Plan Stratégique d'Amélioration et de Fiabilisation de BobApp

Mise en Place d'une Chaîne d'Intégration et de Livraison Continues (CI/CD)

Ce document présente un plan d'action centré sur l'implémentation d'une démarche **CI/CD** pour garantir la fiabilité du système BobApp, améliorer la collaboration *open source* et regagner la confiance des utilisateurs.

---

1. Diagnostic de l'Application BobApp

A. Synthèse des Indicateurs de Qualité SonarQube

L'analyse du code source révèle des faiblesses critiques qui expliquent la difficulté actuelle à corriger les bugs et à maintenir le système :

Catégorie	Score	Nombre de Problèmes	Interprétation et Préconisation
Sécurité	A	0	**Excellent.** Aucune vulnérabilité critique détectée.
Fiabilité	A	1	**Très bon.** Un seul bug potentiel identifié.
Maintenabilité	**D**	**11 *Code Smells***	**Faible.** La note D indique que le code est difficile à maintenir. **Priorité :** Ces 11 problèmes doivent être corrigés pour réduire la **dette technique** existante.
Couverture Tests	**46.9%**	Faible	Seules 46.9% des lignes de code sont couvertes. Cet indicateur est bas et est une cause de l'introduction de **régressions**.
Duplications	0.0%	Bon	**Excellent.** Aucune duplication de code n'a été détectée.
Security Hotspots	Vigilance	2	**Vigilance :** Deux points chauds nécessitent un examen manuel pour confirmer l'absence de failles.

B. Impact des Retours Utilisateurs (Note Moyenne : 2.0/5)

Les problèmes techniques se traduisent directement par la perte d'utilisateurs et un manque de confiance :

Grief Utilisateur	**Origine du Problème**	**Conséquence**
Bug Bloquant (« le bouton tourne et fait planter mon navigateur ! »)	**Manque de tests** et de validation avant déploiement.	**Perte immédiate** de l'utilisateur (bug P0).
Bug Ancien (« encore présent après deux semaines ! »)	**Dette Technique** (Maintenabilité D) et processus de correction lent.	**Manque de réactivité** et sentiment d'abandon.
Manque de Contenu/Réactivité	Déploiements manuels, lents et non sécurisés.	**Perte de rétention** et désintérêt pour le service.

---

2. La Solution : Un Pipeline CI/CD Robuste

L'automatisation du cycle de vie du développement est la seule solution pour garantir la qualité et soutenir la collaboration *open source*.

A. Réponse aux Griefs par l'Automatisation

Objectif Visé	Solution par le CI/CD	Bénéfice Direct
Fiabilité et Stabilité	**Intégration Continue (CI)** : Exécution automatisée des tests et de l'analyse qualité (Sonar).	Garantie que le nouveau code **n'entraîne pas de régression** et ne **plombe pas la dette technique**.
Rapidité des Mises à Jour	**Livraison Continue (CD)** : Déploiement automatisé des images Docker sur Docker Hub.	Mises à jour rapides, fréquentes et sans erreur, répondant au manque de réactivité.
Collaboration Améliorée	**Feedback Immédiat** sur les PR via GitHub Actions.	L'automatisation réduit la friction et encourage la participation.

B. Le Rôle de la *Quality Gate*

La **Quality Gate** est le point de contrôle obligatoire du pipeline CI/CD. Elle utilise les KPIs pour décider si une modification est **acceptable** pour la branche principale ou si elle doit être **bloquée**.

---

3. Proposition des Indicateurs Clés de Performance (KPIs)

Pour garantir que le code fusionné respecte les exigences minimales, nous proposons d'établir les seuils suivants pour la **Quality Gate** SonarCloud :

KPI	Objectif	Seuil Proposé	Justification
1. Taux de Couverture Minimum	Améliorer la Fiabilité	**Minimum 60%** (sur le code global)	Seuil minimal pour commencer à sécuriser les fonctionnalités. **Note :** L'objectif à terme dans le pom.xml est fixé à 80%.
2. Zéro Problème Bloquant	Sécurité et Stabilité	**Égal à 0** (sur le *nouveau* code)	Empêche l'intégration de toute faille critique ou bug majeur.
3. Zéro Nouveau *Code Smell*	Maintenabilité Future	**Égal à 0** (sur le *nouveau* code)	C'est le rempart contre l'augmentation de la dette technique, essentiel pour améliorer le score **D** actuel.
4. Zéro Vulnérabilité Critique	Sécurité des Images	**Égal à 0** (sur l'image Docker)	Bloque le déploiement d'artefacts non sécurisés (vulnérabilités *High* ou *Critical*).

---

4. Plan d'Action Prioritaire pour la Fiabilité

1. **Mise en place de la Quality Gate (CI) :** Déployer immédiatement le workflow GitHub Actions pour **bloquer** toute nouvelle Pull Request qui ne respecte pas les 3 premiers KPIs ci-dessus (Tests, Code Smells, Bloquants).
2. **Sécurité des Artefacts (CI) :** Intégrer un outil de scan Docker pour **bloquer le *build*** de toute image (Front-end et Back-end) présentant des vulnérabilités de sévérité Élevée ou Critique.
3. **Correction :** Prioriser la correction des **11 *Code Smells*** et du **bug bloquant P0** (« Le bouton fait planter ») pour regagner la confiance des utilisateurs.
4. **Déploiement (CD) :** Activer la **Livraison Continue** sur Docker Hub, en s'assurant que l'artefact livré est toujours validé par le CI.