Háttéralkalmazások 1. Házi feladat

A feladat specifikációja

A feladat egy elképzelt logisztikai alkalmazás adatelérési és üzleti logikai rétegének részleges megvalósítása. A logisztikai cég szállítás terveket (TransportPlan) készít, amelyek azt foglalják össze, hogy egy adott szállítmányt milyen szakaszokon (Section) keresztül, milyen mérföldköveket (Milestone) érintve terveznek kiszállítani. Az adatmodell entitásainak rövid leírása:

- TransportPlan: egy szállítási tervet reprezentál, szakaszokat (Section) tartalmaz. Egyedi azonosítóval rendelkezik, és tárolja a kapcsolódó megbízások azonosítóit.
- Section: egy szállítmány egy szakaszát reprezentálja, start- és végmérföldkő tartozik hozzá (fromMilestone, toMilestone). A number mező mutatja meg, hogy ő hányadik szakasz a szállítási tervben. (0-tól számozódva.)
- Milestone: egy mérföldkő a szállítás során. Hivatkozik egy címre (Address) és tartalmazza, hogy a terv szerint mikor kell elérni az adott mérföldkövet (plannedTime). Az időpont mindig helyi idő, az időzónát nem kell eltárolni.
- Address: egy cím adatait (ország 2 betűs ISO kódja, város, utca, irányítószám, házszám, szélesség, hosszúság) tárolja.

Minden entitásnak generált long elsődleges kulcsa van.

A projekt vázban látható, hogy a fenti négy entitás már készen áll, valamint néhány üzleti logikai osztály is. Bővítsd ki az alkalmazást az alábbi feladatok megoldásával:

1. Írj egy új entitást (Partner), amely a logisztikai cég partnereit tudja tárolni. A partnernek az azonosítón kívül van neve, kapcsolattartó e-mail címe és címe, amely értelemszerűen egy Address entitás. (1 pont)
2. Az egyes mérföldköveken különféle tevékenységeket (akár többet) kell elvégezni. Írj egy entitást (Action), amely az adott mérföldkőnél elvégzendő tevékenységeket tudja tárolni. A tevékenységről tudjuk, hogy
    a. mikor végezték el helyi idő szerint (null, ha még nem végezték el),
    b. milyen típusú tevékenység: bepakolás/kipakolás/vámkezelés/ellenőrzés

Definiálj kétirányú kapcsolatot az Action és a Milestone között a megfelelő multiplicitásokkal! (2 pont)

3. A kiszállítási tervekben minden szakaszt (Section) a logisztikai cég valamelyik partnere (Partner) fogja teljesíteni. Hozd létre a teljesítést reprezentáló Compliance entitást a megfelelő kapcsolatokkal! A Compliance entitás a generált elsődleges kulcson és a kapcsolatokon kívül tárolja a kapcsolódó számla string

típusú azonosítóját, és egy mezőt, ami azt jelzi, hogy megtörtént-e már a teljesítés. (2 pont)

4. Bővítsd az AddressRepository (Java), illetve AddressService (.NET) típust az alábbi metódusokkal:
    c. A metódussal adott városban lévő címek kérdezhetők le. A városban kis/nagybetű ne számítson, de ezt leszámítva pontos egyezés szükséges (1 pont)
    d. A metódus egy adott "téglalapon" belül eső címeket adja vissza, ha megadjuk a téglalap bal felső és jobb alsó sarkainak szélesség/hosszúság koordinátáit. (1 pont)
    e. A metódus utca átnevezéseket képes kezelni: paraméterként kapott országkód/irányítószám/utcanév hármasra pontosan illeszkedő címekben nevezi át az utcát a szintén paraméterként kapott új névre (1 pont)

A lekérdezések tesztelésére célszerű az F4_AddressRepositoryIT (Java), illetve F4AddressServiceIT (.NET) (IT = Integration Test) osztály teszt metódusait futtatnod. A teszt osztály használatához az elején ki kell töltened a TODO metódusokat oly módon, hogy az általad írt repository metódusokat meghívod.

5. Valósítsd meg a TransportPlanService osztály alábbi metódusait!
    f. getFirstAndLastMilestone: A metódus adjon vissza egy kételemű Milestone listát, amelyben az adott id-jű szállítási terv legelső és legutolsó mérföldköve található. (1 pont) Ha a tervben még nincs szakasz, akkor üres listát adjon vissza! Ha az adott id-hez nem létezik terv, dobjon (IllegalArgumentException-t / ArgumentException-t)! (1 pont)
    g. registerDelay: a metódus egy adott terv adott mérföldkövénél regisztrál egy percekben megadott várható késést.
        i. Nem létező szállítási terv, vagy mérföldkő esetén kivételt kell dobni (IllegalArgumentException-t / ArgumentException-t) (1 pont)
        ii. Növeld meg az adott mérföldkőnél tervezett időpontot a késés hosszával! (1 pont)
        iii. Ha a mérföldkő a kezdő mérföldkő a szakaszon belül, akkor annak a szakasznak a végmérföldkövénél tervezett időt is növeld meg a késés hosszával! (1 pont)
        iv. Ha a mérföldkő egy szakaszon belüli végmérföldkő, akkor a következő szakasz kezdő mérföldkövének tervezett idejét növeld a késés hosszával! (1 pont)
    h. addSection: Ezzel egy létező szállítási terv szakaszai közé tudunk egy új szakaszt beszúrni, adott sorszámmal (number), két adott id-jű milestone között. A következő szabályokat kell betartani:
        v. Ha nem létezik a szállítási terv, vagy a mérföldkövek közül bármelyik, kivételt kell dobni. (IllegalArgumentException-t / ArgumentException-t) (1 pont)

vi. Az új szakasz sorszáma 0 és MAX között lehet, ahol MAX a tervhez tartozó szakaszok beszúrás előtti darabszáma. Ha ez nem teljesül, dobódjon IllegalArgumentException / ArgumentException. (1 pont)

vii. Ha helyes a sorszám és létezik a szállítási terv és mindkét mérföldkő, mindenképpen új szakaszt kell létrehozni, nem kell azzal foglalkozni, hogy a megadott milestone-ok között esetleg már létezik egy másik szakasz.

viii. A szakasz beszúrása történhet a meglévő szakaszok elé, után, és közé is. Minden esetben meg kell tartani a szakaszok folyamatos sorszámozását. Pl. ha eddig 0, 1, 2-es szakaszok voltak, és az 1-es numberrel szúrunk be, akkor a korábbi 1-es, 2-es numberű szakasz új sorszáma 2-es és 3-as legyen. (Shifteljük a későbbi szakaszokat.)

ix. Az új szakasz előtti vagy utáni szakaszok (ha léteznek ilyenek) mérföldköveit módosítani kell: ha pl. a 0-s szakasz A->B-be vitt, az 1-es szakasz pedig B->C volt, és 1-es helyre szúrjuk be a D->E szakaszt, akkor a 0-s szakasznak A->D, a korábbi 1-es (most már 2-es) szakasznak E->C mérföldköveket kell tartalmaznia.

x. További pontok a tesztesetek alapján:
- helyes beszúrás üres tervbe (1 pont)
- Helyes beszúrás nem-üres terv végére (1 pont)
- Helyes beszúrás nem-üres terv közepére (2 pont)

Az F5a_TransportPlanServiceGetFirstAndLastMilestoneIT, F5b_TransportPlanServiceRegisterDelayIT, F5c_TransportPlanServiceAddSectionIT (Java), illetve F5TransportPlanServiceIt (.NET) teszt osztályokban található metódusok viszonylag alapos tesztelést végeznek. Javasolt tehát ennek futtatása, hogy ellenőrizd a megoldásodat, nem kell adatbázisba teszt adatokat beszúrni, stb.

Tipp: a nem végleges megoldást is már érdemes tesztelned, hogy lásd, mely tesztesetek mennek már át.

A projekt váz elérése, a megoldás beadása

Döntsd el, hogy Java vagy .NET platformon akarod megvalósítani ezt a házi feladatot. (A második házi feladatnál majd a másikat kell választanod.) A választástól függően más-más GitHub Classroom assignmentet kell elfogadnod:

- Java: https://classroom.github.com/a/k1a2eowg
- .NET: https://classroom.github.com/a/CUwQJmSC

Az assignment elfogadása után létrejön a te privát repositoryd amelyben benne lesz a projekt váz.

A repo gyökerében lévő README.md tartalmaz egy kis útmutatót a projekt beüzemeléséhez, használatához. A repoban szabadon dolgozhatsz, az alábbi megkötésekkel:

- A master branchbe nem pusholhatsz, hozz létre egyből a munka elején egy saját branch-et (pl. megoldas néven), és abba dolgozz
- A repo gyökerében lévő neptun.txt fájlba írd bele a Neptun kódodat(ezt is egyből az elején tedd meg)!
- A teszt osztályokat nem módosíthatod, (kivéve, ahol ezt komment jelzi)!
- Amikor véglegesnek ítéled a megoldásodat, hozz létre egy Pull requestet, és rendeld hozzá a megfelelő oktatót. Java esetén: Kövesdán Gábor (gaborbsd), .NET esetén Hideg Attila (SolisarAUT).

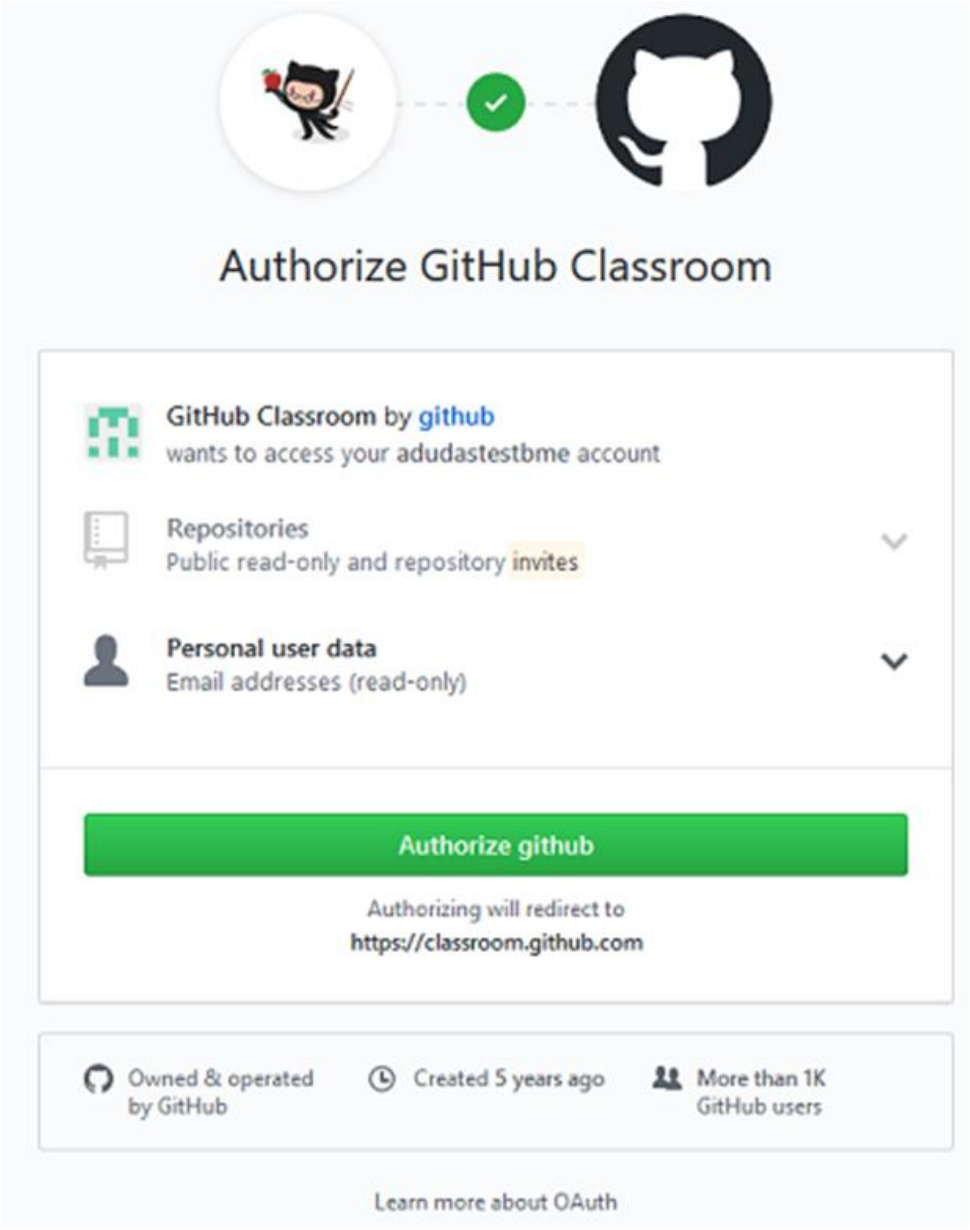
Részletes útmutató a fenti lépések elvégzéséhez

Az alábbi screenshotok egy másik tárgyhoz készültek, így értelemszerűen némileg más tartalmat látsz majd, de a működés lényegében ugyanez.

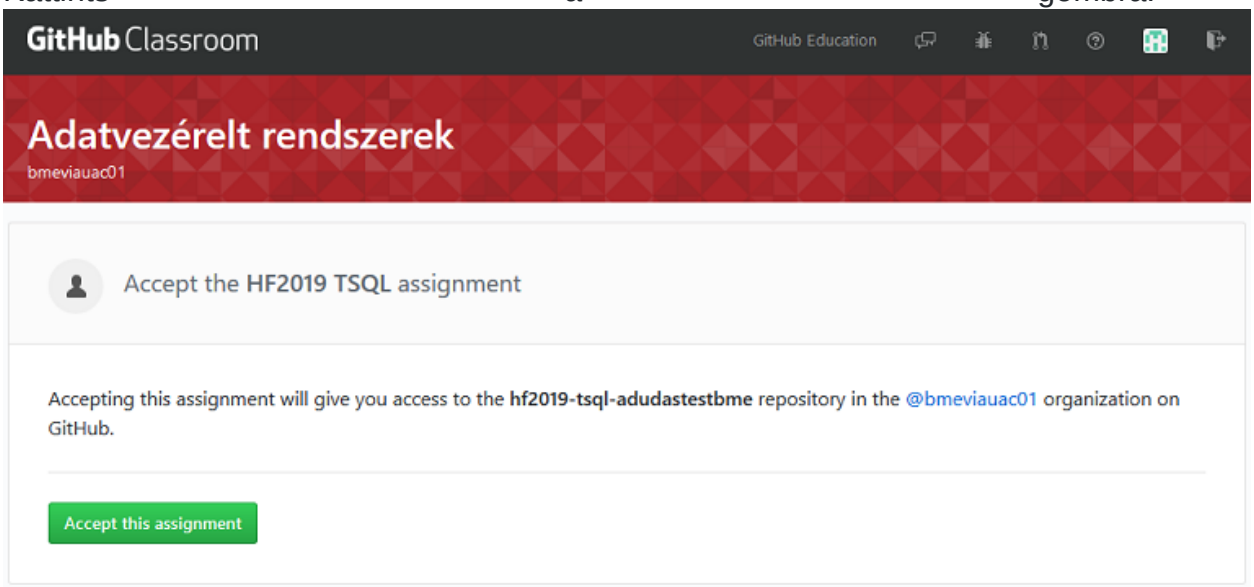
Assignment elfogadása

6. Regisztrálj egy GitHub accountot, ha még nincs.
7. A feladat beadásához tartozó linket nyisd meg.

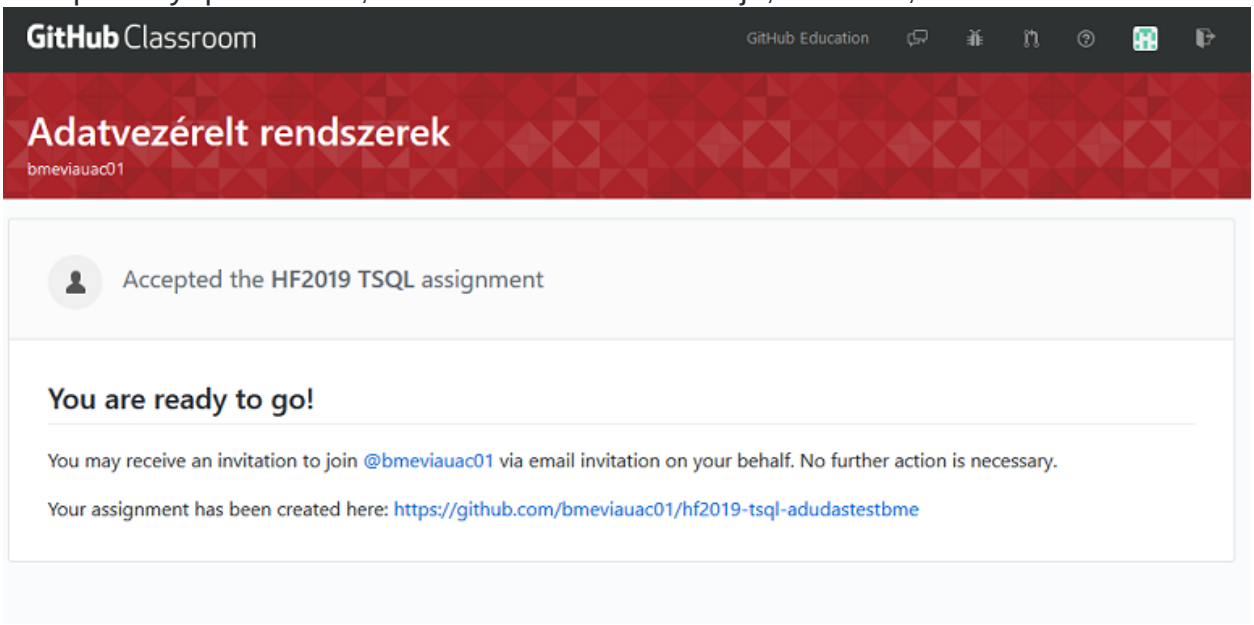
8. Ha kéri, adj engedélyt a *GitHub Classroom* alkalmazásnak, hogy használja az account adataidat.



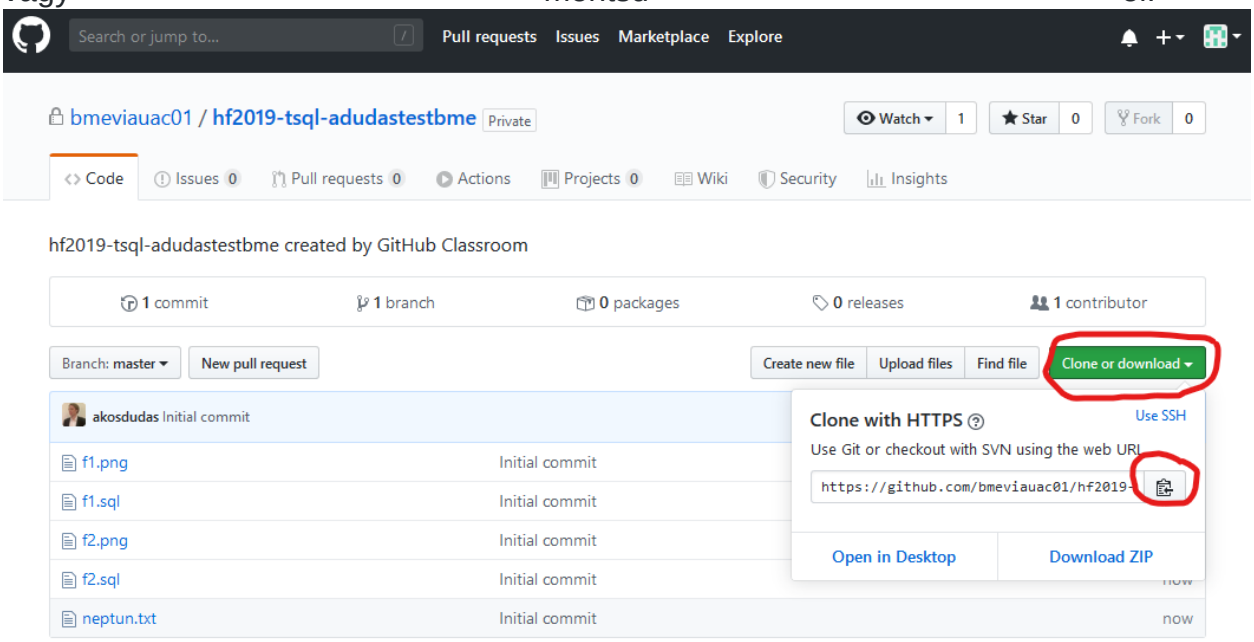
9. Látni fogsz egy oldalt, ahol elfogadhatod a feladatot (“Accept the … assignment”). Kattints a gombra.



10. Várd meg, amíg elkészül a repository. A repository linkjét itt kapod meg. A repository privát lesz, azaz az senki nem látja, csak te, és az oktatók.

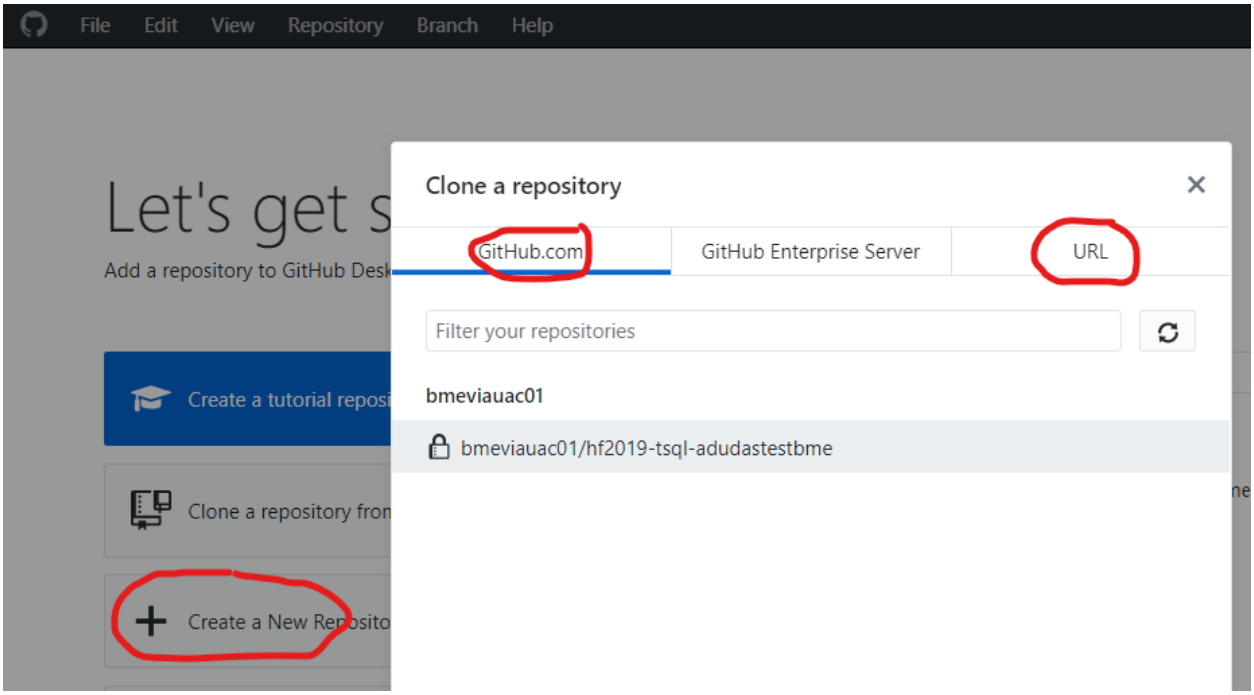


11. Nyisd meg a repository-t a webes felületen a linkre kattintva. Ezt az URL-t írd fel, vagy mentsd el.



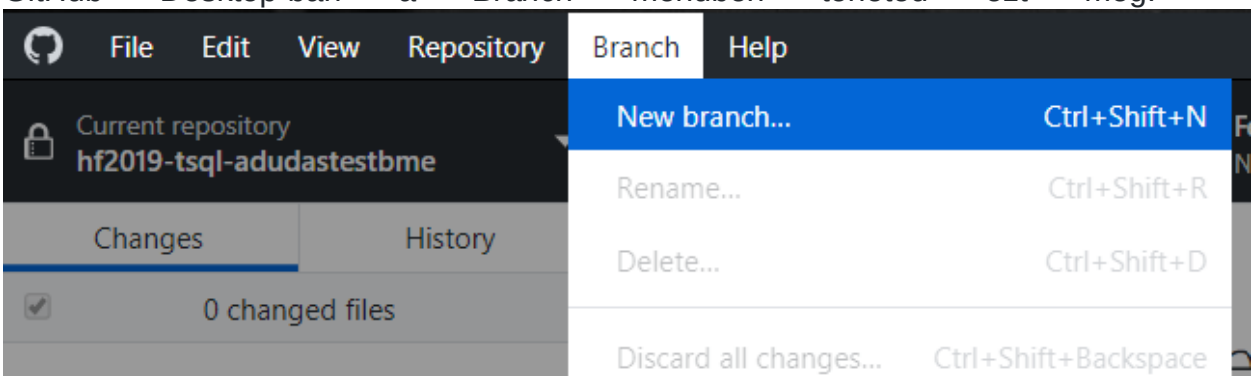
Repository klónozása, megoldas branch létrehozása

12. Klónozd le a repository-t. Ehhez szükség lesz a repository címére, amit a repository webes felületén a *Clone or download* alatt találsz. A git repository kezeléséhez tetszőleges klienst használhatsz. Ha nincs kedvenced még, akkor legegyszerűbb a GitHub Desktop. Ebben az alkalmazásban közvetlenül tudod listázni a repository-kat GitHub-ról, vagy használhatod az URL-t is a klónozáshoz.



Ha konzolt használnál, az alábbi parancs klónozza a repository-t (ha a `git` parancs elérhető): `git clone <repository link>`

13. Ha sikerült a klónozás, MÉG NE KEZDJ EL DOLGOZNI! A megoldást *ne* a repository `master` ágán készítsd el. Hozz létre egy új ágat (branch) `megoldas` néven.
    GitHub Desktop-ban a *Branch* menüben teheted ezt meg.



    Ha konzolt használsz, az új ág elkészíthető ezzel a paranccsal: `git checkout -b megoldas`

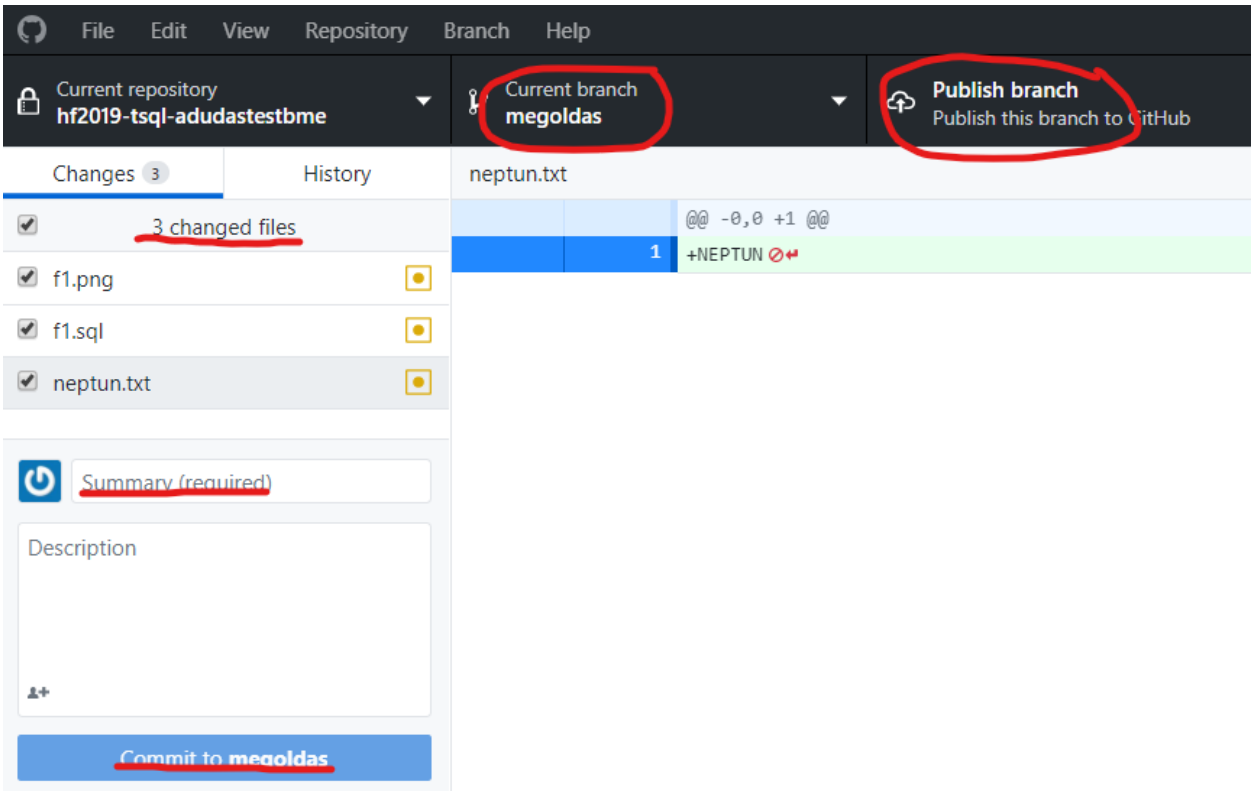
14. Ezen a megoldás ágon dolgozva készítsd el a beadandókat. Akárhányszor kommitolhatsz és pusholhatsz.
    Ellenőrizd, hogy a megfelelő névvel és email címmel kommitolsz-e. Ezt a következő command line paranccsal tudod megtenni.
    git config user.name

```
git config user.email
```

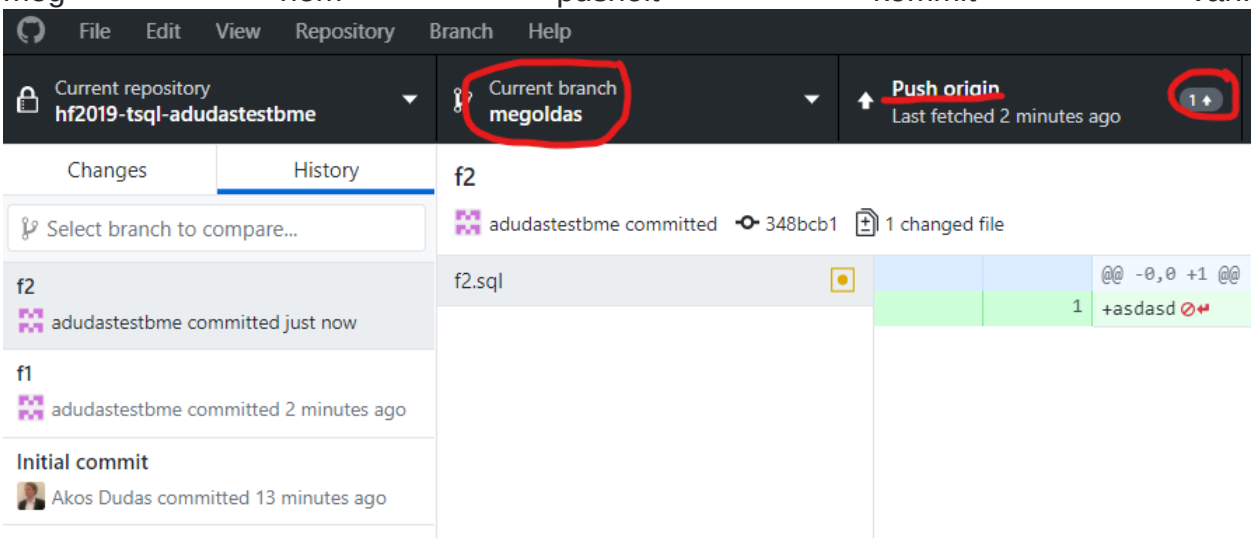
Ha ez nem megfelelő lenne, akkor add ki az alábbi parancsokat a git repository mappájában. Ezzel az adott repository-ra fogod beállítani a kívánt nevet és email címet. (Érdemes olyan email címet, megadni ami a github useretekhez van rendelve)
git config user.name `"John Doe"`

```
git config user.email "john@doe.org"
```

TIP: Érdemes lehet a globális beállításokat vizsgálni és felülírni a `--global` kapcsolóval. GitHub Desktop-ban így tudsz kommitolni. Mindig ellenőrizd, hogy jó ágon vagy-e. Első alkalommal a *megoldas* ág csak helyben létezik, ezért publikálni kell: *Publish this branch*.



A további kommitoknál is mindig ellenőrizd a megfelelő ágat. Ha egy kommit még nincs felöltve, azt a *Push origin* gombbal teheted meg. A kis szám a gombon jelzi, hogy hány, még nem pusholt kommit van.



Ha konzolt használsz, akkor az alábbi parancsokat használd (feltéve, hogy a jó ágon vagy):

Ellenőrizd az ágat, és hogy milyen fájlok módosultak

```
git status
```

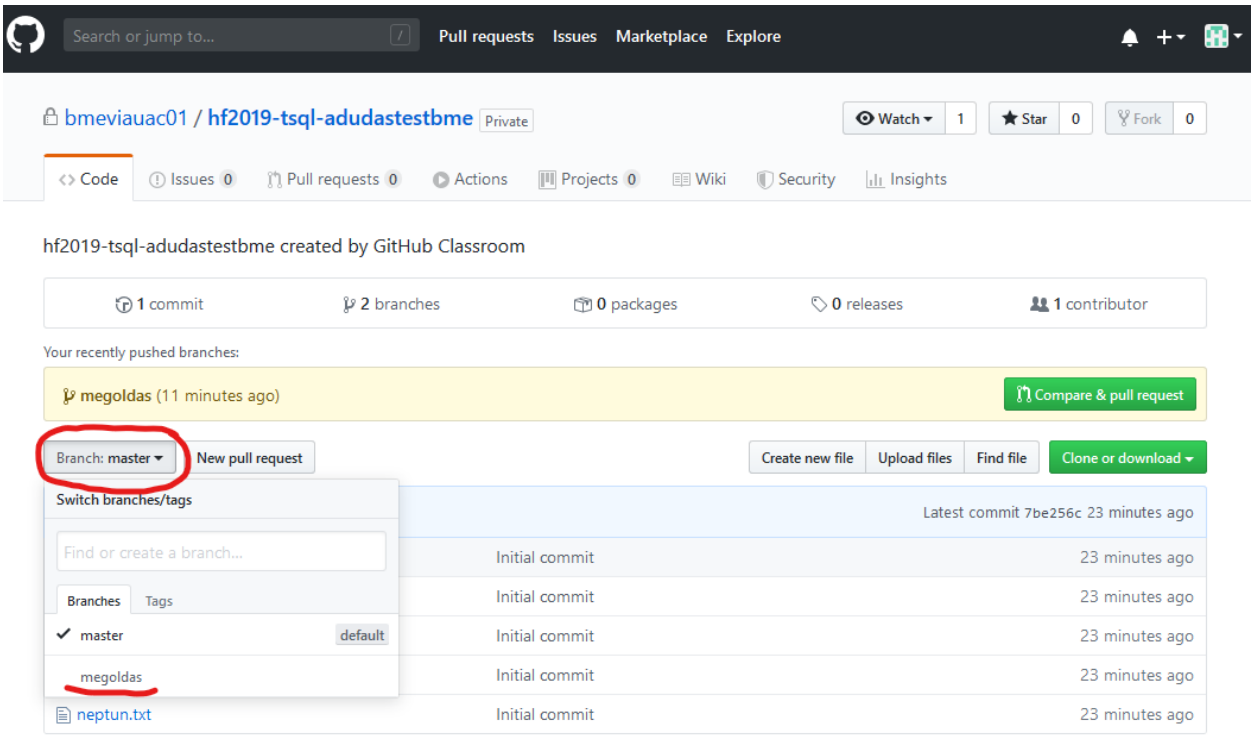
```
# Minden változtatást előkészít kommitolásra
git add .


# Kommit
git commit -m "f1"


# Push első alkalommal az új ág publikálásához
git push --set-upstream origin megoldas


# Push a továbbiakban, amikor az ág már nem új
git push
```

A neptun.txt kitöltése

A projekt gyökerében megtalálható a neptun.txt fájl, ebbe írd bele a neptun kódodat, majd az előzőek szerint commitold és pushold a módosítást.

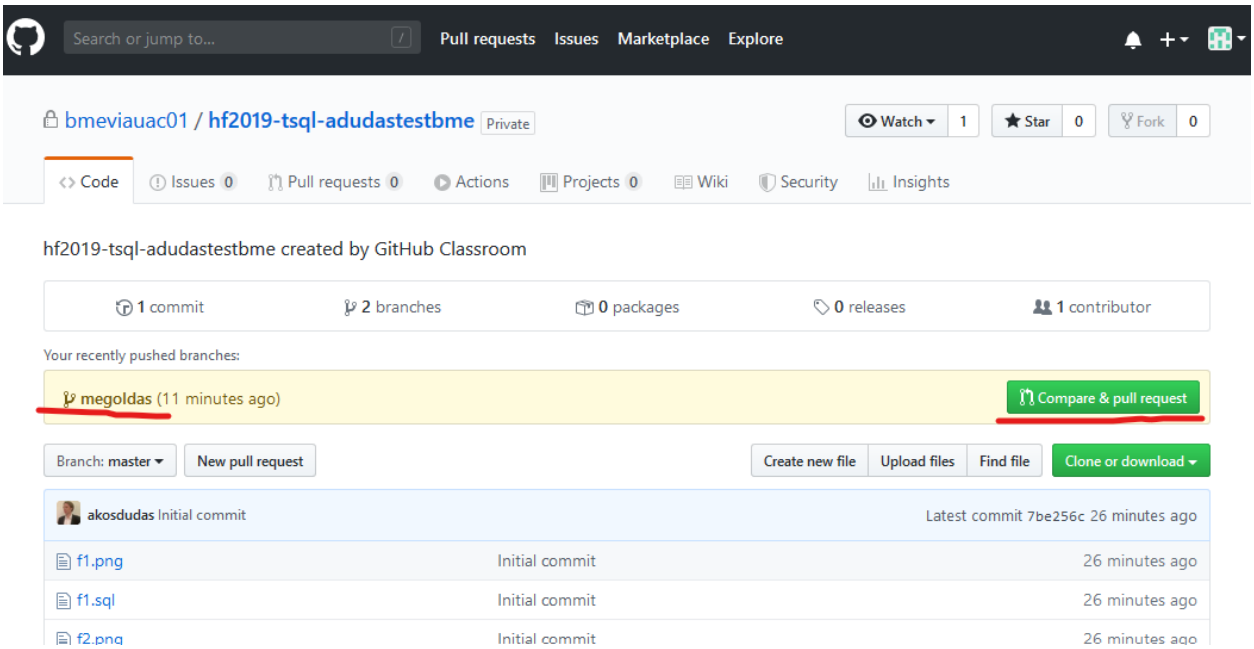
A megoldás beadása

15. Ha végeztél a megoldással, ellenőrizd a GitHub webes felületén, hogy mindent feltöltöttél-e. Ehhez a webes felületen váltanod kell az ágak között.

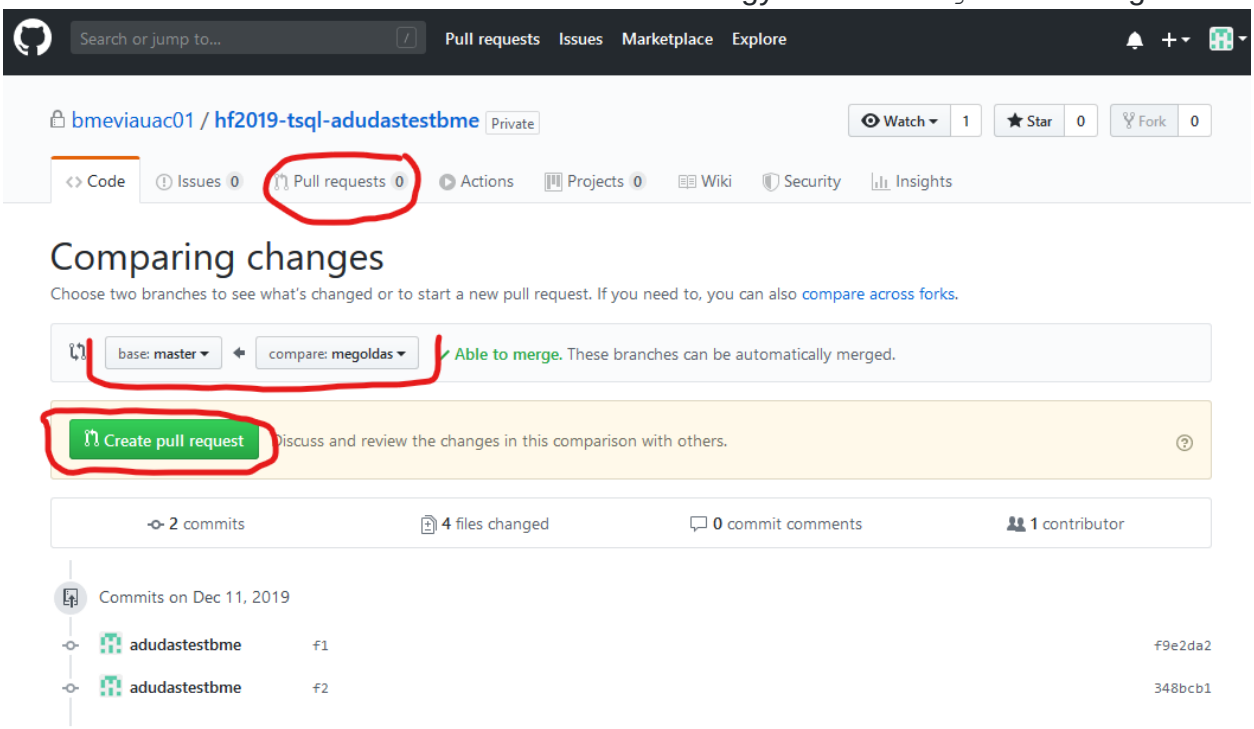


Arra kérünk, hogy NE használd a GitHub fájl feltöltés funkcióját. Ha valami hiányzik, a helyi git repository-ban pótold, és kommitold majd pushold.

16. Ha tényleg kész vagy, akkor nyiss egy *pull request*-et. Ez a *pull request* fogja össze a megoldásodat, és annak "végeredményét" mutatja. Így a laborvezetőnek nem az egyes kommitjaidat vagy fájljaidat kell néznie, hanem csak a releváns, változott részeket látja egyben. A *pull request* jelenti a feladatod beadását is, így ez a lépés nem hagyható ki.
A *pull request* nyitásához a GitHub webes felületére kell menned. Itt, ha nem rég pusholtál, a GitHub fel is ajánlja a pull request létrehozását.

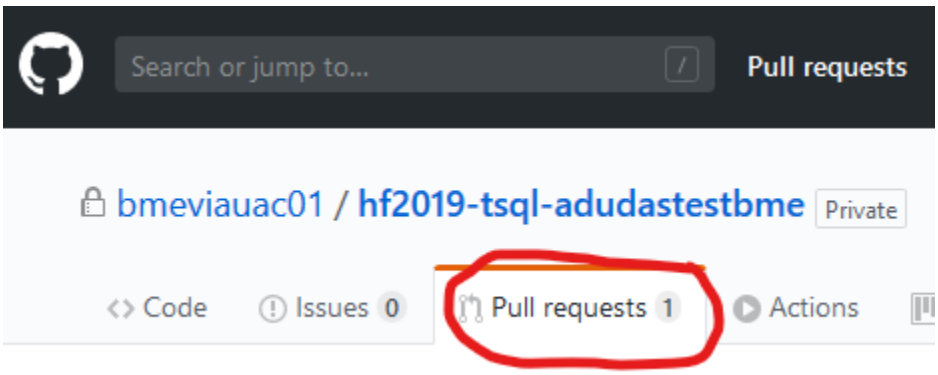


A *pull request*-et a fenti menüben is létrehozhatod. Fontos, hogy a megfelelő brancheket válaszd ki: `master`-be megy a `megoldas` ág.

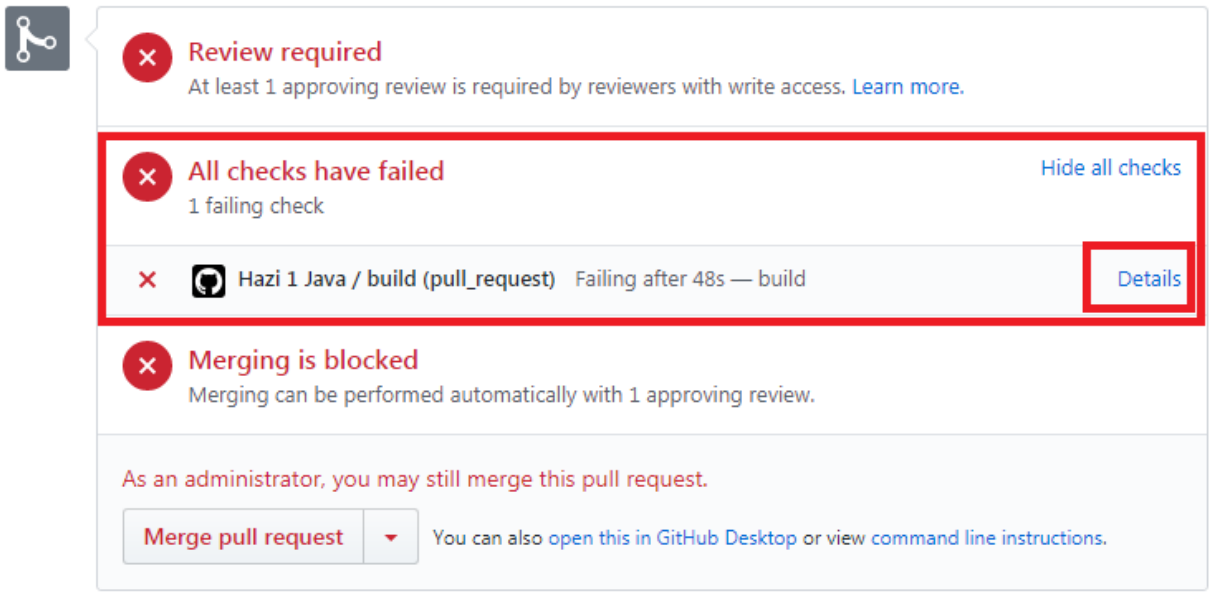


17. Ha minden rendben sikerült, a menüben fent látod a kis “1” számot a *Pull request* elem mellett, jelezve, hogy van egy nyitott pull request. DE MÉG NEM

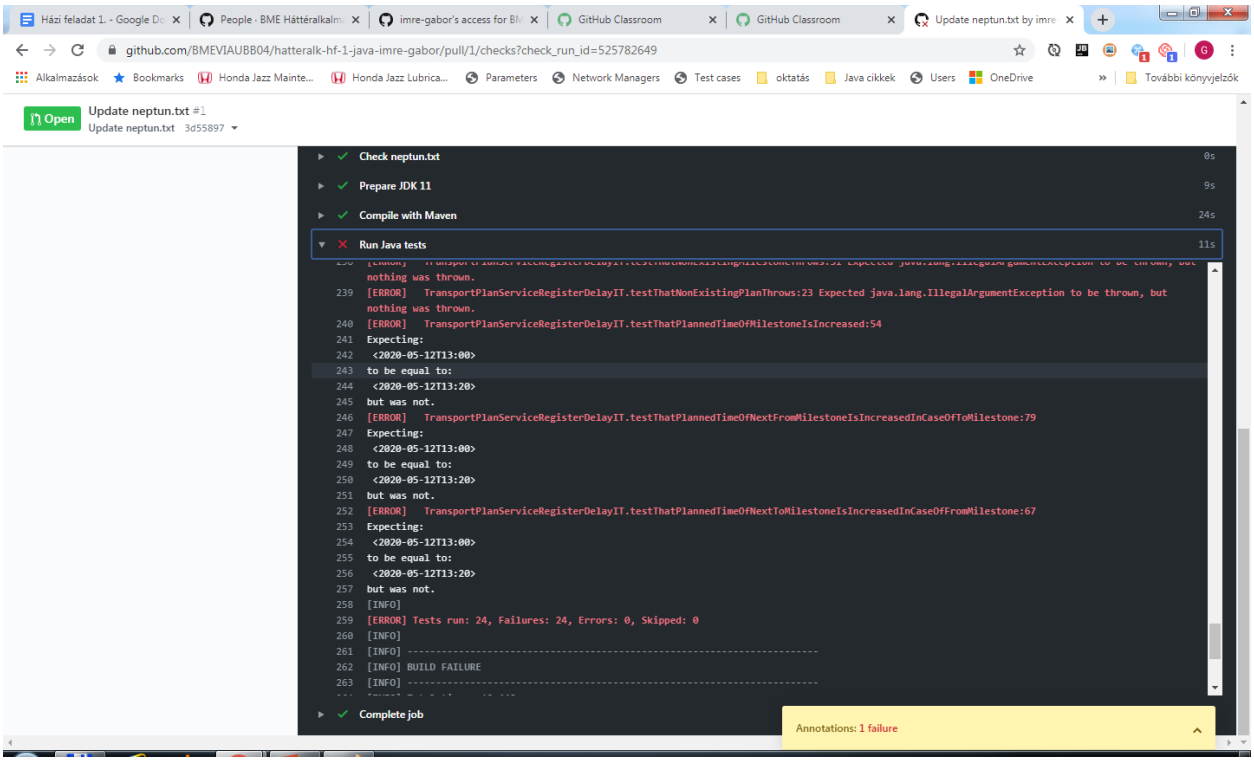
VÉGEZTÉL!



18. A *pull request* létrehozásának hatására le fognak futni a tesztek. Ennek eredményét a pull request alatt a "checks" blokkban fogod látni, pl.

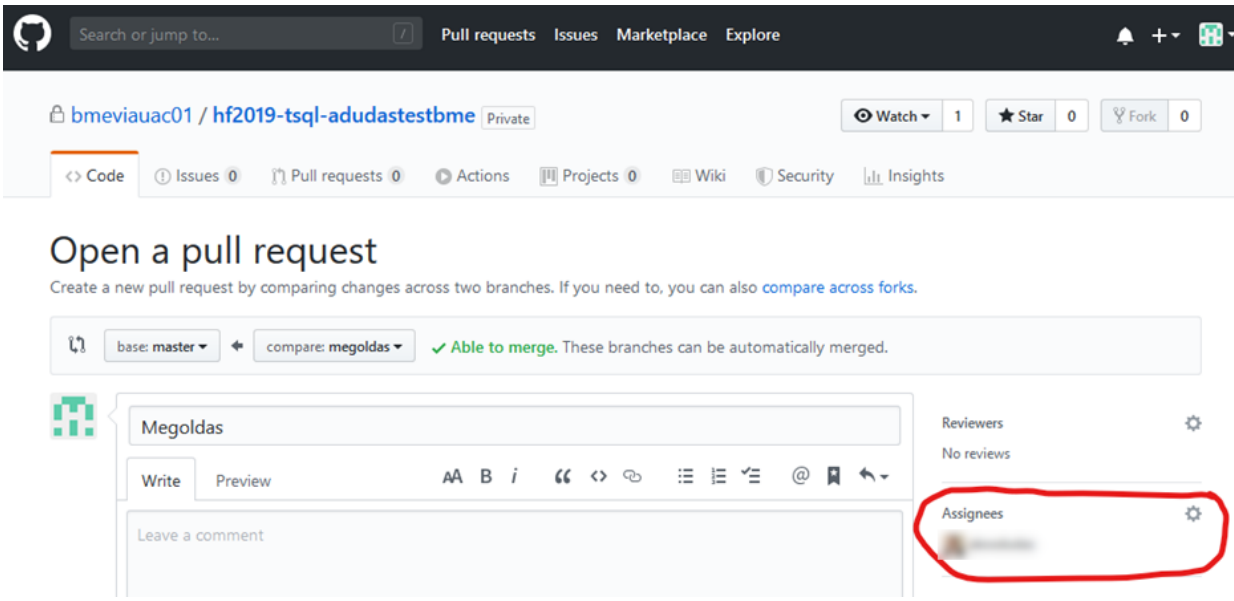


Az értékelés eltarthat egy ideig, addig ez olvasható: Some checks haven't completed yet. Ezután jelenik meg a fenti eredmény (vagy tökéletes megoldás esetén zöld üzenet). Az All checks have failed szöveg némileg félrevezető, az összes teszt futtatása ugyanis egyetlen check-nek számít. Tehát ha a tesztesetek közül akár egy is nem megy át, ez a látvány fogad. A Details linkre kattintva nézheted meg a részleteket, itt látni fogod, melyik tesztesetek nem mentek át, pl.:



Ha itt nem pont azok a tesztesetek futottak hibára, mint nálad lokálisan, az rendellenes viselkedés, ebben az esetben írj az oktatónak.

19. Ha nem vagy megelégedve a munkáddal, akkor még javíthatsz rajta. Ehhez kommitolj és pusholj újra. Ha továbbra is a megfelelő ágon dolgozol, akkor a *pull request* újból le fogja futtatni a teszteket. Arra kérünk, hogy MAXIMUM 5 alkalommal futtasd le a kiértékelést!
    Ha úgy látod, hogy a megoldásodat még javítani akarod, és nem szeretnéd, hogy mindig lefusson az értékelés, akkor vagy állítsd át Draft állapotba a PR-t, vagy zárd le a pull request-et a webes felületen. Ha kész vagy, vissza kell állítanod a Draft állapotból a PR-t, vagy nyiss majd egy újat.
20. VÉGEZETÜL, ha kész vagy, a *pull request*-et rendeld az oktatóhoz. Ez a lépés feltétlenül fontos, ez jelzi a beadást. A beadott verziót javítani már nem lehet. Az automatizált tesztek eredménye jó előrejelzést ad a várható pontszámról. Felhívjuk a figyelmet, hogy az új entitások létrehozására Java esetében nincsenek tesztesetek, ezért azokat különös figyelemmel készítsétek el!



Ha nincs pull request-ed, vagy van, de nincs a laborvezetőhöz rendelve, akkor úgy tekintjük, hogy még nem vagy készen, és nem adtad be a megoldást.

21. Most végeztél. Miután a laborvezetőhöz rendelted a pull request-et, már ne módosíts semmin. A laborvezető értékelni fogja a munkádat, és a pull request lezárásával kommentben jelzi a végeredményt. Erre az utolsó lépésre a határidő **május 2. 23:59**.

A megoldás személyes bemutatása

A megoldást személyesen is be kell mutatni, az május 3-i, erre a célra fenntartott laborgyakorlaton. Az oktató ekkor tetszőleges kérdést feltehet a feladattal kapcsolatban, az önálló megoldás ellenőrzése érdekében. Akár arra is megkérhet, hogy módosíts valamit a kódban. Kivételes esetben, nyomós indokkal távoli, Teams-es bemutatás is egyeztethető a laborvezetőkkel.