



Académie Internationale Mohammed VI de l'Aviation Civile

Rapport de Travaux Pratiques

TP1 : Initiation aux Outils DevOps
Git, GitHub et Maven

Lien du repository : <https://github.com/OUSSAMA-AH/MonProjetJava>

Informations du Projet

Étudiant : OUSSAMA AHAMRI
Formation : Génie Informatique
Module : Méthodes de développement
Enseignant : Pr. AMAL HALLOU
Date : 4 juin 2025
Système : Kali Linux

Année académique 2024-2025

Table des matières

1	Introduction	4
1.1	Contexte du TP	4
1.2	Objectifs Pédagogiques	4
1.3	Environnement de Travail	4
2	Préparation de l'Environnement	4
2.1	Installation et Configuration de Java	4
2.1.1	Vérification de Java Runtime	4
2.1.2	Installation du JDK Complet	5
2.2	Installation et Configuration de Maven	5
2.3	Configuration de Git	5
3	Création du Projet Maven	6
3.1	Génération du Projet avec Archetype	6
3.2	Structure du Projet Générée	6
3.3	Analyse du fichier pom.xml	7
4	Initialisation du Contrôle de Version	8
4.1	Initialisation du Repository Git	8
4.2	Création du fichier .gitignore	8
4.3	Premier Commit	9
5	Tests et Validation Maven	9
5.1	Compilation du Projet	9
5.2	Exécution des Tests Unitaires	10
5.3	Création du Package	10
6	Intégration avec GitHub	10
6.1	Création du Repository GitHub	10
6.2	Configuration du Remote	11
6.3	Authentification : Personal Access Token	11
7	Documentation et README	11
7.1	Création du README.md	11

7.2	Repository Final sur GitHub	12
8	Analyse des Résultats	13
8.1	Objectifs Atteints	13
8.2	Compétences Développées	13
8.2.1	Maîtrise des Outils DevOps	13
8.2.2	Bonnes Pratiques	13
9	Workflow DevOps Établi	13
9.1	Cycle de Développement Mis en Place	13
9.2	Bonnes Pratiques Appliquées	14
10	Conclusion	14
10.1	Bilan du TP	14

Table des figures

1	Vérification de la version Java installée	5
2	Vérification de Maven	5
3	Configuration de Git	6
4	Génération du projet Maven	6
5	Structure du projet Maven générée	7
6	Initialisation du repository Git	8
7	Premier commit du projet	9
8	Compilation réussie avec Maven	10
9	Exécution des tests unitaires	10
10	Push réussi vers GitHub	11
11	Création du fichier README.md	12
12	Repository GitHub final avec README	12

1 Introduction

1.1 Contexte du TP

Ce rapport présente la réalisation du TP1 du module "Méthodes de développement" portant sur l'initiation aux outils DevOps. Ce travail pratique vise à familiariser les étudiants avec les outils fondamentaux du développement logiciel moderne, en particulier Git pour le contrôle de version, GitHub pour la collaboration, et Maven pour la gestion de projets Java.

1.2 Objectifs Pédagogiques

Les objectifs principaux de ce TP sont :

- Maîtriser les concepts de base du contrôle de version avec Git
- Comprendre l'utilisation de GitHub comme plateforme de collaboration
- Apprendre à utiliser Maven pour la gestion de projets Java
- Mettre en place un workflow de développement moderne
- Intégrer les bonnes pratiques DevOps dans un projet simple

1.3 Environnement de Travail

Le TP a été réalisé dans l'environnement suivant :

- **Système d'exploitation** : Kali Linux 2024.2
- **Java** : OpenJDK 23.0.2
- **Maven** : Version 3.9+
- **Git** : Version 2.43+
- **IDE/Éditeur** : Terminal + Nano

2 Préparation de l'Environnement

2.1 Installation et Configuration de Java

La première étape consistait à vérifier et installer Java Development Kit (JDK) sur le système Kali Linux.

2.1.1 Vérification de Java Runtime

```
1 $ java -version
2 openjdk version "23.0.2" 2025-01-21
3 OpenJDK Runtime Environment (build 23.0.2+7-Debian-1)
4 OpenJDK 64-Bit Server VM (build 23.0.2+7-Debian-1, mixed mode, sharing)
```

Listing 1 – Vérification de la version Java

```
(mrx@kali)-[~]
$ java -version
openjdk version "23.0.2" 2025-01-21
OpenJDK Runtime Environment (build 23.0.2+7-Debian-1)
OpenJDK 64-Bit Server VM (build 23.0.2+7-Debian-1, mixed mode, sharing)

(mrx@kali)-[~]
$
```

FIGURE 1 – Vérification de la version Java installée

2.1.2 Installation du JDK Complet

Pour résoudre ce problème, l'installation du JDK complet était nécessaire :

```
1 $ sudo apt update
2 $ sudo apt install openjdk-21-jdk
```

Listing 2 – Installation du JDK

2.2 Installation et Configuration de Maven

Maven étant essentiel pour la gestion de projet Java, son installation était la deuxième étape cruciale.

```
1 $ sudo apt install maven -y
2 $ mvn -version
```

Listing 3 – Installation de Maven

```
(mrx@kali)-[~]
$ mvn -version
Apache Maven 3.9.9
Maven home: /usr/share/maven
Java version: 23.0.2, vendor: Debian, runtime: /usr/lib/jvm/java-23-openjdk-amd64
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.12.25-amd64", arch: "amd64", family: "unix"

(mrx@kali)-[~]
$
```

FIGURE 2 – Vérification de Maven

2.3 Configuration de Git

Git était déjà installé sur le système Kali Linux, mais une configuration était nécessaire :

```
1 $ git --version
2 $ git config --global user.name "OUSSAMA AHAMRI"
3 $ git config --global user.email "oussama.ahamri@aiaac.ma"
```

Listing 4 – Configuration Git

```
(mrX@kali)-[~/Projets-DevOps/MonProjetJava]
$ git config --global user.name "OUSSAMA AHAMRI"

(mrX@kali)-[~/Projets-DevOps/MonProjetJava]
$ git config --global user.email "oussama.ahamri@aiaac.ma"
```

FIGURE 3 – Configuration de Git

3 Création du Projet Maven

3.1 Génération du Projet avec Archetype

La création d'un projet Maven s'effectue via la commande archetype :generate qui génère automatiquement la structure standard d'un projet Java.

```
1 $ mkdir -p ~/Projets-DevOps
2 $ cd ~/Projets-DevOps
3 $ mvn archetype:generate \
4   -DgroupId=com.example \
5   -DartifactId=MonProjetJava \
6   -DarchetypeArtifactId=maven-archetype-quickstart \
7   -DinteractiveMode=false
```

Listing 5 – Génération du projet Maven

```
[INFO] Generating project in Batch mode
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml
Downloaded from central: https://repo.maven.apache.org/maven2/archetype-catalog.xml (17 MB at 1.0 MB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (4.3 kB at 77 KB/s)
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] Parameter: basedir, Value: /home/mrx/Projets-DevOps
[INFO] Parameter: package, Value: com.example
[INFO] Parameter: groupId, Value: com.example
[INFO] Parameter: artifactId, Value: MonProjetJava
[INFO] Parameter: packaging, Value: com.example
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Project created from Old (1.x) Archetype in dir: /home/mrx/Projets-DevOps/MonProjetJava
[INFO] BUILD SUCCESS
[INFO] Total time: 01:04 min
[INFO] Finished at: 2025-05-25T19:25:22+01:00
[INFO]
```

FIGURE 4 – Génération du projet Maven

3.2 Structure du Projet Générée

Après la génération, Maven crée automatiquement la structure de projet suivante :

```
1 $ cd MonProjetJava
2 $ tree .
```

Listing 6 – Exploration de la structure

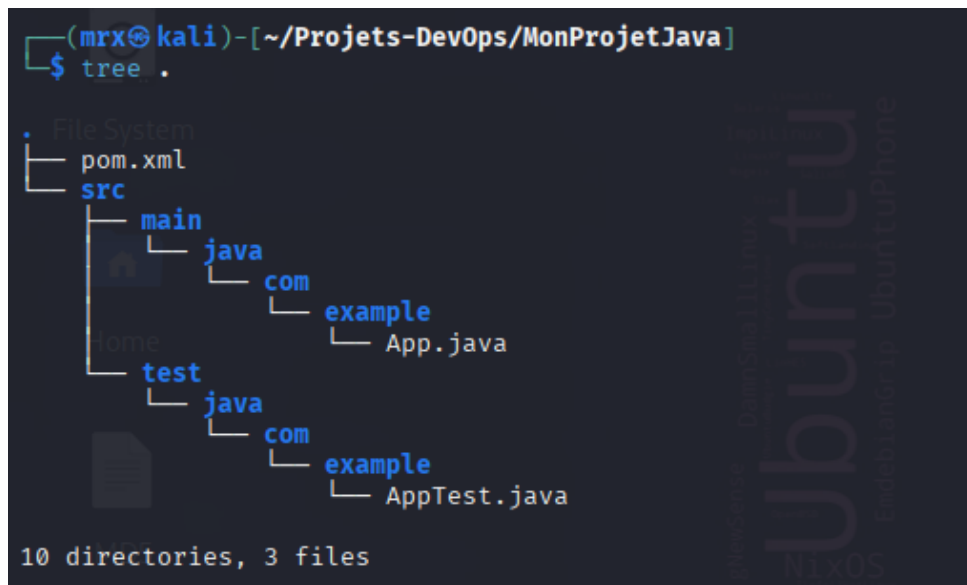


FIGURE 5 – Structure du projet Maven générée

La structure standard comprend :

- `pom.xml` : Fichier de configuration Maven
- `src/main/java/` : Code source principal
- `src/test/java/` : Tests unitaires
- `src/main/java/com/example/App.java` : Classe principale
- `src/test/java/com/example/AppTest.java` : Classe de test

3.3 Analyse du fichier `pom.xml`

Le fichier `pom.xml` contient la configuration du projet :

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5                             http://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7   <groupId>com.example</groupId>
8   <artifactId>MonProjetJava</artifactId>
9   <version>1.0-SNAPSHOT</version>
10  <packaging>jar</packaging>
11
12  <dependencies>
13    <dependency>
14      <groupId>junit</groupId>
15      <artifactId>junit</artifactId>
16      <version>4.11</version>
17      <scope>test</scope>
18    </dependency>
19  </dependencies>
20 </project>
  
```

Listing 7 – Contenu du `pom.xml` initial

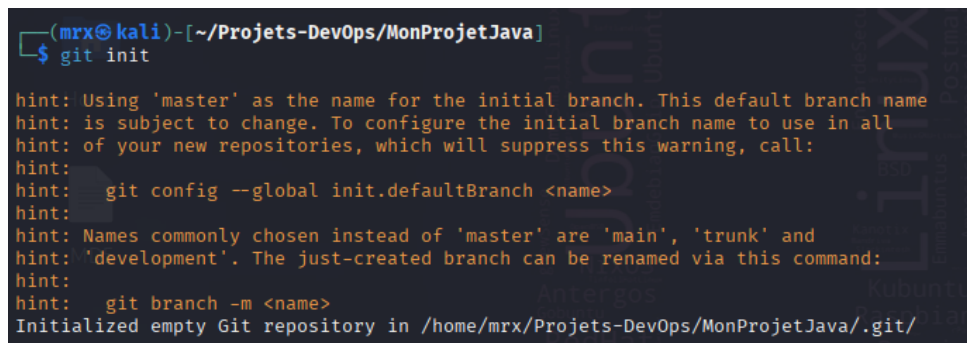
4 Initialisation du Contrôle de Version

4.1 Initialisation du Repository Git

La première étape du contrôle de version consiste à initialiser un repository Git local :

```
1 $ git init
```

Listing 8 – Initialisation Git



```
(mrxc@kali)-[~/Projets-DevOps/MonProjetJava]
$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/mrx/Projets-DevOps/MonProjetJava/.git/
```

FIGURE 6 – Initialisation du repository Git

Git affiche des hints concernant l'utilisation de "main" au lieu de "master" comme branche par défaut, reflétant les bonnes pratiques modernes.

4.2 Création du fichier .gitignore

Un fichier .gitignore est essentiel pour exclure les fichiers temporaires et générés :

```
1 $ nano .gitignore
```

Listing 9 – Création du .gitignore

Contenu du fichier .gitignore :

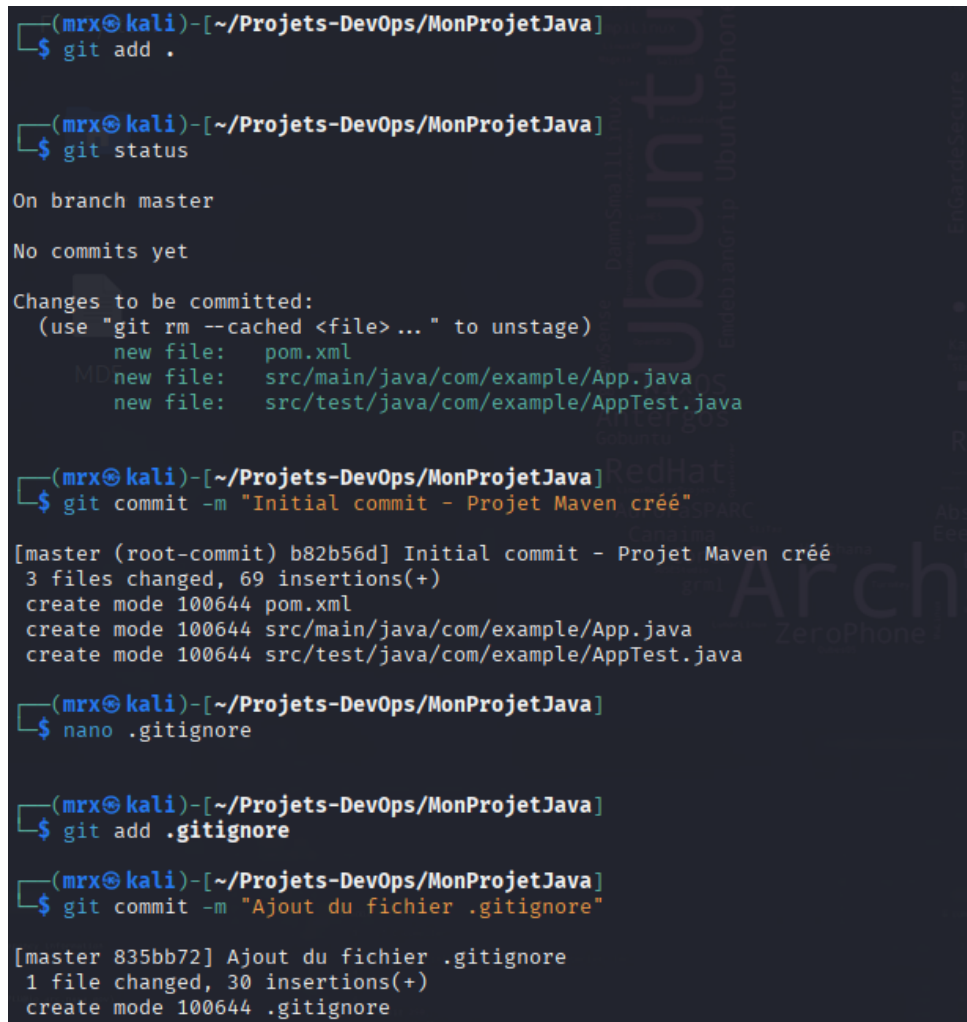
```
1 # Maven
2 target/
3 pom.xml.tag
4 pom.xml.releaseBackup
5 pom.xml.versionsBackup
6
7 # IDE
8 .idea/
9 *.iml
10 .vscode/
11
12 # Logs
13 *.log
14
15 # OS
16 .DS_Store
17 Thumbs.db
```

Listing 10 – Contenu du .gitignore

4.3 Premier Commit

```
1 $ git add .
2 $ git commit -m "Initial commit - Projet Maven cree"
```

Listing 11 – Premier commit



```
(mrx@kali)-[~/Projets-DevOps/MonProjetJava]
$ git add .

(mrx@kali)-[~/Projets-DevOps/MonProjetJava]
$ git status

On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file> ..." to unstage)
        new file:   pom.xml
        new file:   src/main/java/com/example/App.java
        new file:   src/test/java/com/example/AppTest.java

(mrx@kali)-[~/Projets-DevOps/MonProjetJava]
$ git commit -m "Initial commit - Projet Maven créé"

[master (root-commit) b82b56d] Initial commit - Projet Maven créé
 3 files changed, 69 insertions(+)
 create mode 100644 pom.xml
 create mode 100644 src/main/java/com/example/App.java
 create mode 100644 src/test/java/com/example/AppTest.java

(mrx@kali)-[~/Projets-DevOps/MonProjetJava]
$ nano .gitignore

(mrx@kali)-[~/Projets-DevOps/MonProjetJava]
$ git add .gitignore

(mrx@kali)-[~/Projets-DevOps/MonProjetJava]
$ git commit -m "Ajout du fichier .gitignore"

[master 835bb72] Ajout du fichier .gitignore
 1 file changed, 30 insertions(+)
 create mode 100644 .gitignore
```

FIGURE 7 – Premier commit du projet

5 Tests et Validation Maven

5.1 Compilation du Projet

Le test de compilation vérifie que le code source est syntaxiquement correct :

```
1 $ mvn compile
```

Listing 12 – Compilation Maven

```
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0.3/qdox-2.0.3.jar (334 kB at 702 kB/s)
[INFO] Recompiling the module because of changed source code.
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 1 source file with javac [debug target 1.8] to target/classes
[WARNING] bootstrap class path is not set in conjunction with -source 8
not setting the bootstrap class path may lead to class files that cannot run on JDK 8
--release 8 is recommended instead of -source 8 -target 1.8 because it sets the bootstrap class path automatically
[WARNING] source value 8 is obsolete and will be removed in a future release
[WARNING] target value 8 is obsolete and will be removed in a future release
[WARNING] To suppress warnings about obsolete options, use -Xlint:-options.
[INFO] BUILD SUCCESS
[INFO] Total time: 5.956 s
[INFO] Finished at: 2025-05-25T19:36:25+01:00
[INFO]
```

FIGURE 8 – Compilation réussie avec Maven

5.2 Exécution des Tests Unitaires

Maven exécute automatiquement les tests JUnit :

```
1 $ mvn test
```

Listing 13 – Exécution des tests

```
TESTS
Running com.example.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.037 sec - in com.example.AppTest
Results :
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO] BUILD SUCCESS
[INFO] Total time: 7.841 s
[INFO] Finished at: 2025-05-25T19:37:11+01:00
[INFO]
```

FIGURE 9 – Exécution des tests unitaires

5.3 Création du Package

La création du package JAR finalise le processus de build :

```
1 $ mvn package
```

Listing 14 – Création du package

Cette commande génère un fichier JAR exécutable dans le répertoire `target/`.

6 Intégration avec GitHub

6.1 Création du Repository GitHub

Un repository distant a été créé sur GitHub avec les caractéristiques suivantes :

- Nom : MonProjetJava
- Visibilité : Public
- Description : TP1 DevOps - Maven et Git sous Linux

6.2 Configuration du Remote

```
1 $ git remote add origin https://github.com/OUSSAMA-AH/MonProjetJava.git
2 $ git remote -v
```

Listing 15 – Ajout du remote GitHub

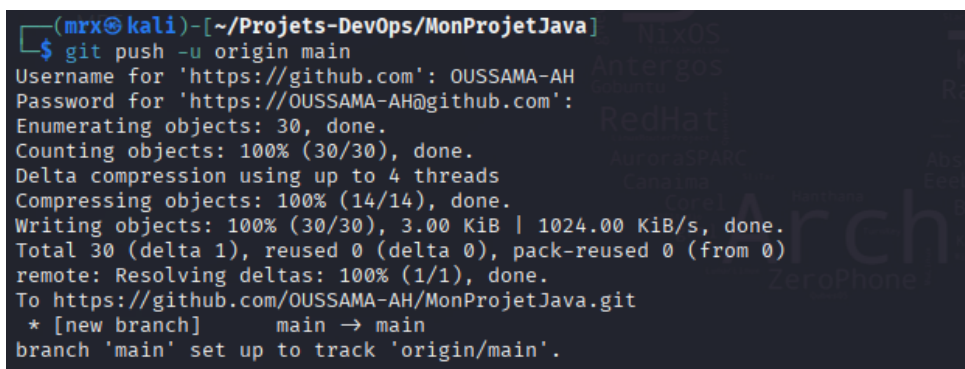
6.3 Authentification : Personal Access Token

La solution consistait à créer un Personal Access Token (PAT) :

1. Accès aux paramètres GitHub : Settings → Developer settings
2. Génération d'un nouveau token avec les permissions **repo**
3. Utilisation du token comme mot de passe lors du push

```
1 $ git push -u origin main
2 # Username: OUSSAMA-AH
3 # Password: [Personal Access Token]
```

Listing 16 – Push réussi avec token



```
(mrX@kali) - [~/Projets-DevOps/MonProjetJava]
$ git push -u origin main
Username for 'https://github.com': OUSSAMA-AH
Password for 'https://OUSSAMA-AH@github.com':
Enumerating objects: 30, done.
Counting objects: 100% (30/30), done.
Delta compression using up to 4 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (30/30), 3.00 KiB | 1024.00 KiB/s, done.
Total 30 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/OUSSAMA-AH/MonProjetJava.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

FIGURE 10 – Push réussi vers GitHub

7 Documentation et README

7.1 Création du README.md

Un fichier README.md complet a été créé pour documenter le projet :

```
1 $ nano README.md
```

Listing 17 – Création du README

```
(mrxc@kali)-[~/Projets-DevOps/MonProjetJava]
$ git add README.md

(mrxc@kali)-[~/Projets-DevOps/MonProjetJava]
$ git commit -m "Ajout du fichier README.md avec documentation complète"
[main 7f0458b] Ajout du fichier README.md avec documentation complète
1 file changed, 29 insertions(+)
create mode 100644 README.md

(mrxc@kali)-[~/Projets-DevOps/MonProjetJava]
$ git push origin main

Username for 'https://github.com': OUSSAMA-AH
Password for 'https://OUSSAMA-AH@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 955 bytes | 955.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/OUSSAMA-AH/MonProjetJava.git
```

FIGURE 11 – Création du fichier README.md

7.2 Repository Final sur GitHub

Après l'ajout du README, le repository GitHub présente une interface complète :

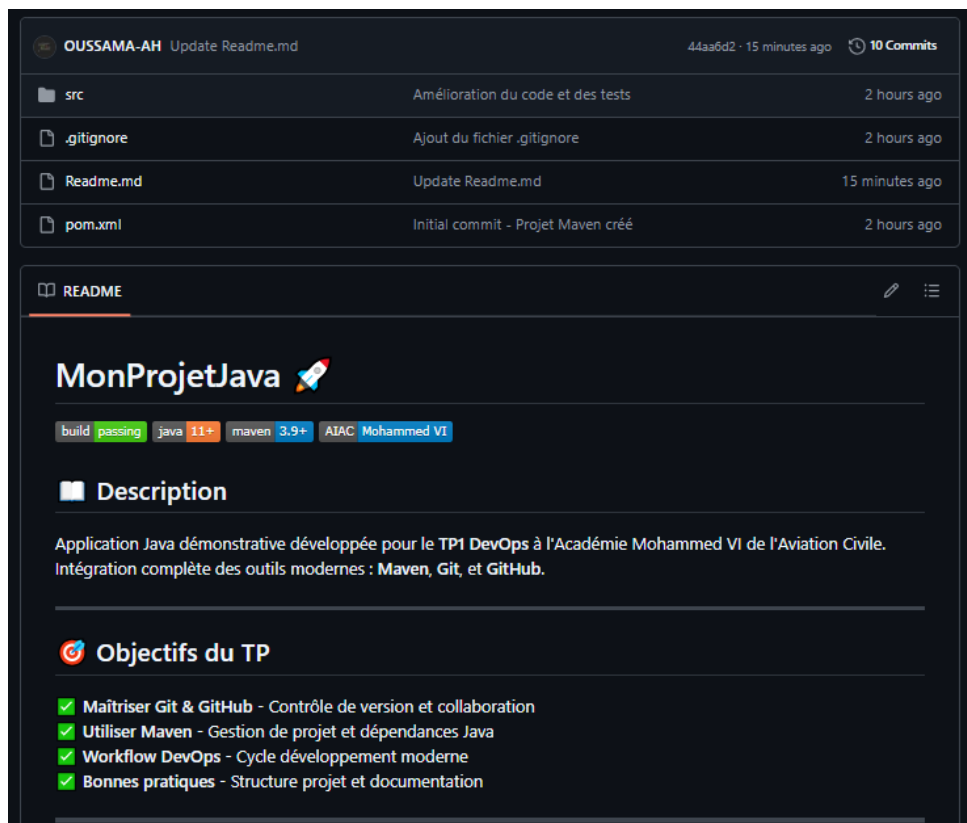


FIGURE 12 – Repository GitHub final avec README

Le repository contient :

- Structure complète du projet Maven
- Documentation détaillée dans le README
- Historique des commits
- Fichier .gitignore approprié

8 Analyse des Résultats

8.1 Objectifs Atteints

Objectifs Réalisés avec Succès

- Installation et configuration complète de l'environnement DevOps
- Création d'un projet Java Maven structuré
- Maîtrise des commandes Git fondamentales
- Intégration réussie avec GitHub
- Mise en place du workflow de développement moderne
- Documentation professionnelle du projet

8.2 Compétences Développées

Ce TP a permis de développer plusieurs compétences techniques essentielles :

8.2.1 Maîtrise des Outils DevOps

- Configuration d'environnement de développement sur Linux
- Utilisation avancée de Git pour le contrôle de version
- Gestion de projets Java avec Maven
- Intégration avec les plateformes de collaboration (GitHub)

8.2.2 Bonnes Pratiques

- Structure de projet standardisée
- Gestion appropriée des fichiers avec .gitignore
- Documentation technique complète
- Workflow de commit et push structuré

9 Workflow DevOps Établi

9.1 Cycle de Développement Mis en Place

Le TP a établi un workflow de développement moderne suivant le schéma :

1. Développement Local

- Modification du code source
- Tests locaux avec Maven (`mvn test`)
- Compilation (`mvn compile`)

2. Contrôle de Version

- Ajout des modifications (`git add`)
- Commit avec message descriptif (`git commit`)
- Vérification du statut (`git status`)

3. Collaboration

- Push vers le repository distant (`git push`)
- Synchronisation avec GitHub
- Documentation mise à jour

9.2 Bonnes Pratiques Appliquées

- Structure de projet standardisée avec Maven
- Nomenclature cohérente des commits
- Documentation technique complète
- Gestion appropriée des secrets (Personal Access Token)
- Exclusion des fichiers temporaires (`.gitignore`)

10 Conclusion

10.1 Bilan du TP

Ce TP1 d'initiation aux outils DevOps a été un succès complet. Il a permis de maîtriser les fondamentaux du développement logiciel moderne en combinant :

- **Gestion de projet** avec Maven
- **Contrôle de version** avec Git
- **Collaboration** avec GitHub
- **Documentation** technique professionnelle