Master 1 GIL - Gestion de projet
Cahier De Recettes général
Agora V3-1

Groupe 1 Agora V3-1 : Partie bibliothèque de jeux

26 mars 2024

Version	3
Date	26 mars 2024
Rédigé par	BINGINOT Etienne CAUCHOIS Niels DUCROQ Yohann KHABOURI Izana MAZUY Axelle MONTAGNE Erwann THIBERVILLE Malvina VAN LIEDEKERKE Florian

Mises à jour du document

Version	Date	Modification réalisée
1	05 décembre 2023	Création du document
2	23 janvier 2024	Mise à jour après les retours de l'enseignant de gestion de projet
3	26 mars 2024	Modifications concernant le déploiement et les outils

Table des matières

1 Objet

L'objectif de ce document est de définir les différentes modalités de tests, afin de fournir au client un logiciel conforme à ses attentes. Dans ce cahier de recettes seront répertoriées les stratégies de tests déployées ainsi que l'ensemble des procédures mises en place.

L'objectif de ce projet étant de développer quatre jeux, chacune de leurs fonctionnalités seront testées une par une, ces dernières étant définies dans les Spécifications Technique du Besoin de chacune des jeux.

2 Documents de référence

Les références utiles à la réalisation de ce cahier de recettes sont :
— **Compte-rendu des réunions :** contenant la liste des décisions prises et des attentes du client ;
— **Spécification technique du besoin (STB) :** détaillant toutes les exigences ;
— **Règles des jeux concernés**.

3 Terminologie

Docker : plateforme permettant de lancer certains logiciels dans des conteneurs logiciels. Elle permet d'accroître la portabilité d'une application, car elle pourra être exécutée sur n'importe quel serveur.

PHP : c'est un langage de programmation libre principalement utilisé pour produire des pages Web dynamiques.

Apache : serveur Web permettant de fournir des pages Web à tous les clients qui le demandent.

MariaDB : Système de gestion de bases de données libre.

framework : les frameworks sont conçus et utilisés pour modeler l'architecture des logiciels applicatifs, applications web et des composants logiciels.

PHPUnit : framework de PHP permettant d'effectuer des tests unitaires ainsi que des tests de régression.

IDE : environnement de développement regroupant un certains nombre d'outils. Ces derniers visant à augmenter la productivité des développeurs.

Sonarlint : extension d'IDE permettant de réaliser, entre autres, des vérifications syntaxiques sur le code. Permet ainsi une production de code de meilleure qualité et plus homogène.

SonarQube : logiciel libre permettant d'évaluer la qualité du code en continu. Il sert à identifier les défauts dans le code, à évaluer le taux de duplication de code, mesurer le niveau de documentation et calculer la couverture de test déployée.

4 Stratégie qualité

Ce projet a pour particularité de consister en l'implantation de 4 différents jeux de société. Le périmètre de chaque jeu est donc bien défini et clairement établi.

Tout jeu de société possédant des règles, notre contrôle qualité sera orienté autour du respect de ces règles. Elles ont été retranscrites dans la STB des jeux correspondants.

Afin de les faire respecter, la plupart de nos tests unitaires seront en réalité des tests fonctionnels, visant à vérifier le respect des règles de chacun des jeux. En effet, une méthode correspondant à une possibilité dans le jeu, nos tests fonctionnels sont donc présents et cruciaux pour nous assurer que nos jeux se déroulent tels que décrits dans leurs règles et, par conséquent, dans nos STB.

L'outil utilisé pour les tests de montée en charge est Gatling. Enfin, nous avons aussi mis en place ELK Stack afin d'avoir une gestion des logs serveurs et de nous assurer de son bon fonctionnement. De plus, nous avons configuré Grafana et Prometheus afin d'avoir des rapports et graphiques réguliers sur l'état du serveur ainsi que sur les codes de retour qu'il renvoie. Tout ceci a été fait afin de nous assurer de la bonne qualité du code et de la santé du serveur.

5 Environnement de test

Afin de réaliser les tests sur le plus d'environnement possible, nous utiliserons les navigateurs
- Firefox ;
- Chrome ;
- Opera.

Les tests se feront sur 3 niveaux :
- en local ;
- en développement ;
- en pré-production ;

Ces différents tests seront nécessaires pour le projet car ils vont permettre de s'assurer du bon fonctionnement de l'application et du respect des règles des différents jeux. Ainsi, ils seront faits tout au long de la réalisation du projet pour vérifier que ces derniers n'apportent pas de problème sur le développement. Les responsables de la réalisation des tests sont Erwann MONTAGNE, Izana KHABOURI et Florian VAN LIEDEKERKE, même si les tests unitaires seront réalisés par le propriétaire du code qui pourra partager son code qu'une fois les bugs réparés. Chacun de ces tests auront un cahier de tests comprenant les rapports justifiant les cas de réussites et d'échecs avec les corrections apportées.

5.1 Tests en local

Le serveur de test local sera sur Docker (STB-REA-06) avec PHP (STB-REA-02). Un serveur Web Apache y est présent, ainsi qu'un serveur de gestion de bases de données MariaDB (STB-REA-04).

Les tests unitaires seront effectués à l'aide de PHPUnit dans sa version 9.6.13. De plus, les tests de syntaxe seront réalisés par PHPStan qui est compatible avec l'IDE PhpStorm.

5.2 Tests en développement

Lors de cette phase seront effectués les tests d'application, d'intégration ainsi que les tests unitaires de chacun des développeurs. Cela permet une plus grande couverture de tests. Ceci sera effectué à l'aide de PHPUnit.

6 Déploiement

Concernant la gestion du GitLab, 4 branches principales sont à distinguer :
— main : correspond à la branche de production ;
— preprod : correspond à la branche de pré-production ;
— develop-game : correspond à la branche servant au développement des jeux ;
— develop-platform : correspond à la branche servant au développement de la plate-forme ;
Notre équipe travaillant sur les jeux nous avons mis en place cette organisation : chaque développeur travaille en local sur sa branche et, lorsqu'il a fini sa fonctionnalité, il merge alors sur develop-game. Develop-game est la branche de mise en commun du code de l'équipe jeux. Elle contient aussi la boîte noire, mise en place suite à notre analyse des risques, qui nous permet de tester nos jeux localement et indépendamment de la plate-forme.

A chaque fin de sprint, ce qui correspond à la livraison d'un livrable, nous faisons un merge sur preprod. Cette dernière contient le code des deux équipes. Preprod correspond à un serveur, similaire au serveur de production, sur lequel nous pouvons tester nos jeux mais aussi la connexion entre la plate-forme et les jeux.

Enfin, il est de la responsabilité de l'équipe plate-forme d'effectuer les merge sur main, lorsque les tests sont concluants.

Nous avons aussi mis en place une CI/CD permettant d'effectuer l'ensemble des tests décrits dans la section précédente et de déployer automatiquement le code sur notre serveur develop-game.

7 Responsabilités

Chaque développeur devra s'assurer que sa couverture de tests pour l'ensemble des tests des unitaires est d'au moins 80%. Ainsi, chaque cas de fonction devra être testé par rapport au résultat attendu, ceci pour des paramètres spécifiques. Les cas de succès mais aussi les cas d'erreurs devront être testés.
Les autres types de tests seront effectués lors de la publication du code.

8 Stratégies de tests

Nous définissons 5 types de tests visant à s'assurer de la qualité du logiciel :
— test unitaire : procédure permettant le bon fonctionnement d'une fonctionnalité ;
— test d'intégration : procédure permettant de vérifier la combinaisons des classes et leurs interactions ;
— test d'application : procédure permettant de vérifier le comportement d'une application complète en effectuant des requêtes HTTP ;
— test de non-régression : procédure visant à s'assurer que la nouvelle fonctionnalité n'impacte pas les précédentes.
— test de montée en charge : procédure visant à tester l'application en jouant différentes parties de différents jeux, afin de nous assurer que le serveur tient la charge.

9 Gestion des anomalies

Après chaque modification effectuée, il est primordial de s'assurer du bon fonctionnement du logiciel. C'est pourquoi nous définissons une procédure à appliquer pour limiter un maximum d'erreurs. Cependant, si elles venaient à survenir, nous définissons aussi un protocole à appliquer pour résoudre les problèmes.

9.1 Procédure pour prévenir des bugs

- Effectuer l'ensemble des tests ;
- Fournir aux développeurs, à intervalle régulier, un rapport décrivant les bugs rencontrés ainsi que leurs démarches de résolution.

9.2 Procédure cas d'anomalie

- Si le bug survient lors d'un test unitaire, il en va de la responsabilité du développeur de le résoudre, ce genre de bug est pris en charge immédiatement.
- Si le bug survient lors d'un test d'intégration, de non-régression ou d'application, la personne se rendant compte du bug écrit un ticket expliquant l'erreur et le contexte dans lequel elle est apparue, ce genre de bug est pris en charge le plus rapidement possible dans un délai de deux jours et compte tenu de la criticité de l'anomalie.
- Si le bug survient chez le client, ce dernier informe l'équipe de développement à l'occasion d'un meeting, puis le bug sera renseigné dans un ticket de la même façon qu'un test d'intégration.
- Lors d'une résolution de bugs, il faut garder un historique des actions réalisées pour tenter de le résoudre.

9.3 Procédure du suivi de bugs :

Le suivi des bugs survenus lors des tests d'intégration est fait avec l'outil de planification Jira, qui permet notamment de suivre et de répertorier des problèmes/bugs à corriger à l'aide de tickets.

- Lorsqu'un bug est détecté, un ticket est émis et est placé dans la colonne "à faire".
- Le ticket est déplacé dans la colonne "en cours" par un développeur qui se chargera de sa résolution et également de consigner les actions entreprises pour tenter de le corriger.
- La résolution est ensuite testée et validée par un autre membre de l'équipe de développement.
- Le ticket est enfin placé dans la colonne "terminé" pour signifier la résolution du bug.

10 Procédures de tests

Pour trouver la procédure de tests d'un jeu, se référer au Cahier De Recettes du jeu correspondant.

10.1 Exigences fonctionnelles

ID	Actions	Résultats attendus	OK/NOK
1	Un message est envoyé en UTF-8	Le message figure dans l'historique des messages et est envoyé à tous les autres joueurs de la partie	
2	Un message est envoyé dans un format d'encodage invalide	Le système bloque le message et ne l'enregistre pas	
3	Un message contenant maximum 255 caractères est envoyé	Le message figure dans l'historique des messages et est envoyé à tous les autres joueurs de la partie	
4	Un message contenant plus de 255 caractères est envoyé	Le message est bloqué par le système et il n'est pas enregistré	
5	Un message est envoyé respectant les conditions	Le système vérifie que le message est enregistré et le développeur vérifie l'affichage sans rafraichissement de la page	
6	Un message ne respectant pas les conditions est envoyé	Le système vérifie que le message n'est pas enregistré et le développeur vérifie qu'il n'apparaît pas dans le chat	

11 Couverture de tests

Pour trouver la couverture de tests d'un jeu, se référer au Cahier De Recettes du jeu correspondant.

11.1 Exigences fonctionnelles

ID	Méthode
STB-GES-01	Le système vérifie l'encodage des caractères dans l'en-tête de réponse/requête HTTP Le développeur envoie un message en UTF-8 et vérifie l'affichage cohérent des caractères.
STB-GES-02	Le système vérifie qu'un message de 255 caractères ou moins est affiché complètement. De même, le système vérifie qu'un message de plus de 255 caractères est refusé et non envoyé
STB-GES-03	Le système envoie des messages sans action utilisateur et vérifie si ces messages apparaissent dans l'historique de discussion, le développeur vérifie ainsi l'affichage des messages sans rafraîchissement.

11.2 Exigences qualité

ID	Méthode
STB-QUA-01	Une liste des étapes clés communes à chaque jeu implémenté doit être rédigée et permettre la vérification de chacune d'entre elles
STB-QUA-02	Chaque méthode doit être documentée et expliquée
STB-QUA-03	Une liste des classes, méthodes et relations doit être réalisée et pour chaque point, il doit être indiqué si l'implantation a changé par rapport à l'analyse.
STB-QUA-04	Une liste des étapes clés de l'implantation d'un jeu que l'on a effectuées doit être réalisée et se retrouver dans le document.
STB-QUA-05	Le système vérifie que sur une moyenne de 100 tests, les joueurs voient leur plateau mis à jour en moins de 50ms.
STB-QUA-06	Les éléments spécifiques aux jeux doivent hériter d'un composant existant dans la bibliothèque
STB-QUA-07	Le système doit vérifier les éléments nécessaires au fonctionnement des jeux, non compris dans le code des dossiers de jeu. Le développeur doit également rédiger une liste des besoins de l'autre équipe et le document doit comprendre l'ensemble de ces éléments.

11.3 Exigences d'interface

ID	Méthode
STB-INT-01	Le système simule des clics utilisateurs sur les différents composants présents sur le site et vérifie si le contenu affiché diffère (changement de routes notamment). Le développeur vérifie à la main si chaque action provoque un changement visuel
STB-INT-02	Le système vérifie que les dimensions d'images soient d'au moins 40px. Le développeur vérifie que sur au moins tous les types d'écrans portables disponibles sur le navigateur en mode développement, les éléments sont visibles sans se recouvrir et qu'il est possible d'effectuer les mêmes actions que sur ordinateur.
STB-INT-03	Le système vérifie qu'un utilisateur n'étant pas dans la liste des joueurs de la partie possède les droits de visionnage sur les plateaux des autres joueurs. Le développeur doit également s'assurer de la présence d'une option permettant de naviguer entre les plateaux des joueurs.
STB-INT-04	Le développeur établit la palette de couleurs du jeu selon les plateaux de jeu composants et logo et vérifie que celle-ci correspond à ce qui est utilisé sur l'implantation du jeu correspondant.
STB-INT-05	Si la surface utilisée est tactile (téléphone, tablette), le développeur doit vérifier qu'une aide sur le composant s'affiche en restant appuyé plus de 2secondes. Si la surface utilisée est un ordinateur, le développeur doit vérifier qu'une aide sur le composé survolé plus de 2 secondes s'affiche. Le système vérifie également la présence d'une description pour chaque élément du jeu.

11.4 Exigences opérationnelles

ID	Méthode
STB-OPE-01	Le système simule une action utilisateur et vérifie si une entrée log de cette action est enregistrée en base de données.
STB-OPE-02	Le système simule un joueur sur une partie et vérifie que l'information sur le pseudo de l'utilisateur est identique au pseudo enregistré sur la partie