Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Metodika verziovania kódu

tím XYZ

**Vedúci projektu:** Ing. Lukáš Kohútka, PhD.

**Študijný program:** Inteligentné softvérové systémy

**Akad. rok:** 2020/2021

**Dátum poslednej zmeny:** 10.10.2020

**Vypracoval:** Bc. Adam Chmara

# Obsah

[**Úvod**](#_yneyaxx3o4oa) **3**

[**Vetvy**](#_p8gvh48umxcv) **3**

[**Commit**](#_gxg4s81i6z41) **4**

[**Pull Request**](#_ed1cgce3cqns) **4**

[**Workflow**](#_in7dncmwx3yn) **5**

# 

# Úvod

Cieľom metodiky verziovania kódu je jasne definovať zakladné pravidlá a najlepšie praktiky verziovania repozitárov projektu. Takto definované pravidlá sprehľadnia a zjednodušia manažment verzií kódu a predídu možným komplikáciám.

Nástroje použité na verziovanie sú **Git a Github**. V rámci celého verziovania bude použitý anglický jazyk.

# Vetvy

Každý repozitár projektu obsahuje permanentne minimálne nasledujúce vetvy:

* **master** - vetva reflektujúca ideálne otestovaný production-ready kód, do *master* vetvy sa môžu mergovat iba pull requesty z *dev* vetvy a po schválení vedúcim projektu
* **dev** - vetva reflektujúca najaktuálnejší stav, do dev vetvy sa mergujú pull requesty z *feature* alebo *fix* vetiev

Výnimkou sú repozitáre asicde-docker, asicde-router, website a documentation. Tieto repozitáre nepotrebujú osobitné vetvy na vývoj, pretože nie sú komplexné, nemenia sa často, alebo neudržiavajú zdrojové kódy.

Ďalšie vetvy pre feature, fix a hotfix:

* **feature/<sprintNum>-<story/task-id>/<description>**
  + vetva vytvorená výhradne z *dev* vetvy
  + určená pre implementáciu nových features
* **fix/<sprintNum>-<task-id>/<description>**
  + vetva vytvorená výhradne z *dev* vetvy
  + určená pre bug fixy existujúce na *dev* vetve
* **hotfix/<sprintNum>-<task-id>/<description>**
  + vetva vytvorená výhradne z *master* branche
  + určená pre rýchle odstránenie bugov na *master* vetve

Pre každú feature/fix/hotfix vetvu musí byť v systéme Jira vytvorená issue alebo task s prislúchajúcim ID.

**Príklad:**

feature/2-47/syntax-highlighting

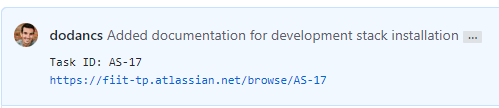
# Commit

Commit by mal spĺňať nasledujúce pravidlá:

* Musí obsahovať podpis autora
* Nadpis správy by mal byť ideálne krátky a výstižný
* Nadpis by mal by začínať veľkým písmenom a slovesami v minulom case (napr. Added, Removed, Fixed, Edited, Changed …)
* Telo správy by malo obsahovať minimálne ID issue/tasku, prípadne aj link na issue/task z Jiry vo formáte:
  + Task ID: XY-123
  + https://fiit-tp.atlassian.net/browse/AS-17

**Príklad:**

git commit -s -m “Added configuration file to auth module” -m “Task ID: <task ID>” -m “https://fiit-tp.atlassian.net/browse/XY-123”



# Pull Request

Pull request sa otvára v prípade ak je vetva pripravená na mergnutie. Po otvorení pull requestu je potrebné vyplniť predpripravenú šablónu na GitHube s požadovanými údajmi a priradiť minimálne jedného člena tímu na review.

Formát šablóny pre pull request:

* **Summary** - zhrnutie a popis vykonaných zmien v danej vetve
* **Issue** - ID tasku alebo story z Trella
* **Test Plan** - krátky opis ako bola daná vetva testovaná
* **Screenshots** - ak sa jedná o UI zmeny tak treba priložiť screenshot rozhrania kde je danú zmenu vidno

**Príklad:**

**Summary**

Hardcoded values in configuration files were fixed to correct values due to multiple errors in the auth module.

**Issue**

TL-134

**Test Plan**

Manually clicking through the UI to check if errors persist.

**Screenshots**

-

# Workflow

Nasledujúce kroky opisujú základný postup pri implementácií novej feature:

1. Vytvorenie (ak neexistuje) tasku alebo story v systéme Trello
   1. Priradenie sa k vytvorenému tasku
2. Dotiahnutie najnovších zmien v dev vetve
3. Vytvorenie a pomenovanie novej vetvy z dev vetvy podľa stanovených konvencií
4. Vývoj vetvy - pridávanie commitov podľa stanovených konvencií
5. Vytvorenie pull requestu z feature vetvy do dev vetvy
6. Po odsúhlasení zmien (review) - merge do dev vetvy