DEPARTEMENT INFORMATIQUE - IUT 2 GRENOBLE

ANNÉE UNIVERSITAIRE 2023-2024

MÉMOIRE DE STAGE

DÉVELOPPEMENT DE FONCTIONNALITÉS SUR LE LOGICIEL OPEN JOURNAL SYSTÈME

MATHDOC

Présenté par

Arnaud Kersual

Jury

IUT : Mme Aude Maignan

IUT : M. Carl Vincent

Société : Mathdoc

Déclaration de respect des droits d'auteurs

Par la présente, je déclare être le seul auteur de ce rapport et assure qu'aucune autre ressource que celles indiquées n'ont été utilisées pour la réalisation de ce travail. Tout emprunt (citation ou référence) littéral ou non à des documents publiés ou inédits est référencé comme tel et tout usage à un outil doté d'IA a été mentionné et sera de ma responsabilité.

Je suis informé qu'en cas de flagrant délit de fraude, les sanctions prévues dans le règlement des études en cas de fraude aux examens par application du décret 92-657 du 13 juillet 1992 peuvent s'appliquer. Elles seront décidées par la commission disciplinaire de l'UGA.

A Grenoble Le 26/05/2024 Signature

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements au professeur Éric Fontenas, de l'Institut Universitaire de Technologie de Grenoble, m'ayant beaucoup aidé dans mes recherches de stage et m'ayant permis de postuler dans cette entreprise.

De plus, j'adresse mes remerciements à la responsable administrative, Mme Céline Talbi, qui a été d'une grande aide pour les papiers administratifs tout au long du stage ainsi que pour sa réactivité quant à mes questions.

En outre, je tiens à remercier vivement mon maître de stage, M. Simon Chevance, développeur au sein de l'entreprise Mathdoc, pour le temps passé ensemble et le partage de son expertise au quotidien. Grâce à sa confiance, j'ai pu m'accomplir totalement dans mes missions. Il fut d'une aide précieuse dans les moments délicats ainsi que pour répondre à mes diverses questions.

Je remercie également toute l'équipe Mersenne, Numdam et mes collègues de travail pour leur accueil, leur esprit d'équipe et leur bienveillance, qui m'ont beaucoup aidé à m'intégrer dans l'entreprise.

Enfin, je tiens à remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce rapport de stage : ma famille, mon tuteur Mme Aude Maignan, et mes camarades de classe pour leurs précieux retours et conseils.

Résumé du rapport de stage

Mon stage en informatique s'est déroulé du 15/04/2024 au 21/06/2024 au sein de l'unité du CNRS Mathdoc, spécialisée dans les services documentaires et l'édition scientifique dans le domaine des mathématiques. Mathdoc compte une équipe de 19 salariés. L'objectif de ce stage était l'élaboration de fonctionnalités supplémentaires sur le logiciel Open Journal Systems (OJS) qui est une plateforme open-source conçue pour gérer et publier des revues académiques en ligne et développé par le groupe Public Knowledge Project (PKP). Ce stage m'a permis de travailler principalement sur le développement de fonctionnalités et de tests logiciels adaptés aux besoins spécifiques de l'entreprise. L'un des projets majeurs sur lesquels j'ai travaillé portait sur le développement de tests fonctionnels pour deux fonctionnalités sur le logiciel OJS* destinées à assurer leur fonctionnement après une mise à jour. Ce projet, utilisant la technologie JavaScript ainsi que l'outil Cypress [1], avait pour objectif de simplifier et d'automatiser le processus de test de ces fonctionnalités pour chaque changement opéré sur elles. Ma contribution a consisté en la réécriture d'anciens tests utilisant l'outil Selenium en tests compatibles avec l'outil Cypress* ainsi que pour le navigateur Chrome et en l'ajout d'autres tests recouvrant tous les cas d'utilisation des fonctionnalités testées. En parallèle, j'ai développé des scripts pour déployer efficacement un environnement de travail dans un conteneur en utilisant l'outil Docker [2]. La technologie utilisée pour développer cela est le langage Bash. Les scripts simplifie la création d'un conteneur et l'implémentation d'OJS dans les conteneurs créés. Ils permettent de rapidement déployer un conteneur opérationnel pour l'écriture de fonctionnalités ou de tests OJS, avec de multiples ports ouverts pour permettre une visualisation du travail réalisé sur un navigateur web et/ou l'utilisation d'un débogueur. J'ai aussi développé des fonctionnalités (plugins) pour le logiciel OJS. Sous la supervision de mon maître de stage, j'ai élaboré un plugin permettant d'ajouter aux noms d'utilisateurs leurs emails en modifiant les données reçues via une requête AJAX(asynchronous javascript and XML) qui permet de faire des requêtes HTTP asynchrones en utilisant la technologie PHP et le framework [3] Vue utilisant la technologie JavaScript ainsi que la méthode de développement piloté par les tests. Un autre aspect important de mon stage fut ma participation aux réunions hebdomadaires du mardi, où tout le personnel de l'entreprise se réunit pour organiser leurs travaux et faire part de leurs avancées. De plus, en tant que développeur, je participais à une deuxième réunion le lundi qui permet aux développeurs de l'entreprise d'échanger des idées, d'apprendre de nouvelles technologies qu'un collègue nous présente ou encore d'aider à la résolution de problèmes techniques. Elle permet aux équipes de développement d'être à jour sur les avancées de chacun. En termes de compétences techniques, ce stage m'a permis de me perfectionner dans plusieurs langages de programmation, en JavaScript, PHP et Bash. J'ai également acquis une expérience pratique avec des outils et frameworks* tels que OJS et Cypress* pour les outils ainsi que Vue comme framework JavaScript et Laravel pour PHP. Les défis rencontrés pendant le stage, tels que la montée en compétences nécessaire sur l'outil OJS et l'utilisation de serveurs de tests, m'ont aidé à développer des compétences en résolution de problèmes et en apprentissage de nouvelles technologies en autonomie précieuses. En conclusion, ce stage a été une expérience enrichissante qui a contribué à mon développement professionnel. Il m'a permis de mettre en pratique les connaissances théoriques acquises lors de mes deux ans de formation, d'améliorer mes compétences techniques en JavaScript et PHP et de développer des aptitudes en travail d'équipe et de gestion de projet.

1. Un outil de test fonctionnel basé sur JavaScript
2. Plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs légers et portables
3. Structure logicielle qui fournit des outils pour simplifier le processus de développement

Table des matières

Table des figures

INTRODUCTION

J'ai effectué mon stage du 15 avril 2024 au 21 juin 2024 au sein des locaux de la société Mathdoc, sous la supervision de mon maître de stage, M. Chevance Simon. Mathdoc est une unité mixte du CNRS et de l'UGA spécialisée dans le développement de solutions logicielles pour la gestion et la diffusion des revues scientifiques. Située à Grenoble Saint-Martin-d'Hères, c'est une équipe qui compte 19 salariés. Mathdoc utilise le logiciel Open Journal Systems (OJS), un logiciel open-source sous licence GNU General Public License (GPL) version 2, conçu pour gérer et publier des revues académiques en ligne. Développé par le Public Knowledge Project (PKP), ce logiciel facilite l'ensemble du processus éditorial, depuis la soumission des articles par les auteurs jusqu'à leur publication et leur diffusion. Nonobstant Mathdoc n'utilise pas OJS* pour la phase de diffusion mais la plateforme PTF qui est un outil interne à Mathdoc. En outre OJS est reconnu à l'international notamment pour sa capacité à indexer entièrement les articles dans des services de découverte mondiaux tels que Google Scholar, Crossref, DOAJ et bien d'autres. De plus Le projet OJS gère plus de 400000 revues à l'international. Mathdoc développe principalement de nouvelles fonctionnalités pour une plateforme de gestion de revues académiques en ligne appelée le centre Mersenne, une infrastructure d'édition scientifique en libre accès couvrant des thèmes divers et variés. Mathdoc développe aussi d'autres sites liés aux mathématiques, comme Numdam, la bibliothèque numérique française de mathématiques. L'entreprise est organisée en plusieurs départements : développement logiciel, éditeurs (avec les maquettistes LaTeX et les mathématiciens en charge de vérifier les articles soumis), administration et direction. Le sujet de ce stage était le développement de fonctionnalités et de tests logiciels pour le logiciel OJS. Pendant ces 10 semaines, durant lesquelles j'ai pu pleinement appréhender le fonctionnement d'une entreprise, j'ai intégré l'équipe de développement en charge du logiciel OJS pour le centre Mersenne. Mes missions incluaient l'écriture de tests fonctionnels, l'automatisation du processus de test avec Cypress [4] en utilisant JavaScript, l'élaboration de nouvelles fonctionnalités pour le logiciel OJS en utilisant PHP et le framework* Laravel ainsi que la méthode de développement piloté par les tests, la création de scripts pour déployer des environnements de travail adaptés à OJS avec Docker [5] ainsi que le langage de programmation Bash et le développement de tests fonctionnels pour les plugins du logiciel OJS. Lors de ces 10 semaines au sein de l'entreprise, mon travail a été organisé en 4 grandes phases : la montée en compétence sur le logiciel OJS, le framework Laravel ainsi que sur les protocoles de tests automatiques ; le développement de tests ; la création de scripts adaptatifs et évolutifs et enfin, le développement de nouveaux plugins répondant à un besoin client en PHP. Le mémoire est organisé de la façon suivante : Présentation du contexte professionnel, cette partie comprend la présentation de l'organisme d'accueil, le cadre général de travail, et la présentation détaillée du sujet, des objectifs et du planning du stage. L'analyse des besoins et spécifications qui comprend les objectifs, l'étude de l'existant, et la spécification des exigences. La conception qui décrit la structure globale. La réalisation qui discute des choix techniques, de la gestion de projet, des phases d'implémentation, et propose un bilan de la réalisation. La conclusion qui permet de résumer les résultats obtenus et les perspectives d'amélioration. Finallement, les annexes qui incluent le cahier des charges, des exemples de code, et d'autres documents pertinents.

4. Un outil de test fonctionnel basé sur JavaScript

5. Plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs légers et portables

II. PRESENTATION DU CONTEXTE PROFESSIONNEL

II.1 PRESENTATION DE L'ORGANISME D'ACCUEIL

L'organisme d'accueil se nomme Mathdoc. C'est une entreprise/laboratoire rattaché au CNRS, situé à Grenoble, au bâtiment CETA, 150 rue de la Chimie. Mathdoc compte 19 salariés, dont 6 développeurs et un administrateur système/réseaux, 2 maquettistes LaTeX, 3 éditeurs, 3 documentalistes, une responsable administrative et financière, et 3 directeurs. Elle touche à plusieurs centres d'activité, comme le centre Mersenne, qui met en libre accès des publications scientifiques et académiques avec des redirections vers les sites des revues créés spécifiquement pour chacune d'entre elles. Elle offre également la possibilité de soumettre des travaux scientifiques avec une procédure réglementée utilisant le logiciel OJS* (GitHub du logiciel OJS).

De plus, Mathdoc développe de nouvelles fonctionnalités pour le logiciel OJS*, permettant aux chercheurs de publier leurs travaux selon une procédure protocolaire, assurant la qualité du travail publié grâce à un système d'évaluation en quatre étapes : soumission par l'auteur, revue par les pairs scientifiques, copyediting et phase de production. Mathdoc est également en charge du développement de Numdam, la bibliothèque numérique française de mathématiques. Mathdoc échange avec des clients internationaux et met à disposition ses revues en plusieurs langues, principalement en anglais et en français.

II.2 CADRE GENERAL DE TRAVAIL

Au sein de l'entreprise, j'ai travaillé dans un open space avec 4 développeurs. J'étais en compagnie de mon maître de stage, qui travaillait sur le logiciel OJS* en même temps que moi. Il y avait aussi un autre stagiaire et un développeur senior. Chaque semaine, nous avions une réunion le lundi, concernant les développeurs de chez Mathdoc. À chaque réunion, un développeur présente ses avancées ou une technologie qu'il utilise pour en faire part à l'équipe. Des problèmes techniques peuvent être abordés avec l'ensemble de l'équipe lors de ces réunions pour réfléchir ensemble.

Les réunions du mardi concernaient l'ensemble de l'équipe Mathdoc, où chaque responsable et membre d'équipe faisait part de ses avancées sur son travail. Ensuite, des nouvelles concernant Mathdoc étaient partagées s'il y en avait. Chaque membre de l'équipe de développement dispose de droits pour pouvoir fork [6] et cloner les répertoires distants depuis l'outil de versionning GitLab et travailler dessus.

II.3 PRESENTATION DU SUJET, OBJECTIFS ET PLANNING

Dans le cadre de mon stage j'étais ammené à travailler sur le logiciel OJS*. Je devais réaliser des tests fonctionnels en utilisant la technologie JavaScript et le framework* Cypress* pour simuler au mieux un utilisateurs pour des tests sur l'interface web. Suite à cela il j'ai été ammené à automatiser ces tests dans un runner [7] qui fait office de serveur de tests sur le répertoire commun GitLab*. Ce runner* permet le déploiement d'un conteneur en utilisant l'application Docker* et une installation du logiciel OJS à partir d'une image (fichier exécutable contenant l'environnement nécessaire pour OJS) créer à partir d'un Dockerfile (fichier de configuration utilisé pour créer une image Docker). Ensuite les tests sont lancées pour s'assurer que les changements lié à l'application pour un ajout de fonctionnalité ou pour une mise à jour des tests soit valide et ne génère pas d'erreurs.

6. Copie d'un référentiel Git existant
7. Logiciel utilisé pour exécuter des tâches automatisées en réponse à des événements spécifiques

L'autre partie du sujet porte sur le développement de nouvelles fonctionnalités dans le logiciel OJS* en utilisant le langage de programmation PHP et l'API REST pour utiliser la base de données associées au logiciel. PHP etant le langage utilisé par le framework Laravel, les plugins sont écrit en PHP avec des classes fournis avec le logiciel.

Mes objectifs était de réécrire les tests pour le plugin Mersenne et Backend, ainsi que de développer un plugin pour rajouter les emails des utilisateurs à côté du nom des utilisateurs lors d'un ajout de juge sur la soumission d'un auteur, ainsi que de faire un script utilitaire pour déployer rapidement et éfficacement un conteneur prêt à d'utilisation par un développeur sur le logiciel OJS*. L'objectif secondaire était développer un plugin dit "read only" permettant à un utilisateur d'OJS de ne seuelement voir une soumission d'un auteur sans pouvoir la modifié. J'ai organisé mon planning avec Trello, un outil de kanban de tâche en ligne et gratuit.

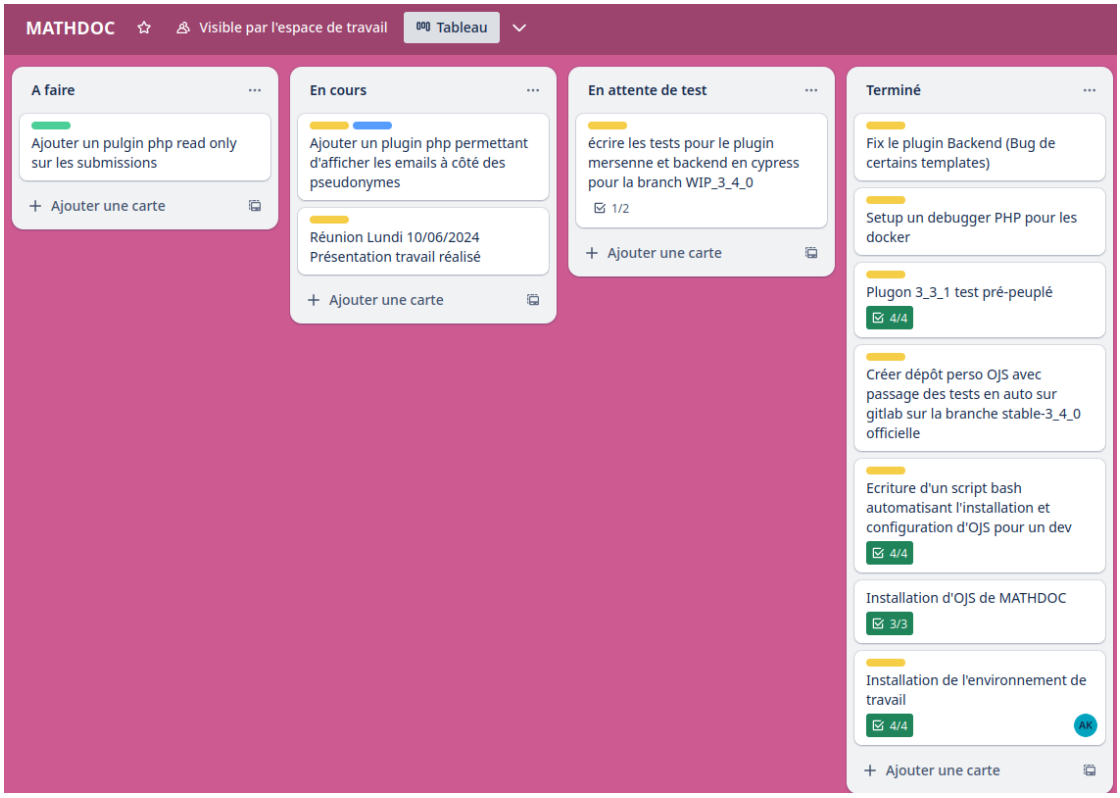

FIGURE 1. Kanban de tâche en ligne "Trello"

Cet outil m'a permis d'organiser mon travail en plusieurs états. En commençant par les tâches qu'il me restait à faire, avec celles en cours et celles en attente de test et de validation de la part de mon maître de stage ou d'autres développeurs. Ce kanban de tâches me permet de garder une trace des tâches effectuées, ce qui permet d'avoir un historique de l'évolution de mon travail dans l'entreprise.

J'ai ensuite noté dans un carnet de bord mon avancée quotidienne sur mes tâches afin de pouvoir prendre du recul sur mon efficacité et mon travail, et de me débloquer si un problème survenait. L'illustration du carnet de bord est disponible à l'annexe cf. Annexe A : Illustration du carnet de bord.

Sur les 10 semaines de stage, il était prévu que je termine les tests fonctionnels sur les plugins Mersenne et Backend, ainsi que de développer un ou plusieurs plugins pour le logiciel OJS*.

III. ANALYSE DES BESOINS ET SPECIFICATION

III.1 OBJECTIF

L'objectif de cet étude de l'existant est de pouvoir comprendre comment le logiciel OJS* fonctionne avant d'apporter des modifications dessus. L'autre objectif est de comprendre comment l'environnement de tests fonctionne avec les runner* sur gitlab et les environnements de développements dédiée comme les conteneurs* en utilisant l'outil Docker [8].

III.2 ETUDE DE L'EXISTANT

Logiciel OJS

Le logiciel OJS* fonctionne avec le framework* Laravel. Il implique donc l'utilisation du langage PHP pour développer avec. La structure du logiciel se base sur le modèle MVC suivant :

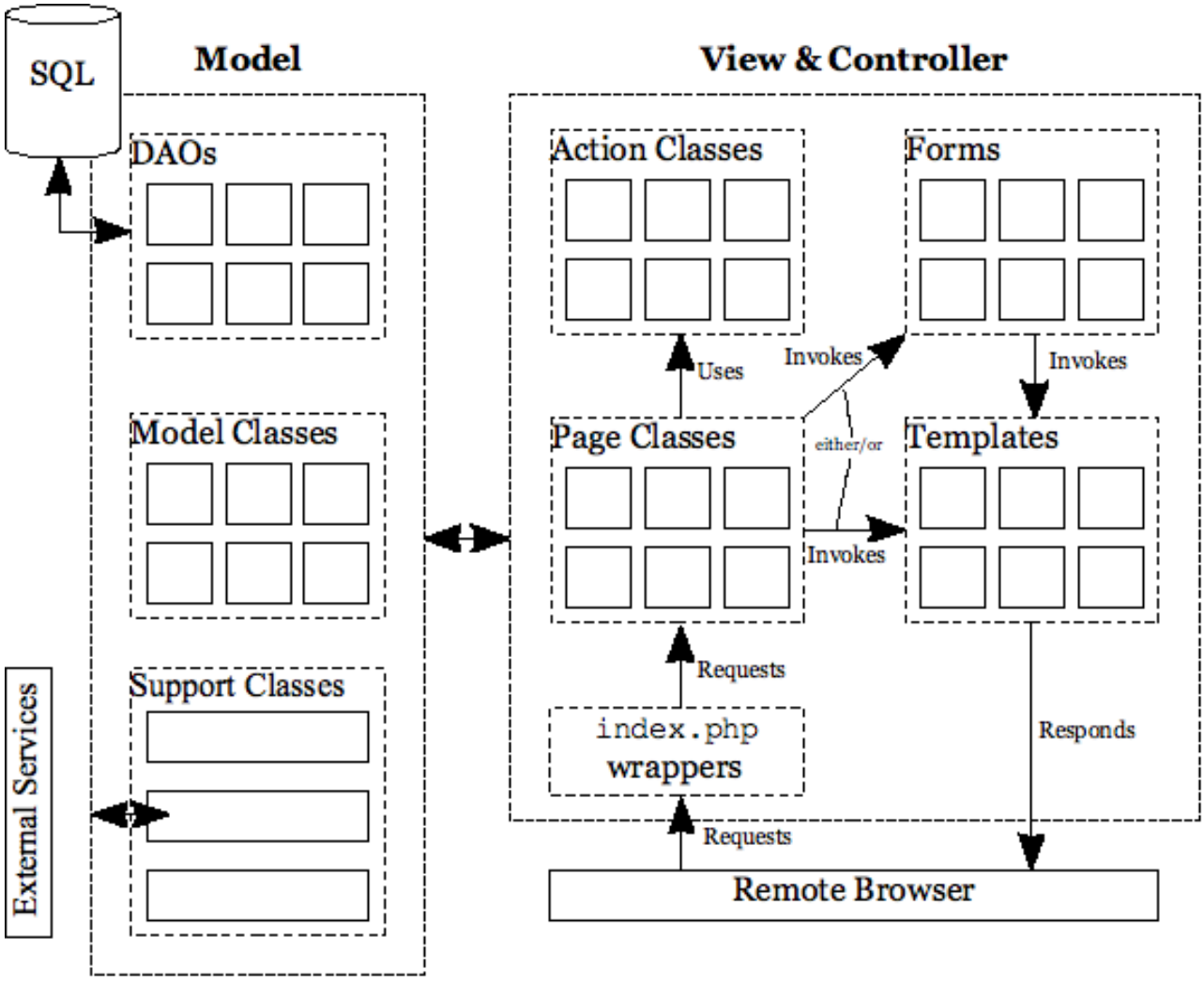

FIGURE 2. Modèle MVC du logiciel OJS

8. Plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs légers et portables

Le modèle ci-dessus est composé d'une base de données avec un DAO* fournissant principalement des fonctions de mise à jour, de création et de suppression pour leurs classes de modèle associées et est responsable de toutes les interactions avec la base de données. De plus, les classes de modèle implémentent des objets PHP représentant les différentes entités du système, telles que les utilisateurs, les articles et les revues. En outre, les classes de support fournissent des fonctionnalités de base telles que des classes et des fonctions communes diverses.

Il y a ensuite les classes de page qui reçoivent les requêtes des navigateurs web des utilisateurs, délèguent le traitement requis à diverses autres classes, et appellent le template Smarty approprié pour générer une réponse. Les templates Smarty sont un système de templates utilisé pour séparer la logique de présentation (HTML) de la logique de l'application (PHP). Cela permet de séparer le développement du design d'une page web. Smarty est également un moteur de templates pour PHP qui permet de créer des templates HTML dynamiques (avec données). Les templates Smarty sont aussi responsables de l'assemblage des pages à afficher aux utilisateurs.

Viennent ensuite les classes d'action utilisées par les classes de page pour effectuer un traitement complexe des requêtes des utilisateurs. Pour finir, il y a les formulaires qui sont des mini-templates pouvant être intégrés dans une page (template) et invoqués par des boutons ou tout autre déclencheur.

Outil Cypress et intégration au versionning

L'autre partie de cette étude porte sur les runners* qui permettent de valider les changements effectués sur l'application par une série de tests fonctionnels en utilisant l'outil Cypress* pour éviter les potentielles erreurs que ces modifications pourraient apporter. Le protocole de tests fonctionne de la manière suivante :

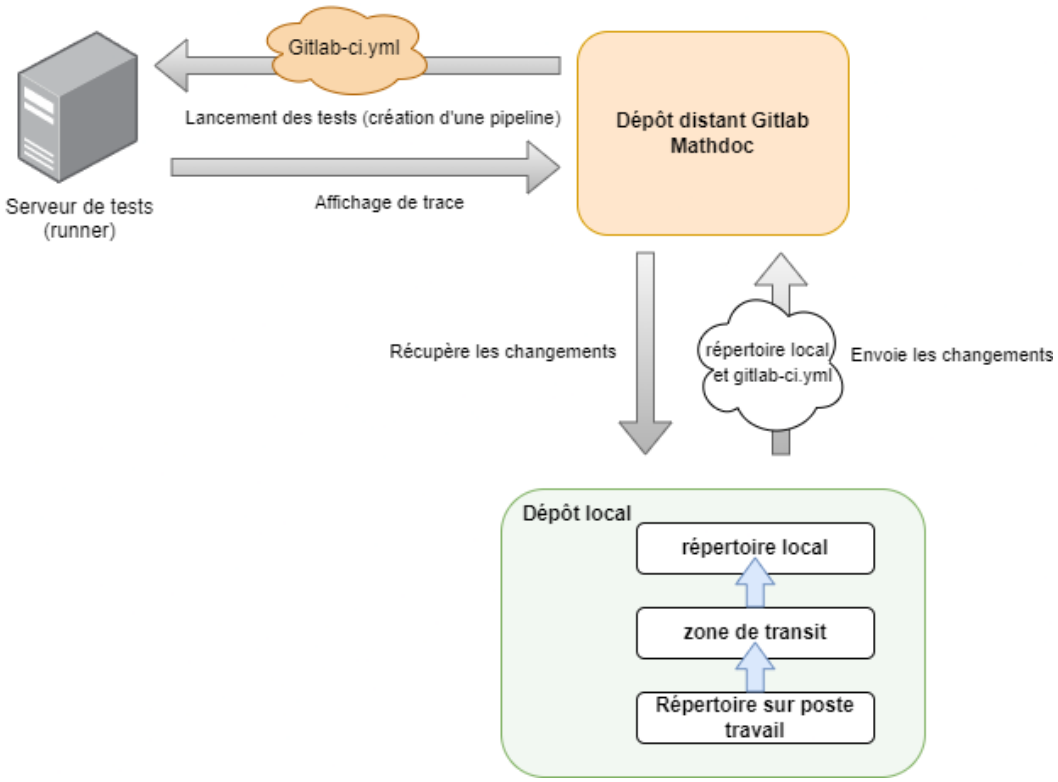

FIGURE 3. Schéma de la procédure de tests sur Gitlab

Sur le schéma ci-dessus nous avons le poste de travail d'un développeur (dépôt local) et le dépôt distant hébergé sur GitLab [9] de Mathdoc. Une fois qu'un développeur souhaite propager ses modifications sur le dépôt officiel de Mathdoc, il devra créer un fichier Gitlab-CI [10] (s'il n'y en a pas). Ce fichier est constitué d'instructions que le runner [11] devra exécuter, notamment le déploiement d'un conteneur avec une image Docker [12] fournie. Le fichier comporte un pattern régulier d'instructions qui sont l'installation du logiciel OJS [13] dans une version compatible avec celle des modifications du développeur, puis l'intégration des changements du développeur et enfin les tests couvrant l'ensemble des modifications apportées. Sur le schéma, nous pouvons voir le gitlab-ci.yml être d'abord émis par le poste local du développeur avant d'être utilisé par le runner* qui ensuite interprète ses instructions dans un environnement Linux sous distribution Debian 12.

Le runner peut également enregistrer des captures d'écran dans un répertoire associé aux tests en cours lorsqu'une erreur survient pendant les tests avec l'outil Cypress*. En voici un exemple :

9. Outil de versionning, gardant un historique du développement
10. Fichier permettant de définir des étapes de traitements à exécuter par des machines virtuelles dans gitlab
11. Logiciel utilisé pour exécuter des tâches automatisées en réponse à des événements spécifiques
12. Plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs légers et portables
13. Logiciel Open Journal System

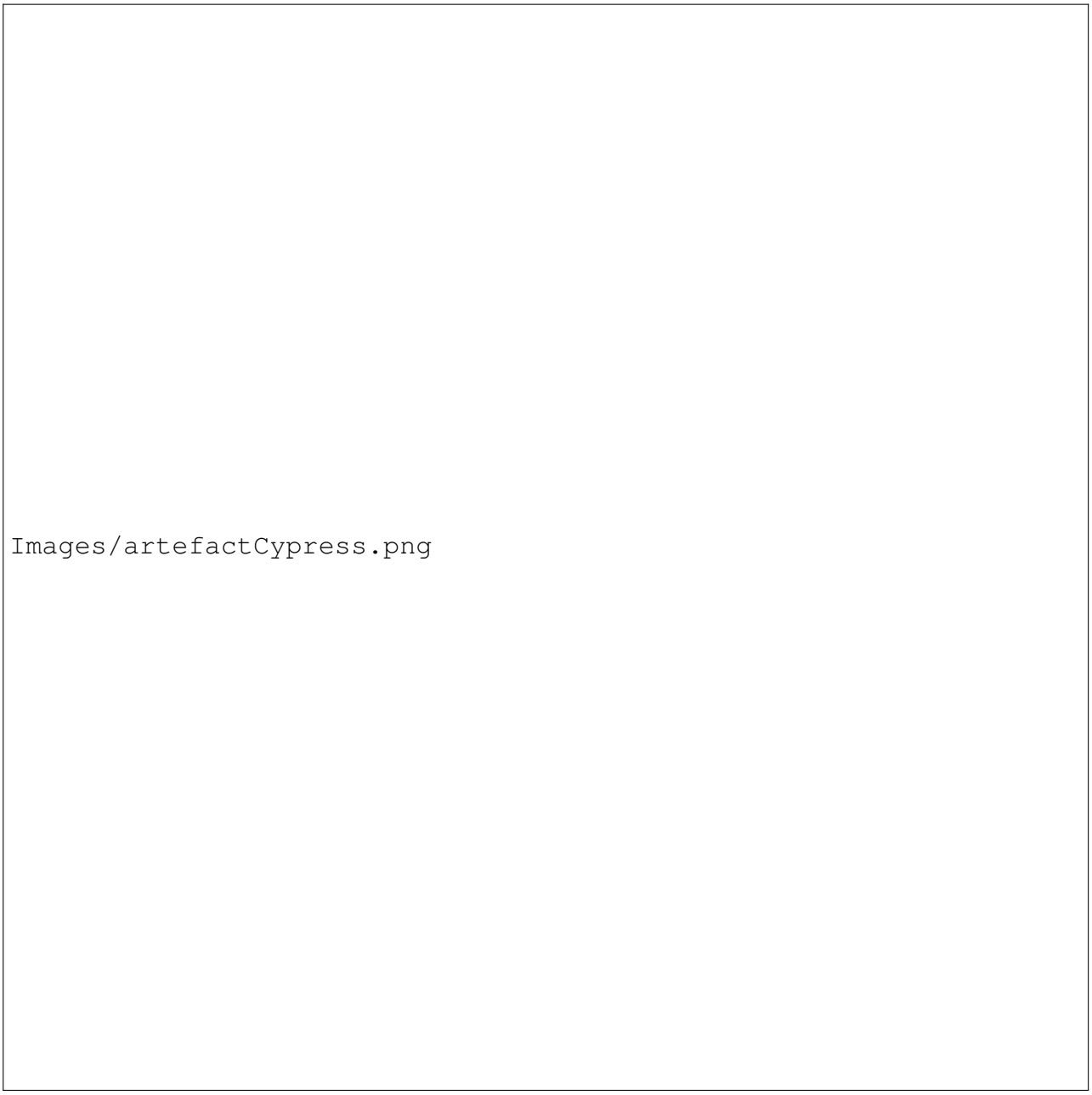

FIGURE 4. Illustration d'une capture d'écran de Cypress sauvegardé

IV. CONCEPTION

IV.1 STRUCTURE GLOBALE

Les plugins que je vais devoir développer nécessitent une arborescence conforme à un modèle qui comporte les fichiers suivants :

```
/ojs
|-+ /plugins
| +-+ /generic
|   +-+ /tutorialExample
|     |-- TutorialExamplePlugin.php
|     +-- version.xml
```

OJS* permet de créer différents types de plugins comme genericPlugin, blockPlugin, reportPlugin, import/exportPlugin. Chaque type de plugin intervient à un moment donné dans le cycle de vie des requêtes HTTP, en utilisant des hooks (hameçons) qui interceptent des événements/actions spécifiques et exécutent du code en conséquence. Voici un schéma décrivant le cycle de vie d'une requête pour OJS* :

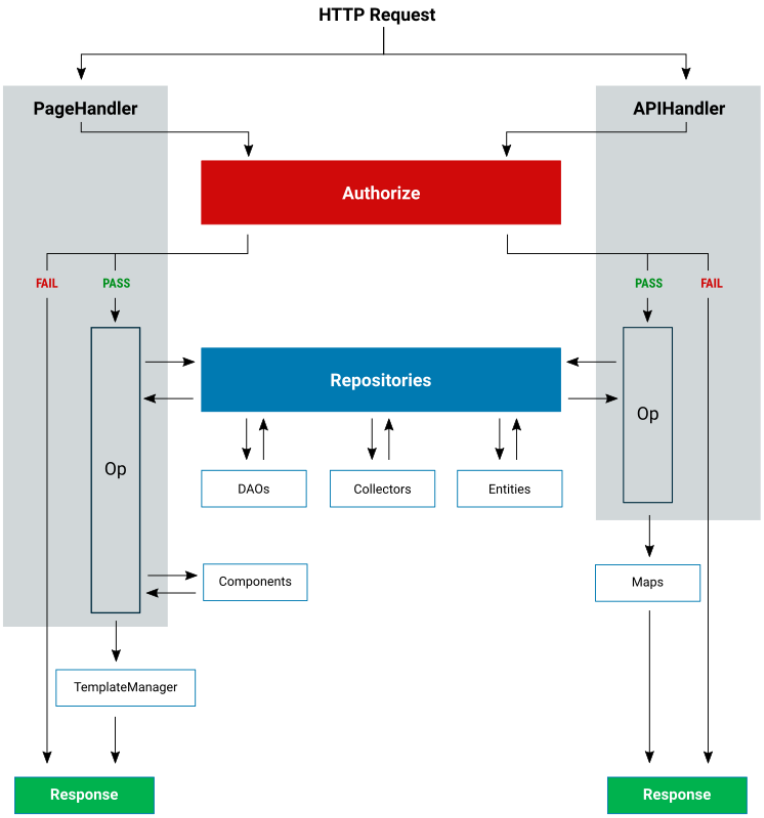

FIGURE 5. Schéma du cycle de vie d'une requête pour OJS*

Ce schéma met en évidence que les plugins de type generic (ceux utilisés par mes plugins) interviennent au niveau du répertoire (représenté par un rectangle bleu). Cela leur permet d'accéder aux données de l'URL ainsi qu'à la base de données, aux entités et d'autres serivces. Qui est souhaitable pour intervenir sur les tempaltes.

V. REALISATION

V.1 CHOIX TECHNIQUES

Choix du standard de développement

Lors de la réalisation de tests fonctionnels ou de plugins, j'ai suivi des normes de codage, comme par exemple l'ajout de commentaires informatifs pour chaque ajout de méthode, classe et test de ma part, en respectant aussi l'indentation et en gardant un code toujours lisible. En ce qui concerne le versionnement, chaque ajout de fonctionnalité ou de test est soumis à un contrôle par un runner [14] de test. Pour le développement, j'utilise mes propres branches et forks (copies) du dépôt commun de Mathdoc afin de faire mes ajouts avant la validation avec mon maître de stage.

La revue de code est effectuée par mon maître de stage qui me fait part de ses retours à l'oral. Pour l'intégration continue, l'outil Gitlab-CI [15] est utilisé, permettant de tester le code à chaque changement apporté. L'outil GitLab-CI* peut aussi réaliser d'autres tâches comme créer des fichiers à partir de commande shell et les stocker sous forme d'artéfact* GitLab qui sont ensuite stocké dans un répertoire GitLab* et accessible par d'autres fichier GitLab-CI pour leurs propre utilisation. Pour obtenir une image via un Dockerfile [16] capable de créer ensuite un environnement adaptées à OJS [17] dans un conteneur on peut par exemple utiliser une image créer par un GitLab-CI stocké dans un répertoire GitLab(GitLab registry) au sein d'un autre GitLab-CI prévut lui pour les tests fonctionnels et ainsi déployer un conteneur adapté.

Les tests fonctionnels sont effectués à l'aide de Cypress*. Les tests unitaires et d'intégration sont écrits en PHP, et il existe également des tests unitaires natifs au logiciel OJS. Avant chaque déploiement pour une revue académique, nous effectuons une série de tests sur les fonctionnalités qu'elle utilise pour garantir que les nouvelles fonctionnalités ou corrections n'introduisent pas de régressions. En adoptant ces standards, nous nous assurons que le développement est structuré, cohérent et maintenable, tout en facilitant la collaboration et en minimisant les erreurs. (Un point négatif est que le temps pour un runner* de faire les tests est très long, suivant la quantité de tests fonctionnels à effectuer.)

V.2 DÉPLOIEMENT DE CONTENEURS

Choix des langages et des outils

La première tâche qui m'a été confiée lors de mon stage était d'élaborer un script en utilisant le langage Bash, adapté pour les utilisations sur des postes Linux afin de simplifier et d'accélérer la mise en place d'un conteneur avec un environnement adapté pour le développement sur OJS.

Le conteneur permet d'isoler le logiciel OJS du poste de travail pour une meilleure organisation sur les postes de chaque développeur. Par exemple, les fichiers installés dans le conteneur ne sont pas visibles sur le poste de l'utilisateur. Pour la création du conteneur, Docker semble un choix judicieux de par sa communauté et ses ressources abondantes, mais aussi par sa compatibilité avec des environnements de production ou de test.

14. Logiciel utilisé pour exécuter des tâches automatisées en réponse à des événements spécifiques
15. Fichier permettant de définir des étapes de traitements à exécuter par des machines virtuelles dans gitlab
16. fichier de configuration utilisé pour créer une image Docker
17. Logiciel Open Journal System

Contraintes

Pour l'écriture du script, il fallait répondre à une contrainte. Il devait offrir une alternative plus rapide et simple d'utilisation que Docker, sans nécessiter de familiarité avec le logiciel pour créer un conteneur opérationnel.

Pratiques adoptées

La première pratique consistait à pouvoir choisir entre l'installation du logiciel vierge ou avec un pré-peuplement de la base de données. Cela avait pour but d'accélérer drastiquement la configuration du logiciel, car l'installation devait être faite manuellement si le logiciel était vierge.

De plus, une deuxième pratique consistait à construire (build) l'image Docker avec un Dockerfile* récupéré depuis un répertoire GitLab, offrant aussi la possibilité de choisir la version du Dockerfile*, ce qui impactait, par exemple, la version de PHP installée à l'intérieur.

La troisième pratique que j'ai mise en place était de configurer le conteneur pour qu'il ouvre deux ports : l'un permettant de se connecter au site OJS contenu dans le conteneur depuis les postes de travail, et l'autre pour activer un débogueur afin de tracer les erreurs survenant lors du développement.

Enfin, la dernière pratique était de pouvoir choisir entre installer le logiciel depuis le répertoire chez Mathdoc ou depuis Public Knowledge Project. Il fallait aussi montrer toutes les branches/versions disponibles du logiciel dans chacun des répertoires.

V.3 TESTS FONCTIONNELS

Choix des langages et des outils

En ce qui concerne les tests fonctionnels, l'outil utilisé est Cypress*. Cet outil a été développé en JavaScript et, par conséquent, le langage adapté à l'outil est le JavaScript. De plus, Cypress permet de supporter des frameworks* JavaScript comme React, Vue et Angular. Cypress fournit également une API facilitant l'élaboration de tests en JavaScript. Développer les tests fonctionnels dans un autre langage reviendrait à se priver des nombreuses aides que Cypress apporte ainsi que d'une communauté autour du langage JavaScript forte.

Contraintes

Pour l'implémentation des tests fonctionnels, plusieurs contraintes devaient être respectées lors du développement. Pour éviter d'augmenter le temps d'attente dû à l'exécution en chaîne des tests fonctionnels, il est impératif d'avoir des tests rapides et peu chronophages. Il est important, par exemple, de ne pas utiliser de pauses fixes entre ou pendant les tests, mais plutôt d'optimiser le temps des requêtes elles-mêmes pour maximiser les gains de temps. En outre, il faut aussi que les tests fonctionnels puissent couvrir l'ensemble des plugins qu'ils tests.

Pour donner une idée du temps nécessaire à l'établissement de l'environnement de tests, le runner* prend environ 30 minutes. Ensuite, selon le nombre de plugins à tester, le temps que met le runner pour effectuer les tests peut varier de 20 à plus de 40 minutes. En cas d'erreur durant les tests, il est

nécessaire de tout recommencer, ce qui est coûteux en termes de temps. Voici un schéma illustrant les étapes effectuées par le runner lors de son lancement :

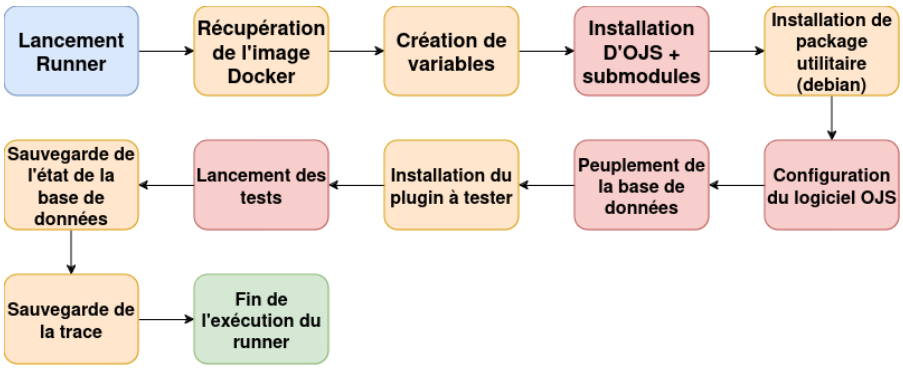

FIGURE 6. Schéma de l'exécution d'un runner non optimisé

Sur ce schéma, les rectangles en orange représentent les tâches peu coûteuses et ceux en rouge les tâches chronophages. Dans un runner* non optimisé, l'accumulation de tâches rouges peut rapidement devenir un problème sur le long terme. Pour pallier au mieux à cela, nous avons adopté de bonnes pratiques.

Pratiques adoptées

Les pratiques adoptées afin de garantir un meilleur gain de temps se porte sur les tests fonctionnels avec des interceptions de requêtes plutôt que des temps d'attente fixe :

```
// Intercept the API request and wait for the request to finish
cy.intercept('POST', '/index.php/publicknowledge/api/v1/submissions/20/submit').as('continueClicked');
cy.contains("button", "Continue").click();
cy.wait('@continueClicked');
```

FIGURE 7. Code interceptant une requête d'API REST

Sur l'image ci-dessus, on peut voir qu'en premier, on attend la requête, c'est-à-dire qu'on se prépare à l'intercepter et qu'on donne un nom à cet événement. Sur la deuxième ligne, on effectue un clic sur un bouton qui va envoyer la requête et déclencher l'événement. Enfin, sur la troisième ligne, on attend que la requête (l'événement) se termine.

De plus, une autre pratique consiste à réduire le temps que mets un runner* pour effectué les tests. Voici en comparaison un schéma illustrant les étapes effectuées par le runner de façon optimiser durant son lancement :

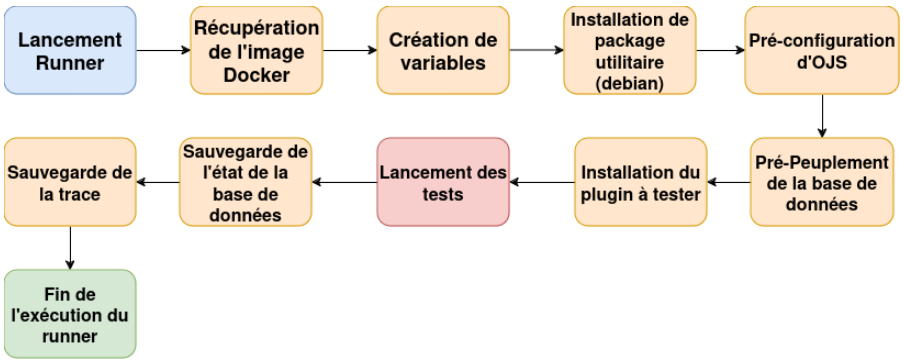

FIGURE 8. Schéma de l'exécution d'un runner optimisé

Le schéma ci-dessus comporte moins de rectangles rouges. L'utilisation d'images avec OJS* déjà installées fournit un gain de temps considérable, et le pré-peuplement de la base de données permet d'avoir en amont des données telles que des soumissions pour tester les plugins sans passer du temps à les créer. Cela permet au runner [18] de se concentrer uniquement sur le lancement des tests et de ne pas perdre de temps inutilement. Enfin, une bonne pratique consiste à limiter le temps d'exécution maximal par instance des runners à une heure, ce qui évite les exécutions de longue durée.

V.4 LOGICIEL OJS

Choix des langages et des outils

Le logiciel OJS* gère ses templates avec le framework [19] Vue.js [20], qui est un outil développé en JavaScript afin de construire des interfaces utilisateur appelées templates. Les templates sont aussi en Smarty [21] côté serveur pour générer des pages web dynamiques (avec des données) et avec Vue.js* pour améliorer l'expérience utilisateur avec ses modules permettant l'intégration d'interfaces utilisateur sans recharger une page et avec un temps d'attente moins long. De plus pour le développement des plugins, les hook [22] ne sont disponible qu'en PHP. Le logiciel repose sur le framework Laravel [23]. Il est écrit en PHP et inspiré du framework Symfony, largement connu des développeurs web avec plus de 1.1 millions de sites utilisant le framework. Par conséquent, il est plus rapide et efficace d'utiliser PHP pour apporter des modifications au logiciel. PHP est également parmi les meilleurs pour le développement web et bénéficie d'une grande communauté et d'un support actif.

Contraintes

Pour le développement des deux plugins, chacun avait ses contraintes. Pour le premier plugin qui affiche les emails des utilisateurs lors de l'ajout d'un juge (reviewer), il fallait trouver une façon durable de modifier le template utilisé pour afficher les juges entre chaque changement de version d'OJS*. Pour voir l'illustration du plugin, se reporter à l'annexe cf. Annexe B : Illustration du plugin userMailPlugin.

Un contre-exemple aurait été de modifier directement le template en écrasant celui par défaut. Cependant, cela nécessite la compilation du nouveau template, ce qui n'est fait qu'à l'installation d'OJS, avant que les plugins soient installés. Il est donc plus problématique de choisir cette méthode car elle nécessite une recompilation supplémentaire ou bien l'utilisation d'une version modifiée d'OJS*, qui demanderait plus de maintenance entre les nouvelles versions.

Le deuxième plugin est une réécriture d'une fonctionnalité déjà présente dans OJS*. Cependant, cette fonctionnalité était intégrée au logiciel et non en tant que plugin, ce qui rendait sa maintenance plus difficile en raison d'une absence de tests et des fusions de code (merge requests) fréquentes avec chaque mise à jour d'OJS*. Il fallait donc la convertir en plugin pour, d'une part, effectuer des tests fonctionnels et, d'autre part, assurer une maintenance plus facile et durable sous forme de fonctionnalité à installer.

18. Logiciel utilisé pour exécuter des tâches automatisées en réponse à des événements spécifiques
19. Structure logicielle qui fournit des outils pour simplifier le processus de développement
20. Framework JavaScript pour développer des applications web
21. Système de templates utilisé pour séparer le développement du design d'une page web
22. intercepte des événements/actions spécifiques et exécutent du code en conséquence
23. Un framework PHP pour le développement web

Pratiques adoptées

Pour le premier plugin, la pratique adoptée consistait à modifier directement les valeurs des requêtes AJAX [24] reçues par les utilisateurs. En passant par la requête, les adresses e-mail des utilisateurs étaient injectées directement dans la variable contenant leurs noms cf. Annexe B : Illustration du plugin userMailPlugin, ce qui avait pour effet d'afficher leurs e-mails à côté de leurs noms cf. Annexe D : Morceau de code d'interception et de modification d'une requête AJAX. Dans la méthode montrée en annexe, l'argument passé en paramètre de la fonction est les données que l'API renvoie à l'utilisateur. On itère ensuite sur chaque composant de la requête et, pour chaque juge, on modifie ses données (items). À l'intérieur de chaque boucle, on transforme la variable contenant le nom d'un juge en ajoutant son mail à côté. On retourne ensuite les nouvelles données (qui comportent les noms modifiés des juges). Cette méthode permet de modifier le template sans avoir à écraser l'original, assurant ainsi une durabilité à travers les mises à jour, sauf en cas de changement majeur.

Pour le deuxième plugin, la pratique adoptée a été de développer un plugin de type generic pour pouvoir modifier les permissions de l'utilisateur lors de son interaction au sein d'OJS* et d'écrire des tests fonctionnels afin de s'assurer du bon fonctionnement du plugin à travers les versions. Pour voir une illustration du plugin, se reporter à l'annexe cf. Annexe C : Illustration du plugin readOnlyPlugin. La différence entre les deux images de l'annexe montre les fonctionnalités restreintes pour un assistant avec l'absence de bouton pouvant provoquer des actions visant à modifier la soumission (le bouton "Add Reviewer" par exemple).

V.5 BILAN DE LA REALISATION, EVALUATION

Rappel du context

Au cours de ce stage, mon objectif a été de migrer les tests de Selenium vers l'outil Cypress* et de développer deux plugins pour le logiciel OJS*. Ces plugins avaient pour but d'améliorer certaines fonctionnalités existantes et d'en ajouter de nouvelles. Le premier plugin visait à afficher les adresses e-mail des utilisateurs (juges) à côté de leurs noms lors de l'ajout d'un juge, tandis que le second consistait en la réécriture d'une fonctionnalité existante sous forme de plugin indépendant.

Résumé des réalisations

Au cours de cette période, j'ai pu :

- Développer un script bash permettant l'installation et la configuration d'un conteneur (Docker) avec un environnement adapté au développement pour OJS.
- Développer et tester un plugin permettant l'affichage des emails des utilisateurs.
- Réécrire une fonctionnalité existante en un plugin indépendant pour une meilleure maintenabilité et testabilité.
- Effectuer la migration des tests fonctionnels en Selenium vers Cypress pour assurer le fonctionnement des plugins à travers les versions.
- Élaborer de nouveaux tests fonctionnels pour assurer le développement piloté par les tests pour les nouveaux plugins.

24. requête HTTP effectuée de manière asynchrone depuis une page web

Analyse des Résultats

Les objectifs fixés au début du projet ont été atteints :

- Le script bash permet un gain de temps pour les développeurs travaillant sur le logiciel OJS, souhaitant isoler le travail réalisé sur le logiciel de leurs stations.
- Le plugin d'affichage des e-mails a été intégré avec succès et fonctionne comme prévu. Il aurait cependant été préférable de créer une zone séparée pour les e-mails afin de pouvoir appliquer un style différent du nom de l'utilisateur pour mieux différencier les deux.
- La réécriture de la fonctionnalité "read Only" en plugin a permis d'améliorer la maintenabilité du code et de faciliter les tests fonctionnels.
- Les tests automatisés avec Cypress* permettent de couvrir l'ensemble des plugins développés ou à tester.

Difficultés et Problèmes Rencontrés

Dès le début de mon stage, lors de ma première mission sur l'élaboration du script Bash, j'ai rencontré des difficultés techniques avec le logiciel Docker*. Les Dockerfiles* n'étaient pas correctement configurés. Python n'était pas à la bonne version, ainsi que PHP, ce qui empêchait OJS de fonctionner correctement.

J'ai aussi rencontré des difficultés lors du développement des tests. Il fallait chercher dans le code des plugins testés pour avoir une compréhension suffisante à l'élaboration des tests. Cela était chronophage étant donné le nouvel environnement dans lequel je suis arrivé et la taille des plugins.

Les dernières difficultés techniques que j'ai rencontrées sont survenues lors du développement des plugins. La modification des templates nécessitait parfois de rechercher lequel modifier, et à cause du framework Vue.js*, il était souvent très long de trouver le bon template. Il y a également des limitations techniques avec Vue.js qui compliquaient la tâche pour modifier certains templates. Certains problèmes n'ont pas été résolus dans le temps imparti, comme pour le plugin ajoutant les emails des utilisateurs où l'email devait avoir un style différent du nom.

Apprentissage et Compétences Développées

Au niveau technique, j'ai pu approfondir mes connaissances en PHP, en développement sur le logiciel OJS et en élaboration de tests fonctionnels avec Cypress*.

J'ai aussi pu améliorer mes compétences en gestion de projet, en communication avec les équipes, et en interaction dans un milieu professionnel grâce aux réunions et à l'open space.

Durant ce stage, j'ai été satisfait de ma performance globale, ayant atteint les principaux objectifs du projet. Cependant, j'identifie une marge d'amélioration dans la gestion du temps par rapport à mes tâches et dans la réactivité lorsqu'un problème survient. Le principale point à améliorer est le temps passé sur chaque tâches qui peu encore diminué avec de l'expérience.

CONCLUSION

Pour conclure, cette expérience a été très positive et enrichissante. J'ai atteint les objectifs fixés par mon maître de stage, et j'ai également développé de nouvelles compétences techniques en PHP avec le framework [25] Laravel [26], en JavaScript avec l'outil Cypress [27], et en Bash avec l'écriture de scripts utilitaires. Pendant deux mois et demi, j'ai pu apprendre et me familiariser avec le milieu professionnel, améliorer ma communication, ma socialisation et mon travail dans un nouvel environnement tel que l'open space, ce qui m'a apporté une expérience précieuse. D'un point de vue personnel, ce stage était l'occasion pour moi de faire l'expérience des entretiens d'embauche et de la recherche d'emploi. Il m'a permis de dissiper beaucoup de doutes sur le monde du travail ainsi que sur le marché de l'informatique. Grâce à cette expérience, j'ai pu élargir ma vision du monde professionnel.

25. Structure logicielle qui fournit des outils pour simplifier le processus de développement
26. Un framework PHP pour le développement web
27. Un outil de test fonctionnel basé sur JavaScript

GLOSSAIRE

AJAX requête HTTP effectuée de manière asynchrone depuis une page web. 16

Cypress Un outil de test fonctionnel basé sur JavaScript. 1, 4, 18, 25

Docker Plateforme de conteneurisation qui permet de créer, déployer et exécuter des applications dans des conteneurs légers et portables. 1, 4, 7, 9

Dockerfile fichier de configuration utilisé pour créer une image Docker. 12

fork Copie d'un référentiel Git existant. 5

framework Structure logicielle qui fournit des outils pour simplifier le processus de développement. 1, 15, 18, 25

GitLab Outil de versionning, gardant un historique du développement. 9

Gitlab-CI Fichier permettant de définir des étapes de traitements à exécuter par des machines virtuelles dans gitlab. 9, 12

hook intercepte des événements/actions spécifiques et exécutent du code en conséquence. 15

Laravel Un framework PHP pour le développement web. 15, 18, 25

OJS Logiciel Open Journal System. 9, 12

runner Logiciel utilisé pour exécuter des tâches automatisées en réponse à des événements spécifiques. 5, 9, 12, 15

Smarty Système de templates utilisé pour séparer le développement du design d'une page web. 15

Vue.js Framework JavaScript pour développer des applications web. 15

Références

[1] Cypress. Documentation API. https://docs.cypress.io/api/table-of-contents. Consulté le 09 Juin 2024.

[2] Laravel. Documentation. https://laravel.com/docs/11.x. Consulté le 07 Juin 2024.

[3] Public Knowledge Project. cycle de vie d'une requête. https://docs.pkp.sfu.ca/dev/documentation/en/architecture-request. Consulté le 04 Juin 2024.

[4] Public Knowledge Project. PKP hooks documentation. https://docs.pkp.sfu.ca/dev/documentation/en/utilities-hooks. Consulté le 04 Juin 2024.

[5] Public Knowledge Project. PKP Plugin/Module documentation. https://docs.pkp.sfu.ca/dev/plugin-guide/en/. Consulté le 08 Juin 2024.

[6] Vue framework. Documentation et guide. https://vuejs.org/guide/introduction.html. Consulté le 29 Mai 2024.

[1] [2] [3] [4] [5] [6]

ANNEXES

VIII.1 Annexe A : Illustration du carnet de bord

Jour 4 18/04/2024

Pause réglementaire à 10H30 - 11H

J'ai discuté avec divers collègues sur la vie à Grenoble et notamment partagé un gouté autour de viennoiserie.

Mon travail

Correction d'erreurs sur la configuration d'OJS dans un docker via le script scriptConfigOjs.sh.
Le problème repose sur la configuration de l'environnement pour faire fonctionner les tests Cypress(Variable d'environnement, préparation aux tests, etc...).

Avancer vers niveau 2 du script de configuration (et ajout d'un dépôt GIT(versionning) personnel (branch)).
étape 1 : testez via un script github la config d'OJS.
étape 2 : testez un plugin via un script github.

J'ai ma propre branche de développement sur gitlab, avec un utilisateur ayant des droits développeurs me permettant certains privilèges comme pouvoir "fork" certains répertoires de MATHDOC pour y travailler.

J'ai mis en place les tests avec des "runners" qui me permette de tester mes fichiers de test "gitlab-ci.yml" à chaque push vers le répertoire gitlab.

cas ERREUR impossible de se connecter au docker pour pull une image via un gitlab-ci et un runner :

```
ERROR: Job failed: failed to pull image "gricad-registry.univ-grenoble-alpes.fr/test-stagia
```

1. aller dans le répo ou l'image a été créer(qui contient le gitlab-ci.yml qui crée l'image) > settings > IC/CD > Token Access > [add token] et rajouter le chemin complet d'un répertoire GITLAB ou vous voulez autoriser un pull d'image du registry.
2. ou Désactiver le Token Access.

FIGURE 9. Partie du carnet de bord (première semaine)

VIII.2 Annexe B : Illustration du plugin userMailPlugin

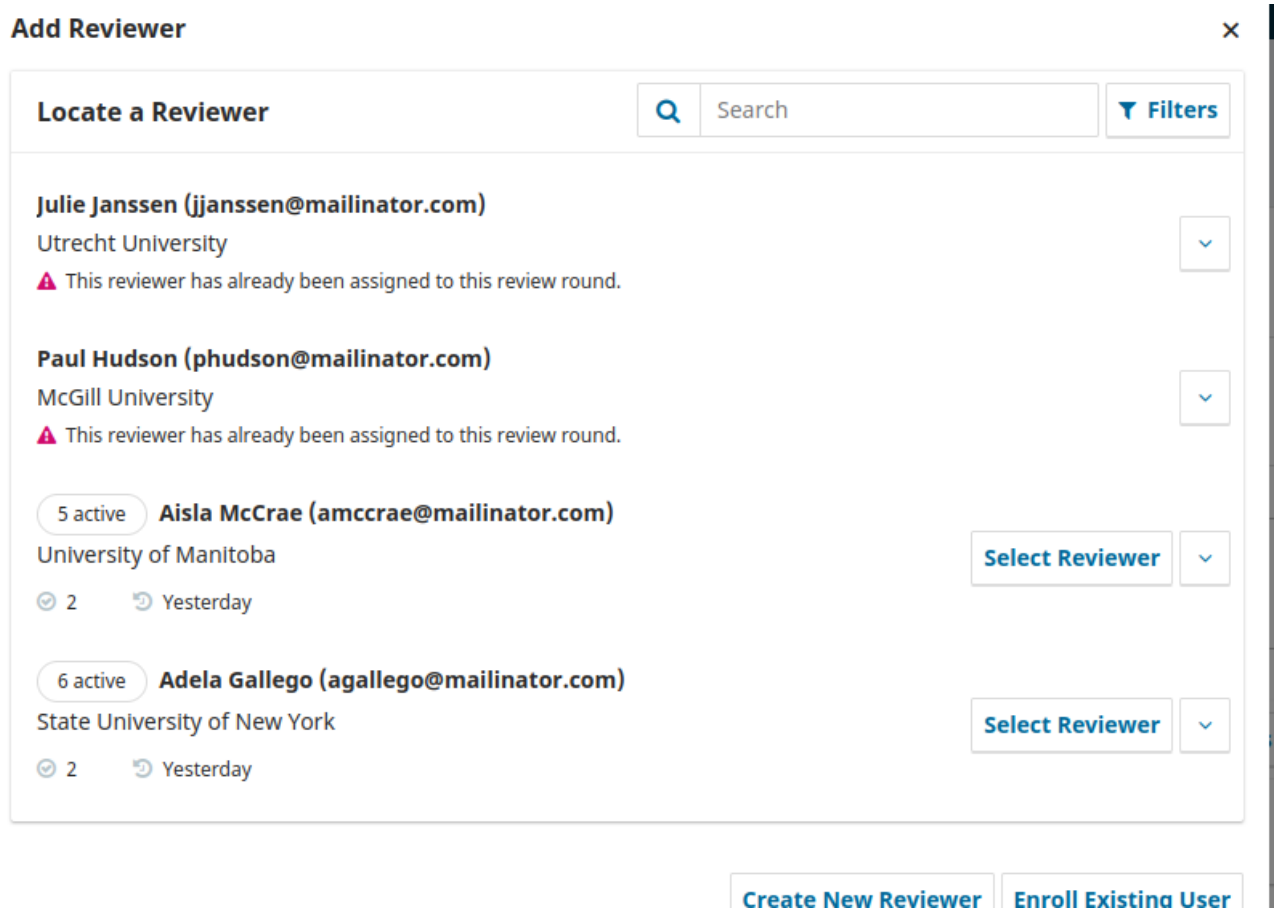

FIGURE 10. Illustration du plugin email actif

VIII.3 Annexe C : Illustration du plugin readOnlyPlugin

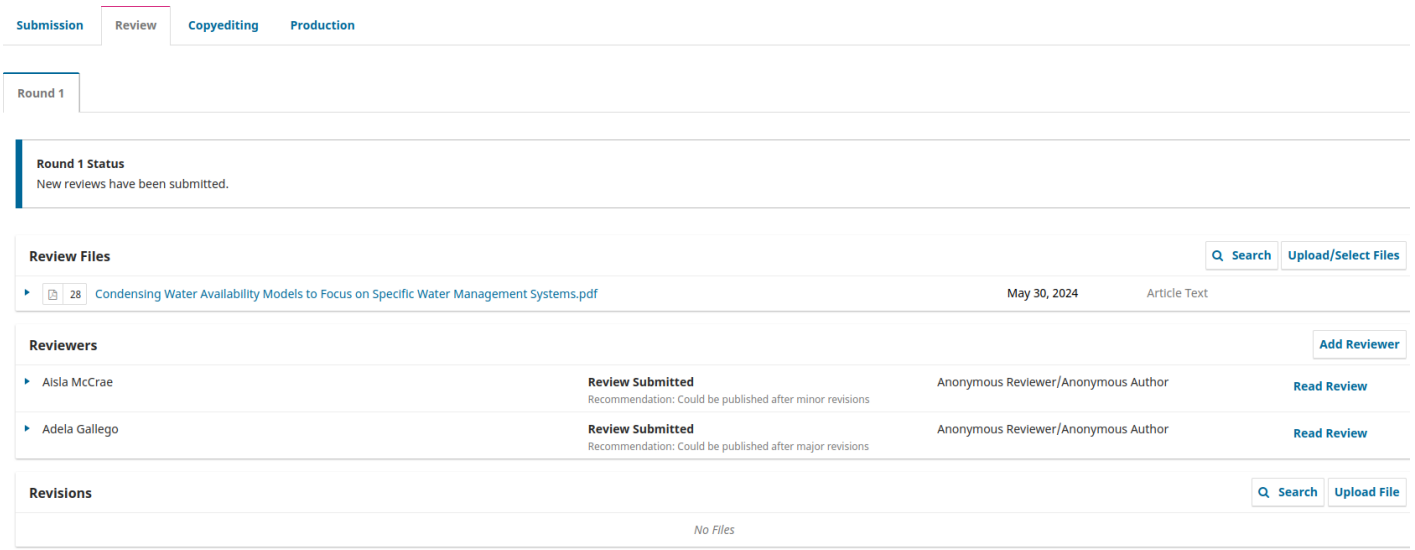

FIGURE 11. Illustration du plugin read only désactivé

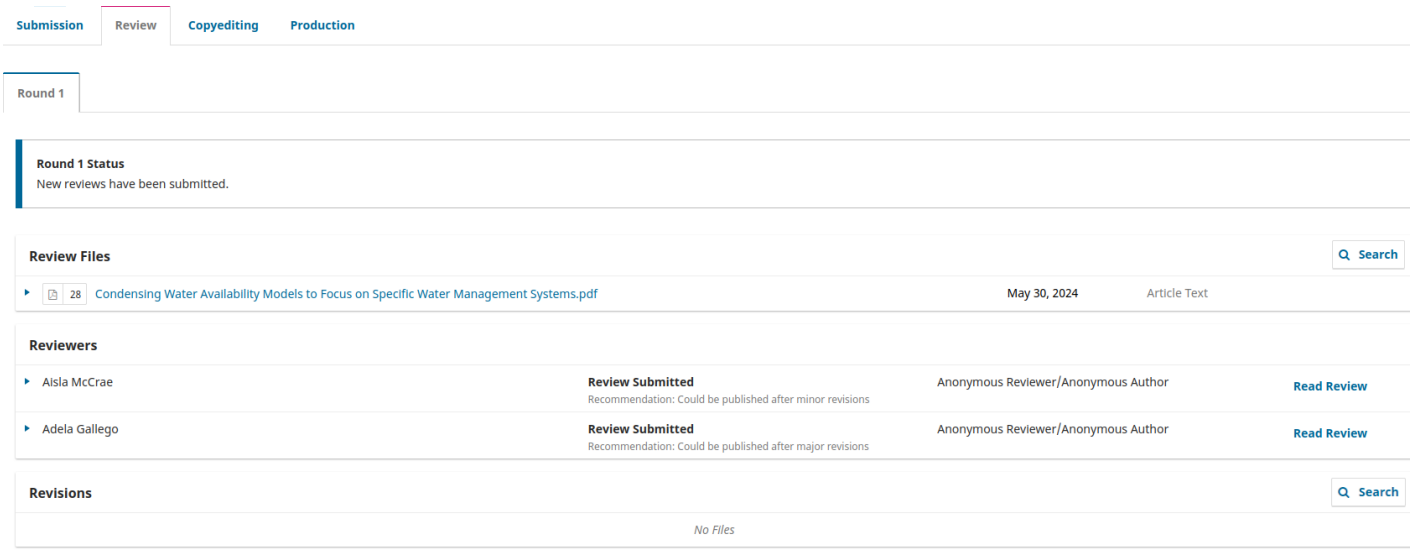

FIGURE 12. Illustration du plugin read only activé

VIII.4 Annexe D : Morceau de code d'interception et de modification d'une requête AJAX

```php
private function addEmailToReviewerList($selectReviewerListData)
{
    foreach ($selectReviewerListData['components']['selectReviewer']['items'] as &$item) {
        $nameAndEmail = $item['fullName'] . ' (' . $item['email'] . ')';
        $item['fullName'] = $nameAndEmail;
    }

    return $selectReviewerListData;
}
```

FIGURE 13. Méthode modifiant le contenue d'une requête AJAX

Résumé du rapport de stage

Mon stage en informatique, du 15/04/2024 au 21/06/2024, s'est déroulé à Mathdoc, unité mixte du CNRS et de l'UGA spécialisée dans les services documentaires en mathématiques. L'objectif était de développer des fonctionnalités supplémentaires pour le logiciel Open Journal Systems (OJS). J'ai travaillé sur des tests fonctionnels en JavaScript avec l'outil Cypress [28] et sur le développement de plugins en PHP avec le framework [29] Laravel [30], Vue.js* et l'outil de template Smarty. J'ai également créé des scripts en Bash pour déployer des environnements de travail avec Docker adaptés aux besoins des développeurs sur OJS*. Ce stage m'a permis d'améliorer mes compétences techniques et de découvrir le milieu professionnel ainsi que ses exigences.

Mots-clés : OJS, PHP, JavaScript, Cypress, Docker, développement web, tests fonctionnels, plugins, CNRS Mathdoc

The Abstract

Test Driven Development of Plugins with Functional Tests inside OJS Software

Kersual Arnaud

abstract: From April 15, 2024, to June 21, 2024, I completed a web development internship at Mathdoc, a joint unit between the CNRS and UGA specializing in documentary services for mathematics and academic journals. The main objective was to develop additional features for the Open Journal Systems (OJS) software. My tasks included creating functional tests in JavaScript using the Cypress tool, developing plugins in PHP with the Laravel framework, Vue.js framework and the Smarty template engine. Additionally, I designed Bash scripts to deploy working environments with Docker, according to the developers' needs on OJS. I also participated in weekly meetings to discuss progress and coordinate with other team members. This internship significantly improved my technical skills in JavaScript, PHP, and Bash, and provided valuable experience in a professional environment, enhancing my technical skills to be more versatile in projects and improved my understanding of project management and teamwork within a professional context.

Keywords: OJS, PHP, JavaScript, Cypress, Docker, développement web, tests fonctionnels, plugins, CNRS Mathdoc, teamwork

28. Un outil de test fonctionnel basé sur JavaScript
29. Structure logicielle qui fournit des outils pour simplifier le processus de développement
30. Un framework PHP pour le développement web