

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:
Alexandru STAMATIN

lector asistent:
Irina COJANU
lector superior:
Radu MELNIC

Lucrarea de laborator #1

1 Scopul lucrării de laborator

Obținerea deprinderilor de utilizare a Version Control Systems și studierea modului de setare a unui server)

2 Obiective

Studierea Version Control Systems (git - bitbucket - mercurial - svn)

3 Efectuarea lucrarii de laborator

3.1 Sarcinile propuse

- Initializarea unui nou repozitoriu
- Generarea si adaugarea cheilor SSH
- Configurarea VCS
- Crearea branch-urilor (cel putin 2)
- Cel putin 1 commit pe fiecare branch
- Setarea unui branch to track a remote origin
- Resetarea unui branch la commit-ul anterior
- Salvarea temporara a schimbarilor fara commit imediat
- Utilizarea fisierului .gitignore
- Merge la 2 branch-uri
- Rezolvarea conflictelor
- Utilizarea tag-urilor

3.2 Analiza lucrarii de laborator

Link la repozitoriu: <https://github.com/AlexStamatin/MIDPS>

- Initializarea unui nou repozitoriu

Aceasta sarcina poate fi indeplinita atat online utilizand site-ul github cat si utilizand comanda *git init* In cazul dat repozitoriul nou a fost creat accesand pagina personala de pe github cu ajutorul optiunii "New Repository" din compartimentul "Your repositories".

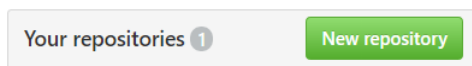
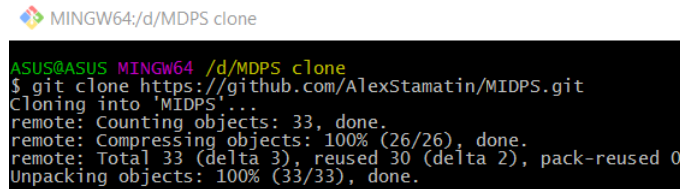


Figure 1: Crearea repozitoriului

Repozitoriul a fost clonat pe masina locala utilizand comanda *git clone*.

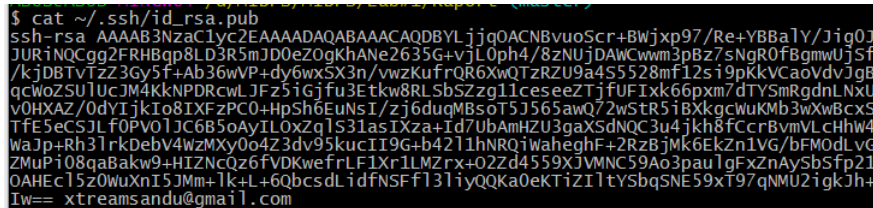


```
MINGW64:/d/MDPS clone
ASUS@ASUS MINGW64 /d/MDPS c\lone
$ git clone https://github.com/AlexStamatin/MIDPS.git
Cloning into 'MIDPS'...
remote: Counting objects: 33, done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 33 (delta 3), reused 30 (delta 2), pack-reused 0
Unpacking objects: 100% (33/33), done.
```

Figure 2: Clonarea pe masina locala

- Generarea si adaugarea cheilor SSH

Utilizand protocolul SSH este posibila autentificarea la servere si servicii aflate la distanta. Cu ajutorul cheilor SSH este posibila conectarea la github fara a fi necesara introducerea username-ului si parolei la fiecare vizita. Cheile SSH pot fi generate cu ajutorul comenzii *ssh-keygen*. Cheile publice existente pot fi afisate cu ajutorul comenzii *cat ~/.ssh/id_rsa.pub*

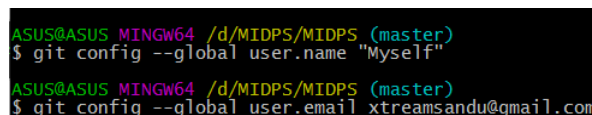


```
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAQDBYLjjgOACNBvuoScr+BWjxp97/Re+YBBaLY/Jig0J
JURiNQCgg2FRHBqp8LD3R5mJD0eZ0gKhAne2635G+vJL0ph4/8zNUjDAwCwmm3pBz7sNgR0fBgmmUjSf
/kjDBTVtZ23Gy5f+Ab36wVP+dy6wx5X3n/vwzKufRQR6XwQTZRZU9a4S5528mf12s19pKkVCaoVdvJgB
qcwoZSUlUcJM4KkNPDRcwlJFz5iGjfu3Etkw8RLSbSZzg11ceseeZTjFUFIXk66pxm7dTYSmRgdnLNxU
v0HXAZ/0dyIjkIo8IXFzPC0+HpSh6EuNsI/zj6duqMBsoT5J565awQ72wStR5iBXkgcwuKMb3wXwBcxS
TfE5eCSJLf0PV01JC6B5oAyILOxZq1S31asIXza+Id7UbAmHZU3gaXSdNQC3u4jkh8fCcrBvmVLcHhw4
waJp+Rh3lrkDebV4WzMXy0o4Z3dv95kucII9G+b4211hNRQiwaheghF+2RzBjmk6EkZn1VG/bFModLvG
ZMuPi08qaBakw9+HIZNcQz6fVDKwefrLF1Xr1LMZrx+02Zd4559XJVMNC59Ao3paulgFxFzNaySbSfp21
OAHEc15z0WuXnI5JMm+1k+L+6QbcsdLidfNSFF13liyQQKa0eKtiZi1tYSbqSNE59xT97qNMU2igkJh+
Iw== xtreamsandu@gmail.com
```

Figure 3: Cheia publica SSH

- Configurarea VCS

Configurarile git de baza pot fi modificate cu ajutorul comenzii *git config --global user.name "name"* si *git config --global user.email "email"*



```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git config --global user.name "Myself"
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git config --global user.email xtreamsandu@gmail.com
```

Figure 4: Configurarea unor parametri

Fisiere noi spre indexare pot fi adaugate cu ajutorul comenzii *git add*. Pentru a inregistra toate schimbarile in comparatie cu fisierele de pe git se utilizeaza comanda *git commit -m "Commit comment"*. Pentru a incarca fisierele indexate pe git utilizam comanda *git push*

```

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git add Hey.txt

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git commit -m "First few chapters"
[master c93018a] First few chapters
 2 files changed, 1 insertion(+)
 create mode 100644 Hey.txt
 create mode 100644 Lab#1/Raport/Lab_template.pdf

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git push
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 63.47 KiB | 0 bytes/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/AlexStamatin/MIDPS.git
 e5e4e6b..c93018a master -> master

```

Figure 5: Adaugarea fisierelor la repozitoriu

- Crearea branch-urilor

Pentru a crea un branch este necesara utilizarea comenzii *git branch "name"* Comanda *git branch* ne afiseaza branch-urile. Pentru a crea un branch nou si a face switch la el se utilizeaza comanda *git branch -b "name"*

```

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git branch NBranch1

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git branch NBranch2

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git branch
  NBranch1
  NBranch2
* master

```

Figure 6: Crearea branch-urilor

- Crearea commit-urilor de pe fiecare branch

Initial facem switch la branch-ul de pe care dorim sa facem commit cu ajutorul comenzii *git checkout "name"*. Apoi cu *git add* adaugam fisierele spre indexare.

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git checkout NBranch1
Switched to branch 'NBranch1'

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch1)
$ git add random.txt

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch1)
$ git commit -m "Add to NBranch1"
[NBranch1 d08ebd5] Add to NBranch1
1 file changed, 1 insertion(+)
create mode 100644 random.txt

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch1)
$ git push origin NBranch1
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 299 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To https://github.com/AlexStamatin/MIDPS.git
* [new branch]      NBranch1 -> NBranch1
```

Figure 7: Commit in NBranch1

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git add random.txt

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git commit -m "Add to NBranch2"
[NBranch2 df50e1c] Add to NBranch2
1 file changed, 1 insertion(+)
create mode 100644 random.txt

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git push origin NBranch2
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 296 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To https://github.com/AlexStamatin/MIDPS.git
* [new branch]      NBranch2 -> NBranch2
```

Figure 8: Commit in NBranch2

- Setarea unui branch to track a remote origin

A fost creat un nou branch `trbranch` to track remote origin utilizand comanda `git branch --track "name" origin/master`. Ulterior a fost creat si adaugat un commit in `trbranch` cu comanda `git push origin "name"`

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git branch --track trbranch origin/master
Branch trbranch set up to track remote branch master from origin.

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git checkout trbranch
Switched to branch 'trbranch'
Your branch is up-to-date with 'origin/master'.
```

Figure 9: Track remote origin

```
$ git push origin trbranch
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 293 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To https://github.com/AlexStamatin/MIDPS.git
 9eacf5b..296825a trbranch -> trbranch
```

Figure 10: Push to trbranch

- Resetarea unui branch la commit-ul anterior

Cu ajutorul comenzii `git log` identificam commit-ul la care dorim sa resetam branch-ul

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git log
commit 9eacf5b7837959bb587d07c881fa535d42ab4402
Author: Myself <xtreamsandu@gmail.com>
Date: Mon Feb 27 00:05:19 2017 +0200

    Removed ignored files

commit 14fd1381311cab01c5a19224dff12558d8ec6098
Author: Myself <xtreamsandu@gmail.com>
Date: Sun Feb 26 23:48:56 2017 +0200

    Remove ignored files

commit 4b1316468afa260204243cfbd671eeecafa594bc
Author: Myself <xtreamsandu@gmail.com>
Date: Sun Feb 26 23:18:56 2017 +0200

    Half Report
```

Figure 11: Identificarea commit-ului

Apoi cu ajutorul `git reset "mode" "commit"` se reseteaza head-ul branch-ului curent la "commit". Poate fi utilizat unul dintre modurile `soft`, `mixed`(optiunea default), `hard`, `merge` sau `keep`.

- Salvarea temporara a schimbarilor fara commit imediat

Uneori este necesar ca in timp ce se lucreaza asupra unei parti din proiect sa se faca switch la un alt branch dar fara a face un commit a modificarilor curente. Acest lucru poate fi realizat cu ajutorul comenzii *git stash*

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git checkout master
error: Your local changes to the following files would be overwritten by checkout:
    random.txt
Please commit your changes or stash them before you switch branches.
Aborting
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git stash
Saved working directory and index state WIP on NBranch2: df50e1c Add to NBranch2
HEAD is now at df50e1c Add to NBranch2
```

Figure 12: Utilizarea git stash

Lista stash-urilor pastrate poate fi vizualizata cu comanda *git stash list*

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git stash list
stash@{0}: WIP on NBranch2: df50e1c Add to NBranch2
stash@{1}: WIP on NBranch1: d08ebd5 Add to NBranch1
stash@{2}: WIP on master: 9eacf5b Removed ignored files
```

Figure 13: Vizualizarea stash-urilor curente

Stash-urile pastrate pot fi aplicate cu ajutorul *git stash apply*

- Utilizarea fisierului .gitignore

Fisierul .gitignore este utilizat pentru a ne asigura ca anumite fisiere nu vor fi indexate si vor fi ignorate de catre git. Acesta nu afecteaza fisierele care deja sunt indexate.

Printre fisierele ce se recomanda a fi ignorate se numara fisiere auxiliare LaTeX, imagini e.t.c.



```
.gitignore - Notepad
File Edit Format View Help
#Fisiere auxiliare LaTeX

*.aux
*.bak
*.bbl
*.blg
*.bib
*.bcf
*.dvi
*.fdb_latexmk
*.glg
*.glo
*.gls
*.idx
*.ilg
*.ind
*.ist
```

Figure 14: Fisierul .gitignore

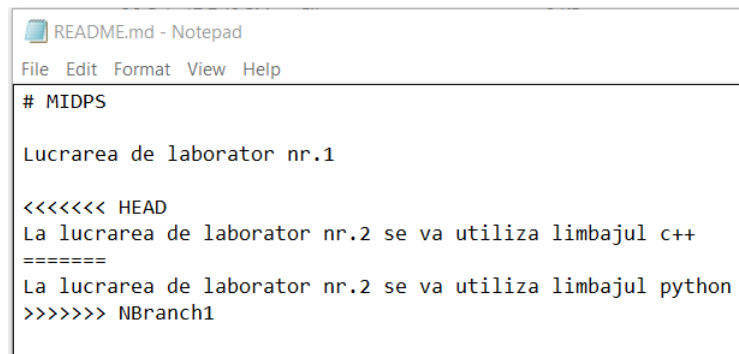
- Merge la 2 branch-uri si rezolvarea conflictelor

Mai multe branch-uri pot fi adaugate la un singur branch cu ajutorul comenzii *git merge*. Dar pot aparea conflicte din cauza diferentelor dintre continuturile fisierelor indexate in aceste branch-uri.

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2)
$ git merge NBranch1 NBranch2
Auto-merging Lab#1/README.md
CONFLICT (content): Merge conflict in Lab#1/README.md
Automatic merge failed; fix conflicts and then commit the result.
```

Figure 15: Conflictul aparut in urma comenzii git merge

Conflictul poate fi rezolvat manual editand fisierele ce il cauzeaza.



```
README.md - Notepad
File Edit Format View Help
# MIDPS

Lucrarea de laborator nr.1

<<<<<< HEAD
La lucrarea de laborator nr.2 se va utiliza limbajul c++
=====
La lucrarea de laborator nr.2 se va utiliza limbajul python
>>>>>> NBranch1
```

Figure 16: Rezolvarea conflictului

Dupa rezolvarea conflictului se poate efectua merge-ul

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2|MERGING)
$ git add Lab#1/README.md
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (NBranch2|MERGING)
$ git commit -m "Merge 2 branches"
[NBranch2 475c54e] Merge 2 branches
```

Figure 17: Commit dupa rezolvarea conflictului

- Utilizarea tag-urilor

Tag-urile sunt utilizate pentru a marca commit-uri importante. Exista doua tipuri de tag-uri: lightweight si annotated. Un tag lightweight este ca un branch care nu se schimba, este un pointer la un commit specific. Pentru a utiliza acest tip de tag se apeleaza comanda *git tag "tagname"*. Tag-urile annotated sunt pastrate ca obiecte in baza de date Git. Pentru a crea asa un tag se utilizeaza comanda *git tag -am "mesaj" "tagname"*

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git tag v0.8lw

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git tag
v0.8lw

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git show v0.8lw
commit c5131891b5210789ccaf2a9aac8b6383201a5f3a
Author: Myself <xtreamsandu@gmail.com>
Date: Mon Feb 27 19:45:29 2017 +0200

Final tasks
```

Figure 18: Crearea unui tag lightweight

```
ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git tag v0.8lw

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git tag
v0.8lw

ASUS@ASUS MINGW64 /d/MIDPS/MIDPS (master)
$ git show v0.8lw
commit c5131891b5210789ccaf2a9aac8b6383201a5f3a
Author: Myself <xtreamsandu@gmail.com>
Date: Mon Feb 27 19:45:29 2017 +0200

Final tasks
```

Figure 19: Crearea unui tag annotated si indexarea tagurilor

Concluzie

În cadrul acestei lucrări de laborator am studiat sistemul de control al versiunilor Git. Am determinat că acesta permite urmărirea istoriei modificărilor suportate de un fișier.

Utilizarea Git facilitează munca inginerilor în domeniul software oferindu-le un sir de posibilități:

- Posibilitatea de a reveni la o versiune mai veche funcțională a codului în cazul comiterii unor greseli
- Compararea diferitelor versiuni a codului propriu
- Urmărirea dezvoltării unui program pe parcursul unei perioade de timp
- Experimentarea cu noi funcționalități fără a interfera cu versiunile de bază a codului

De asemenea utilizarea unui VCS este absolut indispensabilă în cazul lucrului în echipă asupra unui proiect, deoarece:

- Permite vizualizarea informației despre cine a lucrat asupra unui proiect și ce schimbări a introdus
- Permite mai multor persoane să lucreze în paralel asupra diferitelor părți a unui proiect având posibilitatea de a combina ulterior modificările făcute la program
- Oferă posibilitatea de a determina ce secvență de cod a dus la apariția bug-urilor în program

În urma efectuării lucrării ne-am familiarizat cu comenzile de bază ale sistemului Git și am obținut abilități de utilizare a diferitelor funcționalități ale acestuia. Am reușit crearea mai multor branch-uri, combinarea acestora și rezolvarea conflictelor ce au apărut pe parcurs. Am avut posibilitatea de a ne întoarce la versiuni mai vechi a fișierelor în cazul unor schimbări nedorite sau pierderii datelor. Am ajuns la concluzia că cunoașterea sistemului Git poate spori considerabil performanțele programatorilor.

References

- [1] <https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging>
- [2] <https://git-scm.com/book/en/v2/Git-Basics-Tagging>
- [3] <https://www.siteground.com/tutorials/git/commands.htm>