GIT- GITHUB

Git se koristi kao vid komunikacije (u softverskim firmama) među učesnicima projekta (backend, frontend programera, testera, dizajnera i dr.), tako da je teško učestvovati u ovom poslu ako se ne zna šta je Git, a šta GitHub.

Potreban softver za ovu oblast:

|  |
| --- |
| 1. Git software  2. Nalog na GitHub |

U procesu razvoja softvera pišemo, testiramo, čuvamo i skladištimo kodove, tako da su tu nezaobilazni **sistemi za kontrolu verzija** (VSC- version control systems), čija je uloga da prate verzije projekta kada više učesnika radi na njemu.

**Šta su sitemi za kontrolu verzije?**

Sistemi za kontrolu verzija se dele na:

1. sisteme sa centralizovanom verzionom kontrolom (CVCS)- kod njih se svi podaci verzija nalaze na centralnom serveru, a učesnici projekta imaju dostupnu samo verziju na kojoj rade. Ovi sistemi pružaju lakše održavanje i administriranje, ali je sigurnost podataka problematičan, jer ukoliko dođe do otkaza sistema gube se sve informacije.

Ovde pripadaju alati kao što su: SVN, Perforce i CVS.

1. sistemi sa decentralizovanom verzionom kontrolom (DVCS)- kod ovih sistema učesnici projekta pored poslednje verzije preuzimaju i kompletnu bazu o svim verzijama na tom repozitorijumu, i u slučaju otkaza sistema dovoljno je da samo jedan od učesnika postavi podatke na server.

U ove sisteme se ubrajaju sledeći alati: Mercurial, Bazzar, Darcs i **Git.**

**Šta je repozitorijum?**

Repozitorijum (lat. repositorium- polica sa pregradama za spise ili knjige) predstavlja neku vrstu strukture podataka u vidu skladišnog prostora, smeštenog na serveru, u koji se nalaze svi metapodaci i baza podataka vezana za naš projekat. Repozitorijum sadrži:

* skup datoteka i direktorijuma;
* podatke o istoriji promena na njemu (repozitorijumu);
* skup komitovanih objekata;
* skup reference na komitovane objekte.

**Šta je komitovanje?**

Komitovanje (eng. commit- urezati) predstavlja snimak promena stanja na fajlu u odnosu na predhodno stanje. Važno je napomenuti da se u git-u nikad ne commit-uje u udaljeni repozutorijum, nego u lokalni (na našem računaru).

**Istorija razvoja Git-a.**

Git je razvio Linus Torvalds 2005. godine za potrebe razvoja Linuks jezgra, uz doprinos i drugih programera. Naziv “git” je dao Linus Torvalds kada je napisao prvu verziju opisavši ga kao “glupi prikupljač sadržaja”, a za naziv je koristio nasumičnu kombinaciju od tri slova koja mogu lako da se izgovore, a da nisu iskorišćena za neku Juniks komandu (osnovne komande UNIX operativnog sistema), koristivši rečnik slenga (žargonski rečnik):

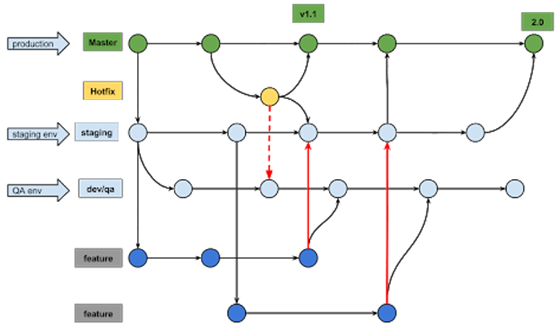
*“Goddamn Idiotic Truckload od sheet”- GIT (kada se pokvari).*

**Šta je Git?**

Git je najpopularniji distribuirani sistem za kontrolu verzija, koji vrši praćenje promena u računarskim fajlovima i koordinaciju rada više učesnika na projektu. Primarno se koristi za verzionisanje koda pri razvoju softvera, ali se može koristiti za praćenje promena u bilo kojim fajlovima.

Kao kod većine drugih distribuiranih sistema kontrole verzija, svaki Git radni direktorijum na svakom računaru predstavlja kompletan repozitorijum sa svom istorijom i upotpunjenom mogućnošću za **praćenje verzija**, nezavisno od pristupa mreži ili centralnog servera.

**Kako se vrši praćenje verzija u GIT-u?**



Git obezbeđuje mahanizam praćenja istorije svake pojedinačne datoteke, počev od njenog kreiranja. Kreirana datoteka se nalazi na glavnoj grani, koja se naziva master ili u novijoj verziji Gita main grana (ali i ne mora tako da se zove) i od nje sve kreće. Ako npr. pretpostavimo da ta datoteka sadrži neki kod u kome ćemo izvršiti promenu sadržaja koda, da bi sačuvali te izmene u master grani Gita moramo ga komitovati. Nakon komitovanja fajla pored sačuvanog izmenjenog sadržaja, imaćemo sačuvane i podatke o osobi koja ga je izmenila i datumu izmene. U master/main grani možemo imati više komitovanih fajlova sa precizno sačuvanim podacima.

|  |  |
| --- | --- |
| Git nas spasava od lavirinta i mreža direktorijuma, kao i mnoštva kopija projekta u kojima je razvijen neki deo koda koji nije u drugom, pa mi jurimo za finalnim a nemamo pojma gde je (kao na slici).  Git nas takođe spasava i od mnogih pitanja kao što su: Ko je odgovoran za ovaj kod? Kome da pošaljem ispravku? Koje rešenje uzeti? Kada postaviti izmenu? Ko je izmenio kod? Kada je izvršeno testiranje? Ovo su sve pitanja koja prave zbrku između dva developera, a šta ako ih je 100? Zbog toga je nastao Git- da olakša kolaboraciju (saradnju) većeg broja programera koji učestvuju u razvoju istog projekta. |  |

Git može da funkcioniše lokalno ili korišćenjem udaljenih repozitorijuma, kao što su GitHub ili GitLab. Ovde ćemo se dalje baviti GitHub-om. Treba napomenuti da se Git često meša sa GitHub-om (GitLab- ima ista svojstva i funkciju kao GitHub samo se drugačije zove, odnosno u vlasništvu je druge firme), ali tu treba napraviti razliku:

* GIT = software;
* GITHUB = website.

**Šta je GitHub?**

GitHub je besplatan server gde se može postaviti neki projekat i gde važi sve ono što smo rekli za Git, jer on funkcioniše na njegovim osnovama.

**Preuzimanje Git-a izvršiti sa stranice:**

<https://git-scm.com/downloads>

i odabrati odgovarajuću verziju u zavisnosti od sistema: macOS, Windows, Linux/Unix.

|  |  |
| --- | --- |
|  |  |

Dalje ne menjamo ništa već samo klikćemo na Next. Jedino što možemo čekirati je da nam se pojavi ikonica na Desctop-u i čekiramo Launch Git Bash pre klika na Finish.

|  |  |
| --- | --- |
|  |  |

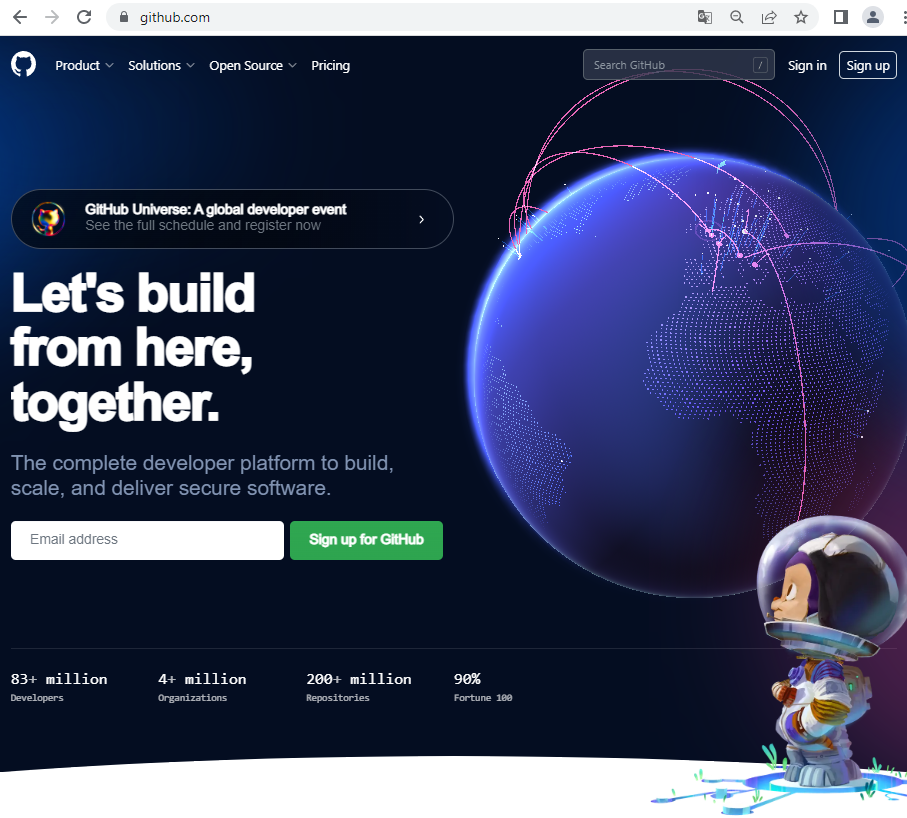
Pokrenuti Git izgleda ovako:



**Registraciju na GitHub izvršiti sa sledećeg linka:**

<https://github.com/>

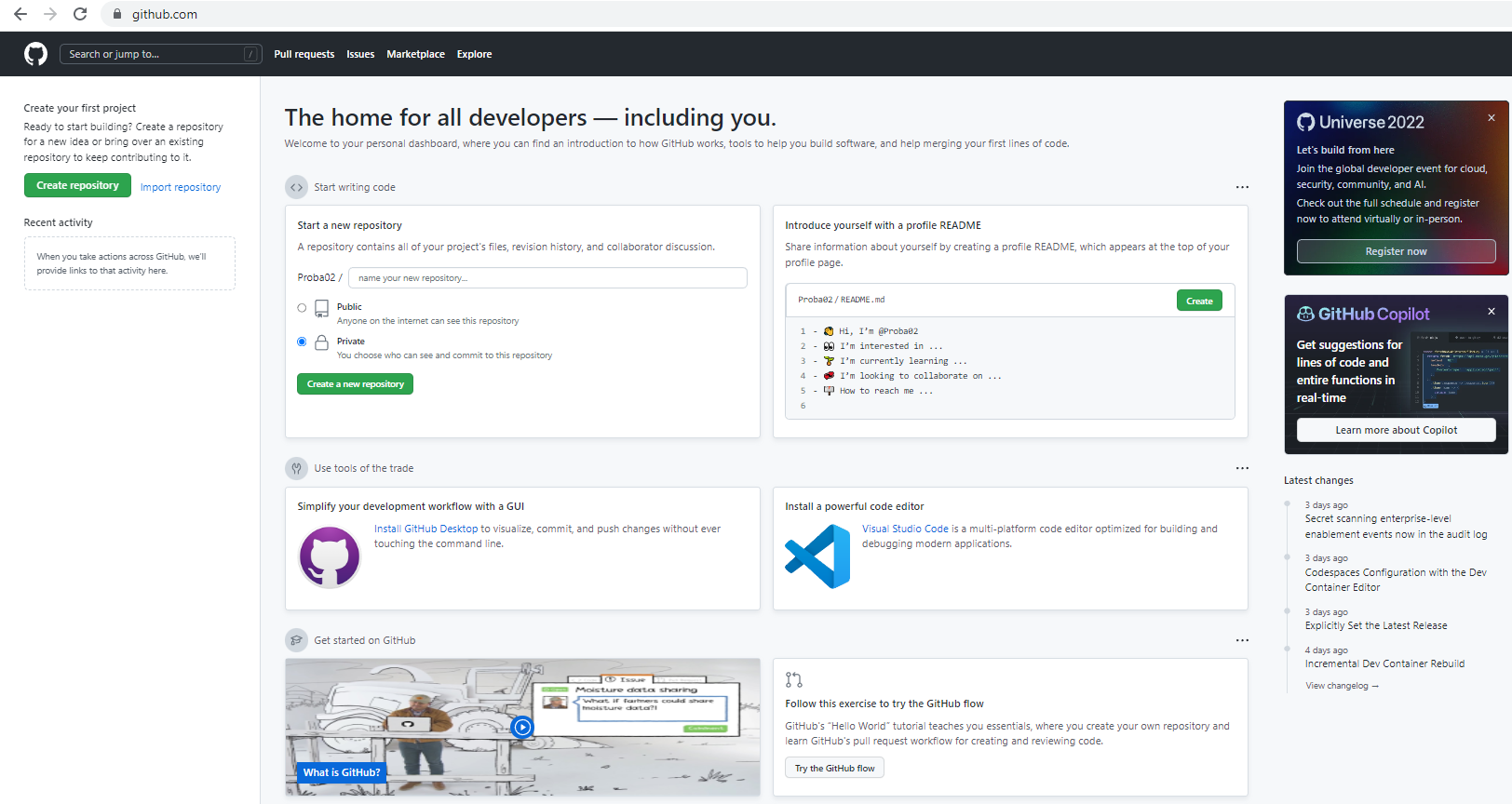
Kako bi napravili nalog kliknemo na Sing up.



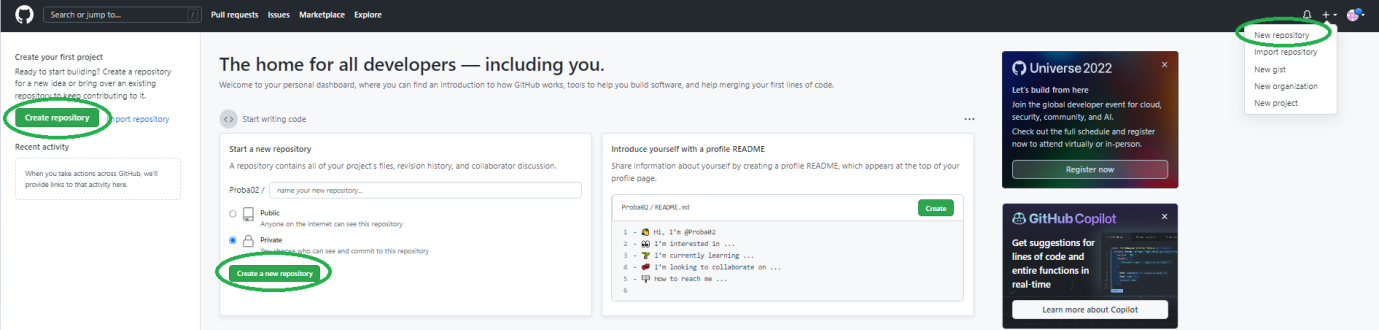
Registrujemo se tako što unesemo mail, šifru i korisničko ime (nakon toga će se prikazati polje gde će nas pitati da li želimo da primamo razna obaveštenja sa GitHub-a: proizvoljno stavimo y- ako hoćemo ili *n- ako nećemo*).

[*usethiscontacts@gmail.com*](mailto:usethiscontacts@gmail.com)*, Blaster1985+, Proba02*

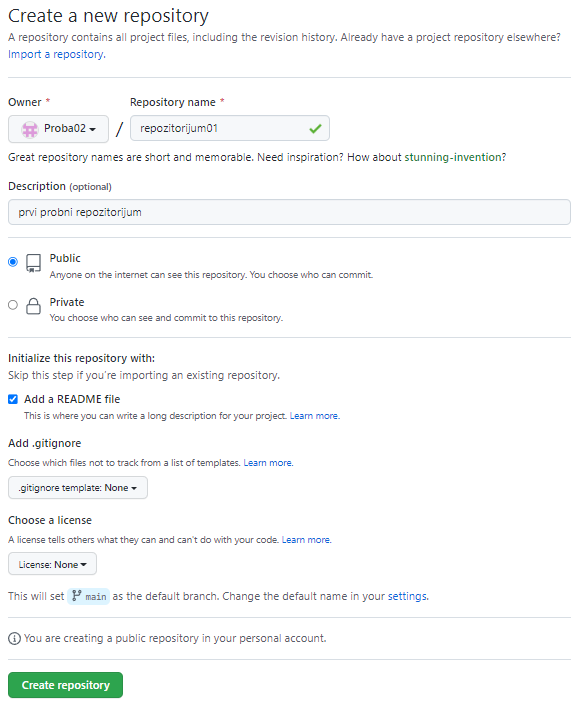
Nakon ovoga nalepimo šifru pristupa koju smo dobili na email, i kliknemo (skroz dole) na *skip personalization* ako ne želimo da dajemo određene informacije, i „ulazimo“ na GitHub.



|  |  |
| --- | --- |
| U gornjem desnom uglu imamo ikonicu koja nam pokazuje podatke o profilu, repozitorijumima, projektima, gde imamo Settings sa podešavanjima, Sing out gde možemo da se odlogujemo i dr.  **Suština GitHub-a su repozitorijumi.** Njih možemo uporediti sa folderima na našem Desctop-u.  Repozitorijum možemo napraviti na jedan od sledećih načina, odnosno klikom na dole prikazana polja, ili ako ih imamo na računaru (u obliku foldera, dokumenta i sl.) možemo ih importovati.  Zvonce u gornjem desnom uglu će prikazivati razna obaveštenja koja nam budu pristizala. |  |



Sada kliknemo na bilo koje od ova tri dugmeta da bi napravili novi repozitorijum.



Prilikom izbora imena repozitorijuma obično se bira ime kojim signaliziramo GitHub-u da nam za njega pruži hosting, npr oblika:

**bilokojeime.github.io**

Međutim pošto ovde vežbamo, daćemo ime tipa repozitorujum01.

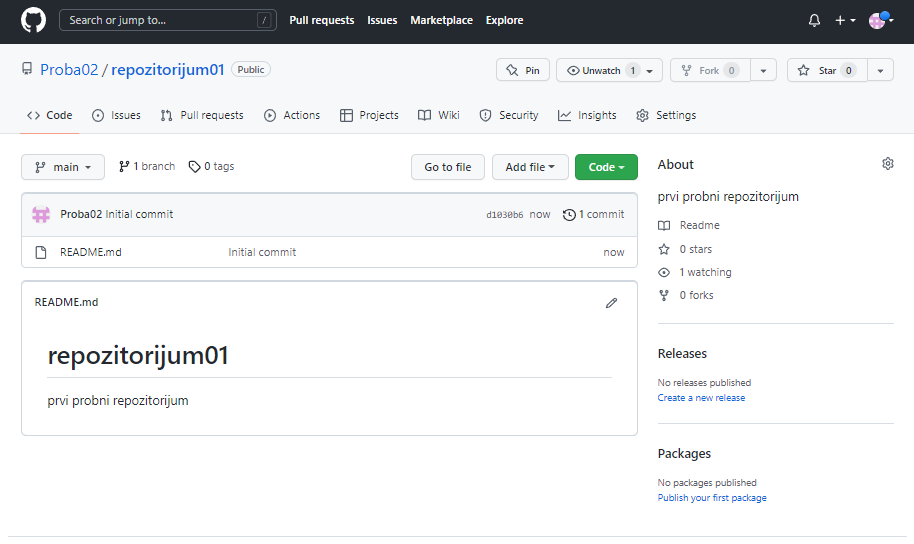
Description (opis) je opcionalan, što znači da nije obavezan.

Nakon toga imamo opcije Public ili Private. Koja je razlika? Što se tiče opcije **Public** to znači da će svako na GitHub-u moći da vidi naš repozitorijum i dokumenta u njemu, dok ako stavimo **Private**, onda moramo da damo dozvolu određenim ljudima da bi mogli videti sadržaj repozitorijuma. Ali ni u jednom ni u drugom slučaju oni neće moći da menjaju fajlove u repozitorijumu ili da brišu nešto bez dozvole i odobrenja.

Mi ćemo odabrati Public.

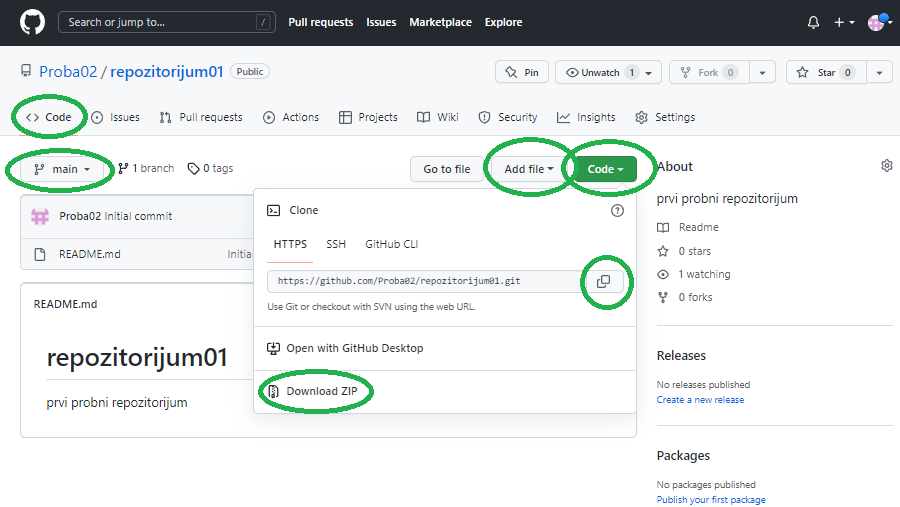
Obično se čekira i README file, koji je koristan jer u njega mogu da se ubace razne beleške, instrukcije i sl.

Nakon tih podešavanja možemo napraviti novi repozitorijum klikom na *Create repository*, i on će izgledati ovako:



**Kako funkcioniše GitHub repozitorijum?**

|  |  |
| --- | --- |
| Ako izađemo i uđemo u GitHub repozitorijum će se nalaziti u gornjoj levoj strani.  Pri kreiranju novog repozitorijuma početna stranica se nalazi u njegovoj kod sekciji, u kojoj se nalaze nekoliko važnih opcija:   * Code; * Add file; * main |  |

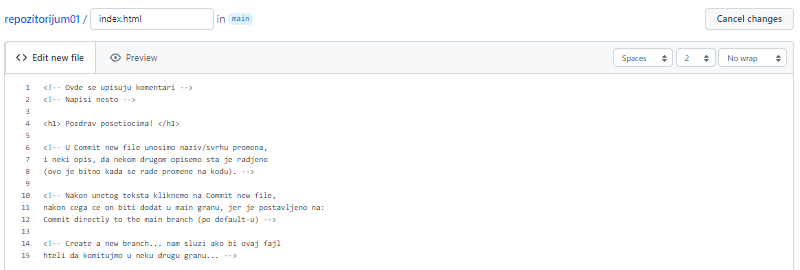


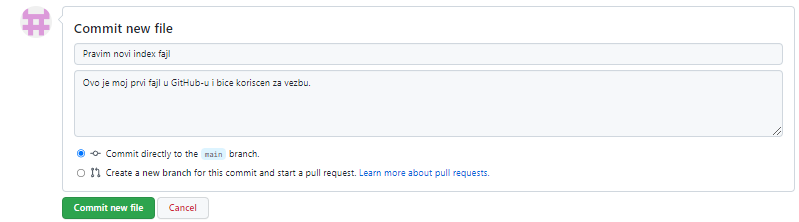
**Opcija Code** nam nudi dve bitne stvari:

1. možemo da kopiramo link (i npr. pošaljemo nekom kako bi mogao klikom na njega da pristupi našem repozitorijumu);
2. možemo da “skinemo” zipovan repozitorijum.

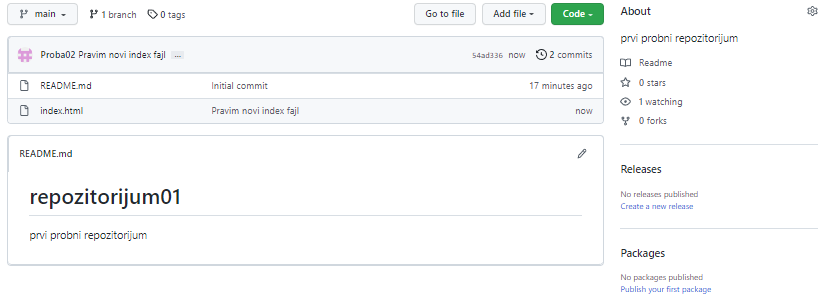
*Probajmo ovo kopirajući link repozitorijuma u bilo koji pretraživač.*

U opciji **Add file** možemo kreirati novi fajl klikom na *Create new file*, nakon čega će nam se pojaviti sledeće:





Klikom na **Commit new file** u repozitorijum vršimo prvu izmenu dodajući novi fajl pod nazivom index.html.



Klikom na repozitorijum01 možemo da ga pogledamo i imaćemo mogućnost za njegovo kopiranje, izmenu, brisanje, odnosno možemo da radimo šta hoćemo sa njim. Čemu ovo služi? Ovo je mesto gde će drugi ljudi pogledati naše fajlove, učestvovati u našim fajlovima, download-ovati ih i sl.

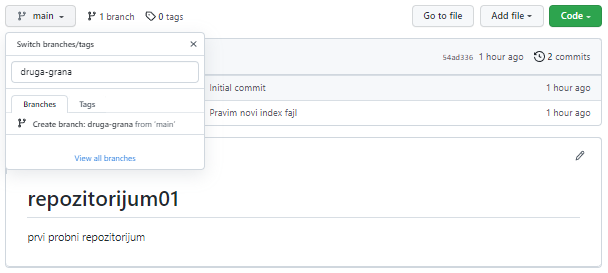
**Šta su GitHub grane?**

Naš novi fajl se automatki komitovao u glavnu-main granu.

Šta se dešava ako mi napravimo novu granu (create a branch…)?

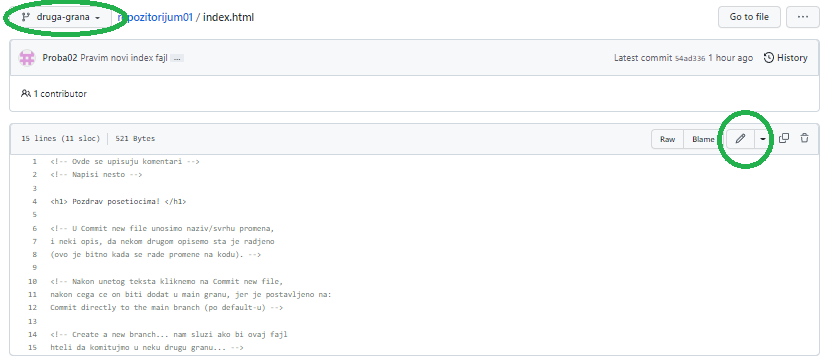
Ta nova grana, kako god je nazvali, će “povući” sve fajlove sa main grane- praktično oni će se pekopirati. Međutim bilo kakve izmene na tom fajlu, u toj novo grani, uticaće samo na njih.

Novi **branch (granu)** pravimo na sledeći način:



Ime nove grane obično nije druga-grana, već se ono daje po datumu izmene koda i imena projekta (npr. 15.05.2022. Skolski sajt-naslovna strana), ali sada ćemo uzeti ovo, i kliknuti na *Create branch: druga-grana*.

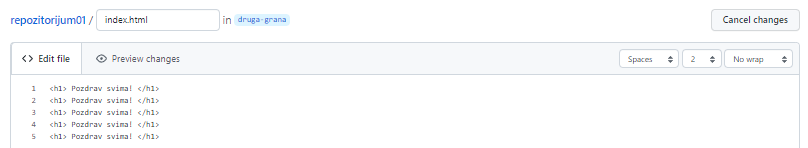
Kreiranjem novog branch-a druga-grana vidimo da su nam fajlovi isti kao u main grani, otvorićemo index.html u druga-grana i izmeniti ga dodajući neki novi unos.

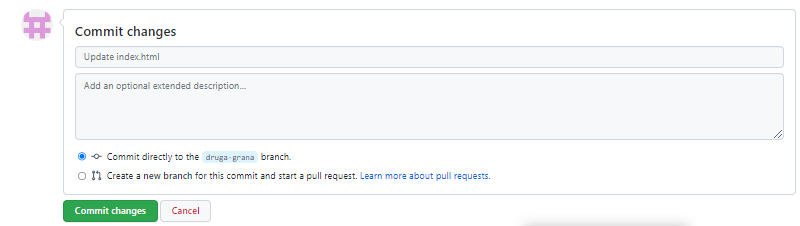


Kursor nam je automatski postavljen na: Commit directly to the druga-grana branch.

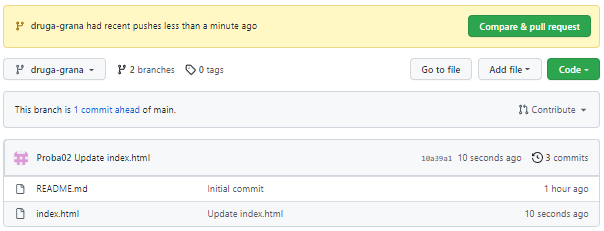
Zatim izvršimo promene na drugoj grani klikom na *Commit changes*.

Nakon ovoga vidimo drugačiji tekst u zavisnosti od izabranih grana.

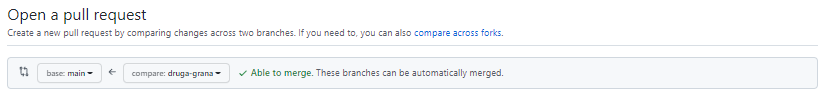




Posle ovoga vratimo se na Code sekciju, u kojoj sada možemo videti opciju ***Compare and pull request.***



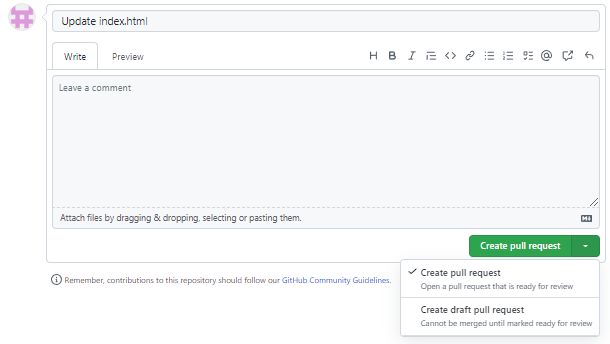
Klikom na ovu opciju videćemo kakve su promene izvršene. Crveno polje i minus označava šta je obrisano, a zeleno polje i plus označava šta je dodato. Ova opcija nam nudi da se dve grane spoje tj. merge-uju, tako što će sadržaj iz druge grane biti kopiran u main granu.



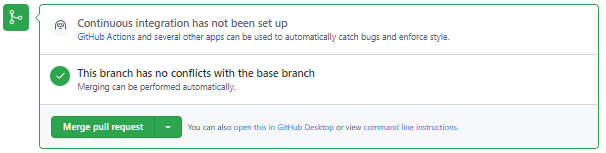
Da smo ovde imali npr. deset grana mogli bi da izaberemo iz koje grane ćemo sadržaj povući u main granu. Odnosno ako deset programera pošalje svoje rešenje nekog koda, mi za main granu izaberemo najbolje.

Ovo će se izvršiti tako što redom kliknemo na:

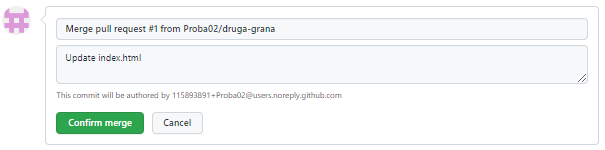
1. *Create pull request* (Ukoliko želimo nešto da zadržimo kliknemo na *Create draft pull request):*



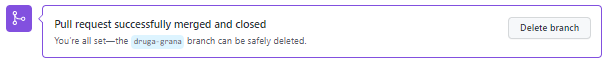
*2. Merge pull request.*



*3. Confirm merge*



Nakon čega će nam biti ispisana poruka:

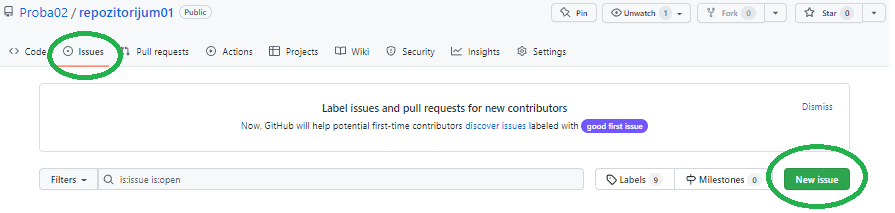


Uspešnost ovoga možemo proveriti tako što se vratimo na Code sekciju i u main grani vidimo sadržaj index.html-a, ako je isti kao u druga-grana dobro smo uradili.

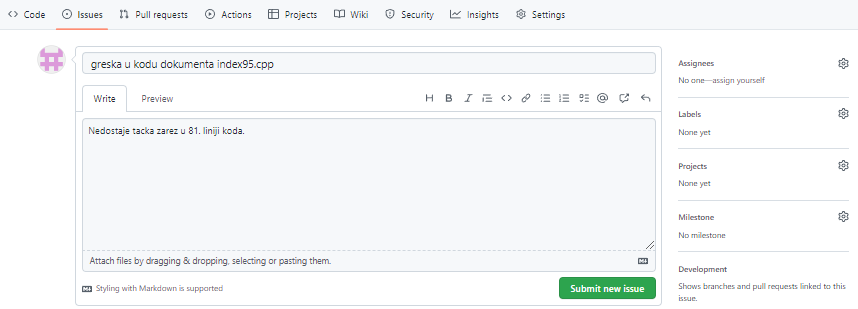
**Ostale sekcije GitHub-a:**

* **Sekcija Issues.**

**Issues (problemi)** je sekcija gde se mogu zabeležiti uočene greške i problemi. Zabelešku može uraditi vlasnik naloga ili korisnik koji ima predlog za neku promenu.

****

Greška se beleži klikom na *New issue*, posle čega se pritisne *Submit new issue*.

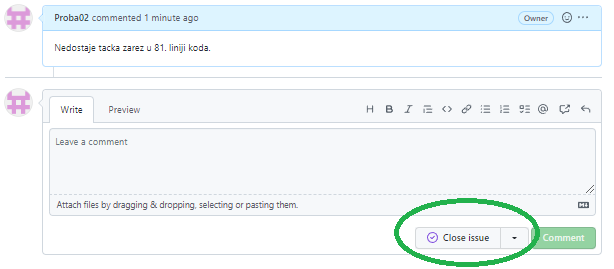


Nakon klika na *Submit new issue* greška će biti zabeležena pod datim nazivom i biće aktivna.

Klikom na sekciju Issue dobićemo sledeće:



Nakon što bude sređena ona se može zatvoriti klikom na nju i na komandu **Close issue**.

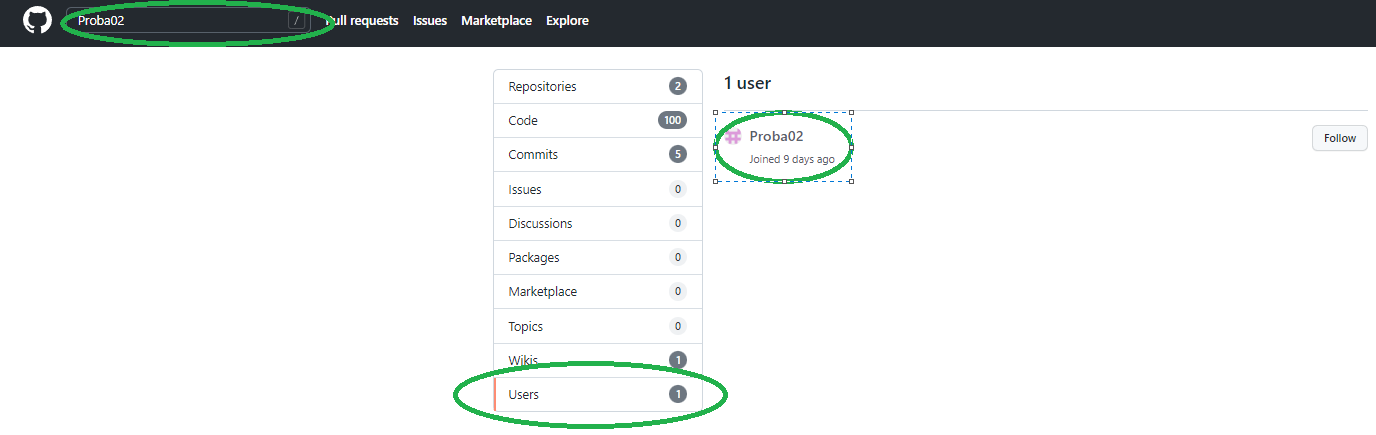


Greška se može dodeliti i drugima da je rešavaju (ako na primer imamo neki tim) klikom na **Assigments**, mogu se dodati rokovi, može se obrisati klikom na **Delete issues** (ako ne želimo da se više vidi u Issue sekciji) i dr.

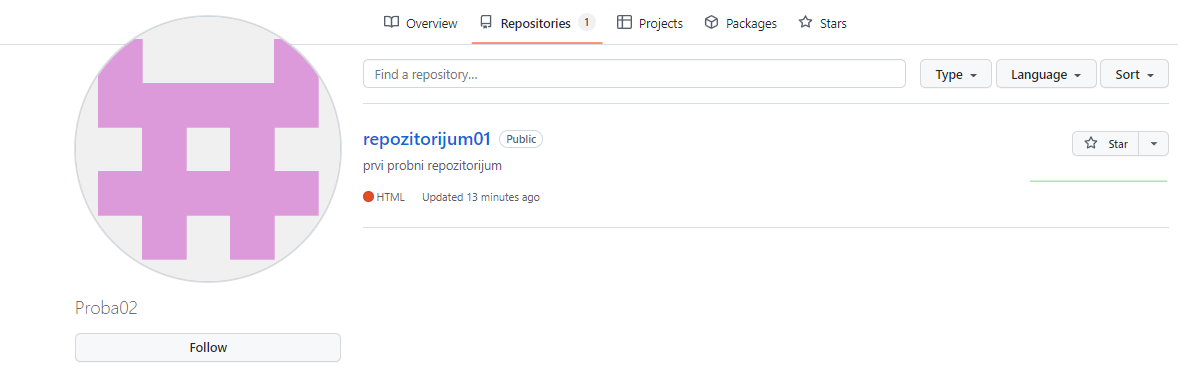
* **Pull requests sekcija *(potreban je jos 1 repozitorijum)***

**Pull requests** sadrži predložene izmene od strane drugih korisnika.

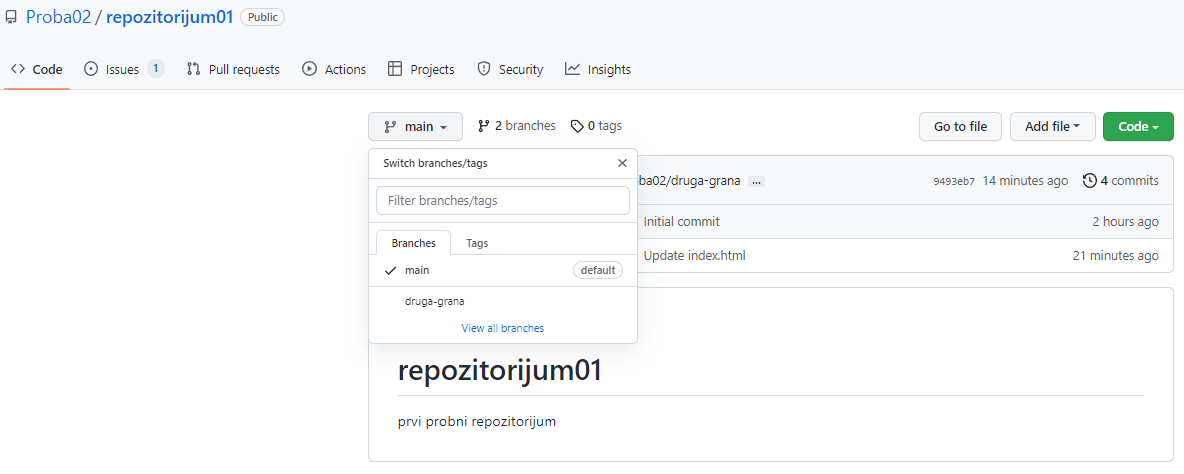
Npr. za ovaj deo treba nam još jedan nalog. U njemu u *Search-u* ukucamo traženi GitHub nalog i u Users odemo na njega (u ovom slučaju Proba02).



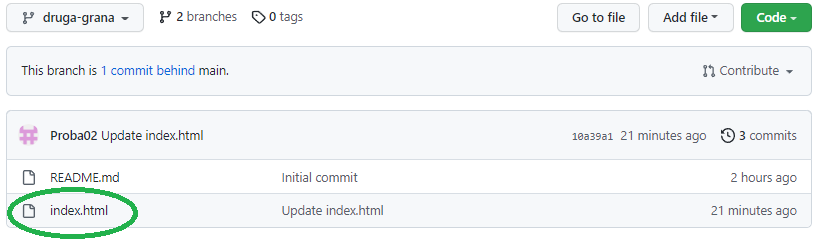
Na novom nalogu u Repositories možemo videti sve repozitorijume tog GitHub naloga.

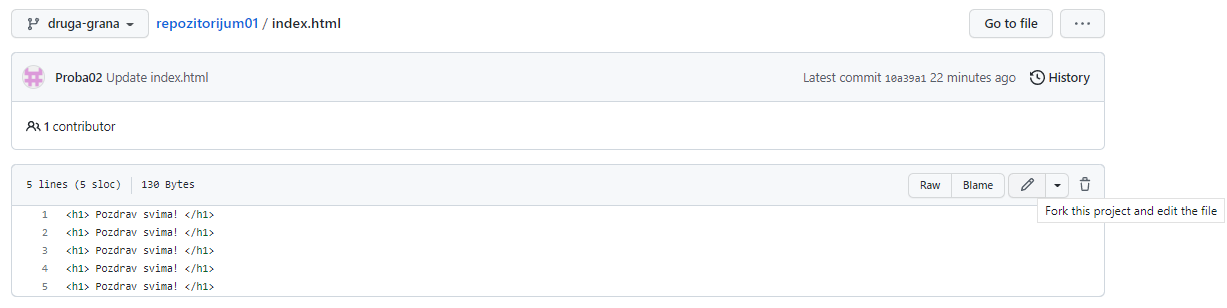


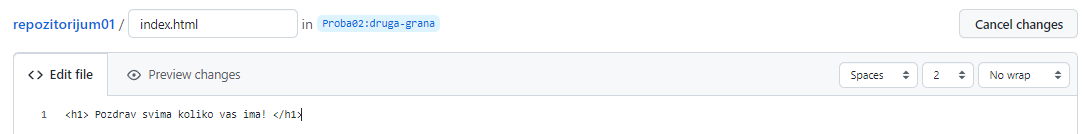
Kada kliknemo na bilo koji od repozitorijuma možemo videti sve što bi videli i sa glavnog profila zato što su Public. Izaberemo repozitorijum01 i uđemo u druga-grana.

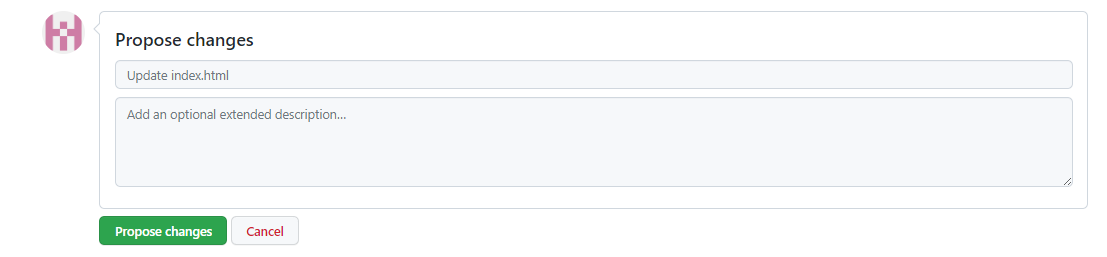


Izaberemo Index.html dokument u kome nešto piše i izmenimo taj tekst, klikom na Fork this project and edit the file, nakon čega snimimo promene klikom na Propose changes (ukoliko hoćemo možemo ostaviti i komentar).

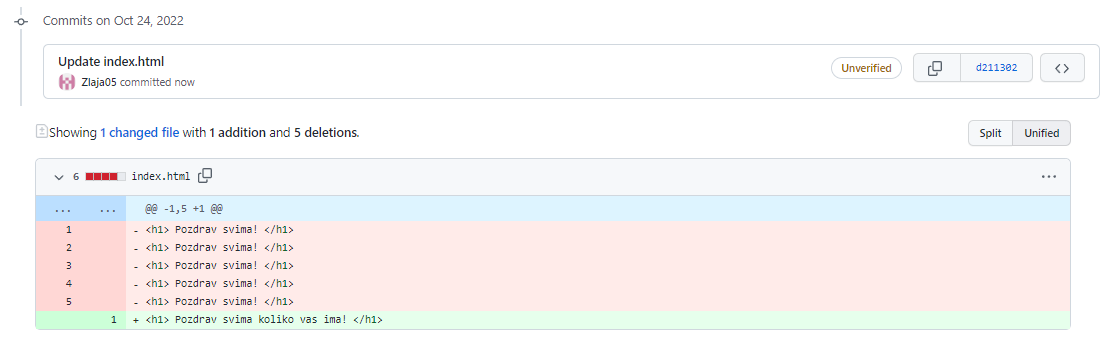




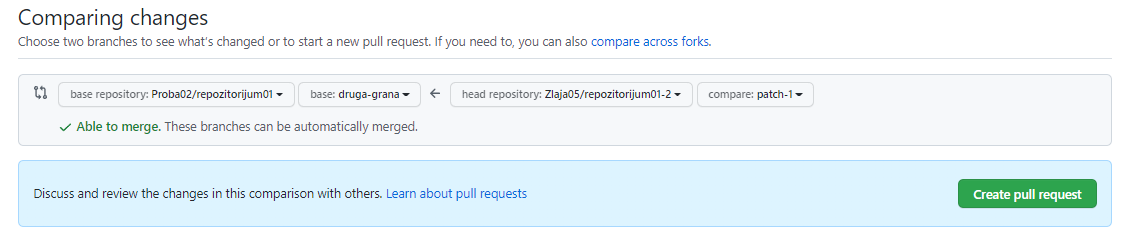




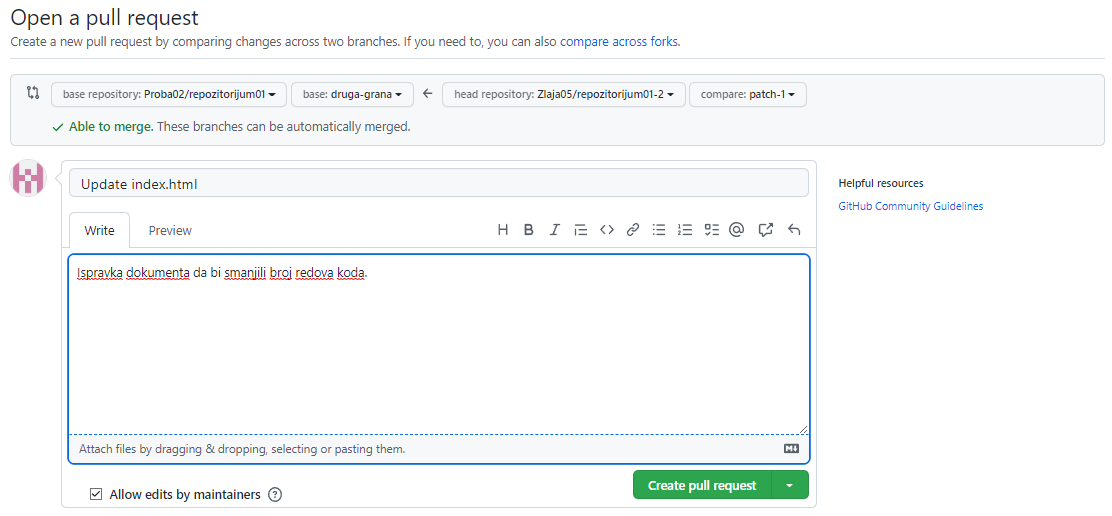
Posle kreiranja zahteva možemo uporediti fajl pre i posle kreiranja promena.



Kada proverimo da li su to naše željene promene kliknemo na *Create pull request*, kako bi tu promenu zahtevali od vlasnika repozitorijuma.

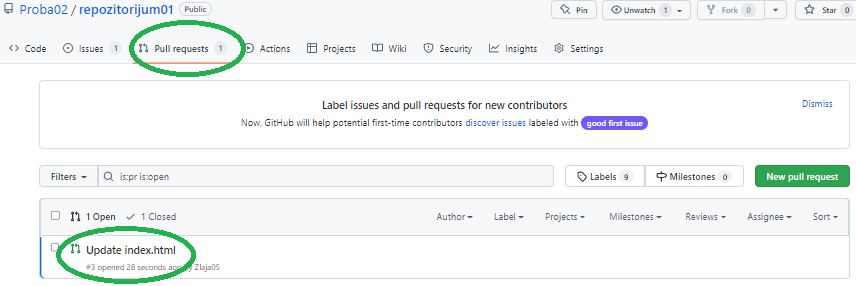


Pružiće nam se i mogućnost ostavljanja komentara gde npr. možemo reći šta smo izmenili, zašto smo to izmenili i slično, nakon čega opet kliknemo na *Create pull request*.



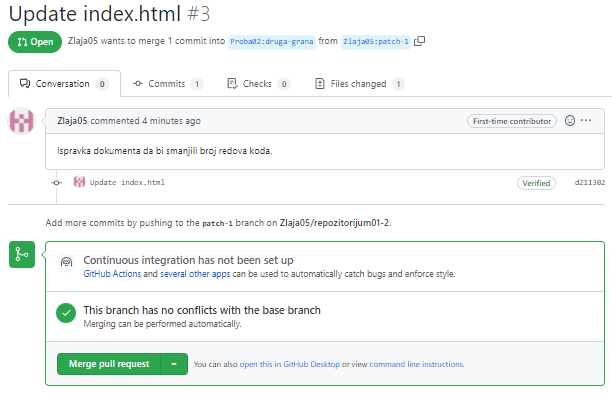
Sada je zahtev otišao na GitHub nalog gde se nalazi ovaj repozitorijum.

Tako da ćemo otići na taj nalog, kliknuti na *Pull requests* i na predloženu izmenu.

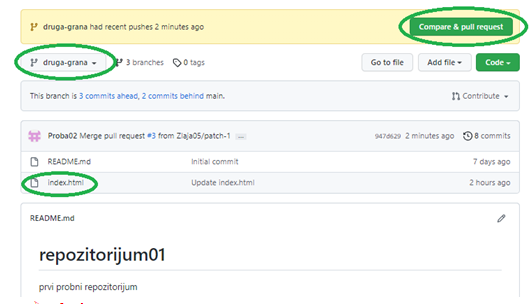


Na *Commits* možemo videti kakva je to promena i na *Files Changed* šta je izmenjeno.

Ako želimo da prihvatimo promenu kliknemo na Klikom na *Merge pull request*, zatim na *Confirm merge*.



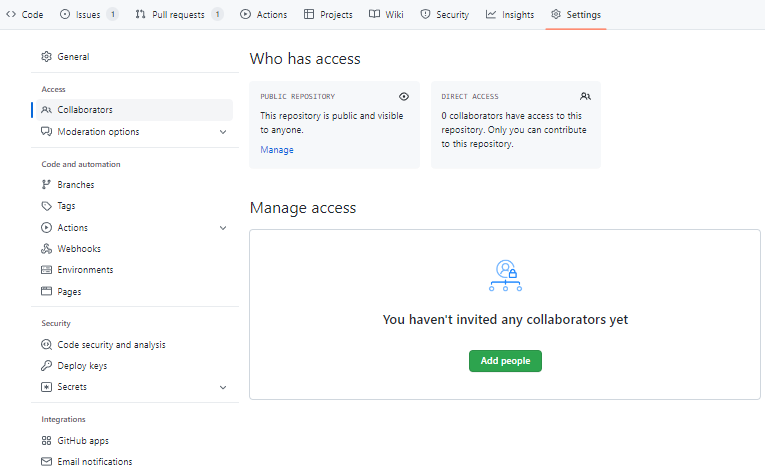
Da li je sve odrađeno kako treba možemo proveriti u druga-grana klikom na index.html, i ako jeste možemo (a i ne moramo) to merge-ovati u main grani.



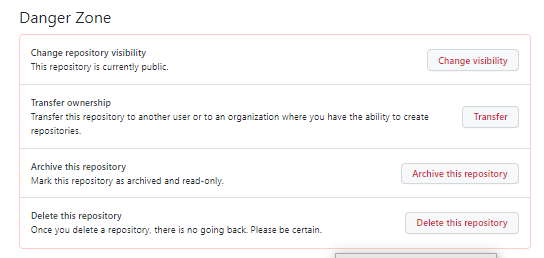
* **Actions**
* **Projects**
* **Wiki**
* **Security**
* **Insights**
* **Settings sekcija**

Ovi oblici i nisu toliko bitni u komunikaciji sa drugim developerima, tako da ćemo ih preskočiti. Možemo jedino napomenuti da u Settings sekciji imamo neke interesantne stvari kao što su Collaborations (Saradnici), gde možemo da:

* dodajemo ljude- *Add people*;
* menjamo vidljivost- Public repository/ Private repository- *Manage***.**



Klikom na **Manage** ne smo da možemo promeniti vidljivost, već možemo izvršiti Transfer repozitorijuma, tj. promenu vlasništva repozitorijuma, možemo da ga arhiviramo, odnosno da bude vidljiv samo za čitanje (ne mogu se vršiti izmene) i možemo ga obrisati.

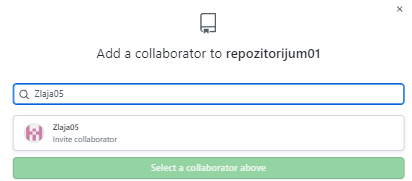


Change Visibility:

1. prva varijanta je prebacivanje određenog repozitorijuma iz Public u Private;
2. druga varijanta je obrnuto.

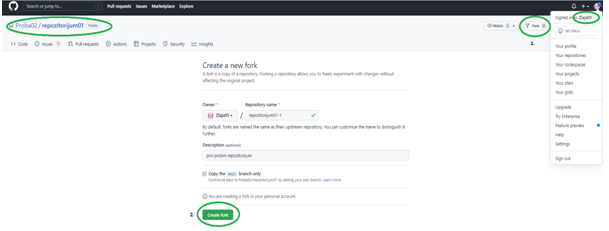
|  |  |
| --- | --- |
|  |  |

Dodavanje saradnika- **Add people** se vrši kad je Private repozitorijum, kako bi ga samo njima načinili Public, tj. on će biti vidljiv za saradnike, koje dodajemo unoseći ime njihovog naloga, i klikom na Select a collaborator above.



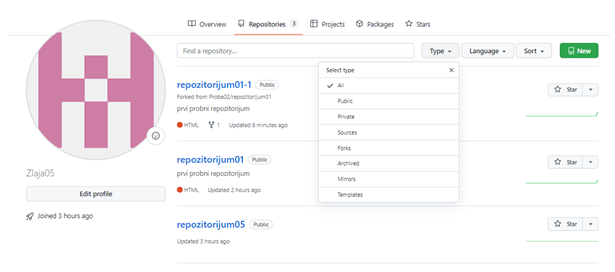
**Koja je uloga komande fork?**

Komanda fork nam služi da “prekopiramo” repozitorijum sa jednog GitHub naloga na drugi.



U profilu nam je jedan nalog a mi “forkujemo” repozitori sa naloga Proba02.

Ako izlistamo sve repozitorijume u Type možemo selektovati samo forkovane (u ovom slučaju to je repozitorijum01-1).

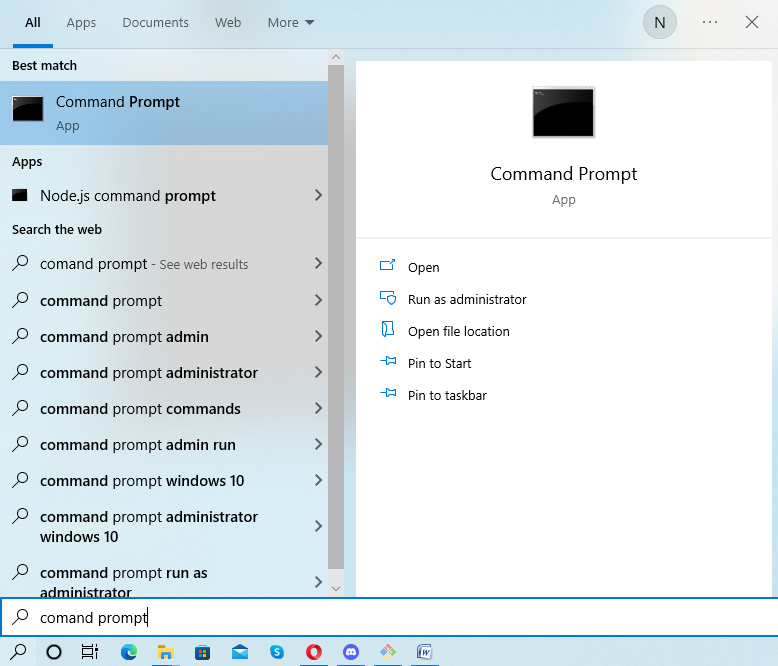


GIT SOFTVER

**Koje su git komande?**

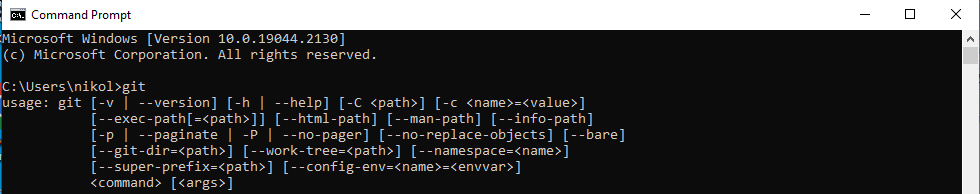
Ovaj odgovor nam daje Command Prompt.

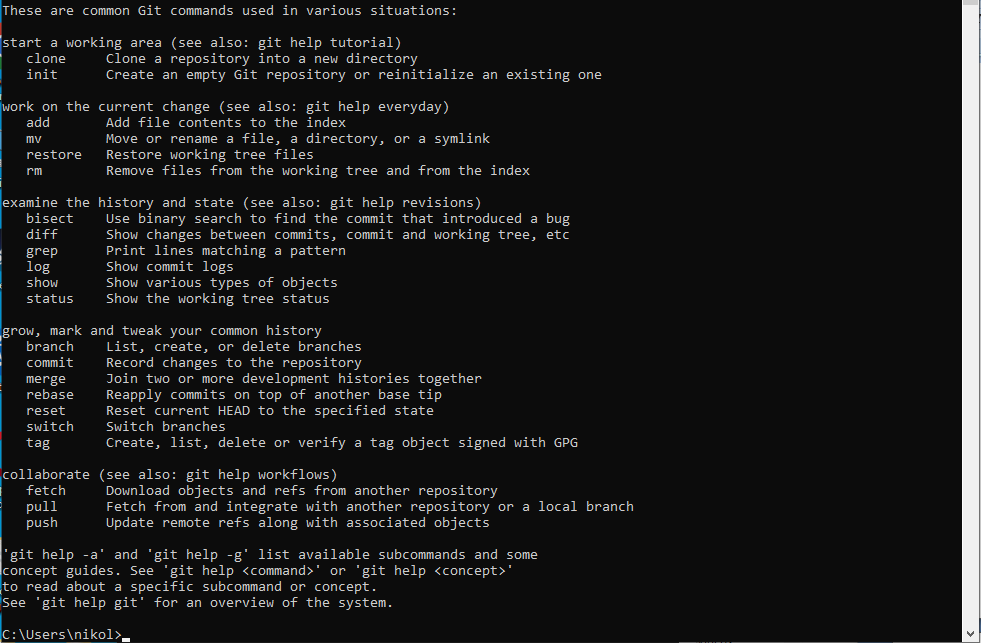
U Search nađemo Command Prompt i u njemu unesemo komandu git.



Nakon pritiska na Enter dobićemo git komande.

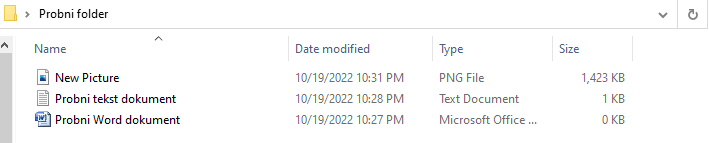




****

**Kako da neki folder sa dokumentima gurnemo u GitHub?**

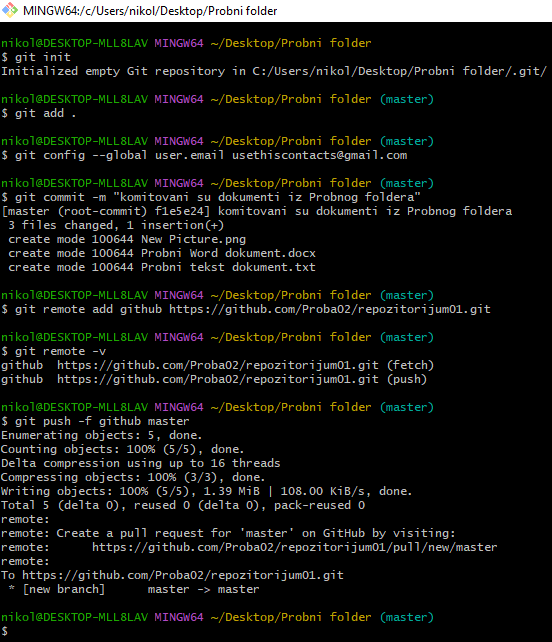
Napravimo folder pod nazivom Probni folder i unutar njega napravimo neke fajlove:



|  |  |
| --- | --- |
| Zatim stisnemo desni klik na *Probni folder* i kliknemo na **Git Bash here**.  Nakon čega će se otvoriti Git (koji pokazuje da je Git pozicioniran na Desctop-u, u Probni folder): |  |

Za prebacivanje dokumenata u folderu na GitHub repozitorijum koristimo sledeće komande:

1. $ git init
2. $ git add .
3. $ git config --global user.email [*naziv.mejl.adrese@mail.com*](mailto:naziv.mejl.adrese@mail.com)
4. $ git commit –m”unesi proizvoljni komentar”
5. git remote add github [*https://github.com/...../kopirana*](https://github.com/...../kopirana) *putanja do git repozitorijuma*
6. git remote –v
7. git push –f github master



**Kako da ovaj dokument preuzmemo na desktop, izvršimo izmenu u novoj grani i “vratimo” na GitHub?**

1. $ git version
2. $ mkdir *gitVezbe*
3. $ cd *gitVezbe*
4. $ git clone *https://github.com/.../...*
5. $ cd *repozitorijum01*
6. $ git status
7. $ ls
8. $ touch test.txt
9. $ ls
10. $ git status
11. $ git add test.txt
12. $ git status
13. $ git commit –m “komentar”
14. $ git branch –M main
15. git push –uf origin main

giKonfiguracija korisnika:

> git config --global user.name "Ana Anic"

> git config --global user.email "mojMejl@gmail.com"

------------------------------------

> mkdir mojFolder // kreiranje novog foldera sa nazivom mojFolder

> cd mojFolder // pozicioniranje u novokreirani folder

> git init // inicijalizacija novog repozitorijuma

> ls // prikaz sadrzaja foldera

> ls -a // prikaz svih fajlova u folderu

> touch test.txt // kreiranje novog fajla

> git status // provera da li je nesto izmenjeno

> git log // pregled istorije komita

> git add . // dodavanje u indeks

> git status

> git commit -m "Initial commit" // komitovanje sa porukom Initial commit

> git status

> git log

-------------------------------------

Grananje i spajanje bez konflikta

-------------------------------------

> git branch // prikaz svih grana

> git branch nova\_grana // kreiranje nove grane

> git checkout nova\_grana // prelaz na novu granu

> git branch

> touch test2.txt

> git add .

> git commit -m "Nova izmena na novoj grani"

> git log

> git checkout master

> git log

> git merge nova\_grana // spajanje grane nova\_grana u master granu

> git log

-------------------------------------

Resavanje konflikta

-------------------------------------

> git checkout nova\_grana

Napraviti izmenu u fajlu i komitovati:

> git add .

> git commit -m "Izmena na novoj grani"

> git checkout master

Napraviti izmenu u istom fajlu na istoj liniji i komitovati:

> git add .

> git commit -m "Izmena na master grani"

> git merge nova\_grana // spajanje grane nova\_grana u master granu

Resiti konflikt u fajlu

> git add .

> git commit -m "Resen konflikt"

> git log

> git status

-------------------------------------

Povezivanje lokalnog repozitorijuma

-------------------------------------

> mkdir noviFolder

> cd noviFolder

> git init

> touch proba.txt

> git add .

> git commit -m "Initial commit"

Napraviti novi repozitorijum na GitLab-u i kopirati komande iz README.md fajl-a:

> git remote add origin URL

> git branch -M main

> git push -uf origin main // ako se prikaze greska otvoriti u projektu: Settings > Repository > Protected branches i dozvoliti force push na main granu

----------------------------------------

Preuzimanje repozitorijuma sa GitLab-a

----------------------------------------

> mkdir folder2

> cd folder2

> git clone URL