Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Metodika kontroly a testovania kódu

tím XYZ

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

**Študijný program:** Inteligentné softvérové systémy

**Akad. rok:** 2020/2021

**Dátum poslednej zmeny:** 13.10.2020

**Vypracoval:** Bc. Dominik Dancs

# Obsah

[**Obsah**](#_nb1q339vciy5) **1**

[**Úvod**](#_yneyaxx3o4oa) **2**

[**Metodika kontroly kódu**](#_p8gvh48umxcv) **2**

[Backend](#_s26nqyczo1kz) 2

[Frontend](#_itjg2p2cn) 2

[**Metodika testovania kódu**](#_3a5hmwfveozl) **3**

[Manuálne testovanie](#_79pkkh7bw63w) 3

[Automatické testovanie](#_ar3mupalxnlp) 3

# 

# Úvod

Cieľom metodiky testovania a kontroly kódu je definovanie základných procesov vykonávaných počas vývoja a po ukončení vývoja určitého modulu projektu. Kód je potrebné revidovať a taktiež testovať aby sa zaručila správnosť novej pridanej funkcionality a taktiež nepoškodila funkčnosť ostatných modulov projektu.

# Metodika kontroly kódu

V tejto sekcii popíšeme nami používané techniky na kontrolu správnosti kódu počas vývoja.

## Backend

Framework Spring používaný na backende projektu poskytuje vývojovému prostrediu IntelliJ IDEA možnosti automatickej kontroly kódu počas písania ako aj pri kompilácii. Automatickou kontrolou sa zaručí správnosť kódu z technickej stránky. Ďalším rozšírením tohto vývojového prostredia, ktoré vykoníva statickú kontrolu kódu je Lombok, ktorý overuje syntax, správnosť dátových typov a prepojenia dátových modelov - toto rozšírenie nám umožňuje generovať tzv. “boilerplate” kód pre každú triedu, a tým sa vyhneme manuálnym chybám pri písaní kódu. Taktiež naše POJO triedy obsahujú oveľa menej riadkov kódu, ako keby sme toto rozšírenie nepoužívali.

Na overenie logickej stránky funkčnosti kódu sú využívané unit testy, ktoré overujú správnosť logiky priamo na výslednej časti implementácie. Počas implementácie nových funkcionalít vytvárame zároveň aj takéto testy, ktoré nám pomáhajú odhaliť nedostatky.

Príkazom mvn test je možné spustiť automatické testy na požiadanie, ktoré sú spúšťané aj pri každej kompilácii kódu. V prípade výskytu problému je takýto kód opravený pred tým, ako je vytvorený pull request.

Okrem automatických kontrol je každý člen tímu zodpovedný za manuálnu kontrolu svojho kódu - lokálne ladenie kódu pomocou funkcií IDE ako aj externých nástrojov, napr.: Postman a pgAdmin. Pri vytvorení pull requestu je tento kód overený ešte minimálne jedným ďalším členom tímu.

## Frontend

Táto časť projektu je kontrolovaná vstavanými funkcionalitami vývojového prostredia Visual Studio Code ako aj ďalšími dostupnými rozšíreniami - Angular Language Service, ESLint, alebo TSLint (statická kontrola).

Rovnako ako pri backende, je aj tento kód ručne kontrolovaný a schválený v dvoch fázach - manuálne autorom funkcionality a ďalším, minimálne jedným, členom tímu.

Angular-cli taktiež poskytuje funkcionality kontroly syntaxe a bezpečnosti kódu pri jeho kompilácií alebo pri manuálnom vyžiadaní kontroly.

# Metodika testovania kódu

Po úspešnej implementácii novej funkcionality a overení napísaného kódu je kód v obidvoch častiach projektu testovaný. Testy sú rozdelené do dvoch kategórií: manuálne a automatické.

## Manuálne testovanie

Každý člen tímu má povinnosť svoj kód poriadne otestovať po akejkoľvek zmene, ktorá bude pridaná do vývojovej vetvy v rámci platformy verziovania kódu. Testovanie prebieha v troch krokoch:

1. **Vytvorenie testovacích scenárov** - pred testovaním si daný člen tímu vytvorí scenáre, ktoré budú overovať správnosť novo-pridanej funkcionality. Tieto scenáre by mali zahŕňať správne aj nesprávne vstupy a dostatočne široké úlohy, ktoré pokryjú celú zmenenú/doplnenú funkcionalitu.
2. **Testovanie aplikácie** - člen tímu následne vykoná testovanie podľa scenárov, ak je to potrebné, scenáre upraví/doplní.
3. **Vyhodnotenie výsledkov / oprava implementácie** - po ukončení testovania vyhodnotí výsledky a následne podľa vyhodnotenia určí, či je daná funkcionalita pripravená na zaradenie do vývojovej vetvy. V prípade nájdenia chýb, sú tieto chyby opravené pred vytvorením pull requestu, a testovanie je opätovne uskutočnené.

## Automatické testovanie

Na automatické testovanie sú v projekte vytvárané testovacie scenáre. Pre backend sú písané vo frameworkoch JUnit a Mockito a pre frontend sú to štandardné E2E testy.

Pri implementácii nových funkcionalít sú súbežne vytvárané aj tieto automatizované testovacie skripty, aby sme zaručili vysokú kvalitu kódu a každá funkcionalita, ako aj projekt ako celok, mohla byť čo najskôr automaticky testovaná.

Tieto testy sú využívané pri manuálnej kompilácii kódu lokálne, ale aj pri vytváraní výstupných nasaditeľných modulov v rámci platformy Jenkins (pomocou pipelines a build tasks). Pri každej úprave kódu vo vývojovej a produkčnej vetve je automaticky spustená kompilácia a testovanie kódu na serveri. Výstup kompilácie a testov je zachytený a v prípade chyby je ihneď odoslané oznámenie stavu na náš komunikačný nástroj Slack. Po zachytení chyby jeden alebo viacerí členovia tímu overia výstupy a problémy odstránia. V prípade bezproblémovej kompilácie a testovania je aplikácia automaticky nasadená.