Sustav za praćenje napretka pisanja programskog koda

Blažević, Josip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij računarstva

Sustav za praćenje napretka pisanja programskog koda

Završni rad

Josip Blažević

Osijek, 2019.godina

Sadržaj:

1. UVOD

Programi za praćenje napretka pisanja programskog koda te njihovo spremanje su postali svakodnevno korišteni kod razvoja programa. Oni omogućavaju spremanje koda na jednostavan i kompaktan način. Programi se spremaju kao izvorni kod te se spremaju na lokalnom računalu ili na nekoj od udaljenih platformi. Kod ovakvog oblika spremanja programskog koda mogu se pregledavati promjene koje su načinjene prilikom razvoja programskog koda. Mogu se vidjeti promjene koje su nastale unutar svakog dokumenta unutar svake linije koda te dodane i obrisane linije programskog koda.

Rad je podijeljen na 6 dijelova: uvod, programi za praćenje napretka programskog koda, primijenjene tehnologije i alati, sustav za automatsko spremanje i praćenje programskog koda, prikaz spremljenih podataka i zaključak.

Unutar poglavlja programi za praćenje napretka programskog koda će biti opisani alati za praćenje napretka programskog koda, Git te poznate platforme za udaljeno spremanje programskog koda. U poglavlju primijenjene tehnologije i alati će biti opisani alati i tehnologije koji su potrebni za izradu sustava za automatsko spremanje programskog koda. Unutar poglavlja sustav za automatsko spremanje programskog koda će biti prezentiran sustav koji automatski sprema promjene prilikom pisanja programskog koda. U poglavlju prikaz spremljenih podataka će biti prezentiran način grafičkog prikaza sustava koji automatski sprema promjene prilikom programskog koda.

1.1. Zadatak završnog rada

Zadatak završnog rada je opisati sustave za praćenje napretka pisanja programskog koda, izraditi sustav koji će automatski spremati programski kod te prikazati dobivene podatke na grafu ovisno o broju promjena načinjenih prilikom pisanja programskog koda. Primjena ovakvog sustava bi mogla biti sustav protiv prepisivanja programskog koda u obrazovnim ustanovama s obzirom da bi u nekim verzijama nastajale poveće promjene, dok u drugima ne bi bilo značajnih promjena.

2. PROGRAMI ZA PRAĆENJE NAPRETKA PROGRAMSKOG KODA

2.1. Sustavi kontrole verzija

Sustavi kontrole verzija su skupovi datoteka koji se najčešće spominju u kontekstu programiranja, ali se ne moraju koristiti nužno za to. Sustavi kontrole verzija imaju razne primjene, može se pregledati tko je napisao kod, kada je napisan kod, što je u kodu promijenjeno, zašto je promijenjeno i slično. Vrlo značajna stvar kod ovakvih sustava je i što se datoteke mogu vratiti na jedno od prijašnjih stanja. Takvi sustavi sadrže povijest načinjenih izmjena kako bi omogućili takvo vraćanje. Kod razvoja programa u timovima je jako bitna suradnja te ovakvi sustavi to poboljšavaju. Postavljanjem verzija koda svaki programer može jednostavno preuzeti dijelove programa koje su napisali suradnici. Osim toga ako nastane neka velika pogreška postoji mogućnost vratiti se na prethodnu verziju te početi ispočetka. Sustavi kontrole verzija su sve više korišteni, a najveću primjenu imaju unutar IT poduzeća koji prate napredak svojih dokumenata i projekata koje razvijaju.

Počevši od definicije sustava kontrole verzija, to je mjesto na računalu (repozitorij), serveru ili nekom uređaju gdje se spremaju datoteke. Mogu biti sačinjeni od običnih tekstualnih datoteka, baza podataka ili u nekom drugom obliku. Takvome mjestu mogu imati pristup svi, ali puno češće samo određeni broj ljudi kojima je dozvoljen pristup ili samo osoba koja razvija taj kod. Ono što je specifično kod sustava kontrole verzija je da korisnik koji ima pristup tome mjestu može preuzeti postavljene verzije te nadodati nove.

Svaka verzija (revizija, inačica) dokumenta ima svoj broj. Od svoje inicijalne verzije pa koliko god verzija bilo, svaka verzija će dobiti svoj broj ili svoju oznaku. Kod korištenja Git-a svaka verzija dobije svoj kriptografski ključ, ali to sve ovisi o sustavu. Upisivanjem oznake verzije može se dobiti ispis svih podataka vezanih uz tu verziju.

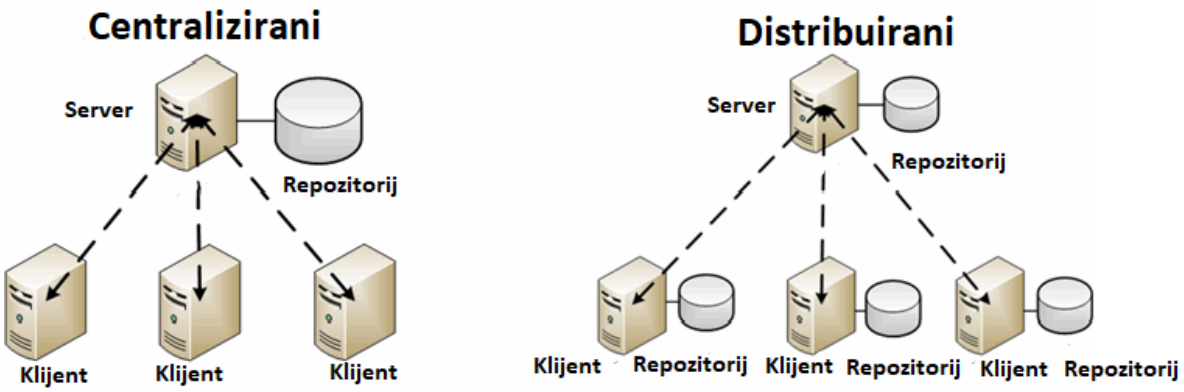
Sustavi za kontrolu verzija su najčešće zasebni alati koji se instaliraju na računala ili servere na kojima će se razvijati aplikacije ili nalaziti repozitoriji podataka. Osim takvih zasebnih aplikacija za kontrolu verzije kao što je Git, aplikacije poput Microsoft Office i Google Docs imaju ili su imale implementiranu povijest spremanja datoteka.

2.2. Vrste sustava za kontrolu verzija

Postoje dvije vrste sustava za kontrolu verzija, centralizirani i distribuirani.

Centralizirani sustavi su bazirani na ideji da postoji jedna centralna kopija koja se najčešće nalazi na serveru te se sve izmjene rade direktno na njoj. Takvi sustavi mogu biti opasni, pogotovo u većim timovima s obzirom da više osoba može raditi na jednoj datoteci istovremeno pa bi moglo doći do brisanja tuđih promjena. Sve izmjene se vide unutar povijesti dokumenta, ali se takvi izmijenjeni dokumenti moraju ručno spojiti u jednu.

Distribuirani sustavi, koji su gledani kao unaprjeđenje na centralizirane sustave, imaju centralizirani repozitorij, ali svaki korisnik ima njegovu lokalnu kopiju na svome računalu te sve izmjene na dokumentu radi lokalno. Tek kada je korisnik gotov sa izmjenama postavlja svoju verziju dokumenta na centralni repozitorij. Trenutno najkorišteniji sustavi kao što su Git, Mercurial i Bazaar su decentralizirani sustavi.



Slika 2.1. Prikaz centraliziranih i distribuiranih sustava za kontrolu verzija [1]

2.3. Najpoznatiji sustavi za kontrolu verzija

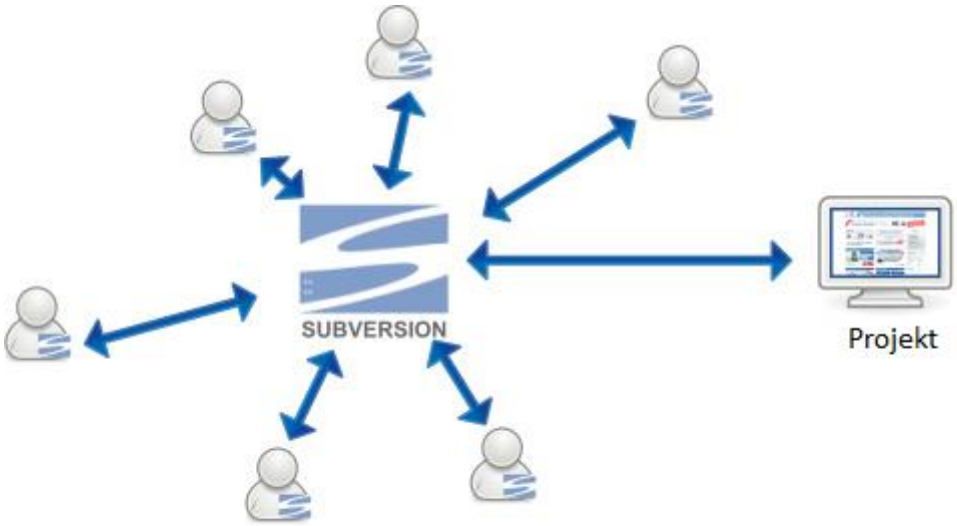
Neki od najpoznatijih sustava za kontrolu verzija su Git, SVN, Mercurial i Bazaar. Najkorišteniji od njih svih je Git čijim korištenjem je razvijen i sustav za automatsko spremanje programskog koda te će time Git biti detaljnije obrađen.

2.3.1. Subversion (SVN)

Subversion (SVN) je centralizirani sustav za upravljanje verzijama izvornoga koda, koji je nastao 2000. godine kao nadogradnja za Concurrent Versions System (CVS). SVN je sustav otvorenog koda što znači da je pogodan za izmjene samog sustava. Napisan je u C jeziku te radi na više operacijskih sustava. Naredbe započinju ključnom riječi svn. 2009. godine je primljen u Apache

Inkubator te mu je puno ime postalo Apache Subversion. Neki od klijenata za Subversion su TortoiseSVN, te direktne implementacije u integrirana razvojna okruženja kao Eclipse i NetBeans.

Sastoji se od većine naredbi koje su se nalazile unutar CVS, uz nadogradnje. Direktoriji su prvoklasni objekti, nalikuju datotekama. Kopiranje, brisanje i promjene imena su promijene koje mijenjaju postojeću te čine novu verziju. Omogućuje grananja i označavanja kako bi razvoj programa mogao ići u više smjerova istovremeno. Omogućuje zaključavanje datoteka kako ih ne bi više korisnika uređivalo u isto vrijeme u datoteke isključivo za čitanje te uz to sadrži mnoge druge značajke.



Slika 2.2. Subversion sustav za upravljanje verzijama izvornoga koda [2]

2.3.2. Mercurial

Mercurial je distribuirani sustav, podržan na većini današnjih operacijskih sustava. Osmišljen je kao sustav koji će se natjecati s Git-om pa je po svome izgledu i funkcionalnosti vrlo sličan Git-u. Najvećim dijelom je napisan u Python programskom jeziku, ali sadržava dijelove poput pretrage razlika u binarnim datotekama koji su napisani u C jeziku, ponajviše zbog performansi. Glavni ciljevi kod razvoja ovog sustava su bili postići visoke performanse, skalabilnost, decentralizaciju, potpuno distribuirani suradnički razvoj, rukovanje robusnim tekstualnim i binarnim datotekama te mnogi drugi. Sadržava internetsko sučelje te je primarno komandno pokretan program, ali sadržava i proširenja s grafičkim sučeljem s programima poput TortoiseHG te većim brojem

proširenja za integrirana razvojna sučelja. Sve njegove naredbe počinju sa ključnom riječi hg prema oznaci za kemijski element merkur.

```
$ hg clone https://www.mercurial-scm.org/repo/hello
$ cd hello
$ hg add datoteka
$ hg commit -m 'Promijenjena verzija'
$ hg push
```

Programski kod 2.1. Primjer upotrebe sustava Mercurial

2.3.3. Bazaar

Bazaar je sustav za kontrolu verzija izvornog koda koji pomaže u praćenju razvoja programa u nekome vremenu i olakšava suradnju s ostalima programerima. Skalabilan je što znači da se može koristiti samostalno ili u timovima bilo koje veličine. Decentraliziran je i distribuiran sustav. Napisan je u Python programskom jeziku radi jednostavnosti, visokih performansi te radi na većini operacijskih sustava. Slogan mu je „kontrola verzije za ljudska bića“ [3] zbog svojeg jednostavnijeg korištenja u usporedbi s ostalim sustavima kontrole verzija. Naredbe mu počinju sa ključnom riječi bzr.

Naredbe su slične CVS i SVN sustavima. Projekti mogu biti započeti i održavani bez udaljenog repozitorija korištenjem naredbe „bzr init“ u direktoriju za kojeg se želi vršiti kontrola verzija. Usprkos tome što je Bazaar distribuirani sustav, on omogućava rad sa centralnim serverom gdje u takvom slučaju se Bazaar može koristiti i kao centralizirani sustav. Omogućava rad s drugim sustavima, SVN verzije i verzije drugih sustava se mogu dohvaćati te spremati kao Bazaar verzije. Verzije podržavaju imena, poruke, imena osoba te drugo iz cjelokupnog Unicode skupa.

```
$ bzr init-repo primjer
$ bzr init primjer/sadrzaj
$ cd primjer/sadrzaj
```

Programski kod 2.2. Primjer upotrebe sustava Bazaar

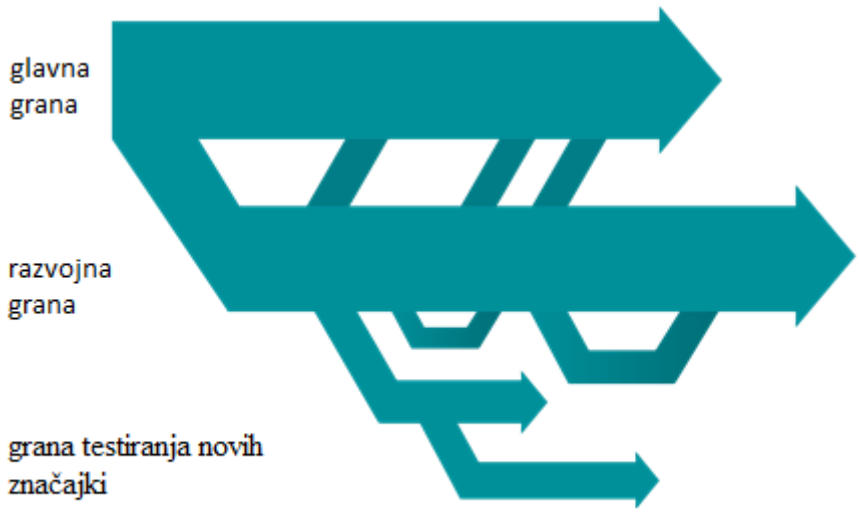
2.3.4. Git

Git je distribuirani sustav za upravljanje izvornim kodom nastao 2005. godine zbog ne zadovoljstva s tadašnjim sustavima za kontrolu verzija. Napisao ga je Linus Torvalds u C programskom jeziku za pomoć pri razvoju Linux jezgre. Besplatan je i otvorenoga koda što

omogućava izmjene sustava ukoliko su potrebne. Usklađen je s postojećim protokolima tipa http, ftp, ssh i drugima te je efikasan za rad na velikim i malim projektima. Pridana je velika pažnja na postizanje što većih performansi gdje prestiže većinu drugih sustava na tržištu. Sadržava značajke poput grananja projekata i više istovremeno pokrenutih radnih procesa. Radi na način da se svaki direktorij nalazi na svakome računalu te se nalaze potpuni repozitoriji na svakome računalu s potpunom poviješću te svim podacima o postojećim verzijama koji su na ovakav način ne ovisni o internetskoj vezi i centralnome serveru.

Grananja i spajanja

Git omogućava i potiče korištenje više lokalnih grana koje mogu biti potpuno nezavisne jedna o drugoj, a izrada, spajanja i brisanja tih linija koda traje do nekoliko sekundi. Na taj način se može vrlo brzo mijenjati kontekst razvoja programskog koda, mogu se izraditi nove grane, obaviti testiranja na njima te na kraju spojiti te grane, nadodati neki dio njih ili se takve grane mogu potpuno odbaciti i nastaviti s ranijim razvojem. Potiče se razdvajanje grana koda koje služe za krajnji proizvod, testiranje sustava te isprobavanje novih značajki.



Slika 2.3. Primjer grananja verzija koda [4]

Kompaktan i brz

Git obavlja većinu svojih operacija lokalno što ga čini bržim od centraliziranih sustava kontrole verzija koji moraju imati konstantu povezanost s centralnim serverom. S obzirom da je napisan u C jeziku, Git izbjegava neka od prevođenja koja su potrebna kod prevođenja još viših programskih

jezika. Provedenim mjerenjem koje je provedeno i objavljeno na službenoj Git stranici [4] su dobiveni podaci da Git 4-16 puta brže postavlja podatke u repozitorij, 4-71 puta brže izračunava razlike u kodu, 31-325 puta brže ispisuje podatke te 3 puta brže ažurira podatke od SVN koji je centralizirani sustav, a sporiji je jedino u kloniranju repozitorija sa centralnog servera. Tim podacima se mogu vidjeti poboljšanja u performansama kod distribuiranih sustava u usporedbi s centraliziranim.

Podatkovno osiguranje

Podatkovni model koji Git koristi kriptografski osigurava integritet svakog dijela projekta. Svaki dokument i svako postavljanje podataka u repozitorij su kriptografski osigurani. Ne moguće je dobiti podatke projekata i dokumenata u njima bez poznavanja cijelog kriptografskog ključa. Osim toga, ne moguće je mijenjati dokumente ili bilo kakve podatke vezane uz dokument bez mijenjanja njegovog identifikatora. Time takav promijenjeni dokument postaje nova verzija dokumenta, ali ne postoji način mijenjanja postojećih verzija.

Područje zadržavanja

Git u sebi sadržava područje zadržavanja kao međusloj između radnih datoteka i repozitorija. Unutar tog međusloja se mogu formatirati i pregledavati datoteke prije njihovog postavljanja u repozitorij. Taj međusloj omogućava da se objave samo određene datoteke ili samo određeni dijelovi datoteka u repozitorij u novoj verziji.

Slika 2.4. Tijek podataka [4]

Naredbe

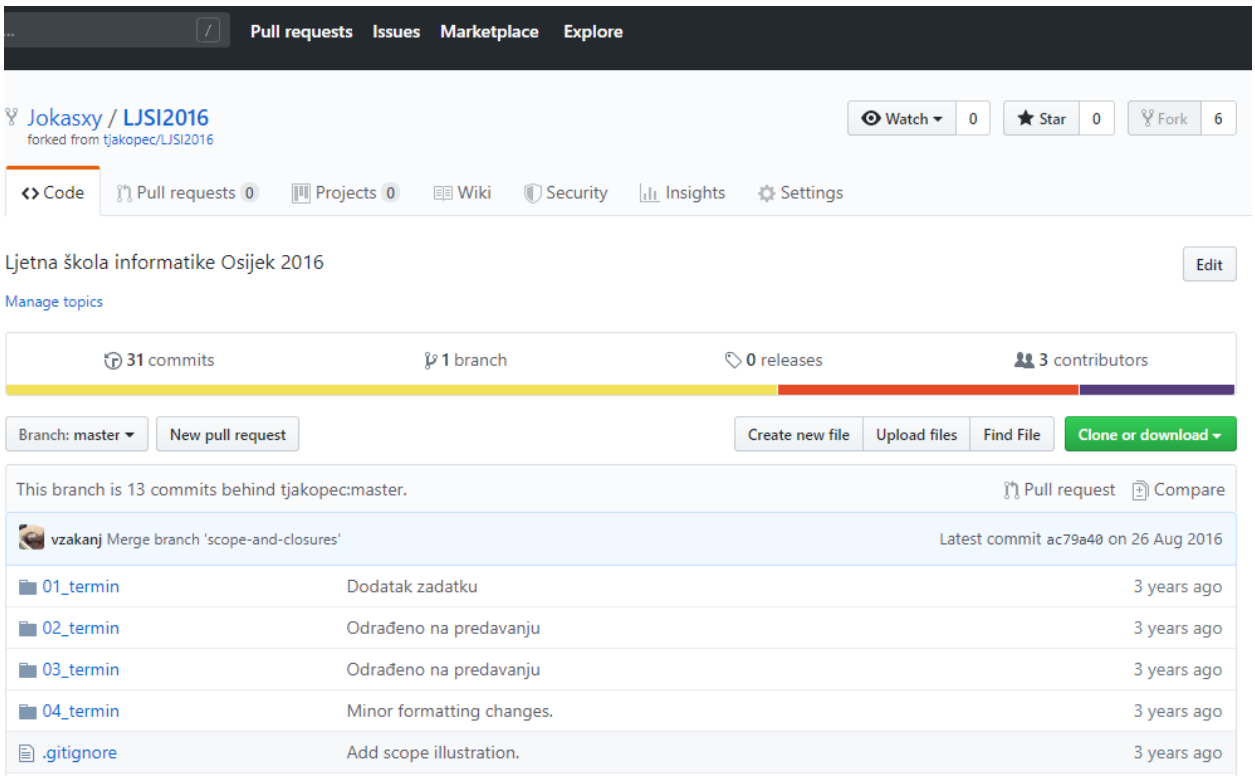
Git unutar svoje dokumentacije ima detaljno opisane naredbe, no opisati će se najkorištenije naredbe u svakodnevnoj uporabi. Git je naredba koja služi samo za praćenje podataka te se nalazi ispred svake ostale Git naredbe.

Tablica 2.1. Git naredbe

init	kreiranje praznog Git repozitorija ili ponovna inicijalizacija postojećeg repozitorija
clone	kloniranje repozitorija u novi direktorij
diff	prikazivanje izmjena između verzija
commit	spremanje izmjena u repozitorij
branch	ispis, kreiranje ili brisanje grana
checkout	promjena grane
merge	spajanje dvaju ili više razvojnih povijesti u jednu
fetch	preuzimanje objekata ili referenci iz drugih repozitorija
pull	preuzimanje objekata udaljenog repozitorija i povezivanje s postojećim repozitorijem
push	ažuriraj udaljeni repozitorij s povezanim objektima
add	dodavanje sadržaja datoteke u indeksnu datoteku

Platforme

Git platforme, poznate kao i Git internetski servisi, omogućavaju postavljanje Git repozitorija na udaljene lokacije. Takve platforme su platforme u oblacima kojima se može pristupiti preko internetskog preglednika. Unutar njih se mogu postaviti postavke tipa tko smije pristupiti postavljenome kodu, tko ga smije preuzimati, mijenjati, postavljati ga te druge postavke. Prilikom korištenja ovakvih platformi se ne mora brinuti za pristup tome kodu, potrebno je imati internetsku vezu te se onda može pristupiti kodu s bilo koje lokacije. Neke od najpoznatijih internetskih platforma koje pružaju usluge poslužitelja za kontrolu verzija koda koristeći Git su GitHub, GitLab, Bitbucket i SourceForge, . Pružaju funkcionalnosti distribuirane kontrole verzija i kontrolu izvornog koda iz samoga Git-a te dodaju i neke od svojih funkcionalnosti. Daju kontrolu pristupa te dodaju značajke poput praćenja programskih pogreški, zahtjeva za dodatne značajke, upravljanje zadacima te dodavanje opisa za svaki projekt. GitHub je trenutno najkorištenija platforma za kontrolu verzija koda koja je u svibnju 2019. godine brojala 37 milijuna korisnika i više od 100 milijuna repozitorija [5].



Slika 2.5. Primjer repozitorija GitHub platforme

3. PRIMIJENJENE TEHNOLOGIJE I ALATI

Za izradu sustava za automatsko spremanje i praćenje programskog koda osim sustava kontrole verzija Git su korišteni HTML i JavaScript opisni jezici te su napisane skripte Windows i Linux operacijskih sustava. Za izradu tih skripti te Html i JavaScript dokumenata je korišten Visual Studio Code tekstualni uređivač. Za iscrtavanje grafova je korištena JavaScript-a internet stranice Highcharts [6].

3.1. Skripte Windows operacijskog sustava (batch)

Skripte Windows operacijskog sustava su skripte koje se izvršavaju u naredbenome retku (cmd.exe). Pokretanjem takve skripte se slijedno pozivaju naredbe te skripte unutar naredbenoga retka. Takve skripte su podržane na svim Windows operacijskim sustavima skroz od vremena MS-DOS-a. Koriste se za izvršavanje osnovnih Windows naredbi. Uz osnovne Windows naredbe ovakve skripte podržavaju grananja i petlje kako bi olakšale rad i proširile mogućnosti u radu. Ovakve skripte imaju ekstenziju .bat.

3.2. Skripte Linux operacijskog sustava (shell)

Skripte Linux operacijskog sustava su skripte koje se izvršavaju u naredbenome retku od strane Unix ljuske. Pokretanjem takve skripte se slijedno pozivaju naredbe te skripte unutar naredbenoga retka. Tipične operacije koje se pišu unutar skripti su manipulacija datotekama, izvršavanje datoteka, te ispisivanje teksta. Skripta koja postavlja okruženje, pokreće program, čini potrebna čišćenja i zapisivanja se zove omotač. . Uz osnovne Linux naredbe ovakve skripte podržavaju grananja i petlje kako bi olakšale rad i proširile mogućnosti u radu. Ovakve skripte imaju ekstenziju .sh ili neku ako imaju proširene mogućnosti, ekstenziju .csh, .rsh, .ssh, .msh, .psh i druge.

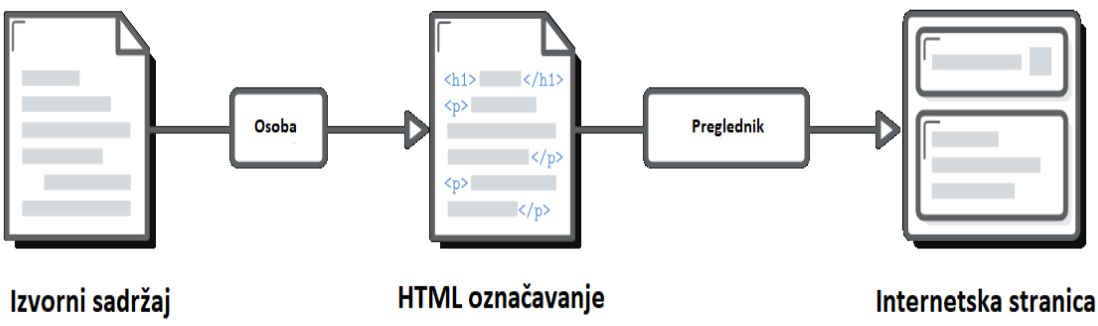
3.3. HTML

HTML je opisni jezik koji služi za izradu internetskih stranica. Njime stvaramo hipertekst dokumente koji imaju ekstenziju .html. Ovim jezikom postavljamo sadržaj u dokument, oblikujemo ga te postavljamo hiperveze u njega. Temeljna zadaća mu je objasniti internetskome pregledniku kako da prikaže hipertekst dokument. Nastoji se napraviti da se svaki dokument prikaže jednako ne ovisno o korištenom internetskome pregledniku, uređaju i operacijskome

sustavu. Njime se ne mogu izvršavati nikakve operacije pa čak ni najjednostavnije aritmetičke operacije, ne može se služiti za programiranje, već samo za opis hipertekst dokumenata. Osnovni elementi ovih dokumenata su oznake koje opisuju kako će se svaki element prikazati unutar internetskoga preglednika te svaka oznaka ima svoje zasebno značenje.

Tablica 3.1. Korištene HTML naredbe

<!DOCTYPE>	tip dokumenta
<html></html>	početak i kraj HTML dokumenta
<head></head>	definiranje podataka o dokumentu
<title></title>	definiranje naslova dokumenta
<meta>	definiranje meta podatka dokumenta
<body></body>	definiranje tijela dokumenta
	definiranje podjeljka dokumenta
<input>	definiranje ulaza u dokument
	definiranje zaglavlja dokumenta
<pre></pre>	definiranje unaprijed formatiranog teksta
<form></form>	definiranje forme za korisnički unos
<script></script>	definiranje klijentske skripte



Slika 3.1. Postupak označavanja teksta za prikaz na internetskome pregledniku [7]

3.4. JavaScript

JavaScript je opisni jezik koji uz HTML i CSS opisuje internetske stranice. JavaScript je orijentiran na funkcionalnost internetskih stranica. Omogućava korištenje funkcija, korisnički unos podataka, sadržaj se piše u uglatim zagradama te je prototipno objektno orijentiran što ga svojim stilom pisanja čini sličnim Java programskom jeziku. JavaScript datoteke imaju ekstenziju .js. Uz HTML i CSS se izvršava na klijentskom računalu te je postao standard za dodavanje funkcionalnosti na internetske stranice. Većina poznati internetskih preglednika u sebi ima ugrađen sustav koji pokreće takve skripte zbog čega je JavaScript postao zastupljen na gotovo svim internetskim stranicama.

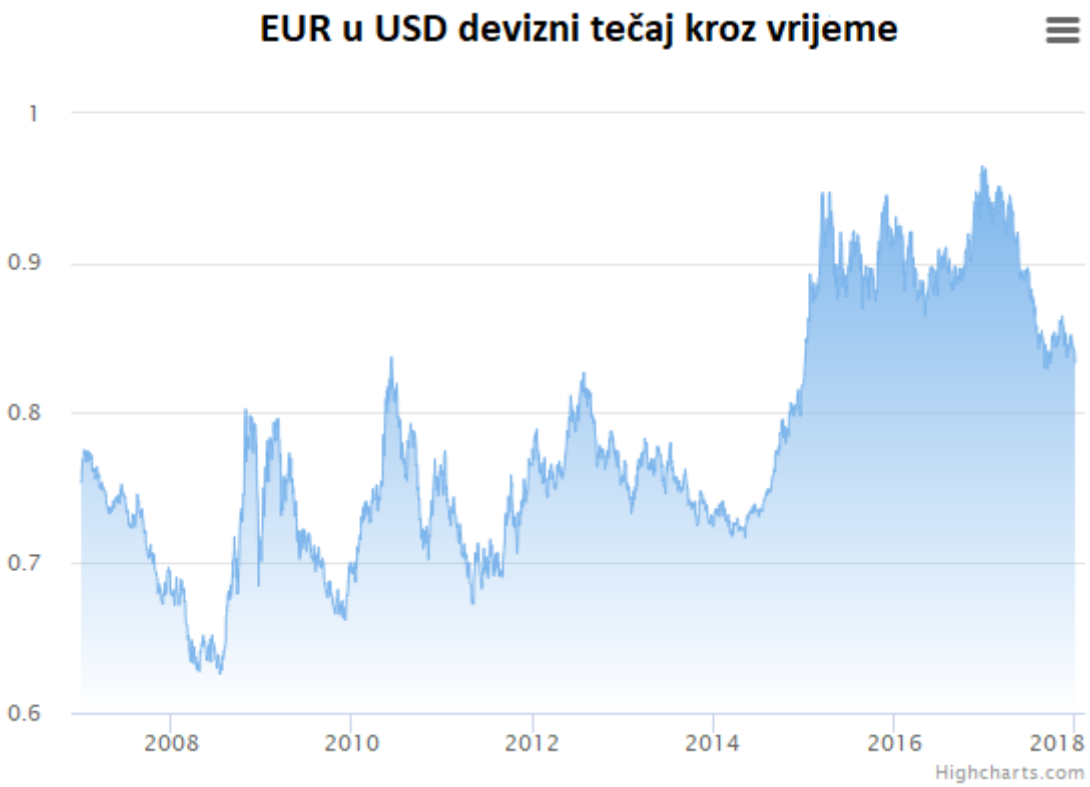
Podržava događajem pokretane, funkcionalne i imperativne programske stilove. Sadrži Aplikacijsko programibilna sučelja (API) za tekst, polja, vrijeme, regularne izraze (RegEx) i model objekta dokumenta (DOM), ali ne sadržava ulazno-izlaznu podršku poput mrežnog korištenja, spremanja podataka i iscrtavanja podataka koje korisnik mora samostalno nadodati kako bi ih JavaScript podržavao. Osim rada na klijentskim računalima gdje je najčešće korišten, JavaScript se može koristiti i na poslužiteljskim računalima, unutar baza podataka te u programima poput PDF čitača kao procesor riječi.

JavaScript omogućava korištenje velikog broja biblioteka koje mu proširuju ili pojednostavljuju korištenje. Biblioteke poput Angular, React i Vue.js olakšavaju izradu korisničkog sučelja, jQuery olakšava rad s dobivenim podacima, HighCharts omogućava iscrtavanje grafova te postoji još velik broj biblioteka koji se svakodnevno povećava s obzirom da se svakodnevno razvijaju nove biblioteke za JavaScript.

3.5. HighCharts

HighCharts je biblioteka napisana koristeći JavaScript kako bi omogućila iscrtavanje grafova na internetskim stranicama koristeći JavaScript. Omogućava dodavanje interaktivnih i responzivnih

grafova koji su optimizirani za korištenje na računalima i mobilnim uređajima. Omogućava korištenje velikih podataka, ima ugrađenu podršku za dodirne pločice, dinamičke promijene podataka, sadrži sustav za otklanjanje pogreški u kodu te druge funkcionalnosti.



Slika 3.2. Primjer grafa HighCharts biblioteke [6]

3.6. Visual Studio Code

Visual Studio Code je tekstualni uređivač koji je razvijen od strane Microsoft-a kako bi olakšao razvoj programskog koda. Kompatibilan je s Windows, Linux i macOS operacijskim sustavima. Ima podršku za otklanjanje programskih pogrešaka, ugrađenu Git kontrolu, pristup GitHub-u kao Git platformi, označavanje sintakse, inteligentno popunjavanje koda, pomoć pri pisanju blokova koda i olakšava refaktoriranje koda. Podržava dodavanje dodatnih proširenja koja dodaju nove funkcionalnosti te podržava različite teme i prečace. Visual Studio Code koristi Electron *framework* koji se koristi i za razvoj Node.js aplikacija.

4. SUSTAV ZA AUTOMATSKO SPREMANJE I PRAĆENJE PROGRAMSKOG KODA

Sustav za automatsko spremanje i praćenje programskog koda je realiziran na način da se pomoću skripti Windows i Linux operacijskog sustava izvršavaju Git naredbe. Nakon što se „git.bat“ ili „git.sh“ skripta izvrši, sustav čeka 300 sekundi te se ponovno izvršava nakon toga. Time dobivamo automatsko spremanje podataka u repozitorij svakih 5 minuta. Nakon što je završeno pisanje programskog koda, izvršavanjem „gitusporedba.bat“ ili „gitusporedba.sh“ skripte se dobiva „gitrazlika.txt“ tekstualni dokument koji sadrži podatke o tome koliko je linija dodano te koliko je linija obrisano za svaki dokument u usporedbi trenutne verzije s početnom verzijom.

Prije izvršavanja Git naredbi potrebno je u naredbenome retku otići na lokaciju gdje će se nalaziti Git repozitorij. Naredbom „cd %~dp0“ unutar skripte Windows operacijskog sustava u naredbenome retku se ulazi u direktorij u kojemu se nalazi skripta koju izvršavamo, nakon koje sljedećom naredbom ulazimo u direktorij „apk“ iz kojega će se cjelokupan sadržaj spremati u repozitorij. Unutar skripte Linux operacijskog sustava nije potrebno ući u direktorij u kojemu se nalazi skripta koju izvršavamo pošto na Linux operacijskom sustavu se naredbeni redak sam postavi u direktorij radni direktorij, nakon čega moramo samo ući u „apk“ direktorij. „git init“ naredba izrađuje novi ili ponovno inicijalizira postojeći repozitorij. Unutar beskonačne petlje naredbom „git add“ uz modifikator „--all“ se postavlja cjelokupan sadržaj trenutnog direktorija u područje zadržavanja čime se pridodaje indeks svakoj datoteci prije postavljanja u repozitorij. Naredbom „git commit“ postavljamo sadržaj iz područja zadržavanja u repozitorij kao novu Git verziju, a s modifikatorom „-m“ dodajemo poruku vezanu uz verziju. Na kraju petlje definiramo čekanje u vremenu od 300 sekundi.

```
cd %~dp0
cd apk
git init
:petlja
    git add --all
    git commit -m "automatsko spremanje"
    TIMEOUT 300
goto petlja
```

Programski kod 4.1. „git.bat“ skripta

```
#!/bin/sh
args=("$@")
cd apk
git init
while true
do
    git add --all
    git commit -m "automatsko spremanje"
    sleep 300
done
```

Programski kod 4.2. „git.bat“ skripta

Kao i u „git.bat“ skripti mora se promijeniti direktorij u naredbenome retku unutar skripte Windows operacijskog sustava, no ovaj put prije nego što se uđe u „apk“ direktorij, obrisati će se sadržaj datoteke „gitrazlika.txt“ ukoliko ona postoji iz direktorija u kojemu se nalazi skripta. Naredba „git rev-list --all --count“ vraća broj verzija koje su spremljene u repozitorij, a prva for petlja skripte Windows operacijskog sustava služi isključivo kako bi spremila rezultat naredbe u varijablu. Ako imamo n verzija, one mogu biti pomaknute u intervalu [1, n-1] od početne verzije pa od broja verzija moramo oduzeti broj jedan kako bismo dobili dobar broj koraka for petlje unutar skripte Windows operacijskog sustava. Unutar skripte Linux operacijskog sustava nije potrebno tako spremati rezultat navedene naredbe kao varijablu, već se samo pridoda vrijednost varijabli, te se ne mora oduzeti vrijednost 1 od broja verzija jer u for petlji postavimo da varijabla „n“ mora biti manja od zadane vrijednosti. Unutar for petlje se u datoteku „gitrazlika.txt“ upisuju razlike između verzija u repozitoriju. Kao parametre naredbe koja izračunava razlike između verzija upisujemo pomak od početne verzije kojeg dobijemo putem for petlje i početnu verziju.

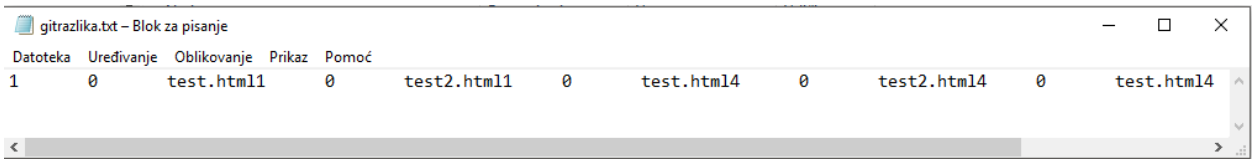
```
cd %~dp0
break>gitrazlika.txt
cd apk
for /f "tokens=* USEBACKQ" %%f in (`git rev-list --all --count`) do (
set var=%%f
)
set /a jedan=1
set /a var=%var%-%jedan%
for /l %%n in (1,1,%var%) do (
    git diff --numstat HEAD~%%n HEAD>>..\gitrazlika.txt
)
```

Programski kod 4.3. „gitusporedba.bat“ skripta

```
#!/bin/sh
args=("$@")
>gitrazlika.txt
cd apk
var=`git rev-list --all --count`
for((n=1;n<$var;n++))
do
    git diff --numstat HEAD~$n HEAD>../gitrazlika.txt
done
```

Programski kod 4.4. „gitusporedba.sh“ skripta

Unutar „gitrazlika.txt“ tekstualne datoteke se podaci zapisuju u formatu „broj dodanih linija broj obrisanih linija ime datoteke“ gdje iza imena datoteke dolazi znak za novu liniju koji je nevidljiv u standardnom tekstualnom pregledniku.



Slika 4.1. „gitrazlika.txt“ tekstualna datoteka

5. PRIKAZ SPREMLJENIH PODATKA

Grafički prikaz podataka koji su spremljeni u „gitrazlika.txt“ je omogućen koristeći „prikaz.html“ HTML datoteke. U njoj je definiran izgled internetske stranice, čija je funkcionalnost napisana unutar <script></script> HTML oznaka umjesto u zasebnoj .js datoteci. Osim napisane skripte preko interneta se povlači i udaljena skripta HighCharts biblioteke koja omogućava iscrtavanje grafova.

Nakon što se stvori tekstualna datoteka „gitrazlika.txt“, pokretanjem „prikaz.html“ HTML datoteke i odabirom te tekstualne datoteke dobivamo ispis sadržaja te datoteke. Unosom imena datoteke koja čiji se razvoj pratio u predviđeno mjesto i klikom na gumb „Pokreni“ dobivamo graf popunjen s podacima o broju promjena između verzija za željenu datoteku.

 definira podjeljak stranice u kojemu će se graf prikazati. Unutar prvog <script> se poziva HighCharts skripta za iscrtavanje grafova. Unutar drugog <script> definiramo izgled grafa koji se prikazuje. Ovdje se definira prazan graf, a njegova svojstva definiraju:

Tablica 5.1. Svojstva praznog grafa

title – text	naslov grafa
yAxis – title – text	ime y osi grafa
xAxis – titile – text	ime x osi grafa
legend – layout:vertical	elementi legende će se nizati jedan ispod drugoga
legend – align:right	poravnanje teksta legende uz desni rub
legend – verticalAlign:middle	vertikalno poravnanje legende sa sredinom grafa
series – name	ime niza podatka koji će se prikazati
responsive – condition – maxWidth:500	ako je širina preglednika 500 ili manje piksela mijenja se raspored elemenata

responsive – chartOptions – legend – layout:horizontal	ako je uvjet ispunjen, mijenja raspored elemenata u horizontalni
responsive – chartOptions – legend – align:center	ako je uvjet ispunjen, mijenja se poravnanje teksta legende na centriran tekst
responsive – chartOptions – legend – verticalAlign:bottom	ako je uvjet ispunjen, tekst prelazi ispod grafa

```
<div id="spremnik" style="width:100%; height:100%;"></div>
<script src="https://code.highcharts.com/highcharts.js"></script>
<script type="text/javascript">

      Highcharts.chart('spremnik', {

                title: {
                    text: 'Git verzije'
                },

                yAxis: {
                    title: {
                        text: 'Količina koda'
                    }
                },

                xAxis: {
                    title: {
                        text: 'Git verzija'
                    }
                },

                legend: {
                    layout: 'vertical',
                    align: 'right',
                    verticalAlign: 'middle'
                },

                series: [{
                    name: 'Dodane linije koda',
                }, {
                    name: 'Obrisane linije koda',
```

```
            }],

            responsive: {
                rules: [{
                    condition: {
                        maxWidth: 500
                    },
                    chartOptions: {
                        legend: {
                            layout: 'horizontal',
                            align: 'center',
                            verticalAlign: 'bottom'
                        }
                    }
                }]
            }
        });
    </script>
```

Programski kod 5.1. Definiranje praznoga grafa

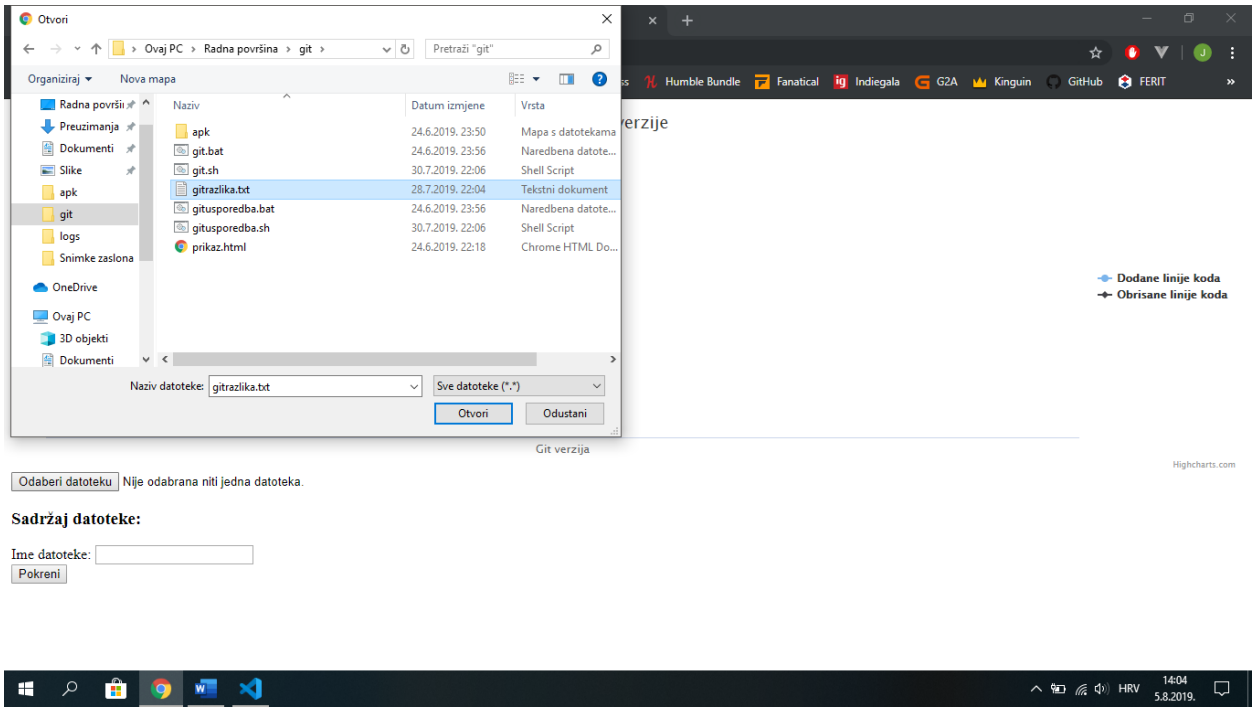
Slika 5.1. Izgled internetske stranice bez podataka

S <input> se označava korisnički unos podataka. „type=file“ govori da je riječ o čitanju datoteke. Kada se klikne na odabir datoteke poziva se funkcija „procitajdatoteku“ preko slušatelja događaja, koji prati promjene, odnosno je li zadana datoteka. Kada se funkcija pozove, provjerava se je li odabrana datoteka, ako nije izlazi se odmah iz funkcije. Ako je datoteka izrađuje se objekt čitača datoteka koji čita i ispisuje sadržaj datoteke. Sadržaj datoteke se popunjava pozivanjem funkcije popuni sadržaj. Popuni sadržaj dijeli sadržaj u polje po oznakama novog reda, gdje je onda svaka

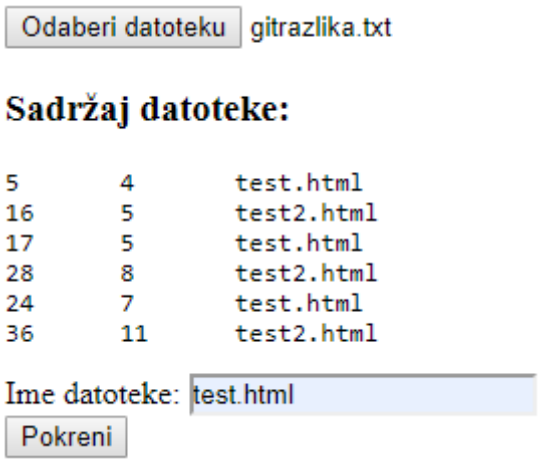
verzija za jedan dokument jedan element polja. Funkcijom „pop()“ se izbacuje zadnji element polja koji je uvijek prazan. U for petlji se svaki element polja dijeli na novo polje gdje se niz dijeli po razmaku. Kada se niz podijeli u polja „dodano“ i „obrisano“ se dodaju objekti koji sadržavaju koliko je dodanih ili obrisanih linija dokumenta i za koju je datoteku to vezano. Brojčane vrijednosti se iz znakova pretvaraju u cjelobrojne brojeve.

```
<input type="file" id="unosDatoteke" />
<h3>Sadržaj datoteke:</h3>
<pre id="sadrzajDatoteke"></pre>

<script type="text/javascript">
      var dodano = [];
      var obrisano = [];

document.getElementById('unosDatoteke')
            .addEventListener('change', procitajDatoteku, false);

function procitajDatoteku(dat) {
                var datoteka = dat.target.files[0];
                if (!datoteka) {
                    return;
                }
                var citac = new FileReader();
                citac.onload = function(dat) {
                var sadrzaj = dat.target.result;
                popuniSadrzaj(sadrzaj);
                };
                citac.readAsText(datoteka);
            }

            function popuniSadrzaj(sadrzaj) {
                var element = document.getElementById('sadrzajDatoteke');
                element.innerHTML = sadrzaj;
                sadrzaj = sadrzaj.split('\n');
                sadrzaj.pop();
                for(i = 0; i < sadrzaj.length; i++){
                    sadrzaj[i] = sadrzaj[i].split(' ');
                }
                for(i = 0; i < sadrzaj.length; i++){
                    dodano.push({dodano: parseInt(sadrzaj[i][0]), datoteka:
sadrzaj[i][2]});
                    obrisano.push({obrisano: parseInt(sadrzaj[i][1]), datoteka:
sadrzaj[i][2]});
                }
            }
</script>
```

Programski kod 5.2. Odabir tekstualne datoteke, čitanje i popunjavanje sadržaja



Slika 5.2. Odabir tekstualne datoteke za prikaz



Slika 5.3. Ispis sadržaja datoteke i unos imena datoteke za koju se želi vidjeti broj promjena

Za unos imena datoteke za koju se žele vidjeti promjene između verzija programskog koda je napravljena forma koja sadrži unos teksta i gumb na kojega kada se klikne poziva funkciju „pokreni“. U funkciji „pokreni“ se u nova polja spremaju vrijednosti samo željene datoteke. For pelja prolazi kroz cijelo polje, a u if grananju se provjerava je li taj podataka iz željene datoteke.

Nakon toga se preuređuje izgled grafa koji je po svemu identičan kao i onaj u ranijoj fazi, osim što sada ima podatke za iscrtati. Podaci se unose kao:

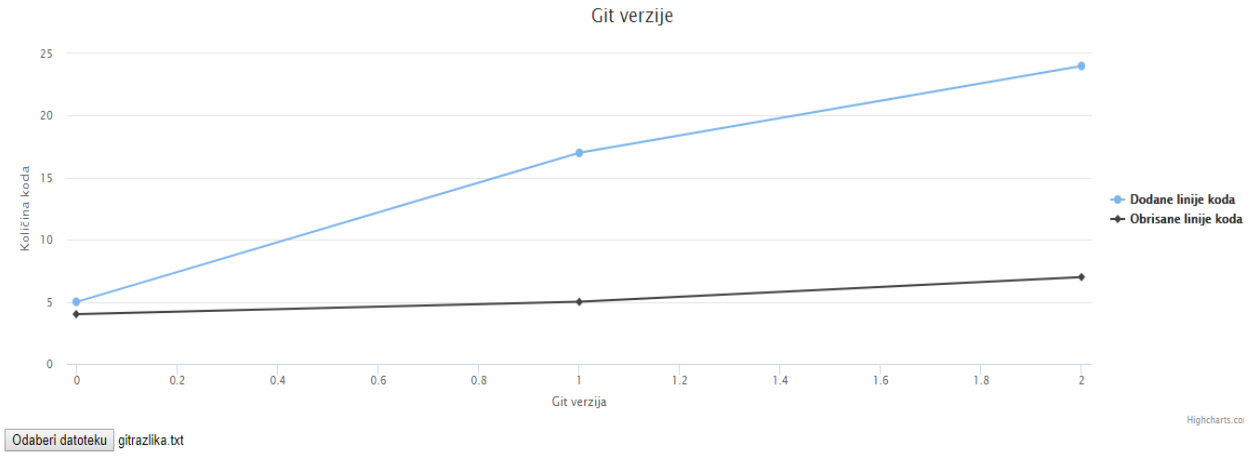
Tablica 5.2. Definiranje podataka koji se unose na graf

series – data	podaci koji se iscrtavaju

```
<form>
      Ime datoteke: <input type="text" name="datoteka" id="datoteka"><br>
      <input type="button" value="Pokreni" onclick="pokreni()">
</form>
<script type="text/javascript">
            function pokreni()
            {
                var ispisDodano = [];
                var ispisObrisano = [];
                for(i = 0; i < dodano.length; i++){
                    if(dodano[i].datoteka ==
document.getElementById('datoteka').value){
                        ispisDodano.push(dodano[i].dodano);
                        ispisObrisano.push(obrisano[i].obrisano);
                    }
                }

                Highcharts.chart('spremnik', {
                    title: {
                    text: 'Git verzije'
                },

                yAxis: {
                    title: {
                        text: 'Količina koda'
                    }
                },

                xAxis: {
                    title: {
                        text: 'Git verzija'
                    }
                },

                legend: {
                    layout: 'vertical',
                    align: 'right',
                    verticalAlign: 'middle'
```

```
            },

                series: [{
                    name: 'Dodane linije koda',
                    data: ispisDodano
                }, {
                    name: 'Obrisane linije koda',
                    data: ispisObrisano
                }],

                responsive: {
                rules: [{
                    condition: {
                        maxWidth: 500
                    },
                    chartOptions: {
                        legend: {
                            layout: 'horizontal',
                            align: 'center',
                            verticalAlign: 'bottom'
                        }
                    }
                }]
            }
            });
        }
    </script>
```

Programski kod 5.3. Odabir datoteke za prikaz rezultata i njezin prikaz

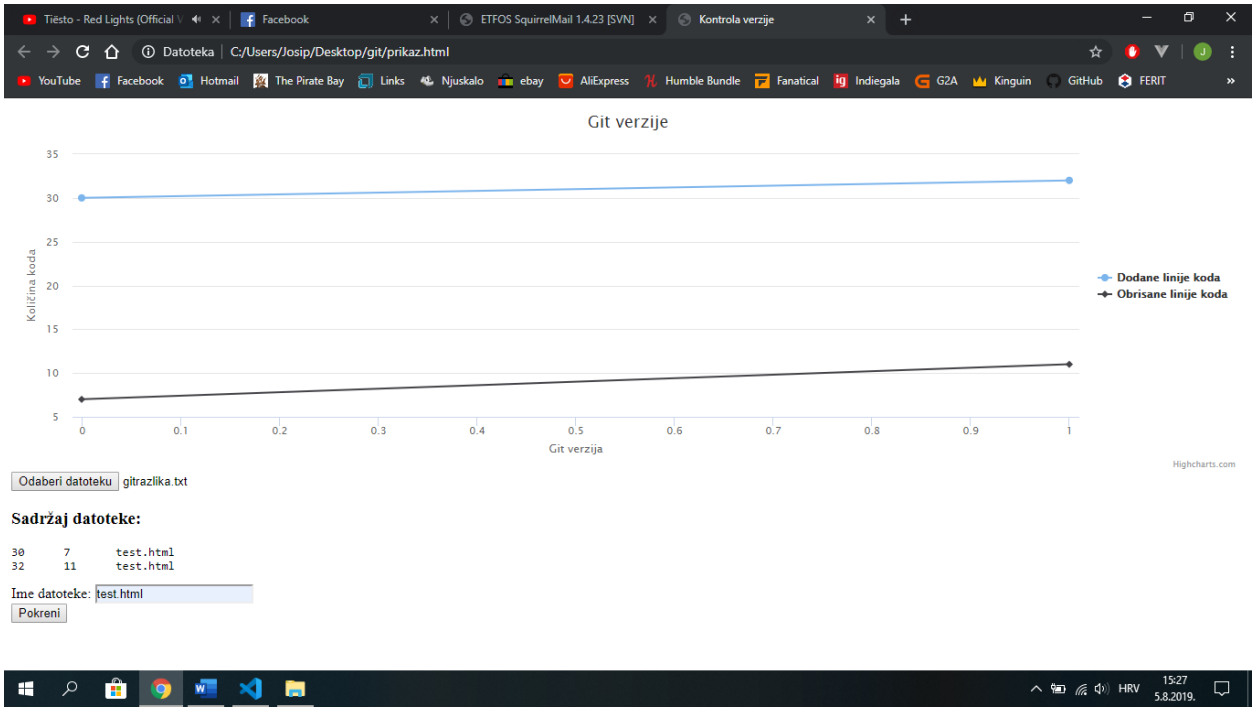


Odaberi datoteku gitrazlika.txt

Sadržaj datoteke:

Slika 5.4. Prikaz podataka

Na slici 5.4. je prikazan normalni tijek razvoja programa. Unutar razvoja će broj dodanih i obrisanih linija polagano rasti i mijenjati se, te će biti veći broj verzija programa.



Slika 5.5. Primjer prepisivanja programskog koda

Na slici 5.5. je prikazan primjer prepisivanja programskog koda. Prepisan programski kod će najčešće imati mal broj verzija gdje će u nultoj Git verziji, odnosno usporedbi prve verzije s inicijalnom biti velik broj dodanih linija. U slučaju prepisivanja i prepravljanja koda će biti više verzija nego kod samoga prepisivanja, ali ovdje može poslužiti broj obrisanih linija koda. Kod prepravljanja programskog koda, svaka prepravljena linija koda će biti zapisana kao jedna dodana i jedna obrisana linija.

6. ZAKLJUČAK

Izrađen je sustav za automatsko spremanje i praćenje programskoga koda te sustav za prikazivanje rezultata prvo navedenog sustava. Kao preduvjet tome svemu, bilo je potrebno proučiti sustave za kontrolu verzija, njihove prednosti i mane te odabrati onaj najpogodniji sustav. Git kao najkorišteniji i vrlo dobro dokumentirani sustav se činio najpogodnijim s najvećim brojem dostupnoga materijala za učenje. Nakon odabira sustava bilo je potrebno proučiti i odabrati alate i tehnologije izrade. Git se može izvršavati direktno iz naredbenoga retka pa se pokazalo logično koristiti ga koristeći skripte Windows i Linux operacijskih sustava. Za prikaz podataka ovoga sustava su odabrani HTML i JavaScript opisni jezici gdje se uz pomoć samo jedne biblioteke omogućilo crtanje grafova, a uz dodavanje poslužitelja, na ovaj način bi se moglo pristupiti ovakvoj stranici s bilo koje lokacije i bilo kojega uređaja. Ovakav izrađeni sustav za automatsko spremanje koda zajedno sa sustavom za prikaz promjena u programskome kodu bi mogli pomoći obrazovnim ustanovama u sprečavanju prepisivanja tuđeg programskog koda jer bi se vidjelo kako bi u nekim vremenskim intervalima naglo rastao broj dodanih linija, dok u drugima ne bi bilo ili bi bilo vrlo malo promjena u napisanim linijama.

Literatura

[1] Slika 2.1. Prikaz centraliziranih i distribuiranih sustava za kontrolu verzija, https://thedailywtf.com/articles/Source-Control-Done-Right [4.8.2019.]

[2] Slika 2.2. Subversion sustav za upravljanje verzijama izvornoga koda, https://www.ovh.ie/items/subversion.xml [29.8.2019.]

[3] Članak o sustavima za kontrolu verzija, https://pcchip.hr/softver/posao-i-financije/sustavi-za-upravljanje-verzijama-dokumenata-i-projekata/ [17.6.2019]

[4] Službena internetska stranica sustava kontrole verzija Git, https://git-scm.com/ [20.6.2019.]

[5] Opis GitHub platforme, https://en.wikipedia.org/wiki/GitHub [20.6.2019.]

[6] Službena internetska stranica JavaScript biblioteke HighCharts, https://www.highcharts.com/ [23.6.2019.]

[7] Slika 3.1. Postupak označavanja teksta za prikaz na internetskome pregledniku, https://internetingishard.com/html-and-css/basic-web-pages/ [5.8.2019.]

[8] Opis Subversion sustava za kontrolu verzija, https://hr.wikipedia.org/wiki/Subversion [19.6.2019.]

[9] Službena internetska stranica sustava kontrole verzija Subversion, https://subversion.apache.org/ [19.6.2019.]

[10] Opis Mercurial sustava za kontrolu verzija, https://en.wikipedia.org/wiki/Mercurial [19.6.2019.]

[11] Službena internetska stranica sustava kontrole verzija Mercurial, https://www.mercurial-scm.org/ [19.6.2019.]

[12] Opis Bazaar sustava za kontrolu verzija, https://en.wikipedia.org/wiki/GNU_Bazaar [19.6.2019.]

[13] Službena internetska stranica sustava kontrole verzija Bazaar, https://bazaar.canonical.com/en/ [19.6.2019.]

[14] Opis Git sustava za kontrolu verzija, https://hr.wikipedia.org/wiki/Git [20.6.2019.]

[15] Opis Git sustava za kontrolu verzija, https://en.wikipedia.org/wiki/Git [20.6.2019.]

[16] Opis skripti Windows operacijskog sustava, https://en.wikipedia.org/wiki/Batch_file [23.6.2019.]

[17] Opis skripti Windows operacijskog sustava, https://hr.if-koubou.com/articles/how-to/how-to-write-a-batch-script-on-windows.html [23.6.2019.]

[18] Opis HTML opisnog jezika za izradu internetskih stranica, https://hr.wikipedia.org/wiki/HTML [25.6.2019.]

[19] Opis JavaScript opisnog jezika za proširivanje funkcionalnosti internetskih stranica, https://en.wikipedia.org/wiki/JavaScript [25.6.2019.]

[20] Opis JavaScript biblioteke HighCharts, https://en.wikipedia.org/wiki/Highcharts [25.6.2019.]

[21] Opis razvojnog okruženja Visual Studio Code, https://en.wikipedia.org/wiki/Visual_Studio_Code [27.6.2019.]

[22] Opis HZML oznaka https://www.w3schools.com/tags/ [27.6.2019.]

[23] Opis skripti Linux operacijskog sustava, https://en.wikipedia.org/wiki/Shell_script [30.7.2019.]

[24] Opis Subversion sustava za kontrolu verzija, https://en.wikipedia.org/wiki/Apache_Subversion [29.8.2019.]

[25] Opis Bazaar sustava za kontrolu verzija, https://better-scm.shlomifish.org/alternatives/bazaar/ [29.8.2019.]

Sažetak

Objašnjeni su sustavi kontrole verzija. Detaljno su opisani sustavi Git, Subversion, Mercurial i Bazaar. Od tih sustava je odabran distribuirani sustav Git za izradu sustava za automatsko praćenje i spremanje programskog koda radi najveće proširenosti i opsežne dokumentacije. Kao alati izrade odabrane su skripte operacijskih sustava Windows i Linux radi direktne uporabe Git-a unutar njih te HTML i CSS za prikaz podataka radi njihove jednostavnosti i kompatibilnosti među uređajima. Izradom ovakvog sustava je prikazano kako Git kao sustav za kontrolu verzija može prikazati nastale promijene između svake verzije programskoga koda. Spremanjem tih podataka i ubacivanjem u internetski dokument uz korištenje vanjske biblioteke za iscrtavanje grafova grafički su prikazane promjene nastale između svake verzije koda. Na grafu su prikazane dodane i obrisane linije koda gdje se uspoređuje svaka verzija s početnom.

Ključne riječi: praćenje napretka programskog koda, Git, internetska stanica

Abstract

Title: Program code progress tracking system

Version control systems are described. Git, Subversion, Mercurial and Bazaar systems are described in detail. From these systems, a distributed Git system was chosen to build a code progress tacking system. It is widest spread system with detailed documentation. As a tool, Windows and Linux operating system scripts were picked because of their direct use of Git inside them and HTML and JavaScript were chosen for data printing because of their simplicity and compatibility across platforms. With development of this system it is presented how Git version control system can show changes between every pushed version of program code. By storing this data and adding them to web document along with using external library for plotting graphs, changes between all of the code versions are graphically presented. Number of added and deleted lines of code are shown on the graph, where every version compares to initial version.

Keywords: version control, Git, website

Životopis

Josip Blažević, rođen je u Osijeku 16.10.1997. godine. Prebivalište mu je u Josipovcu, prigradskom naselju grada Osijeka. Pohađao je osnovnu školu Josipovac, Elektrotehničku i prometnu školu Osijek smjer tehničar za računalstvo te sada pohađa Fakultet elektrotehnike, računarstva i informacijskih tehnologija preddiplomski sveučilišni studij računarstva. Osvojio je 2. mjesto na županijskom natjecanju Osječko-baranjske županije iz osnova informatike na Infokupu. Pohađao je tečaje Ljetne škole programiranja i Software startup academy. Uz brojne poslove koje je do sada obavljao, radio je kao demonstrator na fakultetu na kolegijima programiranje 1, programiranje 2, osnove elektrotehnike 1 i komunikacijskim mrežama, a trenutno radi kao programer u Igloo.

Potpis:

_____