TD4 qualité.

1)
après analyse de sonar, aucun « issue était a noter » cependant, dans « security Hotspots » Une note a était prise, il m'a alors fallu un peu moins de 5min pour comprendre et corriger cette erreur.

2) la librairie présente plusieurs problèmes de qualité et de maintenabilité. Voici les points noter avec les 73 problèmes identidfier :

Types de Problèmes :

complexité Cognitive :
Des méthodes affichent une complexité cognitive élevée donc très compliquer a maintenir

Noms de constantes :
pas de respect des conventions des nom de variable etc

Gestion des erreurs :
La gestion des exceptions est mauvaise

Doubles déclarations :
plusieurs duplication de code sont présent

Vulnérabilités :

Des vulnérabilités critiques ont été détectées, notamment en ce qui concerne l'utilisation d'algorithmes de chiffrement.

La qualité du code de cette librairie semble insuffisante, avec un besoin urgent d'améliorations pour répondre aux normes de maintenabilité et de sécurité. Les problèmes identifiés indiquent une architecture de code peu adaptée et une gestion des exceptions non optimisée.

Protocole d'amélioration pour la librairie

L'analyse de la librairie met en lumière plusieurs points à améliorer. D'abord, il serait bon de simplifié certaines méthode qui sont devenue trop complexes, ce qui rend le code difficile a suivre. Extraire des blocs de code imbriqués dans des méthodes

séparées aiderait à rendre le tout plus lisible et plus réutilisable. De plus, respecté des conventions de nommage pour les constantes pourrait donner un peu plus de cohérence au code en général.

Il est aussi important de se débarrasser des exception inutilisées et de crée des exceptions spécifiques au lieu d'utiliser des exceptions génériques. Cela aiderait à clarifier comment les erreurs sont gérées. Enfin, réduire le nombre de paramètres dans les méthodes serait vraiment utile ; regrouper des paramètres connexes dans des objets pourrait alléger les signatures de méthodes. En gros, ces amélioration rendraient le code plus facile à maintenir, plus robuste et plus sécurisé, tout en facilitant les futures mises à jour de la librairie.

3)

Problèmes Identifiés

Constantes mal nommées :

Tout comme le fichier précédent, des nommages ne sont pas en norme.

Injection de champs au lieu d'injection par constructeur :

- Fichiers :
  - AccountRepository.java
  - AccountRessource.java
  - AccountServiceImpl.java
  - CdiProducer.java

Sonar déconseille ce genre de pratique du a la maintenabilité du code

Code commenté non utilisé :

- Fichiers :
  - AccountDto.java
  - TransactionDto.java

Il faut décommenter pour que le code marche afin que les return face leur travaille.

Déclarations vides :

- Fichier : AccountRessource.java

Une déclaration manque ce qui empeche le bon fonctionnement de code.

En appliquant ces solutions, le code de la librairie quarkus-hexagonal deviendra plus conforme aux meilleures pratiques de développement, améliorera sa maintenabilité, sa lisibilité et sa testabilité.

pour les test, on aurait pu faire comme ca :

```
@Test
public void testToString() {
    AccountDto accountDto = new AccountDto();
    String expectedString = "";
    assertEquals(expectedString, accountDto.toString(), "retourne une liste");
}
```

Grâce a ce test nos deux fonctions aurait était détecter comme ne retournant rien et nous aurions pu corriger cette erreur