Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16, Bratislava 4

Metodika verziovania

Tímový projekt

Tím č. 21

Vedúci: Ing. Ivan Srba, PhD.

tim21.2018.fiit@gmail.com

Autor metodiky: Jakub Sedlář

1 Vymedzenie obsahu

Nasledujúci dokument opisuje pravidlá a postupy pri práci s nástrojom git a platformou Github a pravidlá pre vytváranie a označovanie priebežných a finálnych verzií produktu.

Metodike nepodlieha repozitár tímového webu, nakoľko povaha jeho úprav je veľmi odlišná od ostatných repozitárov a nie je hostovaný na platforme Github. Výnimkou je tiež repozitár Prototypes, ktorý slúži najmä na rýchle zdieľanie testovacích skriptov a prototypov na zahodenie a jeho obsah sa nepovažuje za súčasť vytváraného produktu.

2 Vymedzenie pojmov a skratiek

3 Dedikácia metodiky

Metodikou sa riadia členovia tímu TrafficWatch, v rozsahu predmetu Tímový projekt.

4 Roly a zodpovednosti účastníkov

4.1 Autor pull requestu

Osoba, ktorá vytvára pull request.

Povinnosti:

- Reaguje na pripomienky reviewerov, dostáva pull request do stavu, v ktorom môže byť zlúčený.
- Zlučuje pull request po schválení.

4.2 Reviewer

Osoba vykonávajúca code review

Povinnosti:

- Prezerá kód, ktorý je zahrnutý v pull requeste, kontroluje jeho správnosť, dohliada na dodržiavanie metodiky písania kódu, podáva pripomienky, rozhoduje, kedy je pull request pripravený na zlúčenie.

4.3 Git master

Osoba zodpovedná za metodiku a správu repozitárov.

Povinnosti:

- Vytvára metodiku verzovania a dohliada na jej dodržiavanie.
- Rieši problémy, ktoré nastanú pri práci s nástrojom git.

5 Vetvy

Pre každú user story sa vytvára jedna vetva pre každý nezávislý komponent, na ktorom sa v rámci story pracuje. Novú vetvu možno vytvoriť viacerými spôsobmi:

- git branch <názov vetvy>
- git checkout -b <názov vetvy>

Názov každej vetvy (okrem vetiev master a develop) má tvar <typ>/<názov> kde <názov> pozostáva z kľúča tasku (tak ako je v Jire) a anglického opisu toho, čo sa vo vetve robí (malými písmenami bez diakritiky, s pomlčkami namiesto medzier), napr. TP-01-example-task, a <typ> je buď feature (ak je účelom vetvy pridanie funkcionality), bugfix (ak je účelom vetvy odstránenie chyby), hotfix (ak je účelom vetvy urýchlené odstránenie chyby), alebo release (špeciálne vetvy pre mergovanie do vetvy master). Zvyčajne vytvárame jednu vetvu pre každú user story, ak to však okolnosti vyžadujú je akceptovateľné vytvoriť vetvu aj pre subtask.

Feature a bugfix vetvy sa vetvia z vetvy develop. Hotfix vetvy sa vetvia z vetvy master.

Vetvy develop a master by mali obsahovať hotový, otestovaný kód. Kód vo vetve develop daného komponentu musí byť sám o sebe funkčný. Kódy vo vetvách master jednotlivých komponentov musia byť akceptované product ownermi a navzájom kompatibilné. Zmeny do vetvy develop sa pridávajú výhradne cez pull requesty z feature, bugfix a hotfix vetiev. Zmeny do vetvy master sa pridávajú výhradne cez pull requesty z hotfix vetiev a z vetvy develop.

6 Commity

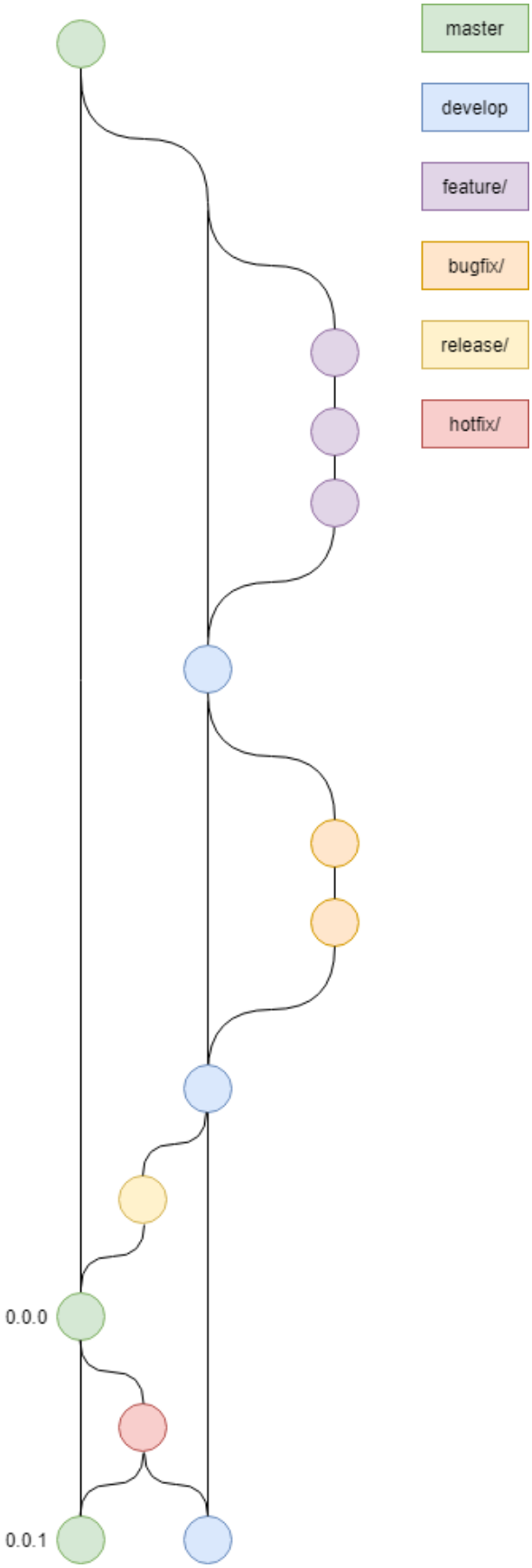
Commit message sa píše v angličtine, v minulom čase bez bodky na konci (napr. Ädded option to track only white cars").

Na konci každého commit message v samostatnom riadku musí byť kľúč úlohy z JIRA, s ktorou commit súvisí (dôležité kvôli integrácií týchto nástrojov).

Pred pushnutím commitu na Github je vhodné ešte raz kód odskúšať a skontrolovať zmeny, ktoré v ňom boli zahrnuté, a commit message. Keď je commit pushnutý, riešenie problémov je náročnejšie. Na kontrolu správnosti commitu je niekoľko užitočných príkazov:

- git show - vypíše všetky zmeny vykonané v poslednom commite
- git log - vypíše zoznam commitov (umožní kontrolu commit messagu)
- ešte pred vykonaním commitu:
  - git diff –cached - vypíše všetky aktuálne zmeny, ktoré budú súčasťou commitu, ak sa hneď vykoná
  - git diff - vypíše všetky aktuálne zmeny, ktoré nebudú súčasťou commitu (je možné pridať ich doňho prostredníctvom príkazu git add)
- riešenie vzniknutých problémov:
  - git commit –amend - podobné ako obyčajný git commit, ale namiesto vytvorenia nového commitu upraví ten predchádzajúci. Takýmto spôsobom je možné pridať do commitu ďalšie súbory, opraviť bugy alebo preklepy, alebo jednoducho zmeniť commit message. Keďže ide o prepisovanie histórie, nie je možné to použiť ak už commit bol pushnutý na Github.

Keď commit spĺňa všetky kritéria, mal by byť čo najskôr pushnutý na Github prostredníctvom git push, resp. git push –set-upstream origin <názov vetvy>, ak ide o novovytvorenú vetvu, ktorá ešte nikdy nebola pushnutá.


master
develop
feature/
bugfix/
release/
hotfix/
0.0.0
0.0.1

7 Pull requesty

Názov pull requestu je totožný s názvom vetvy, z ktorej sa vytvára.

Pull request by mal mať prideleného aspoň jedného reviewera a medzi reviewermi musí byť reporter pridelený na task v Jire, pokiaľ nie je zároveň tým, čo pull request vytvára. Pull request nie je možné mergenúť pokiaľ ho všetci revieweri neschvália.

Maximálne raz za šprint sa z vetvy develop vytvorí vetva s názvom "release/<číslo verzie>", ktorá sa mergne do vetvy master.

Pull request sa neotvára ak sa vo vetve develop od posledného šprintu neudiali žiadne zmeny. Ak by vykonané zmeny narušili kompatibilitu s inými komponentami, pull request sa môže vytvoriť, ale nebude schválený kým sa nevykonajú príslušné zmeny v dotknutých repozitároch. Pull requesty vo všetkých dotknutých repozitároch sa potom mergnú bezprostredne za sebou, čím sa zachová ich vzájomná kompatibilita.

Pull requesty z hotfix vetiev sa najprv vytvoria pre vetvu master, kde sa reviewujú a upravujú. Keď je tento pull request schválený, vytvorí sa nový pull request aj pre vetvu develop, ktorý bude merguntý okamžite, keďže review pre dané zmeny už prebehol.

Schválený pull request merguje ten, čo ho vytvoril.

Pri vytváraní pull requestu a iných relevantných akciách musia byť revieweri a iné zainteresované osoby upozornené podľa metodiky komunikácie.

8 Číslovanie verzií

Každej verzií komponentu aplikácie prináleží práve jeden commit v master vetve príslušného repozitára. Konvencia číslovania verzií vo fáze vývoja pred prvým verejným releasom je s viacerých dôvodov mierne odlišná od tej po ňom (omnoho nižšie nároky na stabilitu rozhrania, nepoužívanie verzie na marketingové účely atď.). Číslo verzie má tvar <major>.<minor>.<patch>, kde <major>, <minor> a <patch> sú prirodzené čísla.

Číslovanie v pre-release fáze Prvá verzia (prvý commit do master vetvy okrem inicializácie repozitára) má označenie 0.0.0. Pravidlá pre inkrementáciu čísel sú nasledovné:

- Major
  - počas pre-release fázy fixovaná na 0
- Minor
  - inkrementuje sa po každej zmene, ktorá mení rozhranie komponentu
    - platí aj pre spätne kompatibilné zmeny
    - platí aj pre zmeny v používateľskom rozhraní, pokiaľ ide o pridanie/odobratie/redesign funkcionality alebo redesign navigácie. Drobné zmeny (zmena farby, posunutie tlačidla o pár cm, ...) sa nepočítajú
  - inkrementuje sa po výraznej zmene vnútornej implementácie. Výrazné zmeny sú:
    - zmena, ktorá môže byť pozorovaná z iných komponentov napriek tomu, že rozhranie sa nezmenilo (výrazná zmena v performance, v presnosti odosielaných dát, frekvencií odosielania dát, ...)
    - veľmi rozsiahla refaktorizácia (úplný rewrite výraznej časti komponentu, alebo aj celého komponentu)
    - podľa vlastného uváženia, po konzultácií s git mastrom/zvyškom tímu
- Patch
  - ak sa inkrementovala minor verzia, resetne sa na 0
  - inak sa inkrementuje

8.1 Číslovanie v release fáze

Prvá produkčná verzia má označenie 1.0.0. Pravidlá pre inkrementáciu čísel sú nasledovné:

- Major
  - inkrementuje sa po každej zmene vo verejnom rozhraní komponentu, ktorá nie je spätne kompatibilná s ostatnými komponentami
    - zmeny v používateľskom rozhraní sa považujú za spätne nekompatibilné, ak bola funkcionalita odstránená alebo presunutá, teda používateľ nemôže dosiahnuť požadovaný výsledok vykonaním postupnosti krokov, na akú bol zvyknutý
- Minor
  - ak bola inkrementovaná major verzia, resetne sa na 0
  - inkrementuje sa po každej spätne kompatibilnej zmene vo verejnom rozhraní
  - inkrementuje sa po výraznej zmene vnútornej implementácie. Výrazné zmeny sú:
    - zmena, ktorá môže byť pozorovaná z iných komponentov napriek tomu, že rozhranie sa nezmenilo (výrazná zmena v performance, v presnosti odosielaných dát, frekvencií odosielania dát, ...)
    - veľmi rozsiahla refaktorizácia (úplný rewrite výraznej časti komponentu, alebo aj celého komponentu)
    - podľa vlastného uváženia, po konzultácií s git mastrom/zvyškom tímu
- Patch
  - ak bola inkrementovaná alebo resetnutá minor verzia, resetne sa na 0
  - inak sa inkrementuje po každej zmene, ktorá nespĺňa požiadavky pre inkrementáciu minor alebo major verzie