Udaljeni repozitoriji

Za rad sa udaljenim repozitorijima prvo će nam trebati račun na nekoj od web platformi za upravljanje verzijama softvera kao što su GitHub, Bitbucket ili GitLab. Za primjere će biti korišten GitHub.

Stvaranje novog repozitorija

Prvo ćemo napraviti novi repozitorij na GitHub-u. U gornjem lijevom rubu ekrana se nalazi zelena tipka sa natpisom "*New*". Pritiskom na nju se otvaraju opcije za podešavanje novog repozitorija:

- Upišite ime svog repozitorija
- Ostavite opciju "Public" kako bi bio javno dostupan
- Odaberite "*Create repository*" bez uključivanja bilo kakvih dodatnih opcija

Nakon toga će vam se prikazati ekran sa informacijama o upravo stvorenom repozitoriju. Trenutno nam je najvažnija adresa repozitorija koja će biti zapisana u obliku `https://github.com/<username>/<ime_repozitorija>.git`.

Povezivanje sa lokalnim repozitorijem

Sada se možemo vratiti u VS Code i povezati naš lokalni repozitorij sa udaljenim repozitorijem kojeg smo stvorili na GitHub stranici. Ponovno ćemo koristiti ugrađeni terminal i upisati naredbu:

```
git remote add origin <adresa>
```

Pri čemu se "`<adresa>`" odnosi na adresu udaljenog repozitorija koju smo napravili u prethodnom koraku. Pozivanjem ove naredbe, lokalnom repozitoriju smo dodali novi udaljeni (*remote*) repozitorij i pridružili mu oznaku "**`origin`**". Dodavanjem oznake ne moramo svaki puta upisivati adresu udaljenog repozitorija već nam je dovoljno zapamtiti njegovo "ime". Osim toga, lokalnom repozitoriju možemo dodijeliti i više udaljenih repozitorija pa je i zbog toga jednostavnije sa njima upravljati kada imaju svoje oznake. Popis pridruženih udaljenih repozitorija (i njihovih adresa) možemo vidjeti sa naredbom:

```
git remote -v
```

Slanje promjena na udaljeni repozitorij

Jednom kada imamo pridruženi udaljeni repozitorij možemo ga početi usklađivati sa lokalnim. Za usklađivanje repozitorija se koriste dvije naredbe - `pull` i `push`. S obzirom da nam je udaljeni repozitorij trenutno prazan, prvo što ćemo napraviti je poslati promjene sa lokalnog repozitorija na udaljeni koristeći naredbu `git push`.

Slanje na udaljeni repozitorij možemo napraviti na sljedeći način:

```
git push -u origin master
```

Pri čemu je potrebno obratiti pozornost na nekoliko detalja (pri čemu je lakše krenuti od kraja):

- Zadnji argument je ime lokalne grane koju želimo poslati na udaljeni repozitorij, u ovom primjeru je to grana *master* (pripazite na ime svoje grane, može biti *main*)
- Prije imena lokalne grane potrebno je navesti udaljeni repozitorij na koji šaljemo promjene. U ovom primjeru to je repozitorij pod imenom "*origin*" kojeg smo stvorili i povezali u prethodnim koracima
- U pozivu naredbe je iskorištena opcija "`-u`" koja stvara vezu (*upstream*) između naše grane i odabranog udaljenog repozitorija. To u praksi znači da pri budućem korištenju *push* (ali i *pull*) naredbe nije potrebno navoditi ime udaljenog repozitorija niti ime lokalne grane već će se ove upisane vrijednosti koristiti kao zadani parametri.

Ako sada osvježimo prozor u pregledniku u kojem nam je bio prikazan repozitorij, vidjeti ćemo kako sada sadrži datoteke koje smo napravili lokalno. Osim samih datoteka, možete vidjeti da je kopirana i čitava povijest naših *commit*-a. U desnom kutu zaglavlja tablice za prikaz datoteka se nalazi brojač *commit*-a koji ujedno služi i kao poveznica za njihov detaljni pregled.

Dohvaćanje promjena sa udaljenog repozitorija

U slučajevima kada više ljudi radi na istom projektu ili kada radimo na različitim računalima potrebno je prije početka rada dohvatiti eventualne promjene sa udaljenog repozitorija kako bi bili sigurni da nastavljamo sa razvojem najnovije verzije našeg projekta.

Ako već imamo povezani *upstream* iz trenutne grane prema udaljenom repozitoriju, sve što moramo napraviti je pozvati naredbu:

```
git pull
```

Nakon pokretanja naredbe sa udaljenog repozitorija će se dohvatiti sve eventualne promjene (novi *commit*-i) te će se automatski primijeniti na našem lokalnom repozitoriju (pod uvjetom da je to moguće).

Ovaj pristup je u redu ako ste sigurni da želite automatski prihvatiti promjene koje su napravljene na udaljenom repozitoriju. Alternativa bi bila korištenje naredbe:

```
git fetch
```

Ova naredba također dohvaća promjene sa udaljenog repozitorija ali ih neće automatski primijeniti. Na taj način možemo prvo provjeriti koje su promjene napravljene uz već spomenutu *git diff* naredbu:

```
git diff master origin/master
```

Nakon pokretanja naredbe u terminalu ćemo vidjeti točne promjene između ova dva repozitorija. Ako smo zadovoljni promjenama, možemo jednostavno pozvati *git pull* kako bi se one automatski primijenile.

> 💡 GitHub omogućava uređivanje sadržaja datoteka direktno iz preglednika. Dovoljno je otvoriti bilo koju datoteku te u gornjem desnom kutu odabrati opciju "*Edit*" (ikona olovke). Nakon promjene možete odmah napraviti i novi *commit*. Na ovaj način možemo jednostavno testirati dohvaćanje promjena sa udaljenog poslužitelja.

Grananje repozitorija

Mogućnost grananja je također jedna od osnovnih (i ključnih) funkcionalnosti *git*-a. Grananje nam dozvoljava stvaranje odvojenih pravaca razvoja unutar jednog repozitorija.

Pogledajmo na primjeru kako bi to moglo izgledati. Trenutno imamo početnu verziju našeg projekta i to je verzija koja radi stabilno i nema pogrešaka. Zamislimo da sada želimo aplikaciji dodati neku novu funkcionalnost ali ne želimo utjecati na trenutnu verziju aplikacije koja je temeljito testirana i ispravna.

Rješenje je napraviti novu razvojnu granu koja će sadržavati nove *commit*-e i funkcionalnosti, a pri tome će nam početna grana ostati nepromijenjena.

Stvaranje novih grana

Nova grana se može napraviti na nekoliko načina, za početak ćemo krenuti sa osnovnom naredbom:

```
git branch novi_dizajn
```

Ovom naredbom smo stvorili granu pod imenom "*novi_dizajn*". Ispis svih grana unutar repozitorija možemo dobiti sa naredbom:

```
git branch
```

Odabir trenutne grane

Ukoliko upišemo ovu naredbu vidjeti ćemo da imamo dvije grane, *master* (ili *main*) i *novi_dizajn*. Pri tome možemo uočiti (zbog boje i zvjezdice) da se još uvijek nalazimo na glavnoj grani. Kako bi se prebacili na novo napravljenu granu možemo upisati jednu od ove dvije naredbe:

```
git checkout novi_dizajn
git switch novi_dizajn
```

Ove dvije naredbe daju isti rezultat u slučaju kada želimo promijeniti grane, ali naredba `checkout` se može koristiti i za vraćanje promjena unutar datoteke pa je u kasnijim verzijama *git*-a dodana naredba `switch` koja služi samo za promjenu grana.

Sada možemo (u skladu sa imenom grane) napraviti nekoliko promjena u dizajnu aplikacije te napraviti novi *commit*. Ako vam je i dalje pokrenut lokalni React poslužitelj, možete iz terminala sa naredbom `git switch` mijenjati trenutnu granu i vidjeti kako se mijenja izgled aplikacije (i naravno kôd).

Spajanje grana

Jednom kada smo zadovoljni sa promjenama koje smo napravili u razvojnoj grani (i dobro ih testirali) vjerojatno ćemo željeti te promjene uključiti u "glavnu" verziju aplikacije. To se može postići tzv. spajanjem (***merge***) grana. Kada želimo spojiti dvije grane, prvo se moramo prebaciti na granu koja nam predstavlja "glavnu" granu (u koju želimo dodati promjene). Nakon toga pomoću naredbe `merge` odabiremo koju granu želimo pripojiti trenutnoj grani. U našem primjeru to bi izgledalo ovako:

```
git switch master
git merge novi_dizajn
```

Sada su svi *commit*-i koje smo napravili na grani "*novi_dizajn*" nadodani na granu *master*. Ukoliko smo u međuvremenu napravili neke odvojene *commit*-e na master grani, oni će također ostati sačuvani, naredba `merge` automatski radi novi *commit* na trenutnoj grani koja sadrži kombinaciju svih promjena sa obje grane.

Konflikti pri spajanju

Ukoliko prilikom spajanja obje grane sadrže promjene na istom dijelu projekta, događa se tzv "***merge conflict***". U tom slučaju će vam se u uređivaču kôda otvoriti novi prozor sa prikazom konfliktnih dijelova nakon čega morate ručno odabrati koja verzija će ostati sačuvana, a koje promjene će se odbaciti.

Ovime se već polako ulazi u složenije koncepte *git*-a koji nisu planirani u sklopu ovog tečaja pa se na njih nećemo dalje osvrtati. Naprednije poznavanje *git*-a zahtijeva upoznavanje sa složenijim konceptima kao što su *pull request*, *stash*, *cherry-pick* ili *interactive rebase*. Kao i svaka vještina, kako bi postali eksperti za rad sa *git*-om potrebno je dosta uloženog vremena i prakse. Prethodna dva poglavlja daju samo kratki uvod u osnovne funkcionalnosti koje su vam dovoljne za početak rada sa *git*-om.

Last updated on April 11, 2023