Réunion de présentation avec le référent technique

29 Novembre 2023

Participants

Nom	Prénom	Groupe
Buresi	Stéphane	Référent technique
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

— Présentation de la mission
— Discussion sur l'architecture du projet
— Débat sur l'utilisation de React

Table des matières

1 Présentation de la mission

Lors de la réunion, les deux parties se sont présentées et l'objectif de l'intervention de l'équipe de développement a été abordé.

L'équipe de développeurs a expliqué son intervention sur la refonte complète de l'application web du côté de la bibliothèque de jeux et a demandé des conseils quant à l'architecture de ce projet sur la partie Symfony.

M. Buresi a également fait part de sa prise de connaissance sur le sujet par le biais des sources du code de l'ancienne application. Il a ainsi pu nous fournir un retour sur les points à changer, améliorer ou revoir pour le projet Agora.

2 Discussion sur l'architecture du projet

2.1 Conseils sur les versions des technologies à utiliser

Lors de la présentation des conseils, M. Buresi a notamment expliqué l'intérêt de prendre les différentes versions des technologies que l'équipe de développeurs comptait utiliser, comme PHP et Symfony.

Concernant PHP, la version préconisée est 8.3, sortie il y a quelques jours qui sera stable d'ici le début de la phase de développement. PHP est un langage devenu fortement typé au fil des versions, ce qui évite les bugs et fiabilise l'ensemble du code. Un conseil évoqué a également été de ne pas fermer les balises PHP, cela induirait une faille de sécurité, et serait une mauvaise pratique.

Concernant Symfony, la version préconisée est la 6.4 qui est une LTS (Long Time Support) ce qui est un atout pour le projet Agora qui doit être maintenable dans le temps.

2.2 Conseils sur l'organisation des fichiers

2.2.1 L'utilisation des scripts

M. Buresi a également souligné que l'utilisation de scripts sh n'était pas toujours des plus recommandées dans certains cas. Il est parfois plus indiqué d'utiliser la CI/CD ou des makefile, ce qui n'a pas été le cas dans les versions précédentes d'Agora.

De plus, lors des précédentes versions du code, l'insertion de valeurs en SQL dans le code, ou encore la création de table était visible dans le code. Toutefois, il existe des outils Symfony permettant de faire cela de manière plus propre et professionnelle : le système de migration utilisant Doctrine. Il existe également des bibliothèques de test en Symfony concernant notamment les tests d'insertion de valeurs, la bibliothèque Doctrine Fixture.

2.2.2 Organisation des dossiers

Etant donné que le projet se base sur une architecture MVC (Modèle, Vue, Contrôleur), les sources du code Agora comportait notamment un dossier Model. Toutefois, Modèle n'existe pas en Symfony, on utilisera à la places des Entity ou DTO (Data Transfert Object).

De plus, l'objectif du projet de cette année étant la constitution d'une Bibliothèque de code facilement complétable, la question de la possibilité de l'héritage a été évoquée. Il est donc possible en PHP de réaliser de l'héritage ou des traits. Un trait est une classe PHP pouvant être appelée n'importe où dans le code et rassemblant du code commun. Toutefois l'utilisation de traits n'est pas des plus sécurisé et ne respecterait pas réellement le modèle Objet.

Dans la structure de ce projet, il sera important de séparer en divers dossiers tels que rassembler le code dans un dossier source (src). Ce dossier serait subdiviser en différents sous-dossiers tels que :

— les contrôleurs (Controller) : contenant les contrôleurs globaux, et possiblement un contrôleur par jeu s'ils ont des spécificités ;

— les entités (Entity) : les objets que l'on souhaite manipuler que ce soit des objets de la Base de données ou des DTO.
Il serait donc possible d'avoir des objets généraux à tous les jeux puis utiliser l'héritage pour définir des comportements plus spécifiques aux jeux ;

— les dépôts (Repository) : ce qui permet de faire des requêtes SQL (SELECT) ;

— les commandes (Command) : contenant les commandes permettant de lancer un jeu notamment (le jeu est spécifié en paramètre de la fonction) ;

— les formulaires (Form) : permet de gérer les formulaires en PHP. Les formulaires ne se déclarant pas dans les contrôleurs, comme vu dans le code source ;

— les données de test (DataFixtures) : permettant de générer des données aléatoirement pour tester ;

— les énumérations (Enum) : permettant de regrouper dans le même dossier les différentes énumérations utilisées dans le code ;

— la sécurité (Security) : permettant de gérer la sécurité du code ;

— les services (Service) : permettant de gérer la logique de l'application,

définir les comportements notamment des différents objets.

Un autre dossier du même niveau que les sources représenterait la partie visuelle et graphique de l'application web via des fichiers. Ce dossier serait de nom templates.

3 Débat sur l'utilisation de React

Une autre question importante a été l'utilisation de React et son intérêt dans le projet. M. Buresi a expliqué ses réserves concernant l'utilisation de la technologie React à l'échelle du projet Agora, qui serait une source de travail importante supplémentaire sans garantie d'un meilleur résultat.

De plus, utiliser React signifierait transformer le projet en API et constituer deux projets en un, l'un concernant uniquement la partie graphique (front) et l'autre concernant l'API avec le code de l'application.

Transformer l'application Symfony en API impliquerait donc d'installer des plug-ins externes afin d'ajouter la compatibilité avec le système de REST API.

De plus, cela impliquerait de modifier les contrôleurs et la manière de penser l'architecture du projet, car ceux-ci ne renverraient plus de l'HTML mais du Json.

Une solution évoquée serait donc l'utilisation du moteur de templates Twig, en remplaçant également JQuery par l'utilisation de Symfony UX. Symfony UX permet d'apporter de la réactivité à une application. En effet celui-ci va directement transformer le code Twig en un code JavaScript fonctionnant comme React. Cette solution permettrait donc d'apporter la réactivité de React tout en gardant la simplicité de Symfony.