

FACULTATEA CALCULATOARE, INFORMATICA SI
MICROELECTRONICA

UNIVERSITATEA TEHNICA A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A
PRODUSELOR SOFT

LUCRARE DE LABURATOR #1

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

Autor:
Mihai LASCU

Lector asistent:
Irina COJANU
Lector asistent:
Radu MELNIC

Lucrarea de laborator #1

1. Scopul lucrării de laborator

De a se învăța utilizarea unui Version Control System și modul de setare a unui server.

2. Obiective

Studierea Version Control System și modul de setare a unui server.

3. Mersul lucrării de laborator

3.1 Cerințele

Initializarea unui nou repository.

Configurarea VCS.

Commit, push pe branch.

Folosirea fișierului .gitignore.

Revenirea la versiunile anterioare.

Crearea branch-urilor noi.

Commit pe ambele branch-uri.

Merge la 2 branch-uri.

Rezolvarea conflictelor.

3.2 Analiza Lucrarii de laborator

Linkul la repozitoriu <https://github.com/mihai1996/MIDPS.git>

Sunt mai multe modalitati de initializare a repozitoriului pe github. Putem crea o mapa goala in care vom plasa gitul nostru prin intermediul comenzii **git init**.

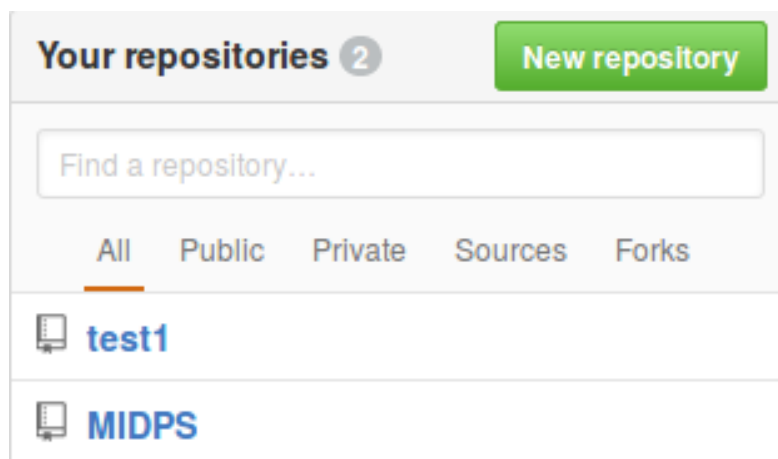
Urmatorul pas este crearea insusi a noului repozitoriu pe care il vom crea utilizand urmatoarea comanda **curl -u 'User' https://api.github.com/user/repos -d '{"name":"NUME"}'**. Cuvintele scrise cu CAPS se vor inlocui cu numele utilizatorului si numele repozitoriului.

In continuare trebuie sa unim gitul nostru gol cu repozitoriul creat.

Pentru aceasta vo folosi comanda **git remote add origin "Linkul la repozitoriu"**

```
mihai@mihai-HP-ProBook-450-G2: ~/test
mihai@mihai-HP-ProBook-450-G2:~$ cd test
mihai@mihai-HP-ProBook-450-G2:~/test$ git init
Initialized empty Git repository in /home/mihai/test/.git/
mihai@mihai-HP-ProBook-450-G2:~/test$ curl -u 'mihai1996' https://api.github.com/user/repos -d '{"name":"test1"}'
mihai@mihai-HP-ProBook-450-G2:~/test$ git remote add origin https://github.com/mihai1996/MIDPS.git
mihai@mihai-HP-ProBook-450-G2:~/test$
```

O alta metoda de a crea un repozitoriu este cea online. Pentru aceasta avem nevoie sa deschidem pagina noastra pe **github**, si sa apasam butonul **new repositories**.



Configurarea git-ului consta in mai multe etape. Pentru inceput vom configura *numele si email-ul*. Aplicam urmatoarele comenzi:

git config -global user.name "Name"

git config -global user.email Email

```
x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git config --list
user.name=miha1996
user.email=mihailascu13@gmail.com
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
remote.origin.url=git@github.com:miha1996/MIDPS.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$
```

Urmatorul pas consta in generarea la cheia **SSH**(Secure Shell). Scriem in CLI **ssh-keygen**, iar cheia obtinuta o copiem in setarile noastre de pe git.

Este de dorit sa initializam repozitoriul nostru cu fisierul **README.md** si un **.gitignore**. In fisierul **README.md** vom adauga niste informatie pentru cei care se vor folosi de repozitoriul creat.

In fisierul **.gitignore** vom adauga toate fisierele ce trebuiesc ignorate in sensul larg al cuvintului fisierele citite in **.gitignore** nu vor putea fi incarcate.

```
x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ vim README.md
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ vim .gitignore
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ cat README.md
# Student
Lascu Mihai gr.TI-153
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ cat .gitignore
# lista de fisiere ce vor fi ignorate:
ignore.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$
```

Vom adauga fisierele noi create pe repozitoriul nostru. La moment vom avea nevoie de urmatoarele comenzi:

git add * - comanda care indexeaza toate fisierele(*git add .*);

git commit -m – comanda face un snapshot la toate schimbarile noastre;

git push origin master – comanda incarca toate fisierele index pe git.

```
x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git add *
The following paths are ignored by one of your .gitignore files:
ignore.txt
Use -f if you really want to add them.
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git commit -m "tt"
[master 51c4029] tt
1 file changed, 1 insertion(+), 1 deletion(-)
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 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 git@github.com:mihai1996/MIDPS.git
b88bad0..51c4029 master -> master
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$
```

Pentru a ne asigura ca am aplicat pasii corect si fara probleme vom utiliza: **git status** si **git show**.

```
x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git status
On branch master
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:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git show
commit 51c402986df8f5e13f7e8aaf739bd42493edea93
Author: miha1996 <mihailascu13@gmail.com>
Date: Thu Feb 16 20:02:46 2017 +0200

    tt

diff --git a/README.md b/README.md
index a0a73de..e4bfb4e 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,2 @@
-# Student
+# Student FCIM
Lascu Mihai gr.TI-153
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$
```

Una dintre caracteristicile principale a unui VCS este faptul ca ne permite sa revenim la o versiune mai veche. Aceasta poate fi efectuata cu ajutorul comenzii `git reset -TYPE "codul comitului"`.

```
x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ vim temp.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md temp.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git add *
The following paths are ignored by one of your .gitignore files:
ignore
ignore.txt
Use -f if you really want to add them.
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git commit -m "type 1"
[master e895f32] type 1
1 file changed, 1 insertion(+)
create mode 100644 temp.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 270 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 git@github.com:mihai1996/MIDPS.git
51c4029..e895f32 master -> master
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md temp.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git rm temp.txt
rm 'temp.txt'
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git add *
The following paths are ignored by one of your .gitignore files:
ignore
ignore.txt
Use -f if you really want to add them.
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git commit -m "type 2"
[master ead79da] type 2
1 file changed, 1 deletion(-)
delete mode 100644 temp.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git push origin master
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 214 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
```

Dupa datele executate observam ca, cind am facut **commit**, s-au ignorat fisierele incluse in `.gitignore`.

Am creat un fisier nou *temp.txt* in *type 1*. Apoi l-am sters si am facut **commit** la *type 2*, in care am sters fisierul *temp.txt* dorim sa revenim la *type 1*. Pentru inceput vom lansa comanda **git -log** care ne arata *log-ul* de *commit-ri* si codul pentru fiecare **commit**. Vom avea nevoie de primele 7 cifre de la *commit-ul* anterior.

```
x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git log
commit ead79da2f0c82c1614b1cdeef84a9c0076c2267f
Author: miha1996 <mihailascu13@gmail.com>
Date: Sat Feb 18 13:31:48 2017 +0200

    type 2

commit e895f32ce3c4d2a3cd4a4d8f412a2d79c8f01612
Author: miha1996 <mihailascu13@gmail.com>
Date: Sat Feb 18 13:28:02 2017 +0200

    type 1
```

Acum vom folosi comenzile descrise mai sus.

```
x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git reset --hard e895f32
HEAD is now at e895f32 type 1
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md temp.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git reset --soft e895f32
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md temp.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$
```

Daca am facut niste schimbari in repository si nu ne satisfac, putem usor reveni la ultima versiune care era pe git utilizind comanda:

git pull origin "branch";

Comanda **git pull** v-a face update la repositoryul nostru pina la ultima versiune.

```
x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git pull origin master
From github.com:miha1996/MIDPS
* branch      master      -> FETCH_HEAD
Updating e895f32..ead79da
Fast-forward
 temp.txt | 1 -
 1 file changed, 1 deletion(-)
 delete mode 100644 temp.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
ignore ignore.txt Lab1 Lab2 Lab3 Lab4 Lab5 README.md
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$
```

Observam ca dupa ce am executat comanda **git pull** repositoriul nostru a revenit la ultima versiune, eliminind fisierul *temp.txt*.

VCS ne permite sa avem mai multe **branch-uri**. Din traducere *branch* semnifica "creanga". *Branch-urile* sunt foarte comod de folosit cind dorim sa lucram paralel la un proiect si apoi dorim sa unim toate modificarile.

git branch "name" - creeaza un *branch* noucu numele "name";
git branch – vizualizarea *branch-urilor*(* indica *branch-ul* curent);
git branch -d "nume" - sterge *branch-ul* nume;
git branch -v – vizualizam ultimele *commit-uri* din lista noastra de *branch-uri*;
git checkout -b "name" - creeaza un *branch* nou cu numele "name" si face switch la el.

```
mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch creanga1
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch
  creanga1
* master
  pas2
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch -d pas2 creanga1
Deleted branch pas2 (was ff3471f).
Deleted branch creanga1 (was ead79da).
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git checkout -b new_b
Switched to a new branch 'new_b'
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ vim exemple.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch
  master
* new_b
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ ls
exemple.txt  ignore  ignore.txt  Lab1  Lab2  Lab3  Lab4  Lab5  README.md
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git add *
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git status
On branch new_b
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   exemple.txt
        new file:   ignore.txt

mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git commit -m "new branch"
[new_b b0181de] new branch
2 files changed, 2 insertions(+)
create mode 100644 exemple.txt
create mode 100644 ignore.txt
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git push origin new_b
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 320 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To git@github.com:mihai1996/MIDPS.git
 * [new branch]      new_b -> new_b
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch -v
  master ead79da type 2
* new_b  b0181de new branch
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$
```


git checkout "nume" - face switch la *branch-ul* "nume";
git branch -u upstream/name – face track la *branch-ul* indicat din *branch-ul* curent;
git branch -u upstream/name – face track din *branch-ul* "nume" la *branch-ul* indicat;
git branch --track "name" upstream/name – creeaza *branch-ul* "nume" si ii face track la *branch-ul* indicat;
git branch -unset-upstream – scoate tracking-ul la *branch-ul* in care ne aflam.

```

x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch
  master
* new_b
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git checkout master
Switched to branch 'master'
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git checkout new_b
Switched to branch 'new_b'
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch -u origin/master
Branch new_b set up to track remote branch master from origin.
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch -u origin/master new_b
Branch new_b set up to track remote branch master from origin.
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch --track "new_b2" origin/master
Branch new_b2 set up to track remote branch master from origin.
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch
  master
* new_b
  new_b2
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git checkout master
Switched to branch 'master'
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git checkout new_b
Switched to branch 'new_b'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git checkout new_b2
Switched to branch 'new_b2'
Your branch is up-to-date with 'origin/master'.
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$
  
```

Putem avea *conflicte* in cazul cind dorim sa facem *merge* la 2 *branch-uri* si unele rinduri sunt diferite. In asemenea caz ne vine in ajutor un *mergetool*. Drept *mergetool* am ales **kdiff3**. Pentru a seta *kdiff3* ca *mergetool* default folosim comanda: **git config -global merge.tool kdiff3**.

In continuare vom lucra cu 2 *branch-uri* - "master" si "new_b". Vom crea in fiecare *branch* un fisier "ft_merge" continutul caruia va fi diferit.

```

x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch
* master
  new_b
  new_b2
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ vim ft_merge
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ cat ft_merge
tuctuc
pokemongo
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git checkout new_b
Switched to branch 'new_b'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ vim ft_merge
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ cat ft_merge
tuctuc
Scrie ceva alt ceva
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ █

```

In continuare vom incerca sa facem merge si sa rezolvam acest conflict.

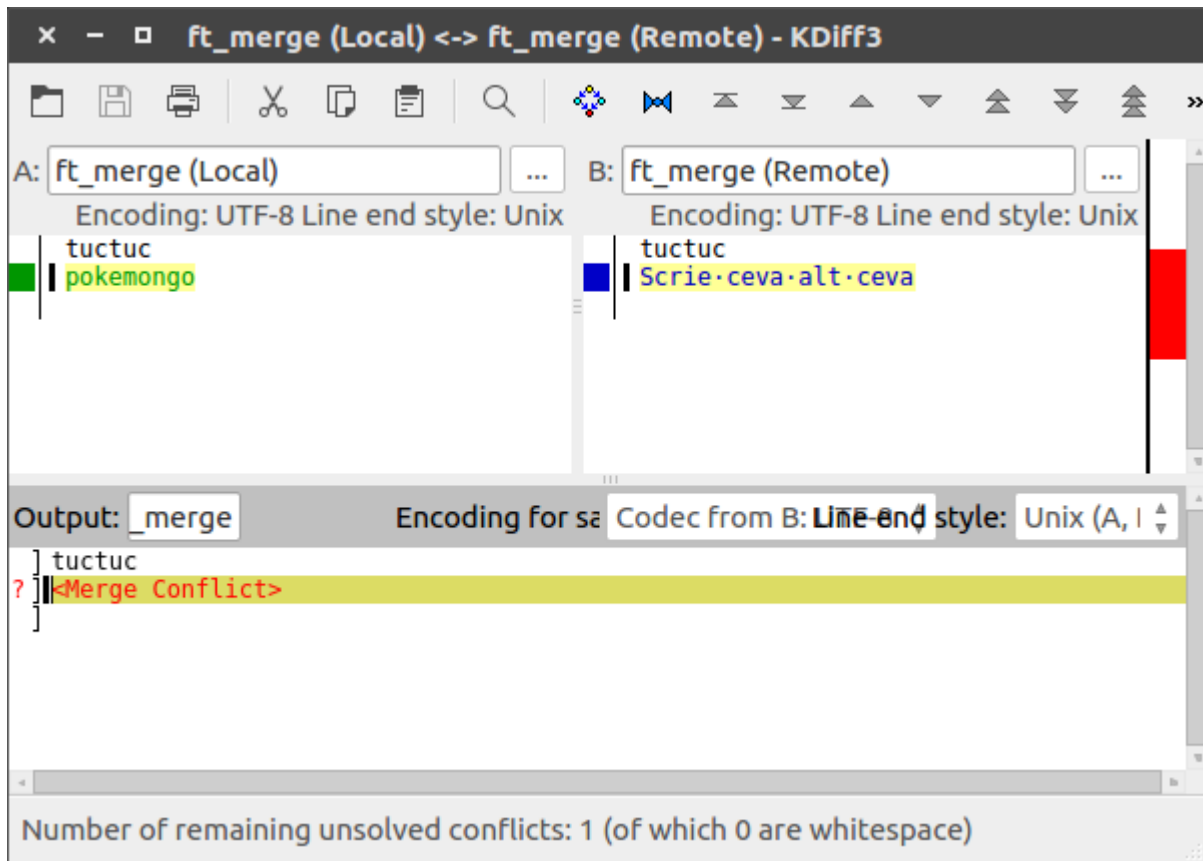
```

x - □ mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git branch
* master
  new_b
  new_b2
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git merge new_b
Auto-merging ft_merge
CONFLICT (add/add): Merge conflict in ft_merge
Automatic merge failed; fix conflicts and then commit the result.
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git mergetool
Merging:
ft_merge

Normal merge conflict for 'ft_merge':
{local}: created file
{remote}: created file
█

```

Dupa acest pas rezolvam conflictul cu ajutorul **kdiff3**. Acesta ne afiseaza urmatoarea fereasta:



Tagurile sunt foarte comod de folosit si cu ajutorul lor putem face un track eficient al versiunelor. Exemplu de tag.

```
mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git tag -a v1.0 -m "This is final git!"
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git tag
v1.0
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git push --tags
```

Pentru a putea vedea versiunea taguita folosim comanda:
git show "tag"

```
mihai@mihai-HP-ProBook-450-G2: ~/MIDPS
mihai@mihai-HP-ProBook-450-G2:~/MIDPS$ git show v1.0
tag v1.0
Tagger: miha1996 <mihailascu13@gmail.com>
Date: Sat Feb 18 23:48:27 2017 +0200

This is final git!
```

3.3. Concluzie

In lucrarea data am studiat lucrul cu **VCS**. Am facut cunostinta cu platforma **github**. Toate lucrarile si comenzile le-am efectuat in terminal pe sistema de operare **Linux**. Folosire **VCS** contine un sir de avantaje pentru lucrarile cu produsele soft ce ne permite sa efectuam lucrul mai rapid,eficient si fara probleme.El ne permite lucrul in paralel sau echipa si revenirea la versiunile anterioare.In laburator am inclus majoritatea lucrarilor esentiale ce stau la baza platformei **git**. Nu este prima mea experinta cu git-ul, am facut cunostinta cu el doar la nivel de baza(`git clone`, `git add *`, `git status`, `git commit -m " "`, `git push`, `git show`) ce imi dadeau posibilitatea sami incarc programele pe platforma.

Insa in aceasta lucrare am facut cunostinta cu lucruri noi si foarte utile precum ar fi *branchurile*, *merge la branchuri*, si rezolvarea *conflicteleor* utilizind **kdiff3**). In opinia mea **VCS** este bine de stiut pentru orice programator, deoarece el este foarte practic si eficient atat in lucru cit si in dezvoltarea softului.