1. Ukratko o poveznici verzioniranja koda s gradivom

Poveznica verzioniranja koda i teorije grafova se može opisati tako da se kaže da je repozitorij reprezentacija grafa. Početak verzioniranja koda najčešće započinje inicijalnim commitom kojega možemo promatrati kao korijen grafa, a ostale commitove kao vrhove. Također u verzioniranju koda prisutna je i kriptografija. Grafovi se sažimaju i spremaju. Podatci o grafovima dobijemo čitanjem njihovog SHA-1 sažetka.

2. Osnovni pojmovi u verzioniranju koda

Prije zadavanja zadatka potrebno je opisati neke osnovne pojmove koji su prisutni u verzioniranju koda, koji će vam pomoći bolje razumjeti zadatak jer će u zadatku biti korištena git terminologija.

Dok smo uspoređivali repozitorij sa teorijom grafova, spomenuli smo inicijalni commit. Općenito pojam commit opisuje stanje repozitorija u datom trenutku, ono pokazuje trenutno stanje, ali i svog roditelja (prethodnu verziju). Commit-om trenutna verzija postaje aktualna najnovija verzija, ali ponekad postoji potreba zadržati stare verzije. Taj problem rješava Branch (eng. grana). Branch je zapravo imenovani pokazivač koji pokazuje na specifičan commit ili liniju commit-ova. To omogućuje imati jednu radnu verziju dostupnu dok se u isto vrijeme radi na najnovijoj verziji. Sve grane se granaju od glavne (master/main) grane. Pojam Feature Branch se tipično pojavljuje u razvijanju programskog koda. Kod kreiranja nove funkcionalnosti poželjno je promjene, vezane za tu funkcionalnost, spremati na granu koja nije glavna (master) grana. Za funkcionalnost se kreira posebna grana koja služi realizacijom te funkcionalnosti. Nakon dovršavanja funkcionalnosti se ta grana spaja (merge) s glavnom ili nekom drugom granom. Merge (eng. spajanje) je operacija u kojem se spajaju dvije grane u jednu. Rezultat spajanja je jedna grana koja u sebi sadrži sadržaj i promjene s obje prethodne grane. Spajanje se provodi automatski ili ručno u slučaju konflikata. Konflikt se najčešće dešava kada u granama, koje pokušavamo spojiti, postoje različite promjene na isti tekst ili resurs. Postoje različiti načini spajanja, ali njih će te sami proučiti kao dio zadatka. U zadatcima se često spominje pojam Checkout. On označava operaciju u kojoj se ažurira lokalni direktori na način da se podudara s granom koju smo odabrali. Ona služi za „šetanje" po granama i prikaz aktualnog stanja na odabranoj grani. Prije rada na git projektu moramo inicijalizirat projekt, odnosno repozitorij. Inicijalizacija je operacija kreiranja novog repozitorija u kojem će se spremati podaci sa svih grana. Inicijalizacijom možemo dobiti novi prazan repozitorije ili možemo postaviti već napravljen, prije neverzioniran projekt. Pokretanjem inicijalizacije se kreira sav metadata koji opisuje projekt. To su podatci poput referenca, podataka o poddirektorijima i pokazivač na aktualnu najnoviju verziju.

3. Općenite informacije o projektnom zadatku

Projektni zadatak koji će vam biti zadan sastojati će se od tri podzatatka, po jedan za svakog člana tima. Naglasak je na načinima spajanja grana u verzioniranju koda pa preporučamo da proučite koji sve načini postoje i po čemu se razlikuju. Mi smo odabrali tri načina spajanja, svaki će biti vezan za jedan podzatak, to su spajanja "Octopus", "Ours/Theirs" I " Fast-forward merge". Svakako je potrebno detaljnije proučiti ova tri načina spajanja (merganja).

Zadatke je potrebno realizirati u nekom programskom jeziku po želji, potrebno je instalirati Git na računalo ukoliko već nije instalirano. Git možete preuzeti ovdje: https://git-scm.com/downloads. Potrebno je uz zadatak predati i sliku dobivenog grafa nakon uspješnih spajanja grana u repozitoriju, preporučamo da takav graf crtate na papiru. Ne mora biti jedan prikaz grafa za jedan zadatak, jedan zadatak može imati i više prikaza grafova/dijelova grafa, dokle god je jasno i razumljivo što se događa u verzioniranju. Preporučamo da u grafu/ovima označavate commitove, grane i spajanja. Ukoliko ćete za git naredbe koristiti konzolu, možete si pomoći nekim „cheat sheetom" koji su dostupni na internetu https://education.github.com/git-cheat-sheet-education.pdf. Možete koristiti i neki alat za vizualizaciju git aktivnosti, npr. https://www.gitkraken.com/.

Za svaki podzadatak, potrebno je inicijalizirati repozitorij i stvoriti Master granu koja će biti glavna grana.

3.1. Zadatak 1

Iz Master grane se checkouta feature grana A. Grana A sadržava proizvoljna dva commita A1 i A2. Nakon što su commitovi obavljeni u A grani, vraća se na Master granu i checkouta se na granu B, gdje se također radi proizvoljna dva commita B1 i B2, ali commitovi moraju biti različiti od commitova u A grani. (Savjet: ako A grana sadržava print(„Hello World"), u grani B se smatra da je različitost tako da u njoj piše print(„Hello Universe")). Checkouta se opet na Master granu i merga se A grana u master te se nakon uspješnog merga A grane u Master, se pokuša mergati i B grana u Master, dogodio se problem. Kako se naziva problem koji se dogodio? Zašto se on događa? Mi zasad ne želimo promjene s B grane, želimo zadržati promjene s Master grane, koristimo li zastavicu --ours ili -- theirs? Dalje se checkoutamo na granu B te želimo prenijeti promjene s Mastera na granu B, opet se dogodio problem, je li to isti problem kao kod merganja grane B u Master? Koristimo li u ovom slučaju zastavicu --ours ili --theirs? Sada su prenesene sve promjene s Mastera na granu B, može se napraviti još jedan commit B3, checkoutat na Master i mergat granu B na Master.

Opišite kako izgleda dobiveni graf. Je li se ovaj zadatak mogao bolje realizirati? Proučite pojam „cherry - pick", je li on mogao biti bolji izbor za realizaciju zadatka?

3.2. Zatadak 2

Iz Master grane se stvore feature grane A,B,C i D, a iz feature grane A se stvore još grane E,F,G. Svaka od tih grana E,F, G će imati barem jedan commit. Sada se treba vratiti na granu A i jednu po jednu granu (E,F,G) spojiti nazad u granu A. Kada su uspješno spojene grane E,F,G u granu A, treba se premjestiti u grane B,C i D te u svakoj od njih izvršiti barem jedan commit. Ukoliko svaka grana ima barem jedan commit, potrebno se vratiti nazad u Master granu i iskoristiti Octopus merge kako bi grane A, B, C, D spojili odjednom. Napišite tu naredbu kako biste Octopus mergom spojili sve te grane u isto vrijeme. Dogoditi će se jedan od dva scenarija, ili će se uspješno mergati sve u Master granu ili će nastati konflikt. Ukoliko nastane konflikt te ako ga znate riješiti, riješite ga, no svakako na papiru pri crtanju grafa naznačite merganje koje nema konflikata.

Opišite kako izgleda dobiveni graf. U kojim situacijama mislite da je korištenje Octopus merga dobra ideja, a u kojima mislite da je loša? Ukoliko ste imali konflikt pri zadnjem spajanju, mislite li da se to trebalo spojiti na drugi način i ako da, na koji?

3.3. Zadatak 3

U Master grani potrebno je napraviti barem tri commita te se iz zadnjeg commita checkoutat na novu feature granu A. U novoj feature grani A potrebno je napraviti barem dva commita te se od zadnjeg commita checkoutati na granu B. Na grani B je potrebno napraviti još barem dva commita. Izvršite fast - forward merge tako da zadnji commit na grani B postane Master grana. Nacrtajte i taj graf.

Opišite kako izgleda dobiveni graf, po čemu je specifičan naspram ostalih grafova?. U kojim slučajevima je fast - forward merge moguć? Što se točno dogodi u slučaju korištenja fast - forward merga sa Master granom i sa ostalim feature granama? Navedite situacije u kojima fast - forward merge neće proći. Kako glasi komanda ukoliko ne želimo da se dogodi fast - forward merge?