École Nationale d'Ingénieurs de Carthage

RAPPORT DE TEST STATIQUE

Analyse de la Qualité du Code (Backend & Frontend)

Équipe QA :

Mohamed Dhia Eddine Thabet *Test Manager*
Mohamed Aziz Dridi *Tester*

Novembre 2025

1 Synthèse de la Qualité (Executive Summary)

L'analyse statique a été réalisée à l'aide de **SonarQube** pour évaluer la dette technique, la fiabilité et la sécurité du code avant toute exécution.

Module	Fiabilité	Sécurité	Maintenabilité
Backend (Java)	**Grade C (Critique)**	Grade A	Grade A
Frontend (React)	**Grade B (Majeur)**	Grade A	Grade A

TABLE 1 – Tableau de bord Qualité SonarQube

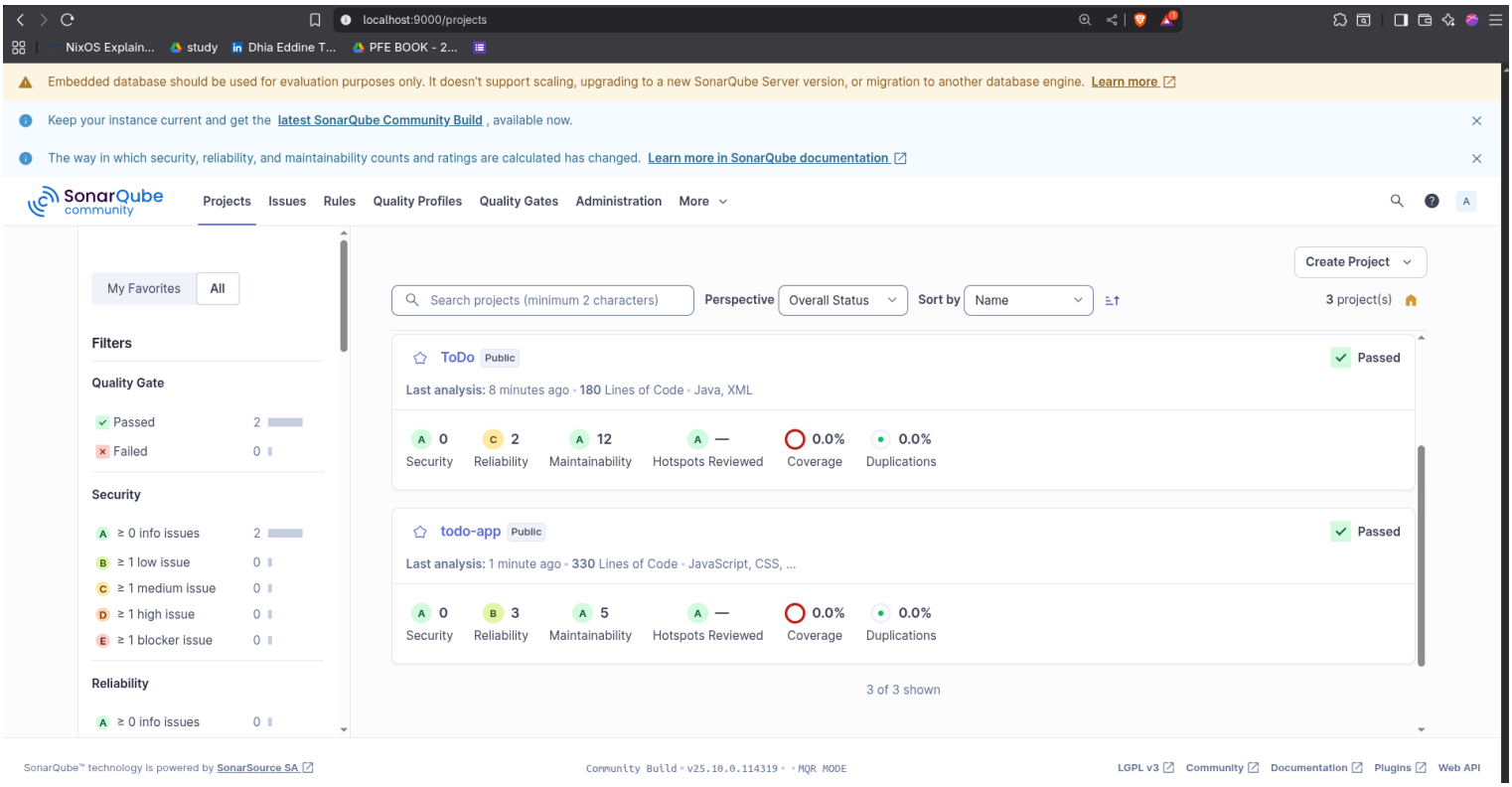

FIGURE 1 – Preuve d'audit SonarQube

2 Analyse Approfondie : Backend (Java)

Une revue manuelle combinée à l'analyse automatisée a révélé une faille critique de conception.

2.1 2.1 Violation de Thread-Safety (Critique)

Composant : `TodoRepository.java`

- **Problème :** Utilisation d'un compteur simple (`Integer idCounter`) incrémenté via `++` dans un Bean Singleton.
- **Impact ISTQB :** *Défaut de Fiabilité*. En cas d'accès concurrent, deux tâches peuvent hériter du même ID, entraînant une perte de données.
- **Preuve Code :**

```
@Repository
public class TodoRepository {
	private Integer idCounter = 0; // NON ATOMIQUE
	...
	item.setId(idCounter++); // DANGER
}
```

FIGURE 2 – Snippet du code incriminé

3 Analyse Approfondie : Frontend (React)

3.1 3.1 Gestion des Hooks (Fiabilité)

Composant : `App.js`

- **Code Smell :** Le hook `useEffect` possède une dépendance sur l'état qu'il modifie lui-même.
- **Risque :** Boucle de rendu infinie (Infinite Re-render) pouvant bloquer le navigateur client.

4 Conclusion & Recommandations

Le code est propre et bien formaté, mais la gestion de la concurrence côté serveur est **inacceptable pour une mise en production**.

Actions requises :

1. Remplacer `Integer` par `AtomicInteger` dans le Repository.
2. Refactoriser le `useEffect` du Frontend.

Document validé par l'équipe QA de l'ENICarthage.