

Dezvoltarea profesională a software-ului implică aproape întotdeauna existența unor echipe de dezvoltare, adică participarea unui număr mare de programatori, ceea ce este adevărat, mai ales atunci când vine vorba despre proiecte mai complexe. Prin urmare, cooperarea reciprocă, respectiv colaborarea, reprezintă un aspect foarte important al dezvoltării. Potrivit acestui fapt, de-a lungul timpului au fost dezvoltate numeroase instrumente care asigură colaborarea și facilitează partajarea codului. Pentru început, sunt șanse mari să fi auzit sau să vă fi întâlnit până în prezent cu platforma online GitHub. Cu siguranță este cea mai populară platformă pentru partajarea codului. În această lecție, printre altele, vom vorbi despre platforma GitHub. Totuși, pentru a ajunge la povestea respectivă, este necesar să înțelegem câteva concepte fără de care GitHub nu ar exista. Pentru început, ne vom familiariza cu noțiunea - versiune.

Conceptul versiune cod

Inima oricărui produs software este codul său sursă. Cantitatea de cod din care este alcătuit un program depinde de o serie de parametri, unde *linia* este folosită ca unitate stabilită pentru exprimarea cantității de cod. De exemplu, se poate spune că un program este alcătuit din 1354 de linii de cod. Programele mai simple dispun de mai puține linii de cod, în timp ce cele mai complexe pot avea zeci sau chiar sute de mii de linii de cod.

Pe măsură ce numărul de linii de cod din care este alcătuit un program crește în timpul dezvoltării, devine din ce în ce mai dificil să se urmărească modificările realizate asupra codului. Dacă se adaugă și munca în echipă, care presupune actualizarea aceluiasi cod de către un număr de programatori, situația devine și mai complicată. Pentru a se organiza munca programatorului și facilita întreținerea, pe parcurs au fost create numeroase sisteme privind versiunea codului (*version control systems*, în limba engleză). Pe lângă această denumire de bază, aceste sisteme sunt adesea numite și *revision control* și sisteme *source control*.

Versiunea codului este o abordare care permite monitorizarea modificărilor codului sursă. Mai simplu spus, versiunea permite ca

fiecare modificare a codului să fie înregistrată și să se permită revenirea la o astfel de versiune în orice moment. Pentru a înțelege mai bine acest lucru, vom prezenta un exemplu. Persoana care a comandat programul solicită o modificare a unei anumite funcționalități care necesită corectarea codului sursă. Codul nou este scris peste codul existent, iar clientul decide că dorește funcționalitatea veche. Dacă nu se folosește un sistem privind versiunile, programatorul trebuie din nou să scrie codul vechii funcționalități. Pe de altă parte, cu un sistem privind versiunile, procesul de revenire la o versiune veche presupune doar câteva clicuri.

Datorită utilizării unui sistem de versiuni, programatorii pot accesa în orice moment informații importante privind trecutul proiectului. Așadar, pot vedea:

- toate modificările care au fost efectuate,
- cine a întreprins modificările,
- când au avut loc modificările,
- motivul pentru care au fost întreprinse modificările.

În prezent, versiunea se atinge folosind numeroase sisteme, adică programe de calculator. Două astfel de programe, numite Git și SVN, sunt în frunte.

Ce reprezintă Git și SVN?

În momentul de față, cel mai cunoscut sistem privind versiunea, cu siguranță este Git. Este vorba despre un sistem care a fost creat de fondatorul Linux kernel - Linus Torvalds, lucrând tocmai la cel mai popular produs al său.



Imaginea 10.1. Git

Datorită simplității, vitezei și faptului că este un software open source, pe parcurs, Git a câștigat o mare popularitate. Se folosește în combinație cu toate tipurile de dezvoltare, adică pentru proiecte de diferite dimensiuni, de la programe simple cu care muncește un singur programator, până la sisteme software complexe, care implică echipe mari de dezvoltare.

SVN este un alt sistem popular de versiuni. Numele său complet este Subversion, SVN fiind abrevierea, un nume alternativ (imaginea 10.2.).



Imaginea 10.2. SVN

SVN se folosește pentru întreprinderea unor activități similare precum Git. Cu toate acestea, există diferențe subtile între aceste două sisteme. Principala caracteristică care distinge SVN de Git este centralizarea. Acest lucru înseamnă, practic, că SVN dispune de un server central unde se află fișierele de proiect. Stațiile de lucru sunt clienții care, la începutul activității, descarcă fișierele de pe server, iar apoi trimit modificările codului către depozitul central. La sistemul Git nu există o astfel de centralizare, fiecare client poate fi un server în același timp.

Practic, datorită setului identic de funcționalități ale acestor două sisteme la nivelul lor de bază, în continuare noi ne vom familiariza doar cu sistemul Git.

Instalarea și utilizarea sistemului Git

Git este în întregime gratuit pentru utilizare. În funcție de sistemul de operare, procedura de instalare a acestuia diferă.

Pe Linux, este suficient să rulați următoarea comandă:

```
$ sudo apt install git-all
```

Pe macOS versiunea 10.9. și altele mai recente, Git poate fi instalat rulând următoarea comandă pe Terminal:

```
$ git --version
```

Pentru versiunile mai vechi de macOS, Git poate fi descărcat de la următoarea adresă:

<https://git-scm.com/download/mac>

În final, instalarea sistemului Git pentru Windows poate fi descărcată de la următoarea adresă:

<https://git-scm.com/download/win>

Notă

Dacă încercați să instalați Git pe sistemele de operare Windows sau macOS, folosind fișierele de instalare care se află la adresele web tocmai menționate, astfel de fișiere de instalare trebuie să fie pornite după descărcare, iar progresul instalării trebuie monitorizat. Este necesar să utilizați setările implicite de instalare, adică nu este necesar să faceți nicio modificare a setărilor implicite pe parcursul instalării. După ce instalarea este finalizată, veți avea ocazia să încercați comenzile din Git care sunt menționate în continuare.

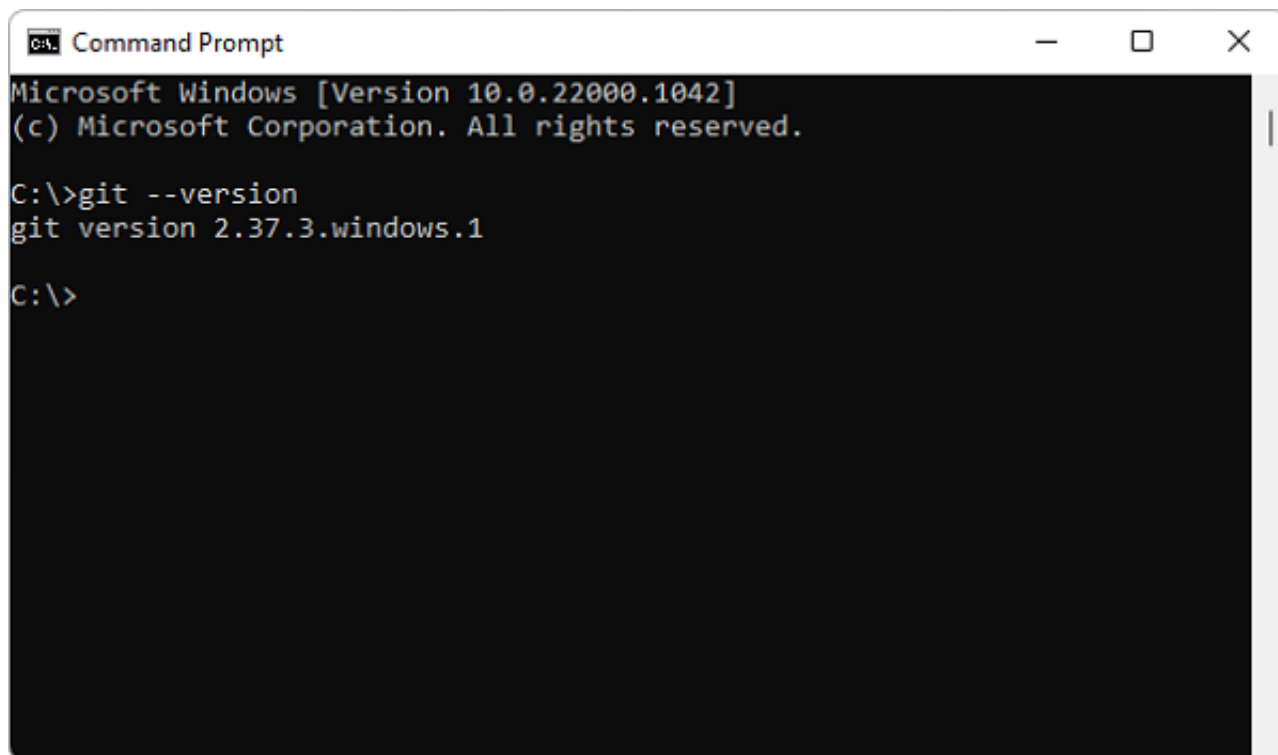
Verificarea disponibilității sistemului Git

Git a fost conceput și creat ca un instrument fără interfață cu utilizatorul. Prin urmare, utilizarea acestuia presupune folosirea unei console sau a unui terminal, în funcție de sistemul de operare care se utilizează. Desigur, din cauza absenței unui mediu grafic de utilizator, de-a lungul timpului, a fost creat un număr mare de așa-numiți clienți Git, adică programe care facilitează utilizarea Git. Am decis să încercăm Git în forma sa originală/de bază, astfel încât să puteți înțelege în mod corect modul de funcționare.

Funcționarea sistemului Git se bazează pe emiterea diferitelor comenzi de tip text. De exemplu, pentru început, vom emite o comandă pentru a verifica existența Git:

`git --version`

O Aceasta este comanda prin care se verifică existența sistemului Git, precum și versiunea (imaginea 10.3.).



```
Command Prompt
Microsoft Windows [Version 10.0.22000.1042]
(c) Microsoft Corporation. All rights reserved.

C:\>git --version
git version 2.37.3.windows.1

C:\>
```

Imaginea 10.3. Verificarea existenței sistemului Gi

În imaginea 10.3. este ilustrată emiterea primei comenzi Git pe sistemul de operare Windows, folosind programul Command Prompt. Țineți cont de faptul că este necesar să fie instalat Git, iar locația sa este plasată în variabila de sistem PATH. Instalând Git pentru Windows de la adresa URL menționată mai devreme, acest lucru se întreprinde în mod automat. În cazul în care Git este prezent pe sistem, se primește un mesaj cu conținut asemănător cu cel care se vede în poză - *git version 2.37.3.windows.1*. Desigur, mesajul diferă în funcție de versiunea instalată.

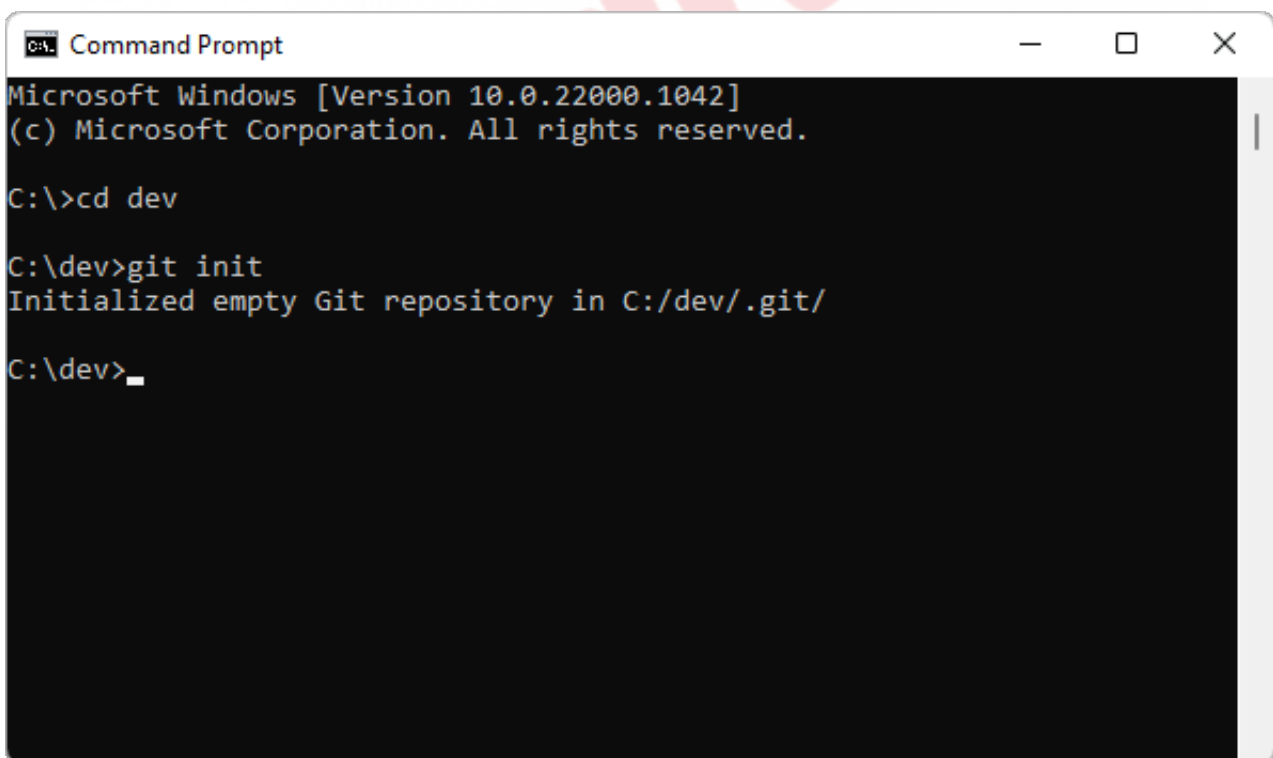
Depozite Git

Modul în care funcționează Git se bazează pe noțiunea de **depozit**. Depozitul reprezintă figura centrală a sistemului Git. Uneori se numește proiect Git și reprezintă o colecție completă de fișiere și foldere care alcătuiesc un proiect software.

Un depozit Git se creează prin emiterea următoarei comenzi:

`git init`

Emiterea acestei comenzi trebuie întreprinsă din folderul în care se dorește să fie creat depozitul (imaginea 10.4.).



```
Microsoft Windows [Version 10.0.22000.1042]
(c) Microsoft Corporation. All rights reserved.

C:\>cd dev

C:\dev>git init
Initialized empty Git repository in C:/dev/.git/

C:\dev>_
```

Imaginea 10.4. Crearea depozitului Git

Emiterea comenzii `init`, Git în fundal, în folderul care devine depozit se creează un folder special cu denumirea `.git`. Toate metadatele necesare pentru funcționarea depozitului sunt plasate în folderul

respectiv. Fișierul menționat este ascuns pe majoritatea sistemelor de operare și utilizatorul nu are nevoie să se ocupe direct de el

Prin emiterea comenzii `init`, Git creează și ceva ce se numește ramură (*branch*, în limba engleză). Fiecare depozit trebuie să dispună de cel puțin o ramură și, dacă este necesar, de un număr mai mare de ramuri denumite arbitrar.

Ramificarea sistemului Git

Într-un depozit Git pot exista un număr mare de ramuri (*branches*, în limba engleză). Git creează în mod automat prima ramură care poartă denumirea de ramură principală/master (*master branch*, în limba engleză). Programatorul are posibilitatea să creeze un număr arbitrar de ramuri suplimentare.

Ideea principală a ramificării este posibilitatea dezvoltării paralele a mai multor versiuni diferite ale unui singur produs software. De exemplu, o ramură poate servi ca principală și cea de-a doua pentru testare. Git permite crearea de noi ramuri, dar și împreunarea a două sau mai multor ramuri într-una singură.

Staging și committing

După crearea depozitului Git, este necesar să plasați în el fișierele cu codul sursă al proiectului la care se lucrează. Plasarea fișierelor și folderelor în depozit se efectuează prin intermediul a doi pași:

- **staging** – poate fi pasul pregătitor,
- **committing** – acțiunea care inițiază înregistrarea modificărilor în depozit, prin care numai modificările acelor fișiere care se află în starea de *staging*, adică asupra cărora *staging-ul* a fost efectuat anterior, sunt înregistrate în depozit.

Motivul existenței unui astfel de mecanism este foarte simplu. Mai precis, proiectele constau, de obicei, dintr-un număr mare de fișiere. Uneori nu veți dori să înregistrați în depozit modificările din toate fișierele. În astfel de situații, acest pas intermediar (*staging*) este mai mult decât util, deoarece permite determinarea cu precizie a modificărilor care vor fi introduse în depozit.

Staging-ul, adică pregătirea pentru livrare, se întreprinde în modul următor:

```
git add PythonHelloWorld.py
```

În acest fel, fișierul PythonHelloWorld.py este plasat într-o stare staging, unde totul este pregătit pentru înscrierea lui în depozit.

Commit, respectiv livrarea sau finalizarea, se întreprinde folosind următoarea comandă:

```
git commit -m "Initial Commit"
```

Pe lângă textul comenzii, comanda prezentată conține și descrierea modificărilor întreprinse în cod (*Initial Commit*). Acest lucru vă va ajuta ulterior să vă regăsiți mai ușor în versiuni diferite ale aceluiași cod

Crearea identității de utilizator

Înainte de înscrierea modificărilor în depozit, Git trebuie să dispună de informații privind utilizatorul care efectuează înscrierea. Dacă Git nu dispune de astfel de date, crearea identității de utilizator se poate face prin emiterea următoarelor două comenzi:

```
git config --global user.email you@example.com
```

```
git config --global user.name "Your Name"
```

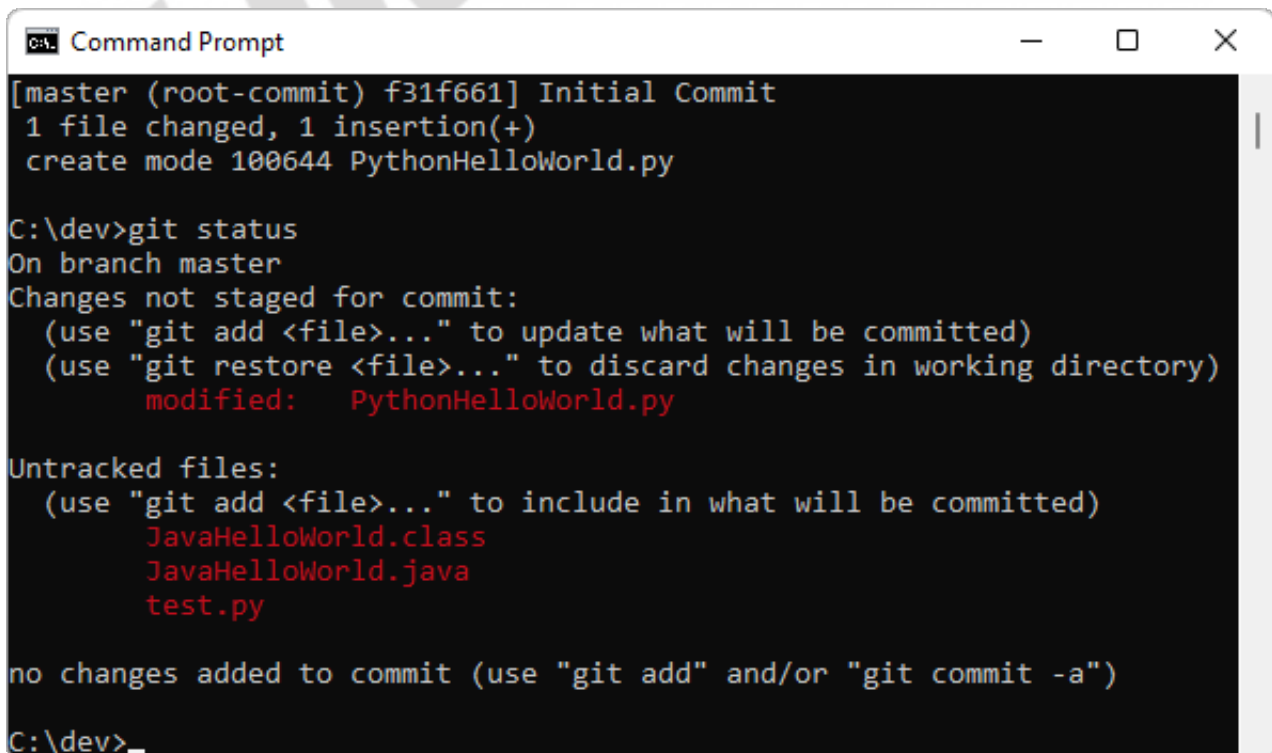
Este foarte important să știți că nu se poate efectua niciun commit dacă identitatea utilizatorului nu a fost setată anterior. Prin urmare, dacă vi se cere să definiți o identitate de utilizator după o încercare de

commit, trebuie să știți că un astfel de commit nu a reușit. Prin urmare, într-o astfel de situație, după definirea identității utilizatorului în modul în care este prezentat în rândurile anterioare, este necesar să se repete commit-ul inițial.

Prin parcurgerea pașilor până în prezent, a fost creat un depozit Git cu un singur fișier. Pentru a vedea cu adevărat puterea sistemului Git, este suficient ca, după pașii descriși, să întreprindeți câteva modificări în fișierul Python care este plasat în depozit. De exemplu, puteți adăuga o altă comandă pentru a imprima un mesaj la ieșire. După efectuarea unor astfel de modificări, salvați fișierul. Git dispune de comanda pentru verificarea stării fișierelor care sunt înscrise în depozit:

git status

Rularea acestei comenzi, după efectuarea modificărilor în fișierul PythonHelloWorld.py, va avea rezultatul ca în imaginea 10.5.



```
[master (root-commit) f31f661] Initial Commit
1 file changed, 1 insertion(+)
create mode 100644 PythonHelloWorld.py

C:\dev>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   PythonHelloWorld.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        JavaHelloWorld.class
        JavaHelloWorld.java
        test.py

no changes added to commit (use "git add" and/or "git commit -a")
C:\dev>
```

Imaginea 10.5. Verificarea stării fișierelor din depozit

În imaginea 10.5. puteți vedea că Git a detectat că există modificări în fișierul `PythonHelloWorld.py`, adică fișierul a fost modificat de la ultimul commit întreprins. Dacă am dori acum să înscriem o astfel de modificare în depozit, este necesar să întreprindem din nou *staging* și *committing*

În final, Git oferă și o privire de ansamblu asupra tuturor acțiunilor commit:

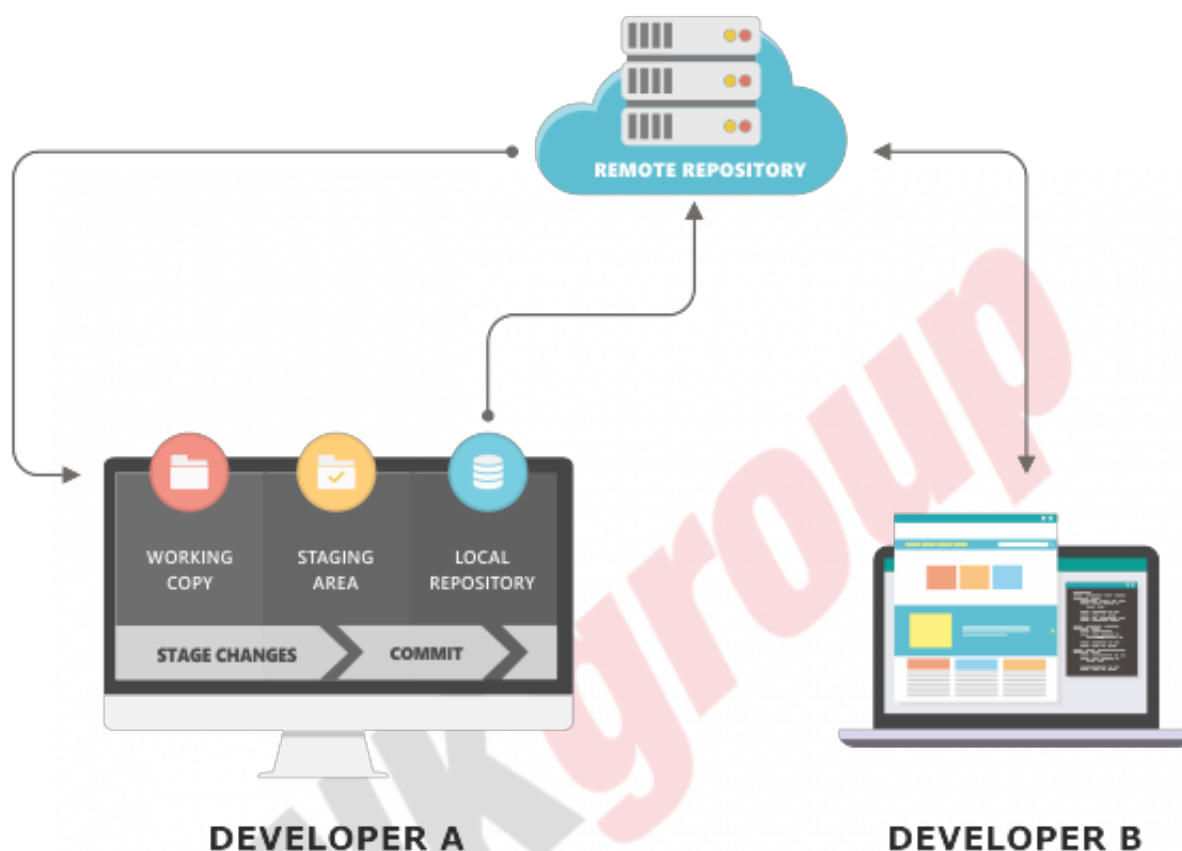
`git log`

Prin emiterea acestei comenzi, se obține o privire de ansamblu asupra tuturor acțiunilor commit din depozitul curent, cu informații însoțitoare.

Ce reprezintă GitHub?

Pentru începători, pot apărea numeroase confuzii atunci când, în povestea despre Git, se introduce termenul GitHub. Este unul dintre cele mai populare sisteme online pentru găzduirea/hosting-ul proiectelor Git. Având în vedere că, în rândurile anterioare, Git s-a prezentat ca un instrument de consolă fără o interfață grafică cu utilizatorul, GitHub poate fi privit ca o învelitoare grafică pentru Git, disponibil pe web. Astfel, GitHub permite crearea de depozite, care sunt disponibile pe web, ceea ce reprezintă o mare oportunitate pentru lucrul în echipă. GitHub nu este singurul sistem de acest gen. Pe lângă el, există multe altele, precum Bitbucket, GitLab, SourceForge etc.

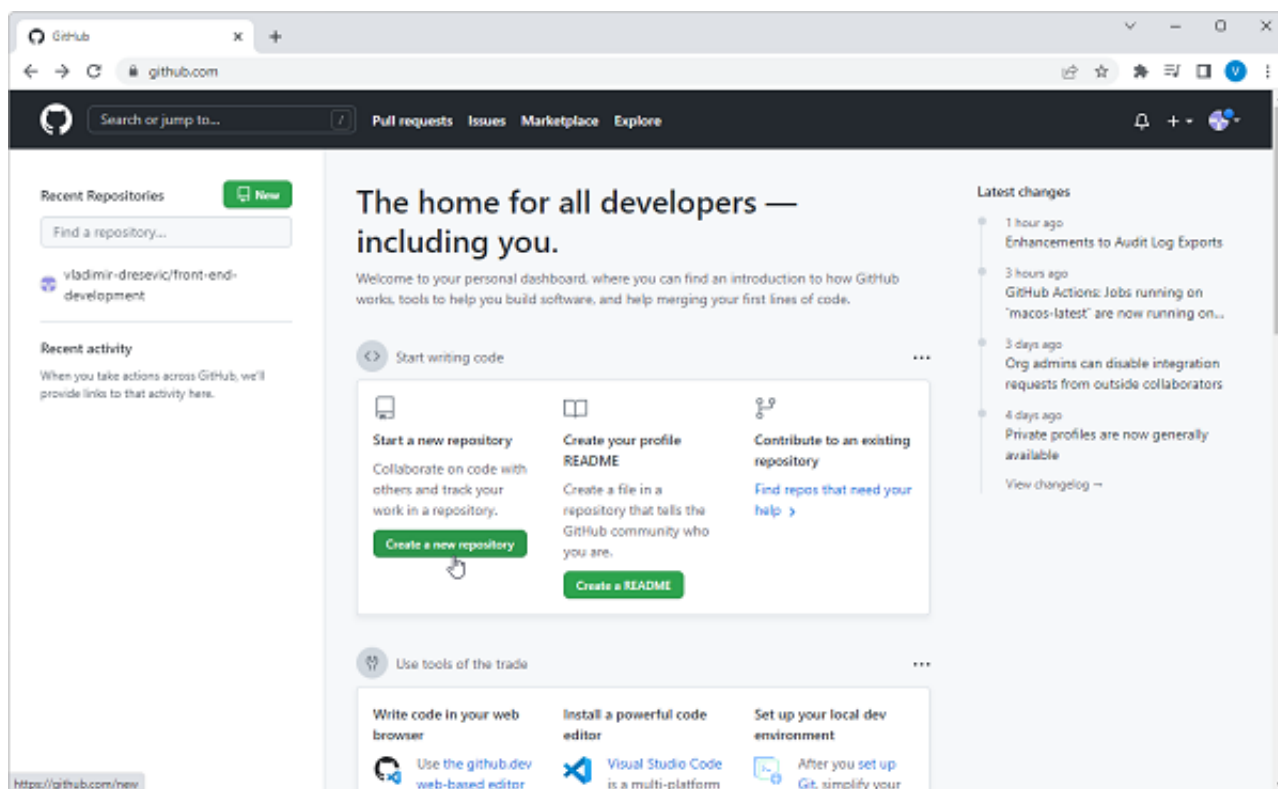
Rândurile anterioare ale acestei lecții au ilustrat crearea unui depozit Git local. Cu toate acestea, pe parcursul lucrului în echipă, de obicei există un depozit central la distanță, în timp ce fiecare dintre programatori dispune de un depozit local pe propriul calculator, tocmai cum ilustrează imaginea 10.6



Imaginea 10.6. Principiul de funcționare a depozitelor locale și a celor la distanță

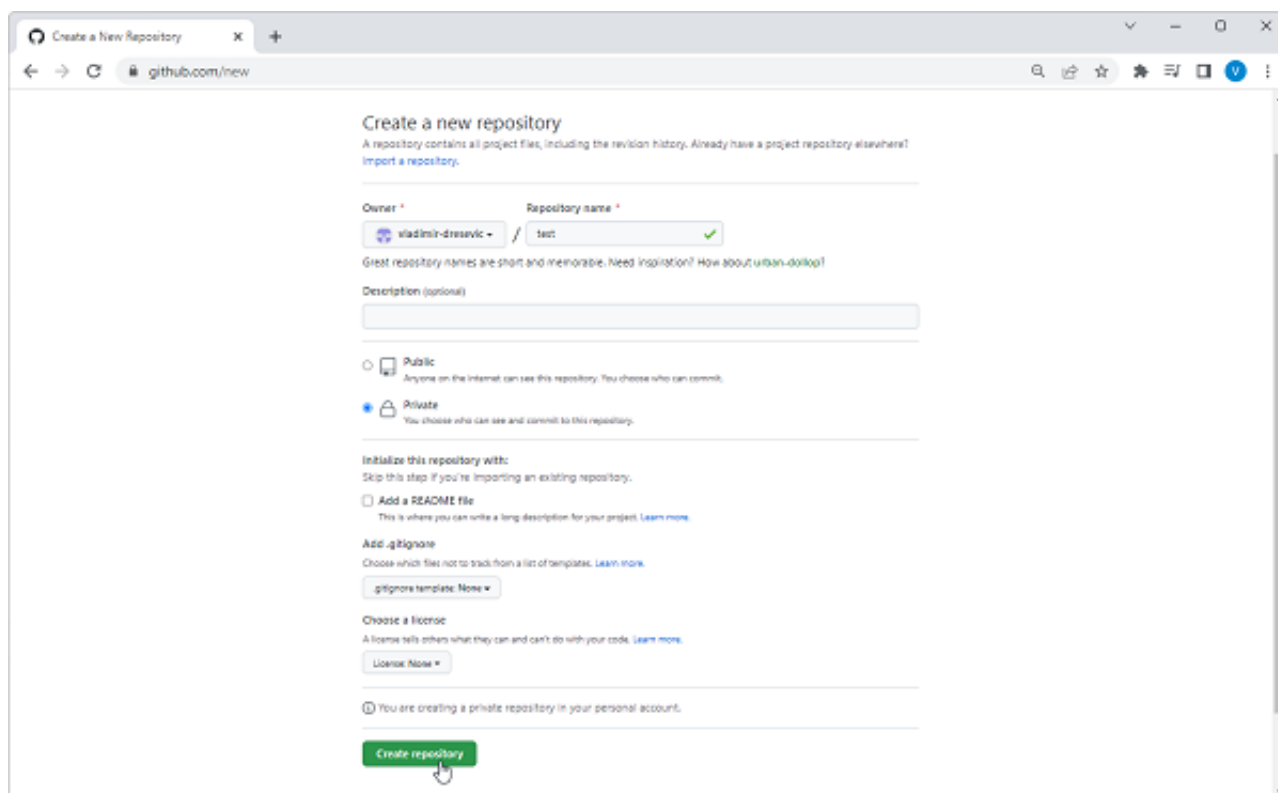
În imaginea 10.6. este ilustrată logica lucrului cu depozitele locale și cele la distanță. Dezvoltatorii lucrează cu depozitele locale și, după efectuarea modificărilor în cod, astfel de modificări transferă în depozitul central, care este partajat cu toți ceilalți colegi de echipă. Astfel, toți membrii echipei își încep munca prin sincronizarea depozitului local cu cel central, pentru a se asigura că lucrează la cea mai actualizată versiune a codului, care conține toate modificările efectuate de restul colegilor din echipă.

Utilizarea GitHub presupune crearea în prealabil a unui cont de utilizator. După crearea contului de utilizator, este necesar să se creeze depozitul (imaginea 10.7.).



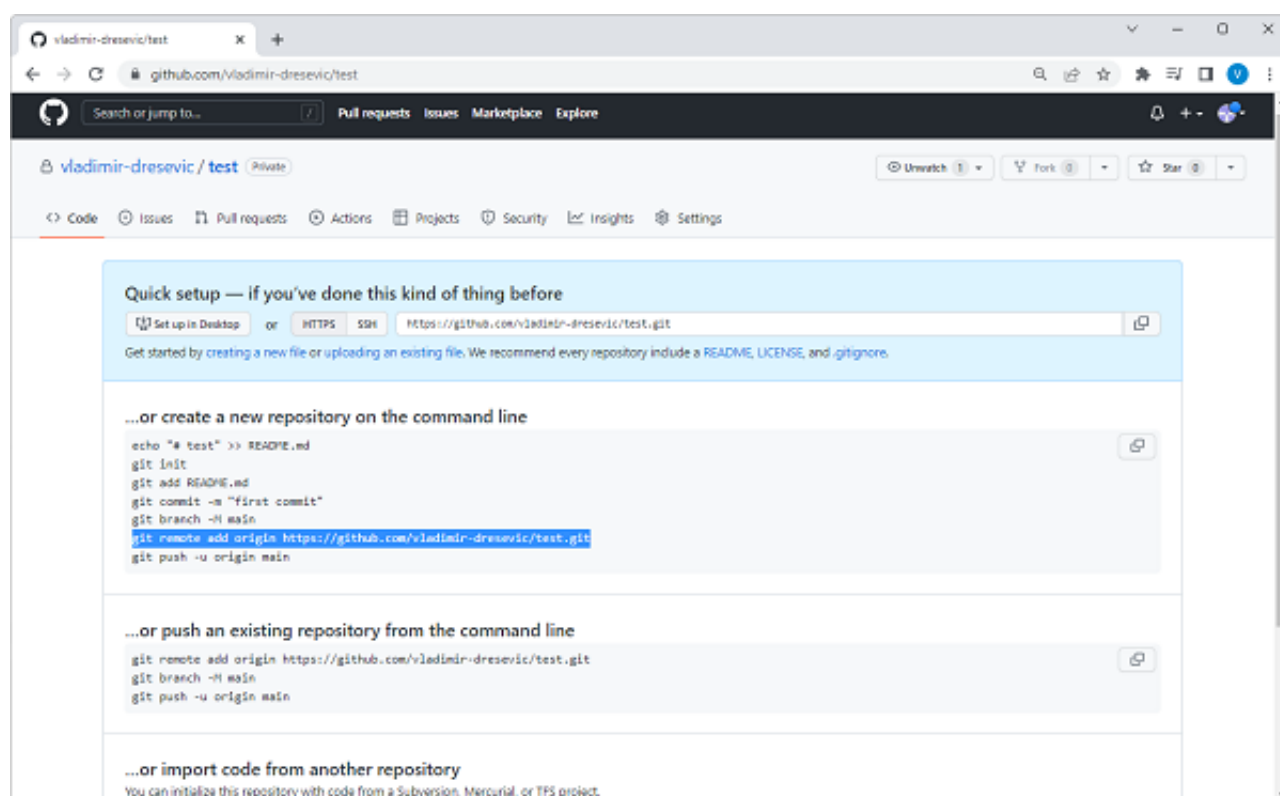
Imaginea 10.7. Opțiunea pentru crearea depozitului (Create a new repository)

Crearea unui nou depozit pe GitHub necesită o setare minimă (imaginea 10.8.).



imaginea 10.8. Setarea depozitului GitHub care se creează

După crearea depozitului pe GitHub, este posibil să conectați depozitul local la depozitul la distanță. Diverse opțiuni privind efectuarea unei astfel de activități pot fi văzute pe pagina care se obține după crearea depozitului (imaginea 10.9.).



Imaginea 10.9. Diverse opțiuni pentru conectarea depozitului local și cel la distanță, respectiv pentru adăugarea fișierelor în depozitul crea

În imaginea 10.9. puteți vedea comanda, care este cea mai simplă modalitate de a conecta depozitul local și cel la distanță:

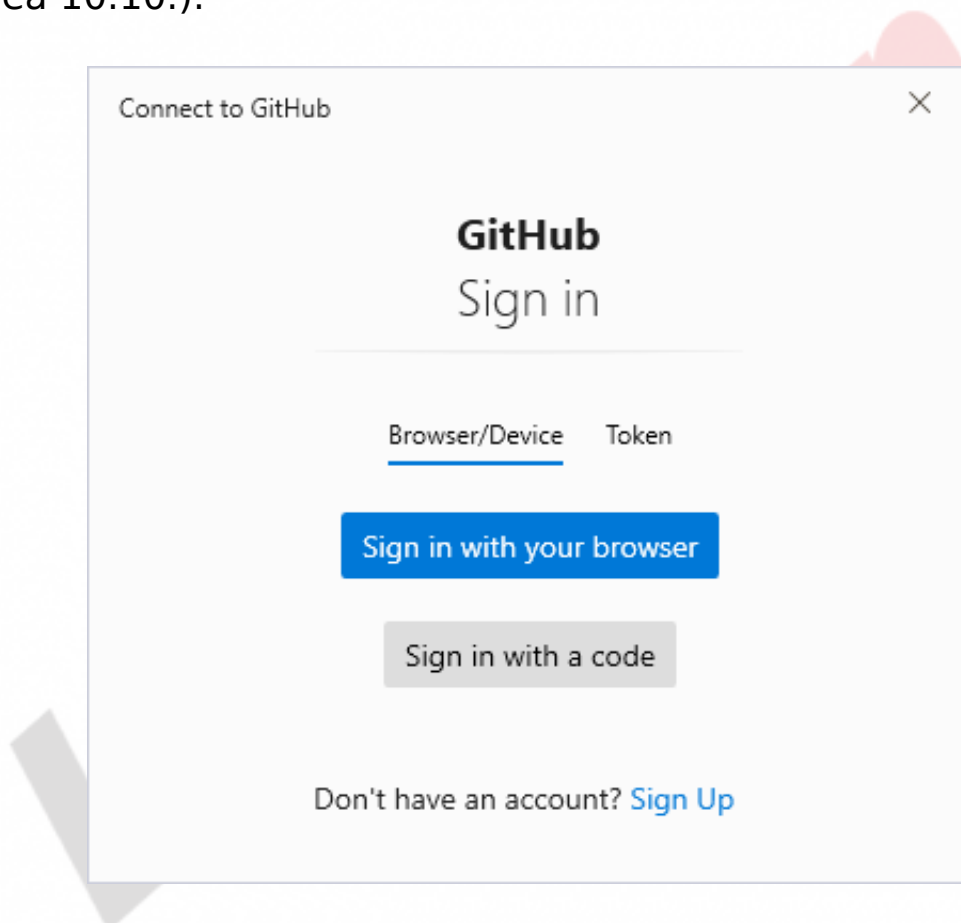
```
git remote add origin https://github.com/vladimir-dresevic/test.git
```

Comanda este remote add origin, în timp ce restul este adresa URL la care se află depozitul la distanță. Desigur, comanda trebuie să fie emisă din consolă, în timp ce sunteți poziționați în fișierul depozitului.

Pentru a se transfera fișierele din depozitul local în depozitul de la distanță, se utilizează comanda **push**:

`git push -u origin master`

Această comandă inițiază upload-ul, adică încărcarea fișierelor locale în depozitul de la distanță. Dacă întreprindeți acest lucru pentru prima dată, va fi necesar să vă conectați la GitHub folosind datele de acces (imaginea 10.10.).



Imaginea 10.10. Fereastra care apare pentru a se conecta consola locală la contul GitHub

Dacă, după conectarea depozitului local și remote, doriți să preluați toate fișierele din depozitul local, puteți utiliza comanda **pull**:

`git pull origin master`

În principiu, comanda `pull` este o comandă care caracterizează începerea lucrului la un proiect în care este încadrată o echipă,

deoarece implică conformarea depozitului local cu depozitele la distanță.

Rândurile anterioare au ilustrat conectarea depozitului local și cel la distanță. GitHub (Git) permite, de asemenea, clonarea depozitelor existente, ceea ce înseamnă crearea unei copii locale identice a unui depozit la distanță. Acest lucru se întreprinde folosind comanda **clone**:

```
git clone https://github.com/vladimir-dresevic/test.git
```

Comanda `clone` nu necesită crearea anterioară a unui depozit local, ea poate fi executată în mod direct și astfel, împreună, se obține depozitul local precum și toate fișierele din acesta.

Notă

Este important de știut că, atunci când un depozit Git se creează folosind GitHub, implicit se obține ramura principală cu denumirea `main`. GitHub a introdus o astfel de noutate în anul 2020, considerând că termenii precum *master* și *slave* pot fi jignitori pentru unele grupuri de oameni. Totuși, atunci când un depozit este creat folosind Git, se obține ramura cu denumirea `master` (dacă nu este definit altfel în fișierul de configurare). În acest curs, am creat depozitul folosind Git și apoi l-am trecut la GitHub, ceea ce a produs ca rezultat faptul ca ramura principală să rămână `master`. Cu toate acestea, este important să știți că un depozit creat pe GitHub, pentru denumirea ramurii principale, va avea `main`.