**ELEKTRO IN RAČUNALNIŠKA ŠOLA**

**POROČILO IZVEDENE VAJE PRI PREDMETU:**

**IPO**

**Številka in naslov vaje: VAJA 6 – GIT**

**Datum: Šolsko leto: 2024/2025**

**Ime in priimek dijaka:**

**Razred:**

**Učitelj:**

**Ocena in podpis učitelja:**

# UVOD V GIT

V nadaljevanju je iz knjige povzetih par pomembnih nastavitev oz. osnovnih informacij potrebnih za izvajanje vaj. Predlagam, da si pred pričetkom pregledate literaturo dostopno na: [GIT Knjiga](https://git-scm.com/book/sl/v2/Za%c4%8detek-O-nadzoru-razli%c4%8dic)

Začni s tem, da greš na njihovo spletno stran in si ustvariš nov račun (če ga še nimaš). Naslednji korak je ustvarjanje novega repozitorija. Repozitorij je lokacija kamor boš naložil kodo svojega projekta. Lahko je bodisi javen, bodisi zaseben. Določenim uporabnikom lahko daš pravice, da spreminjajo njegovo vsebino.

Na svoji strani najdi in pritisni *+New Repository*. Izberi si dobro ime, dodaj opis in si končal. Repozitorij se nahaja na naslovu github.com/<up. ime lastnika>/<ime repozitorija>.

**NAMESTITEV NA RAČUNALNIK**

Predlagam, da si predhodno na sistem namestite Notepad++ in kasneje namestite orodje za verzioniranje [**Git**](https://git-scm.com/) (<https://git-scm.com/>). Pri izbiri urejevalnika izberite Notepad++, ostalo le sledite korakom namestitve. Pravilnost nameščenega sistema git preverite z ukazom v ukazni vrstici git --version, kjer se vam kot odgovor izpiše nameščena različica Git sistem.

V ukazni vrstici vašega lokalnega okolja (npr. Command prompt v OS Windows, terminal v Linux ali Mac OS ali GIT MingW64), preverite nastavitve sistema git, kjer sta pomembna predvsem **uporabniško ime oz. ime in priimek ter elektronski naslov** avtorja uveljavitev.

Najprej preverite z naslednjima ukazoma, če sta omenjena parametra nastavljena.

*git config --global user.name*

*git config --global user.email*

V primeru da nista, jih nastavite s spodnjima ukazoma, kjer so podatki o imenu in priimku ali uporabniškem imenu ter elektronskem naslovu skladni z GitHub uporabniškim računom.

*git config --global user.name "uporabniskoime"*

*git config --global user.email "elektronskinaslov@gmail.com"*

Če želite preveriti svoje nastavitve, lahko uporabite ukaz git config --list za izpis vseh nastavitev, ki jih lahko Git najde v tistem trenutku

Sedaj, ko je vaša identiteta nastavljena, lahko nastavite privzeti tekstovni urejevalnik, ki bo uporabljen, ko Git pričakuje, da vpišete sporočilo. Če to ni nastavljeno, bo Git uporabljal vaš privzeti urejevalnik sistema. Predlagam, da si najprej namestite urejevalnik Notepad++ ali kakšen drug primerljiv in ga kasneje izberete med postopkom namestitve.

**POMOČ**

Če kadarkoli potrebujete pomoč med uporabo Gita, so na voljo trije enakovredni načini, da dobite izčrpno pomoč strani priročnika (**manpage**) za katerikoli ukaz Git:

git help <verb>

git <verb> --help

man git-<verb>

Na primer, dobite lahko pomoč manpage za ukaz git config, če poženete:

git help config

Če ne potrebujete obsežne pomoči v obliki man-strani, ampak samo potrebujete hitro osvežitev o možnostih za določen ukaz v Git, lahko zaprosite za bolj jedrnat izhod help z uporabo možnosti -h, kot na primer:

*git add -h*

**Pridobivanje repozitorija Git**

Projekt Git običajno pridobite z dvema pristopoma:

1. Vzamete lahko lokalni direktorij, ki trenutno ni pod nadzorom različic, in ga spremenite v repozitorij Git, ali pa
2. **Klonirate** obstoječi repozitorij Git od drugod.

V kateremkoli od obeh primerov dobite na vaši lokalni napravi repozitorij Git, ki je pripravljen za delo.

**Inicializacija repozitorija v obstoječem direktoriju**

Če imate projektni direktorij, ki trenutno ni pod nadzorom različic, in ga želite začeti nadzirati z Gitom, morate najprej iti v ta projektni direktorij. Če tega še niste počeli, je ukaz odvisen od sistema, ki ga poganjate:

za Linux:

cd /home/user/my\_project

za macOS:

*cd /Users/user/my\_project*

za Windows:

*cd C:/Users/user/my\_project*

in vpišete:

git init

To ustvari nov poddirektorij z imenom .git, ki vsebuje vse vaše potrebne datoteke repozitorija — skelet repozitorija Git.

**Kloniranje obstoječega repozitorija**

Če želite dobiti kopijo obstoječega repozitorija Git, na primer projekt, ki mu želite prispevati, je ukaz, ki ga potrebujete, git clone. Git dobi polno kopijo skoraj vseh podatkov, ki jih ima strežnik. Vsaka različica vsake datoteke zgodovine projekta je privzeto povlečena, ko poženete git clone. V bistvu, če se disk vašega strežnika pokvari, lahko pogosto uporabite skoraj katerikoli klon kateregakoli odjemalca, da strežnik povrnete v stanje, v katerem je bil, ko je bil kloniran.

Repozitorij klonirate z *git clone <url>.* Na primer, če želite klonirati Git povezano knjižnico, lahko to naredite na naslednji način:

*git clone https://github.com/username/folder*

To povleče vse podatke za ta repozitorij in izvleče delovno kopijo zadnje različice. Če greste v novi direktorij folder, boste tam videli projektne datoteke, ki so pripravljene za delo ali uporabo.

Če želite klonirati repozitorij v direktorij, imenovan nekaj drugega kot folder, lahko to določite kot naslednji argument v ukazni vrstici:

*git clone https://github.com/username/folder vaja*

Ukaz izvede enako stvar kot prejšnji, vendar ciljni direktorij je imenovan vaja.

**Snemanje sprememb v repozitorij**

Pomnite, da je lahko vsaka datoteka v vašem delovnem direktoriju v dveh stanjih: **sledena** ali **nesledena**. Sledene datoteke so datoteke za katere Git ve.

Nesledene datoteke so vse ostale — katerakoli datoteka v vašem delovnem direktoriju, ki ni bila v vašem zadnjem posnetku in ni v vašem področju priprave. Ko prvič klonirate repozitorij, bodo vse vaše datoteke sledene in nespremenjene, ker jih je Git ravnokar izvlekel in jih še niste kakorkoli urejali.

Ko boste urejali datoteke, jih Git vidi kot spremenjene, ker ste jih spremenili od zadnje potrditve.

**Preverjanje statusa vaših datotek**

Glavno orodje, ki ga uporabljate, da določite, katere datoteke so v kakšnem stanju, je ukaz:

*git status.*

**Sledenje novih datotek**

Da začnete slediti novi datoteki, uporabite ukaz git add. Da začnete slediti datoteki README, lahko poženete naslednje:

git add README

Če ponovno poženete svoj ukaz statusa, lahko vidite, da je vaša datoteka README sedaj sledena in dana v pripravo za potrjevanje.

Da je dana v pripravo, lahko veste, ker je pod naslovom »Changes to be committed«. Če na tej točki izvedete potrditev, bo različica datoteke v času, ko ste pognali git add, v naknadni zgodovini posnetka.

**Priprava spremenjenih datotek**

Spremenimo datoteko, ki je bila že sledena. Če spremenite prej sledeno datoteko in nato ponovno poženete vaš ukaz *git status*, se pojavi pod razdelkom imenovan »Changes not staged for commit« — kar pomeni, da je bila sledena datoteka spremenjena v delovnem direktoriju, vendar še ni bila dana v področje priprave. Za dodajanje v področje priprave, poženite ukaz git add.

**Odstranjevanje datotek**

Da odstranite datoteko iz Gita, jo morate odstraniti iz svojih sledenih datotek (bolj točno, odstraniti iz vašega področja priprave) in nato narediti potrditev. To naredi ukaz *git rm* in prav tako odstrani datoteko iz vašega delovnega direktorija, da je naslednjič ne vidite kot nesledeno datoteko.

# Vaja 1: OSNOVE GIT 70%

# Naloga 1: Namestitev Gita in nastavitev GitHub računa

1. Namesti Git na svoj računalnik (<https://git-scm.com/>) z vsemi privzetimi nastavitvami.
2. Ustvari račun na [GitHub](https://github.com/).
3. Nastavi svoje uporabniško ime in e-pošto v Git (*Command Prompt, Git Bash, Terminal*..)
4. Preveri nastavitve.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

# Naloga 2: Ustvarjanje in kloniranje repozitorija

**Navodila**

1. Na GitHub-u ustvari nov repozitorij in mu dodaj ime Vaja01.
2. V repozitorij naloži vsaj 5 datotek. Lahko tekstovne, batch iz prejšnjih vaj ali kakšne druge datoteke s programsko kodo drugih predmetov.
3. V ukaznem pozivu kloniraj repozitorij na svoj računalnik.
4. Premakni se v mapo repozitorija in preveri ali je ustvarilo klon.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 3: Dodajanje in potrjevanje (*commit*) datotek**

**Navodila**

1. Ustvari novo besedilno datoteko na lokalnem računalniku.
2. Dodaj jo v Git in jo potrdi (*commit*).
3. Preveri zgodovino (*log*)
4. Pošlji svoje spremembe v GitHub repozitorij.
5. Dodaj še več datotek in preveri spremembe z git status.
6. Preveri zgodovino z git log.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 4: Ustvarjanje veje (*branching*)**

**Navodila**

1. Ustvari novo vejo, katero poljubno poimenuj.
2. Ustvari novo datoteko in jo dodaj v novo ustvarjeno vejo.
3. Vse skupaj dodaj v Git in potrdi.
4. Preklopi nazaj na glavno vejo.
5. V datoteki ustvari nekaj sprememb in ponovno dodaj na Git in potrdi.
6. Primerjaj novo ustvarjeno vejo in main vejo (*git diff*).

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 5: Združevanje vej in reševanje konfliktov**

**Navodila**

1. V mapi z Git repozitorijem ustvari novo datoteko z naključno vsebino.
2. Vse skupaj dodaj v Git in potrdi.
3. Ustvari novo vejo *veja1* in preklopi na njo.
4. Uredi oz. naredi spremembo v prej ustvarjeni datoteki; spremembe dodaj v Git in potrdi.
5. Preklopi nazaj na *main* vejo in naredi spremembo v isti vrstici datoteke ter dodaj v Git in potrdi.
6. Poskusi združit veji – morala bi biti napaka (konflikt).
7. Odpri datoteko in ročno odpravi napako.
8. V Git dodaj popravljeno datoteko in potrdi. – Konflikt bi moral biti rešen.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 6: Delo z git ignore**

**Navodila**

1. V mapi z Git repozitorijem ustvari novo datoteko »*secret.txt«*, ki vsebuje občutljive podatke npr. geslo.
2. Ustvari datoteko .gitignore.
3. Preveri, da secrets.txt ni v staging area.
4. Dodaj .gitignore v staging area in naredi commit.
5. Dodaj več datotek v .gitignore, na primer vse datoteke s končnico .log ali vse mape z imenom temp. Datoteke ali mapo si moraš seveda prej pripraviti.
6. Preveri, da se te datoteke ne sledijo v Git-u.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 7: Delo z zgodovino (log)**

**Navodila**

1. Uporabi *git log*, da vidiš zgodovino commitov.
2. Poišči commit, kjer si dodal poljubno tekstovno datoteko (iz prejšnjih vaj). Poglej sporočila commitov, da najdeš pravega.
3. Poglej podrobnosti o tem commitu.
4. Poglej skrajšano zgodovino commitov.
5. Poglej statistiko sprememb za vsak commit.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 8: Povrnitev sprememb**

**Navodila**

1. Dodaj novo vsebino v poljubno datoteko in naredi commit.
2. Uporabi *git revert*, da povrneš zadnji commit.
3. Preveri ali je vsebina povrnjena na prejšnje stanje.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 9: Delo z git stash**

**Navodila**

1. V mapi z Git repozitorijem ustvari novo datoteko z naključno vsebino.
2. Datoteko dodaj v Git in potrdi.
3. Naredi spremembo v obstoječi datoteki in jo ne potrdi.
4. Preveri stanje *status*. Videti moraš, da je datoteka spremenjena.
5. Uporabi git stash za začasno shranjevanje.
6. Preveri seznam shranjenih sprememb.
7. Odpri in preveri kaj se je zgodilo v ustvarjeni datoteki.
8. Obnovi vse spremembe.
9. Odpri in preveri kaj se je zgodilo v ustvarjeni datoteki.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 10: Uporaba git rebase**

**Navodila**

1. V mapi z Git repozitorijem ustvari novo datoteko z naključno vsebino.
2. Datoteko dodaj v Git in potrdi.
3. Ustvari novo vejo in se pomakni na to vejo.
4. Uredi oz. naredi spremembo v prej ustvarjeni datoteki; spremembe dodaj v Git in potrdi.
5. Preklopi nazaj na main vejo in naredi še eno spremembo.
6. Preklopi na prej ustvarjeno vejo in namesto merge uporabi rebase. Če pride do konflikta, ga reši v datoteki in nato nadaljuj

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 11: Kloniranje in delo na tujem repozitoriju**

**Navodila**

1. Poišči en javni GitHub repozitorij in ga kloniraj (npr. <https://github.com/public-apis/public-apis>).
2. Ustvari novo vejo za spremembe.
3. Spremeni eno datoteko npr. README (dodaj novo vrstico). Dodaj v Git in potrdi.
4. Naredi *fork* repozitorija v svoj GitHub.
5. Nato pojdi na svoj GitHub repozitorij in klikni *New Pull Request*

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 12: GitHub Issues in Pull Requests**

**Navodila**

1. Odpri *Issue* na GitHub-u.

* Pojdi v GitHub repozitorij. Lahko obstoječi iz prejšnjih vaj, vendar naj ima vsaj 3 datoteke. V nasprotnem primeru jih dodaj.
* Klikni *Issues* → *New Issue*.
* Opiši težavo in jo oddaj.

1. Na svojem računalniku v mapi z Git repozitorijem ustvari novo vejo. POZOR: V kolikor še nimaš na računalniku tega repozitorija ga najprej kloniraj in potem ustvari vejo.
2. Popravi napako v eni izmed datotek (spremembo). Shrani datoteko in dodaj spremembe v Git.
3. Potisni (*push*) svojo vejo v repozitorij.
4. Pojdi na GitHub. Pojavilo se bo obvestilo, da si naložil novo vejo in da lahko odpreš *Pull Request*
5. Ustvari Pull Request in v opisi razloži kaj si spremenil.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 13: Deljenje kode z več sodelavci (Fork, Pull Request, Review)**

Predstavljajte si, da delate na odprtokodnem projektu in želite prispevati svojo kodo.

**Navodila**

1. Fork repozitorija

* Pojdite na GitHub in poiščite javni repozitorij.
* Kliknite Fork (zgornji desni kot). S tem ustvarite svojo kopijo repozitorija.

2. Kloniranje fork-anega repozitorija

* Klonirajte svojo različico repozitorija.
* Dodajte originalni repozitorij kot upstream (da lahko pozneje sinhronizirate spremembe).

3. Naredite spremembo in jo objavite

* Ustvarite novo vejo.
* Dodajte spremembo v eno od datotek (npr. index.html).
* Naredite commit in push.

4. Ustvarjanje Pull Requesta

* Pojdite na vaš fork na GitHub.
* Kliknite Compare & pull request in izberite originalni repozitorij.
* Dodajte opis in oddajte Pull Request.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

# Vaja 1: BASH in GIT 30%

**Naloga 1: Fork repozitorija**

Na vmesniku github.com opravite **fork**repozitorija <https://github.com/romanh89/vajaIPO> in v ukazni lupini prenesite vašo verzijo repozitorija na računalnik. **Opišite razliko med ukazoma git clone in git fork.**

V ukazni lupini dodajte v repozitorij vse vaše predhodno narejene **BATCH**datoteke (vsaj 5) in jih objavite na vašem git repozitoriju. **Opišite razliko med git commit in git push.**

**Naloga 2: Ustvari skripto za avtomatski commit**

Ustvari skripto auto\_commit.sh, ki vsako minuto shrani in objavi vse spremembe v Git repozitoriju. Skripta naj:

* Doda vse spremenjene datoteke v Git.
* Ustvari commit z avtomatskim sporočilom, kjer doda trenutno uro in datum.
* Pošlje spremembe (*push*) na GitHub.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**

**Naloga 3: Autosave**

Ustvari skripto sync\_with\_github.sh, ki bo vsakih 5 minut preverila ali so na GitHub-u nove spremembe in jih potegnila (pull) v lokalni repozitorij.

Skripta naj:

* Preveri, ali so na oddaljenem repozitoriju nove spremembe.
* Če so, jih potegne v lokalni repozitorij.
* Izpiše sporočilo o uspehu ali napaki.

Skripto zaženi in preveri njeno delovanje.

**Na kratko opiši vse uporabljene ukaze in opravljene korake oz. aktivnosti.**