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 laburator #1

1. Scopul lucrarii de laburator

De a se invata utilizarea unui Version Control System si modul de setare a unui server.

2. Obiective

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

3. Mersul lucrarii de laburator

3.1 Cerintele

Initializarea unui nou repozitoriu.

Configurarea VCS.

Commit, push pe branch.

Folosirea fiserului .gitignore.

Revenirea la versiunile anterioare.

Crearea branch-urilor noi.

Commit pe ambele branch-uri.

Merge la 2 branch-uri.

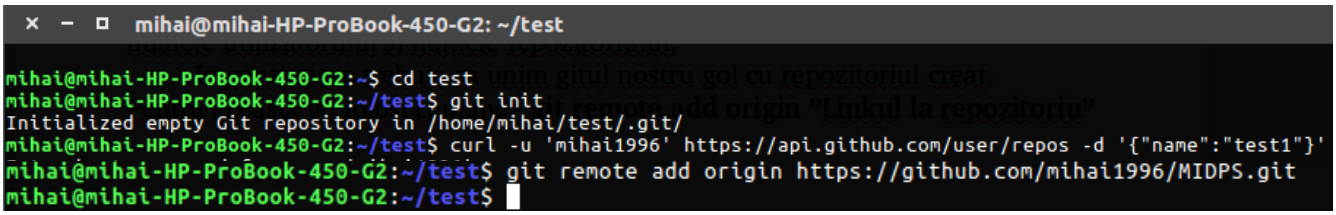
Rezolvarea conflictelor.

3.2 Analiza Lucrarii de laburator

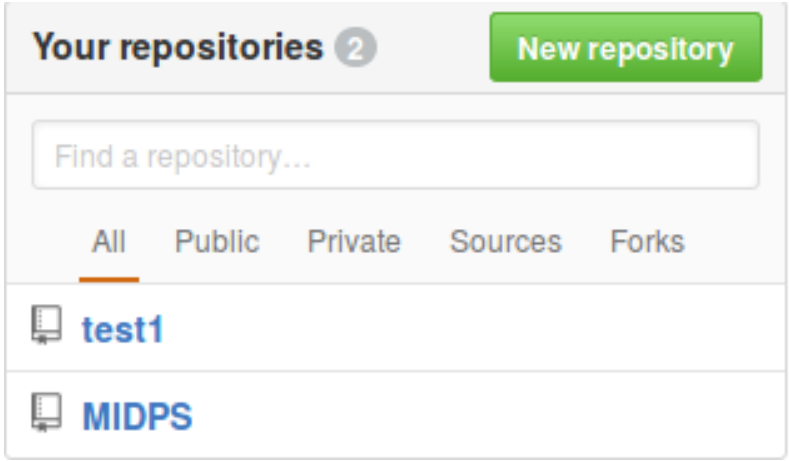
Linkul la repozitoriu **https://github.com/mihai1996/MIDPS.git**
Sunt mai multe modalitati de initializare a repozitoriului pe github.Putem crea o mapa goala in care vom plasa gitul nostru prin intermediul comenzii **git int.**

Urmatorul pas este crearea insusi a noului repozitoriu pe care il vom creea utilizind 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"**



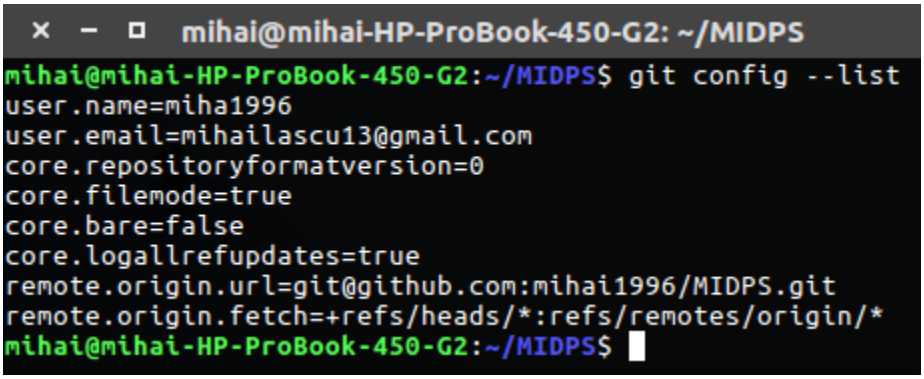
O alta metoda de a crea un repozitoriu este cea online. Pentru aceasta avem nevoie sa deschidem pagina noastra pe ***github,*** si sa apasam butonul **new repositories.**



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"*

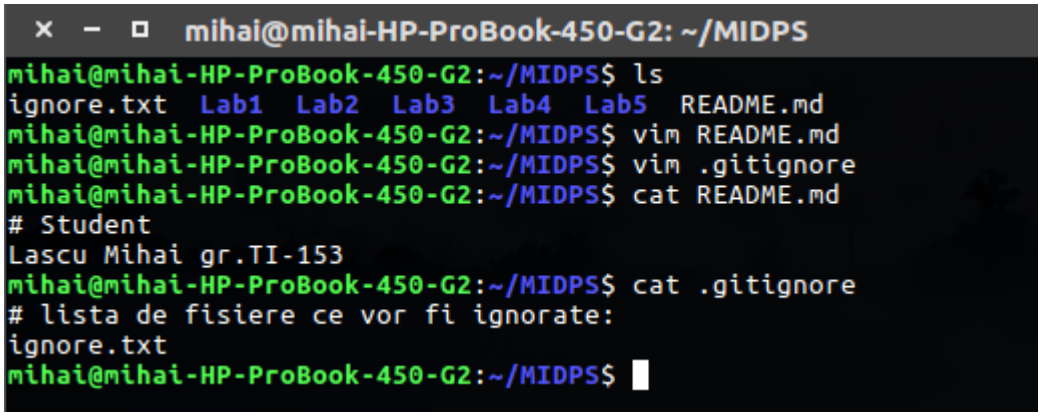
git config -global user.email Email



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 fiserul **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.

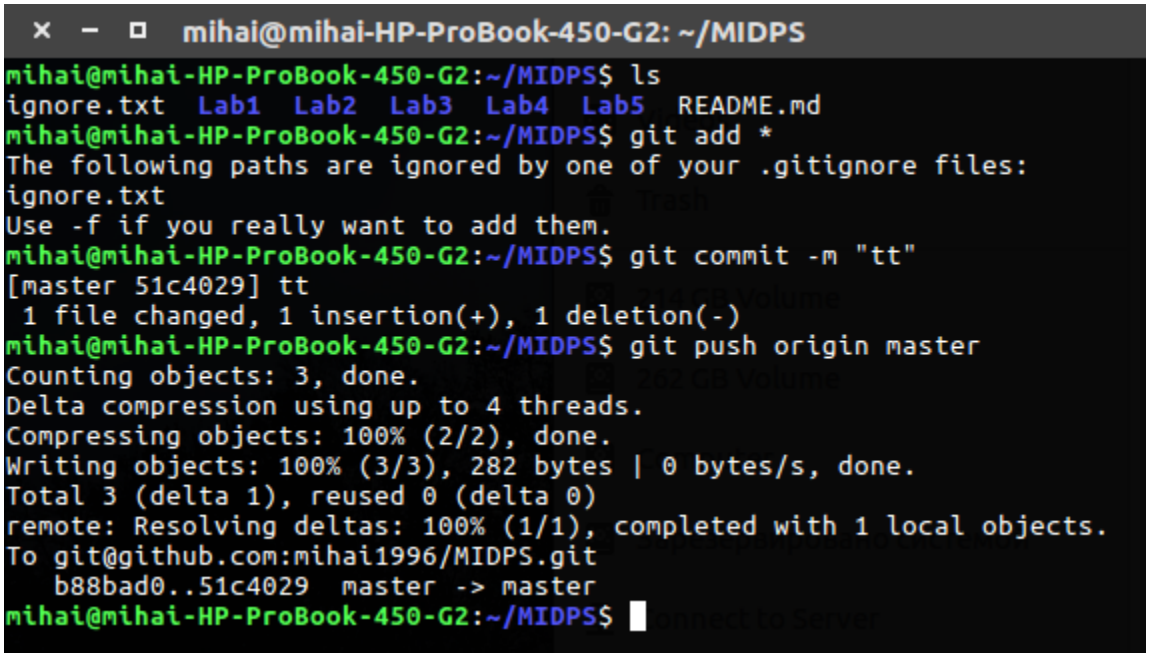


Vom adauga fisierele noi create pe repozitoriul nostru. La moment vom avea nevoie de urmatoarele comenzi:

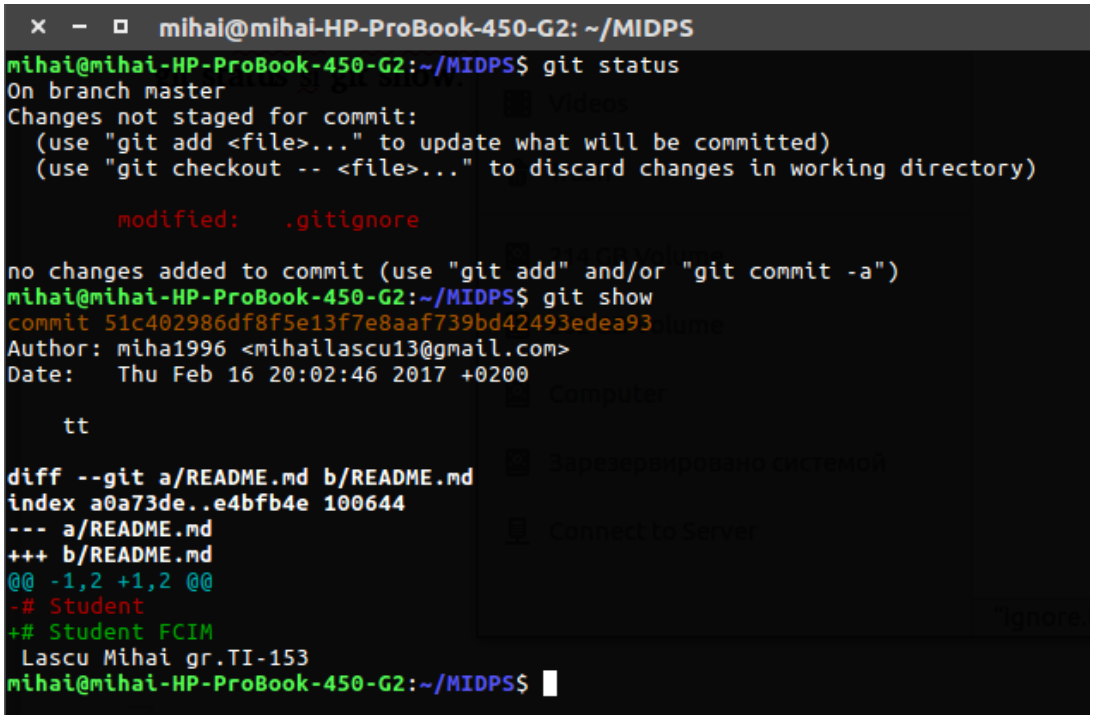
git add * - comanda care indexeaza toate fisierele(*git add .)*;

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

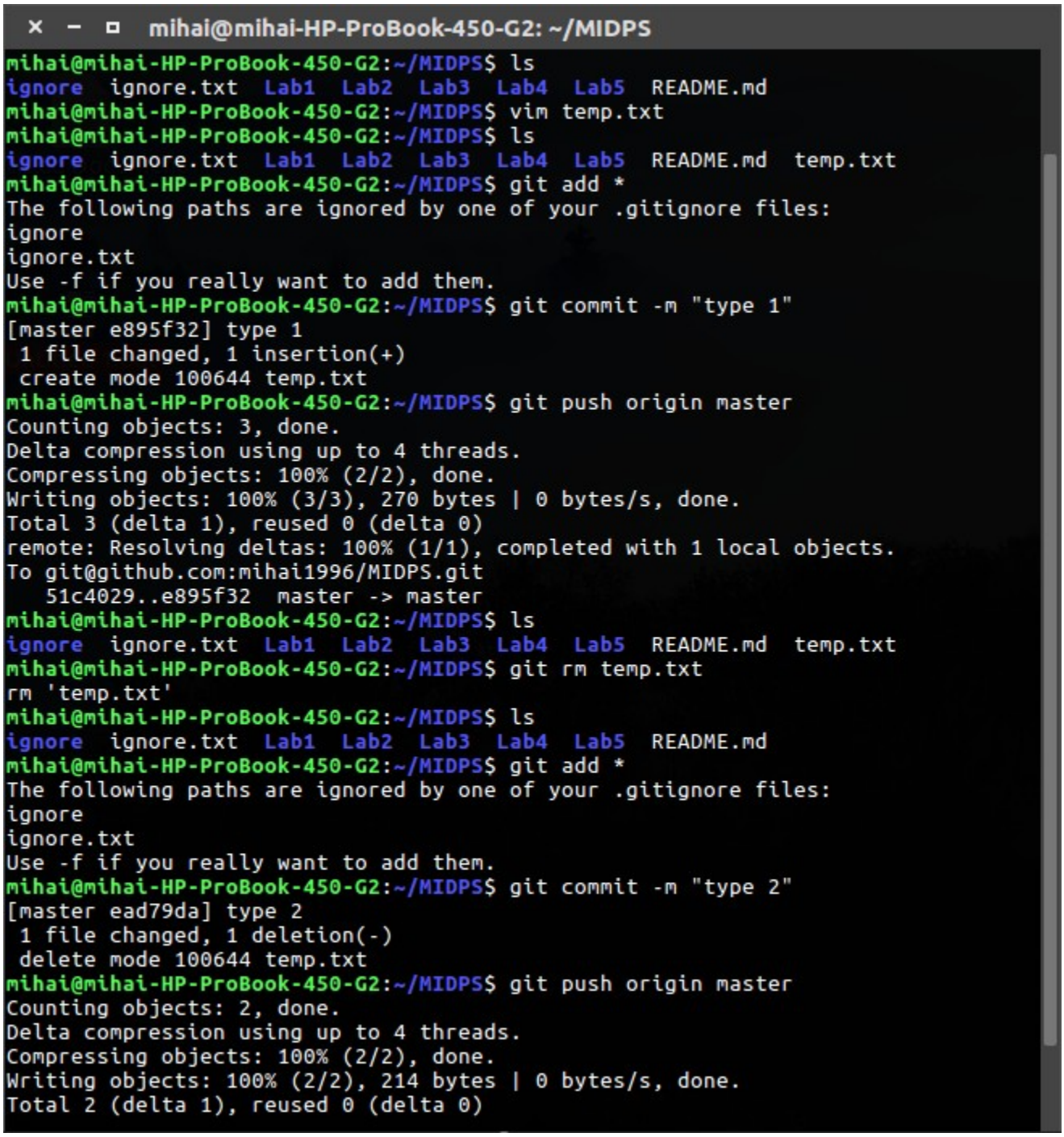
git push origin master – comanda incarca toate fisierele index pe git.



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

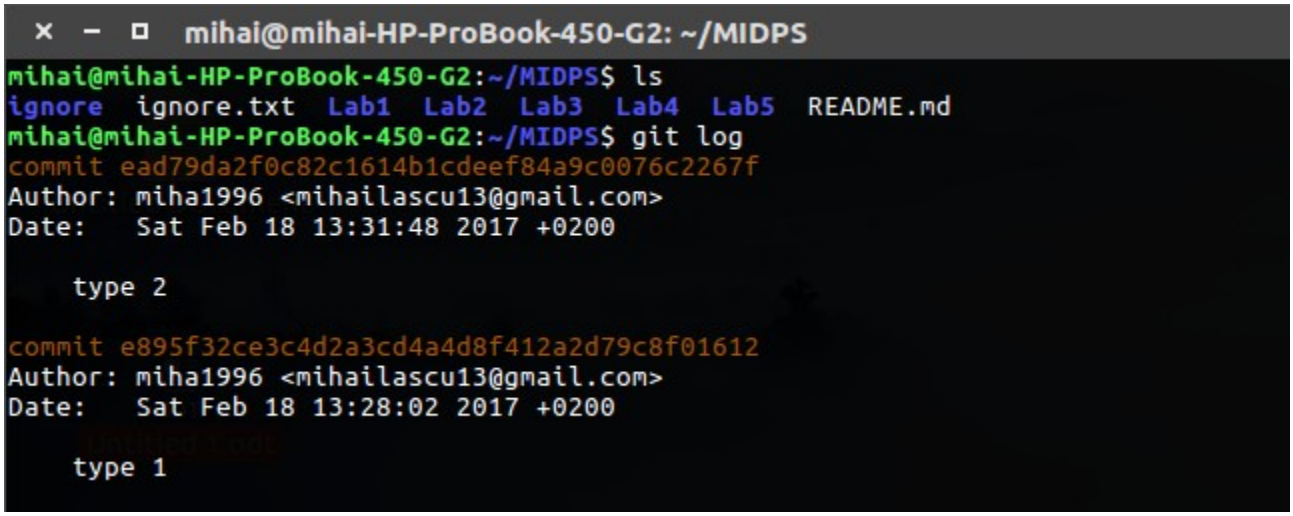


Una dintre caracteristicile principale a unui **VCS** este faptul ca ne permite sa revenim la o versiune mai veche. Aceasta paote fi efectuata cu ajutorul comenzii **git reset -TYPE ”codul comitului”.**

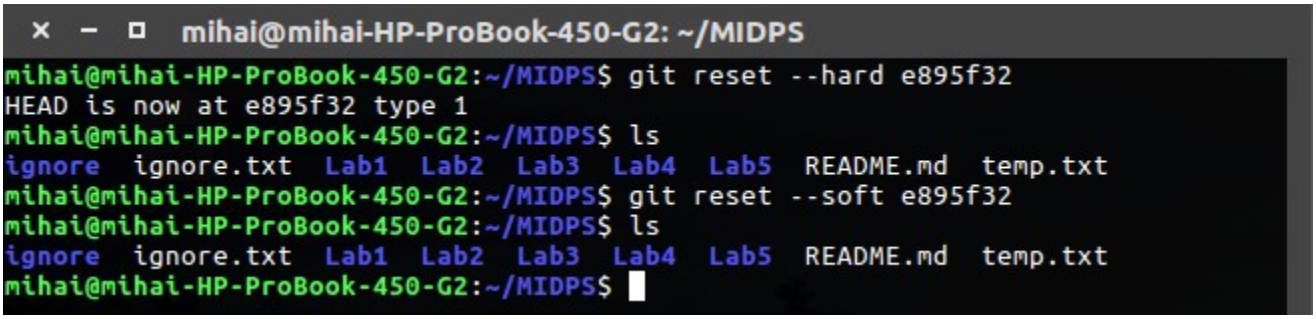


Dupa datele executate observam ca, cind am facut **commit,** s-au ignorat fisierele incluse in .gitignore.

Am creat un fiesier nou *temp.text* 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.



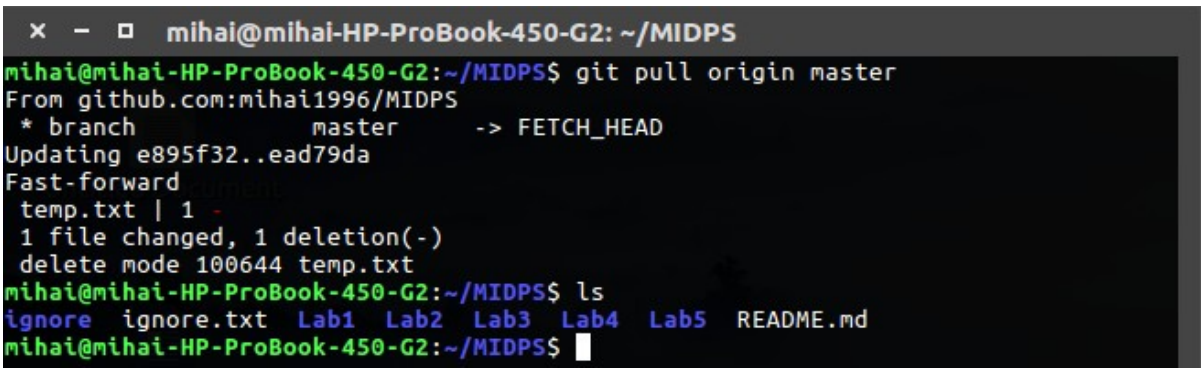
Acum vom folosi comenzile descrise mai sus.



Daca am facut niste schimbari in repozitoriu 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 repozitoriul nostru pina la ultima versiune.



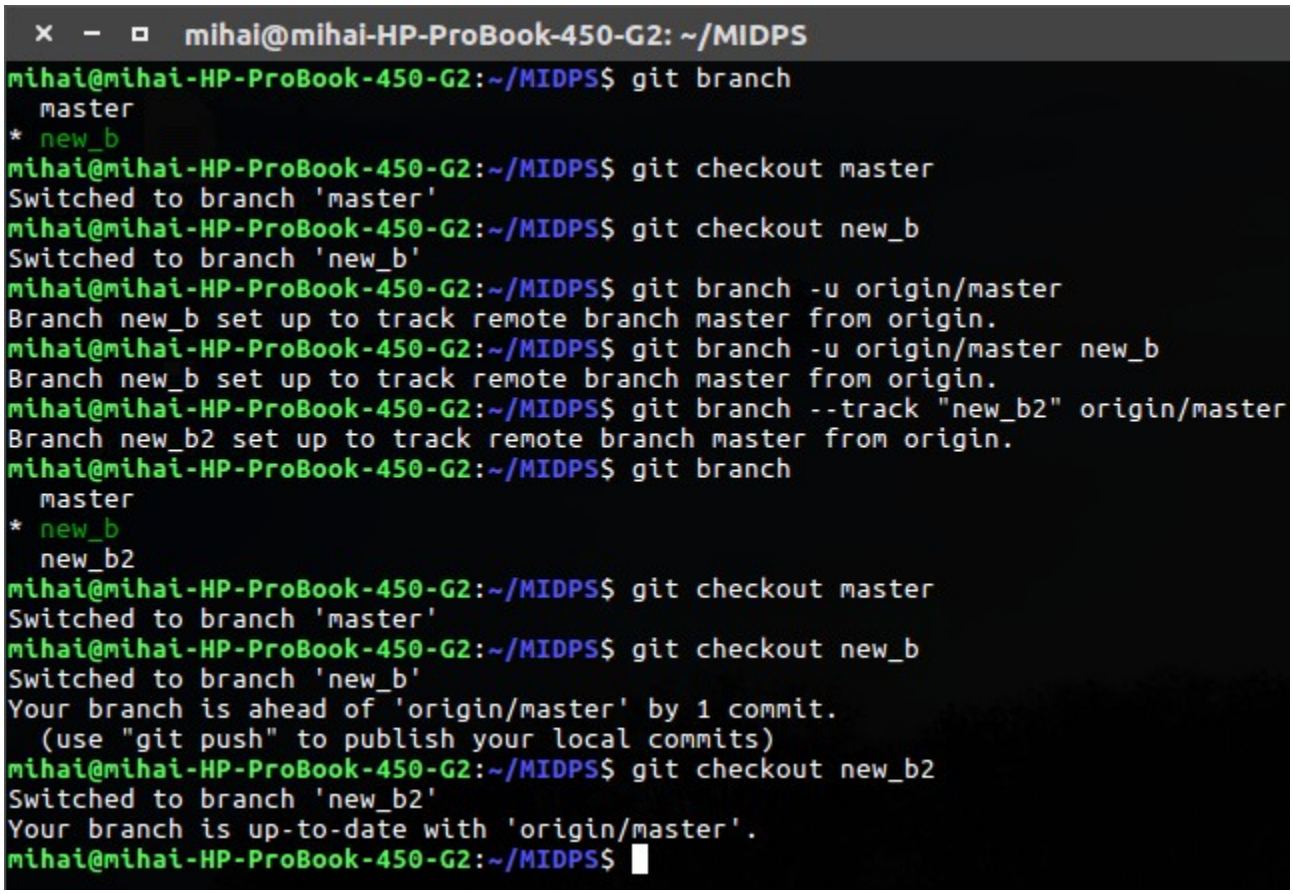
Observam ca dupa ce am executat comanda **git pull** repezitoriul nostru a revenit la ultima versiune, eliminind fisierul *temp.txt*.

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 – vizulaizam 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.

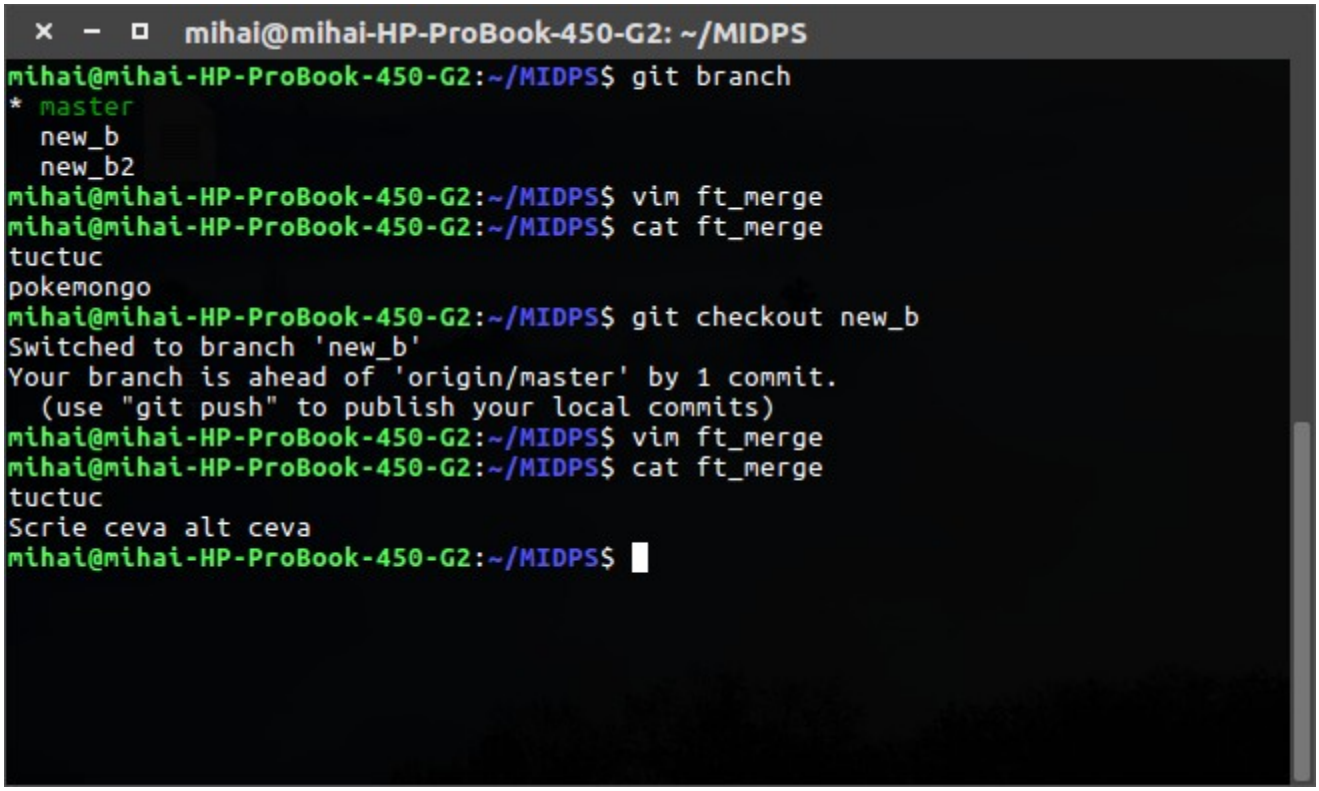
```
mihai@mihai-HP-ProBook-450-G2: ~/MIDPS

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-usptream – scoate tracking-ul la *branch-ul* in care ne aflam.

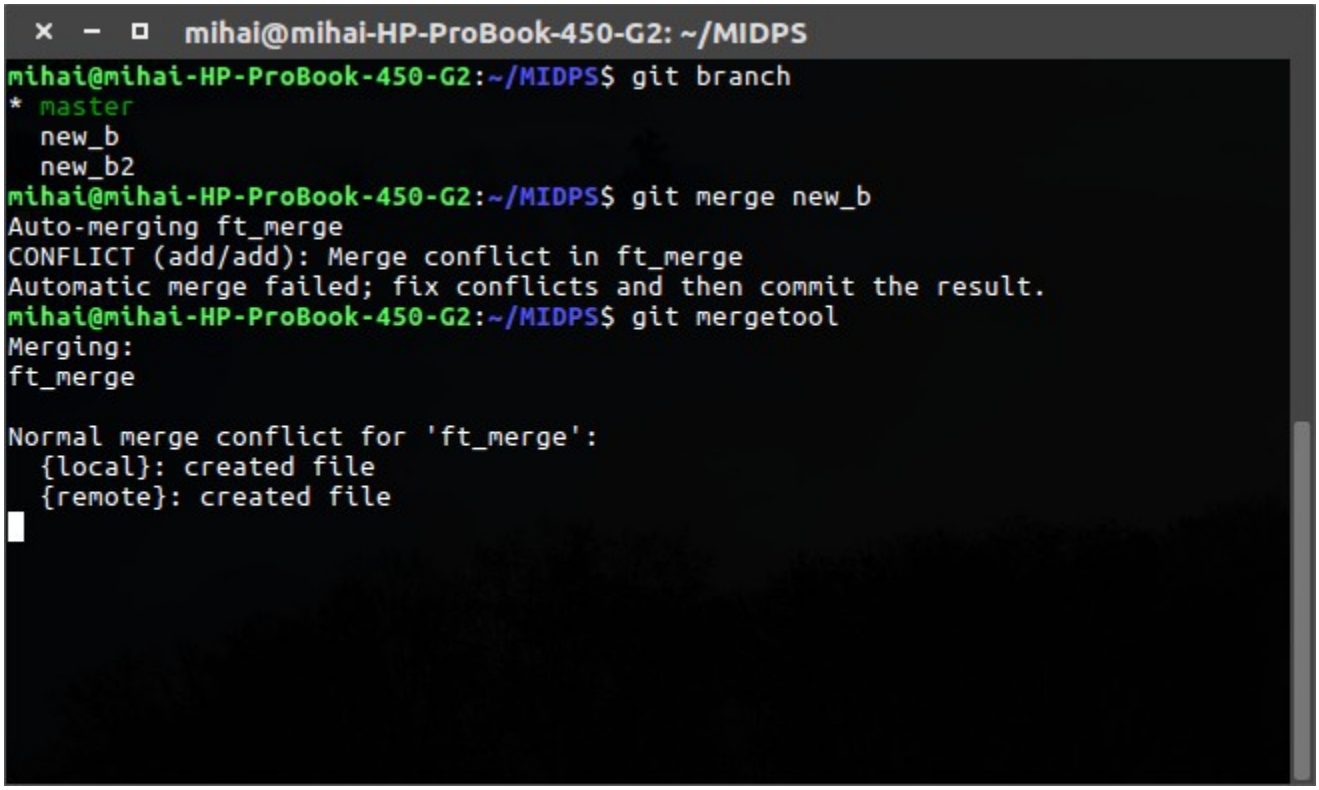


Putem avea *conflicte* in cazul cind dorim sa facem *merge* la 2 *branch-uri* si unele rinduri sunt diferite.In asemenea caz ne vine in ajutor un *mergetool*. Drept *mergetool* am ales **kdiff3.** Pentru a seta *kdiff3* ca *mergetool* default folosim comanda: **git config -global merge.tool kdiff3.**

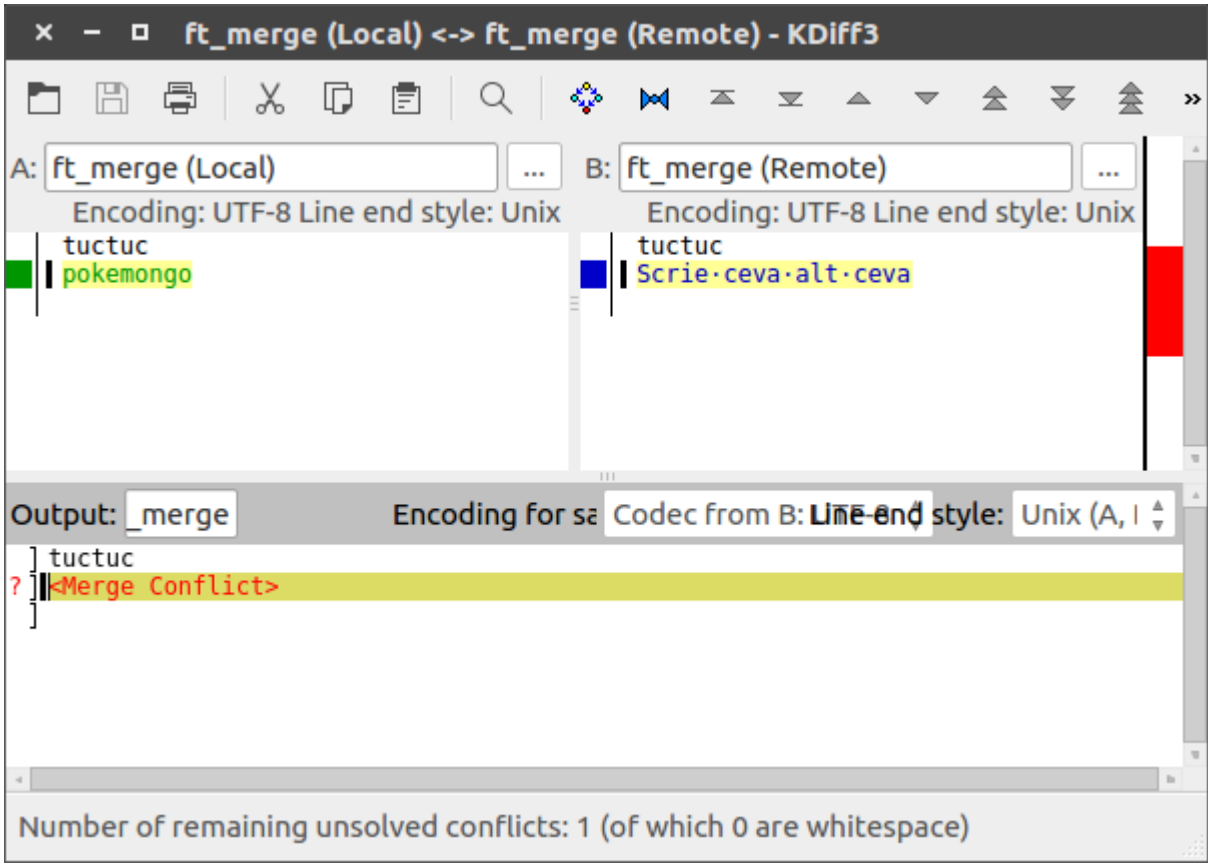
In continuare vom lucra cu 2 *branch-uri* - ”master” si ”new_b”. Vom crea in fiecare *branch* un fisier *”ft_merge”* continutul caruia va fi diferit.



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



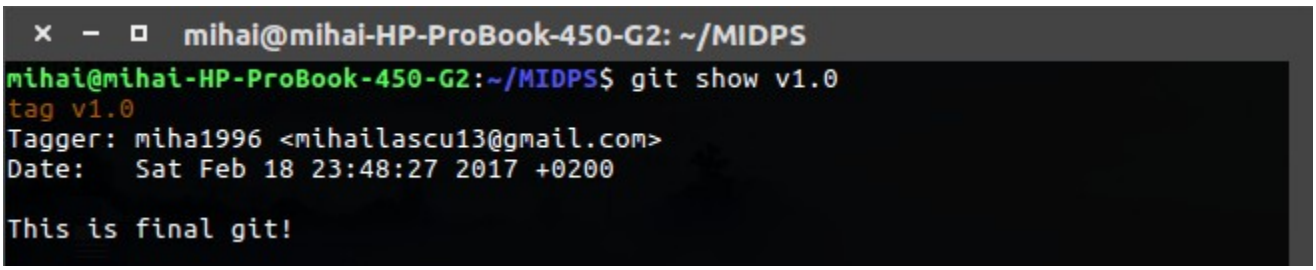
Dupa acest pas rezolvam conflictul cu ajutorul **kdiff3.** Acesta ne afiseaza urmatoarea fereastra:



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

Pentru a putea vedea versiunea taguita folosim comanda:

git show ”tag”



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 atit in lucru cit si in dezvoltarea softului.