

# Uvod u sisteme za kontrolu verzija datoteka i Git

---

## 1 UVOD

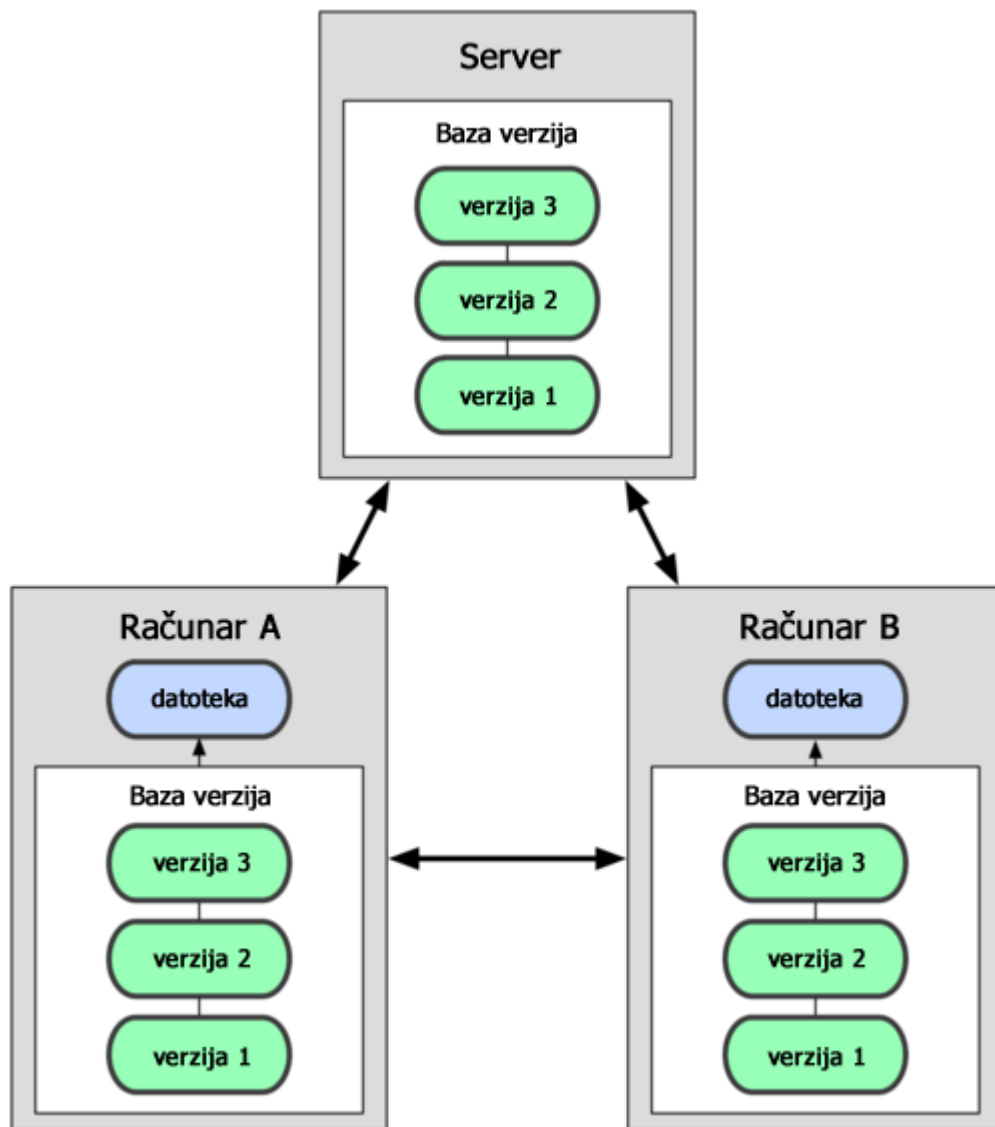
Sistemi za kontrolu verzija datoteka (eng. *Version Control Systems*, skraćeno: VCS) prate promjene napravljene na jednoj ili više datoteka, što omogućava vraćanje datoteka u neko od prijašnjih stanja, vraćanje cijelog projekta na staru verziju, pregled svih promjena urađenih nad datotekom itd. Isto tako, korištenje ovih sistema omogućava vraćanje slučajno obrisanih datoteka. Datoteke koje su pod kontrolom VCS-a su uglavnom izvorni kod, ali one mogu biti bilo kojeg tipa (slike, razni dokumenti ili dodatni resursi potrebni za projekat).

Sistemi za verzioniranje datoteka su prešli dug put u svom razvoju. Najjednostavniji oblik verzioniranja je ručno pravljenje kopija direktorija i označavanje direktorija vremenom pravljenja verzije i brojem verzije. Međutim, ovakav pristup je podložan greškama, i ideja koja se razvila je držanje verzija datoteka u bazi podataka. Kod lokalnih sistema za verzioniranje, na računaru na kojem se radi postoji baza podataka koja čuva historiju promjena. Međutim, i ovaj pristup se pokazao nedovoljno dobrim: nije moguć rad u timovima. Zbog toga su uvedeni centralizovani sistemi za verzioniranje koji su riješili problem timskog rada, obzirom da su se sve datoteke čuvale na udaljenom serveru, a svaki korisnik je sa servera povlačio samo verziju projekta na kojoj je trenutno radio. Ovaj pristup ima problem što se, u slučaju kvara servera (ili serverskog diska), cijeli projekat nepovratno gubi.

### **Distribuirani sistemi za kontrolu verzija**

Svi ovi problemi su riješeni pojavljivanjem distribuiranih sistema za verzioniranje, kao što je Git. U ovakvim sistemima, klijenti sa servera ne preuzimaju samo jednu verziju, već *kloniraju cijeli repozitorij*. U slučaju kvara servera, obzirom da se svi podaci nalazi na klijentskom računaru, dovoljno je kopirati lokalni repozitorij na udaljeni server i šteta je popravljena. Rad distribuiranih sistema je ilustrovan na slici

1.



Slika 1. Rad distribuiranih sistema za verzioniranje

Ukoliko se Git ispravno koristi, sve aktivnosti poput kreiranja, mijenjanja i brisanja datoteka se bilježe u Gitovu bazu podataka i praktično je nemoguće prouzrokovati nepovratni gubitak podataka.

## 2 Osnovne upute za rad sa Gitom

### Kreiranje repozitorija

1. Kreirati korisnički račun na <https://github.com/>
2. Poslati predmetnom asistentu mail na (ec15261@etf.unsa.ba) u kojem je potrebno napisati naziv tima, korisničko ime upotrebljeno pri kreiranju korisničkog računa svih članova tima i tema koja će se raditi na predmetu.
3. Predmetni asistent će dodati dato korisničko ime u organizaciju ooad-2015-2016 i kreirati repozitorij za tim na kojem će se raditi

U nastavku su opisani koraci koje je potrebno uraditi nakon što je repozitorij kreiran.

### Instalacija gita

Preuzeti instalaciju gita i instalirati (<http://git-scm.com/downloads>). Uz instalaciju Gita dolaze *Git Bash* (*shell* kroz koji ćemo raditi sa Gitom) i *Git GUI* (jedna varijanta rada sa Gitom kroz GUI, mada ima i drugih besplatnih alata koji izgledaju malo ljepše).

Otvoriti *Git Bash* tako da radni direktorij bude trenutni direktorij (može se napraviti desni klik na prazan prostor u *File exploreru* i izabrati *Git Bash*).

### Konfiguracija identiteta korisnika

Prije početka rada je potrebno napraviti par postavki Git okruženja. Ove postavke se rade samo jednom, ali ih je moguće naknadno promijeniti.

Konfiguracijom identiteta zapravo se daje identitet korisnika Gitu, što je važno jer Git koristi ove informacije prilikom pohranjivanja *commita* (nakon što je *commit* već pohranjen, nije moguće naknadno promijeniti ove informacije za taj *commit*).

**NAPOMENA:** Komande otkucati ručno u konzolu. Moguće je da se crtice i navodnici neće ispravno kopirati iz PDFa.

Pokrenuti Git Bash i ukucati:

```
$ git config --global user.name "Ime Prezime"
$ git config --global user.email imeprezime@etf.unsa.ba
```

Prosljeđivanjem parametra `--global`, Git uvijek koristi ove postavke (u slučaju prijave sa vlastitim korisničkim računom).

## Provjeravanje postavki

Da bi se provjerilo da li je konfiguracija ispravna, može se koristiti komanda `git config --list`.

```
git config --list
user.name=Ime Prezime
user.email=imeprezime@etf.unsa.ba
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

Za provjeru vrijednosti samo jednog od parametara (naprimjer ime ili email), može se koristiti komanda:

```
$ git config user.name
Ime Prezime

$ git config user.email
imeprezime@etf.unsa.ba
```

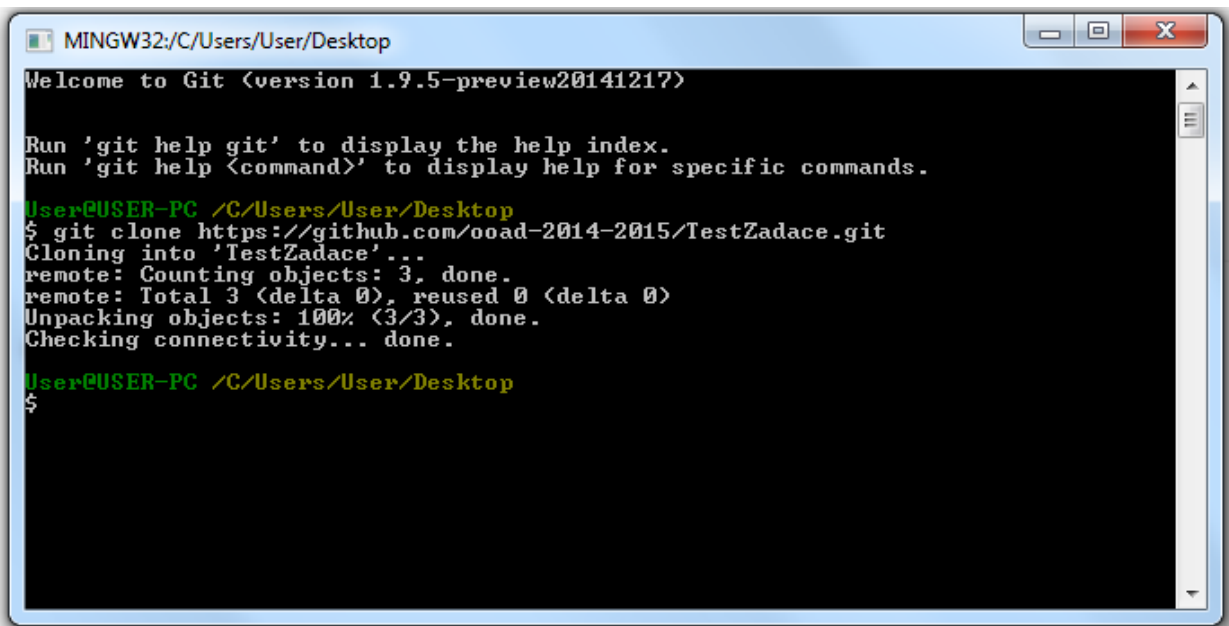
## Kloniranje repozitorija

Do Git repozitorija se može doći na dva načina. Prvi je da se projekat koji je već na računaru importuje na udaljeni Git server. Drugi način je da se projekat koji već postoji na udaljenom Git repozitoriju klonira na računar. Koristiti će se drugi pristup, jer je repozitorij već kreiran i potrebno ga je preuzeti na lokalni računar.

U nastavku teksta biće korišten link na repozitorij TestZadace (<https://github.com/ooad-2015-2016/TestZadace.git>). Tim će umjesto ovo linka koristiti link na vlastiti repozitorij.

Komanda za preuzimanje kopije Git repozitorija koji će se koristiti na predmetu je:

```
$ git clone https://github.com/ooad-2015-2016/TestZadace.git
```



```
MINGW32:/C:/Users/User/Desktop
Welcome to Git (version 1.9.5-preview20141217)

Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.



User@USER-PC /C:/Users/User/Desktop
$ git clone https://github.com/ood-2014-2015/TestZadace.git
Cloning into 'TestZadace'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.

User@USER-PC /C:/Users/User/Desktop
$
```

Slika 2. Kloniranje projekta

Ova komanda će kreirati direktorij TestZadace, inicijalizirati direktorij TestZadace /.git i preuzeti kopiju repozitorija. U linku je navedeno da se koristi HTTPS protokol za preuzimanje podataka, što će omogućiti rad sa repozitorijem korištenjem vlastitim korisničkim imenom i lozinkom.

Repozitorij koji je upravo kloniran je trenutno prazan. U njemu će biti potrebno kreirati direktorije u kojima će se raditi na zadaćama i projektu.

Name	Type	Size
 .git	File folder	
 README.md	MD File	1 KB

Slika 3. Početni kloniran sadržaj sa repozitorija

## Kreiranje direktorija i datoteka u repozitoriju

Prva datoteka koju je potrebno dodati u repozitorij je .gitignore, koja sadrži upute gitu koje datoteke ne treba slati na udaljeni repozitorij, odnosno datoteke koje treba ignorisati. Naprimjer, izvršne datoteke, privremene lokalne datoteke i slično. Ova datoteka se automatski kreira pri kreiranju repozitorija koristeći github stranicu.

Uputa: Otvoriti direktorij repozitorija, uraditi desni klik na radnu površinu i izaberati Git Bash. Ukoliko nema te opcije, onda pokrenuti program Git Bash i otvoriti direktorij repozitorija, kao u primjeru:

```
User@USER-PC /C/Users/User/Desktop
$ cd /C/Users/User/Desktop/TestZadace/

User@USER-PC /C/Users/User/Desktop/TestZadace (master)
$ _
```

Kreirati novu datoteku imena .gitignore:

```
User@USER-PC /C/Users/User/Desktop/TestZadace (master)
$ touch .gitignore
```

Datoteku kreirati na ovaj način a ne kroz file explorer, jer bi se Windows bunio što datoteka nema ime, nego samo ekstenziju.

Kreiranu datoteku otvoriti u nekom text editoru i kopirati u nju sadržaj sa linka:

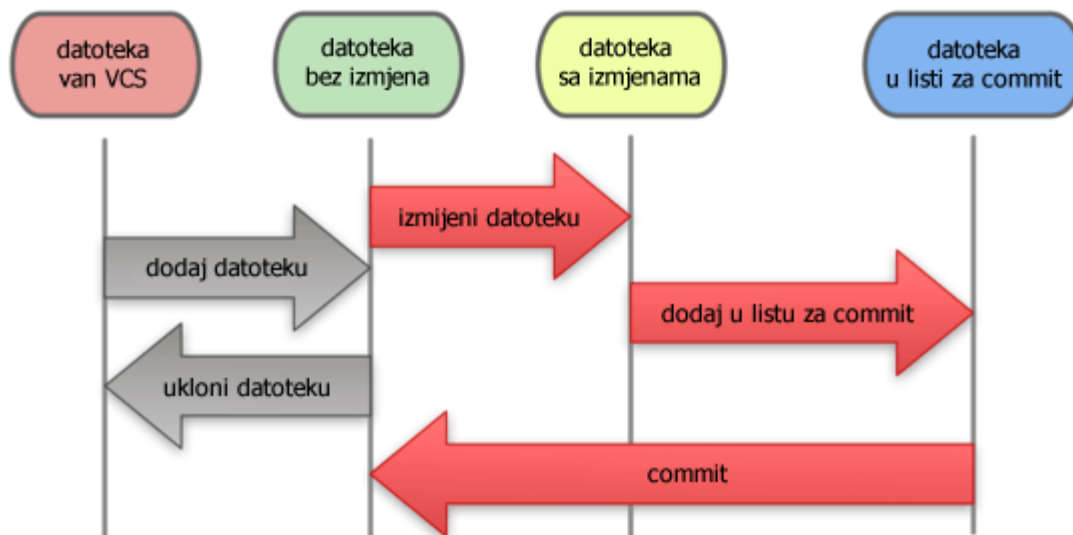
<https://raw.githubusercontent.com/github/gitignore/master/VisualStudio.gitignore>

Potrebno je da se za svaku zadaću kreira poseban direktorij.

Poštujte pravila za imenovanje direktorija koja će biti objavljena zajedno sa projektnim zadatkom. Konzistentna organizacija direktorija može uticati na bodove ostvarene u projektu. Svaki projektni zadatak će imati poseban naziv za datoteku.

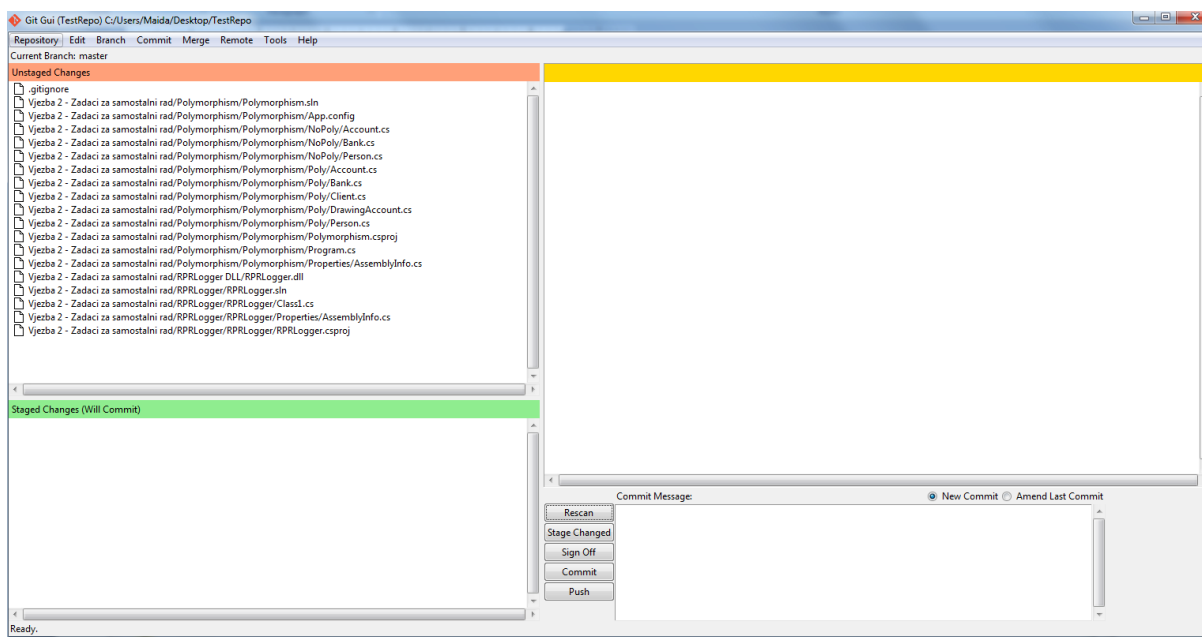
## Zapisivanje promjena na repozitorij

Svaka datoteka u radnom direktoriju može biti u jednom od dva stanja: van VCS (promjene na datoteci se ne prate) ili unutar VCS (prate se sve promjene na datoteci). Datoteke koje su unutar VCS mogu biti bez izmjena, sa izmjenama ili mogu biti u listi za *commit*. Kada se repozitorij tek klonira, sve datoteke će biti unutar VCS i bez izmjena. Pri radu na projektu, datoteke se mijenjaju. Nakon što se završi željena promjena, datoteku je potrebno dodati u listu za *commit* i onda uraditi *commit*, odnosno sačuvati promjene u Gitovu bazu. Životni ciklus statusa datoteke je prikazan na slici 3.



Slika 4: Životni ciklus statusa datoteke

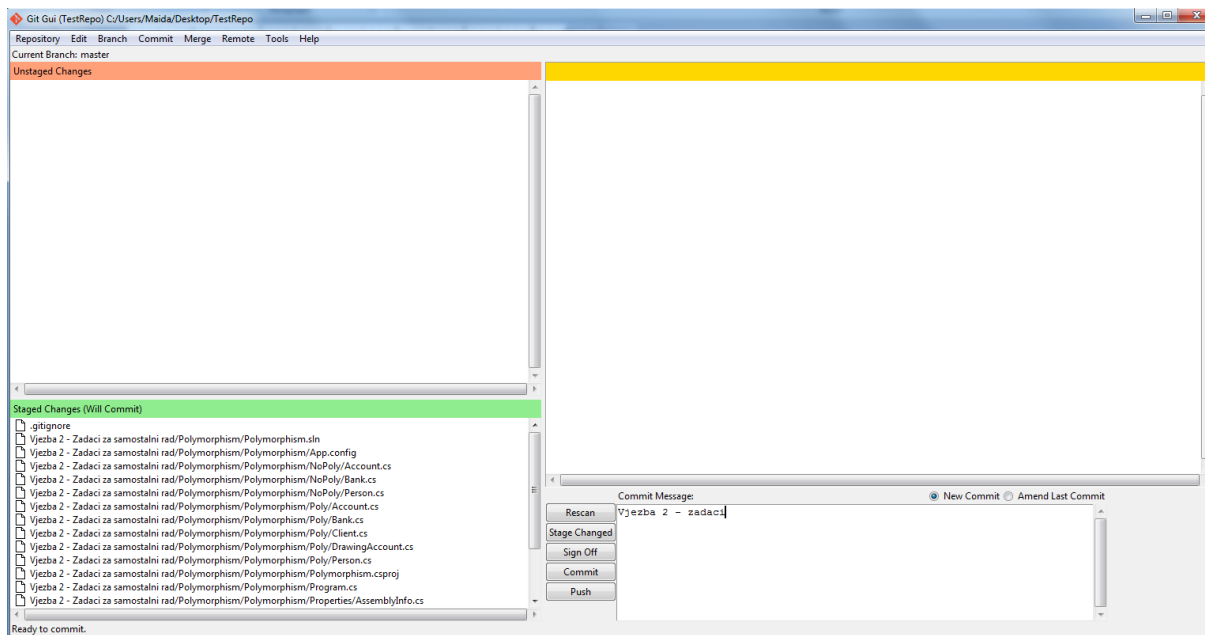
Pri radu je dozvoljeno koristiti grafički alat Git GUI koji je instaliran zajedno sa Git Bash. U direktoriju u kojem je kloniran repozitorij napraviti desni klik na prazan prostor i odaberati Git GUI, koji je prikazan na sljedećoj slici. U gornjem lijevom uglu je lista svih datoteka koje ili nisu uopšte ušle u VCS, ili su unutar VCS ali nisu uključene u listu datoteka koje će se naći u narednom *commitu*. U gornjem desnom uglu je pregled promjena napravljenih na datoteci.



Slika 5: Git GUI - promjene na datotekama koje nisu dodane u listu za *commit*

Direktoriji koji su prazni neće biti dodani u listu za commit. Na udaljeni repozitorij se šalju samo direktoriji koji sadrže barem jednu datoteku.

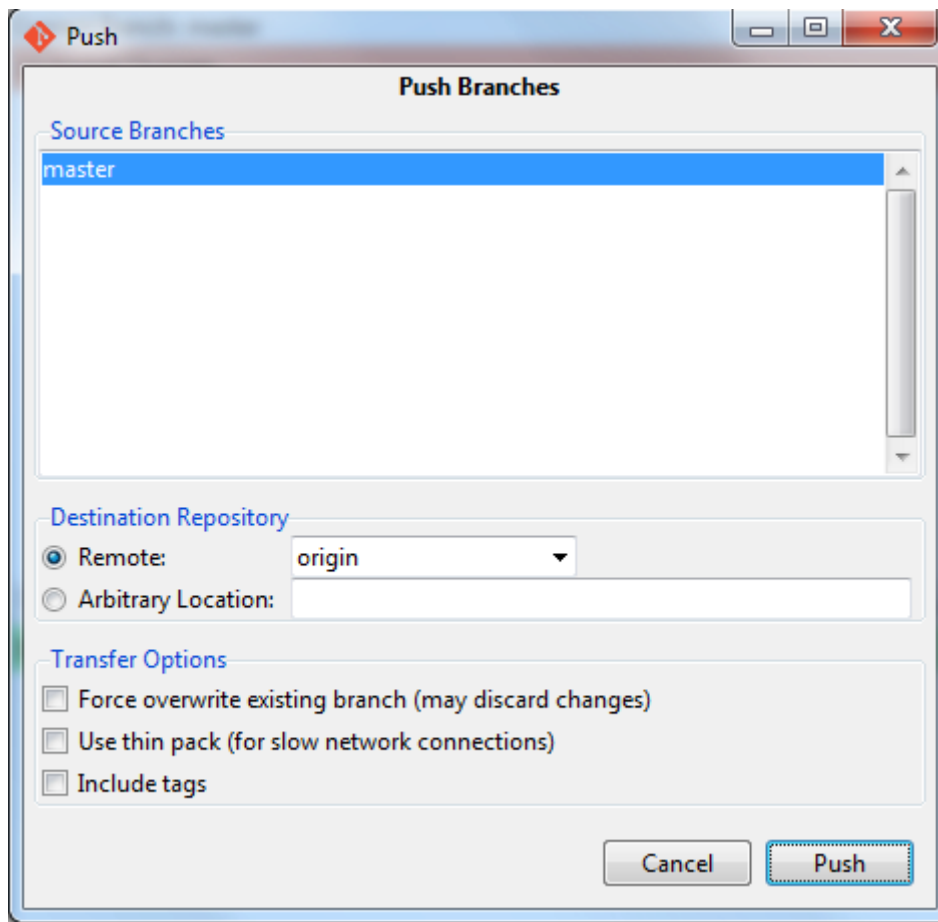
Označiti datoteke koje želite sačuvati i u meniju odaberite *Commit -> Stage to Commit*. Time ste te datoteke dodali u listu datoteka koje će biti sačuvane sljedećim *commitom*.



Slika 6: Git GUI - promjene na datotekama su dodane u listu za *commit*

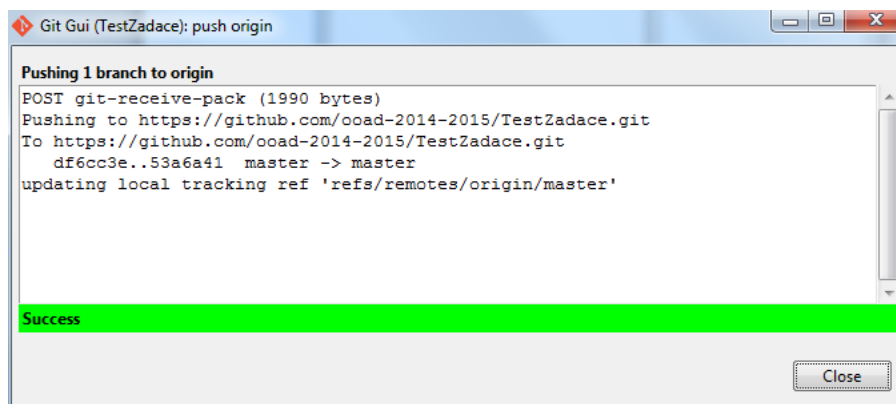
U donjem desnom uglu potrebno je napisati poruku koja objašnjava šta je to što ovim *commitom* dodajete ili koje su to promjene koje ste napravili. Nakon toga možete uraditi *commit*. Nakon što je *commit* završen, potrebno je te promjene sačuvati i na udaljenom Git repozitoriju. To ćete uraditi odabirom opcije *Push*.





Slika 7: Git GUI - bilježenje promjena na udaljeni repozitorij

Moraćete unijeti prvo korisničko ime, pa lozinku, da biste poslali datoteke na udaljeni repozitorij.



Slika 8: Slanje podataka na udaljeni repozitorij.

Potrebni su korisničko ime i lozinka, da bise provjerilo da li korisnik ima pravo da mijenja datoteke na repozitoriju.

**Svaki put kada uradite neku bitnu promjenu ili završite neki dio koda, uradite *commit - push*. Na ovaj način vam se nikad neće desiti da nepovratno izgubite neku datoteku ili dio koda.**

Često će se desiti da dva člana tima povuku istu verziju sa repozitorija i u isto vrijeme prave izmjene. Nakon što prvi član tima povuče najnoviju verziju, drugi član tima u međuvremenu je već napravio push na repozitoriju. Prvi član nema izmjene koje je napravio drugi član tima. Tada će pri obavljanju njegove push operacije iskočiti upozorenje kao na slici 9. Ovo upozorenje znači da član tima treba prvo da povuče izmjene drugog člana tima koristeći fetch (Remote - Fetch From - Origin) a zatim spojiti svoju verziju sa tom verzijom koristeći merge. U slučaju da članovi tima nisu radili izmjene nad istim datotekama, merge će se obaviti bez problema.

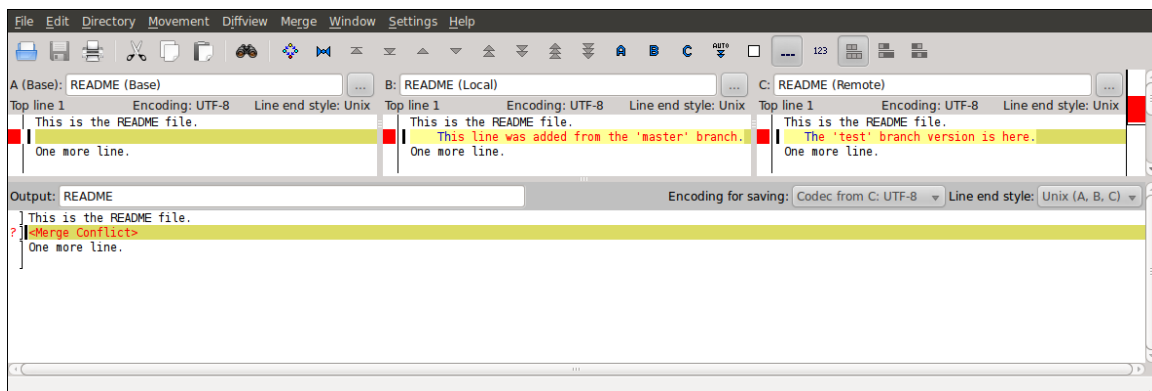


Slika 9: Nije urađjen pull (fetch + merge) prije commit

Ako je ista datoteka modifikovana u obje verzije tada će merge operacija pokušati obje izmjene primjeniti nad istom datotekom. Ako je datoteka modifikovana u istim linijama koda tada će doći do konflikta (Slika 10). Konflikt je potrebno razrješiti (resolve) tako što se bira za svaki blok koda, čije izmjene je potrebno prihvatiti. Za svaki blok koda koji se preklapa može se uzeti lokalna verzija (use mine) ili remote verzija (use other) tog bloka koda. Konflikti se mogu razrješiti ručno, ili koristeći gui alate poput kdiff ili meld (preporučuje se korištenje gui alata).



Slika 10: Konflikt poruka pri merge operaciji



Slika 11: Rješavanje konflikta koristeći kdiff

Savjeti za pravilno korištenje git repozitorija pri razvoju projekta u timu:

- **Nikad ne raditi push koda koji se ne može kompajlirati**
- Po mogućnosti ne raditi push koda koji je pokvario određenu funkcionalnost
- **Push treba raditi često.** Čim je određena aktivnost završena (Naprimjer kreirana klasa student sa osnovnim properties, ili popravljen određeni bug), odmah uraditi push.
- **Napisati u commit poruci sve što je bitno za taj commit.** Naprimjer: “Dodana klasa student i popravljen bug u formi OcjeneStudenta koji je blokirao resize forme. Izmjena je

prouzrokovala da se sada forma IndexStudenta ne može otvoriti”. Na ovaj način se pri kasnijem traženju kako je nastao bug može tačno naći koje su linije izmjenjene da se lakše može pronaći uzrok buga. Često su bugovi zabilježeni u sistema za praćenje bugova (poput Jira i drugih) gdje svaki bug ima svoj detaljni opis i ID. Tada se u commit poruci stavi samo “Popravljen bug #id\_buga\_forme\_Ocjena + link\_na\_bug\_u\_sistemu, prouzrokovan bug #21432 + link\_na\_bug\_u\_sistemu” i onda se lako može doći do opisa buga.

- Raspodijeliti odgovornosti članova tima za određene klase tako da se što više izbjegne nastajanje konflikta između verzija.
- Ne praviti projekte V1, V2 pa ih odvojeno commitati. Sve je to jedan projekat koji ima različite verzije, za šta i služi sistem za verzioniranje.

## Grananje

Gotovo svaki sistem za verzioniranje ima podršku za grananje (eng. *branching*). Kreiranjem grane za rad, odvaja se od glavne linije razvoja projekta, što omogućava pravljenje promjena koje neće utjecati na glavnu verziju projekta. Projekat uvijek mora imati jednu granu koja sadrži kod koji je potpuno ispravan i spreman je za produkciju. Na većim projektima, sav razvoj, popravljavanje *bugova* ili dodavanje novih funkcionalnosti se radi na odvojenoj grani.

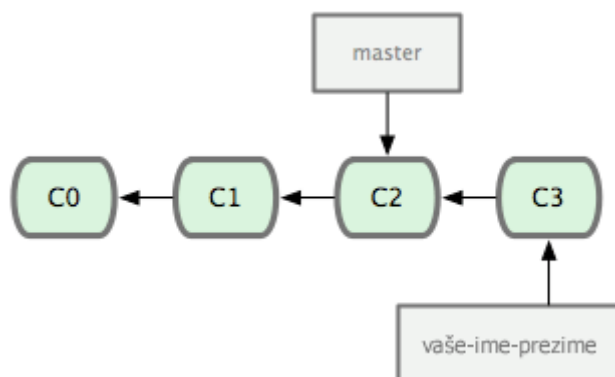
Da se kreira novu grana i istovremeno se prebaci na nju, vrši se komanda:

```
$ git checkout -b nova-grana
```

Nakon izvršenja, dobije se poruka:

```
$ git checkout -b nova-grana
```

Na ovaj način se odvaja od glavne grane (*master*) i nastavlja rad na grani koja je upravo kreirana. Grana *master* ostaje netaknuta u daljem radu (slika 2).



Slika 12: Rad na odvojenoj grani nema utjecaj na resurse na glavnoj grani

Kada je popravljjanje *buga* završeno, vraća se na granu `master`, i spajaju se promjene napravljene na grani `nova-grana` sa kodom u grani `master`.

```
$ git checkout master
```

```
$ git merge nova-grana
```