

Nem kell külön fájlokba menteni, hanem egy fájlunk van és annak az előtörténetét, egyrészt mi nyomon tudjuk követni, hogy alakult másrészt visszatudunk tekerni ha valami rossz volt és onnan folytatni tovább. Így nem kell külön nyilvántartani.

Egy személyes verziókezelés.

Verziókezelni, egy személyes házi projektet is lehet.

Egy fájlnak a verziótörténetét nyílvántarthatjuk inkrementális lépésekben,vagyis ezt adtuk hozzá, ezt módosítottuk.Történt vele valami ezt nyílván tudjuk tartani.

A verziókezelés azért hasznos, mert egy fájlt látunk és az alkalmazás amiben mi a verziókat kezeljük abban tudjuk nézni ezen a napon, ezen a módosítással, ilyen kommentel ez történt.

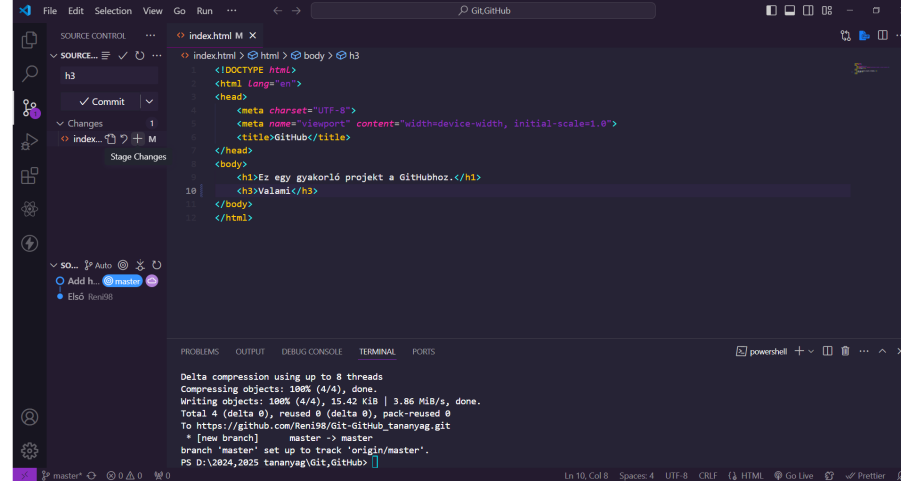
A másik amire jó megoldást ad a verziókezelés, a kollaboráció amikor nem egyedül dolgozunk rajta hanem csapatban.

Lehetővé teszi az együttműködést.

Verziókezelésnek van saját módszertana, erre a Git-et és konkrét szolgáltatása a GitHub.

Egy repository egy projekt és abban lehet sok fájlunk

Amit szeretnék egy commitba beletenni azt fájl szinten hozzáadom, akkor a Staged-be kerülnek, tehát odakészítődnek,hogy ezek a változások lesznek commitolva a commit-hoz tartozik egy üzenet és utána lehet commit-ra kattintani.

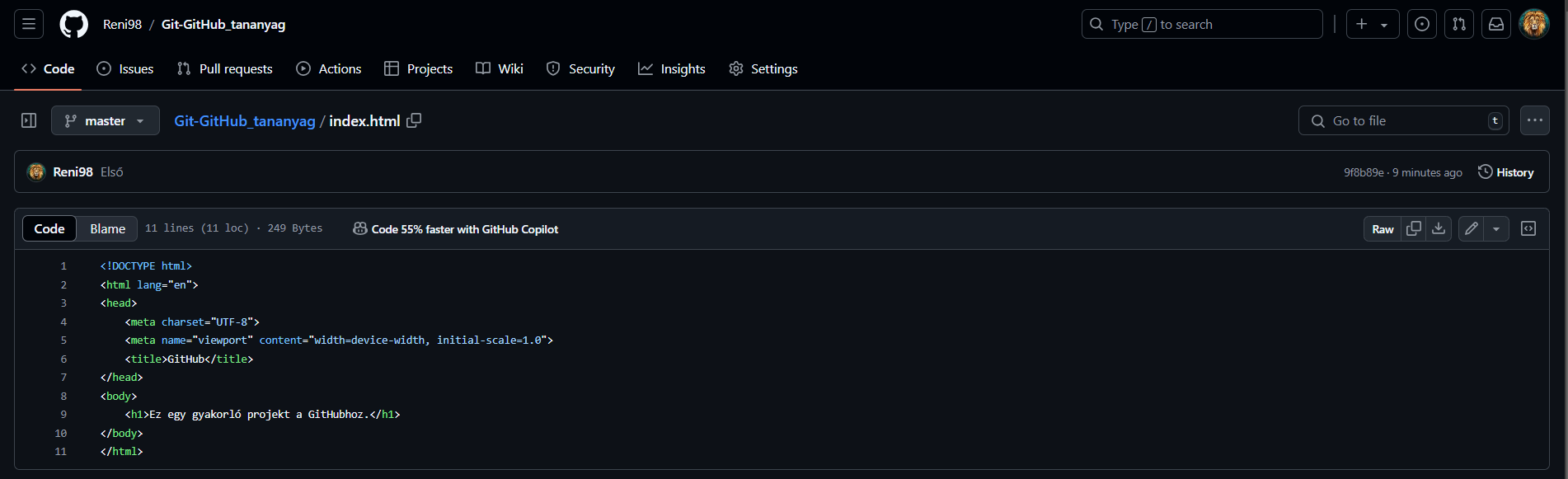


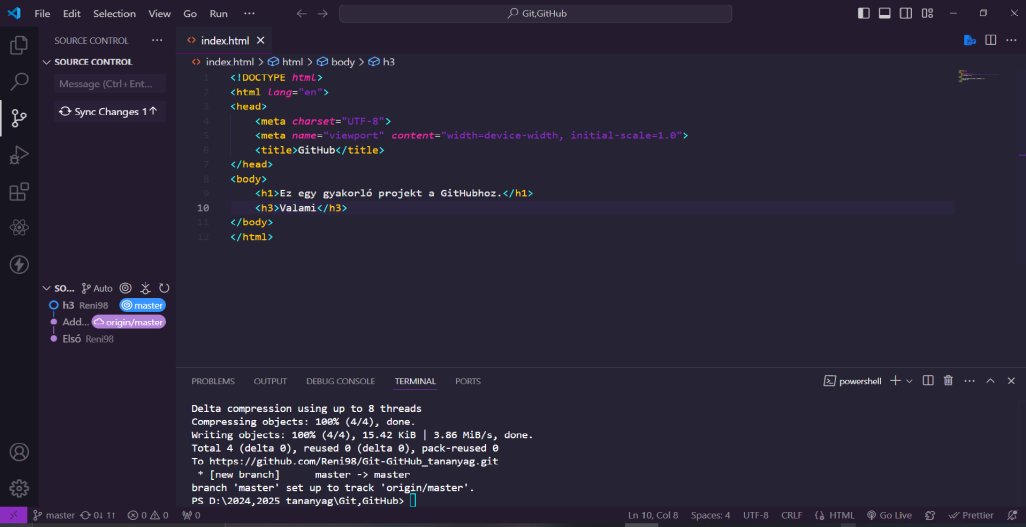
Lekönyvelt változás lett, ez egy állapot. Továbbra is egy fájlunk van de annak már 2 verziója.

Commitoltuk a változást de csak nálunk érhető el.

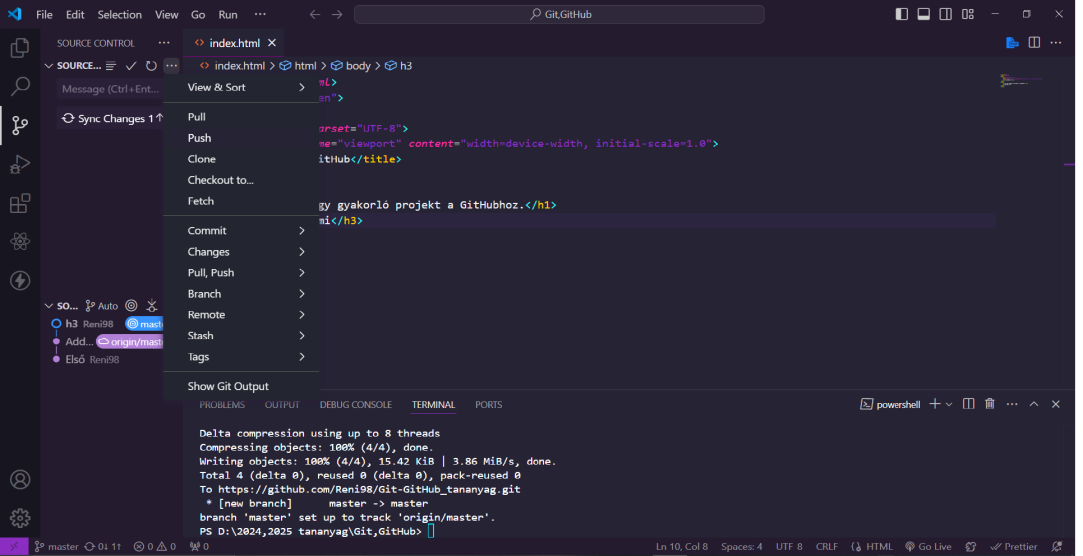
Ahhoz, hogy másik is hozzáférjenek pushol-nunk kell.

Origin a szerver, annak a main vagy master branch-ére szeretnénk feltenni.



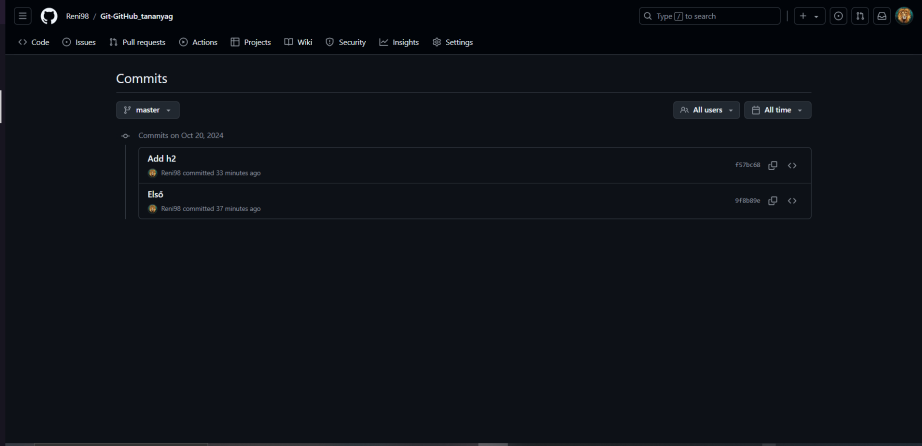


A Sync Change felteszi a mi módosításainkat, és lehúzza az újabb módosításokat is, tehát push-ol és pull-ol egyszerre..



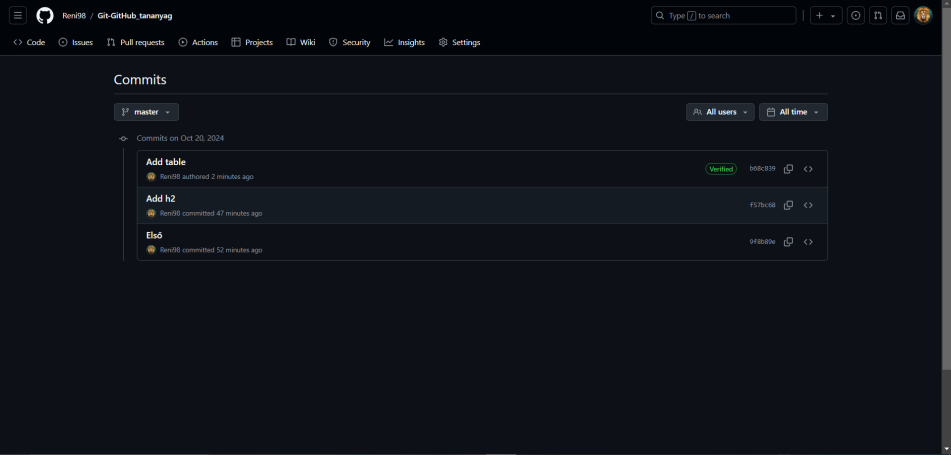
De lehet csak a push-ot is használni.

commit history(commitok története)

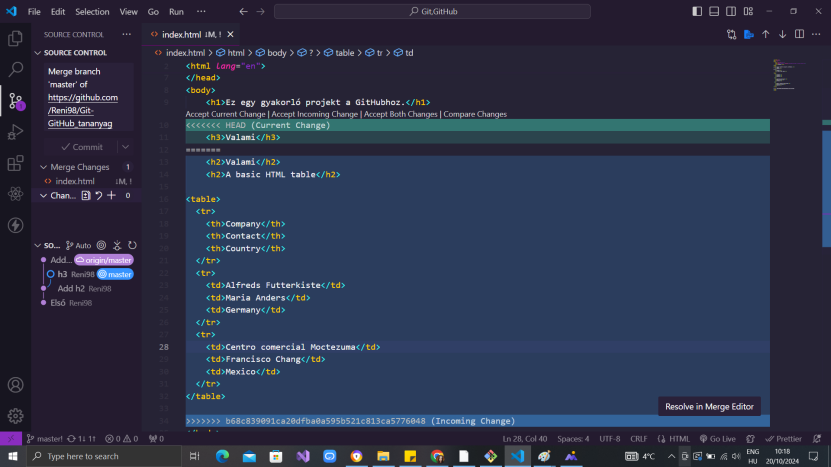


A commitok története azt mutatja meg, milyen változtatásokat végeztünk a projektünkben az idő során.

A commit egyszerre egy fotó, hogy itt tartottunk és a commit messages miatt üzenet, hogy mit csináltunk.



Git és verziókezelésben: A "Verified" státusz egy commit esetében azt jelzi, hogy a commit digitális aláírással lett ellátva, és a rendszer ellenőrizte, hogy az aláírás hiteles. Ez azt bizonyítja, hogy a commit valóban az adott személytől származik, és nem lett módosítva.



<<<<<<< HEAD: Ez a kódrészlet azt mutatja, hogy ez a tartalom jelenleg a te helyi verziódban van.

=======: Ez a vonal választja el a két konfliktusos tartalmat.

>>>>>>> b68c839091ca20dfba0a595b521c813ca5776048: Ez a kódrészlet jelzi a másik verzióból származó módosítást (pl. a távoli repozitóriumból vagy egy másik ágból).

A konfliktus akkor keletkezik, ha két különböző módosítást próbálsz egyesíteni egy fájl ugyanazon részén, és Git nem tudja automatikusan eldönteni, hogy melyik verziót kellene használnia.

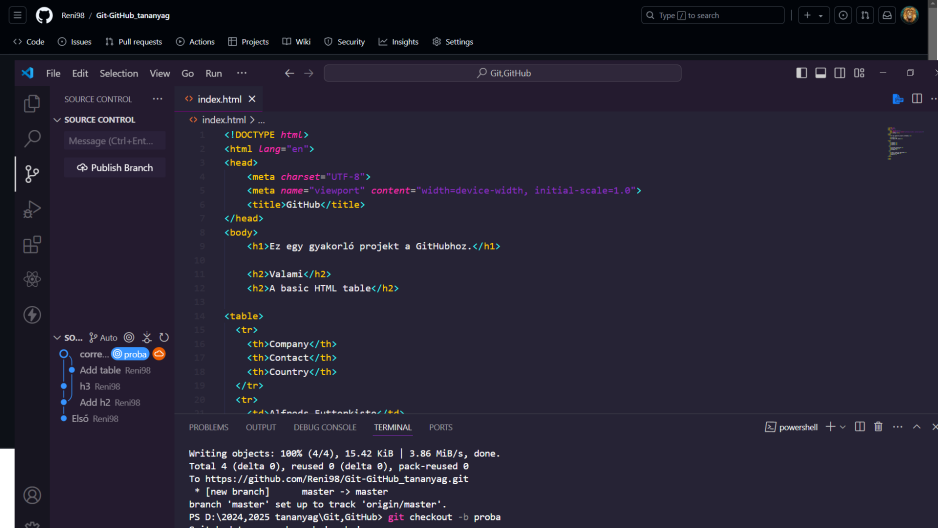
Main-be nem commitolunk, mert ezzel felül tudjuk írni mások munkáját!!!!

Tudunk csinálni különböző alverziókat. Mindenki egy saját verziócsomagon tud dolgozni.

Amíg a saját Branchaman dolgozok

Main-ről, jelenlegi állapotról leágazunk, nyitunk egy b,c stb tervet,elmondjuk mit szeretnénk ezzel a branch-al csinálni.

Ha jól csináltuk visszatudjuk származtatni a főágba, összefésüljük a kettőt, és ekkor minden ott lesz a mainen ami kell.

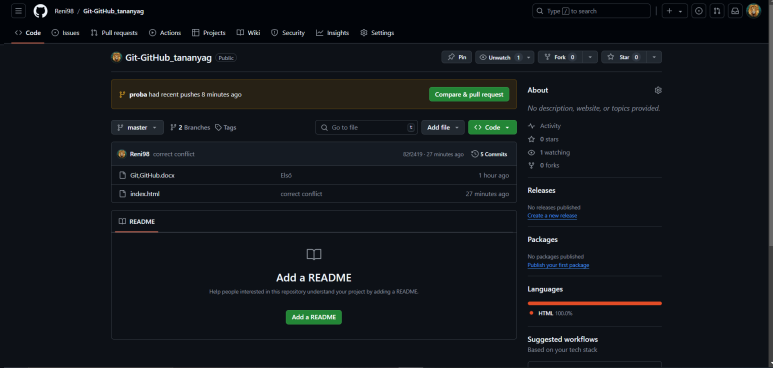
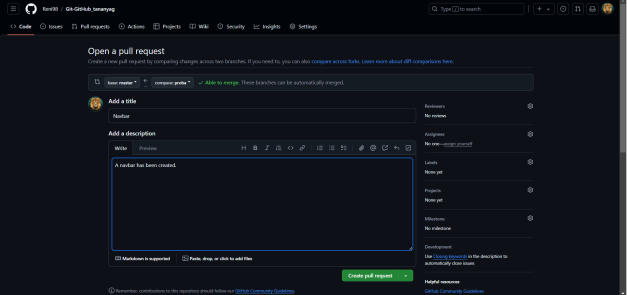
Új branch-et, ágat hoztam létre: git checkout -b új-branch-neve

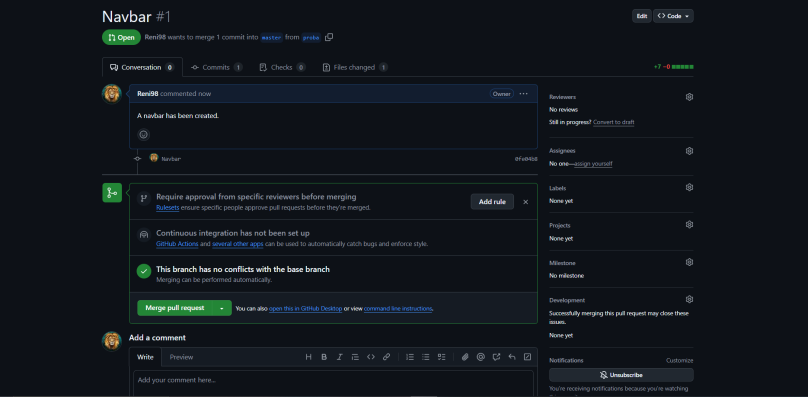
Git status-al megtudom nézni melyik branch-ben vagyok.

Publish-ra kell kattintanom ha azt szeretném, hogy mások is lássák az új brancham.

Hogyha működik majd rá lehet fűzni a main-re.

Adott feladathoz készül egy merge request.



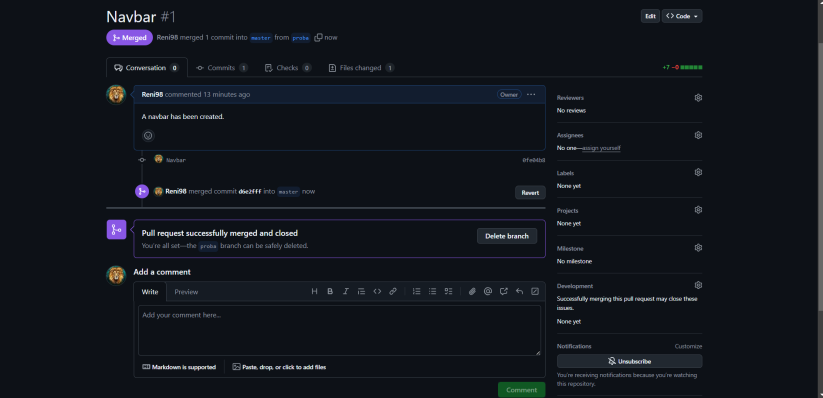
Meg lesz nézve manuálisan, hogy össze lehet-e őket mergelni conflict nélkül.

A conflict az amikor egyik branchen így a másikon amúgy változtattak és amikor összeadjuk a két változatatást akkor nem tudja a gép magától eldönteni melyik módostást kéne átvennie. Ekkor kell manuálisan merge konfliktusokat feloldani.

Egy személyes projektnél ellehet fogadni, viszont csapatban dolgozás esetén van egy személy aki review-ut csinál,Reviewers az a személy aki valamilyen módosítást, javítást kérhet.

Azért van, hogy a kódbázisunk jól működjön.

Be lehet állítnai amig nincs leokézva addig nem lehet mergelni és minden látszik, ott fog szerepelni, hogy ki fogadta el a merget.

Ha egy branch végzet, ellátta a feladatát, amit akartunk beraktuk a mainbe fő featurként működik,azt a branch-et lehet törölni. Vissza lehet tekerni a Revert-el ha nem működne.

A változások nem tűntek el, mert azok a commitok bekerültek a mainbe.

Repóban vannak fájlok, a fájlokat commitokban adjuk hozzá,modósítjuk, töröljük stb.

Brancheken fejlesztünk és általában van egy main branchunk ami tökéletes állapotban van. Vannak al ágaink és majd azokat mergel-jük vissza a mainre, merge vagy pull requistekkel.