Ministerul Educaţiei al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică şi Microelectronică
Catedra Automatică şi Tehnologii Informaţionale

Raport

Lucrare de laborator nr. 1
la disciplina: *Programarea în rețea*

Tema: Versionarea codului sursă utilizând GIT

A verificat: lector asistent Ostapenco S.

A efectuat: st. gr. TI-142 Cornita Constantin

Chișinău 2017

Obiectivele lucrării

Crearea unui repozitoriu distant, localizat de serviciul github, și sincronizarea tuturor modificărilor efectuate asupra repozitoriului local.

Sarcina lucrării

Lucrarea de laborator are ca scop studiul și înțelegerea principiilor de funcționare și utilizare a sistemului distribuit de control al versiunilor numit GIT, utilizând repozitoriul creat- https://github.com/CornitaConstantin/PR.git

Introducere

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 autorul schimbării. Important de menționat că în orice moment de timp se poate reveni la o versiune anterioară a entității.

Motivația principală constă în posibilitatea ca diferiți membri ai echipei, aflați eventual în spații geografice îndepartate, sa poata lucra simultan la proiect, urmând ca, la final, modificarile lor sa fie reunite în noi versiuni ale proiectului. De asemenea, există și alte avantaje. Când se observă un bug, se poate reveni la o versiune anterioară, în vederea determinării momentului introducerii acestuia în program. În același timp, se poate urma o dezvoltare pe ramuri (branches), în care se lucrează, în paralel, la multiple versiuni ale proiectului- de exemplu, una în care se dorește înlaturarea bug-urilor, iar cealaltă, în care se urmărește adăugarea de noi functionalitați, înaintea șlefuirii celor existente [1].

Există două modele de VCS-uri.

Modelul centralizat (ex: SVN): codul sursa este situat pe un server central, de unde clienţii pot obţine variante de lucru pe maşina locala (***working copy***). După efectuarea locală a modificărilor, dezvoltatorul solicită actualizarea variantei de pe server.

Avantajele utilizării modelului SVN sunt portabilitatea și suportul de integrare cu numeroase IDE-uri ca, de exemplu, Eclipse [2].

Principalele dezavantaje ale utilizării modelului centralizat SVN sunt:

- dependența de accesul la server;
- mentenanța și backup-urile serverului;
- dificultăți de comunicare cu serverul;

- dificultatea de a utiliza instrumentele pentru gestionarea ramurilor și de unificare a ramurilor.

Modelul distribuit (ex: GIT): nu există un server central, procesul de sincronizare desfășurându-se la nivel peer-to-peer.

Principalele beneficii ale sistemelor GIT și Mercurial sunt:

- urmărirea schimbărilor mai avansată și mai detaliată, lucru care duce la mai puține conflicte;
- lipsa necesității unui server, toate operațiile cu excepția schimbului de informații între repozitorii se realizează local;
- operațiile pentru gestionarea ramurilor și de unificare a ramurilor (prin intermediul comenzii `merge`) sunt mai sigure, și prin urmare folosite mai des;
- rapiditatea marea a operațiilor datorită lipsei necesității comunicării cu serverul.

Dezavantajele utilizării sistemelor GIT și Mercurial:

- modelul distribuit este mai greu de înțeles;
- nu există așa de mulți clienți GUI datorită faptului că aceste sisteme sunt mai noi;
- reviziile nu sunt numere incrementale, lucru care le face mai greu de referențiat;
- riscul apariției de greșeli este mare dacă modelul nu este familiar.

Chei SSH

SSH este un protocol de rețea care asigură o comunicare securizată a datelor între două stații (calculatoare, tablete, telefoane sau alte dispozitive dintr-o rețea). Pentru a asigura confidențialitatea și integritatea informațiilor interschimbate, SSH se folosește de criptarea cu chei asimetrice. Criptografia asimetrică este un tip de criptografie care utilizează o pereche de chei: o cheie publică și o cheie privată. Un utilizator care deține o astfel de pereche își publică cheia publică astfel încât oricine dorește să o poată folosi pentru a îi transmite un mesaj criptat. Numai deținătorul cheii secrete (private) este cel care poate decripta mesajul astfel criptat `[3]`.

Utilizatorul deține o pereche de chei: una publică și una privată. În timp ce cheia publică se trimite stației de la distanță (eng. *remote*) cu care se dorește comunicarea, cea privată rămâne tot timpul pe stația locală și trebuie protejată de public. Cheia publică trebuie trimisă stației de la distanță, pentru că aceasta să o poată folosi la decriptarea datelor primite în format securizat.

Fingerprint: `1c:8a:33:39:27:26:4c:ce:bd:ef:9c:b6:55:ff:05:e1`

Comenzi GIT utilizate

În cadrul acestei lucrări de laborator, pentru realizarea sarcinilor propuse, care vizează versionarea codului sursă la dezvoltarea unui proiect, au fost utilizate o serie de comenzi absolut necesare pentru realizarea scopului acestei lucrări.

Pentru a gestiona eficient dezvoltarea unui proiect în cadrul sistemului de versionare GIT, este necesar de a crea un repozitoriu distant și unul local. Pentru a inițializa un repozitoriu local GIT, într-un directoriu, se utilizează comanda `git init`. Inițial se crează un repozitoriu gol. În cadrul acestui repozitoriu se vor păstra fișierele necesare, local.

Odată ce a fost creat un repozitoriu local, utilizatorul are posibilitatea de a configura numele și adresa poștei electronice a acestuia. Această informație este inclusă în cadrul fiecărui `commit` realizat în cadrul dezvoltării proiectului și ulterior nu poate fi modificată. Aceste setări sunt suficient de realizat doar o singură dată, la crearea repozitoriului local. Ele vor fi utilizate de sistemul GIT pentru toate operațiunile pe care utilizatorul le realizează în cadrul acestuia. Sintaxa acestor comenzi sunt: `git config --global user.name "User Name"` - pentru a seta numele utilizatorului, `git config --global user.email user@gmail.com` - pentru a seta poșta electronică a utilizatorului. În cazul în care, pentru dezvoltarea altui proiect este necesar de specificat alte date personale privind poșta electronică sau nume utilizatorului, se poate executa aceiași comandă fără parametrul -global, în catalogul cu proiectul necesar.

În cadrul repozitoriului GIT, având o copie a fișierelor de lucru pentru un anumit proiect, urmează să se realizeze anumite modificări în cadrul fișierelor acestora. Modificările vor fi fixate în așa numite snapshots a acestor modificări în repozitoriu, de fiecare dată, când proiectul ajunge la starea, la momentul căreia proiectul ar trebui fi salvat.

Un fișier în repozitoriul local poate să se afle în două stări: `tracked` și `untracked`. Fișierele aflate în starea `tracked`, sunt fișierele care în ultimul snapshot, ele pot fi nemodificate, modificate sau pregătite pentru `commit (aflat în starea staged)`. Fișierele aflate în starea untracked sunt toate celelalte fișiere, oricare fișiere în repozitoriu, care nu fac parte din ultimul snapshot și nu este pregătit pentru commit. Odată cu clonarea proiectului de pe repozitoriul distant, toate fișierele se vor afla în starea tracked și nemodificate, deoarece ele sunt importate din repozitoriu distant și până când ele nu au fost redactate. Atunci când aceste fișiere vor fi redactate, GIT le va considera în calitate de fișiere modificate, deoarece au fost modificate după realizarea ultimului commit. Ciclul de viață a fișierelor în perioada dezvoltării și gestionării unui proiect sunt este reprezentat în figura de mai jos.

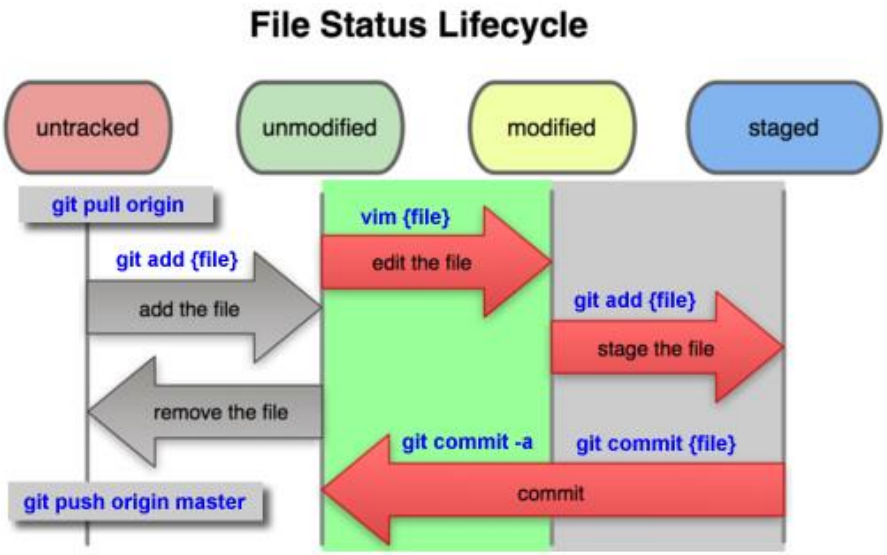

Figura 1- File Status Lifecycle

Pentru a determina starea fișierului, se utilizează comanda de bază `git status`. La fel această comandă arată la moment în ce ramură se modifică fișierele.

Pentru a urmări fișierele care au fost modificate și adăugarea acestora în cadrul controlului de versionare a codului, se utilizează comanda `git add`. Fișierele urmărite vor fi indexate, aflându-se în secțiunea `Changes to be committed`. Dacă se va realiza un `commit` la acest moment, versiunea fișierelor proiectului care există la momentul executării comenzii `git add`, se va adăuga în istoria stărilor snapshots.

Odată ce fișierele modificate au fost indexate prin intermediul comenzii precedente, se pot fixa toate modificările. Metoda cea mai simplă de fixare a modificărilor în cadrul fișierelor proiectului, este de a executa comanda `git commit -m "first commit"`. GIT crează un `commit` cu mesajul spcificat. La fixarea modificărilor fișierelor în cadrul repozitoriului local, se raportează anumite informații despre denumirea ramurii în cadrul căruia s-a efectuat `commit`, câte fișiere au fost modificate, statistica rândurilor adăugate sau șterse în cadrul fișierelor, identificatorul unic al acestui `commit`. Fișierele care nu sunt indexate nu vor nimeri în snapshots. Această comandă permite crearea unui snapshot a stării proiectului la momentul dat de timp, care ulterior ar putea fi restabilit sau comparat cu versiunea actuală a proiectului.

Pentru a avea posibilitatea de a lucra cu un proiect GIT, se utilizează repozitorii distante. Un repozitoriu distant se păstrează în rețea. Ele pot fi mai multe, fiecare dintre ele de regulă, ar trebui să fie disponibil sau doar pentru citire, sau pentru citire și înscriere. Pentru a obține accesul la un repozitoriu distant, se utilizează comanda `git remote add origin [URL]`.

Pentru a distribui fișierele în repozitoriul distant, se utilizează comanda `git push -u [serverul distant][ramura]`. Această comandă se va executa numai în cazul în care accesul la repozitoriul distant nu este restricționat, și utilizatorul are drepturi de a efectua modificări în repozitoriu.

Sistemul de versionare GIT permite de a lucra la același proiect din mai multe ramuri. Pentru a crea o ramură și de a trece imediat în cadrul acesteia se utilizează comanda: `git checkout -b "NewBranch". Astfel orice modificare în fișierele proiectului vor fi realizate în ramura în care se află sistemul la momentul actual. Aceste modificări nu vor afecta starea sau conținutul fișierelor din alte ramuri.`

Aceste ramuri pot fi într-un final unite, adică modificările efectuate, să fie adăugate în ramura de bază. Aceasta se efectuează prin intermediul comenzii `git merge NewBranch`. La executarea acestei comenzi ar putea apărea eventuale conflicte, care specifică că aceeași secvență de cod sunt modificate diferit în cadrul celor două ramuri. Odată cu soluționarea acestor conflicte, cu succes se poate crea un snapshot, efectuând un `commit [4]`.

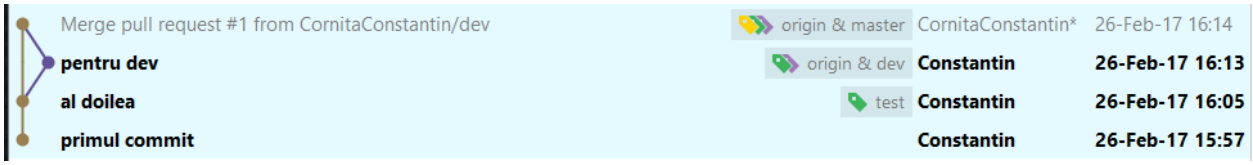

Figura 2 – rezultatul

In imaginea de mai sus (Figura 2) vedem reprezentarea grafica a comenzilor realizate.

Concluzii

Sistemul de versionare GIT este un mod eficient de management al fișierelor care permite păstrarea istoricului tuturor modificărilor aduse fișierelor urmărite. Unul dintre avantajele folosirii sistemului GIT este posibilitatea de a reveni la o versiune mai veche a proiectului. Utilizatorul dispune de o copie de siguranță a codului sursă. Cea mai recentă versiune a codului sursă este mereu disponibilă tuturor dezvoltatorilor. GIT simplifică procesul de distribuire a codului sursă și să înlesnește colaborarea pe proiecte.

Bibliografie

1 Scott Chacon, Pro Git [Resursă electronică]. – Regim de acces: http://git-scm.com/book

2 Lars Vogel, Git - Tutorial [Resursă electronică]. – Regim de acces: http://www.vogella.com/tutorials/Git/article.html

3 Atlassian, Git Tutorials [Resursă electronică]. – Regim de acces: https://www.atlassian.com/git/tutorial

4 Vincent Driessen, A successful Git branching model [Resursă electronică]. – Regim de acces: http://nvie.com/posts/a-successful-git-branching-model/