STU FIIT
SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Metodika verziovania kódu
DUtí(m)

Tím číslo: 18
Členovia tímu: Bende Tomáš, Búcsiová Veronika, Horniczká Veronika, Melicherčík Jozef, Pacher Marek, Pavkovček Filip, Slíž Boris, Sojka Michal
Akademický rok: 2018/2019
Dátum vypracovania: 17.11.2018
Dátum poslednej úpravy: -
Vypracoval: Jozef Melicherčík

1. Úvod

Metodika verziovania kódu opisuje spôsob práce s verziovacím nástrojom Bitbucket a definuje základné štandardy pre verziovanie kódu. Je určená všetkým členom tímu, ktorí v rámci plnenia svojich úloh implementujú nejakú časť funkcionality v tímovom projekte a následne ju pripájajú k zdrojovému kódu projektu.

Nadväzujúce a súvisiace metodiky a dokumenty:

- Inštalačná príručka
- Metodika pre manažment úloh
- Metodika pre code review

2. Postup práce s verziovacím nástrojom Bitbucket

V tejto sekcii je popísaný postup práce s nástrojom Bitbucket za účelom pripojenia novej funkcionality projektu k celému zdrojovému kódu projektu. Pri práci s verziovacím nástrojom nie je presne stanovený spôsob komunikácie so serverom. To znamená, že je na vlastnom uvážení člena tímu, či použije príkazový riadok, alebo grafické používateľské rozhranie.

Postup práce pre verziovanie vypracovanej implementačnej úlohy cez príkazový riadok:

1. Aktualizácia projektu

Základné príkazy:

- ***cd „dir_nazov“*** - presunúť sa do priečinku s projektom
- ***git checkout develop*** - prepnúť sa do vetvy develop
- ***git pull*** - stiahnúť všetky zmeny zo servera na lokálne prostredie
- ***git fetch -p origin*** - update lokálnych vetiev po príkaze pull

Pred tým ako člen tímu začne pracovať na implementácií novej funkcionality projektu, je nutné zabezpečiť niekoľko základných úkonov. V prvom rade je potrebné nastaviť príkazový riadok do priečinku s projektom a prepnúť sa do vetvy *„develop“*. Následne je potrebné vykonať príkaz na stiahnutie aktualizácií zo server a updatnuť lokálne vetvy.

2. Vytvorenie novej vetvy

Základné príkazy:

- ***git checkout -b ‘nazov_novej_vetvy‘*** - vytvorenie novej vetvy na develope

Po vykonaní základných úkonov a implementácií zmien/vývoji je možné vykonať nasledovné kroky k nahratiu lokálnej verzie na server. Každý samostatný commit by mal v sebe zahŕňať ucelený kus funkcionality, nie nutne dokončený a funkčný kus. Podľa interných štandardov tímu sa pre každú implementačnú úlohu vytvára nová vetva. Z tohto dôvodu je prvým krokom vytvorenie novej vetvy. Pomenovávanie vetiev sa riadi nasledovnými pravidlami:

- maximálne trojslovný názov vetvy v slovenskom jazyku
- slová oddelené podtržníkom
- slová vystihujúce charakter zmien (napr. čo sa stalo? úprava, v čom nastala zmena? ciest, kde nastala? AktivityView -> uprava_ciest_AktivityView)

3. Zobrazenie zmien

Základné príkazy:

- ***git status*** - kontrola zmien na lokálnom prostredí
- ***git diff*** - rozdiel v zmenených riadkoch

Pred pridaním jednotlivých zmien z pracovného prostredia do tzv. staging area je možné vidieť informácie o zmenách pomocou vyššie uvedených príkazov.

4. Označenie zmien

Základné príkazy:

- ***git add .*** - označenie zmien, ktoré sa pridajú k nasledujúcemu commitu
- ***git reset „cesta“*** - vylúčenie zmien z commitu
- ***git checkout -- „cesta“*** - trvalé odstránenie neoznačených zmien

Vyššie uvedený príkaz pridá všetky zmeny do nasledujúceho commit. V prípade, že nie je potrebné pridať všetky upravené súbory do commitu, je možné súbory explicitne vybrať pomocou príkazu *git add „cesta“*. Pre výber konkrétnych riadkov súbory slúži príkaz *git add –p „cesta“*.

V základnom priečinku projektu sa nachádza aj súbor s názvom *.gitignore*. V tomto súbore sa nachádza zoznam priečinkov a súborov, ktoré git nemá pridávať k zmenám. To znamená, že je prostredníctvom tohto súboru možné trvale vylúčiť pridávanie ľubovoľného priečinku alebo súboru k zmenám.

5. Vytvorenie commitu

Základné príkazy:

- ***git commit -m „informacna_sprava“*** – zverejnenie zmien zo staging area do tzv lokálneho repozitára

Pridanie všetkých zmien do staging area umožňuje zverejnenie týchto zmien na server. Ku každému commitu je nutné pridať aj správu. Pri vytváraní správy je nutné sa riadiť nasledovnými pravidlami:

- správa sa nevetví do odstavcov ani odkazov
- správa má charakter oznamovacej vety v slovenskom jazyku
- správa zahŕňa stručný a jasný opis zmien (napr. Pridaný základný layout pre informatívne testovanie)

6. Aktualizovanie stavu vlastnej vetvy

Základné príkazy:

- ***git merge origin/nazov_vetvy*** – aktualizácia stavu vlastnej vetvy o nové zmeny zo servera (nazov_vetvy – vetva, z ktorej sa ťahajú zmeny na vlastnú vetvu)

V prípade, ak sú na dokončenie implementačnej úlohy potrebné zmeny zverejnené v inej vetve, je potrebné zavolať tento príkaz z vlastnej vetvy.

7. Zverejnenie commitu

Základné príkazy:

- ***git push -u origin „nazov_novovytvorenej_vetvy“*** – presunie zmien z lokálneho repozitára na server

Zdieľanie zmien na server je po ich pridaní do lokálneho repozitára v predošlom kroku možno zadaním príkazu *push.* V tomto prípade však môže nastať konflikt, keď medzičasom iný člen tímu zverejnil na servery zmeny v rovnakých súboroch (vznikne konflikt). V takom prípade je nutné pomocou príkazu *pull* (krok 1.) stiahnuť zmeny zo servera za účelom ich napojenia na lokálne zmeny a vyriešiť prípadné konflikty. Následne je možné pokračovať zadaní príkazu *push.*

8. **Vytvorenie pull request**

Základné príkazy:

- ***podržať tlačidlo ctrl a kliknúť na odkaz v príkazovom riadku, ktorý sa objaví po push príkaze*** – vytvorenie pull reguestu

Po nahratí zmien na server je nutné vytvoriť *pull request* pre kontrolu zmien (funkčnosť a konvecie) iným členom tímu. Bližšie je tento krok opísaný v *Metodike pre code review*.

9. **Aktualizovanie hlavnej vývojovej vetvy**

Základné príkazy:

- schválenie pull request
- ***git merge origin/nazov_vetvy*** – pridanie commitu do vetvy develop a uzatvorenie aktuálnej vetvy

Po schválení *pull request* je možné pridať *commit* do vetvy *develop* a uzavrieť aktuálnu vetvu. Tento krok vykonáva výhradne *Git master*, ktorý je zodpovedný za spravovanie verziovacieho nástroja. Bližšie je tento krok opísaný v *Metodike pre code review*.

3. Štruktúra vetiev

Hlavnou vetvou repozitára je *master,* ktorý reprezentuje aktuálny stav celého projektu s dokončenými súčasťami funkcionality. Táto vetva sa vždy nahráva na produkčný server, kde sa udržuje aktuálna funkčná časť projektu. Vedľajšou, vývojovou, vetvou je vetva *develop,* na ktorej prebiehajú všetky zmeny a vývoj novej funkcionality.

Za spravovanie *master* vetvy a mergeovanie *develop* vetvy je zodpovedný *Git master.* Jeho úlohou je zabezpečenie funkčnosti projektových zmien a minimalizácia možných problémov v súvislosti s mergeovaním týchto zmien. Vývoj prebieha vo vetve *develop,* kde sa pre každú implementačnú úlohu vytvárajú, podľa interných konvencií tímu, vždy nové vetvy. Pre každú implementačnú úlohu sa vytvorí jedna vetva, ktorá je po schválení *pull requestu* uzatvorená a mergenutá k vetve *develop*.

4. Obsah a štruktúra správ v pull request

Postup práce s verziovacím nástrojom Bitbucket opisuje v 7. kroku vytvorenie *pull request*. Pri vytváraní *pull request* je nutné vyplniť nasledovné polia v online nástroji Bitbucket.

Title: predvyplní sa z *commit* správy (nemeniť)

Description: formát správy sa neriadi presnými konveciami, je teda na autorovi, aký formát zvolí. Opis zmien by mal byť však dostatočne výstižný a presný na to, aby ktorýkoľvek člen tímu dokázal pochopiť, aké zmeny daný *commit* v sebe obsahuje.

Reviewers: vybrať jedného člena tímu, ktorý vykoná review (bližšie je tento proces opísaný v *Metodike pre code review*.)

Close branch: zaškrtnúť pole pre uzatvorenie vetvy po schválení *pull request*