

# Objavljivanje i kolaboracija

Proces razvoja web sajtova veoma često podrazumeva učestvovanje većeg broja programera, što je posebno istinito ukoliko je reč o kompleksnijim projektima. Stoga je međusobna saradnja, odnosno kolaboracija, veoma važan aspekt web razvoja. U lekciji koja je pred vama biće objašnjeni osnovni principi timskog razvoja web sajtova.

U posao razvoja web sajtova ubraja se i proces njegovog objavljivanja. Jednostavno, nakon kreiranja funkcionalnog proizvoda potrebno ga je objaviti, što podrazumeva postavljanje fajlova web sajta na server sa koga će web sajt biti javno dostupan. Proces objavljivanja web sajtova biće još jedna tema koja će biti obrađena u ovoj lekciji.

## Pojam verzionisanja koda

Srce svakog softverskog proizvoda jeste njegov izvorni kod. Kada su u pitanju frontend programiranje i razvoj web sajtova, takav izvorni kod jeste zapravo kod HTML, CSS i JavaScript jezika. Tako se može reći da bez izvornog koda ovih jezika ne može postojati ni web sajt.

Količina koda od kojeg su sačinjeni web sajtovi zavisi od brojnih parametara, pri čemu se kao neka ustaljena jedinica za izražavanje količine koda koristi *linija*. Tako se, na primer, može reći da je neki web sajt sačinjen iz 1.354 linije koda. Jednostavniji web sajtovi, sasvim razumljivo, poseduju manji broj linija koda, dok kompleksniji sajtovi mogu imati i desetine, pa i stotine hiljada linija koda.

Kako se tokom razvoja povećava broj linija koda od kojih je sačinjen neki web sajt, razumljivo je da je sve teže i teže ispratiti promene koje se sprovode na kodu. Ukoliko se tome doda i timski rad, koji podrazumeva ažuriranje jednog istog koda od strane većeg broja programera, situacija postaje još komplikovanija. Kako bi se uveo red u programerski posao i olakšalo održavanje, tokom vremena su kreirani brojni sistemi za verzionisanje koda (*engl. version control systems*). Pored ovog osnovnog naziva, ovakvi sistemi se veoma često nazivaju i *revision control* i *source control* sistemi.

Verzionisanje koda je pristup koji omogućava praćenje promena izvornog koda. Jednostavnije rečeno, verzionisanje omogućava da se svaka izmena na kodu zabeleži i da se omogući povratak na takvu verziju u bilo kom trenutku. Kako biste ovo bolje razumeli, razmotrićemo jedan primer. Naručilac web sajta zatraži izmenu određene funkcionalnosti, koja zahteva korigovanje izvornog koda. Postojeći kod se zameni novim, a onda klijent odluči da želi da se na sajtu vrati stara funkcionalnost. Ukoliko se ne koristi neki sistem za verzionisanje, programeru ne preostaje ništa drugo, nego da ponovo napiše kod stare funkcionalnosti. Sa druge strane, sa sistemom za verzionisanje proces povratka na staru verziju podrazumeva samo nekoliko klikova.

Korišćenjem sistema za verzionisanje, programeri u svakom trenutku mogu doći do važnih informacija o istoriji projekta:

- sve promene koje su napravljene;
- ko ih je napravio;
- kada su one napravljene;
- zbog čega su napravljene.

Verzisionisanje je danas realizovano korišćenjem brojnih sistema, odnosno kompjuterskih programa.

## Šta je Git?

Danas najpoznatiji sistem za verzionisanje je svakako Git. Reč je o sistemu koji je kreirao tvorac Linux kernela – Linus Torvalds, upravo radeći na svom najpopularnijem proizvodu. Zbog svoje jednostavnosti, brzine i činjenice da je reč o softveru otvorenog koda, Git je stekao veliku popularnost tokom vremena. Posebno se intenzivno koristi prilikom web razvoja, za projekte različite veličine, od jednostavnih web sajtova na kojima radi jedan programer, pa sve od kompleksnih web sistema, na čijem radu učestvuju veliki razvojni timovi. U nastavku ovoga kursa mi ćemo se upoznati sa osnovama korišćenja Gita.

### Pitanje

Šta je Git?

- **Sistem za verzionisanje koda**
- Sistem za otklanjanje grešaka
- Sistem za testiranje
- Razvojno okruženje

### Objašnjenje:

*Git je danas najpoznatiji sistem za verzionisanje koda, koji je kreirao tvorac Linux kernela – Linus Torvalds.*

## Instalacija i korišćenje Gita

Git je potpuno besplatan za korišćenje. U zavisnosti od operativnog sistema razlikuje se i procedura njegove instalacije.

Na Linuxu je dovoljno pokrenuti sledeću komandu:

```
$ sudo apt install git-all
```

Na macOS-u verzije 10.9 i više Git se može instalirati pokretanjem sledeće komande u terminalu:

```
$ git --version
```

Za starije verzije macOS-a, Git se može preuzeti sa sledeće adrese:

<https://git-scm.com/download/mac>

Na kraju, instalacija Gita za Windows se može preuzeti sa sledeće adrese:

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

## Napomena

Ukoliko pokušavate da instalirate Git na Windows ili macOS operativnim sistemima korišćenjem instalacionih fajlova koji se nalaze na upravo navedenim web adresama, takve instalacione fajlove je nakon preuzimanja potrebno pokrenuti i treba pratiti tok instalacije. Potrebno je koristiti podrazumevana instalaciona podešavanja, odnosno nije potrebno vršiti bilo kakve promene podrazumevanih postavki prilikom instalacije. Tek nakon završene instalacije bićete u mogućnosti da isprobate Git komande koje su navedene u nastavku lekcije.

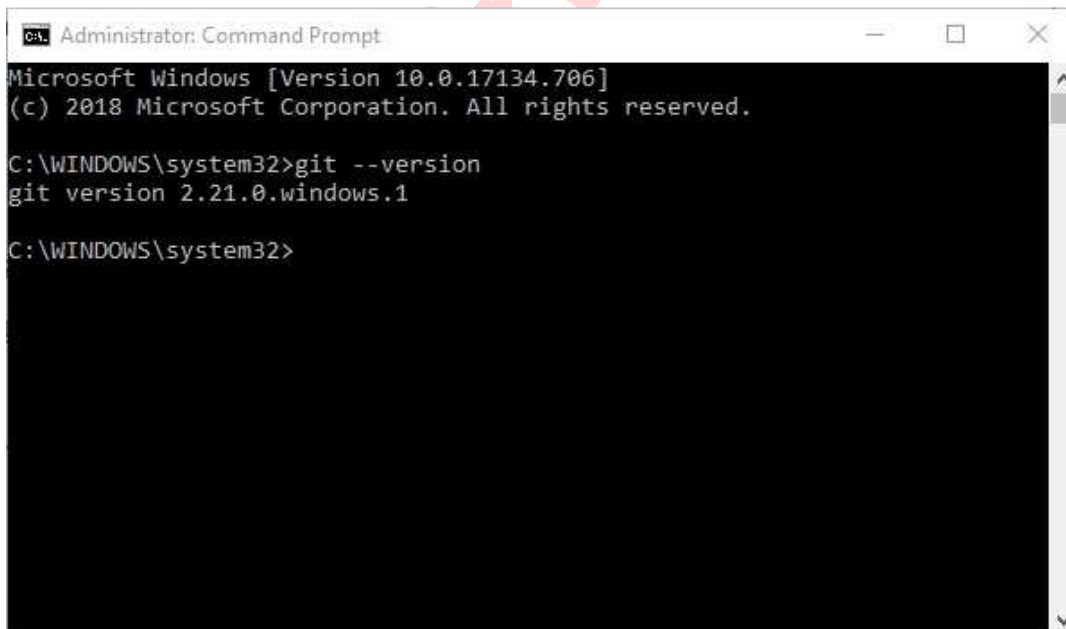
## Osnovne Git komande

Git je zamišljen i kreiran kao alat bez korisničkog okruženja. Zbog toga njegovo korišćenje podrazumeva upotrebu konzole ili terminala, u zavisnosti od operativnog sistema koji koristite. Naravno, zbog odsustva grafičkog korisničkog okruženja, tokom vremena je kreiran veliki broj takozvanih Git klijenata, odnosno programa koji omogućavaju jednostavnije korišćenje Gita. Mi smo se sa razlogom odlučili da Git isprobamo u njegovom izvornom obliku kako biste na pravi način razumeli njegov način funkcionisanja.

Funkcionisanje Gita zasniva se na upućivanju različitih tekstualnih komandi. Na primer, za početak uputićemo jednu komandu kojom ćemo proveriti postojanje Gita:

```
git --version
```

Ovo je komanda kojom se proverava postojanje Gita, kao i njegova verzija (slika 11.1).



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>git --version
git version 2.21.0.windows.1

C:\WINDOWS\system32>
```

*Slika 11.1. Provera postojanja Gita*

Na slici 11.1. ilustrovano je upućivanje prve Git komande na Windows operativnom sistemu, korišćenjem Command Prompt programa. Obratite pažnju na to da je neophodno da Git bude instaliran, a njegova lokacija smeštena unutar sistemske Path promenljive. Instalacijom Gita za Windows, sa nešto ranije navedene URL adrese, sve ovo se obavlja automatski. U slučaju prisustva Gita na sistemu, dobija se poruka slične sadržine onoj koja se može videti na slici – *git version 2.21.0.windows.1*. Naravno, poruka se kod vas može razlikovati u zavisnosti od instalirane verzije.

Funkcionisanje Gita zasniva se na pojmu repozitorijuma.

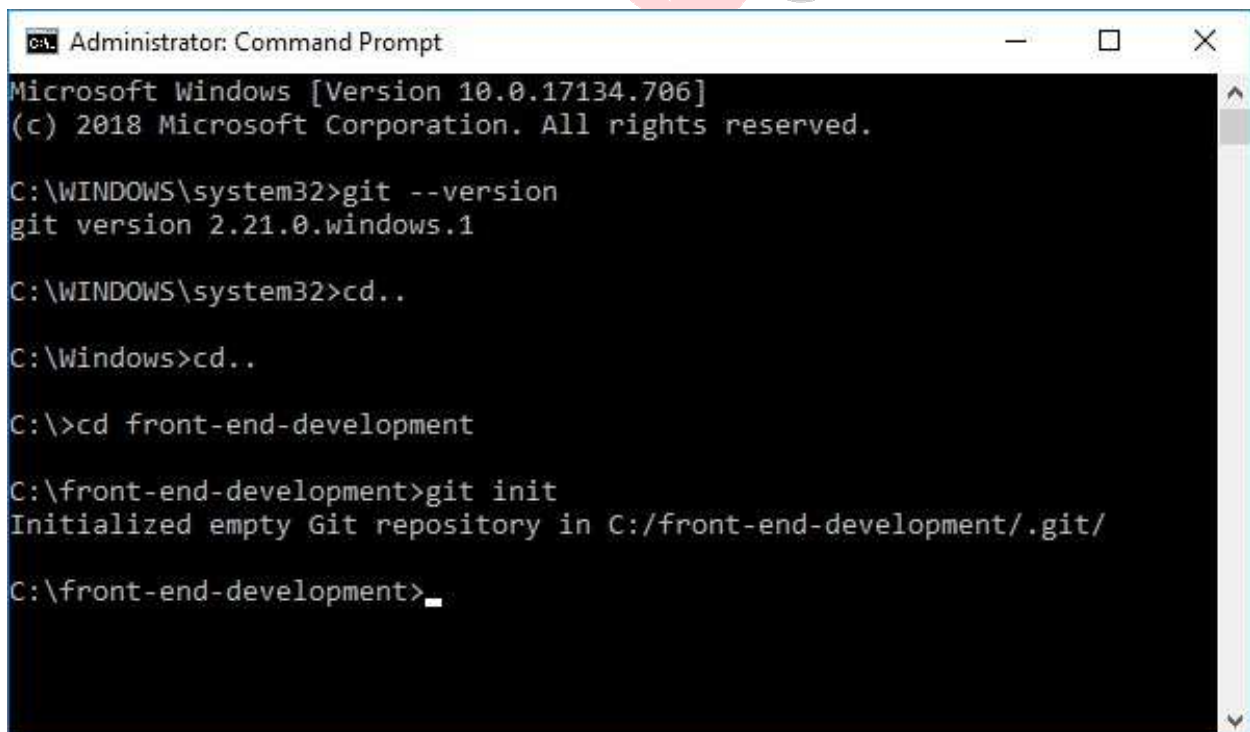
### Git repozitorijum

Repozitorijum je centralna figura Git sistema. Često se naziva i Git projekat, a predstavlja kompletnu kolekciju fajlova i foldera od kojih je sačinjen jedan softverski projekat. U našem slučaju, reč je o svim fajlovima i folderima jednog web sajta.

Kreiranje Git repozitorijuma obavlja se upućivanjem sledeće komande:

`git init`

Upućivanje ove komande je potrebno obaviti iz foldera unutar kog se želi kreirati repozitorijum. Mi ćemo repozitorijum kreirati unutar foldera koji sadrži naš prvi HTML dokument, kreiran u jednoj od prethodnih lekcija (slika 11.2).



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>git --version
git version 2.21.0.windows.1

C:\WINDOWS\system32>cd..

C:\Windows>cd..

C:\>cd front-end-development

C:\front-end-development>git init
Initialized empty Git repository in C:/front-end-development/.git/

C:\front-end-development>
```

Slika 11.2. Kreiranje Git repozitorijuma

Nakon kreiranja Git repozitorijuma, potrebno mu je dodeliti i fajlove. Unutar repozitorijuma mi ćemo smestiti fajl koji predstavlja naš prvi HTML dokument. Smeštanje fajlova i foldera u repozitorijum obavlja se sprovođenjem dva koraka koji se nazivaju *staging* i *committing*. Staging se može doživeti kao pripremni korak, a committing kao akcija koja pokreće upisivanje promena u repozitorijum. Pri tome se u repozitorijum upisuju samo promene na onim fajlovima koji su u *staging* stanju, odnosno nad kojima je prethodno obavljen *staging*. Svrhu ovakvog mehanizma, koji podrazumeva dva koraka, možda ne možete da razumete u ovom trenutku. Ipak, pozadina je i više nego jednostavna. Naime, projekti se najčešće sastoje iz većeg broja fajlova, što je slučaj i prilikom razvoja web sajtova. Nekada nećete želiti da promene u svim fajlovima upišete u repozitorijum. U takvim situacijama ovaj jedan međukorak (*staging*) je i više nego koristan, jer omogućava da se precizno odrede promene koji će biti unete u repozitorijum.

Staging, odnosno priprema za predaju, obavlja se na sledeći način:

```
git add index.html
```

Na ovaj način je fajl index.html postavljen u staging stanje, pri čemu je sve spremno za njegovo upisivanje u repozitorijum.

Commit, odnosno predaja ili završetak, obavlja se korišćenjem sledeće komande:

```
git commit -m "Initial Commit"
```

Pored teksta komande, prikazana naredba sadrži i opis promena koje su napravljene u kodu. Tako nešto olakšava kasnije snalaženje u različitim verzijama jednog istog koda.

### **Kreiranje korisničkog identiteta**

Pre upisivanja promena u repozitorijum neophodno je da Git poseduje podatke o korisniku koji obavlja upisivanje. Ukoliko Git ne poseduje takve podatke, kreiranje korisničkog identiteta je moguće obaviti upućivanjem sledeće dve komande:

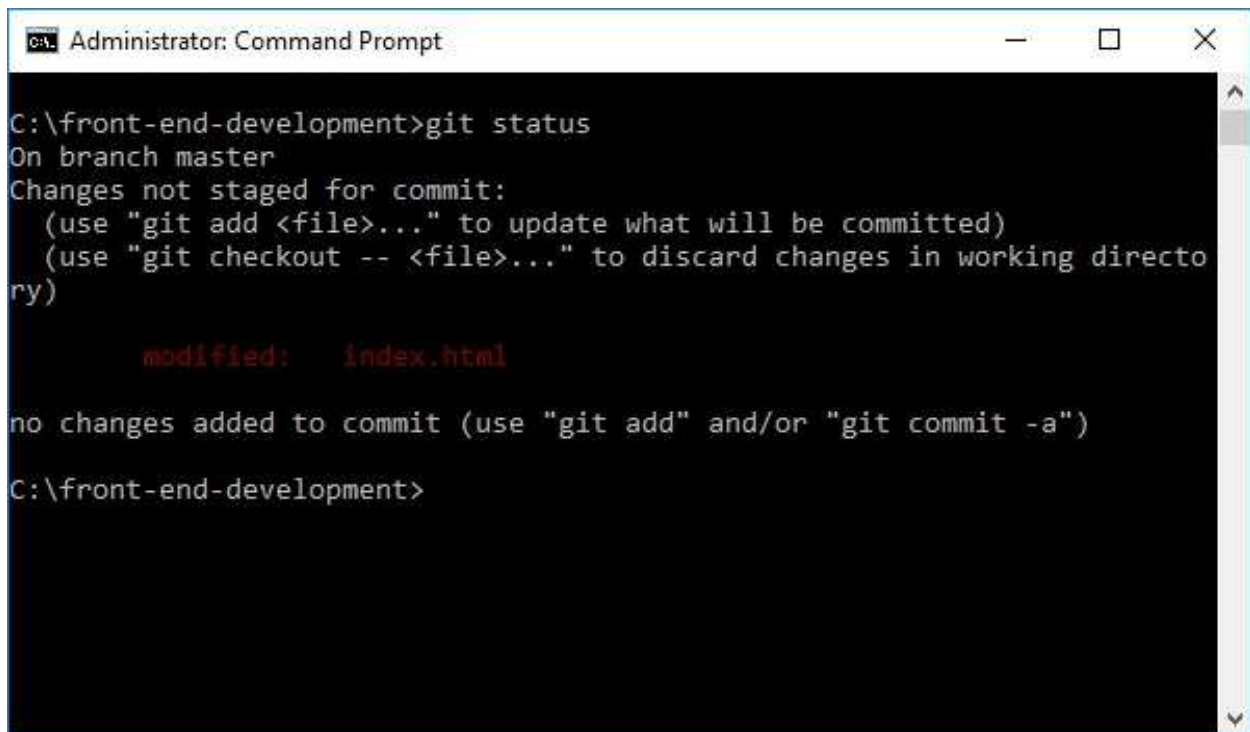
```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```

Veoma je bitno znati da nijedan commit ne može biti obavljen ukoliko prethodno nije postavljen korisnički identitet. Stoga, ukoliko je od vas zatraženo da definišete korisnički identitet nakon pokušaja commita, potrebno je znati da takav commit nije uspeo. Zato toga je u takvoj situaciji, nakon definisanja korisničkog identiteta na način koji je prikazan u prethodnim redovima, potrebno ponoviti inicijalni commit.

Obavljanjem dosadašnjih koraka uspostavljen je Git repozitorijum sa jednim fajlom. Kako biste uvideli pravu moć Git sistema, dovoljno je da nakon opisanih koraka napravite neku izmenu na našem prvom HTML dokumentu. Na primer, možete dodati još jedan paragraf i, nakon tako napravljene izmene, sačuvajte fajl. Git poseduje komandu za proveru statusa fajlova koji su upisani u repozitorijum:

```
git status
```

Pokretanje ovakve komande, nakon sprovedene izmene na index.html fajlu, imaće rezultat kao na slici 11.3.



```
C:\front-end-development>git status
On branch master
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
C:\front-end-development>
```

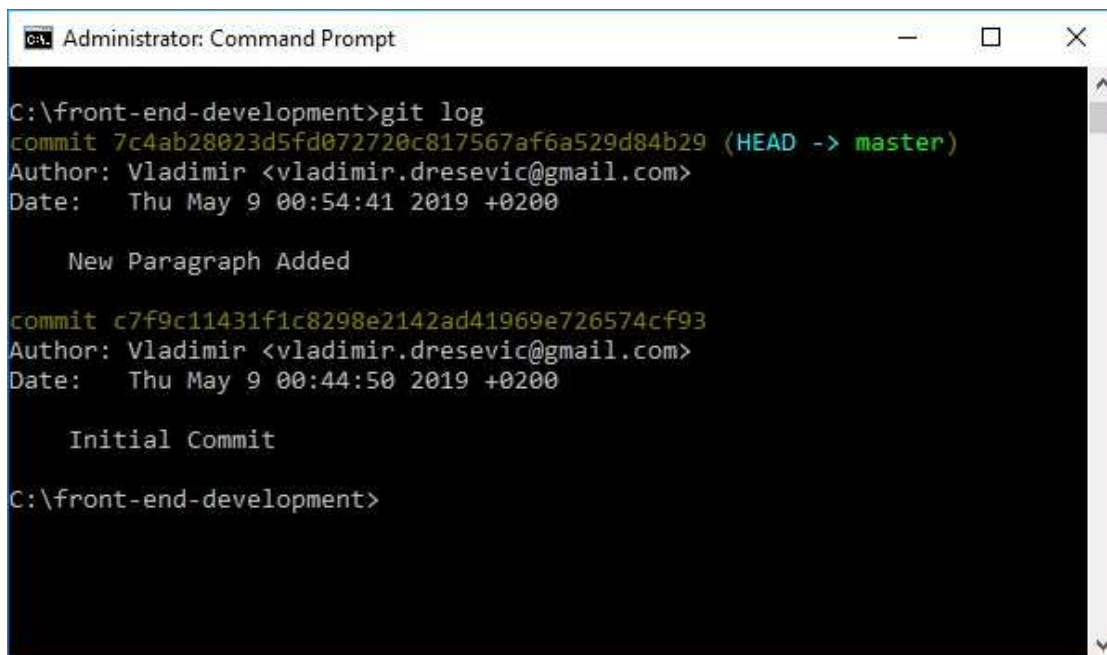
Slika 11.3. Provera statusa fajlova u repozitorijumu

Na slici 11.3. možete videti da je Git detektovao da na fajlu index.html postoje izmene, odnosno da je fajl modifikovan nakon poslednjeg sprovedenog commita. Ukoliko bismo takvu promenu sada želeli da upišemo u repozitorijum, bilo bi neophodno ponovo obaviti *staging* i *committing*.

Na kraju, Git omogućava i pregled svih commit akcija:

git log

Upućivanjem ove komande dobija se pregled svih commitova u tekućem repozitorijumu, sa propratnim informacijama (slika 11.4).



```
Administrator: Command Prompt

C:\front-end-development>git log
commit 7c4ab28023d5fd072720c817567af6a529d84b29 (HEAD -> master)
Author: Vladimir <vladimir.dresevic@gmail.com>
Date: Thu May 9 00:54:41 2019 +0200

    New Paragraph Added

commit c7f9c11431f1c8298e2142ad41969e726574cf93
Author: Vladimir <vladimir.dresevic@gmail.com>
Date: Thu May 9 00:44:50 2019 +0200

    Initial Commit

C:\front-end-development>
```

Slika 11.4. Pregled svih commitova

### Pojam Git grananja

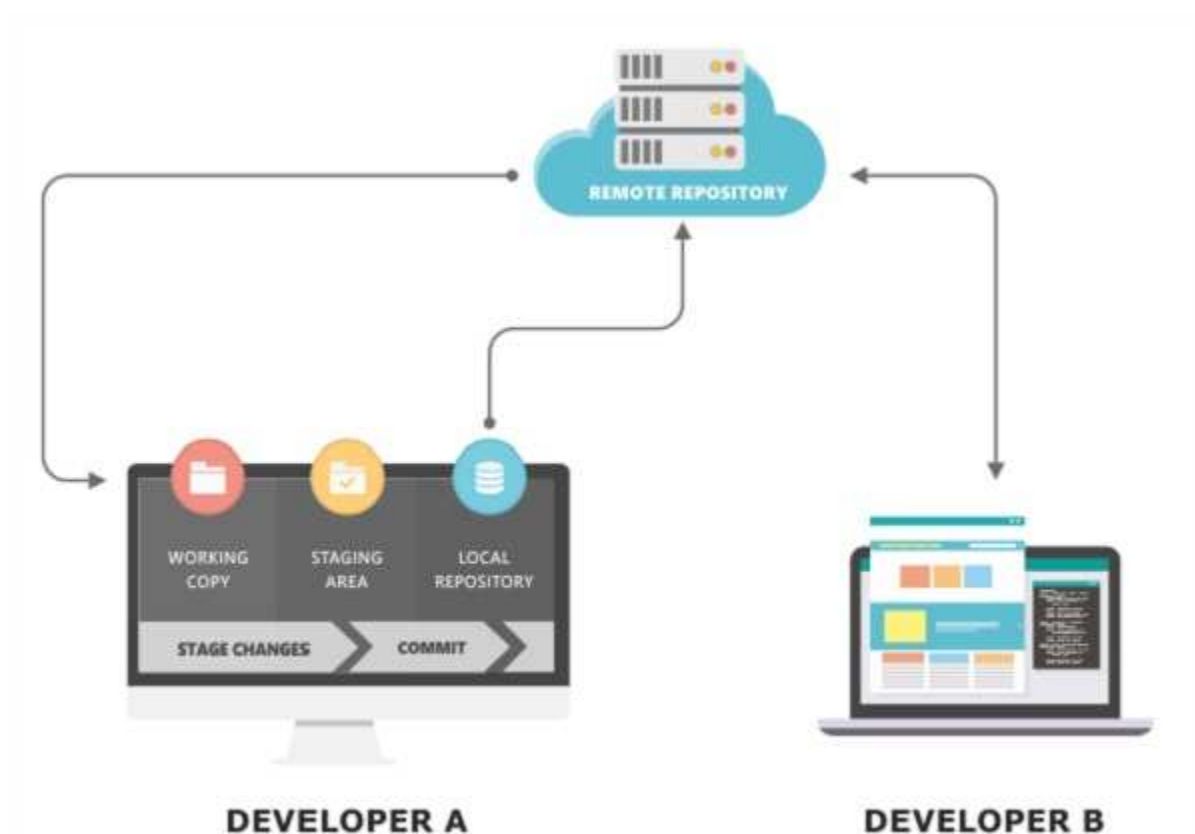
Pored repozitorijuma, grananje je jedan od osnovnih Git pojmova. Naime, jedan Git projekat sačinjen je iz jedne ili više grana (*engl. branches*). Git automatski kreira početnu granu koja se naziva master grana (*engl. master branch*). Programer ima mogućnost da kreira proizvoljan broj dodatnih grana. Osnovna poenta grananja jeste mogućnost paralelnog razvoja nekoliko različitih verzija jednog softverskog proizvoda. Na primer, jedna grana može služiti kao primarna, a druga za testiranje i isprobavanje. Git omogućava kreiranje novih grana, ali i spajanje dve ili više grana u jednu.

### Šta je GitHub?

Velika zabuna kod početnika može nastati kada se u priču o Gitu umeša pojam GitHub. Reč je zapravo o jednom od najpopularnijih online sistema za hostovanje Git projekata. S obzirom na to da je nešto ranije rečeno da je Git konzolni alat bez grafičkog korisničkog okruženja, GitHub se može doživeti kao grafički omotač za Git, dostupan na webu. Tako GitHub omogućava kreiranje udaljenih repozitorijuma, koji su dostupni preko weba, što je sjajna mogućnost za rad u timu.

Jednostavno, u prethodnim redovima ove lekcije ilustrovano je kreiranje jednog lokalnog Git repozitorijuma. Ipak, prilikom timskog rada uglavnom postoji i jedan udaljeni, centralni repozitorijum, dok svaki od programera na svom kompjuteru poseduje i lokalni repozitorijum, baš kao što to ilustruje slika 11.5.





*Slika 11.5. Princip funkcionisanja lokalnih i udaljenih repozitorijuma*

Na ovaj način programeri rade na lokalnim repozitorijumima i, nakon izvršenih izmena na kodu, takve izmene prosleđuju u centralni repozitorijum, koji je deljen sa svim ostalim kolegama iz tima. Tako svi članovi tima svoj rad započinju usklađivanjem svog lokalnog repozitorijuma sa centralnim kako bi osigurali rad na najažurnijoj verziji koda, koja sadrži sve izmene koje su načinile ostale kolege u timu.

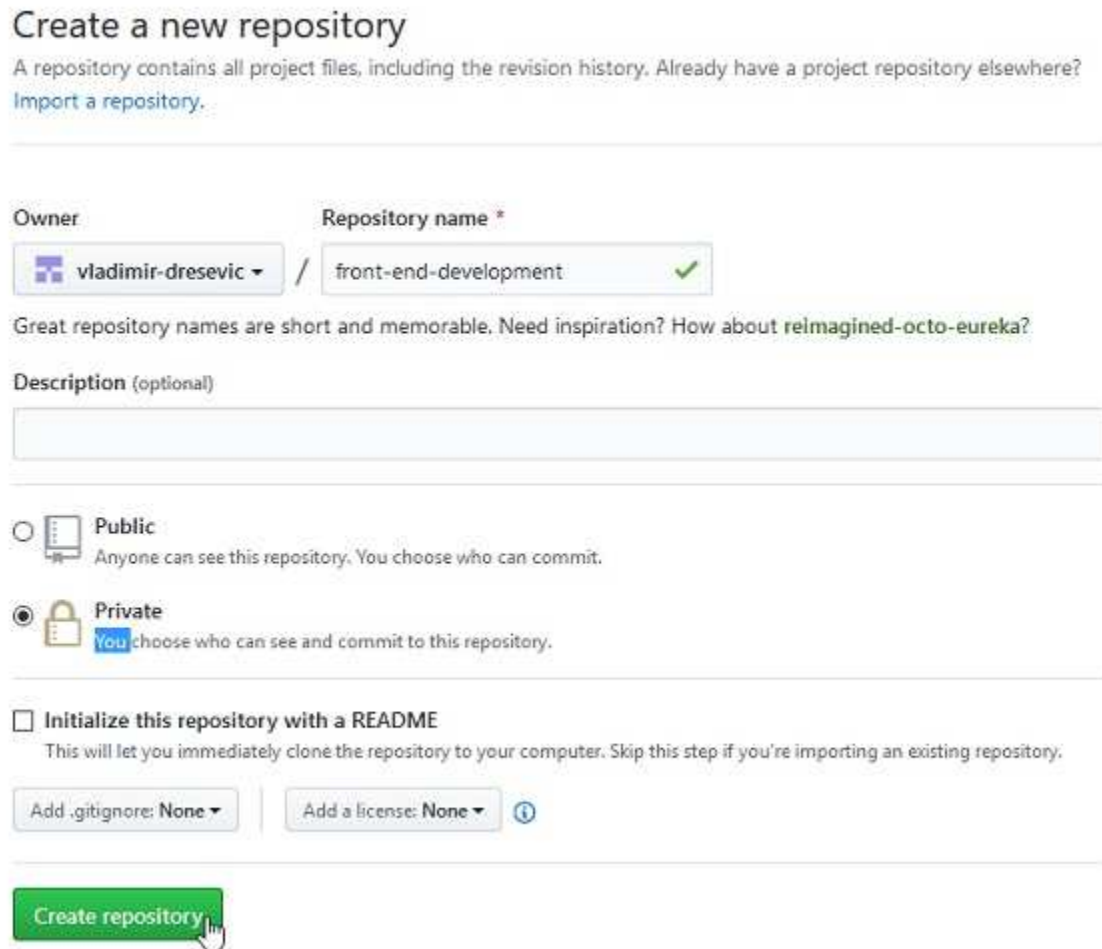
Korišćenje GitHuba podrazumeva prethodno kreiranje korisničkog naloga. Nakon kreiranja korisničkog naloga potrebno je kreirati repozitorijum (slika 11.6)



*Slika 11.6. Opcija za kreiranje repozitorijuma*



Kreiranje novog repozitorijuma na GitHubu zahteva minimalno podešavanje (slika 11.6).



**Create a new repository**

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner: **vladimir-drešević** / Repository name \*: **front-end-development** ✓

Great repository names are short and memorable. Need inspiration? How about [reimagined-octo-eureka?](#)

Description (optional):

☐ **Public**  
Anyone can see this repository. You choose who can commit.

☒ **Private**  
**You** choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** Add a license: **None** ⓘ

**Create repository**

*Slika 11.7. Podešavanje GitHub repozitorijuma koji se kreira*

Nakon kreiranja remote repozitorijuma na GitHubu, moguće je obaviti povezivanje lokalnog repozitorijuma sa onim udaljenim. To se najlakše postiže upućivanjem sledeće komande:

```
git remote add origin
```

<https://github.com/vladimir-drešević/front-end-development.git>

Komanda je remote add origin, dok je ostatak URL adresa na kojoj se nalazi udaljeni repozitorijum. Prikazanu komandu možete pronaći unutar samog GitHuba, nakon kreiranja repozitorijuma.

Kako bi se fajlovi iz lokalnog repozitorijuma prebacili u udaljeni, koristi se komanda push:

```
git push -u origin master
```

Ova komanda inicira upload, odnosno podizanje lokalnih fajlova na udaljeni repozitorijum.

Nešto ranije je rečeno da početak rada na projektu na kome učestvuje tim podrazumeva prethodno usklađivanje lokalnog repozitorijuma sa udaljenim. To se postiže korišćenjem komande pull:

```
git pull origin master
```

## Objavljivanje web sajta

Nakon završetka faze koja podrazumeva kodiranje i programiranje u procesu razvoja web sajtova i nakon uspešno obavljenih testova moguće je pristupiti objavljivanju web sajta. Proces objavljivanja web sajta zapravo se zasniva na omogućavanju pristupa korisnicima kojima je sajt namenjen.

Najčešći način objavljivanja web sajtova podrazumeva njihovo smeštanje na poseban serverski kompjuter, pri čemu oni postaju dostupni pod određenim domenskim imenom, kao što su *google.com* ili *wikipedia.org*. Bitno je znati da je za ovakvo objavljivanje neophodno osigurati postojanje:

- hostinga – reč je o prostoru na nekom web serveru;
- domenskog imena – jedinstvena URL adresa, preko koje će korisnici moći da pristupe web sajtu.

**Hosting** je prostor na nekom web serveru koji se može popuniti fajlovima web sajta. Takav web server je zadužen za isporuku resursa web sajta svim klijentima koji mu žele pristupiti. Bitno je znati da nabavka hostinga funkcioniše po principu iznajmljivanja serverskog prostora, što uglavnom nije besplatno. Svakako na webu postoje i hosting provajderi koji poseduju besplatne pakete, ali su uglavnom ograničenih mogućnosti i brzine. Za bilo kakvo ozbiljnije hostovanje neophodno je kupiti hosting. To se može obaviti online, posredstvom brojnih hosting provajdera. Dovoljno je da izguglate pojam web hosting i dobićete veliki broj hosting provajdera.

Pored hostinga, objavljivanje sajta podrazumeva postojanje i **domenskog imena**. U jednoj od prethodnoj lekcija bilo je reči o strukturi URL adresa i tom prilikom ste mogli da pročitate da se neki web sajt jednoznačno određuje domenskim imenom. Baš kao i prostor na nekom web serveru, i domensko ime se iznajmljuje. Iznajmljivanje domenskih imena postiže se posredstvom specijalnih ustanova, koje se nazivaju domenski registrati. U zavisnosti od toga koji tip domena želite registrovati, razlikuje se i proces registracije. Registrovanje nekog od takozvanih top-level domena može se obaviti posredstvom velikog broja online registratora. Registrovanje domena specifičnih za određene države (.rs, .co.uk, .at i slično) nešto je komplikovanije i može podrazumevati različite preduslove koje je neophodno ispuniti (državljanstvo, dokumenta i slično). Iznajmljivanje domenskih imena se uglavnom naplaćuje na godišnjem nivou.

Nakon obezbeđivanja postojanja hostinga i domenskog imena dovoljno je fajlove web sajta prebaciti na iznajmljeni prostor na web serveru. Za obavljanje takvog posla se najčešće koristi neki od FTP klijenata, odnosno programa koji su sposobni da obave transfer fajlova korišćenjem pravila FTP protokola. Veoma često i sami hosting provajderi poseduju određeni grafički online alat, koji omogućava transfer fajlova bez korišćenja posebnog FTP klijenta.

Proces objavljivanja web sajtova ne spada u najjednostavnije teme web razvoja. Takođe, ne može se ni podvesti pod nekoliko jednostavnih koraka, čijim praćenjem se dolazi do cilja s obzirom na to da postoji veliki broj nepoznatih i promenljivih. Zbog toga će u nastavku ove lekcije biti ilustrovan još jedan pristup za objavljivanje web sajtova, koji je veoma praktičan i koristan za početnike koji su u procesu učenja.

### Objavljivanje korišćenjem GitHub Pages servisa

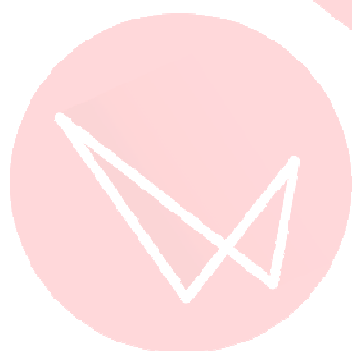
U dosadašnjem toku ove lekcije već je ilustrovano korišćenje GitHub online servisa. Ova online platforma poseduje još jednu veoma zanimljivu funkcionalnost. Naime, GitHub se može koristiti kao besplatni hosting provajder kako bi se brzo i lako mogli pregledati web sajtovi koji se razvijaju. Takva funkcionalnost se naziva GitHub Pages.

Preduslov za korišćenje GitHub Pagesa jeste postojanje javnog GitHub repozitorijuma. Nakon kreiranja javnog repozitorijuma i smeštanja projektnih fajlova unutar njega, jednostavnim aktiviranjem GitHub Pages opcije kompletan web sajt postaje javno dostupan online. U video-lekciji možete pogledati kompletno uputstvo za objavljivanje web sajta korišćenjem GitHub Pages servisa.

## Rezime

- Kako bi se uveo red u programerski posao i olakšalo održavanje, tokom vremena su kreirani brojni sistemi za verzionisanje koda.
- Sistemi za verzionisanje drugačije se nazivaju i version control systems, revision control systems i source control systems.
- Verzionisanje koda je pristup koji omogućava praćenje promena izvornog koda.
- Najpoznatiji sistem za verzionisanje danas je svakako Git.
- Tvorac Gita je Linus Torvalds, a reč je o sistemu koji je zbog svoje jednostavnosti, brzine i činjenice da je reč o softveru otvorenog koda stekao tokom vremena veliku popularnost.
- Git je alat bez korisničkog okruženja, pa zbog toga njegovo korišćenje podrazumeva upotrebu konzole ili terminala, u zavisnosti od operativnog sistema koji se koristi.
- Komanda za proveru postojanja Gita i njegove verzije je `git -version`.
- Repozitorijum je centralna figura Git sistema; često se naziva i Git projekat, a predstavlja kompletnu kolekciju fajlova i foldera od kojih je sačinjen jedan softverski projekat.
- Kreiranje Git repozitorijuma obavlja se upućivanjem komande `git init`.
- Smeštanje fajlova i foldera u repozitorijum obavlja se sprovođenjem dva koraka, koji se nazivaju *staging* i *committing*.
- *Staging* se može doživeti kao pripremni korak, a *committing* kao akcija koja pokreće upisivanje promena u repozitorijum.
- *Staging*, odnosno priprema za predaju, obavlja se korišćenjem komande `git add`.
- *Commit*, odnosno predaja ili završetak, obavlja se korišćenjem komande `git commit`.
- Provera statusa fajlova u Git repozitorijumu obavlja se korišćenjem komande `git status`.
- Pregled istorije commita u jednom repozitorijumu se može obaviti korišćenjem komande `git log`.
- GitHub je najpopularniji online sistem za hostovanje Git projekata.
- Najčešći način objavljivanja web sajtova podrazumeva njihovo smeštanje na poseban serverski kompjuter, pri čemu oni postaju dostupni pod određenim domenskim imenom.
- Objavljivanje web sajtova zahteva postojanje hostinga i domenskog imena.
- Hosting je prostor na nekom web serveru koji se može popuniti fajlovima web sajta.

- Domensko ime jeste adresa na kojoj će web sajt koji se objavljuje biti dostupan.
- GitHub Pages može se koristiti kao besplatni hosting provajder kako bi se brzo i lako mogli pregledati web sajtovi koji se razvijaju.



linkgroup