Metodika verziovania kódu

Metodika verziovania kódu

Tím č. 03 FireAnt

Vedúci tímu: Branislav Pecher, Ivan Srba

Autor metodiky: Peter Mačinec

Dátum vytvorenia: 8.10.2019

Poslednú zmenu vykonal: Peter Mačinec

Dátum poslednej zmeny: 3.11.2019

Obsah

Pravidlá commitovania

Pre vytváranie commit správ platia nasledujúce pravidlá:

- **Commit správy sa píšu výlučne v jazyku angličtina**
- **Skratky pred commit správy**
    - kratšie: `ADD, FIX, REMOVE, MERGE, CHANGE, REVERT, WIP (Work In Progress - použi len v špeciálnych prípadoch)`
    - skratka pokrýva pôvodný zámer
        - pridávam niečo, aby som niečo opravil (FIX, nie ADD)
        - skratky `FIX` a `CHANGE` pokrývajú najmä jednoduché zmeny (rýchla oprava, malá zmena napr. na žiadosť zákazníka)
        - `FIX` a `CHANGE` nepokrývajú veľké zmeny, ktoré tvoria nové funkcionality (ak som celú špecifikáciu zle pochopil, tak celá prerábka nebude 20 FIX commitov)
- **Jeden commit pokrýva presne jednu zmysluplnú zmenu v kóde**
    - **Pomôcka:** ak to dokážem vyjadriť jednou vetou, je to v poriadku, akonáhle ma to núti použiť spojku 'a', už je to viac commitov

    - **Nesprávne:** 1 commit s viacerými správami, pokrývajúcimi viacero zmien (antipattern) - následne ak vidíme jednu správu, tak za commitom hľadáme práve túto funkcionalitu, ostatné sú skryté a teda veľmi ťažko dohľadateľné
- **Konzistentnosť CRLF a LF - používať len LF (týka sa najmä OS Windows)**
- **Commit nesmie obsahovať veci, ktoré k projektu nepatria**
    - vlastné nastavenia a konfigurácie
    - súbory z operačného systému či vývojového prostredia (._.DS_STORE, .idea)
- **Commit nemá vysvetľovať celý kód, na to je kód samotný a komentáre**
    - **Nesprávne:** `ADD: homepage slider that is handled by HomepageController and sliding.js library`
    - **Správne:** `ADD: homepage slider`
- **Dĺžka commit správy by nemala presiahnuť dĺžku 50 znakov**
    - doplnkové informácie a vysvetlenia je možné poskytnúť v dlhšom opise pod commit správou (medzi commit správou a opisom je jeden voľný riadok)
    - dĺžka 50 znakov neplatí pre MERGE commity pri zapracovaní Pull requestov
- **Commit nesmie pokrývať viac funkcionality, ako je opísané v správe**
    - skryté zmeny - pridám niečo a odrazu aj niečo fixnem, ale správa pokrýva len pridanie funkcionality
    - malé zmeny by tam nemali byť tiež obsiahnuté (CSS zmeny - pridávam funkcionalitu, ale narýchlo fixnem pomocou 3 riadkov ccs-ka aj dizajn)
- **Kód v commitoch nesmie byť tak rozsiahly, že je náročné sa v ňom orientovať**
    - treba zvážiť viacero commitov, aj ak ide o podobnú vec (napr. refaktor je možné rozdeliť na refaktor domovskej stránky, administrácie, ...)
- **Necommitovať zmeny vo formatovaní a zároveň zmeny v kóde**
    - formátovanie kódu by mal byť 1 commit, ďalší môže byť upravená funkcionalita - miešanie následne zakryje reálne zmeny.
- **Zmeny z vetvy nadradenej (napr. develop nad feature vetvou) zapracovávame metódou rebase**
    - napr.: ak chceme/potrebujeme zapracovať zmeny, ktoré boli zapracované do vetvy *develop*, použijeme metódu **rebase**, nie **merge**.

Pravidlá git vetiev (branches)

Vetvy vo všetkých git repozitároch majú nasledovnú štruktúru:

- **master**
    - hlavná vetva, ktorá odráža aktuálny stav na produkčnom serveri
    - je zamknutá na priame pushovanie do vetvy, povolené len cez pull requesty
    - pull request do tejto vetvy môže byť len z vetiev *develop* a *hotfix*
- **develop**
    - vývojová vetva, ktorá odráža aktuálny stav na testovacom (staging) serveri
    - je zamknutá na priame pushovanie do vetvy, povolené len cez pull requesty
- **ostatné vetvy podľa Gitflow:**
    - **feature/name-of-feature** - pridanie novej funkcionality, následne sa zlučuje s vetvou *develop* pomocou pull requestu
    - **hotfix/name-of-bug** - oprava chýb, následne sa zlučuje s vetvami *master* a *develop* pomocou pull requestu, ako to určuje Gitflow

Každá **feature** vetva odráža jednu user (ale nielen user) story v šprinte.

Pravidlá pull requestov

Pull request môže vytvoriť ktorýkoľvek člen tímu, ktorý na zmenách v pull requeste pracoval (zväčša ten, čo pridal posledné zmeny, prípadne ako sa členovia tímu dohodnú).

Pre vytváranie pull (merge) requestov platia nasledovné pravidlá:

- Názov pull requestu v krátkosti opíše pridanú funkcionalitu a tvorí akoby abstrakciu nad jednotlivými commitmi
- Formát názvu pull requestu je nasledujúci (príklad): *Add support for elasticsearch*
- Opis pull requestu má vždy na prvom riadku značenie *user story,* nasledovaný **jedným voľným riadkom** (v prípade, že bude mať PR ďalší popis). Funkcionality jednoducho vymenujeme v *bullet liste*. Príklad opisu:

```
Story: FA-1

The following functionalities have been added:
- simple description of new functionality 1
- simple description of new functionality 2

Longer description if needed.
```

Pri jednoduchších PRs vytvoríme len jednoduché popisy:

```
Story: FA-1

Integration with elasticsearch have been added.
```

- Záludné časti, prípadne časti potrebné vysvetlenia (aj) pre reviewera zdokumentované v opise pull requestu
- Pri vytvorení PR sa vyberú členovia tímu, ktorí kód majú prezrieť (aspoň 1) - výber členov určuje **Metodika prehliadok kódu**
- Názov aj opis sa píše výlučne v jazyku angličtina
- Po schválení a zlúčení vetvy sa vetva vymaže (pokiaľ nejde o vetvu **develop**)
- Pull requesty sa zapracúvajú výlučne metódou **merge**
- Commit message pri zapracovaní pull requestu sa riadi nasledovnými pravidlami:
    - Používať skratku *MERGE*
    - Predpísaná štruktúra: *MERGE: branch_from into branch_to #{PR_NUMBER}*
    - Konkrétny príklad, spĺňajúci štruktúru: *MERGE: feature/elasticsearch-integration into develop #3*

Pravidlá verziovania systému a nasadzovania

Metodika verziovania systému

Pri verziovaní systému sa riadime pravidlami **Semantic versioning** (https://semver.org/).

Verzia systému má nasledujúci formát: **v{MAJOR}.{MINOR}.{PATCH}** (napríklad **v1.2.5**)

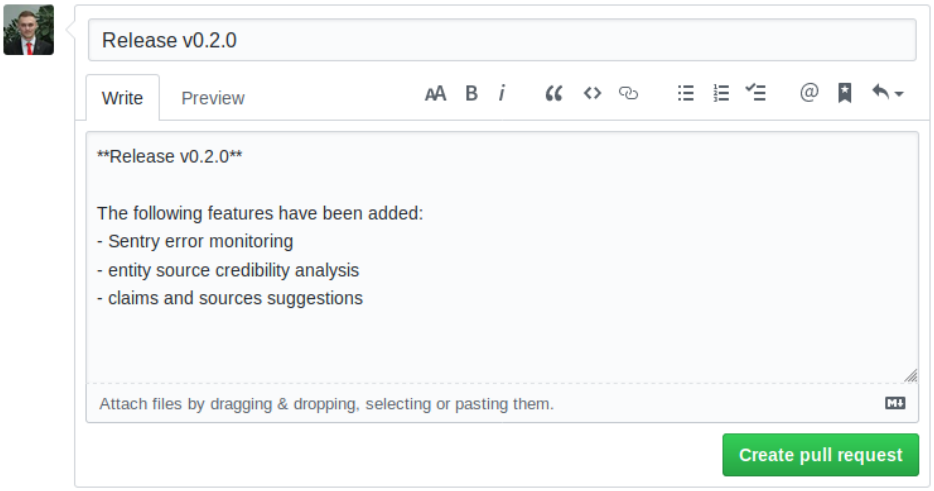
Pravidlá pre inkrementovanie verzií sú nasledovné:

- MAJOR verzia sa zvýši, ak sa vytvárajú nekompatibilné zmeny (napr. úplne nová verzia systému)
- MINOR verzia sa zvýši, ak pridávame kus funkcionality, avšak zachovávame spätnú kompatibilitu
- PATCH verzia sa zvýši, ak vytvárame malé zmeny, opravy chýb, oprava kompatibility

Nasadzovanie systému

Nasadzovanie kódu do produkcie zahŕňa nasledujúce kroky:

1. Vytvorenie Pull requestu na zapracovanie zmien z vetvy **develop** do vetvy **master**. Musí byť dodržaný formát názvu a opisu pull requestu, zobrazený na príklade:



2. Zapracovanie Pull requestu, pričom správa merge commitu musí spĺňať body v metodike **Pravidlá pull requestov** uvedenej v tomto dokumente.
3. Pridanie *tagu*, určujúceho verziu nasadenia (POZOR, potrebné priradiť k vetve **master**):
    a. Verzia musí spĺňať pravidlá, uvedené v časti **Metodika verziovania systému**.
    b. Pridanie nadpisu, ktorý spĺňa nasledujúcu formulu: **FireAnt Backend vX.X.X** alebo **FireAnt Frontend vX.X.X** (podľa aplikácie). Verzia je rovná zadanému tagu z bodu č. 3.
    c. Pribaliť potrebné súbory (napr. RASA model v zip súbore *rasa_model.zip,* obsahujúci zložku *models* a v nej model vo formáte . *tar.gz*).
    d. Pridanie opisu pridaných funkcionalít. Príklad: