plateforme. Ce point devait aborder différents sujet tels que :

— Qualité du code
— Communication controlleurs - services

Dans un second temps, il était question de discuter de solutions sur certaines fonctionnalités en cours d'implémentation durant le sprint en cours (à savoir le sprint 7), comme l'invitation à une table, ou bien l'exclusion d'un joueur d'une table en cours de partie. En cas de non implémentation de ces fonctionnalités, les différents retours émis par le référent technique devaient être compilés dans le document dédié aux fonctionnalités non implémentées, afin de donner des pistes de réflexions et des potentiels solutions pour les groupes travaillant sur les itérations suivantes du projet. Il devait aussi être question de l'utilisation du protocole Mercure sur le serveur de développement, qui ne fonctionnait pas au moment de la réunion.

2 Retours et validation du référent

2.1 Qualité syntaxique du code

M. Buresi a dans l'ensemble validé la qualité du code, qu'il a considéré comme convenable dans l'ensemble. Cependant il a souhaité souligner les quelques points d'amélioration concernant le codestyle, par exemple les espaces entre une instruction if et les parenthèses décrivant les conditions, le nombre de retours à la ligne jugé trop nombreux à certains endroits, créant des espaces vides non nécessaires, ou bien le respect de la norme PSR pour le nom des variables ou des fonctions.

Le référent technique a pu nous conseiller quelques solution pour améliorer le codestyle global, comme utiliser la commande $CTRL + ALT + L$ sur PHPStorm, ainsi que les logiciels PHPCodeFixer et PHPCodeSniffer, qui sont des outils permettant l'automatisation de la correction de code PHP.

2.2 Communication controlleur - service

Concernant la séparation des tâches entre les controlleurs et les services associés, qui s'est trouvé être une notion difficile à comprendre et à mettre en place pour l'équipe de développement, M.Buresi a validé l'architecture mise en place par l'équipe, soulignant le bon respect des appels aux services dans les contrôleurs et la présence faible de code métier dans ceux-ci.

Il a toutefois tenu à rappeler que la déclaration des attributs d'un contrôleur devait s'effectuer directement via les paramètres du constructeurs du contrôleur, et non dans le corps de la fonction. Aussi, l'équipe doit faire plus attention à bien tester les valeurs de retours des méthodes des services lors de leur appel dans un contrôleur, afin de vérifier au mieux les cas d'erreurs.

Enfin il a conseillé à l'équipe la création d'un service dédié aux opérations administrateurs (par exemple AdminService), car le contrôleur AdminController lié à ces opérations contenait encore trop de code métier.

2.3 Autres remarques

Concernant la génération de données persistées en base de données présents dans certains fichiers comme l'AdminController (pour la génération des comptes temporaires) ou bien les fichiers de migration (pour la création des différents entités Game représentant les jeux disponibles sur la plateforme), M.Buresi nous a indiqué il était mieux de passer par des fixtures pour ce type de génération, plutôt que par les solutions prises par l'éauipe pour ces différents cas.

Il a aussi conseillé de ne pas tester l'existence d'un utilisateur connecté sur la session dans les différents

instructions conditionnelles présentes dans le code avec l'instruction *($user != null)* mais en utilisant seulement *(user)*, qu'il considère comme une optimisation du code et une façon plus élégante de procéder.

Enfin, une miese en garde a été faite sur l'utilisation de la commande symfony *make :entity*, qui génère automatiquement les getter et setter des attributs de l'entité sans forcément envoyer de code d'erreur en cas de mauvais retours. Il est possible d'utiliser cette commande, mais il devient alors impératif de revoir l'entiereté de ces fonctions afin de bien veiller que des codes de retours soient émis.

3 Questions techniques

3.1 Utilisation du protocole Mercure

Concernant le protocole Mercure utilisé par l'application, l'équipe rencontrait un problème pour le faire démarrer lors des test de de la plateforme en local (et aussi sur le serveur de déploiement), sur la branche de développement de la partie plateforme. Interrogé à ce sujet, M.BURESI a pu nous conseiller de faire attention lors de la définition des variables d'environnement.

3.2 Expiration d'une invitation à une table

L'une des fonctionnalités de la plateforme consiste à pouvoir inviter des joueurs à une table lors de la création de celle-ci. Cette invitation doit par contre expirer au bout d'une semaine et, après demande du client, une notification devait être envoyer aux joueurs invités pour indiquer que l'invitation qui leur avait été envoyée avait expirée. Robin, étant chargé du développement de cette fonctionnalité, a questionné M.BURESI sur des solutions d'implémentations du système d'expiration des invitations. Il a été discuté de la solution actuellement envisagé, à savoir l'utilisation du composent Messenger de Symfony pour exécuter la fonction d'expiration de l'invitation par le biais de l'envoi d'un message 1 semaine arpès la création du Board, reçu par une classe MessageHandler réalisant l'expiration de l'invitation.

M.Buresi a plutôt conseillé l'utilisation de cron jobs, qui vérifieraient chaque jour, dans toutes les tables en attente, la validité des invitations, et déclencherait l'expiration dans les cas nécessaires.

3.3 Proposition d'exclusion d'un joueur d'une partie en cours

Concernant le système d'inactivité d'un joueur durant une partie, et de son exclusion après le dépassement du délai d'inactivité paramétré pour la table, le client souhaitait pouvoir proposer aux autres joueurs un bouton (activable seulement après le dépassement du délai) afin d'exclure le joueur inactif. Étant donné que cette fonctionnalité partageait des similitudes de fonctionnements avec l'expiration des invitations, l'équipe a souhaité discuter avec le référent technique de la meilleure solution à entreprendre.

Après discussion, M.Buresi a indiqué que pour ce cas-ci la solution utilisant le composant Messenger pouvait être envisagé, étant donné qu'il est nécessaire que la proposition d'exclusion se fasse précisément après le dépassement du délai d'inactivité, ce qui ne serait pas possible avec une solution via les cron jobs qui effectuerait seulement cette tâche à certains moments précis définis à l'avance. Cette précision dans le moment d'activation du bouton est nécessaire , afin de garantir l'intégrité de la partie et de ne pas tricher en exploitant ce temps d'inactivité possible. Toutefois l'utilisation de cronjobs n'était pas à écarter si une approche alternative à ce délai avant exclusion était trouvé.