Poglavlje 16

Alat za kontrolu verzije

16.1 Uvod

Git je trenutno najpopularniji i najrasprostranjeniji sistem za kontrolu verzije, namenjen praćenju promena izvornog koda tokom procesa razvoja softvera. Iako je razvijen sa idejom podrške koordinaciji tima programera koji zajedno radi na određenom projektu, može biti korišćen i u generalne svrhe, za praćenje promena u okviru proizvoljne kolekcije datoteka. Kako se projekat razvija i evoluira, članovi tima obavljaju testiranje, otklanjaju greške i dodaju nove delove koda znajući da bilo koja verzija koda može biti rekonstruisana u proizvoljnom trenutku. Osobe koje učestvuju u razvoju mogu na osnovu uvida u projektnu istoriju da saznaju:

- Koje su promene unete?
- Ko ih je uneo?
- Kada su promene nastale?
- Zašto su promene bile neophodne?

Git je primer *distribuiranog sistema* za kontrolu verzije koji omogućava pun pristup svakoj datoteci, grani i iteraciji razvoja i svakom korisniku dozvoljava pristup potpunoj i zaokruženoj istoriji svih promena. Za razliku od centralizovanih sistema za kontrolu verzije, distribuiranim sistemima kao što je Git nije neopodna permanentna veza sa centralnim repozitorijumom.

Repozitorijum, odnosno Git projekat predstavlja kolekciju datoteka i direktorijuma u vezi sa projektom, uključujući i istoriju revizija za svaku pojedinačnu datoteku. Istorija predstavlja niz verzija iste datoteke u različitim vremenskim trenucima kada je izvršena *operacija postavljanja (eng. commit)*. Svaka novoizvršena operacija postavljanja nadovezuje se na prethodnu, tako da se u logičkom smislu dobija povezana lista koja može biti razgranata tako da prati različite grane razvoja

istog projekta. Svaki učesnik koji poseduje kopiju repozitorijuma ima na raspolaganju kompletnu bazu koda koji pripada projektu, zajedno sa istorijom njegovog razvoja. Korišćenjem osnovnih alata u komandnoj liniji, ili nekog od grafičkih okruženja (GUI), korisnicima su omogućene osnovne operacije interakcije sa istorijom, kloniranja repozitorijuma, postavljanja novih verzija koda, kreiranja i spajanja grana, poređenja različitih verzija koda i sl.

Rad sa repozitorijumima olakšava organizovanje i zaštitu koda, bez čega bi rad na velikim *open-source* projektima bio praktično nezamisliv. Učesnici u razvoju mogu ispravljati greške ili dodavati nove funkcionalnosti bez bojazni da će time pokvariti ili poremetiti glavni tok razvoja projekta. Ovo je omogućeno primenom tematskih grana koje se mogu na jednostavan način kreirati, pripojiti glavnom toku ili proglasiti zastarelim, po potrebi. Uz pomoć mrežnih platformi kao što su **GitHub** ili **Bitbucket**, otvaraju se dodatne mogućnosti kada je u pitanju saradnja i transparentnost tokom razvoja projekta. Javni repozitorijumi omogućavaju razvojnim timovima da sarađuju i doprinose zajedničkom cilju stvaranja najboljeg mogućeg proizvoda.

16.2 Osnovne komande

Odličan izvor informacija u vezi sa osnovnim i naprednim funkcionalnostima Git-a dostupan je u formi elektronske knjige:

`https://git-scm.com/book/en/v2`

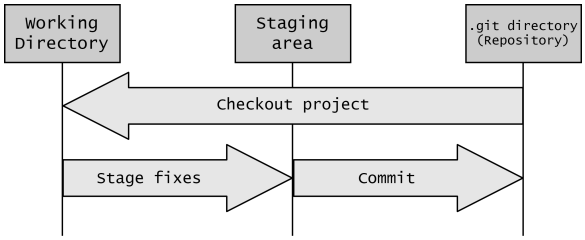
Nakon instalacije odgovarajuće verzije Git-a (za Linux, Windows ili macOS sistem), operacije je moguće obavljati putem komandi koje se izvršavaju iz komandne linije.

Datoteke koje pripadaju projektu u svakom trenutku se mogu nalaziti u jednom od tri osnovna stanja:

- **Postavljeno** (eng. *committed*) – podrazumeva da je datoteka bezbedno smeštena u lokalnu bazu podataka
- **Izmenjeno** (eng. *modified*) – sadržaj datoteke je izmenjen od poslednjeg postavljanja, ali sadržaj još nije pohranjen u bazu
- **Spremno za postavljanje** (eng. *staged*) – izmenjena datoteka je označena za postavljanje

U skladu sa tim, razlikujemo tri glavne sekcije Git projekta: Git direktorijum (eng. *.git directory*), radni direktorijum (eng. *working directory*) i pripremnu zonu (eng. *staging area*), kao što je i prikazano na slici 16.1.

Git direktorijum je mesto gde se čuvaju metapodaci i baza podataka u vezi sa projektom. Ovo je najvažniji deo Git sistema i tu se smeštaju podaci prilikom kloniranja repozitorijuma, kao i prilikom postavljanja novih verzija.



Slika 16.1: Glavne sekcije GIT projekta

Radni direktorijum sadrži datoteke koje predstavljaju stanje tekuće verzije projekta. Tekuća verzija se generiše na osnovu podataka iz baze i na raspolaganju je korisniku za unos izmena.

Pripremna zona je datoteka, koja je obično smeštena u Git direktorijumu i koja sadrži informacije o tome šta tačno treba da sadrži sledeća verzija koja će biti generisana prilikom postavljanja (*commit*). U Git terminologiji ova zona se naziva još i "*index*".

Tok rada se obično odvija u tri faze koje se sukcesivno smenjuju:

1. Korisnik modifikuje sadržaj datoteka u radnom direktorijumu
2. Korisnik selektivno označava one datoteke koje trebaju biti postavljene u repozitorijum, čime se one prebacuju u pripremnu zonu
3. Obavlja se operacija postavljanja (*commit*), čime se preuzimaju datoteke iz pripremne zone i permanentno se smeštaju u Git repozitorijum

U nastavku je dat pregled osnovnih Git komandi, koje se najčešće upotrebljavaju u praksi.

git init

Komanda *git init* inicijalizuje novi Git repozitorijum u tekućem direktorijumu i počinje sa njegovim praćenjem. Na ovaj način se dodaje skriveni direktorijum (*.git* direktorijum) u kojem je smeštena interna baza podataka neophodna za praćenje verzije. Način upotrebe komande je dat u nastavku.

```
$ git init
```

git clone

Komanda *git clone* kreira lokalnu kopiju postojećeg repozitorijuma sa servera. Klonirani repozitorijum sadrži sve izvorne datoteke, zajedno sa istorijom promena i granama razvoja. Primer upotrebe ove komande je dat u nastavku.

```
$ git clone https://github.com/rszes/test.git
```

git status

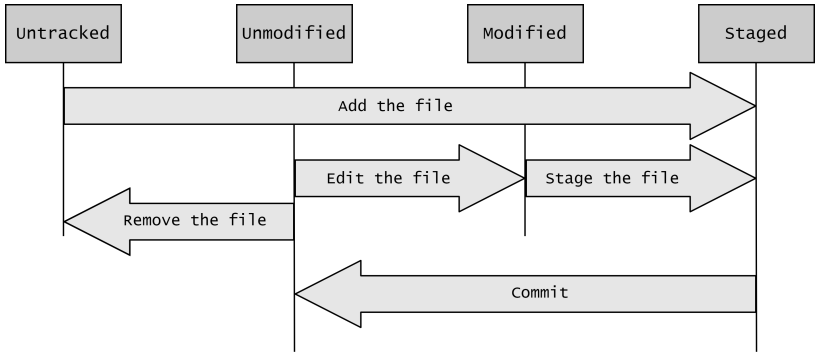
Svaka datoteka u radnom direktorijumu može se nalaziti u jednom od dva stanja:

- **Nepraćena** datoteka (eng. *untracked*)
- **Praćena** datoteka (eng. *tracked*)

Praćena je ona datoteka o čijim eventualnim promenama Git sistem vodi računa. Takva datoteka može se nalaziti u jednom od tri stanja:

- **Neizmenjena** (eng. *unmodified*) – datoteka čiji sadržaj nije izmenjen od poslednjeg postavljanja
- **Izmenjena** (eng. *modified*) – datoteka čiji sadržaj je promenjen
- **Pripremljena za postavljanje** (eng. *staged*) – prilikom sledećeg postavljanja, biće postavljene samo one datoteke koje se nalaze u ovom stanju, tj. koje se nalaze u pripremnoj zoni (eng. *staging area*)

Ilustracija ovih stanja je data na slici 16.2.



Slika 16.2: Stanja GIT projekta

Komanda *git status* služi za proveru stanja datoteka u radnom direktorijumu. Primer upotrebe je prikazan u nastavku.

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working directory clean
```

git add

Komanda *git add* označava izmenjene datoteke za postavljanje, čime ih prebacuje u pripremnu zonu. Ovo je prvi korak u kreiranju nove verzije koda. Sve promene koje su prebačene u pripremnu zonu će prilikom postavljanja postati integralni deo sledeće verzije, a samim tim i projektne istorije.

git commit

Komanda *git commit* snima promene iz pripremne zone u repozitorijum, čime se kompletira proces kreiranja nove verzije. Novostvorena verzija predstavlja presek stanja projekta u određenom vremenskom trenutku. Sve što je prethodno prebačeno u pripremnu zonu pomoću komande *git add*, postaće deo nove verzije kreirane komandom *git commit*.

git branch

Komanda *git branch* kreira novu granu razvoja koda.

git merge

Komanda *git merge* spaja promene u kodu nastale u dvema odvojenim granama razvoja koda.

git pull

Komanda *git pull* ažurira sadržaj lokalnog repozitorijuma sadržajem sa udaljenog servera.

git push

Komanda *git push* ažurira repozitorijum na udaljenom serveru promenama napravljenim u lokalu.

16.3 Instalacija GIT alata

Instalacija alata se može besplatno preuzeti sa sledećeg linka:

`https://git-scm.com/downloads`

Kako bi se instalirao GIT na Linux operativnom sistemu potrebno je u zavisnosti od distribucije izvršiti odgovarajuću komandu u komandnoj liniji terminala. Komande za pojedine distribucije nalaze se na sledećem linku:

`https://git-scm.com/download/linux`

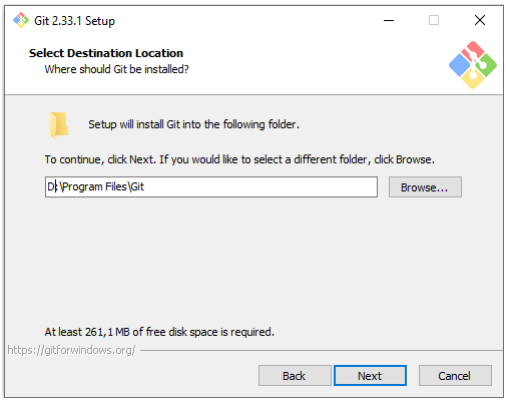
Sa druge strane, kako bi se instalirao GIT na operativnom sistemu Windows, potrebno je izvršiti sledeći niz koraka koji su prikazani na slikama koje slede u nastavku. Instalacionu datoteku za odgovarajuću verziju Windows operativnog sistema je moguće preuzeti na sledećem linku:

`https://git-scm.com/download/win`

Nakon preuzimanja, pokretanjem ove datoteke se otvara prozor za instalaciju prikazan na slici 16.3. Potrebno je isključiti opciju *Only show new options*, kako bi bilo omogućeno detaljnije podešavanje prilikom instalacije. Nakon toga, prihvatiti Licencu klikom na *Next*.

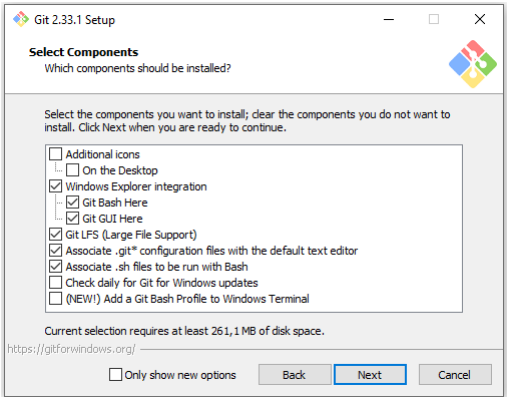
Slika 16.3: Instalacija Git-a korak 1

Odabrati lokaciju na kojoj će biti instaliran Git, kao što je prikazano na slici 16.4.



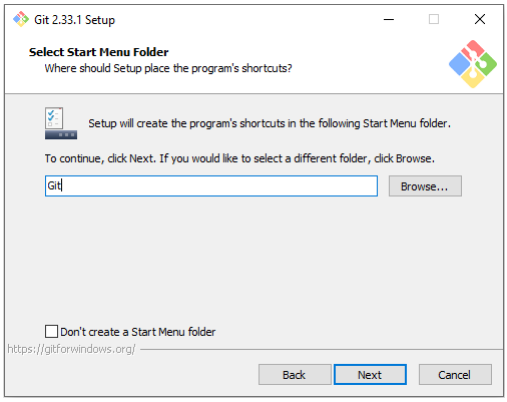
Slika 16.4: Instalacija Git-a korak 2

Slika 16.5 prikazuje prozor u kom je potrebno selektovati komponente koje će biti instalirane. Takođe, moguće je odabrati načine integracije Git-a. Ponuđene su dve opcije, prva predstavlja integraciju Git-a unutar Bash-a, gde se upotreba Git-a zasniva na primeni komandi u kozoli. Kao alternativa je ponuđena integracija u vidu GUI okruženja, u kojem se komande izvršavaju automatizovano, pokretanjem određenih akcija.



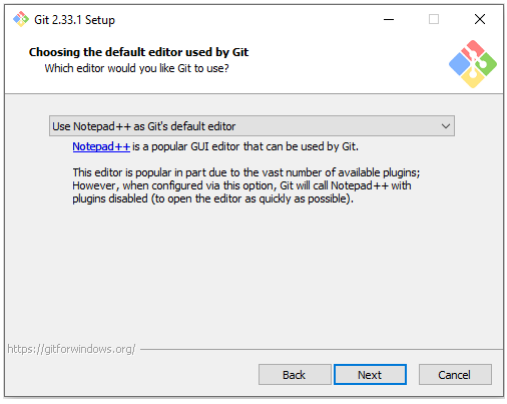
Slika 16.5: Instalacija Git-a korak 3

Nakon odabira komponenti, potrebno je imenovati direktorijum u kom će biti sačuvan Git unutar *Start* menija, kao što je pikazano na slici 16.6.



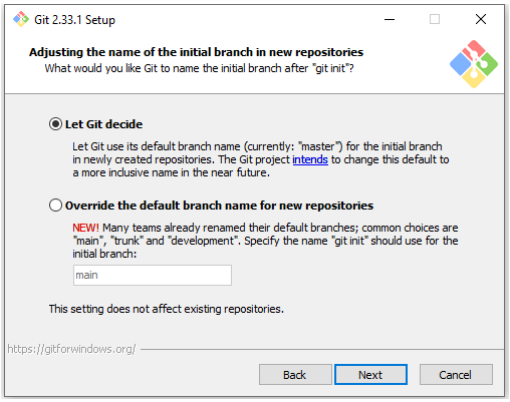
Slika 16.6: Instalacija Git-a korak 4

Prozor prikazan na slici 16.7 omogućava izbor podrazumevanog tekstualnog editora koji će Git upotrebljavati.



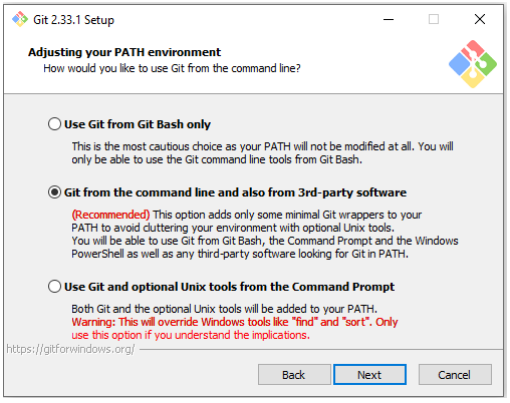
Slika 16.7: Instalacija Git-a korak 5

U okviru prozora sa slike 16.8 vrši se odabir imena inicijalne grane novog repozitorijuma. Prva opcija ostavlja podrazumevano ime inicijalne grane, koje je trenutno *"master"*. Druga opcija omogućava korisniku da sam navede ime inicijalne grane koje će se koristiti u svim novim repozitorijumima.



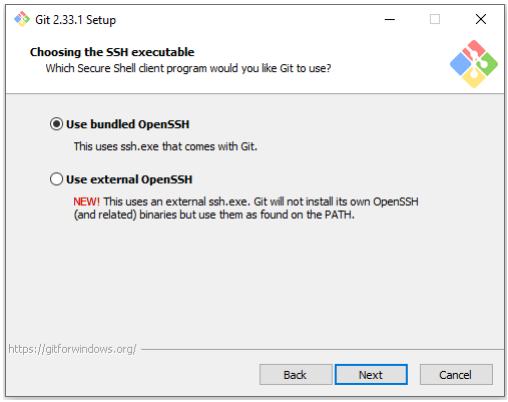
Slika 16.8: Instalacija Git-a korak 6

Nakon toga, potrebno je odabrati kako će Git biti upotrebljen u okviru komandne linije. Odabirom prve opcije ograničava se upotreba Git komandi samo unutar *Git bash*-a. Druga opcija omogućva upotrebu Git komandi unutar drugih konzola i programa. Preporučuje se izbor druge opcije, kao što je prikazano na slici 16.9, zbog toga što se Git može uključiti kao *plugin* za druge programe, što omogućava jednostavniju upotrebu.



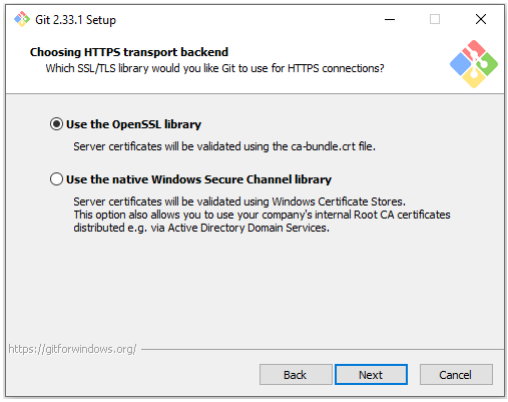
Slika 16.9: Instalacija Git-a korak 7

Slika 16.10 prikazuje prozor u kom je potrebno izabrati prvu opciju, koja omogućava Git-u da koristi ugrađen SSH.



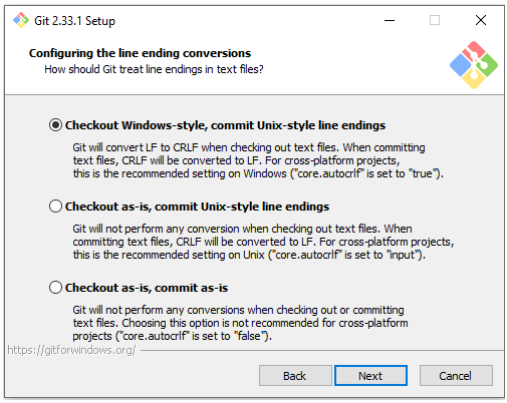
Slika 16.10: Instalacija Git-a korak 8

U prozoru sa slike 16.11 je potrebno izabrati OpenSLL biblioteku za upotrebu prilikom HTTPs povezivanja.



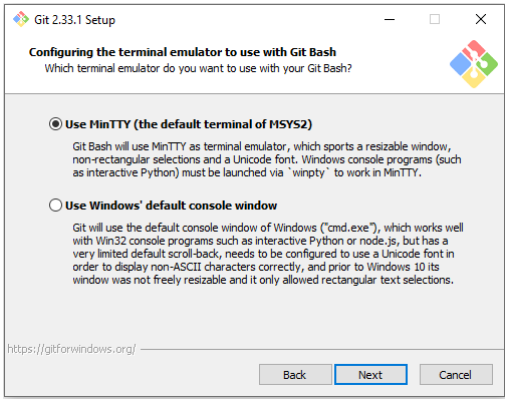
Slika 16.11: Instalacija Git-a korak 9

Sledeći prozor omogućava izbor načina postupanja sa krajevima linija u tekstualnim datotekama. Odabrati prvu opciju, kao što je prikazano na slici 16.12.



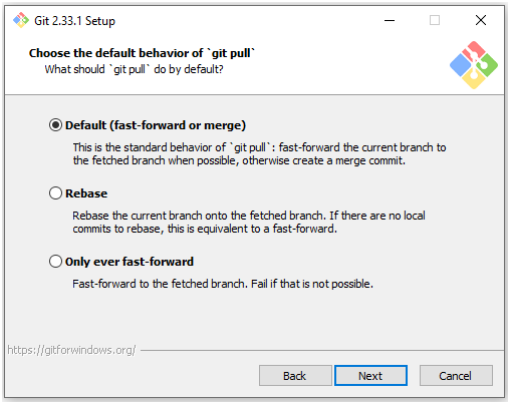
Slika 16.12: Instalacija Git-a korak 10

Slika 16.13 prikazuje prozor u okviru kog je potrebno izabrati konzolu koja će biti upotrebljena za *Git Bash*.



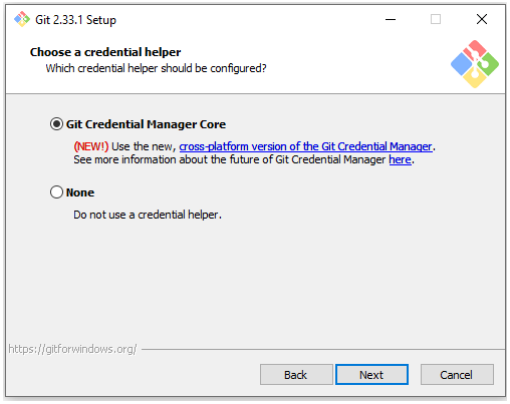
Slika 16.13: Instalacija Git-a korak 11

Nakon toga je potrebno izabrati ponašanje komande *git pull*. Ostaviti označeno podrazumevano ponašanje, kao na slici 16.14.



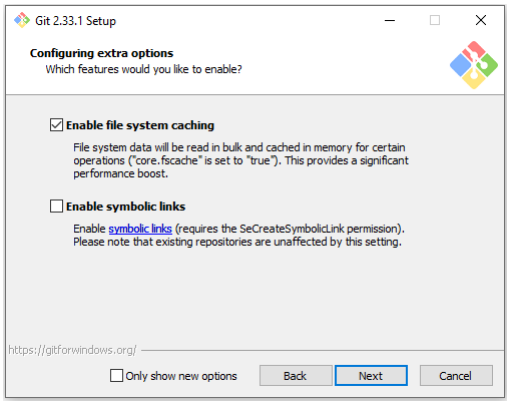
Slika 16.14: Instalacija Git-a korak 12

Ukoliko je potrebno preuzeti privatan repozitorijum sa servera, često je neophodno uneti kredencijale u vidu korisničkog imena i šifre. Prozor sa slike 16.15 omogućava izbor *Git Credential Manager*-a kao pomoćnog alata za upravljanje kredencijale.



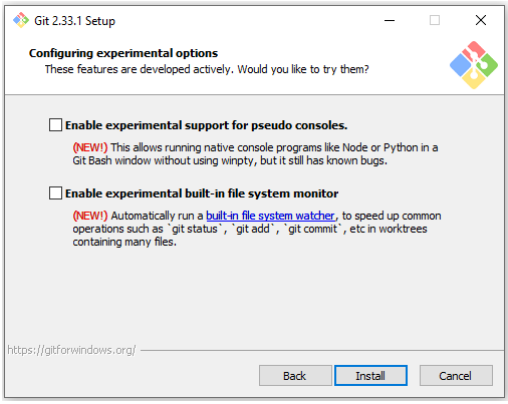
Slika 16.15: Instalacija Git-a korak 13

U nastavku odaberite *feature*-e koje želite da budu omogućeni. Primer odabira je prikazan na slici 16.16.



Slika 16.16: Instalacija Git-a korak 14

Konačno, ukoliko želite da omogućite eksperimentalne *feature*-e, to možete uraditi u okviru prozora prikazanog na slici 16.17. Nakon odabira, klikom na taster *Install* se pokreće instalacija.



Slika 16.17: Instalacija Git-a korak 15

16.4 Zadatak za vežbu

Zadatak 16.4.1. Kreirati javni (eng. *public*) repozitorijum na *Github*-u pod nazivom *GIT_exercise*. Nakon toga, potrebno je izvršiti kloniranje repozitorijuma na lokalnu mašinu. U repozitorijumu je potrebno napraviti direktorijum *LED_blink*. U okviru *LED_blink* direktorijuma, potrebno je napraviti datoteku *main.c*, čiji je sadržaj dat na listingu 16.1.

```
#include <avr/io.h>
#include <stdint.h>

int16_t main()
{
    while(1)
    ;
    return 0;
}
```

Listing 16.1: Datoteka main.c

Nakon izmene sadržaja ove datoteke, potrebno je ažurirati trenutni repozitorijum uz komentar "*Initial commit*". Potom, potrebno je izmeniti sadržaj ove datoteke sa kodom datim na listingu 16.2.

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>

int16_t main()
{
    int16_t high_time = 500;
    int16_t low_time = 500;

    DDRB |= 1 << 5; //PB5 je izlaz

    while (1)
    {
        PORTB |= 1 << 5; // LED ON
        _delay_ms(high_time); // Pauza 1s
        PORTB &= ~(1 << 5); // LED OFF
        _delay_ms(low_time); // Pauza 1s
    }

    return 0;
}
```

Listing 16.2: Modifikovana datoteka main.c

Kada je ova izmena napravljena, potrebno je napraviti novu objavu uz propratni komentar "*Initial code added*". Sledeći korak podrazumeva izmenu vrednosti pro-

menljivih `high_time` i `low_time` sa 300 i 700, respektivno. Nakon izmene, potrebno je komandom `git diff main.c` proveriti koje su izmene realizovane u odnosu na prethodnu verziju. Izmene je potrebno objaviti pod nazivom "*high_time and low_time values changed*".

Naredni korak podrazumeva kreiranje nove grane pod nazivom *branch1*. Izmene koje je potrebno izvršiti na ovoj grani se sastoje od kreiranja nove funkcije pod nazivom *ledBlink()* čija je implementacija prikazana na listingu 16.3 i izmene sadržaja *main()* funkcije tako da se umesto ovog dela koda poziva kreirana funkcija.

```
void ledBlink(int16_t high_time, int16_t low_time)
{
    PORTB |= 1 << 5; // LED ON
    _delay_ms(high_time); // Pauza 1s
    PORTB &= ~(1 << 5); // LED OFF
    _delay_ms(low_time); // Pauza 1s
}
```

Listing 16.3: Funkcija ledBlink()

Izmene je, zatim, potrebno postaviti na granu *branch1* uz komentar "*ledBlink function added*". Sa druge strane, na grani main je potrebno, u okviru datoteke *main.c*, kreirati funkciju *ledInit()*, prikazanu na listingu 16.4, i izmeniti sadržaj *main()* funkcije na odgovarajući način. Izmene postaviti pod nazivom "*ledInit function added*".

```
void ledInit()
{
    DDRB |= 1 << 5; //PB5 je izlaz
}
```

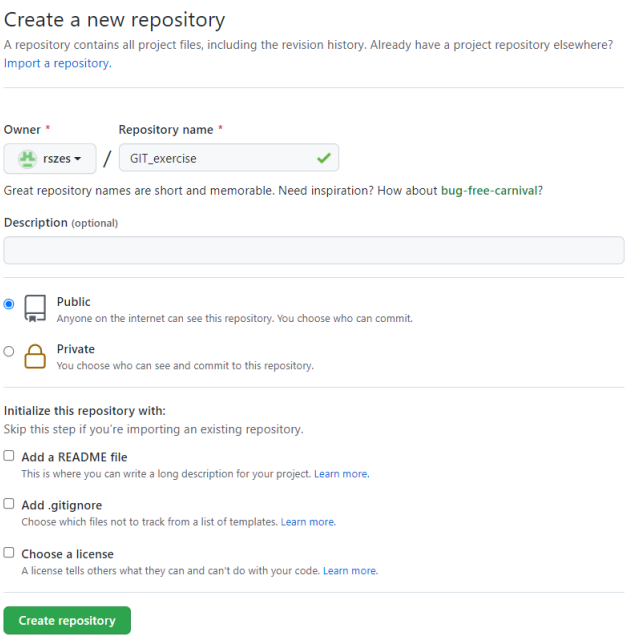
Listing 16.4: Funkcija init()

Naredni korak podrazumeva spajanje *main* i *branch1* grane, gde je potrebno rešiti konflikte tako što se izmene iz obe datoteke uvrste u konačnu verziju. Novu verziju je potrebno objaviti uz propratni komentar "*main and branch1 merged*".

Nakon toga potrebno je kreirati datoteku *end.txt*, izbrisati datoteku *main.c* iz repozitorijuma i postaviti novokreiranu datoteku *end.txt* pod nazivom "*main.c removed, end.txt added*".

Rešenje:

Na početku, potrebno je kreirati nalog na sajtu `https://github.com` i izvršiti prijavu. Pritiskom na dugme *New* otvara se odgovarajući meni. Potrebno je uneti ime repozitorijuma koji se kreira, odabrati da li će repozitorijum biti dostupan svima (eng. *Public*) ili samo odabranim korisnicima (eng. *Private*) i po potrebi uneti opis onoga šta će biti smešteno u njemu. Prikaz ovog je dat na slici 16.18.



Slika 16.18: Kreiranje repozitorijuma na GitHub-u

Klikom na dugme *Create repository* će repozitorijum biti napravljen, nakon čega će biti prikazan link do tog repozitorijuma. Ovaj link služi za kreiranje lokalnih kopija repozitorijuma i njihovo povezivanje sa originalom koji se nalazi na serveru. Kreiranje lokalne kopije je prikazano na listingu 16.5.

```
$ git clone https://github.com/rszes/GIT_exercise.git
Cloning into 'GIT_exercise'...
warning: You appear to have cloned an empty repository.
```

Listing 16.5: Kloniranje repozitorijuma

Nakon toga, potrebno je promeniti trenutni direktorijum na *GIT_exercise*. Pored toga, potrebno je podesiti ime i email adresu korisnika upotrebom komandi datih u nastavku.

```
$ git config --global user.email "kel.rszes@gmail.com"
$ git config --global user.name "rszes.admin"
```

Listing 16.6: Konfigurisanje imena i email adrese korisnika

Nakon što su konfigurisani ovi parametri, unutar lokalnog repozitorijuma napravljen je direktorijum LED_blink, a zatim, unutar njega datoteka *main.c*. Potom, potrebno je uneti sadržaj iz postavke zadatka u datoteku i sačuvati izmene. Upotrebom komande *git status*, unutar datog repozitorijuma, proverava se stanje

datoteka u repozitorijumu, gde se može primetiti da se direktorijum LED_blink i njegov sadržaj ne prati.

```
$ git status
On branch main

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
LED_blink/

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

Listing 16.7: Provera statusa

Kako bi se otpočelo praćenje ovog direktorijuma i njegovog sadržaja, potrebno je izvršiti njegovo prebacivanje u pripremnu zonu koristeći komandu *git add*. Komanda, data u nastavku, omogućava prebacivanje isključivo datoteke *main.c* i direktorijuma u okviru kojeg se on nalazi, bez ostalog sadržaja koji se nalazi u njemu (u ovom slučaju, ostali sadržaj ne postoji).

```
$ git add LED_blink/main.c
```

Listing 16.8: Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Sačuvavanje izmena repozitorijuma izvršava se upotrebom komande *git commit* koja kao parametar prihvata poruku. U ovom slučaju koristi se poruka definisana u postavci zadatka.

```
$ git commit -m "Initial commit"
[main (root-commit) b78d167] Initial commit
1 file changed, 9 insertions(+)
create mode 100644 LED_blink/main.c
```

Listing 16.9: Sačuvavanje promena

Nakon poziva ove komande, izmene su sačuvane na lokalnoj mašini, ali ne i na originalnom repozitorijumu koji se nalazi na serveru. Ažuriranje repozitorijuma na serveru izvršava se upotrebom komande *git push origin main* gde se ažurira *main* grana repozitorijuma.

```
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 329 bytes | 329.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rszes/GIT_exercise.git
* [new branch]      main -> main
```

Listing 16.10: Sačuvavanje promena

Nakon objavljivanja inicijalnog sadržaja direktorijuma, potrebno ga je zameniti sadržajem datim u postavci zadatka. Upotrebom komande `git status` može se videti da je promena datoteke *main.c* primećena.

```
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   LED_blink/main.c

no changes added to commit (use "git add" and/or "git commit -a")
```

Listing 16.11: Provera statusa

Korišćenjem komande *git add ** vrši se prebacivanje svih datoteka, koje se nalaze unutar trenutnog direktorijuma, u pripremnu zonu. Karakter * (eng. *Asterisk*) je specijalni karakter koji omogućava ovu funkcionalnost.

```
$ git add *
```

Listing 16.12: Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Nakon toga, potrebno je sačuvati promene uz odgovarajuću poruku koja opisuje koje promene su izvršene u odnosu na prethodnu verziju.

```
$ git commit -m "Initial code added"
[main ac399c7] Initial code added
1 file changed, 14 insertions(+), 2 deletions(-)
```

Listing 16.13: Sačuvavanje promena

Repozitorijum na serveru se zatim ažurira sačuvanim izmenama.

```
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 482 bytes | 482.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rszes/GIT_exercise.git
8a30824..ac399c7  main -> main
```

Listing 16.14: Ažuriranje repozitorijuma

Posle izmena promenljivih `high_time` i `low_time` u 300 i 700 respektivno, promene u odnosu na prethodno sačuvane datoteke u repozitorijumu mogu se videti upotrebom komande *git diff*. Kao parametar se prosleđuje ime datoteke čiji sadržaj se želi uporediti. Znakovi minusa označavaju sadržaj koji se nalazi u poslednjoj sačuvanoj verziji datoteke, dok plus karakteri označavaju izmene tog sadržaja.

```
$ git diff LED_blink/main.c
diff --git a/LED_blink/main.c b/LED_blink/main.c
index 43ea662..102295f 100644
--- a/LED_blink/main.c
+++ b/LED_blink/main.c
@@ -4,8 +4,8 @@

int16_t main()
{
    -       int16_t high_time = 500;
    -       int16_t low_time = 500;
    +       int16_t high_time = 300;
    +       int16_t low_time = 700;

    DDRB |= 1 << 5; //PB5 je izlaz
```

Listing 16.15: Pregled izmena

Potom, izmene je potrebno prebaciti u pripremnu zonu kako bi mogle biti sačuvane.

```
$ git add *
```

Listing 16.16: Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Nakon čega se vrši i njihovo sačuvavanje uz odgovarajući deskriptivni komentar.

```
$ git commit -m "high_time and low_time values changed"
[main 7450ac5] high_time and low_time values changed
1 file changed, 2 insertions(+), 2 deletions(-)
```

Listing 16.17: Sačuvavanje promena

Potom, vrši se ažuriranje repozitorijum na *main* grani na serveru.

```
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 360 bytes | 360.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rszes/GIT_exercise.git
ac399c7..7450ac5  main -> main
```

Listing 16.18: Ažuriranje repozitorijuma

Nakon toga, shodno postavci zadatka potrebno je kreirati novu granu u okviru repozitorijuma pod nazivom *branch1*. Ovo je urađeno upotrebom naredbe *git branch branch1*, gde parametar *branch1* predstavlja naziv grane koja se kreira.

```
$ git branch branch1
```

Listing 16.19: Kreiranje grane branch1

Budući da smo u ovom trenutku pozicionirani na glavnoj (*main*) grani, potrebno je izvršiti prelaz na granu *branch1* koja je kreirana u prethodnom koraku. Ovo se može učiniti upotrebom komande *git checkout branch1*. Nakon poziva ove komande, ispisuje se poruka koja signalizira uspešnu promenu grane.

```
$ git checkout branch1
Switched to branch 'branch1'
```

Listing 16.20: Prelazak na granu branch1

U narednom koraku, potrebno je izmeniti sadržaj *main.c* datoteke odgovarajućim kodom (funkcija *ledBlink()*) datim u postavci zadatka. Tako izmenjen kod, potrebno je prebaciti u pripremnu zonu kako bi izmene mogle biti kasnije sačuvane.

```
$ git add *
```

Listing 16.21: Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Na sličan način kao i do sada, promene se čuvaju uz odgovarajući deskriptivni komentar u skladu sa postavkom zadataka.

```
$ git commit -m "ledBlink function added"
[branch1 d9c7639] ledBlink function added
1 file changed, 9 insertions(+), 4 deletions(-)
```

Listing 16.22: Sačuvavanje promena

Nakon sačuvavanja izmena, potrebno je ažurirati repozitorijum na serveru, ali ovog puta na grani *branch1*.

```
$ git push origin branch1
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 410 bytes | 410.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'branch1' on GitHub by visiting:
remote:      https://github.com/rszes/GIT_exercise/pull/new/branch1
remote:
To https://github.com/rszes/GIT_exercise.git
* [new branch]      branch1 -> branch1
```

Listing 16.23: Ažuriranje repozitorijuma

Nakon ažuriranja *branch1* grane, potrebno je, ponovo, pozicionirati se na glavnu (*main*) granu upotrebom komande `git checkout main`. Potom, potrebno je izvršiti odgovarajuće promene u datoteci *main.c* (implementiranje *ledInit()* funkcije). Izmenjena datoteka se, nakon toga, prebacuje u pripremnu zonu.

```
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

$ git add *
```

Listing 16.24: Promena grane i prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Izmene se, zatim, sačuvavaju uz odgovarajući komentar dat u tekstu zadatka.

```
$ git commit -m "ledInit function added"
[main 7477211] ledInit function added
1 file changed, 6 insertions(+), 1 deletion(-)
```

Listing 16.25: Sačuvavanje promena

Glavna grana repozitorijuma na serveru se zatim ažurira ovim izmenama.

```
$ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 353 bytes | 353.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rszes/GIT_exercise.git
7450ac5..7477211  main -> main
```

Listing 16.26: Ažuriranje repozitorijuma

Kako su obe grane u ovom trenutku različite, u odnosu na trenutak pre grananja, spajanje njihovog sadržaja se izvršava upotrebom *git merge* komande. Ova komanda spaja datoteke iz grane u kojoj smo trenutno pozicionirani (*main*), sa onom koja se prosleđuje kao parametar (u ovom slučaju to je grana *branch1*).

```
$ git merge branch1
Auto-merging LED_blink/main.c
CONFLICT (content): Merge conflict in LED_blink/main.c
Automatic merge failed; fix conflicts and then commit the result.
```

Listing 16.27: Spajanje grana

U toku spajanja dveju verzija datoteki *main.c* alat nije u stanju da prepozna da li je u pitanju jedna funkcija koja se razlikuje u imenu i sadržaju ili se radi o dve zasebne funkcije. Zbog toga, on ukazuje na konflikt. Konflikt rešava programer, tako što datoteku, nastalu spajanjem, menja u skladu sa svojim potrebama, kao što je prikazano na listinzima 16.28 i 16.29.

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>
```

```
<<<<<<< HEAD
void ledInit()
{
    DDRB |= 1 << 5; //PB5 je izlaz
=======
void ledBlink(int16_t high_time, int16_t low_time)
{
    PORTB |= 1 << 5; // LED ON
    _delay_ms(high_time); // Pauza 1s
    PORTB &= ~(1 << 5); // LED OFF
    _delay_ms(low_time); // Pauza 1s
>>>>>>> branch1
}

int16_t main()
{
    int16_t high_time = 300;
    int16_t low_time = 700;

    ledInit();

    while (1)
    {
        ledBlink(high_time, low_time);
    }

    return 0;
}
```

Listing 16.28: Konfliktna datoteka main.c

```
#include <avr/io.h>
#include <util/delay.h>
#include <stdint.h>

void ledInit()
{
    DDRB |= 1 << 5; //PB5 je izlaz
}

void ledBlink(int16_t high_time, int16_t low_time)
{
    PORTB |= 1 << 5; // LED ON
    _delay_ms(high_time); // Pauza 1s
    PORTB &= ~(1 << 5); // LED OFF
    _delay_ms(low_time); // Pauza 1s
}

int16_t main()
{
    int16_t high_time = 300;
    int16_t low_time = 700;

    ledInit();

    while (1)
    {
```

```
        ledBlink(high_time, low_time);
    }

    return 0;
}
```

Listing 16.29: Rešavanje konflikta

Izmenjena datoteka se nakon toga prebacuje u pripremnu zonu.

```
$ git add *
```

Listing 16.30: Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Nakon toga se izmene sačuvavaju uz odgovarajući komentar.

```
$ git commit -m "main and branch1 merged"
[main 79cd7b4] main and branch1 merged
```

Listing 16.31: Sačuvavanje promena

Konačno, repozitorijum na serveru se ažurira izmenama na glavnoj grani (*main*). Posle ovog koraka glavna grana i grana *branch1* su spojene na serveru.

```
$ git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 6 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 377 bytes | 377.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/rszes/GIT_exercise.git
7477211..79cd7b4  main -> main
```

Listing 16.32: Ažuriranje repozitorijuma

Nakon spajanja grana, po uslovu zadatka, potrebno je napraviti novu datoteku *end.txt* i izvršiti brisanje datoteke *main.c* iz repozitorijuma, upotrebom komande `git rm`.

```
$ git rm LED_blink/main.c
rm 'LED_blink/main.c'
```

Listing 16.33: Brisanje datoteke main.c iz repozitorijuma

Nova datoteka *end.txt* se prebacuje u pripremnu zonu, čime se ona prvi put uvrštava u repozitorijum, ali samo na lokalnoj mašini.

```
$ git add LED_blink/end.txt
```

Listing 16.34: Prebacivanje izmenjenih datoteka iz repozitorijuma u pripremnu zonu

Nakon toga, potrebno je sačuvati izmene uz odgovarajući komentar.

```
$ git commit -m "main.c removed, end.txt added"
[main 64026ee] main.c removed, end.txt added
2 files changed, 31 deletions(-)
create mode 100644 LED_blink/end.txt
delete mode 100644 LED_blink/main.c
```

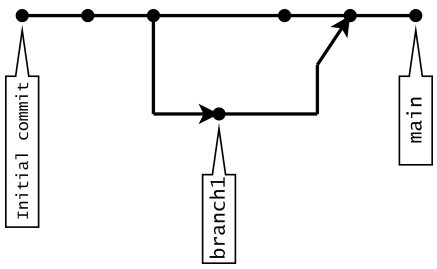
Listing 16.35: Sačuvavanje promena

Konačno, potrebno je izvršiti ažuriranje repozitorijuma na serveru, čime se iz ovog repozitorijuma briše datoteka *main.c*, a uvrštava datoteka *end.txt*. Datoteka *main.c* je i dalje sačuvana na prethodnim objavama, međutim ona nije vidljiva u poslednjoj objavi. Upotrebom komande *git checkout* čiji je parametar broj prethodne objave, može se vratiti repozitorijum na odgovarajuću sačuvanu verziju, čime se može pristupiti različitim verzijama repozitorijuma, tj. datoteka koje se u njemu nalaze. Brojeve verzija prethodnih objava možemo dobiti upotrebom komande *git log*.

```
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Writing objects: 100% (4/4), 295 bytes | 295.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/rszes/GIT_exercise.git
79cd7b4..64026ee  main -> main
```

Listing 16.36: Ažuriranje repozitorijuma

Na slici 16.19 je prikazan dijagram stabla koje je kreirano ovim zadatkom i predstavlja dobar vizuelni uvid u to šta je rađeno, koje objave su napravljene, koje grane su kreirane itd. Tačke na dijagramu predstavljaju pojedinačne objave koje su napravljene (u hronološkom redu). Dodatno, ovom dijagramu je moguće pristupiti i na *GitHub*-u, u odeljku *Insights/Network*.



Slika 16.19: Prikaz dijagrama stabla koje je kreirano ovim zadatkom