SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Tím 16 - MI16

Inteligentný importér verejných datasetov

Metodika pre verziovanie

Vedúci tímu:
Ing. Jakub Šimko, PhD.

Členovia tímu:
Bc. Ladislav Bari
Bc. Adam Ševčík
Bc. Adam Talian
Bc. Filip Varga
Bc. Milan Vaško
Bc. Jana Vrabľová

Dedikácia

Tento dokument predstavuje metodiku pre verziovanie zdrojového kódu na tímovom projekte tímu 16 (MI-16) na FIIT STU v akademickom roku 2017/2018.
Cieľom dokumentu je sumarizácia pravidiel a odporúčaní pri verziovaní zdrojového kódu.
Metodika je určená pre všetkých členov tímu.

Zaužívané pojmy

Verzovací systém (Git)

- systém umožňujúci verzovať súbory a z nich vytvárať vetvy. V projekte ide o systém Git[1].

Repozitár

- online úložisko zdrojového kódu kam majú všetci členovia tímu prístup
- využitie - synchronizácia / záloha / vytvorenie spustiteľných súborov (build) / nasadzovanie
- členovia tímu sem nahrávajú svoje úpravy zdrojových kódov
- náš repozitár - https://code.xit.camp/upvii

Vetva

- obsahuje množinu commit-ov - teda množinu zmien zdrojového kódu
- typy vetiev (podľa Gitflow):
    - master - špeciálna vetva - obsahuje kód, ktorý je vždy plne funkčný a nasaditeľný
    - development - špeciálna vetva - obsahuje kód, ktorý je nasadzovaný na vývojový server, sp
    - feature vetvy - obsahuje zmeny sú súčasťou tej istej funkcionality
    - hotfix vetvy - obsahuje zmeny opravujúce chybu
- spájanie vetiev sa nazýva mergovanie

Commit

- zachytáva zmeny v zdrojovom kóde
- pomenovaný podľa konvencie (uvedená nižšie v metodike - sekcia Vytváranie kontrolných bodov (commit-y))

[1] https://git-scm.com/

Manažment verziovania

Gitflow branching model

Gitflow predstavuje branching model zameraný na agilný spôsob vývoja. Tento model definuje rôzne typy vetiev, ktoré majú špecifické účely.

Hlavné vetvy:

- master vetva
    - obsahuje kód, ktorý je vždy plne funkčný, spĺňa definition of done a je okamžite nasaditeľný
- development vetva
    - kópia master vetvy
    - oproti master vetve obsahuje implementáciu, ktorá ešte nespĺňa definition of done, ale je potrebné ju otestovať na serveri
    - robí sa z nej deploy na vývojový server (staging server)
    - merge do master vetvy prebieha v čase vydania (release) - na konci šprintu

Vedľajšie vetvy:

- feature vetva
    - odvodzuje sa z development vetvy
    - používa sa pri vývoji novej funkcionality => vyvíja sa v nej nová funkcionalita (feature)
    - spojí sa development pomocou merge request-u
- bugfix vetva
    - odvodzuje sa z development vetvy
    - používa sa pri oprave chýb, ktoré nie sú kritické
    - spojí sa development pomocou merge request-u
- hotfix vetva
    - odvodzuje sa z master vetvy
    - používa sa ak sa v produkčnej vetve (master) vyskytne kritická chyba, ktorú treba opraviť
    - keď je chyba opravená, prebehne merge do master aj development vetvy

Vytváranie vetiev

Feature vetvy:

- sú odvodzované z development vetvy a pomenovávané nasledovne "*feature/[Jira-CisloTasku][kratky-popis-v-anglictine]*"

Hotfix vetvy:

- sú odvodzované z master vetvy a pomenovávané nasledovne "*hotfix/[Jira-CisloTasku][kratky-popis-v-anglictine]*"

Príklad pre task - “*IM-34 Vytvorenie ErrorHandlera a úprava súčasného chytania chýb*” by bol názov vetvy “*feature/IM-34-error-handling-refactor*”

Vytvorenie novej vetvy
“*git checkout -b* [NAZOV_NOVEJ_VETVY] [development / master]”.
Príklad
“*git checkout -b feature/IM-34-error-handling-refactor development*”.

Vytváranie kontrolných bodov (commit-y)

Commit:

- sa vytvára vždy keď je implementovaná ucelená časť funkcionality, pričom kód musí skompilovateľný
- je potrebné nahrať na repozitár (pushnúť)
- obsahuje deskriptívnu správu v angličtine - je nutné aby podľa nej bolo možné identifikovať zmeny, ktoré daný commit obsahuje (príklady commitov - https://chris.beams.io/posts/git-commit/#seven-rules)

Vytvorenie commitu:

- na vytvorenie môžeme použiť terminál alebo možnosti IDE. Uvádzame príklad pre terminál.
- “ *git status* ” - zobrazí zmenené súbory
- “ *git add [FILE]* ” - pridá súbor do vytváreného commitu
- “ *git commit -m \”message\”* ” - vytvorí commit
- “ *git push origin [NAZOV VETVY]* “ - nahrá zmeny na repozitár

V prípade potreby je možné na vytvorenie commitu použiť bez prepínača -m, kde môžeme zadať dlhšiu správu.
Po nahratí je na repozitári pustená kompilácia. Vždy treba skontrolovať, či prebehla korektne. Z vetiev development a master sa vytvára aj docker kontajner pre nasadenie na server.

Merge

Po implementovaní funkcionality vo feature (resp. hotfix) vetve sú tieto zmeny merge-nuté do development vetvy. Z development vetvy prebehne nasadenie na vývojový (staging) server. Po schválení produktovým vlastníkom (product owner, vedúci TP) sú zmeny mergnuté do master vetvy, odkiaľ sú nasadené na produkčný kód.

Poradie merge-ov:

- podľa gitflow - merge z feature vetiev prebieha vždy cez development!

Merge do developmentu
Merge môžem spraviť vždy pokiaľ verzia, ktorú chcem mergnúť:

- skompilovateľná
- funkčná stará funkcionalita

- nová funkcionalita nesmie vyhadzovat chyby
- v novej verzii nebolo zmenené API - pokiaľ bolo zmenené môžem mergnúť len ak bude upravená aj development vetva opačnej strany (frontend - backend)

Merge do master

- len z developmentu (prípadne hotfix)
- ak bol development nasadený a otestovaný na vyvojovom serveri
- ak bola verzia na vývojom serveri akceptovaná product ownerom (vedúcim)

Postup pri merge:

- vytvorenie merge request-u - najjednoduchšie cez webové rozhranie
- prebehne prehliadka zdrojového kódu (code review), ktorá sa riadi metodikou pre prehliadky zdrojového kódu
- po zapracovaní zmien vývojárom je možné spraviť merge
    - väčšinou možné cez webové rozhranie - neodporúčané
    - lokálne, postup je uvedený vo webovom rozhraní gitlab-u. Odporúča sa po mergi minimálne skompilovať program.

Nasadenie

Zp zdrojových kódov vo vetvách development a master sa po kompilácii vytvoria docker kontajnery, ktoré sa nasadzujú na vývojový, resp. produkčný server. Postup nasadzovania je uvedený v metodike pre nasadzovanie.
vetva development -> staging server
vetva master -> production server