Ministerul Educaţiei 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

Chişinău 2017

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

### **Scop și Obective**

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.

### **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.

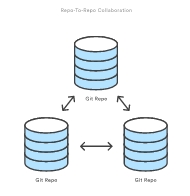
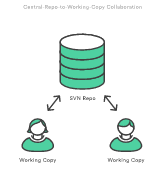


Figura 1. Modul de interacțiune cu dipozitorile

### **Terminologie**

* *repository -* pe server, conține ierarhia de fișiere și informațiile de versiune;
* *working copy -* varianta locală, obținuta 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, intre 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 repozitoriu sau la o versiune la alegere.
* *branch - c*reează o “copie” a unui fișier/proiect pentru modificări „în paralel” fără a afecta starea actuală a unui proiect.
* *merge - a*plică ultimele modificări dintr-o versiune a unui fișier peste alt fișier;
* *conflict - s*ituația 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 Lucrări**

1. Instalam sistemul de control al versiunilor

* GitBash - Downloads,  <https://git-scm.com/downloads>

1. Crem 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.

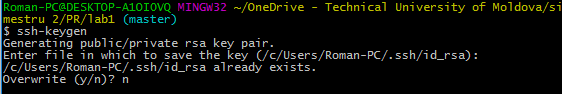


Figura 2. Generare parolă

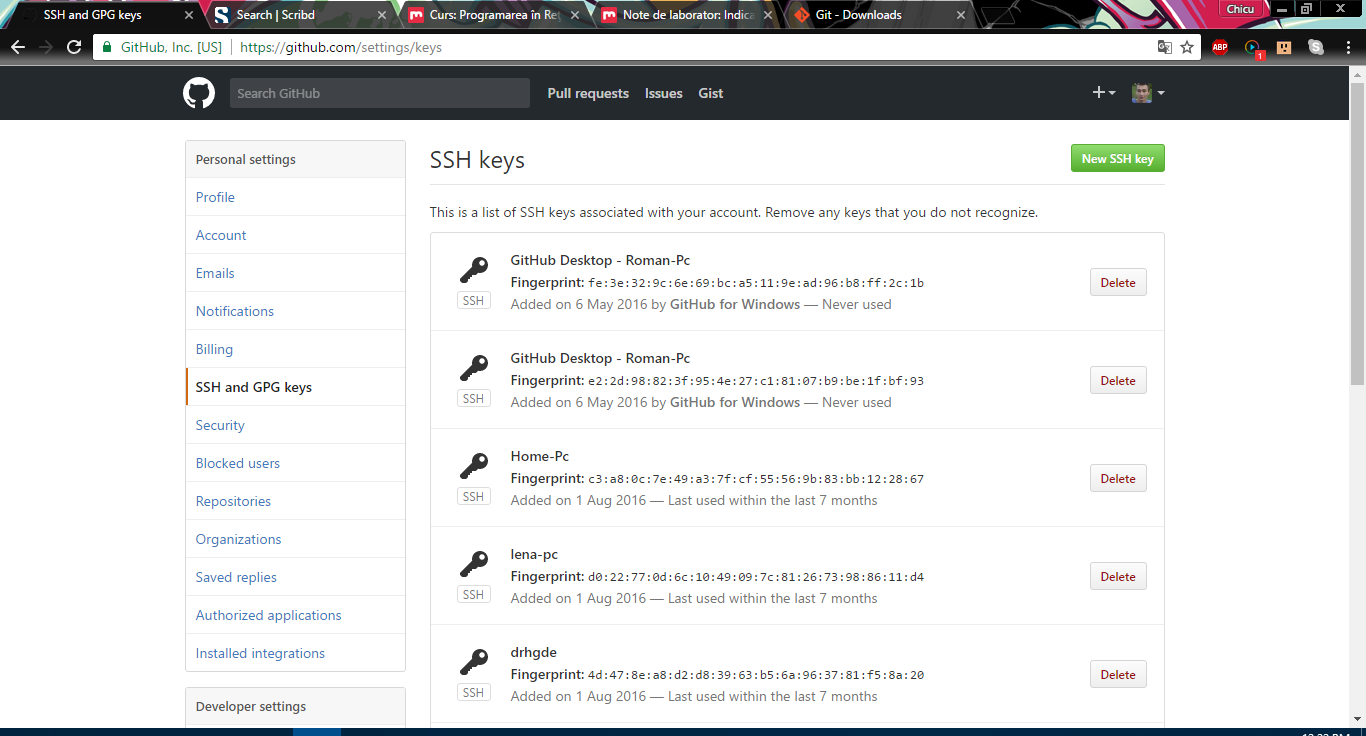


Figura 3. Adăugarea parolei pe Github

1. Crem 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**.

1. Inițializați repozitorul GIT în acest director figura 4.

* ***git init***

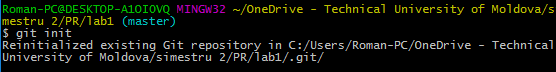


Figura 4. Inițializarea unui de pozitoriu local

1. Modificați conținutul directorului figura 5.

* adăugam 1-2 fișiere...

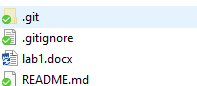




Figura 5. Inițializarea unui fișier gitignore

1. 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)*



Figura 6. Adăugarea in depozitoriu.

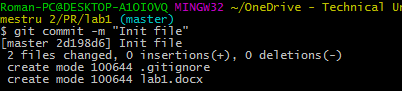


Figura 7. Commitul inițial

1. Adăugam repozitoriul distant figura 8.

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



Figura 8. Inițializarea noilor configurări

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

* git push -u***origin***master

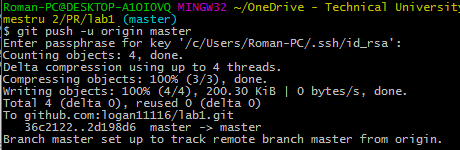


Figura 9. Adăugarea la depozitoriul extern

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

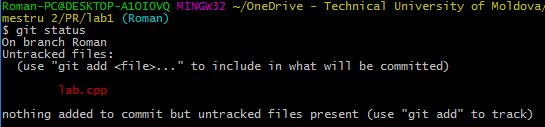


Figura 10. Starea fișierelor

1. Crem o ramură nouă de dezvoltare ***Roman, numebranch figura 11.***

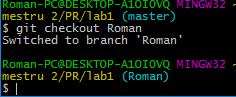
* ***git branch Roman***



Figura 11. Crearea unui nou branch

1. 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***



**Figura 12. Migrarea la noua ramură**

1. Modificam directorul de lucru și repozitoriul local ***figura 13***

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

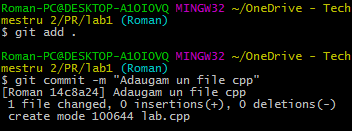


Figura 13. Adăugarea in depozitorilul local

1. Verificam dacă sunt modificări în repozitoriul distant  figura 14

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

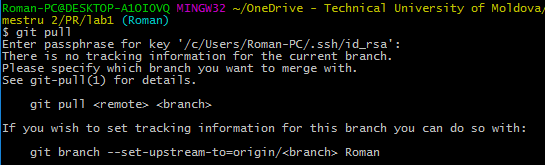


Figura 14. Monitorizarea schimbărilor la nivel de repozitoare

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

* ***git push -u origin Roman***

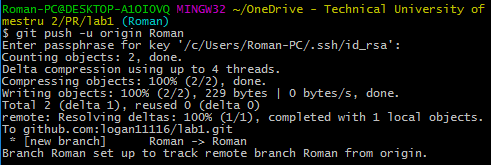


Figura 15. Adăugarea in repozitoriul extern

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

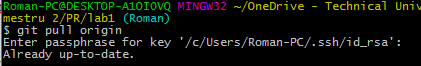
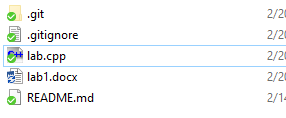


Figura 16. Monitorizarea schimbărilor

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

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



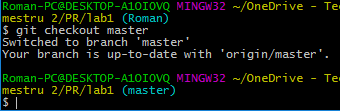


Figura 17. Mergerea la nouă ramură

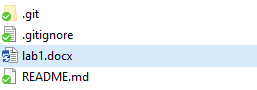
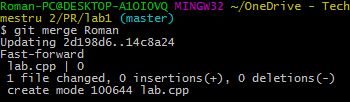
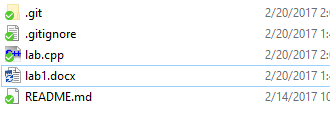


Figura 18. Schimbările apărute in repozitoriu

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

* git merge***Roman***





**Figura 19. Unificarea ramurilor**

1. Reînnoim repozitoriul distant **origin,**ramura **master**(smilar p. 8)

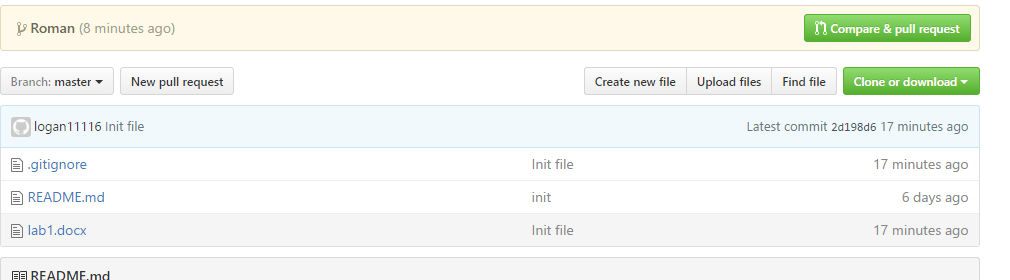


Figura 20. Schimbările in ramura master

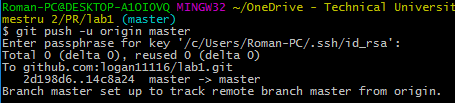


Figura 21. Adăugăm fișiere în dipozitoriul extern

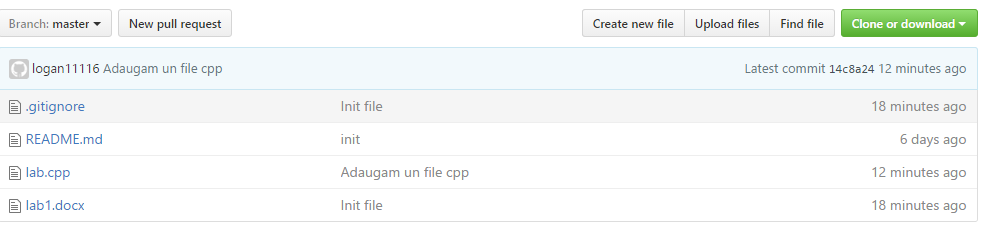
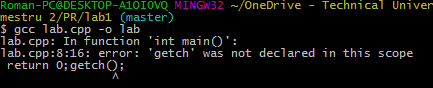
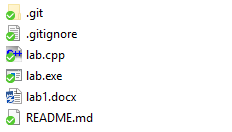


Figura 22. Schimbările survenite in repozitoriu

1. Observa modificările realizate în proiectul localizat în <https://github.com/logan11116/lab1>
2. Compilăm un fișier cpp in git bash figura 23.



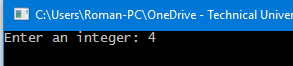


Figura 23. Compilarea, lansarea în execuție

### **Construirea proiectelor Java utilizînd Apache Maven**

Obiectivele lucrării: înțelegerea procesului de construire automată a uni 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.

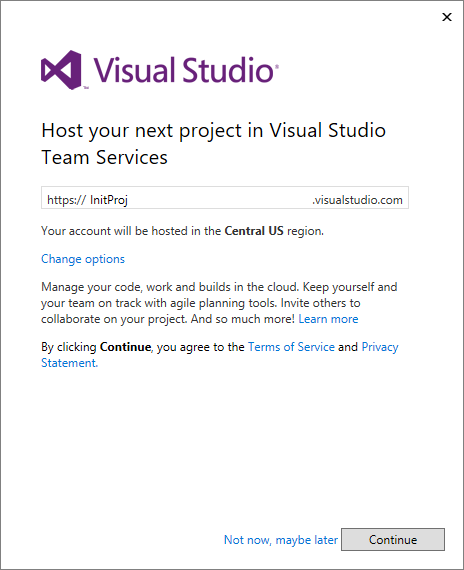


Figura 24. Crearea unui dipozitoriu VS

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

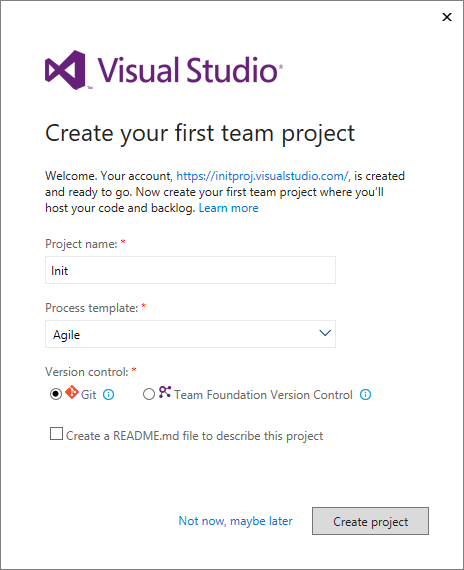


Figura 25. Crearea unui project

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

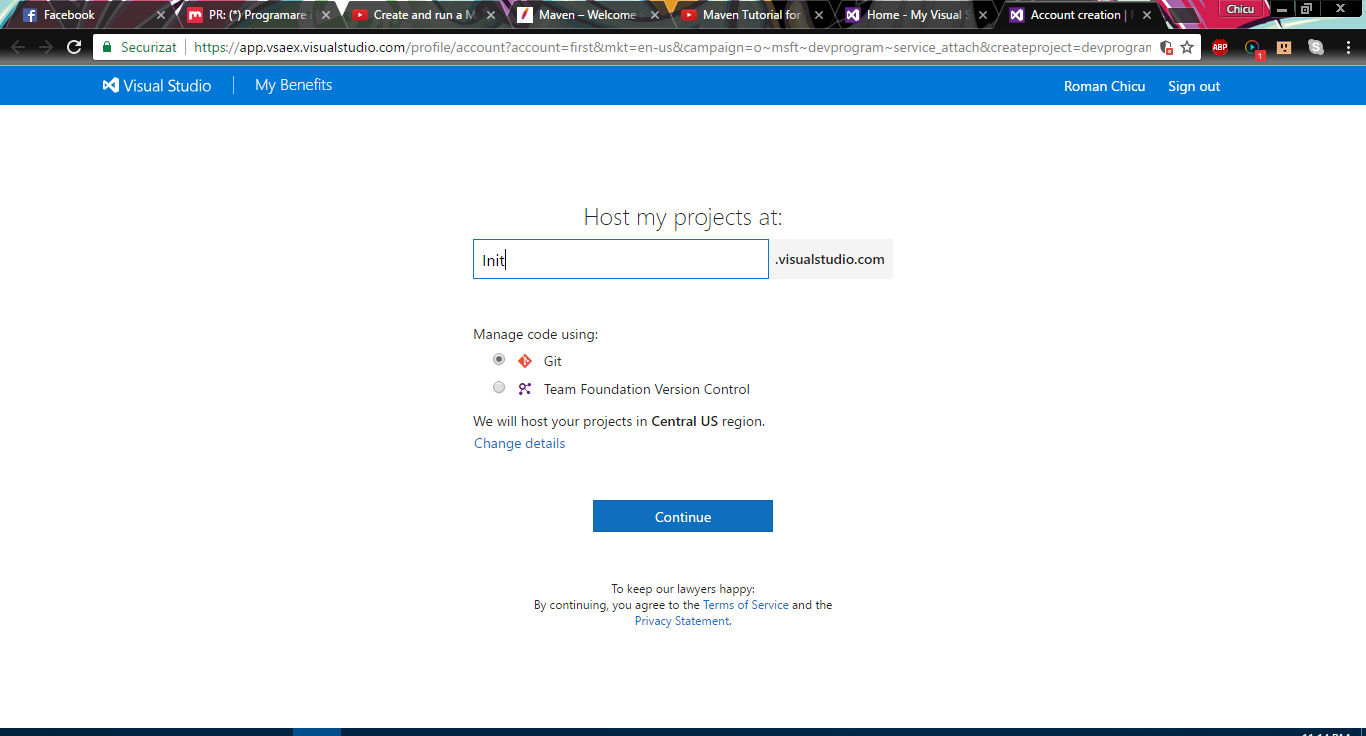


Figura 26. Crearea unui nume de host project

După iniațilizare a unui proiect crem un fișier Maven figura 4. Cu reprezentarea reprozitoriului și metoda de conlucrare a acestuia figura 27.

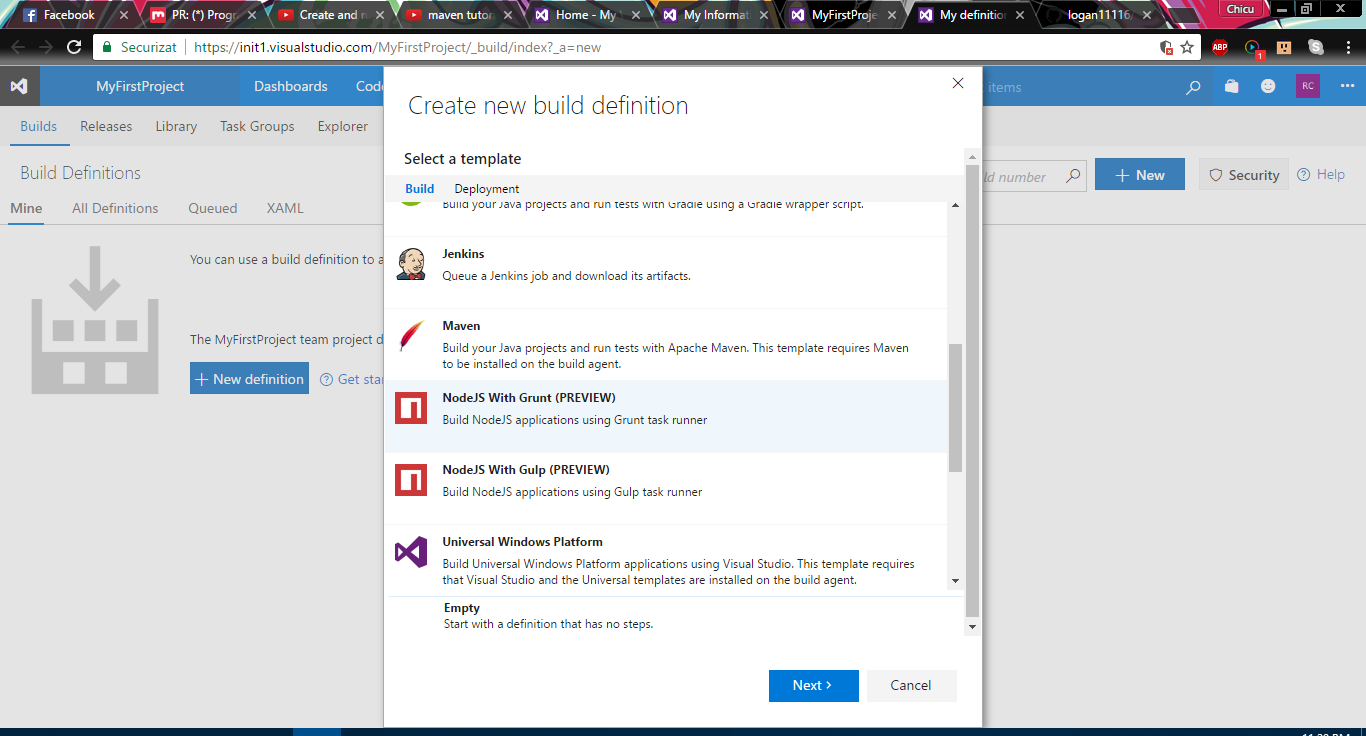


Figura 27. Creare unui fișier Maven

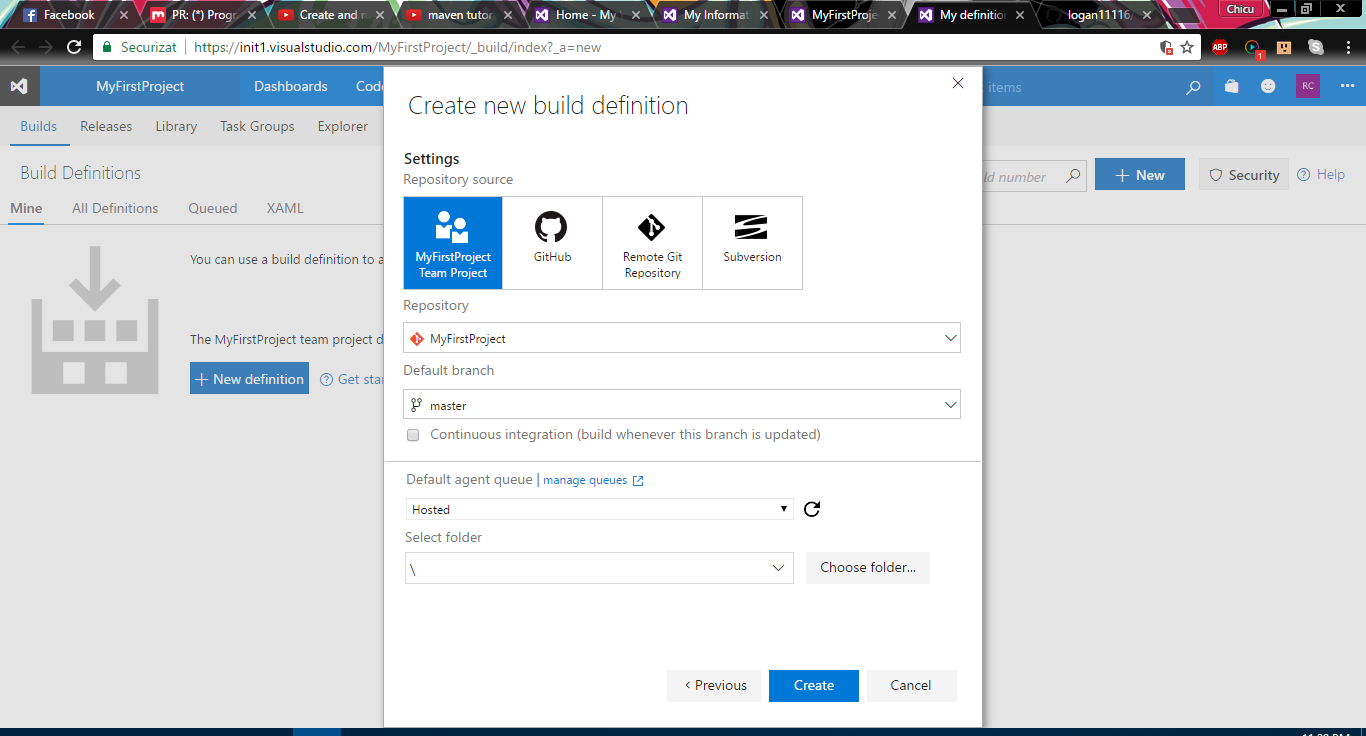


Figura 28. Init project reprository

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

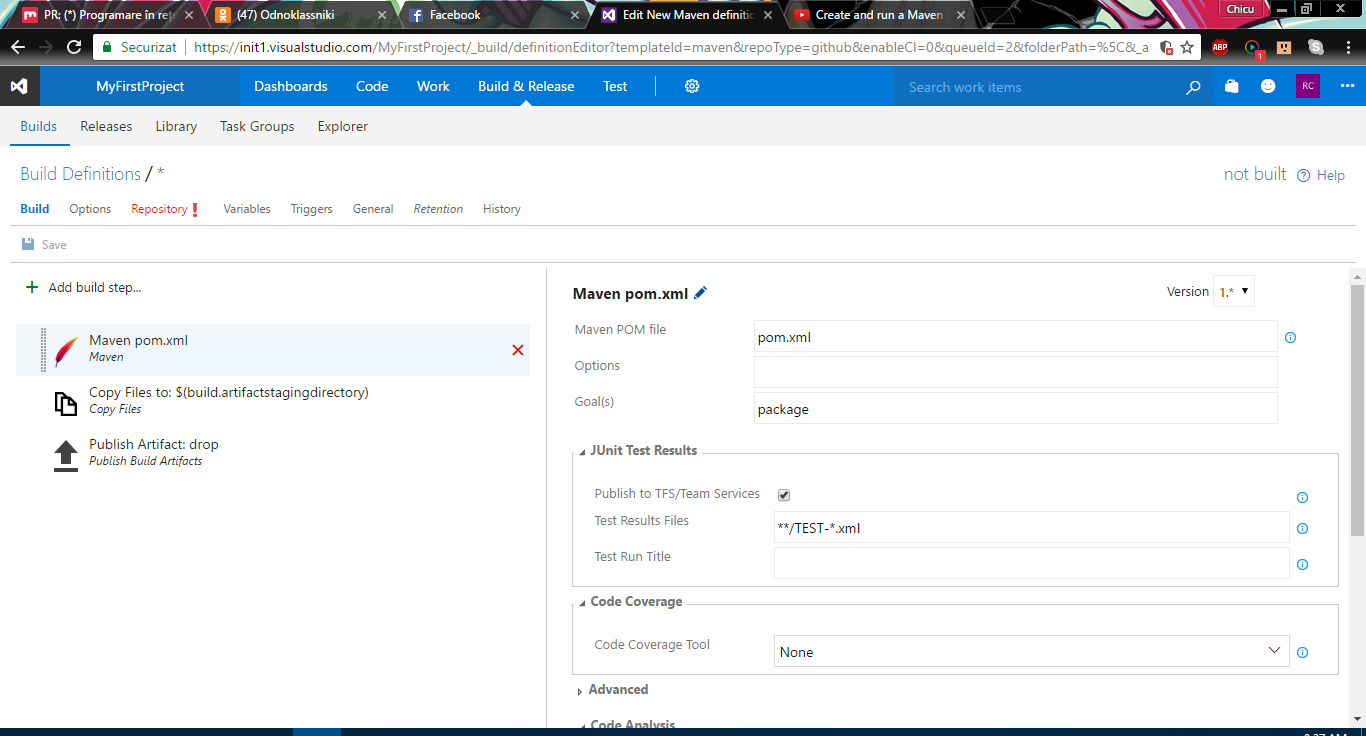


Figura 29. Configurarea unui Pub. Artifact și Maven file

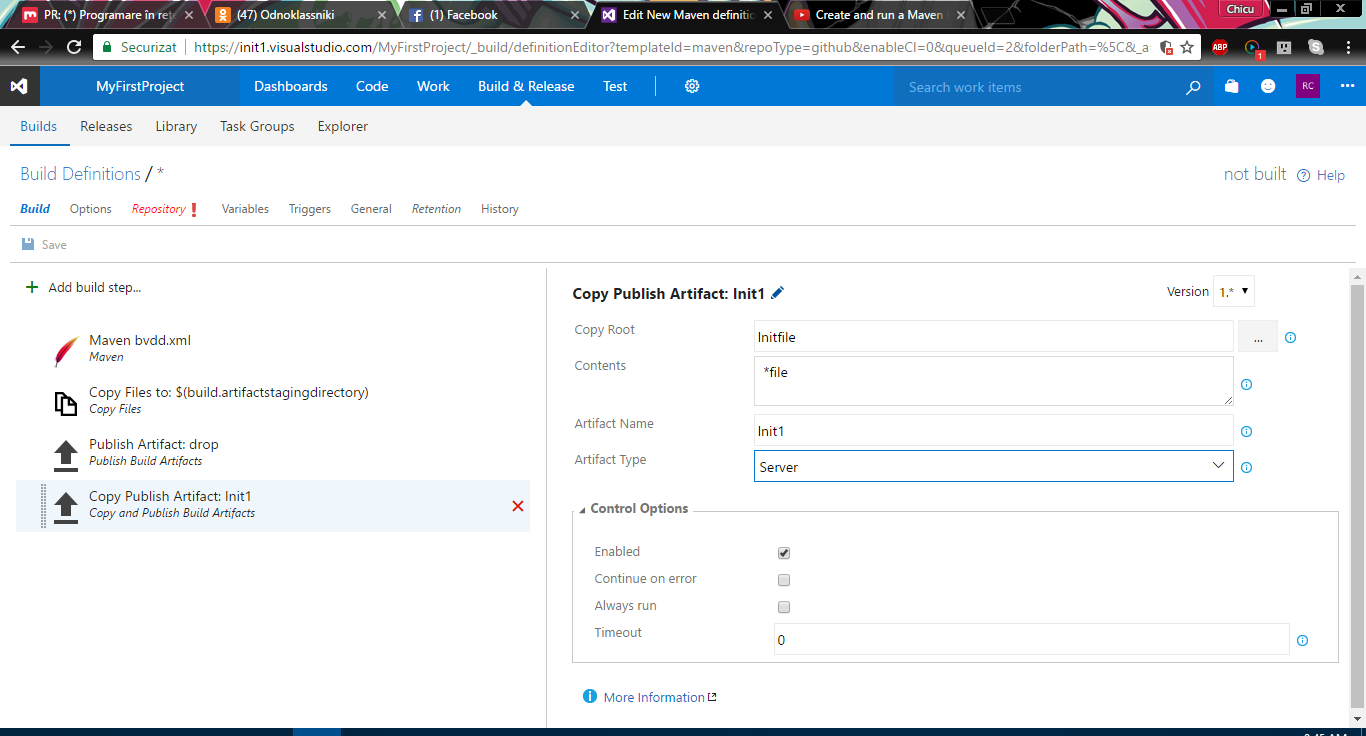


Figura 30. Crearea unui task

Facem build la project asemănător ca în figura 31.

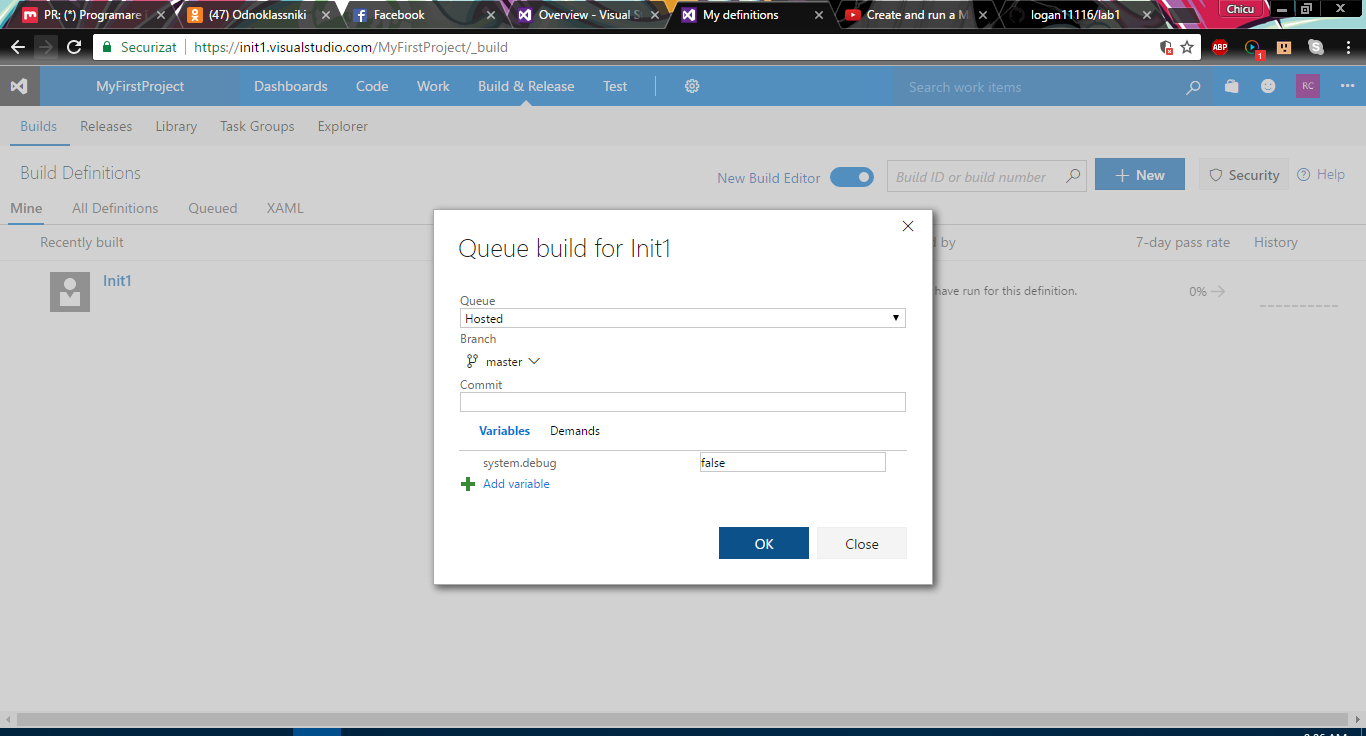


Figura 31. Build project

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

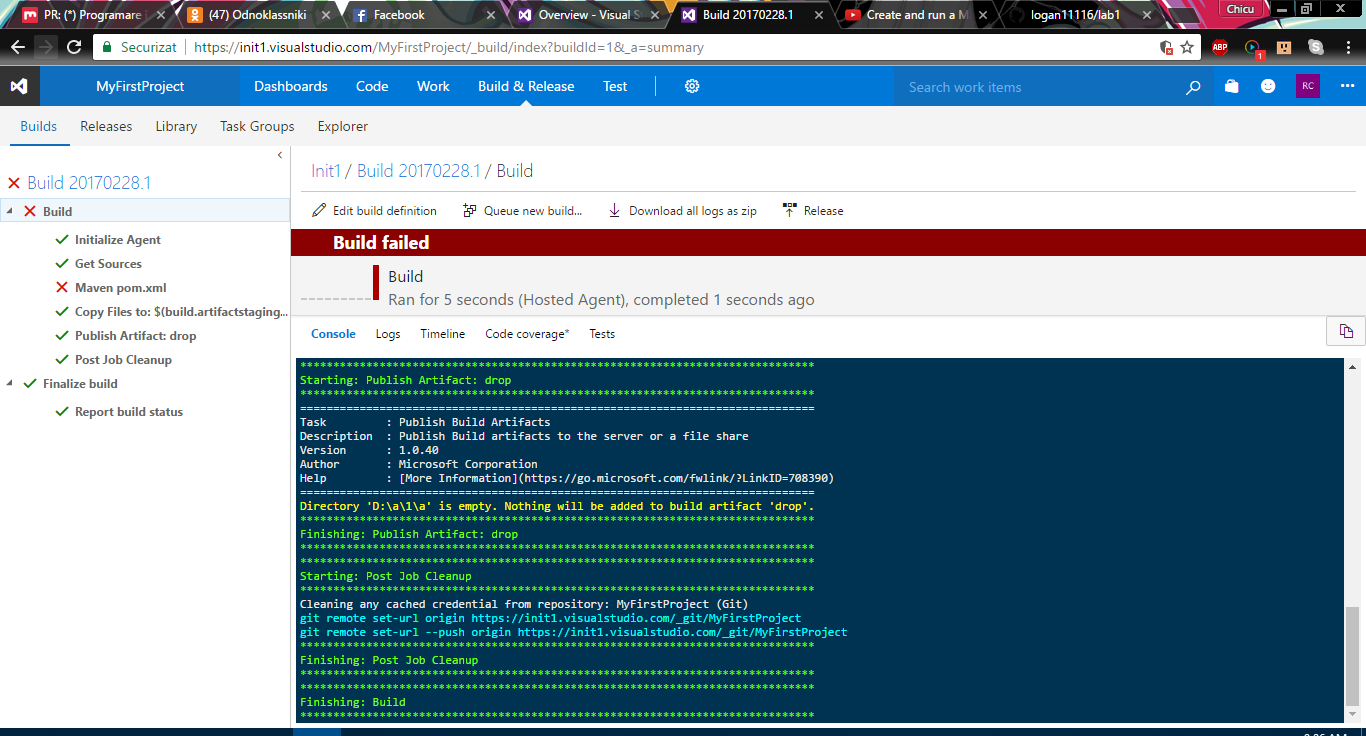


Figura 32. Rezultatul buildulu

### **Concluzie**

Ni-am făcut familiari cu mediul github și înspecial am lucrat cu consola git bash în care am creat un dipozitoriu local și l-am importat într-un extern. Am studiat diverse comenzi git și importanța lor intr-un mediu unix. Am compilat un cod sursă in 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](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>