SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Metodika verzií zdrojového kódu

Tímový projekt – Stratos FIIT

2016 **Jakub Findura**

1 Úvod

Táto metodika je určená členom tímu, ktorí v rámci svojich úloh tvoria zdrojový kód niektorej zo súčastí systému. V rámci nášho projektu existuje viacero samostatných celkov systému, ktoré sú odlišné z pohľadu zdrojového kódu. Preto aj v rámci správy zdrojového kódu sú jednotlivé súčasti logicky oddelené.

Pre zdieľanie zdrojového kódu v tíme a uchovávanie jednotlivých verzií využívame systém git. Tento systém nám poskytuje služba GitHub. Pre každú samostatnú časť systému je vytvorený samostatný repozitár.

1.1 Roly

Správca repozitára – Člen tímu, ktorý spravuje daný repozitár. Zabezpečuje jeho vytvorenie, správu vetiev, pull requestov, kontroluje vykonávanie code review.

Vývojár – člen tímu, ktorý implementoval daný zdrojov kód. Spravuje svoj lokálny repozitár a odosiela vytvorený zdrojový kód na vzdialený repozitár.

Kontrolór kódu – člen tímu, ktorý vykonáva kontrolu kódu. Kontrolór v rámci kontroly kódu, kód nevytvára ani neopravuje.

1.2 Súvisiace metodiky

Metodika riadenia

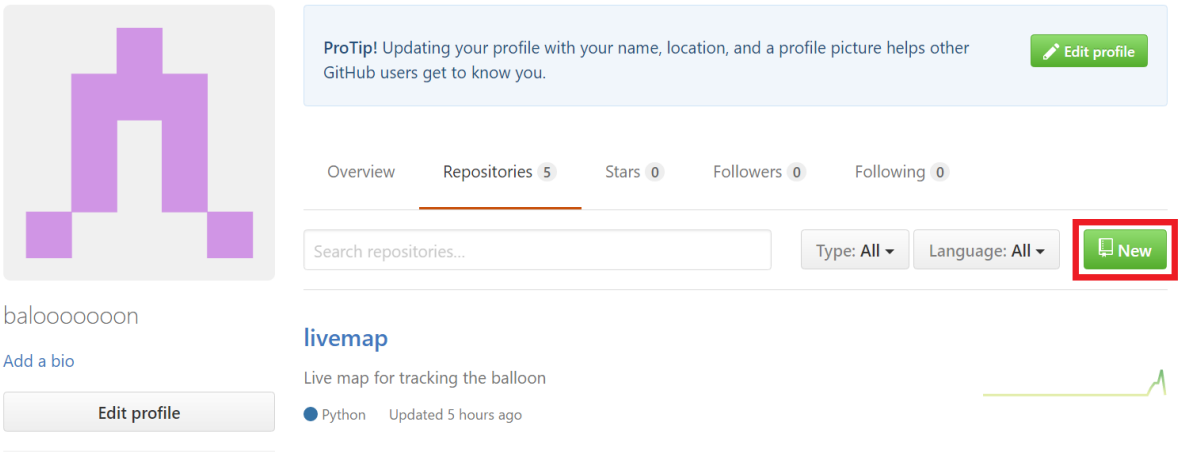
1.3 Zoznam pojmov a skratiek

Git	systém na správu a verziovanie zdrojového kódu
Branch	Vetva, možnosť robiť nezávislé zmeny zdrojového kódu.
Pull Request	Požiadavka na zlúčenie nového zdrojového kódu, ktorý musí prejsť kontrolou
Commit	Súbor zmien zdrojového kódu v repozitári
Merge	Spojenie dvoch vetiev
Code review	Prehliadka kódu, slúži na kontrolu zdrojového kódu, či je vhodný na pridanie do funkčnej vetvy.
Pull	Prevzatie zmien ostatných členov tímu zo vzdialeného repozitára.
Push	Odoslanie lokálnych zmien na vzdialený repozitár.

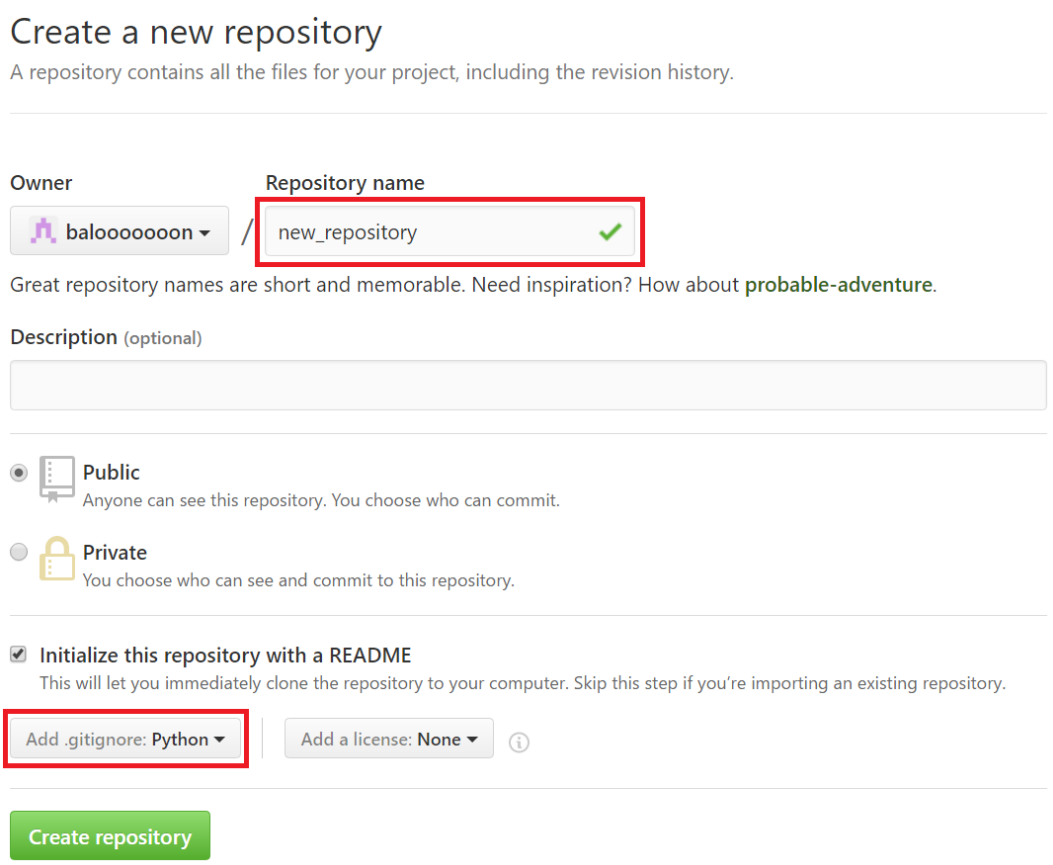
2 Procesy

2.1 Vytvorenie repozitára

Nový repozitár je možné vytvoriť vo webovom rozhraní serveru GitHub. Názov repozitára by mal jedno(dvoj)slovne vystihovať danú logickú časť nášho systému. Pri vytváraní je vhodné ihneď vytvoriť súbor README a .gitignore, ktorý vynechá súbory, ktoré je nepotrebné alebo nežiadúce do repozitára ukladať.



Obrázok 1 GitHub.com - Vytvorenie nového repozitára



Obrázok 2 - GitHub.com - formulár pre vytvorenie repozitára

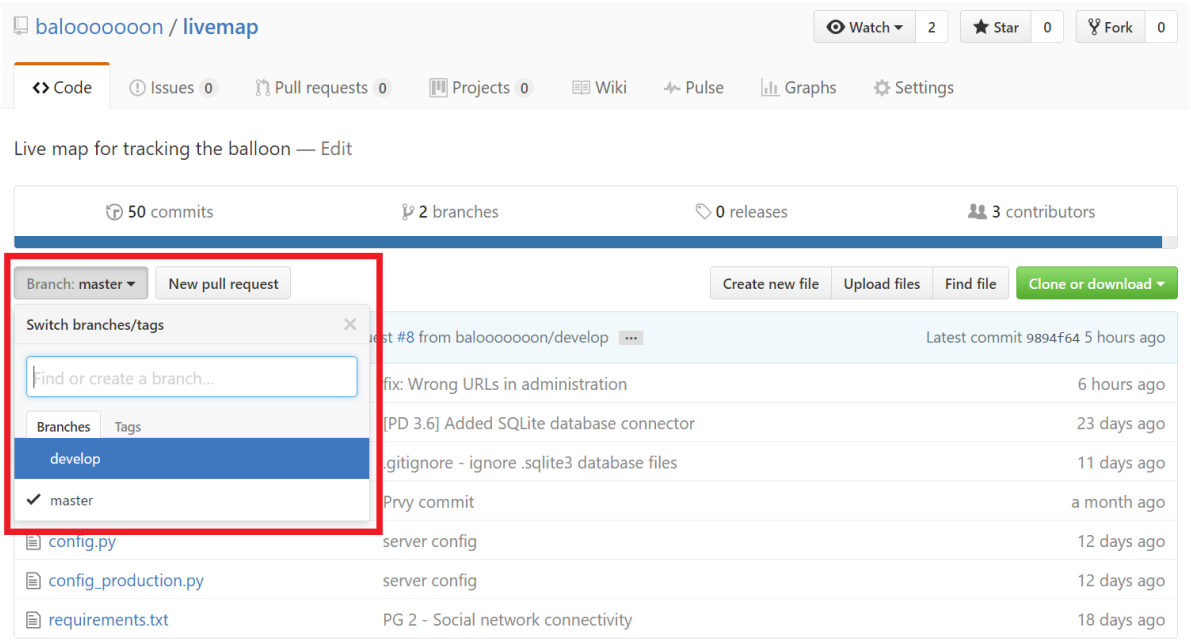
2.2 Vytvorenie vetvy

Každý repozitár obsahuje minimálne dve základné vetvy.

Master – Táto vetva obsahuje otestovaný a funkčný kód, ktorý môže byť nasadený do prototypu. Do tejto vetvy nie je možné priamo commitovať, je nutné použiť tzv. Pull Request. Pull request v tejto vetve musí prejsť cez Code Review aspoň jedným ďalším členom tímu.

Develop – V tejto vetve je zdrojový kód vo fáze vývoja. Zdrojový kód v tejto vetve by mal byť funkčný a spustiteľný. Autor commitu je zodpovedný za zdrojový kód a jeho otestovanie pred vykonaním commitu.

V rámci vzdialeného repozitára nie je nutné vytvárať novú vetvu pre každú úlohu/feature, ktorú člen tímu implementuje. Pokiaľ na menšej úlohe pracuje člen tímu sám, môže tieto vetvy používať v lokálnom repozitári.

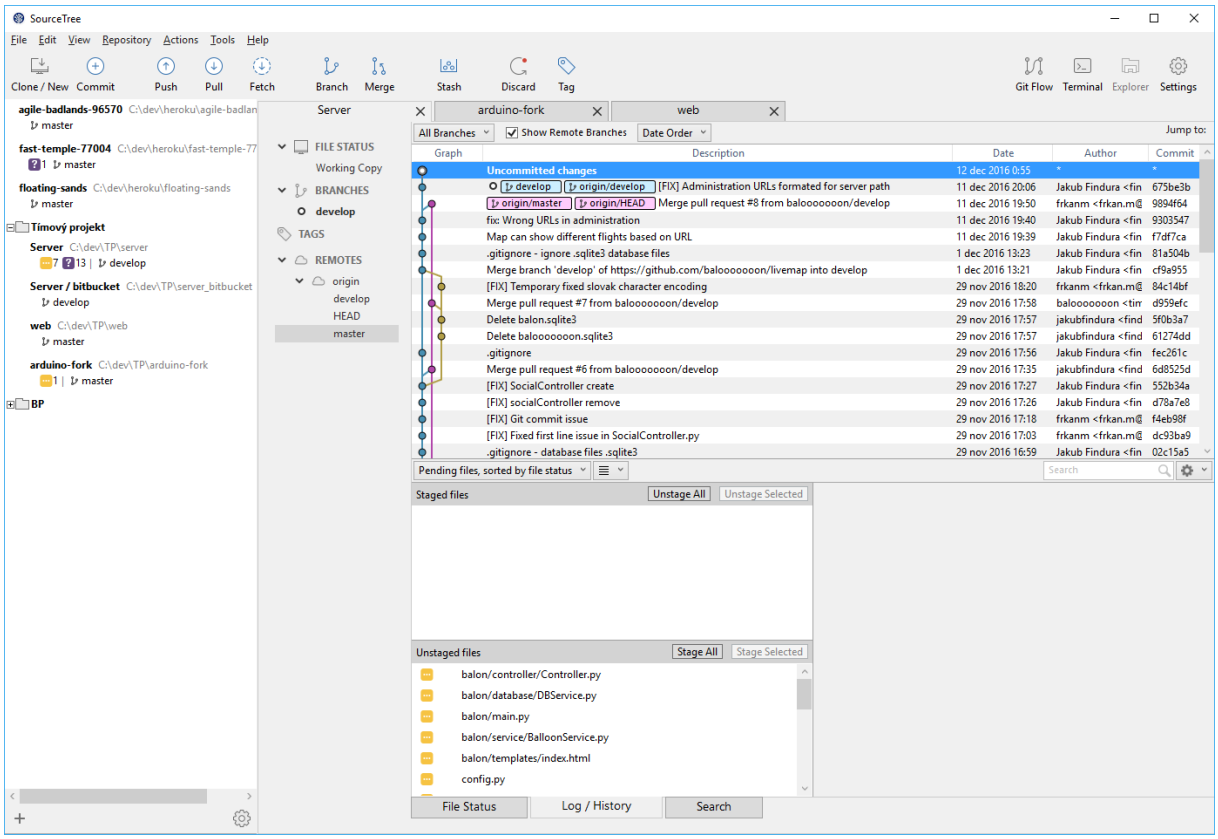


Obrázok 3 GitHub.com - vytvorenie novej vetvy

2.3 Vytvorenie lokálneho repozitára

Pre správu verzií na lokálom počítači je nutné mať nainštalovaný klientský program pre správu git. Odporúčaný je program SourceTree. Tiež je možné použiť git priamo v konzole.

```
Git init
```

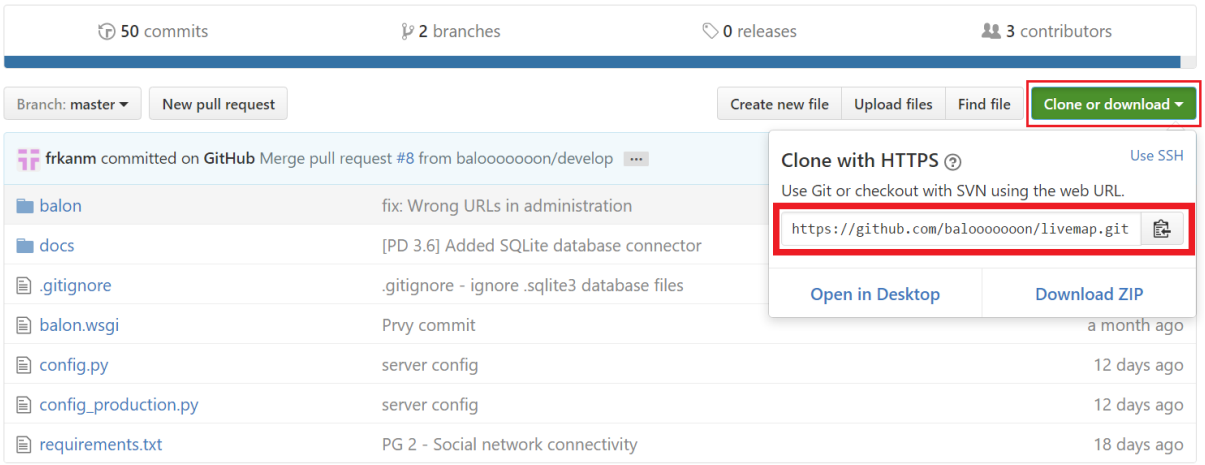


Obrázok 4 SourceTree - Program pre správu git

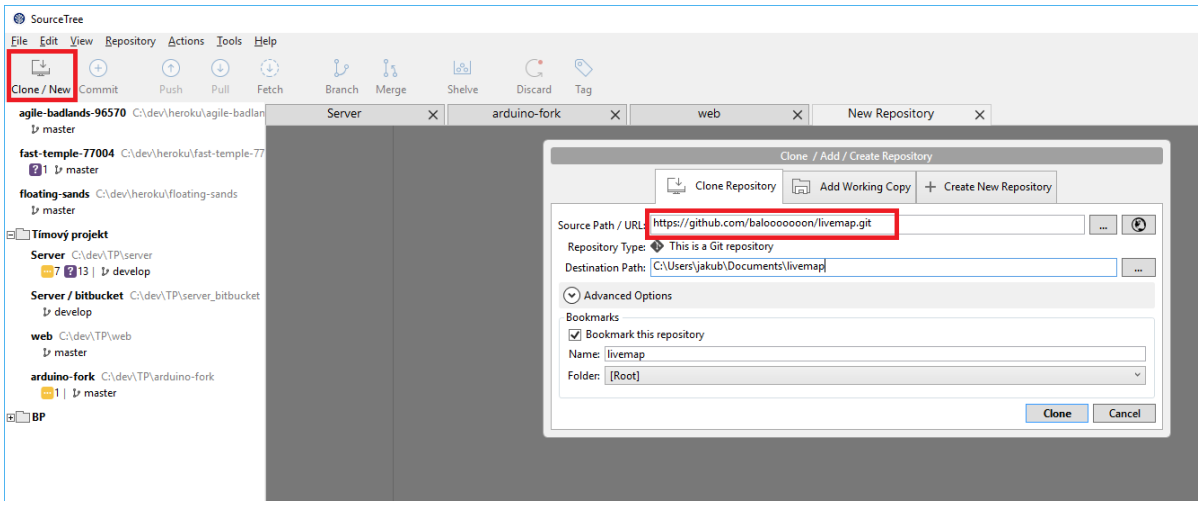
2.4 Prepojenie vzdialeného repozitára

Aby bolo možné zdrojový kód zdieľať so vzdialeným repozitárom, je nutné vytvoriť lokálny repozitár a pridať odkaz na vzdialený repozitár.

```
git remote add [name] [git_url]
git fetch [name]
git checkout develop
```



Obrázok 5 GitHub.com - odkaz na vzdialený repozitár

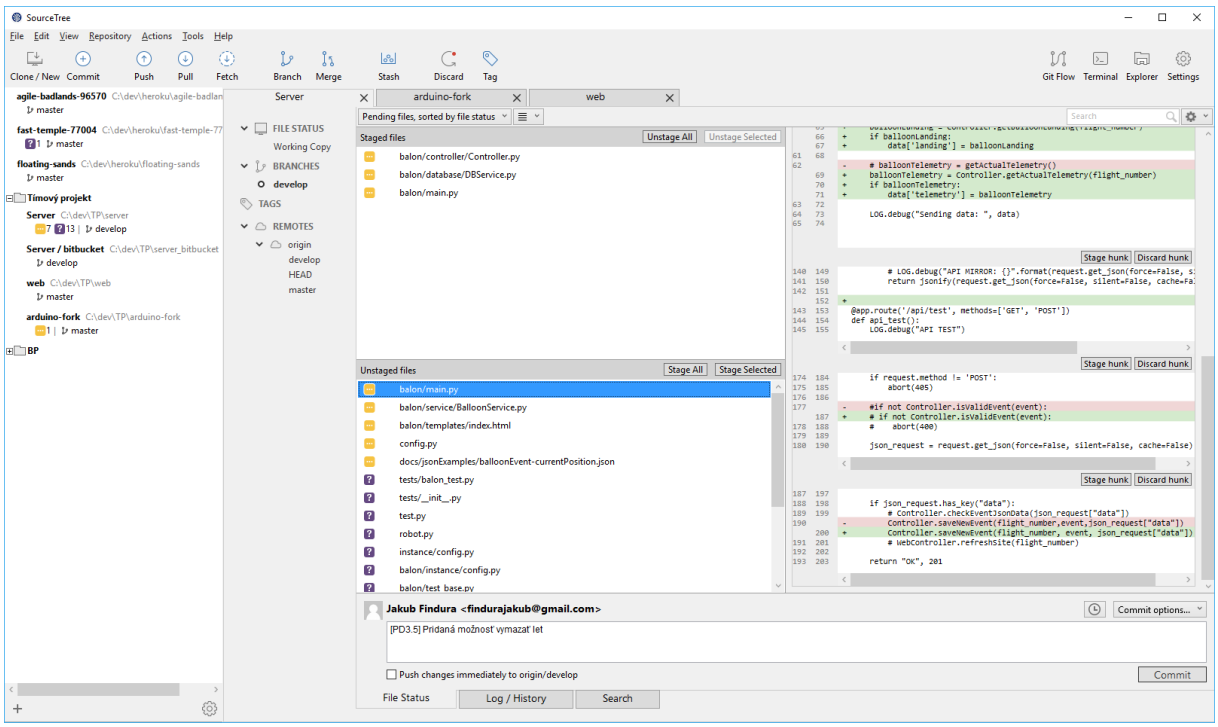


Obrázok 6 SourceTree - Pripojenie vzdialeného repozitára

2.5 Vytvorenie commitu

Prvým krokom je označenie zmenených súborov alebo častí súborov, ktoré majú byť súčasťou commitu. Každý commit obsahuje správu, ktorá popisuje, čo je predmetom daného commitu.

```
Git add .
Git commit -m „message“
```



Obrázok 7 SourceTree - Rozhranie pre vytvorenie nového commitu

2.6 Synchronizácia so vzdialeným repozitárom

Vytvorený commit sa nachádza len v lokálnom repozitári. Pre odoslanie všetkých lokálnych zmien je potrebné vykonať príkaz *push*. Pred vykonaní príkazu *push* je nutné vykonať príkaz *pull*, ktorý stiahne všetky vykonané zmeny vo vzdialenom repozitári ostatných členov tímu. Pokiaľ by nastal konflikt v prípade, že dvaja členovia urobili zmeny v rovnakej časti kódu, je nutné ho vyriešiť. Vhodná je externá aplikácia DiffMerge od SourceGear.

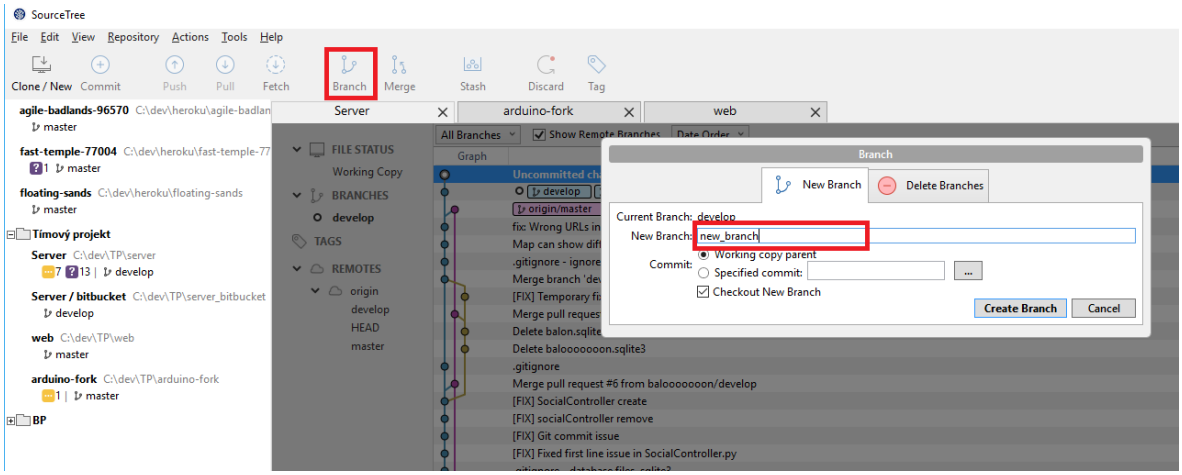
```
Git pull
Git push
```

2.7 Vytvorenie novej vetvy

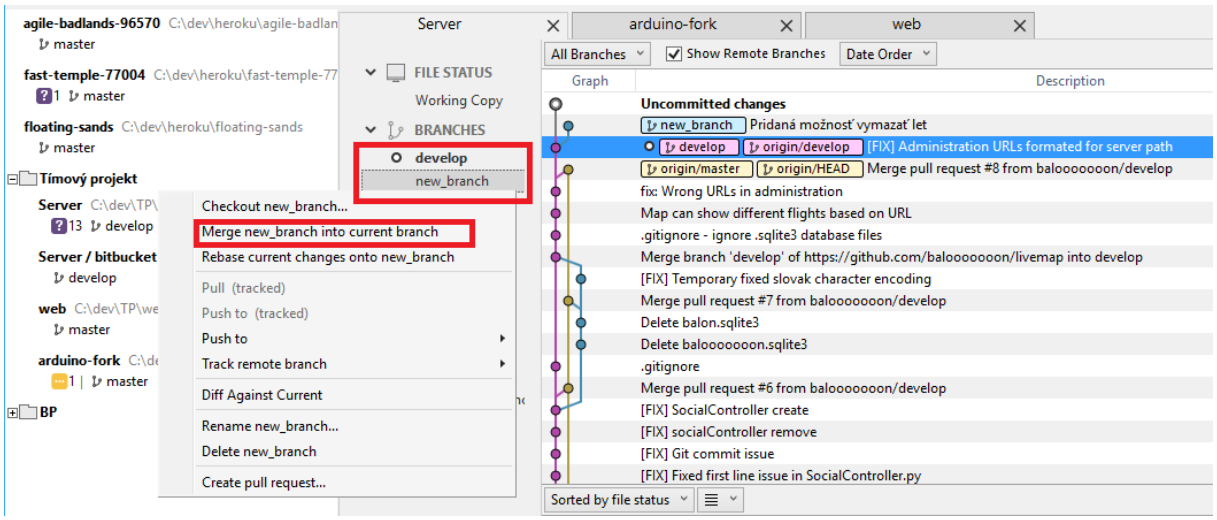
Pokiaľ je vyvíjaná nová funkcionalita, ktorá výrazne zasahuje do už existujúceho kódu, prípadne je predpoklad, že nakoniec bude zahodená, je vhodné vytvoriť novú vetvu. Pre správu vetiev existujú dva hlavné príkazy

Branch – vytvorenie novej vetvy

Merge – spojenie dvoch vetiev



Obrázok 8 SourceTree - vytvorenie novej vetvy



Obrázok 9 SourceTree - spojenie vetiev (merge)

3 Pravidlá písania

3.1 Názvy vetiev

Názvy vetiev by mali jasne vystihovať, čo je ich primárnym zameraním. Názov vetvy by sa mal skladať z prefixu a jedného, prípadne dvoch vystihujúcich slov.

Prefixy:

`feature_` pre samostatné časti, ktoré sa neviažu na konkrétnu priradené úlohu.

`pd-1-6_` pre funkcie, ktoré sa týkajú konkrétnej úlohy pridelenej členovi tímu alebo celého user story. Prefix vychádza zo značenia úloh, ktoré je opísané v Metodike riadenia.

Príklad:

`feature_database`

`db-1_lokalizacia`

`pg-3-6_mapa`

3.2 Commit message

Správa pri commite by mala krátkym textom vystihovať zmeny, ktoré boli v commite vykonané. Správa môže obsahovať prefix, ktorý obsahuje typ commitu (napríklad fix) alebo číslo úlohy, ak je viazaný priamo na úlohu.

Príklad:

[FIX] Opravené cesty k statickým súborom

[PD3.6] Zobrazovanie trasy letu na mape