Explication et Analyse du Travail CI/CD

1. Introduction

Dans le cadre du projet, j'ai mis en place une pipeline CI/CD complète permettant d'automatiser les tests, l'analyse de qualité du code, la construction et le déploiement des images Docker. Cette démarche assure un processus de développement fiable, itératif et conforme aux bonnes pratiques DevOps.

---

2. Explication des GitHub Actions

J'ai défini plusieurs workflows GitHub Actions pour automatiser le cycle de vie du projet :

A.Workflows des tests

Il y a deux fichiers pour les tests :

- Tests Frontend : frontend-tests-coverage.yml
- Tests Backend : backend-tests-coverage.yml

Frontend

Le workflow commence par :

- Nom du workflow : name
- Déclencheurs : on définit les branches sur lesquelles les push et pull_request activent le workflow.
- Jobs : la section jobs décrit les actions à exécuter. Ici, un seul job Frontend-test :
  - OS : tourne sous Ubuntu (runs-on).
  - Dossier : dans le dossier front (defaults).

- Version Node.js : utilise la version 16 (strategy).
- Étapes (steps) :
    1. Checkout du repository
    2. Installation de Node.js
    3. Installation des dépendances : npm install
    4. Exécution des tests et génération des rapports de couverture

Backend

Même procédure, adaptée à :

- Version du JDK (Java)
- Exécution des tests via Maven

B.Workflows d'analyse de code

Fichiers :

- sonarcloud_front.yml
- sonarcloud_back.yml

Étapes :

1. Connexion à SonarCloud (via secrets GitHub)
2. Exécution des analyses
    - Front avec Node.js
    - Back avec Maven

Automatisation et intégration

- Les workflows s'exécutent à chaque commit.
- Un commit échoue si un workflow échoue.

C.Workflows CI/CD

Le fichier docker.yml déploie les images Docker sur Docker Hub.

Étapes :

1. Vérifications préliminaires
2. Connexion à Docker Hub
    - Récupération des secrets DOCKER_USERNAME et DOCKER_PASSWORD
3. Construction et envoi des images Docker

- Création des images pour le front et le back
- Push des images sur Docker Hub

---

3. KPIs proposés

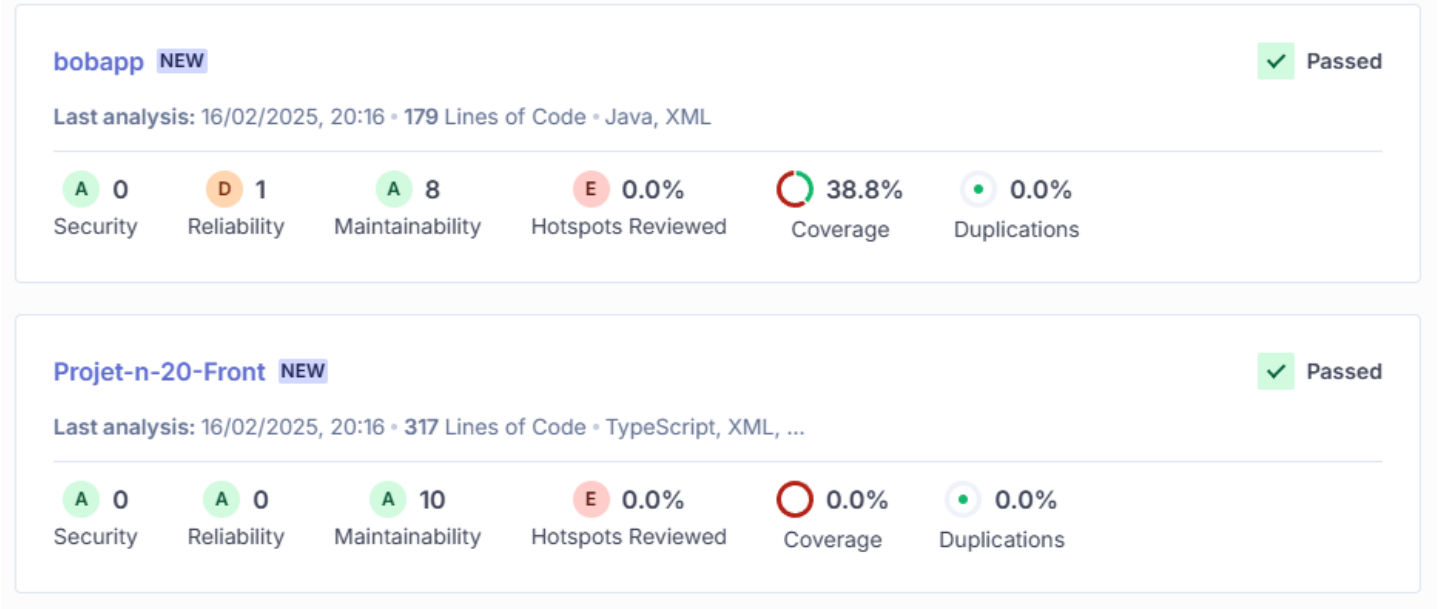
A. Couverture du code
   - Minimum 80% de couverture pour le front et le back.

B. Qualité du code (SonarCloud)
   - Bugs : note A (0 bugs)
   - Vulnérabilités : note A
   - Revue de sécurité : note A (100% Hotspots Reviewed)
   - Maintenabilité : note A (Code smells)
   - Issues :
     - 0 issues High
     - Max 3 issues Medium
     - Issues Low non restreintes mais minimisées
   - Blocker Issues : 0 (exemple : fuite de mémoire)

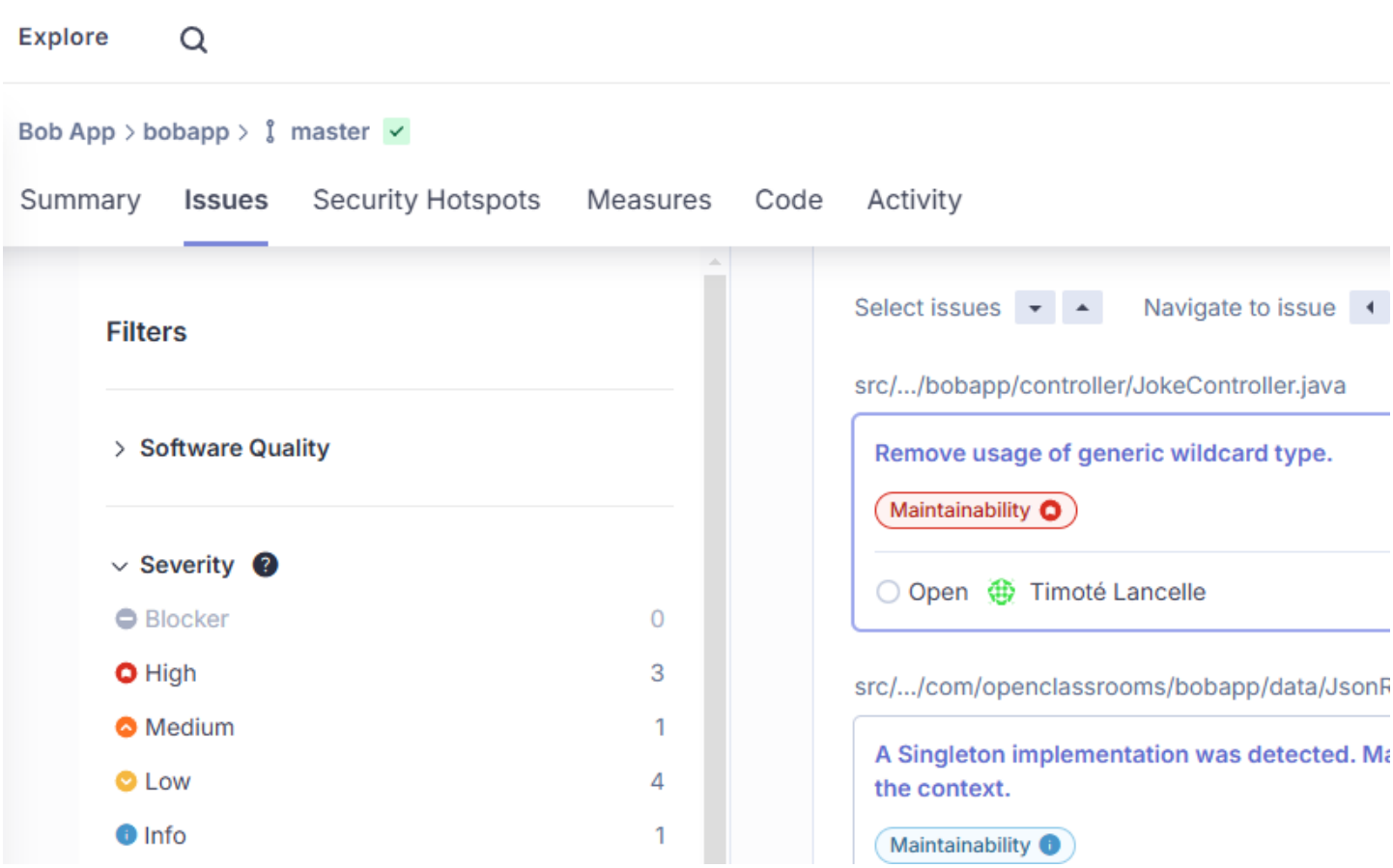
---

4. Analyse des métriques

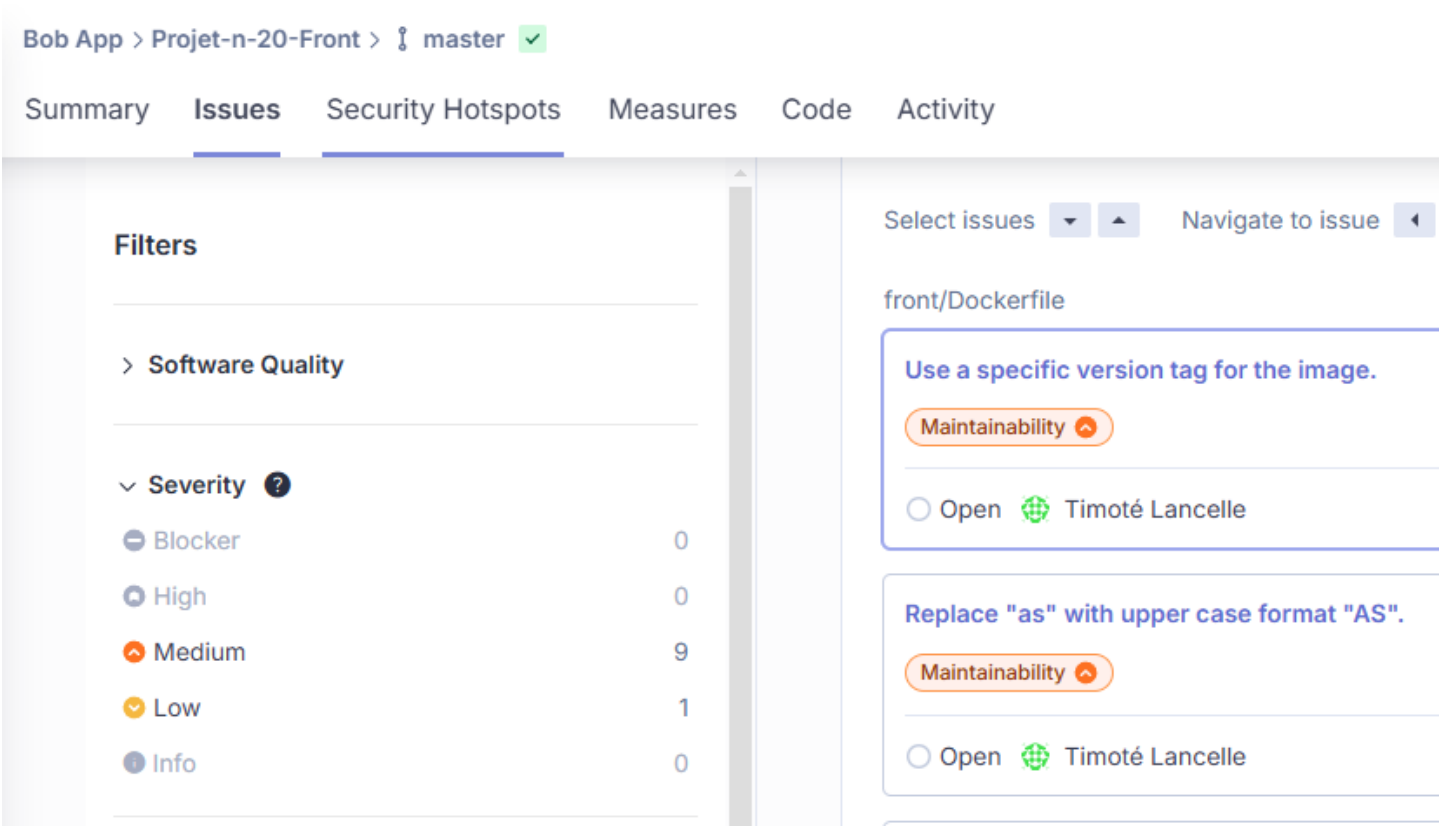
A. Analyse générale



B.Analyse issues backend



C.Analyse issues frontend

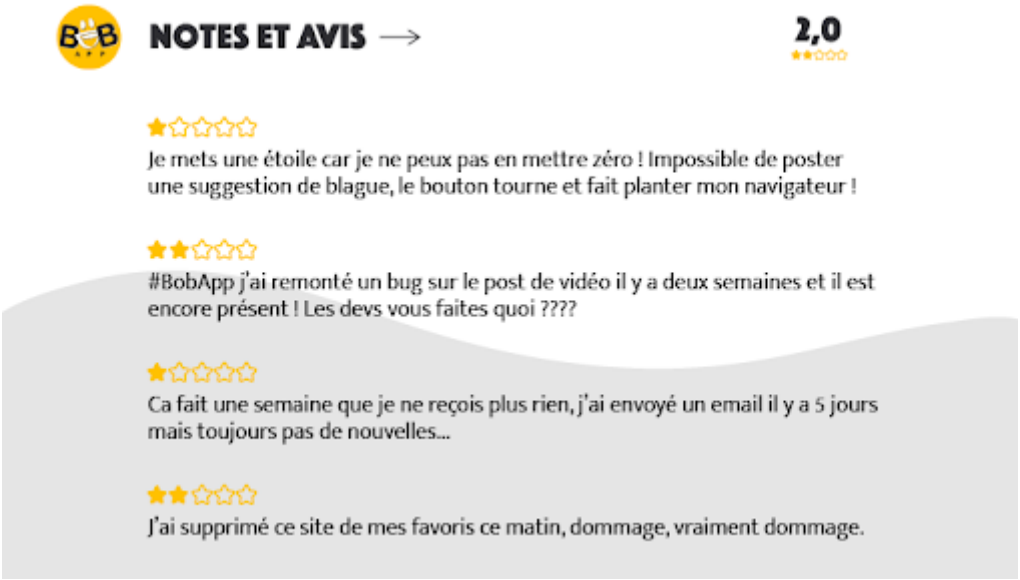


Métriques	Backend	Frontend
Security (Vulnérabilités)	0 (note A)	0 (note A)
Reliability (Bugs)	1 (note D)	0 (note A)
Maintainability (Maintenabilité)	8 (note A)	10 (note A)
Hotspots Reviewed (Revue de sécurité)	0.0 % (note E)	0.0 % (note E)
Coverage (Couverture)	38.8 %	0.0 %
Duplications	0.0 %	0.0 %
Issues (High, Medium, Low)	3 high, 1 medium, 4 low	0 high, 9 medium, 1 low

---

5. Retours utilisateurs et améliorations

Les retours utilisateurs ont mis en avant plusieurs points :



- **Problème de lenteur sur certaines pages du front** → Optimisation des requêtes API nécessaire.
- **Affichage non responsive sur certains écrans** → Correction CSS et tests sur plusieurs résolutions à prévoir.
- **Besoin de logs plus détaillés en cas d'erreur** → Amélioration de la gestion des erreurs backend.

Améliorations à venir :

- Correction des vulnérabilités détectées par SonarCloud.
- Optimisation des performances front-end.

Ajout de logs détaillés et monitoring applicatif.

---

6. Conclusion

- Les workflows fonctionnent correctement et automatisent les tâches de test, d'analyse et de déploiement.
- Les KPIs doivent être améliorés : couverture de code insuffisante et problèmes de qualité du code.
- Des bugs utilisateurs critiques sont recensés, priorité à leur correction.