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:
Ernest Bitca

lector asistent:
Irina Cojanu
lector superior:
Radu Melnic

Lucrarea de laborator #1

1 Scopul lucrarii de laborator

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

2 Obiective

Studierea Version Control Systems (git).

3 Mersul lucrarii de laborator

3.1 Cerintele

Initializarea unui nou repositoriu.
Configurarea VCS.
Commit, Push pe branch.
Folosirea fisierului .gitignore.
Revenire la versiunele anterioare.
Crearea branch-urilor noi.
Commit pe ambele branch-uri.
Merge la 2 branchuri.
Rezolvarea conflictelor.

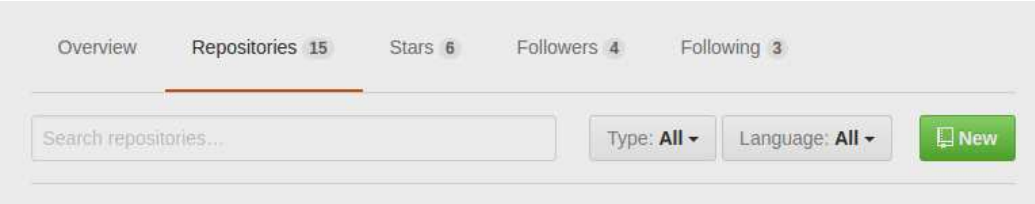
3.2 Analiza Lucrarii de laborator

Linkul la repozitoriu **https://github.com/Ernest96/MIDPS**
Sunt mai multe modalitati de a initializa un repozitoriu pe github. Putem crea o mapa goala in care vom plasa gitul nostru prin intermediul comenzii **git init**.

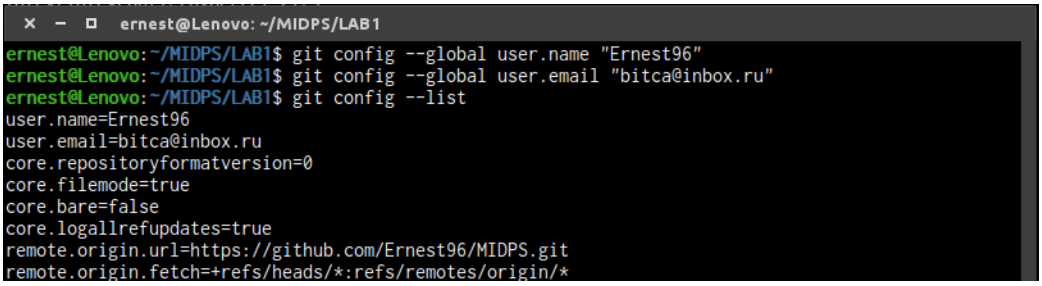
Urmatorul pas este crearea insusi a noului repozitoriu pe care il vom crea utilizind urmatoarea comanda **curl -u 'USER' https.//api.github.com /user/repos -d '{"name":"NUME"}'**. Unde cuvintele scrise cu CAPS se vor inlocui cu numele utilizatorului si numele repozitoriului.

Dupa aceasta este necesar sa unim gitul nostru gol cu repozitoriul creat. Vom folosi urmatoarea comanda **git remote add origin "Linkul la repo"**

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

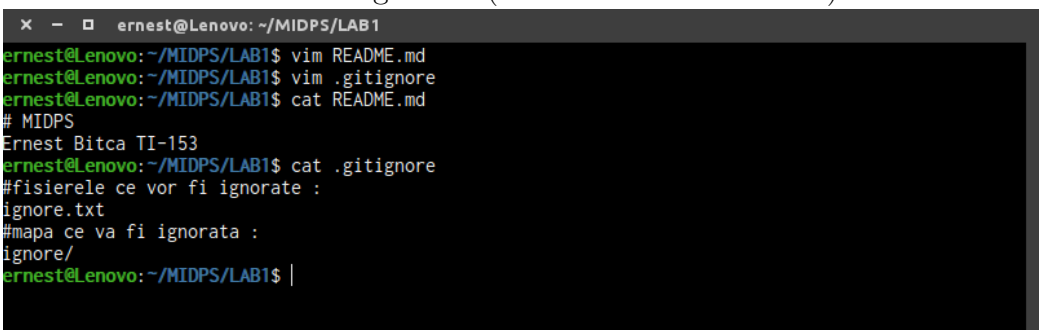


Configurarea gitului const in mai multe etape. La inceput vom configura numele si emailul. Scrim urmatoarele comenzi:
git config –global user.name "NUMELE"
git config –global user.email EMAIL

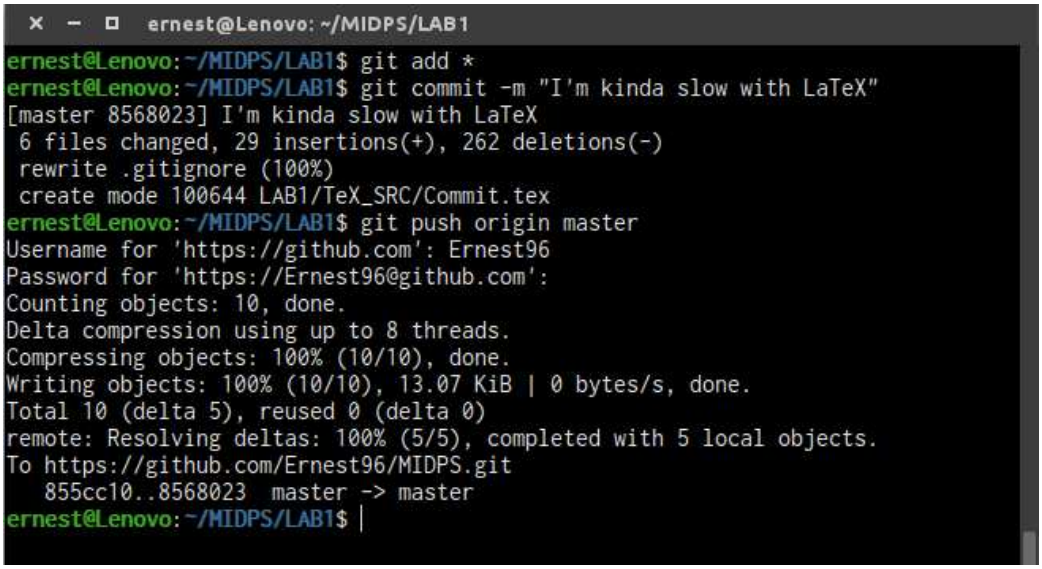


Urmatorul pas consta in generarea la cheia **SSH** (Secure Shell). Scriem in CLI **ssh-keygen**, iar cheia obtinuta o copiem in setarile noastre de pe git.

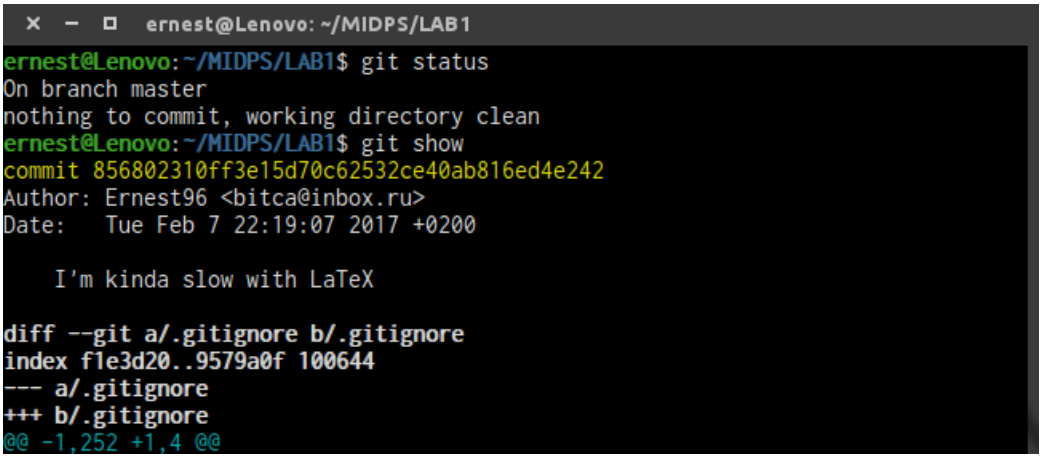
Este de dorit sa initializam repozitoriul nostru cu un fisier **README.md** si un **.gitignore**. In fisierul README.md vom adauga niste informatie pentru cei care se vor folosi de repozitoriu iar in fisierul .gitignore vom adauga toate fisierele ce trebuiesc ignorate (adica sa nu fie incarcate).



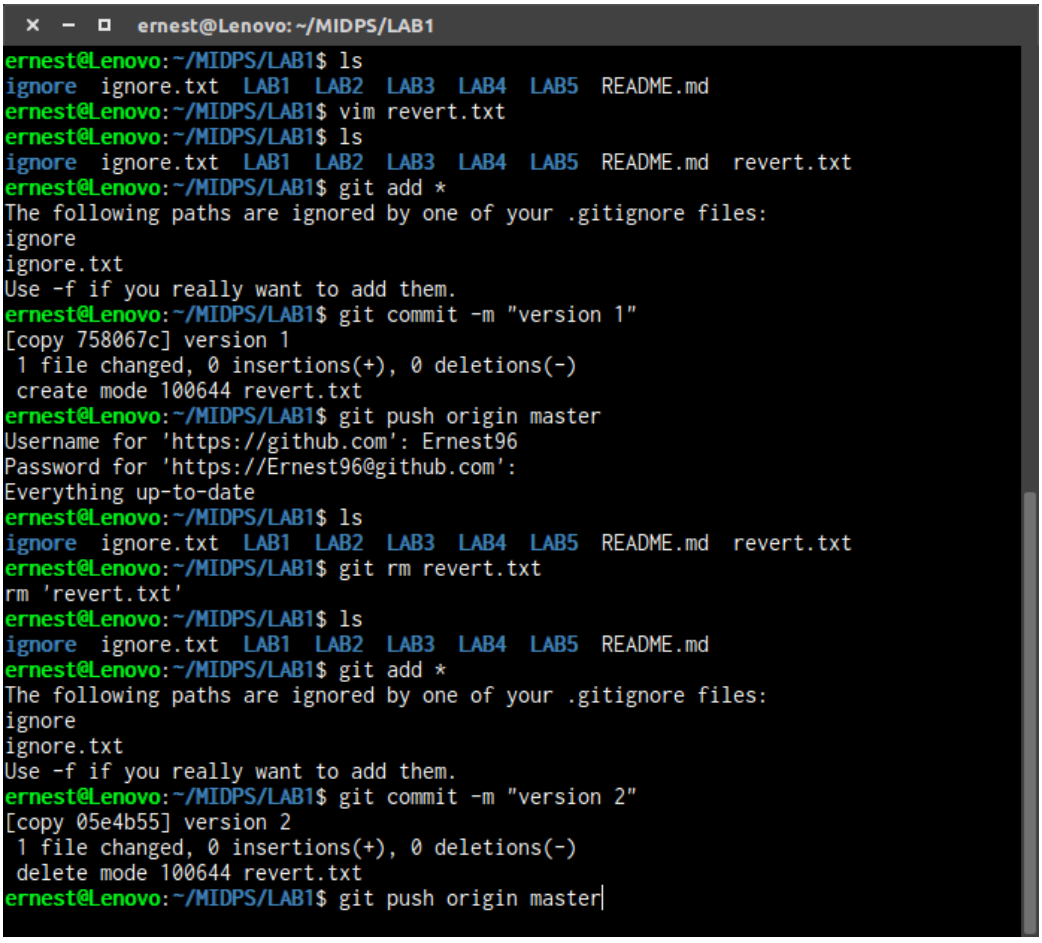
Vom adauga fisierele noi create pe repozitoriul nostru. Pentru aceasta vom avea nevoie de urmatoarele comenzi :
git add * - comanda indexeaza toate fisierele.
git commit -m - comanda face un snapshot la toate schimbarile noastre.
git push origin master - comanda incarca toate fisierele indexate pe git.



Pentru a ne asigura ca am facut totul bine si nu avem probleme vom utiliza :**git status** si **git show**

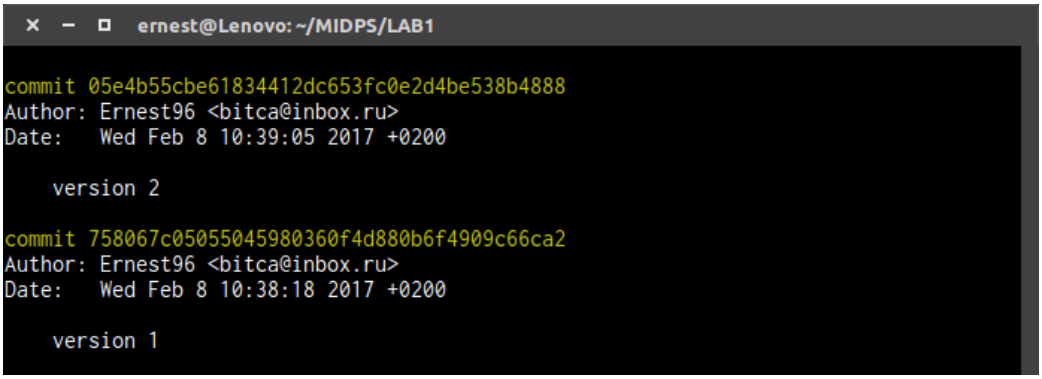


Una dintre caracteristicele principale a unui **VCS** este faptul ca ne permite sa revenim la o versiune mai veche. Aceasta poate fi efectuata cu ajutorul comenzii **git reset –TYPE "codul comitului"**. Exista diferenta intre –**soft** si –**hard** , cind facem soft reset indexurile ramin neschimbate. Iar in cazul cind facem hard reset , pierdem indexurile.

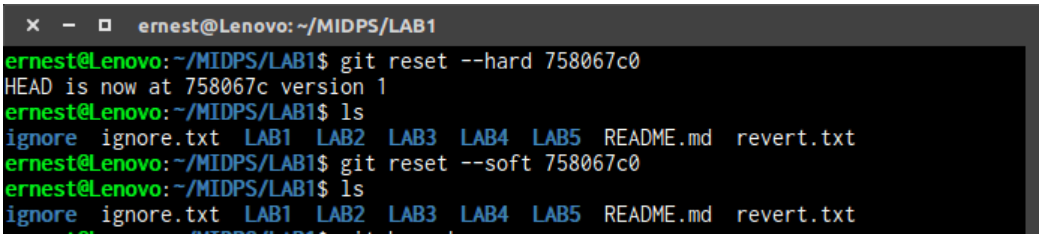


Dupa cum observam cind am facut commit, s-au ignorat fisierele incluse in .gitignore.

Am creat un fisier nou revert.txt in versiunea 1. Dupa care l-am sters si am facut commit la versiunea 2 in care am sters fisierul revert.txt dorim sa revenim la versiunea1. La inceput vom lansa comanda **git –log** care ne arata logul de commituri si codul pentru fiecare commit. Vom avea nevoie de primele 7 cifre la commitul anterior.



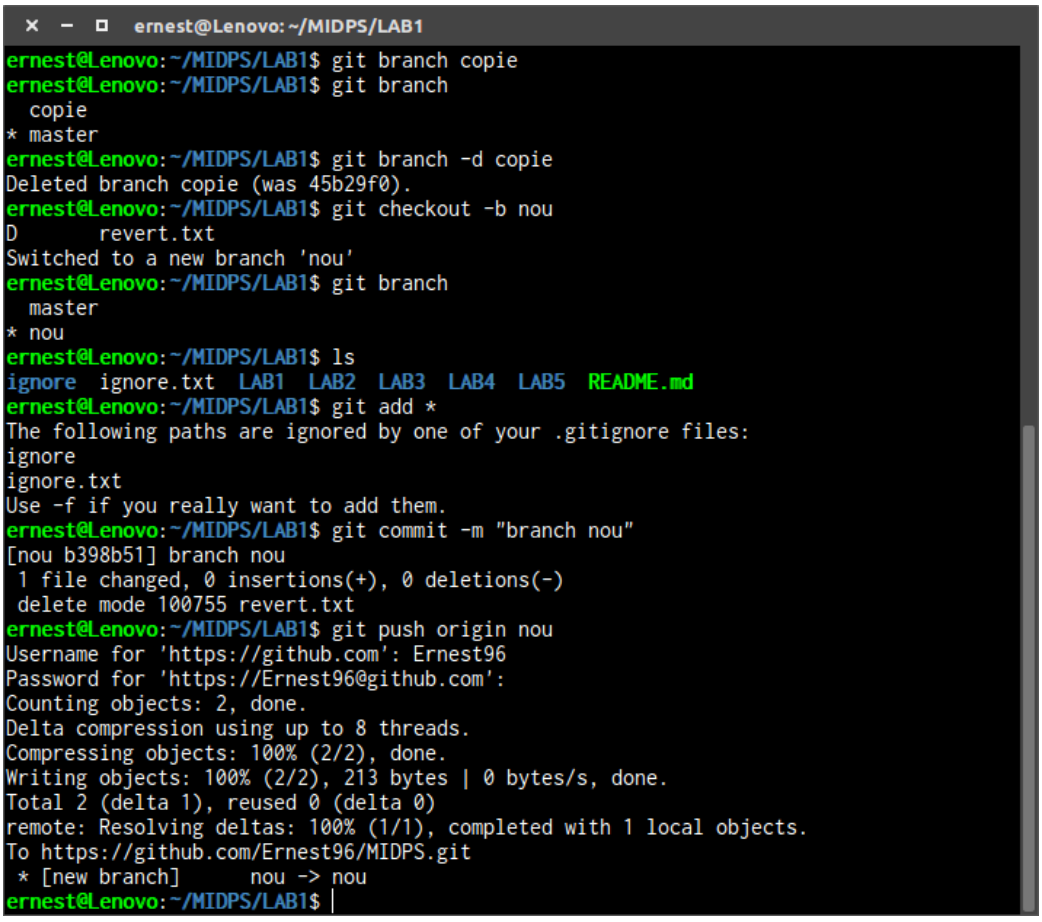
Acum vom folosi comenzile descrise anterior.



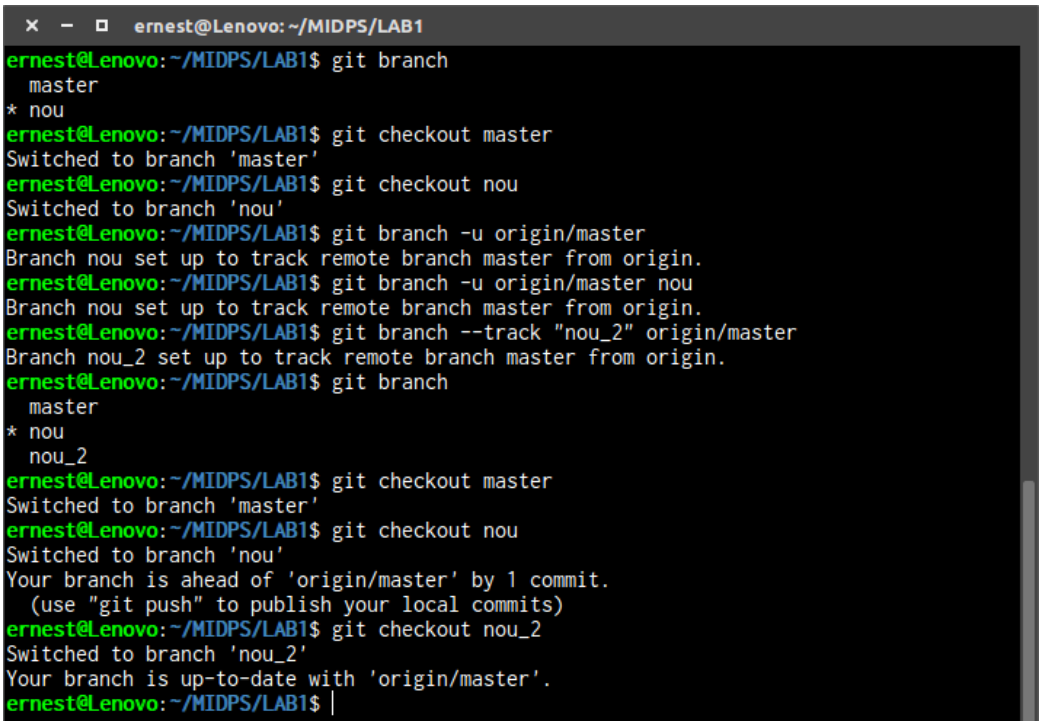
Daca am facut niste schimbari in repozitoriu si nu ne satisfac, putem usor reveni la ultima versiune care era pe git utilizind comanda **git pull origin "branch"** Comanda **git pull** va face update la repozitoriul nostru pina la ultima versiune.

VCS ne permite sa avem mai multe **branchuri**. Din traducere branch semnifica "creanga". Branchurile 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 nou cu numele "name".
git branch - vizualizarea branchurilor (* indica branchul curent).
git branch -d "nume" - sterge branchul "nume".
git checkout -b "name" - creeaza un branch nou cu numele "name" si face switch la el.

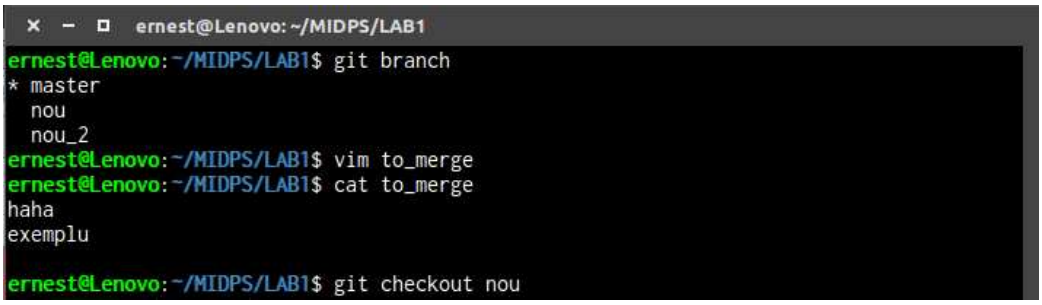


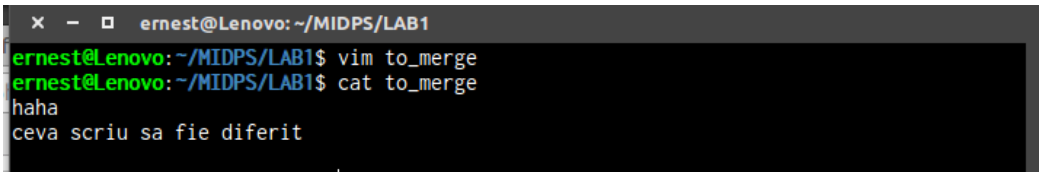
git checkout "nume" - face switch la branchul "nume".
git branch -u upstream/name - face track la branchul indicat din branchul curent.
git branch -u upstream/name "nume" - face track din branchul "nume" la branchul indicat.
git branch –track "name" upstream/name - creeaza branchul "name" si ii face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



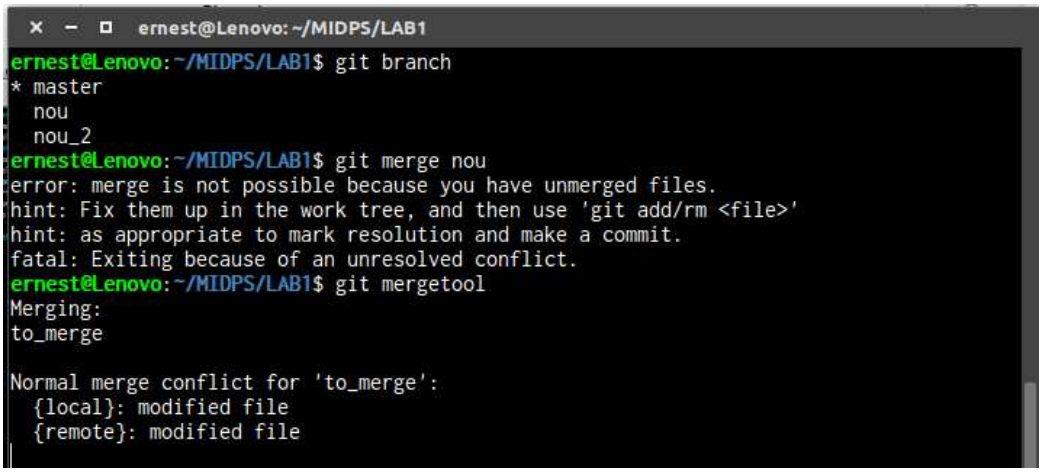
Putem avea conflicte in cazul cind dorim sa facem merge la 2 branchuri si unele rinduri sunt diferite. In asa caz ne vin in ajutor un mergetool. Drept mergetool am ales **kdiff3**. Pentru a seta kdiff3 ca mergetool default folosim comanda : **git config –global merge.tool kdiff3**

In continuare vom lucra cu 2 branchuri - "master" si "nou". Vom crea in fiecare branch cite un fisier "tomerge" continutul caruia va fi diferit.



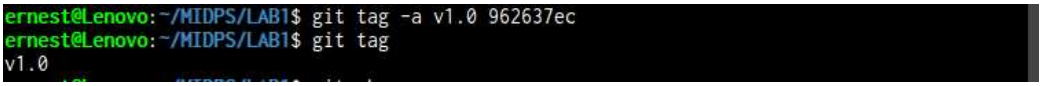


In continuare vom incerca sa facem merge si sa rezolvam acest conflict.

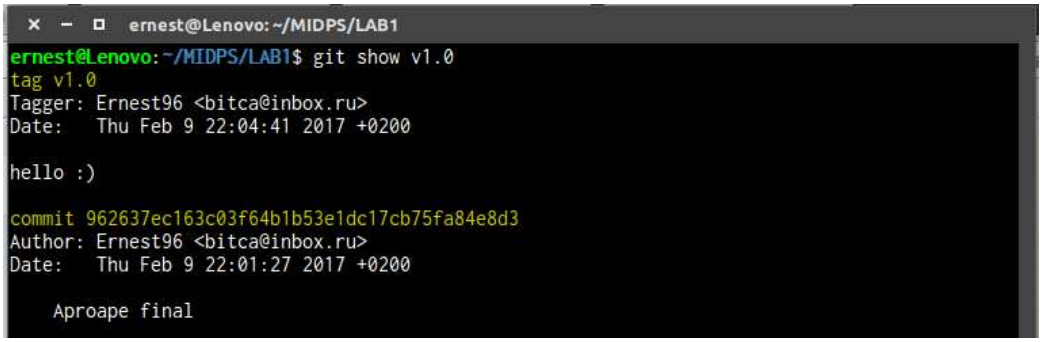


Dupa acest pas rezolvam conflictul cu ajutorul **kdiff3**. De exmplu eu am ales sa fac merge in felul urmator.

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



Dupa aceasta putem vedea usor o versiune taguita. Folosim comanda **git show "tag"**



4 Concluzii

In lucrarea data am studiat lucrul cu **VCS**. Am luat cunostinta cu platforma **github**. Toate lucrurile, comenzile le-am indeplinit in terminal pe Linux. Sunt foarte multe plusuri in folosirea VCS. Fara el producerea produselor soft ar fi foarte lenta si problematica. El ne permite lucrul paralel, menajarea versiunelor, revenire la versiuni anterioare. In lucrare am pus in practica majoritatea comenzilor esentiale. Nu este prima mea experienta cu git-ul, insa mi-am imbunatatit nespus de mult lucrul cu el. Am luat cunostinta cu branchurile, merge la branchuri si rezolvarea conflictelor (utilizind kdiff3). Dupa parerea mea orice programator contemporan necesita cunostinta unui VCS. El contribuie nu doar la dezvoltarea hard-skillurilor dar si a celor soft.