

Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică și Microelectronică



Departamentul ISA

Raport

Lucrarea de laborator Nr. 1
la Programarea în rețea

Tema: Versionarea codului sursă folosind GIT

A elaborat st. gr. TI-144

D. Gorduz

A verificat lect. asist.

S. Ostapenco

Chișinău 2017

Scopul lucrării

Lucrarea de laborator are ca scop studiul și înțelegerea principiilor de funcționare și utilizare a sistemului distribuit de control al versiunilor numit GIT.

Obiectivul lucrării

Crearea unui repository distant, localizat de serviciul gitlab.ati.utm.md, și sincronizarea tuturor modificărilor efectuate asupra repositoryului local.

Sarcină tip pentru lucrarea de laborator

Reprezentarea vizuală a pașilor necesari pentru efectuarea lucrării de laborator include elemente grafice cu semnificații corespunzătoare comenzilor git. Prin urmare este necesar de luat în considerare următoarele:

- a) @ asociază repositoryul distant prin `git remote add origin`;
 - b) + adaugă fișiere/directorii și modifică repositoryul local prin `commit`;
 - c) - șterge fișiere/directorii și modifică repositoryul local prin `commit`;
 - d) ~ modifică fișiere și repositoryul local prin `commit`;
 - e) * (branch) îmbină o ramură în cea indicată, `git merge`;
 - f) ↓ `git pull`, trage modificările din repositoryul distant;
 - g) ↓↑ `pull and push`, sincronizează repositoryul local și distant;
- În corespundere cu elementele de notație, pașii efectuați în indicațiile metodice pot corespunde grafului de mai jos :

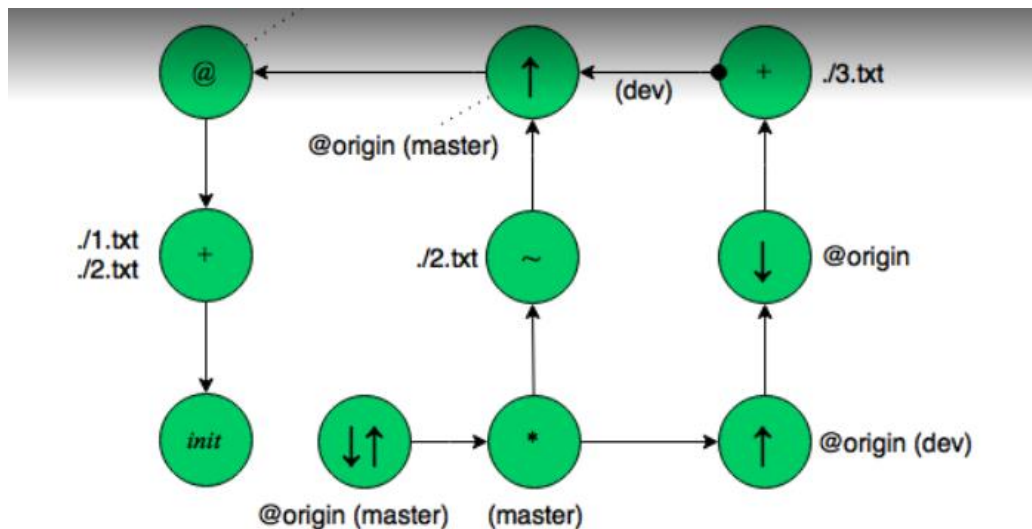


Figura 1 – Variantă pentru lucrare de laborator

Mersul Lucrării

Pentru a efectua această lucrare de laborator avem nevoie de sistemul de control Git pe care îl putem descărca de pe adresa: <http://git-scm.com/downloads>. Fișierele unui proiect pot fi păstrate atât local cât și pe https://github.com/GorduzDaniel/Lab1_PR. Pentru aceasta avem nevoie să creăm cont nou și să generăm o cheie ssh pentru a conecta repoziitoriul local cu cel de pe server. Primul pas pe care trebuie să-l facem pentru a începe lucrul cu Git este să creăm un director local, să pornim de acolo Git și să inițializăm repoziitoriul în acest director (vezi în figura 2).[1]

```
danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1
$ git init
Initialized empty Git repository in D:/UTM/University_III_2/PR/laborator_1/.git/

danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ touch README.txt
```

Figura 2 – Inițializarea GIT-ului

Pentru a verifica cum sistemul funcționează am creat în directorul nou două fișiere apoi am testat instrucțiunea *git status* (vezi în figura 3).

```
danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ touch file1.txt

danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ touch file2.txt

danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file1.txt
        file2.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Figura 3 – GIT status

Modificarea repoziitoriului local se face prin instrucțiunea *git add* și *git commit*. Pentru ca aceste modificări să ajungă pe repoziitoriul distant este nevoie de a testa instrucțiunea *git remote add origin git@gitlab.ati.utm.md:username/projectname.git* și respectiv *git push* (vezi în figura 4).

```
danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git add .

danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git commit -m "added 2 files"
[master 0e517ef] added 2 files
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file1.txt
create mode 100644 file2.txt

danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git push -u origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 287 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
Branch master set up to track remote branch master from origin.
To github.com:GorduzDaniel/Lab1_PR.git
262df31..0e517ef master -> master
```

Figura 4 – Adaugarea în repoziitoriu

Crearea unei ramure și trecerea de la una la alta se face cu ajutorul instrucțiunii *git branch **nume ramură*** și respectiv *git checkout **nume ramură***. Pe ramura **dev** a fost create un fișier nou *file3.txt* după care modificări au fost salvate cu **git add .** și **git commit-m"---"** și încărcate pe repoziitoriul distant, iar pentru trage modificările din repoziitoriul distant folosim **git pull**(vezi în figura 5,6,7 și 8).

```
danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git branch dev

danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git checkout dev
Switched to branch 'dev'

danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (dev)
$ touch file3.txt

danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (dev)
$ git status
On branch dev
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        file3.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Figura 5 – Crearea pe ramura dev a fișierului file3.txt

```
danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (dev)
$ git add .

danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (dev)
$ git commit -m "added file 3 on branch dev"
[dev 26c5294] added file 3 on branch dev
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 file3.txt
```

Figura 6 – Salvarea fișierului

```
danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (dev)
$ git pull origin master
From github.com:GorduzDaniel/Lab1_PR
 * branch          master       -> FETCH_HEAD
Already up-to-date.

danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (dev)
$ git push -u origin dev
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 267 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
Branch dev set up to track remote branch dev from origin.
To github.com:GorduzDaniel/Lab1_PR.git
 * [new branch]      dev -> dev
```

Figura 7 – Tragerea modificărilor și încărcarea pe repoziitoriul distant

```
danie@DESKTOP-63Q5EOG MINGW64 /d/UTM/University_III_2/PR/laborator_1 (dev)
$ git checkout master
Your branch is up-to-date with 'origin/master'.
Switched to branch 'master'
```

Figura 8 – Trecerea pe ramura master

Pentru a verifica cum sistemul funcționează am modificat **fișier2** de pe master apoi am testat instrucțiunea *git status* , apoi a fost adăugată și comentată(vezi în figura 9).

```
danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
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:   file2.txt

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

Figura 9 – Verificarea modificărilor efectuate

Iar pentru a uni două ramure este nevoie de a testa instrucțiunea *git merge* (vezi în figura 10).

```
danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git merge dev
Merge made by the 'recursive' strategy.
 file3.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file3.txt

danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git push -u origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 483 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local objects.
Branch master set up to track remote branch master from origin.
To github.com:GorduzDaniel/Lab1_PR.git
 0e517ef..259aa2c master -> master
```

Figura 10 – Folosirea comenzii git merge

Pentru ca toate aceste modificări să ajungă pe repoziitoriul distant este nevoie de a folosi instrucțiunea *git push* (vezi în figura 11,12).

```
danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git push -u origin master
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 483 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local objects.
Branch master set up to track remote branch master from origin.
To github.com:GorduzDaniel/Lab1_PR.git
 0e517ef..259aa2c master -> master

danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ |
```

Figura 11 – Încărcarea pe repoziitoriul distant

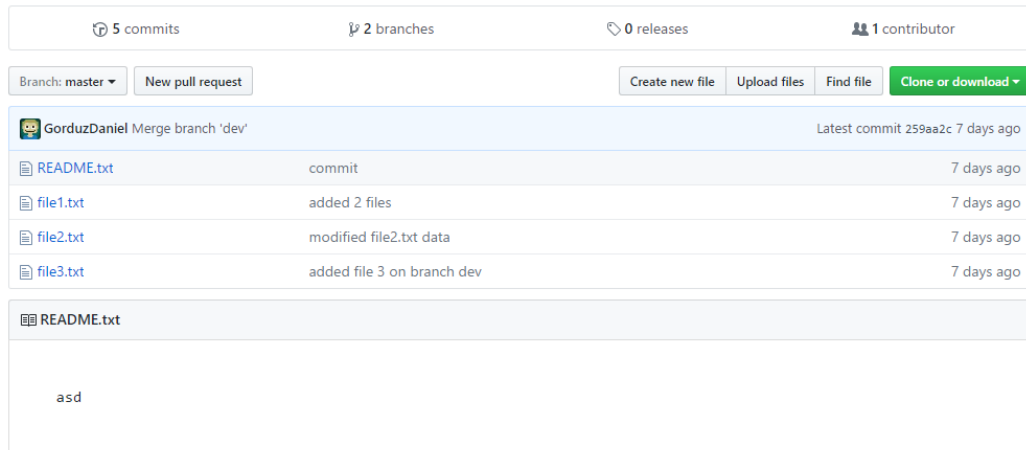


Figura 12 – Rezultatul modificărilor repoziatoriului distant

Pentru a vizualiza graficul comiturilor este nevoie de a folosi instrucțiunea ***git log --graph --all*** (vezi în figura 13).

```
danie@DESKTOP-63Q5E0G MINGW64 /d/UTM/University_III_2/PR/laborator_1 (master)
$ git log --graph --all
* commit 259aa2c25540bdd59432f10f966b9291bf11d4e8
  Merge: 6d64744 26c5294
  Author: Daniel Gorduz <daniel_gorduz@yahoo.com>
  Date: Tue Mar 7 08:33:56 2017 +0200

    Merge branch 'dev'

* commit 26c52942a330478d6a9a46ac09e3a02072047bcc
  Author: Daniel Gorduz <daniel_gorduz@yahoo.com>
  Date: Tue Mar 7 08:27:28 2017 +0200

    added file 3 on branch dev

* commit 6d64744f5e507bf6a05c5d24eb337995d96cb532
  Author: Daniel Gorduz <daniel_gorduz@yahoo.com>
  Date: Tue Mar 7 08:33:41 2017 +0200

    modified file2.txt data

* commit 0e517ef96c182086e4a8b904d7896d018c5037da
  Author: Daniel Gorduz <daniel_gorduz@yahoo.com>
  Date: Tue Mar 7 08:21:46 2017 +0200

....skipping...
```

Figura 13 – Vizualizarea grafului modificarilor asupra repoziatoriului local

Construirea proiectelor Java utilizând Apache Maven

În capitolul respectiv, va fi analizată metoda de creare a proiectelor Java, utilizând Apache Maven. În figurile 14, 15, 16 observăm etapele inițiale de creare a proiectului utilizând IntelliJ IDEA.[2]

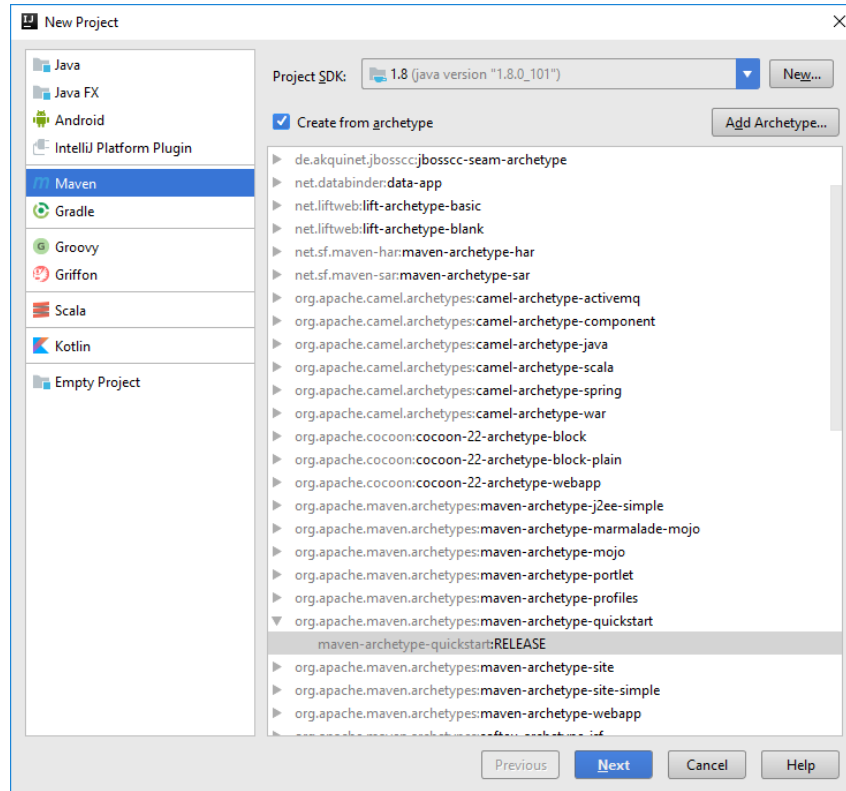


Figura 14 – Crearea unui proiect Maven

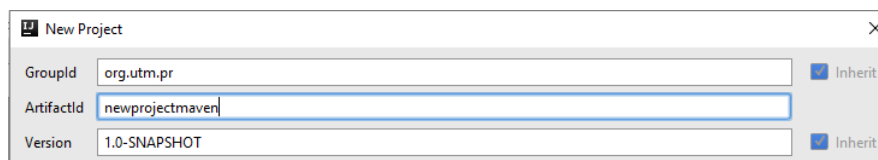


Figura 15 – Adăugarea datelor

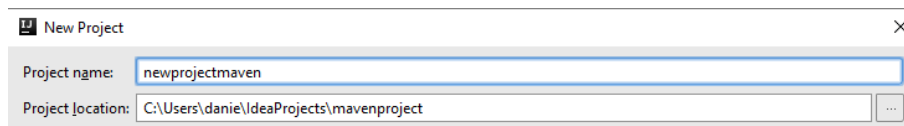


Figura 16 – Denumirea proiectului

În figura 17, poate fi observată configurarea *run/debug* necesară executării unui scop Maven:

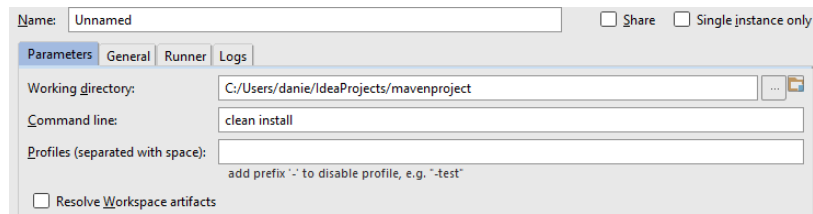
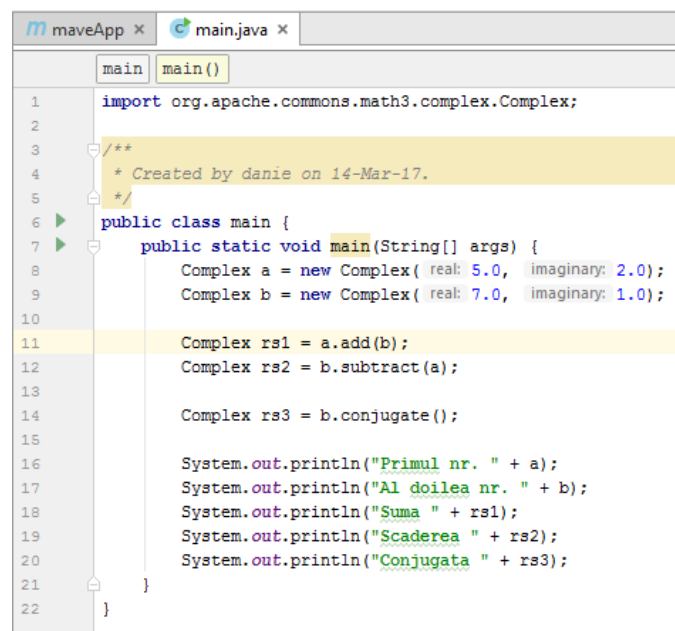


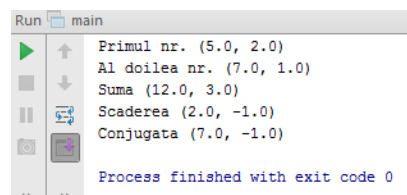
Figura 17 – configurarea run/debug

În scopul realizării lucrării de laborator, a fost utilizată dependența *commons-math 3.2*, ce a fost inclusă în proiect prin intermediul POM-ului (figura 18). Librăria respectivă include un set vast de opțiuni ce permit prelucrarea matematică a datelor.[3]



```
1 import org.apache.commons.math3.complex.Complex;
2
3 /**
4  * Created by daniel on 14-Mar-17.
5  */
6 public class main {
7     public static void main(String[] args) {
8         Complex a = new Complex( real: 5.0, imaginary: 2.0);
9         Complex b = new Complex( real: 7.0, imaginary: 1.0);
10
11         Complex rs1 = a.add(b);
12         Complex rs2 = b.subtract(a);
13
14         Complex rs3 = b.conjugate();
15
16         System.out.println("Primul nr. " + a);
17         System.out.println("Al doilea nr. " + b);
18         System.out.println("Suma " + rs1);
19         System.out.println("Scaderea " + rs2);
20         System.out.println("Conjugata " + rs3);
21     }
22 }
```

Figura 18 – Implementarea codului



```
Run main
Primul nr. (5.0, 2.0)
Al doilea nr. (7.0, 1.0)
Suma (12.0, 3.0)
Scaderea (2.0, -1.0)
Conjugata (7.0, -1.0)
Process finished with exit code 0
```

Figura 19 - Rezultatul execuției

Concluzie

În urma efectuării acestei lucrări de laborator au fost căpătate deprinderi în lucru cu GIT, aceasta reprezintă o sistemă de control al versiunilor. La momentul actual este una dintre cele mai folosite sisteme de acest gen.

În acest laborator au fost folosite cele mai des folosite posibilități ale GIT-ului, așa ca adăugarea unui fișier în indexate, efectuarea commit-urilor, crearea ramurilor, unirea ramurilor ș.a.

Cunoștințele căpătate la acesta lucrare de laborator vor servi drept bază pentru studierea mea în continuare a GIT-ului și pentru folosirea lui pentru controlul stării unui proiect.

Studiul Apache Maven a permis crearea unui proiect Java, ce utiliza dependența *common-math* 3.2, în cadrul căruia a fost testată un nou tip de date definit de acesta, și anume Complex, ce permite crearea nr. complexe și realizarea a diferite operații asupra acestora.

Bibliografie

1. NetworkProgrammingGuide.pdf, [Resursă electronică] – regim de acces: <https://drive.google.com/file/d/0B0vf11XUnLc2Q0NrLTk3czlOTWM/view>
2. Data Generation, [Resursă electronică] – regim de acces: <http://commons.apache.org/proper/commons-math/userguide/random.html>
3. Commons Math: The Apache Commons Mathematics Library, [Resursă electronică] – regim de acces: <http://commons.apache.org/proper/commons-math/>