# Facultatea Calculatoare, Informatica si Microelectronica

# Universitatea Tehnica a Moldovei


Medii Interactive de Dezvoltare a

Produselor Soft

Lucrarea de laborator Nr.1

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


*A efectuat* : **Negru Igor**

*lector asistent* : **Cojanu Irina**

*lector superior :* **Melnic Radu**

# Lucrarea de laborator Nr.1

- ## Scopul lucrarii de laborator :

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

- ## Obiective

Studierea Version Control Systems (git).

- ## Mersul lucrarii de laborator

### *3.1 Cerintele :*

* Initializare unui nou repositoriu.

* Configurarea VCS.

* Commit, Push branch.

* Folosirea fisierului .gitignore.

* Revenire la versiunele anterioare.

* Crearea branch-urilor noi.

* Commit pe ambele branch-uri.

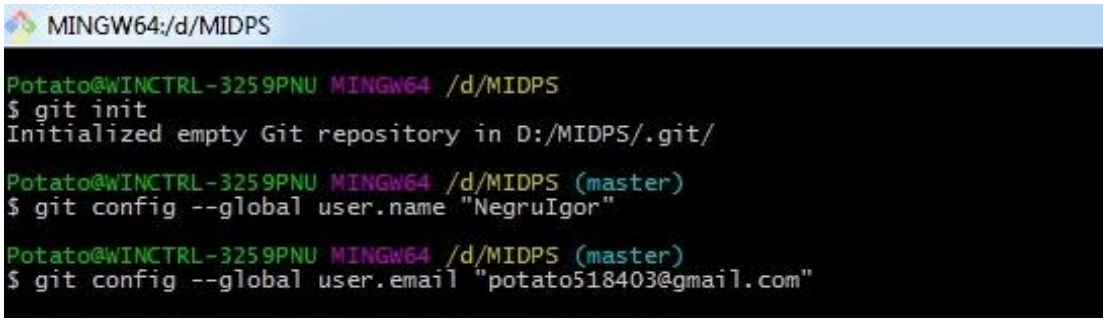
* Merge la 2 branchuri.

* Rezolvarea conflictelor.

- ## Analiza lucrarii de laborator :

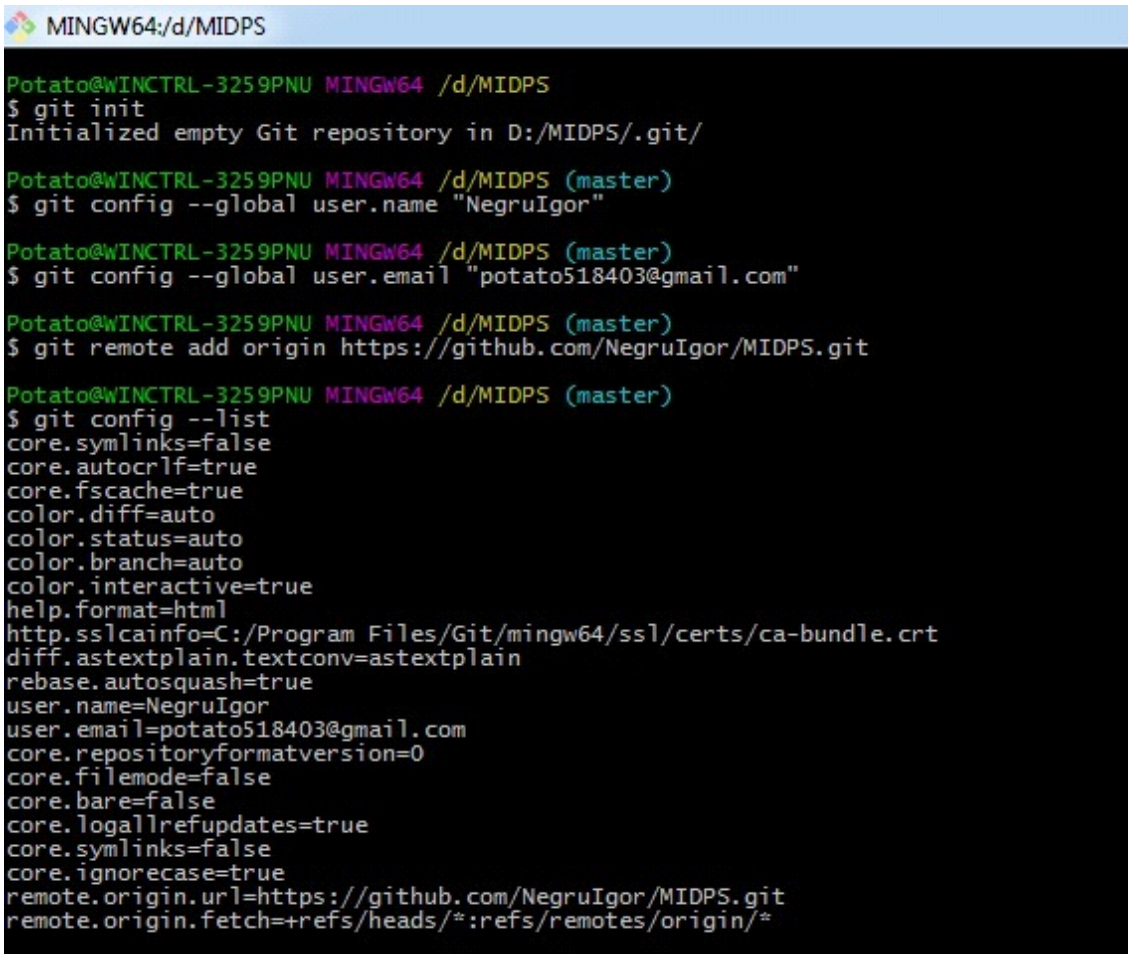
Linkul la repozitoriu **https://github.com/NegruIgor/MIDPS**
Sunt mai multe modalitati de a initializa un repozitoriu pe github. Putem crea o mapa goala in care vom plasa gitul nostru prin intermediul comenzii **git init.**

Urmatorul pas este crearea 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 repozitoruui. Dupa aceasta este necesar sa unim gitul nostru gol cu repozitoriul creat. Vom folosi urmatoare camonda **git remote add origin "Linkul la repozitoriul nostru"**




**Configurarea gitului** consta in mai multe etape. La inceput vom configura numele si emailul. Scriem urmatoarele comenzi :

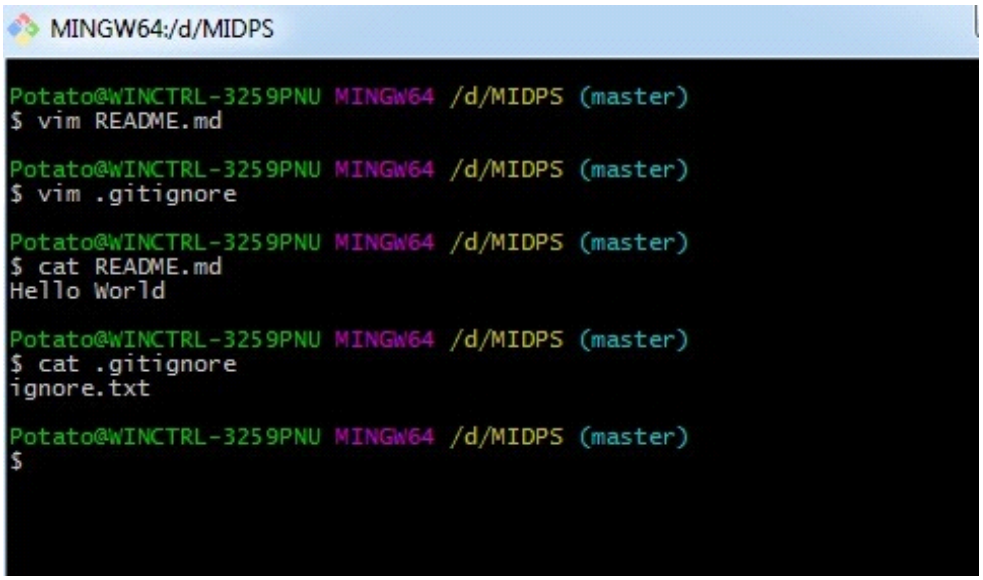
**git config –global user.name “Numele”**
**git config –global user.email “Email”**



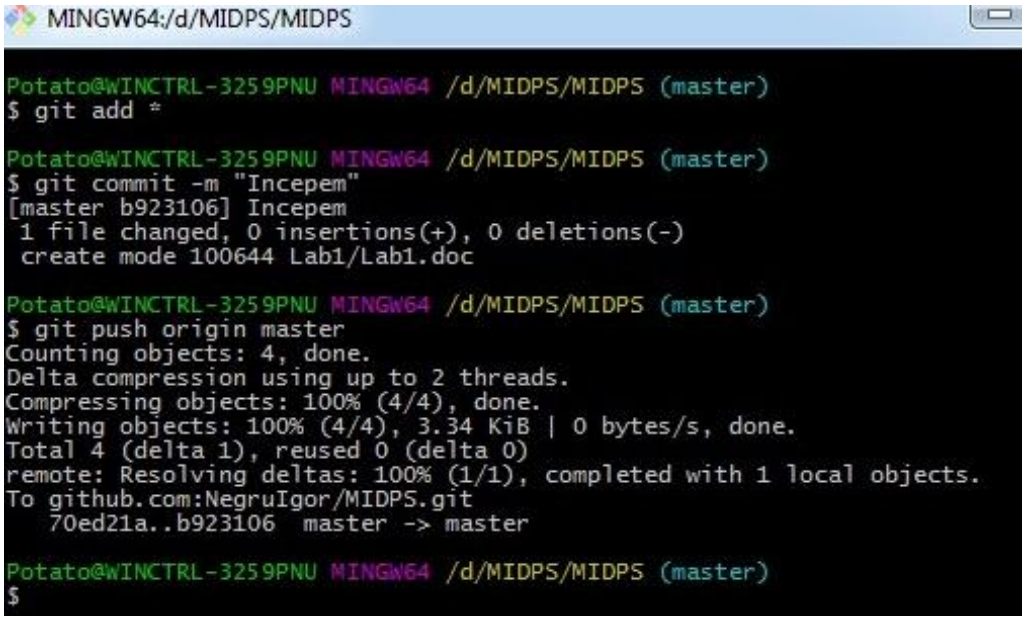

Urmatorul pas consta in generarea la cheis **SSH.** Scriem in CLI **ssh-keygen,** iar cheia obtinuta o copiem in setarile noastre de pe github.com.

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

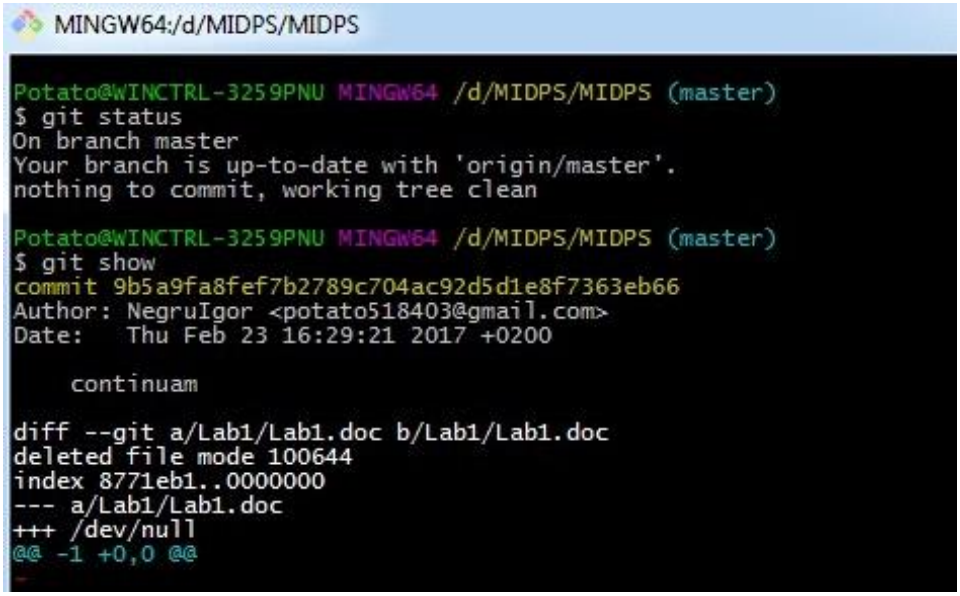
repozitoriu iar in fisierul .gitignore vom adauga toate fisierele ce trebuiesc ignorate (adica sa nu fie incarcate ).




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 "TEXT" –** comanda face un snapshot la toate schimbarile noastre.
**git push origin master –** comanda incarca toate fisierele indexate pe **github.com**

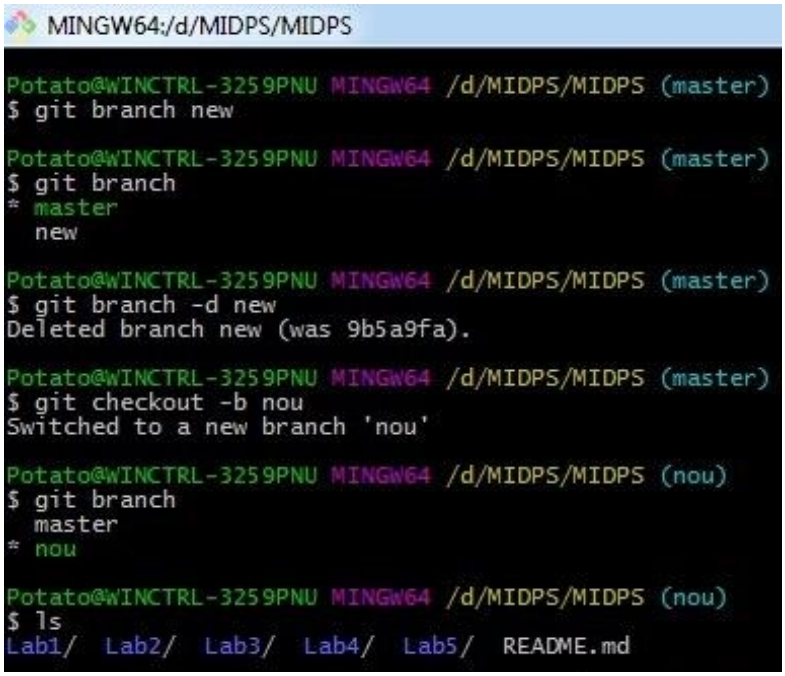



Pentru a ne asigura ca am facut totul bine si nu avem probleme vom utiliza :
***git status**
***git show**




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 “name” –** sterge branchul “name”.
**git checkout –b “name” -** creeaza un branch nou cu numele “name” si face switch la el.




**git checkout “name” –** face switch la branchul “name”.
**git branch –u upstream/name –** face track la branchul indicat din branchul curent.

**git branch –u upstream/name “name” –** face track din branchul “name” la branchul indicat.
**git branch –track “name” upstream/name –** creeaza branchul “name” si ii face track la branchul indicat.
**git branch –unset-upstream –** scoate trackingul la branchul in care ne aflam.

```
Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git add *

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git commit -m "branch nou"
On branch nou
nothing to commit, working tree clean

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git push origin nou
Total 0 (delta 0), reused 0 (delta 0)
To github.com:NegruIgor/MIDPS.git
 * [new branch]      nou -> nou
```

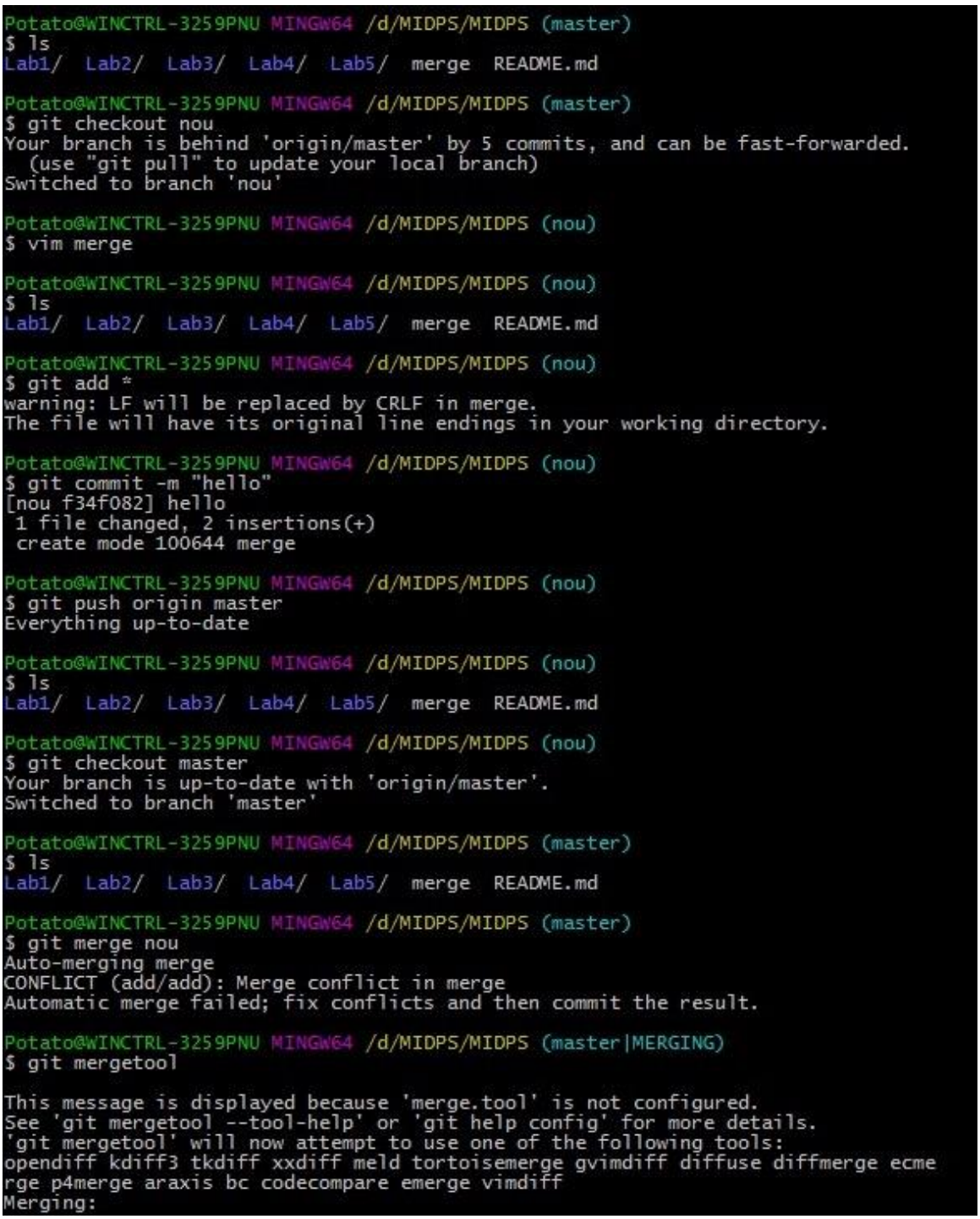
MINGW64:/d/MIDPS/MIDPS

```
Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git branch
  master
* nou

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git checkout master
Your branch is up-to-date with 'origin/master'.
Switched to branch 'master'

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git checkout nou
Switched to branch 'nou'

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git branch -u origin/master
Branch nou set up to track remote branch master from origin.

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git branch -u origin/master nou
Branch nou set up to track remote branch master from origin.

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git branch --track "nou_2" origin/master
Branch nou_2 set up to track remote branch master from origin.

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git branch
  master
* nou
  nou_2

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git checkout master
Your branch is up-to-date with 'origin/master'.
Switched to branch 'master'

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git checkout nou
Your branch is up-to-date with 'origin/master'.
Switched to branch 'nou'

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git checkout nou_2
Your branch is up-to-date with 'origin/master'.
Switched to branch 'nou_2'

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou_2)
$
```

```
MINGW64:/d/MIDPS/MIDPS

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou_2)
$ git branch
  master
  nou
* nou_2

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou_2)
$ vim to_merge

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou_2)
$ cat to_merge
haha
lol

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou_2)
$ git checkout nou
Your branch is up-to-date with 'origin/master'.
Switched to branch 'nou'

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ vim to_merge

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ cat to_merge
haha
ceva diferit
```

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 mergetool. Drept mergetool am ales **kdiff3.** Pentru kdiff3 ca mergetool default folosim comanda : **git config –global merge.tool kdiff3**
In continuare vom lucra cu 2 branchuri – “master” si “new”. 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 rezovam conflictul co ajutorul **kdiff3.** De exemplu eu am ales sa fac merge in felul urmator.




- ***Concluzie***

In lucrarea nr.1 la MIDPS am studiat lucrul cu **VCS**. Am cunoscut platforma **github**. Toate lucrurile, comenzile le-am indeplinit in terminal pe Windows. Sunt o multime de plusuri in folosirea VCS. Fara VCS elaborarea produselor soft ar fi foarte lenta si problematica. El ne permite lucrul paralel, menajarea versiunelor, revenire la versiuni anterioare. In lucrare am practicat majoritatea comenzilor esentiale. Este prima mea experienta cu github.com si mi-am imbunatatit nespus de mult lucrul pe aceasta platforma. Am cunoscut branchurile, merge la

branchuri si rezolvarea conflictelor. Dupa parerea mea orice programator contemporan necesita cunostinta unui VCS. El contribuie nu doar la dezvoltarea hard-skillurilor dar si a celor soft.

- ***Referinte :***

1.https://github.com/BestMujik/MIDPS-labs/blob/master/MIDPS_LAB%231.md

2. https://github.com/Ernest96/MIDPS/blob/master/LAB1/Lab%231.pdf