Compte Rendu - DevOps & CI/CD

Continuous Testing & TDD

Auteur : Jean Jireh
Groupe : ING4 App DATA Gr3
Date : 3 février 2026

1 ◎ Objectif du Lab

But du TP

L'objectif principal de ce laboratoire était de mettre en pratique la méthodologie **TDD (Test Driven Development)** dans un environnement Node.js utilisant une base de données Redis.

Il s'agissait concrètement de :

- Comprendre la structure d'une application API REST existante.
- Écrire des tests unitaires (pour le contrôleur) et des tests d'intégration (pour le routeur API) avant d'implémenter la fonctionnalité.
- Implémenter la fonctionnalité de récupération d'un utilisateur (`GET /user/:username`) pour valider ces tests.

2 Possible application dans le monde réel

Dans le monde professionnel, cette approche est cruciale pour garantir la qualité et la maintenabilité du code.

Non-régression En écrivant les tests, on s'assure que les futures modifications du code ne casseront pas les fonctionnalités de récupération d'utilisateurs existantes.

Documentation Les tests servent de documentation technique vivante sur le comportement attendu de l'API.

Confiance CI/CD Une suite de tests complète permet de déployer en production automatiquement avec l'assurance que l'application fonctionne comme prévu.

3 Étape dans le cycle DevOps

Ce lab se situe principalement dans les étapes de **Code**, **Build** et **Test** du cycle DevOps.

Justification

- ✔ Le TDD intervient directement pendant la phase de **codage**.
- ✔ L'automatisation des tests via `npm test` fait partie intégrante de l'Intégration Continue (CI). Dans un pipeline DevOps réel, ces tests seraient exécutés automatiquement à chaque *commit* ou *pull request*.

4 Problèmes rencontrés et Résolution

Problème 1 : Connexion Redis

Erreur de Connexion

Message d'erreur : `Error: Redis connection to 127.0.0.1:6379 failed`
Analyse : L'application n'arrivait pas à joindre la base de données, car le serveur Redis n'était pas lancé sur la machine locale.

Résolution

1. J'ai ouvert un nouveau terminal.
2. J'ai exécuté la commande `redis-server` pour démarrer le service.
3. J'ai vérifié le statut avec `redis-cli ping` qui a retourné `PONG`.

Problème 2 : Échec des tests (Assertion Error)

TDD - Red Phase

Message d'erreur : `Uncaught AssertionError: expected null to be deep equal to { username: ... }`
Analyse : Lors de la première exécution des tests, le contrôleur n'était pas encore implémenté, il retournait donc `null` ou `undefined`. C'est le comportement normal du TDD.

Résolution

1. J'ai analysé le fichier `src/controllers/user.js`.
2. J'ai utilisé la méthode `db.hgetall` de la librairie Redis pour récupérer les données.
3. J'ai relancé `npm test` et le test est passé au vert.

5 Finalité du Lab

L'objectif du lab est-il rempli ? OUI.

Pourquoi ?

- ☑ La fonctionnalité `GET user` est pleinement opérationnelle.
- ☑ Tous les tests (unitaires et d'intégration) passent avec succès lors de l'exécution de la commande `npm test`.
- ☑ L'approche TDD a été respectée en écrivant les tests dans `user.controller.js` et `user.router.js` avant l'implémentation finale.