Uputstvo za koriscenje GIT alata

1. Instalacija i konfiguracija GIT-a

- Windows - https://git-scm.com/download/win
- Linux - https://git-scm.com/download/linux

Nakon instaliranja mozemo proveriti da li je pravilno instaliran koristeci komandu:

- `git –version`

Ukoliko je GIT pravilno instaliran potrebno je konfigurisati ime i Email koji se koriste kao identifikacija unutar GIT-a.

To mozemo uraditi koristeci sledece komande:

- `git config --global user.name "Ime"`
- `git config --global user.email Email@email.com`

2. Repozitorijumi

Kako bih smo poceli sa radom, potrebno je da imamo lokalnu verziju repozitorijuma. To mozemo uraditi koriscenjem komande:

- `git clone <link>`

Gde je **<link>** link GIT repozitorijuma.

Ova omanda se izvrsava samo priliom reiranja lokalnog repozitorijuma.

Uolio zelimo da povucemo najnoviju verziju projekta koristimo:

- `git pull`

3. Priprema i commit-ovanje fajlova

Commit-ovanje je proces u kojem se nase promene “zavanicno” dodaju na GIT repozitorijum.

Commit-ovanje se moze videti kao cuvanje trenutne verzije naseg projekta, i ukoliko postoji potreba moguce je vratiti se na prethodnu verziju.

3.1 Provera statusa

Najpre moramo videti koji fajlovi su nam izmenjeni ili se ne prate od strane GIT-a, to mozemo uraditi komandom:

- `git status`

Koristeci informacije dobijene izvrsavanjem ove komande mozemo odrediti koje fajlove zelimo da pripremimo.

3.2 Priprema fajlova

Nakon sto znamo koje fajlove zelimo da commit-ujemo prvo ih moramo pripremiti, to mozemo uraditi koristeci **git add** omandu.

Ova omanda se moze koristiti na sledece nacine:

- `git add file.c file1.txt`

ukoliko zelimo da dodamo jedan ili vise specificnih fajlova,

- `git add .`

ukoliko zelimo da dodamo sve fajlove i foldera unutar foldera u kojem se nalazimo.

3.3 Commit-ovanje fajlova

Nakon pripreme fajlova spremni smo da te fajlove i commit-ujemo. To mozemo uraditi sledecom komandom:

- `git commit –m “Commit poruka”`

Gde je “commit poruka” poruka koja sluzi za organizaciju historije projekta.

Poruka treba biti deskriptivna i povezana sa promenama koje smo commit-ovali.

4. Istorija projekta

Ukoliko zelimo da vidimo celokupnu istoriju projekta, sve commit-ove i njihove kodove, to mozemo uraditi koristeci komandu:

- `git log`

Ukoliko zelimo da se prebacimo na neku od tih prethodnih verzija to mozemo uraditi komandom :

- `git checkout <commit-hash>`

Gde je **<commit-hash>** hash kod koji mozemo videti koristeci **git log**.

- `git checkout master`

Umesto <commit-hash> mozemo staviti **master** kako bih smo se vratili na glavnu verziju.

5. Grane

Grane(branches) se mogu videti kao alternativne verzije projekta, kako bi razlicite verzije projekta mogle paralelno da se razvijaju i prate.

Na ovaj nacin mozemo dodati nove, eksperimentalne, funkcionalnosti u nas projekat bez da menjamo ista unutar glavne grane.

Za pravljenje nove grane koristi se komanda:

- `git branch <ime-grane>`

Gde je **<ime-grane>** ime koje zelimo da damo novoj grani.

Ukoliko zelimo da promenimo na kojoj smo grani, to mozemo uraditi koristeci:

- `git checkout <ime-grane>`

5.1. Spajanje i brisanje grana

Ukoliko zelimo da spojimo dve grane u jedno kako bi smo implementirali promene to mozemo uraditi koristeci:

- `git merge <ime-grane>`

Gde se **<ime-grane>** treba zameniti imenom grane koju zelimo da spojimo sa granom u kojoj se trenutno nalazimo.

Ukoliko zelimo da obrisemo granu koristimo komandu:

- `git branch -d <ime-grane>`

Gde je **<ime-grane>** ime grane koju zelimo da izbrisemo.