



FILOZOFSKI FAKULTET

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

Odsjek za informacijske znanosti
Dvopredmetni diplomski studij informacijske tehnologije
Informacijsko komunikacijska infrastruktura

Seminarski rad na temu

Git

Mentor:

Doc. dr. sc. Tomislav Jakopec

Izradila:

Anja Labak

U Osijeku, 10.12.2021.

SADRŽAJ

SAŽETAK	1
1. UVOD.....	2
2. VERZIONIRANJE IZVORNOG KODA	3
3. GIT.....	6
3.1. POVIJEST GIT-A	6
3.2. GIT KAO ALAT ZA VERZIONIRANJE KODA.....	7
4. PRIMJER KORIŠTENJA GIT NAREDBI	8
4.1. KONFIGURACIJA GIT BASH-A I KREIRANJE REPOZITORIJA	8
4.1. KREIRANJE NADOPUNA	12
4.2. GRANANJE I SPAJANJE.....	15
4.3. VERZIONIRANJE KODA	16
4.4. ZAHTJEVI ZA GURANJE I POVLAČENJE.....	18
4.5 GIT CHERRY PICK.....	22
5. ZAKLJUČAK.....	24
LITERATURA	25

SAŽETAK

Seminarski rad bavi se alatom Git koji se koristi za kontrolu distribuiranih verzija izvornog koda. Cilj ovog rada je predstaviti alat Git pomoću njegovih naredbi. Rad je podijeljen u nekoliko cjelina, a počinje s definiranjem sustava za verzioniranje izvornog koda, te njihovom podjelom. U nastavku se daje povijesni pregled Git-a. Govori se o tome tko ga je osnovao i kada, iz kojeg razloga i u što se on razvio danas. Git kao alat je u širokoj uporabi, stoga će se u sljedećem poglavlju nabrojati njegove prednosti i za što se sve može koristiti. Zadnja cjelina ovoga rada je praktični dio. U praktičnom djelu prikazan je konkretan primjer korištenja raznih Git naredbi. Korištene su naredbe za kreiranje repozitorija, grananje i spajanje, te naredbe za zahtjeve povuci i poguraj. Kao zadnji primjer korištenja Git naredbi, odabrana je naredba Git Cherry Pick.

Ključne riječi: verzioniranje izvornog koda, sustavi za verzioniranje izvornog koda, Git, naredbe, nadopuna, grananje

1. UVOD

Cilj ovog rada je predstaviti alat Git za kontrolu distribuiranih verzija izvornog koda. Rad je podijeljen na dvije cjeline: teorijski dio o Git-u, te praktični dio u kojemu je prikazane naredbe Git-a na konkretnom primjeru.

Prvi dio teorijskog djela definira pojam verzioniranja izvornog koda. Govori o sustavima za verzioniranje izvornog koda, te kako se dijele. U nastavku govori se o razlici navedenih sustava, te kako zapravo funkcioniraju. Drugi dio rada govori o Git-u i njegovoj povijesti. U povijesnom pregledu Git-a saznajemo tko ga je osnovao i kada, koja je njegova prvobitna svrha i kako je nastala konačna verzija. Sljedeći dio govori o brojnim prednostima Git-a te zašto je on zapravo u širokoj upotrebi.

U praktičnom djelu seminarskog rada koristi se emulator Git Bash za izvedbu Git naredbi. Praktični dio podijeljen je na dio koji govori o konfiguraciji Git-a i kreiranju lokalnog repozitorija. U ovom djelu objašnjen je način instalacije Git-a, zašto se koristi Git Bash, kako stvoriti lokalni repozitorij i kako ga pratiti pomoću Git-a. Nakon toga slijede osnovne naredbe Git-a. Prva naredba koja je prikazana u radu je kreiranje nadopuna. Slijedeće naredbe su grananje i spajanje. Grananje i spajanje su jedna od brojnih prednosti Git-a, stoga su detaljno opisane u praktičnom djelu. Nadalje, u praktičnom djelu opisano je pomoću kojih naredbi se mogu vidjeti sve verzije koda te tehnika povuci i poguraj. Zadnji dio objašnjava što je to Cherrypicking model te kako se izvodi pomoću naredbi. Praktični dio popraćen je grafičkim prikazima kako bi se vidjele naredbe Git-a.

2. VERZIONIRANJE IZVORNOG KODA

Verzioriranje izvornog koda (eng. *Version Control*) je pojam koji se odnosi na sustav koji bilježi promijene u datoteci ili skupu datoteka tijekom vremena.¹ Sustav za verzioriranje koda (eng. *Version Control System*) koristi se za praćenje i upravljanje programskim kodom jer pomaže programerskim timovima da upravljaju promjenama izvornog koda tijekom vremena.² Dakle, on omogućuje vraćanje programskog koda ili projekta na prethodno stanje, uspoređivanje promjena tijekom vremena, uvid u zadnje izmjene na programskom kodu, tko je i kada uveo promijene i dr. Poznati sustavi za verzioriranje koda su CVS, SVN, Mercurial i Git.³ ⁴Sustav za verzioriranje koda dijelimo na dvije vrste sustava, a to su: centralizirani sustav i distribuirani sustav. Centralizirani sustav kontrole verzija (eng. *Centralized Version Control System*), dalje u tekstu CVCS, koristi središnji poslužitelj za pohranu svih datoteka, te omogućuje timsku suradnju. CVCS koristi jedan repozitorij pomoću kojeg korisnici mogu pristupiti poslužitelju. (Slika 1.) ⁵ Repozitorij koji se nalazi na vrhu hijerarhije predstavlja središnji poslužitelj. Središnji poslužitelj može biti lokalni ili udaljen koji je direktno povezan s radnom jedinicom programera. Svaka promjena koja se uvodi, uvodi se na glavnom repozitoriju. Nedostatak centraliziranog sustava kontrole verzija je što onemogućava rad izvan mreže. Kao što je već navedeno, postoji samo jedan glavni repozitorij, što znači ukoliko dođe do pada ili oštećenja središnjeg poslužitelja, dolazi i do gubitka cjelokupnih podataka projekta. Dakle, kod distribuiranih sustava kontrole verzija nema takvih nedostataka.

¹ Chacon, Scott; Straub, Ben. Git: About Version Control, 2014. URL: <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control> (2021-12-05)

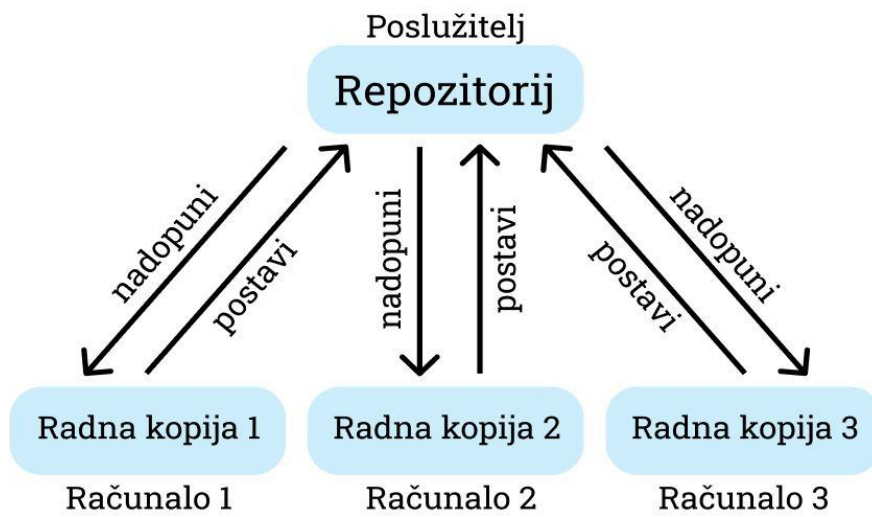
² What is version control. URL: <https://www.atlassian.com/git/tutorials/what-is-version-control> (2021-12-05)

³ Chacon, Scott; Straub, Ben. Nav. dj.

⁴ Version Control Software Comparison: Git, Mercurial, CVS, SVN. URL: <https://medium.com/@derya.cortuk/version-control-software-comparison-git-mercurial-cvs-svn-21b2a71226e4> (2021-12-05)

⁵ Chacon, Scott; Straub, Ben. Nav. dj.

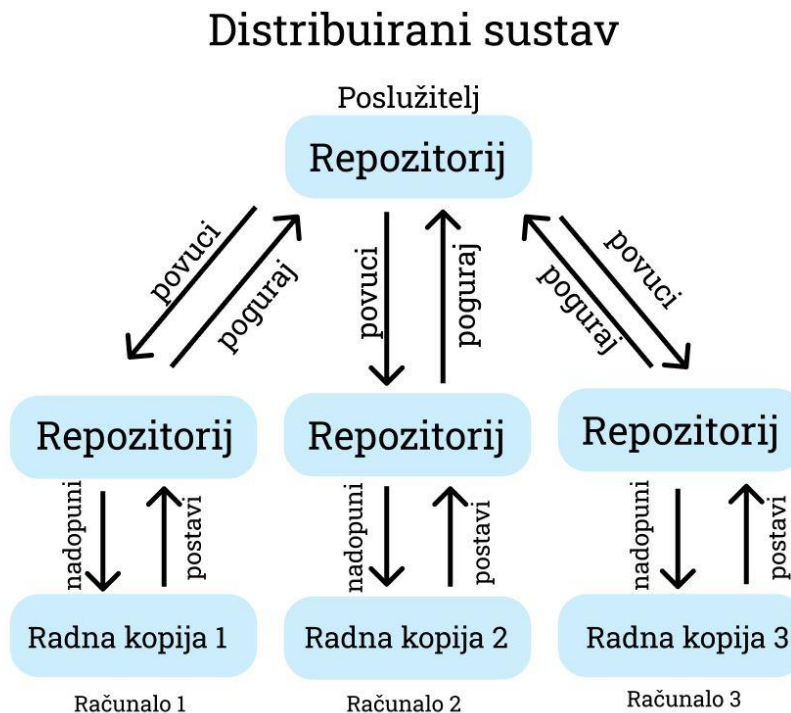
Centralizirani sustav



Slika 1. Centralizirani sustav za verzioniranje koda⁶

⁶ What is Git. URL: <https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2016/11/Centralized-Version-Control-System-Workflow-What-Is-Git-Edureka.png> (2021-12-05)

Distribuirani sustav za verzioniranje koda (eng. Distributed version control system), dalje u tekstu DVCS, je sustav koji se ne oslanja samo na središnjeg poslužitelja prilikom pohrane svih verzija programskog koda, već svaki suradnik ima lokalni repozitorij ili tzv. „klon“ glavnog repozitorija. Time se postiže da je programski kod distribuiran na više repozitorija i glavni repozitorij.⁷ (Slika 2.)



Slika 2. Distribuirani sustav za verzioniranje koda⁸

Dakle, distribuirani sustav omogućava ažuriranje kopija repozitorija s novim podacima sa središnjeg servera s naredbama *povuci* (eng. *pull*). S druge strane, ukoliko se želi utjecati na središnji repozitorij s lokalnog repozitorija, koristi se naredba *poguraj* (eng. *push*). Prednosti su rad izvan mreže, te nadopune se mogu dodati lokalno bez da se upravlja podacima iz glavnog

⁷ What is Git. URL: <https://www.edureka.co/blog/what-is-git/> (2021-12-05)

⁸ What is Git. URL: <https://dl.jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2016/11/Distributed-Version-Control-System-Workflow-What-Is-Git-Edureka-768x508.png> (2021-12-05)

repozitorija. Također, ukoliko dođe do pada središnjeg poslužitelja, podatci se zadržavaju na lokalnim repozitorijima suradnika.⁹

3. GIT

U nastavku seminarskog rada govori se o povijesti Git-a, što je Git, te koju se njegove prednosti. Na 3. slici možemo vidjeti logo Git-a.



Slika 3. Git logo

3.1. POVIJEST GIT-A

Git je razvio Linus Torvalds 2005. godine kao osnovu drugim sustavima za razvijanje programskog koda. Linus Torvalds je prvobitno razvio Git zbog Linux-a.¹⁰ Linux je operativni sustav koji se počeo razvijati sredinom 90-ih godina 20. stoljeća.¹¹ U početnoj fazi održavanja Linux-a, promijene na programu prenosile su se kao zakrpe i arhivirane datoteke. Od 2002. godine, za potrebe održavanja Linuxa počinje se koristiti DVCS BitKeeper. Godine 2005., Linus Torvalds prestaje koristiti BitKeeper što ga potiče da razvije vlastiti alat na temelju značajki BitKeeper-a. Cilj mu je bio razviti brzi alat, s jednostavnim korisničkim sučeljem, koji podržava nelinearni razvoj projekata. Također, htio je razviti distribuirani alat na kojemu se mogu razvijati složeni

⁹ What is Git. URL: <https://www.edureka.co/blog/what-is-git/> (2021-12-05)

¹⁰ Krajina, Tomo. Uvod u Git. URL: <https://tkrajina.github.io/uvod-u-git/git.pdf> (2021-12-05)

¹¹ What is Linux. URL: <https://www.linux.com/what-is-linux/> (2021-12-05)

projekti poput Linux-a. Od svog početka, Git se razvio u jednostavan alat za korištenje, međutim zadržao je svoje početne kvalitete poput brzine, učinkovitosti i sustava grananja za nelinearni razvoj projekata.¹²

3.2. GIT KAO ALAT ZA VERZIONIRANJE KODA

Git je alat za kontrolu distribuiranih verzija izvornog koda. On podržava distribuirane nelinearne tokove rada tako što pruža osiguranje podatka za razvoj kvalitetnih projekata.¹³ To znači da je lokalni repozitoriji kopija centralnog repozitorija kontrole verzija. Prednost potpuno funkcionalnih lokalnih repozitorija je što omogućavaju rad na daljinu ili izvan mreže. Što znači, kada klijenti rade na svom kodu, predaju ga lokalno. Nakon predaje, klijenti sinkroniziraju svoju kopiju repozitorija s kopijom na poslužitelju. S druge strane, kada bi Git podržavao centralizirani sustav, tada bi klijent morali sinkronizirati kod s poslužiteljem prije nego što stvore novu verziju koda.¹⁴

Zbog svojih brojnih prednosti, Git je postao standard za verzioniranje izvornog koda. Velika prednost Gita je što je on besplatan alat u otvorenom pristupu.¹⁵ Odlikuje ga brzina jer ne zahtjeva povezivanje na mrežu prilikom korištenja. Fleksibilan je, što znači da bezobzira kolika se količina podataka pohranjuje na Git, on će se toj količini podataka i prilagoditi. Prilagodba velikoj količini podataka je moguća zbog smanjenja veličine podataka sa strane klijenta bez ikakvog gubitka. Zahvaljujući lokalnim repozitorijima koji svaki klijent posjeduje, u slučaju pada Git sustava, uvijek postoji mogućnost vraćanja izgubljenih podataka iz bilo kojeg lokalnog repozitorija. Drugim riječima, uvijek će postojati sigurnosna kopija podataka.¹⁶ Nadalje, ukoliko se Git kombinira s alatom za verzioniranje izvornog koda, primjerice GitHub, povećava se produktivnost tima tako što se potiče suradnja, automatiziraju se procesi, te se poboljšava vidljivost i slijed rada. Git-a omogućuje korištenje zahtjeva za povlačenjem (*eng. pull request*) za raspravljanje o promjenama na programskom kodu s timom prije nego što se postave na glavnu

¹² Chacon, Scott; Straub, Ben. Nav. dj.

¹³ Ahmed, Reshma. What is Git, 2020. URL:<https://www.edureka.co/blog/what-is-git/> (2021-12-05)

¹⁴ Jacobs, Mike; Kaim, Ed. What is Git, 2021. URL:<https://docs.microsoft.com/en-us/devops/develop/git/what-is-git> (2021-12-05)

¹⁵ Jacobs, Mike; Kaim, Ed. Nav. dj.

¹⁶ Ahmed, Reshma. Nav. dj.

granu(*eng. main branch*). Korištenje zahtjeva za povlačenjem predstavlja veliku prednost jer omogućava uvid u promjenu datoteka, ostavljanje komentara, pregledavanje nadopuna(*eng. commit*) i nadogradnji(*eng. build*) te na kraju odobravanje programskog koda na temelju glasovanja.¹⁷

4. PRIMJER KORIŠTENJA GIT NAREDBI

4.1. KONFIGURACIJA GIT BASH-A I KREIRANJE REPOZITORIJA

U zadnjem djelu seminarskog rada prikazan je konkretan primjer korištenja Git-a. Prvi korak je preuzimanje datoteke za instalaciju sa službene stranice Git-a. Trenutna verzija za operativni sustav Windows 10 je Git 2.34.1. Uz instalaciju Git-a, instalirat će se i Git Bash (*eng. Bourne Again Shell*). Git Bash je emulator za Microsoft Windows okruženja koji se koristi za pokretanje Git-a iz naredbenog retka.¹⁸ (Vidi Slika 4.)



Slika 4. Git Bash emulator


¹⁷ Jacobs, Mike; Kaim, Ed. Nav. dj.

¹⁸Git Bash.

URL:<https://www.atlassian.com/git/tutorials/git-bash#:~:text=What%20is%20Git%20Bash%3F,operating%20system%20through%20written%20commands> (2021-12-08)

Kako bi koristili Git, prvo se moramo identificirati putem naredbenog retka pomoću sljedećih naredbi: `git config --global user.name "Vaše ime"` i `git config --global user.email vas@email.com`. (Slika 5.)

Identifikacija je neophodan korak jer pomoću nje možemo vidjeti tko je radio promijene na datotekama koje se prate pomoću Git-a.

A screenshot of a Windows command prompt window titled "MINGW64/c/Users/anja0". The window has a red title bar and a black background. The text inside shows three lines of commands being entered at a prompt. The first line is "anja0@DESKTOP-980S7VK MINGW64 ~ (master)" followed by "\$ git config --global user.name 'Anja Labak'". The second line is "anja0@DESKTOP-980S7VK MINGW64 ~ (master)" followed by "\$ git config --global user.email anja.labak12345@gmail.com". The third line is "anja0@DESKTOP-980S7VK MINGW64 ~ (master)" followed by "\$" and a cursor. There is a vertical scrollbar on the right side of the window.

```
anja0@DESKTOP-980S7VK MINGW64 ~ (master)
$ git config --global user.name "Anja Labak"

anja0@DESKTOP-980S7VK MINGW64 ~ (master)
$ git config --global user.email anja.labak12345@gmail.com

anja0@DESKTOP-980S7VK MINGW64 ~ (master)
$
```

Slika 5. Identifikacija na Git Bash-u

Sljedeća slika prikazuje izradu repozitorija i datoteka koja će se koristiti kako bi se prikazao primjer verzioniranja koda. (Slika 6.) Pomoću naredbe `pwd` dobiva se uvid koji se točno repozitorij prati. Naredbom `cd /c/` premjestili smo se u local disk c gdje kreiramo repozitorij pomoću naredbi `mkdir seminar_git`. Naredbom `cd /c/seminar_git` premještamo se u repozitorij koji koristimo za ovaj primjer te kreiramo datoteke `index.html` i `style.css` pomoću naredbi `touch index.html` i `touch style.css`.



```
MINGW64:/c/seminar_git
anja0@DESKTOP-980S7VK MINGW64 ~ (master)
$ git config --global user.name "Anja Labak"

anja0@DESKTOP-980S7VK MINGW64 ~ (master)
$ git config --global user.email anja.labak12345@gmail.com

anja0@DESKTOP-980S7VK MINGW64 ~ (master)
$ pwd
/c/Users/anja0

anja0@DESKTOP-980S7VK MINGW64 ~ (master)
$ cd /c/

anja0@DESKTOP-980S7VK MINGW64 /c
$ pwd
/c

anja0@DESKTOP-980S7VK MINGW64 /c
$ mkdir seminar_git

anja0@DESKTOP-980S7VK MINGW64 /c
$ cd /c/seminar_git/

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git
$ pwd
/c/seminar_git

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git
$ touch index.html

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git
$ touch style.css

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git
$
```

Slika 6. Kreiranje datoteka index.html i style.css pomoću Git naredbi u Git Bash-u

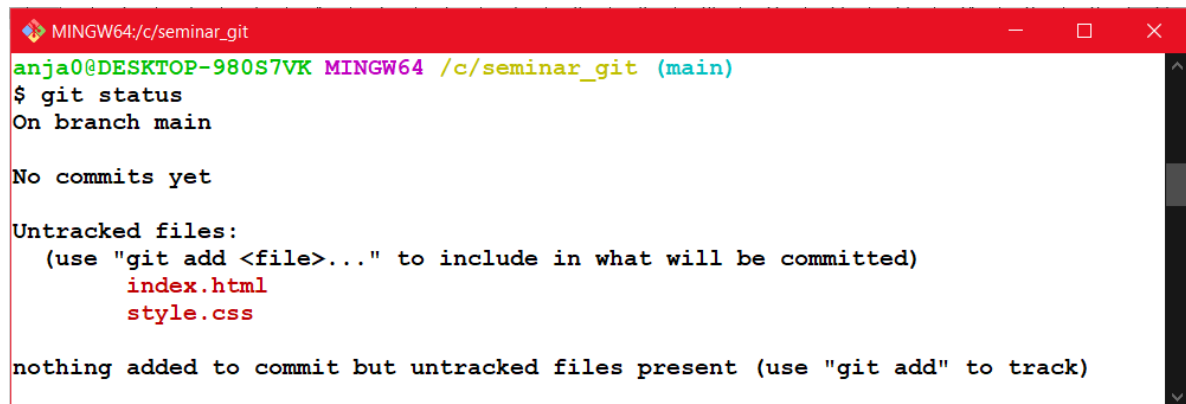
Nakon što smo kreirali repozitorij i datoteke u koje će se ispisivati programski kod pomoću naredbe `git init` počinjemo pratiti repozitorij s Git-om. (Slika 7.)¹⁹



```
MINGW64:/c/seminar_git
git init
Initialized empty Git repository in C:/seminar_git/.git/
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$
```

Slika 7.

Naredba `git status` daje nam uvid u stanje našeg repozitorija na terminalu. Na slici 8, dobivamo uvid na kojoj se grani nalazimo, postoje li nove nadopune i koje datoteke pratimo.



```
MINGW64:/c/seminar_git
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html
        style.css

nothing added to commit but untracked files present (use "git add" to track)
```

Slika 8.

¹⁹Introduction to Git. URL: <https://www.notion.so/zarkom/Introduction-to-Git-ac396a0697704709a12b6a0e545db049> (2021-12-08)

4.1. KREIRANJE NADOPUNA



```
MINGW64/c/seminar_git
git add index.html style.css

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   index.html
        new file:   style.css

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$
```

Slika 9. Dodavanje datoteka za praćenje

Pomoću naredbe `git add` dodajemo datoteke koje želimo pratiti putem Git-a. (Slika 9.) Naredbom `git commit -m "Praćenje datoteka index.html i style.css"` dodajemo nadopunu. Dodavanje nadopuna je neophodan korak prilikom praćenja programskog koda jer nam daje uvid u sve izmijene na datotekama u prošlosti. Zahvaljujući nadopunama koje generiraju jedinstveni identifikator nadopune (eng. *commit hash*), uvijek postoji mogućnost vraćanja na tu verziju koda.



```
MINGW64/c/seminar_git
$ git add index.html style.css

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git commit -m "Praćenje datoteka index.html i style.css"
[main (root-commit) 945e618] Praćenje datoteka index.html i style.css
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
 create mode 100644 style.css

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git status
On branch main
nothing to commit, working tree clean

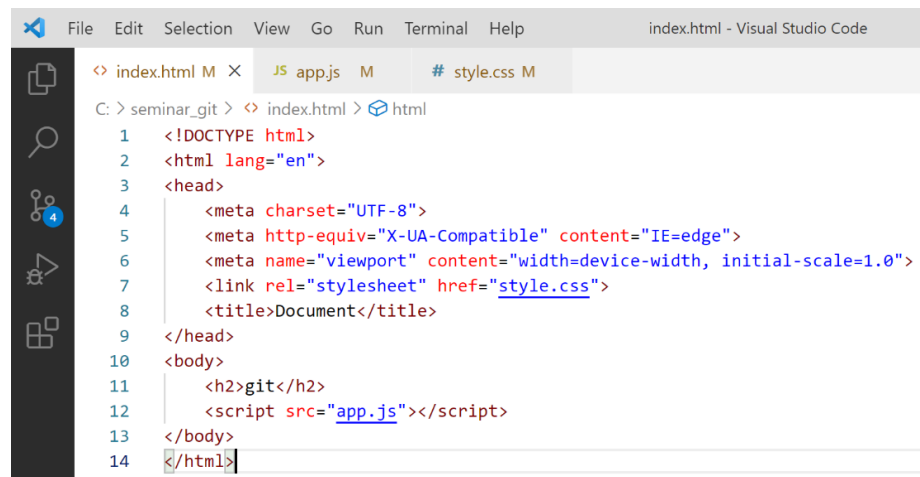
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git log
commit 945e61861fd713f62e92477d578adbbel6e77012 (HEAD -> main)
Author: Anja Labak <anja.labak12345@gmail.com>
Date:   Wed Dec 8 20:43:27 2021 +0100

    Praćenje datoteka index.html i style.css

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$
```

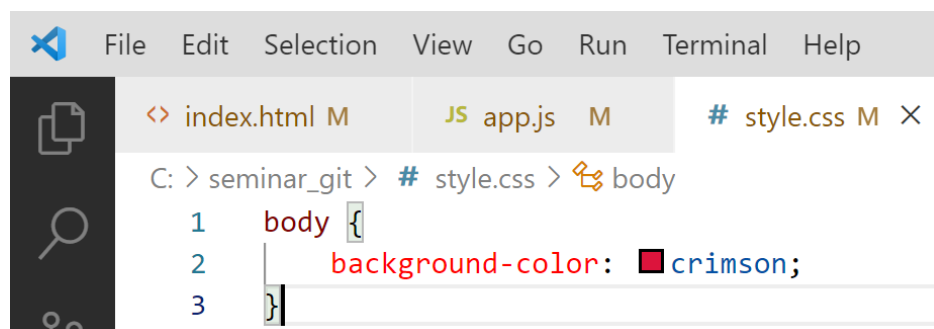
Slika 10. Kreiranje prve nadopune

U datotekama index.html, style.css i app.js napravljene su izmjene na programskom kodu kako bi u sljedećem primjeru pokazali kako napraviti više nadopuna odjednom.(Slika 11-13.)



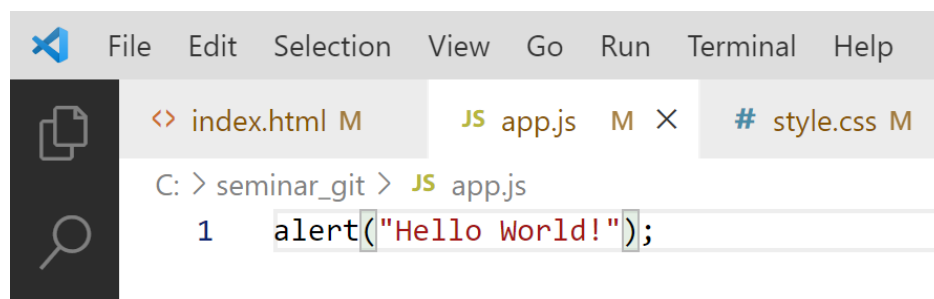
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <link rel="stylesheet" href="style.css">
8   <title>Document</title>
9 </head>
10 <body>
11   <h2>git</h2>
12   <script src="app.js"></script>
13 </body>
14 </html>
```

Slika 11. Izmijene na programskom kodu u datoteci index.html



```
1 body {
2   background-color: crimson;
3 }
```

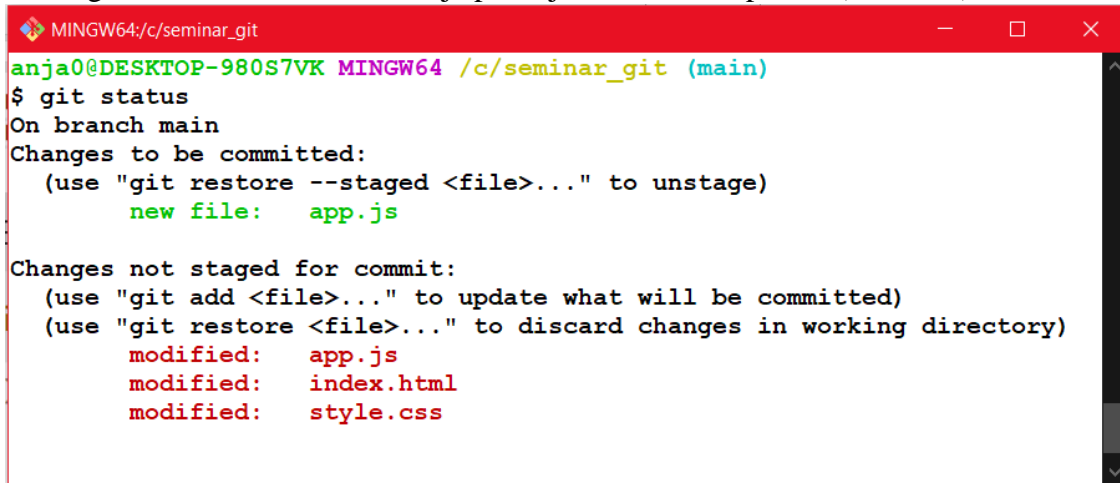
Slika 12. Izmijene na programskom kodu u datoteci style.css



```
1 alert("Hello World!");
```

Slika 13. Izmijene na programskom kodu u datoteci app.js

Naredbom `git status` dobivamo uvid koje promijene treba nadopuniti. (Slika 14.)



```
MINGW64:/c/seminar_git
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   app.js

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   app.js
        modified:   index.html
        modified:   style.css
```

Slika 14. Izmijene na programskom kodu za nadopunu

Ukoliko želimo napraviti nadopunu na sve datoteke, naredbom `git add index.html style.css app.js` označit ćemo datoteke koje želimo nadopuniti. Nakon toga naredbom `git commit -m` napisat ćemo nadopunu za označene datoteke. (Slika 15.)



```
MINGW64:/c/seminar_git
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   app.js
        modified:   index.html
        modified:   style.css

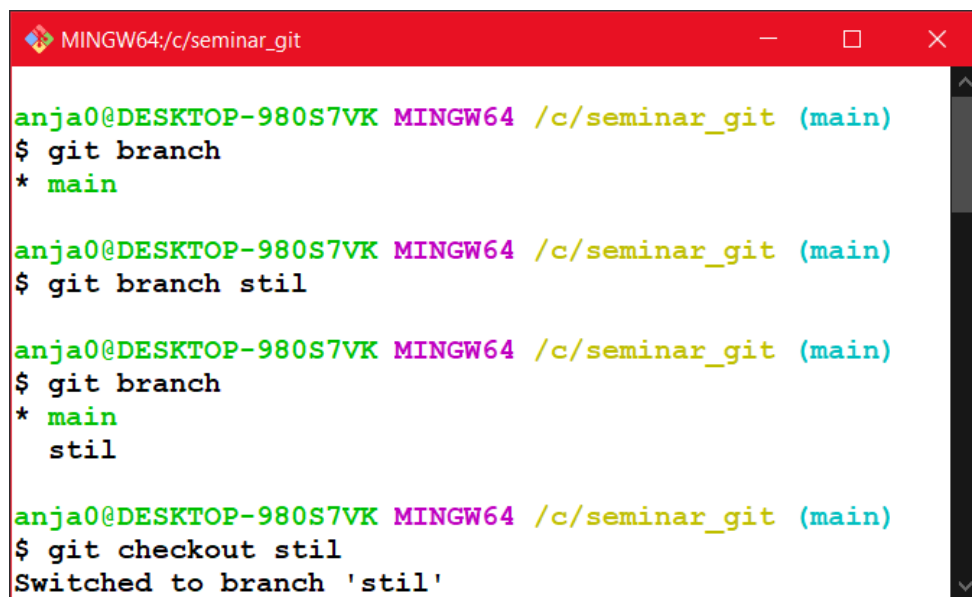
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git commit -m "U app.js je dodano upozorenje. U style.css je promjenjena
  boja pozadine. U index.html je dodan naslov i poveznice na style.css i ap
  p.js"
[main d35e9ba] U app.js je dodano upozorenje. U style.css je promjenjena b
  oja pozadine. U index.html je dodan naslov i poveznice na style.css i app.
  js
 3 files changed, 19 insertions(+)

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$
```

Slika 15. Nadopuna datoteka index.html, style.css i app.js

4.2. GRANANJE I SPAJANJE

Grane(eng. *branch*) predstavljaju samostalnu liniju razvoja programskog koda. Git naredbe branch koriste se za kreiranje, listanje, preimenovanje i brisanje grana. Grananje (eng. *branching*) omogućuje svakom razvojnom programeru da se odvoji od glavne grane te da odvojeno radi na vlastitoj grani bez obzira na veličinu promjena. Proces odvajanja od glavne grane omogućava pravljenje pogrešaka, ispravljanje pogrešaka, testiranje novih značajki itd. Grananjem se osigurava da glavna grana održi svoju kvalitetu.²⁰ Na slici 16. prikazan je primjer korištenja git branch naredbi. Naredba git branch pokazuje granu koja se trenutno koristi. Naredbom git branch stil dodana je grana stil. Naredbom git checkout stil izvršava se prebacivanje na granu stil.



```
MINGW64:/c/seminar_git

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git branch
* main

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git branch stil

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git branch
* main
  stil

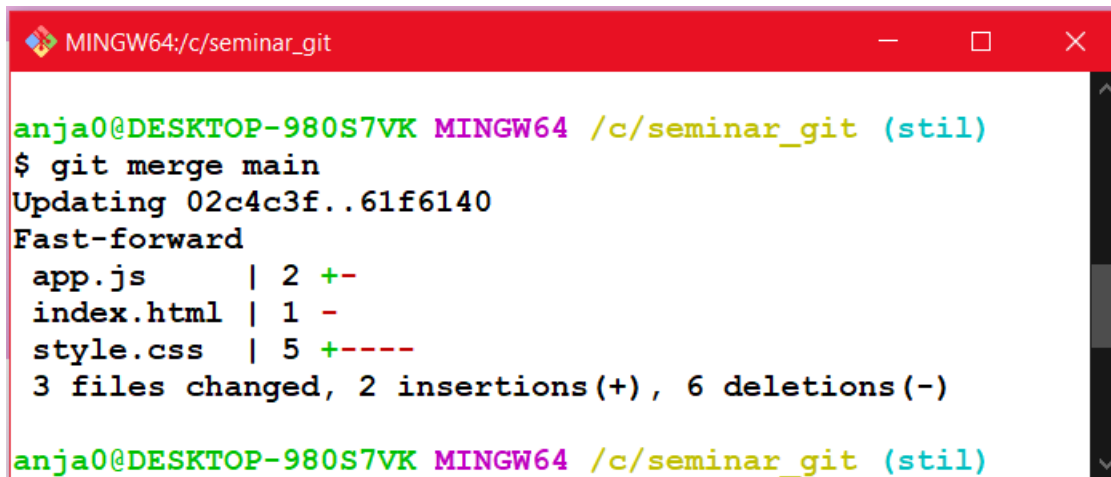
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git checkout stil
Switched to branch 'stil'
```

Slika 16. Naredbe za prikaz svih grana i kreiranje novih grana

U nastavku, na grani stil u datoteci style.css promijenjena je boja pozadine. Ova promjena neće utjecati na glavnu granu. Međutim, ukoliko se vraćamo na glavnu granu, ta promjena se neće vidjeti. Ukoliko želimo vidjeti tu promjenu na glavnoj grani potrebno je spojiti glavnu granu sa granom stil.

²⁰Why Git for your organization. URL: <https://www.atlassian.com/git/tutorials/why-git> (2021-12-09)

Spajanje(eng. *merge*) je naredba Git-a koja se implementira ukoliko želimo da se promjene s programskog koda jedne grane spoje na drugu granu. Primjerice, kada se na sporednoj grani razvije i testira nova značajka koja se želi implementirati u glavnu granu. Za spajanje grana koristi se naredba `git merge <naziv grane>`.²¹ (Slika 17.)



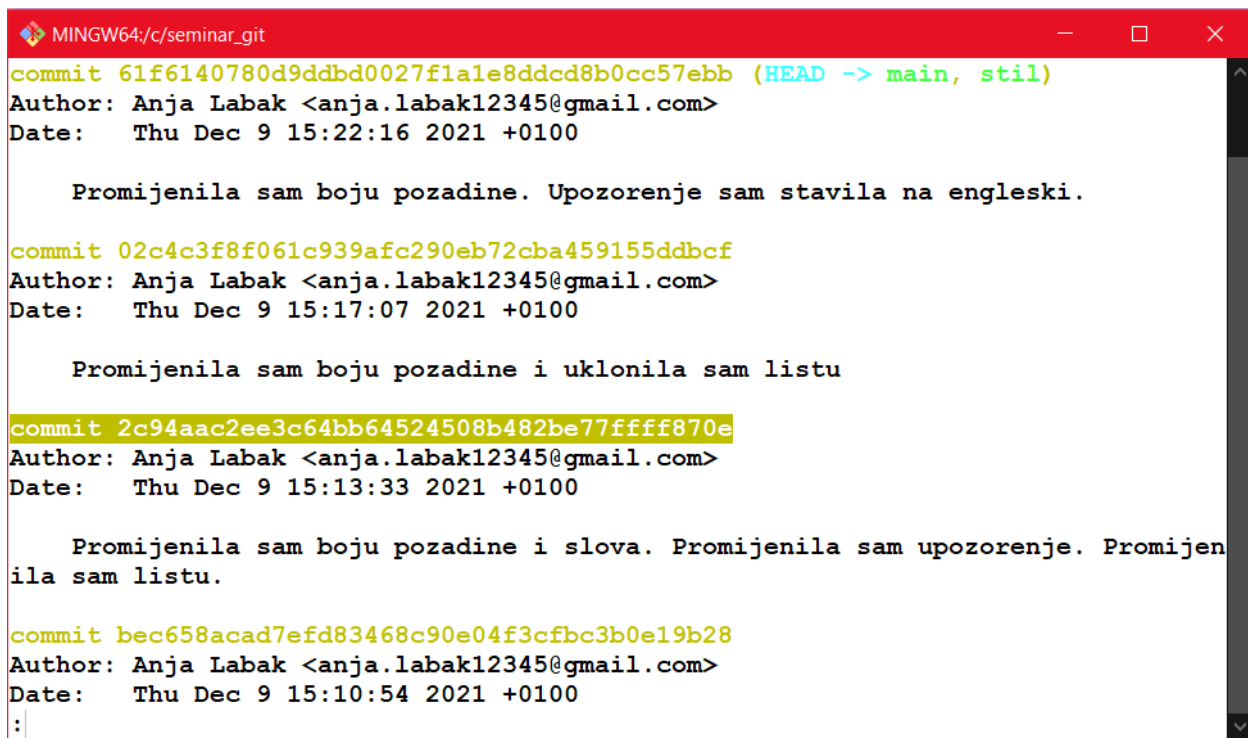
```
MINGW64:/c/seminar_git
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (stil)
$ git merge main
Updating 02c4c3f..61f6140
Fast-forward
 app.js      | 2 +-
 index.html  | 1 -
 style.css   | 5 +---
 3 files changed, 2 insertions(+), 6 deletions(-)
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (stil)
```

Slika 17. Spajanje grana pomoću naredbe `git merge <naziv grane>`

4.3. VERZIONIRANJE KODA

Prilikom kreiranja nadopuna, uz opis nadopune generira se jedinstveni identifikator nadopune(eng. *commit hash*). Ukoliko se želi dobiti uvid u prošlost izmjena na programskom kodu, koristi se naredba `git checkout <jedinstveni-identifikator-nadopune>`. (Slika 18.) Ta naredba omogućuje uvid u svaku verziju koda.

²¹Introduction to Git. URL: <https://www.notion.so/zarkom/Introduction-to-Git-ac396a0697704709a12b6a0e545db049> (2021-12-08)



```
MINGW64/c/semnar_git
commit 61f6140780d9ddbd0027f1a1e8ddcd8b0cc57ebb (HEAD -> main, stil)
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:22:16 2021 +0100

    Promijenila sam boju pozadine. Upozorenje sam stavila na engleski.

commit 02c4c3f8f061c939afc290eb72cba459155ddbcbf
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:17:07 2021 +0100

    Promijenila sam boju pozadine i uklonila sam listu

commit 2c94aac2ee3c64bb64524508b482be77ffff870e
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:13:33 2021 +0100

    Promijenila sam boju pozadine i slova. Promijenila sam upozorenje. Promijenila sam listu.

commit bec658acad7efd83468c90e04f3cfbc3b0e19b28
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:10:54 2021 +0100

:
```

Slika 18. Jedinstveni identifikator nadopune

4.4. ZAHTJEVI ZA GURANJE I POVLAČENJE

U ovom djelu rada obradit ćemo zahtjeve za guranje i povlačenje(eng. *push and pull requests*). Naredba za guranje(eng. *git push*) služi za prijenos sadržaja iz lokalnog repozitorija u udaljen repozitorij. Guranje je način na koji se prenose i nadopune iz lokalnog repozitorija u udaljeni repozitorij.²²

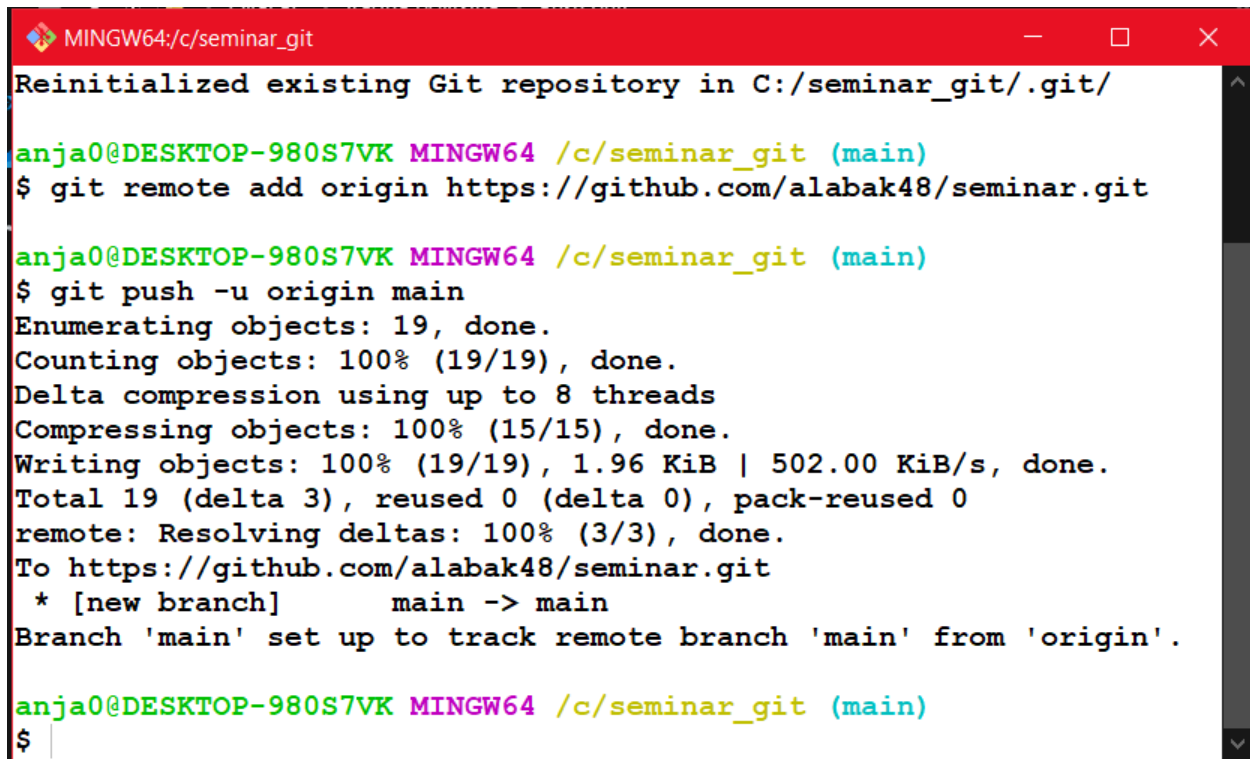
Naredba za povlačenje(eng. *git pull*) koristi se za dohvaćanje i preuzimanje sadržaja iz udaljenog repozitorija kao i trenutno ažuriranje lokalnog repozitorija i sadržaja. Nakon što se izvede naredba za povlačenje, potrebno je koristiti naredbu merge(eng. *spajanje*). Dakle, naredba za povlačenje zapravo čini kombinaciju naredbi dohvati(eng. *git fetch*) i naredbe spoji. Razlika između naredbe dohvati i naredbe guranje je u tome što se naredba dohvati koristi samo za lokalne repozitorije, a naredba za guranje se koristi za izvoz nadopuna na udaljene grane.

U nastavku, prikazan je primjer provedbe zahtjeva za guranje i povlačenje. Prije svega, potrebno je kreirati udaljen repozitorij. U ovom primjeru, repozitorij je kreiran na GitHub-u. (slika)

Slika 19. Kreiranje repozitorija na GitHub-u

²² Prajapati, Ameet. What is git commit, push, pull, log, aliases, fetch, config & clone, 2019. URL: <https://medium.com/mindorks/what-is-git-commit-push-pull-log-aliases-fetch-config-clone-56bc52a3601c>(2021-12-08)

Nakon kreiranja repozitorija na GitHub-u, potrebno je povezati lokalni repozitorij sa udaljenim repozitorijem. Za povezivanje repozitorija potrebne su sljedeće naredbe: `git remote add origin https://github.com/alabak48/seminar.git` i `git push -u origin master`. (Slika 20.)



```
MINGW64:/c/seminar_git
Reinitialized existing Git repository in C:/seminar_git/.git/

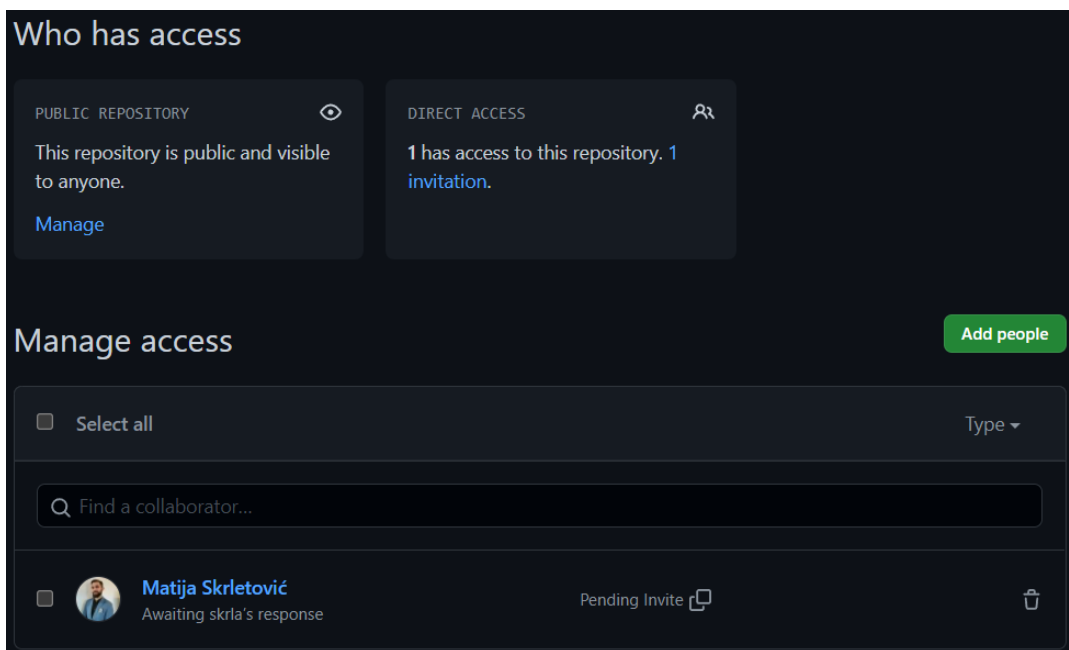
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git remote add origin https://github.com/alabak48/seminar.git

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git push -u origin main
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (19/19), 1.96 KiB | 502.00 KiB/s, done.
Total 19 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/alabak48/seminar.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

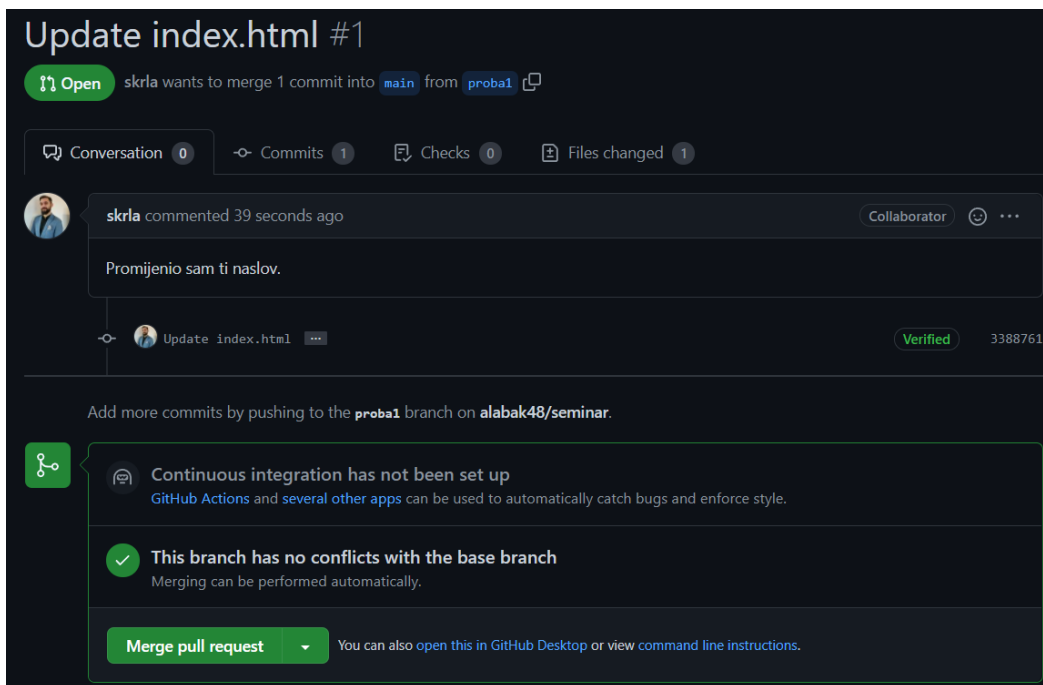
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$
```

Slika 20. Povezivanje lokalnog repozitorija i udaljenog repozitorija

Nakon što smo povezali repozitorije, potrebno je poslati zahtjev za suradnju na projektu. (Slika 21.) Kada je zahtjev prihvaćen, tada suradnik može raditi promijene na projektu. Na slici 22. zahtjev za povlačenje od strane suradnika koji možemo spojiti.

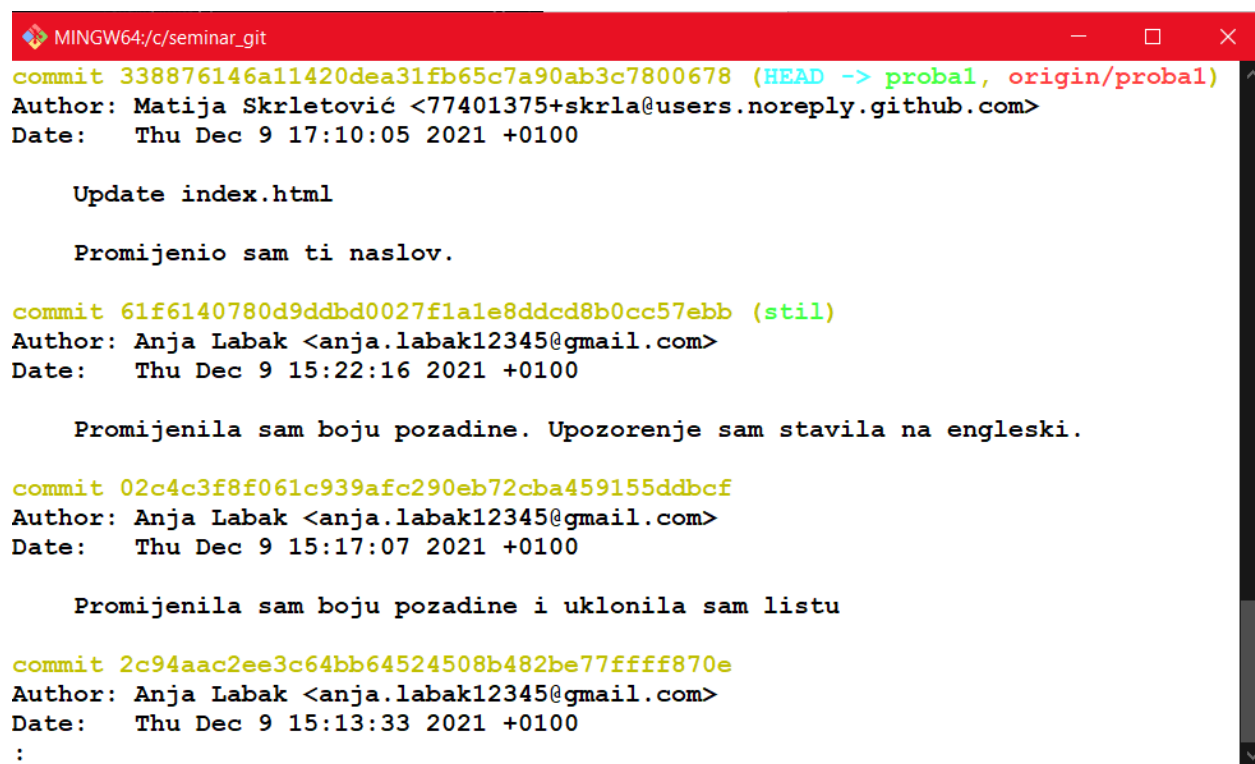


Slika 21. Slanje zahtjeva za suradnju na projektu



Slika 22. Zahtjev za povlačenje od strane suradnika

Spajanje zahtjeva za povlačenjem može se izvršiti jednostavnim klikom na „Merge pull request“. Međutim, GitHub nam omogućuje i popis naredbi koje možemo ispisati u Git-u. Naredbe su sljedeće: `git fetch origin git checkout -b proba1 origin/proba1 git merge main git checkout main git merge --no-ff proba1 git push origin main`. Nakon izvršenog spajanja, potrebno je izvršiti naredbu `git pull` koja će povući sve promijene na udaljenom repozitoriju. Pošto je nadopuna bila na grani `proba1`, potrebno je prebaciti se na granu `proba1` i upisati git log naredbu kako bi dobili ispis izmjena na datoteci. Ukoliko to želimo, granu `proba1` možemo spojiti s granom `main`. Na grani `proba1` izmijenjen je naslov u `index.html` (Slika 23.)



```
MINGW64:/c/seminar_git
commit 338876146a11420dea31fb65c7a90ab3c7800678 (HEAD -> proba1, origin/proba1)
Author: Matija Skrletović <77401375+skrla@users.noreply.github.com>
Date: Thu Dec 9 17:10:05 2021 +0100

    Update index.html

    Promijenio sam ti naslov.

commit 61f6140780d9ddbd0027f1a1e8ddcd8b0cc57ebb (stil)
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:22:16 2021 +0100

    Promijenila sam boju pozadine. Upozorenje sam stavila na engleski.

commit 02c4c3f8f061c939afc290eb72cba459155ddbcf
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:17:07 2021 +0100

    Promijenila sam boju pozadine i uklonila sam listu

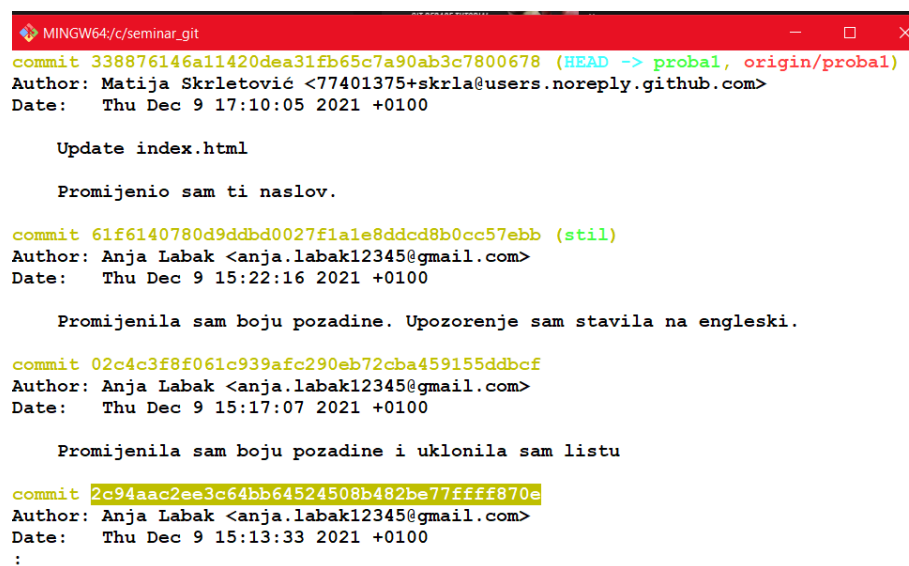
commit 2c94aac2ee3c64bb64524508b482be77ffff870e
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:13:33 2021 +0100
:
```

Slika 23. Nadopuna na udaljenom repozitoriju od strane suradnika

4.5 GIT CHERRY PICK

Naredba `git cherry-pick` omogućuje odabir nadopune iz jedne grane i premještane te nadopune na drugu granu. Često se koristi za poništavanje promjena i vraćanje izgubljenih nadopuna. Primjerice, ukoliko se nadopuna ispiše na krivoj grani, naredbom `git cherry-pick` nadopunu možemo prenijeti na ispravnu granu, te ju smjestiti na odgovarajuće mjesto. Git Cherry Pick omogućuje prijenos jedne ili više nadopuna s jedne grane na drugu. Problem Git Cherry Pick-a je što ponekad može napraviti duple nadopune. Također, umjesto ove naredbe, puno češće se koristi tradicionalno spajanje. Često se koristi kod timske suradnje kada više suradnika radi na istom programskom kodu. Nadalje, koristi se kod rješavanja problema u programskom kodu kada je potrebno u vrlo kratkom vremenu ponuditi rješenje.²³

U nastavku, prikazan je primjer korištenja naredbe `git cherry-pick`. U ovom primjeru prikazuje se korištenje `git cherry-pick` naredba u slučaju kada nadopunu iz jedne grane prebacujemo na drugu granu. Iz grane `proba1` kopiran je jedinstveni identifikator nadopune. (Slika 24.)



```
MINGW64/c/seminar_git
commit 338876146a11420dea31fb65c7a90ab3c7800678 (HEAD -> proba1, origin/proba1)
Author: Matija Skrletović <77401375+skrla@users.noreply.github.com>
Date: Thu Dec 9 17:10:05 2021 +0100

    Update index.html

    Promijenio sam ti naslov.

commit 61f6140780d9ddbd0027f1a1e8ddcd8b0cc57ebb (stil)
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:22:16 2021 +0100

    Promijenila sam boju pozadine. Upozorenje sam stavila na engleski.

commit 02c4c3f8f061c939afc290eb72cba459155ddbcbf
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:17:07 2021 +0100

    Promijenila sam boju pozadine i uklonila sam listu

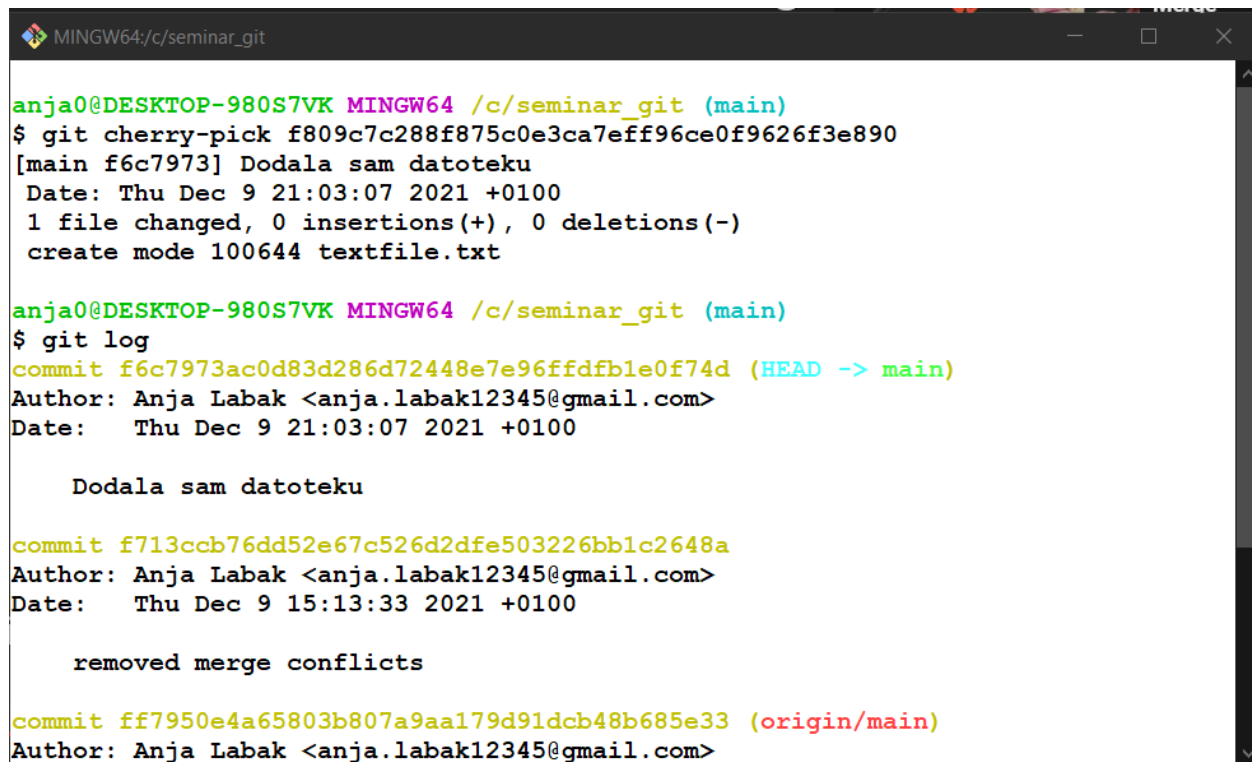
commit 2c94aac2ee3c64bb64524508b482be77ffff870e
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:13:33 2021 +0100

:
```

Slika 24. Jedinstveni identifikator nadopune

²³ Git Cherry Pick. URL:<https://www.atlassian.com/git/tutorials/cherry-pick#:~:text=git%20cherry%2Dpick%20is%20a,be%20useful%20for%20undoing%20changes> (2021-12-09)

Naredbom git checkout main vračamo se na glavnu granu. Naredbom git cherry-pick <jedinstveni identifikator nadopune> prebacujemo nadopunu iz grane proba1 na glavnu granu. (Slika 25.)

A screenshot of a terminal window titled 'MINGW64:/c/seminar_git'. The terminal shows the execution of 'git cherry-pick' followed by 'git log'. The cherry-pick command successfully adds a file 'textfile.txt' to the main branch. The log shows three commits: the first by Anja Labak adding the file, the second by the same author removing merge conflicts, and the third which is the current commit (HEAD -> main) resulting from the cherry-pick operation.

```
anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git cherry-pick f809c7c288f875c0e3ca7eff96ce0f9626f3e890
[main f6c7973] Dodala sam datoteku
Date: Thu Dec 9 21:03:07 2021 +0100
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 textfile.txt

anja0@DESKTOP-980S7VK MINGW64 /c/seminar_git (main)
$ git log
commit f6c7973ac0d83d286d72448e7e96ffdfb1e0f74d (HEAD -> main)
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 21:03:07 2021 +0100

    Dodala sam datoteku

commit f713ccb76dd52e67c526d2dfe503226bb1c2648a
Author: Anja Labak <anja.labak12345@gmail.com>
Date: Thu Dec 9 15:13:33 2021 +0100

    removed merge conflicts

commit ff7950e4a65803b807a9aa179d91dcb48b685e33 (origin/main)
Author: Anja Labak <anja.labak12345@gmail.com>
```

Slika 25. Korištenje naredbe git-cherry pick

5. ZAKLJUČAK

Cilj ovog rada bio je predstaviti Git. Također, htjeli smo prikazati konkretan primjer korištenja Git naredbi na programskom kodu. Git je jednostavan alat za korištenje u otvorenom pristupu. Verzioniranje izvornog koda pomoću nadopuna čini se jednostavnim, te može uvelike olakšati rad na projektima među timovima ili suradnicima. Pomoću nadopuna možemo vidjeti povijest pisanja programskog koda. To nosi brojne prednosti sa sobom. Primjerice, uvijek se možemo vratiti na prethodno stanje ukoliko je došlo do pogreške ili možemo se vratiti na nadopunu u kojoj je greška napravljena. Kao najveću prednost korištenja Git-a izdvojila bih grananje jer omogućava rad na jednom projektu bez ometanja suradnika. Također, grananje omogućava individualan rad na programskom kodu, stoga pravljenje grešaka ne utječe na glavni projekt. Ukoliko se razvije nova značajka koja je važna za projekt, nju možemo vrlo jednostavno implementirati pomoću spajanja. U ovom radu, prikazan je samo dio Git-a. Međutim, s prikazanim naredbama već možemo napraviti značajnu promjenu u suradnji.

LITERATURA

Ahmed, Reshma. What is Git, 2020. URL: <https://www.edureka.co/blog/what-is-git/> (2021-12-05)

Chacon, Scott; Straub, Ben. Git: About Version Control, 2014. URL: <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control> (2021-12-05)

Git Bash. URL: <https://www.atlassian.com/git/tutorials/git-bash#:~:text=What%20is%20Git%20Bash%3F,operating%20system%20through%20written%20commands> (2021-12-08)

Git Cherry Pick. URL: <https://www.atlassian.com/git/tutorials/cherry-pick#:~:text=git%20cherry%2Dpick%20is%20a,be%20useful%20for%20undoing%20changes> (2021-12-09)

Introduction to Git. URL: <https://www.notion.so/zarkom/Introduction-to-Git-ac396a0697704709a12b6a0e545db049> (2021-12-08)

Jacobs, Mike; Kaim, Ed. What is Git, 2021. URL: <https://docs.microsoft.com/en-us/devops/develop/git/what-is-git> (2021-12-05)

Krajina, Tomo. Uvod u Git. URL: <https://tkrajina.github.io/uvod-u-git/git.pdf> (2021-12-05)

Prajapati, Ameet. What is git commit, push, pull, log, aliases, fetch, config & clone, 2019. URL: <https://medium.com/mindorks/what-is-git-commit-push-pull-log-aliases-fetch-config-clone-56bc52a3601c> (2021-12-08)

Version Control Software Comparison: Git, Mercurial, CVS, SVN. URL: <https://medium.com/@derya.cortuk/version-control-software-comparison-git-mercurial-cvs-svn-21b2a71226e4> (2021-12-05)

What is Git. URL: <https://www.edureka.co/blog/what-is-git/> (2021-12-05)

What is Git. URL: <https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2016/11/Centralized-Version-Control-System-Workflow-What-Is-Git-Edureka.png> (2021-12-05)

What is Git. URL: <https://d1jnx9ba8s6j9r.cloudfront.net/blog/wp-content/uploads/2016/11/Distributed-Version-Control-System-Workflow-What-Is-Git-Edureka-768x508.png> (2021-12-05)

What is Linux. URL: <https://www.linux.com/what-is-linux/> (2021-12-05)

What is version control. URL: <https://www.atlassian.com/git/tutorials/what-is-version-control> (2021-12-05)

Why Git for your organization. URL: <https://www.atlassian.com/git/tutorials/why-git> (2021-12-09)