KOLEGIJ

Mrežno programiranje

Laboratorijska vježba br.2

***Teme:***

* GIT
* Verzioniranje i upravljanje kodom

U ovom laboratorijskoj vježbi ćemo proći osnove GIT sustava, kontrole verzija i GitHub-a (nap. a. Možete koristiti bilo koju Git platformu; Vježba je pisana za Github).

# ŠTO JE GIT

**GIT** je sustav za kontrolu verzioniranja, tj. upravljanje revizijama izvornog koda. U ovoj vježbi ćemo se orijentirati na GIT. Postoji i na primjer, SVN koji je isto jako popularan. O SVN-u možete naći više informacija na Internetu.

VCS (*Version Control Software*) je softver koji pomaže timovima ili pojedincima upravljanje izvornim kodom u nekom vremenu. On prati promjene u projektu tako da ako se nešto „izjalovi“, korisnik može usporediti razlike između trenutne verzije i neke prošle verzije projekta na kojemu se radi. Može se čak napraviti i “checkout” prethodne verzije projekta, te izvršiti izmjene na takvoj verziji. Također, GIT podržava i odvojene „grane“ (*branches*), koji su zapravo odvojene, paralelne verzije projekta na kojima se može raditi simultano. Naprimjer, jedan član tima može raditi na nekoj funkcionalnosti u isto vrijeme kada drugi član radi na drugoj funkcionalnosti (ili pak isti pojedinac radi na dva odvojena brancha za dvije odvojene funkcionalnosti, itd.) Pri završetku razvoja, na primjer, obje grane se mogu spojiti iliti *merge-ati* natrag u glavnu granu koja se zove ***master branch***.

### ZADATAK:

<https://git-scm.com/>- instalirajte ispravnu verziju za vaš OS

# ŠTO JE GITHUB

GitHub je je platforma koja pruža sučelje prema Git-u za pohranu projekata koji koriste Git u oblaku. GitHub je koristan kada postoji više pojedinaca ili računala koji surađuju na projektu, iz razloga što se može relativno jednostavno ažurirati verzije projekta. GitHub je koristan i za osiguravanje pričuvne pohrane za projekt, jer se u slučaju gubitka lokalnog projekta može jednostavno klonirati cijeli projekt uključujući cijelu povijest razvoja tog projekta s GitHub-a.

### ZADATAK:

Otvorite račun na GitHub-u ili bilo kojem drugom sličnom rješenju.

# ZAŠTO KORISTITI GIT I GITHUB

S GITom i GitHubom možete jednostavno:

* Imati povijesni pregled projekta, tako da se u bilo kojem trenutku možete vratiti na stariju verziju u slučaju problema s novom, ili pak usporediti novu s nekom starijom verzijom, itd.
* Cijela povijest projekta je spremljena unutar tog projekta u direktoriju imena .git. Jako je teško izgubiti kod s Gitom.
* Upravljati različitim verzijama. Različite verzije se mogu označavati s tagovima, kao npr. v1.0.1, v2, itd. Tada se jednostavno može napraviti i checkout prethodnih verzija koristeći tagove.
* Surađivati s drugima.
* Imati različite grane. S različitim granama, možete na primjer, imati različite grane za različite funkcionalnosti, možete imati granu za dokumentaciju, itd.
* Jednostavno duplicirati bilo koju verziju projekta u neku drugu granu, napraviti promjene na njoj, te je spojiti natrag u originalnu granu.
* Git je super brz.
* Možete držati backup svih svojih projekata.
* Javno objavljivati svoj kod.
* Imati jedno online mjesto gdje držite svoje projekte,

dokumentaciju, itd.

# OSNOVE

S naredbom **git** u cmd-u ili Powershellu provjerite da li je ispravno instaliran.

## ZADATAK 1. – proći vježbu (prva 4 scenarija) na Katacoda online platformi

<https://www.katacoda.com/courses/git>

### Kao dokaz priložiti screenshot uspješnog završetka prva 3 scenarija na GIT tečaju.

### 

### 



**ZADATAK 2. – POSTAVLJANJE VLASTITOG PROJEKTA:**

1. Kreirajte prazan direktorij (**mkdir**)(nap. dajte mu smisleno ime za ovaj kolegij; dodajte neki obični tekst dokument), prebacite se u njega (**cd**), te pokrenite sljedeću naredbu za inicijaliziranje tog direktorija u git-u (**git init**).

Git će inicijalizirati taj direktorij i stvoriti će direktorij imena .git u root-u vašeg projektnog direktorija (onog kojeg ste kreirali).

1. Sada treba podesiti online kopiju projekta na GitHub-u. Na GitHubu dodajte novi repozitorij (postavite ime, public repositorij; **nemojte** staviti kvačicu na „initialize this repositorij with a README“).

Kopirajte URL vrijednost koju dobijete (za HTTPS).

Sada trebate signalizirati lokalnom Git projektu gdje mu je online, tj. remote projekt. Git URL udaljenog (*remote*) projekta naziva „origin“, pa pokrenite ovu naredbu da bi postaviti:

### git remote add origin <URL>

1. Postavit osnovne podatke u GIT – svaki git commit koristi ove informacije:

### git config –-global user.name „Ime Prezime“

**git config –-global user.email „e-mail adresa“**

Ukoliko želite možete postaviti i defaultni text editor, npr.

Notepad++:

**git config –-global core.editor "'putanja do exe' -multiInst -nosession"**

S **git config –-list** možete provjeriti postavke

1. Postoji opcija i kloniranja postojećeg projekta s GitHub-a naredbom git clone <URL>, ali to nam sada nije potrebno.

### GIT ADD, COMMIT i PUSH:

Git radi na način da uzme „*snapshot*“, tj. sliku u vremenu, vašeg projekta kad god izdate naredbu **commit**. Ti snapshoti su povijest vašeg projekta. Git ne sprema cijeli projekt, nego se referencira na prošle datoteke u projektu. Također ih i kompresira.

Tri osnovna nivoa GIT projekta su:

* + RADNI DIREKTORIJ – Direktorij u kojemu je smješten projekt.

Promjene se rade nad kodom u ovom direktoriju.

* + INDEKS ili „staging area“ – Ovdje se dodaju stvari za commit- anje. To se sprema u datoteku u .git direktoriju. Može se koristiti **git add** naredba za dodavanje datoteka iz radnog direktorija u stage, gdje se pripreme za commit-anje. Tada se može iskoristiti **commit** naredba za uzimanje snapshota svega na stage-u i spremanje u povijest projekta.
  + HEAD – referenca na prethodni commit ili checkout u trenutnoj grani.

**git push** naredba se koristi za ažuriranje udaljenog repozitorija sa

svim recentnim commit-ovima koji su zadani.

Recimo da imate neke nove ažurirane datoteke ili pak nove datoteke u vašem radnom direktoriju koje želite commit-ati. Prvo trebamo dodati te datoteke na stage iliti indeks:

**git add <filename>** - dodaje jednu datoteku na stage

**git add .** – dodaje sve na stage. Najčešće ćete je koristiti pošto dodaje sve promjene i nove datoteke u vašem projektu na stage.

Tada koristite **commit** za snapshot svih promjena u stageu, i slanje na HEAD. Svaki commit treba jedinstvenu poruku uz njega koja identificira u manje od jedne rečenice koje su promjene napravljene.

Npr.

### git commit –m „prvi commit“

Sada treba pushati promjene na remote repositorij na GitHubu. Prvi put kada ovo radimo treba specificirati koji remote origin „upstream“ bi željeli koristiti. S obzirom da smo još na defaultnoj grani, tj. master branch, damo sljedeću naredbu:

### git push -u origin master

Ukoliko pita za korisničke podatke za GitHub, unesite ih i pritisnite

Enter.

Na sljedećim pushevima na remote master granu, ne treba specificirati upstream pa se može koristiti samo naredba:

### git push

Git push samo pusha promjene u trenutnoj grani (master u ovom slučaju.)

**GRANE (eng. *BRANCHES*)**

Grane su kao različite kopije radnog direktorija koje se mogu odvojeno

modificirati, te zajedno spojiti u jednu granu. Ova naredba prikazuje u kojoj ste trenutno grani: **git status**

Ukoliko ste na glavnoj grani, ova naredba duplicira glavnu granu u odvojenu granu:

### git branch branch2

Branch2 je nakon ove naredbe identičan master grani. Da bi se prebacili

u novu granu koristimo naredbu checkout:

### git checkout branch2

Kreirajte novi dokument (novi tekstualni file iz naredbenog retka: npr. echo „“ > test.txt; stavite neki tekst unutar nakon kreiranja) u radnom direktoriju. Dodajte ga i commit-ajte. Pushajte promjene na

ovoj grani na remote repositorij. S obzirom da je ovo prvo pushanje na ovoj grani, morate specificirati upstream:

### git push -u origin branch2

Sada ćemo „merge-ati“, tj. spojiti promjene u master granu. Prvo se

prebacimo natrag u master granu, te tek onda merge radimo:

### git checkout master git merge grana2

Grana „master“ bi sada trebala sadržavati i novo kreirani tekstualni dokument.

Sada izbrišemo granu koju smo bili kreirali:

### git branch -d grana2

**POVLAČENJE PROMJENA SA GITHUB-A**

Da bi ažurirali lokalnu granu s posljednjim promjenama na remote origin grani, prvo se napravi checkout u granu koju želite ažurirati, te se napravi dohvat, tj. *fetch origin*:

### git checkout imegrane git fetch origin

Onda spojite, tj. merge-ate udaljenu granu s origin-a u lokalnu granu

### git merge origin/imegrane

**ILI (**sljedeća naredba napravi oboje odjednom**):**

### git pull

**ZAKLJUČNO**

**Sav kod koji ste dosada u ovim vježbama napravili stavite u remote repositorij na GitHub (prvo napravite lokalno pa push na remote repositorij). Pošaljite mi link do vaših javnih repositorija koji sadržava kod. Svaka sljedeća vježba će morati biti na GitHubu kao i ove prethodne koje morate staviti gore.**