

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 .

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar, which lists the project structure. A red arrow points from the terminal window down to the 'node_modules' folder in the Explorer. The terminal window shows the command \$ npm install angular bootstrap angular-route being run, followed by several npm WARN messages about deprecated packages. The status bar at the bottom indicates there are 1 problem in the current workspace.

```
Korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ npm install angular bootstrap angular-route
npm WARN deprecated angular-route@1.8.3: For the actively supported Angular, see https://www.npmjs.com/package/@angular/core. AngularJS support has officially ended. For extended AngularJS support options, see https://goo.gle/angularjs-path-forward.
npm WARN deprecated angular@1.8.3: For the actively supported Angular, see https://www.npmjs.com/package/@angular/core. AngularJS support has officially ended. For extended AngularJS support options, see https://goo.gle/angularjs-path-forward.

added 4 packages in 4s

2 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New minor version of npm available! 9.5.0 -> 9.8.1
npm notice Changelog: <https://github.com/npm/cli/releases/tag/v9.8.1>
npm notice Run `npm install -g npm@9.8.1` to update!
npm notice

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

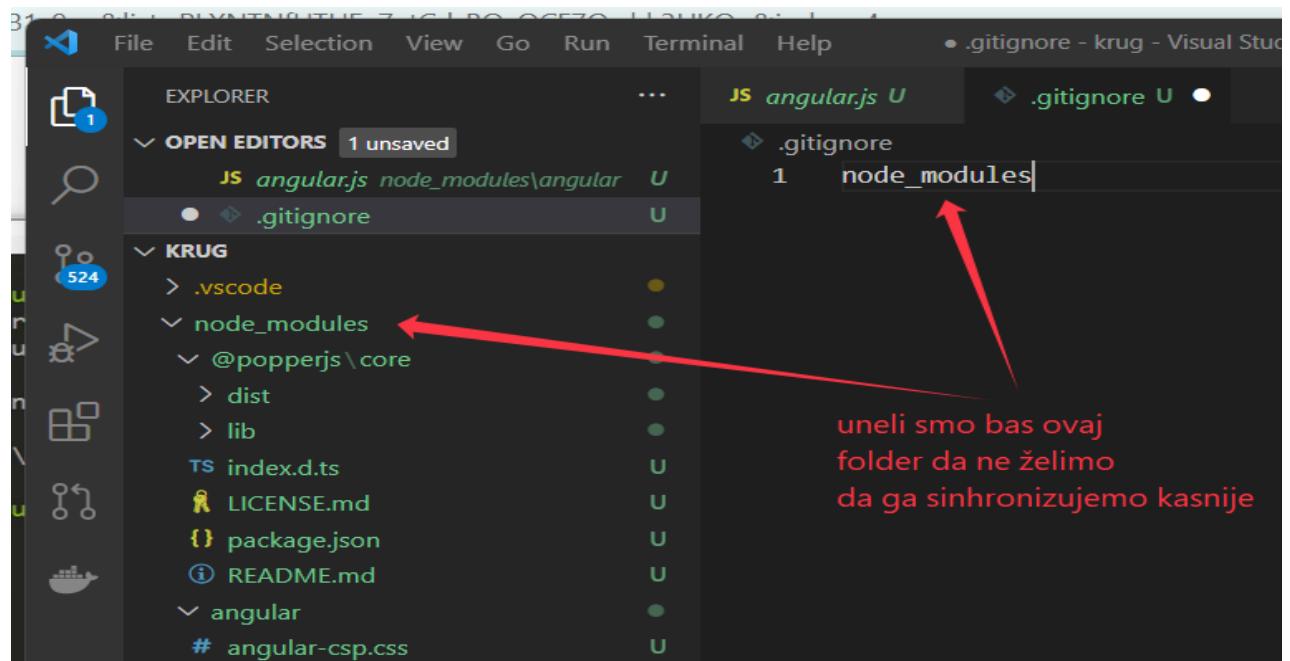
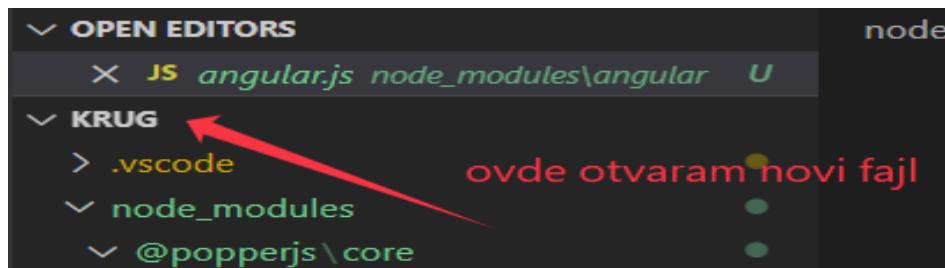
previše nepotrebnih fajlova

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 čemo staviti baš ovaj folder koji nam je nepotreban a učitali smo ga demonstrativno **node_modules**



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

```
korisnik@Djordje80 MINGW64 ~/Desktop/krug (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore ←
    .vscode/
    package-lock.json
    package.json

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

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

ono sto nas zanima

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 replaced 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.

	main	1 branch	0 tags	Go to
 djolemladenovic	samo proba			6d
 .vscode	samo proba			
 .gitignore	samo proba			
 README.md	Initial commit			
 main.cpp.c	Update main.cpp.c			
 package-lock.json	samo proba			vidimo naš fajl koji smo pravili
 package.json	samo proba			

krug / .gitignore				
Code	Blame	1 lines (1 loc) - 12 Bytes	kao što vidimo u tom fajlu je samo ovo	
1	node_modules			

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.

The screenshot shows the VS Code interface with the following details:

- EXPLORER** pane on the left:
 - OPEN EDITORS**: Shows a list of open files, with `.gitignore` currently selected.
 - SAMPLE-PROJECT**: Shows the project structure with the following files and folders:
 - `dist`
 - `logs`: Contains `build.log`, `example.log`, and `tests.log`.
 - `node_modules`
 - `src`: Contains `index.js`
 - `tests`
 - `.env`
 - `.eslintrc.js`
 - `.gitignore`
 - `LICENSE`
 - `package-lock.json`
 - OUTLINE**, **TIMELINE**, and **NPM SCRIPTS** are also listed.- EDITOR** pane on the right:
 - The title bar says `.gitignore M X`.
 - The code editor displays the `.gitignore` file with the following content:

```
1 .env
2 dist/
3 node_modules/
4
5 # Log files
6 *.log
7 !example.log
8
```

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

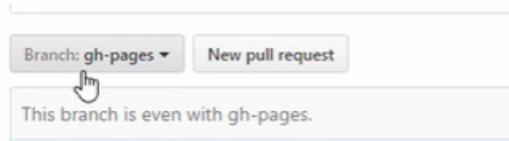
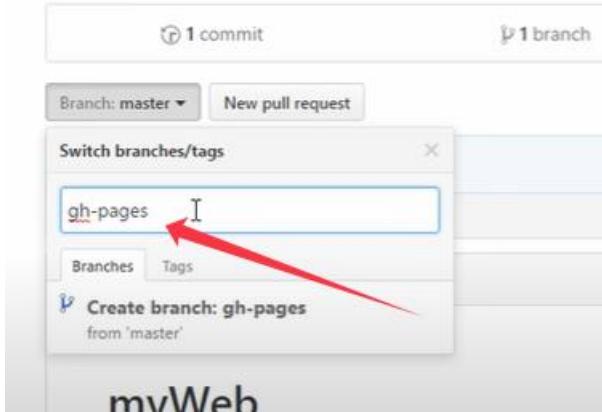
Ako kucamo *.log to se podrazumeva na sve fajlove koji se u logs

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 čemo ukucati gh-pages



I kreiraćemo taj novi Branch

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

A screenshot of the GitHub 'Settings' page for the repository 'myWeb'. The top navigation bar includes 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Settings' tab is active. On the left, a sidebar has sections for 'General', 'Access', 'Collaborators', 'Moderation options', 'Code and automation', and 'Branches' (which is selected). In the main area, the 'Branch protection rules' section is shown. It lists a single rule for the branch 'gh-pages'. A red arrow points to this rule entry. Below the rule, it says 'Currently applies to 1 branch' and has 'Edit' and 'Delete' buttons.

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.

The screenshot shows the GitHub Pages settings for a repository named 'myWeb'. On the left, there's a sidebar with various project management options like General, Access, Collaborators, and Pages. A red arrow labeled '1' points to the 'Pages' option. Another red arrow labeled '2' points to the main content area where it says 'Your site is live at <https://djolemladenovic.github.io/myWeb>'. A third red arrow labeled '3' points to the 'Pages' section in the sidebar. At the bottom right of the main content area, there's a note in Serbo-Croatian: 'ovde možemo videti naš sajt'.

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

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

The screenshot shows the GitHub code editor interface for the 'myWeb' repository. On the left, there's a sidebar with 'Code' selected, showing a dropdown for the branch ('main') and a search bar. Below that is a 'Go to file' input field with 'index.html' typed in. A red arrow points to this input field. On the right, the main panel shows the file content area with a placeholder: 'Enter file contents here'. A red arrow points to this placeholder with the text: 'kod čemo kucati u VSCodeu pa čemo ga ovde preneti'.

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

myWeb / index.html in main

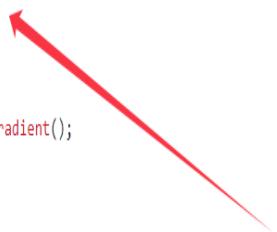
Cancel changes

Commit changes...

Edit Preview

Spaces 2 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 GitHub Pages

Access Collaborators Moderation options

Code and automation Branches Tags Rules Actions Webhooks Environments Codespaces 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

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

Sada možemo otici na naš sajt

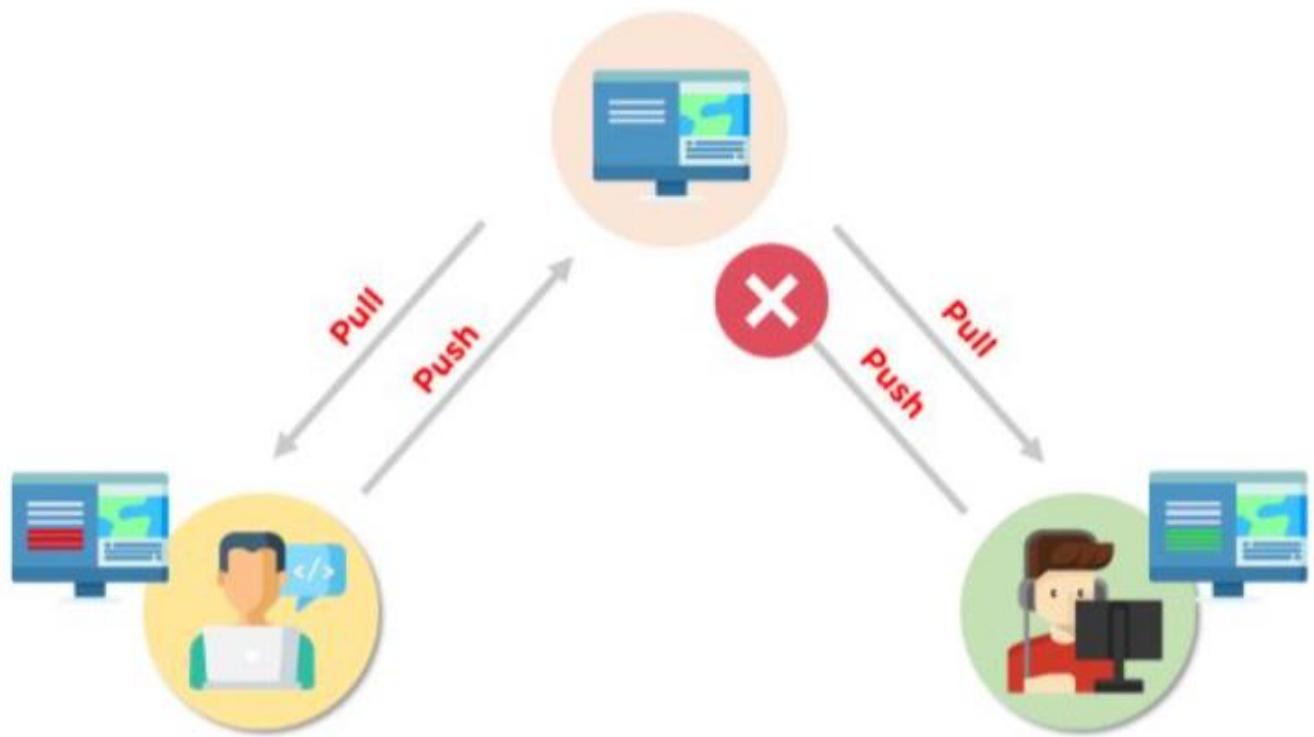
Learn how to add a Jekyll theme to your site.

Security

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 dosjelu. 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 sprecili 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 command helps in creating a list of commits that cause conflicts.

2. git diff

The git diff command helps in identifying differences between branches or files.

3. git checkout

The git checkout command is used for reverting changes made to files or switching branches.

4. git reset --mixed

The git reset --mixed command is used for reverting changes in the working directory and暂存区.

5. git merge --abort

The git merge --abort command helps in exiting the merge process without committing changes.

6. git reset

The git reset command is used to revert changes made to files in the working directory to their previous state.

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.

The screenshot shows a GitHub repository interface. The top navigation bar includes links for 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. A search bar is located at the top right. On the left, there are buttons for '+', 'Q', and 't'. The main content area displays a file named 'file1 / hrana'. A commit by 'djmladenovic' titled 'Update hrana' is shown. Below the commit, there are two tabs: 'Code' (which is selected) and 'Blame'. The 'Code' tab shows the following content:

```
1 Doručak
2 Užina
3 Ručak
4 Užina
5 Večera
```

```
Terminal ▾ Tue 12:05 ●
File Edit View Search Terminal Help djordje80@ubuntu: ~/Desktop/file1
```

```
changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)

    modified:   hrana

no changes added to commit (use "git add" and/or "git commit -a")
jordje80@ubuntu:~/Desktop/file1$ git add hrana
jordje80@ubuntu:~/Desktop/file1$ git commit -m"Promena u redosledu hrane"
[master 578b92e] Promena u redosledu hrane
 1 file changed, 2 insertions(+)
jordje80@ubuntu:~/Desktop/file1$ git push
to github.com:djmladenovic/file1.git
! [rejected]      main -> main (fetch first)
error: failed to push some refs to 'git@github.com:djmladenovic/file1.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
jordje80@ubuntu:~/Desktop/file1$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:djmladenovic/file1
  fc594ac..e26f384  main      -> origin/main
Auto-merging hrana
CONFLICT (content): Merge conflict in hrana
Automatic merge failed; fix conflicts and then commit the result.
jordje80@ubuntu:~/Desktop/file1$
```

Tada se pojavio git conflit.

```
File Edit View Search Terminal Help
hint: to the same ref. You may want to first integrate the
remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --hel
p' for details.
djordje80@ubuntu:~/Desktop/file1$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused
0
Unpacking objects: 100% (3/3), done.
From github.com:djmladenovic/file1
      fc594ac..e26f384  main          -> origin/main
Auto-merging hrana
CONFLICT (content): Merge conflict in hrana
Automatic merge failed; fix conflicts and then commit the r
esult.
djordje80@ubuntu:~/Desktop/file1$
```

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

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

Activities Visual Studio Code

File Edit Selection View Go Run Terminal Help

prvifajl.c 1 hrana M X

hrana

1 Doručak
2 Užina
3 Ručak

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes

<<<<< HEAD (Current Change)

5 Vecera
6 kolac
7 =====

8 Užina
9 Večera

>>>> e26f384366b8eb56faec82133afed49fcc30ea2d (Incoming Change)

11

The screenshot shows a diff view in Visual Studio Code for a file named 'hrana'. The code editor displays the following content:

```
1 Doručak
2 Užina
3 Ručak
<<<<< HEAD (Current Change)
5 Vecera
6 kolac
7 =====
8 Užina
9 Večera
>>>> e26f384366b8eb56faec82133afed49fcc30ea2d (Incoming Change)
```

The 'Current Change' section (lines 4-6) and the 'Incoming Change' section (lines 8-9) are color-coded. The commit hash 'e26f384366b8eb56faec82133afed49fcc30ea2d' is highlighted in blue.