DOCUMENTATION ET ANALYSE DU WORKFLOW CI/CD POUR BOBAPP

Ce document a pour objectif de présenter le travail de mise en place d'un workflow d'Intégration Continue et de Déploiement Continu (CI/CD) pour le projet BobApp, de proposer des indicateurs de performance (KPIs) pour le suivi de la qualité, et d'analyser les premières métriques ainsi que les retours des utilisateurs pour définir les prochaines actions.

I. DESCRIPTION DES ÉTAPES DU WORKFLOW CI/CD

Le workflow, configuré avec GitHub Actions, est conçu pour automatiser la validation, l'analyse et le déploiement de l'application. Il se déclenche à chaque push ou pull request sur la branche main.

Voici les étapes principales :

1. Détection des changements
2. Le build et les tests BackEnd
3. Le build et les tests FrontEnd
4. Envoie des données à SonarCloud
5. Déploiement du du Backend et FrontEnd sur Docker Hub

1 Détection des changements

Objectif :
Détecter les modifications apportées aux dossiers front et back pour n'exécuter que les jobs pertinents.

Outil :
dorny/paths-filter

2 Le build et les tests BackEnd

Objectif:
Compiler le code source du back-end, exécuter les tests unitaires et générer le rapport de couverture de code.

Déclenchement :
Uniquement si des changements sont détectés dans le dossier back.

Actions :
Mise en place de Java 11.
Lancement des tests avec Maven (mvn -B test).
Sauvegarde du rapport de couverture Jacoco (jacoco.xml) en tant qu’artefact.

3 Le build et les tests FrontEnd

Objectif :
Installer les dépendances du front-end, exécuter les tests et générer le rapport de couverture.

Déclenchement :
Uniquement si des changements sont détectés dans le dossier `front`.

Actions :
Mise en place de Node.js 18.
Installation des dépendances (npm install).
Lancement des tests avec Karma/Jasmine en mode headless (npm test -- --no-watch --no-progress --browsers=ChromeHeadless --code-coverage).
Sauvegarde du rapport de couverture LCOV (lcov.info) en tant qu'artefact.

4 Envoie des données à SonarCloud

Objectif :
Analyser la qualité du code (back-end et front-end) pour détecter les bugs, vulnérabilités, "code smells" et mesurer la couverture de tests.

Déclenchement :
Après les étapes de build et de test, si celles-ci ont réussi.

Actions :
Récupération des rapports de couverture (jacoco.xml et lcov.info).
Lancement de l'analyse avec SonarCloud. Le "Quality Gate" de SonarCloud doit passer pour que le workflow continue.

5 Déploiement du du Backend et FrontEnd sur Docker Hub

Objectif :
Construire les images Docker pour le back-end et le front-end et les publier sur Docker Hub.

Déclenchement :
Uniquement pour les push sur la branche main et si l'analyse SonarCloud a réussi.

Actions :
Connexion à Docker Hub.
Build de l'image Docker à partir du Dockerfile correspondant.
Push de l'image sur Docker Hub avec le tag « latest ».

II. PROPOSITION DE KPIS (INDICATEURS CLÉS DE PERFORMANCE)

Pour garantir une qualité de code maintenable et évolutive, je propose de suivre les KPIs suivants, à intégrer dans le "Quality Gate" de SonarCloud :

1. Taux de Couverture de Code (Coverage) : supérieur à 80%

Description : Ce KPI mesure le pourcentage de lignes de code qui sont exécutées par les tests automatisés. Un taux élevé assure qu'une grande partie du code a été testée, réduisant ainsi le risque de régressions.

Seuil proposé : 80%. C'est un standard de l'industrie qui offre un bon équilibre entre l'effort de test et la confiance dans le code. Nous commencerons avec ce seuil pour le nouveau code ajouté.

2. Aucun Nouveau "Blocker Issue »

Description : Les "Blocker Issues" sont des problèmes critiques identifiés par SonarCloud, tels que des fuites de mémoire ou des vulnérabilités majeures, qui peuvent entraîner des pannes ou des failles de sécurité.

Seuil proposé : 0. Il est impératif qu'aucun nouveau code ne vienne introduire de problème aussi grave. Toute Pull Request ajoutant un "Blocker Issue" doit être refusée.

III. ANALYSE INITIALE

Métriques Actuelles

Après la première exécution complète de la pipeline, voici les métriques initiales remontées par SonarCloud.

Taux de Couverture (Coverage) :

Back-end : 0%
Front-end : 66.7%
Total : 18.5%

Fiabilité : Note D
Bugs : 1 : C'est le problème le plus critique qui cause probablement l'instabilité de l'application.

Sécurité : Note A
Vulnérabilités : 0 : Le code ne présente pas de faille de sécurité exploitable.

Dette Technique : 45 minutes Note A : 10
Code Smells : 10. Ces problèmes sont mineurs, le code reste donc globalement de bonne qualité.
Dette Technique : 45 minutes (Note : La dette technique est le temps estimé pour corriger les 10 "Code Smells". La note 'A' indique que ce temps est très court.)

Analyse des Retours Utilisateurs

Les avis des utilisateurs sur les stores sont une source précieuse d'information. Voici un résumé des points pertinents :

Avis 1 : "L'application est lente et plante souvent, dommage car les blagues sont bonnes."

Avis 2 : "L'application ne marche plus depuis la dernière mise à jour."
Avis 3 : "Impossible de charger la blague du jour, l'application reste bloquée sur un écran blanc."

Ces retours mettent en évidence deux problèmes majeurs :
Fiabilité : Des plantages fréquents et des régressions après les mises à jour.
Performance : Lenteur de l'application et problèmes de chargement.

III. PROCHAINES ACTIONS ET PRIORITÉS

En croisant l'analyse des métriques et les retours utilisateurs, voici les actions prioritaires à mener :

Priorité 1 : Corriger les Bugs et Vulnérabilités critiques remontés par SonarCloud.
Justification : Les problèmes identifiés par Sonar (notamment les "Bugs" et "Vulnérabilités") sont probablement la cause directe des plantages et de l'instabilité rapportés par les utilisateurs. Leur correction améliorera immédiatement la fiabilité de l’application.

Priorité 2 : Augmenter la couverture de test.
Justification : Le faible taux de couverture actuel explique pourquoi des régressions apparaissent après chaque mise à jour ("L'application ne marche plus depuis la dernière mise à jour"). En ajoutant des tests pour les fonctionnalités existantes, nous créerons un filet de sécurité qui empêchera de casser ce qui fonctionne déjà. Cela répond directement au besoin de fiabiliser les déploiements.

Priorité 3 : Analyser et optimiser les performances.
Justification : Les problèmes de lenteur et de chargement nécessitent une investigation plus poussée (analyse de requêtes, temps de réponse du back-end, etc.). Une fois la stabilité améliorée, des optimisations pourront être mises en place pour rendre l'expérience utilisateur plus fluide.

Ce plan d'action, soutenu par le workflow CI/CD, permettra d'améliorer itérativement la qualité de BobApp, de rassurer les utilisateurs et d'encourager les contributions externes.