RPO Vaja 1

Opis projekta

Enostaven kalkulator z uporabniškim vmesnikom.

Osnovni Pojmi

**Kaj je nadzor verzij in zakaj je pomemben?**

Nadzor verzij je sistem, ki spremlja spremembe v datotekah skozi čas. Beleži naslednje stvari:

Kaj se je spremenilo?

Kdo je naredil spremembo?

Kdaj je bila sprememba narejena?

Zakaj je bila potrebna (glede na sporočilo ob shranjevanju)?

Zakaj je pomemben?

Zgodovina in sledljivost: Vedete lahko, kako se je vaš projekt razvijal in zakaj so bile določene spremembe narejene.

Povrnitev sprememb: Če naredite napako, se lahko preprosto vrnete na prejšnjo, delujočo verzijo.

Sodelovanje: Več ljudi lahko hkrati dela na istem projektu brez, da bi si medsebojno prepisovali datoteke ali povzročali kaos.

Obnovitev: Če izgubite trenutno delo ali pokvari računalnik, je vaš projekt varen v "oblaku" (npr. na GitHubu).

**Razlike med centraliziranimi in distribuiranimi sistemi**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| | Lastnost |  |  | | --- | --- | --- | | **Centralizirani nadzor verzij (CVCS)** | **Distribuirani nadzor verzij (DVCS)** |
| **Primeri** | SVN, CVS | **Git**, Mercurial |
| **Arhitektura** | En sam centralni strežnik, ki vsebuje celotno zgodovino in različice. Uporabniki imajo le svoje trenutne datoteke. | Vsak uporabnik ima **popolno kopijo** celotnega repozitorija z vso zgodovino na svojem računalniku. |
| **Delo brez povezave** | Omejeno. Za večino operacij potrebujete povezavo do strežnika. | Zelo dobro. Lahko delate commit, pregledujete zgodovino in ustvarjate veje brez povezave. Ko se ponovno povežete, vse sinhronizirate. |
| **Hitrost** | Počasi, ker so operacije odvisne od omrežja. | Zelo hitro, ker se večina operacij izvaja lokalno. |
| **Odvisnost od strežnika** | Visoka. Če centralni strežnik odpove, sodelovanje se ustavi in obstaja tveganje izgube podatkov. | Nizka. Če glavni strežnik odpove, je mogoče projekt obnoviti iz katere koli lokalne kopije. |

**Ključni pojmi: repozitorij, commit, veja (branch), združevanje (merge), kloniranje, fork**

**Repozitorij (Repo):**

"Skladišče" ali mapa, kjer sistem za nadzor verzij hrani vse datoteke vašega projekta in njihovo zgodovino.

**Commit:**

Posamezna skupina sprememb, ki je naredjena v datotekah, skupaj z opombami (sporočilom), ki pojasni, kaj in zakaj je spremenjeno.

**Veja (Branch):**

Vzporedna različica repozitorija. Privzeta veja se običajno imenuje main ali master. Lahko se ustvari nova veja, da bi se delalo na novi funkciji ali popravljal hrošče, brez vpliva na glavno vejo.

**Združevanje (Merge):**

Postopek združevanja sprememb iz ene veje v drugo. Ko se konča z delom na lastnoj veji se "združi" nazaj v glavno vejo.

**Kloniranje (Clone):**

Postopek ustvarjanja popolne kopije obstoječega repozitorija (običajno z oddaljenega strežnika, npr. GitHub) na lokalni računalnik.

**Fork:**

Ustvarjanje osebne kopije nekega repozitorija na platformi (kot je GitHub). To omogoča, eksperimentiranje s spremembami, brez vplivanja na originalni projekt. Pogosto je predhodnica za prispevanje tujim projektom.

**Osnove Gita**

**Namestitev in konfiguracija Gita (uporabnik, email, SSH ključi)**



Uporabljam osebni email če že imam obstoječi github račun.

**Inicializacija repozitorija**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Dodajanje sprememb v "staging" in potrjevanje (commit)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Pregled zgodovine (log, diff, status)**

**Uporaba .gitignore za ignoriranje datotek**

****

Del .gitignore datoteke. Datoteka je prevzeta z [naslova](https://github.com/github/gitignore/blob/main/Python.gitignore).

**Veje in združevanje**

**Ustvarjanje in preklapljanje med vejami**

Če ni bilo mogoče ustvariti sliko zaslona, postopek gre na naslednji način:  
Klik na „main“ ali na „master“ -> Create new branch -> branch name

**Združevanje vej z glavno vejo**

Če ni bilo mogoče ustvariti sliko zaslona, postopek gre na naslednji način:

V „master“ veji se narede naslednji koraki Command Palette → "Git: Merge Branch". Potem se izbere veja katero želimo združiti.

**Reševanje konfliktov pri združevanju**

Ni bilo konfliktov pri združevanju.

**Razlika med fast-forward in non-fast-forward merge**

Fast-Forward Merge:

Najpreprostejši način združevanja, ki se lahko zgodi samo, če ciljna veja ni imela novih commit-ov odkar ste se razvejili.

Non-Fast-Forward Merge:

Zgodi se ko so se obe veji razvijali neodvisno in imajo različne commit-e.

**Oddaljeni repozitoriji (GitHub)**

**Kloniranje oddaljenih repozitorijev**

**A computer screen with text on it

AI-generated content may be incorrect.**

**Dodajanje oddaljenega repozitorija (origin)**

**A computer screen shot of a black screen

AI-generated content may be incorrect.**

**Uporaba forkov in prispevanje k javnim projektom**

Na github stranici (v tem primeru <https://github.com/octocat/Spoon-Knife>) kliknemo gumb “fork”. Potem se ustvari lastna kopija.

A screenshot of a computer

AI-generated content may be incorrect.

**Sodelovalni potek dela**

**Najboljše prakse pri pregledih kode**

**Najboljše prakse za pregled kode**

Avtorji kode morajo preveriti ali koda dela in napisati jasen opis. Tudi je priporočano imati več manjših PR-ja. Za pregledalce se priporoča da bojo konstruktivni in da se osredoče na funkcionalnost, varnost in berljivost kode.

**Pravila zaščite vej**

Blokirano neposredno dodajanje v zaščitene veje.

Obvezna zahteva za pull pred združitvijo.

Vsi testi morajo biti uspešni.

Rešeni konflikti v pogovoru.

Posodobljena veja z osnovno vejo.

**Uporaba "issues" za sledenje nalogam**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Napredna uporaba Gita**

**Rebase v primerjavi z merge**

Rebase:

Prepisuje zgodovino

Linearna zgodovina brez združitev

Nevarno za javne veje

Merge:

Shrani celotno zgodovino

Ustvari zapis združitve

Bolje za javne veje