SEMINARSKI RAD I OBJEKTNO OBJEKTNO ORIJENTISANOG PROGRAMIRANJA

Uvod u GIT

Radila : Vucenic Elena III-10

Treca beogradska gimnazija 2022

SADRZAJ

[1. Uvod 2](#_Toc120974274)

[2. GIT istorija 2](#_Toc120974275)

[3. Upotreba 2](#_Toc120974276)

[4. Repozitorijum 2](#_Toc120974277)

[5. Datoteke 3](#_Toc120974278)

[6. GITHUB 4](#_Toc120974279)

[7. Proces rada sa komandama 4](#_Toc120974280)

[8. Osnovne komande 5](#_Toc120974281)

[git config 5](#_Toc120974282)

[git init 5](#_Toc120974283)

[git clone 5](#_Toc120974284)

[git status 6](#_Toc120974285)

[git add 6](#_Toc120974286)

[git commit 6](#_Toc120974287)

[git branch 6](#_Toc120974288)

[git push 7](#_Toc120974289)

[git fetch 7](#_Toc120974290)

[git merge 7](#_Toc120974291)

[git pull 8](#_Toc120974292)

[git revert 8](#_Toc120974293)

[git log 8](#_Toc120974294)

[9. SSH kljuc 8](#_Toc120974295)

[10. Jednostavan primer upotrebe osnovnih komandi 9](#_Toc120974296)

[1. Korak: Konfiguracija 9](#_Toc120974297)

[2. Korak: Pravimo novi folder 10](#_Toc120974298)

[3. Korak: Pravimo novi repozitorijum 11](#_Toc120974299)

[4. Korak: Inicijalizacija repozitorijuma 12](#_Toc120974300)

[5. Korak : Ubacivanje fajla 12](#_Toc120974301)

[6. Korak: Komitovanje promena 13](#_Toc120974302)

[7. Korak: Pushovanje fajlova 14](#_Toc120974303)

# Uvod

GIT je :

1. najpopularniji *version control system*
2. sofver koji služi za praćenje promena u kodu
3. koordinaciju rada više članova tima koji rade na istom projektu

# GIT istorija

Git je kreiran od strane Linux Torvalds-a, 2005 godine za razvijanje Linux kernels.

Torvalds je želeo raspoloživ sistem koji bi koristio kao bolju i besplatnu alternativu BitKeeper-u (source-control management (SCM) sistem koji su koristili za održavanje projekata početkom 2002. godine).

# Upotreba

Kada ga koristimo?

Git vam pomaže da **sinhronizujete vaš kod** sa drugom osobom, ili čak sa celim timom.

Zamislite da radite sa vašim kolegom na istom projektu, i oboje radite na istim fajlovima unutar projekta. Git uzima vaše promene i promene vašeg kolege zasebno i spaja ih u jedan “Master” repozitorijum. Dakle koristeći Git možete biti sigurno da svi radite na istoj verziji repozitorijuma, što isključuje brigu o tome da li ste poslali fajlove između sebe putem mejla ili USB stick-a. Samim tim kolaboracija, tj. zajednički rad postaje vrlo lagan.

# Repozitorijum

* Šta je repozitorijum (eng.“repository”)?

**Repozitorijum,** odnosno Git projekat predstavlja kolekciju datoteka i direktorijuma u vezi sa projektom, uključujući i istoriju revizija za svaku pojedinačnu datoteku. Istorija predstavlja niz verzija iste datoteke u različitim vremenskim trenucima kada je izvršena operacija postavljanja (eng. commit).

* Zasto ih koristimo?

Rad sa repozitorijumima olakšava **organizovanje i zaštitu koda**, bez čega bi rad na velikim open-source projektima bio praktično nezamisliv. Učesnici u razvoju mogu ispravljati greške ili dodavati nove funkcionalnosti bez bojazni da će time pokvariti ili poremetiti glavni tok razvoja projekta. Ovo je omogućeno primenom tematskih grana koje se mogu na jednostavan način kreirati, pripojiti glavnom toku.

Uz pomoć mrežnih platformi kao što su *GitHub* ili *Bitbucket*, otvaraju se dodatne mogućnosti kada je u pitanju saradnja i transparentnost tokom razvoja projekta.

# Datoteke

Svaka datoteka u radnom direktorijumu može se nalaziti u jednom od dva stanja:

 Nepraćena datoteka (untracked)

 Praćena datoteka (tracked)

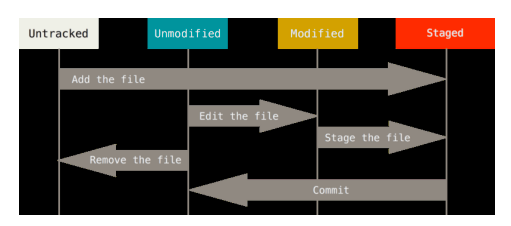
Datotetke koje pripadaju projektu (pracene datoteke) u svakom trenutku se mogu nalaziti u jednom od tri osnovna stanja:

 **Postavljeno (committed)** – podrazumeva da je datoteka bezbedno smeštena u lokalnu bazu podataka

 **Izmenjeno (modified)** – sadržaj datoteke je izmenjen od poslednjeg postavljanja, ali sadržaj još nije u bazi

 **Spremno za postavljanje (staged)** – izmenjena datoteka je označena za postavljanje

Promene stanja datoteka i operacije koje dovode do njih prikazane su na slici:



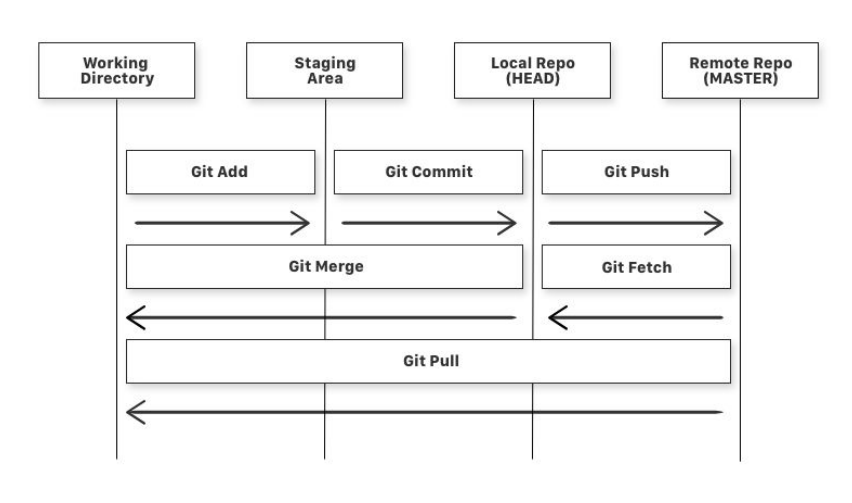
# GITHUB

* Da li je Git isto kao i GitHub?

Git i GitHub su dve različite stvari.

* GitHub je kompanija osnovana 2008. godine, koja razvija alatku čija je svrha integracija sa Git-om.
* Nije vam potreban GitHub kako bi koristili Git, ali **ne možete koristiti GitHub bez Git-a**.

# Proces rada sa komandama



Slika 1

# Osnovne komande

### git config

Ovom komandom vrsimo nasu konfiguraciju sa username-om i email-om (naloga na github-u).

### git init

Ovim se inicijalizuje git projekat.

### ****git clone****

Ako ne incijalizujemo sami git projekat, već trebamo raditi na nekom koji već postoji negde. Onda moramo koristiti ovu funkciju. Kao što piše, ovim kloniramo kompletan repozitorijum sa istorijom svih promena, sa svakim njihovim commitom.

Postoje dva načina za kloniranje : ssh i https, svaki zahteva drugačiji nivo autentifikacije, dok neki i ne zahtevaju(public repo).

### ****git status****

Ova komada ispisuje stagovane promene, fajlove koje ne pratite, obrisane fajlove ili jednostavno sve ono što će se desiti ako uradite neku akciju.

### ****git add****

Kako bi git pratio vaše fajlove, morate ih dodati na listu za praćenje. To se radi ovom komandom i unutar direktorijuma projekta.

### ****git commit****

Kada komitujete trenutno sačuvano stanje direktorijuma, pravite snapshoot. Tačku u vremenu koju će pratiti commit hash koji se automatski generisao. Za commitovane a ne pushovane promene kažemo da su Stageovane. Dobro je uvek ostaviti deskrpitivni komentar svakog commita, kako bi znali šta smo radili u tom trenutku.

### ****git branch****

Rekli smo da je namena gita prvenstveno za verzionisanje vašeg koda. Brancheve možete posmatrati kao odvojene vremenske linije istog projekta. Ovo je komanda za grane. Koristimo sledece njene komande:

git branch <naziv\_nove\_grane > - definisemo novu granu

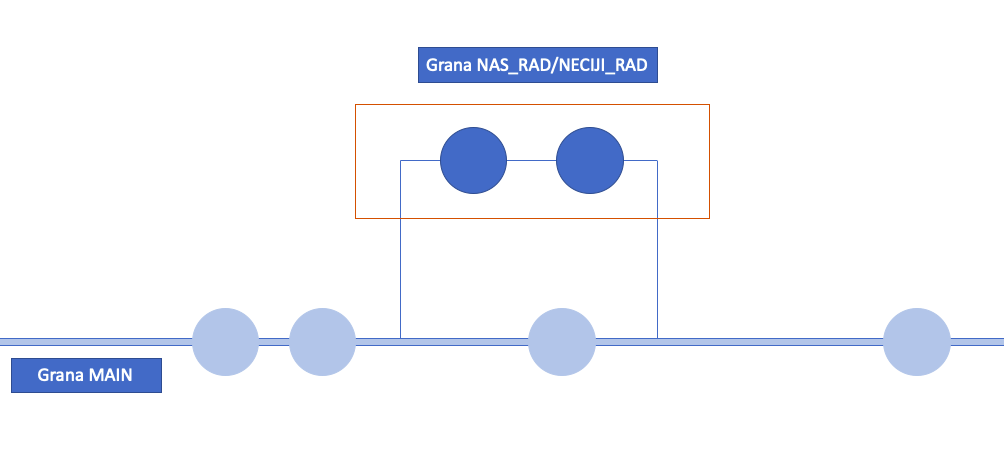
git branch -d <naziv\_grane\_koju\_zelimo\_obrisati > - brisemo datu granu

git branch --list – lista svih grana

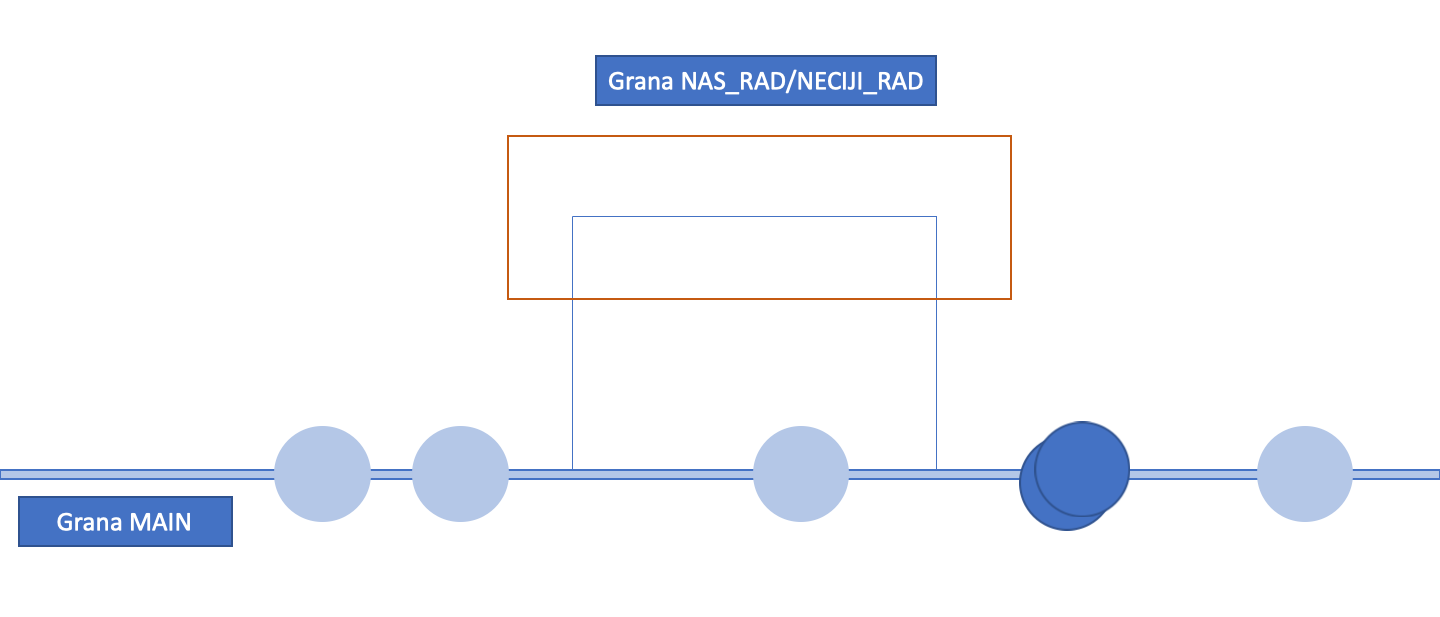
#### **Malo o Git branching-u**

Git grananje omogućuje razvojnim programerima odstupanje od proizvodne verzije koda kako bi popravili grešku. Programeri stvaraju grane za rad s kopijom koda bez menjanja postojeće verzije. Grane stvarate kako biste izolovali promene koda, koje testirate pre spajanja s glavnom granom.

PRE



POSLE



### ****git push****

Sve ono što ste dodali u commit(stageovane promene) sada treba gurnuti na server. I tako završiti kompletan ciklus jednog commita.

### git fetch

Ova komanda se koristi kako bi preuzeli fajlove iz remote repozitorijuma u lokalni, ali ne i u radni direktorijum. (pogledati Slika 1)

### git merge

Ova komanda služi kako bi fajlove iz lokalnog repozitorijuma preuzeli u radni repozitorijum. (pogledati Slika 1)

### git pull

Ova komanda služi kako bi se svi fajlovi preuzeli iz remote repozitorijuma direktno u radni repo, komanda je slična git fetch i git merge komandama. (pogledati Slika 1)

### git revert

Ovo je komanda koju koristimo kada zelimo da uzmemo prethodni commit I dodamo ga kao novi, a da se pritom da log ostane netaknut.

Da biste se vratili na ranije predaje, upotrebite *git revert HEAD~x* (x je broj - 1 povratak, 2 povratka, itd.)

### git log

Naredba git log prikazuje zadani izlaz za brzi pregled commit istorije.

# SSH kljuc

Git SSH ili sigurna ljuska mrežni je protokol za sigurno šifrovanje svih podataka koji se šalju s računara na server putem Interneta.

* Vaš **javni SSH ključ** je kao vaše korisničko ime ili identitet i možete ga deliti sa bilo kim.
* Vaš **privatni SSH ključ** je poput lozinke i čuva se lokalno na vašem računaru. Ali to je kao da čuvate svoje lozinke na lepljivom listu – svako može da ih vidi ako im ima pristup. Iz bezbednosnih razloga, SSH će od vas tražiti lozinku kada generišete ključeve (nadam se da niste preskočili ovaj korak) i koristiće tu lozinku za šifrovanje i dešifrovanje vašeg privatnog ključa.

Međutim, to znači da ćete morati da unesete svoju lozinku svaki put kada budete morali da koristite svoj privatni ključ, što će postati dosadno. Da bi se ovo rešilo, većina SSH implementacija će koristiti **agenta**, koji pamti vaš dešifrovani ključ. To znači da ćete morati da ga otključate samo jednom i trajaće sve dok ga ponovo ne pokrenete, omogućavajući vam da se bezbedno prijavite na svoje servere bez zatražene lozinke.

! Za generisanje ssh kljuca pogledajte sledeci link :

<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

# Jednostavan primer upotrebe osnovnih komandi

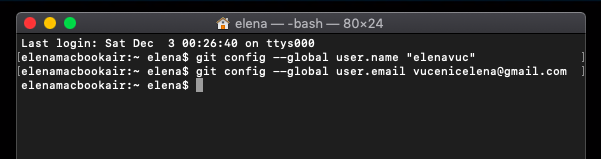
Na narednim stranicama je objasnjena upotreba navedenih osnovnih komandi.

## Korak: Konfiguracija

Prvo moramo potvrditi naš identitet u Gitu sledecim komandama:

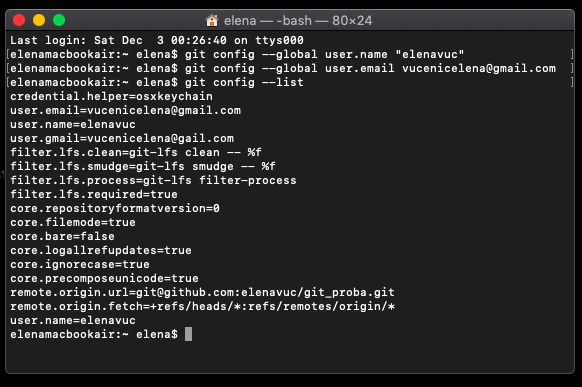
git config --global user.name “username”

git config --global user.email [example@gmail.com](mailto:example@gmail.com) – (kod email-a se ne stavljaju navodnici)



Uz pomoc sledece komande proveravamo nasu konfiguraciju:

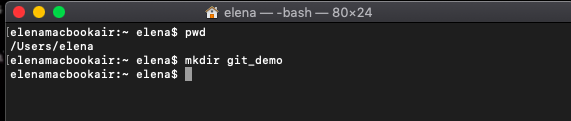
git config --list



## Korak: Novi folder

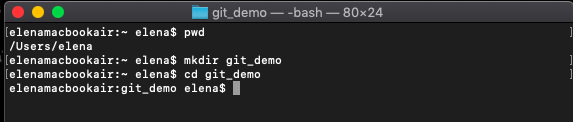
Sa sledecmo komandom kreiramo novi folder. Ovaj korak mozemo i uraditi rucno tj. Na desktop-u, ali cemo ovde raditi sve preko Terminal-a.

mkdir ime\_foldera

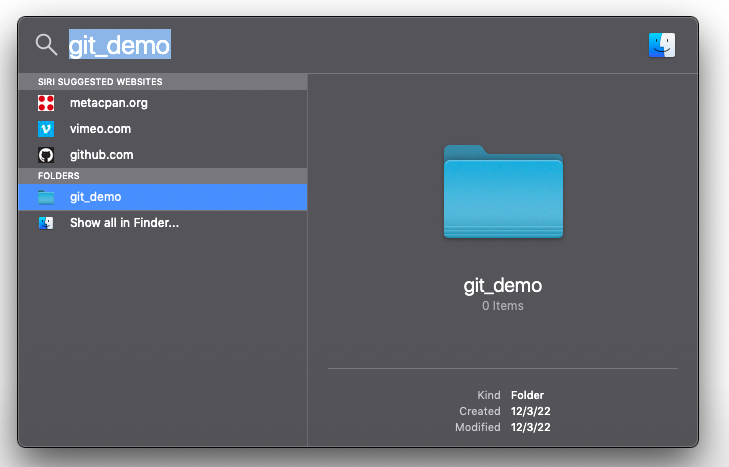


Sledecom komandom pristupamo folderu koji navedemo – u nasem slucaju je to novonastali folder.

cd ime\_foldera

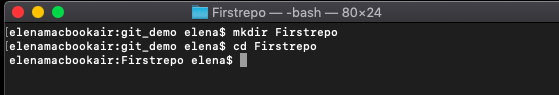


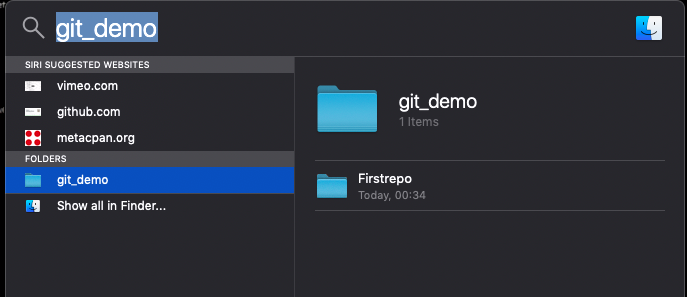
Nas folder mozemo proveriti preko Finder-a:



## Korak: Novi repozitorijum

Sve komande koje smo koristili za stvaranje foldera koristicemo i ovde:

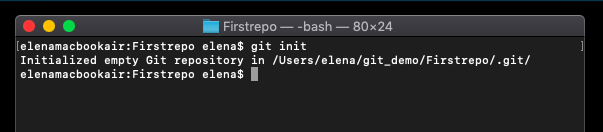




## Korak: Inicijalizacija repozitorijuma

Ovo postizemo komandom:

git init



## Korak : Ubacivanje fajla

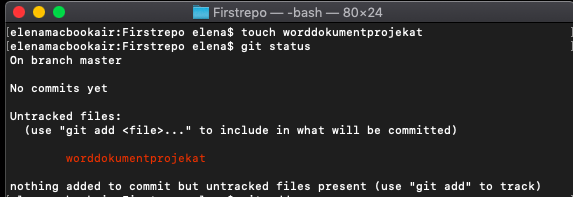
U ovom primeru koristicemo vec napravljen fajl, ali ukoliko zelimo preko Terminal-a da napravimo novi fajl, mozemo koristiti iste komande kao I kod foldera I repozitorijuma:

Sledecom komandom dolazimo do fajla:

touch ime\_fajla

Sledecom komandom mozemo proveriti status fajlova. (primetimo da je nas fajl crvene boje i da pokazuje time da nista nije dodato)

git status



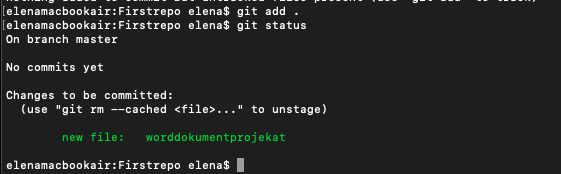
Sledecom komandom dodajemo fajl:

git add ime\_fajla

Ukoliko zelimo da ubacimo vise fajlova, da ne bismo kucali redom za svaki fajl ovu komandu I time mozda I izostavili nesto, koristimo sledecu komandu:

git add .

(primetimo da sada kada proverimo status, videcemo da je nas fajl zelene boje, tj. da postoji)



To mozemo proveriti i u Finder-u:

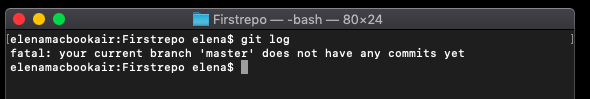


## Korak: Komitovanje promena

Ovo je jedan od najvaznijih koraka, zato sto ovim zapisujemo promene u project history i time olaksavamo ostalim programerima oko snalazenja u nasem kodu.

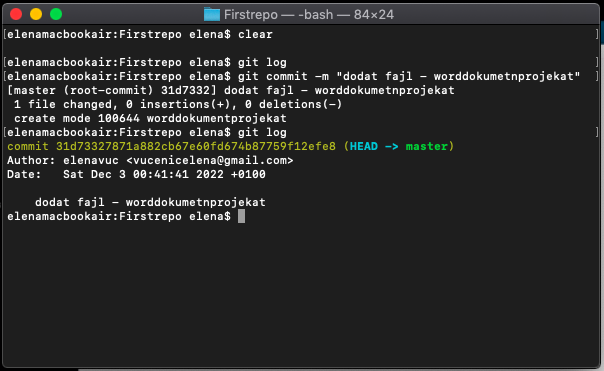
Sledecom komandom otvaramo prostor u kome imamo jasan pregled promena: (s obzirom da mi jos nista nismo dodavali ispisace da nema commit-a)

git log



Sledecom komandom komitujemo nas dodani fajl:

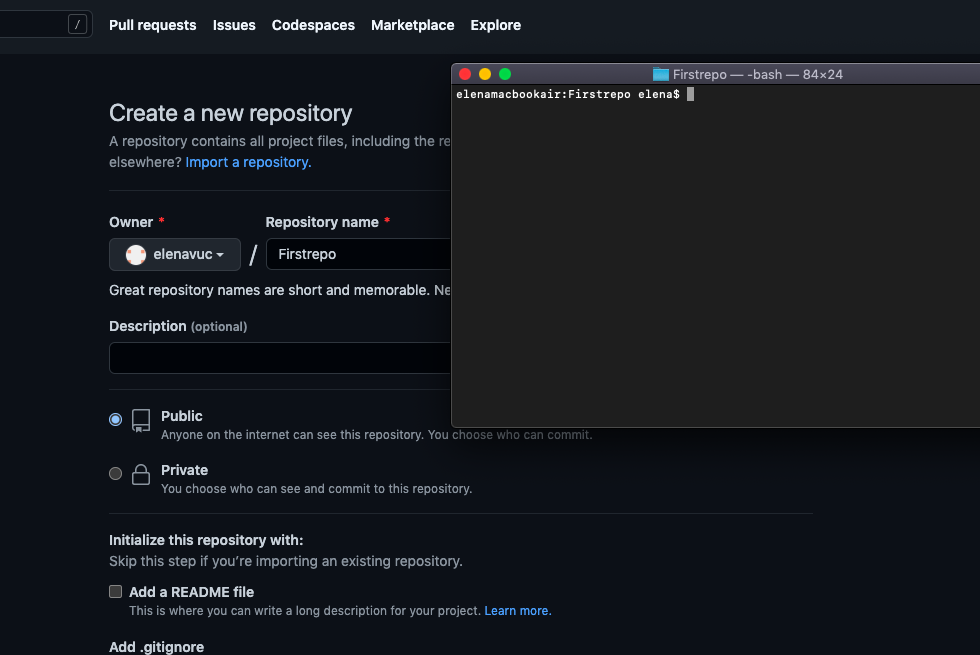
git commit -m “neka poruka”



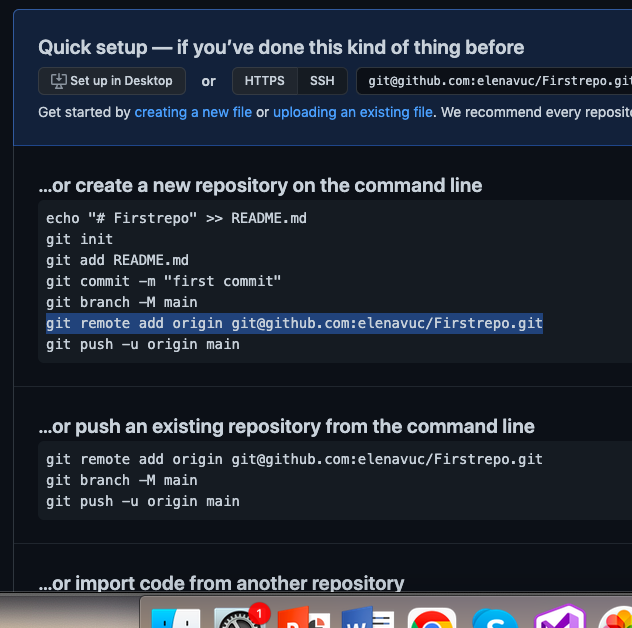
## Korak: Pushovanje fajlova

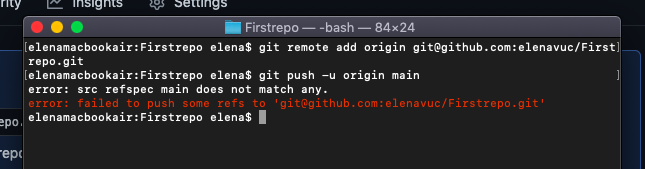
Sve ono što ste dodali u commit(stageovane promene) sada treba gurnuti na server. I tako završiti kompletan ciklus jednog commita.

Prvo pravimo novi repozitorijum na github-u:

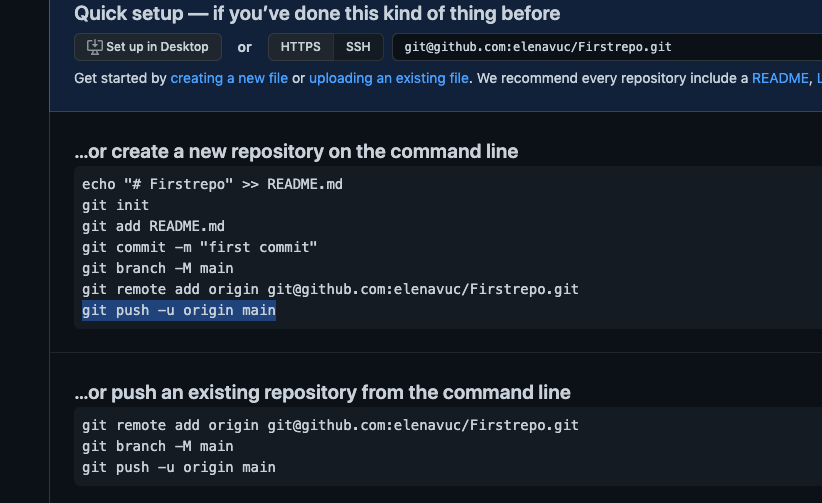


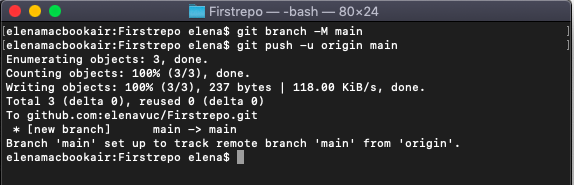
Zatim kopiramo ovu liniju dole selektovanu :



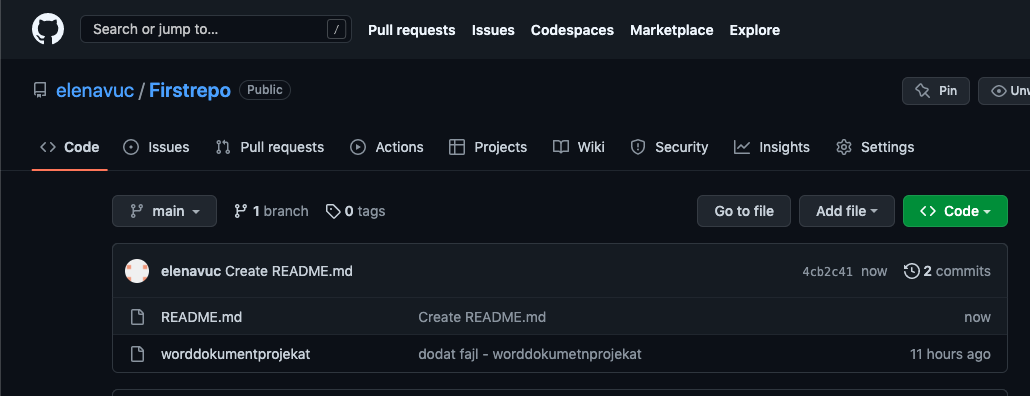


Zatim potrebno je da sledecom komandom definisemo granu gde cemo push-ovati sve promene, u ovom slucaju to je grana MAIN. Pa onda sve push-ujemo sledecom komandom na tu granu:





Kada proverimo na github-u, videcemo da su se svi push-ovani fajlovi pojavili na grani main:



# Korisni linkovi:

1. Ukoliko imate Windows onda pogledajte ovaj video za gore navedeni primer: <https://www.youtube.com/watch?v=QJ0iUNe27c8>
2. Link za github: <https://github.com/>
3. Link za instaliranje: <https://www.youtube.com/watch?v=RGOj5yH7evk&t=1513s>