Háttéralkalmazások 2. 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, ezt kell néhány REST végponttal kibővíteni. Minden feladatra igazak az alábbiak:

- A kérés/válasz törzse, ha van, JSON formátumú
- A törzsben küldhetsz/fogadhatsz közvetlenül entitásokat JSON-ba szerializálva, vagy ha indokoltnak látod, Data Transfer Objektumokat (DTO-kat) is. Ez utóbbi esetben a property nevek pontosan egyezzenek meg az entitás property nevekkel.

A konkrét feladatok:

1. Írj REST API-t az Address entitáshoz, a /adresses URI-n, amely az alábbi műveleteket támogatja:
    a. POST /addresses: új címet hoz létre az adatbázisban. A kérés törzsében az Address entitás mezőinek megfelelő propertykkel rendelkező JSON objektumot vár. (2 pont)
        i. Válasz siker esetén: 200 OK, a törzsben olyan JSON objektummal, amelyben “id” nevű propertyben tartalmazza az újonnan beszúrt entitás generált id-jét
        ii. Válasz hibák esetén: 400 Bad Request

        - Ha a törzs üres
        - Ha a JSON törzsben ki van töltve az id property
        - Ha a country, city, zipCode, street, number közül bármelyik null vagy üres string
    b. GET /addresses: az összes cím lekérdezése (1 pont)
        i. Válasz: 200 OK, a törzsben egy JSON tömb, benne az összes DB-ben tárolt cím (üres DB esetén üres tömb)
    c. GET /addresses/{id}: adott id-jű cím lekérdezése (1 pont)
        i. Válasz siker esetén: 200 OK, a törzsben az adott id-jű cím adatai
        ii. Válasz hibák esetén: 404 Not Found, ha az adott id-hez nem található cím
    d. DELETE /addresses/{id}: adott id-jű cím törlése (1 pont)
        i. Válasz siker esetén: 200 OK, üres törzs, akkor is, ha az adott id-hez nem található cím, és emiatt nem is történik törlés
    e. PUT /addresses/{id}: az adott id-hez tartozó cím módosítása az adatbázisban. A kérés törzsében az Address entitás mezőinek megfelelő propertykkel rendelkező JSON objektumot vár, az ebben lévő id definiálja hogy melyik entitást módosítjuk. (2 pont)
        i. Válasz siker esetén: 200 OK, a törzsben az entitás új állapota
        ii. 400 Bad Request, ha a törzs üres, vagy ha a JSON törzsben ki van töltve az id property, és az nem egyezik az url-ben lévővel, vagy ha a country, city, zipCode, street, number közül bármelyik null vagy üres string
        iii. 404 Not Found, ha a megadott id-vel nem létezik cím az adatbázisban
2. REST API címek keresésére
    a. POST /addresses/search: keresés címre, ország, város, utca alapján. A törzsben a keresési feltételeket reprezentáló address objektum érkezik. Amely propertyk alapján szűrni szeretnénk, azok vannak kitöltve (csak a fent említett 3 mező szerinti szűrést kell támogatni!). Az egyes propertykre vonatkozó feltételek ÉS kapcsolatban értendők. Kis- és nagybetű nem számít, valamint nem szükséges pontos egyezés, elegendő, ha a megadott stringgel kezdődik az adott mező. (5 pont)
        i. Válasz: 200 OK, a törzsben a megtalált címek tömbje, amely értelemszerűen üres, ha nincs találat.
        ii. 400 Bad Request, ha a törzs üres
    b. Bővítsd ki az előző végpontot, hogy elfogadjon query paraméterben a lapozásra, rendezésre vonatkozó információkat is, az alábbi módon (4 pont):
        i. page: hányadik oldalt kérik, 0-tól számolva

ii. size: az oldal mérete (az utolsó oldal lehet, hogy nem ennyi elemet tartalmaz)
iii. sort: a mező, amely szerint rendezni szeretnénk, illetve opcionálisan vesszővel hozzáfűzve a rendezés iránya (asc/desc). Ha az irány nem specifikált, növekvő rendezés az alapértelmezett.

Egy teljes példa: /addresses/search?page=2&size=10&sort=city,desc
Alapértelmezett értékek:

- Ha a size paraméter hiányzik, az összes találatot adjuk vissza
- Ha a page hiányzik, a 0. oldalt adjuk vissza
- Ha a sort hiányzik, id szerinti rendezés legyen az alapértelmezett
- Ha a rendezés iránya hiányzik, növekvő legyen az alapértelmezett

A válaszban értelemszerűen csak az adott oldalra eső címek tömbje szerepeljen. Ezen kívül a válasz tartalmazzon egy X-Total-Count nevű header-t is, amiben a szerver megmondja, hány találat van összesen.

3. REST API adott időpontnál később befejezendő szállítási tervekre (4 pont)
   a. GET /transportplans
   b. dateTime nevű, kötelező query paraméterben egy dátum+időt fogad el, az ennél később befejezendő transportplan-ek tömbjét kell majd visszaadni, az alábbi módon értelmezve: a kiszállítási terv tervezett befejezési időpontja az utolsó Section-jében lévő toMilestone-jában tárolt plannedTime. Feltételezheted, hogy minden tervre igaz, hogy az utolsó sectiont megelőző összes section toMilestone-jában tárolt plannedTime korábbi az utolsó section toMilestone-jában tárolt időpontnál.
   c. Válasz siker esetén: 200 OK, törzsben a kritériumoknak megfelelő transport planek (esetleg üres) tömbjével.
   d. 400 Bad Request, ha a dateTime paraméter hiányzik, vagy nem követi a yyyy-MM-ddTHH-mm-ss formátumot

A 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 megoldásra is már érdemes lefuttatni a teszteket, hogy lásd, mely tesztesetek mennek már át.

A projekt váz elérése, a megoldás beadása

Ezt a házi feladatot a másik platformon kell megoldanod az első házi feladathoz képest. A választástól függően más-más GitHub Classrooms assignmentet kell elfogadnod:

- Java: https://classroom.github.com/a/0AhwQEEr
- .NET: https://classroom.github.com/a/5c71WxyN

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
- 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).

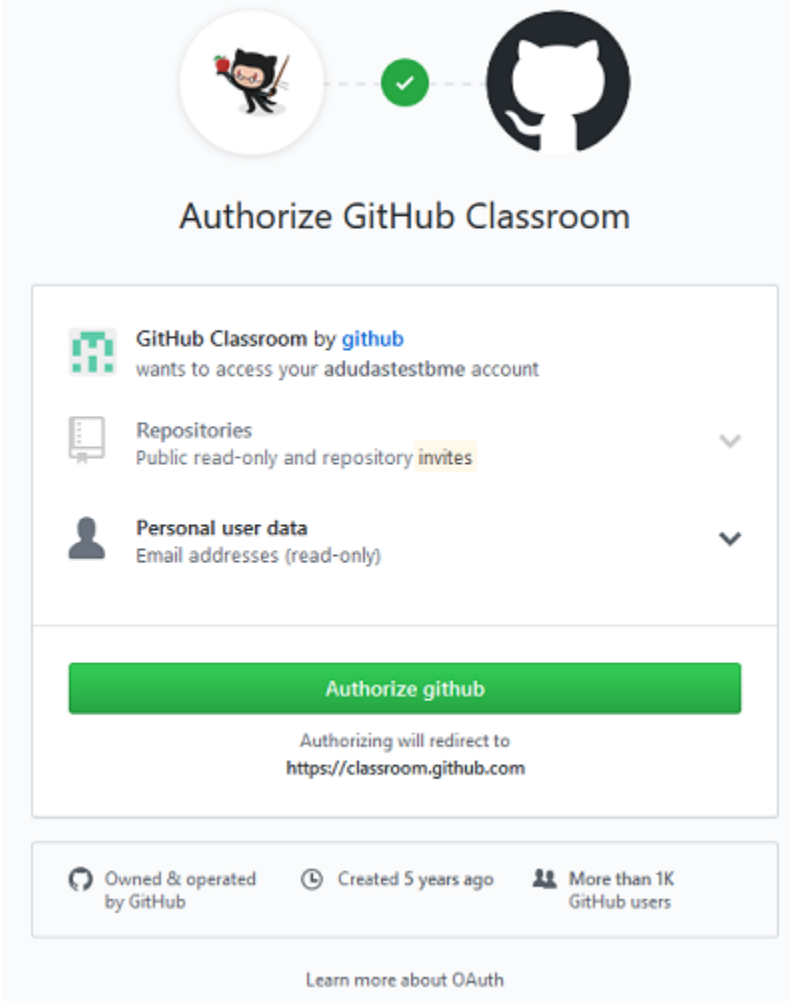
Részletes útmutató a fenti lépések elvégzéséhez

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

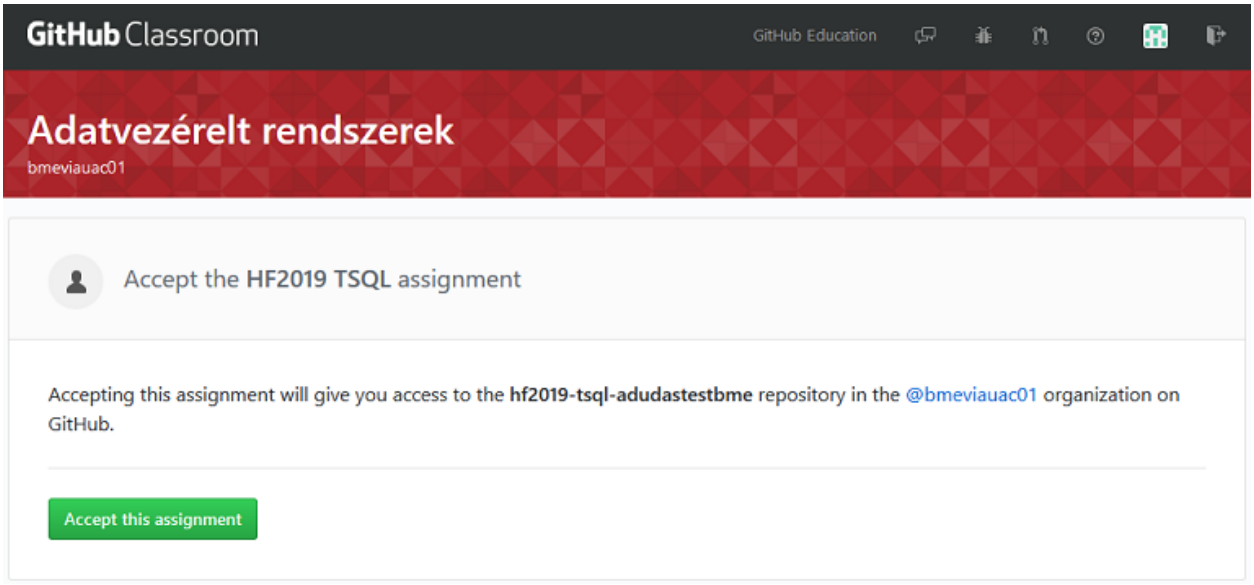
Assingment elfogadása

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

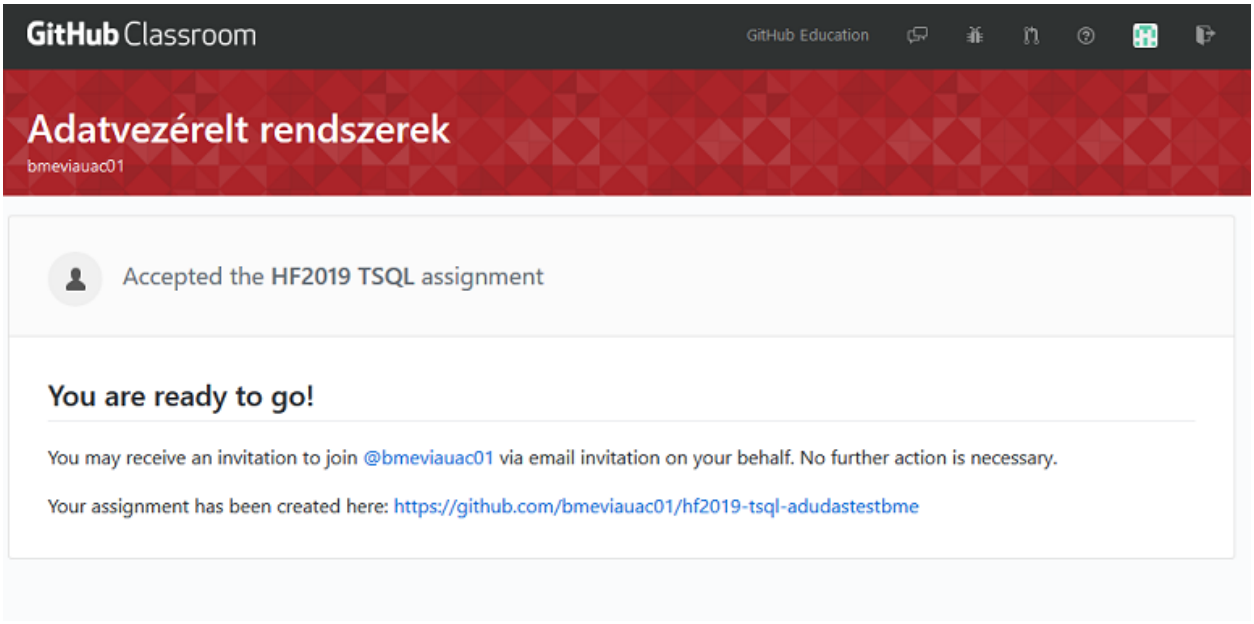
3. Ha kéri, adj engedélyt a *GitHub Classroom* alkalmazásnak, hogy használja az account adataidat.



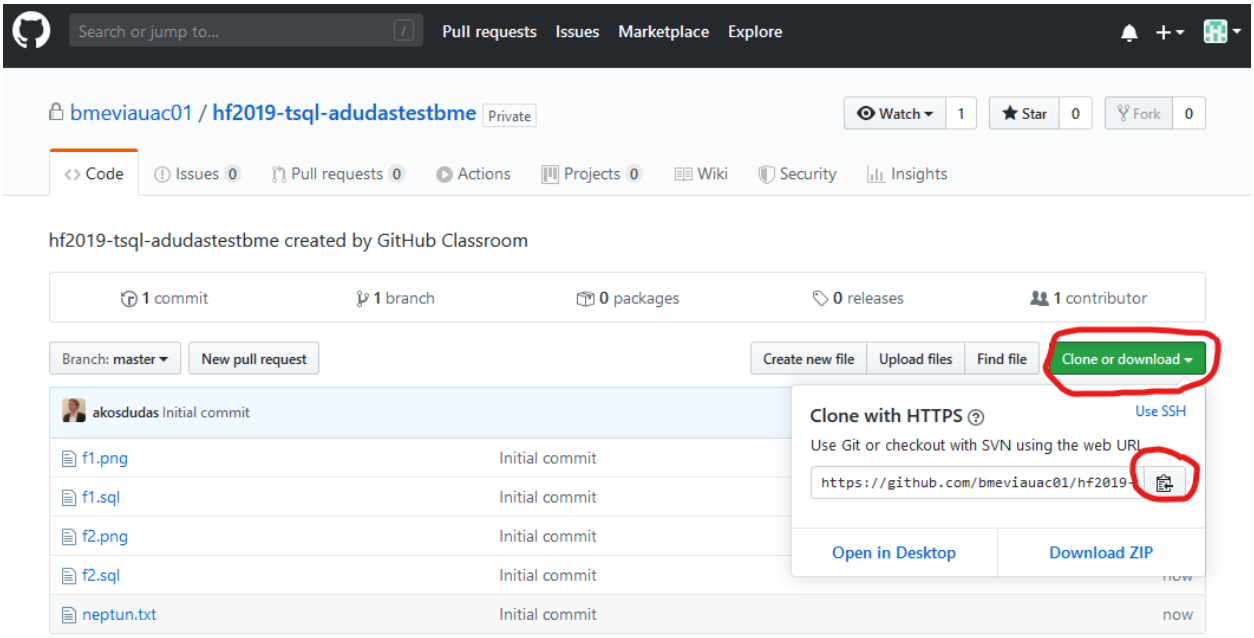
4. Látni fogsz egy oldalt, ahol elfogadhatod a feladatot (“Accept the … assignment”). Kattints a gombra.



5. 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.

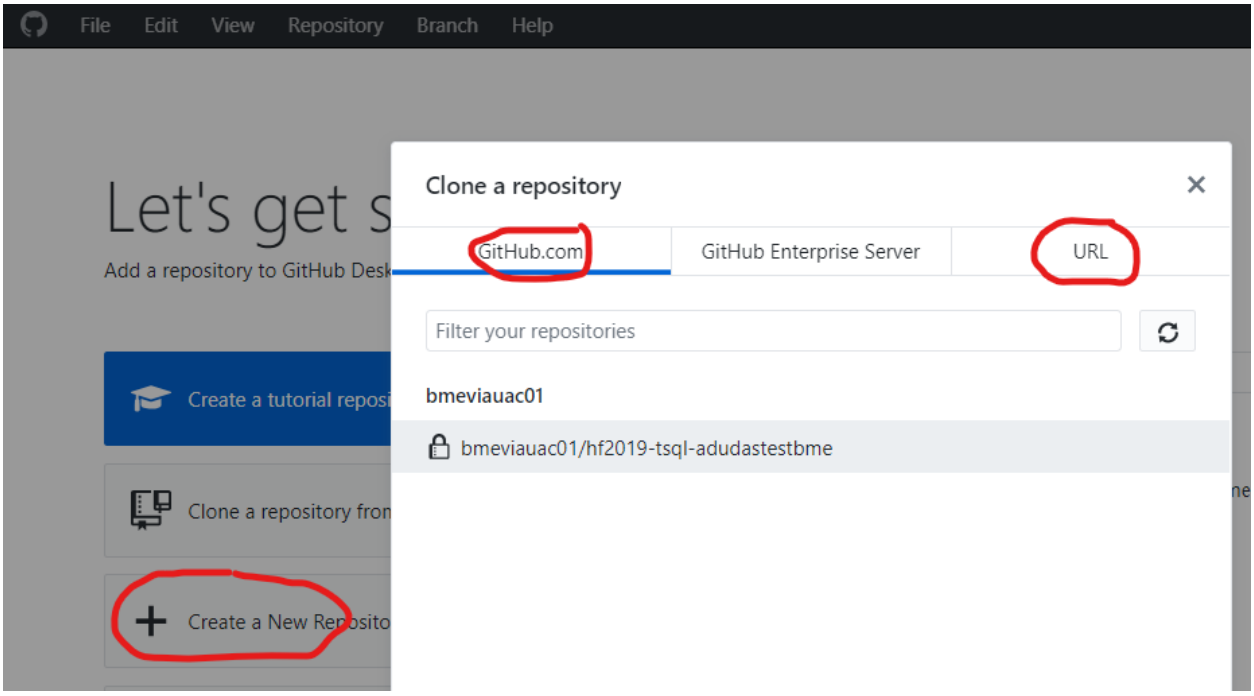


6. Nyisd meg a repository-t a webes felületen a linkre kattintva. Ezt az URL-t írd fel, vagy mentsd el.



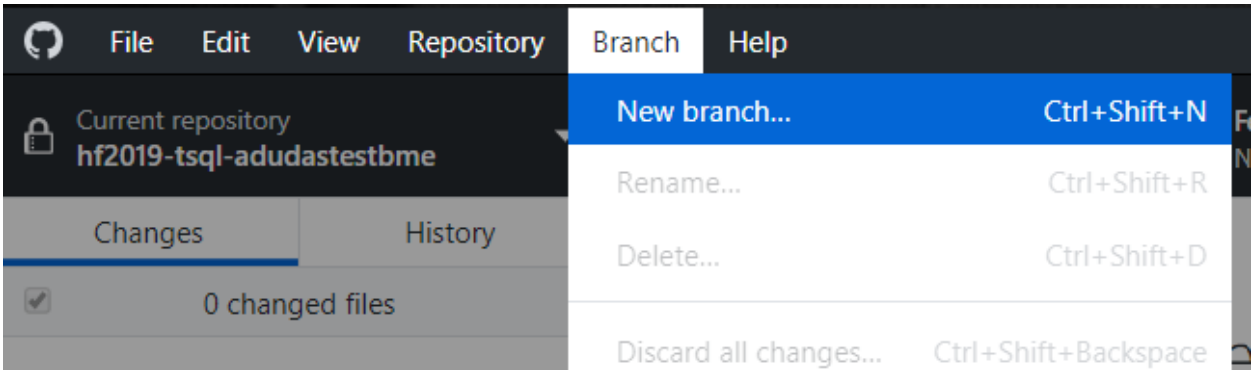
Repository klónozása, megoldas branch létrehozása

1. 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>`

2. 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`

3. Ezen a megoldás ágon dolgozva készítsd el a beadandókat. Akárhányszor kommitolhatsz és pusholhatsz. Laborgépeken mindig ellenőrízd, 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`

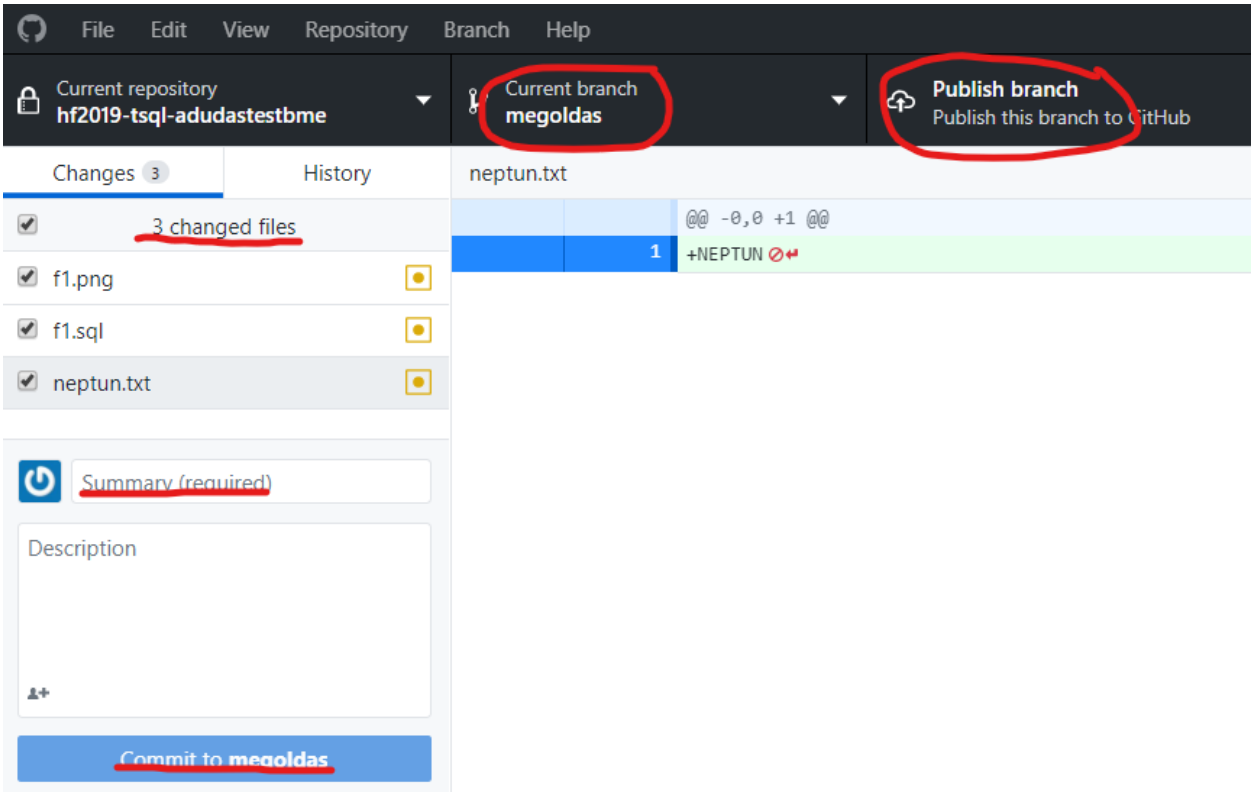
   Ha ez nem megfelelő lenne, akkor add ki az alábbi parancsokat a git repository

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)

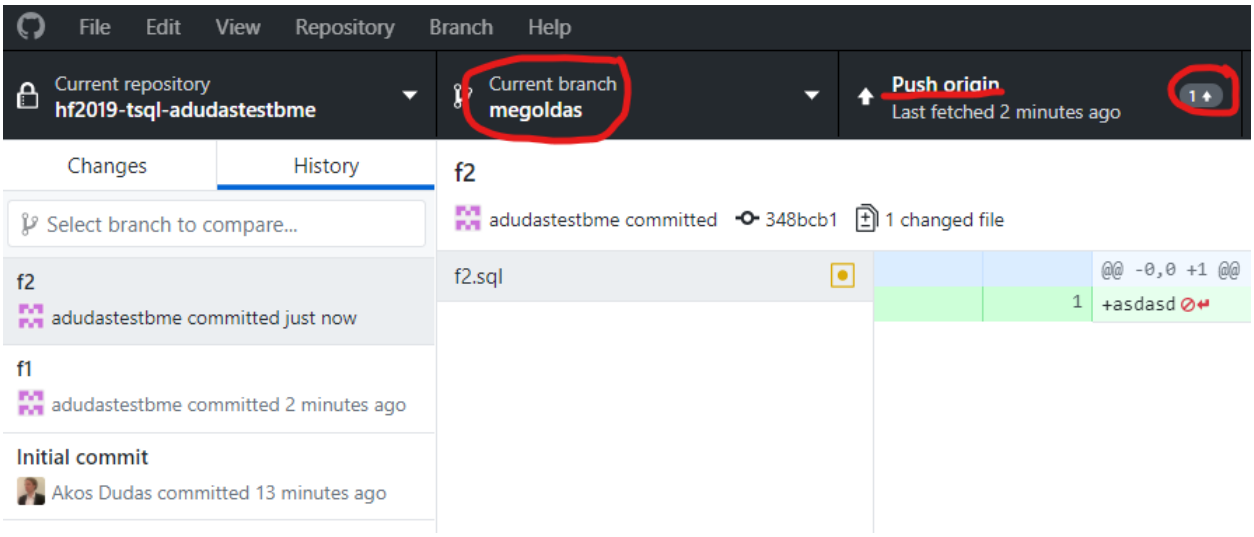
```
git config user.name "John Doe"
git config user.email "john@doe.org"
```

TIP: Otthon 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):

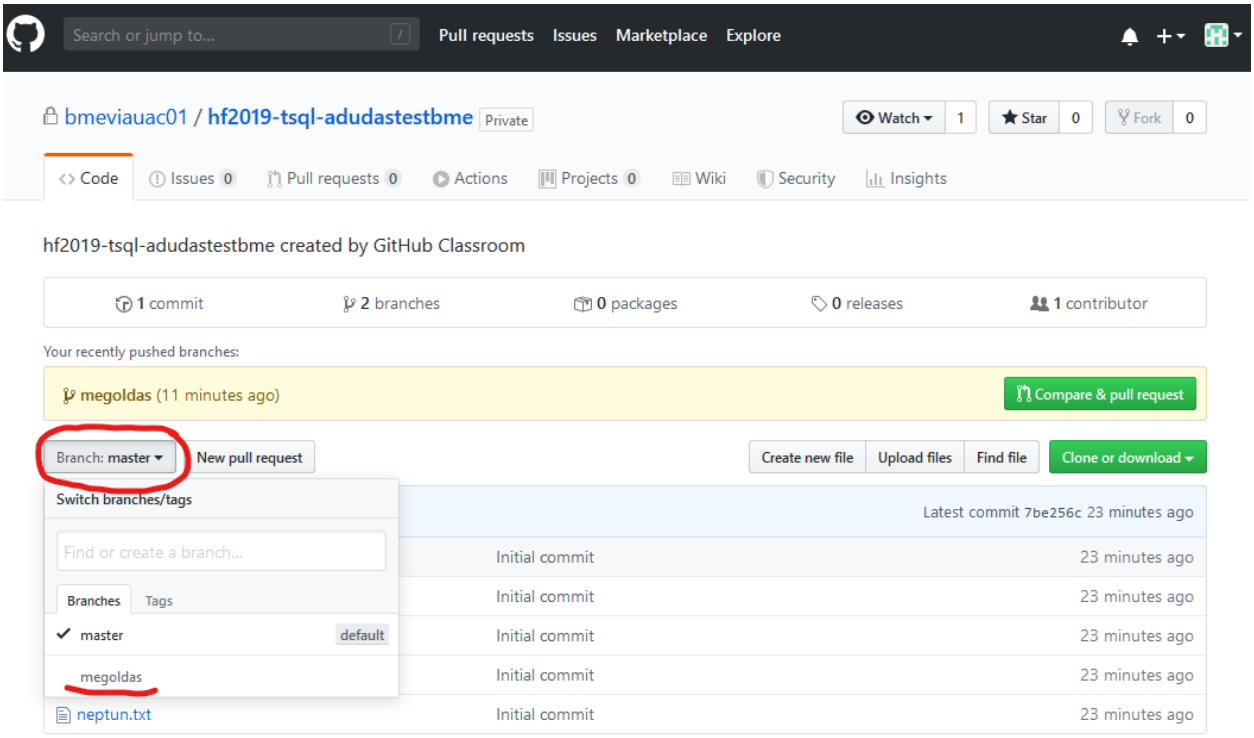
```
# Ellenőrizd az ágat, és hogy milyen fájlok módosultak
git status

# 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.

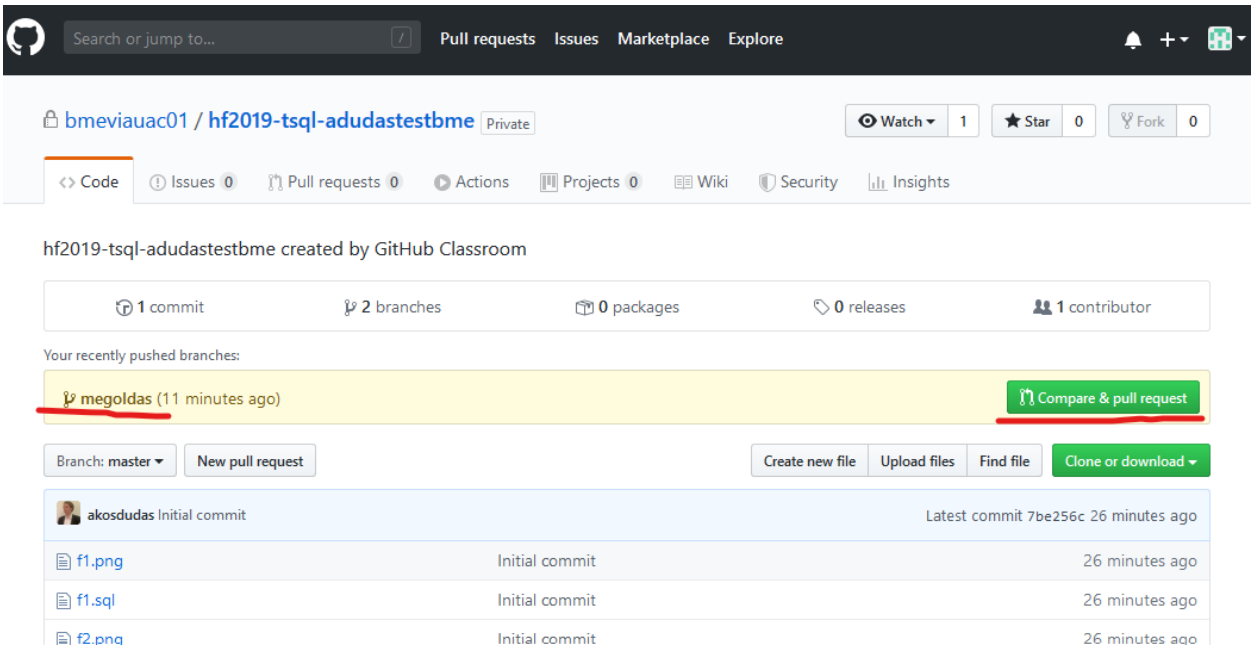
A megoldás beadása

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

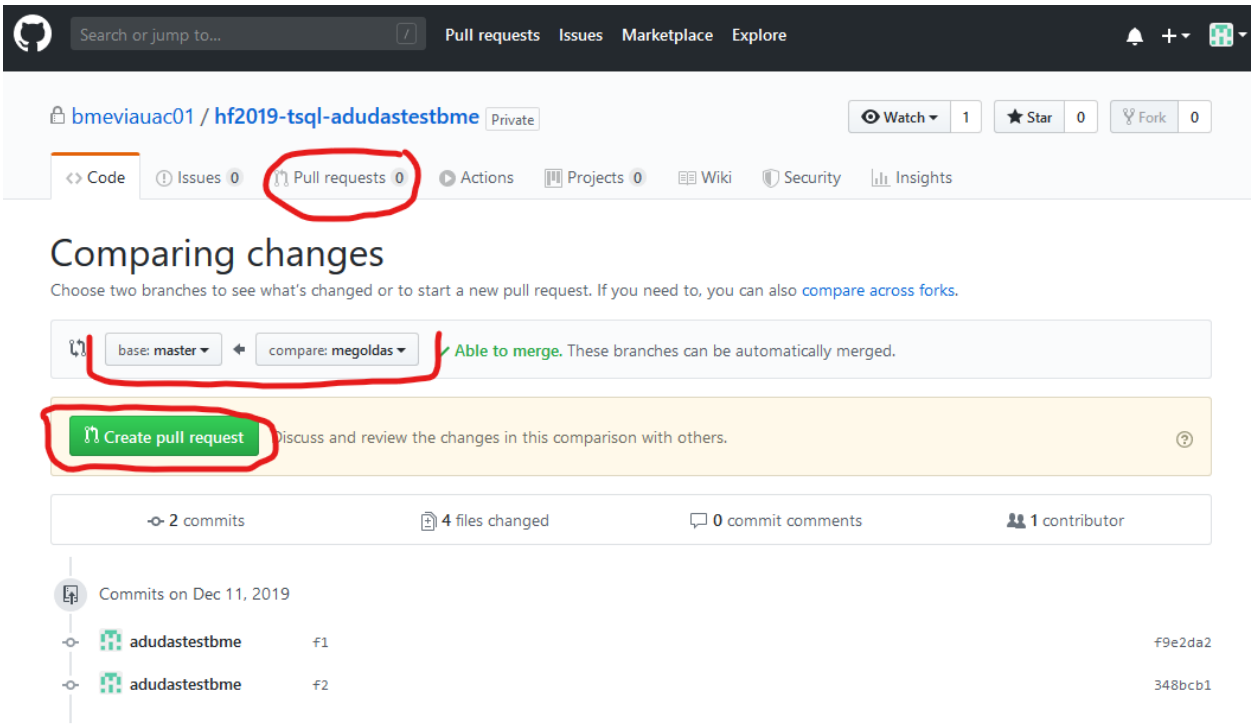


Arra kérünk, hogy NE használd a GitHub fájl feltöltés funkcióját. Ha valami hiányzik, a helyi git repository-ban pótold, és kommitold majd pushold.

2. 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.

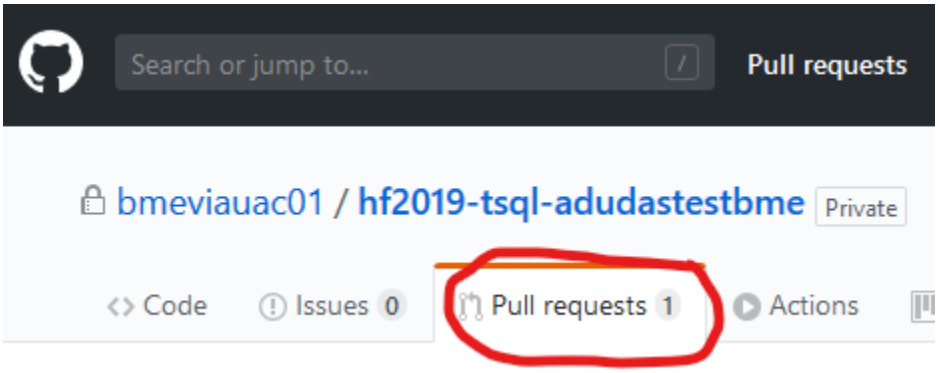


A *pull request*-et a fenti menüben is létrehozhatod. Fontos, hogy a megfelelő brancheket válaszd ki: `master`-be megy a `megoldas` ág.

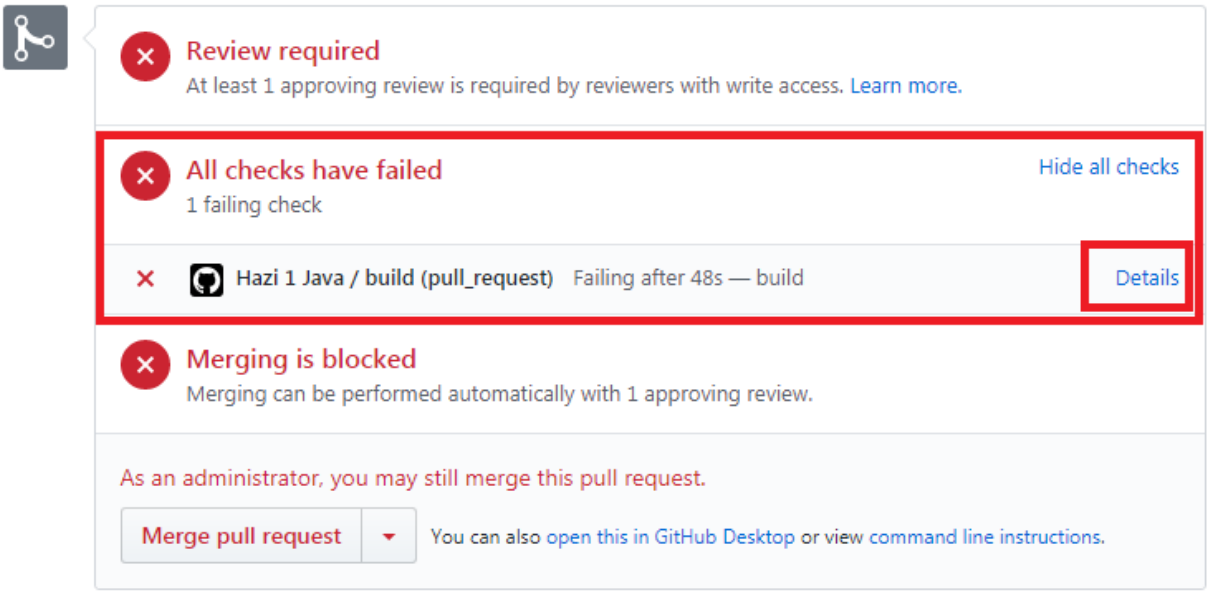


3. Ha minden rendben sikerült, a menüben fent látod a kis “1” számot a *Pull request* elem mellett, jelezve, hogy van egy nyitott pull request. DE MÉG NEM

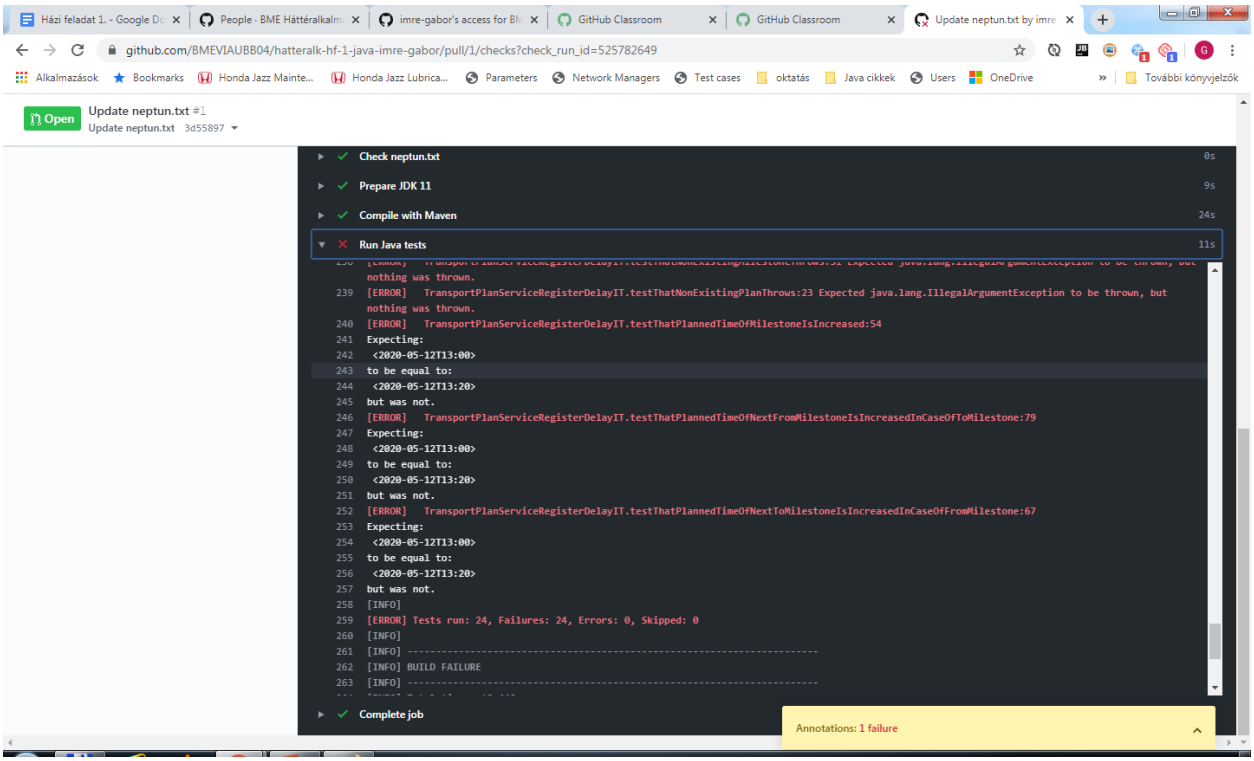
VÉGEZTÉL!



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

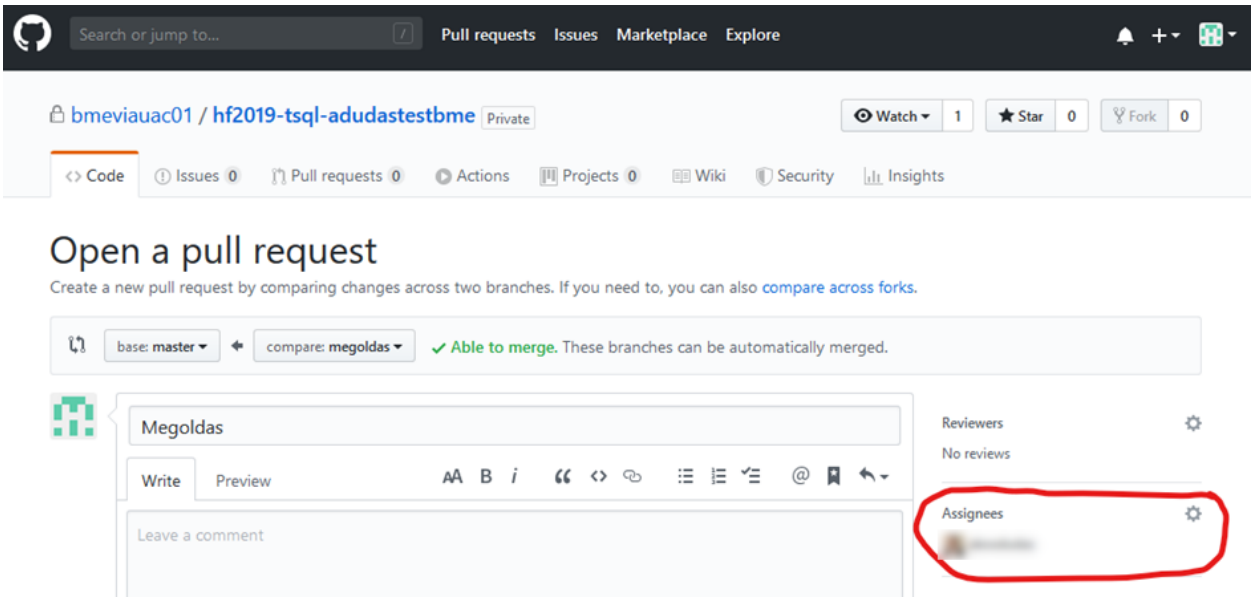


Az értékelés eltarthat egy ideig, addig ez olvasható: Some checks haven’t completed yet. Ezután jelenik meg a fenti eredmény (vagy tökéletes megoldás esetén zöld üzenet). Az All checks have failed szöveg némileg félrevezető, az összes teszt futtatása ugyanis egyetlen check-nek számít. Tehát ha a tesztesetek 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.

5. 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 zárd le a pull request-et a webes felületen. Ha kész vagy, nyiss majd egy újat.
6. 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.



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.

7. 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 **2023. Május 23. 23:59**.

A megoldás személyes bemutatása

A megoldást személyesen is be kell mutatni, a május 24-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.