

***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 lucrării de laborator :***

De a se învăța utilizarea unui Version Control System și modul de setare a unui server.

- ***Obiective***

Studierea Version Control Systems (git).

- ***Mersul lucrării de laborator***

### ***3.1 Cerințele :***

- \* Initializare unui nou repository.
- \* Configurarea VCS.
- \* Commit, Push branch.
- \* Folosirea fișierului .gitignore.
- \* Revenire la versiunile anterioare.
- \* Crearea branch-urilor noi.
- \* Commit pe ambele branch-uri.
- \* Merge la 2 branchuri.
- \* Rezolvarea conflictelor.

- ***Analiza lucrării de laborator :***

Linkul la repository <https://github.com/NegruIgor/MIDPS>

Sunt mai multe modalități de a inițializa un repository pe github. Putem crea o mapă goală în care vom plasa gitul nostru prin intermediul comenzii **git init**.

Următorul pas este crearea a noului repository pe care îl vom crea utilizând următoarea comandă **curl – u ‘USER’ https://api.github.com/user/repos/ -d ‘{“name”:”NUME”}’**. Unde cuvintele scrise cu CAPS se vor înlocui cu numele utilizatorului și numele repository-ului. După aceasta este necesar să adăugăm gitul nostru gol cu repository-ul creat. Vom folosi următoarea comandă **git remote add origin “Linkul la repository-ul nostru”**

```
MINGW64:/d/MIDPS

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS
$ git init
Initialized empty Git repository in D:/MIDPS/.git/

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ git config --global user.name "NegruIgor"

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ git config --global user.email "potato518403@gmail.com"
```

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"
```

```
MINGW64:/d/MIDPS

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS
$ git init
Initialized empty Git repository in D:/MIDPS/.git/

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ git config --global user.name "NegruIgor"

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ git config --global user.email "potato518403@gmail.com"

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ git remote add origin https://github.com/NegruIgor/MIDPS.git

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ git config --list
core.symlinks=false
core.autocrlf=true
core.fscache=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
user.name=NegruIgor
user.email=potato518403@gmail.com
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=https://github.com/NegruIgor/MIDPS.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
```

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 ).

```
MINGW64:/d/MIDPS

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ vim README.md

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ vim .gitignore

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ cat README.md
Hello World

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS (master)
$ cat .gitignore
ignore.txt

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

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/MIDPS

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

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git commit -m "Incepem"
[master b923106] Incepem
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Lab1/Lab1.doc

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git push origin master
Counting objects: 4, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 3.34 KiB | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To github.com:NegruIgor/MIDPS.git
70ed21a..b923106 master -> master

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

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

**\*git status**

**\*git show**

```
MINGW64:/d/MIDPS/MIDPS

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git show
commit 9b5a9fa8fef7b2789c704ac92d5d1e8f7363eb66
Author: NegruIgor <potato518403@gmail.com>
Date: Thu Feb 23 16:29:21 2017 +0200

    continuum

diff --git a/Lab1/Lab1.doc b/Lab1/Lab1.doc
deleted file mode 100644
index 8771eb1..0000000
--- a/Lab1/Lab1.doc
+++ /dev/null
@@ -1,0 @@
```

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.

```
MINGW64:/d/MIDPS/MIDPS

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

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

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git branch -d new
Deleted branch new (was 9b5a9fa).

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

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

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ ls
Lab1/  Lab2/  Lab3/  Lab4/  Lab5/  README.md
```

**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 “to\_merge” 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 urmat.

```

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ ls
Lab1/ Lab2/ Lab3/ Lab4/ Lab5/ merge README.md

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git checkout nou
Your branch is behind 'origin/master' by 5 commits, and can be fast-forwarded.
(use "git pull" to update your local branch)
Switched to branch 'nou'

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

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ ls
Lab1/ Lab2/ Lab3/ Lab4/ Lab5/ merge README.md

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git add *
warning: LF will be replaced by CRLF in merge.
The file will have its original line endings in your working directory.

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git commit -m "hello"
[nou f34f082] hello
1 file changed, 2 insertions(+)
create mode 100644 merge

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ git push origin master
Everything up-to-date

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (nou)
$ ls
Lab1/ Lab2/ Lab3/ Lab4/ Lab5/ merge README.md

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)
$ ls
Lab1/ Lab2/ Lab3/ Lab4/ Lab5/ merge README.md

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master)
$ git merge nou
Auto-merging merge
CONFLICT (add/add): Merge conflict in merge
Automatic merge failed; fix conflicts and then commit the result.

Potato@WINCTRL-3259PNU MINGW64 /d/MIDPS/MIDPS (master|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:

```

## • **Concluzie**

In lucrarea nr.1 la MIDPS am studiat lucrul cu VCS. Am cunoscut platforma **github**. Toate lucrrurile, 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](https://github.com/BestMujik/MIDPS-labs/blob/master/MIDPS_LAB%231.md)

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