

**SLOBOMIR P UNIVERZITET
FAKULTET ZA INFORMACIONE TEHNOLOGIJE**



**SEMINARSKI RAD
ELEKTRONSKO POSLOVANJE – TEORIJA I PRIMJENA
GIT**

**Predmetni profesor:
Prof. dr Dragoljub Pilipović**

**Student:
Belmin Kadušić
Broj indexa: 496/23**

Doboj, 2024

SADRŽAJ

1. UVOD	3
2. VERZIONIRANJE KODA I OSNOVNI POJMOVI	4
2.1. Linearno verzionisanje koda	4
2.2. Grafovi, grananje i spajanje grana	5
3. INSTALACIJA I POSTAVLJANJE.....	8
2.3. Windows	8
2.4. macOS.....	9
2.5. Linux (Debian/Ubuntu).....	10
4. OSNOVNE OPERACIJE SA GIT-OM	11
5. DODATNI ALATI I INTEGRACIJE.....	16
2.1. GIT GUI alati.....	16
2.2. Integracija s razvojnim okolinama.....	16
2.3. GIT hosting platforme	16
2.4. Hookovi (kuke) GIT repozitorija.....	16
2.5. Alati za upravljanje verzijama datoteka.....	16
6. NAJBOLJE PRAKSE I SAVJETI	18
7. PRIMJER UPOTREBE.....	19
8. ZAKLJUČAK	32
9. LITERATURA.....	33
10. SPISAK SLIKA.....	34

1. UVOD

Verzioniranje koda obuhvata razvoj i evoluciju metoda i alata koji su korišteni za praćenje i upravljanje promjenama u kodu tokom vremena. U počecima programiranja, kada su projekti bili manji i manje složeni, programeri su često pratili promjene ručno. To bi moglo uključivati čuvanje kopija datoteka s različitim imenima ili dodavanje komentara u kod koji opisuju izmjene. U 1980-ima i 1990-ima razvijeni su centralizirani sistemi za verzioniranje kao što su CVS (Concurrent Versions System) i SVN (Subversion). Ti su alati omogućavali programerima da spremljeni kod dijele s drugima te prate istoriju promjena. Međutim, ti su sistemi imali svoje nedostatke, uključujući sporu brzinu rada i probleme s grananjem i spajanjem koda. Ranih 2000-ih godina, počeo je razvoj distribuiranih sistema za verzioniranje, poput Bitkeepera. Distribuirani sistemi donose promjenu paradigme u upravljanju verzijama, omogućavajući svakom programeru da ima puni lokalni repozitorij s punom istorijom promjena. Ovo je poboljšalo brzinu rada i omogućilo fleksibilnije načine saradnje.

GIT je stvoren 2005. godine od strane Linusa Torvaldsa, autora Linux kernela. Razvio ga je kao odgovor na potrebu Linux kernel tima za bržim i efikasnijim sistemom za upravljanje kodom¹. GIT je distribuirani sistem za verzioniranje koji se brzo proširio i postao dominantan alat u softverskoj industriji. Nakon što je GIT postao popularan, razvili su se razni alati i usluge koji ga podržavaju, poput GitHuba, GitLaba i Bitbucket. Ovi su servisi omogućili programerima da jednostavno dijele svoj kod s drugima, prate promjene, upravljaju projektima i sarađuju u razvoju softvera.

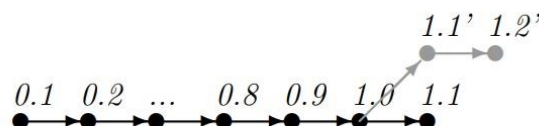
¹ <https://en.wikipedia.org/wiki/Git>

2. VERZIONIRANJE KODA I OSNOVNI POJMOVI

Svaka aplikacija koja ima stvarnog korisnika kome rješava neki stvarni problem ima i korisničke zahtjeve. Taj korisnik možemo biti mi sami, može biti neko hipotetsko tržište (kome planiramo prodati rješenje) ili može biti naručilac. Korisničke zahtjeve ne možemo nikad tačno predvidjeti u trenutku kad krenemo pisati program. Možemo satima, danima i mjesecima sjediti s budućim korisnicima i planirati šta će sve naša aplikacija imati, ali kad korisnik sjedne pred prvu verziju aplikacije, čak i ako je pisana tačno prema njegovim specifikacijama, on će naći nešto što ne valja. Radi li se o nekoj maloj izmjeni, možda ćemo je napraviti na licu mjesta. Možda ćemo trebati otići kući, potrošiti nekoliko dana i napraviti novu verziju.

Desit će se, na primjer, da korisniku damo na testiranje verziju 1.0. On će istestirati i, naravno, naći nekoliko sitnih stvari koje treba ispraviti. Otići ćemo kući, ispraviti ih, napraviti verziju 1.1 s kojom će klijent biti zadovoljan. Nekoliko dana kasnije, s malo više iskustva u radu s aplikacijom, on zaključuje kako sad ima bolju ideju kako je trebalo ispraviti verziju 1.0. Sad, dakle, treba "baciti u smeće" posao koji smo radili za 1.1, vratiti se na 1.0 i od nje napraviti npr. 1.1b.

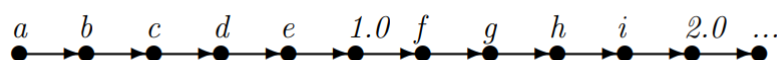
Grafički bi to izgledalo ovako:



U trenutku kad je korisnik odlučio da mu trenutna verzija ne odgovara trebamo se vratiti korak unazad i započeti novu verziju, odnosno novu granu projekta te nastaviti projekt sa tom izmjenom.²

2.1. Linearno verzionisanje koda

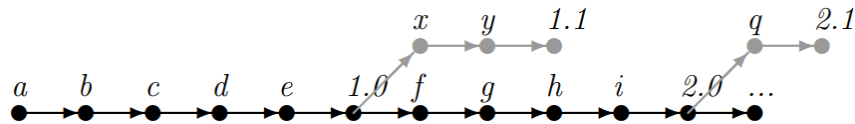
Linearni pristup verzionisanju koda se najbolje može opisati slijedećom ilustracijom.



² Tomo Krajina, „Uvod u Git“

To je idealna situacija u kojoj tačno unaprijed znamo kako aplikacija treba izgledati. Započnemo projekat sa početnim stanjem *a*, pa napravimo izmjene *b*, *c*, ... sve dok ne zaključimo da smo spremni izdati prvu verziju za javnost i proglasimo to verzijom 1.0.

Postoje mnoge varijacije ovog linearnog modela, jedna česta je:



Ona je česta u situacijama kad nemamo kontrolu nad time koja je tačno verzija programa instalirana kod klijenta. S web aplikacijama to nije problem jer vi jednostavno možete prebaciti aplikaciju na server i odmah svi klijenti koriste novu verziju. Međutim, ako je vaš program klijentima snimljen na CD i takav poslan klijentu može se desiti da jedan ima instaliranu verziju 1.0, a drugi 2.0.³

2.2. Grafovi, grananje i spajanje grana

Grafovi, grananje i spajanje grana su ključne funkcionalnosti u GIT-u koje omogućavaju organizaciju i upravljanje promjenama u kodu.

Grafovi u GIT-u prikazuju istoriju razvoja projekta i odnose između različitih grana. Svaki komit u repozitoriju povezan je s prethodnim komitom, stvarajući grafičku strukturu promjena. Ova struktura omogućava programerima da vizualiziraju kako se razvijao kod, koji su komiti bili uključeni u određenu granu i kako su grane spojene ili odvojene tokom vremena.

Grananje u GIT-u omogućava programerima da razvijaju nove karakteristike, ispravljaju greške ili eksperimentišu s različitim idejama bez uticaja na glavnu granu projekta. Kada programer stvori novu granu, stvara se odvajanje od trenutne tačke u istoriji projekta. Sve promjene napravljene u toj grani ne utiču na ostale grane projekta sve dok se ne spoje.

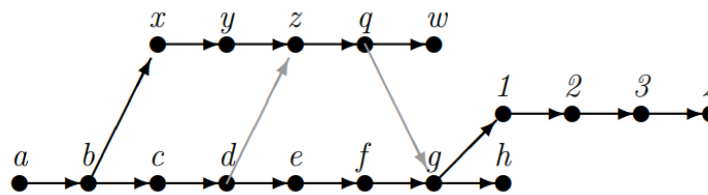
Spajanje grana je proces kombiniranja promjena iz jedne grane u drugu. To se obično radi kako bi se integrirale promjene iz različitih grana u glavnu granu projekta. Kada su promjene u nekoj grani spremne za integraciju u glavnu granu, programer može koristiti komanda ``git merge`` kako bi spojio te promjene s glavnom granom.

³ Tomo Krajina, „Uvod u Git“

GIT će automatski pokušati spojiti promjene, ali ako dođe do konflikata (npr. dvije grane mijenjaju istu datoteku na istom mjestu), programer mora riješiti te konflikte ručno prije nego što spoji grane.

Grafovi, grananje i spajanje grana su ključni za organizaciju i upravljanje razvojem složenih projekata u GIT-u. Omogućavaju programerima da rade paralelno na različitim karakteristikama, eksperimentiraju s novim idejama i održavaju glavnu granu projekta stabilnom i funkcionalnom.

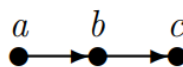
Uzet ćemo jedan grafički primjer grafa, te opisati moguće tokove razvoja istog:



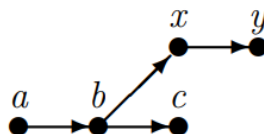
Svaka tačka grafa je stanje projekta. Projekt s gornjim grafom započeo je s nekim početnim stanjem a . Programer je napravio nekoliko izmjena i snimio novo stanje b , zatim c , ... i tako sve do h , w i 4 .

Ovakav graf je stanje istorije projekta, ali iz njega ne možemo zaključiti koji su redom čvorovi nastajali. Neke stvari možemo zaključiti: vidi se na primjer, da je d nastao nakon c , e nakon d ili z nakon y . Ne znamo da li je prije nastao c ili x . Ili, čvor 1 je sigurno nastao nakon g , ali iz grafa se ne vidi da li je nastao prije x ili poslije x .

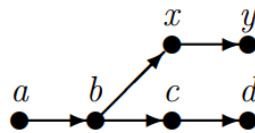
Primjer kako je navedeni graf mogao nastati:



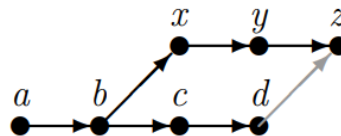
Programer je započeo aplikaciju, snimio stanje a , b i c i tada se sjetio da ima neki problem kojeg može riješiti na dva načina, vratio se na b i napravio novu granu. Tamo je napravio izmjene x i y :



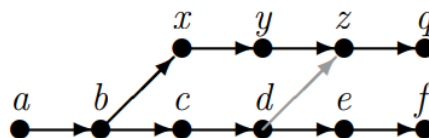
Zatim se sjetio izmjene koju je mogao napraviti u originalnoj verziji, vratio se tamo i dodao d :



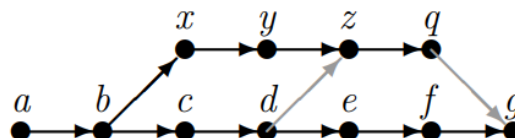
Nakon toga se vratio na svoj prvobitni eksperiment i odlučio da bi bilo dobro tamo imati izmjene koje je napravio u *c* i *d*. Tada je preuzeo te izmjene u svoju granu:



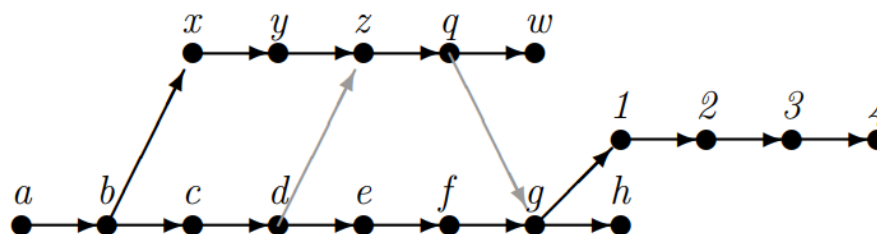
Na eksperimentalnoj grani je napravio još jednu izmjenu *q*. Tada je odlučio privremeno napustiti taj eksperiment i posvetiti se izmjenama koje mora napraviti u glavnoj grani. Vratio se na originalnu granu i tamo napredovao s *e* i *f*:



Sjetio se da bi mu sve izmjene iz ekperimentalne grane odgovarale u originalnoj, preuzeo ih u početnu granu:



... zatim je nastavio i napravio još jednu eksperimentalnu granu (1, 2, 3, ...). Na onoj je prvoj eksperimentalnoj grani dodao još i *w* i tako dalje...



Jedna od velikih prednosti gita je lakoća stvaranja novih grana i preuzimanja izmjena iz jedne u drugu granu.⁴

⁴ Tomo Krajina, „Uvod u Git“

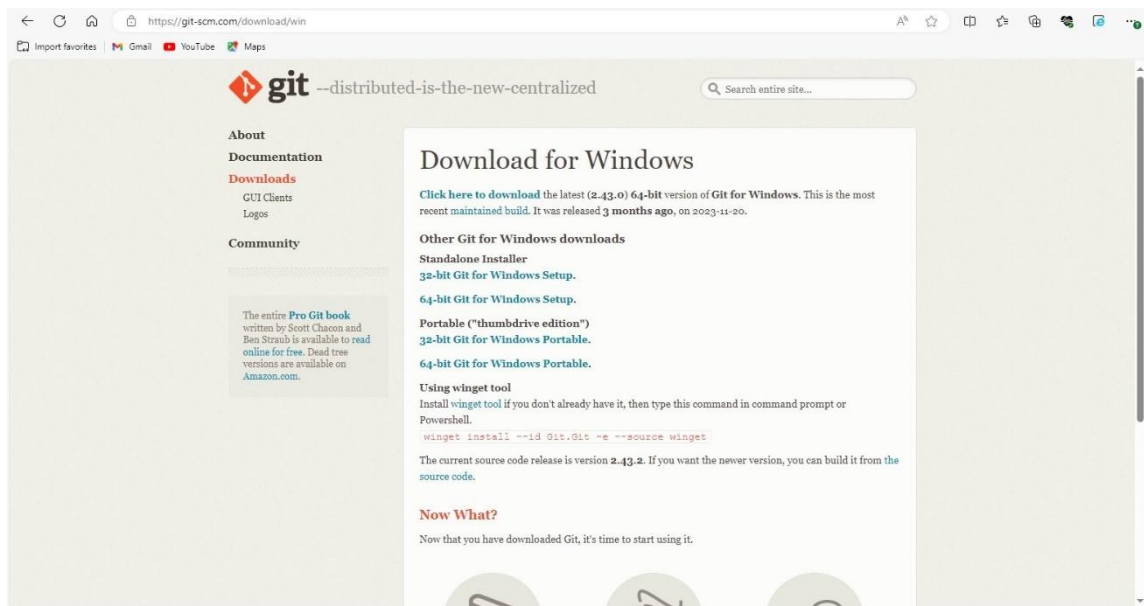
3. INSTALACIJA I POSTAVLJANJE

Instalacija i postavljanje GIT-a su relativno jednostavni procesi, a osnovni koraci za instalaciju i postavljanje GIT-a su različiti na različitim operativnim sistemima:

2.3. Windows

Preuzeti instalacijski program s službene web stranice GIT-a (<https://git-scm.com/>).

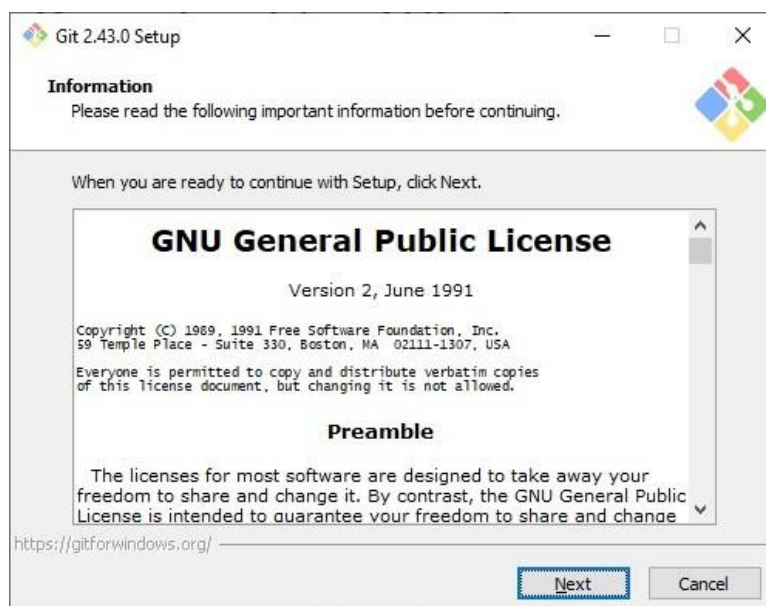
Slika 1 – Službena stranica GIT-a



Izvor: <https://git-scm.com/download/win>

Pokrenuti preuzeti instalacijski program.

Slika 2 – Pokretanje instalacije



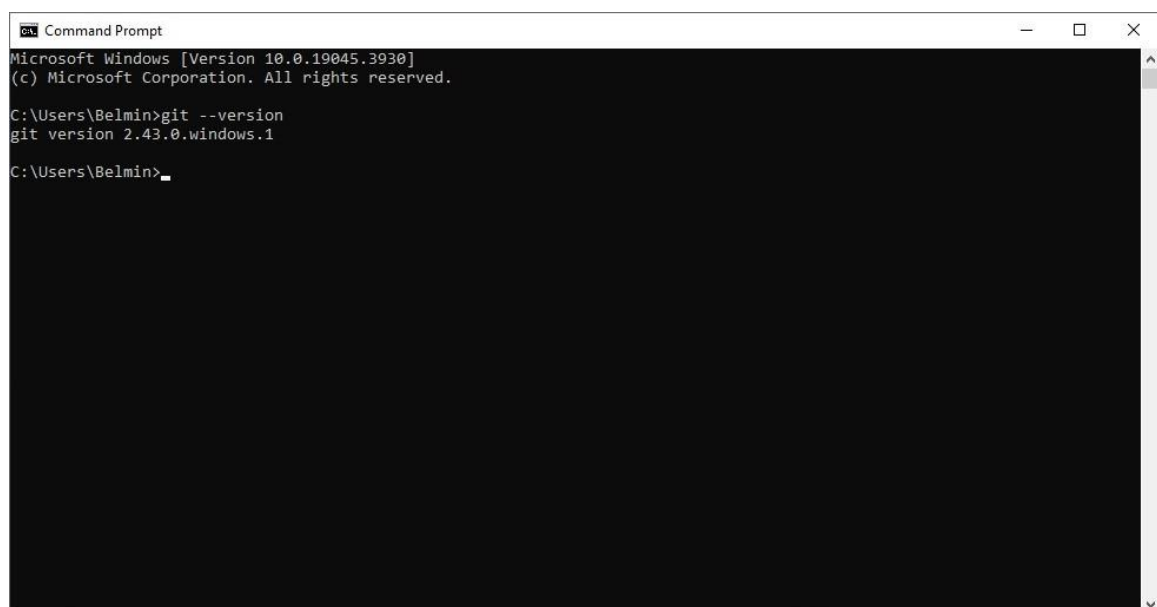
Slijediti upute u instalacijskom wizardu, prihvatajući zadane postavke ili prilagođavajući ih prema željama.

Slika 3 – Kraj instalacije



Kada instalacija završi, u Command Prompt ili PowerShell komandom `git --version` moguće je provjeriti da li je uspješno instaliran GIT.

Slika 4 – Provjera uspješnosti instalacije GIT-a



2.4. macOS

Instalacija GIT-a na macOS-u može se izvršiti putem Homebrew paketnog menadžera ili preuzimanjem instalacijskog programa s web stranice GIT-a.

U slučaju korištenja Homebrew, pokrenuti Terminal i izvršiti komandu *brew install git*.

U slučaju preuzimanja instalacijskog programa, posjetiti službenu web stranicu GIT-a (<https://git-scm.com/>) i preuzeti instalacijski program za macOS. Pokrenuti preuzeti instalacijski program i slijedite upute u wizardu za instalaciju.

Nakon instalacije, otvoriti Terminal i provjeriti je li GIT uspješno instaliran komandom *git --version*.

2.5. Linux (Debian/Ubuntu)

Instalacija GIT-a na Debian-based distribucijama (poput Ubuntu-a) može se izvršiti putem paketnog menadžera.

Otvoriti Terminal i izvršiti komandu *sudo apt update* za osvježavanje informacija o dostupnim paketima. Zatim izvršiti komandu *sudo apt install git* za instalaciju GIT-a.

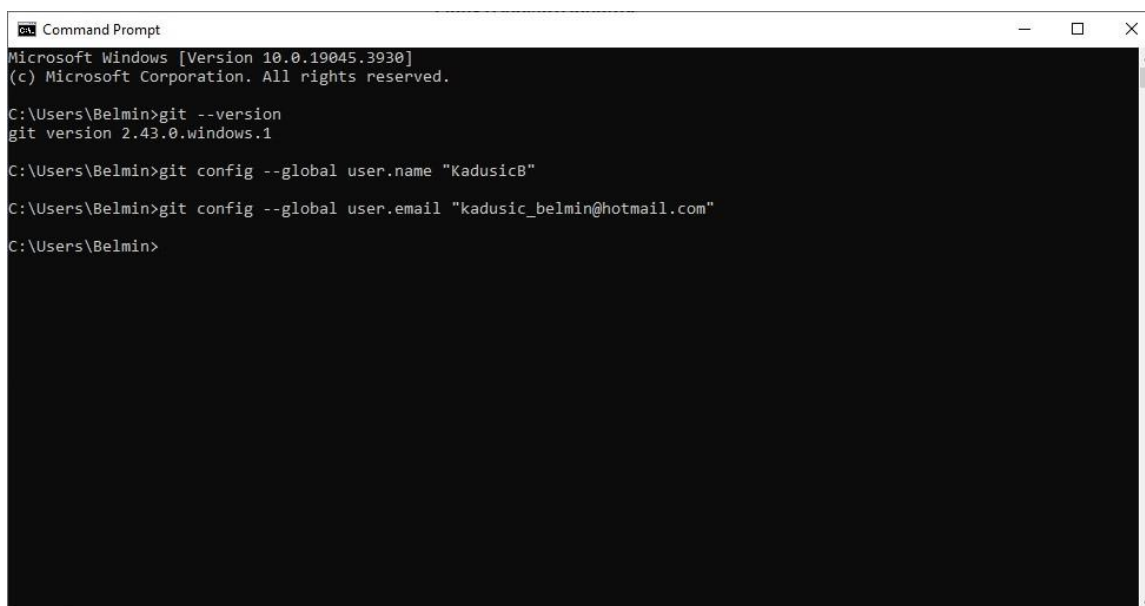
Provjera uspješnosti instalacije GIT-a se izvodi komandom *git --version*.

Nakon uspješne instalacije, moguće je konfigurisati neke osnovne postavke GIT-a kao što su korisničko ime i e-mail adresu pomoću komandi:

```
git config --global user.name "Vaše ime"
```

```
git config --global user.email vaš@email.com
```

Slika 5 – Konfiguracija korisničkog imena i e-mail adrese



```
Command Prompt
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>git --version
git version 2.43.0.windows.1

C:\Users\Belmin>git config --global user.name "KadusicB"

C:\Users\Belmin>git config --global user.email "kadusic_belmin@hotmail.com"

C:\Users\Belmin>
```

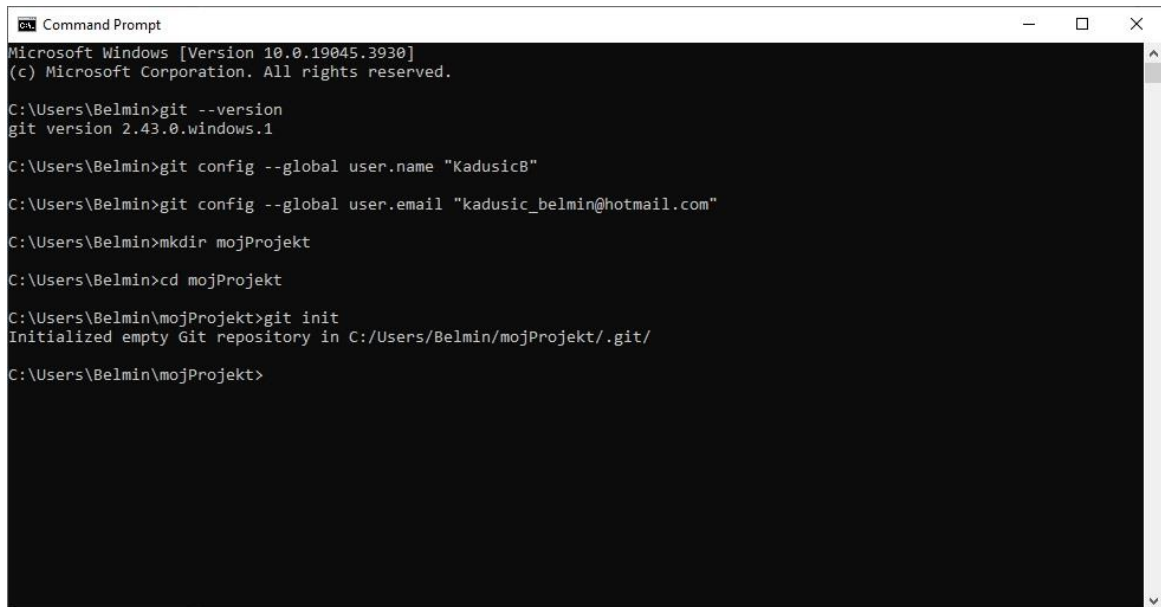
Ove komande postavljaju globalne postavke, koje će se primjenjivati na sve vaše projekte. Nakon ovih koraka, GIT je spreman za korištenje na vašem sistemu.

4. OSNOVNE OPERACIJE SA GIT-OM

Nekoliko osnovnih operacija sa GIT-om:

Inicijalizacija repozitorija: Kreiranje novog Git repozitorija u direktoriju projekta. To se obično radi komandom *git init*.

Slika 6 – Inicijalizacija GIT repozitorija „mojProjekt“



```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>git --version
git version 2.43.0.windows.1

C:\Users\Belmin>git config --global user.name "KadusicB"

C:\Users\Belmin>git config --global user.email "kadusic_belmin@hotmail.com"

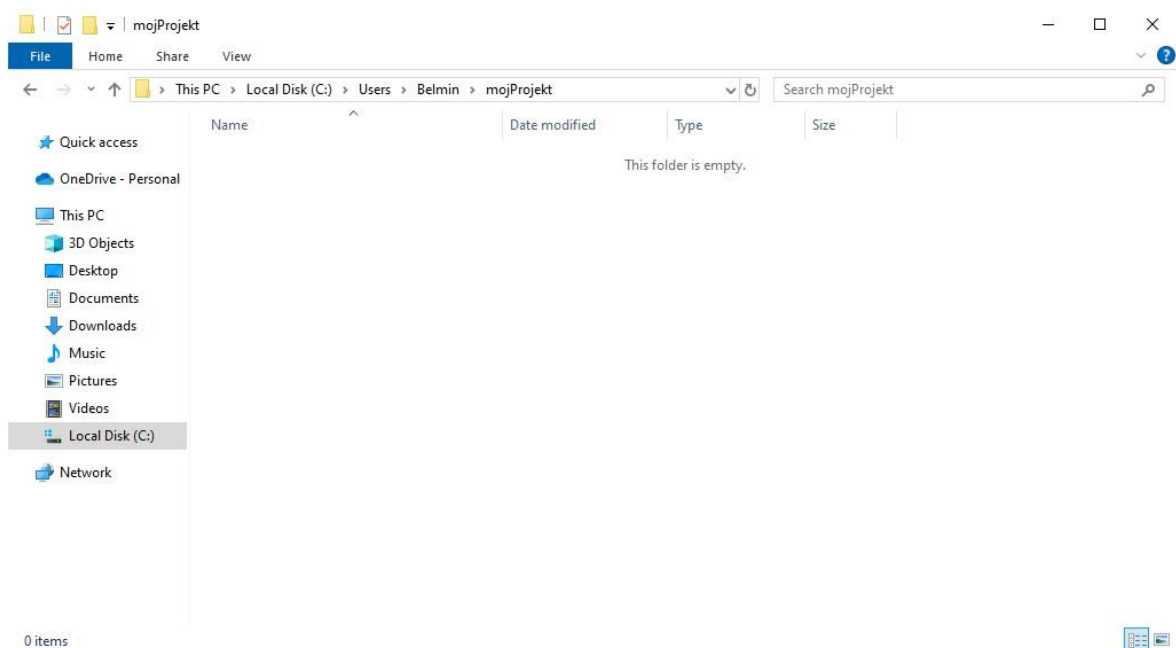
C:\Users\Belmin>mkdir mojProjekt

C:\Users\Belmin>cd mojProjekt

C:\Users\Belmin\mojProjekt>git init
Initialized empty Git repository in C:/Users/Belmin/mojProjekt/.git/

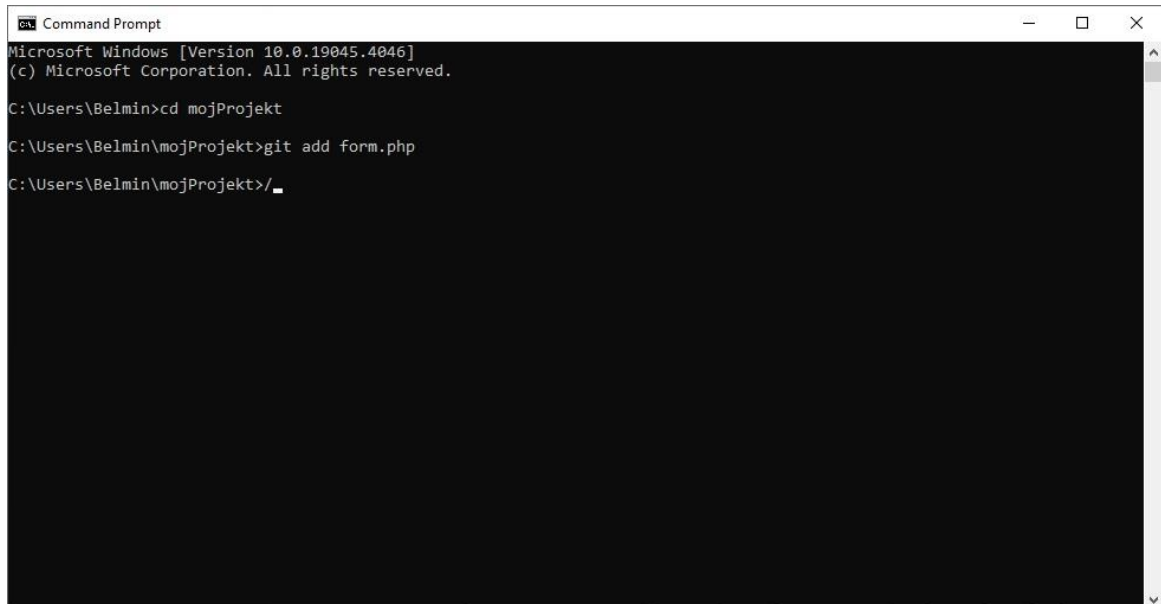
C:\Users\Belmin\mojProjekt>
```

Slika 7 – Prazan GIT repozitorij na putanji „C:\Users\Belmin\mojProjekt“



Dodavanje fajlova: Kada napravite izmjene u kodu, koristite *git add* da dodate te izmjene u staging area. Na primjer: *git add form.php*

Slika 8 – Dodavanje izmjene u staging area komandom *git add form.php*

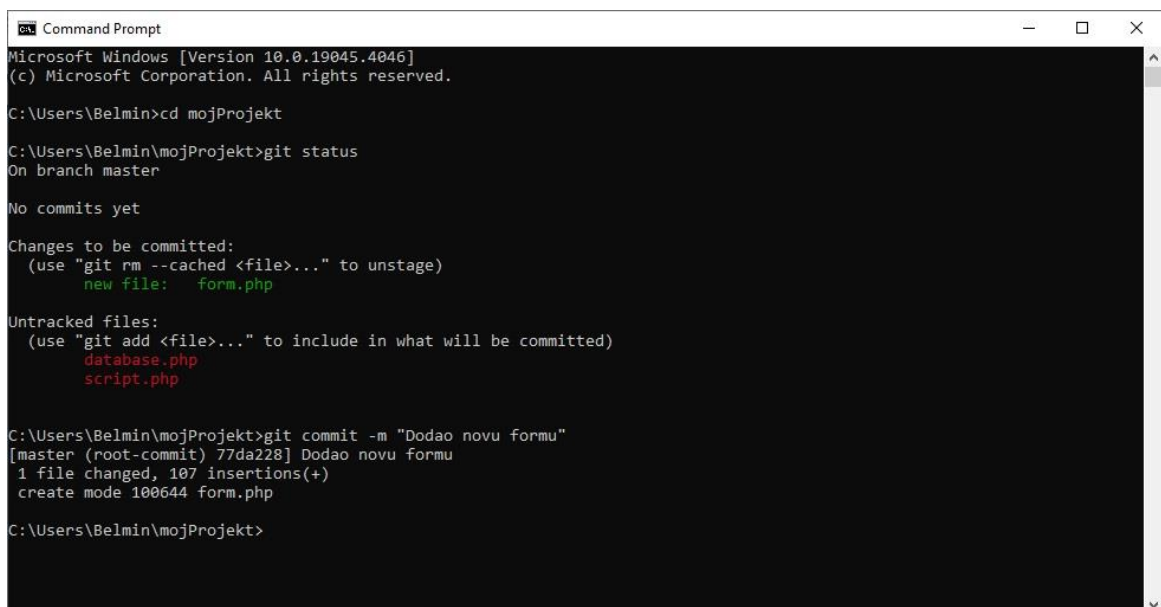


```
Command Prompt
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>cd mojProjekt
C:\Users\Belmin\mojProjekt>git add form.php
C:\Users\Belmin\mojProjekt>_
```

Komit: Kada ste zadovoljni sa izmjenama u staging area, koristite *git commit* da trajno sačuvate te izmjene. Na primjer: *git commit -m "Dodao novu formu"*.

Slika 9 – Upotreba komande komit sa dodavanjem komentara



```
Command Prompt
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>cd mojProjekt
C:\Users\Belmin\mojProjekt>git status
On branch master

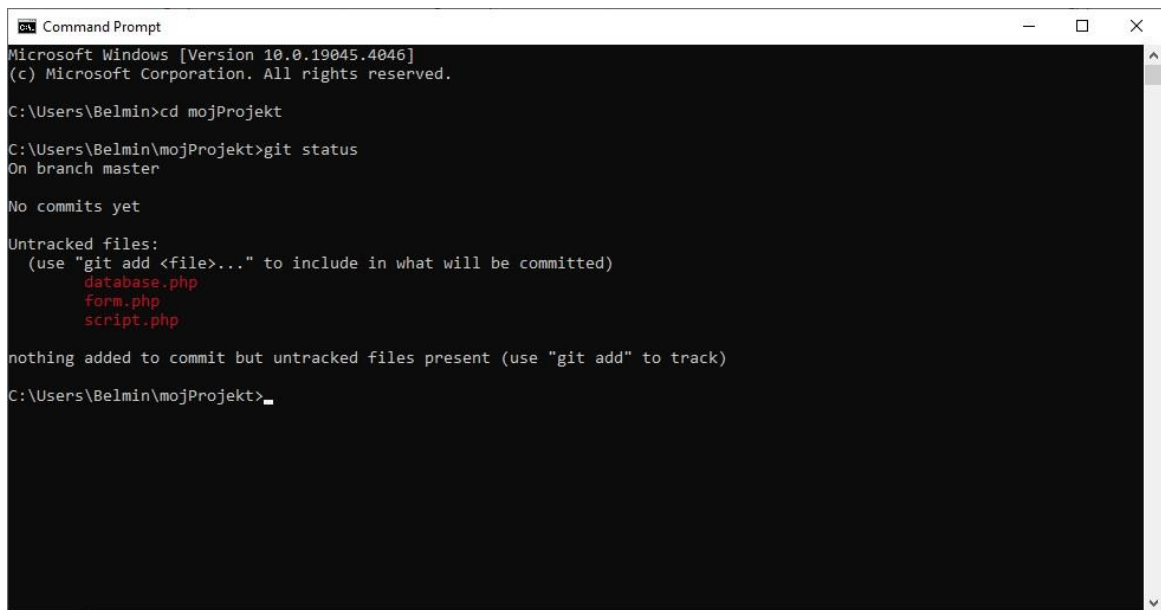
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   form.php

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

C:\Users\Belmin\mojProjekt>git commit -m "Dodao novu formu"
[master (root-commit) 77da228] Dodao novu formu
 1 file changed, 107 insertions(+)
 create mode 100644 form.php
C:\Users\Belmin\mojProjekt>
```

Pregled statusa: Komanda *git status* prikazuje trenutno stanje repozitorija, uključujući izmjene koje su spremne za komit i one koje nisu.

Slika 10 – Upotreba komande git status kada smo izvršili promjene u folderu

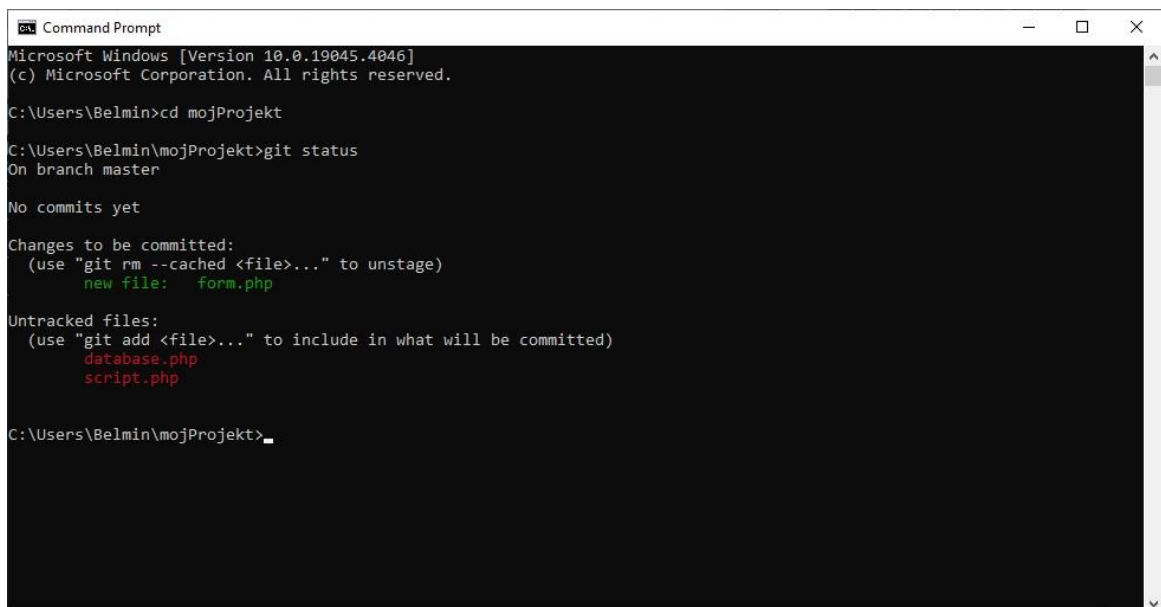
```
Command Prompt
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>cd mojProjekt
C:\Users\Belmin\mojProjekt>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        database.php
        form.php
        script.php

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\Belmin\mojProjekt>
```

Slika 11 – Upotreba komande git status poslije komande git add

```
Command Prompt
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>cd mojProjekt
C:\Users\Belmin\mojProjekt>git status
On branch master

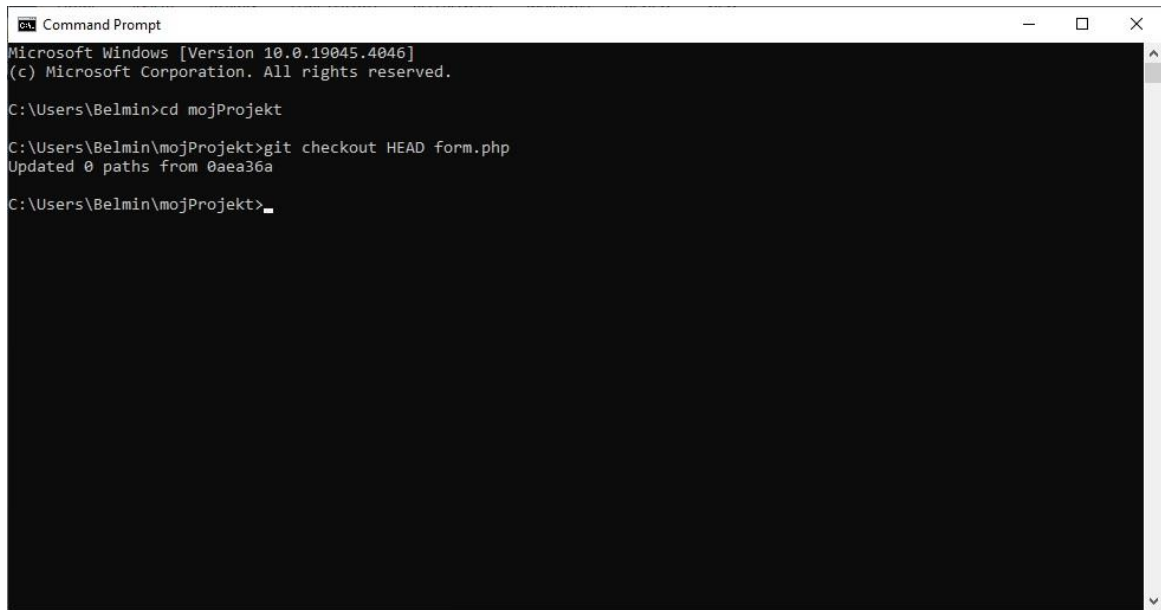
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   form.php

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

C:\Users\Belmin\mojProjekt>
```

Povratak na prethodnu verziju: Ako želite da se vratite na prethodnu verziju fajla, koristite *git checkout*. Na primjer: *git checkout HEAD form.php*.

Slika 12 – Upotrebe komande git checkout HEAD form.php

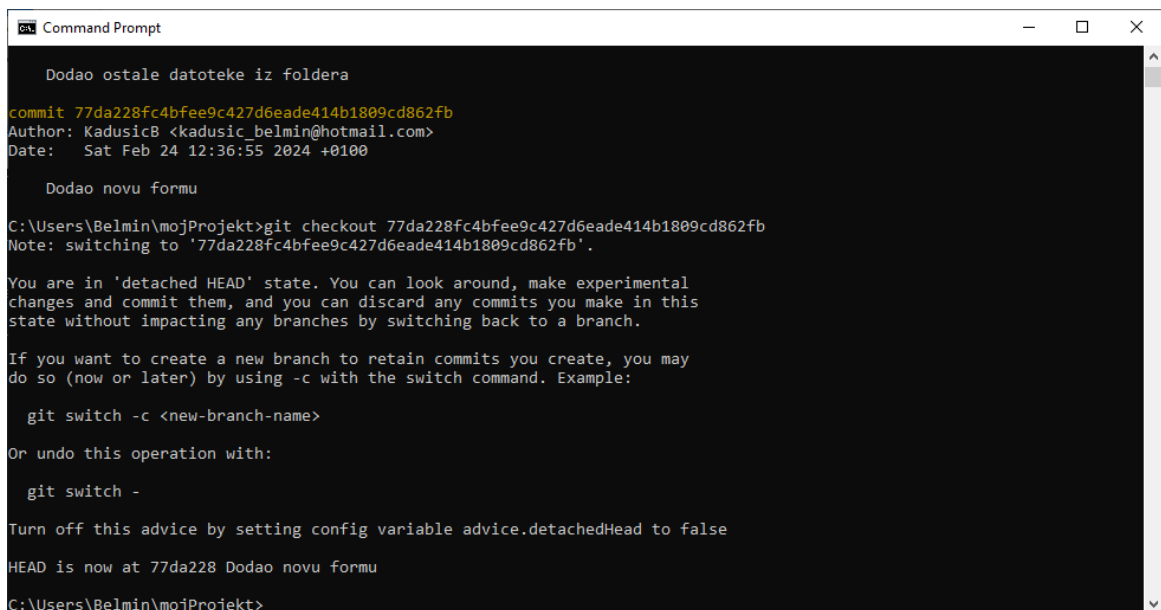
```
Command Prompt
Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>cd mojProjekt

C:\Users\Belmin\mojProjekt>git checkout HEAD form.php
Updated 0 paths from 0aea36a

C:\Users\Belmin\mojProjekt>
```

Povratak na određeni komit: Ako želite da se vratite na određeni komit, koristite *git checkout* sa odgovarajućim hash kodom komita.

Slika 13 – Vraćanje na određeni komit komandom git checkout

```
Command Prompt

Dodao ostale datoteke iz foldera
commit 77da228fc4bfee9c427d6eade414b1809cd862fb
Author: KadusicB <kadusic_belmin@hotmail.com>
Date: Sat Feb 24 12:36:55 2024 +0100

Dodao novu formu

C:\Users\Belmin\mojProjekt>git checkout 77da228fc4bfee9c427d6eade414b1809cd862fb
Note: switching to '77da228fc4bfee9c427d6eade414b1809cd862fb'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

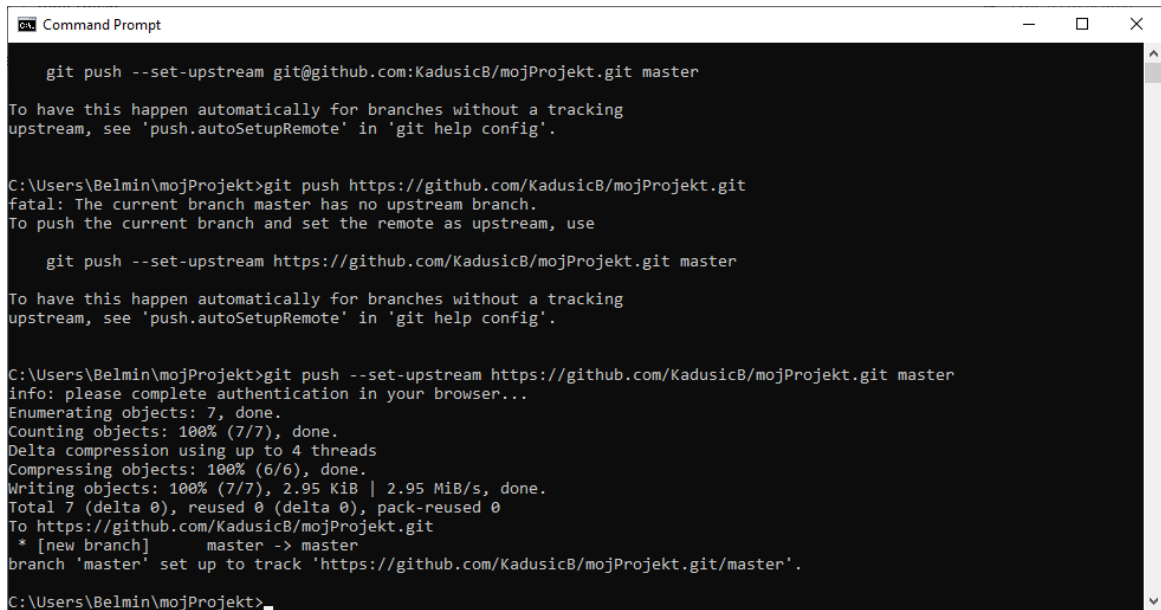
Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 77da228 Dodao novu formu

C:\Users\Belmin\mojProjekt>
```

Slanje izmjena na udaljeni repozitorij: Komanda *git push* šalje vaše izmjene na udaljeni repozitorij (na primjer, GitHub).

Slika 14 - Upotreba komande git push



```
Command Prompt

git push --set-upstream git@github.com:KadusicB/mojProjekt.git master

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

C:\Users\Belmin\mojProjekt>git push https://github.com/KadusicB/mojProjekt.git
fatal: The current branch master has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream https://github.com/KadusicB/mojProjekt.git master

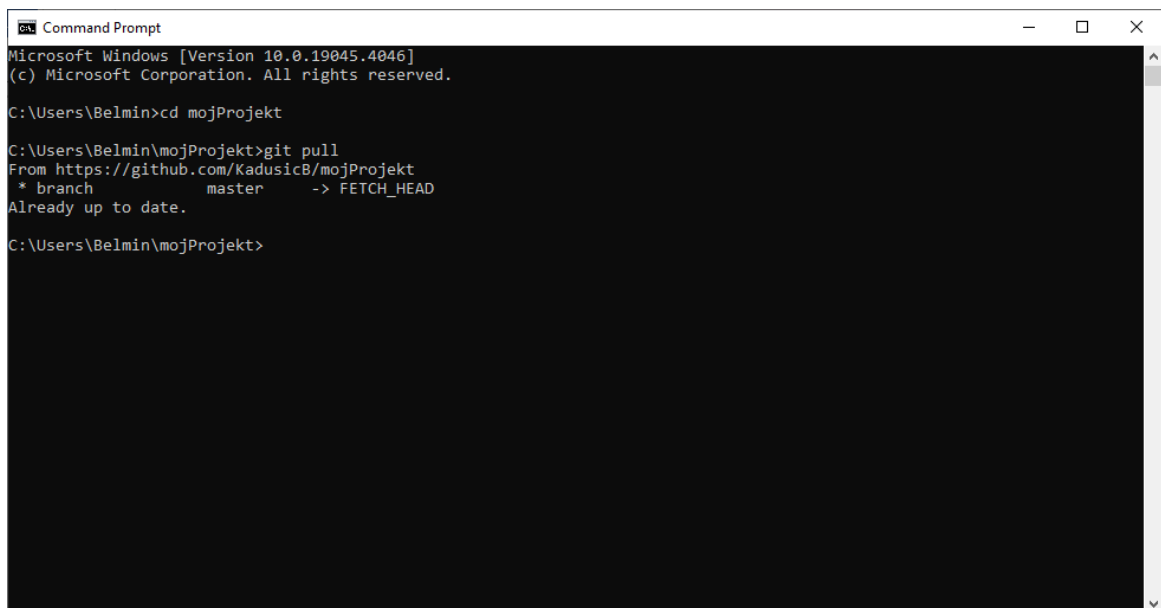
To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

C:\Users\Belmin\mojProjekt>git push --set-upstream https://github.com/KadusicB/mojProjekt.git master
info: please complete authentication in your browser...
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 2.95 KiB | 2.95 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/KadusicB/mojProjekt.git
 * [new branch]      master -> master
branch 'master' set up to track 'https://github.com/KadusicB/mojProjekt.git/master'.

C:\Users\Belmin\mojProjekt>
```

Povlačenje izmjena sa udaljenog repozitorija: Komanda *git pull* povlači najnovije izmjene sa udaljenog repozitorija na vaš lokalni repozitorij.

Slika 15 - Upotreba komande git pull



```
Command Prompt

Microsoft Windows [Version 10.0.19045.4046]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>cd mojProjekt

C:\Users\Belmin\mojProjekt>git pull
From https://github.com/KadusicB/mojProjekt
 * branch      master      -> FETCH_HEAD
Already up to date.

C:\Users\Belmin\mojProjekt>
```

Kloniranje repozitorija: Ako želite da kopirate cijeli repozitorij sa udaljenog servera, koristite *git clone*.

5. DODATNI ALATI I INTEGRACIJE

Postoji mnogo dodatnih alata i integracija koji mogu poboljšati produktivnost i iskustvo rada s GIT-om.

2.1. GIT GUI alati

GIT GUI alati pružaju grafičko okruženje za upravljanje GIT repozitorijima. Neki popularni alati uključuju Sourcetree, GitKraken, GitHub Desktop i TortoiseGIT.

Ovi alati omogućuju korisnicima da vizualno pregledaju istoriju komitova, rad s granama, pregledaju promjene u datotekama i obavljaju druge GIT operacije putem intuitivnog korisničkog okruženja.

2.2. Integracija s razvojnim okolinama

Mnoge razvojne okoline (IDE) imaju ugrađenu podršku za GIT. Primjeri uključuju Visual Studio Code, IntelliJ IDEA, Eclipse, Atom i mnoge druge.

Ove integracije omogućuju programerima da rade s GIT-om izravno unutar svog omiljenog razvojnog okruženja, uključujući pregled promjena, upravljanje granama, komitanje i slanje promjena na udaljene repozitorije.

2.3. GIT hosting platforme

GIT hosting platforme poput GitHuba, GitLaba i Bitbucket pružaju dodatne alate i funkcionalnosti za saradnju, upravljanje projektima i praćenje problema.

Ove platforme omogućuju programerima da dijele svoj kod s drugima, pregledavaju kod drugih, upravljaju radom na projektima kroz issues i pull requeste, te koriste napredne alate za praćenje performansi i saradnju.

2.4. Hookovi (kuke) GIT repozitorija

GIT hookovi su skripte koje se automatski pokreću u određenim točkama u GIT procesu, poput komitanja ili spajanja grana.

Korisnici mogu koristiti hookove za izvođenje automatskih provjera koda, izvještavanje o promjenama, automatsko pokretanje testova itd.

2.5. Alati za upravljanje verzijama datoteka

Neke aplikacije za upravljanje datotekama, poput Dropboxa i Google Drivea, imaju ugrađenu podršku za verzioniranje datoteka.

Ovo omogućuje korisnicima da pregledaju istoriju promjena u datotekama i vraćaju se na prethodne verzije, iako ovi alati obično nisu toliko napredni kao pravi sistemi za upravljanje verzijama kao što je GIT.

Ovi dodatni alati i integracije pružaju dodatne mogućnosti i olakšavaju korištenje GIT-a u razvoju softvera, saradnji s timom i upravljanju projektima.

6. NAJBOLJE PRAKSE I SAVJETI

Nekoliko najboljih praksi i savjeta za učinkovito korištenje GIT-a u razvoju softvera:

Redovito komitanje - Često radite komit kako biste zadržali istoriju projekta detaljnom i čistom. Preporučuje se komitanje nakon svake logičke promjene ili jedinice rada.

Opisivanje komit poruka - Pišite jasne i koncizne komit poruke koje opisuju svrhu i sadržaj promjene. Ovo olakšava razumijevanje istorije promjena i olakšava saradnju s drugima.

Korištenje grana za razvoj karakteristika - Koristite grane za izoliranje razvoja novih karakteristika, ispravaka grešaka i eksperimentiranje s kodom. To omogućava paralelan rad na različitim dijelovima projekta bez uticaja na glavnu granu.

Redovno ažuriranje lokalnog repozitorija - Redovno ažurirajte svoj lokalni repozitorij kako biste dobili najnovije promjene iz udaljenog repozitorija. To se može postići komandama poput *git fetch* i *git pull*.

Rješavanje konflikata pri spajanju - Ako dođe do konflikata pri spajanju grana, rješavajte ih pažljivo i sistemski. Pregledajte promjene, komunicirajte s ostalim članovima tima i osigurajte da spojene promjene ostanu funkcionalne i ispravne.

Korištenje .gitignore datoteke - Iskoristite *.gitignore* datoteku kako biste definisali koje datoteke ili direktoriji ne bi trebali biti praćeni GIT-om. To uključuje privremene datoteke, datoteke iz build procesa, konfiguracijske datoteke s lozinkama itd.

Reviw kode i pull requesti - Koristite pull requeste i revizije koda kako biste omogućili timsko pregledavanje i reviziju promjena prije nego što se spoje s glavnom granom. To pomaže u pronalaženju i ispravljanju grešaka te promovisanju najboljih praksi u razvoju softvera.

Redovito čišćenje repozitorija - Redovito provjeravajte i čistite repozitorij od nepotrebnih grana i komitova kako biste održali istoriju projekta preglednom i smanjili rizik od konfuzije ili grešaka.

Učenje naprednih komandi i funkcionalnosti - Učite i istražujte napredne komande i funkcionalnosti GIT-a kako biste postali učinkovitiji u upravljanju promjenama i rješavanju problema koji se mogu pojaviti tokom razvoja.

7. PRIMJER UPOTREBE

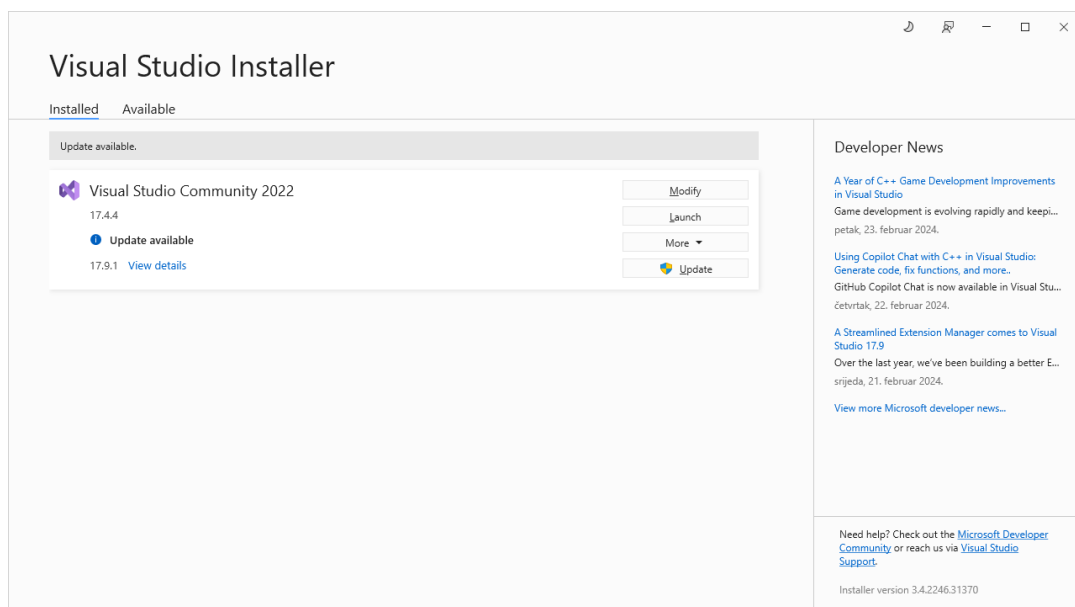
Za naš primjer upotrebe GIT-a biće potrebno da prezmemo neke aplikacije sa interneta, prvenstveno GIT, zatim Visual Studio, MySQL, te XAMPP-a.

Slika 16 - Instalacija GIT-a

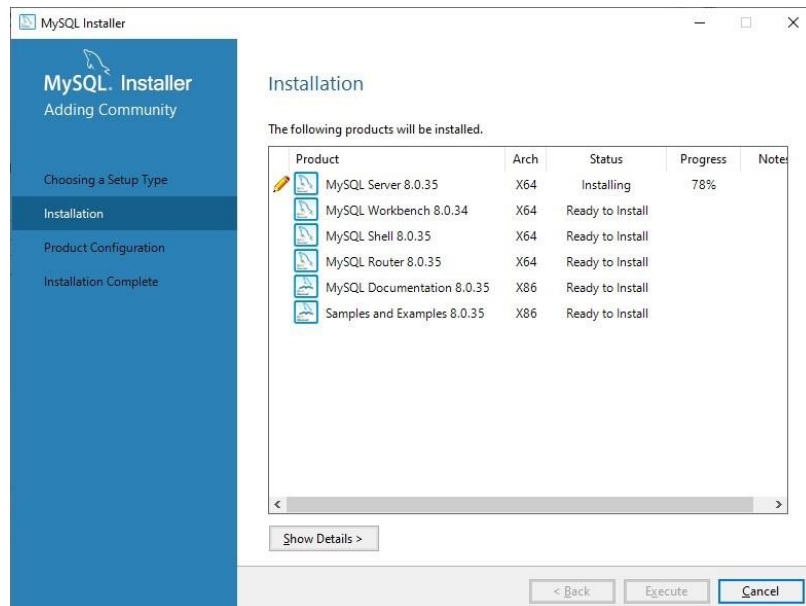


Za potrebe ovog primjera prvo se pristupilo instalaciji GIT-a.

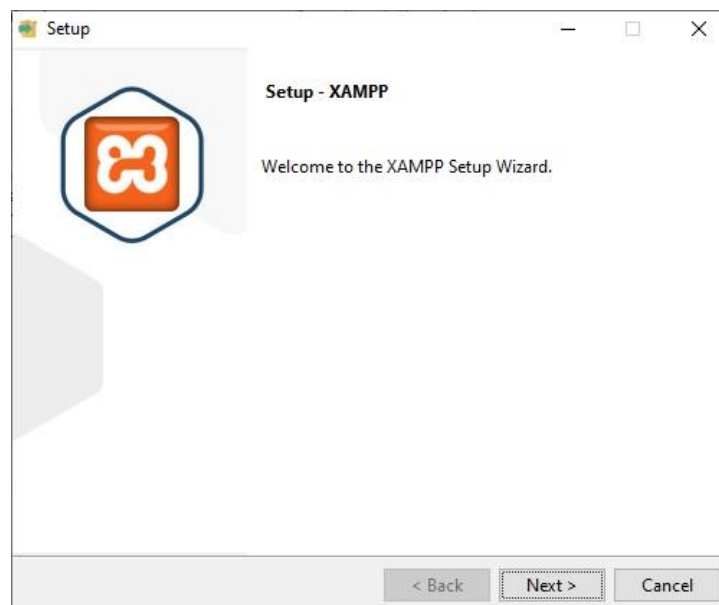
Slika 17 – Instalacija Visual Studio 2022



Zatim Visual Studio 2022.

Slika 18 – Instalacija MySQL-a

Nakon toga instalaciji MySQL-a.

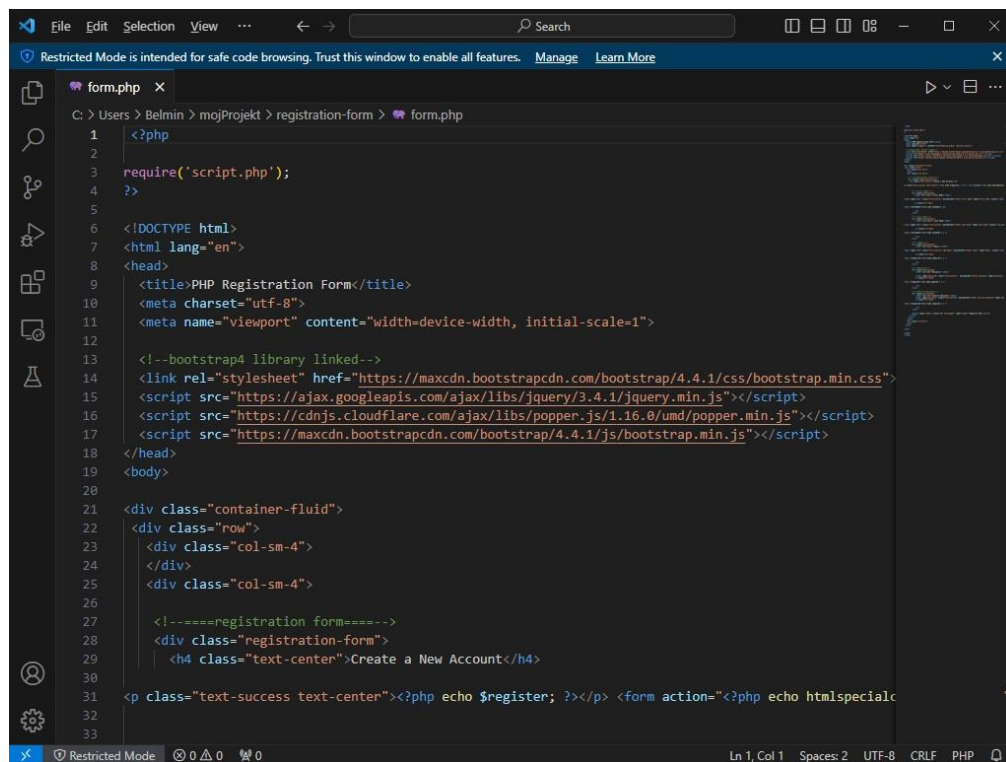
Slika 19 – Instalacija XAMPP-a

I konačno instalacija XAMPP-a.

U našem primjeru ćemo napraviti jednostavnu formu za registraciju u PHP-u, koju ćemo povezati sa SQL bazom podataka, koju ćemo napraviti u MySQL, te putem XAMPP aplikacije testirati u localhostu.

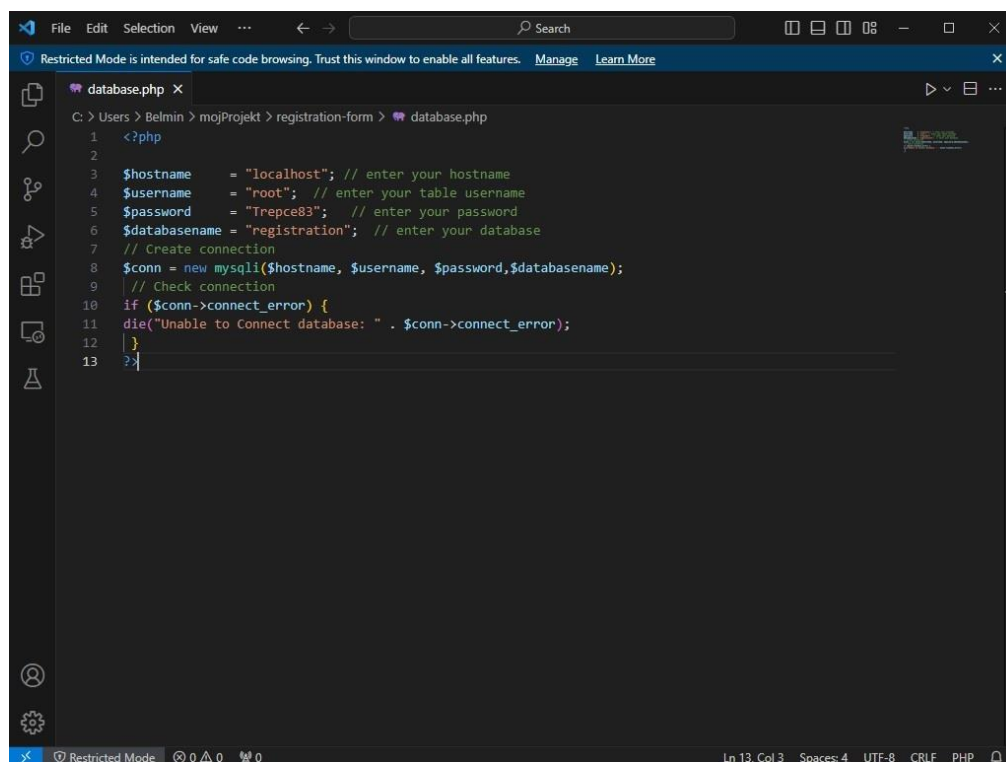
Kodovi za registracijsku formu su preuzeti sa web stranice <https://codingstatus.com/php-registration-form/>⁵

Slika 20 – Kod za formu registracije form.php



```
1 <?php
2
3 require('script.php');
4 ?>
5
6 <!DOCTYPE html>
7 <html lang="en">
8 <head>
9 <title>PHP Registration Form</title>
10 <meta charset="utf-8">
11 <meta name="viewport" content="width=device-width, initial-scale=1">
12
13 <!--bootstrap4 library linked-->
14 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
15 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
16 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
17 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>
18 </head>
19 <body>
20
21 <div class="container-fluid">
22 <div class="row">
23 <div class="col-sm-4">
24 </div>
25 <div class="col-sm-4">
26
27 <!--====registration form====-->
28 <div class="registration-form">
29 <h4 class="text-center">Create a New Account</h4>
30
31 <p class="text-success text-center"><?php echo $register; ?></p> <form action="<?php echo htmlspecialchars($register); ?>">
32
33
```

Slika 21 – Kod za povezivanje s bazom podataka database.php



```
1 <?php
2
3 $hostname = "localhost"; // enter your hostname
4 $username = "root"; // enter your table username
5 $password = "Trepce83"; // enter your password
6 $databasename = "registration"; // enter your database
7 // Create connection
8 $conn = new mysqli($hostname, $username, $password, $databasename);
9 // Check connection
10 if ($conn->connect_error) {
11 die("Unable to Connect database: " . $conn->connect_error);
12 }
13 ?>
```

⁵ <https://codingstatus.com/php-registration-form/>

Slika 22 – Kod za provjeru upisa u registracijsku formu i bazu podataka script.php

```

1  <?php
2
3  require_once('database.php');
4  $db= $conn; // update with your database connection
5  // by default, error messages are empty
6  $register=$valid=$fnameErr=$lnameErr=$emailErr=$passErr=$cpassErr='';
7  // by default, set input values are empty
8  $set_firstName=$set_lastName=$set_email='';
9
10 extract($_POST);
11 if(isset($_POST['submit']))
12 {
13
14
15 //input fields are Validated with regular expression
16 $validName="/^[a-zA-Z ]*$/";
17 $validEmail="/^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*\\.\\w{2,3}$/";
18 $uppercasePassword = "/(?=.*[A-Z])/";
19 $lowercasePassword = "/(?=.*[a-z])/";
20 $digitPassword = "/(?=.*[0-9])/";
21 $spacesPassword = "/$\\s+/";
22 $symbolPassword = "/(?=.*[!@#$%^&*])/";
23 $minEightPassword = "/.{8,}/";
24
25 // First Name Validation
26 if(empty($first_name)){
27     $fnameErr="First Name is Required";
28 }
29 else if (!preg_match($validName,$first_name)) {
30     $fnameErr="Digits are not allowed";
31 }else{
32     $fnameErr=true;
33 }
34

```

Slika 23 – Kreiranje baze podataka i tabele u MySQL-u

```

1  create database registration;
2  create table `users` (
3    `id` int(10) UNSIGNED PRIMARY KEY NOT NULL AUTO_INCREMENT,
4    `first_name` varchar(255) DEFAULT NULL,
5    `last_name` varchar(255) DEFAULT NULL,
6    `email` varchar(255) DEFAULT NULL,
7    `password` varchar(255) DEFAULT NULL,
8    `created_at` timestamp(6) DEFAULT NULL
9  );

```

#	Time	Action	Message	Duration / Fetch
1	11:20:45	create database registration	1 row(s) affected	0.016 sec
2	11:20:45	CREATE TABLE `users` (`id` int(10) UNSIGNED PRIMARY KEY NO...	Error Code: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ')' at line 8	0.000 sec
3	11:21:33	create database registration	Error Code: 1007 Can't create database; database exists	0.000 sec
4	11:21:33	CREATE TABLE `users` (`id` int(10) UNSIGNED PRIMARY KEY NO...	Error Code: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ')' at line 8	0.000 sec
5	11:24:16	CREATE TABLE `users` (`id` int(10) UNSIGNED PRIMARY KEY NO...	Error Code: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ')' at line 8	0.000 sec
6	11:24:48	CREATE TABLE `users` (`id` int(10) UNSIGNED PRIMARY KEY NO...	Error Code: 1050 Table 'users' already exists	0.000 sec

Po preuzimanju i instaliranju GIT aplikacije pristupili smo inicijalizaciji GIT repozitorija komandom *git init mojProjekt*. U folder na putanji C:\Users\Belmin\mojProjekt smo dodali tri dokumenta sa kodovima: form.php, script.php i database.php.

Pozivanjem komande *git status* u komandnoj liniji našeg projekta dobili smo poruku:

```
Microsoft Windows [Version 10.0.19045.4046]
```

```
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\Belmin>cd mojProjekt
```

```
C:\Users\Belmin\mojProjekt>git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

```
(use „git add <file>...” to include in what be committed)
```

```
database.php
```

```
form.php
```

```
script.php
```

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

```
C:\Users\Belmin\mojProjekt>
```

Zatim smo pozvali komandu *git add form.php*, kojom smo dodali fajl *form.php* u staging area. Nakon dodavanja fajla *form.php*, te ponovnog pozivanja komande *git status* dobili smo poruku sljedećeg sadržaja:

```
Microsoft Windows [Version 10.0.19045.4046]
```

```
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\Belmin>cd mojProjekt
```

```
C:\Users\Belmin\mojProjekt>git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use „git rm --cached <file>...” to unstage)
```

```
new file:   form.php
```

```
Untracked files:
```

```
(use „git add <file>...” to include in what be committed)
```

```
database.php
```

```
script.php
```

```
C:\Users\Belmin\mojProjekt>
```

Zatim smo komandom *git commit -m „Dodao novu formu“* izvršili komitanje fajla *form.php*, a po izvršenju iste dobili poruku:

```
C:\Users\Belmin\mojProjekt>git commit -m „Dodao novu formu“
```

```
[master (root-commit) 77da228] Dodao novu formu
```

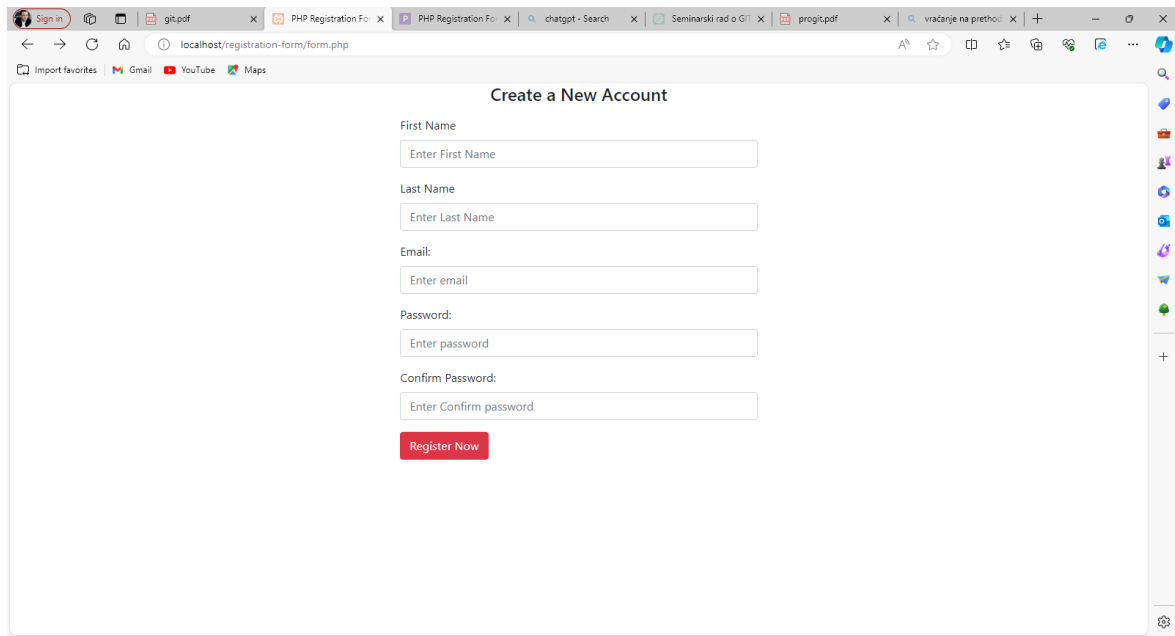
```
1 file changed, 107 insertions(+)
```

```
create mode 100644 form.php
```

Izvršili smo dodavanje i ostala dva fajla komandom *git add*, te ih komitovali komandom *git commit -m "Dodao ostale datoteke iz foldera"*.

Pokretanjem Apache servera u XAMPP-u možemo pokrenuti svoju registracijsku formu.

Slika 24 – Izgled registracijske forme

The image shows a web browser window with multiple tabs. The active tab is titled 'PHP Registration Fo' and shows a registration form. The browser's address bar displays 'localhost/registration-form/form.php'. The form itself is titled 'Create a New Account' and contains five input fields: 'First Name' (placeholder: 'Enter First Name'), 'Last Name' (placeholder: 'Enter Last Name'), 'Email:' (placeholder: 'Enter email'), 'Password:' (placeholder: 'Enter password'), and 'Confirm Password:' (placeholder: 'Enter Confirm password'). Below these fields is a red button labeled 'Register Now'. The browser's sidebar on the right shows various icons for extensions and settings.

Izvršit ćemo jezičke izmjene na našoj formi, te ćemo nakon toga dodati izmjene u naš repozitorij na lokalnom računaru. Nakon izmjene fajla u našem repozitoriji, i pozivanjem komande *git status* dobijamo poruku slijedećeg sadržaja:

```
Microsoft Windows [Version 10.0.19045.4046]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\Belmin>cd mojProjekt
```

```
C:\Users\Belmin\mojProjekt>git status
```

```
On branch master
```

```
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:   form.php
```

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

```
C:\Users\Belmin\mojProjekt>
```


Pozivom komande *git diff* dobićemo razlike koje smo napravili u kodu, u odnosu na kod koji je već komitovan u repozitorij, sa izgledom koda:

```
C:\Users\Belmin\mojProjekt>git diff
diff --git a/form.php b/form.php
index 12ee450..a88053d 100644
--- a/form.php
+++ b/form.php
@@ -6,7 +6,7 @@ require('script.php');
<!DOCTYPE html>
<html lang="en">
<head>
- <title>PHP Registration Form</title>
+ <title>PHP Registracijska forma</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

@@ -26,16 +26,16 @@ require('script.php');

  <!--====registration form====-->
  <div class="registration-form">
-     <h4 class="text-center">Create a New Account</h4>
+     <h4 class="text-center">Kreirajte novi račun</h4>

  <p class="text-success text-center"><?php echo $register; ?></p> <form
action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>"
method="post">

    <!--//first name//-->
    <div class="form-group">
-         <label for="email">First Name</label>
+         <label for="email">Ime</label>

- <input type="text" class="form-control" placeholder="Enter First Name"
name="first_name" value="<?php echo $set_firstName;?>">
+ <input type="text" class="form-control" placeholder="Unesite Vaše ime"
name="first_name" value="<?php echo $set_firstName;?>">

    <p class="err-msg">

@@ -46,9 +46,9 @@ require('script.php');
```

```

        <!--//Last name/-->
        <div class="form-group">
-           <label for="email">Last Name</label>
+           <label for="email">Prezime</label>
-       <input type="text" class="form-control" placeholder="Enter Last Name"
name="last_name" value="<?php echo $set_lastName;?>">
+       <input type="text" class="form-control" placeholder="Unesite Vaše
prezime" name="last_name" value="<?php echo $set_lastName;?>">

        <p class="err-msg">

@@ -61,7 +61,7 @@ require('script.php');
        <div class="form-group">
            <label for="email">Email:</label>

-       <input type="text" class="form-control" id="email" placeholder="Enter
email" name="email" value="<?php echo $set_email;?>">
+       <input type="text" class="form-control" id="email" placeholder="Unesite
Vaš email" name="email" value="<?php echo $set_email;?>">

        <p class="err-msg">

@@ -72,9 +72,9 @@ require('script.php');

        <!--//Password/-->
        <div class="form-group">
-           <label for="pwd">Password:</label>
+           <label for="pwd">Lozinka:</label>

-           <input type="password" class="form-control"
placeholder="Enter password" name="password"
+           <input type="password" class="form-control"
placeholder="Unesite lozinku" name="password"
            <p class="err-msg">

        <?php if($passErr!=1){ echo $passErr; } ?>
@@ -84,8 +84,8 @@ require('script.php');

        <!--//Confirm Password/-->
        <div class="form-group">

```

```

-         <label for="pwd">Confirm Password:</label>
-         <input type="password" class="form-control"
placeholder="Enter Confirm password" name="cpassword">
+         <label for="pwd">Potvrda lozinke:</label>
+         <input type="password" class="form-control"
placeholder="Unesite ponovo lozinku" name="cpassword">
         <p class="err-msg">

```

```

<?php if($cpassErr!=1){ echo $cpassErr; } ?>
@@ -93,7 +93,7 @@ require('script.php');
        </p>
    </div>

```

```

-         <button type="submit" class="btn btn-danger"
name="submit">Register Now</button>
+         <button type="submit" class="btn btn-danger"
name="submit">Registruj se</button>
        </form>
    </div>
</div>

```

(END)

Sada imamo i novi izgled naše stranice za registraciju:

Slika 25 – Stranica za registraciju sa prevodom

The screenshot shows a web browser window with the URL `localhost/registration-form/form.php`. The page title is "Kreirajte novi račun". The form contains the following fields:

- Ime: Unesite Vaše ime
- Prezime: Unesite Vaše prezime
- Email: Unesite Vaš email
- Lozinka: Unesite lozinku
- Potvrda lozinke: Unesite ponovo lozinku

At the bottom of the form is a red button labeled "Registruj se".

Sada ćemo komandom `git add form.php`, dodati fajl `form.php` u stage area, te komandom `git commit -m „Promjena jezika“` dodati naš izmjenjeni fajl u novi komit.

```
Microsoft Windows [Version 10.0.19045.4046]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\Belmin>cd mojProjekt
```

```
C:\Users\Belmin\mojProjekt>git status
```

```
On branch master
```

```
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:   form.php
```

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

```
C:\Users\Belmin\mojProjekt>git add form.php
```

```
C:\Users\Belmin\mojProjekt>git commit -m "Promjena jezika"
```

```
[master b088364] Promjena jezika
```

```
1 file changed, 12 insertions(+), 12 deletions(-)
```

```
C:\Users\Belmin\mojProjekt>
```

Komandom *git log* ćemo dobiti spisak svih komita koje smo napravili u navedenom projektu:

```
Microsoft Windows [Version 10.0.19045.4046]  
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Users\Belmin>cd mojProjekt
```

```
C:\Users\Belmin\mojProjekt>git log
```

```
commit b08836482acef9598ffb5cbf6bc38b9c5cd6f89a (HEAD -> master)
```

```
Author: KadusicB <kadusic_belmin@hotmail.com>
```

```
Date: Sat Feb 24 18:49:31 2024 +0100
```

Promjena jezika

```
commit 625ab419e8faed73d15023078f8919f6e8bbb2d7
```

```
Author: KadusicB <kadusic_belmin@hotmail.com>
```

```
Date: Sat Feb 24 14:39:26 2024 +0100
```

Dodao ostale datoteke iz foldera

commit 77da228fc4bfee9c427d6eade414b1809cd862fb

Author: KadusicB <kadusic_belmin@hotmail.com>

Date: Sat Feb 24 12:36:55 2024 +0100

Dodao novu formu

C:\Users\Belmin\mojProjekt>

Posljednjom komandom koju ćemo koristiti u ovom primjeru ćemo izvršiti dodavanje izmjena u repozitorij na Github-u

Microsoft Windows [Version 10.0.19045.4046]

(c) Microsoft Corporation. All rights reserved.

C:\Users\Belmin>cd mojProjekt

C:\Users\Belmin\mojProjekt>git push

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 4 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 522 bytes | 522.00 KiB/s, done.

Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

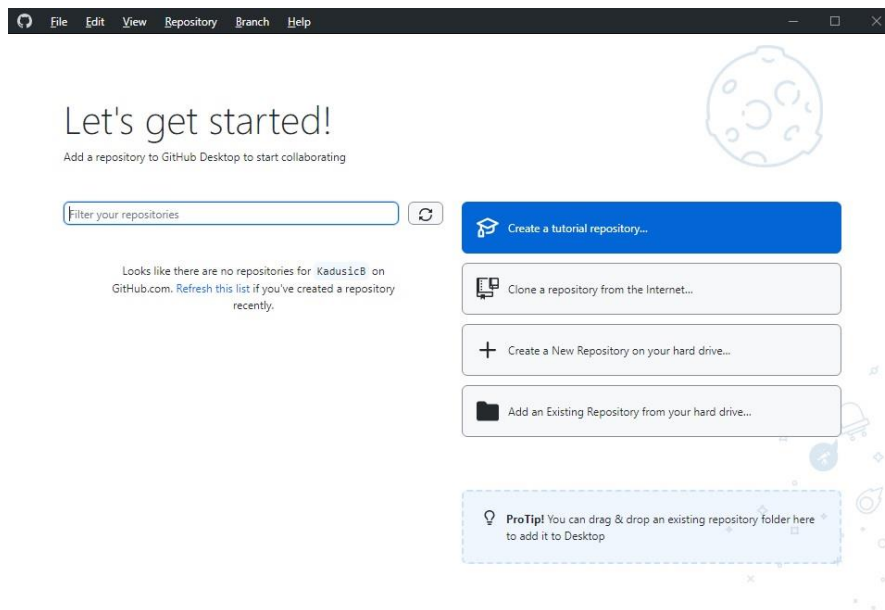
To https://github.com/KadusicB/mojProjekt.git

625ab41..b088364 master -> master

C:\Users\Belmin\mojProjekt>

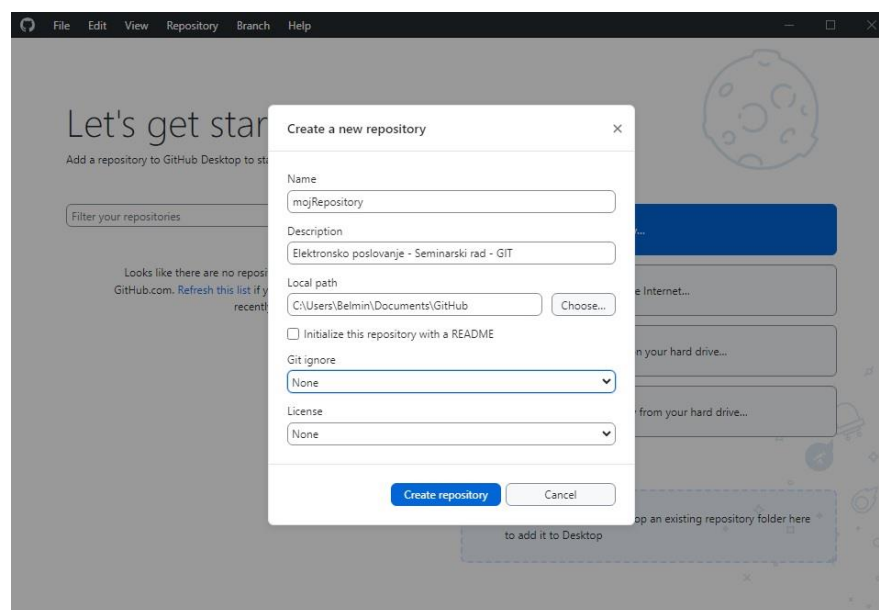
Također za ovaj projekat smo preuzeli i Git desktop aplikaciju, sa kojom se također može raditi sve ovo što smo radili u komandnoj liniji, ukoliko je nekom lakše raditi sa grafičkim interfejsom.

Slika 26 – Git desktop aplikacija



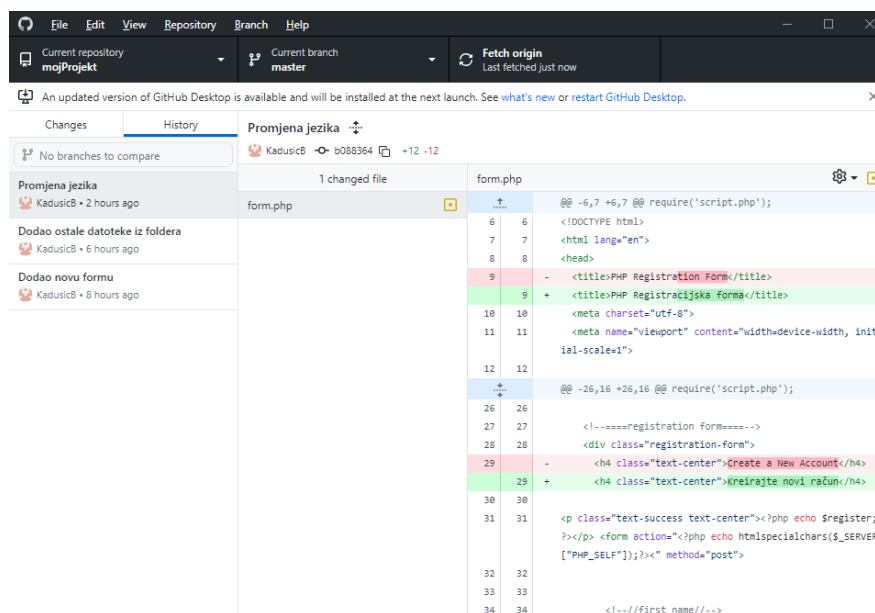
U desktop aplikaciji se može klonirati repozitorij sa interneta, kreirati novi repozitorij na lokalnom disku, dodati postojeći repozitorij sa lokalnog diska...

Slika 27 – Kreiranje novog repozitorija u Git grafičkom interfejsu



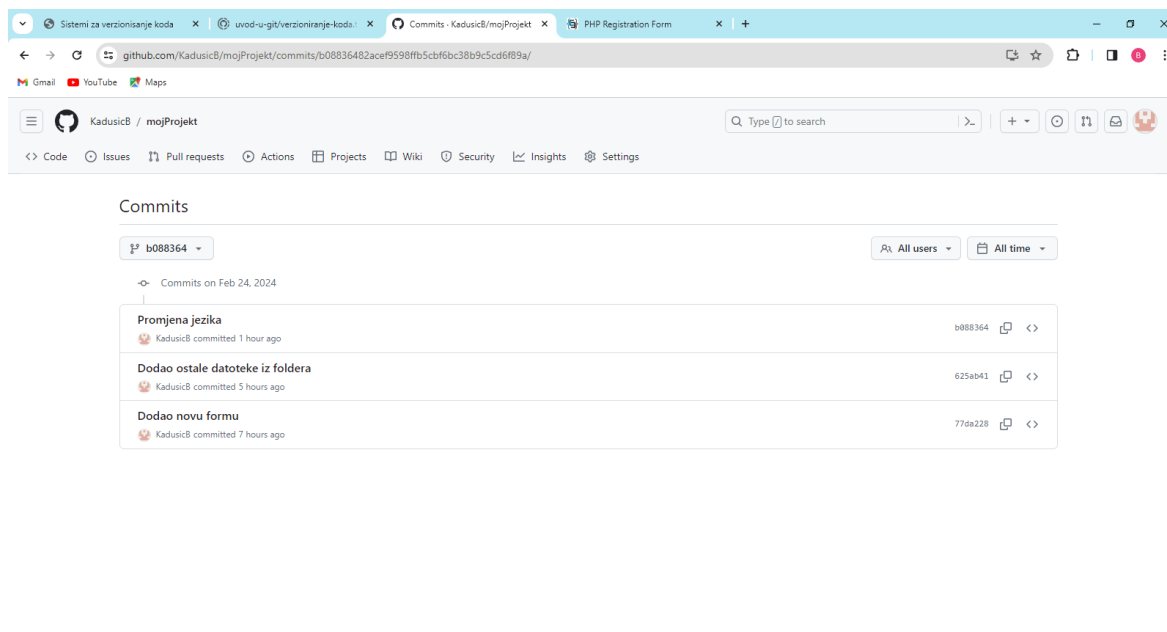
Također možemo vidjeti komite, promjene koje su napravljene u komitima, odnosno koji dio koda smo mijenjali ili dodavali u kojem komitu.

Slika 28 – Izgled komita u Git grafičkom interfejsu



Pristupanjem web stranici githuba, kreiranjem korisničkog imena i lozinke također se mogu pratiti sve radnje i promjene koje su se dešavale u repozitoriju, te se mogu dijeliti sa ostalim korisnicima radi unapređenja koda.

Slika 29 – Web stranica githuba korisnika KadusicB, sa pogledom na komite u mojProjekt repozitoriju



8. ZAKLJUČAK

U ovom seminarskom radu istražili smo sistem za upravljanje verzijama GIT, njegovu istoriju, osnovne operacije, dodatne alate i najbolje prakse. GIT je moćan alat koji omogućuje programerima učinkovito upravljanje promjenama u kodu, saradnju s drugima i održavanje istorije projekta.

Kroz analizu istorije verzioniranja koda, razvoja GIT-a i rastuće složenosti softverskih projekata, shvatili smo potrebu za naprednim sistemim za upravljanje verzijama kao što je GIT. Distribuirana priroda GIT-a, brzina, učinkovitost i podrška za razne operacije kao što su grananje i spajanje grana, čine ga neizostavnim alatom za moderni softverski razvoj.

Uz to, istražili smo dodatne alate i integracije koji nadopunjuju funkcionalnosti GIT-a, omogućujući programerima da još učinkovitije rade na projektima i surađuju s timom. Savjeti i najbolje prakse koje smo prezentirali pomažu u održavanju čistog, organiziranog i stabilnog repozitorija, olakšavajući razvoj softvera i saradnju unutar tima.

GIT ostaje esencijalni alat za programere i razvojne timove diljem svijeta, pomažući im u upravljanju kompleksnošću softverskih projekata, održavanju kvalitete koda i ostvarivanju uspješnih rezultata.

Na kraju, primjerima iz prakse, od potrebnih instalacija, preko kodova, do git komandi napravili smo repozitorij u kojem smo radili naš program, mijenjali ga, a sve to pratili preko komandne linije, GIT desktop aplikacije, i web stranice Github-a.

9. LITERATURA

1. Tomo Krajina, *Uvod u GIT*
2. Scott Chacon and Ben Straub, 2022, *ProGIT*
3. http://si3psi.etf.rs/materijali/labs/lab1/SistemiKontroleVerzija_v3.0.pdf, pristupljeno 09.02.2024.
4. Jon Loeliger, *Version Control with Git*
5. Ferdinando Santacroce, 2015, *Git essentials*, Birmingham - Mumbai

10.SPISAK SLIKA

Slika 1 – Službena stranica GIT-a	8
Slika 2 – Pokretanje instalacije	8
Slika 3 – Kraj instalacije	9
Slika 4 – Provjera uspješnosti instalacije GIT-a	9
Slika 5 – Konfiguracija korisničkog imena i e-mail adrese	10
Slika 6 – Incijalizacija GIT repozitorija „mojProjekt“	11
Slika 7 – Prazan GIT repozitorij na putanji „C:\Users\Belmin\mojProjekt“	11
Slika 8 – Dodavanje izmjene u staging area komandom git add form.php	12
Slika 9 – Upotreba komande komit sa dodavanjem komentara	12
Slika 10 – Upotreba komande git status kada smo izvršili promjene u folderu	13
Slika 11 – Upotreba komande git status poslije komande git add	13
Slika 12 – Upotrebe komande git checkout HEAD form.php	14
Slika 13 – Vraćanje na određeni komit komandom git checkout	14
Slika 14 - Upotreba komande git push	15
Slika 15 - Upotreba komande git pull	15
Slika 16 - Instalacija GIT-a	19
Slika 17 – Instalacija Visual Studio 2022	19
Slika 18 – Instalacija MySQL-a	20
Slika 19 – Instalacija XAMPP-a.....	20
Slika 20 – Kod za formu registracije form.php	21
Slika 21 – Kod za povezivanje s bazom podataka database.php	21
Slika 22 – Kod za provjeru upisa u registracijsku formu i bazu podataka script.php	22
Slika 23 – Kreiranje baze podataka i tabele u MySQL-u.....	22
Slika 24 – Izgled registracijske forme.....	24
Slika 25 – Stranica za registraciju sa prevodom.....	27
Slika 26 – Git desktop aplikacija.....	30
Slika 27 – Kreiranje novog repozitorija u Git grafičkom interfejsu	30
Slika 28 – Izgled komita u Git grafičkom interfejsu	31
Slika 29 – Web stranica githuba korisnika KadusicB, sa pogledom na komite u mojProjekt repozitoriju	31