JUNIORDev DIGITALNA DALMACIJA

Upravljanje verzijama softvera

Git je distribuirani sustav za upravljanje verzijama softvera i koristi se za praćenje promjena u kôdu za vrijeme razvoja aplikacije. Git dozvoljava istovremeni rad više programera na istom projektu i pruža kompletnu povijest promjena koje su napravljene u kôdu. Na taj način se jednostavno mogu pratiti promjene na projektu, olakšana je međusobna suradnja te je po potrebi moguće vratiti projekt (kôd) u neku od prijašnjih verzija.

Korištenjem *git*-a osigurava se praćenje svih promjena u kôdu i njihovo ispravno spremanje tj. praćenje povijesti čime se smanjuje mogućnost pogreške ili različitih problema koji mogu nastati kada više ljudi istovremeno radi na istom projektu. Programeri (developeri) mogu raditi na svojim odvojenim verzijama (tzv. "grane" tj. "*branch*) koje se u nekom trenutku spajaju sa glavnom verzijom projekta. Korištenjem *git*-a taj cijeli proces se odvija na organizirani i kontrolirani način kako bi se izbjegle bilo kakve potencijalne pogreške. Dodatno, *git* nam pruža sigurnosnu kopiju (*backup*) cijelog projekta te omogućava jednostavan povratak na stariju verziju kôda u slučaju problema.

Pri radu sa *git*-om potrebno se upoznati sa nekoliko osnovnih koncepata:

- **Repozitorij** - (eng. ***repository***) ili skraćeno ***repo*** predstavlja kolekciju datoteka i direktorija koje pratimo pomoću *git*-a. Repozitorij sadrži sav kôd čiju povijest i promjene pratimo te predstavlja ključni koncept *git*-a. Razlikujemo **lokalni** i **udaljeni** repozitorij.
- ***Commit*** - predstavlja zapis promjena koje su napravljene unutar repozitorija. U pravilu svaki *commit* sadrži poruku sa opisom promjena te autora i vrijeme promjene. *Commit* možemo predstaviti kao stanje projekta (kôda) u nekom vremenskom trenutku.
- **Grana** (eng. ***branch***) - predstavlja odvojeni razvojnu liniju u *git*-u. Grane dozvoljavaju da više osoba radi na različitim funkcionalnostima aplikacije bez međusobnog ometanja. Također dozvoljava testiranje novih funkcionalnosti bez utjecaja na trenutnu (stabilnu) verziju aplikacije. Svaka grana ima svoj skup *commit*-ova (tj. promjena) koje se naknadno mogu pripojiti (***merge***) glavnom dijelu projekta tj. glavnoj grani.

Inicijalizacija repozitorija

Za početak je potrebno provjeriti je li *git* instaliran na računalu. To možemo napraviti pomoću terminala i naredbe:

```
git --version
```

Nakon čega bi vam se trebala ispisati trenutna verzija *git*-a koja je instalirana na računalu. Ukoliko se ispiše pogreška sa porukom da "*git*" naredba nije prepoznata, potrebno je sa službene stranice preuzeti i instalirati najnoviju verziju *git*-a.

Prije početka rada sa *git*-om, napraviti ćemo novi početni predložak React aplikcije (koristeći Vite kao i do sad)

```
npm create vite@latest vjezba06
```

Jednom kada je proces instalacije dovršen, možemo pokrenuti aplikaciju i testirati je li sve u redu. Nakon toga ćemo započeti sa inicijalizacijom *git* repozitorija.

Otvorite u VS Code uređivaču novi prozor terminala (ostaviti ćemo upaljen terminal koji pokreće razvojni React poslužitelj za pokretanje aplikacije) te provjerite nalazite li se u direktoriju projekta ("*vjezba06*" ako ste pratili gornji primjer). U tom slučaju sve što nam treba za inicijalizaciju novog *git* repozitorija je pokretanje naredbe:

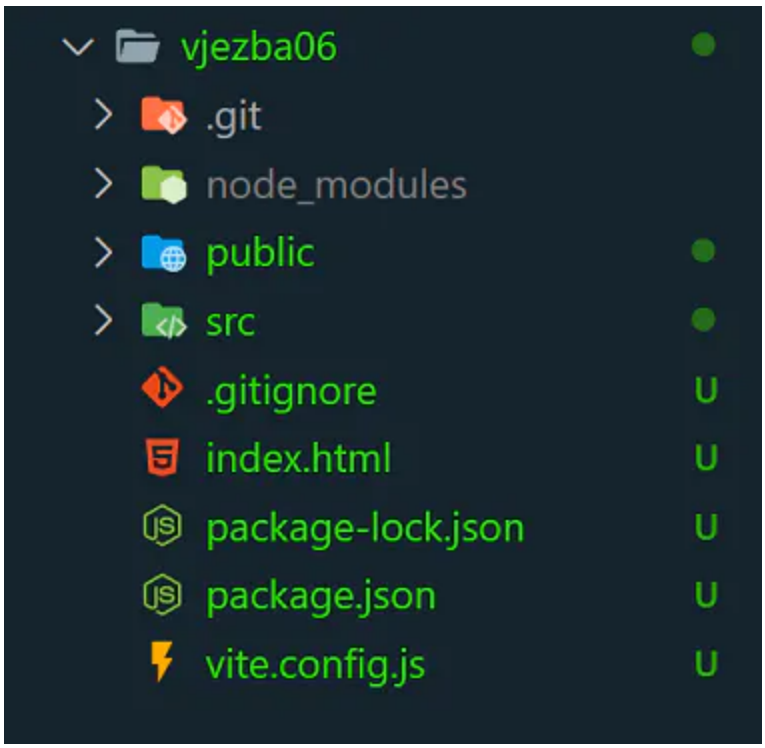
```
git init
```

Pokretanjem ove naredbe ispisati će vam se poruka `Initialized empty Git repository in ...` koja potvrđuje da ste u trenutnom direktoriju stvorili novi *git* repozitorij.

Ovisno o postavkama VS Code-a, unutar projekta možete uočiti da se stvorio novi direktorij

("**.git**" - ovaj direktorij je skriven pa se neće vidjeti ako nije uključena opcija da se prikazuju i skrivene datoteke) te jedna nova datoteka ("**.gitignore**"). To nam je jedan od pokazatelja da smo unutar glavnog direktorija projekta stvorili *git* repozitorij.

Također ovisno o postavkama VS Code-a (ili uređivača kôda kojeg koristite) moguća je promjena u prikazu datoteka projekta kao na slici u primjeru ispod.



U primjeru na slici datoteke unutar projekta (točnije unutar repozitorija) su sada prikazane zelenom bojom. Također, kraj svih sa desne strane stoji oznaka "**U**". To je kratica od ***untracked*** i govori nam da se te datoteke nalaze unutar repozitorija ali ne pratimo njihove promjene.

Stanje repozitorija možemo provjeriti sa naredbom:

```
git status
```

Nakon čega ćemo dobiti poruku sličnoj ovoj:

```
On branch master
No commits yet
Untracked files:
```

```
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        index.html
        package-lock.json
        package.json
        public/
        src/
        vite.config.js
nothing added to commit but untracked files present (use "git add" to track)
```

Iz gore navedenog ispisa možemo vidjeti nekoliko informacija:

- nalazimo se na grani ***master*** (u nekim verzijama će pisati ***main***)
- trenutno repozitorij ne sadrži niti jedan *commit*
- unutar repozitorija se nalaze direktoriji i datoteke čiju povijest ne pratimo

Git nam dozvoljava odabir kojim datotekama ćemo pratiti povijest. Iako možemo ručno definirati jednu po jednu datoteku sa naredbom `git add <ime_datoteke>` najčešće želimo pratiti promjenu nad svim datoteka unutar projekta. Tu funkcionalnost možemo jednostavno postići naredbom:

```
git add .
```

Provjerimo li sada stanje repozitorija, vidjeti ćemo da su sve datoteke u projektu u tzv. ***staging*** stanju. To znači da pratimo promjene nad njima i da su spremljene promjene u odnosu na prethodnu verziju repozitorija.

S obzirom da smo tek inicijalizirali repozitorij, nije bilo prethodnog stanja pa moramo napraviti prvi *commit* tj. spremiti početnu verziju našeg repozitorija. To radimo pomoću naredbe:

```
git commit -m "Prvi commit"
```

Opcija `-m` nam služi za pisanje poruke uz *commit*. To je uobičajena praksa kako bi jednostavnije znali koje su promjene napravljene u kojem *commit*-u te tako olakšali rad sa repozitorijem sami sebi ali i drugim osobama koje rade na istom repozitoriju.

Konačno sada bi nam ispis statuta repozitorija trebao pokazati poruku na kojoj se grani nalazimo i da ne postoji ništa za *commit*.

Praćenje promjena

Nakon što smo napravili prvi *commit* unutar repozitorija, možemo početi sa izmjenama u projektu kako bi se upoznali sa još nekoliko mogućnosti *git*-a.

Započeti ćemo sa komponentom "*App.jsx*" i zamijeniti početni izgled aplikacije.

App.jsx

```jsx
import './App.css'

function App() {

  return (
    <div className="App">
      <h1>Uvod u git</h1>
      <p>Praćenje promjena u repozitoriju</p>
    </div>
  )
}

export default App
```

Ispisom statusa repozitorija dobiti ćemo poruku otprilike ovakvog sadržaja:

```
(use "git add <file>..." to update what will be committed)
(use "git restore <file>..." to discard changes in working directory)
      modified:   src/App.jsx

no changes added to commit (use "git add" and/or "git commit -a")
```

Na terminalu se prikazuje koja je datoteka promijenjena u odnosu na prethodni *commit* te imamo na raspolaganju dvije opcije

- možemo odbaciti promjene koristeći naredbu `git reset` uz ime datoteke čije promjene želimo poništiti
- možemo prihvatiti promjene i označiti ih (*stage*) spremne za uključivanje u idući *commit* korištenjem naredbe `git add <ime_datoteke>`

Ukoliko želimo vidjeti koje su to točno promjene napravljene, to možemo postići uz pomoć naredbe:

```
git diff src/App.jsx"
```

Pozivanjem ove naredbe na terminalu se prikazuju sve promjene na odabranoj datoteci. Obrisani dijelovi datoteke su prikazani crvenom, a novi dijelovi zelenom bojom. U slučaju kada imamo veći broj promjena, kroz ispis promjena u terminalu se pomičemo sa tipkama "*PageUp*" i "*PageDown*" dok se za izlazak iz ispisa promjena koristi tipka "***q***".

S obzirom da želimo spremiti promjene na glavnoj komponenti, to ćemo i napraviti sa naredbom:

```
git add src/App.jsx
```

Kada imamo više promjena koje želimo istovremeno označiti za uključivanje u sljedeći *commit*, to naravno možemo napraviti sa naredbom koju smo koristili i prilikom prvog *commit*-a:

```
git add .
```

Nakon što smo označili sve promjene koje želimo sačuvati, vrijeme je za napraviti novi *commit*, uz pripadajuću poruku:

```
git commit -m "Nova struktura glavne komponente"
```

Sada naš repozitorij sadrži više *commit*-ova tj. imamo čitavu povijest promjena koje su

napravljene od trenutka kada smo inicijalizirali repozitorij i napravili prvi *commit*. Povijest repozitorija možemo vidjeti sa naredbom:

```
git log
```

Ova naredba će ispisati povijest *commit*-ova na repozitoriju, počevši od najnovijeg prema starijim. Uz svaki *commit* se vidi njegova jedinstvena *hash* oznaka, autor tog *commita*, datum kada je napravljen te poruka koju je autor napisao.

Ispis promjena

Ukoliko želimo vidjeti promjene između različitih *commit*-ova postoji više načina kako to napraviti, ovisno što želimo usporediti:

- sa naredbom `git log -p` možemo dobiti detaljni ispis svih promjena za svaki *commit*
- sa naredbom `git diff A B` možemo usporediti bilo koja dva *commit*-a (umjesto A i B pišemo njihove oznake)
- pomoću naredbe `gitk` možemo pokrenuti ugrađeni alat sa korisničkim sučeljem koji će nam prikazati cijelu povijest repozitorija

Sve navedeno u ovom dijelu se odnosi na upravljanje lokalnim repozitorijem ali gotovo uvijek ćemo naš lokalni repozitorij povezati sa nekim udaljenim repozitorijem. To radimo kako bi i drugi imali pristup našem projektu ili kako bi mi mogli pristupiti repozitoriju sa nekog drugog računala. Zato ćemo se sada kratko osvrnuti na rad sa udaljenim repozitorijima.

Last updated on April 10, 2023

‹ anje verzijama

Udaljeni repozitorij ›