

FACULTATEA CALCULATOARE, INFORMATICĂ ȘI MICROELECTRONICĂ
UNIVERSITATEA TEHNICĂ A MOLDOVEI

MEDII INTERACTIVE DE DEZVOLTARE A PRODUSELOR SOFT
LUCRAREA DE LABORATOR#1

Version Control Systems și modul de setare a unui server

Autor:

Marina JECHIU

lector asistent:

Irina COJANU

Laboratory work #1

1 Scopul lucrării de laborator

Însusirea noțiunii de Version Control Systems și a modului de setare a unui server.

2 Obiective

Version Control Systems (git || *bitbucket* || *mercurial* || *svn*)

3 Condiția sarcinii

- inițializează un nou repository;
- configurează-ți VCS;
- crearea branch-urilor (cel puțin 2);
- commit pe ambele branch-uri;
- setează un branch to track a remote origin pe care vei putea să faci push ;
- salvarea temporară a schimbărilor care nu se vor face commit imediat;
- resetează un branch la commit-ul anterior;
- folosirea fișierului .gitignore;
- merge 2 branches;
- rezolvarea conflictelor a 2 branches.

4 Laboratory work implementation

4.1 Analiza lucrării de laborator

<https://github.com/MarinaJechiuTI154/MIDPS> - linkul repoziatoriului.

Principalele noțiuni cu care voi opera:

- **repository** componenta server ce conține informații privind ierarhia de fișiere și reviziile.
- **branch** este o ramură secundară de dezvoltare a unui proiect.
- **checkout** preluarea în mediul local a unei anumite revizii publicate pe server.
- **commit** cerere de publicare pe server a unor modificări.
- **pull** acțiunea de actualizare (update) a informațiilor locale cu cele de pe server.
- **conflict** apare atunci când mai mulți utilizatori vor să publice modificări aplicate acelorași fișiere din proiect, însă sistemul de aplicare a versiunilor diferite nu poate îmbina modificările.
- **revert** revenirea la o versiune anterioară pe un anumit fir de dezvoltare (branch).
- **tag** branch “read-only” ce nu mai permite modificări ulterioare (folosit uneori pentru versiunile stabile și derivă dintr-un branch)

Am creat un cont public pe github cu denumirea mdps. Pentru a putea gestiona repoziatoriul am instalat GitBash. Pentru a activa contul avem nevoie să introducem în setări cheia, care se obține prin tastarea în linia de comandă: `ssh-keygen`. Pentru a deschide cheia obținută folosim următoarele instrucțiuni: `cat ~/.ssh/id - rsa.pub`. Pentru a face legătura între repoziatoriul pe github și cel local este necesar să clonăm repoziatoriul online cu instrucțiune: *git clone*.

La crearea repoziatoriului se creează un branch implicit numit master. Însă, dacă trebuie să creăm un nou branch folosim comanda: *git checkout 'denumire-branch'*. Astfel, nu doar se creează un nou branch, dar și se trece automat pe acest branch. În cazul în care dorim să trecem pe un branch deja existend, la tastarea denumirii acestuia sistemul îl recunoaște și face automat salt către acesta.

Deoarece repoziatoriile git sunt repozitorii de tip distrib, pentru a încărca modificările efectuate local pe server este necesar pentru a efectua următoarii pași:

- *git add* . adăugarea în index a modificărilor realizate în directoriul meu, fișiere ce se intenționează a fi publicate.

git commit efectuarea commit-urilor în baza informației din index. Acestea pot conține denumiri pentru a gestiona mai ușor modificările.

- *git push* publicarea modificărilor pe repoziatoriu.

Pentru a putea face push pe branch-urile create, este necesar ca acestea să fie setate to track a remote origin. Pentru asta vom folosi comanda: *git push -u origin den-branch*.

Ultimul commit efectuat în repoziatoriu poartă numele de HEAD. Astfel, pentru a reveni la un commit anterior este suficient să scriem în linia de comandă: *git reset --hard HEAD*.

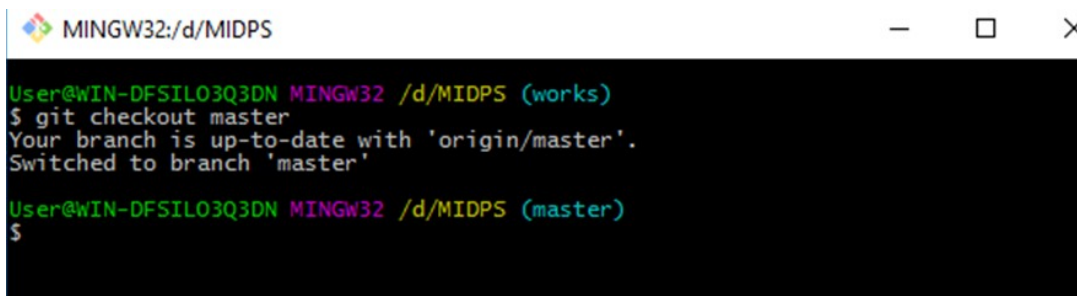
Uneori, vrem să facem anumite schimbări temporare, dar fără a face deodată commit asupra acestora. Pentru asta utilizăm comanda *git stash*. Putem totodată să revenim asupra acestor modificări mai târziu.

Nu întotdeauna dorim să versionăm anumite fișiere, întrucât acestea pot conține executabile generate de procesul de compilare sau efectiv parametrii de configurare ai aplicației. Fișierele sau directoarele ce se doresc a fi ignorate de sistemul de versionare pot fi trecute într-un fișier numit *.gitignore*. Acesta este evaluat în mod recursiv, putând astfel avea mai multe fișiere *.gitignore* în folderele proiectului nostru. Cu toate acestea, pentru a nu căuta foarte mult path-urile ignorate, se folosește în mod convențional un singur astfel de fișier plasat în radacina proiectului.

După crearea a două sau mai multe branch-uri acestea sunt dezvoltate separat, conținând informații diferite de obicei. Dacă dorim să unim aceste două branch-uri într-unul singur folosim comanda *git merge*. Aceasta se va executa numai dacă între aceste branch-uri nu există conflict, adică conținutul lor este identic. Pentru a rezolva aceste conflicte trebuie sincronizate cele două branch-uri, ambele să aibă informație identică.

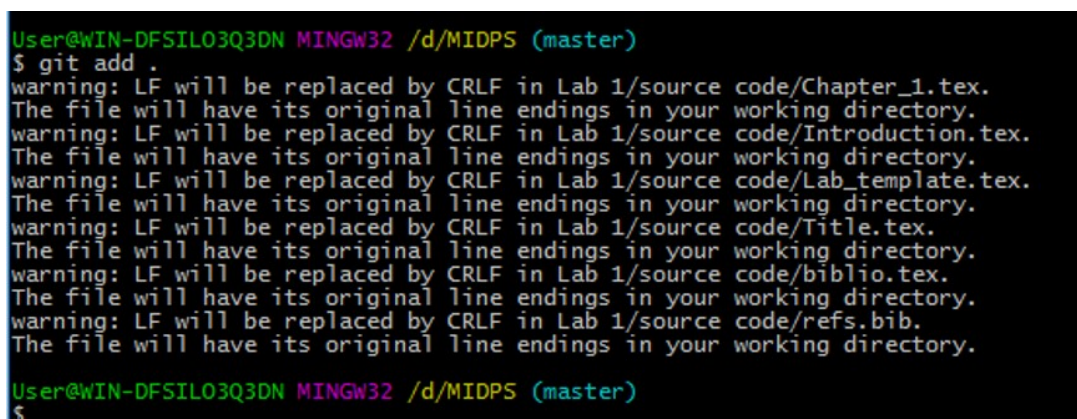
În istoria modificărilor pot fi marcate cele mai importante modificări cu ajutorul tag-urilor. Git-ul folosește două tipuri de tag-uri: simple și adnotate. Numele tag-urilor au următoarea formă: v0.1, v3.4 etc. Pentru a inițializa un tag adnotat introducem: *git tag -a v1.2 -m "nume-tag"*. Tag-urile adnotate sunt stocate ca obiecte complete în baza de date. Tag-ul ușor este de fapt un pointer către un anumit commit. Acesta se inițializează: *textitgit tag v1.2 -lw*.

4.2 Imagini

A screenshot of a terminal window titled "MINGW32:/d/MIDPS". The prompt is "User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (works)". The user enters the command "\$ git checkout master". The output is "Your branch is up-to-date with 'origin/master'. Switched to branch 'master'". The prompt then changes to "User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)".

```
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (works)
$ git checkout master
Your branch is up-to-date with 'origin/master'.
Switched to branch 'master'
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$
```

Figure 4.1 – Trecerea de pe un branch pe altul

A screenshot of a terminal window titled "MINGW32:/d/MIDPS". The prompt is "User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)". The user enters the command "\$ git add .". The output shows several warnings about line endings being replaced by CRLF in various files. The prompt then changes to "User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)".

```
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git add .
warning: LF will be replaced by CRLF in Lab 1/source code/Chapter_1.tex.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in Lab 1/source code/Introduction.tex.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in Lab 1/source code/Lab_template.tex.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in Lab 1/source code/Title.tex.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in Lab 1/source code/biblio.tex.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in Lab 1/source code/refs.bib.
The file will have its original line endings in your working directory.
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$
```

Figure 4.2 – Adăugarea modificărilor efectuate din directoriu în index

```
MINGW32:/d/MIDPS

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git add .

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git push
Counting objects: 18, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (16/16), done.
Writing objects: 100% (18/18), 68.42 KiB | 0 bytes/s, done.
Total 18 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To github.com:MarinaJechiuTI154/MIDPS.git
   ca2cab7..fadeddd  master -> master

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git commit -m 'titlu'
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

Figure 4.3– Comanda commit

```
MINGW32:/d/MIDPS

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git commit -m 'test'
[master ed3c913] test
Committer: Marina <Marina>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly:

    git config --global user.name "Your Name"
    git config --global user.email you@example.com

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

1 file changed, 2 deletions(-)
delete mode 100644 README.md

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git reset --hard HEAD
HEAD is now at ed3c913 test
```

Figure 4.4– Revenirea la ultimul commit

```
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git stash
Saved working directory and index state WIP on master: ed3c913 test
HEAD is now at ed3c913 test

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ |
```

Figure 4.5– Salvarea temporară a modificărilor

```

MINGW32:/d/MIDPS
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git checkout works
error: The following untracked working tree files would be overwritten by checkout:
    README.md
Please move or remove them before you switch branches.
Aborting
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$

```

Figure 4.6– Conflict

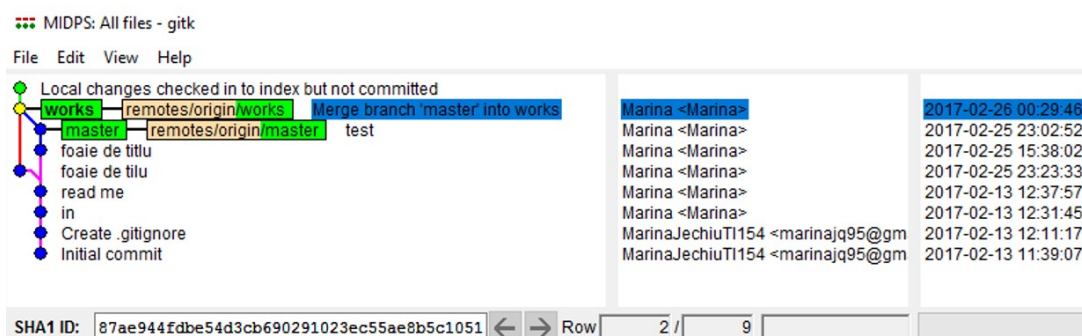


Figure 4.7– Imaginea grafică a repozitoriului

```

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS/Lab 1 (master)
$ git checkout 'test'
A      README.md
Switched to branch 'test'
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS/Lab 1 (test)
$

```

Figure 4.8– Crearea unui nou branch

```

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (test)
$ git push -u origin test
Total 0 (delta 0), reused 0 (delta 0)
Branch test set up to track remote branch test from origin.
To github.com:MarinaJechiuTI154/MIDPS.git
 * [new branch]      test -> test
User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (test)
$

```

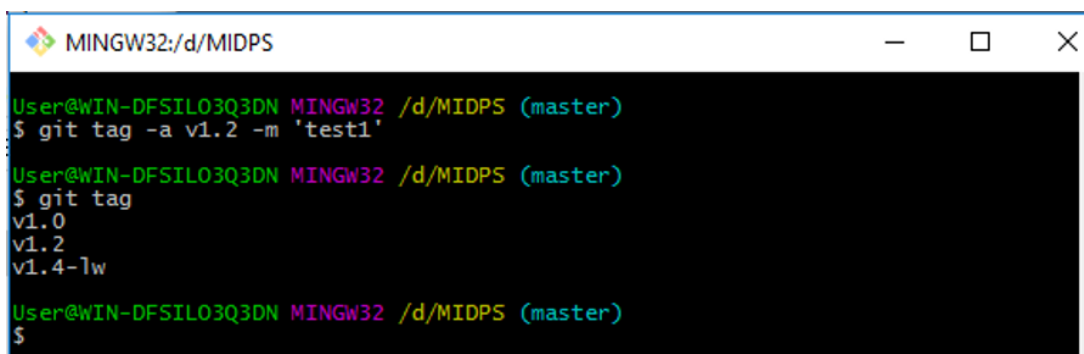
Figure 4.9– Set a branch to track a remote origin

```

User@WIN-DFSILO3Q3DN MINGW32 /d/MIDPS (master)
$ git clone https://github.com/MarinaJechiuTI154/MIDPS
Cloning into 'MIDPS'...
remote: Counting objects: 36, done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 36 (delta 5), reused 29 (delta 3), pack-reused 0
Unpacking objects: 100% (36/36), done.

```

Figure 4.10– Clonarea unui repozitoriu

A screenshot of a Windows terminal window titled "MINGW32:/d/MIDPS". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The terminal shows a series of commands and their outputs. The first command is "git tag -a v1.2 -m 'test1'", which is executed successfully. The second command is "git tag", which lists the existing tags: "v1.0", "v1.2", and "v1.4-lw". The prompt returns to the shell.

```
MINGW32:/d/MIDPS
User@WIN-DFSIL03Q3DN MINGW32 /d/MIDPS (master)
$ git tag -a v1.2 -m 'test1'
User@WIN-DFSIL03Q3DN MINGW32 /d/MIDPS (master)
$ git tag
v1.0
v1.2
v1.4-lw
User@WIN-DFSIL03Q3DN MINGW32 /d/MIDPS (master)
$
```

Figure 4.11 – Implementarea tag-urilor

Concluzie

În lucrarea de laborator numărul 1 am făcut cunoștință cu VCS GIT. A fost creat un repository pe serverul online și a fost sincronizat cu cel local. A fost efectuate toate setările necesare. Putem conchide că acest VCS este o metodă foarte eficientă și rapidă de lucru asupra unui proiect, în mod special când este vorba de o echipă.

La îndeplinirea obiectivelor stabilite am întâlnit numeroase dificultăți, dar pot afirma cu încredere că există foarte multă documentație online și forumuri ce ajută la soluționarea acestor probleme.