Master 1 GIL - Document de développement - Fonctionnalités à implanter

Groupe 1 Agora V3-1 : Partie bibliothèque de jeux

14 mai 2024

Version	1
Date	14 mai 2024
Rédigé par	BINGINOT Etienne MAZUY Axelle

Mises à jour du document

Version	Date	Modification réalisée
1	14 mai 2024	Création du document

Table des matières

1 Introduction

Ce document a pour objectif de regrouper l'ensemble des fonctionnalités qui n'ont pas pu être implantés au cours du développement par manque de temps. Chacune de ces différentes fonctionnalités possède une solution ou une piste de solution permettant d'aider au mieux à sa réalisation.

2 Fonctionnalités à implanter

2.1 Bibliothèque de code

2.1.1 Logs complets permettant de facilement rejouer une partie

Le client souhaite pouvoir dans le futur ajouter une fonctionnalité permettant de rejouer une partie à partir des logs de celle-ci. La solution actuelle ne permet pas de facilement réaliser cette solution, l'historique des coups de la partie étant géré par des chaînes de caractères descriptives.

Cependant, il est possible de facilement modifier ce format par un format JSON récoltant à chaque coup l'entièreté des informations du coup du joueur. Ce JSON pourra ainsi facilement être interpreté, que ce soit pour un affichage descriptif quand il s'agira de visualiser l'historique, ou alors pour permettre de facilement rejouer la partie en récoltant les données du JSON.

2.1.2 Utilisation d'exceptions personnalisées

Tout au long du projet, des exceptions sans sémantiques ont été utilisées. Cependant, ceci est une mauvaise pratique et il est préférable d'utiliser des exceptions définies par le développeur et ayant un sens plus fin. Le traitement peut ainsi être modifié en fonction du type d'erreurs, facilitant la gestion des exceptions.

2.1.3 Découpage des contrôleurs

Les contrôleurs des différents jeux sont, pour certains, extrêmement longs et regroupent de nombreuses fonctionnalités ensembles. Un découpage plus précis de ces contrôleurs serait un atout pour le projet, en clarifiant ainsi le rôle des différents contrôleurs et des routes.

2.1.4 Application des normes PSR-12

Les normes PSR-12 ne sont pas appliqués dans l'entièreté du projet, le rendant parfois compliqué à lire. L'utilisation d'un outil comme PHPCS Fixer pourrait permettre de corriger la majorité de ces problèmes, mais un tour complet via PHPStan et PHPCS est nécessaire pour régler la totalité des problèmes.

2.1.5 Transformation des migrations de données de jeu en fixtures

Les données permanentes des différents jeux sont actuellement initialisées sous la forme de migrations Symfony. Cependant, les migrations ne doivent concerner que la structure de données et non les données en elles-mêmes. Il est donc nécessaire de transformer ces migrations en fixtures qui ont elle un rôle de création de données (données de tests ou non).

2.1.6 Amélioration de la bibliothèque d'animation

Certaines animations peuvent être très similaires entre les différents jeux. Une bibliothèque d'animation améliorée peut être réalisé afin de faciliter la réalisation d'animation pour les futurs jeux, et factoriser le code.

2.2 6 qui prend

2.2.1 Correction de l'animation de dépôt de cartes

Dans certains cas, l'animation de dépôt de cartes ne va pas se lancer pour la première carte. Ce bug est probablement dû à un problème de synchronisation avec la file des animations. Il est nécessaire de voir

comment le message de dépôt de cartes est traité dans le code Javascript pour voir si celui-ci lance comme il faut l'animation.

Cela peut également être dû à une possible duplication de publish dans le contrôleur, une autre piste à explorer pour la résolution de ce bug.

Le dernier point à voir est le code de l'animation. Celui-ci possède peut-être un problème bloquant l'animation dans certains cas.

2.3 Myrmes

2.3.1 Correction des publish Mercure

Dans certains cas, les publish Mercure ne sont pas complets. Une action entraînant une mise à jour du plateau personnel ne mettra pas forcément à jour la pré-visualisation du plateau personnel du joueur. Il est donc nécessaire de faire un tour du jeu et des différentes actions possibles pour détecter ces soucis et rajouter les différents publish nécessaires et le traitement associé.

2.3.2 Nouveau découpage des services

Le découpage actuel de Myrmes au niveau des services n'est pas satisfaisant. En effet, ce découpage se concentre sur les différentes phases du jeu, ce qui n'est pas la bonne approche à prendre. Un découpage d'un service réussi doit se concentrer sur les différents objets métiers et le traitement associé à ces objets. Ainsi, il est plus simple de combiner les différents services entre eux et éviter les cycles de dépendance / duplication de code.

2.4 Environnement de développement

2.4.1 Finaliser la configuration de ELK Stack

Pour pouvoir facilement traiter les logs avec ELK Stack, il est nécessaire de définir un moyen d'interpréter ces logs via la création de grammaire. Ainsi, il est possible de récupérer les différents champs séparément et réaliser efficacement des recherches dessus (comme par exemple récupérer l'ensemble des logs d'erreurs).

Par manque de temps, une grammaire complète n'a pas pu être réalisé pour interpréter les logs de Symfony. Réaliser cette grammaire permettrait de facilement analyser les nombreux logs générés par l'application notammement pour un environnement de production.

2.4.2 Améliorer le déploiement de l'application

Une amélioration est possible au niveau du déploiement automatique de l'application. Actuellement, le script de déploiement va exécuter un docker-compose up –build sur la machine à déployer, ce qui aura pour effet de recompiler l'entiereté du projet et de le lancer. Cependant, il est en général plus propre de compiler une image Docker à la fin de la CI/CD en cas de succès puis de push cette image Docker sur la machine qui n'aura qu'à l'exécuter.

Il faut cependant faire attention aux autres microservices qu'il est nécessaire de synchroniser avec ce nouveau conteneur.