Git segédlet

Mi a feladat

Automatizált tesztek eljuttatása a fejlesztésnek, hogy a termék minőségét egyszerűen igazolni tudják. Ezt ők aztán felhasználják új kiadások ellenőrzésére.

A teszteket otthon írjuk, ezért a teszt kódot, ami otthon (továbbiakban lokálisan) van kéznél, el kell juttatni olyan helyre (továbbiakban remote), ahol a fejlesztők (és más tesztelők) ellenőrizni és futtatni tudják őket.

Hogy tudjuk megcsinálni

A teszteket úgy írjuk, ahogy akarjuk. Néhány dolgot kell figyelembe venni:

* A teszteknek akkor van értékük, ha az alábbiak teljesülnek:
  + Tudjuk, hogy mit ellenőriznek
  + Relevánsak a termék minőségére vagy a fejlesztés menetére nézve
  + Bármikor futtathatóak
  + Értelmezhező a kimenetük minimális előzetes ismeret birtokában (megmondja neked, hogy jó vagy nem jó)
  + (meg még pár dolog, ami nem lényeges itt)
* A teszteket akik akarják, azonos eredménnyel tudják futtatni a saját környezetükön
  + Ez gyakorlatilag azt jelenti, hogy egyrészt meg kell beszélni, hogy ha valamit olyat akarsz használni, amire a többiek esetleg nem számítanak és problémát okozhat náluk, másrészt debugolni kell, miután elkezdtük haszálni
  + Lényegében fenn kell tartani az alapállapotot, hogy minden teszt mindenkinél futtatható
* A teszteknek illeszkedniük kell a dokumentációs kontextusba.
  + Követni kell tudni tesztesethez, amit mások is elérnek. Ez lehet dokumentáció vagy ember által olvasható adatfile. Ha adatfile-t használunk, legalább use case-hez kössük a tesztesetet.
  + Tudni kell hozzá hibajegyet felvenni, amit mások is elérnek.

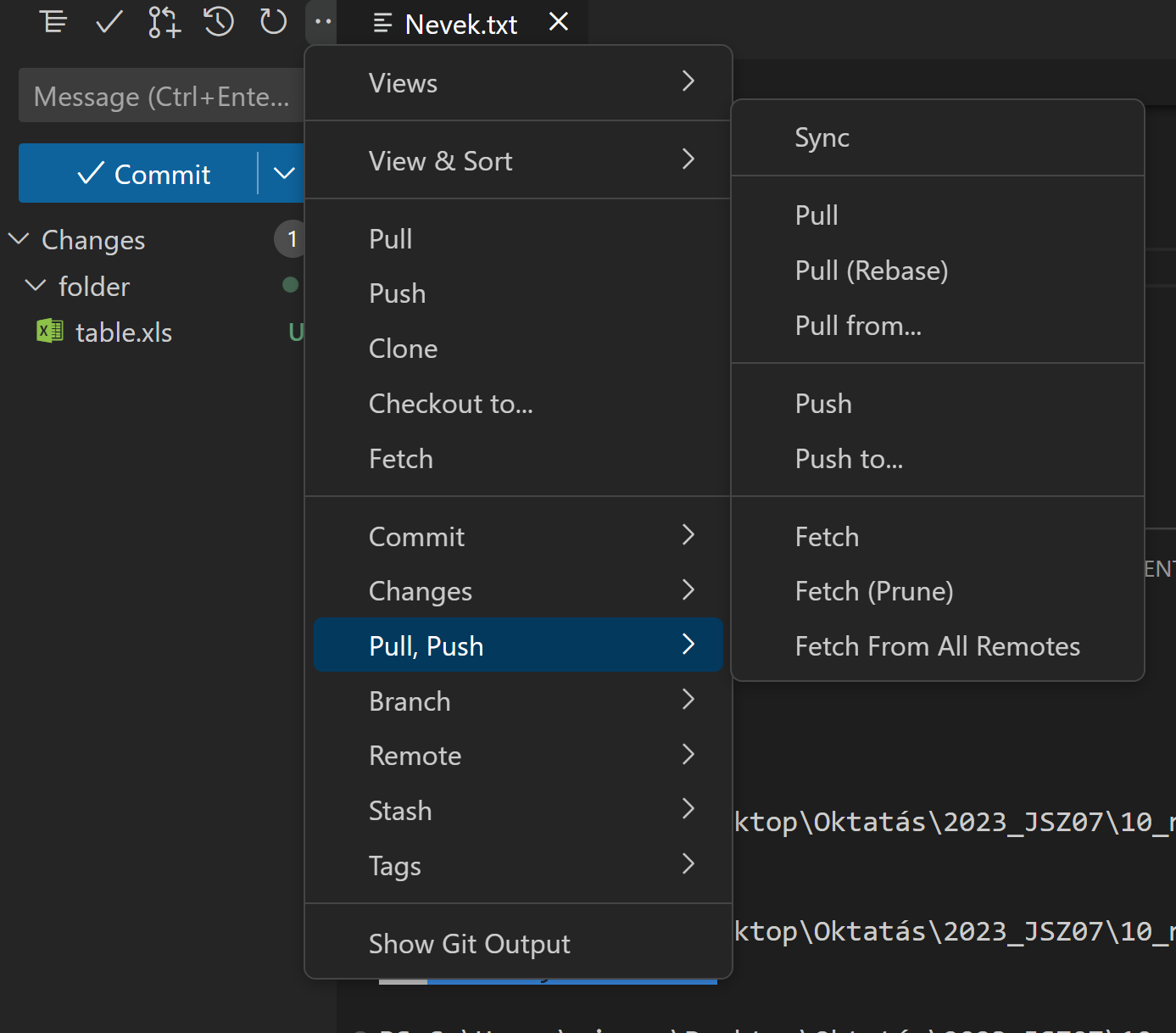
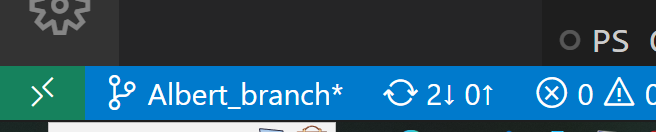
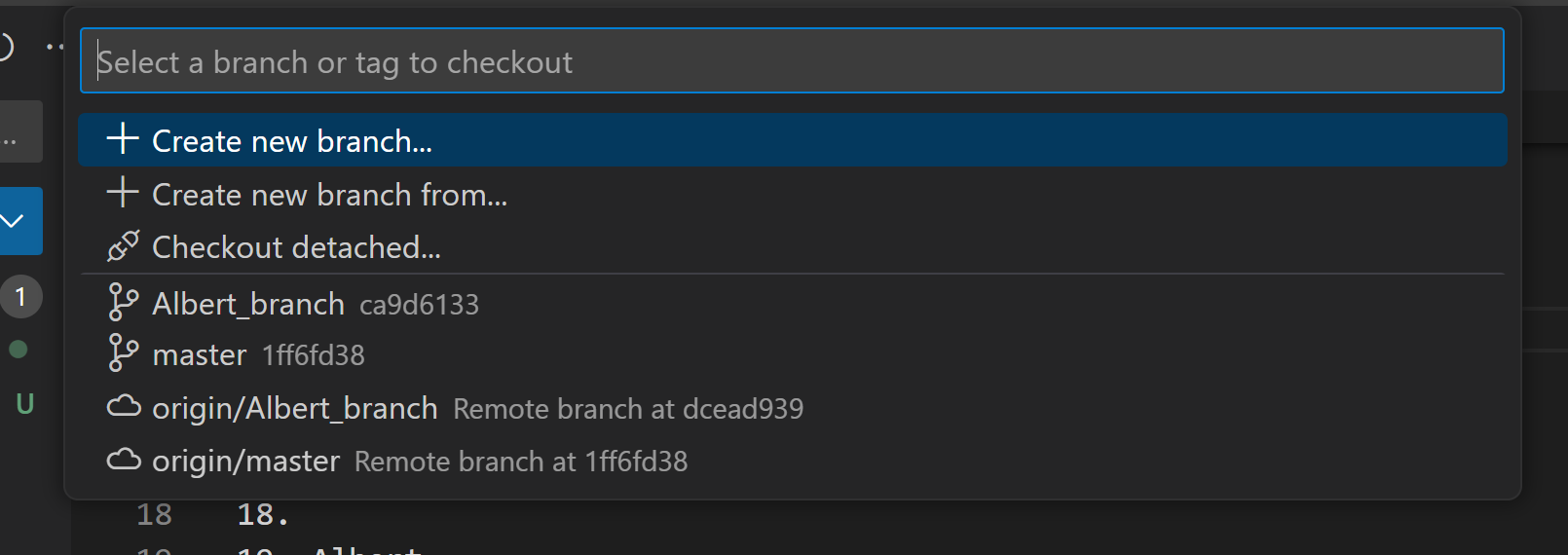
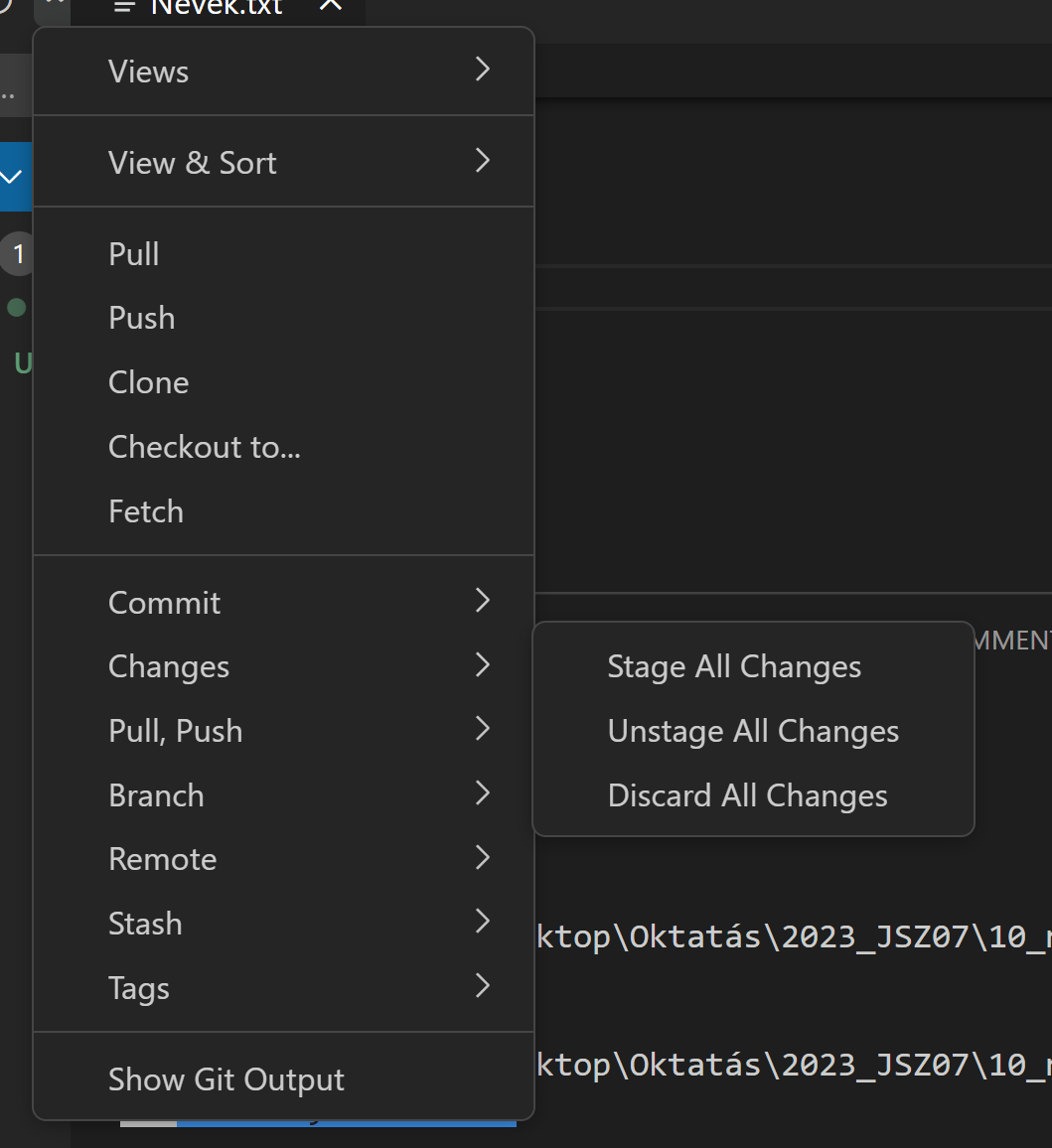
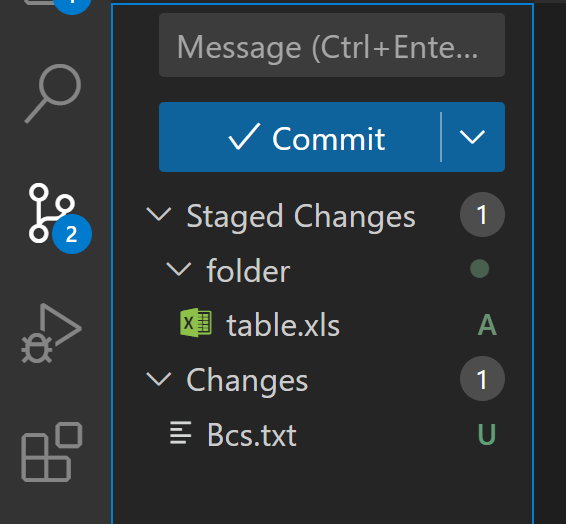
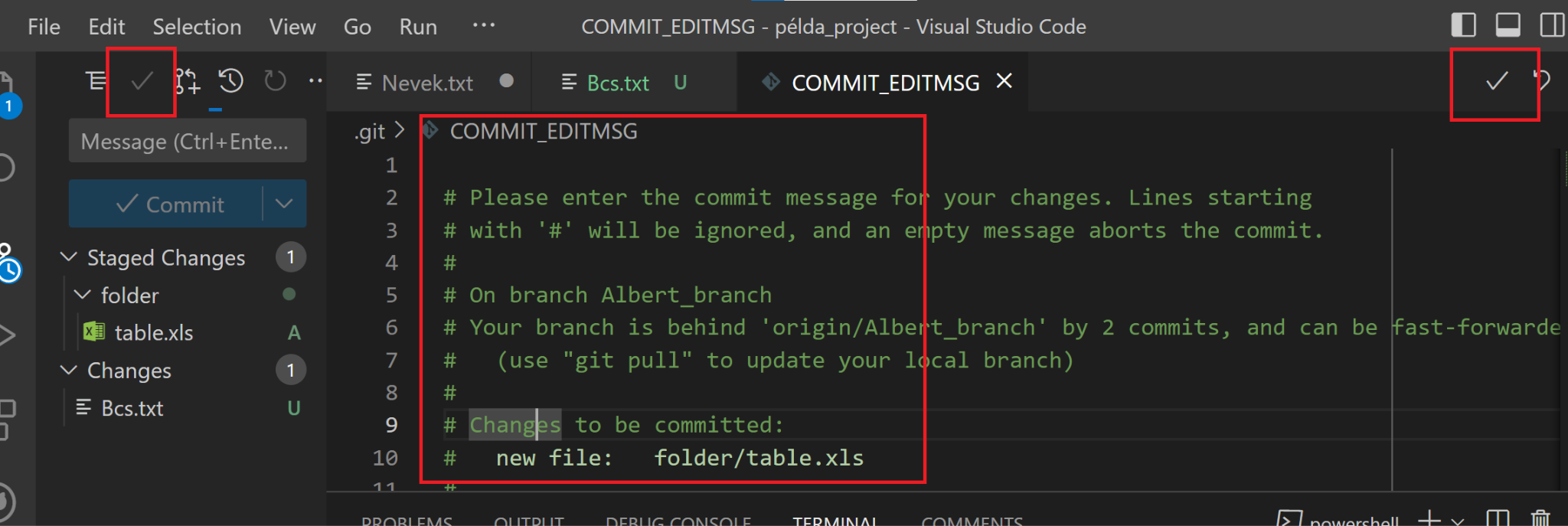
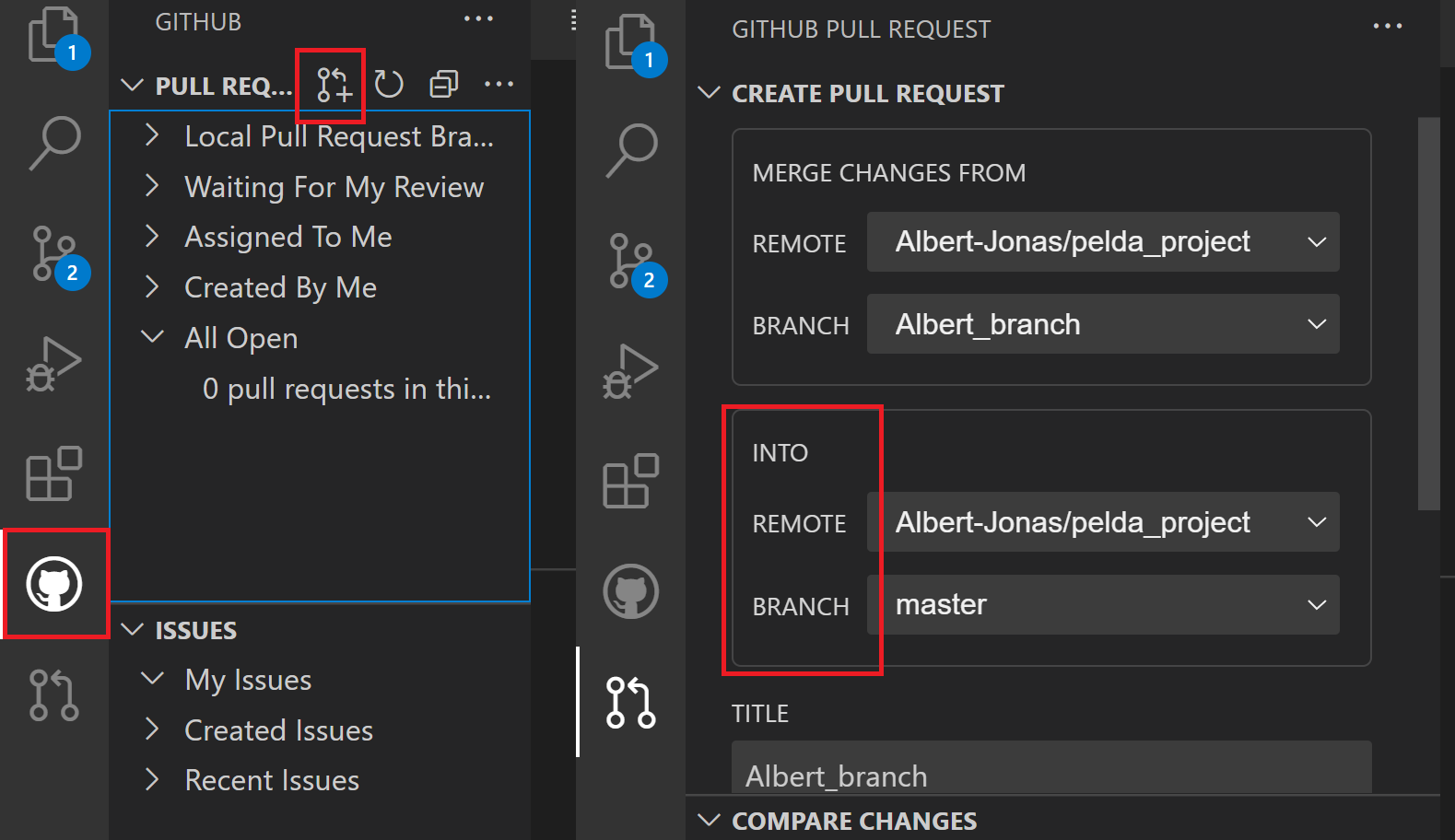
Fentiekhez használjuk a git-et, meg a githubot.

A remote teszt repó már létezik, ezt tudjuk magunkhoz klónozni és lokálisan kiegészíteni a saját tesztjeinkkel.

Kb ez a folyamat írható le a tesztíráshoz:

1. Kiválasztunk egy teszt task-ot, amivel foglalkozni szándékozunk
2. Azonosítjuk a teszteseteket, amiket meg fogunk valósítani
3. Frissítjük a repót
4. Létrehozunk, vagy átváltunk a branch-re, amin szeretnénk dolgozni
5. Ha használni akarunk olyat, amit addig nem (lásd feljebb), értesítjük róla a fejlesztőket
6. Lokálisan megírjuk a teszteket, vagy beillesztjük
7. Lokálisan ellenőrizzük, hogy a tesztek megvalósítják a tesztesetet
8. Frissítjük a repót és merge-eljük az esetleges új fejlesztéseket a branch-ünkbe
9. Lokálisan ellenőrizzük, hogy a tesztek megvalósítják a tesztesetet
10. Vagy kiírunk egy PR-t, vagy értesítjük a fejlesztőket, hogy kész vagyunk a tesztekkel
11. A PR vagy a branch lehetővé teszi a fejlesztőknek, vagy más tesztelőknek, hogy ellenőrizzék, hogy a tesztek ugyanazt az eredményt adják, mint lokálisan
12. Felvesszük a hibajegyeket, ha vannak

VS Code eszközök

* Repó frissítés
  + Egyirányú
    - 
    - Pull - frissíti a branch-et amin vagyunk és leszedi a többi branch remote adatait
    - Fetch - leszedi az összes remote branch adatait, de nem merge-eli bele az aktív branch-be az esetleges remote commitokat
    - További infó: https://stackoverflow.com/questions/292357/what-is-the-difference-between-git-pull-and-git-fetch
  + Kétirányú
    - 
    - Ez frissíti a remote-ot a locale-hoz képest és a locale-t a remote-hoz képest. (az aktív branch-en leszedi meg feltolja a commitokat)
* Branch műveletek
  + Váltás és készítés  
    
* Commit
  + Staging
    - 
    -   
      A change-eken ha hoverelsz a + és - jelekkel lehet stageingelni és unstagingelni.
  + Commit
    - 
    - Bal oldali pipával lehet elindítani a commitot
    - Középen a nem kommentelt sorok lesznek a cím, a kommentelt sorok a leírás
    - Jobb oldali pipával lehet befejezni a commitot
* Push
  + Frissítéssel lehet pusholni és pullolni egyszerre
* Pull request kiírás
  + VS code
    -   
      Cica ikonnal nyílik meg a felület  
      Fenti ikonnal tudsz PR-t csinálni  
      Jobb oldalon MINDIG ellenőrizzük, hogy HOVA akarunk merge-elni
  + Github
    - Push után van egy bazi nagy zöld gomb, hogy create pull request