

Preuve par test unitaire (sans base de données)

Ce document vous explique **comment fournir une preuve par test unitaire**, sans entrer dans du code détaillé, afin de démontrer que votre **logique métier est correctement découplée du stockage**.

L'objectif n'est pas de produire des tests complexes, mais de montrer que :

- vos règles métier sont bien appliquées,
- votre service métier fonctionne **sans base de données**,
- la persistance est correctement simulée.

Principe général

Un test unitaire, dans ce livrable, sert à valider deux points essentiels :

1. **La règle métier est respectée** (ou refusée si elle est violée).
2. **Le métier fonctionne avec une persistance simulée** (repository en mémoire).

Vous ne testez pas la base de données.

Vous ne testez pas l'ORM.

Vous testez **le comportement métier**.

Ce que vous devez mettre en place (conceptuellement)

1. Utiliser un repository en mémoire

Vous devez disposer d'une implémentation simulée du repository (par exemple [InMemoryTicketRepository](#)).

Ce repository :

- stocke les objets en mémoire (liste ou map),
- ne dépend d'aucune base de données,
- respecte l'interface du repository.

C'est ce repository qui sera utilisé dans les tests.

2. Instancier le service métier avec ce repository

Le service métier (par exemple `TicketService`) doit être instancié en lui fournissant le repository en mémoire.

Cela permet de démontrer que :

- le service dépend uniquement de l'**abstraction** (interface du repository),
- l'implémentation concrète du stockage est interchangeable.

Aucune dépendance technique (SQL, ORM, configuration de base) ne doit être nécessaire.

Structure logique d'un test (sans code)

Chaque test suit la même logique, souvent appelée :

- **Given** : un état initial (les données de départ),
- **When** : une action métier,
- **Then** : un résultat attendu.

Cette structure doit être lisible, même sans regarder le code.

Les deux tests recommandés

Il est fortement recommandé de fournir **au moins deux tests simples**.

Test 1 – Cas négatif : règle métier violée

Objectif : prouver que la règle métier est bien appliquée.

- **Given** :
 - un ticket existant en mémoire,
 - un ticket ouvert,
 - aucune résolution associée.
- **When** :
 - vous demandez au service métier de clôturer le ticket **sans fournir de résolution**.
- **Then** :
 - l'opération est refusée (erreur, exception ou échec contrôlé),
 - le ticket reste dans son état initial (non clôturé).

Ce test montre que la règle métier est protégée.

Test 2 – Cas positif : règle métier respectée

Objectif : prouver que le flux métier fonctionne correctement.

- **Given :**
 - un ticket existant en mémoire,
 - un ticket ouvert.
- **When :**
 - vous demandez au service métier de clôturer le ticket **avec une résolution valide.**
- **Then :**
 - le ticket passe à l'état « clôturé »,
 - le ticket est correctement sauvegardé dans le repository en mémoire,
 - la récupération du ticket confirme le nouvel état.

Ce test montre que le service métier fonctionne de bout en bout, sans base.

Comment démontrer que vous n'utilisez pas de base de données

Votre preuve est considérée valide si :

- aucun service de base de données n'est démarré,
- aucune configuration de datasource n'est utilisée,
- aucun ORM n'est initialisé,
- seuls des objets métier et un repository en mémoire sont manipulés.

Si le test peut s'exécuter dans n'importe quel environnement, sans dépendance externe, alors le découplage est réussi.

Ce que vous pouvez rendre

Pour cette preuve, vous pouvez fournir :

- un ou deux tests unitaires simples,
- et éventuellement une courte explication écrite (README ou REFLEXION.md) indiquant :

« Les règles métier sont validées par des tests unitaires utilisant une implémentation InMemory du repository, sans base de données. »

Erreurs fréquentes à éviter

- Tester la base de données au lieu du métier.
- Initialiser un ORM dans les tests.
- Faire dépendre le test d'une configuration technique.
- Tester plusieurs règles métier dans un seul test.
- Oublier de vérifier l'état final du ticket.

Critère simple de réussite

Si vos tests :

- valident une règle métier,
- utilisent uniquement une persistance simulée,
- fonctionnent sans aucune base de données,

alors la preuve attendue est conforme.

Le test unitaire n'est pas là pour tester la technique.

Il est là pour prouver que :

- le métier est isolé,
- les règles sont au bon endroit,
- l'architecture est saine.