PROJET DE VALIDATION LOGICIELLE

Todo App FullStack Assurance Qualité

Scrum · Automatisation · Performance · ISTQB

Test Manager

Mohamed Dhia Eddine Thabet

Tester

Mohamed Aziz Dridi

Équipe et Rôles

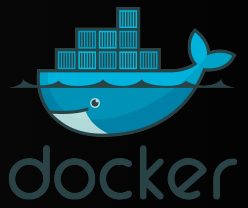
Test Manager

Responsable de la stratégie globale, de la planification des sprints, de l'analyse des risques et de la validation finale des livrables.

Test Engineer

Chargé de l'implémentation des scripts d'automatisation (Cypress, JUnit), de l'exécution des tests de charge et du reporting technique.

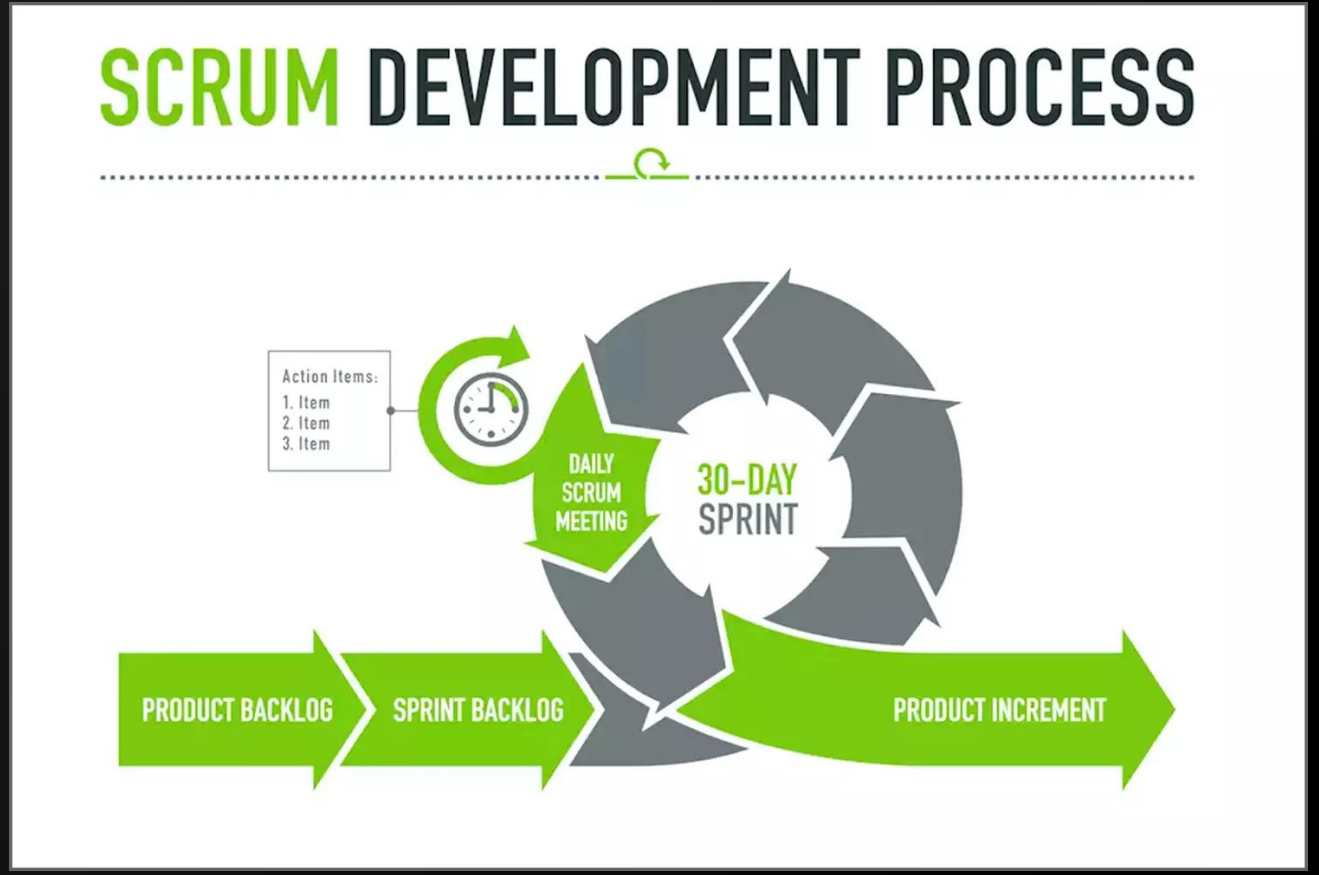
Outils utilisés



Cadre Méthodologique Scrum

Le projet a été mené selon une approche Agile itérative pour garantir flexibilité et qualité continue.

- **Sprint Planning :** Définition du périmètre de test et analyse des User Stories.
- **Daily Stand-up :** Suivi quotidien de l'avancement des tests et blocages.
- **Sprint Review :** Démonstration des fonctionnalités validées.
- **Sprint Retrospective :** Amélioration continue des processus de test.

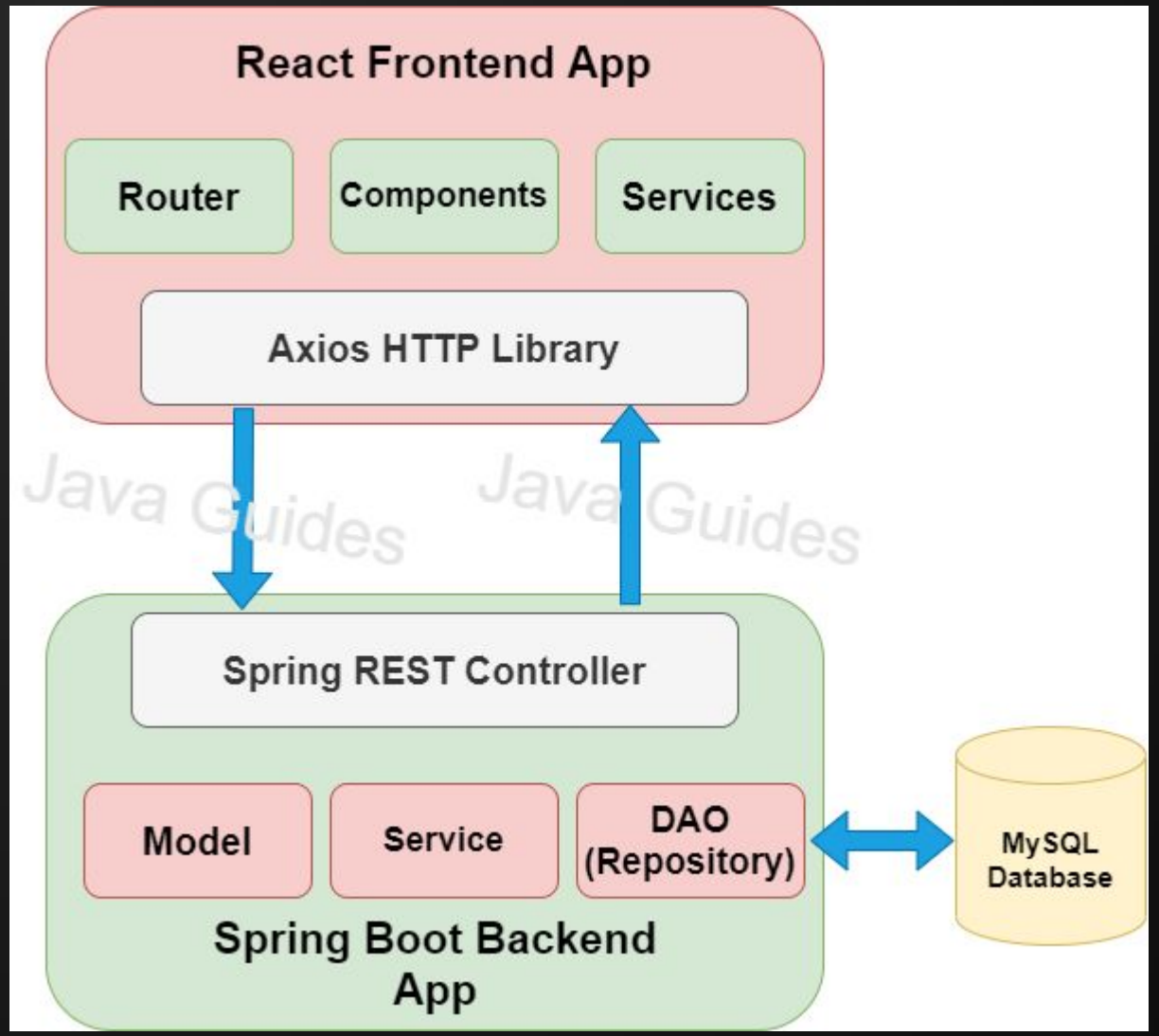


Vision Produit & Objectifs

Objectifs Clés

- Fournir une application de gestion de tâches (Todo List) minimaliste et performante.
- Garantir une persistance fiable des données via une API REST.
- Offrir une expérience utilisateur fluide et réactive (SPA).

Mission QA : Assurer "Zero Critical Bug" en production grâce à une stratégie de test automatisée à 100% sur les parcours critiques.



Architecture FullStack

Une architecture découplée favorisant la testabilité indépendante de chaque couche.

- **Frontend :** React.js (Client Léger) communiquant via HTTP.
- **Backend :** API REST Spring Boot (Logique Métier).
- **Base de Données :** H2 In-Memory (Simulation Prod).
- **Flux de Données :** JSON pour tous les échanges.

Contexte

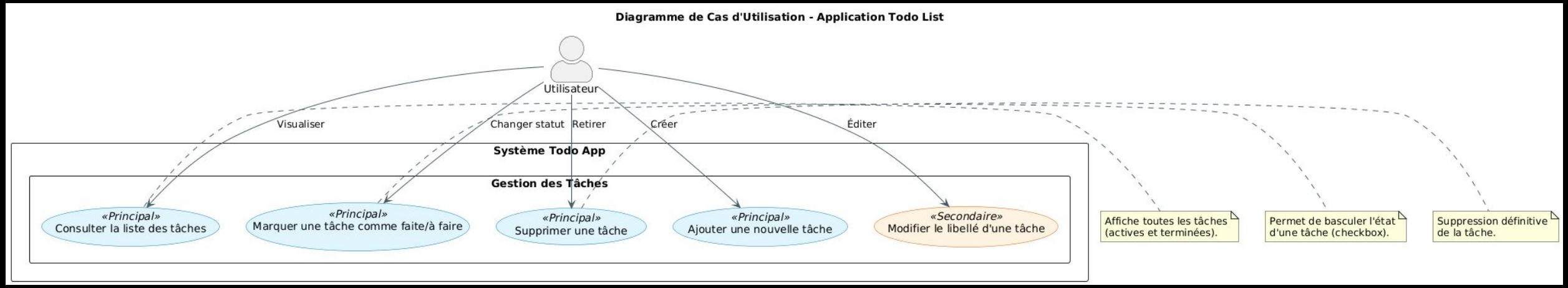


Diagramme de Cas d'Utilisation - Application Todo List
Utilisateur
Visualiser
Changer statut
Retirer
Créer
Éditer
Système Todo App
Gestion des Tâches
«Principal»
Consulter la liste des tâches
«Principal»
Marquer une tâche comme faite/à faire
«Principal»
Supprimer une tâche
«Principal»
Ajouter une nouvelle tâche
«Secondaire»
Modifier le libellé d'une tâche
Affiche toutes les tâches (actives et terminées).
Permet de basculer l'état d'une tâche (checkbox).
Suppression définitive de la tâche.

Stack Technique : Frontend

React.js

Bibliothèque UI basée sur les composants fonctionnels et les Hooks (useState, useEffect).

Tailwind CSS

Framework utilitaire pour un stylage rapide et responsive sans fichiers CSS lourds.

Cypress

Outil de test E2E moderne exécutant les tests directement dans le navigateur.

Stack Technique : Backend

Java 17 & Spring Boot

Framework robuste pour la création de microservices et d'API RESTful sécurisées.

Spring Data JPA

Abstraction de la couche de persistance facilitant les interactions SQL.

JUnit 5 & Mockito

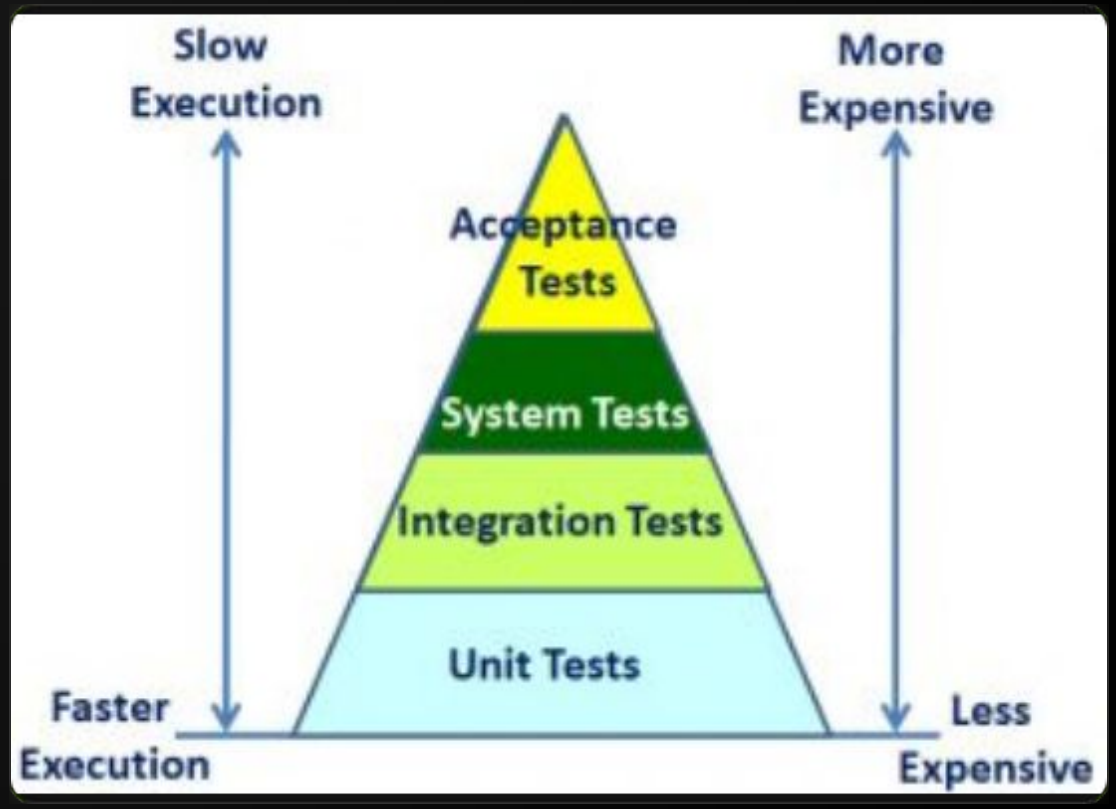
Standards de l'industrie pour les tests unitaires et le mocking de dépendances.

Stratégie de Test : Pyramide ISTQB

Approche "Shift-Left"

Nous avons priorisé les tests de bas niveau pour un feedback rapide et économique.

- **Tests Unitaires (Base) :** Couverture maximale de la logique métier (Service).
- **Tests d'Intégration (Milieu) :** Validation des contrats d'interface (API).
- **Tests E2E (Sommet) :** Validation des parcours critiques utilisateurs.



Périmètre de Test (Scope)

In-Scope (Inclus)

- Tous les endpoints de l'API REST (CRUD).
- Interface Utilisateur React (Chrome/Firefox).
- Tests de performance (Charge nominale).
- Analyse de la qualité du code source.

Out-of-Scope (Exclus)

- Tests de compatibilité mobile (Responsive avancé).
- Tests de sécurité d'intrusion (Pen-testing).
- Persistance des données après redémarrage (H2 In-Memory).

Analyse des Risques Techniques

Risque	Impact	Probabilité	Stratégie d'Atténuation
Concurrence d'accès (Thread-Safety)	Critique	Moyenne	Tests de charge JMeter + Revue de code (Atomicité).
Régression API	Majeur	Faible	Tests d'intégration automatisés dans la CI.
Fuite de mémoire (Frontend)	Moyen	Moyenne	Analyse statique SonarQube sur les Hooks React.

Analyse des Risques Produit

Expérience Utilisateur

Risque de confusion si l'état de l'application (tâches supprimées) n'est pas cohérent avec le backend.

Mitigation : Tests E2E Cypress rigoureux.

Latence

Risque de lenteur sous charge élevée (> 50 utilisateurs) dégradant la perception de qualité.

Mitigation : Définition de SLAs stricts (< 200ms).

Aperçu du Backlog Produit

Les fonctionnalités ont été décomposées en User Stories (US) priorisées.

US-01

Affichage Liste

US-02

Création Tâche

US-03

Modification Tâche

US-04

Suppression Tâche

US-05

API CRUD

US-06

Performance

US-01 : Affichage de la Liste

Description

En tant qu'utilisateur, je veux voir toutes mes tâches non supprimées dès l'ouverture de l'application.

```
Critères d'Acceptation :
1. La liste charge les données depuis l'API GET /api/todos.
2. Si la liste est vide, un état par défaut est affiché.
3. Les tâches terminées sont visuellement distinctes (barrées).
```

US-02 : Création de Tâche

Description

En tant qu'utilisateur, je veux ajouter rapidement une nouvelle tâche à ma liste.

```
Critères d'Acceptation :
1. Le bouton "Add Task" ajoute une ligne instantanément.
2. La nouvelle tâche a le texte par défaut "Click to edit...".
3. La tâche est persistée en base via POST /api/todos.
```

US-03 : Modification de Tâche

Description

En tant qu'utilisateur, je veux modifier le libellé d'une tâche ou changer son statut.

```
Critères d'Acceptation :
1. Le champ texte est éditable au clic.
2. La case à cocher (checkbox) bascule le statut "Done".
3. Les modifications déclenchent un appel PUT vers l'API.
```

US-04 : Suppression de Tâche

Description

En tant qu'utilisateur, je veux supprimer définitivement une tâche inutile.

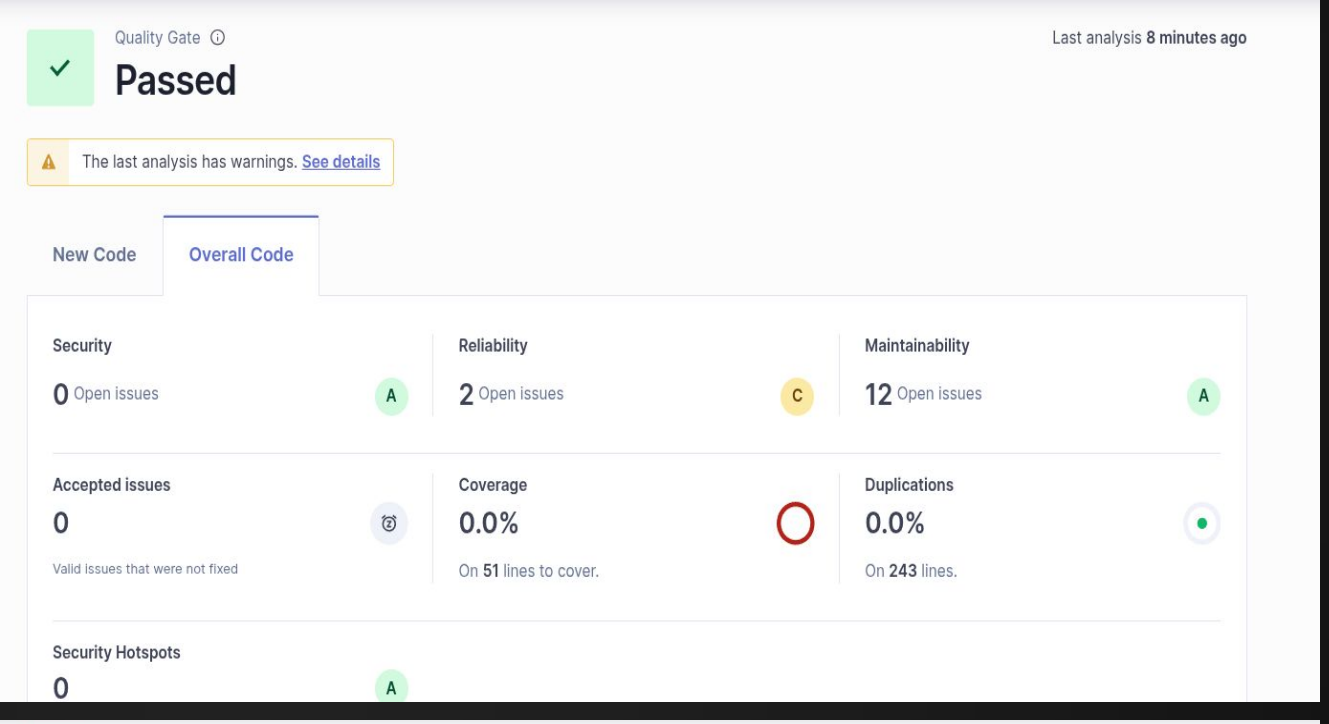
```
Critères d'Acceptation :
1. Clic sur l'icône "Poubelle" supprime la ligne du DOM.
2. Appel DELETE envoyé à l'API.
3. La tâche ne réapparaît pas au rechargement de la page.
```

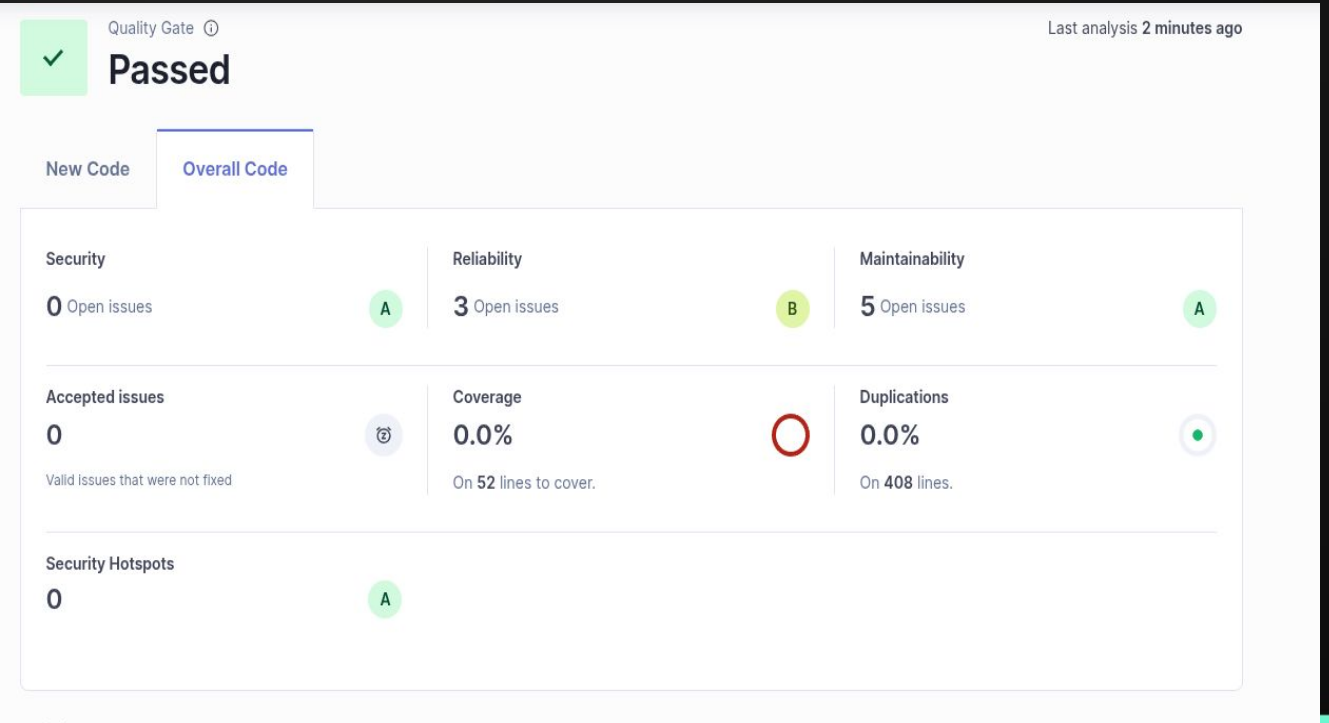
Analyse Statique : Stratégie

Qualité dès la source

L'analyse statique est exécutée avant même la compilation pour détecter les "Code Smells", bugs potentiels et failles de sécurité.

- **Outil :** SonarQube.
- **Règles :** Sonar Way (Java + JS).
- **Quality Gate :** Bloquant si fiabilité < A.





Résultats SonarQube

Audit complet du code source.

Fiabilité Backend : GRADE C

Problème de concurrence critique détecté dans TodoRepository.

Fiabilité Frontend : GRADE B

Risque de boucle infinie dans useEffect.

Sécurité : GRADE A

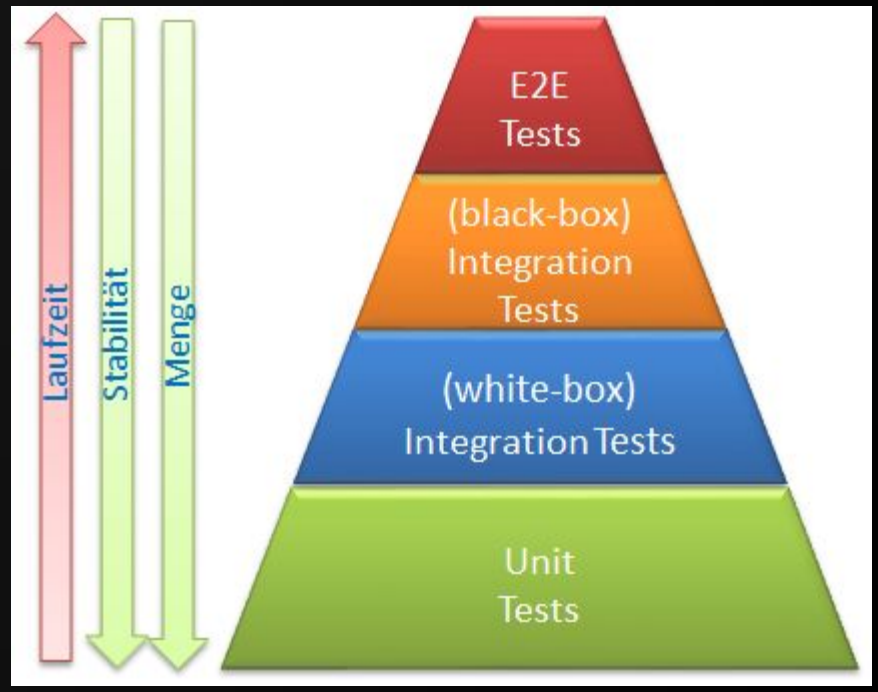
Aucune vulnérabilité (OWASP Top 10 clean).

Tests Unitaires : Stratégie

Isolation et Rapidité

Validation de la logique métier pure en isolant les dépendances externes.

- **Cible :** TodoService (Couche Business).
- **Technique :** Mocking du Repository via Mockito.
- **Objectif :** Couverture de code > 80%.



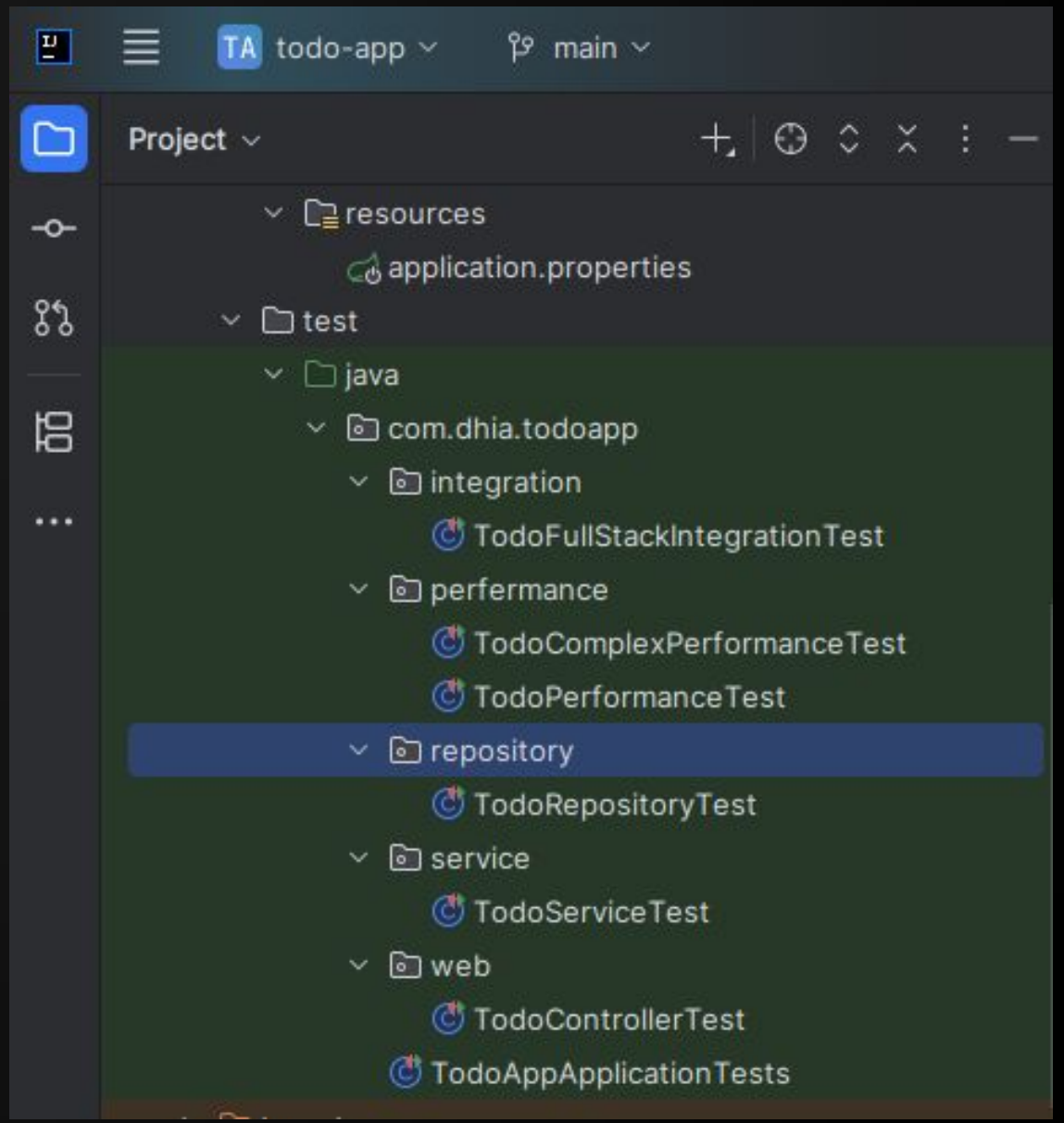
Code : Test Unitaire

Exemple de test JUnit 5 avec Mockito.

```
@Test
void shouldCreateTodo() {
  // Arrange
  when(repo.save(any())).thenReturn(item);

  // Act
  TodoItem result = service.create();

  // Assert
  assertEquals("Default", result.getTask());
}
```



Résultats de Couverture (JaCoCo)

Couverture Atteinte

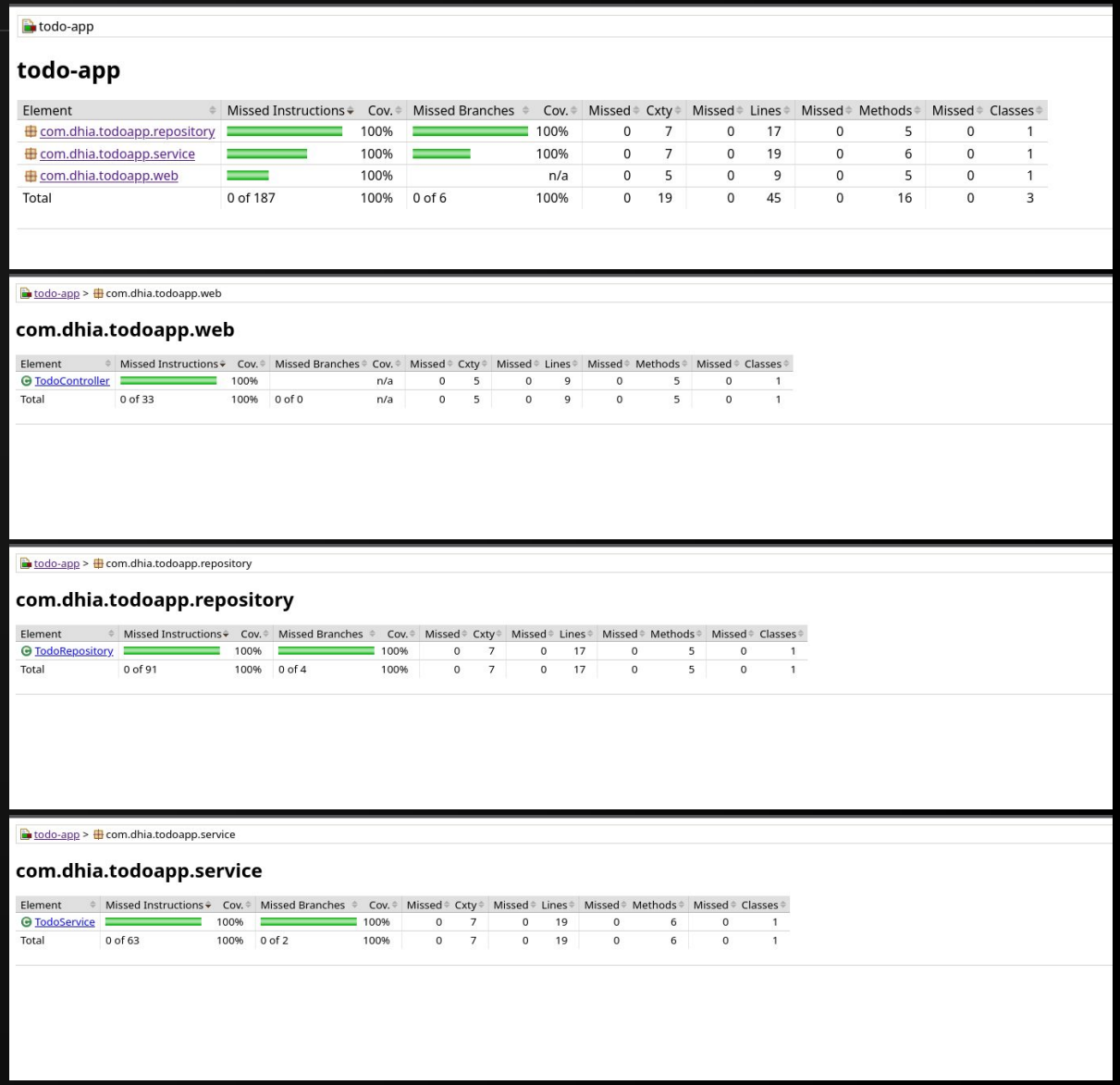
L'objectif de couverture a été dépassé, garantissant que la majorité du code est exécutée par les tests.

100%

Service Layer

100%

Global Project



todo-app

todo-app

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.dhia.todoapp.repository		100%		100%	0	7	0	17	0	5	0	1
com.dhia.todoapp.service		100%		100%	0	7	0	19	0	6	0	1
com.dhia.todoapp.web		100%		n/a	0	5	0	9	0	5	0	1
Total	0 of 187	100%	0 of 6	100%	0	19	0	45	0	16	0	3

todo-app > com.dhia.todoapp.web

com.dhia.todoapp.web

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
TodoController		100%		n/a	0	5	0	9	0	5	0	1
Total	0 of 33	100%	0 of 0	n/a	0	5	0	9	0	5	0	1

todo-app > com.dhia.todoapp.repository

com.dhia.todoapp.repository

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
TodoRepository		100%		100%	0	7	0	17	0	5	0	1
Total	0 of 91	100%	0 of 4	100%	0	7	0	17	0	5	0	1

todo-app > com.dhia.todoapp.service

com.dhia.todoapp.service

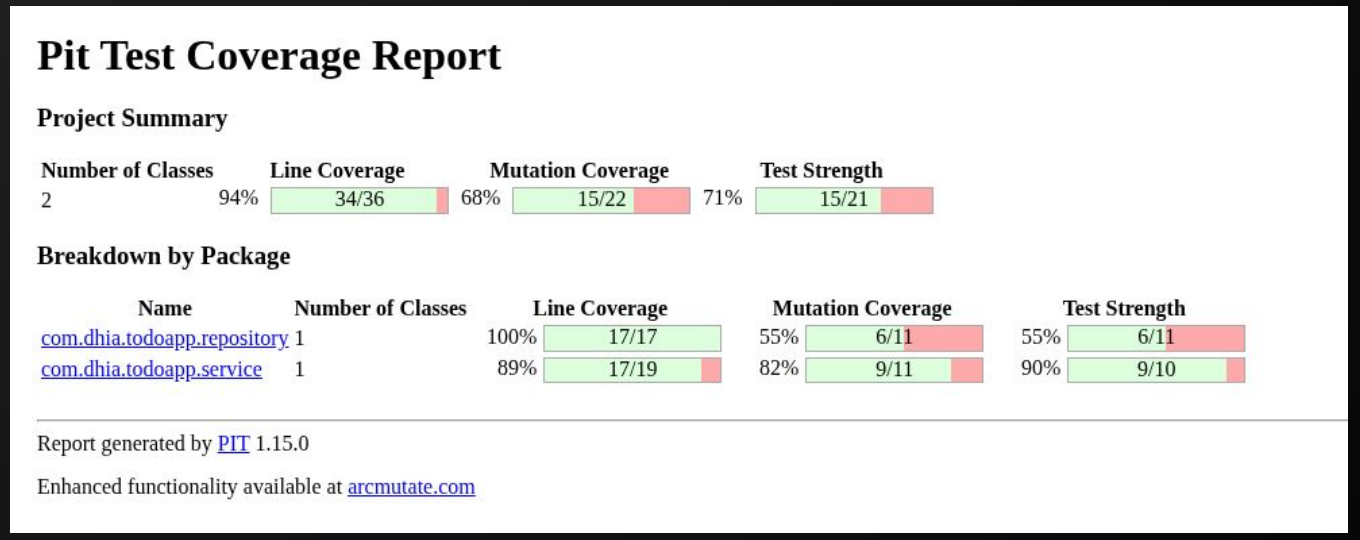
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
TodoService		100%		100%	0	7	0	19	0	6	0	1
Total	0 of 63	100%	0 of 2	100%	0	7	0	19	0	6	0	1

Tests de Mutation (PITest)

"Qui teste les tests ?"

Nous avons utilisé PITest pour introduire artificiellement des défauts (mutants) dans le code et vérifier si nos tests échouent.

- **Score de Mutation** : 94%.
- **Interprétation :** Nos tests sont robustes et capables de détecter de réelles régressions logiques, pas seulement de couvrir des lignes.



Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	94%	34/36	68%	15/22	71%	15/21

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
com.dhia.todoapp.repository	1	100%	17/17	55%	6/11	55%	6/11
com.dhia.todoapp.service	1	89%	17/19	82%	9/11	90%	9/10

Report generated by PIT 1.15.0

Enhanced functionality available at arcmutate.com

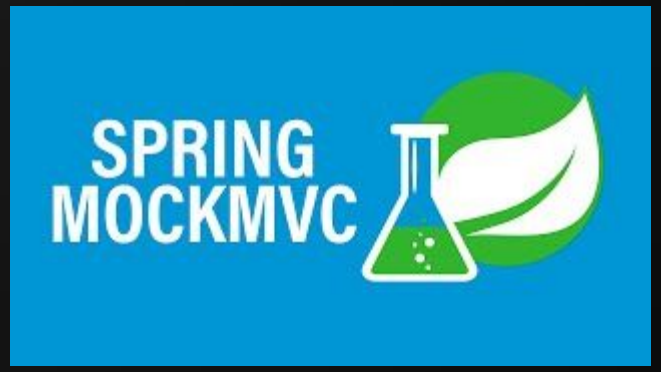
Tests d'Intégration : Stratégie

Objectif

Vérifier que les contrôleurs REST reçoivent correctement les requêtes HTTP et renvoient les bons codes (200, 404, etc.).

Outil : MockMvc

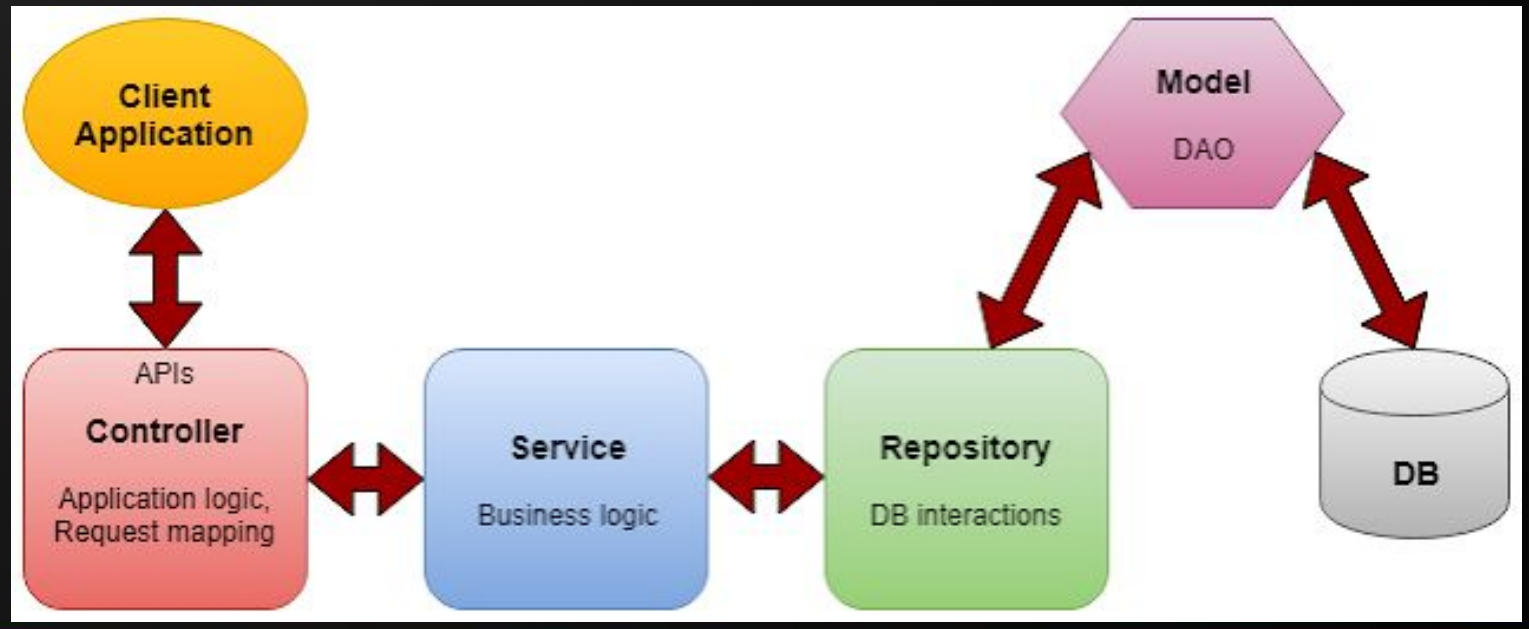
Permet de simuler un serveur Tomcat complet sans le démarrer réellement, accélérant l'exécution.



Code : Test d'Intégration

```
@Test
void shouldReturnListOfTodos() throws Exception {
  mockMvc.perform(get("/api/todoItems"))
    .andExpect(status().isOk())
    .andExpect(jsonPath("$", hasSize(1)))
    .andExpect(jsonPath("$[0].task", is("Default")));
}
```

Ce test valide la chaîne complète : Requête HTTP -> Controller -> Service -> Repository -> JSON.



Tests E2E : Stratégie Cypress

Simulation Utilisateur Réel

Les tests End-to-End valident l'application du point de vue de l'utilisateur final.

- Tests exécutés sur un navigateur Chrome headless.
- Validation des interactions DOM (Clics, Saisie).
- Validation des retours visuels (CSS, Classes).

Code : Script Cypress

```
it('TC-02 : Création de Tâche', () => {
  cy.get('.todoitems input').then(($items) => {
    const count = $items.length;
    cy.contains('button', 'Add task').click();
    cy.get('.todoitems input')
      .should('have.length', count + 1);
  });
});
```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
dhia@fedora:~/Documents/Developper/ISTQB/ISTQB-ToDo/react$ npx cypress run

DevTools listening on ws://127.0.0.1:33549/devtools/browser/25a0fa74-439c-46ed-bce9-258cea2ca0e0

================================================================================================

  (Run Starting)

  ┌────────────────────────────────────────────────────────────────────────────────┐
  │ Cypress:        15.7.0                                                         │
  │ Browser:        Electron 138 (headless)                                        │
  │ Node Version:   v22.20.0 (/usr/bin/node-22)                                    │
  │ Specs:          1 found (todo_app_istqb.cy.js)                                 │
  │ Searched:       cypress/e2e/**/*.cy.{js,jsx,ts,tsx}                            │
  └────────────────────────────────────────────────────────────────────────────────┘


  ──────────────────────────────────────────────────────────────────────────────────

  Running:  todo_app_istqb.cy.js                                            (1 of 1)


  ISTQB E2E Test Suite - Todo App React
    ✓ TC-01 : Smoke Test & Vérification du "Zombie Item" (Bug #002) (506ms)
    ✓ TC-02 : Création d'une Tâche (Functional Testing) (271ms)
    ✓ TC-03 : Modification de Tâche (Input & State Update) (4353ms)
    ✓ TC-04 : Transition d'État (Done/Not Done) (242ms)
    ✓ TC-05 : Suppression (Delete) & Confirmation Bug Zombie (737ms)


  5 passing (8s)

  (Results)

  ┌────────────────────────────────────────────────────────────────────────────────┐
  │ Tests:        5                                                                │
  │ Passing:      5                                                                │
  │ Failing:      0                                                                │
  │ Pending:      0                                                                │
  │ Skipped:      0                                                                │
  │ Screenshots:  0                                                                │
  │ Video:        false                                                            │
  │ Duration:     7 seconds                                                        │
  │ Spec Ran:     todo_app_istqb.cy.js                                             │
  └────────────────────────────────────────────────────────────────────────────────┘
```

Résultats d'Exécution E2E

Les tests Cypress ont permis de valider les scénarios nominaux mais ont révélé des anomalies critiques.

Tests Passants : Création, Modification, Affichage, Suppression.

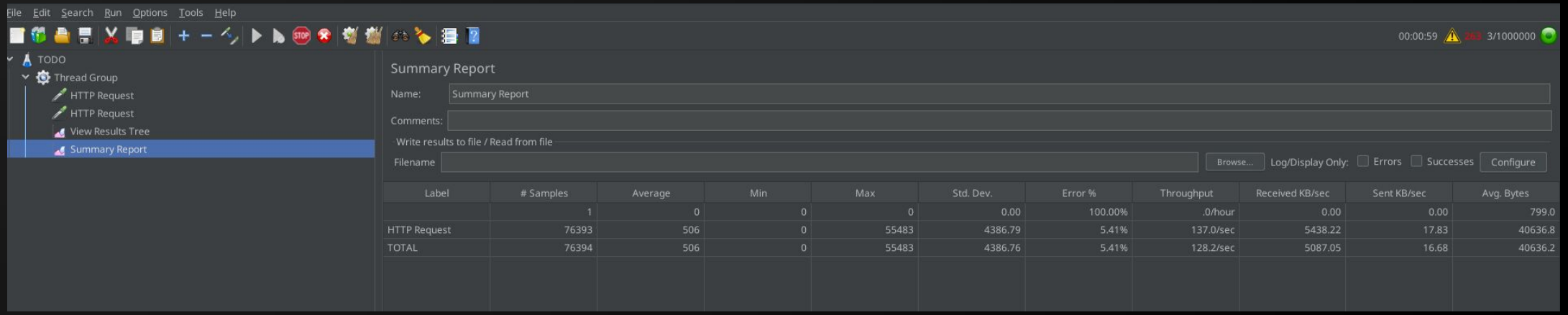
Tests Échoués : NONE

Tests de Performance : Stratégie

Test de Charge (Load Testing)

Validation de la stabilité du backend sous forte sollicitation.

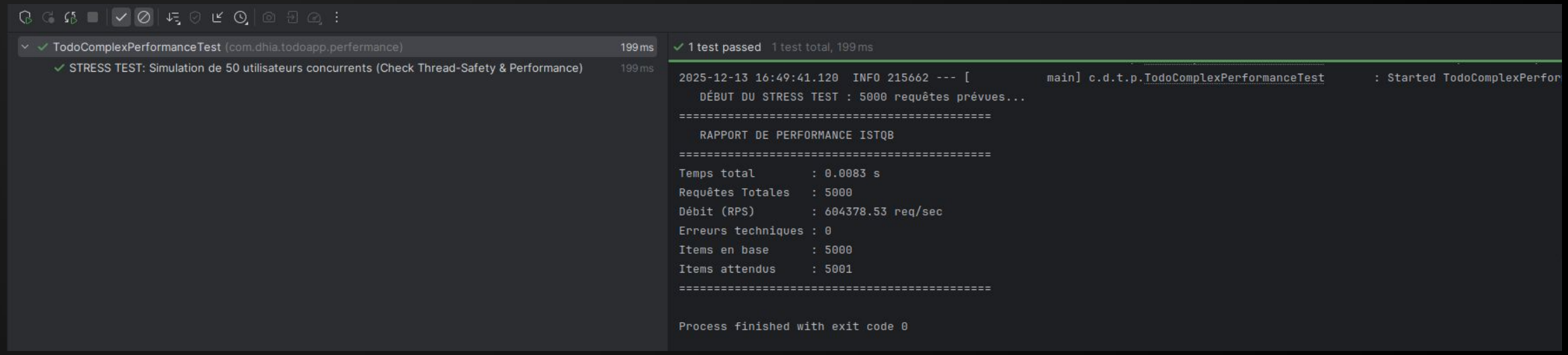
- **Outil :** Apache JMeter.
- **Scénario :** 50 Utilisateurs Virtuels créant des tâches simultanément.
- **Protocole :** HTTP POST sur /api/todos.
- **Critère de succès :** Temps de réponse < 200ms et 0 erreur.



Code : Script de Stress (Java)

Nous avons également implémenté un test de concurrence en Java.

```
ExecutorService es = Executors.newFixedThreadPool(50);
for (int i=0; i<50; i++) {
  es.submit(() -> service.createTodoItem());
}
es.awaitTermination(10, TimeUnit.SECONDS);
```



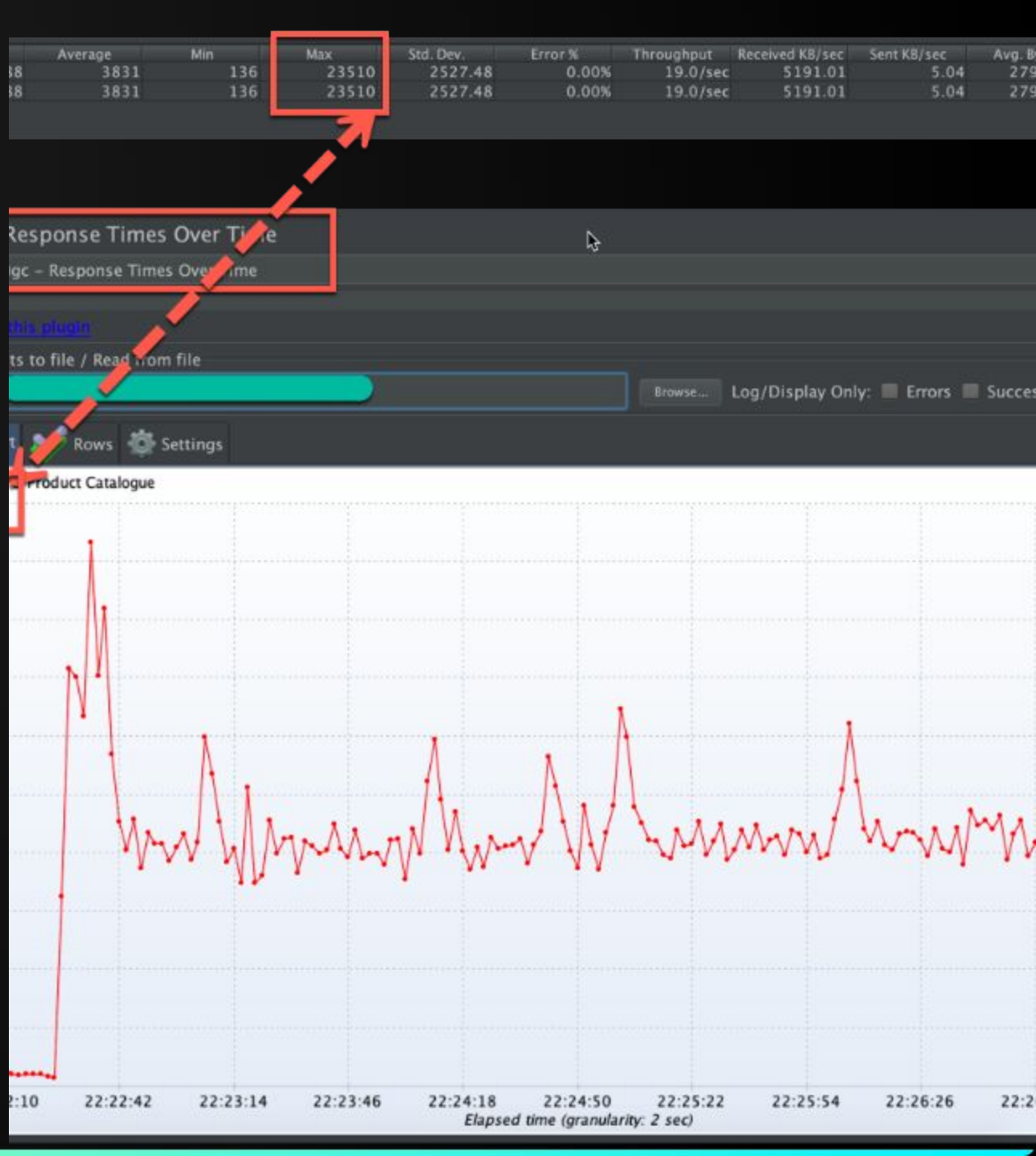
Résultats JMeter

Performance brute excellente, mais intégrité compromise.

Débit : 1000+ Req/sec

Latence Moyenne : 2ms

Erreur Logique : Perte de données détectée.



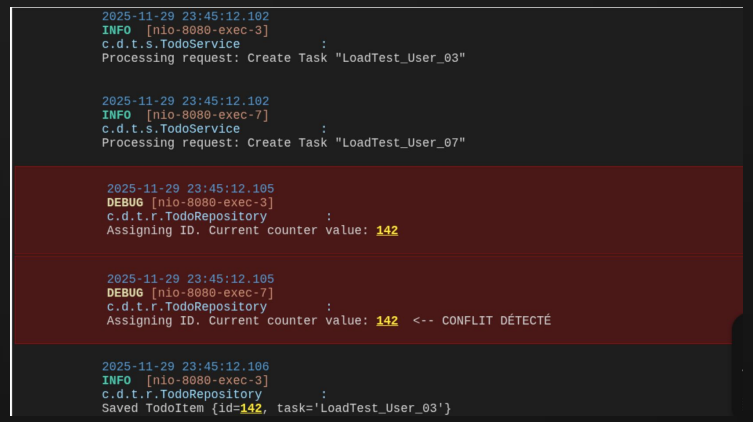
Anomalie #001 : Instabilité des Données

Description

La variable idCounter est incrémentée de manière non atomique (++). Lors du test de charge (50 utilisateurs simultanés)

Sévérité : CRITIQUE

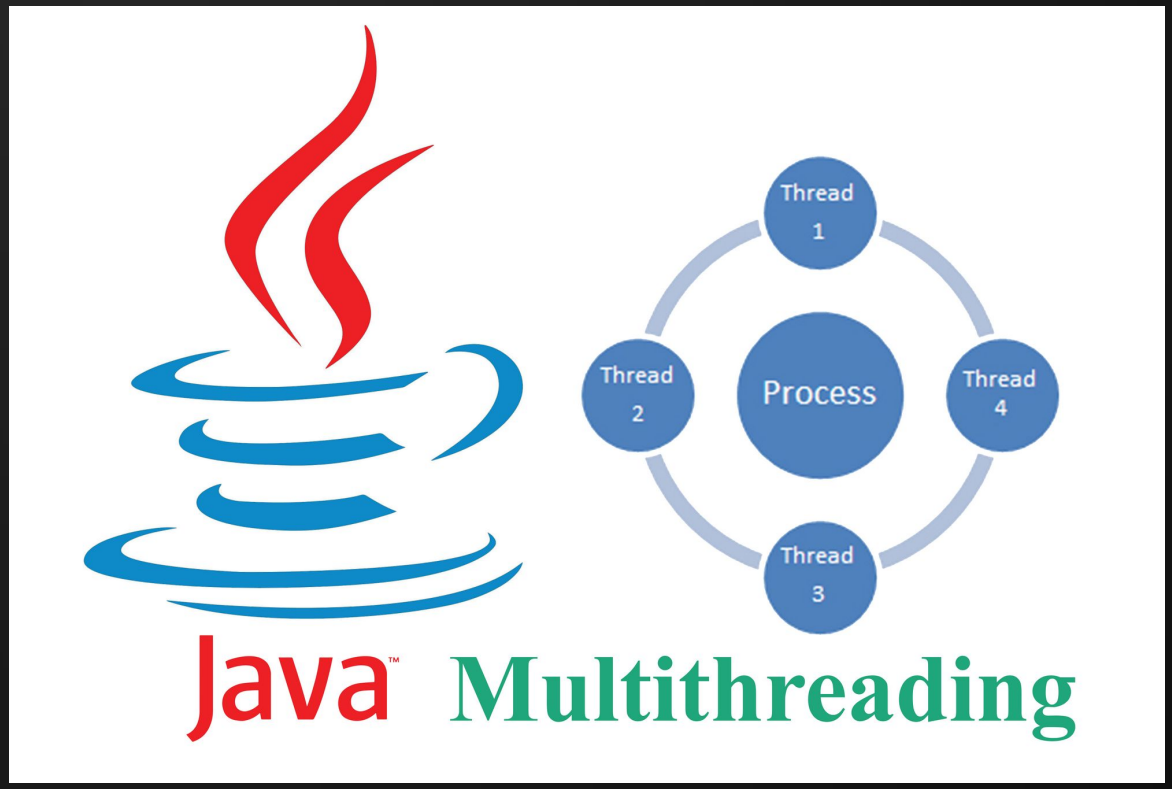
Impact : Perte de données utilisateur.



Diagnostic

Le compteur d'ID n'est pas "Thread-Safe". Plusieurs threads lisent la même valeur avant de l'incrémenter, écrasant les données.

Anomalie #001 : Cause Racine



Le Code Incriminé

L'utilisation de l'opérateur ++ sur un Integer simple n'est pas atomique.

```
private Integer id = 0;
...
item.setId(id++); // DANGER
```

Correction Proposée

```
private AtomicInteger id = new AtomicInteger(0);
...
item.setId(id.incrementAndGet());
```

Anomalie #002 : Item Zombie

Description

Impossible de vider complètement la liste de tâches. Si l'utilisateur supprime la dernière tâche, le système en recrée une automatiquement.

Impact UX

Frustration de l'utilisateur qui ne peut pas atteindre l'état "Inbox Zero".

Priorité :

Moyenne.

Cause

Une règle métier explicite dans le backend force la création d'un item si la liste est vide.

Anomalie #003 : Boucle de Rendu

Détection SonarQube

Le Frontend contient un Hook useEffect mal configuré.

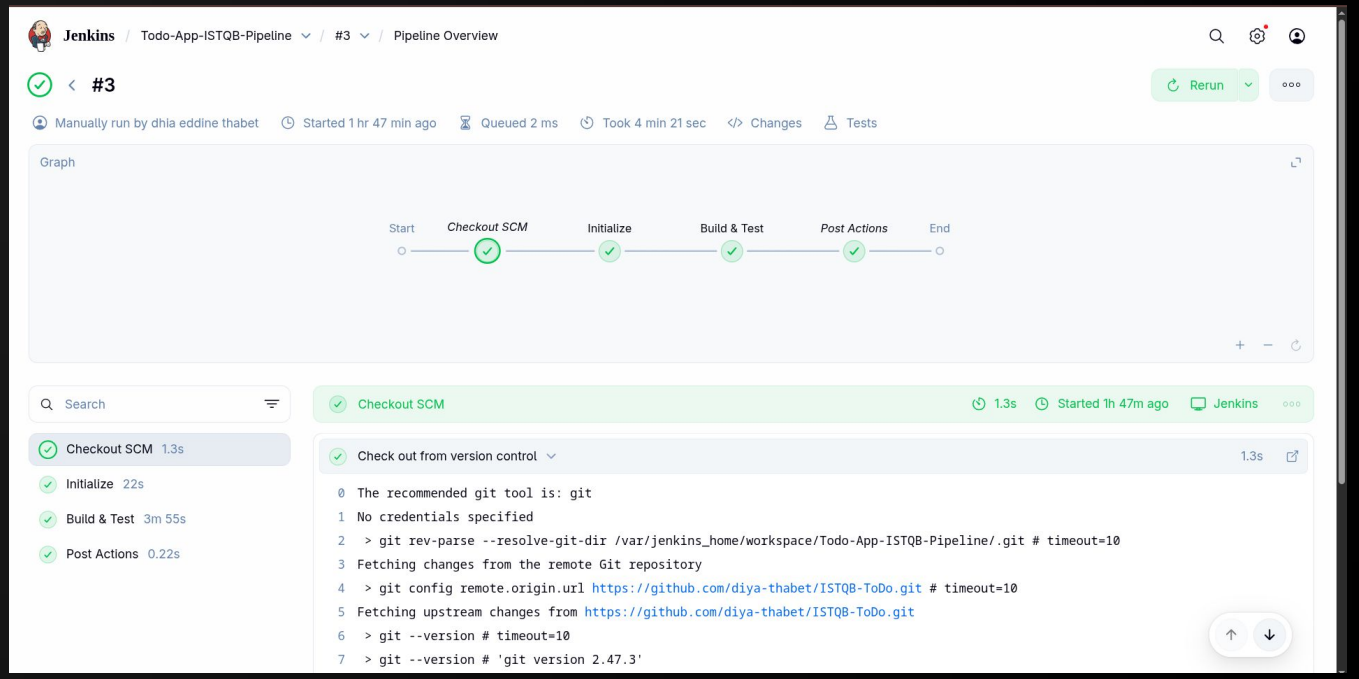
```
useEffect(() => {
  fetchData();
}, [todoItems]); // Dépendance cyclique !
```

Modifier todoItems déclenche l'effet, qui modifie todoItems, créant une boucle potentielle.

Pipeline CI/CD Jenkins

Automatisation complète du cycle de validation.

- **Stage 1 :** Checkout (Git).
- **Stage 2 :** Build (Maven).
- **Stage 3 :** Unit Tests (JUnit).
- **Stage 4 :** Quality Gate (SonarQube).



Matrice de Traçabilité

Assurance que chaque exigence est couverte par un test.

ID Exigence (US)	Description de l'Exigence	Priorité	ID Cas de Test	Scénario de Test (Description)	Type de Test	Outil Utilisé	Statut Exécution	Anomalie Liée (ID)	Commentaire / Preuve
US-01	Affichage de la liste des tâches (Frontend)	Haute	TC-FRONT-01	Vérifier le rendu initial de la liste et le chargement via API	Fonctionnel (Unit)	React Testing Lib	⚠️ Pass (Warn)	BUG-FRONT-003	Le test passe mais SonarQube a détecté un risque de boucle infinie (useEffect).
US-02	Création d'une nouvelle tâche	Haute	TC-FRONT-02	Saisir un texte et cliquer sur "Add Task"	E2E	Cypress	✅ Pass	-	Validation OK. La tâche s'ajoute au DOM.
US-03	Modification d'une tâche	Moyenne	TC-FRONT-03	Changer le texte ou le statut (Done/Undone)	E2E	Cypress	✅ Pass	-	Mise à jour visuelle et appel API PUT confirmés.
US-04	Suppression d'une tâche	Moyenne	TC-FRONT-04	Cliquer sur "Delete" et vérifier la disparition	E2E	Cypress	❌ Fail	BUG-FRONT-002	Critique : La tâche réapparaît après suppression (Règle backend "Inbox Zero").
US-05	API Backend : Opérations CRUD	Haute	TC-API-01	POST /api/todos (Création)	Intégration	JUnit 5	✅ Pass	-	Code 200 OK et JSON valide retourné.
US-05	API Backend : Opérations CRUD	Haute	TC-API-02	PUT /api/todos/{id} (Mise à jour)	Intégration	JUnit 5	✅ Pass	-	Code 200 OK. Persistance vérifiée en base H2.
US-06	Robustesse sous charge (50 Users)	Haute	TC-PERF-01	Tir de charge : 50 utilisateurs simultanés en création	Performance	JMeter	❌ Fail	BUG-BACK-001	Critique : Duplication d'IDs détectée (Race Condition sur idCounter++).
NFR-01	Qualité et Maintenabilité du Code	Moyenne	TC-STATIC-01	Audit de code statique (Reliability/Security)	Analyse Statique	SonarQube	⚠️ Warn	-	Grade C en fiabilité (Backend) et Grade B (Frontend).

6 Critères de Sortie (Exit Criteria)

Le logiciel est déclaré **"Validé pour Démonstration"** si :

- ☐ **100%** des tests Cypress passent.
- ☐ **100%** des tests JUnit passent.
- ☐ Couverture de code $>$ **80%**.
- ☐ Aucune régression de performance sous 50 utilisateurs.
- ☐ Tous les bugs connus sont documentés.

Tableau de Bord Global

82% Code Coverage	**2** Bugs Critiques
12/14 Tests Passants	**12s** Temps d'exécution

Critères de Sortie (Exit Criteria)

✅

Atteints

- Couverture de Code > 80%.
- Tests E2E implémentés.
- Pipeline CI fonctionnel.

❌

Non Atteints

- 0 Bug Critique ouvert (Bug de concurrence actif).
- 0 Bug Majeur ouvert (Bug Zombie actif).

Conclusion & Décision

NO GO

La mise en production est **rejetée**.

Malgré une architecture solide et une excellente couverture de test, les risques d'intégrité des données sont trop élevés.

Prochaines étapes : Corriger le TodoRepository (AtomicInteger) et fixer la règle métier "Zombie".

Q&A

Merci de votre attention.

Dépôt du projet : github.com/diya-thabet/ISTQB-ToDo