

Ministerul Educației al Republicii Moldova

Universitatea Tehnică a Moldovei

Catedra Automatică și Tehnologii Informaționale

# **RAPORT**

Lucrare de laborator Nr.1

la PR

Tema: Versionarea codului sursă utilizind GIT

A elaborat :

st.gr. TI-142 : Chicu Roman

A verificat :

Ostapenco Stepan

## **Tema** Versionarea codului sursă utilizind GIT

### **Scop și Obiective**

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

### **Noțiuni Teoretice**

Sistemele de versionare (VCS, Version Control Systems - eng.) servesc la gestionarea versiunilor multiple ale fișierelor incluse într-un proiect colaborativ. Fiecare modificare efectuată asupra elementului de proiect se memorizează împreună cu autorului schimbării. Important de menționat că în orice moment de timp se poate reveni la o versiune anterioară a entității.

Motivatia principala consta in posibilitatea ca diferiti membri ai echipei, aflati eventual in spatii geografice indepartate, sa poata lucra simultan la proiect, urmand ca, la final, modificarile lor sa fie reunite in noi versiuni ale proiectului. De asemenea, exista si alte avantaje. Cand se observa un bug, se poate reveni la o versiune anterioara, in vederea determinarii momentului introducerii acestuia in program. In acelasi timp, se poate urma o dezvoltare pe ramuri (branches), in care se lucreaza, in paralel, la multiple versiuni ale proiectului - de exemplu, una in care se doreste inlaturarea bug-urilor, iar cealalta, in care se urmareste adaugarea de noi functionalitati, inaintea slefuirii celor existente.

Exista doua modele de VCS-uri:

- centralizat (ex: SVN): codul sursa este situat pe un server central, de unde clientii pot obtine variante de lucru pe masina locala (working copy). Dupa efectuarea locala a modificarilor, dezvoltatorul solicita actualizarea variantei de pe server.
- distribuit (ex: Git): nu exista un server central, procesul de sincronizare desfasurandu-se la nivel peer-to-peer.

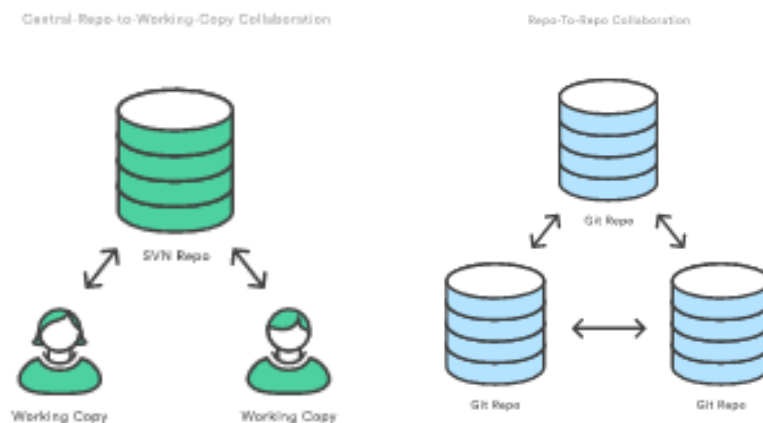


Figura 1. Modul de interacțiune cu dipozitorile

## Terminologie

- *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 mesaje 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 repozitoriu 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 repozitoriu.
- *resolve*: rezolvarea (de obicei manuală) a conflictelor apărute într-un fișier după un *merge*.

## Mersul Lucrării

1. Instalăm sistemul de control al versiunilor
  - GitBash - Downloads, <https://git-scm.com/downloads>
2. Creăm cont <https://github.com/>
  - Vă adresați profesorului pentru acest punct
  - Adăugați cheia publică generată de ssh-keygen (în git-bash terminal) în profilul utilizatorului. (Atenție! Cheia publică se regăsește în conținutul fișierului generat cu extensia **.pub**; cheile fiind localizate în directorul ascuns **.ssh** al utilizatorului de sistem: `./../Users/NumeUtilizator/.ssh/id_rsa`) reprezentare figura 1,2.

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Technical University of Moldova/si  
mestru 2/PR/lab1 (master)  
$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/c/Users/Roman-PC/.ssh/id_rsa):  
/c/Users/Roman-PC/.ssh/id_rsa already exists.  
Overwrite (y/n)? n
```

Figura 2. Generare parolă

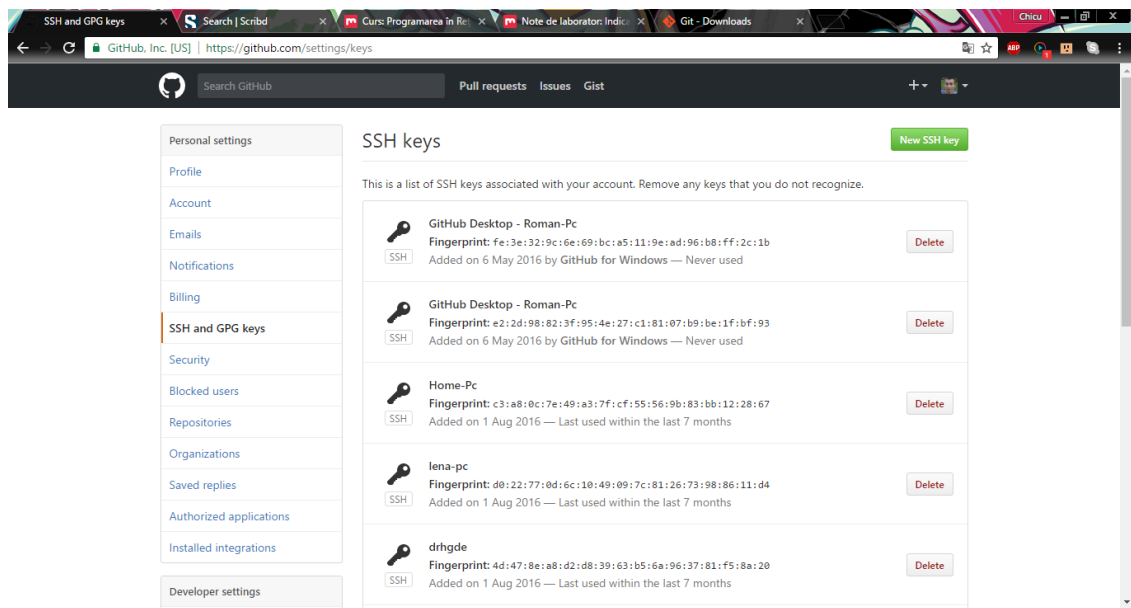


Figura 3. Adăugarea parolei pe Github

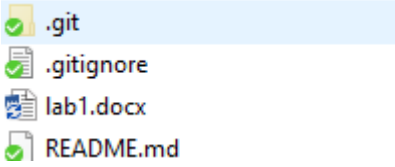
3. Creăm un director local de lucru în care se vor afla fișierele de versionat
  - Puteți utiliza comanda **mkdir** pentru crearea directorului în linia de comandă a terminalului, iar navigarea o puteți realiza prin: **cd NumeDirector**.
4. Inițializați repozitorul GIT în acest director figura 4.
  - **git init**

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University of Moldova/s
mestru 2/PR/lab1 (master)
$ git init
Reinitialized existing Git repository in C:/Users/Roman-PC/OneDrive - Technical
University of Moldova/simestru 2/PR/lab1/.git/
```

Figura 4. Inițializarea unui de pozitoriu local

5. Modificați conținutul directorului figura 5.

- adăugam 1-2 fișiere...



```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University
mestru 2/PR/lab1 (master)
$ touch .gitignore
```

Figura 5. Inițializarea unui fișier gitignore

6. Modificam repozitoriul local, ori de câte ori se modifică directorul de lucru:

- STAGE - *git add .* (figura 6)
- HEAD - *git commit -m "First commit"* (figura 7)

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive
mestru 2/PR/lab1 (master)
$ git add .
```

Figura 6. Adăugarea in depozitoriu.

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Un
mestru 2/PR/lab1 (master)
$ git commit -m "Init file"
[master 2d198d6] Init file
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 .gitignore
create mode 100644 lab1.docx
```

Figura 7. Commitul inițial

7. Adăugam repozitoriul distant figura 8.

- *git remote add origin [git@github.com:logan11116/lab1.git](https://github.com/logan11116/lab1.git)*

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Universi
mestru 2/PR/lab1 (master)
$ git remote add origin git@github.com:logan11116/lab1.git
```

Figura 8. Inițializarea noilor configurări

8. Reînnoim repozitoriul distant *origin*, ramura *master* figura 9

- *git push -u origin master*

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University of Moldova
mestru 2/PR/lab1 (master)
$ git push -u origin master
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 200.30 KiB | 0 bytes/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:logan11116/lab1.git
   36c2122..2d198d6  master -> master
Branch master set up to track remote branch master from origin.
```

Figura 9. Adăugarea la depozitoriul extern

9. Utilizam frecvent comanda *git status* pentru a vedea starea directorului de lucru și repozitoriului local figura 10.

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University of Moldova
mestru 2/PR/lab1 (Roman)
$ git status
On branch Roman
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        lab.cpp

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

Figura 10. Starea fișierelor

10. Creăm o ramură nouă de dezvoltare **Roman**, *numebranch* figura 11.

- *git branch Roman*

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University of Moldova
mestru 2/PR/lab1 (master)
$ git branch Roman
```

Figura 11. Crearea unui nou branch

11. Trecem în ramura nou creată **Roman**

- *git checkout Roman*
- ultimele două operații de creare și migrare la branch pot fi combinate prin: *git checkout -b Roman*

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University of Moldova
mestru 2/PR/lab1 (master)
$ git checkout Roman
Switched to branch 'Roman'

Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical University of Moldova
mestru 2/PR/lab1 (Roman)
$ |
```

Figura 12. Migrarea la noua ramură

12. Modificăm directorul de lucru și repozitoriul local figura 13

- Adăugați un nou fișier pe lângă cele existente
- Reînnoim repozitoriul local (similar p. 6)

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Tech
mestru 2/PR/lab1 (Roman)
$ git add .

Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Tech
mestru 2/PR/lab1 (Roman)
$ git commit -m "Adaugam un file cpp"
[Roman 14c8a24] Adaugam un file cpp
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 lab.cpp
```

Figura 13. Adăugarea în depozitorul local

13. Verificam dacă sunt modificări în repozitoriul distant figura 14

- *git pull*
- studiem care alte comenzi *git* cuprinde această comandă!

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Technical University of Moldova,
mestru 2/PR/lab1 (Roman)
$ git pull
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

git branch --set-upstream-to=origin/<branch> Roman
```

Figura 14. Monitorizarea schimbărilor la nivel de repozitoare

14. Reînnoim repozitoriul distant, aflîndu-vă pe ramura **Roman** figura 15

- *git push -u origin Roman*

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Technical University of
mestru 2/PR/lab1 (Roman)
$ git push -u origin Roman
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':
Counting objects: 2, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 229 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To github.com:logan1116/lab1.git
 * [new branch]      Roman -> Roman
Branch Roman set up to track remote branch Roman from origin.
```

Figura 15. Adăugarea în repozitoriul extern






15. Verificam dacă sunt modificări în repozitoriul distant **origin** (similar p. 13) (figura 16)

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Technical Univ
mestru 2/PR/lab1 (Roman)
$ git pull origin
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':
Already up-to-date.
```

Figura 16. Monitorizarea schimbărilor

16. Trecem pe ramura principală de dezvoltare **master** figura 17.

- `git checkout master`
- Observați conținutul directorului de lucru.
- Opțional puteți modifica directorul de lucru și repoziitoriul pe ramura **master** (pp. 5-6)

	.git	2/21
	.gitignore	2/21
	lab.cpp	2/21
	lab1.docx	2/21
	README.md	2/14

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Techmestru 2/PR/lab1 (Roman)
$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Techmestru 2/PR/lab1 (master)
$ |
```

Figura 17. Mergerea la nouă ramură





	.git
	.gitignore
	lab1.docx
	README.md

Figura 18. Schimbările apărute in repoziitoriu

17. Unificam ramurile **master** și **Roman** figura 19

- `git merge Roman`

```
Roman-PC@DESKTOP-A10I0VQ MINGW32 ~/OneDrive - Techmestru 2/PR/lab1 (master)
$ git merge Roman
Updating 2d198d6..14c8a24
Fast-forward
 lab.cpp | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 lab.cpp
```






	.git	2/20/2017 2:
	.gitignore	2/20/2017 1:
	lab.cpp	2/20/2017 2:
	lab1.docx	2/20/2017 1:
	README.md	2/14/2017 10

Figura 19. Unificarea ramurilor



18. Reînnoim repozitoriul distant **origin**, ramura **master** (similar p. 8)

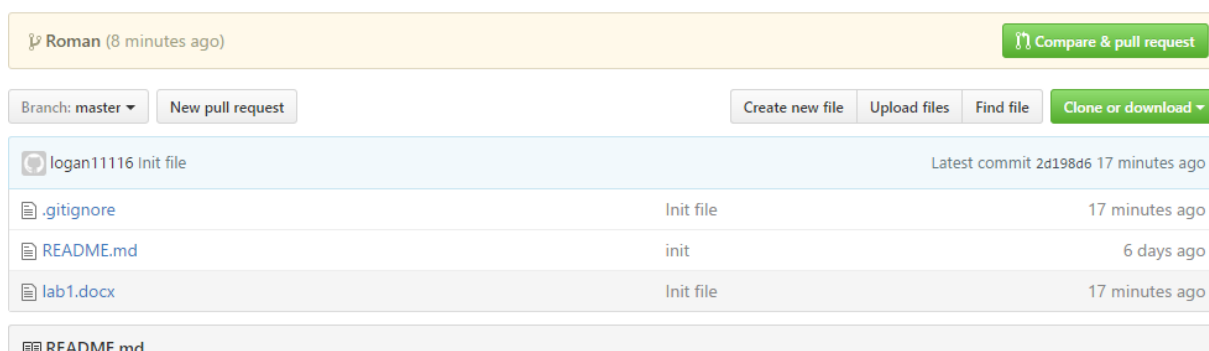


Figura 20. Schimbările în ramura master

```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Universit  
mestru 2/PR/lab1 (master)  
$ git push -u origin master  
Enter passphrase for key '/c/Users/Roman-PC/.ssh/id_rsa':  
Total 0 (delta 0), reused 0 (delta 0)  
To github.com:logan11116/lab1.git  
2d198d6..14c8a24 master -> master  
Branch master set up to track remote branch master from origin.
```

Figura 21. Adăugăm fișiere în dipozitoriul extern

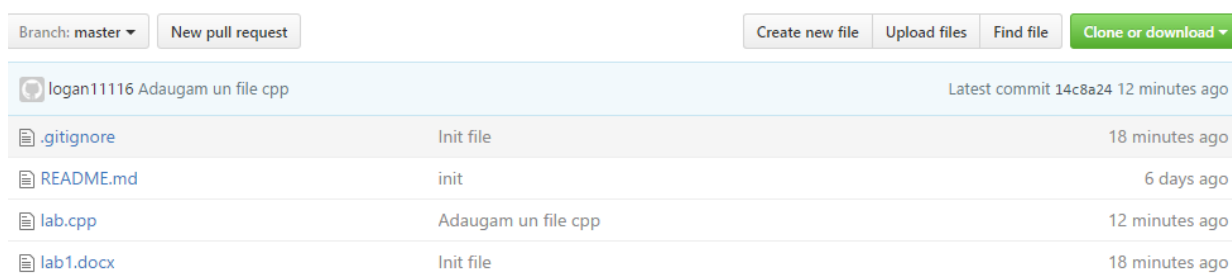
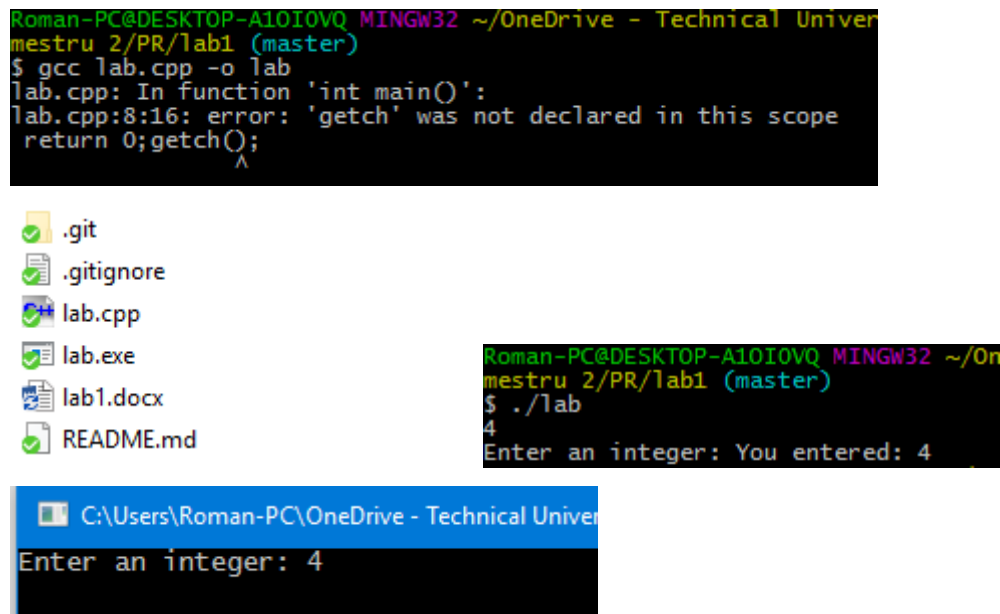


Figura 22. Schimbările survenite în repozitoriu

19. Observa modificările realizate în proiectul localizat

în <https://github.com/logan11116/lab1>

20. Compilăm un fișier cpp în git bash figura 23.



```
Roman-PC@DESKTOP-A10IOVQ MINGW32 ~/OneDrive - Technical Univer
mestru 2/PR/lab1 (master)
$ gcc lab.cpp -o lab
lab.cpp: In function 'int main()':
lab.cpp:8:16: error: 'getch' was not declared in this scope
return 0;getch();
               ^
$ ./lab
4
Enter an integer: You entered: 4
```

Files in the directory:

- .git
- .gitignore
- lab.cpp
- lab.exe
- lab1.docx
- README.md

Figura 23. Compilarea, lansarea în execuție

## Construirea proiectelor Java utilizând Apache Maven

Obiectivele lucrării: înțelegerea procesului de construire automată a unui proiect Java, cunoașterea fazelor esențiale Maven; obiectivul specific constând în setarea unui proiect Maven dependent de librării externe localizate pe servere centrale și analiza avantajelor oferite de acest

Crem un dipozitoriu visual studio InitProj ca în figura 24 instrument.

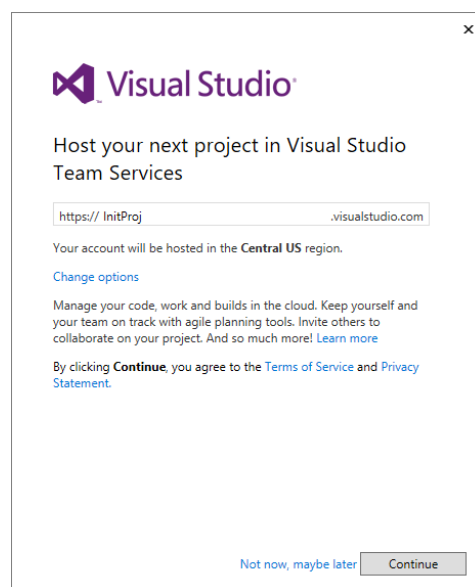
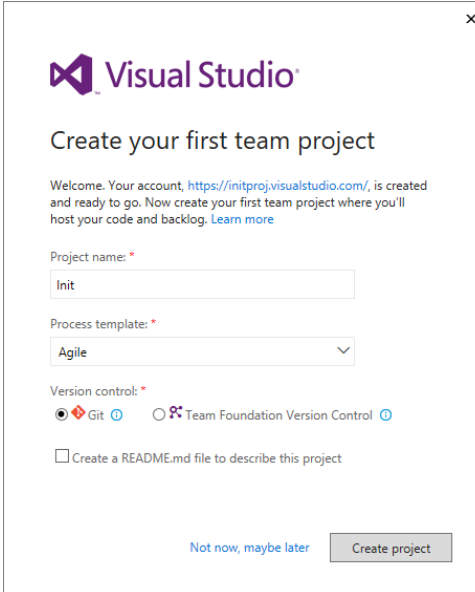


Figura 24. Crearea unui dipozitoriu VS

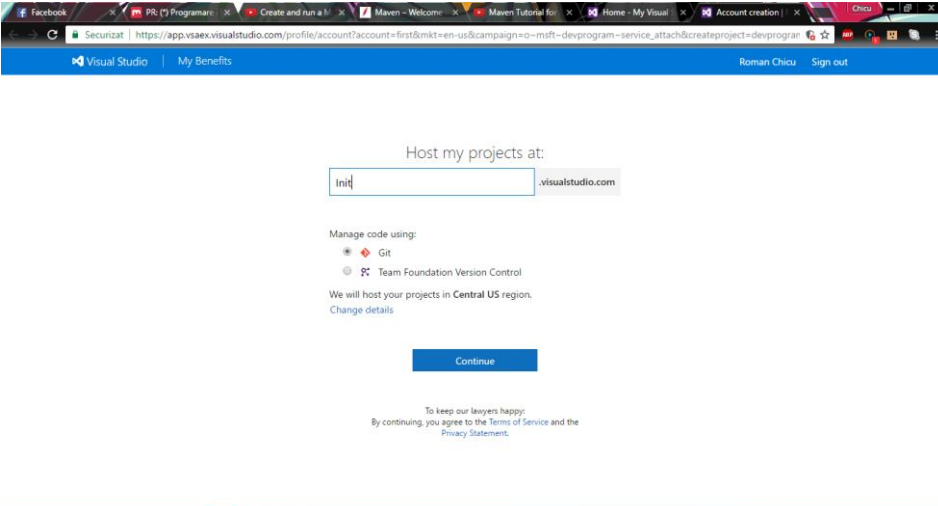
Creare un proiect comun unde mai mulți v-a avea acces pentru modificări și redactare (figura 25).



The image shows a Visual Studio dialog box titled "Create your first team project". It includes a welcome message, a "Project name" field with the value "Init", a "Process template" dropdown set to "Agile", and "Version control" options with "Git" selected. There is a checkbox for "Create a README.md file" and buttons for "Not now, maybe later" and "Create project".

Figura 25. Crearea unui proiect

Asemănător putem inițializa un host project în <https://app.vsaex.visualstudio.com> reprezentare în figura 26



The image shows a web browser window displaying the Visual Studio portal. The main form is titled "Host my projects at:" and has a text input field containing "Init" and a ".visualstudio.com" domain field. Below this, there are "Manage code using:" options for "Git" and "Team Foundation Version Control". A note states "We will host your projects in Central US region." with a "Change details" link. A blue "Continue" button is at the bottom, followed by a legal disclaimer: "To keep our lawyers happy: By continuing, you agree to the Terms of Service and the Privacy Statements."

Figura 26. Crearea unui nume de host project

După inițializare a unui proiect creăm un fișier Maven figura 4. Cu reprezentarea repozitoriului și metoda de conlucrare a acestuia figura 27.

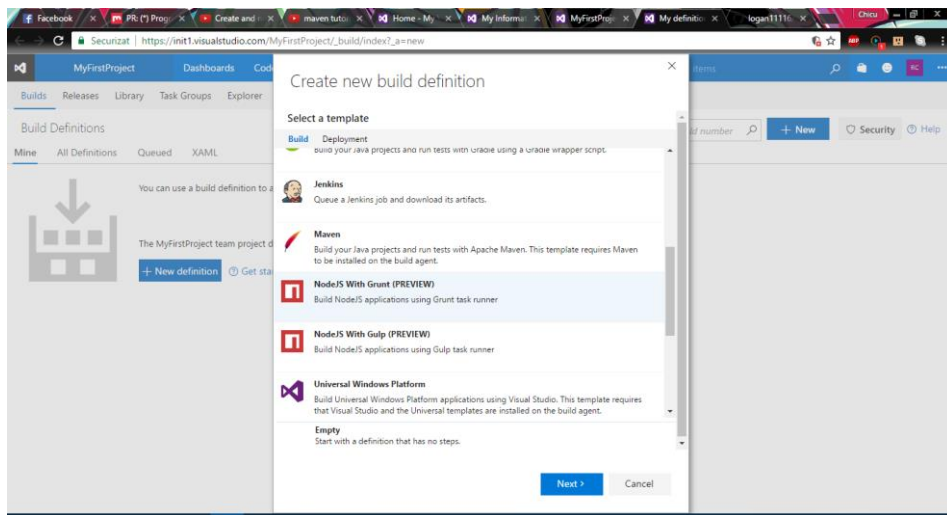


Figura 27. Creare unui fișier Maven

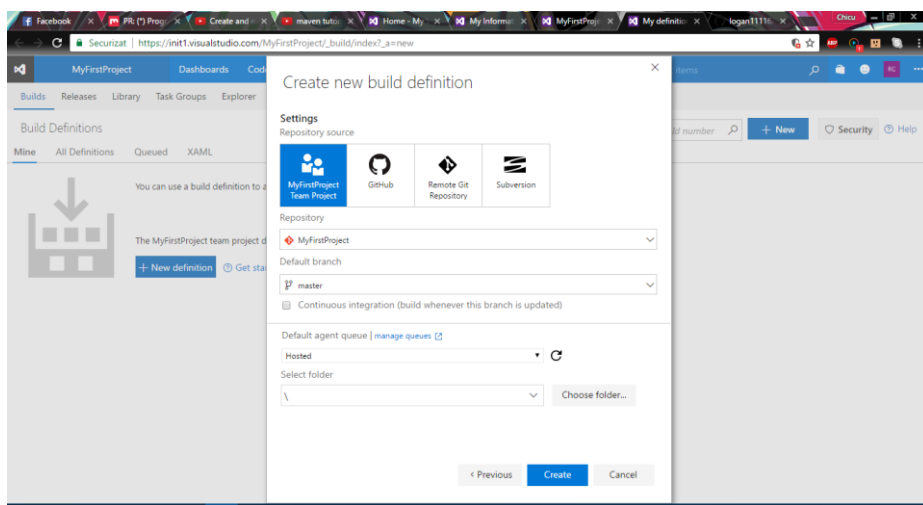


Figura 28. Init project repository

Configurare unui project Maven se va face în modul reprezentat în figura 29,30 .

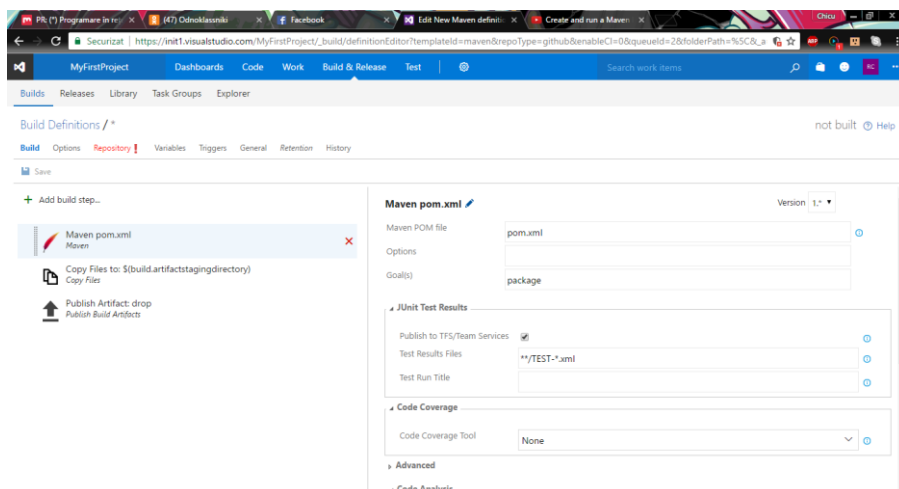


Figura 29. Configurarea unui Pub. Artifact și Maven file

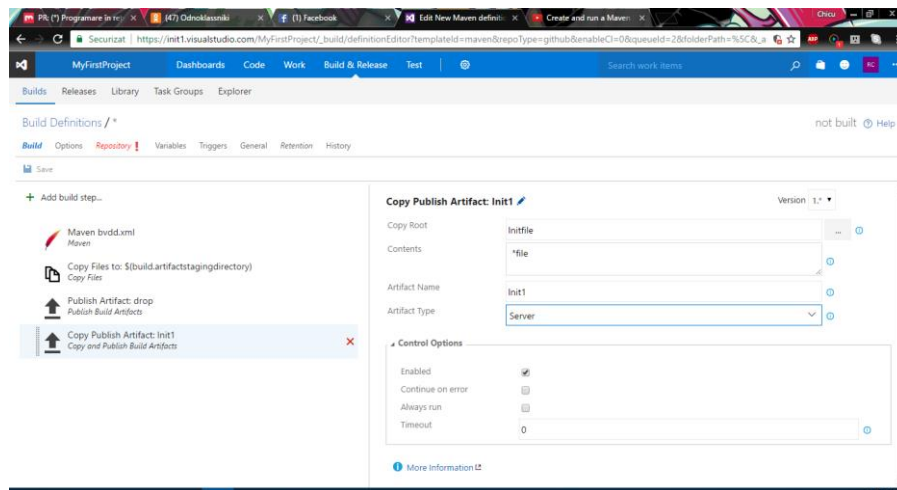


Figura 30. Crearea unui task

Facem build la project asemănător ca în figura 31.

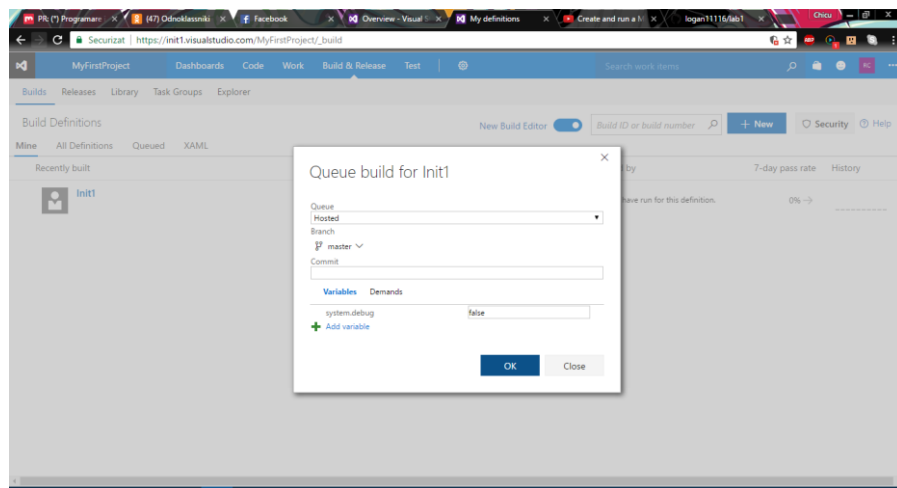


Figura 31. Build project

În următoarea figură este reprezentată rezultatul buildului proiectului Maven figura 9.

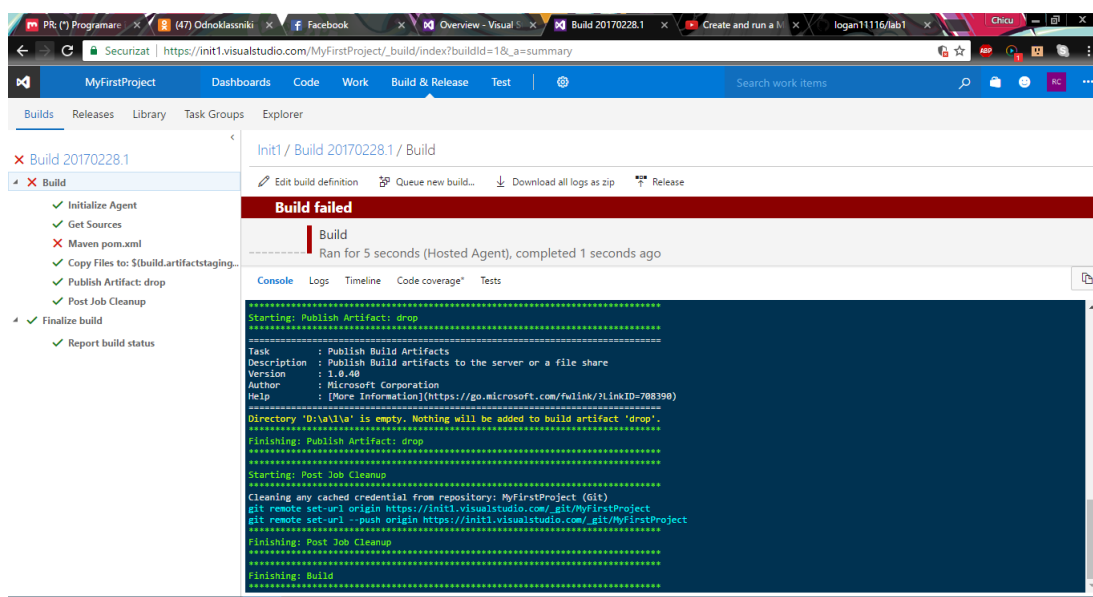


Figura 32. Rezultatul buildului

## **Concluzie**

Ni-am făcut familiari cu mediul github și în special am lucrat cu consola git bash în care am creat un depozitoriu local și l-am importat într-un extern. Am studiat diverse comenzi git și importanța lor într-un mediu unix. Am compilat un cod sursă în terminal

## Bibliografie

1. Scott Chacon, Pro Git, July 29, 2009 <http://git-scm.com/book>
2. Lars Vogel, Git - Tutorial, actualizat  
- 14.12.2014, <http://www.vogella.com/tutorials/Git/article.html>
3. Git How To, <http://githowto.com/>
4. Atlassian, Git Tutorials, <https://www.atlassian.com/git/tutorial>
5. Vincent Driessen, A successful Git branching model, January 05, 2010, <http://nvie.com/posts/a-successful-git-branching-model/>
6. [Linux.conf.au 2013] - Git For Ages 4 And Up,  
Youtube, <https://www.youtube.com/watch?v=1ffBJ4sVUb4>
7. Visualizing Git Concepts with D3, <http://onlywei.github.io/explain-git-with-d3/>
8. tryGit, <https://try.github.io>
9. Learn Git Branching, <http://pcottle.github.io/learnGitBranching/>
10. Code School, Git Real, Free preview, <https://www.codeschool.com/courses/git-real>
11. Code School, Git Real 2, Free preview, <https://www.codeschool.com/courses/git-real-2>