Da vidimo sada kako se koristi komanda git ignore (kako da ne prikazujemo neke stvari)

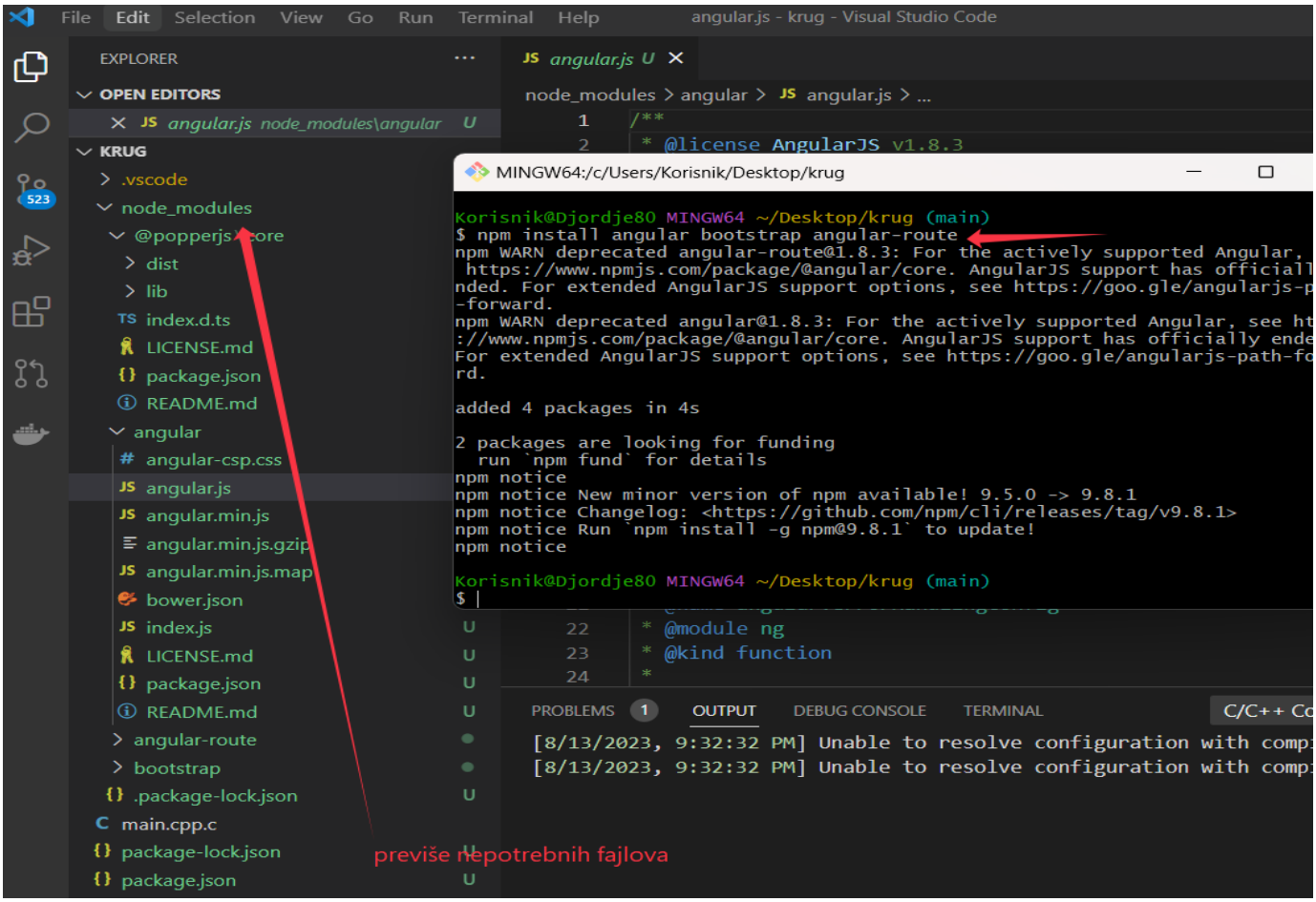
Kopirao, klonirao sam reposetorijum sa programom krug.

Instalirajmo sada npr node module sa npm

Tj kucamo.

npm install angular bootstrap angular-route

Postoje fajlovi, folderi koje je nepotrebno deliti .

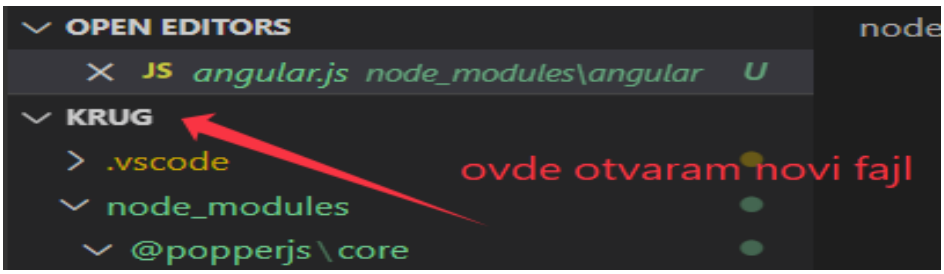


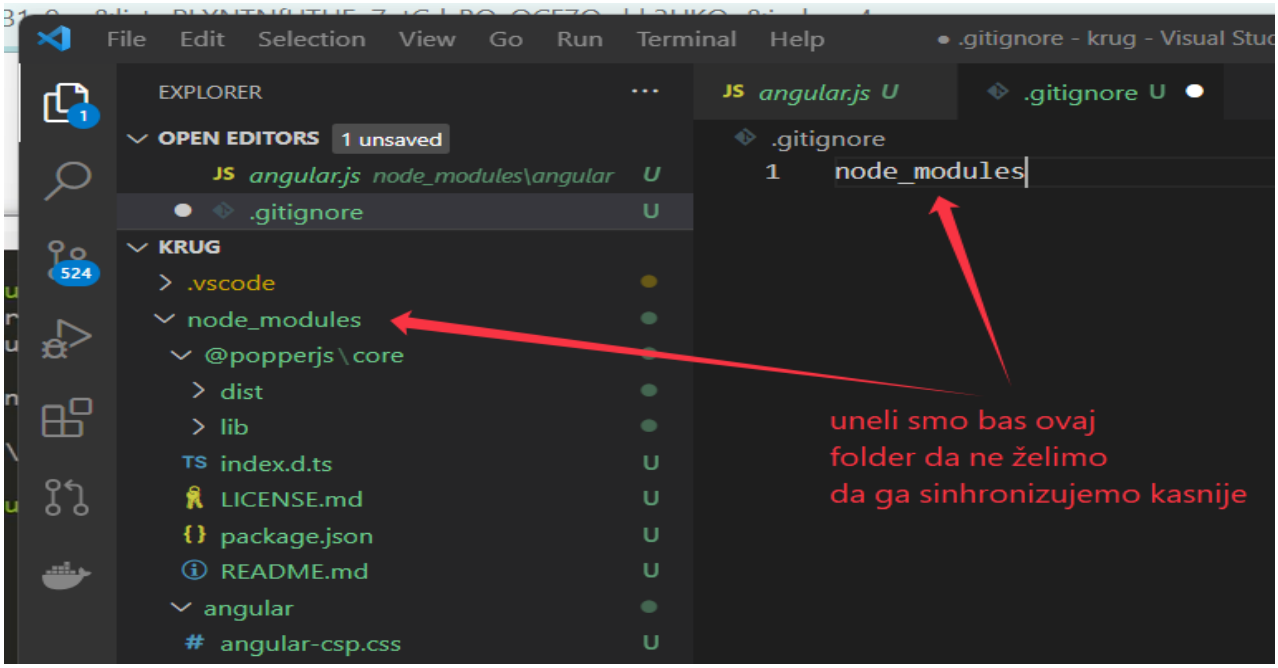
Instalirao sam sada neke node module čisto da vidite kada stavim opciju git ignore na neke nepotrebne stvari.

Ima kao što vidimo previše nebitnih fajlova tako da ćemo kucati u svom glavnom folderu novi fajl

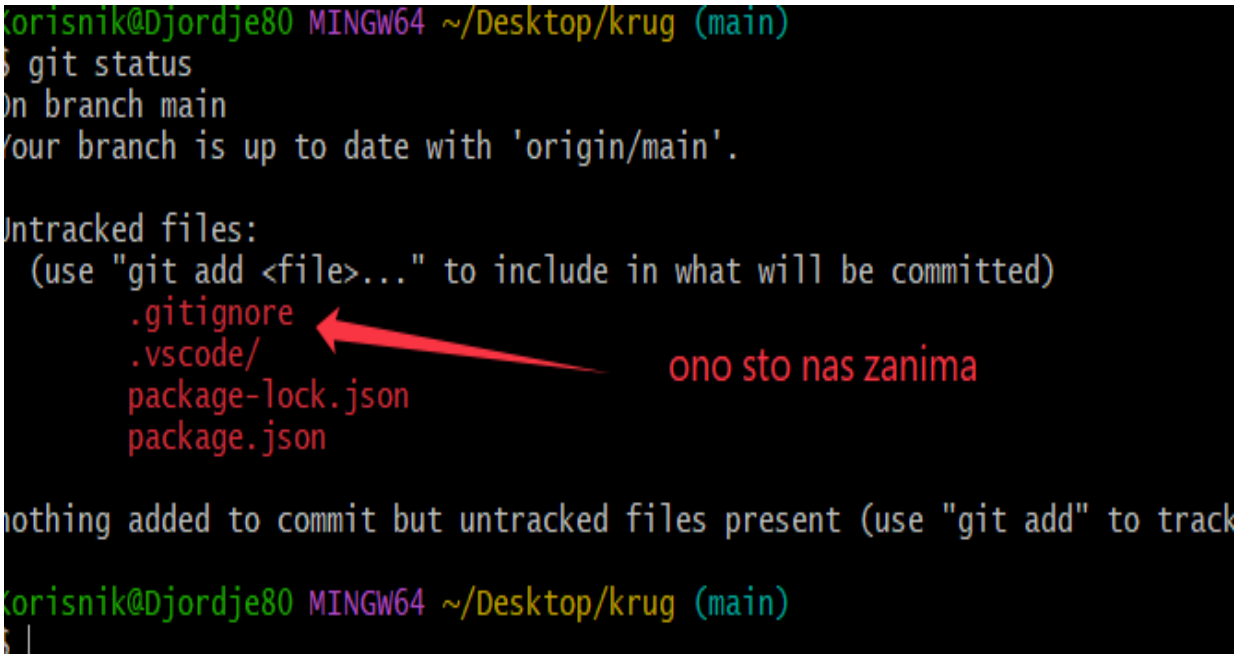
.gitignore (tu ćemo upisati šta da sam git softver ignoriše kada treba da sinhronizuje naše fajlove)

Mi ćemo staviti baš ovaj folder koji nam je nepotreban a učitali smo ga demostrativno node_modules





Vratimo se sada u naš terminal i pogledajmo git status da vidimo šta on pokazuje.



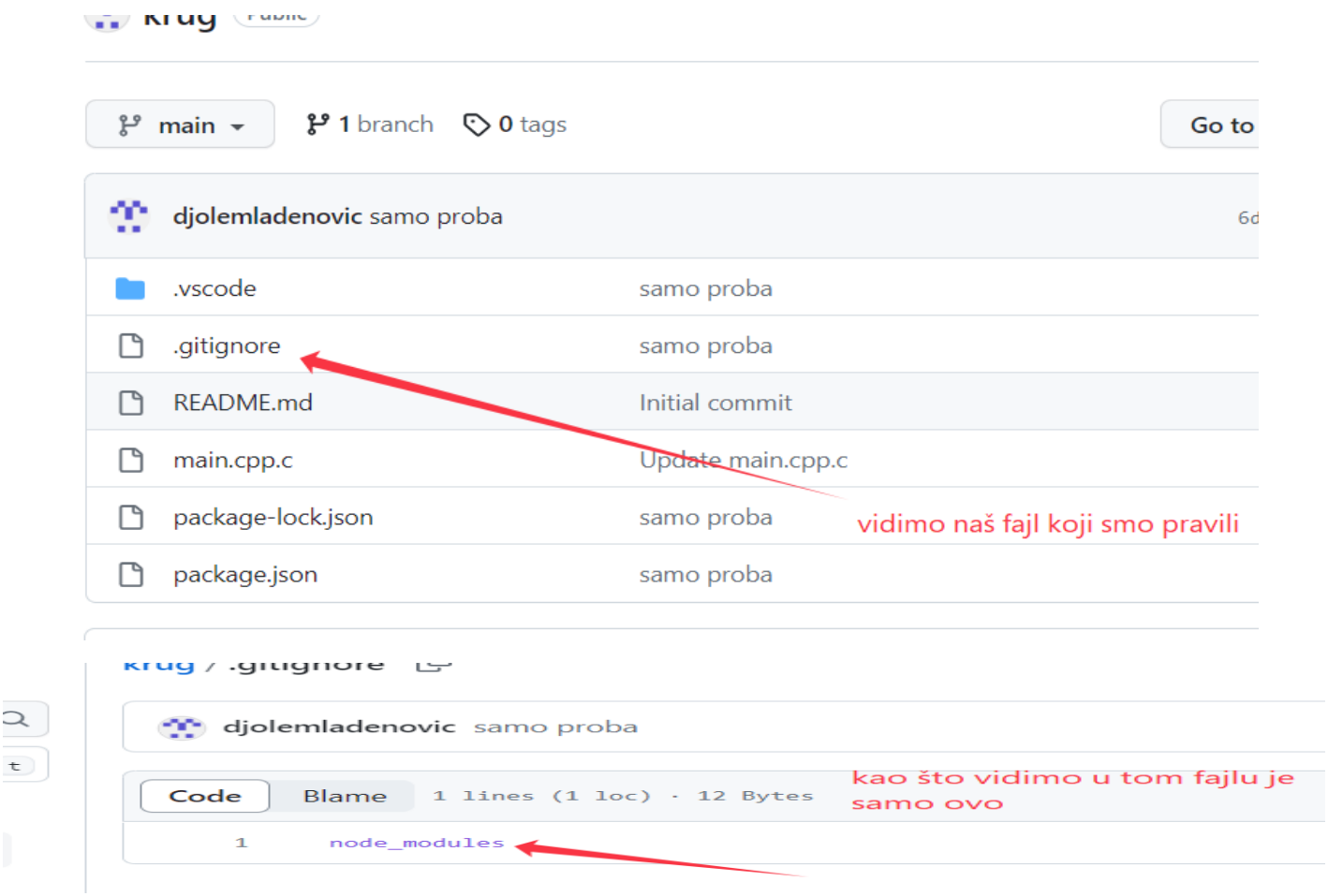
Imamo napravljeni gitignore fajl koji želimo da push-ujemo i idemo sa komandom git add . (. -Označava da sve dodajemo izmene na github)

Onda ide za potvrdu komanda git commit -m " Samo test ..."

Pa da onda to sinhronizujemo sa git push

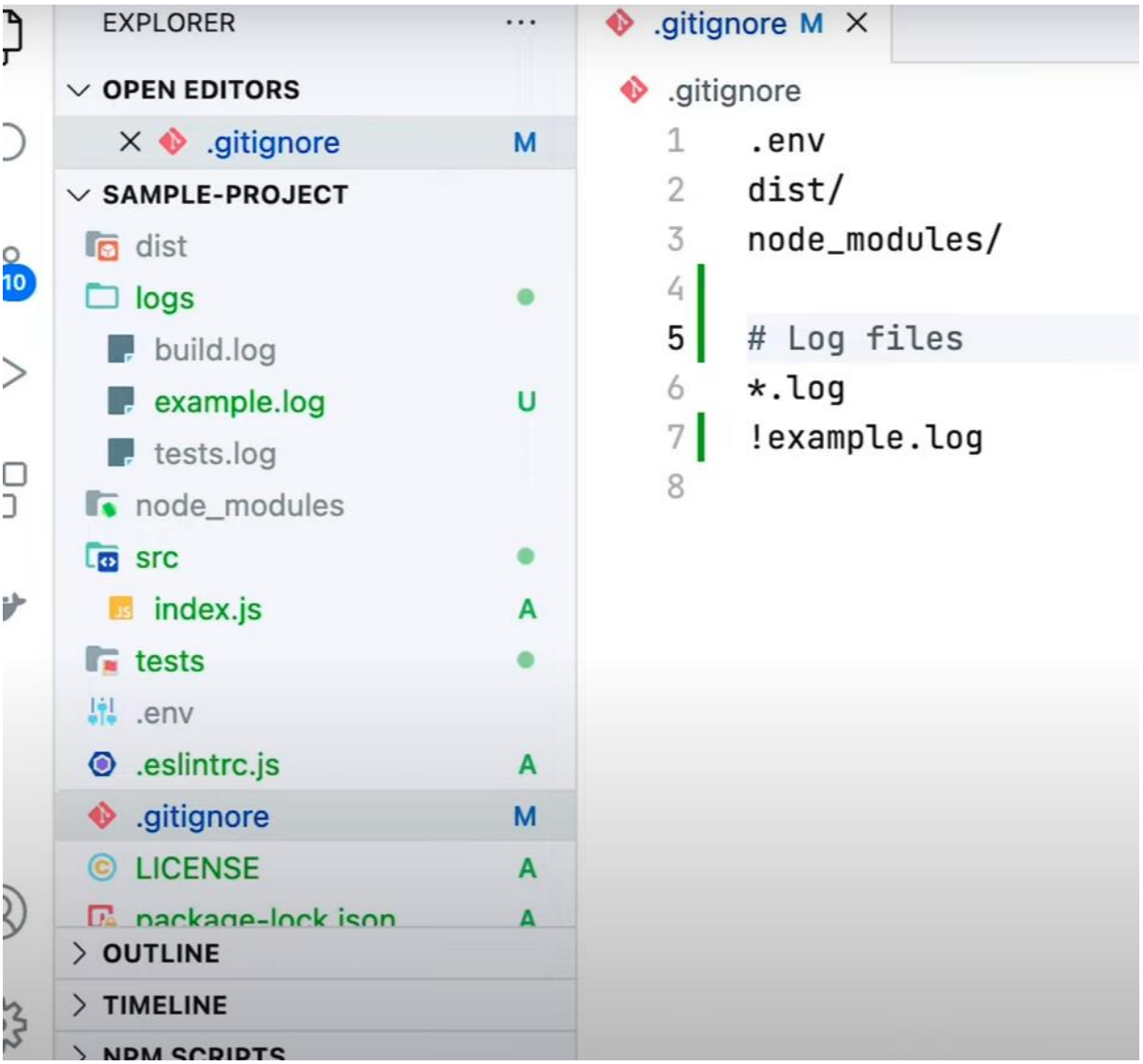
```
Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ git add .
warning: in the working copy of '.vscode/c_cpp_properties.json', LF will be repl
aced by CRLF the next time Git touches it
warning: in the working copy of '.vscode/launch.json', LF will be replaced by CR
LF the next time Git touches it
warning: in the working copy of '.vscode/settings.json', LF will be replaced by
CRLF the next time Git touches it
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF
 the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the
next time Git touches it

Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ git commit -m " samo proba "
[main 6d9b388]  samo proba
 6 files changed, 162 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 .vscode/c_cpp_properties.json
 create mode 100644 .vscode/launch.json
 create mode 100644 .vscode/settings.json
 create mode 100644 package-lock.json
 create mode 100644 package.json

Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 2.34 KiB | 599.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/djolemladenovic/krug.git
   fabddc8..6d9b388  main -> main

Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$
```

Idemo sada na naš GitHub i pogledajmo naš folder krug da vidimo da li se i ikako izvršila sinhronizacija.



Kao što vidimo u git ignore fajlu je samo jedna stvar a ne brdo fajlova koje bi korisnici morali svaki put da kopiraju kod sebe .

Ovde smo videli kako da napravimo fajl kada želimo da se nešto ne uključimo u naš projekat.



Ako želimo da ipak neki log fajl izuzmemo , kucaćemo !example.log

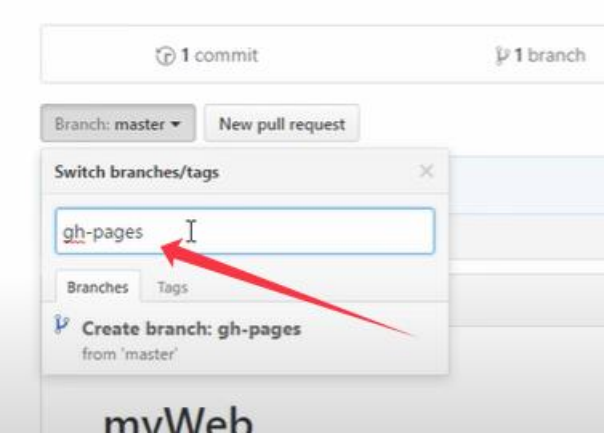
Ako kucamo *.log to se podrazumeva na sve fajlove koji se u logs

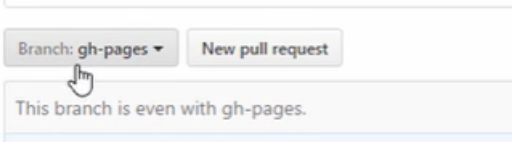
Da vidimo šta je GitHub Pages

To nije ništa drugo nego besplatan hosting za naš web sajt.

Krećemo od pravljenja new repository na GitHubu. Ukucajte kako želite da se zove novi repozitorijum.

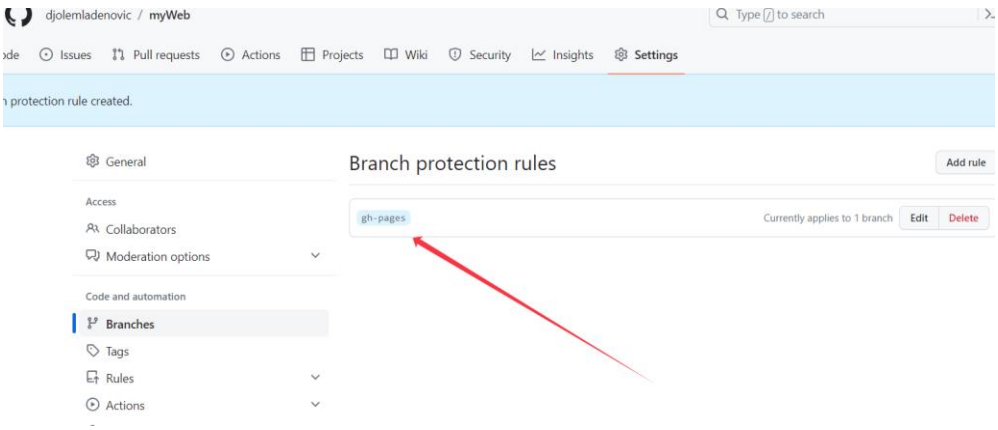
Mi smo tu na našem master Branch-u , tu ćemo ukucati gh-pages





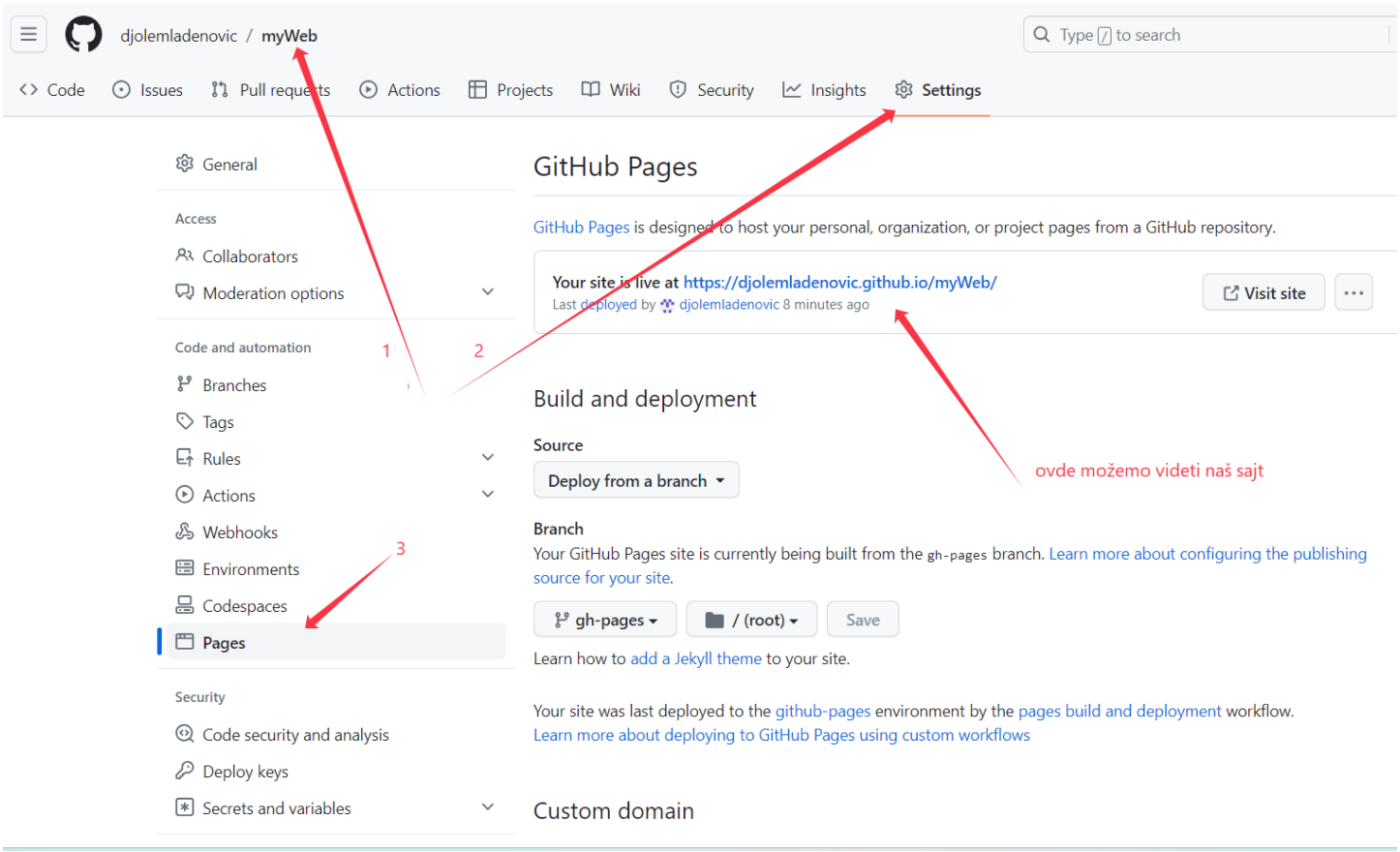
I kreiraćemo taj novi Brench

Tu i dalje postoji master brench (main) i on je da kažem i dalje glavni brench e to moramo promeiti u setings-u.



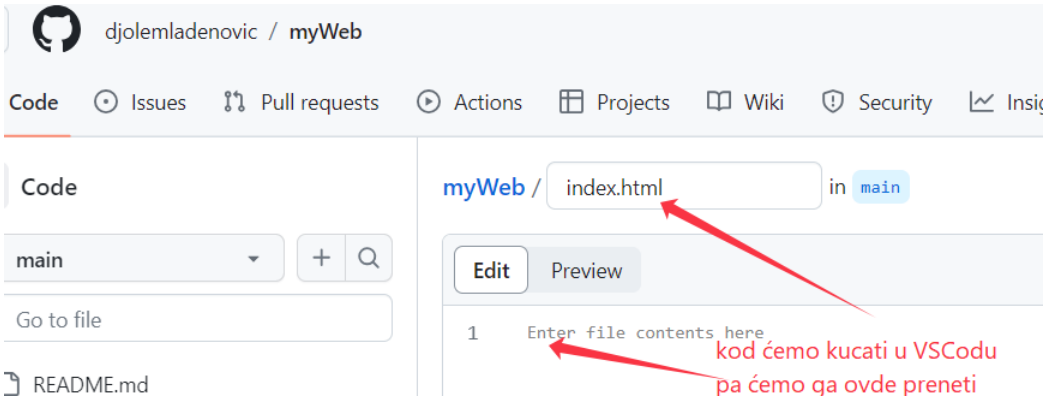
Kada ponovo odemo na naš dokument myWeb i u setings-u odemo na opciju Pages možemo videti da je sada prikazana

putanja do naše veb strane.

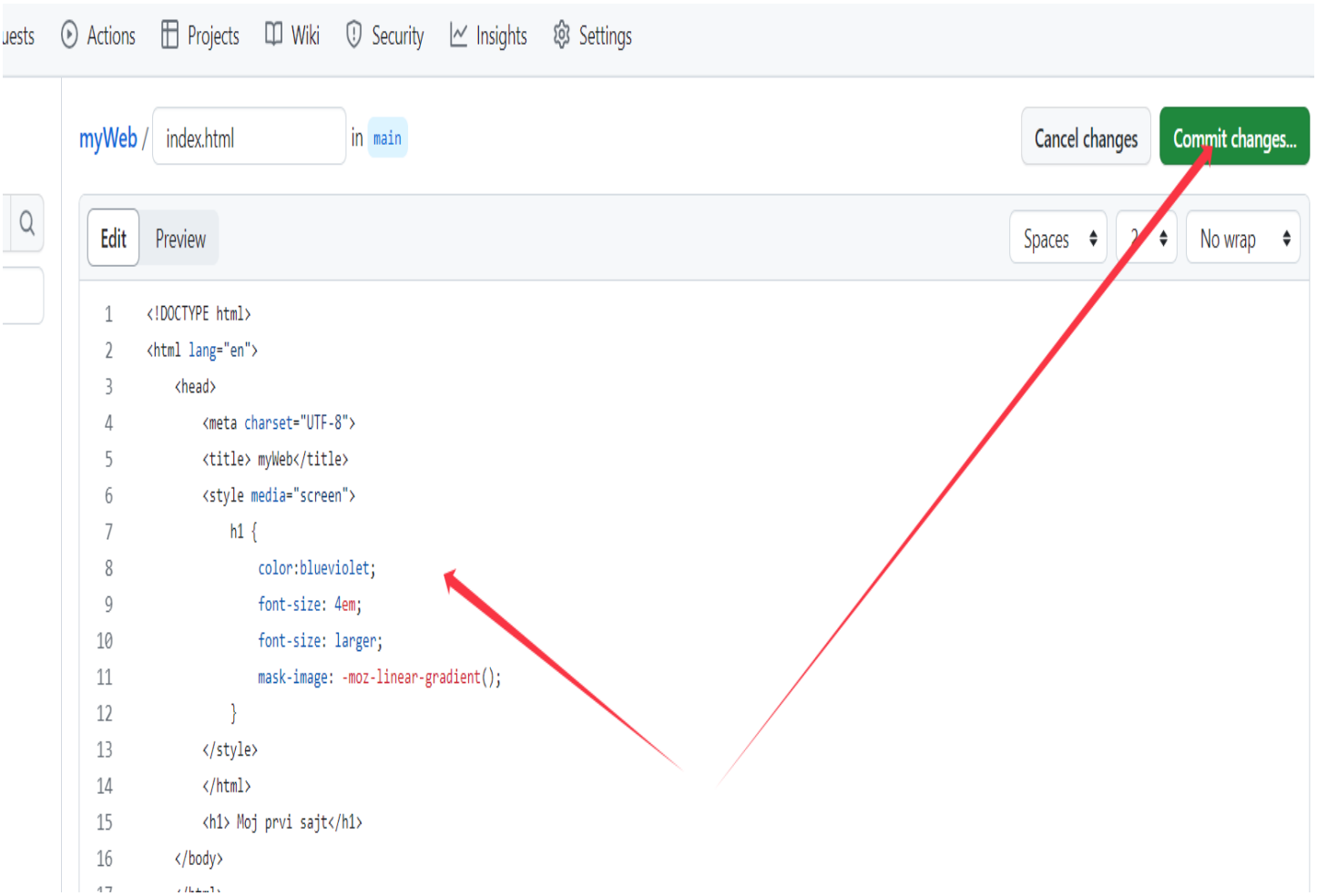


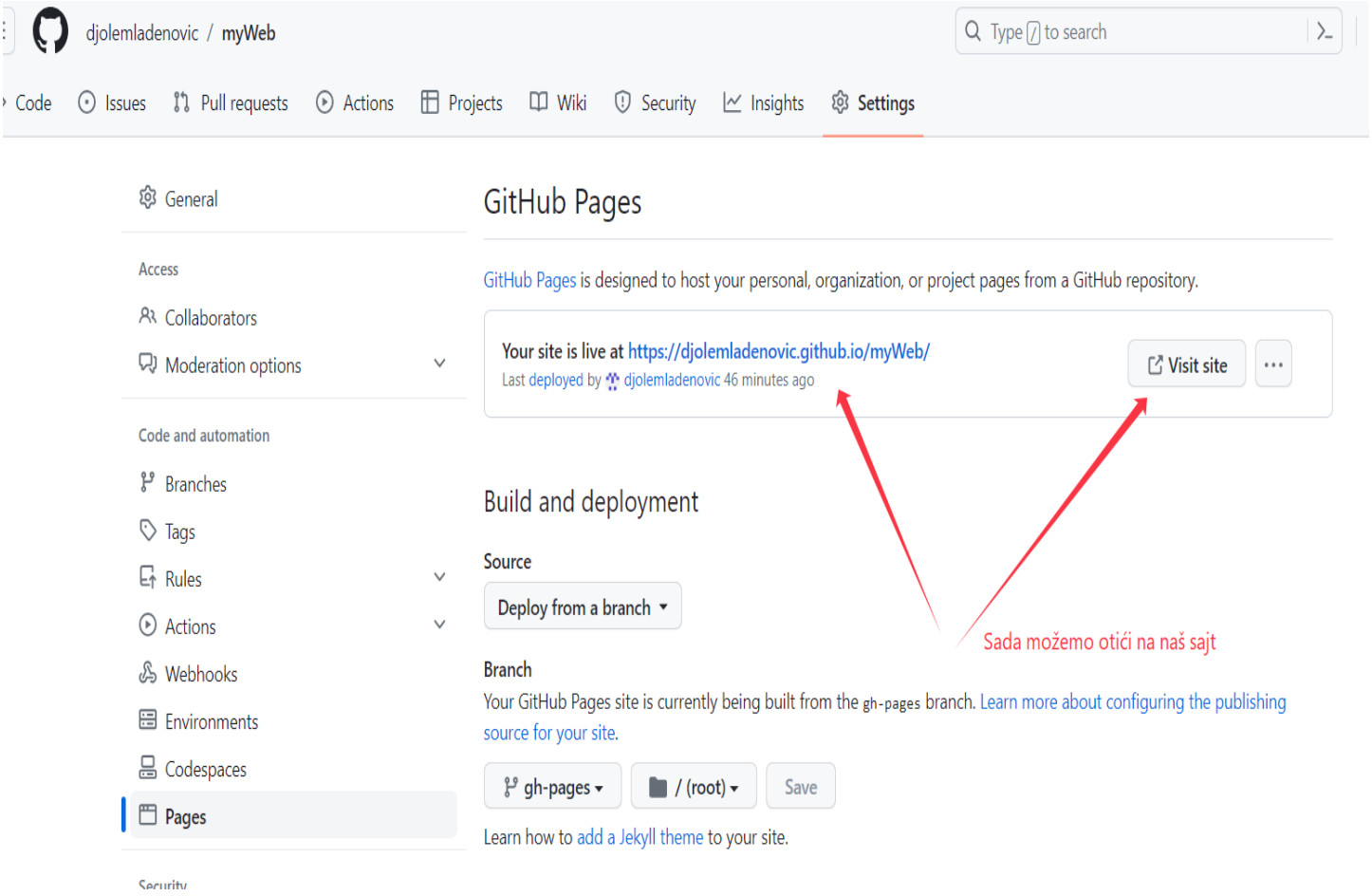
Sada naravno nemam ništa još uvek zato što mije folder prazan ali hajde da stavimo neki fajl i da napišemo neki kod.

Nazovimo ga index.html (naravno prva stranica koja će se otvoriti)



Uneo sam sada neki program pisan u html-u i sacuvacu izmene tj. commit changes

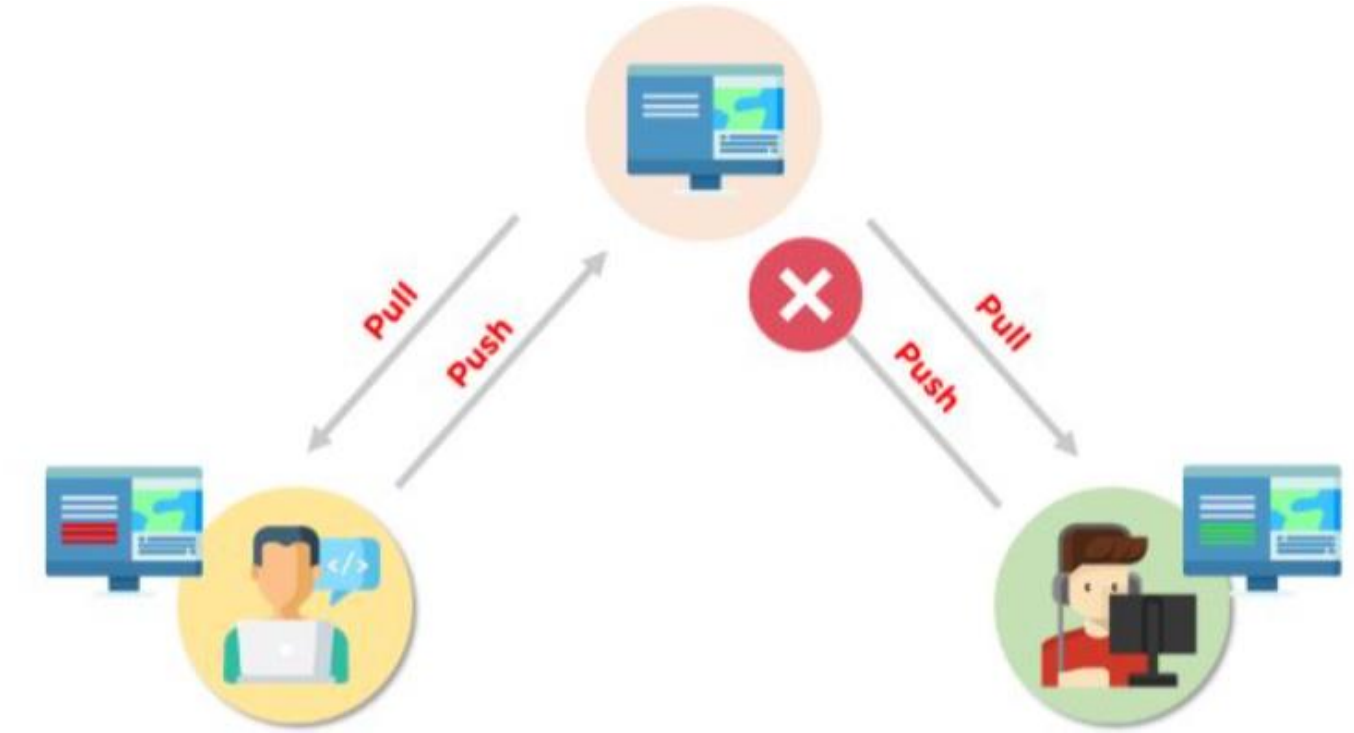

uests
Actions
Projects
Wiki
Security
Insights
Settings
myWeb /
index.html
in
main
Cancel changes
Commit changes...
Edit
Preview
Spaces
No wrap
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <title> myWeb</title>
6 <style media="screen">
7 h1 {
8 color:blueviolet;
9 font-size: 4em;
10 font-size: larger;
11 mask-image: -moz-linear-gradient();
12 }
13 </style>
14 </html>
15 <h1> Moj prvi sajt</h1>
16 </body>


djolemladenovic / myWeb
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://djolemladenovic.github.io/myWeb/
Last deployed by djolemladenovic 46 minutes ago
Visit site
Build and deployment
Source
Deploy from a branch
Sada možemo otići na naš sajt
Branch
Your GitHub Pages site is currently being built from the gh-pages branch. Learn more about configuring the publishing source for your site.
gh-pages
/ (root)
Save
Learn how to add a Jekyll theme to your site.

What is a Git Merge Conflict? (Šta je neusaglašenost pri objedinjavanju)

Neusaglašenost pri objedinjavanju git je događaj koji se dešava kada Git nije u stanju da automatski reši razlike u kodu između dva komita. Git može automatski da objedini promene samo ako su komiti na različitim linijama ili granama.

Sledi primer kako funkcioniše neusaglašenost git objedinjavanja:



Pretpostavimo da postoje dva programera: Developer A i Developer B. Obojica izvlače isti kodni fajl iz udaljenog skladišta i pokušavaju da naprave razne amandmane u tom dosijeu. Nakon što izvrši izmene, Developer A gura datoteku nazad u udaljeno skladište iz svog lokalnog skladišta. Sada, kada Developer B pokuša da pritisne tu datoteku nakon što izvrši promene sa svog kraja, on to nije u mogućnosti, jer je datoteka već promenjena u udaljenom skladištu.

Da bi sprečili takve konflikte, programeri rade u **odvojenim izolovanim granama**. Komanda za objedinjavanje Git kombinuje zasebne grane i rešava sva neusaglašena uređivanja.

Sada kada smo prošli kroz osnove sukoba Git mergea, pogledajmo sledeće različite vrste konflikata.

Kako otkloniti neusaglašenosti objedinjavanja u Gitu?

Postoji nekoliko koraka koji bi mogli da smanje korake potrebne za rešavanje neusaglašenosti objedinjavanja u Git-u.

1. korak: Najlakši način za otklanjanje neusaglašene datoteke je da je otvorite i izvršite sve neophodne promene.

2. korak: Nakon uređivanja datoteke, možemo da koristimo git dodavanje komande za fazu novog objedinjenog sadržaja.

Korak 3: Poslednji korak je stvaranje novog posvećivanja uz pomoć komande git commit.

4. korak: Git će kreirati novo objedinjavanje da bi dovršio objedinjavanje.

Git Commands to Resolve Conflicts

1. git log --merge

The git log --merge komanda pomaže u izradi liste posvećivanja koja uzrokuje neusaglašenost.

2. git diff

The git diff command pomaže u identifikovanju razlika između depoa ili datoteka država.

3. git checkout

The git checkout command koristi se za opozivanje promena izvršenih u datoteci ili za promenu grana.

4. git reset --mixed

The git reset --mixed command koristi se za opoziv promena u radnom direktorijumu i prostoru za pripremu.

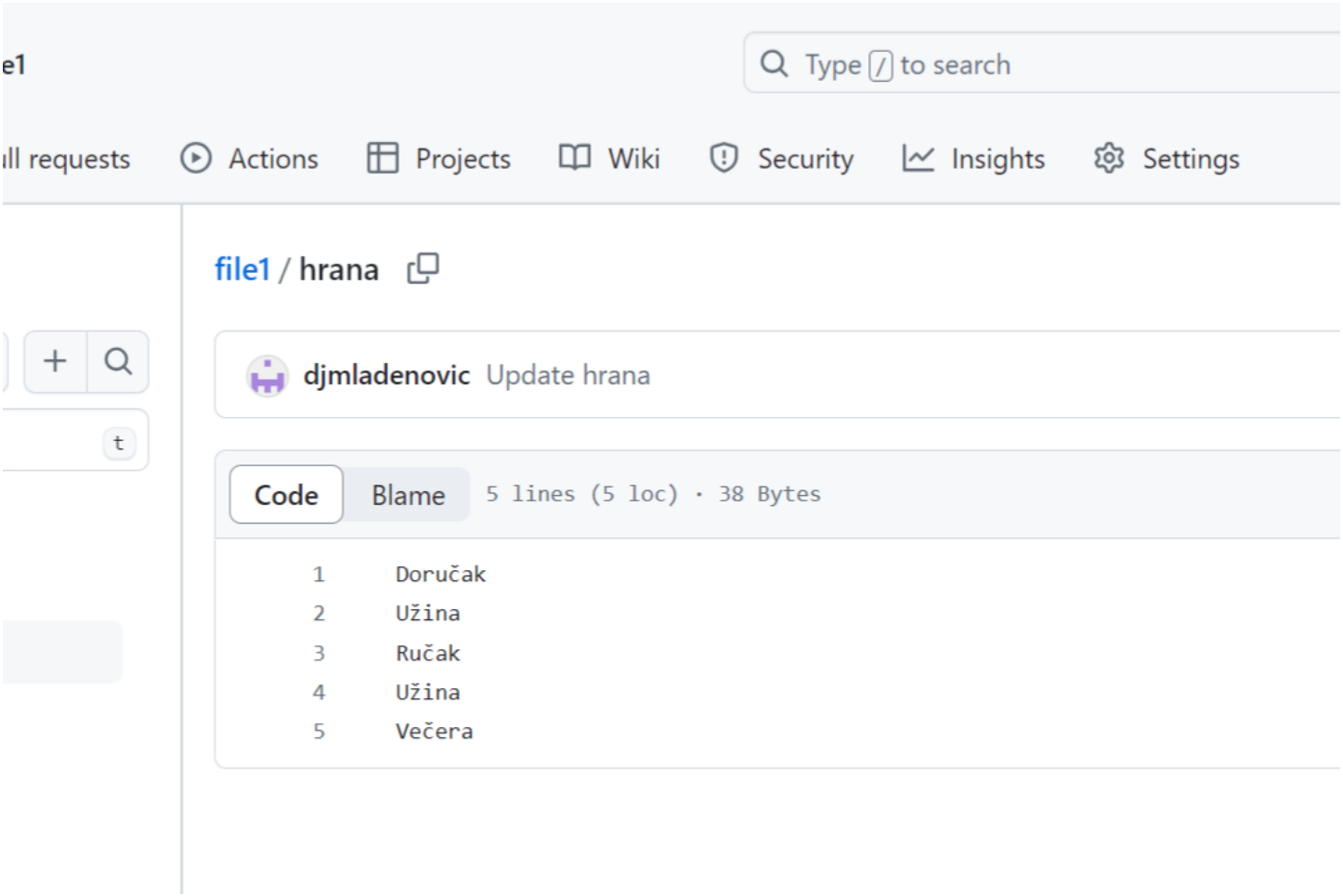
5. git merge --abort

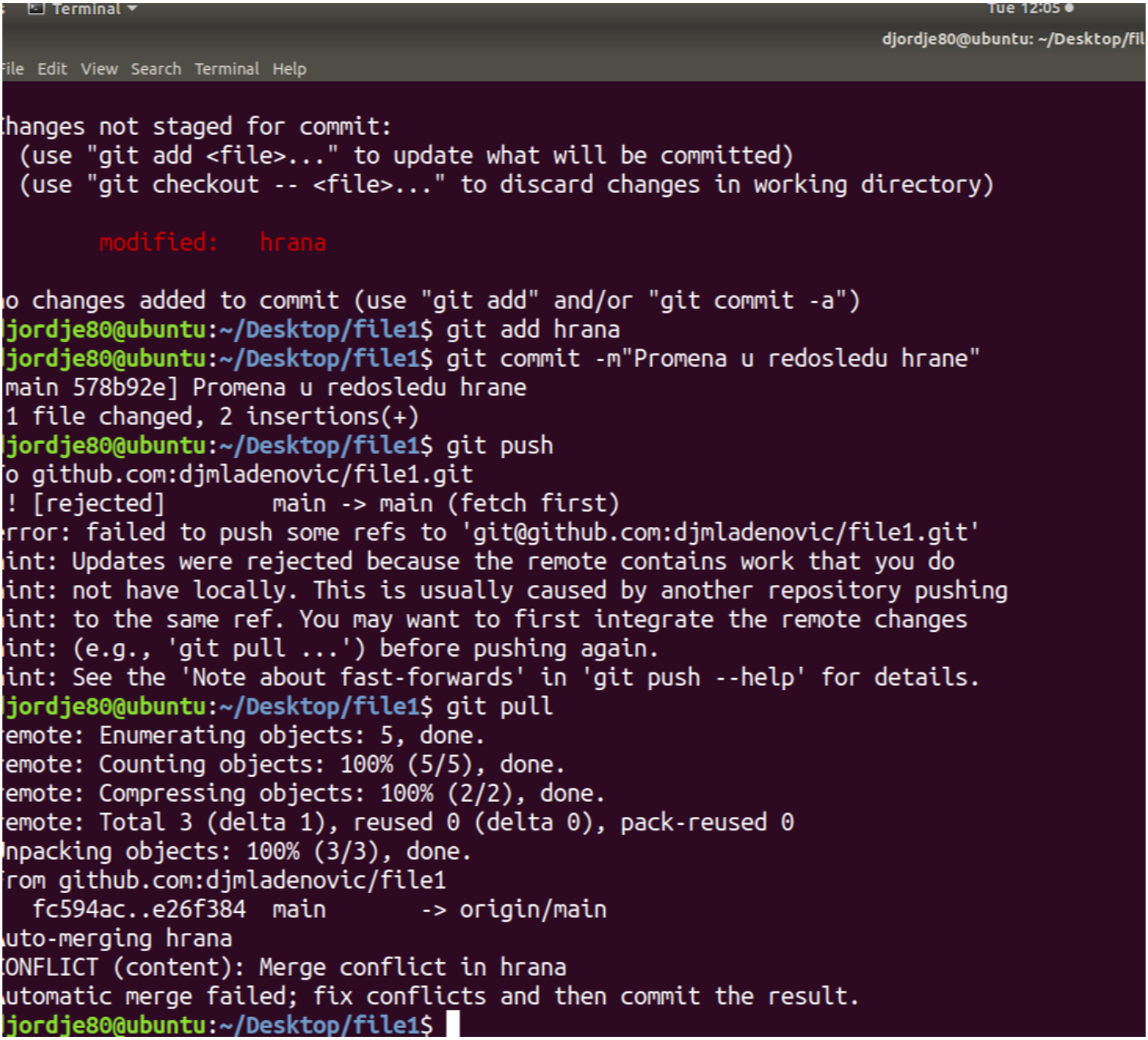
The git merge --abort command pomaže u izlasku iz procesa objedinjavanja i povratku u državu pre početka spajanja.

6. git reset

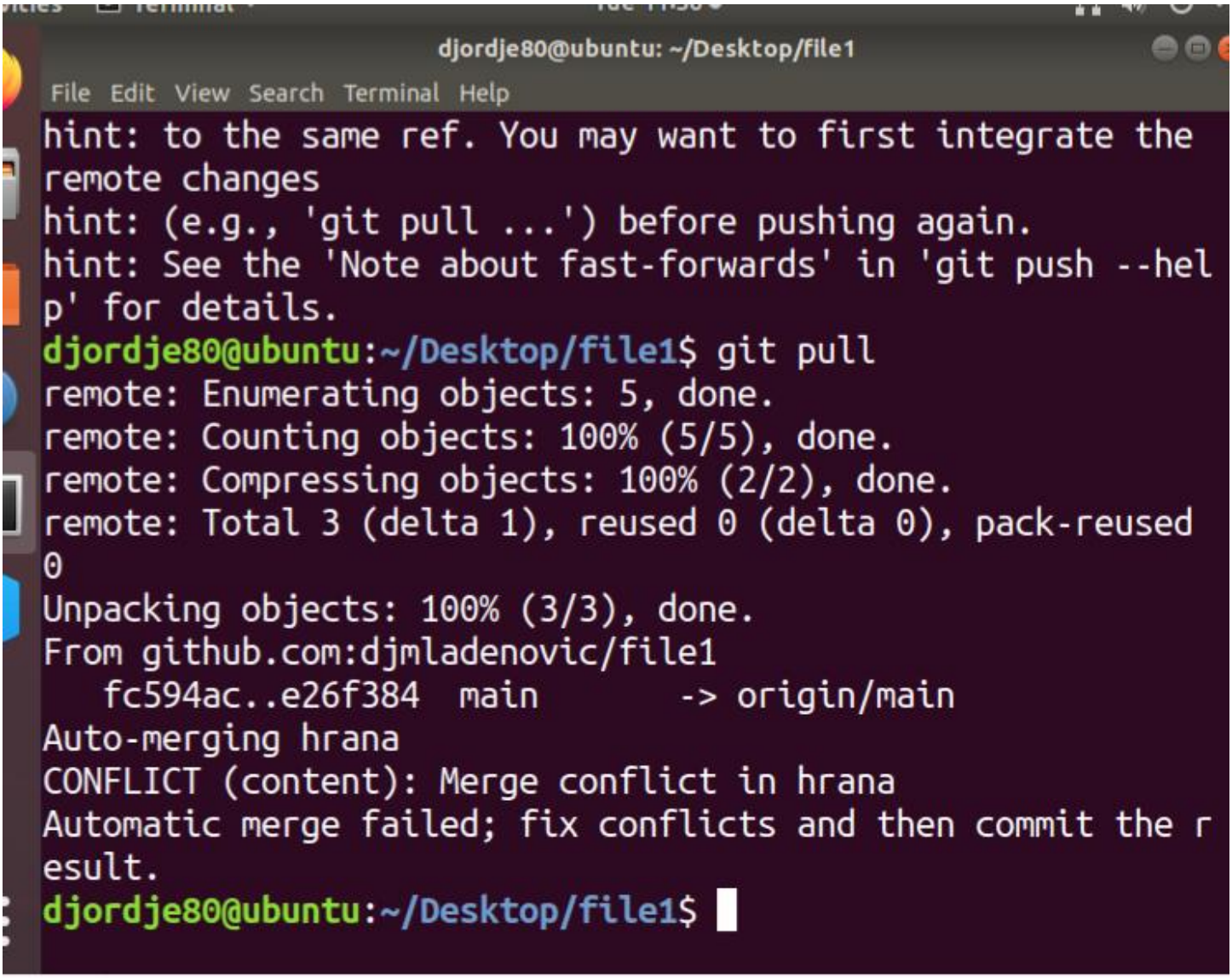
The git reset command koristi se u vreme neusaglašenosti objedinjavanja da bi se neusaglašene datoteke vratile u prvobitno stanje.

Kao primer u svom Git hab nalogu u file1 sam napravio ismene . Takođe sam na istom fajlu na istom mestu u cloniranom fajlu na terminal napravio isto ismene na istom mestu ali različite.



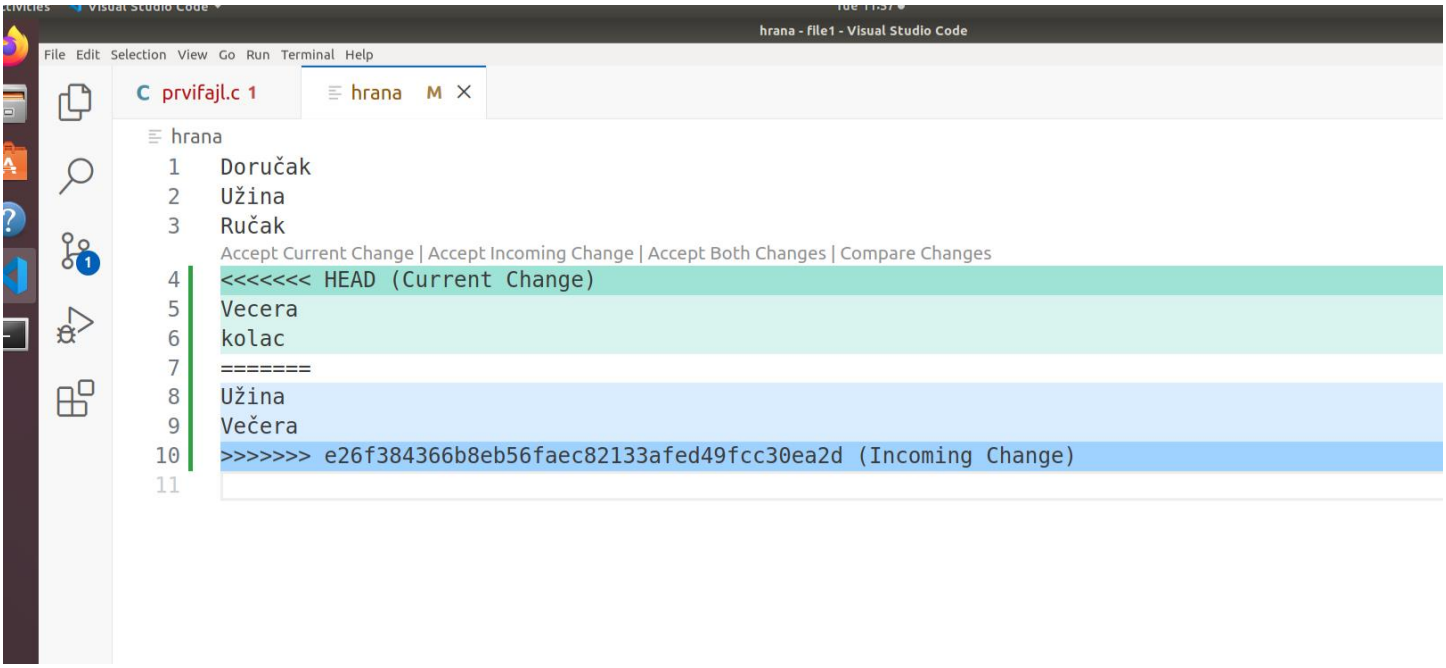


Tada se pojavio git conflit.



Konflikt je tu , sada se vratimo u VC I pogledamo te koflikte

Videcete koje opcije postoje. Imamo mogucnost da zadržimo prvi, drugi ili oba . Pa I da ih uporedimo.


hrana - file1 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
prvifajl.c 1
hrana M
hrana
1 Doručak
2 Užina
3 Ručak
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
4 <<<<<<< HEAD (Current Change)
5 Vecera
6 kolac
7 =======
8 Užina
9 Večera
10 >>>>>>> e26f384366b8eb56faec82133afed49fcc30ea2d (Incoming Change)
11