**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare Informatică şi Microelectronică**

**Departamentul Ingineria Software și Automatică**

Disciplina: *Instrumente de dezvoltare în web*

**Raport**

**Lucrare de laborator nr.1**

**Tema :** "Interacțiunea cu sistemul de control al versiunilor Git.

Utilizarea hostingului de repozitorii Github."

**A efectuat : Danilenco Mihail st.TI-206**

**A verificat : asist.univ. Buldumac Oleg**

**Chișinău 2022**

**Scopul lucrării:** Însușirea sistemului de control al versiunilor GIT, utilizarea GitHub

**Considerații teoretice:**

**Git** reprezintă la momentul actual, unul dintre cele mai populare sisteme ce pot să fie folosite pentru a păstra o evidență a modificărilor aduse într-un proiect. Oferă de asemenea un mod de a ne întoarce la o anumită versiune, în cazul în care apar diverse probleme în momentul în care se fac actualizari la nivel de cod.

Evidența modificărilor aduse se face la nivel de fișier individual, nu doar la nivel de proiect, iar acest lucru facilitează și lucrul în echipă. Mai mult decât atât, în momentul în care cineva vrea să aducă o actualizare într-un fișier, aceasta poate să fie revizuită de ceilalți membrii ai echipei înainte de a ajunge în versiunea principală a proiectului.

Cum putem folosi Git?

Ulterior instalării Git pe sistemul nostru acesta poate să fie folosit direct din linia de comandă, dar avem și posibilitatea de a instala alte aplicații care să ofere o interfață grafică. Funcționalitatea în sine este aceeași, indiferent de modul de utilizare, dar cu ajutorul interfeței grafice putem să vedem mai clar operațiile pe care urmează să le realizăm și rezultatul acestora. Câteva exemple de aplicații ce pot fi instalate ”peste” Git sunt: TortoiseGit, SourceTree, SmartGit etc.

Funcțiile puse la dispoziție de Git pot de asemenea să fie integrate și în majoritatea IDE-urilor profesioniste: PyCharm, IntelliJ, PHPStorm, NetBeans etc. Acestea oferă în mod direct o interfață intuitivă pentru comenzile de Git.

Sistemul de versionare Git oferă posibilitatea de a lucra pe mai multe **branch**-uri (sau ramuri) ale proiectului principal. Acestea minimizează impactul problemelor ce pot să apară dacă un defect este introdus în aplicații, facilitând în paralel lucrul în echipă. Branch-urile reprezintă copii ale proiectului, realizate de cele mai multe ori din **main** (sau master, pentru proiecte mai vechi) – acesta fiind rădăcina proiectului în sine.

Dacă doi programatori lucrează în paralel la două lucruri diferite: o nouă funcționalitate și repararea unui defect existent, atunci fiecare dintre aceștia își va crea un branch propriu unde își va aduce modificările. Mai mult decât atât, dacă cei doi vor să colaboreze la un moment dat, pot trece pe același branch pentru a vedea modificările aduse până la un moment dat și să propună îmbunătățiri.

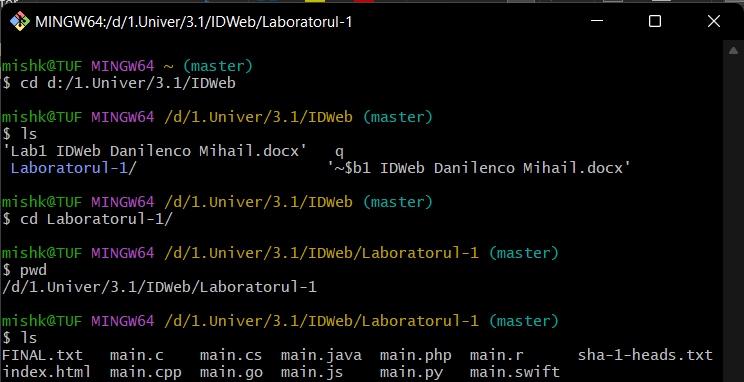
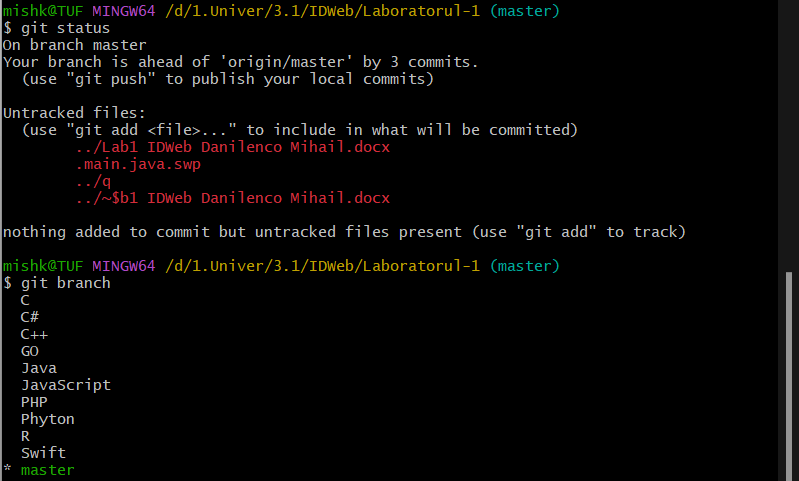
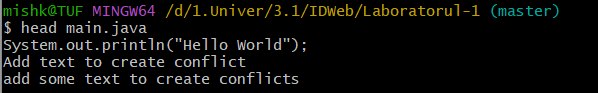
**Executarea lucrării**

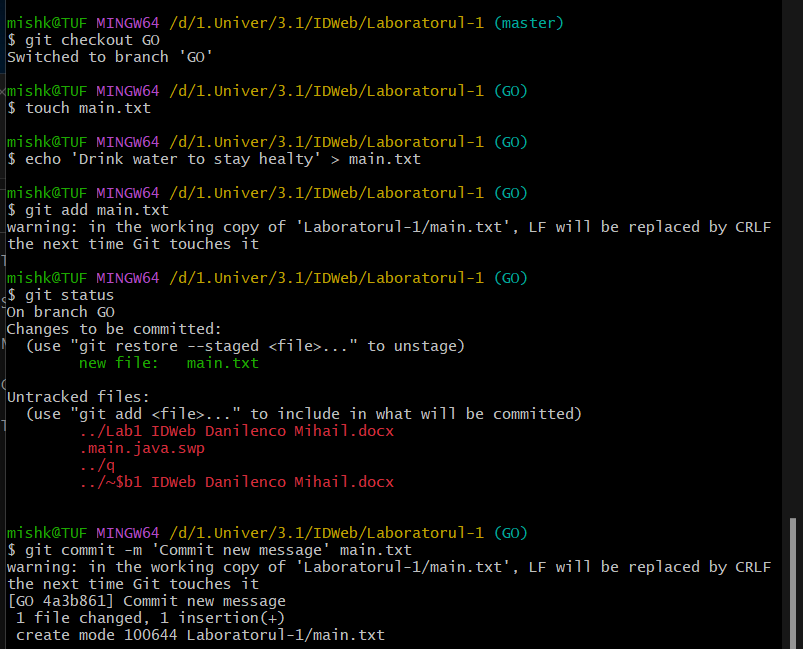
Comenzi CLI

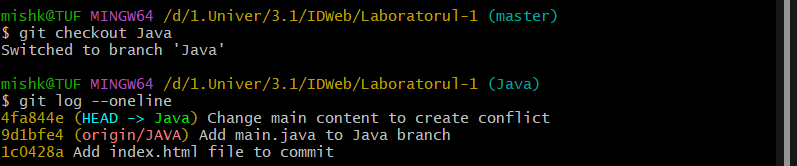
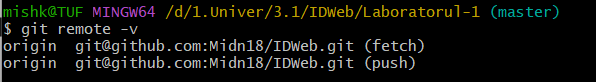
|  |  |
| --- | --- |
| **Comanda** | **Explicația** |
| **pwd** | Afișează locația în care ne aflăm |
| **ls** | Afișează conținutul directoriului în care ne aflăm |
| **cd** | Schimbăm directoriul |
| **mkdir** | Crează foldere |
| **touch** | Creaza fișiere |
| **less** | Afișează conținutul fișierului în consolă |
| **head** | Afișează primele 10 rânduri unui fișier text |
| **vim|nano** | Editori de text |
| **rm** | Elimină fișiere și directorii |

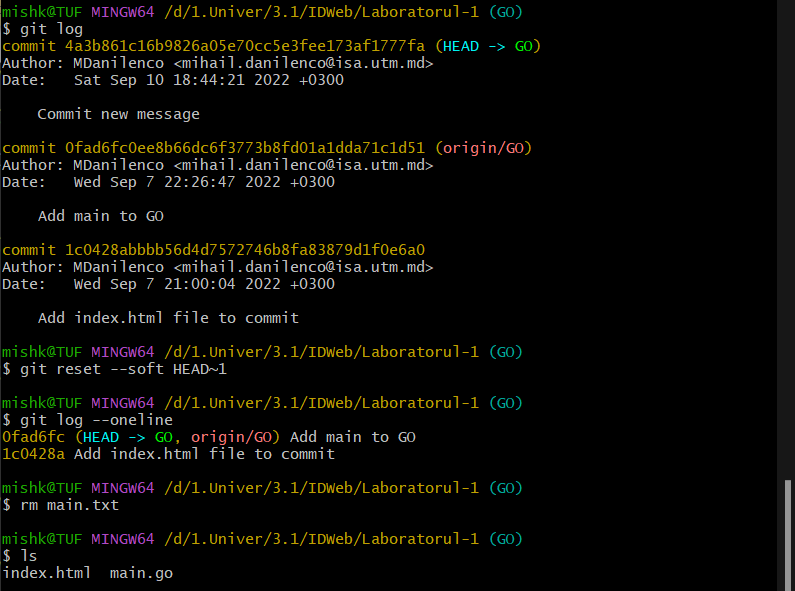
Comenzi GIT

|  |  |
| --- | --- |
| **Comanda** | **Explicația** |
| **git init** | Inițializează un repozitoriu nou |
| **git clone** | Clonează repozitoriu |
| **git status** | Afișează statutul branch-ului |
| **git log** | Afișează commit-urile efectuate |
| **git branch** | Afișează toate branch-urile |
| **git checkout** | Mutarea de pe un branch pe altul |
| **git merge** | Actualizarea unui branch cu altul |
| **git add** | Adaugarea unui fișier în git |
| **git commit** | Salvarea schimbărilor pe repozitoriul local |
| **git reset -hard** | Resetează branch-ul la un anumit commit ștergând modificarile |
| **git pull** | Actualizează branch-ul actual cu schimbarile de pe remote |
| **git push** | Trimite schimbările de pe local pe remote |
| **git remote** | Efectuează modificări cu repozitoriul remote |
| **git config** | Setează schimbări în git |

**Exemple de comenzi utilizate pentru realizarea lucrării de laborator**





**Concluzii**

Efectuând lucrarea de laborator nr.1 am avut ca scop familiarizarea cu git și GitHub și însușirea comenzilor.

Pentru efectuarea lucrării de laborator trebuia să creăm un repozitoriu local în care urma să efectuăm sarcinile propuse în lucrare. Am creat un folder unde ulterior am creat și editat diferite fișiere text. Aceste sarcini erau executate în cadrul repozitoriului local, însă pe diferite branch-uri. După efectuarea sarcinilor pe branch-uri separate, am finalizat cu actualizarea branch-ului principal – **master**. După actualizare, toate fișierele create pe branch-uri separate au fost aduse în master.

Am creat un repozitoriu remote pe [**GitHub**](https://github.com/Midn18/IDWeb)unde, prin intermediul consolei am transmis toate sarcinile efectuate pe repozitoriul local. Tot prin consolă și GitHub am setat conexiunea de tip **SSH**. În consolă am generat cheia de conectare și ulterior am încărcat cheia publică pe GitHub.

În concluzie pot spune că laboratorul a fost unul educativ în care am însușit așa un instrument important în domeniul IT – GIT.