

FACULTATEA CALCULATOARE, INFORMATICA SI MICROELECTRONICA

UNIVERSITATEA TEHNICA A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A PRODUSELOR SOFT

LUCRAREA DE LABORATOR#1

---

## Version Control Systems si modul de setare a unui server

---

*Autor:*

Octavian COROLETCHI

*lector asistent:*

Victor GOJIN

CHISINAU 2017

## Laboratory work #1

### 1 Scopul lucrarii de laborator

Initializarea in Version Control Systems.

### 2 Obiective

Version Control Systems (git)

### 3 Laboratory work implementation

#### 3.1 Tasks and Points

- a) Initializarea unui nou repository
- b) Configurarea VCS
- c) Crearea branch-urilor (cel putin 2)
- d) Commit pe ambele branch-uri( cel putin 1 commit per branch)
- e) Setarea unui branch to track a remote origin pe care sa se faca push(GitHub)
- f) Resetarea unui branch la commit-ul anterior
- g) Salvarea temporara a schimbarilor care nu se vor face commit imediat
- h) Folosirea fisierului .gitignore
- i) Merge 2 branches
- j) Rezolvarea conflictelor a 2 branches
- k) Folosirea tag-urilor pentru marcarile semnificative

#### 3.2 Analiza lucrarii de laborator

Click on "Link" or copy <https://github.com/OctavianCoroletchiTI154/MIDPS.git> pentru repositoryul meu.

Task a:

Pentru inceput, mi-am facut cont pe github.com. Mi-am creat repositoryul meu(vezi imaginea a) public. Apoi am facut download la clientul Git.

Task b:

Mi-am configurat repositoryul meu, generindu-mi o cheie publica cu ajutorul comenzii `ssh-keygen -t rsa -b4096 -C"youremail@example.com"`. Apoi mi-am adaugat cheia mea publica la contul meu. Dupa mi-am clonat repositoryul meu pe masina mea locala cu ajutorul comenzii `git clone 'repositoryul meu'`

Task c:

Am creat branch-uri cu ajutorul comenzii "*gitbranch < noulmeubbranch >*". (vezi imaginea c). Apoi i-am dat push pe github, cu comanda *git push origin lab1*. (vezi imaginea c-1).

Task d:

Am facut commit pe branchurile create cu ajutorul comenzii *gitcommit -m "name"*. (vezi imaginea d).

Task e:

Am setat branch-ul *lab1*, pentru a urmări remote origin pe care să facem push. Am folosit comanda *gitbranch -u origin/lab1* (vezi imaginea e)

Task f:

Am eliminat branch-ul *lab1* din origin cu comanda *gitpushorigin : lab1*. Apoi am readus *lab1* la comitul care avem nevoie. *gitreset --hard 'hashid'*. Apoi am facut push din nou. *gitpushoriginlab1*. (pentru toate vezi imaginea f)

Task g:

Am salvat schimbarile temporare cu ajutorul Stash-ului. Aceasta permite salvarea temporara a unor modificari pentru care nu avem nevoie să facem commit. Se folosește comanda *gitstash*. După ce se salvează comenzile modificate, ne putem reîntoarce în orice moment spre salvări cu comanda *gitstashapplystash@0*, sau în loc de *stash@0* cu id-ul salvat. Pentru a vedea întreaga listă de stash-uri, folosim *gitstashlist*. (vezi imaginea g).

Task h:

.gitignore, este un fișier, în care specificăm ce tipuri de fișiere nu dorim să fie prezente în repoziitoriul nostru Git. În imaginea răspunzătoare de task-ul h, se va observa fișierul nostru .gitignore și tipurile de fișiere ce vor fi ignorate. (vezi imaginea h).

Task i:

Ajunând la un moment, când modificările pentru un anumit branch ajung la un statut final, atunci putem să îmbinăm branch-ul adițional cu cel principal. Pentru început, ne schimbăm pe branch-ul principal cu comanda *gitcheckoutmaster*, apoi le îmbinăm cu comanda *gitmergelab1* unde *master* și *lab1*, sunt branch-uri. Apoi putem face delete pentru branch-ul adițional *lab1*, pentru că nu mai avem nevoie de el cu comanda *gitbranch -d lab1*. (vezi imaginea i pentru implementari).

Task j:

Conflictele apar atunci când încerci să faci merge la 2 branch-uri, în care în ambele au loc modificări în același fișier. Cum ar fi situația întâlnită de mine, în care 2 branch-uri, aveau modificări diferite în README.md. Conflictul întâlnit (vezi imaginea j). Apoi trebuia să decid ce schimbări doresc să păstrez pentru a continua peste conflict. (vezi j-1). După aia, totul a continuat cu succes.

Task k:

Tagurile sunt folosite pentru a marca anumite evenimente importante din modificări. Ele se implementează cu ajutorul comenzii *gittag -a 'name' -m 'description'*. După se da push pentru a vedea și alții tagurile noastre. (vezi imaginea k, pentru implementare).

### 3.3 Imagini

#### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: OctavianCoroletchiTI154 / Repository name:

Great repository names are short and memorable. Need inspiration? How about [animated-lamp](#).

Description (optional):

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None ⓘ

[Create repository](#)

Imaginea - "a"

```
Octavian@Oca MINGW64 ~/Desktop (master)
$ ssh-keygen -t rsa -b 4096 -C "octavian.c96@gmail.com"
Generating public/private rsa key pair.
```

Imaginea - "b"

```
Octavian@Oca MINGW64 /d/MIDPS (master)
$ git branch lab1

Octavian@Oca MINGW64 /d/MIDPS (master)
$ git branch
 2stbranch
 lab1
* master
 newmodel
 trackingremote
```

Imaginea - "c"

```
Octavian@Oca MINGW64 /d/MIDPS (master)
$ git push origin lab1
Total 0 (delta 0), reused 0 (delta 0)
To github.com:OctavianCoroletchiTI154/MIDPS.git
 * [new branch]      lab1 -> lab1
```

Imaginea - "c-1"

```

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git push --set-upstream origin lab1
Counting objects: 10, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 82.30 KiB | 0 bytes/s, done.
Total 10 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
Branch lab1 set up to track remote branch lab1 from origin.
To github.com:OctavianCoroletchiTI154/MIDPS.git
  2b4c970..e095f8c  lab1 -> lab1

```

Imaginea - "d"

```

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git branch -u origin/lab1
Branch lab1 set up to track remote branch lab1 from origin.

```

Imaginea - "e"

```

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git push origin :lab1
To github.com:OctavianCoroletchiTI154/MIDPS.git
- [deleted]          lab1

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git reset --hard dba4b64f39504959b90f9e3dd54a5a9899738e26
HEAD is now at dba4b64 Task f

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git push origin lab1
Counting objects: 17, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (17/17), done.
Writing objects: 100% (17/17), 123.13 KiB | 0 bytes/s, done.
Total 17 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local objects.
To github.com:OctavianCoroletchiTI154/MIDPS.git
 * [new branch]      lab1 -> lab1

```

Imaginea - "f"

```

MINGW64:/d/MIDPS

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git stash

Saved working directory and index state WIP on lab1: 2b6b5d8 new mod
HEAD is now at 2b6b5d8 new mod

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git stash list
stash@{0}: WIP on lab1: 2b6b5d8 new mod
stash@{1}: WIP on lab1: dba4b64 Task f
stash@{2}: WIP on lab1: dba4b64 Task f

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git stash apply stash@{0}
On branch lab1
Your branch is up-to-date with 'origin/lab1'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

```

Imaginea - "g"

---

```

## Ignore Visual Studio temporary files, build results, and
## files generated by popular Visual Studio add-ons.

# User-specific files
*.suo
*.user
*.userosscache
*.sln.docstates

# User-specific files (MonoDevelop/Xamarin Studio)
*.userprefs

# Build results
[Dd]ebug/
[Dd]ebugPublic/
[Rr]elease/
[Rr]eleases/
x64/
x86/
bld/
[Bb]in/
[Oo]bj/
[Ll]og/

# Visual Studio 2015 cache/options directory
.vs/
# Uncomment if you have tasks that create the project's static files in wwwroot
#wwwroot/

# MSTest test Results
[Tt]est[Rr]esult*/
[Bb]uild[Ll]og.*

# NUNIT
*.VisualState.xml
TestResult.xml

```

Imaginea - "h"



```
MINGW64:/d/MIDPS
Your branch is up-to-date with 'origin/lab1'.

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git checkout master
Your branch is up-to-date with 'origin/master'.
Switched to branch 'master'

Octavian@Oca MINGW64 /d/MIDPS (master)
$ git branch
  2stbranch
  lab1
* master
  newmodel
  trackingremote

Octavian@Oca MINGW64 /d/MIDPS (master)
$ git merge lab1
Updating 2b4c970..9f871c5
Fast-forward
 Lab 1/Raport/Lab1introduction.tex | 8 ++++++
 Lab 1/Raport/Screenshot_1.jpg     | Bin 0 -> 59863 bytes
 Lab 1/Raport/b.jpg                | Bin 0 -> 11008 bytes
 Lab 1/Raport/c-1.jpg              | Bin 0 -> 12976 bytes
 Lab 1/Raport/c.jpg                | Bin 0 -> 11392 bytes
 Lab 1/Raport/d.jpg                | Bin 0 -> 33035 bytes
 Lab 1/Raport/e.jpg                | Bin 0 -> 9095 bytes
 Lab 1/Raport/f.jpg                | Bin 0 -> 43762 bytes
 Lab 1/Raport/g.jpg                | Bin 0 -> 56856 bytes
 README.md                        | 4 +++-
10 files changed, 11 insertions(+), 1 deletion(-)
create mode 100644 Lab 1/Raport/Lab1introduction.tex
create mode 100644 Lab 1/Raport/Screenshot_1.jpg
create mode 100644 Lab 1/Raport/b.jpg
create mode 100644 Lab 1/Raport/c-1.jpg
create mode 100644 Lab 1/Raport/c.jpg
create mode 100644 Lab 1/Raport/d.jpg
create mode 100644 Lab 1/Raport/e.jpg
create mode 100644 Lab 1/Raport/f.jpg
create mode 100644 Lab 1/Raport/g.jpg

Octavian@Oca MINGW64 /d/MIDPS (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 5 commits.
(use "git push" to publish your local commits)
nothing to commit, working tree clean

Octavian@Oca MINGW64 /d/MIDPS (master)
$ git push
Total 0 (delta 0), reused 0 (delta 0)
To github.com:OctavianCoroletchiTI154/MIDPS.git
2b4c970..9f871c5 master -> master
```

Imaginea - "i"

```
Octavian@Oca MINGW64 /d/MIDPS (master)
$ git merge test1 test2
Fast-forwarding to: test1
Trying simple merge with test2
Simple merge did not work, trying automatic merge.
Auto-merging README.md
ERROR: content conflict in README.md
fatal: merge program failed
Automatic merge failed; fix conflicts and then commit the result.

Octavian@Oca MINGW64 /d/MIDPS (master|MERGING)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Imaginea - "j"

```

1  # MIDPS
2  # Hello
3
4  ##2nd Branch Modify
5
6  ##New Commit v0.1
7  ##Added new branch
8  Adaugarea comiturilor pentru noul branch.
9  Trying to return a commit;
10 Before stash 1
11
12 <<<<<< .merge_file_a09932
13 Additional modification for conflict test. 1
14 1
15 =====
16 Additional modification for conflict test.2
17 2
18 2
19 >>>>>> .merge_file_a08484
20

```

Imaginea - "j-1"

```

Octavian@Oca MINGW64 /d/MIDPS (master)
$ git tag -a v1.0 -m "Final Version of Lab1"

Octavian@Oca MINGW64 /d/MIDPS (master)
$ git push origin v1.0
Counting objects: 1, done.
Writing objects: 100% (1/1), 171 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To github.com:OctavianCoroletchiTI154/MIDPS.git
 * [new tag]          v1.0 -> v1.0

Octavian@Oca MINGW64 /d/MIDPS (master)

```

Imaginea - "k"

## Concluzie

In concluzie, pot spune ca version control system este o utilitate foarte importanta in urmarirea dezvoltarii aplicatiilor. Aceasta permite dezvoltarea unei aplicatii prin diverse cai, de diferiti developeri in acelasi timp. Version control protejeaza codul sursa de o eroare umana care poate duce la consecinte foarte mari. Dezvoltatorii de aplicatii care lucreaza in echipe, mereu schimba codul sursa, si aceasta necesita intr-o organizare bine pusa la punct. Principala problema este cea a conflictelor, ce permite o rezolvare adecvata a acestei probleme.



## References

- 1 [http://www.manniwood.com/starting\\_a\\_project\\_with\\_git.html](http://www.manniwood.com/starting_a_project_with_git.html)
- 2 <http://www.vogella.com/tutorials/Git/article.html>
- 3 <https://git-scm.com/>