

FACULTATEA CALCULATOARE, INFORMATICA SI MICROELECTRONICA

UNIVERSITATEA TEHNICA A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A PRODUSELOR SOFT

LUCRAREA DE LABORATOR#1

Version Control System

Autor:

Eugen MIROVSCHI

lector asistent:

Irina COJANU

lector superior:

Svetlana COJOCARU

Laboratory work #2

1 Scopul lucrării de laborator

Studiere și utilizarea unui sistem de control a versiunilor. Inițializarea unui repository și adăugarea versinilor noi

2 Obiective

- a) Înțelegerea și folosirea CLI (basic level)
- b) Version Control Systems (git — bitbucket — mercurial — svn)

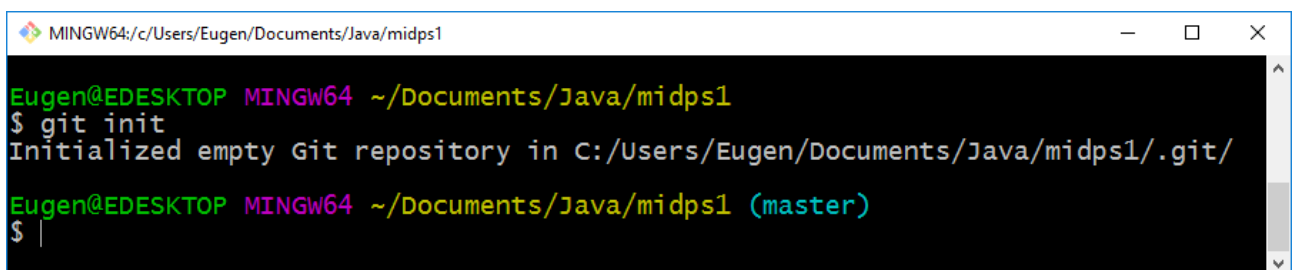
3 Laboratory work implementation

3.1 Tasks and Points

- a) Inițializarea unui nou repozitoriu
- b) Configurarea VC
- c) Crearea branch-urilor (cel puțin 2)
- d) Commit pe ambele branch-uri (cel puțin 1 commit per branch)
- e) Setarea unui branch sa urmărească un remote origin pe care se poate să faci push (GitHub)
- f) Resetarea unui branch pe commit-ul anterior
- g) Folosirea fișierului .gitignore
- h) Merge dintre 2 branch-uri
- i) Rezolvarea conflictelor a 2 branch-uri
- j) Folosirea tag-urilor pentru marcarea schimbarilor semnificative precum release

3.2 Analiza lucrarii de laborator

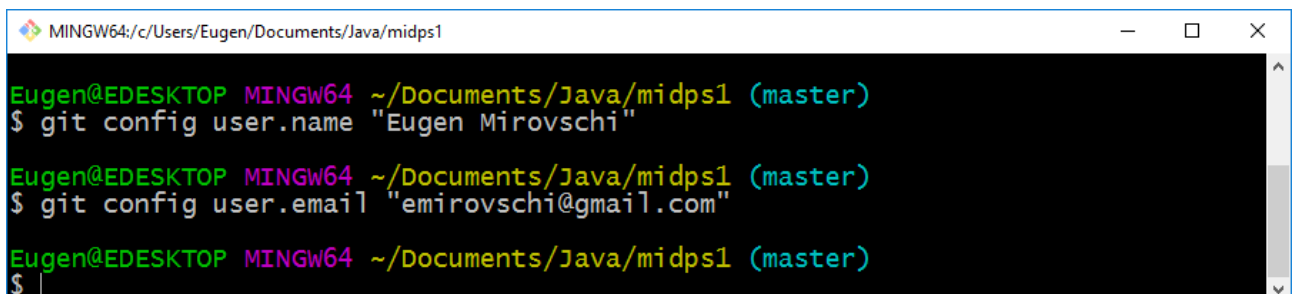
- a) Primul pas a fost inițializarea unui nou repozitoriu



```
MINGW64:/c:/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1
$ git init
Initialized empty Git repository in C:/Users/Eugen/Documents/Java/midps1/.git/
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ |
```

Figure 3.1 – Inițializarea repozitoriului

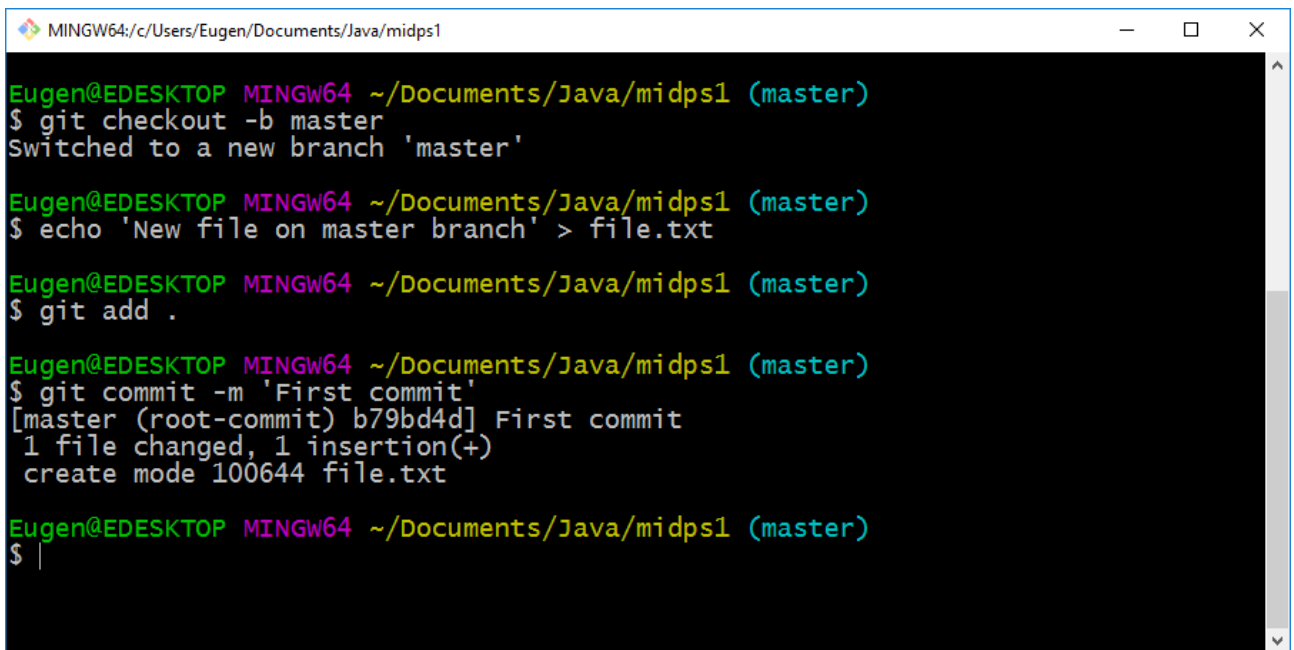
- b) Pentru a defini autorul versiunilor am folosit comanda **git config**



```
MINGW64:/c:/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git config user.name "Eugen Mirovski"
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git config user.email "emirovski@gmail.com"
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ |
```

Figure 3.2 – Configurarea repozitoriului

- c) Branch-ul master a fost create utilizând comanda **git checkout -b master** după care a fost adăugat un fișier nou care este inclus în primul commit



```
MINGW64:/c/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git checkout -b master
Switched to a new branch 'master'

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ echo 'New file on master branch' > file.txt

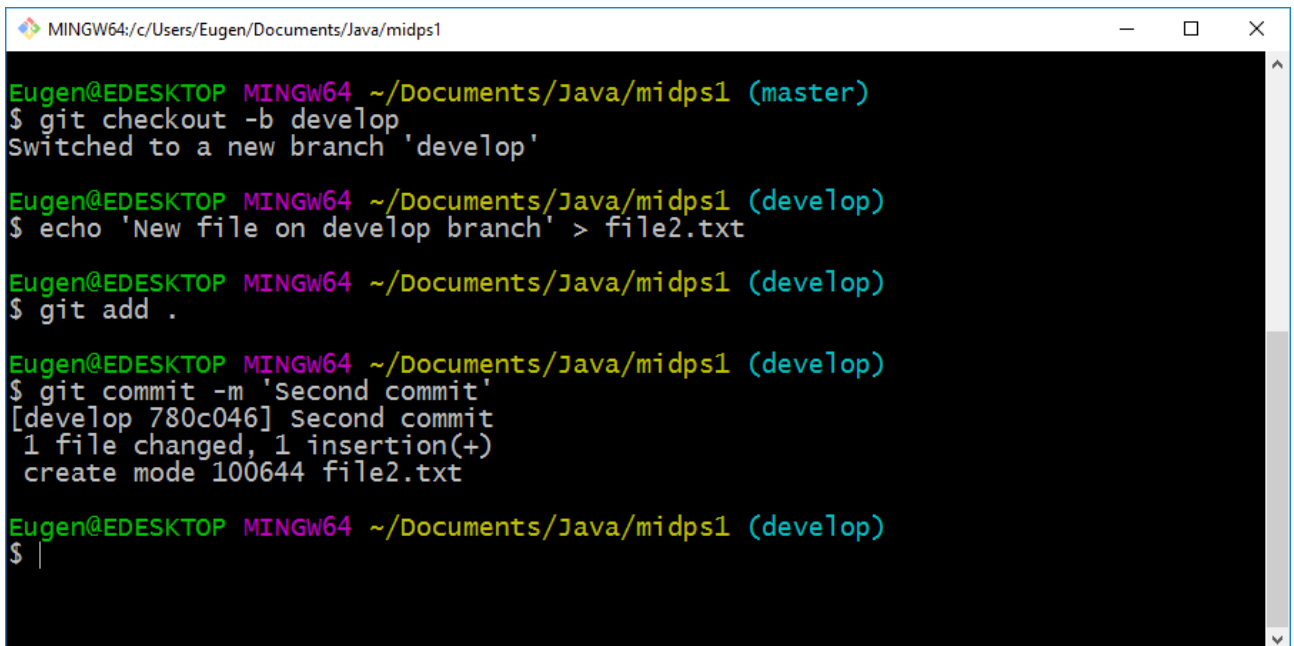
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git add .

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git commit -m 'First commit'
[master (root-commit) b79bd4d] First commit
1 file changed, 1 insertion(+)
create mode 100644 file.txt

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$
```

Figure 3.3– Crearea master branch

- d) Branch-ul develop a fost creat similar ca și master însă acesta are automat ca parent primul commit din master.



```
MINGW64:/c/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git checkout -b develop
Switched to a new branch 'develop'

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ echo 'New file on develop branch' > file2.txt

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ git add .

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ git commit -m 'Second commit'
[develop 780c046] Second commit
1 file changed, 1 insertion(+)
create mode 100644 file2.txt

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$
```

Figure 3.4– Crearea develop branch

- e) Pentru a adăuga un remote, inițial am creat un repository nou în GitHub după care am rulat comanda de adăugare a referinței în repositoryul local **git remote add**. După aceasta am mutat toate schimbarile pe origin folosind **git push**. Repositoryul folosind în acest caz este <https://github.com/emirovschi/MIDPS-1>.

```
MINGW64:/c/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git remote add origin https://github.com/emirovschi/MIDPS-1.git

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git push --set-upstream origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 240 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/emirovschi/MIDPS-1.git
* [new branch]      master -> master
Branch master set up to track remote branch master from origin.

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git push origin develop
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/emirovschi/MIDPS-1.git
* [new branch]      develop -> develop

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ |
```

Figure 3.5– Adăugare remote

- f) Pentru a reseta branch-ul curent la comitul anterior am folosit comanda **git reset HEAD** .

```
MINGW64:/c/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ echo 'Third file' > file3.txt

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git add .

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git commit -m 'Add new file'
[master faf089f] Add new file
1 file changed, 1 insertion(+)
create mode 100644 file3.txt

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git reset HEAD~

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

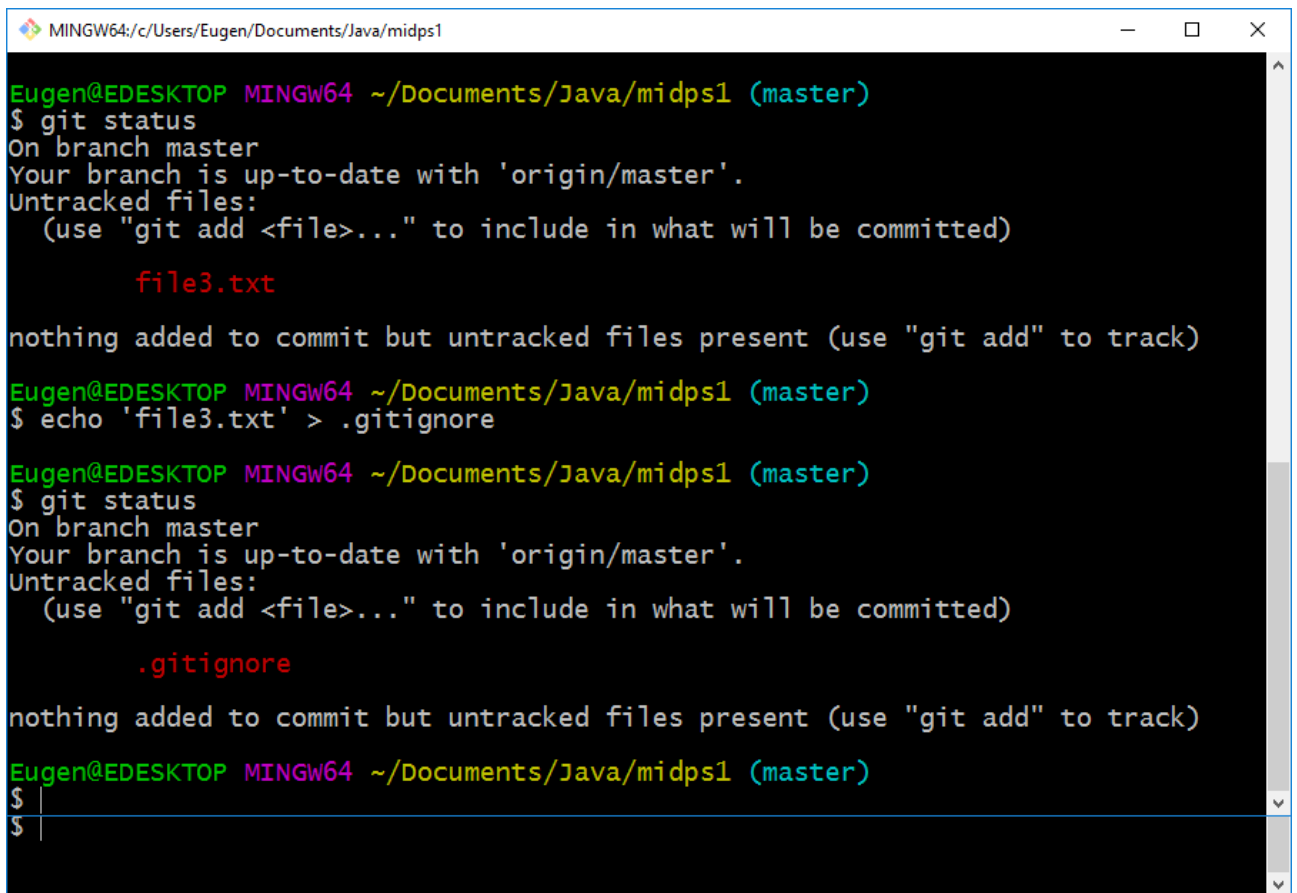
    file3.txt

nothing added to commit but untracked files present (use "git add" to track)

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ |
```

Figure 3.6– Resetarea ultimului commit

- g) Fișierul *.gitignore* permite excluderea anumitor fișiere în dependență de denumirea acestora. În exemplul dat am exclus fișierul creat în pasul anterior.



```
MINGW64:/c:/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file3.txt

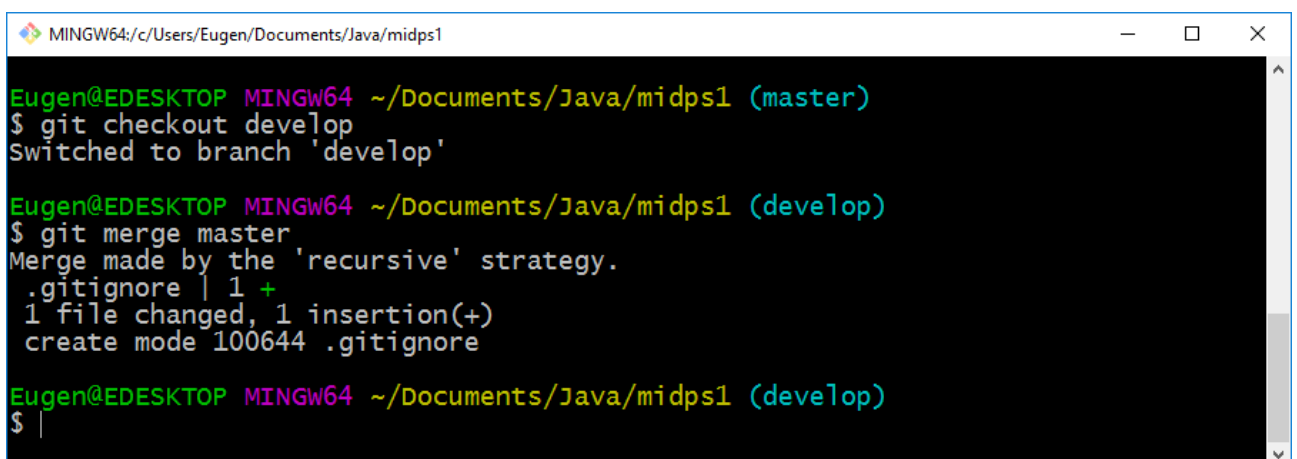
nothing added to commit but untracked files present (use "git add" to track)
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ echo 'file3.txt' > .gitignore
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ 
$
```

Figure 3.7– Adăugarea unui fișier în *.gitignore*

- h) Merge între branch-uri se face utilizând comanda **git merge**. În acest exemplu am facut merge la develop branch în master



```
MINGW64:/c:/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git checkout develop
Switched to branch 'develop'
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ git merge master
Merge made by the 'recursive' strategy.
 .gitignore | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ 
$
```

Figure 3.8– Branch merge

- i) În caz că o linie dintr-un fișier a fost redactată pe ambele branch-uri, atunci posibil să conflicteze în procesul de merge a branch-urilor.

```
MINGW64; c:/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ echo 'New content' > file.txt

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ git add .

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ git commit -m 'Changed file on develop branch'
[develop 60ac1c0] Changed file on develop branch
1 file changed, 1 insertion(+), 1 deletion(-)

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (develop)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ echo 'New content2' > file.txt

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git add .

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git commit -m 'Changed file on master branch'
[master 178a188] Changed file on master branch
1 file changed, 1 insertion(+), 1 deletion(-)

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git merge develop
Auto-merging file.txt
CONFLICT (content): Merge conflict in file.txt
Automatic merge failed; fix conflicts and then commit the result.

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master|MERGING)
$
```

Figure 3.9– Simularea unui conflict

- j) Soluționarea unui conflict poate fi efectuată prin mai multe metode: redactarea manuală, utilizarea unui instrument de comparare sau folosirea uneia din versiuni. În acest caz am forțat folosirea versiunii curente.

```
MINGW64:/c/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master|MERGING)
$ cat file.txt
<<<<<<< HEAD
New content2
=====
New content
>>>>>>> develop

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master|MERGING)
$ git checkout --ours -- file.txt

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master|MERGING)
$ cat file.txt
New content2

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master|MERGING)
$ git add .

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master|MERGING)
$ git commit
[master 78d5739] Merge branch 'develop'

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$
```

Figure 3.10 – Soluționarea unui conflict

- k) Crearea unui tag este efectuată utilizând comanda **git tag**. Aceasta atașează tag-ul nou la commit-ul current.

```
MINGW64:/c/Users/Eugen/Documents/Java/midps1
Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git tag -a 'R1.0' -m 'First release'

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$ git tag
R1.0

Eugen@EDESKTOP MINGW64 ~/Documents/Java/midps1 (master)
$
```

Figure 3.11 – Crearea unui tag

Concluzie

Elaborând această lucrare am utilizat sistemul de control al versiunilor GIT. Acest instrument lucrează în mod implicit cu modulul de interacțiune CLI. Astfel toate acțiunile au fost efectuate folosind comenzile prezente în git. Unele din aceste comenzi sunt:

git init Inițializarea unui repozitoriu

git config Configurarea unui repozitoriu

git checkout -b Crearea unui branch

git brach Afișarea listei de branch-uri

git add Înregistrează schimbarile

git commit Crează o versiune nouă cu schimbarile înregistrate

git remote add Adaugă remote

git push Salvează schimbarile local pe repozitoriu extern

git merge Execută merge între 2 branch-uri

git tag Modificarea tag-urilor din acest repozitoriu

Un element la fel de important în crearea și întreținerea repozitoriului GIT este fișierul *.gitignore*. Acesta permite excluderea altor fișiere care nu ar trebui să fie împărtășite cu alți dezvoltatori care lucrează la același repozitoriu.

References

- 1 GIT, *offical documentation*, <https://git-scm.com/documentation>
- 2 GitHub, *offical page*, <https://github.com/>