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:
Octavian Coroletchi

lector asistent:
Victor Gojin

Chisinau 2017

Laboratory work #1

1 Scopul lucrarii de laborator

Initializarea in Version Control Systems.

2 Obiective

Version Control Systems (git)

3 Laboratory work implementation

3.1 Tasks and Points

a) Initializarea unui nou repositoriu

b) Configurarea VCS

c) Crearea branch-urilor (cel putin 2)

d) Commit pe ambele branch-uri(cel putin 1 commit per branch)

e) Setarea unui branch to track a remote origin pe care sa se faca push(GitHub)

f) Resetarea unui branch la commit-ul anterior

g) Salvarea temporara a schimbarilor care nu se vor face commit imediat

h) Folosirea fisierului .gitignore

i) Merge 2 branches

j) Rezolvarea conflictelor a 2 branches

k) Folosirea tag-urilor pentru marcarile semnificative

3.2 Analiza lucrarii de laborator

Click on "Link" or copy `https://github.com/OctavianCoroletchiTI154/MIDPS.git` pentru repozitoriul meu.

Task a:

Pentru inceput, mi-am facut cont pe github.com. Mi-am creat repozitoriul meu(vezi imaginea a) public. Apoi am facut download la clientul Git.

Task b:

Mi-am configurat repozitoriul meu, generindu-mi o cheie publica cu ajutorul comenzii $ssh-keygen-trsa-b4096-C"youremail@example.com"$. Apoi mi-am adaugat cheia mea publica la contul meu.Dupa mi-am clonat repozitoriul meu pe masina mea locala cu ajutorul comenzii "$gitclone'repozitoriuln$

Task c:

Am creat branch-uri cu ajutorul comenzii "$gitbranch < noulmeubranch >$ ". (vezi imaginea c). Apoi i-am dat push pe github, cu comanda git push origin lab1.(vezi imaginea c-1).

Task d:

Am facut commit pe branchurile create cu ajutorul comenzii $gitcommit - m"name"$. (vezi imaginea d).

Task e:

Am setat branch-ul lab1, pentru a urmari remote origin pe care sa facem push. Am folosit comanda $gitbranch - uorigin/lab1$ (vezi imaginea e)

Task f:

Am eliminat branch-ul lab1 din origin cu comanda $gitpushorigin : lab1$. Apoi am readus lab1 la comitul care avem nevoie. $gitreset--hard'hashid'$. Apoi am facut push din nou. $gitpushoriginlab1$. (pentru toate vezi imaginea f)

Task g:

Am salvat schimbarile temporare cu ajutorul Stash-ului. Aceasta permite salvarea temporara a unor modificari pentru care nu avem nevoie sa facem commit. Se foloseste comanda *gitstash*. Dupa ce se salveaza comenzile modificate, ne putem reintoarce in orice moment spre salvari cu comanda $gitstashapplystash@0$, sau in loc de stash@0 cu id-ul salvat. Pentru a vedea intreaga lista de stash-uri, folosim *gitstashlist*. (vezi imaginea g).

Task h:

.gitignore, este un fisier, in care specificam ce tipuri ce file-uri nu dorim sa fie prezente in repozitoriul nostru Git. In imaginea raspunzatoare de task-ul h, se va observa fisierul nostr .gitignore si tipurile de fisiere ce vor fi ignorate. (vezi imaginea h).

Task i:

Ajungind la un moment, cind modificarile pentru un anumit branch ajung la un statut final, atunci putem sa imbinam branch-ul aditional cu cel principal. Pentru inceput, ne schimbam pe branch-ul principal cu comanda *gitcheckoutmaster* , apoi le imbinam cu comanda $gitmergelab1$ unde master si lab1, sunt branch-uri. Apoi putem face delete pentru branch-ul adition lab1, pentru ca nu mai avem nevoie de el cu comanda $gitbranch - dlab1$. (vezi imaginea i pentru implementari).

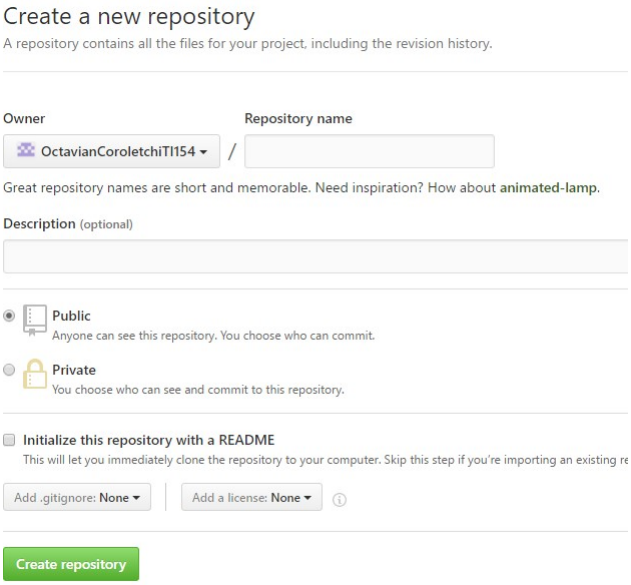
Task j:

Conflictele apar atunci cind incerci sa faci merge la 2 branchu-uri, in care in ambele au loc modificari in acelasi fisiel. Cum ar fi situatia intilnita de mine, in care 2 branch-uri, aveau modificari diferite in README.md. Conflictul intilnit (vezi imaginea j). Apoi trebuia sa decid ce schimbari doresc sa pastrez pentru a continua peste conflict. (vezi j-1). Dupa aia, totul a continuat cu succes.

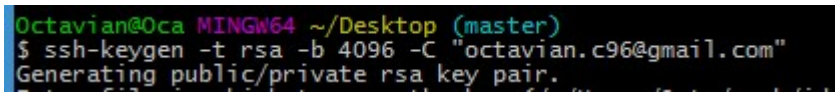
Task k:

Tagurile sunt folosite pentru a marca anumite evenimente importante din modificari.Ele se implementeaza cu ajutorul comenzii $gittag - a'name' - m"description"$. Dupa se da push pentru a vedea si altii tagurile noastre. (vezi imaginea k, pentru implementare).

3.3 Imagini

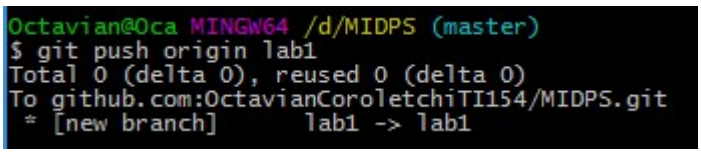


Imaginea - "a"



Imaginea - "b"

Imaginea - "c"



Imaginea - "c-1"

```
Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git push --set-upstream origin lab1
Counting objects: 10, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (10/10), done.
Writing objects: 100% (10/10), 82.30 KiB | 0 bytes/s, done.
Total 10 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
Branch lab1 set up to track remote branch lab1 from origin.
To github.com:OctavianCoroletchiTI154/MIDPS.git
   2b4c970..e095f8c  lab1 -> lab1
```

Imaginea - ”d”

```
Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git branch -u origin/lab1
Branch lab1 set up to track remote branch lab1 from origin.
```

Imaginea - ”e”

```
Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git push origin :lab1
To github.com:OctavianCoroletchiTI154/MIDPS.git
 - [deleted]         lab1

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git reset --hard dba4b64f39504959b90f9e3dd54a5a9899738e26
HEAD is now at dba4b64 Task f

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git push origin lab1
Counting objects: 17, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (17/17), done.
Writing objects: 100% (17/17), 123.13 KiB | 0 bytes/s, done.
Total 17 (delta 4), reused 0 (delta 0)
remote: Resolving deltas: 100% (4/4), completed with 1 local objects.
To github.com:OctavianCoroletchiTI154/MIDPS.git
 * [new branch]      lab1 -> lab1
```

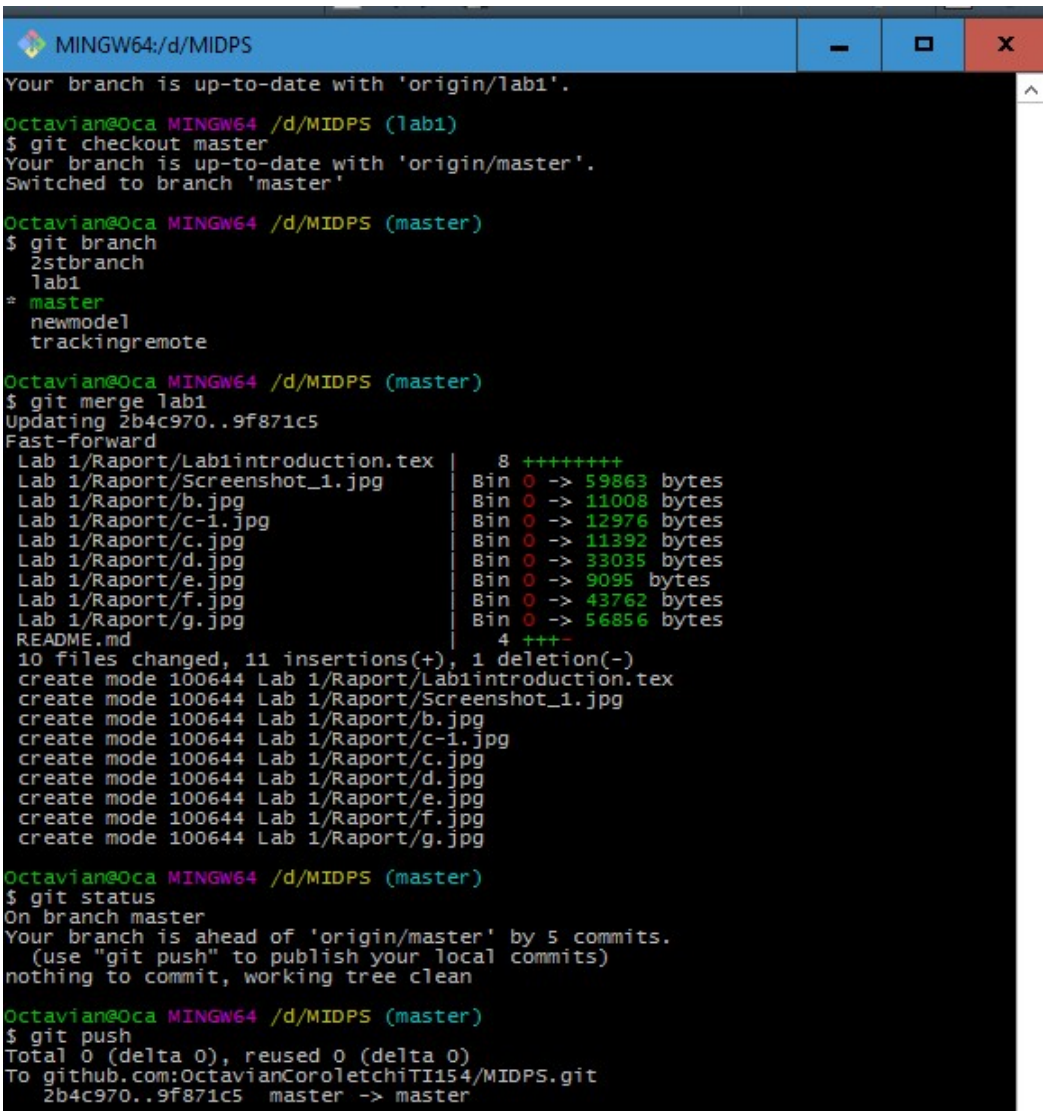
Imaginea - ”f”

```
MINGW64:/d/MIDPS

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git stash

Saved working directory and index state WIP on lab1: 2b6b5d8 new mod
HEAD is now at 2b6b5d8 new mod

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git stash list
stash@{0}: WIP on lab1: 2b6b5d8 new mod
stash@{1}: WIP on lab1: dba4b64 Task f
stash@{2}: WIP on lab1: dba4b64 Task f

Octavian@Oca MINGW64 /d/MIDPS (lab1)
$ git stash apply stash@{0}
On branch lab1
Your branch is up-to-date with 'origin/lab1'.
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

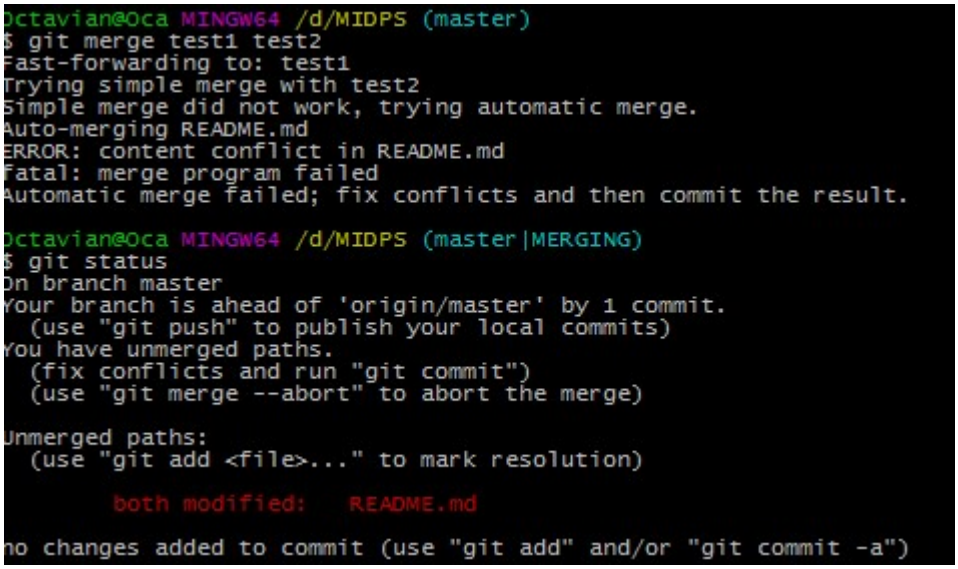
Imaginea - ”g”

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

# User-specific files
*.suo
*.user
*.userosscache
*.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
```

Imaginea - ”h”



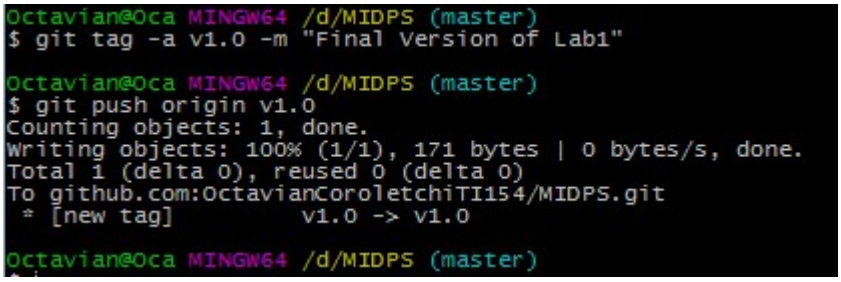
Imaginea - ”i”



Imaginea - ”j”

```
# MIDPS
# Hello

##2nd Branch Modify

##New Commit v0.1
##Added new branch
Adaugarea comiturilor pentru noul branch.
Trying to return a commit;
Before stash 1

<<<<<<< .merge_file_a09932
Aditional modification for conflict test.  1
1
=======
Aditional modification for conflict test.2
2
2
>>>>>>> .merge_file_a08484

```

Imaginea - ”j-1”



Imaginea - ”k”

Concluzie

In concluzie, pot spune caci version control system este o utilitate foarte importanta in urmarirea dezvoltarii aplicatiilor. Aceasta permite dezvoltarea unei aplicatii prin diverse cai, de diferiti developeri in acelasi timp. Version control protejeaza codul sursa de o eroare umana care poate duce la consecinte foarte mari. Dezvoltatorii de aplicatii care lucreaza in echipe, mereu schimba codul sursa, si aceasta necesita intr-o organizare bine pusa la punct. Principala problema este cea a conflictelor, ce permite o rezolvare adecvata a acestei probleme.

References

1 http://www.manniwood.com/starting_a_project_with_git.html

2 http://www.vogella.com/tutorials/Git/article.html

3 https://git-scm.com/