Rapport des Tests - SR10

Léopold Chappuis - Alexandre Eberhardt

1 Introduction

Ce document présente un compte-rendu des tests effectués sur les modèles nous permettant d'extraire et d'importer des données de la base de données ainsi que les routes dans notre application développée avec Express.js. Les tests ont été réalisés en utilisant le framework Jest et Supertest.

2 Parties Testées

Les tests ont été effectués sur les fonctions principales de chaque modèle, ainsi que sur les routes de l'application, notamment :

- **Offer Model** : `read`, `readAll`, `offreDetail`, `already`, `postule`, `enterFile`, `getCandId`, `getFile`, `areValid`.
- **passwd** : `generateHash`, `comparePassword`.
- **Recruteur modèle** : `readCandidatures`, `readNewRecruteur`, `readAllOffres`, `getAllCandidats`, `getAllCandidatsVerified`, `readAllRequests`, `getOrgForRecruteur`, `getNewApplicantsRecruters`, `quitOrg`, `creerFichePoste`, `creerOffre`, `getSiret`, `getIdOffre`, `offreOrga`, `getIntitule`, `modifyCand`, `disableOffre`, `modifyIntitule`, `modifyRTravail`, `modifySalMin`, `modifySalMax`.
- **session** : `init`, `createSession`, `isConnected`, `deleteSession`.
- **Users modèle** : `readall`, `applied`, `areValid`, `read`, `already`, `alreadyadmin`, `alreadyrecruteur`, `alreadyDetail`, `verifsiret`, `readById`, `create`, `update`, `unpostule`, `supporga`, `findorga`, `unaskadmin`, `unaskrecruteur`, `makeAdmin`, `makeRecruteur`, `addaddress`, `createorg`, `getAllFromCand`, `deleteFile`.
- **Routes de l'API** : Authentification, Utilisateurs, Recruteurs, Offres.

3 Règles de Test Spécifiées

Les règles de test spécifiées pour chaque fonction incluent :

- Vérification des résultats attendus pour les entrées valides.

- Gestion des erreurs pour les entrées invalides ou en cas de défaillance de la base de données.
- Utilisation de mocks pour isoler les fonctions des dépendances externes, notamment les appels à la base de données.
- Vérification des interactions avec les autres modèles ou fonctions.
- Tests des réponses des routes pour vérifier les statuts HTTP appropriés.

4 Taux de Couverture

Le taux de couverture des tests a été mesuré en termes de couverture des lignes, des branches, des fonctions et des instructions. Voici un résumé du taux de couverture pour chaque module :

- **Modèle Offer** : 100%
- **passwd** : 100%
- **Modèle Recruteur** : 100%
- **session** : 100%
- **Modèle Users** : 100%
- **Routes de l'API** : 92%

5 Conclusion

Les tests réalisés couvrent l'ensemble des fonctionnalités principales des modules de l'application ainsi que les routes critiques. Les résultats des tests montrent une bonne couverture du code et une gestion efficace des erreurs. Les tests automatisés permettent de garantir la fiabilité et la robustesse de l'application.