

# Organizacija repozitorijuma na Github-u

Svaki projekat, a pogotovo projekti na kojima radi više ljudi, zahteva dobru organizaciju. Korišćenjem Git-a se rešava većina problema “gremlin” tipa “kako je ovo radilo juče, a danas ne”, međutim postoji čitav niz problema koji mogu nastati i upravo zbog loše organizacije ili nepravilnog korišćenja.

Na ovim vežbama ćemo proći kroz neke osnovne načine kako sebi možemo olakšati život i uštedeti vreme organizacijom projekta na Github-u. Sve ovo što se priča u vezi sa Github-om je ustaljena industrijska praksa, alati su slični.

Gotov primer ćemo iskoristiti kao osnovu, menjaćemo neke funkcionalnosti, a usput ćemo prolaziti kroz korake za organizaciju.

**Napomena (pročitati ponovo pri izradi projekta):** očekuje se da projekti imaju dobro napisan Readme.md fajl i .gitignore obavezno. Što se tiče Issues, pull requests i actions, obavezno je pravljenje barem jednog Issue-a i jednog Pull request-a, ali se ne zahteva aktivno korišćenje, samo je preporučljivo.

## 1. *Readme.md* fajl

Šta treba da sadrži (kome pišem)?

Pre svega, osnovni fajl koji bi svaki projekat trebao da sadrži je Readme fajl. U ovom fajlu bi trebalo da bude opisano ukratko **čemu** projekat **služi**, šta je **urađeno**, osnovnu **dokumentaciju** i **uputstva** kako pokrenuti i testirati kod.

Ovo je dobro mesto i da se navedu **zavisnosti** koda.

Ovde je bitno, kako drugima koji bi eventualno koristili ili nadogradili kod, tako i vama jer se lako zaboravljaju stvari kad se neko vreme ne radi na njima.

[Primer Readme fajla](#)

## 2. *.gitignore* fajl

Šta Git vidi, a šta u potpunosti ignoriše?

U ovom fajlu se navode sve datoteke, tipovi datoteka, kompletni direktorijumi, praktično sve što želimo da Git u potpunosti ignoriše.

Treba biti pažljiv pri pisanju .gitignore fajla jer se mogu desiti neželjene stvari, na primer:

1. Ne ignorišu se datoteke koje nisu kod (na primer iskompajlirane datoteke, privremene *cash* datoteke i slično). U ovom slučaju imamo problem što razmenjujemo datoteke koje ne moramo. Na primer kod iz primera za autoelektroniku zauzima 1.3 MB, a kada se iskompajlira zauzima 200 MB.  
Dakle potrebno je ignorisati sve što nije obavezan deo koda ili bild sistema.
2. Ignorise se fajl bez kojeg kod ne može u potpunosti da radi. Tipični primeri su ignorisanje nekog .lib/.dll fajla ili fajla koji govori na koji način se projekat kompajlira (projektni fajl, fajl bild sistema)...

Najbolji način za generisanje .gitignore fajla je pomoću gotovih Template-a. Na primer za Visual Studio postoji template, kao i za većinu popularnih IDE-a. Kao i za neke bild sisteme ili programske jezike. Template je često ponuđen od strane online Git alata (ima na Github), a može se generisati i preko sledećeg linka.

Za potrebe našeg projekta OK je da koristimo [Template za VisualStudio](#).

[Generator .gitignore fajla](#)

### 3. Issues

Šta treba sledeće raditi?

Issues se koriste da se prate problemi u kodu, ali takođe i zahtevi za funkcionalnostima (feature requests), TO-DO objekti i slično.

Na Github-u, pri kreiranju issue-a moguće je dodati i Labelu i Milestone. Labela može da označava na primer koji je tip issue-a, a Milestone govori do kada treba da se reši neki Issue.

Šta između ostalog treba da sadrži tekst Issue-a:

- Šta je issue (Bug report, Feature request, Enhancement i slično)
- Žašto se javlja, koji scenario ga izaziva (koraci da se izazove)
- Predlog kako rešiti

[Primer jednog Issue-a](#)

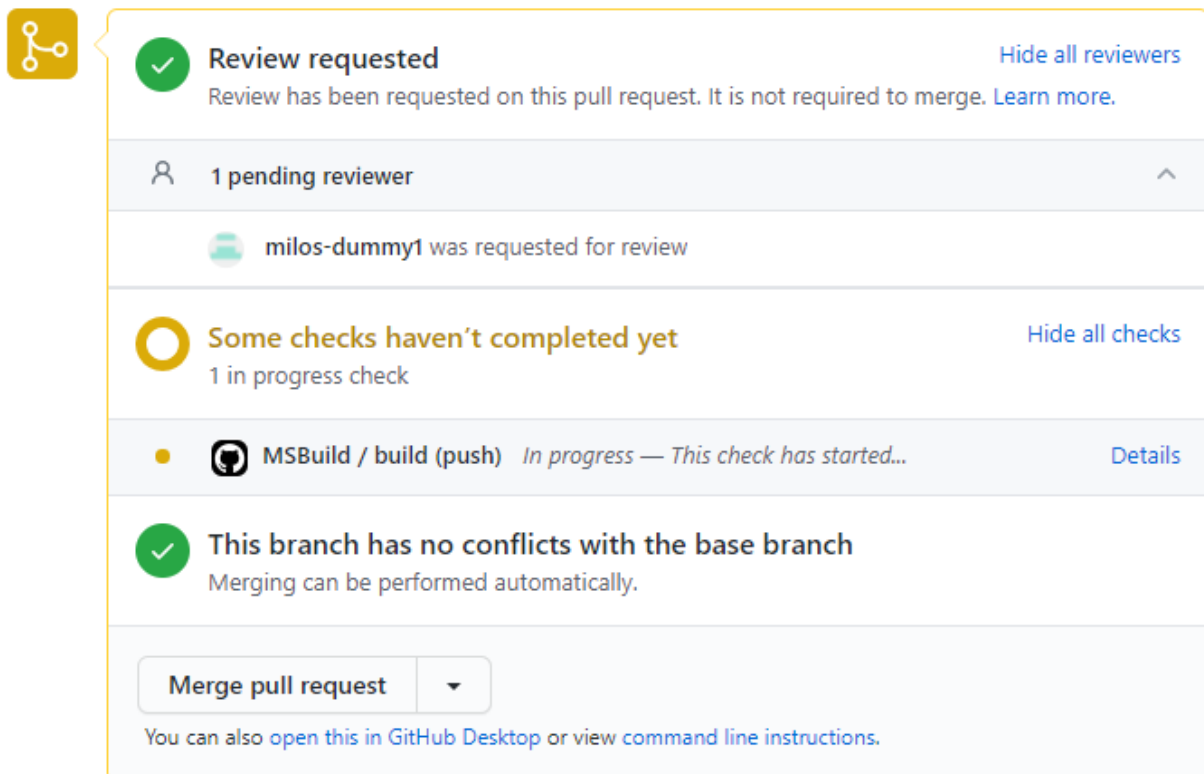
## 4. Pull Requests

Da li je urađeno kako treba, nek pogleda još neko?


Osnovna uloga Pull Request-ova je da se svaki kod koji ide na glavnu granu revidira od strane više ljudi. Takođe je moguće napraviti automatsku proveru koda (kompajl + testovi) svaki put kada se nešto pošalje na granu.



Pri kreiranju Issue-a uglavnom se zadaje lista ljudi koji pregledaju izmene (to ne mogu biti ljudi koji su i napravili te izmene).


Svaki put kada se izmene push-uju na granu koja ima neki Pull Request, prvo se proverava da li je moguće uopšte da se merge-uje (da li ima konflikata), potom se čeka odobrenje svih koji treba da potvrde izmene, i takođe ako postoji neki Actions, čeka se potvrda da je to OK.






The screenshot shows a GitHub Pull Request status bar with a yellow border. On the left is a yellow icon of a branching diagram. The bar contains several sections: 1. A green checkmark icon followed by the text 'Review requested' and a link 'Hide all reviewers'. Below this is a sub-header '1 pending reviewer' with an upward arrow, and a list item 'milos-dummy1 was requested for review'. 2. An orange circle icon followed by the text 'Some checks haven't completed yet' and a link 'Hide all checks'. Below this is a list item 'MSBuild / build (push) In progress — This check has started...' with a link 'Details'. 3. A green checkmark icon followed by the text 'This branch has no conflicts with the base branch' and the text 'Merging can be performed automatically.'. At the bottom is a button 'Merge pull request' with a dropdown arrow, and a link 'You can also open this in GitHub Desktop or view command line instructions.'


 **Review requested** [Hide all reviewers](#)  
Review has been requested on this pull request. It is not required to merge. [Learn more.](#)


 **1 pending reviewer** 

 milos-dummy1 was requested for review

 **Some checks haven't completed yet** [Hide all checks](#)  
1 in progress check

  **MSBuild / build (push)** *In progress — This check has started...* [Details](#)

 **This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request** 

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Ako su svi uslovi ispunjeni, moguće je uraditi Merge i zatvoriti taj Pull request, a ako su uvezani, automatski zatvoriti i Issue.

Na narednoj stranici se vidi primer jednog kompletnog scenarija.



**miloshunter** commented 28 minutes ago

Owner



Ovde pisete razlog zašto bi trebalo merge-ovati ovu granu bla bla.



**miloshunter** added 2 commits 29 minutes ago



**Fixed issue #1**

7da2190



**Merge branch 'main' into led\_bar\_tsk-fix**

✓ dc75ff4



**miloshunter** linked an issue that may be closed by this pull request 16 minutes ago

**Bug report: led\_bar\_tsk - ne radi kako treba #1**

🔒 Closed



**miloshunter** requested a review from **milos-dummy1** 4 minutes ago



**minor fix**

✓ fa3dfc0



**milos-dummy1** approved these changes 3 minutes ago

[View changes](#)

**milos-dummy1** left a comment

Collaborator



ovo se cini kao dobro resenje



**milos-dummy1** merged commit 4e3c5b2 into **main**

[View details](#)

[Revert](#)

2 minutes ago

1 check passed

## 5. Github Actions (workflow)

Push-ovao sam neki kod, kako znam da radi?

Postoji mogućnost i automatske provere koda da li radi, svaki put kad se nešto desi (najčešće novi push na grani).


Workflows made for your C repository Suggested

**MSBuild based projects**  
By GitHub Actions

Build a MSBuild based project.

[Set up this workflow](#)

```
nuget restore ${env.SOLUTION_FILE_PATH}
msbuild /m /p:Configuration=${env.BUILD_CONFIGURATION}
${env.SOLUTION_FILE_PATH}
```


 actions/starter-workflows C ●

**CMake based projects**  
By GitHub Actions

Build and test a CMake based project.

[Set up this workflow](#)

```
cmake -B ${github.workspace}/build -DCMAKE_BUILD_TYPE=${env.BUILD_TYPE}
cmake --build ${github.workspace}/build --config ${env.BUILD_TYPE}
ctest -C ${env.BUILD_TYPE}
```


 actions/starter-workflows C ●

**C/C++ with Make**  
By GitHub Actions

Build and test a C/C++ project using Make.

[Set up this workflow](#)

```
./configure
make
make check
```

 actions/starter-workflows C ●

Ovo su neki gotovi template-i koji omogućavaju kompajliranje koda svaki put kad se push-uje. Najčešće je praksa da se prave i automatski testovi koji proveravaju osnovne funkcionalnosti svakog većeg paketa (unit testovi, koji su brzi ali dobro pokrivaju kod).

Na ovaj način znamo da li će naš Merge na glavnu granu da unese neke greške i da ih predupredimo.