Slovenská technická univerzita v Bratislave Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16, Bratislava 4

METODIKA VERZIOVANIA

BlockPay
Blockchain Busters

Tímový projekt
Tím č. 20
Vedúci: Ing. Kristián Košťál
fiittim201920@gmail.com

Vypracoval: Jozef Daxner,
Lenka Koplíková

1 Vymedzenie obsahu

Tento dokument popisuje postupy a pravidlá pri práci s nástrojom git a platformou GitHub v tíme Blockchain Busters. Táto metodika sa nevzťahuje na git repozitár slúžiaci na verziovanie tímového webu, lebo jeho štruktúra je odlišná od štruktúry projektu.

2 Dedikácia metodiky

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

3 Informácie o repozitári

Repozitár je umiestnený na GitHub cloude. Administrátorom je Jozef Daxner (používateľ camperko). Ostatní členovia tímu sa k repozitáru pripájajú pomocou ssh kľúča. Adresa repozitára je https://github.com/camperko/TP20.git

- `git clone https://github.com/camperko/TP20.git`

4 Vetvy

Vetvy feature a bugfix sa tvoria z vetvy develop, vetva develop vytvára vetvy release. Vetva develop by mala obsahovať iba kompletné úlohy (definované pomocou Metodiky definition of done), ktoré sú integrované do zvyšku projektu a sú navzájom kompatibilné.

V štruktúre git sa nachádzajú tieto vetvy:

- `master`
- `release_<číslo verzie>` – vetva, do ktorej sa mergujú develop vetvy pri skončení šprintu
- `develop` – hlavná vetva počas jednotlivých šprintov, do ktorej sa mergujú feature a bugfix vetvy po tom, ako sú *"done"* [1]
- `feature_<číslo úlohy podľa Azure DevOps>` – vetvy pre jednotlivé tasky
- `bugfix_<číslo bugu podľa Azure DevOps>` – vetvy pre jednotlivé bugfixy

Novú vetvu možno vytvoriť dvoma spôsobmi:

[1] Ako je uvedené v dokumente Metodika definition of done

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

Užitočné príkazy na správu vetiev:

- `git branch` - zobrazenie všetkých aktívnych vetiev v repozitári, aktuálna vetva je označená symbolom `*`.
- `git fetch` - aktualizovanie mapovania vzdialených vetiev
- `git checkout <názov vetvy>` - prepnutie na vetvu, na ktorej chceme pracovať

5 Commit

Commit sa vytvára po dokončení časti úlohy. Commit message je písaná v angličtine v minulom čase, je krátka a výstižná. Stručne popisuje pridávanú funkcionalitu/pridané zmeny. Na konci commit message sa pripíše `AZ#<číslo úlohy>`

Keď jednotlivý commit spĺňa všetky kritéria, je pushnutý na server. Commit sa kontroluje pomocou nasledovných príkazov:

- `git show` - zmeny vykonané v poslednom commite
- `git log` - zoznam commitov
- `git diff –cached` - zmeny, ktoré sú súčasťou commitu
- `git diff` - zmeny, ktoré nie sú súčasťou commitu
- `git commit –amend` - možnosť upraviť predchádzajúci commit

Iné užitočné príkazy na prácu s commitom:

- `git add [<súbory>, *]` - pridanie súborov do commitu, ktoré boli lokálne zmenené/pridané/odstránené
- `git status` - zobrazenie súborov ktoré sú/ nie sú pridané do commitu
- `git commit -m <správa>` - samotný commit s commit message
- `git checkout -- .` - discardnutie vykonaných zmien

6 Push

Commit sa vkladá na server pomocou príkazu:

- `git push origin <názov vetv>`
- `git push --set-upstream origin <názov vetvy>` - ak ide o ešte nepushnutú vetvu
- `git revert <hash commitu>` - zvrátenie commitu

7 Pull request

Pull request musí mať aspoň jedného kontrolóra, ktorý schvaľuje a kontroluje pull request okrem jeho žiadateľa. Bez jeho schválenia nie je dovolené vetvy mergenúť. Schválený pull request merguje ten, čo ho vytvoril. Názov pull requestu je rovnaký ako názov vetvy, z ktorej sa vytvára.

Raz za šprint sa mergne vetva develop do vetvy release.

Pri vytváraní pull requestu a iných dôležitých akciách musia byť oboznámené všetky zainteresované osoby v tíme.

8 Číslovanie verzií

Číslovanie verzie má tvar `<major>.<minor>.<patch>` a toto číslo sa používa pri mergovaní do vetvy release.

Prvá produkčná verzia má číslo 1.0.0.

Konvencia číslovania je nasledovná:

- major
    - v pre-release fáze má hodnotu 0
    - inkrementuje sa po každej zmene, ktorá nie je spätne kompatibilná s ostatnými komponentami
- minor
    - inkrementuje sa po každej zmene, ktorá mení rozhranie komponentu
    - inkrementuje sa po každej výraznej zmene vnútornej implementácie (napríklad veľká refaktorizácia, zmena pozorovaná z iných komponentov)
    - ak bola inkrementovaná major verzia, resetne sa na 0
- patch
    - ak sa inkrementovala minor verzia, resetne sa na 0
    - inak sa inkrementuje