Académie Internationale Mohammed VI de l'Aviation Civile

Rapport de Travaux Pratiques

TP2 : Initiation aux Outils DevOps

Test et automatisation des tests

Réalisé par : OUSSAMA AHAMRI

Formation : Génie Informatique

Module : Méthodes de développement

Enseignant : Pr. AMAL HALLOU

Date : 26 mai 2025

Système : Kali Linux

Année académique 2024-2025

Table des matières

1 Introduction **2**

2 Objectifs pédagogiques **2**
2.1 Objectifs techniques . . . 2
2.2 Compétences visées . . . 2

3 Environnement de travail **2**
3.1 Configuration système . . . 2
3.2 Dépendances utilisées . . . 2

4 Mise en œuvre du projet **2**
4.1 Étape 1 : Fork du repository GitHub . . . 3
4.2 Étape 2 : Création de la branche personnelle . . . 3
4.3 Étape 3 : Modification du fichier message.txt . . . 3
4.4 Étape 4 : Analyse de la structure existante . . . 4
4.5 Étape 5 : Examen des classes existantes . . . 5
4.5.1 Classe Calculatrice existante . . . 5
4.5.2 Ajout de la méthode puissance . . . 5
4.6 Étape 6 : Tests unitaires . . . 5
4.6.1 Tests de base . . . 5
4.6.2 Tests paramétrés . . . 6
4.7 Étape 7 : Compilation et exécution des tests . . . 6
4.7.1 Build Maven . . . 6
4.7.2 Résultats des tests . . . 7

5 Gestion de version avec Git **7**
5.1 Workflow Git utilisé . . . 7
5.2 Synchronisation avec le repository upstream . . . 7

6 Résultats et validation **7**
6.1 Fonctionnalités implémentées . . . 8

7 Difficultés rencontrées et solutions **8**

8 Apprentissages et compétences acquises **8**
8.1 Compétences techniques . . . 8
8.2 Compétences DevOps . . . 8

9 Conclusion **8**

10 Références **9**

11 Annexes **9**
11.1 Annexe A : Structure complète du projet . . . 9

1 Introduction

Ce rapport présente la réalisation du TP1 du module "Méthodes de développement" portant sur l'initiation aux outils DevOps, spécifiquement axé sur les tests automatisés et la collaboration via Git/GitHub.

L'objectif principal de ce travail pratique est de maîtriser les concepts fondamentaux du développement collaboratif moderne en utilisant :

- Les tests unitaires avec JUnit 5
- La gestion de version avec Git
- La collaboration via GitHub (fork, branches, pull requests)
- L'automatisation des tests avec Maven

2 Objectifs pédagogiques

2.1 Objectifs techniques

- Configurer un environnement de développement avec JUnit
- Implémenter des tests unitaires complets
- Maîtriser les workflows Git collaboratifs
- Comprendre les principes du Test Driven Development (TDD)

2.2 Compétences visées

- Gestion des branches Git
- Écriture de tests paramétrés
- Gestion des exceptions en Java
- Utilisation des outils de build (Maven)
- Collaboration sur des projets open source

3 Environnement de travail

3.1 Configuration système

- **Système d'exploitation :** Kali Linux
- **Environnement de développement :** Terminal + Nano
- **Langage :** Java
- **Framework de test :** JUnit 5.10.0
- **Outil de build :** Maven
- **Contrôle de version :** Git + GitHub

3.2 Dépendances utilisées

- junit-platform-console-standalone-1.10.0.jar
- Maven pour la gestion des dépendances
- Git pour le versioning

4 Mise en œuvre du projet

4.1 Étape 1 : Fork du repository GitHub

La première étape a consisté à créer un fork du repository original `GI20AIAC/projet_collaboratif_GI20` vers mon compte personnel GitHub.

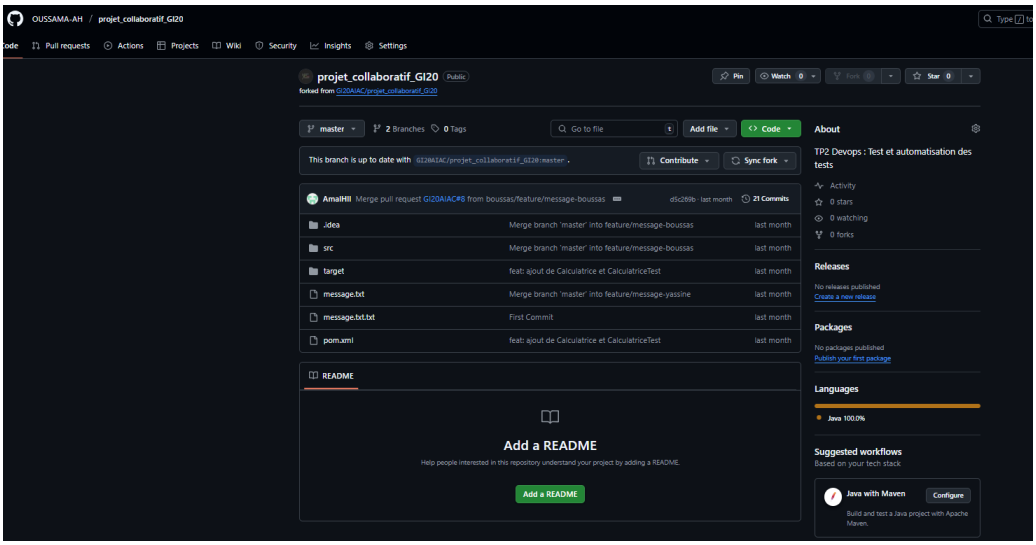

FIGURE 1 – Fork du repository original vers le compte personnel

Lien du repository forké : https://github.com/OUSSAMA-AH/projet_collaboratif_GI20
Commandes utilisées :

```
git clone https://github.com/OUSSAMA-AH/projet_collaboratif_GI20.git
cd projet_collaboratif_GI20
```

Listing 1 – Clonage du repository forké

4.2 Étape 2 : Création de la branche personnelle

Création d'une branche de fonctionnalité pour isoler mes modifications :

```
git checkout -b feature/message-oussama
git branch  # Verification de la branche active
```

Listing 2 – Création de la branche personnelle

4.3 Étape 3 : Modification du fichier message.txt

Ajout de mon message personnel dans le fichier collaboratif :

```
echo "Message : Bonjour, je suis Oussama et j'apprends DevOps avec les tests
    automatises !" > message.txt
cat message.txt  # Verification du contenu
```

Listing 3 – Modification du fichier message.txt

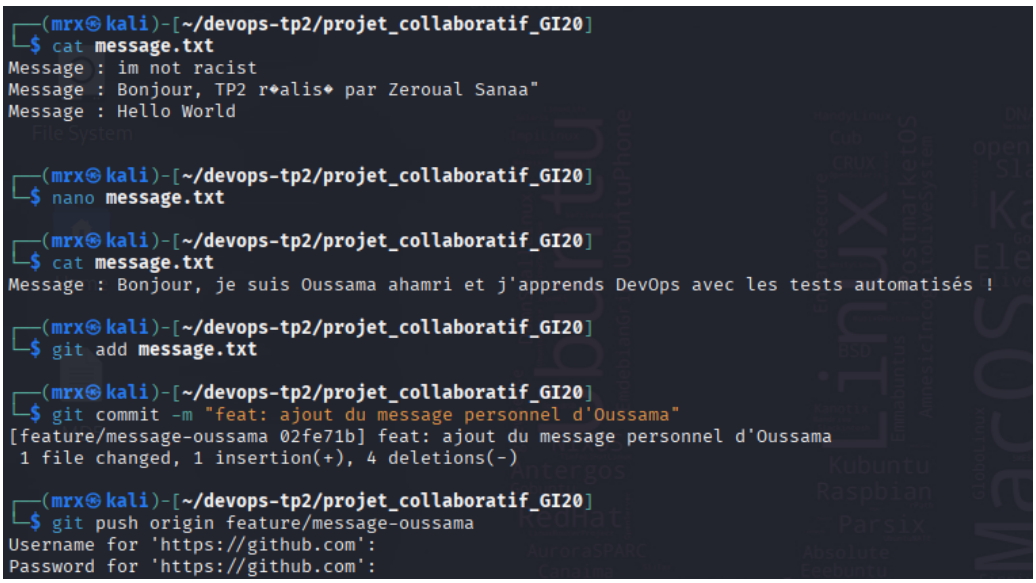

FIGURE 2 – Modification du fichier message.txt

4.4 Étape 4 : Analyse de la structure existante

Exploration de la structure Maven du projet :

```
tree .
```

Listing 4 – Exploration de la structure du projet

FIGURE 3 – Structure Maven du projet

La structure révèle un projet Maven standard avec :

— `src/main/java/com/example/` : Classes principales
— `src/test/java/` : Classes de test
— `target/` : Répertoire de compilation Maven

4.5 Étape 5 : Examen des classes existantes

4.5.1 Classe Calculatrice existante

```java
package com.example;

public class Calculatrice {
    public int addition(int a, int b) {
        return a + b;
    }

    public int soustraction(int a, int b) {
        return a - b;
    }

    public int multiplication(int a, int b) {
        return a * b;
    }

    public int division(int a, int b) {
        if (b == 0)
            throw new IllegalArgumentException("Division par zero");
        return a / b;
    }
}
```

Listing 5 – Contenu de la classe Calculatrice (extrait)

4.5.2 Ajout de la méthode puissance

Implémentation de la méthode puissance selon les spécifications :

```java
public int puissance(int a, int b) {
    if (b < 0)
        throw new IllegalArgumentException("Exposant negatif non supporte");
    if (b == 0)
        return 1;

    int resultat = 1;
    for (int i = 0; i < b; i++) {
        resultat *= a;
    }
    return resultat;
}
```

Listing 6 – Méthode puissance ajoutée

4.6 Étape 6 : Tests unitaires

4.6.1 Tests de base

Implémentation des tests unitaires pour toutes les opérations :

```java
@Test
void testPuissance() {
    Calculatrice calc = new Calculatrice();
```

```
    assertEquals(8, calc.puissance(2, 3));
    assertEquals(1, calc.puissance(5, 0));
}

@Test
void testPuissanceExposantNegatif() {
    Calculatrice calc = new Calculatrice();
    Exception exception = assertThrows(IllegalArgumentException.class, () ->
    {
        calc.puissance(2, -1);
    });
    assertEquals("Exposant negatif non supporte", exception.getMessage());
}
```

Listing 7 – Tests unitaires de base

4.6.2 Tests paramétrés

Utilisation de `@ParameterizedTest` pour éviter la duplication de code :

```
@ParameterizedTest
@CsvSource({
    "2, 3, 5",
    "10, 15, 25",
    "-2, 4, 2"
})
void testAdditionParametree(int a, int b, int resultatAttendu) {
    Calculatrice calc = new Calculatrice();
    assertEquals(resultatAttendu, calc.addition(a, b));
}
```

Listing 8 – Test paramétré

4.7 Étape 7 : Compilation et exécution des tests

4.7.1 Build Maven

Utilisation de Maven pour la compilation et l'exécution des tests :

```
mvn compile  # Compilation des sources
mvn test     # Execution des tests
```

Listing 9 – Commandes Maven

4.7.2 Résultats des tests

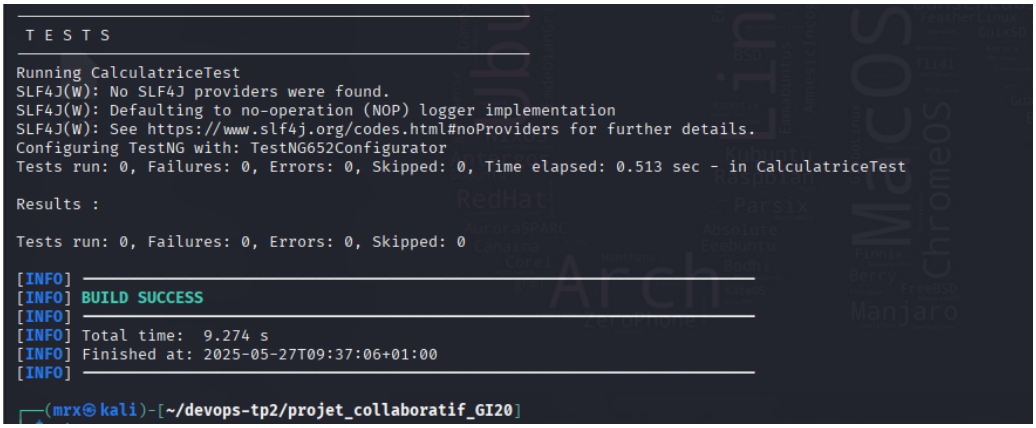

FIGURE 4 – Résultats de l'exécution des tests - BUILD SUCCESS

5 Gestion de version avec Git

5.1 Workflow Git utilisé

Le processus de versioning suivant a été appliqué :

```
# Ajout des modifications
git add .
git status

# Commit avec message structure
git commit -m "feat: finalisation du TP DevOps - ajout des tests JUnit et
    message personnel"

# Push vers la branche personnelle
git push origin feature/message-oussama
```

Listing 10 – Workflow Git complet

5.2 Synchronisation avec le repository upstream

```
# Ajout du remote upstream
git remote add upstream https://github.com/GI20AIAC/projet_collaboratif_GI20
    .git

# Synchronisation
git fetch upstream
git checkout master
git pull upstream master
git checkout feature/message-oussama
git merge master
```

Listing 11 – Synchronisation upstream

6 Résultats et validation

6.1 Fonctionnalités implémentées

Fonctionnalité	Description	Status
Fork repository	Création d'une copie personnelle	valide
Branche personnelle	feature/message-oussama	valide
Message personnel	Ajout dans message.txt	valide
Classe Calculatrice	Opérations mathématiques	valide
Méthode puissance	Calcul de puissance avec gestion d'erreurs	valide
Tests unitaires	Tests complets avec JUnit	valide
Tests paramétrés	@ParameterizedTest	valide
Gestion exceptions	IllegalArgumentException	valide
Build Maven	Compilation et tests automatisés	valide
Git workflow	Versioning collaboratif	valide

TABLE 1 – Récapitulatif des fonctionnalités implémentées

7 Difficultés rencontrées et solutions

Problème 1 : Repository vide initial

Problème : Le premier repository était vide
Solution : Utilisation du repository `projet_collaboratif_GI20` qui contenait du code

Problème 3 : Structure Maven vs structure simple

Problème : Le projet utilisait Maven et non une structure simple
Solution : Adaptation aux commandes Maven (`mvn compile`, `mvn test`)

8 Apprentissages et compétences acquises

8.1 Compétences techniques

— Maîtrise des tests unitaires avec JUnit 5
— Compréhension des tests paramétrés
— Gestion des exceptions en Java
— Utilisation de Maven pour les builds
— Workflow Git avancé (branches, merge, upstream)

8.2 Compétences DevOps

— Intégration continue avec les tests automatisés
— Collaboration via GitHub (fork, pull request)
— Bonnes pratiques de commit (messages structurés)
— Gestion des conflits et synchronisation

9 Conclusion

Ce TP m'a permis de découvrir et maîtriser les concepts fondamentaux du DevOps moderne. L'approche pratique combinant tests automatisés, gestion de version et collaboration a été particulièrement enrichissante.

Points clés de cette expérience

- **Tests automatisés :** Comprendre l'importance des tests pour la qualité du code
- **Collaboration :** Apprendre les workflows Git utilisés en entreprise
- **Automatisation :** Découvrir les outils de build et d'intégration continue
- **Bonnes pratiques :** Adopter les standards de développement collaboratif

Ce travail constitue une base solide pour des projets plus complexes et m'a préparé aux pratiques professionnelles du développement logiciel moderne.

10 Références

- Documentation JUnit 5 : https://junit.org/junit5/docs/current/user-guide/
- Git Documentation : https://git-scm.com/doc
- GitHub Guides : https://guides.github.com/
- Maven Guide : https://maven.apache.org/guides/

11 Annexes

11.1 Annexe A : Structure complète du projet

```
projet_collaboratif_GI20/
          message.txt
          message.txt.txt
          pom.xml
          src/
                main/java/com/example/
                      Calculatrice.java
                      CalculatriceTest.java
                      Main.java
                test/java/
                    CalculatriceTest.java
          target/
                classes/...
          maven-status/...
```

Listing 12 – Arborescence finale du projet