Verziókezelés segédlet „Programozás alapjai” c. tárgyhoz

Készítette: Tompa Tamás, 2021

Mi ez?

A verziókezelés olyan eljárások összessége, amelyek lehetővé teszik egy adathalmaz változatainak (verzióinak) együttes kezelését. Szoftverek estében ez a szoftver életciklusa során a forráskódban végzett módosítások tárolását jelenti.

Miért van rá szükség?

A fejlesztés során a források sok iteráción megy keresztül, így szükséges lehet, hogy esetleges probléma eseten vissza lehessen térni egy korábbi verzióra. Nagy segítséget nyújt továbbá csapatmunkában történő szoftverfejlesztés esetében is.
Egyszerű verziókövetés: minden változtatást elmentünk külön jegyzékekben. Megvalósítható, de nehézkes, időidényes, nem hatékony, nem átlátható.
Megoldás: verziókövető rendszerek (pl. **Git**, Mercurial, stb.). Lehet használni helyi (local) illetve távoli (remote) repokat, ilyen távoli repok például a Github, Bitbucket stb.
A Git-et a használathoz telepíteni szükséges: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
Hasznos GUI-val rendelkező Git kliens a **SmartGit:** https://www.syntevo.com/smartgit/

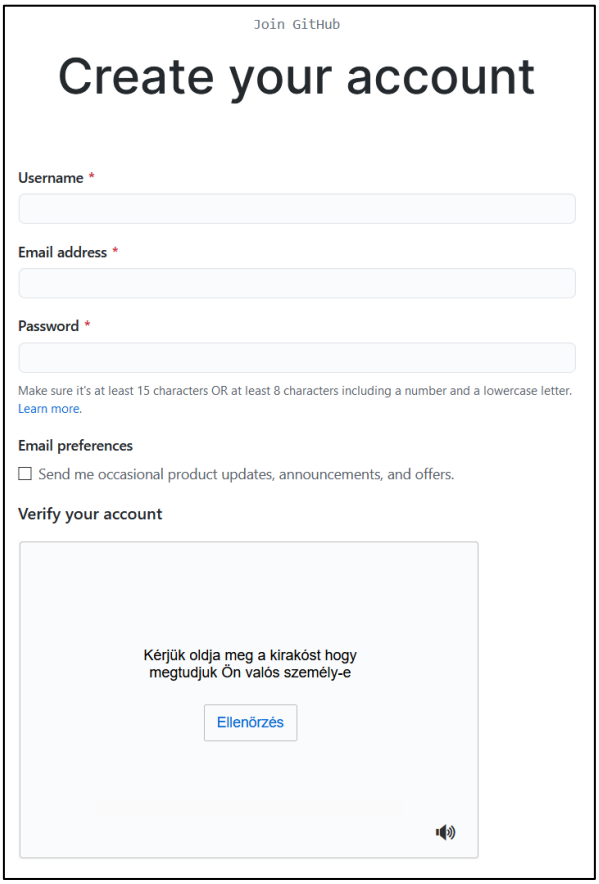
Alapfogalmak

Repository: Maga a tárolónk. Röviden csak repo.
Working copy: A kód egy részének egy példánya, amelyen a fejlesztő éppen dolgozik a saját gépén
Commit: A kódon eszközölt változtatásokat úgynevezett kommitok formájában érvényesíthetjük a tárolókon belül. A tárolók mintegy pillanatképként tartalmazzák azokat, illetve projektünk aktuális állapotát. Célszerű minden nagyobb módosítást követően kommitolnunk. Az adott kommithoz általában megjegyzés is írható, hogy milyen módosítás történt az adott kommit hatására.
Revision: verzió
Checkout: Lokális másolat készítése valamely verziókezelt fájlról.
Head: a legfrissebb kommitot (verziót) jelöli, az aktuális ág teteje.
Push: adatok feltöltése a központi repoba
Pull: változások letöltése
Diff/Change/Delta: két file között változás megtalálása/mutatása.
Branch: fejlesztési ág
Merge: összefésülés. A fejlesztési ágak létrehozása mellett lehetőségünk van ezek egyesítésére is.
Conflict: Ágak összefésülése során keletkező jelenség. A két ág verziója olyan kódot tartalmaz, amit nem lehet automatikusan összefésülni
Clone: repo tartalmának lehozása lokálisan (adott jegyzékbe)

Forrás: Dr. Mileff Péter, Szoftverfejlesztés, Verziókövetés, Verziókövető rendszerek, jegyzet https://users.iit.uni-miskolc.hu/~mileff/szf/Verziokezeles_V5.pdf

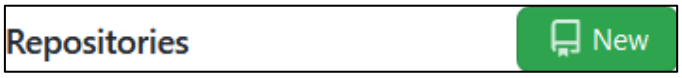
Github
https://github.com/
Távoli repository, amely a forráskódok távoli tárolását valósítja meg (részben ingyenes), használata regisztrációhoz kötött. Mérete folyton csak nő (forráskódból való törlés esetében is), hiszen minden egyes commit-olt verziót eltárol és bármelyik verzióra vissza lehet térni.

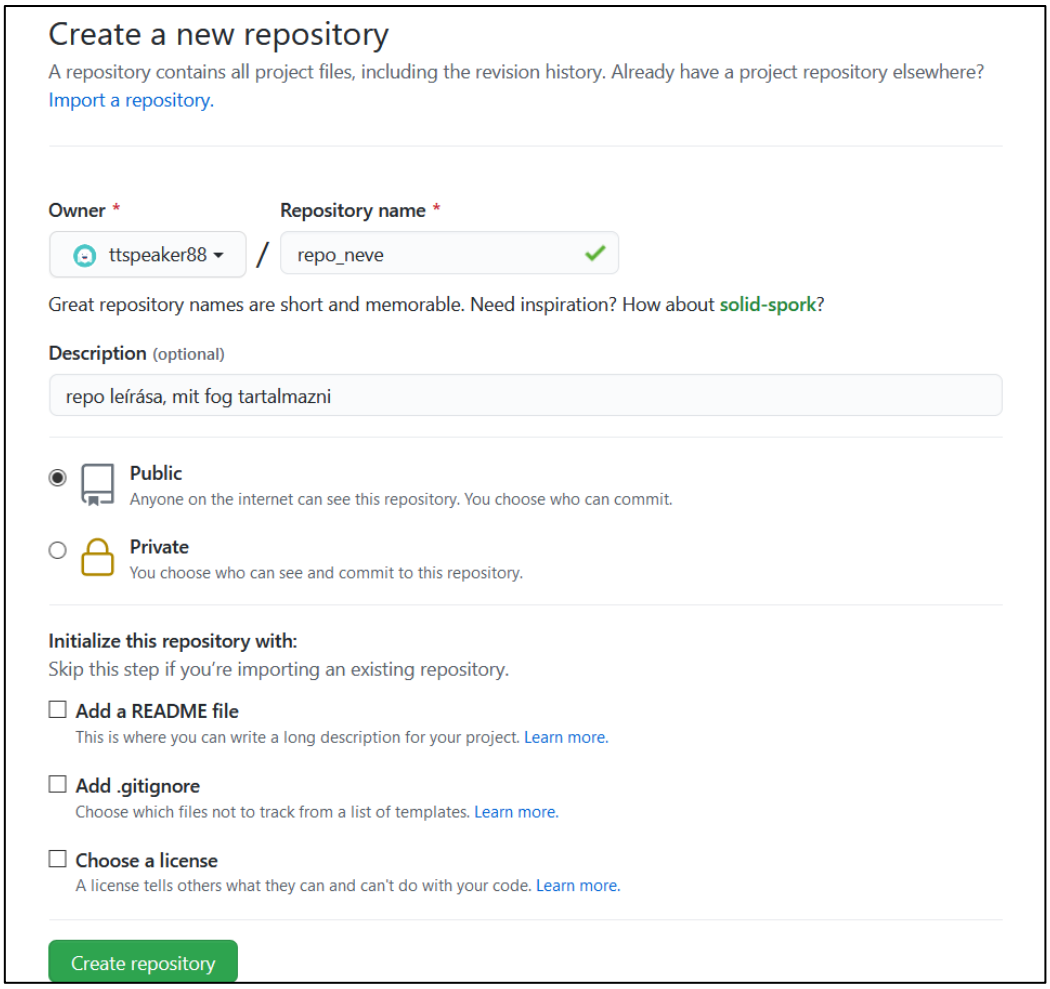
Regisztrációs felület:



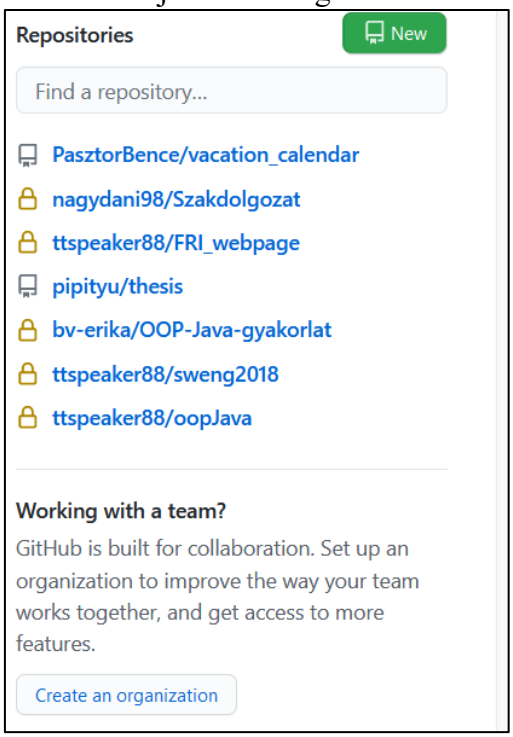
Regisztráció után repo létrehozása szükséges, amely lehet **public vagy private**. Public esetében a weben bárki láthatja a repo tartalmát, private esetében csak az adott jogkörrel rendelkező személyek.

Új repo létrehozása:

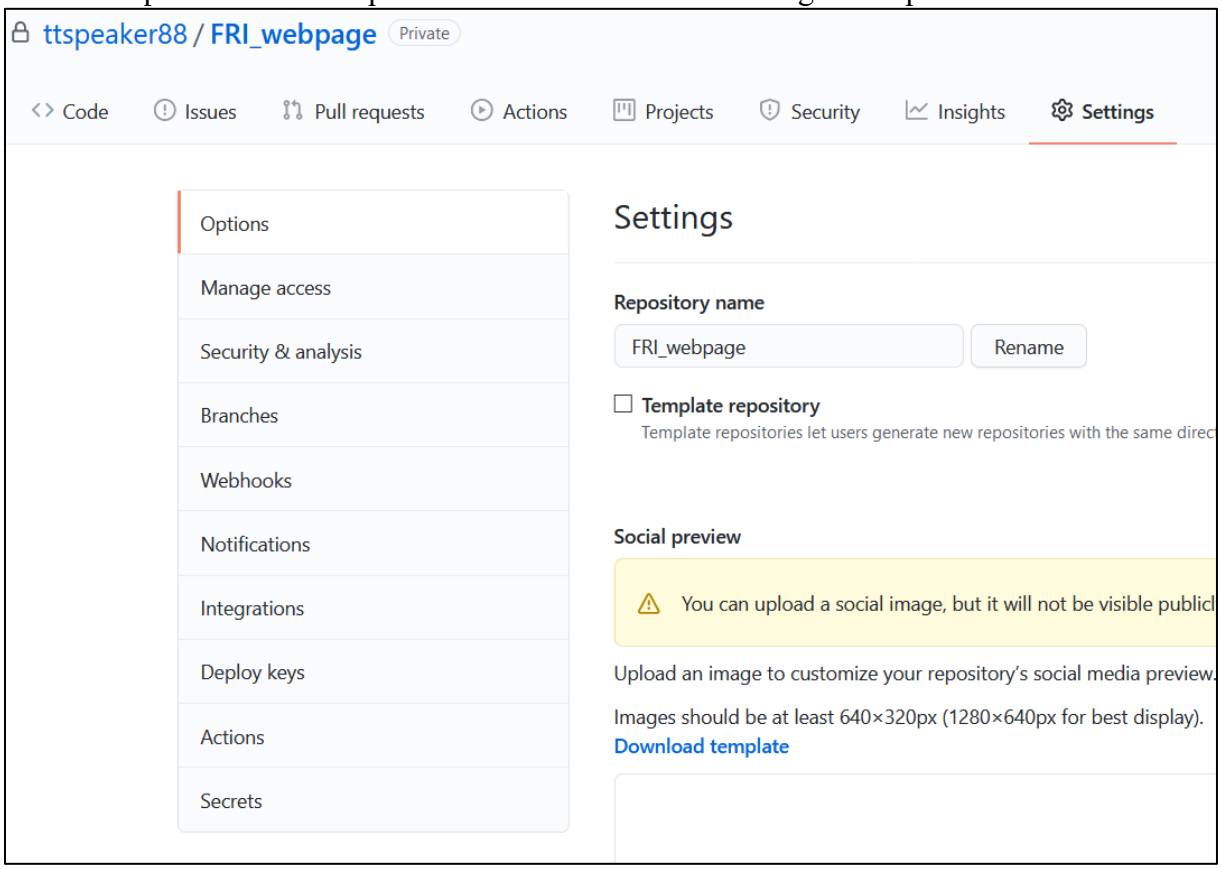




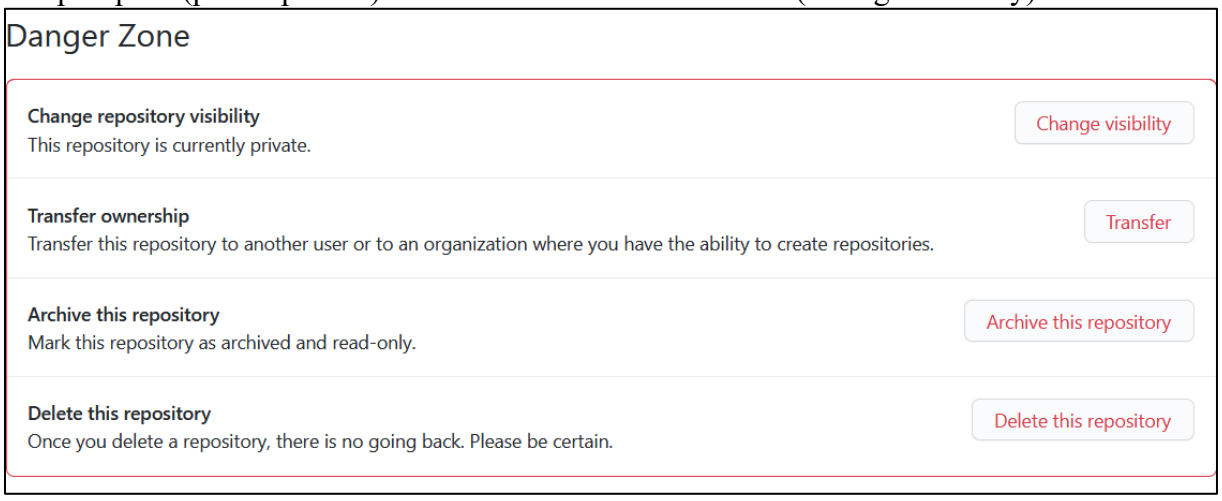
Létrehozott repo-k a felület bal oldalán jelennek meg:



Az adott repo beállításai a repo-ra való kattintás utána a Settings menüpont alatt találhatók:



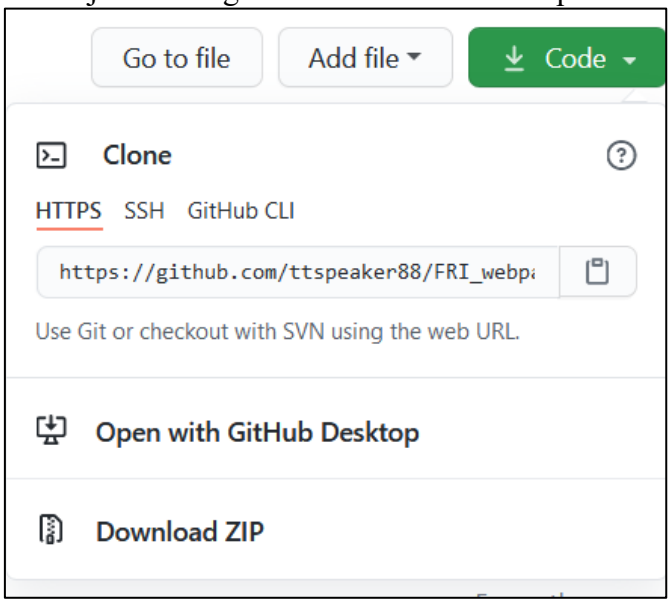
A repo típusa (public/private) az ablak alsó részében állítható (Change visibility):



Manage access menüpont alatt (Invite a collaborator) lehet hozzáadni a private repo-hoz kollaborátorokat (résztvevőket), akik láthatják a repo tartalmát és commit-olhatnak is oda:

A repo tartalma és elérhetőségét biztosító link a Code menüpont alatt található meg:

A zöld Code fülre kattintva jelenik meg a link és itt tölthető le zip-elve a repo tartalma.



Git használata parancssorból, fontosabb parancsok

1. `Git Bash` indítása

2. `mkdir repo`
   „repo” nevű könyvtár létrehozása

3. `cd repo`
   „repo” nevű könyvtárba lépés

4. `git config --global user.name "Saját Nevünk"`
   git konfigurálása, saját név beállítása

5. `git config --global user.email saját@email.címünk`
   git konfigurálása, saját email cím beállítása

6. `git init`
   git munkaterület (üres repository) inicializálása

7. `git remote add origin ~/repo/.git`
   az előző lépésben létrehozott lokális repo hozzáadása

8. `touch valami.txt`
   Tetszőleges fájl létrehozása a git könyvtárán („repo”) belül, pl. egy txt is megfelel. Töltsük fel tetszőleges tartalommal!

9. `git add valami.txt`
   A „valami.txt” fáj hozzáadása verziókövetésre, ezt a fájlt figyelje a Git. Változás felvétele.

10. `git commit –m „commit szövege”`
    Commit-olás, ahol a „commit szövege” helyett pl. „initial commit”

11. `git push origin master`
    Változtatás feltöltése („push”-olása) az „origin” tárhelyen a „master” ágba.

12. Változtatás a txt állományon, pl új sor hozzáírása -> mentés

13. `git add valami.txt`
    Változás felvétele.

14. `git commit –m „új sor beszúrása”`
    Commit-olás

15. `git push origin master`
    Változtatás feltöltése („push”-olása) az „origin” tárhelyen a „master” ágba.

16. `git log`
    Git történelem kiíratása, commit-ok (azonosítója, szövege stb.)

17. `git diff commit1 commit2`
    A „commit1” és a „commit2” közötti különbség kiíratása

18. `git checkout „commit id”`
    A „commit id”-jű commit verzióra történő visszaállítás

    `git checkout master`
    Hatására visszaáll a legutolsó commit állapotú verzió.

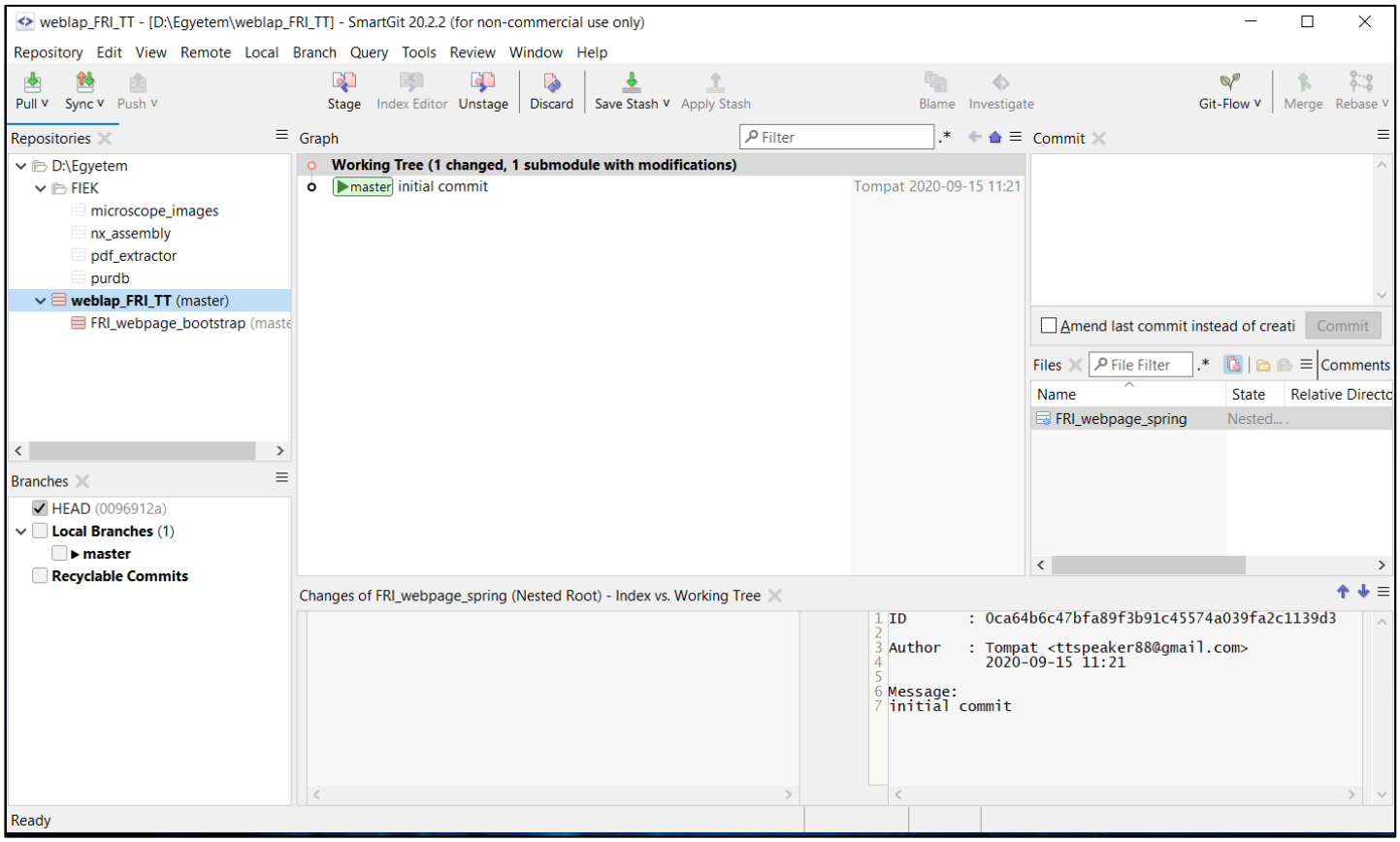
További hasznos Git tutorial: https://www.tutorialspoint.com/git/index.htm

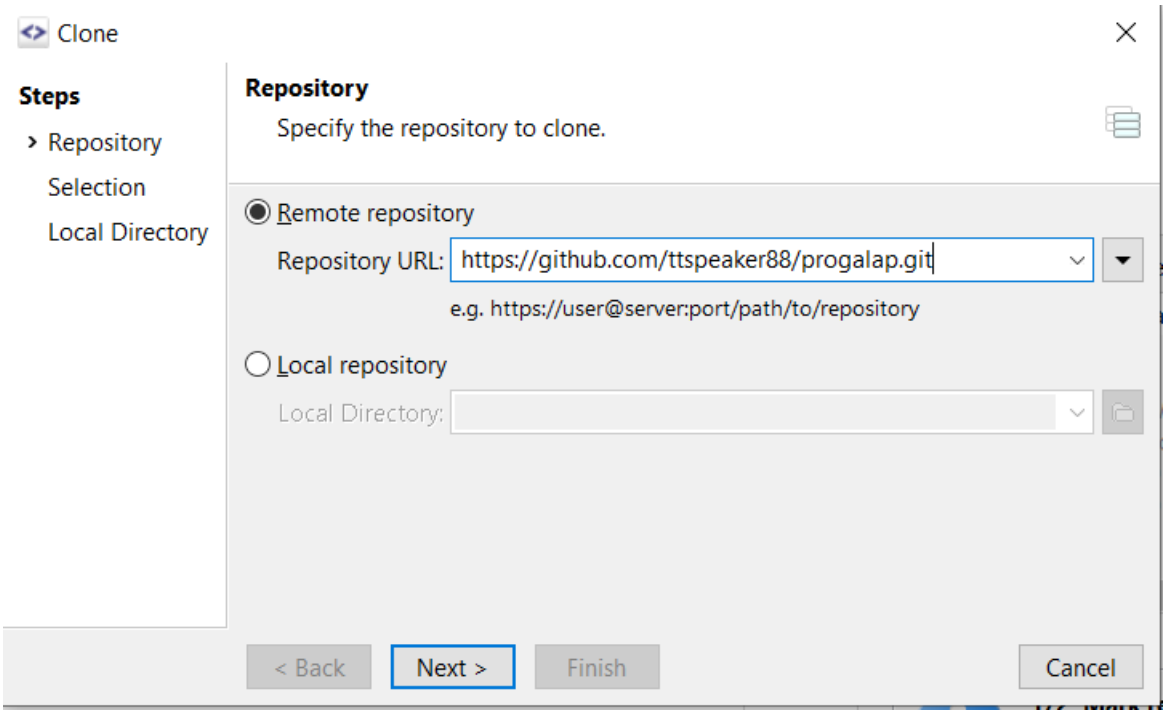
A SmartGit szoftver telepítése és használata

A SmartGit egy (bizonyos mértékig ingyenes) grafikus felhasználói felülettel rendelkező Git kliens, amely nagy segítséget nyújt repository-aink kezelésében. Több ehhez hasonló GUI-val rendelkező Git kliens szoftver is létezik, például a GitKraken (https://www.gitkraken.com/), a továbbiakban a SmartGit alapvető használat lesz részletezve.

A szoftver letöltése: https://www.syntevo.com/smartgit/download/

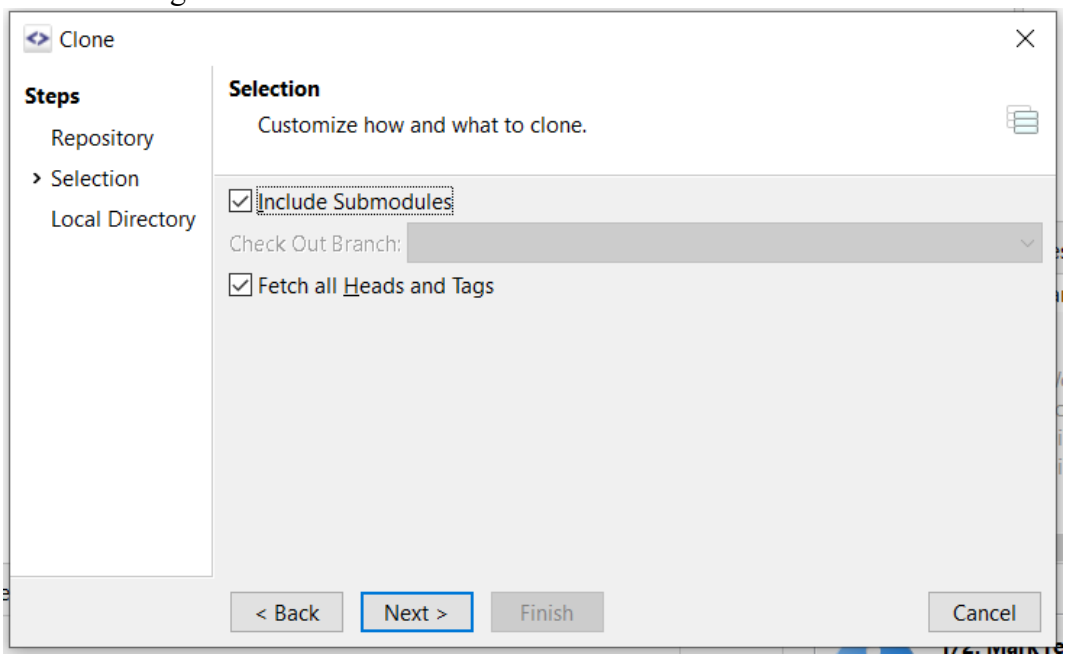
A SmartGit felülete az alábbi:



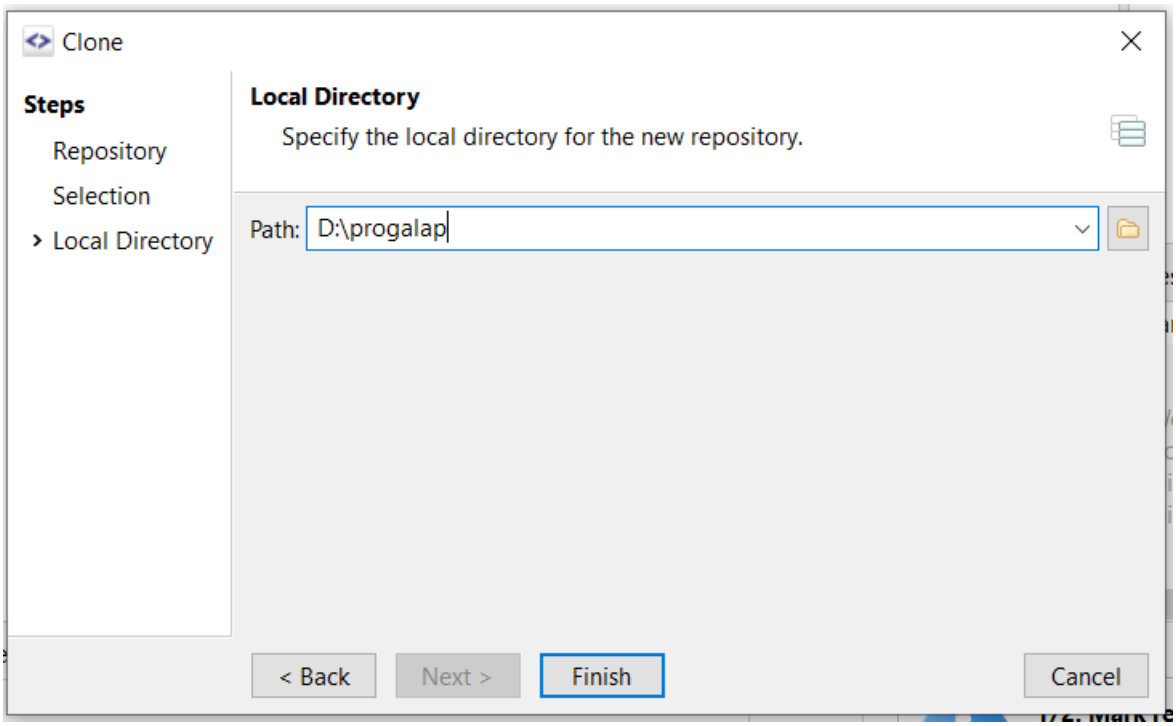


Már létrehozott (pl. GitHub repo) repository-t lehozni gépünkre (clone) „Repository -> Clone…” menüpont alatt lehetséges, amely után az alábbi Repository ablak jelenik meg:

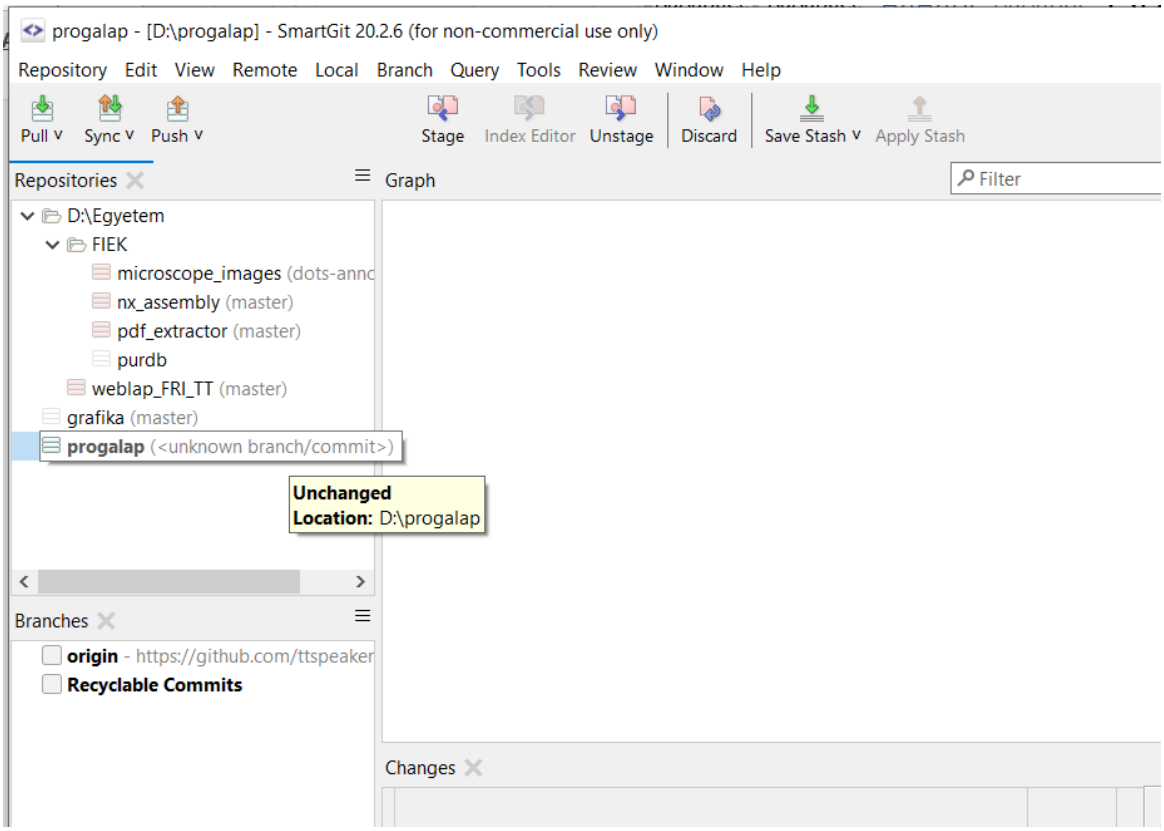
Itt a „Remote repositroy” és a „Local repositroy” lehetőség közül választhatunk. A Remote egy távoli szerveren lévő (Github, Bitbucket, stb.) repo lehozására alkalmas, a Local pedig a saját háttértárunk lévő repo hozzáadását valósítja meg.
Esetünkben a Remote használata szükséges, itt kell megadni a Github-on (vagy más távoli szerveren) létrehozott repo URL-jét (a lehozott repo kezdetben üres lesz, majd ide töltjük fel a tartalmakat).
A Next gombra való kattintás után a következő ablak jelenik meg, ahogy az alapbeállításokat egyelőre nem szükséges módosítani:



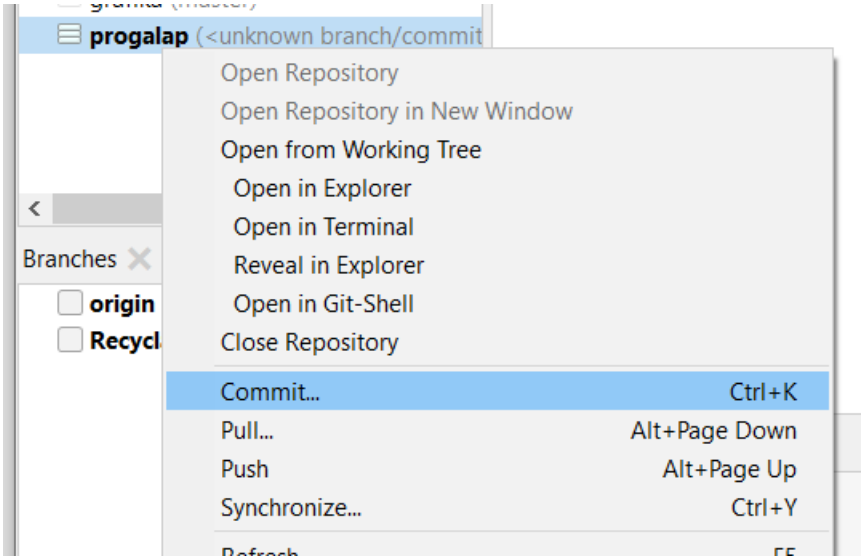
Következő lépésben az a jegyzék (könyvtár) megadás szükséges ahová a távoli repo tartalmát le szeretnénk hozni, le szeretnénk menteni. Az itt megadott könyvtárban lévő tartalom lesz verziókövetve:



A Finish gombra való kattintás után a megadott könyvtárba lehozza a távoli repo tartalmát a rendszer, majd ha minden rendben ment akkor a főablak bal oldalán a „Repositories” listában vastagon szedve megjelenik a repo neve, a „Graph” részben pedig majd a commit-ot listája:



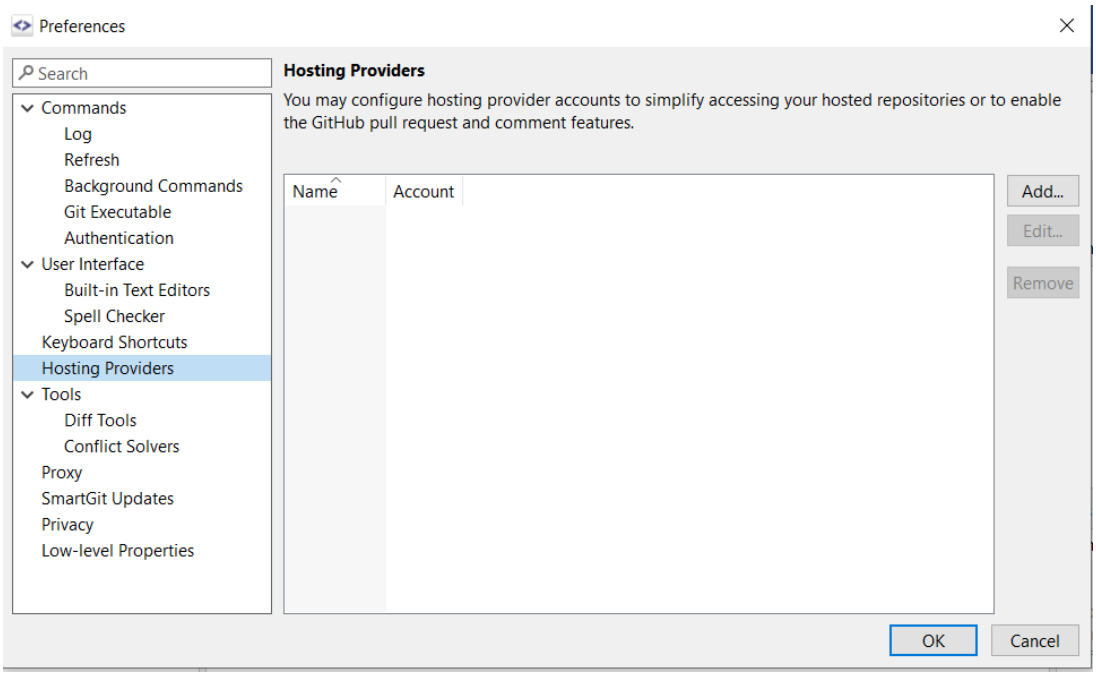
A repo-ra jobb gombbal történő kattintás utána nyílik meg egy lista, amelyből a főbb parancsok kiadása (commit, pull, push, stb) valósítható meg:



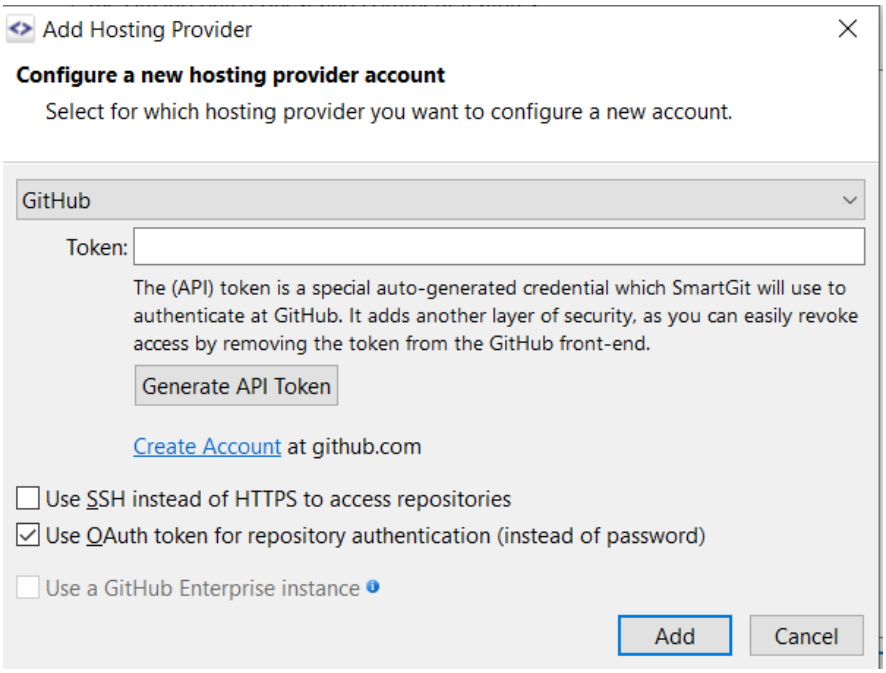
SmartGit autentikáció beállítása

Ahhoz, hogy a SmartGit kilenssel lehessen GitHub reponkat kezelni, szükséges lehet a GitHub-ban beállítani jogusultságot, hogy az engedélyezze a SmartGit-en keresztüli Git műveleteket végrehajtását. Ezeket a beállításokat elsőként a SmartGit szoftverben kell elvégeznünk, majd a GitHub-on is szükséges néhány lépést elvégezni majd.

Nyissuk meg az Edit -> Preferences fület, majd itt válasszuk a Hosting Providers menüpontot:

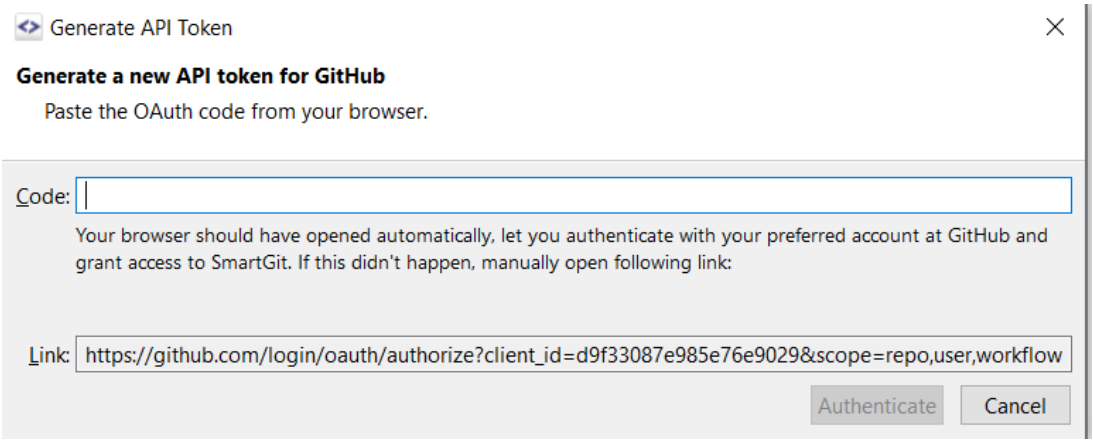


Az Add gombra való kattintás utána az alábbi panel jelenik meg, ahol kattintsunk a Generate API Token gombra:

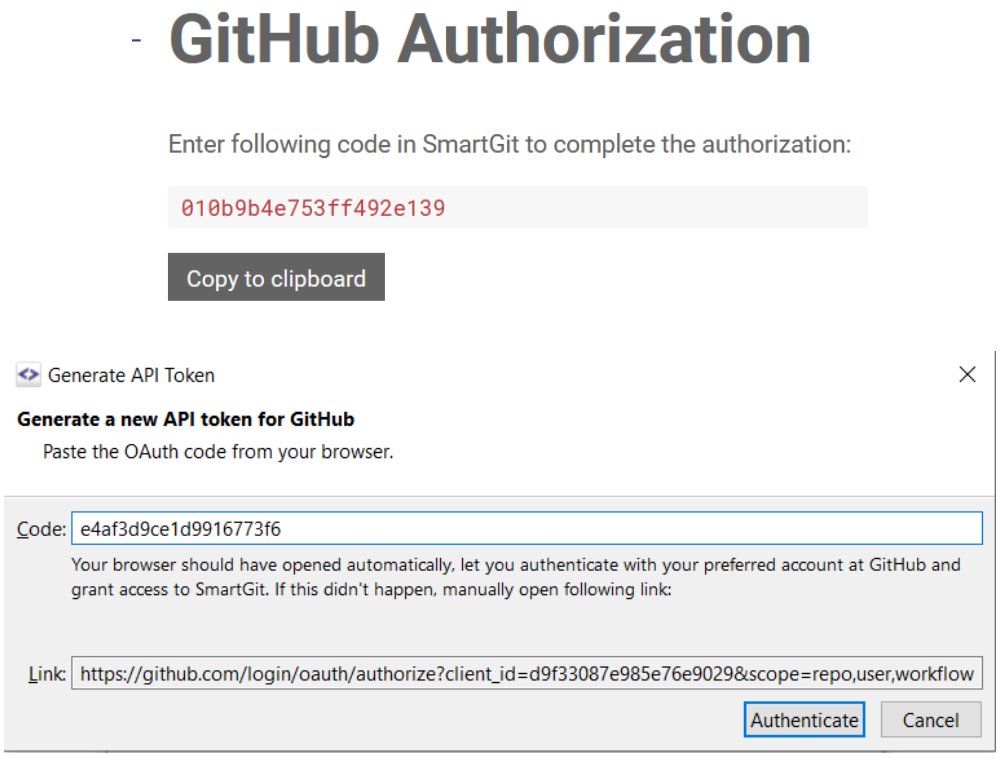


A következő lépésben a GitHub jelszó megadás szükséges:

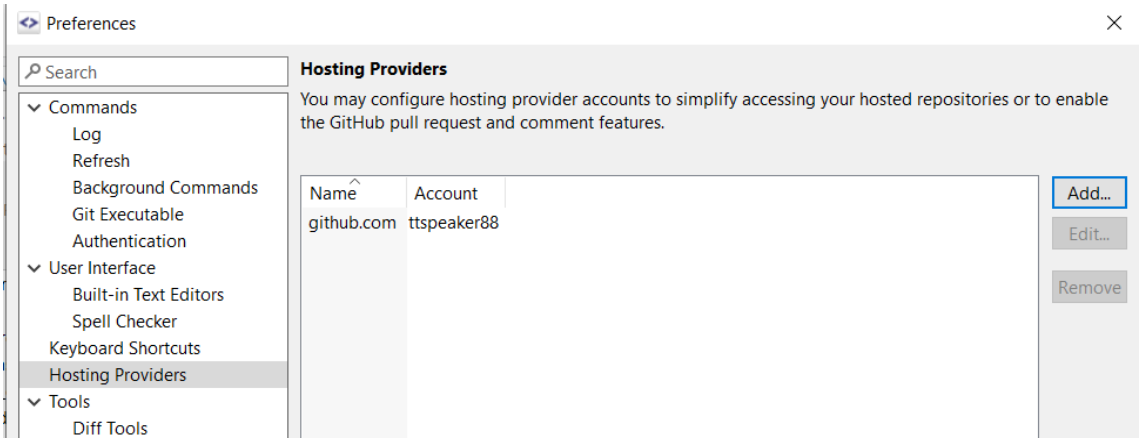
A jelszó megadása után a következő ablak jelenik meg, ahogy a kapott link-et illeszük be egy böngészőbe:



A link hatására megnyíló felületen válasszuk az Authorize syntevo gombot (siker esetén zöld színű), majd adjunk meg a GitHub jelszót ha az szükséges. Ha minden rendben ment akkor, egy új lapon meg fog jelenni egy token (String sorozat), amit be kell illeszteni a Code szövegmezőbe:



Ha minden rendben ment akkor az Authenticate gombra való kattintás után megjelenik a SmartGit-hez hozzáadott GitHub account-unk:



A továbbiakban a GitHub engedélyezni fogja a SmartGit számára, hogy Git műveleteket (Commit, Push, stb) végezhessen el a GitHub reponkunk.