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

Efectuat : TI-151 **Poseletchi Cristian**

Verificat : lector asistent **Cojanu Irina**

2017

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).

Intelegerea si aplicarea comenzilor GIT.

3. Mersul lucrarii de laborator

3.1 Cerintele :

* Initializare unui nou repositoriu.

* Configurarea VCS.

* Crearea branch-urilor si commit pe ambele branch-uri

* Resetarea branch-urilor la commit-urile anterioare

* Merge la 2 branchuri.

* Folosirea fisierului .gitignore..

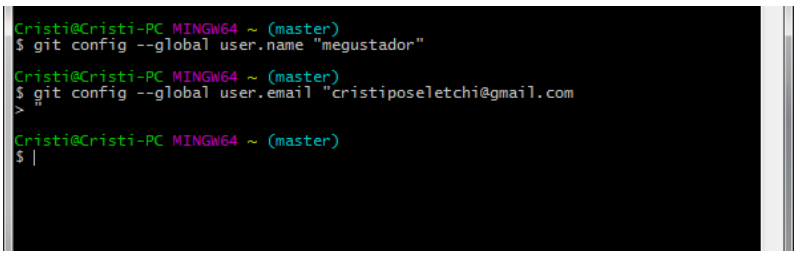
* Rezolvarea conflictelor.

3.2 Analiza lucrarii de laborator :

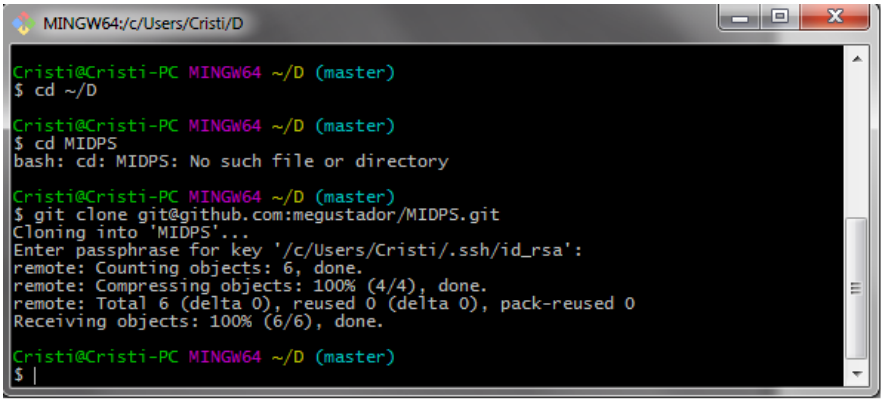
Linkul repositoriului **https://github.com/megustador/MIDPS**

Configurarea gitului consta in mai multe etape. La inceput vom configura numele si emailul prin intermediul urmatoarelor comenzi :

git config --global user.name “Numele”
git config --global user.email “Email”

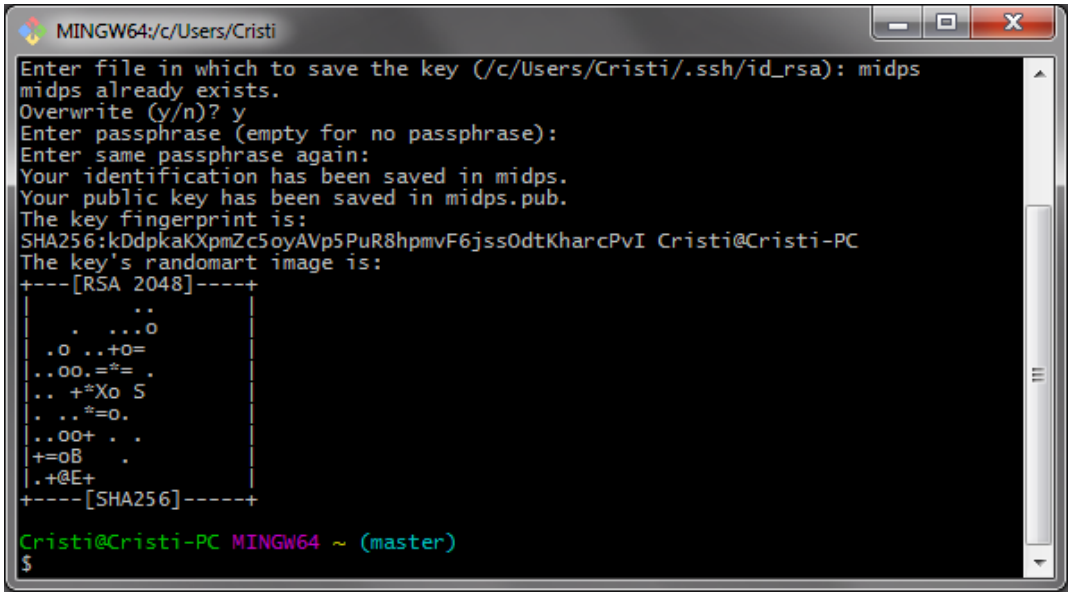


Exista mai multe metode de a crea un repozitoriu. Eu am creat repozitoriul direct pe github apoi cu **comanda git clone repo_url** si **SSH adresa** la repo mi-am creat o copie a repozitoriului pe local. Se putea de facut acest lucru si cu comanda **git init**

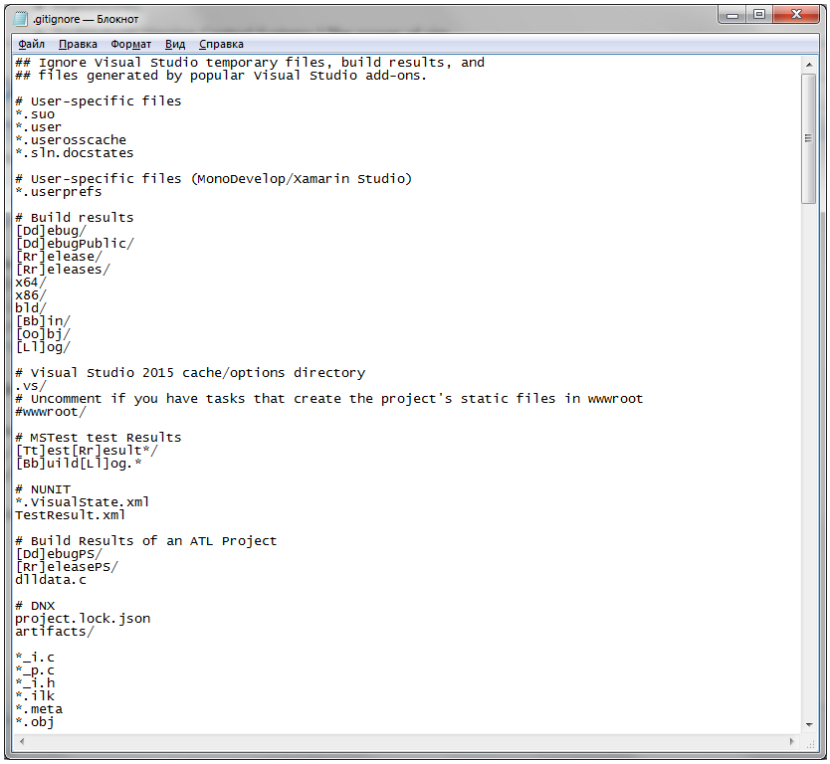


Urmatorul pas consta in generarea **SSH** key**.** Scriem **ssh-keygen,** iar cheia (publica) obtinuta o copiem in setarile noastre de pe github.com.

Cum e mentinut si in conditiile laboratorului, este de dorit sa initializam repozitorul nostru cu un fisier **README.md** si un **.gitignore.** In fisierul README.md vom adauga informatii pentru cei care se vor folosi de repozitoriu iar in fisierul .gitignore vom adauga toate fisierele ce trebuiesc ignorate (adica sa nu fie incarcate la moment).



Fisierul gitignore



.gitignore — Блокнот

Файл Правка Формат Вид Справка

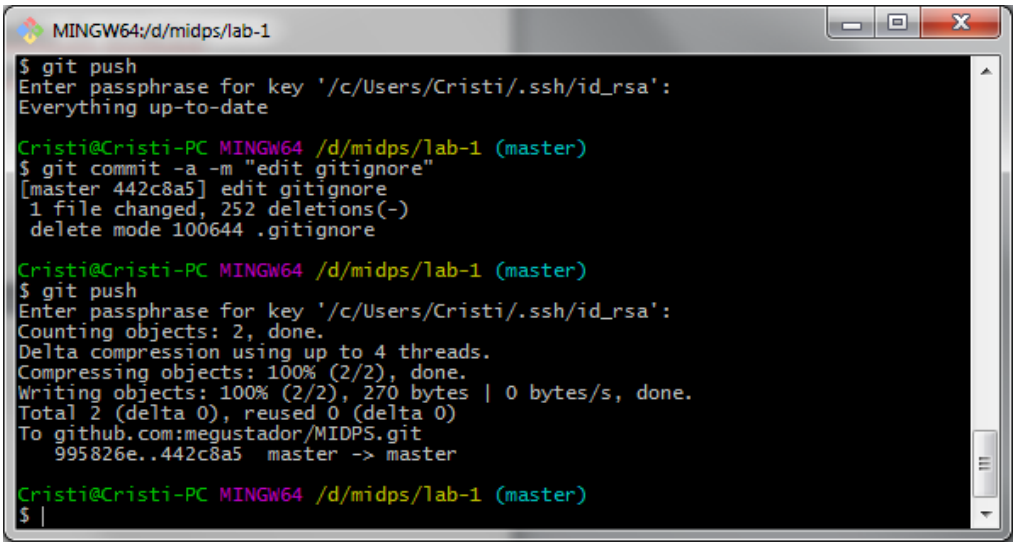
```
## Ignore Visual Studio temporary files, build results, and
## files generated by popular Visual Studio add-ons.

# User-specific files
*.suo
*.user
*.usersscache
*.sln.docstates

# User-specific files (MonoDevelop/Xamarin Studio)
*.userprefs

# Build results
[Dd]ebug/
[Dd]ebugPublic/
[Rr]elease/
[Rr]eleases/
x64/
x86/
bld/
[Bb]in/
[Oo]bj/
[Ll]og/

# Visual Studio 2015 cache/options directory
.vs/
# Uncomment if you have tasks that create the project's static files in wwwroot
#wwwroot/

# MSTest test Results
[Tt]est[Rr]esult*/
[Bb]uild[Ll]og.*

# NUNIT
*.VisualState.xml
TestResult.xml

# Build Results of an ATL Project
[Dd]ebugPS/
[Rr]eleasePS/
dlldata.c

# DNX
project.lock.json
artifacts/

*_i.c
*_p.c
*_i.h
*.ilk
*.meta
*.obj
```

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**



MINGW64:/d/midps/lab-1

```
$ git push
Enter passphrase for key '/c/Users/Cristi/.ssh/id_rsa':
Everything up-to-date

Cristi@Cristi-PC MINGW64 /d/midps/lab-1 (master)
$ git commit -a -m "edit gitignore"
[master 442c8a5] edit gitignore
 1 file changed, 252 deletions(-)
 delete mode 100644 .gitignore

Cristi@Cristi-PC MINGW64 /d/midps/lab-1 (master)
$ git push
Enter passphrase for key '/c/Users/Cristi/.ssh/id_rsa':
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 270 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
To github.com:megustador/MIDPS.git
   995826e..442c8a5  master -> master

Cristi@Cristi-PC MINGW64 /d/midps/lab-1 (master)
$ |
```

Pentru a ne asigura ca am facut totul bine si nu avem probleme utilizam urmatoarele comenzi git:
***git status**
***git show**

```
MINGW64:/D/midps

Cristi@Cristi-PC MINGW64 /D/midps (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   Lab-1/file.txt

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        deleted:    Lab-1/Raport1.pdf


Cristi@Cristi-PC MINGW64 /D/midps (master)
$ git show
commit 995826e8b4083f2750570d126a23c9ada7d2249a
Author: megustador <cristiposeletchi@gmail.com>
Date:   Sun Feb 19 20:12:30 2017 +0200

    Deleting file
```

VCS ne permite sa avem mai multe **branchuri.** Din ENG branch semnifica “creanga”. Branch-urile sunt utilizate cind lucram paralel la un proiect si apoi dorim sa combinam 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.

```
MINGW64:/D/MIDPS/lab-1

Cristi@Cristi-PC MINGW64 /D/MIDPS/lab-1 (master)
$ git checkout -b MySecondBranch
fatal: A branch named 'MySecondBranch' already exists.

Cristi@Cristi-PC MINGW64 /D/MIDPS/lab-1 (master)
$ git checkout MySecondBranch
D       Lab-1/Raport1.pdf
A       Lab-1/file.txt
Switched to branch 'MySecondBranch'

Cristi@Cristi-PC MINGW64 /D/MIDPS/lab-1 (MySecondBranch)
$ git push origin MySecondBranch
Enter passphrase for key '/c/Users/Cristi/.ssh/id_rsa':
Total 0 (delta 0), reused 0 (delta 0)
To github.com:megustador/MIDPS.git
 * [new branch]      MySecondBranch -> MySecondBranch

Cristi@Cristi-PC MINGW64 /D/MIDPS/lab-1 (MySecondBranch)
$ git branch
* MySecondBranch
  master

Cristi@Cristi-PC MINGW64 /D/MIDPS/lab-1 (MySecondBranch)
$
```

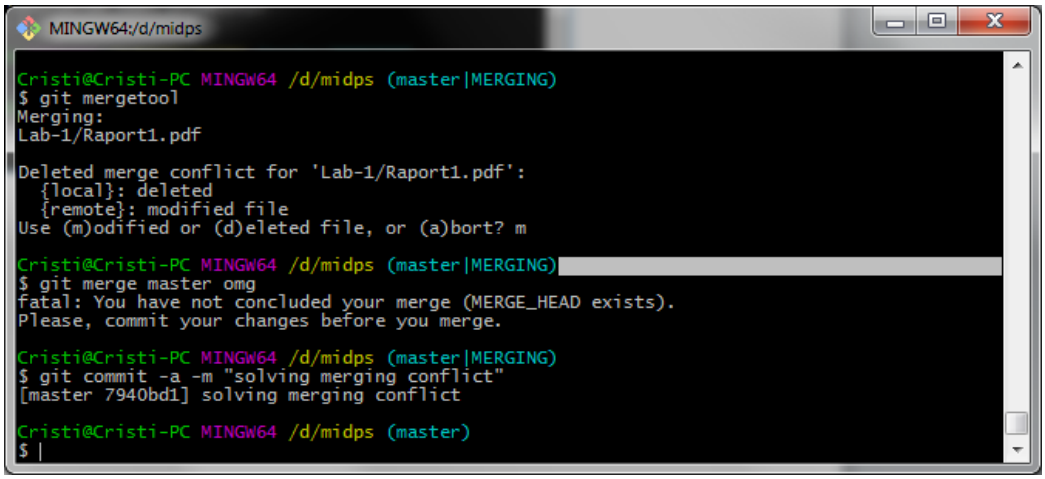
git checkout "name" – face switch la branchul "name".
git branch –u upstream/name – face track la branchul indicat din branchul curent.
git branch –u upstream/name "name" – face track din branchul "name" la branchul indicat.
git branch –track "name" upstream/name – creeaza branchul "name" si ii face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

```
MINGW64:/D/midps
Cristi@Cristi-PC MINGW64 /D/midps (master)
$ git branch
  MySecondBranch
* master
  omg

Cristi@Cristi-PC MINGW64 /D/midps (master)
$ git checkout MySecondBranch
D       Lab-1/Raport1.pdf
A       Lab-1/file.txt
Switched to branch 'MySecondBranch'

Cristi@Cristi-PC MINGW64 /D/midps (MySecondBranch)
$ git checkout omg
D       Lab-1/Raport1.pdf
A       Lab-1/file.txt
Switched to branch 'omg'

Cristi@Cristi-PC MINGW64 /D/midps (omg)
$
```

In caz ca dorim sa schimbat istoria unui commit, sau sa **resetam un branch la commitul anterior**. Pentru asta putem folosi comanda **git reset commit_index.** Pentru a demonstra asta am ales branchul "MySecondBranch" de pe repo-ul meu si l-am resetat la ultimul commit facut.

```
MINGW64:/d/midps/lab-1
Enter passphrase for key '/c/Users/Cristi/.ssh/id_rsa':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 319 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To github.com:megustador/MIDPS.git
   442c8a5..c02b1d7  master -> master

Cristi@Cristi-PC MINGW64 /d/midps/lab-1 (master)
$ git checkout MySecondBranch
Switched to branch 'MySecondBranch'

Cristi@Cristi-PC MINGW64 /d/midps/lab-1 (MySecondBranch)
$ ls
Raport1.pdf  readme

Cristi@Cristi-PC MINGW64 /d/midps/lab-1 (MySecondBranch)
$ git reset --hard HEAD
HEAD is now at 4f3259c Edit the readme

Cristi@Cristi-PC MINGW64 /d/midps/lab-1 (MySecondBranch)
$
```

Pot aparea conflicte in cazul cind dorim sa facem **merge** la 2 branch-uri si unele rinduri sunt diferite. In asa caz,pentru a elimina conflictele, folosim **mergetool**. Drept mergetool am ales **kdiff3.** Pentru kdiff3, in mod implicit folosim comanda : **git config –global merge.tool kdiff3.**



4. Concluzie

In aceasta lucrare de laborator am pus in practica si am insusit cele mai importante functii ale Git-ului. Am stiut de GitHub pina acum, insa nu am stiut ca el atitea posibilitati. VCS ne face viata de x99 ori mai usora cind lucram asupra unui produs soft. Cel mai mult ma impresionat posibilitatea de a face **branchuri**, asta chiar este ceva extreme de necesar cind asupa unui produs lucreaza mai multi oameni simultan. In urma acestei lucrari am invatat crearea si controlarea unui repozitoriu si a fisierelor din interior. La fel un avantaj este **commitul** si posibilitatea de a vedea cind si ce schimbari au fost facute. GitHub-ul este un must-learn pentru orice developer in devenire!

Referinte :

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

2. https://www.siteground.com/tutorials/git/commands.htm

3. https://www.atlassian.com/git/tutorials/

4. https://git-scm.com/book/en/v2/Git-Branching-Basic

5. https://learn.sparkfun.com/tutorials/using-github