Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

# Metodika verziovania kódu

Tím DILEMA

**Členovia tímu:** Bc. Zuzana Bachárová
Bc. Peter Bakoš
Bc. Martin Baláž
Bc. Marek Bernád
Bc. Imrich Nagy
Bc. Peter Tibenský
Bc. Juraj Volko
**Vedúci projektu:** Ing. Ján Lang, PhD.
**Tím:** č. 10 [FIIT DU]
**Akademický rok:** 2017/2018
**Dátum poslednej zmeny:** 12.11.2017
**Vypracoval:** Imrich Nagy

# 1 Úvod

Účelom tejto metodiky je opísať spôsob verziovania zdrojového kódu v tíme pri práci na tímovom projekte v akademickom roku 2017/2018. Je dôležité, aby pravidlá dodržiavali všetci členovia tímu.

Ako verziovací systém sme si zvolili systém git. Úložisko pre repozitár so zdrojovým kódom projektu je poskytované službou Bitbucket.

# 2 Postup

Každý člen tímu má pri práci s gitom povolený ľubovoľný výber nástroja na verziovanie. Voľba použitia príkazového riadku alebo grafického používateľského rozhrania záleží čisto na preferencii. Zaužívanými grafickými nástrojmi v našom tíme je aplikácia Atlassian SourceTree a verziovacie funkcie vývojového prostredia JetBrains PhpStorm.

Postup pri práci na zvolenej úlohe (*task*) by sa mal riadiť nasledujúcimi krokmi:

1. **Zvolenie vetvy**
   Pred začatím práce na úlohe je potrebné zabezpečiť, že sa člen tímu nachádza na vetve (*branch*), prislúchajúcej úlohe, na ktorej ide pracovať. V prípade, že sa nachádza na inej vetve, prepne sa na správnu vetvu použitím príkazu **git checkout <vetva>**. Ak sa táto vetva v lokálnej verzii repozitára nenachádza, je potrebné aktualizovať vzdialene sledované vetvy (*remote-tracking branches*) pomocou príkazu **git fetch origin**.
   Keby vetva nebola vytvorená ani vo vzdialenej verzii repozitára, je možné prepnúť sa na hlavnú vetvu (**git checkout master**), aktualizovať ju (**git pull origin master**), vytvoriť novú vetvu (**git checkout -b <vetva>**), a poslať ju na vzdialený repozitár (**git push origin <vetva>**). Alternatívnym spôsobom je vytvorenie vetvy pomocou webového používateľského rozhrania služby Bitbucket.

2. **Aktualizovanie vetvy**
   Pre zachovanie súvislej histórie commitov a minimálneho množstva vetvenia (s potenciálnymi konfliktmi) je pred začatím práce vhodné si vetvu aktualizovať. Spojenie vetvy zo vzdialeného repozitára (*origin*) do lokálneho sa vykonáva príkazom **git pull origin <vetva>**.

3. **Zobrazenie zmien**

   Prehľadnú a čitateľnú históriu vetvy a bezproblémovú prácu s jednotlivými commitmi pomocou nástrojov gitu je možné zaručiť dostatočnou frekvenciou commitov a ohľaduplným zoskupovaním zmien. Každý commit by mal tvoriť súvislý celok zmien v riadkoch súborov, ktoré spolu vytvárajú jednu pomenovateľnú zmenu (napríklad pridanie jednej funkcionality alebo oprava jednej chyby). Treba tiež brať do úvahy, že verzia repozitára v jednom commite by mala obsahovať spustiteľnú aplikáciu, ale zavedené zmeny nemusia byť hotové ani plne funkčné. Je preto lepšie tvoriť commity častejšie než menej často. Prehľad uvedených zmien od aktuálnej verzie repozitára je možné zobraziť príkazom **git status**. Rozdiel v zmenených riadkoch sa dá prehliadať po zadaní príkazu **git diff**.

4. **Označenie zmien**

   Označenie súborov relevantných k nasledujúcemu commitu sa uskutoční označením ciest k jednotlivých súborom alebo adresárom príkazom **git add <cesta>**. Pridanie všetkých súborov v repozitári je možné použitím znaku bodky ako cesta, ak je pracovným adresárom najvyšší priečinok repozitára. Interaktívny výber iba niektorých zmenených skupín riadkov zo zvolených súborov sa začína príkazom **git add -p <cesta>**. Vylúčenie zmien sa zadáva príkazom **git reset <cesta>**. Na permanentné odstránenie neoznačených zmien sa používa príkaz **git checkout -- <cesta>**.

5. **Vytvorenie commitu**

   V prípade, že všetky požadované zmeny sú označené, vytvorenie commitu prebieha príkazom git commit. Na zadanie commit správy bez editora priamo na príkazovom riadku sa používa príkaz **git commit -m <správa>**, kde nadpis a jednotlivé odstavce sú oddelené prepínačmi **-m** (napríklad **git -m "Message title" -m "Paragraph 1" -m "Paragraph 2"**). Posledný commit sa dá upraviť alebo doplniť ďalšími označenými zmenami (výlučne iba pred jeho zverejnením vo vzdialenom repozitári) pomocou príkazu **git commit --amend**.

6. **Zverejnenie commitov**

   Na zdieľanie commitov s ostatnými členmi tímu je potrebné zadať príkaz **git push origin <vetva>**. Môže sa pri tom stať, že táto akcia zlyhá, pretože iný člen tímu poslal svoje commity do vzdialeného repozitára skôr. V takom prípade je nutné použiť príkaz **git pull origin <vetva>** za účelom spojenia zmien z oboch sád commitov, vyriešiť prípadné konflikty, a zopakovať príkaz **git push origin <vetva>**.

# 3 Štruktúra vetiev

Hlavnou vetvou repozitára je **master**, ktorá reprezentuje aktuálny stav celého projektu s dokončenými súčasťami funkcionality.

Pre každú novú úlohu je vytvorená oddelená vetva (*feature branch*), ktorá sa s pôvodnou spája pri jej dokončení. Názvy týchto vetiev podliehajú formátu ***sprintN-task-name***, kde *N* je číslo šprintu a *task-name* je stručné meno riešenej úlohy, vyjadrené maximálne dvoma (prípadne troma) slovami oddelenými pomlčkou.

Manuálne commity sú vytvárané výlučne mimo master vetvy. Napredovanie tejto vetvy je riešené pomocou *pull requests*, ktoré predstavujú požiadavku o spojenie *feature branch* s hlavnou vetvou. V prípade výskytu konfliktov, ktoré zabraňujú spojeniu vetiev, je potrebné najprv naopak pripojiť master vetvu do feature vytvorením merge commitu, v ktorom sa dané konflikty manuálne odstránia.

# 4 Obsah a štruktúra commit správ

Každý commit vytvorený v rámci git repozitára musí povinne obsahovať správu, ktorá opisuje zmeny, ktoré nastanú po aplikovaní daného commitu. Keďže samotné zmeny v zdrojovom kóde, ktoré commit uvádza, je možné zobraziť pomocou nástrojov gitu, hlavným účelom tejto správy je opísať **účel, za akým bol commit vytvorený** (teda nie spôsob, akým bol dosiahnutý). Commit správy teda opisujú zmeny na relatívne vysokej úrovni, teda na hladine pridanej, zmenenej, alebo odstránenej funkcionality.

Commit správy v repozitári nášho projektu sú písané výlučne po anglicky. Obsah správy commitu pozostáva zo stručného nadpisu, ktorý môže byť nasledovaný jedným alebo viacerými odstavcami doplňujúceho textu. Odstavce sú navzájom a od nadpisu oddelené jedným prázdnym riadkom.

**Nadpis správy:** Na tomto riadku by sa malo nachádzať najviac 50 znakov. Pre zaručenie konzistentnosti správ by sa nadpisy vždy mali začínať veľkým písmenom a končiť bez bodky. Pre jednoduchú čitateľnosť je zavedený zaužívaný formát nadpisu commit správy: mal by tvoriť jednu vetu v rozkazovacom spôsobe, ktorá sa začína slovesom, ktoré najlepšie vystihuje zmenu, ktorú commit uvádza (napríklad *create, add, implement, start, fix, use, optimize, change, edit, alter, update, refactor, stop, remove*).

**Telo správy:** Formát odstavcov nasledujúcich po nadpise majú voľnú štruktúru, záleží čisto na autorovi commitu, akým spôsobom commit opíše, aby najlepšie vystihol uvedené zmeny. Túto časť commit správy nie je potrebné uviesť v prípade, že zmena je dostatočne malá, aby

účel commitu bol úplne obsiahnutý už v jeho nadpise. Telo commit správy sa preto píše iba v tedy, ak jeho nadpis dostatočne nevystihuje, čo ním bolo zmenené.