Ministerul Educaţiei al Republicii Moldova

Universitatea Tehnică a Moldovei
Catedra Tehnologii Informaţionale

RAPORT

Lucrarea de laborator#2

la Medii Interactive de Dezvoltare a Produselor Soft

A efectuat: Profir Andrei
st.gr. TI – 143

A verificat: Cojanu Irina
lect.asist.

Chişinău 2016

Tema: Version Control Systems si modul de setare a unui server

Scopul lucrării:

- Intelegerea si folosirea CLI (basic level)
- Administrarea remote a masinilor linux machine folosind SSH (remote code editing)
- Version Control Systems (git || mercurial || svn)
- Compileaza codul C/C++/Java/Python prin intermediul CLI, folosind compilatoarele gcc/g++/javac/python

Formularea condiţiei problemei (sarcina de lucru):

- *Basic Level* (nota 5 || 6) :
  - conecteaza-te la server folosind SSH
  - compileaza cel putin 2 sample programs din setul HelloWolrdPrograms folosind CLI
  - executa primul commit folosind VCS
- *Normal Level* (nota 7 || 8):
  - initializeaza un nou repositoriu
  - configureaza-ti VCS
  - crearea branch-urilor (creeaza cel putin 2 branches)
  - commit pe ambele branch-uri (cel putin 1 commit per branch)
- *Advanced Level* (grade 9 || 10):
  - seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)
  - reseteaza un branch la commit-ul anterior
  - merge 2 branches
  - conflict solving between 2 branches
- *Bonus Point*:
  - Scrie un script care va compila HelloWolrdPrograms projects.

Implimentare task-uri:

- *Basic Level* (nota 5 || 6) :
  - conecteaza-te la server folosind SSH

```
Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2 (master)
$ git clone git@github.com:TUM-FAF/IDE.git clone
Cloning into 'clone'...
remote: Counting objects: 297, done.
remote: Total 297 (delReceivinta 0), reused 0 (delta 0)g objects:  83% (247, pac
k-/297)reused 297
Receiving objects: 100% (297/297), 68.13 KiB | 0 bytes/s, done.
Resolving deltas: 100% (163/163), done.
Checking connectivity... done.
```

- compileaza cel putin 2 sample programs din setul HelloWolrdPrograms folosind CLI

```
Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2 (master)
$ cd ./clone/HelloWorldPrograms/java

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2/clone/HelloWorldProg
rams/java (master)
$ javac hello.java

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2/clone/HelloWorldProg
rams/java (master)
$ java HelloWorld
Hello World
```

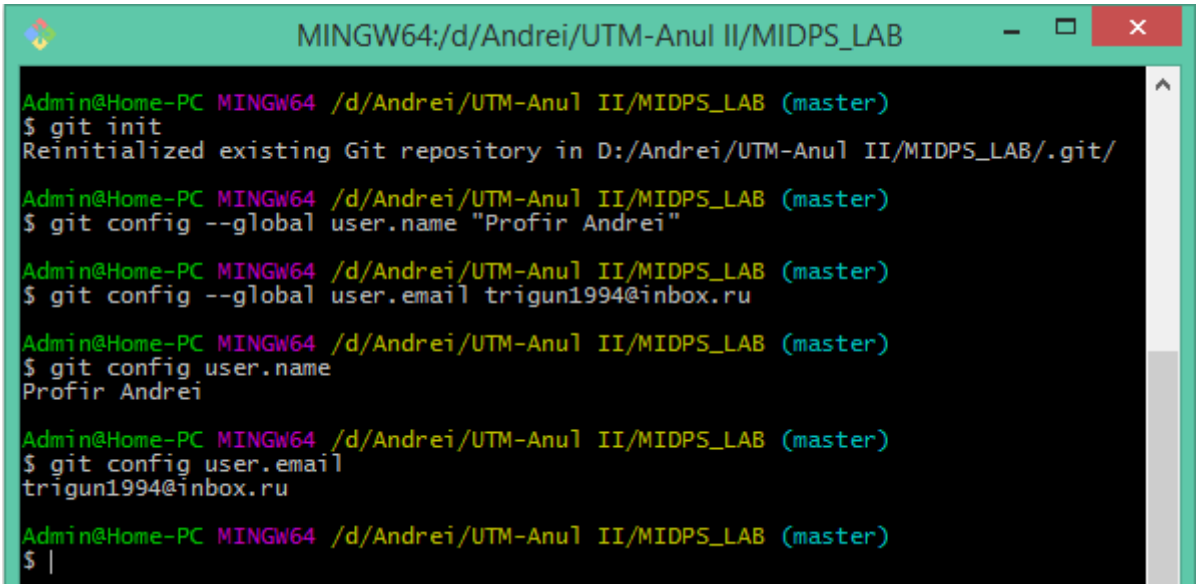
```
Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2/clone/HelloWorldProg
rams/c (master)
$ gcc -o hello.exe hello.c

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2/clone/HelloWorldProg
rams/c (master)
$ ./hello.exe
Hello world

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2/clone/HelloWorldProg
rams/c (master)
$ cd ../cpp

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2/clone/HelloWorldProg
rams/cpp (master)
$ g++ -o hello.exe hello.cpp

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2/clone/HelloWorldProg
rams/cpp (master)
$ ./hello.exe
Hello World!
```

- executa primul commit folosind VCS

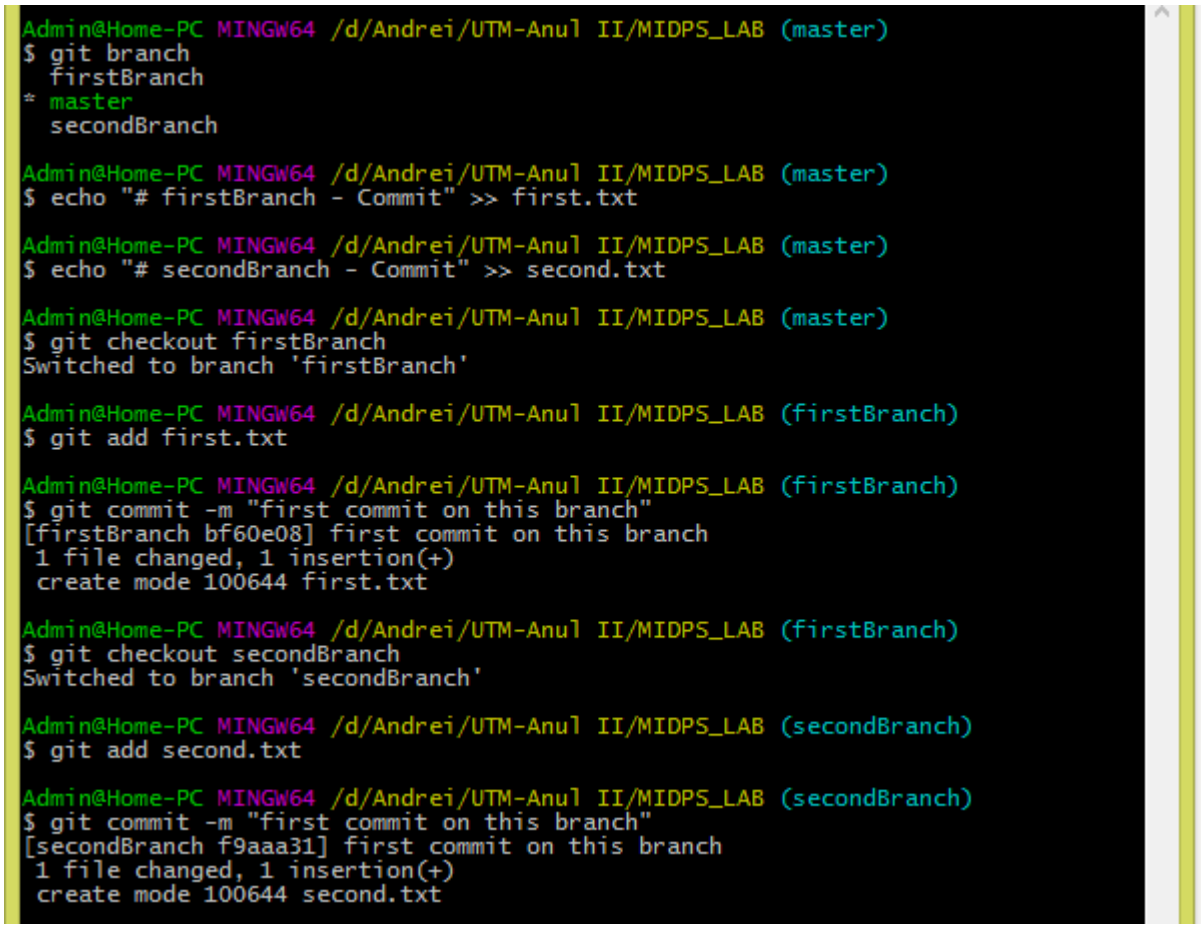
```
Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (master)
$ cd ./Lab-2

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2 (master)
$ echo "# Lab#2 - MIDPS" >> README.md

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2 (master)
$ git add README.md
```

```
Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB/Lab-2 (master)
$ git commit -m "lab-2 first commit"
[master cdf03ef] lab-2 first commit
 1 file changed, 2 insertions(+)
 create mode 100644 Lab-2/README.md
```

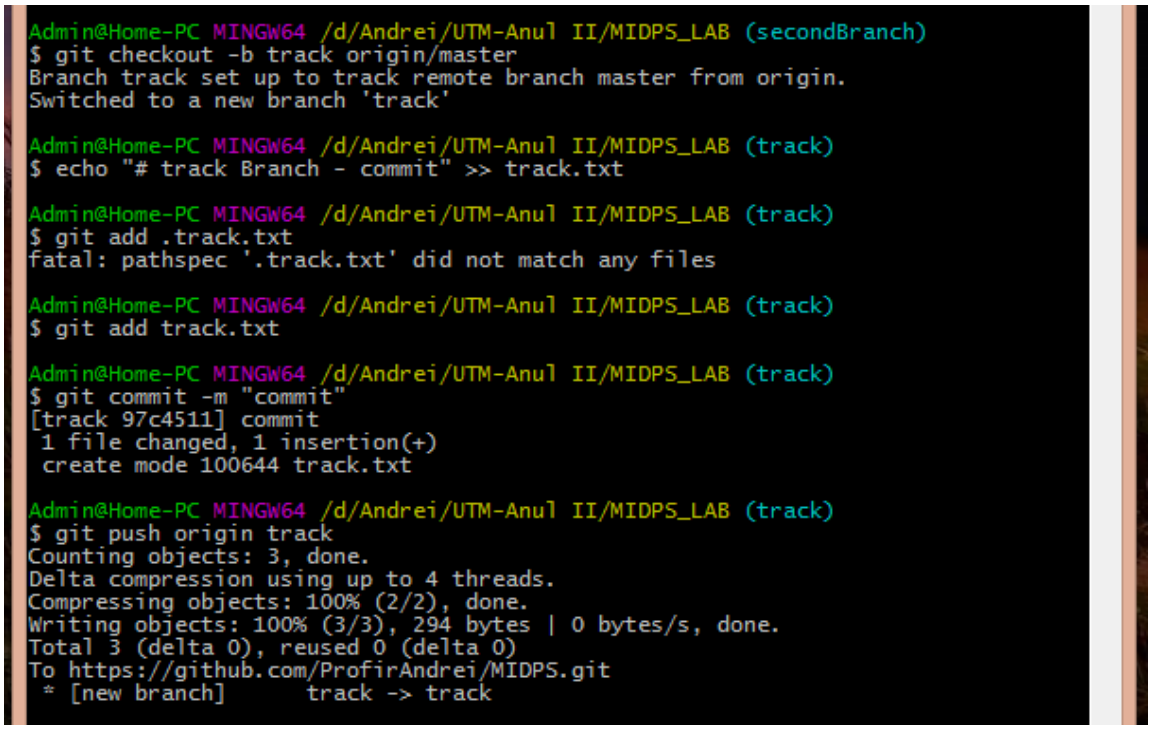
- *Normal Level* (nota 7 || 8):
    - initializeaza un nou repositoriu
    - configureaza-ti VCS



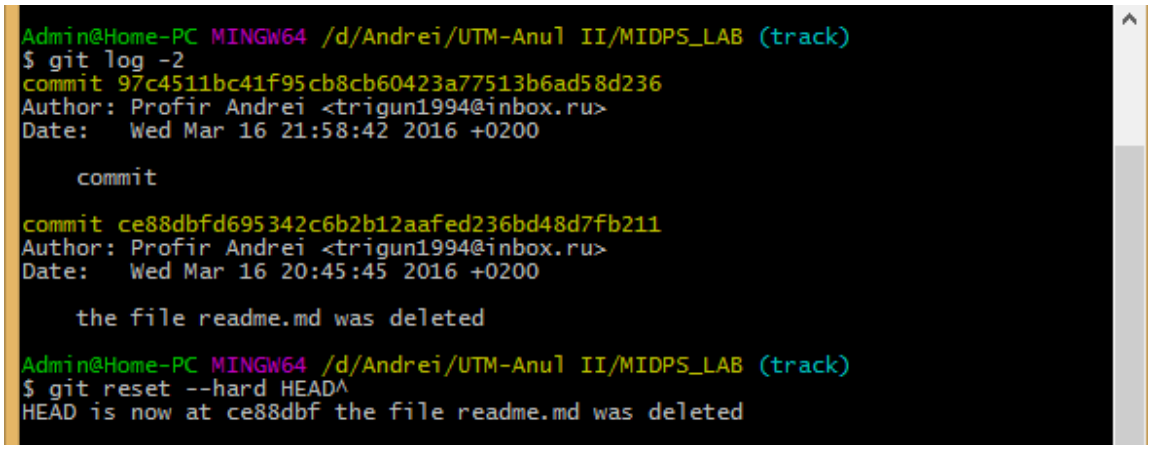
    - crearea branch-urilor (creeaza cel putin 2 branches)
    - commit pe ambele branch-uri (cel putin 1 commit per branch)



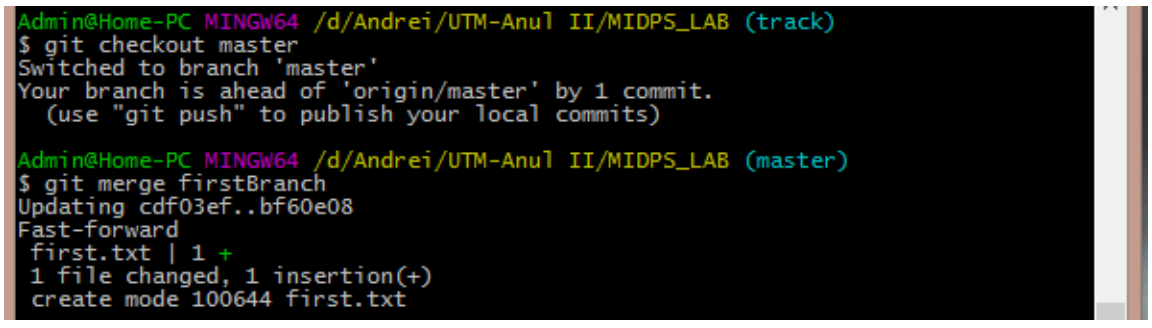
- *Advanced Level* (grade 9 || 10):
    - seteaza un branch to track a remote origin pe care vei putea sa faci push (ex. Github, Bitbucket or custom server)



    - reseteaza un branch la commit-ul anterior



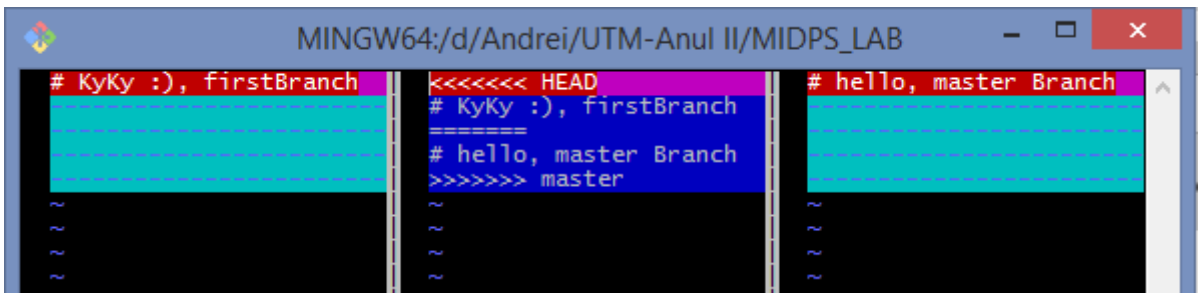
    - merge 2 branches



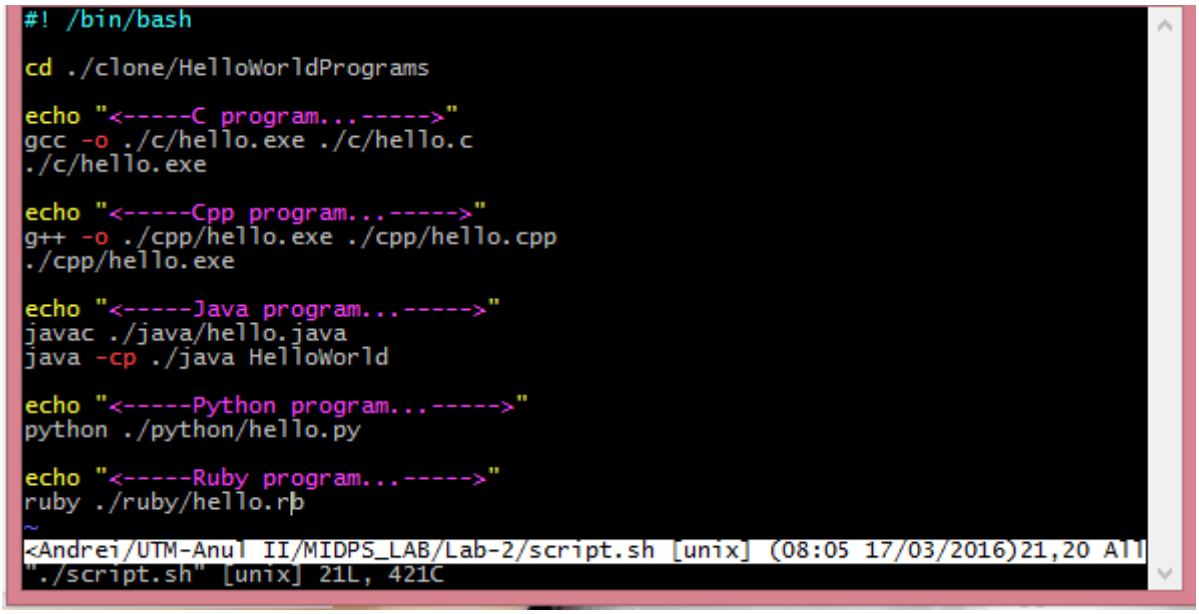
```
Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (master)
$ git merge secondBranch
Merge made by the 'recursive' strategy.
 second.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 second.txt
```

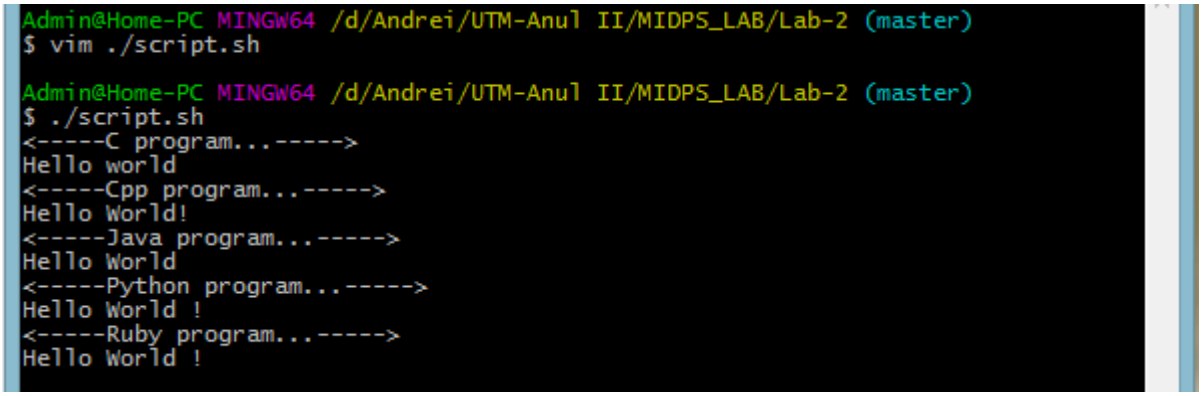
- conflict solving between 2 branches

```
Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (master)
$ echo "# hello, master Branch" >> hi.txt

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (master)
$ git add hi.txt

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (master)
$ git commit -m "added hi.txt"
[master 37759e4] added hi.txt
 1 file changed, 1 insertion(+)
 create mode 100644 hi.txt

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (master)
$ git checkout firstBranch
Switched to branch 'firstBranch'

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (firstBranch)
$ echo "# KyKy :), firstBranch" >> hi.txt

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (firstBranch)
$ git add hi.txt

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (firstBranch)
$ git commit -m "added hi.txt on the firstBranch"
[firstBranch 202e923] added hi.txt on the firstBranch
 1 file changed, 1 insertion(+)
 create mode 100644 hi.txt

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (firstBranch)
$ git merge master
Auto-merging hi.txt
CONFLICT (add/add): Merge conflict in hi.txt
Automatic merge failed; fix conflicts and then commit the result.

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (firstBranch|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecme
rge p4merge araxis bc codecompare emerge vimdiff
Merging:
hi.txt

Normal merge conflict for 'hi.txt':
  {local}: created file
  {remote}: created file
Hit return to start merge resolution tool (vimdiff):
3 files to edit

Admin@Home-PC MINGW64 /d/Andrei/UTM-Anul II/MIDPS_LAB (firstBranch|MERGING)
$ git commit -m "solved merge conflict"
[firstBranch f59f443] solved merge conflict
```



- *Bonus Point*:
    - Scrie un script care va compila HelloWolrdPrograms projects.





Link-ul catre repozitoriu online:

https://github.com/ProfirAndrei/MIDPS

Concluzie:

In urma efectuarii acestei lucrari de laborator au fost capatate primele deprinderi de baza in lucrul cu un VCS. In aceasta lucrare de laborator a fost folosit GIT ca fiind unul dintre cele mai populare DVCS si unul dintre cele mai comode in lucru.

Am capatat deprinderi in lucrul cu repozitoriul atit online cit si local. M-am invatat sa initializez un repozitoriu sa-l clonez s.a. Pe linga aceste doua manipulari cu repozitoriul am folosit si alte posibilitati asa cum: crearea unui nou branch, am facut unirea a doua branch-uri, am invatat cum sa hotarim unele conflicte care pot aparea cind facem "merge" la branch-uri.

De asemenea am invatat cum sa compilez cod sursa in diferite limbaje atit direct din CLI cit si prin intermediul unui script creat anterior. Evident ca prin intermediul scriptului este mult mai usor din cauza ca scriptul o data creat nu necesita o multime de comenzi pentru a face ceea ce deha a fost facut o data.