Depuis longtemps, je souhaitais mettre en place des tests unitaires pour améliorer la qualité de nos développements. Actuellement, nos processus de test se concentrent principalement sur des séries de tests manuels, notamment des tests unitaires, d'intégration et de qualification du produit.

Récemment, j'ai entrepris la mise en place de cas de tests avec Oracle SQL Developer en suivant le tutoriel "Performing a Unit Test of Your PL/SQL in Oracle SQL Developer 2.1, [Performing a Unit Test of Your PL/SQL in Oracle SQL Developer 2.1](https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/11g/r2/prod/appdev/sqldev/sqldev_unit_test/sqldev_unit_test_otn.htm)", qui est très bien documenté. Ce tutoriel m'a permis de réaliser un premier test sur une fonction que j'ai écrite.

La fonction que j'ai développée retourne la somme de deux nombres passés en paramètres. J'ai trouvé cette expérience très instructive. Je suis convaincu que cette démarche contribuera à réduire les erreurs et à renforcer la robustesse de notre application.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Puis documenté le test sous SQL Developer

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Sous SQL Developer

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

Une image contenant texte, capture d’écran, ligne, logiciel

Description générée automatiquement

Exécution du test (test OK)

Une image contenant texte, capture d’écran, Police, nombre

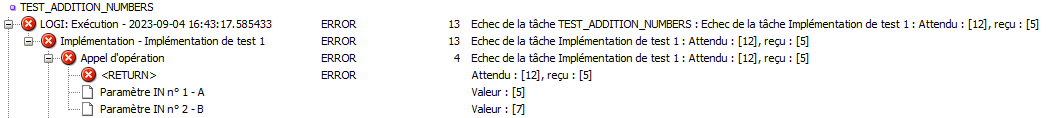
Description générée automatiquement

Si je modifie ma fonction et que le test ne répond plus à sa définition

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Exécution du test (test KO)



**C’est très bien tout ça, mais pourquoi tester ? Pourquoi ce n’est pas mis en place ?**

En réfléchissant à l'intégration de tests unitaires automatisés, il est essentiel de comprendre leurs avantages indéniables. Ces tests permettent de valider les comportements de chaque fonction de manière rigoureuse et systématique. Ils ouvrent également la porte à l'optimisation et à la refactorisation du code sans altérer son comportement, grâce à un mécanisme de détection instantanée des erreurs.

L'effet de bord positif est significatif : l'amélioration de la qualité du code se fait de manière moins risquée, tandis que le nombre d'erreurs diminue drastiquement lorsque les tests sont corrects. De plus, ils offrent une couverture complète de la solution, renforçant ainsi sa fiabilité.

Les tests automatisés sont également un pilier fondamental pour envisager le déploiement continu de la solution, en assurant une validation automatique lors de l'intégration. Cette approche permet de livrer plus sereinement, plus rapidement et d'apporter davantage de valeur à chaque étape du projet.

**Du coup pourquoi ne l’a-t-on jamais mis en place ?**

La mise en place de tests unitaires automatisés peut être entravée par plusieurs facteurs. Tout d'abord, l'architecture du produit actuel, basé sur la solution Forms Builder, peut limiter la possibilité d'automatiser les tests sans déplacer une partie du code côté base de données.

Ensuite, la culture des tests automatisés semble absente de l’entreprise, cela ne veut pas dire qu’il n’y a pas de culture de tests, mais qu’il n’y a pas de culture de tests automatisés au niveau des équipes de développement.

Enfin, l'héritage d'un projet basé sur des milliers de livrables peut compliquer le début de cette transition.

Néanmoins, si l'opportunité se présentait de reconstruire un projet avec une architecture différente, déplaçant le code côté base de données, cela ouvrirait la voie à la mise en place de ces méthodes de tests. Cela apporterait incontestablement de la qualité à la solution, tout en répondant aux exigences de l'innovation et de la rapidité de livraison dans le monde du développement logiciel moderne.