Tema: Çfarë është GitHub?

Lista e figurave

Pëshëndetje të gjithëve!

Ky është një vazhdim i shkrimit të mëparshëm “Çfarë është Git?”. Më poshtë do të shpjegojmë se çfarë është GitHub dhe si ta përdorim atë.

GitHub është një platformë hostim-i e kodit për *version control*. Kjo na lejon të punojmë së bashku me persona të tjerë në projekte nga kudo që ndodhemi. Git përdoret nga *command line* ndërsa GitHub ka një ndërfaqe grafike në Web. GitHub gjithashtu siguron kontroll për aksesim si dhe disa veti për të thjeshtësuar bashkëpunimin, të tilla si një *wikis* dhe tools për menaxhimin e detyrave për çdo projekt. Për më tepër, çdokush mund të regjistrohet dhe të krijojë një repository publik të kodit *falas*, gjë që e bën GitHub veçanërisht popullor me projektet *open-source*.

1. Krijimi i një repository

Një **repository** përdoret për të organizuar një projekt të vetëm. Mund të përmbajë file, foto, video, kod, folders etj. Gjithashtu, është e këshillueshme që çdo *repository* të ketë një file README ose një file i cila mban informacion në lidhje me projektin për shembull: çfarë bën, pse është krijuar, si funksionon etj.

Për të krijuar një *repository* të ri nga ndërfaqja e GitHub klikojmë në butonin **New.**

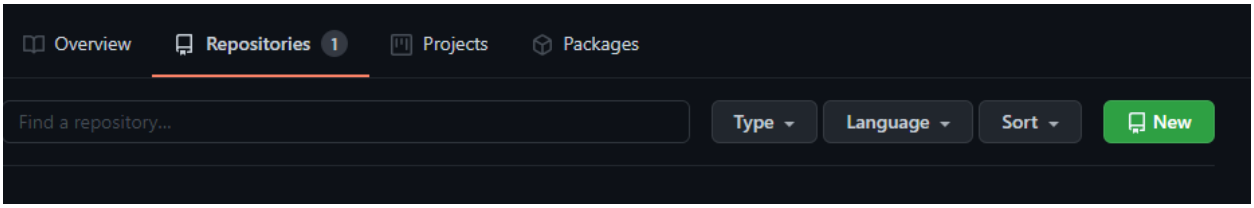

Fig. 1 New repository

Më pas plotësojmë fushat me të dhënat që duam dhe klikojmë **‘Create repository’**.

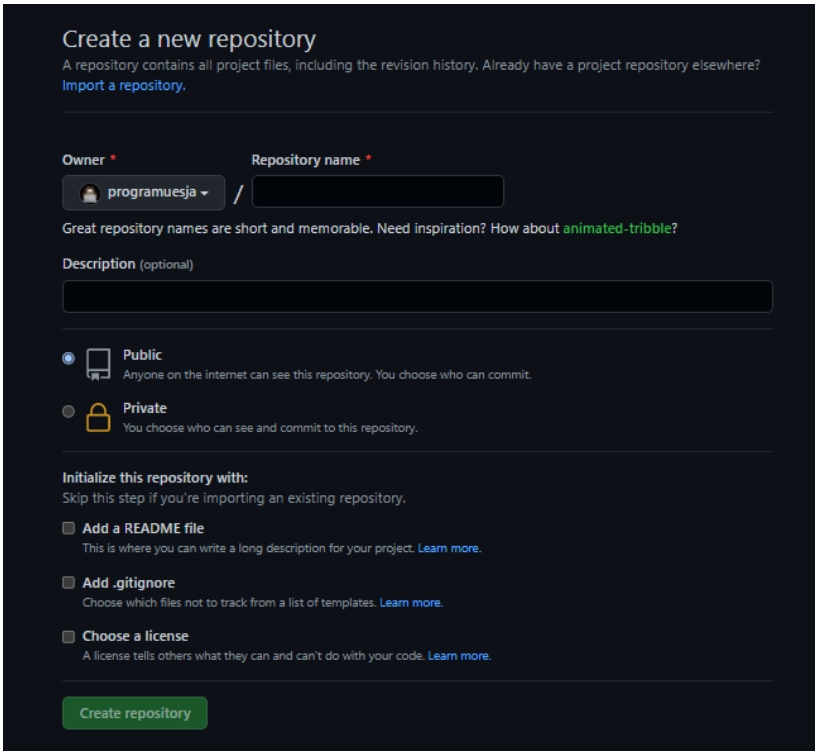

Fig. 2 Repository details

2. Krijimi i një branch

Degëzimi është mënyra për të punuar në versione të ndryshme të një *repository* në të njëjtën kohë. Si default vetë *repository* ka një **branch kryesor** zakonisht të quajtur **main** ose **master**. Përdorim *branches* për të bërë ndryshime dhe për t'i testuar ato para se t'i bëjmë *commit* në branch-in **main**. Kur krijojmë një *branch* të ri nga *main*, krijojmë një kopje të *main* si është në kohën e krijimit. Nëse dikush më pas bën ndryshime dhe i ruan në *main*, mund t'i marrim këto ndryshime duke bërë *pull*.

Për të krijuar një branch të ri veprojmë si më poshtë:
Shkojmë në repository-n ku duam të krijojmë branch-in. Klikojmë menunë drop down në të majtë. Në kutinë e inputit ku lexon *'Find or create a branch…'* shkruajmë emrin e branch-it që duam të krijojmë (në këtë shembull *test-branch*) dhe më pas klikojmë **'Create branch: ….'** ose shtypim **Enter**.

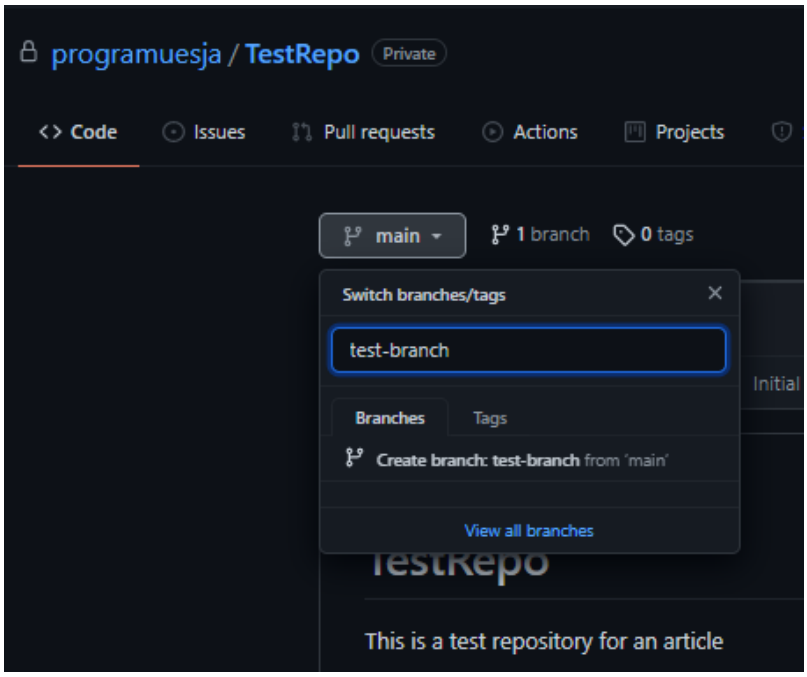

Fig. 3 Create branch

3. Bërja e ndryshimeve dhe *commit*

Pasi krijojmë branch-in ***test-branch*** kalojmë në këtë branch. Mund të bëjmë një ndryshim në këtë branch si për shembull: shtimi i një file të ri. Për të kryer këtë ndjekim hapat e mëposhtëm.

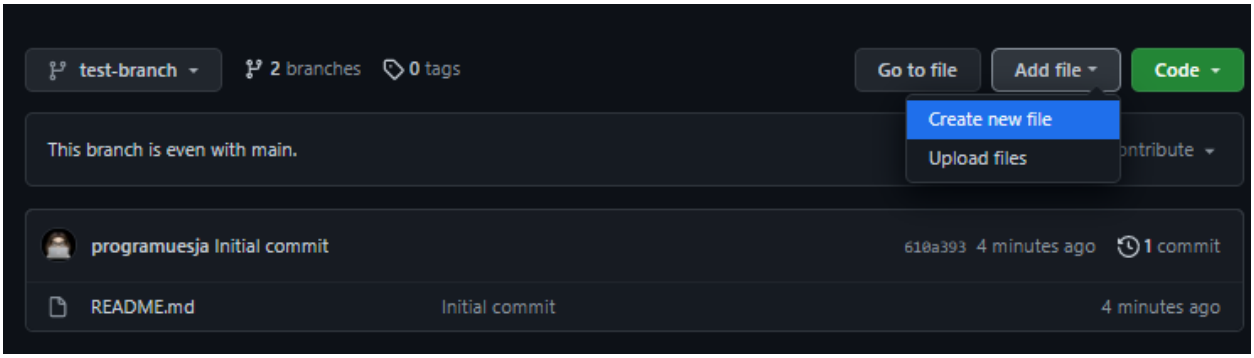

Fig. 4 Add file

- Klikojmë **Add file** në të djathtë
- Zgjedhim nëse duam të krijojmë apo shtojmë një file nga kompjuteri
- Nëse zgjedhim **Create new file** hapet faqja e mëposhtme ku vendosim emrin e file-t si dhe tekstin që do të përmbajë ky file.

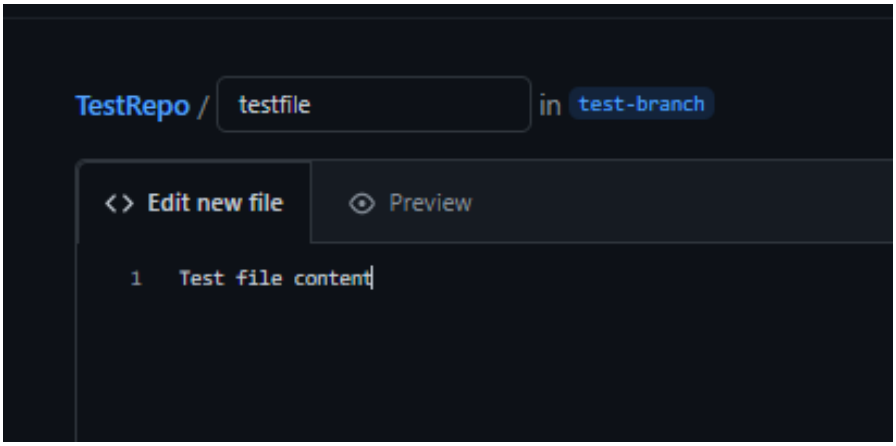

Fig. 5 Create new file

- Në fund shohim dy fusha input për të bërë commit këtë file të ri. Si e shpjeguam dhe në artikullin "Çfarë është Git?" kur bëjmë një commit mund të vendosim një mesazh përshkrues për ndryshimet e zhvilluara. Pasi plotësojmë fushat klikojmë **Commit new file** për të bërë *commit*.

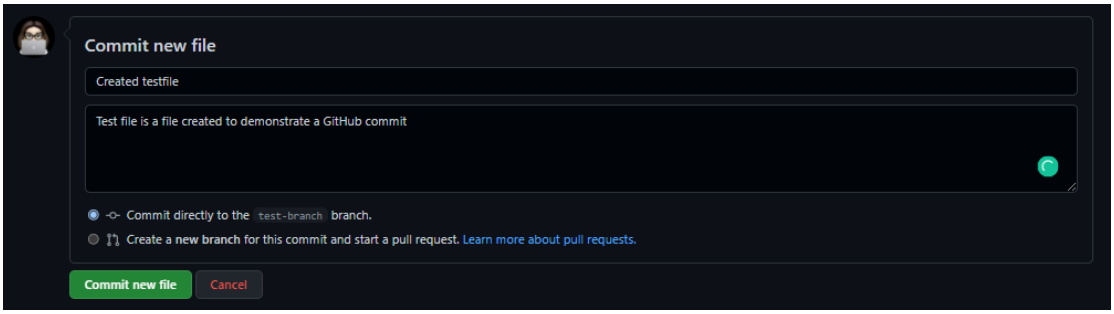

Fig. 6 Commit changes

Pasi bëjmë *commit* dhe shkojmë përsëri te branch-i na shfaqet si më poshtë:

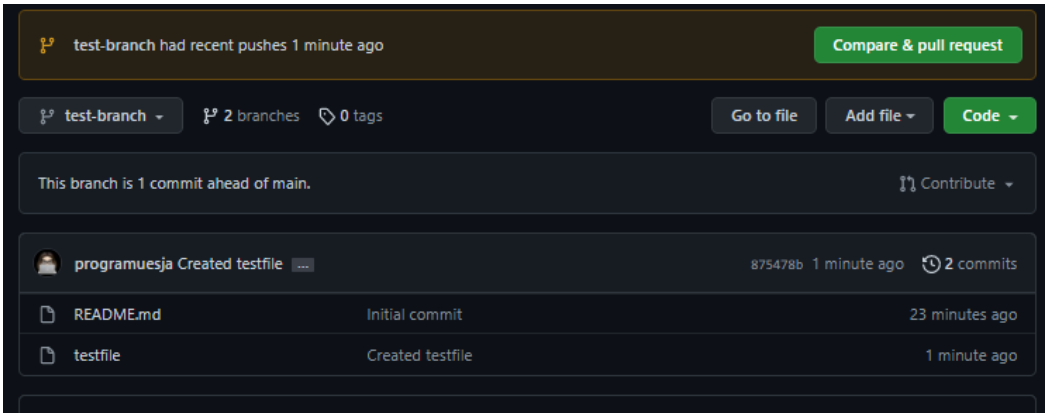

Fig. 7 Committed file

4. Hapja e një Pull Request

Pull requests janë thelbi i bashkëpunimit në GitHub. Kur hapim një *pull request*, propozojmë ndryshimet tona dhe kërkojmë që dikush të rishikojë dhe të bëjë ***pull*** ndryshimet tona dhe t'i bashkojë me *branch*-in e vet. *Pull requests* tregojnë ndryshimet mes dy *branch*-ve. Ndryshimet tregohen me ngjyrë të gjelbër dhe të kuqe, në varësi të ndryshimeve që kemi bërë, respektivisht kemi shtuar diçka të re ose hequr diçka ekzistuese. Për të hapur një *pull request* të re shkojmë në **Pull requests > New pull request**, si në foton në vazhdim.

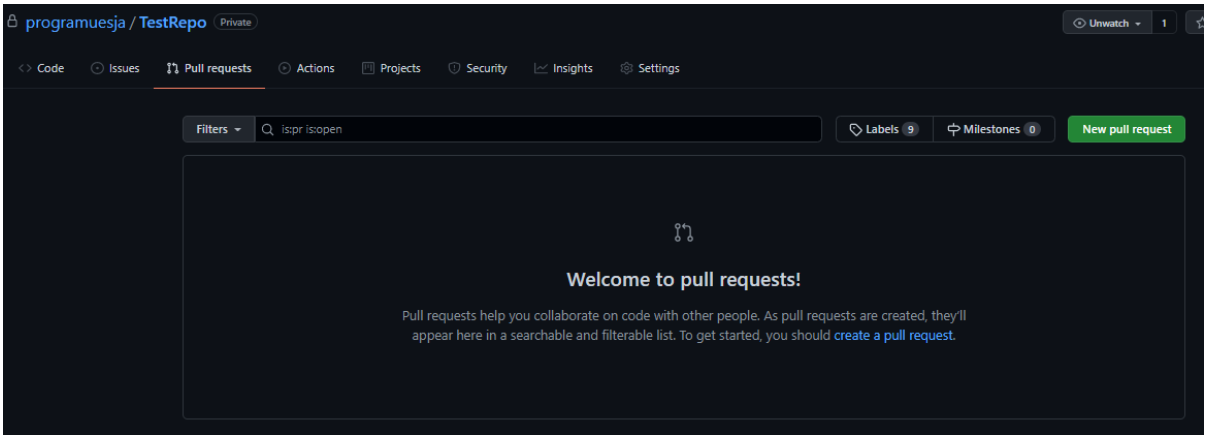

Fig. 8 Creating new pull request

Pasi klikojmë **New pull request**, hapet faqja e mëposhtme ku mund të zgjedhim të shohim ndryshimet mes *branches* të ndryshëm. Në shembullin tonë shohim ndryshimet mes branch ***main*** dhe branch ***test-branch***. Si mund të vëzhgojmë dhe në Fig. 9, në branch-in *test-*

branch tregohet se është bërë 1 commit dhe ka ndryshuar 1 file (testfile që shtuam më parë).

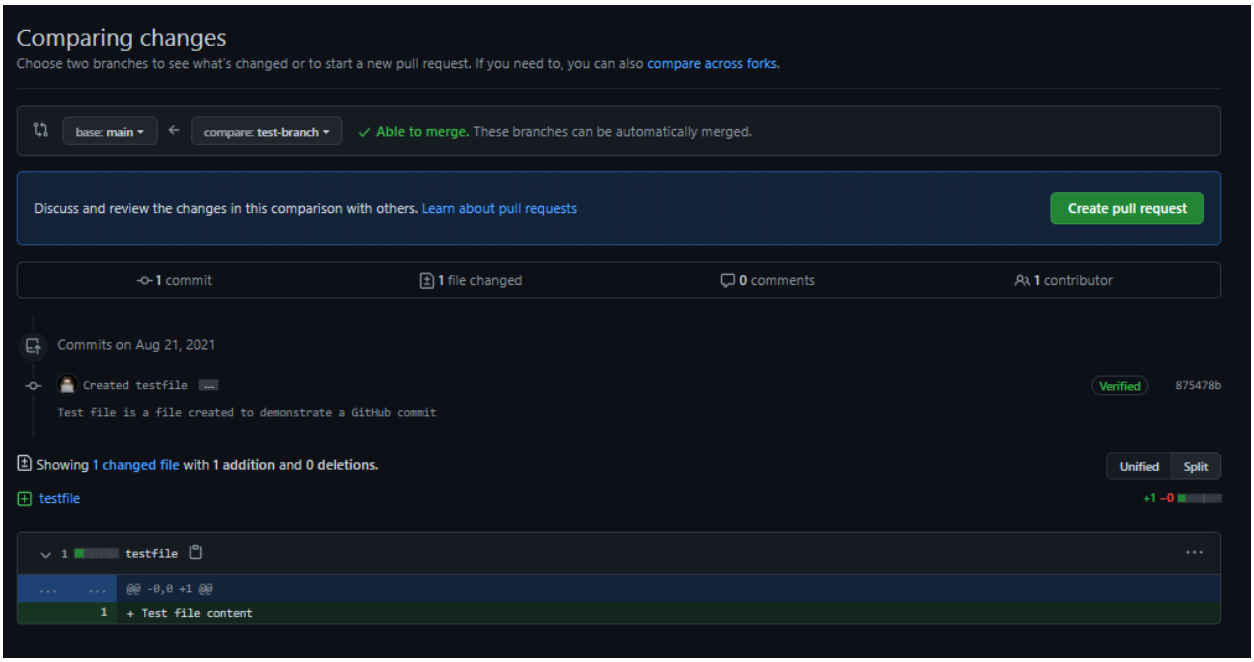

Fig. 9 Comparing branches

Në fund klikojmë **Create pull request** dhe shfaqen detajet si më poshtë. Në të djathtë mund të caktojmë se cili do të kryejë rishikimin, cili do të pranojë ndryshimet etj. Pasi i kemi plotësuar gjithë fushat që duam klikojmë **Create pull request.**

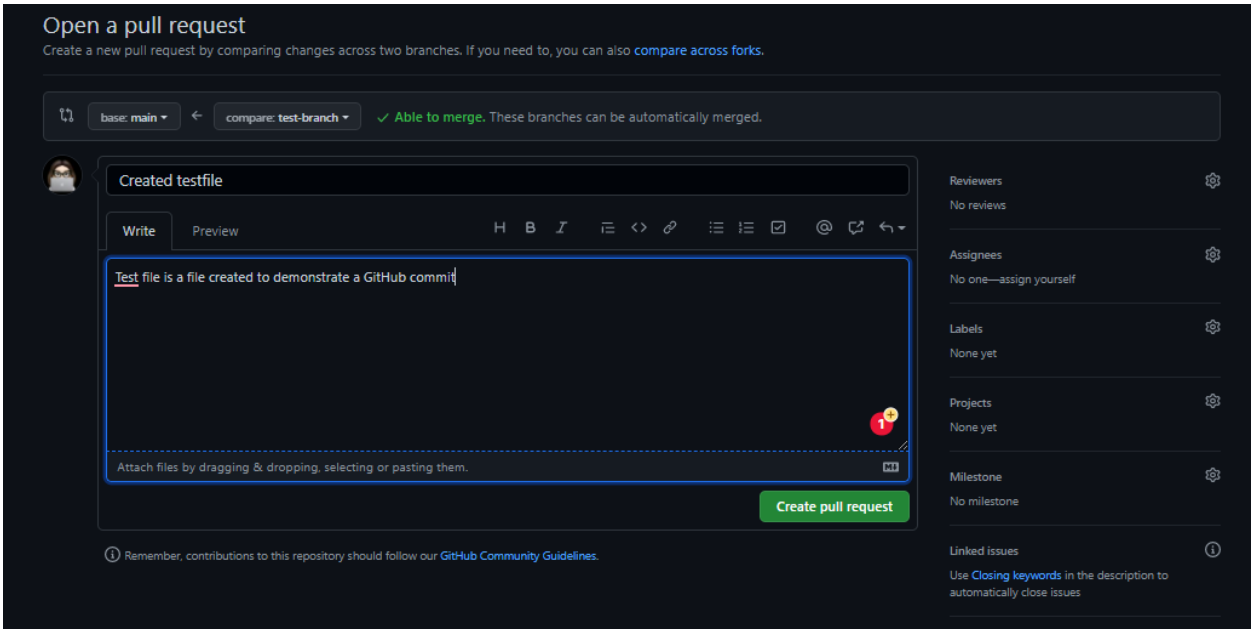

Fig. 10 Fill in pull request details

Pasi krijojmë *pull request* mund të bëjmë *merge* ndryshimet nga branch **test-branch** në **main**. Gjithashtu, na tregon nëse ka konflikte të cilat duhen zgjidhur mes branch-it të cilin duam të bëjmë merge dhe branch-it ku duam t'i bëjmë merge.

5. Merge

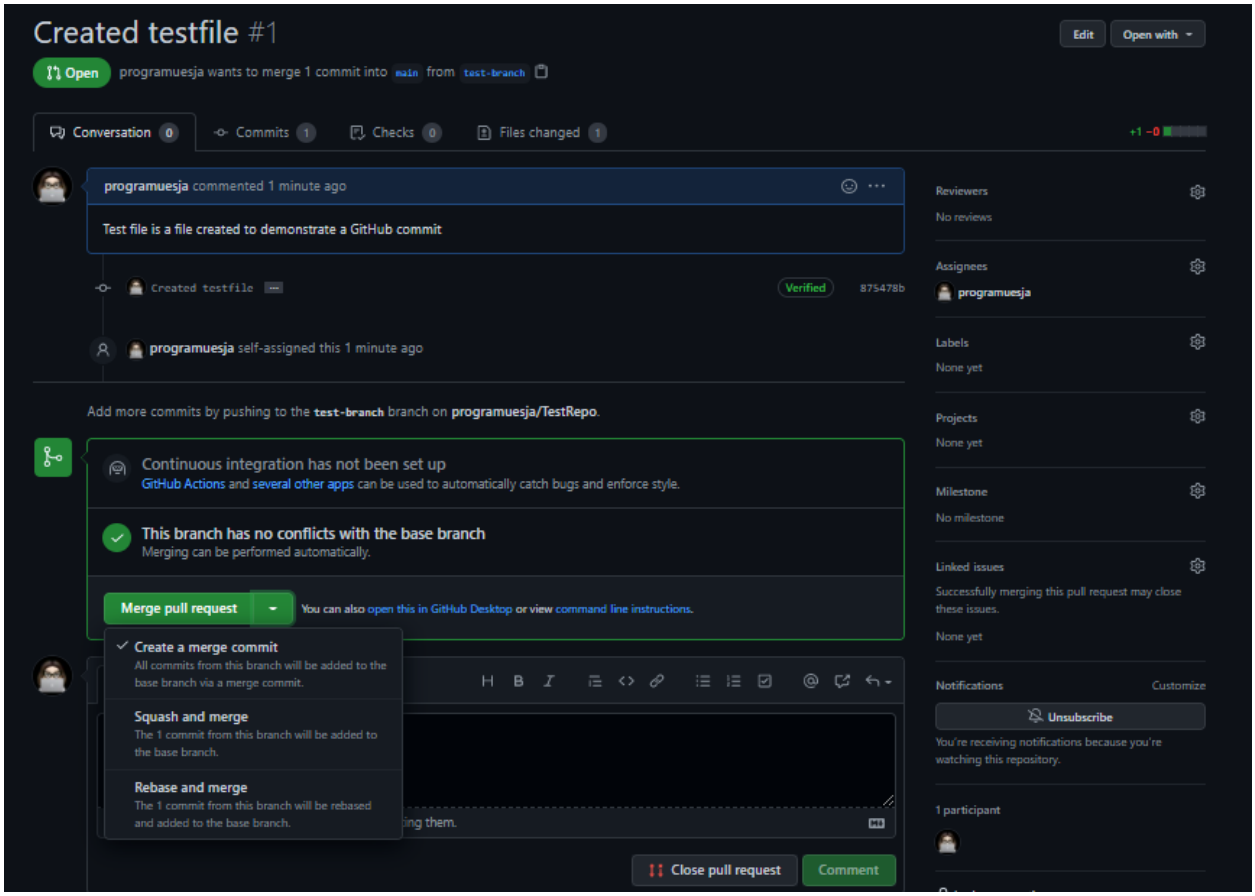

Fig. 11 Created pull request

Për të bërë **merge** kemi tre opsione të cilat mund t'i shohim në Fig. 11.

- **Create a merge commit** (default): Github do të marrë të gjitha *commits* nga *pull request* dhe do t'i shtojë ato në branch-in kryesor me një commit të ri në merge commit. Marrim një shembull. Në branch-in test-branch shtojmë dy file të ndryshme dhe për secilën pasi i shtojmë bëjmë *commit*. Kështu do të të kemi dy *commits* të ndryshëm (Created file.txt & Add git photo).

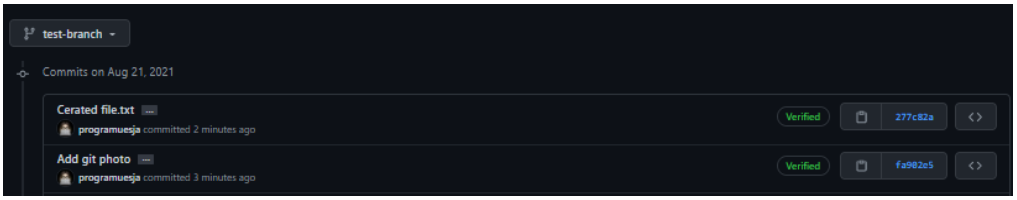

Fig. 12 Add two files to test-branch

Pasi bëjmë merge me branch-in **main** shohim se është shtuar një commit i ri **"Merge pull request #2 ….."**, e cila shton commits të mëparshëm *"Created file.txt"* dhe *"Add git photo"* në main.

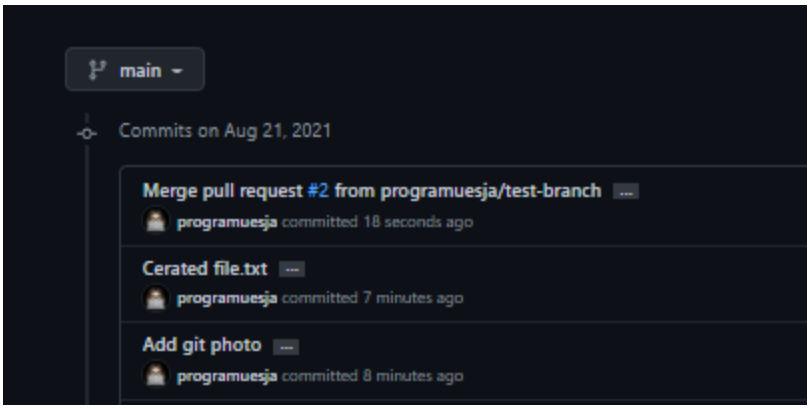

Fig. 13 Create a Merge Commit

- **Squash and merge**: *squashing* merr disa commits dhe i "ngjesh" në një të vetëm. E shohim me shembullin në vazhdim për të kuptuar ndryshimin nga *Create a merge commit*. Bëjmë përsëri dy commits në test-branch për disa ndryshime. Më pas krijojmë pull request dhe pas kësaj zgjedhim opsionin *Squash and merge*. Si shihet në Fig. 14 tani shfaqet krijojmë vetëm një *commit* **Test branch (#3)** ku i **"ngjeshim"** të dy *commits* bashkë, ndryshe nga metoda e parë ku krijuam një *commit* **"Merge pull request #2…."** ku **shtuam** dy *commit*s.

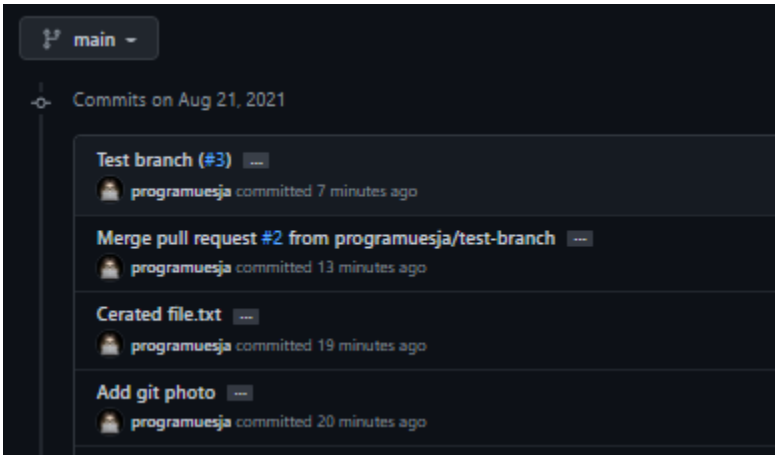

Fig. 14 Squash and merge

Nëse klikojmë mbi këtë *commit* shohim detajet si më poshtë. Në këtë *commit* tregohet se janë modifikuar dy files dhe *commit*-et e këtyre ndryshimeve janë "ngjeshur" bashkë në një *commit* të vetëm të quajtur **Test branch (#3)**.

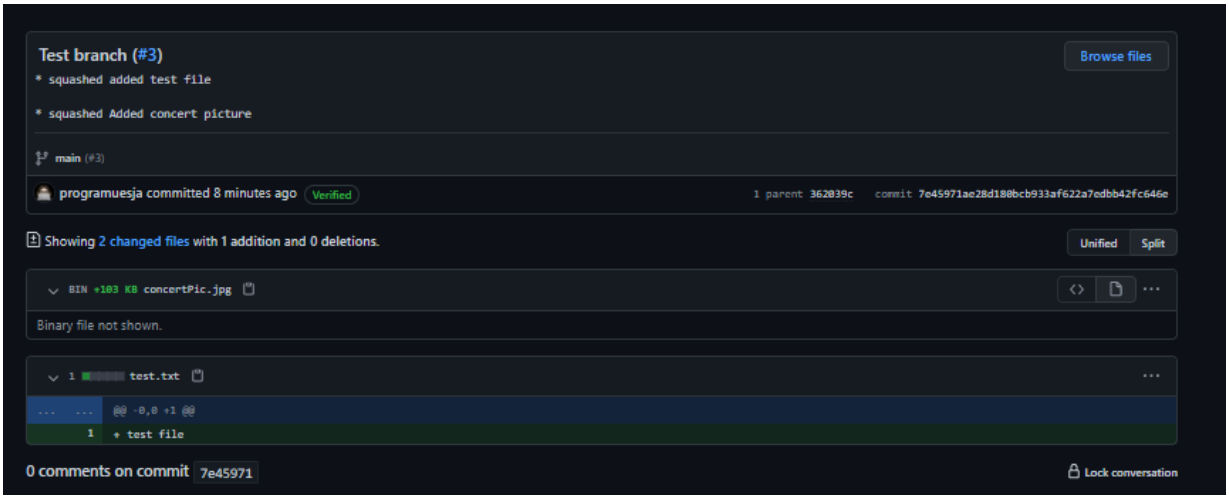

Fig. 15 Test branch commit (squashed commits)

- **Rebase and merge**: nëse ndryshimet janë të vogla ose janë *commits* që bëhen vetëm një herë, atëherë mund të përdorim **Rebase and merge**. Kjo e mban historinë e Git të pastër. Si shihet dhe në Fig. 16 kemi dy *commits* të reja të shtuara, por ndryshe nga **Create a merge commit** nuk kemi një *commit "Merge pull request…"* për të shtuar dy *commits* nga test-branch.

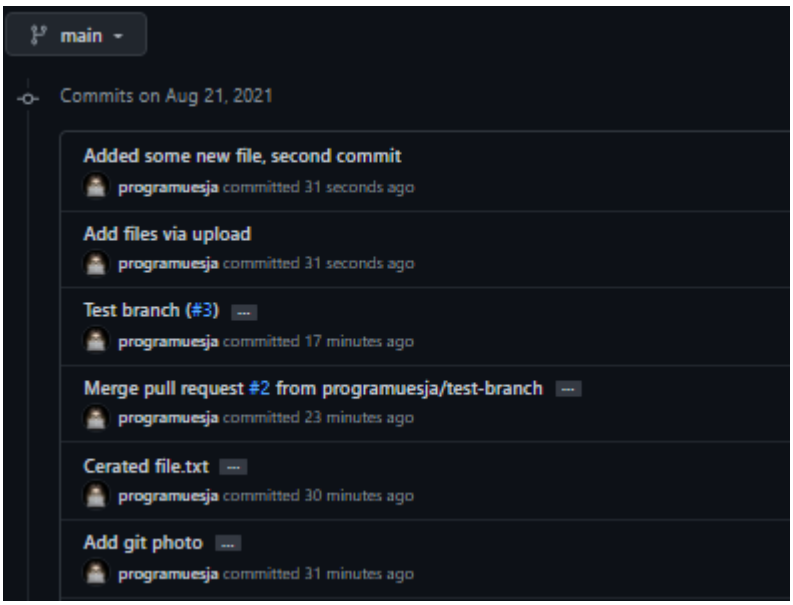

Fig. 16 Rebase and merge

6. Issues

Issues janë një mënyrë për të gjurmuar *tasks*, përmirësimet dhe problemet në projektet tuaja. Mund të krijoni issue për të vënë në dijeni pjesëtarët e tjerë në projekt për një *bug* në kod, mund të caktosh *tasks* për zhvillues të tjerë ose veten etj. Në Fig. 17 kemi krijuar një *issue* për një *feature* të re në projekt. Në menunë djathtas mund të përzgjedhim personin të cilit do ia caktojmë këtë detyrë, mund të vendosim një *label* e cila tregon se për çfarë është krijuar ky issue për shembull: përmirësim, bug, për të kërkuar ndihmë, për të bërë pyetje etj.

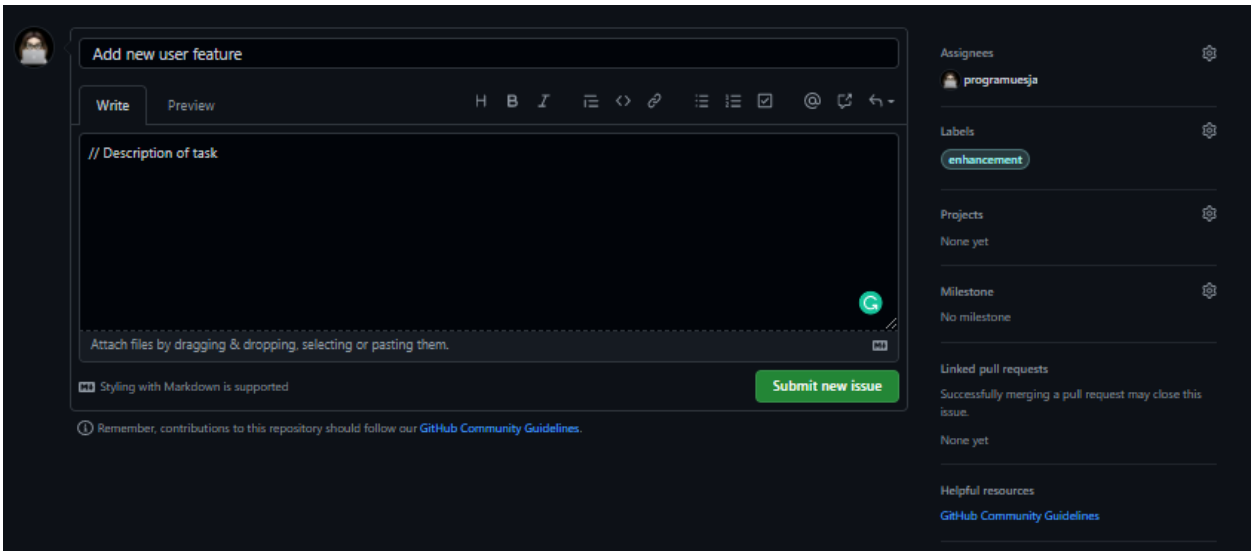

Fig. 17 Create issue

Përfundim

Në këtë artikull mësuam çfarë është GitHub dhe nga ndryshon prej Git. Gjithashtu, eksploruam dhe disa prej *features* më të përdorura të tij.

Lexime të mëtejshme:

https://guides.github.com/introduction/flow/

https://education.github.com/pack