**Projet de développement web à base de composant et des services web**

# I- TDD (Test-Driven Development)

1. Définition

Le **TDD (Test-Driven Development)**, ou développement dirigé par les tests, est une méthodologie de développement de logiciels qui se concentre sur la création de tests automatisés avant d'écrire le code de l'application elle-même.

1. Principes du TDD

Voici les principes et étapes clés de TDD :

* **Rédiger un test avant le code** : Avant de coder une fonctionnalité ou une unité, un test est écrit pour spécifier et valider ce que le code va faire. Ce test échouera initialement car la fonctionnalité n'a pas encore été implémentée
* **Rendre le test réussi** : Écrire le code minimum nécessaire pour passer le test. Cela garantit que le développement est axé sur les exigences spécifiques et réduit la probabilité de surdéveloppement
* **Refactoriser** : Nettoyer et améliorer le code tout en s'assurant que tous les tests continuent de réussir. Cela inclut l'optimisation du code, l'élimination des redondances et l'amélioration de la lisibilité sans changer son comportement

1. Les avantages du TDD

Le TDD offres d’importants avantages dans le développement logiciel à savoir :

* **Une meilleure conception de code** : En écrivant des tests avant le code, les développeurs sont obligés de réfléchir à la manière dont le code sera utilisé et testé, ce qui conduit souvent à une conception plus modulaire et plus maintenable.
* **Une réduction des bugs et des régressions** : Les tests automatisés identifient rapidement les bugs et les régressions, ce qui améliore la qualité globale du logiciel.
* **Confiance dans les modifications** : Avec une suite de tests complète, les développeurs peuvent apporter des modifications et des améliorations au code sans craindre de casser les fonctionnalités existantes.

1. Les limites du TDD

Malgré ses avantages le TDD présente quelques limites que sont :

* **Augmentation du temps initial de développement** : L'écriture de tests avant le code peut sembler lente au début et augmenter le temps de développement initial.
* **Maintien des tests** : Les tests doivent être maintenus et mis à jour en même temps que le code de production, ce qui peut nécessiter des efforts supplémentaires.
* **Complexité des tests** : Certains tests peuvent être difficiles à écrire, surtout pour les interfaces utilisateur ou les intégrations complexes.

1. Exemples de test Unitaire et d’intégrations avec Spring Boot

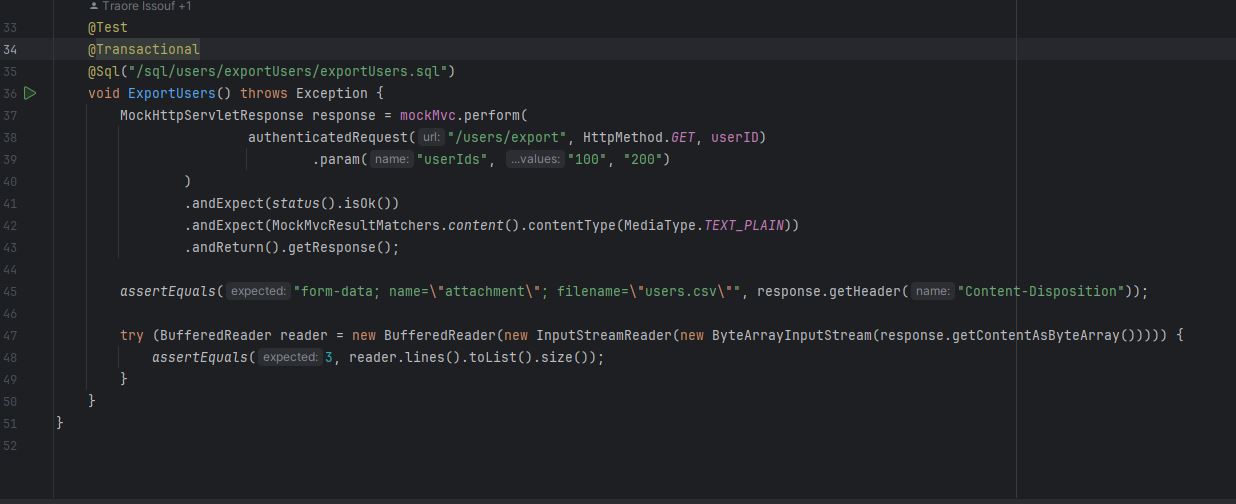


Figure 1: test d'intégration

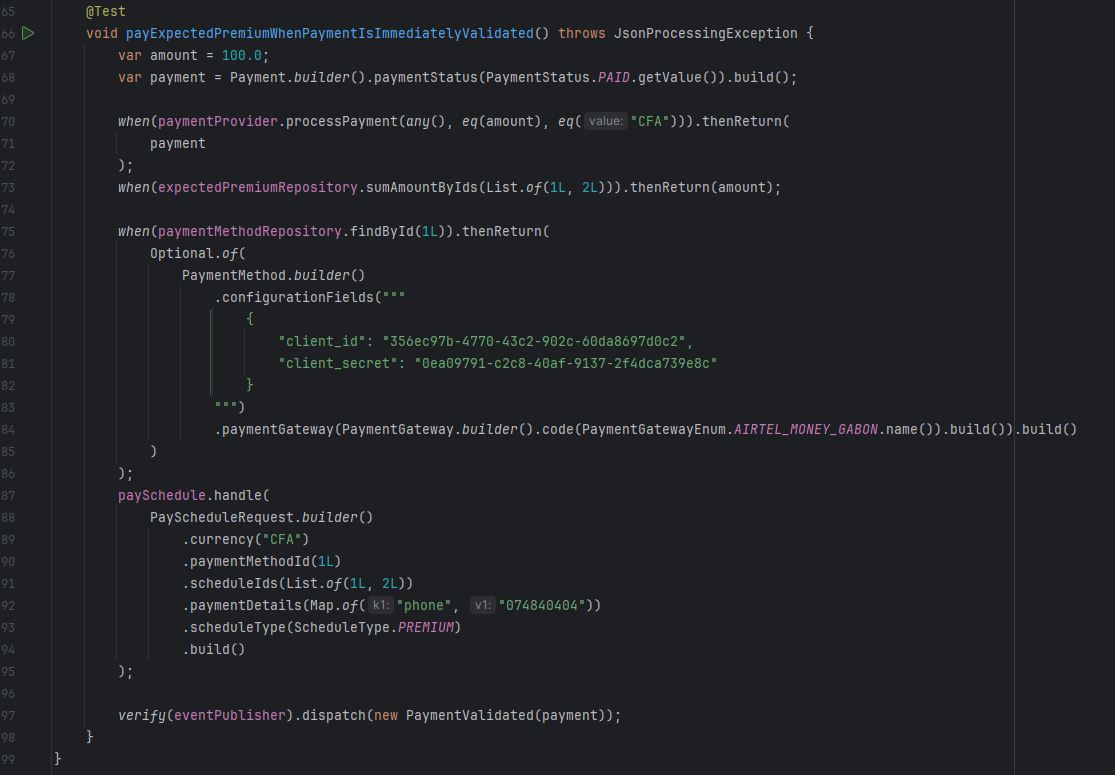


Figure 2: test unitaire

# II- Security by Design

1. Définition

Security by design (ou « sécurité par la conception ») **désigne un produit tel qu'un logiciel dont la sécurité et la notion de risque sont au cœur de sa conception**. Cette approche permet d'augmenter la sécurité du produit pour le protéger de potentielles menaces en réduisant les risques de failles.

1. Les principes d’une conception secure by design

* **Modélisation des Menaces** : Identifier les menaces et les vulnérabilités potentielles dès le début du processus de développement. Cela implique de comprendre les attaquants potentiels, leurs capacités et leurs motivations, ainsi que les vecteurs d'attaque possibles.
* **Architecture Sécurisée** : Concevoir des systèmes avec la sécurité comme préoccupation principale. Cela inclut le choix de configurations sécurisées, l'utilisation de pratiques de codage sécurisées et la garantie que l'architecture du système soutient les objectifs de sécurité tels que la confidentialité, l'intégrité et la disponibilité.
* Distinguer et restreindre les privilèges : S'assurer que les utilisateurs et les systèmes fonctionnent avec le niveau d'accès minimum nécessaire pour effectuer leurs fonctions. Ce principe limite les dommages potentiels dus à une mauvaise utilisation accidentelle ou malveillante des privilèges.
* **Tests de Sécurité Réguliers** : Tester continuellement le système pour détecter les vulnérabilités tout au long du cycle de développement. Cela inclut l'analyse statique du code, l'analyse dynamique, les tests de pénétration et les audits de sécurité.

# III- Test By Design

1. Définition

**Test by Design** est une approche qui vise à inclure la planification et la conception des tests dès les premières phases du cycle de développement de logiciels

1. Les caractéristiques du test by design

* **Planification Anticipée des Tests** : Les tests sont planifiés et conçus en parallèle avec la définition des exigences et la conception du système. L'objectif est de garantir que chaque exigence fonctionnelle et non fonctionnelle est testable et que les tests appropriés sont conçus pour valider ces exigences.
* **Collaboration** : Cette approche implique souvent une collaboration étroite entre les développeurs, les testeurs, les analystes métier, et parfois même les utilisateurs finaux, pour s'assurer que les tests reflètent bien les attentes et les besoins réels.
* **Documentation** : Les plans de test et les cas de test sont documentés dès le début, ce qui aide à clarifier les exigences et à s'assurer que tous les aspects du système sont pris en compte dans les tests.
* **Couverture Complète** : L'objectif est de garantir une couverture de test complète et d'identifier les scénarios de test dès la phase de conception du système, ce qui permet de détecter les problèmes potentiels avant qu'ils ne deviennent critiques.