

Lab Python 1 - Lab 2

Module 1.2 - Controlul versiunilor cu GIT (Version Control)

Note:

Acest laborator cuprinde laboratorul 1 din secțiunea 2.2 -Controlul versiunilor cu GIT 6: Modulul Teorie Python 1.

Obiective:

Învățarea serviciului GitHub și cum se lucrează pe aceasta platforma.

Ce este Git/ GitHub?

Există numeroase servicii disponibile care oferă spații pentru depozitare și colaborare, cum ar fi GitHub, GitLab și altele. Toate aceste instrumente sunt construite pe baza capacităților fundamentale ale Git, un sistem de control al versiunilor open-source, care este o componentă esențială în dezvoltarea software-ului.

GitHub, în particular, este un serviciu de cloud care aparține și este operat în prezent de Microsoft. Oferă un ecosistem bogat pentru stocarea, lucru și colaborare pe proiectele bazate pe Git. Acesta nu este doar un simplu loc de depozitare, ci și o platformă socială și de colaborare. Fiecare depozit pe GitHub reprezintă practic un folder care conține întregul context al unui proiect, inclusiv codul sursă, documentația și alte resurse relevante.

În cadrul GitHub, utilizatorii pot colabora cu alții, pot propune modificări, pot raporta probleme (issues), pot crea ramuri (branches) pentru dezvoltare paralelă și multe altele. Acesta este un mediu în care dezvoltatorii pot interacționa, revizui și îmbunătăți în mod colectiv proiectele lor.

Este important să rețineți că, pe lângă funcționalitatea fundamentală a Git, GitHub adaugă un strat suplimentar de utilități și funcționalități care facilitează colaborarea și dezvoltarea software-ului într-un mod mai eficient și mai social.

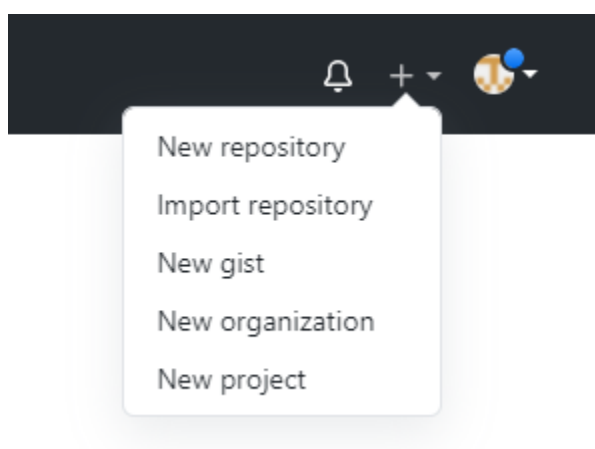
Cerinte:

Intrați pe platforma github.com și creați-vă un cont nou.

Crearea unui repo

Odată ce aveți un cont pe GitHub, puteți începe să lucrați. Primul lucru este să creați un repository (depozit).

În colțul din dreapta sus, apăsați pe butonul '+' și dați clic pe new repository.



Acum o să vă apară o nouă pagină unde puteți să adăugați detalii despre depozitul (repo) pe care urmează să îl creați:


- După ce îi puneți un nume și o descriere depozitului dvs, puteți alege dacă acesta va fi public sau privat.
- Într-un depozit privat, puteți alege cine îl poate vedea și da commit.
- Pe un depozit public, oricine îl poate vedea. Chiar și cu depozitele publice, numai persoanele aprobate pot face modificări.
- După ce ați ales dacă doriți să faceți un proiect privat sau public, vă puteți initializa depozitul cu următoarele:
 - README file = este destinat utilizatorilor depozitului sau vizitatorilor depozitului să înțeleagă despre ce este vorba, instrucțiuni și explicații despre modul de utilizare a codului. Oferă documentația pentru cod.
 - .gitignore = se creează în directorul rădăcină al depozitului dvs. pentru a spune lui Git ce fișiere și directoare trebuie ignorate atunci când dați commit.
 - License = Depozitele publice de pe GitHub sunt adesea folosite pentru a partaja software open source. Pentru ca depozitul dvs. să fie cu adevărat open source, va trebui să îl licențiați, astfel încât alții să poată utiliza, modifica și distribui software-ul.

- Licențele pot fi studiate pe linkul:
<https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/creating-a-repository-on-github/licensing-a-repository>

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)


Owner * Repository name *

 /

Great repository names are short and memorable. Need inspiration? How about **stunning-dollop**?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Termeni des întâlniți în Git:

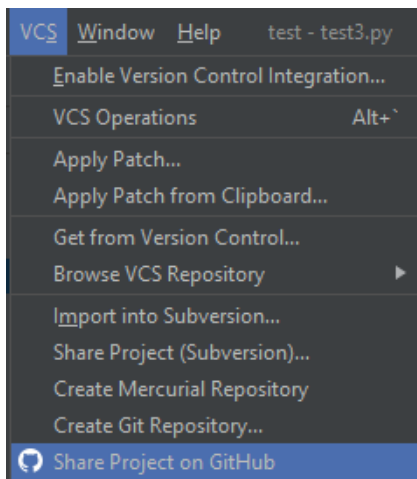
- Commit și push
 - Un "commit" în Git este o acțiune prin care se salvează modificările făcute în fișierele din cadrul unui repository local.
 - Un "push" este o acțiune care încarcă (trimite) commit-urile locale către un repository remote. Astfel, modificările devin disponibile și pentru alți colaboratori ai proiectului.
- Pull requests:
 - vă ajută să colaborați la cod cu alte persoane. Pe măsură ce sunt create pull-requests (PR), acestea vor apărea aici într-o listă de căutare și filtrare.
 -

- O folosință comună a pull request-urilor este de a face commit și push pe branch-ul principal (de obicei master), unde se găsește versiunea funcțională și originală a codului. De obicei, se adaugă una sau mai multe persoane ca recenzori. Aceste persoane trebuie să verifice pull request-ul și să aprobe sau să lase comentarii pentru îmbunătățirea codului.
- Branch:
 - Crearea unui branch permite dezvoltatorilor să lucreze independent pe noi funcționalități sau modificări fără a afecta codul din ramura principală (master branch). Ramurile oferă un mediu izolat pentru dezvoltarea și testarea noilor caracteristici, iar când acestea sunt completate, pot fi încorporate înapoi în ramura principală printr-un proces numit "îmbinare" (merge). Ramurile facilitează colaborarea și gestionarea codului în cadrul unui proiect, permițând echipelor să lucreze în paralel fără conflicte.

PyCharm si Github

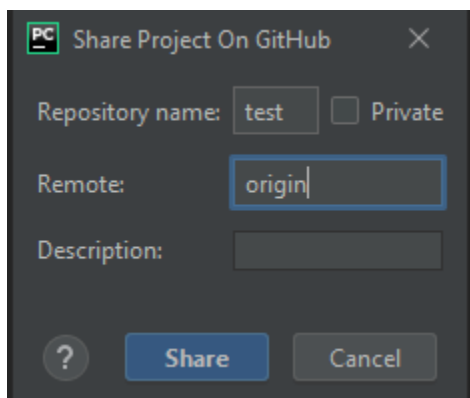
Puteți adăuga pe GitHub un proiect pe care îl dezvoltați local, astfel încât alții să îl poată vedea sau să contribuie la acesta.

1. În PyCharm, deschideți proiectul pe care doriți să îl încarcați pe GitHub.
2. Din meniul principal, dați click pe VCS și dați click pe *Share Project on GitHub*.

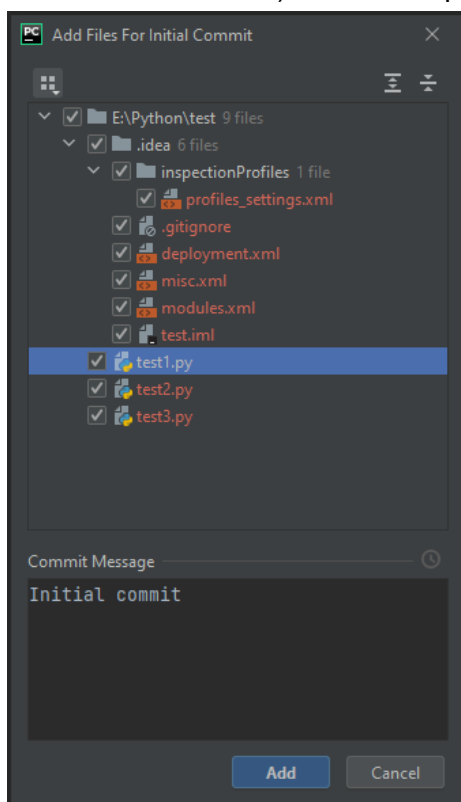


3. Dacă v-ați înregistrat deja contul GitHub în PyCharm, conexiunea va fi stabilită folosind aceste acreditări. Dacă nu v-ați înregistrat contul în PyCharm, se deschide dialogul Conectare la GitHub. Specificați jetonul de acces sau solicitați unul nou cu datele de conectare și parola.

4. Când conexiunea la GitHub a fost stabilită, se deschide caseta de dialog Partajare proiect pe GitHub. Specificați numele noului depozit, numele depozitului la distanță și introduceți o descriere a proiectului dumneavoastră. Puteți selecta opțiunea *Private* dacă nu doriți să permiteți accesul public la depozitul dvs. pentru alți utilizatori GitHub.



5. Selectați sursele proiectului care doriți să le încarcați pe GitHub, un mesaj de commit (by default Initial Commit) și dați click pe Add.



6. Intrati pe GitHub si dati click pe *Repositories*. Acum ar trebui sa va apara noul depozit creat.
7. Intrați pe depozit, click pe Settings. La secțiunea manage access puteți invita alți colaboratori, schimba de la privat la public sau invers, etc.