

Ministerul Educației al Republicii Moldova
Universitatea Tehnică a Moldovei
Departament Automatică și Tehnologii Informaționale

RAPORT

Lucrare de laborator Nr.1
la Programarea în Rețea

Tema: Sistemul distribuit de control al versiunilor, GIT

A elaborat :

st.gr. TI-141 : Guba Dumitru

A verificat :

conf. univ. Ciorbă Dumitru

Chișinău 2017

Cuprins

Scopul lucrării.....	3
Sarcina lucrării.....	3
1. Versionarea GIT.....	3
2. NuGet package manager.....	4
3. Realizarea sarcinii.....	5
Concluzie.....	8
Bibliografie.....	9
Anexa A.....	10

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.

Obiectiv: Crearea unui repozitoriu distant, localizat de serviciul gitlab.ati.utm.md, și sincronizarea tuturor modificărilor efectuate asupra repozitoriului local.

1. Versionarea GIT

Noțiuni teoretice

Git este un sistem revision control care rulează pe majoritatea platformelor, inclusiv Linux, POSIX, Windows și OS X. Ca și Mercurial, Git este un sistem distribuit și nu întreține o bază de date comună. Este folosit în echipe de dezvoltare mari, în care membrii echipei acționează oarecum independent și sunt răspândiți pe o arie geografică mare.

Git este dezvoltat și întreținut de Junio Hamano, fiind publicat sub licență GPL și este considerat software liber.

SVC îți permite să vezi toate modificările făcute pe parcursul dezvoltării proiectului și ajută foarte mult în situațiile următoare:

* **Revenirea la o versiune anterioară** . De exemplu, când ai făcut un update care nu merge așa cum trebuie, poți reveni la ultima versiune stabilă. Sau în cazul în care ai scos un feature din proiect la cererea clientului, dar acesta s-a răzgândit și tu l-ai șters între timp.

* **Lucrul în echipă**. Permite ca mai mulți oameni să lucreze în același fișier simultan.

* **Backup**. Dacă ai șters un fișier din greșală, poți oricând să-i dai restore.

Principalele comenzi puse la dispoziție de sistemul de versiune distribuit GIT sunt:

- *repository* - pe server, conține ierarhia de fișiere și informațiile de versiune;
- *working copy* - varianta locală, obținută de la server, pe care se fac modificările;
- *revision* - o versiune a unui document. (v1, v2, v3...).
- *checkout* - aducerea pe masina locala a versiunii de pe server, sub forma unei working copy
- *update/pull*: actualizarea repozitoriului local în funcție de modificările survenite, între timp, pe server. Se aduc doar fișierele modificate;
- *commit* - înregistrează o nouă versiune a fișierului (fișierelor) modificat în repozitoriu.
- *commit message* - un mesaj asociat unei acțiuni *commit* care descrie schimbările făcute în noua versiune.
- *changelog* - o listă a versiunilor (commit-urilor) unui fișier/proiect de obicei însoțită de mesajele asociate fiecărui *commit*.

- *diff*: Afișează diferențele dintre două versiuni a unui fișier sau dintre fișierul modificat local (pe working copy) și o versiune de pe repository.
- *revert* - renunțarea la ultimele modificări (locale) făcute într-un fișier din *working copy*, și revenirea la ultima versiune aflată în repozițoriu sau la o versiune la alegere.
- *branch* - creează o “copie” a unui fișier/proiect pentru modificări „în paralel” fără a afecta starea actuală a unui proiect.
- *merge* - aplică ultimele modificări dintr-o versiune a unui fișier peste alt fișier;
- *conflict* - situația în care un *merge* nu se poate executa automat și modificările locale sunt în conflict cu modificările din repozițoriu.
- *resolve*: rezolvarea (de obicei manuală) a conflictelor apărute într-un fișier după un *merge*.

2. NuGet package manager

În realizarea laboratorului am utilizat ca mediu de dezvoltare Visual Studio (C#). Am creat o aplicație de procesare a imaginilor folosind librăria AForge.NET. Pentru a putea folosi această librărie în program am folosit instrumentul *NuGet*.

NuGet reprezintă un instrument pentru lucrul cu pachetele. Datorită acestui instrument putem descărca și utiliza librăriile necesare în dezvoltarea aplicațiilor soft (figura 1).

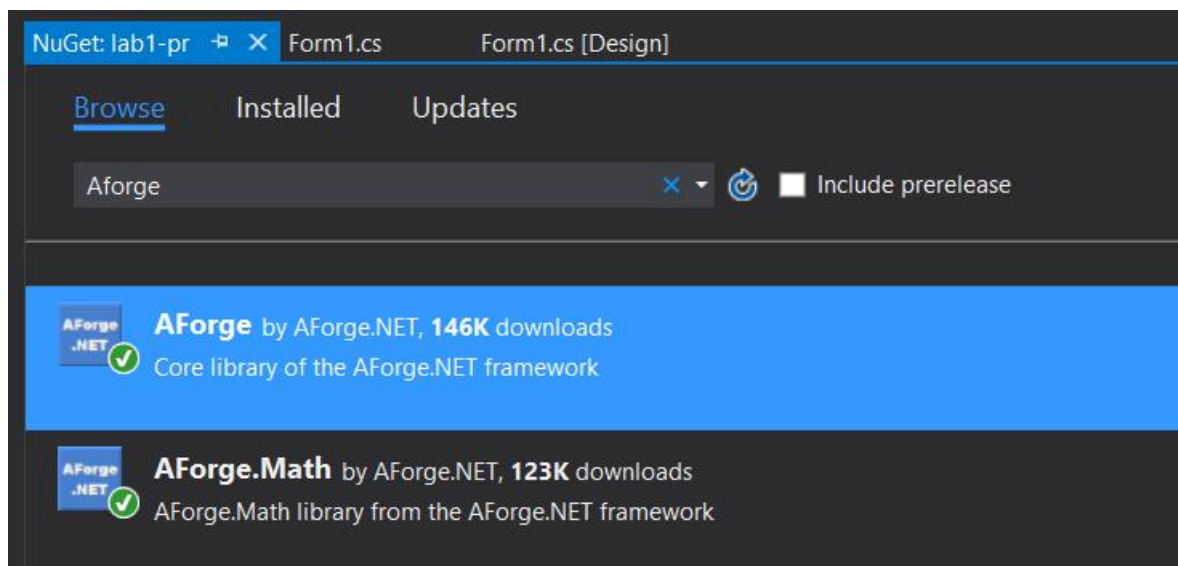


Figura 1 - Descărcarea librăriei AForge.NET

NuGet este un manager de pachete gratuit și open-source proiectat pentru platforma de dezvoltare Microsoft (cunoscut anterior ca NuPack). De la introducerea sa în 2010, NuGet a evoluat într-un ecosistem mai larg de instrumente și servicii. NuGet este distribuit ca o extensie Visual Studio. Începând cu Visual Studio 2012, NuGet vine preinstalat în mod implicit. NuGet este de asemenea integrat cu SharpDevelop. NuGet poate fi, de asemenea, utilizat din linia de comandă și automat cu script-uri. Aceasta susține mai multe limbaje de programare, inclusiv: pachete NET Framework pachete native scrise în C ++, cu crearea de pachete ajutat de CoApp.

3. Realizarea sarcinii:

- 1) Descarcarea aplicatiei de instal - git.exe
- 2) Crearea repozitoriului nou
- 3) Configurarea GIT
- 4) Crearea repozitoriului pe GitHub si setarea lui
- 5) Realizarea exercitiului de baza

- Aplicatia pentru instalarea sistemului de control a versiunilor git poate fi descarcata la adresa: <https://git-scm.com/downloads>

- Pentru crearea unui repozitoriu local am efectuat urmatoarele comenzi in Git bash:

- `mkdir PR`
- `cd PR`
- `git init` - pentru initializarea git-ului in repozitoriul dat

- Pentru configurarea GIT este nevoie de a specifica username-ul si email-ul nostru prin comenzile necesare.

Pentru username:

```
git config user.name "dumitruguba"
```

iar pentru email:

```
git config user.email "dumitru.guba@gmail.com"
```

Deasemenea este nevoie de a seta ssh key (public and private) prin:

- `ssh-keygen -t rsa -b 4096 -C "dumitru.guba@gmail.com"`
- Enter a file in which to save the key (/Users/you/.ssh/id_rsa): yourTextFileName

- Pentru a crea un nou repozitoriu pe GitHub este nevoie sa selectam optiunea New Repository. Dupa care avem nevoie sa facem legatura intre repozitoriul nostru local si cu cel public de pe GitHub prin comanda: `git remote add origin remote repository URL` .

- Ca exercitiu am avut de lucrat pe 2 ramuri(master si dev), de adaugat un fisier fiind pe ramura master, de schimbat pe ramura dev si de adaugat un nou fisier, iar apoi de trecut pe master si de creat al 3-lea fisier si de facut merge intre master si dev.

Efectuarea sarcinii este descrisa in detalii in imaginile ce urmeaza (vezi figura: 2, 3, 4, 5), mentionez - dupa efectuarea modificarilor pe un branch este nevoie de a actualiza noile operatii si pe GitHub prin comenzile: `git add .` , `git commit -m "info about modifications"`, `git push origin branchName`.

```
wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (master)
$ git status
On branch master
nothing to commit, working tree clean

wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (master)
$ touch f1.txt
```

Figura 2 - Crearea primului fisier pe master

În figura 2 are loc crearea fisierului f1.txt în directorul /PR aflîndumă pe branch-ul master.

```
wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (master)
$ git checkout -b dev
fatal: A branch named 'dev' already exists.

wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (master)
$ git checkout dev
Switched to branch 'dev'

wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (dev)
$ touch f2.txt
```

Figura 3 - Crearea fisierului 2 pe dev

În figura 3 are loc crearea unui nou branch “dev” și aflîndumă deja pe acest branch se creează fișierul f2.txt.

```
wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (dev)
$ git checkout master
Switched to branch 'master'

wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (master)
$ touch f3.txt
```

Figura 4 - Crearea fisierului 3 pe master

În figura precedenta se trece de pe branch-ul dev înapoi pe master pentru crearea fișierului 3 f3.txt.

```
wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (dev)
$ git checkout master
Switched to branch 'master'

wert Jr@DESKTOP-FS3BT1U MINGW64 /PR (master)
$ git merge dev
```

Figura 5 - Merge între master și dev

Graficul de lucru asupra exercitiului este prezentat în *figura 6*.

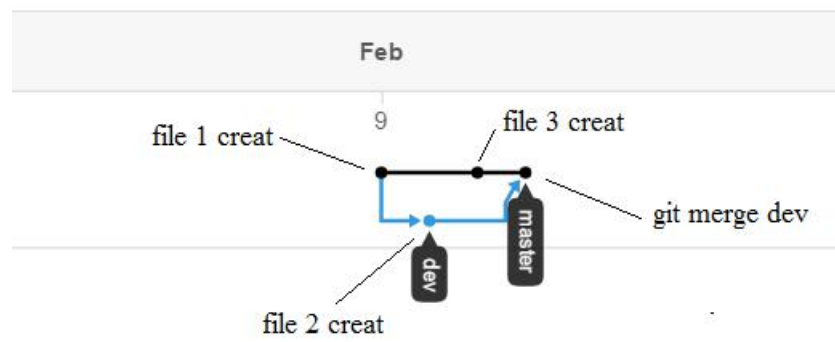


Figura 6 - Graful interactiunii intre ramuri

Concluzii:

În lucrarea data am studiat și am deprins lucrul cu sistemul de control al versiunilor GIT. Am învățat comenzile de bază în lucrul cu acest sistem. Am instalat git.exe, am creat repozitoriul de lucru și am configurat GIT-ul pentru a putea lucra asupra proiectelor dorite.

Prin crearea unui repozitoriu distant, și sincronizarea acesteia cu modificărilor efectuate asupra repozitoriului local, s-a determinat principalele beneficii acestui sistem cum ar fi: operearea a diverse modificări pe același proiect a mai multor membri a diferitor echipe de dezvoltare.

Bibliografie

- **Sistemul de versionare GIT** [Resursă electronică] - Regim de acces: <https://www.git-tower.com/learn/git/ebook/en/command-line/remote-repositories/introduction#start>
- **Ciorbă Dumitru, Beșliu Victor, Poștaru Andrei, Rostislav Călin, Programarea în rețea.** Introducere în dezvoltarea aplicațiilor Java [Resursă electronică] – Regim de acces: <https://moodle.ati.utm.md/mod/page/view.php?id=2089>
- <https://marketplace.visualstudio.com/items?itemName=NuGetTeam.NuGetPackageManager>

Anexa A

<https://github.com/dumitruguba/PR/tree/master/lab1-pr>