|  |
| --- |
| FESB, SVEUČILIŠTE U SPLITU |
| Git |
| Projektiranje Informacijskih sustava |
|  |
|  |
| **23.3.2018.** |

Josip Zelić

|  |
| --- |
| PROJEKTIRANJE INFORMACIJSKIH SUSTAVA – VJEŽBA 1 |

## Sustavi za upravljanje verzijama (*Version Control)*

Sustav za upravljanje verzijama prati promjene datoteka u nekom vremenskom periodu, pri čemu je moguće, u bilo kojem trenutku pregledavati bilo koju verziju datoteke. U ovakvim sustavima, najčešće se prate verzije izvornog koda programa, ali je moguće pratiti i verzije bilo kojih drugih datoteka. Sustavi za upravljanje verzijama omogućavaju jednostavno vraćanje sadržaja datoteke na neku prethodnu verziju, praćenje promjena tijekom vremena, kao i uvid o osobi koja je napravila zadnju promjenu. Jedna od bitnijih mogućnosti je i vraćanje izbrisanih/izgubljenih datoteka.

### Centralizirani sustavi za upravljanje verzijama

Današnji projekti se većinom razvijaju u projektnim timovima što dovodi do potrebe za usklađivanjem projekta između članova projektnog tima. Da bi se taj problem riješio, došlo je do razvoja centraliziranih sustava za upravljanje verzijama (npr. CVS, SVN) kod kojih obično postoji jedan, centralni poslužitelj na kojem se čuvaju svi verzije projektnih datoteka. Članovi tima, na taj poslužitelj šalju sve promjene projektnih datoteka koje naprave, ali i dohvaćaju najnovije verzije datoteka koje su poslali ostali članovi tima.

Jedan od problema centraliziranih sustava za upravljanje verzijama je što ukoliko dođe do pada glavnog poslužitelja, članovi projektnog tima ne mogu koristiti sustav, bilo za slanje svojih promjena datoteka, bilo za dohvaćanje najnovijih verzija datoteka.

### Distribuirani sustavi za upravljanje verzijama

Kod distribuiranih sustava za upravljanje verzijama (npr. Git, Mercurial, Bazaar), klijenti ne skidaju samo zadnju verziju datoteka, nego potpuni projekt. U slučaju da dođe do pada glavnog servera, bilo koji klijentski projekt se može iskoristiti za vraćanje podataka.

U današnjoj vježbi, bavit ćemo se osnovnim operacijama **Git** distribuiranog sustava za upravljanje verzijama.

## Git – Osnovni koncepti

**Repozitorij** – analogno projektu, sadrži sve datoteke vezane za projekt (uključivo i dokumentaciju), kao i potpunu povijest verzija za svaku datoteku. Na repozitoriju može surađivati više suradnika. Repozitorij može biti javan (svi ga vide i svi mogu čitati kod, ali ne i pisati) i privatan (i čitanje i pisanje je dopušteno samo za članove time).

**Clone** (**„Klon“**) – klon je kopija repozitorija koja se nalazi na lokalnom računalu, a kloniranje je proces izrade te kopije. Na lokalnom klonu, moguće je raditi izmjene i koristiti Git za praćenje tih izmjena, bez potrebe da se kontaktira središnji poslužitelj. U bilo kojem trenutku, ukoliko postoji veza prema poslužitelju, moguće je napraviti sinkronizaciju između lokalnog klona i verzije na središnjem poslužitelju.

**Commit** ili revizija, je pojedinačna promjena datoteke (ili skupa datoteka). Ima značenje spremanja datoteke unutar sustava verzija (slično kao što se datoteka može spremiti na disk). Prilikom svakog spremanja, čuvaju se informacije o točnim promjenama kao i informacije o osobi koja ih je napravila. Uz svako spremanje mora se dodati i opis napravljenih promjena.

**Diff** (**„Razlika“**) **–** usporedba promjena između dvaju *commitova*; vizualno pokazuje promjene datoteke između dva commita.

**Push** (**„pošalji“**) – odnosi se na spremanje commitanih promjena na središnji repozitorij.

**Branch** (**„Grana“**)– paralelna verzija repozitorija. Sadržana je unutar repozitorija, ali nema utjecaja na glavnu granu, na taj način dopuštajući rad bez utjecanja na glavnu verziju. Cilj grananja je razvoj dodatnih funkcionalnosti ili testiranje ideja pri čemu se imaju sve prednosti repozitorija (spremanje novih revizija, praćenje razlika i sl). Ako su napravljene promjene prihvatljive grana se spaja („merge“) u glavni repozitorij (pri čemu je potrebno riješiti sve konflikte do kojih može doći), a ako se zaključi da testirana ideja ne funkcionira, grana se jednostavno miče, bez ikakvog utjecaja na glavni repozitorij. Na taj način je moguće da članovi tima rade na različitim funkcionalnostima aplikacije, bez velikog utjecaja jednih na druge.

**Merge** (**„spajanje“**) – spajanje uzima promjene iz grane ili račvanog projekta i spaja ih u neku drugu granu. Spajanje može biti obavljeno automatski, ukoliko nema konflikata između datoteka ili ručno.

**Fetch („dohvati“)** – odnosi se na dohvaćanje zadnje verzije repozitorija sa središnjeg poslužitelja, pri čemu ne dolazi do spajanja ukoliko je, u međuvremenu, došlo do promjena određenih datoteka.

**Pull („povuci“)** – proces dohvaćanja i spajanja promjena sa središnjeg repozitorija.

**Blame(„Okrivljivanje“) –** dohvaća informacije o tome tko je zadnji napravio modifikaciju određene linije i kad (u sklopu koje revizije). Na taj način moguće je pronaći tko je dodao određenu funkcionalnost i napravio promjene koje su dovele do neke pogreške.

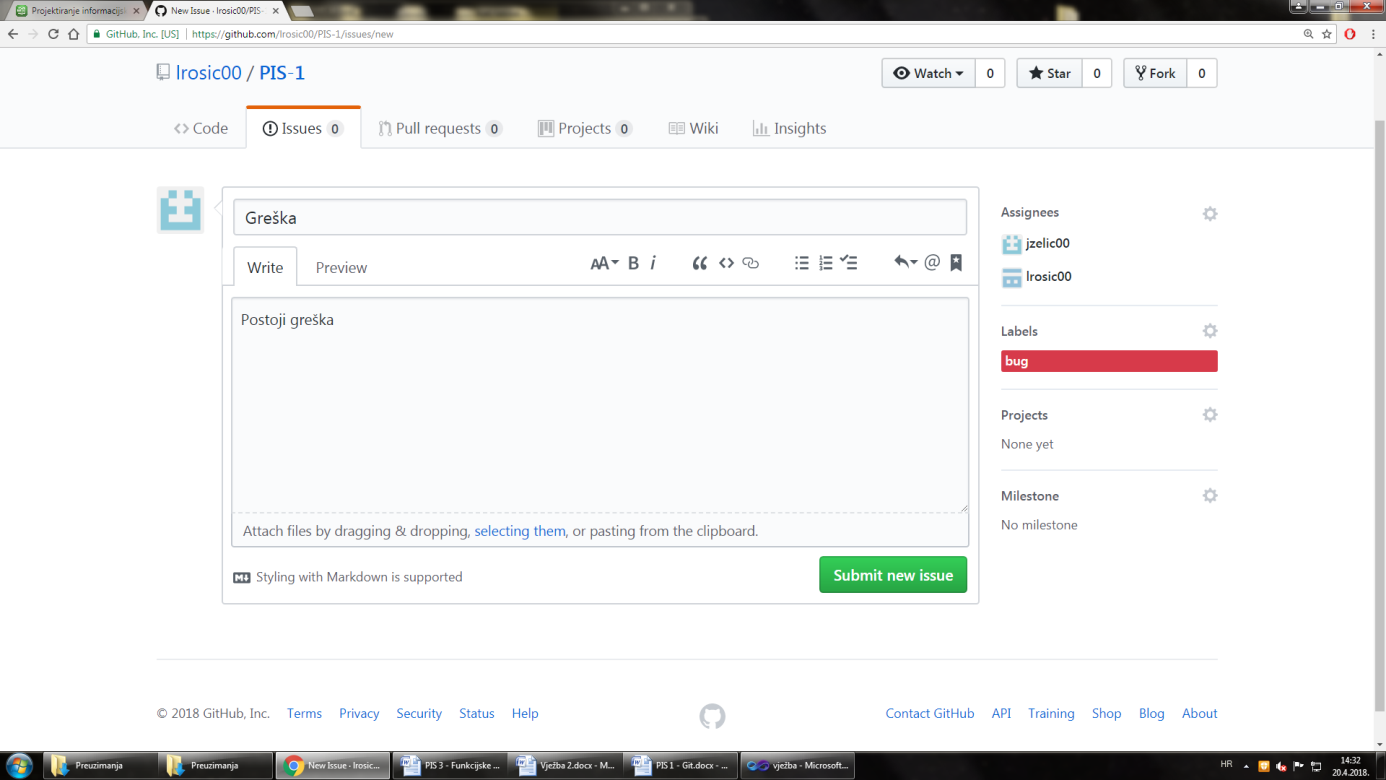
**Fork** (**„Račvanje“**) **–** račvanje je osobna kopija repozitorija drugog korisnika. Moguće je slobodno raditi bilo kakve promjene na račvanom projektu, bez ikakvog utjecaja na original. Račvani projekt je povezan s originalnim projektom i moguće je zatražiti nove verzije datoteka iz originalnog projekta. Također, moguće je autoru originalnog projekta poslati prijedlog za povlačenje račvanog projekta i na taj način spojiti račvani projekt i originalni projekt. Obično je namijenjeno za programere izvan tima.

**Issue** – je predloženo poboljšanje, zadatak ili upit vezan za repozitorij. Kod javnih repozitorija, bilo tko ih može stvoriti. Oko svakog pitanja je moguće voditi raspravu i svako pitanje može biti pridijeljeno nekom od korisnika repozitorija.

## Rad s GIT-om

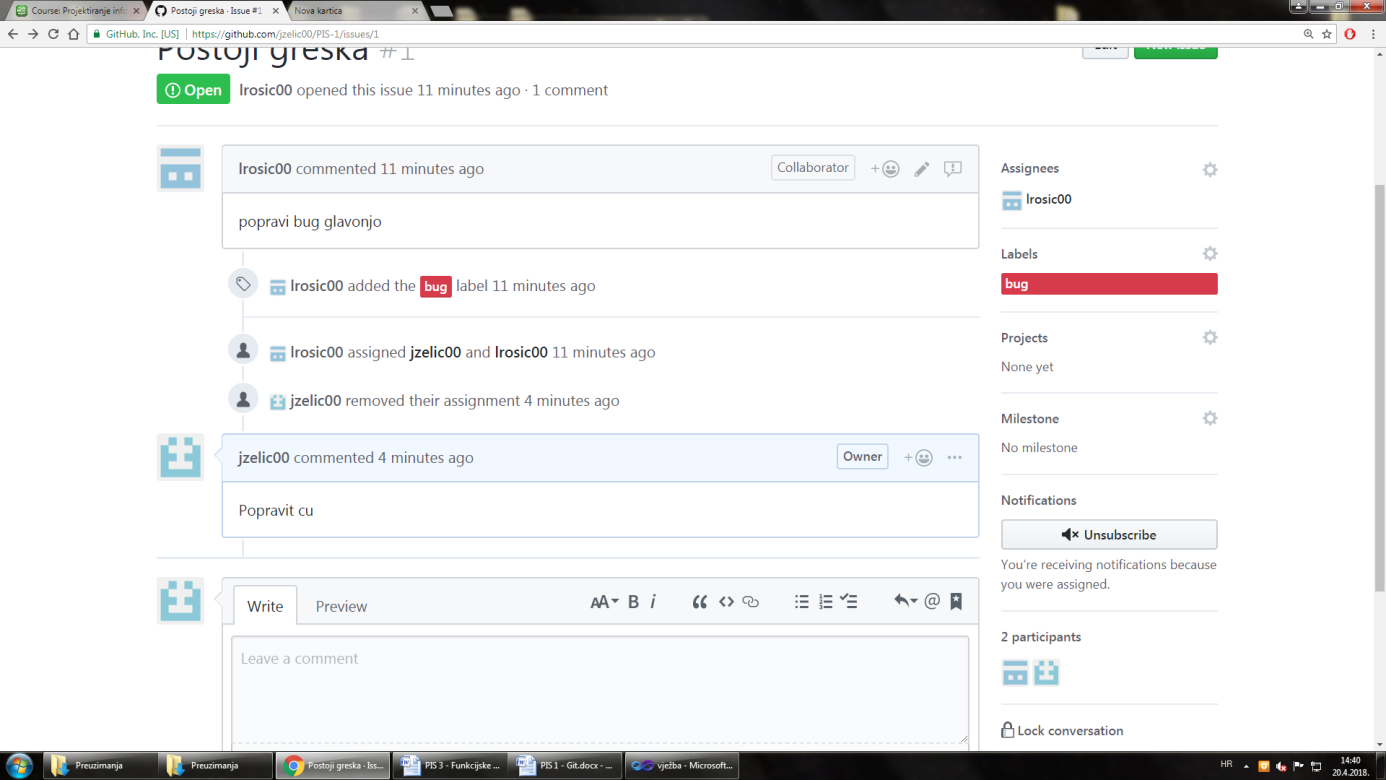
Obaviti sljedeće korake:

1. **Registracija na github-u:** Otvoriti github stranicu (<https://github.com/>), upisati željeno korisničko ime, e-mail adresu i šifru te kliknuti „Sign up“. U idućem koraku odabrati besplatnu opciju i kliknuti „Finish Sign up“.
2. **Stvaranje repozitorija:** Nakon registracije otvara se nova stranica na kojoj je u donjem desnom kutu moguće kliknuti „New repository“. Na sljedećoj stranici unijeti ime repozitorija (npr: PIS-1) i opis projekta. Kliknuti „Create repository“ (ne uključivati ništa drugo!)
3. Skinuti i instalirati GIT za windows: <http://git-scm.com/download/win>
4. Instalirati GIT plugin za Visual studio:
   1. Otvoriti Visual Studio
   2. Tools -> Extension Manager (ili Extension and Updates) -> Online gallery (ili Online)
   3. Pronaći „Git Tools“ i instalirati (za Visual Studio 2015 i više), za Visual Studio 2010 pronaći i instalirati „Git Source control provider“
   4. Resetirati Visual Studio
5. Povezati VS i Git za Windows
   1. Tools->Options->Source Control
   2. Plugin Selection: Odabrati Git. Samo za plugin „Git Source control provider“ napraviti i sljedeće korake
   3. Git Source Control Provider Options -> Postaviti put do Git-a (vjerojatno u „C:\Users\student\AppData\Local\Programs\Git\bin\git.exe“)
   4. Odabrati „Use Visual Studio Diff Window“
6. Stvoriti Visual Studio C# console projekt
7. Postaviti repozitorij za projekt
   1. Desnim klikom na Solution odabrati Add Solution to Source Control za plugin„Git Tools“ , a za plugin „Git Source control provider“ desni klik na projekt: „Create Git repository“, ukoliko nemate opciju „Create Git repository“, onda otvorite Git bash i pozicionirajte se u direktorij vašeg Solutiona i pozovite naredbu git init sami.
   2. Iz izbornika GitTools odabrati -> Git bash , a za plugin „Git Source control provider“ desni klik na projekt -> Git -> Git bash
   3. U konzoli koja se otvori (Ako se konzola ne otvara ili se otvori, pa odmah zatvori, otvorite konzolu direktno preko Program Files -> Git Bash i pozicionirajte se u direktorij vašeg C# console projekta):
      1. git config user.name "username" (upisati svoj username)
      2. git config user.email ime@fesb.hr (upisati svoj mail)
      3. git add . (dodate direktorij projekta u git index)
      4. git commit –m "First commit"(napravi novu verziju s opisom)
      5. git remote add origin url\_repozitorija\_sa\_githuba (poveže lokalni repozitorij sa github repozitorijem; url repozitorija je formata: <https://github.com/username/ime_repozitorija.git> ili samo copy/past iz GitHub kreiranog repozitorija)
      6. git push –u origin master (pošalje zadnju verziju lokalnog repozitorija na centralni github server, ukoliko vam se javlja greška provjeriti u Control Panel->Credential Manager (Upravitelj vjerodajnica) da li je spremljena nečija tuđa vjerodajnica za github pa je izbrišite i ponovo pozovite naredbu za push)
   4. Provjeriti github repozitorij i vidjeti da li se slaže s lokalnom podatkovnom strukturom
8. Modifikacija projekta
   1. Otvoriti Program.cs u Visual Studiu i u Main metodi dodati Console.WriteLine("Hello World");
   2. Usporediti lokalnu i centralnu verziju git fetch pa git diff u konzoli na vašem računalu
   3. Dodati promjenu u područje postavljanja (staging area, index) s git add vašpodirektorij/Program.cs (vi ste u root direktoriju projekta pa treba i put do datoteke)
   4. Napraviti novi git commit (otvori se editor povezan s git-om s komentarom, odkomentirajte liniju (izbrišite #) u kojoj piše sama izmjena te spremite promjenu, te zatvorite editor da se proces nastavi, ukoliko se radi o VI editoru (konzolnom tekstualnom editoru) briše se tipkom delete, nakon toga tipka escape i naredba :wq! za izlazak iz editora)
   5. Napraviti git push.
   6. Pogledati github stranicu da li je prošlo uspješno
9. Suradnja
   1. Dodati novog suradnika na githubu: Na github stranici repozitorija, u gornjem, desnom kutu kliknuti na „+“ ikonu i odabrati „New collaborator“. Dodajte korisničko ime osobe koja sjedi do vas. Ta osoba bi trebala dobiti mail s poveznicom na projekt.
   2. Modifikacija projekta čiji ste suradnik: nakon primanja maila, u githubu otvorite projekt i koristeći web sučelje, napravite neku modifikaciju Program.cs datoteke (klikom na Edit), upišite neki komentar i pri dnu stranice opišite napravljenu promjenu.
   3. Napraviti git fetch i pogledati razliku između lokalnog i centralnog repozitorija (git diff master origin/master)
   4. Napraviti git pull i dohvatiti zadnju verziju sa centralnog repozitorija
   5. Otvoriti github web stranicu, datoteku Program.cs. Kliknuti na „Blame“ dugme i pogledati tko je što modificirao.
10. Dodatne modifikacije projekta:
    1. U Visual studiu dodati novu klasu: „NewClass“. Dodati je u repozitorij s naredbama:
    2. git add vašpodirektorij/NewClass.cs
    3. git commit –m „Added new class“
    4. git push
    5. Provjeriti na githubu da li je promjena uspješno sinkronizirana.
11. Konflikti
    1. Napraviti promjenu na svom projektu – dodati praznu metodu u NewClass
    2. git add vašpodirektorij/NewClass.cs
    3. Napraviti git commit –m „Dodana metoda“ (ALI NE NAPRAVITI git push!)
    4. Otići preko github web stranice do projekta na kojem surađujete i napraviti neku drugu promjenu u NewClass.cs. Spremiti promjenu i napisati opis promjene
    5. Lokalno napraviti:
       1. git fetch
       2. git diff master origin/master
       3. git pull (git će javiti da je došlo do konflikta)
       4. Otvoriti file u visual studiu i popraviti konflikt.
       5. Napraviti commit: git commit –am „Merged“
       6. git push
12. Zadaci:
    1. **Spajanje na projekt na kojem surađujete**: preko githuba se spojiti na projekt na kojeg ste dodani kao suradnik i dodati jedan „issue“ (na desnoj strani stranice Issues -> New Issue), u kojem će te napisati da postoji neka greška i s desne strane označite da se radi o bug-u:



Slika screenshota browsera s dodanim „issue“

* 1. **Dodjela buga članu projektnog tima:** Spojiti se na svoj github projekt, pregledati dodani issue i dodijeliti ga osobi koja ga je prijavila (s desne strane - „Assignee“).



Slika screenshota browsera s dodjeljenim „issuem“

* 1. **Ispravka bug-a na projektu na kojem surađujete**: preko github sučelja, spojiti se na projekt i napraviti neku ispravku. Spojiti se na issue i napisati da je bug ispravljen i kliknuti „Comment and Close“.