

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Projektová dokumentácia - riadenie

Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller-04@googlegroups.com
Akademický rok: 2016/2017

Obsah

1	Úvod.....	1-1
2	Roly členov tímu a podiel práce.....	2-1
2.1	Roly členov tímu	2-1
2.2	Pridelené zodpovednosti členov tímu.....	2-1
2.3	Podiel práce	2-2
3	Aplikácie manažmentov	3-1
3.1	Manažment dokumentácie.....	3-1
3.2	Manažment komunikácie.....	3-1
3.3	Manažment plánovania.....	3-1
3.4	Manažment verziovania.....	3-1
3.5	Manažment testovania	3-2
3.6	Manažment kvality písania zdrojového kódu	3-2
4	Sumarizácie šprintov	4-1
4.1	Šprint 1 – Baldr.....	4-1
4.2	Šprint 2 – Fenrir.....	4-3
4.3	Šprint 3 – Freya	4-6
4.4	Šprint 4 – Heimdallr	4-9
4.5	Šprint 5 – Huginn	4-10
5	Používané metodiky	5-1
5.1	Metodika plánovania	5-1
5.2	Metodika konvencií písania zdrojového kódu – Coding Standards	5-1
5.3	Metodika testovania zdrojového kódu.....	5-1
5.4	Metodika prehliadok zdrojového kódu – Code review	5-2
5.5	Metodika verziovania zdrojového kódu	5-2
5.6	Metodika písania dokumentácie	5-2
6	Globálna retrospektíva	6-1

1 Úvod

Tento dokument predstavuje dokumentáciu riadenia projektu, ktorý je vytváraný v rámci predmetu Tímový projekt v akademickom roku 2016/2017. Dokument zahŕňa roly členov tímu, podiel ich práce na dokumentácii projektu, aplikácie manažmentov, sumarizácie šprintov, zoznam používaných metodík a globálnu retrospektívu pre jednotlivé semestre.

Názov témy nášho projektu je *Story Teller – Zber a vyhodnocovanie požiadaviek* a jeho cieľom je vytvoriť informačný systém, ktorý umožní pohodlné zadávanie a sledovanie splnenia používateľských scenárov (funkcionálnych požiadaviek). Zber požiadaviek bude realizovaný prostredníctvom vizuálneho skicovania obrazoviek scenáru a ich komentovania, na základe čoho môžu byť vygenerované akceptačné testy. Ďalšou funkcionalitou systému bude vykonávanie akceptačných testov a komentovanie ich výsledkov.

Prvá kapitola dokumentu obsahuje prehľad manažérskych rolí každého člena tímu a prehľad podielu práce členov tímu na jednotlivých kapitolách dokumentácie riadenia projektu a dokumentácie inžinierskeho diela.

V druhej kapitole dokumentu je bližšie popísaná aplikácia jednotlivých manažmentov v projekte, čo zahŕňa opis jednotlivých činností potrebných pre riadenie projektu.

Tretia kapitola dokumentu slúži na sumarizáciu jednotlivých šprintov. Pre každý šprint je uvedený burndown graf a zoznam činností, ktoré sa nám podarilo, resp. nepodarilo spraviť.

Štvrtá kapitola dokumentu obsahuje zoznam a stručný opis metodík, ktorými sme sa riadili pri práci na projekte. Pri každej metodike je uvedená referencia na dokument, v ktorej je spísaná.

V piatej kapitole dokumentu je uvedená globálna retrospektíva pre zimný semester. Táto časť zahŕňa zoznam činností, ktoré boli vyhodnotené za pozitívne, zoznam činností, ktorým by sme sa mali v budúcnosti vyhnúť, a zoznam činností, ktoré by sme mali vykonávať v ďalších šprintoch.

2 Roly členov tímu a podiel práce

Táto kapitola opisuje manažérske roly členov tímu a podiel práce členov tímu na jednotlivých kapitolách dokumentácie riadenia a dokumentácie inžinierskeho diela.

2.1 Roly členov tímu

Každý člen tímu má pridelenú aspoň 1 manažérsku rolu, za ktorú je zodpovedný. V tabuľke č. 1 sa nachádza prehľad manažérskych rolí každého člena tímu, ktoré si vybral pri ich pridelovaní.

Tabuľka 1: Roly členov tímu

Zodpovedný člen tímu	Manažérska rola
Bc. Jakub Ondik	Manažér vývoja a kvality
Bc. Patrik Januška	Manažér testovania
Bc. Adam Neupauer	Manažér integrácie a nasadzovania
Bc. Martin Olejár	Manažér dokumentácie a plánovania
Bc. Miroslav Hurajt	Manažér komunikácie a verziovania
Bc. Ondrej Hamara	Manažér rizík

2.2 Pridelené zodpovednosti členov tímu

Bc. Jakub Ondik

- rozhodovanie o smerovaní tímového projektu
- definovanie krokov pre vývoj projektu
- sledovanie pokroku vývoja pri riešení projektu
- integrácia pomocných systémov
- infraštruktúra
- zodpovednosť za vykonávanie prehliadok zdrojového kódu
- dohľad nad kvalitou systému

Bc. Patrik Januška

- sledovanie písania testov a testovania
- zodpovednosť za vykonávanie prehliadok zdrojového kódu

Bc. Adam Neupauer

- vytvorenie, správa a štylovanie tímovej webovej stránky
- správa tímového servera
- kontinuálna integrácia

Bc. Martin Olejár

- aktualizácia dokumentov na webovej stránke
- vytvorenie šablóny pre písanie dokumentácie a jej údržba

- zodpovednosť za zadeľovanie úloh a sumarizovanie šprintov
- sledovanie termínov
- dohľad nad plnením zadaných úloh

Bc. Miroslav Hurajt

- zodpovednosť za verziovanie zdrojového kódu
- dohľad nad prácou s vetvami
- sledovanie tímovej komunikácie

Bc. Ondrej Hamara

- zaznamenávanie a vyhodnotenie rizík počas práce na tímovom projekte

2.3 Podiel práce

V tabuľke č. 2 sa nachádza prehľad podielu práce členov tímu na jednotlivých kapitolách dokumentácie riadenia projektu a v tabuľke č. 3 sa nachádza prehľad podielu práce členov tímu na jednotlivých kapitolách dokumentácie inžinierskeho diela.

Tabuľka 2: Podiel práce na dokumentácii riadenia projektu

Názov kapitoly	Vypracoval
Úvod	Bc. Martin Olejár
Roly členov tímu a podiel práce	Bc. Martin Olejár
Aplikácie manažmentov	Bc. Miroslav Hurajt
Sumarizácie šprintov	Bc. Jakub Ondik
Používané metodiky	Bc. Patrik Januška
Globálna retrospektíva ZS	Bc. Martin Olejár

Tabuľka 3: Podiel práce na dokumentácii inžinierskeho diela

Názov kapitoly	Vypracoval
Úvod	Bc. Martin Olejár
Globálne ciele pre ZS	Bc. Jakub Ondik
Celkový pohľad na systém	Bc. Jakub Ondik

3 Aplikácie manažmentov

Každý člen nášho tímu zodpovedá za určitý manažment, ktorý si sám vybral z pomedzi jednotlivých tímových rolí.

3.1 Manažment dokumentácie

Dokumentácia k nášmu vyvíjanému projektu je písaná pravidelne v rámci jednotlivých šprintov. Po dokončení úloh, aby bola práca na používateľskom príbehu považovaná za ukončenú, musí obsahovať aj príslušnú dokumentáciu. Za kontrolu tejto dokumentácie zodpovedá pridelený manažér.

Taktiež každé tímové stretnutie vzniká zápisnica, ktorú píše každý týždeň iný člen tímu. Táto zápisnica obsahuje opísaný priebeh stretnutia a taktiež zoznam úloh, ktoré je potrebné spraviť. Na túto zápisnicu bola v úvode práce na tímovom projekte vytvorená šablóna, aby každá mala jednotný tvar. Zápisnica sa publikuje každý týždeň aj na tímovú stránku, aby bola dostupná každému členovi tímu.

Každú vytvorenú dokumentáciu kontroluje manažér dokumentácie a dáva jej výslednú podobu.

3.2 Manažment komunikácie

Komunikácia mimo tímové stretnutia prebieha prevažne cez komunikačný nástroj Slack, ktorý umožňuje pohodlné vyhľadávanie vo veľkom počte správ a organizáciu tém do samostatných kanálov. Cez tento komunikačný nástroj môžeme pohodlne kontaktovať všetkých členov tímu a taktiež diskutovať o vzniknutých problémoch, nápadoch a návrhoch.

Komunikácia na tímovom stretnutí je voľná. Ak má nejaký člen tímu nejaký návrh alebo nápad, tak ho predstaví ostatným členom a diskutuje sa o tom. Taktiež po konci šprintu sa na tímovom stretnutí vytvára retrospektíva šprintu, ktorú vedie scrum master (v našom tíme vedúci) a každý člen tímu sa vyjadrí, čo sa mu páčilo a nepáčilo, a následne sa spoločne hľadajú riešenia k týmto problémom.

3.3 Manažment plánovania

Na manažment naplánovaných úloh používame nástroj Team Foundational Server (TFS), v ktorom naplníme backlog používateľskými príbehmi, z ktorých si potom jednotliví členovia tímu vyberajú úlohy.

Práca v tíme je organizovaná do dvojtýždňových šprintov. Na začiatku každého šprintu sa vytvárajú úlohy, ktoré sa dokončia daný šprint. Pri vytváraní úloh vzniká diskusia, aby každému členovi tímu bolo jasné, čo sa ide implementovať. Následne scrum master organizuje scrum poker, aby sa úlohy ohodnotili story pointami a upresnilo zadanie jednotlivých úloh.

3.4 Manažment verziovania

Po dokončení ucelených funkcionalít sa tieto funkcionality spájajú do novej verzie projektu s novou funkcionalitou. Na manažment verziovania používame systém Git a prácu vo vetvách.

Každý člen tímu pracuje vo svojej vetve vytvorenej podľa používateľského príbehu. Po ukončení práce vytvorí pull request, kde sa jeho práca skontroluje z hľadiska kvality zdrojového kódu a požadovanej funkcionality. Pre správne fungovanie verziovania sme vytvorili metodiku verziovania, aby každý člen tímu mal definovaný správny postup ako pristupovať k verziovaniu.

3.5 Manažment testovania

Každý člen tímu zodpovedný za úlohu vytvára príslušne testy k danej úlohe. V rámci nášho projektu je testovanie zložené z 2 častí: testovaniu backendu, na ktoré sa vytvárajú unit testy a testovanie frontendu (AngularJS komponentov) simulovaním priamo cez rozhranie webového prehliadača. Ako pomôcku pre vytváranie týchto testov manažér testovania vytvoril prehľadnú metodiku testovania, v ktorej detailne popisuje ako vytvoriť tieto testy.

3.6 Manažment kvality písania zdrojového kódu

Každý člen tímu zodpovedá za vytvorenú úlohu, ktorú si vyberie a teda aj za zdrojový kód, ktorý napíše. Každý člen tímu by mal dodržiavať pravidlá písania zdrojového kódu, pre ktoré bola vytvorená aj metodika konvencií písania zdrojového kódu, kde sú uvedené všetky štandardy, ktoré dodržiavame, aby bol zdrojový kód kvalitný a prehľadný. Napísaný zdrojový kód sa po pull requeste kontroluje aj druhým členom tímu s cieľom nájsť nedostatky v kóde a upovedomiť ho prostredníctvom komentáru, aby tieto nedostatky opravil. Tento postup pri kontrolovaní zdrojového kódu a celkový postup ku kontrole práce iného člena tímu sú uvedené v metodike ku code review.

4 Sumarizácie šprintov

4.1 Šprint 1 – Baldr

Prvý šprint mal za úlohu oboznámenie s vybranými technológiami. Z tohto dôvodu boli do tohto šprintu vybrané menej náročné používateľské príbehy a im zodpovedajúce úlohy, ktoré môžeme vidieť na obrázkoch 1 a 2.

Title	Story Points	State	Assigned To
Registracia prostrednictvom Emailu z webovej stránky	2	Closed	Bc. Martin Olejar
Vytvorenie entity pouzivatela		Closed	Jakub Ondik
REST POST metoda na zaregistrovanie pouzivatela		Closed	Bc. Martin Olejar
Vytvorit formular pre registraciu		Closed	Bc. Martin Olejar
Dokumentacia a review pre entitu pouzivatela		Closed	Jakub Ondik
Dokumentacia a review formularu registracie a prisluhujuceho ...		Closed	Bc. Martin Olejar
Nastavenie zakladnych udajov o pouzivatelovi	2	Closed	Bc. Patrik Januska
Pridanie atributov do DB entity usera		Closed	Bc. Patrik Januska
REST GET na ziskanie aktualneho profilu		Closed	Bc. Adam Neupauer
Formular na zobrazenie profilu		Closed	Bc. Miroslav Hurajt
PUT na uplnu aktualizaciu udajov o pouzivatelovi		Closed	Bc. Patrik Januska
Vytvorenie Angular controllera pre profil pouzivatela		Closed	Jakub Ondik
Uprava vyberu lokalizacie a realny upload foto pouzivatela		Closed	Jakub Ondik
Dokumentacia		Closed	Jakub Ondik
Definovanie zakladov prace s git	2	Closed	Bc. Martin Olejar
Definovanie zakladov (pull, commit)		Closed	Bc. Martin Olejar
Definovanie prace s vetvami		Closed	Bc. Martin Olejar
Prihlasenie na Web stránke	5	Closed	Jakub Ondik
Prihlasovaci formular		Closed	Bc. Miroslav Hurajt
Service a controler pre Angular, aj pre registraciu		Closed	Jakub Ondik
Token interceptor (angular)		Closed	Jakub Ondik
Ukladanie tokenu do local storage (angular)		Closed	Jakub Ondik
Filter pre prihlasenie		Closed	Jakub Ondik
Filter pre neuspesne prihlasenie		Closed	Jakub Ondik
Filter pre token		Closed	Jakub Ondik
Konfiguracia springu		Closed	Jakub Ondik
Dokumentacia		Closed	Jakub Ondik

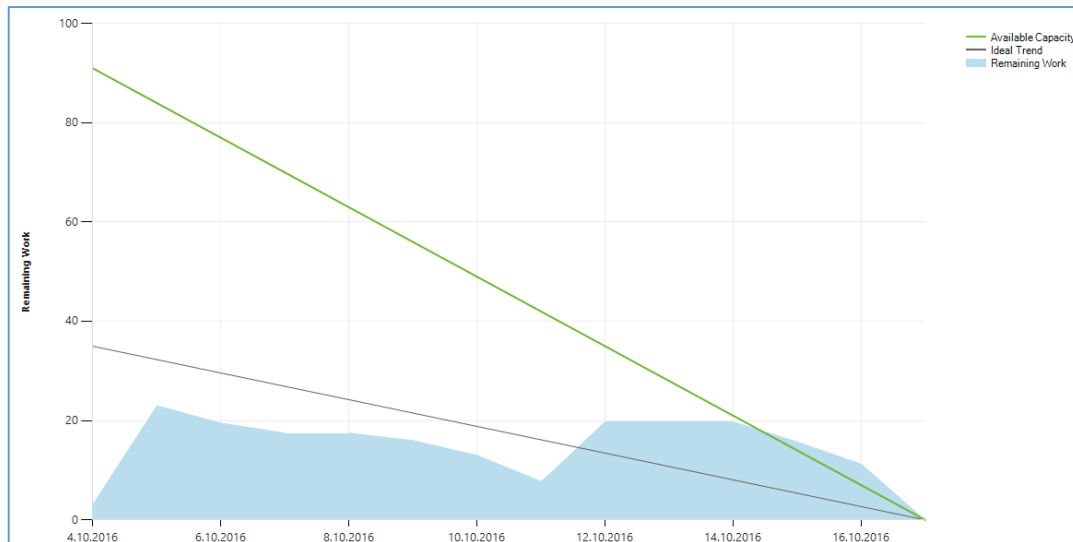
Obrázok 1: Úlohy šprintu Baldr

Overenie emailu	3	Closed	Jakub Ondik
Odoslanie emailu s aktivacnym a deaktivacnym linkom		Closed	Bc. Martin Olejar
Deaktivacia uctu po 12h		Closed	Jakub Ondik
Deaktivacia uctu po pouziti deaktivacneho linku		Closed	Bc. Ondrej Hamara
Aktivacia uctu po potvrdeni aktivacnym linkom		Closed	Bc. Ondrej Hamara
Sablony na emaily v DB		Closed	Bc. Miroslav Hurajt
Dokumentacia		Closed	Jakub Ondik
Prihlasenie z mobilneho zariadenia	3	Closed	Bc. Adam Neupauer
Nastavenie deployu pre mobilne zariadenia		Closed	Bc. Adam Neupauer
Konfiguracia servera		Closed	Bc. Adam Neupauer
Nastavenie continuous integration		Closed	Bc. Adam Neupauer
Vytvorenie unix service scriptu pre Ionic		Closed	Jakub Ondik
Nastavenie certifikatov pre subdomeny		Closed	Jakub Ondik
Dokumentacia		Closed	Bc. Adam Neupauer
Po zmene lokalizácie v profile sa zmení lokalizácia GUI	3	Closed	Jakub Ondik
Translate modul angularu		Closed	Jakub Ondik
Java Bean, jej endpoint a service		Closed	Jakub Ondik
Preklady pre existujúce rozhranie		Closed	Jakub Ondik
Dokumentacia		Closed	Jakub Ondik
Vytvorenie šablóny pre dokumentácie	1	Closed	Bc. Martin Olejar
Vytvorenie dokumentu a nastavenie šablóny dokumentácie		Closed	Bc. Martin Olejar
Hlavička a päta		Closed	Bc. Martin Olejar
Šablóna pre technickú dokumentáciu		Closed	Bc. Martin Olejar
Vytvorenie šablóny pre zápisnicu	1	Closed	Bc. Martin Olejar
Vytvorenie dokumentu a nastavenie šablóny zápisnice		Closed	Bc. Martin Olejar
Definovanie základného postupu na CodeReview	2	Closed	Bc. Ondrej Hamara
Sposob code review		Closed	Bc. Ondrej Hamara
Pribeh code review		Closed	Bc. Ondrej Hamara
Definovanie coding standards	2	Closed	Jakub Ondik
Definovanie coding standards pre Javu (Spring)		Closed	Jakub Ondik
Definovanie coding standards pre JavaScript (AngularJS)		Closed	Jakub Ondik

Obrázok 2: Úlohy šprintu Baldr – pokračovanie

Všetky uvedené úlohy sa podarilo dokončiť (a boli akceptované vlastníkom produktu) a teda nebolo potrebné ich presúvať. Postup práce je možné vidieť na obrázku 3. Nárast dňa 11. októbra bol spôsobený pridaním ďalších úloh, keďže sme nesprávne odhadli počet bodov, ktoré sme schopní vyriešiť počas šprintu, z dôvodu oboznamovania sa s novými technológiami pre niektorých členov tímu.

V tomto šprinte prebehlo nastavenie nástrojov pre kontinuálnu integráciu, vytvoril sa základný autentifikačný mechanizmus a rozhodli sme sa, že budeme podporovať viacjazyčnosť aplikácie. Taktiež sa zaviedli prvotné metodiky.



Obrázok 3: Burndown chart pre šprint Baldr

Velocity šprintu bola 26. Na začiatku šprintu bola velocity 15, ale z dôvodu pridania úloh v strede šprintu sa zvýšila.

4.2 Šprint 2 – Fenrir

Vzhľadom na úspešné oboznámenie sa s technológiami sme sa rozhodli v druhom šprinte zvýšiť počet bodov. Súčasťou tohto šprintu boli aj prehliadky kódov (z angl. code review) a práca v samostatných vetvách – jedna vetva na používateľský príbeh. Úlohy tohto šprintu je možné vidieť na obrázkoch 4 a 5. Výsledkom tohto šprintu bol kompletný používateľský modul vrátane bezpečnostných prvkov (odstránenie autentifikačného tokenu v prípade odcudzenia zariadenia). Taktiež sme sa rozhodli zúčastniť sa súťaže TP Cup.

Postup práce v tomto šprinte je možné vidieť na obrázku 6. V tomto šprinte sa nám nepodarilo dokončiť všetky úlohy, čo malo za následok nedokončenie dvoch používateľských príbehov a ich následne presunutie do ďalšieho šprintu. Zvyšné používateľské príbehy boli úspešne dokončené a akceptované vlastníkom produktu. Rast v začiatkovej časti grafu je spôsobený odhadovaním trvania úloh členmi až po začiatku šprintu.

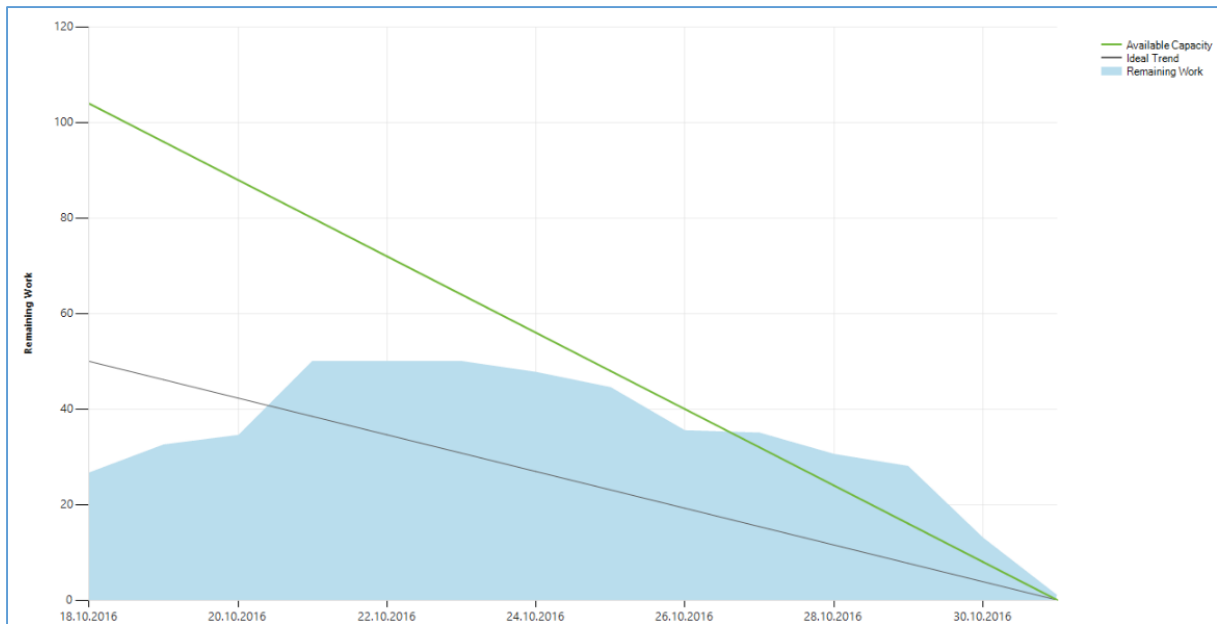
Za úlohy, ktoré neboli dokončené bol zodpovedný jednotlivец, ktorý ich nedokončil bez udania relevantného dôvodu.

Title	Story Points	State	Assigned To
Retrospektiva sprintu Baldr	1	Closed	Bc. Patrik Januska
Vytvorenie dokumentu s retrospektivou sprintu Baldr		Closed	Bc. Patrik Januska
Manuálne odhlásenie zo Story Telleru	2	Closed	Jakub Ondik
Vytvorenie metód pre AngularJS vrátane tlačidla na odhlásenie		Closed	Jakub Ondik
Konfigurácia Springu pre odhlásenie		Closed	Jakub Ondik
Vytvorenie logout filtra		Closed	Jakub Ondik
Zmena hesla	2	Closed	Jakub Ondik
REST endpoint pre zmenu hesla		Closed	Jakub Ondik
AngularJS modul na zmenu hesla		Closed	Jakub Ondik
Odstránenie tokenu	2	Active	Bc. Ondrej Hamara
Vytvorenie REST endpointu pre zrušenie tokenu		Closed	Bc. Patrik Januska
Doplnenie hlasok pre angular		Active	Bc. Ondrej Hamara
Obnovenie hesla cez email	3	Closed	Bc. Miroslav Hurajt
Doplnenie emailovej šablony pre obnovu hesla		Closed	Bc. Miroslav Hurajt
Pridanie AngularJS komponentu pre obnovenie hesla		Closed	Bc. Miroslav Hurajt
Pridanie Spring komponentu pre obnovenie hesla		Closed	Bc. Miroslav Hurajt
Zobrazenie zariadení, z ktorých bol používateľ prihlásený	3	Active	Bc. Martin Olejar
Uprava entity tokenu		Closed	Bc. Martin Olejar
Vytvorenie REST endpointu pre platne tokeny		Closed	Bc. Martin Olejar
Vytvorenie komponentu pre AngularJS		Active	Bc. Ondrej Hamara

Obrázok 4: Úlohy šprintu Fenrir

<ul style="list-style-type: none"> ▶ Používateľ ma základné zobrazenie po prihlásení, ktoré mu bude p... 5 <ul style="list-style-type: none"> ■ Layout rozmiestnenia jednotlivých funkčných prvkov ■ Dodefinovať CSS ■ Hlavná obrazovka s odnavigovaním na súčasný user config ■ Nastýľovanie User Configu 	Closed	Bc. Adam Neupauer
<ul style="list-style-type: none"> ▶ Používateľ ma responzívne používateľské rozhranie pri prihlásení 5 <ul style="list-style-type: none"> ■ Framework pre responzívny dizajn ■ CSS so spoločnými štýlmi ■ Aplikovanie štýlov na login a register screen 	Closed	Bc. Adam Neupauer
<ul style="list-style-type: none"> ▶ Vytvorenie nového projektu 5 <ul style="list-style-type: none"> ■ Vytvorenie entity pre projekt ■ Formulár pre základné detaily projektu ■ REST endpoint ■ Layout rozhrania projektu +CSS + HTML + JS 	Closed	Bc. Patrik Januska
<ul style="list-style-type: none"> ▶ Obnovenie aktivacieho tokenu 3 <ul style="list-style-type: none"> ■ Vytvorenie REST endpointu pre znovuposlanie emailu ■ Vytvorenie AngularJS modulu pre znovuposlanie aktivacieho to... 	Closed	Bc. Martin Olejar
<ul style="list-style-type: none"> ▶ Code Review a support pre sprint Fenrir <ul style="list-style-type: none"> ■ CR a support - Jakub 	Closed	Jakub Ondik
<ul style="list-style-type: none"> ▶ Prihláška 1 <ul style="list-style-type: none"> ■ Napísať prihlasku do TP cupu ■ Odovzdať analogovú verziu 	Closed	Jakub Ondik

Obrázok 5: Úlohy šprintu Fenrir - pokračovanie



Obrázok 6: Burndown chart pre šprint Fenrir







































Velocity tohto šprintu sme sa rozhodli zvýšiť na 32, čo sa ukázalo ako nezvládnuteľné (z dôvodu zlyhania jednotlivca) a teda dosiahnutá velocity bola 27, čiže približne na úrovni predchádzajúceho šprintu.

4.3 Šprint 3 – Freya

V tomto šprinte sme sa rozhodli zobrať vyšší počet bodov, keďže nedokončené úlohy boli problémom jednotlivca a nie tímu. Rozhodli sme sa zaviesť roly používateľov a autorizačné anotácie špecifické pre tieto roly. Taktiež sme sa rozhodli o zavedenie notifikácií o dianí v systéme. Príkladom diania je priradenie roly používateľovi pre konkrétny projekt. Tiež sme vytvorili základné rozhranie pre projekty. Okrem týchto funkcií boli predmetom tohto šprintu aj úlohy prenesené z predchádzajúceho šprintu a dokumentácia potrebná pre prvý kontrolný bod. Tiež sme zaviedli analýzu zdrojového kódu pomocou nástroja SonarQube a zaviedli sme systém Sentry pre vzdialené logovanie klientskej aplikácie, ktorý zatiaľ nie je prepojený s JavaScript knižnicou na logovanie RavenJS. Všetky úlohy je možné vidieť na obrázkoch 7 a 8.

Výsledkom tohto šprintu bola základná správa projektov – prehľad, zobrazenie, pridávanie používateľov a priradovanie používateľských rolí. Taktiež sa úspešne podarilo nainštalovať systémy SonarQube a Sentry a bol implementovaný systém notifikácií pre používateľov. Postup práce v tomto šprinte je možné vidieť na obrázku 9.

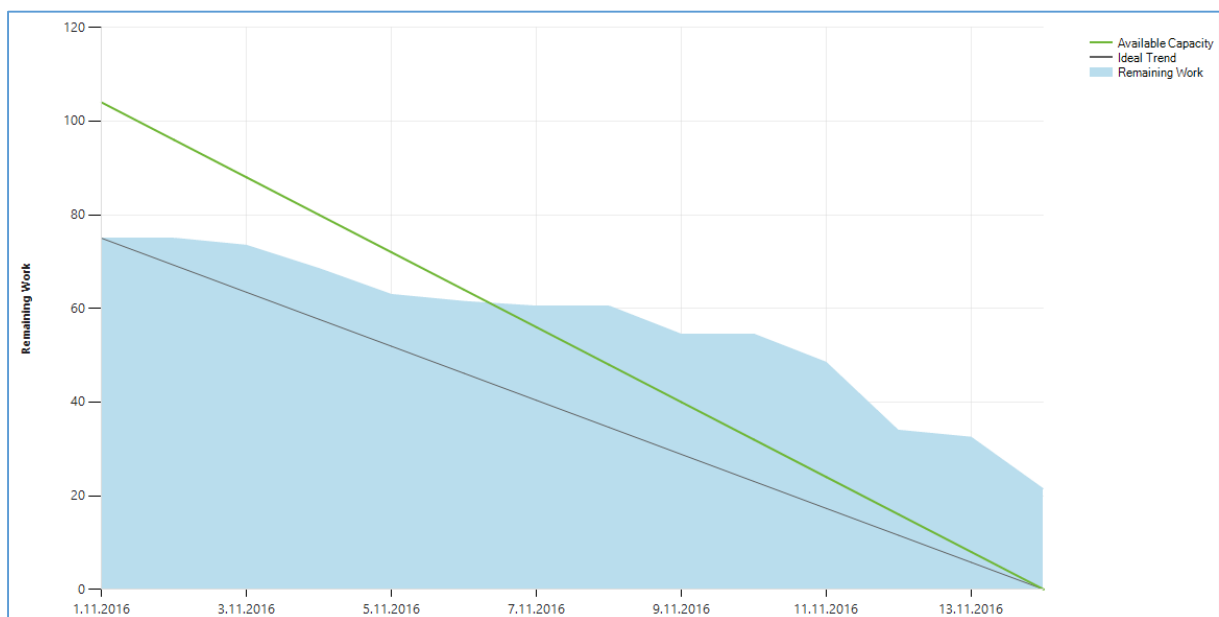
Veľkým negatívom tohto šprintu je nedokončenie pomerne veľkého počtu používateľských príbehov a úloh, čo bolo spôsobené nárazovou prácou tesne pred ukončením šprintu. Taktiež sa nepodarilo dokončiť ani jednu prenesenú úlohu z minulého šprintu, za ktoré bol zodpovedný jednotlivec, ktorý nedokončil (nezačal riešiť) úlohy tohto šprintu, ktoré si sám vybral.

Title	Story Points	State	Assigned To
4  Odstranenie tokenu	2	Active	Bc. Ondrej Hamara
 Doplnenie hlasok pre angular		Active	Bc. Ondrej Hamara
4  Prehľad uz existujucich projektov	5	Closed	Bc. Patrik Januska
 Zmeny nad DB		Closed	Bc. Martin Olejar
 REST endpoint na získanie zoradených projektov pre prihlasene...		Closed	Bc. Patrik Januska
 Zobrazenie zoznamu projektov		Closed	Bc. Patrik Januska
 REST endpoint pre nactanie informácii o konkrétnom projekte		Closed	Jakub Ondik
4  Zobrazenie zoznamu používateľov v projekte	2	Closed	Bc. Patrik Januska
 Zobrazenie zoznamu		Closed	Bc. Patrik Januska
 REST endpoint na nactanie zoznamu pouzivatelov v projekte		Closed	Bc. Patrik Januska
4  Zobrazenie zariadení, z ktorých bol používateľ prihlásený	3	Active	Bc. Martin Olejar
 Vytvorenie komponentu pre AngularJS		Closed	Bc. Ondrej Hamara
4  Priradenie používateľa k role v rámci projektu	3	Resolved	Bc. Patrik Januska
 Vytvorenie REST endpointu pre priradenie roly		Closed	Bc. Patrik Januska
 Uprava AngularJS modulu pre priradenie roly		Closed	Bc. Adam Neupauer
 Zaslanie notifikácii o pridelení role		Closed	Jakub Ondik
4  Definovanie roly administrátora	3	Resolved	Bc. Adam Neupauer
 Vytvorenie authorization metod a nastavenie anotácii pre rolu a...		Closed	Bc. Adam Neupauer
 Vytvorenie angularJS modulu špecifického pre administrátora		Closed	Bc. Adam Neupauer
4  Notifikovanie používateľa o dianí v StoryTelleri	8	Active	Jakub Ondik
 DBO notifikácie		Closed	Bc. Miroslav Hurajt
 Websocket modul pre Java na publish subscribe		Closed	Jakub Ondik
 Templatey pre notifikácie		Active	Bc. Miroslav Hurajt
 Angular modul na pripojenie na websockety ktorý detekuje notif...		Closed	Jakub Ondik
4  Pozvanie do projektu cez StoryTeller	3	Active	Bc. Miroslav Hurajt
 Rozhranie pre pridanie používateľa		Closed	Bc. Martin Olejar
 REST endpoint pre príjem priradenia používateľa k projektu + sp...		New	Bc. Martin Olejar
 Template na notifikáciu pozvania používateľa do projektu		Active	Bc. Miroslav Hurajt
4  Definovanie roly analytika	3	New	Bc. Ondrej Hamara
 Vytvorenie authorization metod a nastavenie anotácii pre rolu a...		New	Bc. Ondrej Hamara
 Vytvorenie angularJS modulu špecifického pre analytika		New	Bc. Ondrej Hamara
4  Definovanie roly zákazníka	3	New	Bc. Ondrej Hamara
 Vytvorenie authorization metod a nastavenie anotácii pre rolu z...		New	Bc. Ondrej Hamara
 Vytvorenie angularJS modulu špecifického pre zákazníka		New	Bc. Ondrej Hamara
4  Finalizácia inžinierskeho diela k prvému kontrolnému bodu	2	Closed	Jakub Ondik
 Úvod		Closed	Bc. Martin Olejar
 Globálne ciele projektu		Closed	Jakub Ondik
 Celkový pohľad na systém		Closed	Jakub Ondik

Obrázok 7: Úlohy šprintu Freya

4	Finalizacia riadenia k prvemu kontrolnemu bodu	2	Active	Bc. Martin Olejar
	Uvod		Closed	Bc. Martin Olejar
	Roly členov tímu		Closed	Bc. Martin Olejar
	Podiel prace		Closed	Bc. Martin Olejar
	Aplikacie manažmentov		New	Bc. Miroslav Hurajt
	Sumarizácie šprintov		Closed	Jakub Ondik
	Používané metodiky		Closed	Bc. Patrik Januska
	Globálna retrospektíva		Active	Bc. Martin Olejar
4	Konfigurácia SonarQube	3	Closed	Jakub Ondik
	Konfigurácia serverovej casti		Closed	Jakub Ondik
	Nastavit pom.xml		Closed	Jakub Ondik
4	Konfiguracia Sentry	5	Closed	Jakub Ondik
	Nastavit bridge na úrovni DC a zosietovat VM		Closed	Jakub Ondik
	Nastavit reverzne proxy a ssl certifikat		Closed	Jakub Ondik
	Nastavit Postgres na pripojenia z VM s private IP		Closed	Jakub Ondik
	Nainstalovat redis		Closed	Jakub Ondik
	Nainstalovat sentry		Closed	Jakub Ondik

Obrázok 8: Úlohy šprintu Freya - pokračovanie



Obrázok 9: Burndown chart pre šprint Freya

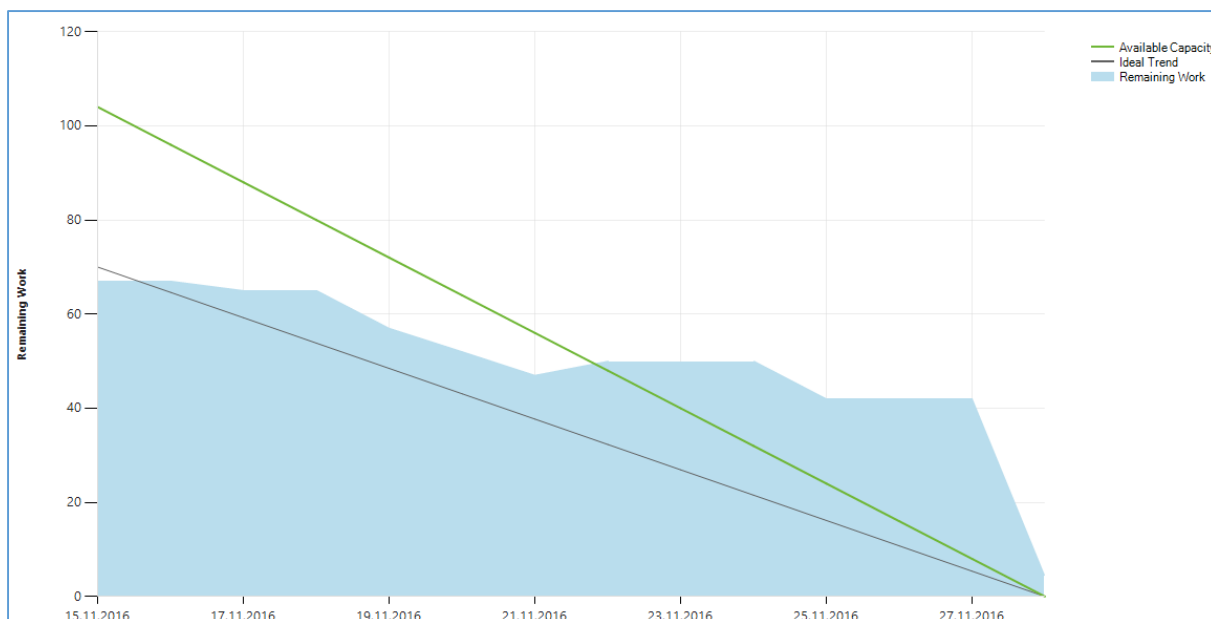
Velocity tohto šprintu sme sa rozhodli zvýšiť na 37 aj z dôvodu prenášaných príbehov. Dosažená velocity bola na úrovni 26, čiže na podobnej úrovni ako predchádzajúce šprinty. Problém znovu nastal na úrovni jednotlivca, ktorému sa nepodarilo dokončiť ani úlohy prenášané z predchádzajúcich šprintov.

4.4 Šprint 4 – Heimdallr

V tomto šprinte sme sa rozhodli zobrať vyšší počet používateľských bodov. Rozhodli sme sa vytvoriť riadenie prístupu pre roly používateľa – pre rolu analytika a administrátora (vlastníka) projektu. Taktiež sme vytvorili funkcionality na pridávanie používateľov do projektu používateľom, ktorý má oprávnenú rolu. Konečne sme sa dostali k implementácii tvorby skíc. Vytvorili sme plátno určené na kreslenie skíc, panel kresliacich nástrojov a tvarov, ktoré fungujú na princípe drag and drop a vytvorili sme základný nástroj – textové pole. Taktiež sme sa rozhodli zaviesť vzdialené logovanie chýb vyskytujúcich sa na klientskej strane aplikácie. Na toto sme použili nástroj Sentry.io v kombinácii s javascriptovým pluginom RavenJS, pre ktorý sme vytvorili wrapper z dôvodu zjednodušenia logovania chýb. Dané úlohy je možné vidieť na obrázku 10. Postup práce je zobrazený na obrázku 11.

Title	Story Points	State	Assigned To
Notifikovanie používateľa o dianí v StoryTelleri	8	Active	Jakub Ondik
Templaty pre notifikácie		Active	Bc. Miroslav Hurajt
Pozvanie do projektu cez StoryTeller	3	Active	Bc. Miroslav Hurajt
Template na notifikáciu pozvania používateľa do projektu		Active	Bc. Miroslav Hurajt
REST endpoint pre príjem priradenia používateľa k projektu + sp...		Closed	Bc. Martin Olejar
Definovanie roly zakaznika	3	Active	Bc. Ondrej Hamara
Vytvorenie authorization metod a nastavenie anotácii pre rolu z...		Closed	Bc. Ondrej Hamara
Vytvorenie angularJS modulu specifickeho pre zakaznika		Closed	Bc. Ondrej Hamara
Definovanie roly analytika	3	Active	Bc. Ondrej Hamara
Vytvorenie authorization metod a nastavenie anotacji pre rolu a...		Closed	Bc. Ondrej Hamara
Vytvorenie angularJS modulu specifickeho pre analytika		Closed	Bc. Ondrej Hamara
Používateľ má možnosť drag'n'drop tvarov na plátno	13	Closed	Bc. Adam Neupauer
Vytvorenie canvasu		Closed	Bc. Adam Neupauer
Vytvorenie panelu s nástrojmi na kreslenie		Closed	Bc. Adam Neupauer
Funkcia drag n drop/pridanie na základe kliku		Closed	Bc. Adam Neupauer
Vytvorenie geom. objektov : textové pole		Closed	Jakub Ondik
Vytvorenie wrappera pre RavenJS	3	Active	Jakub Ondik
Wrapper metody pre všetky typy logovacích správ, vrátane vyni...		Closed	Jakub Ondik
Perzistencia skice	8	Active	Jakub Ondik
Navrh spôsobu perzistencie skice		Active	Jakub Ondik
Implementácia perzistencie skice		Active	Jakub Ondik
Registrácia resource pre skicu v AngularJS module		New	Bc. Martin Olejar
CRUD REST endpointy		New	Bc. Martin Olejar

Obrázok 10: Úlohy šprintu Heimdallr



Obrázok 11: Burndown chart pre šprint Heimdallr

Počiatková velocity tohto šprintu bola 41, avšak z dôvodu odovzdania zadaní na iných predmetoch a prenášania úloh medzi šprintami určitých jednotlivcov sa nám podarilo dosiahnuť velocity iba 13. Používateľské príbehy, ktoré neboli dokončené, boli štandardne prenesené do nasledujúceho šprintu.

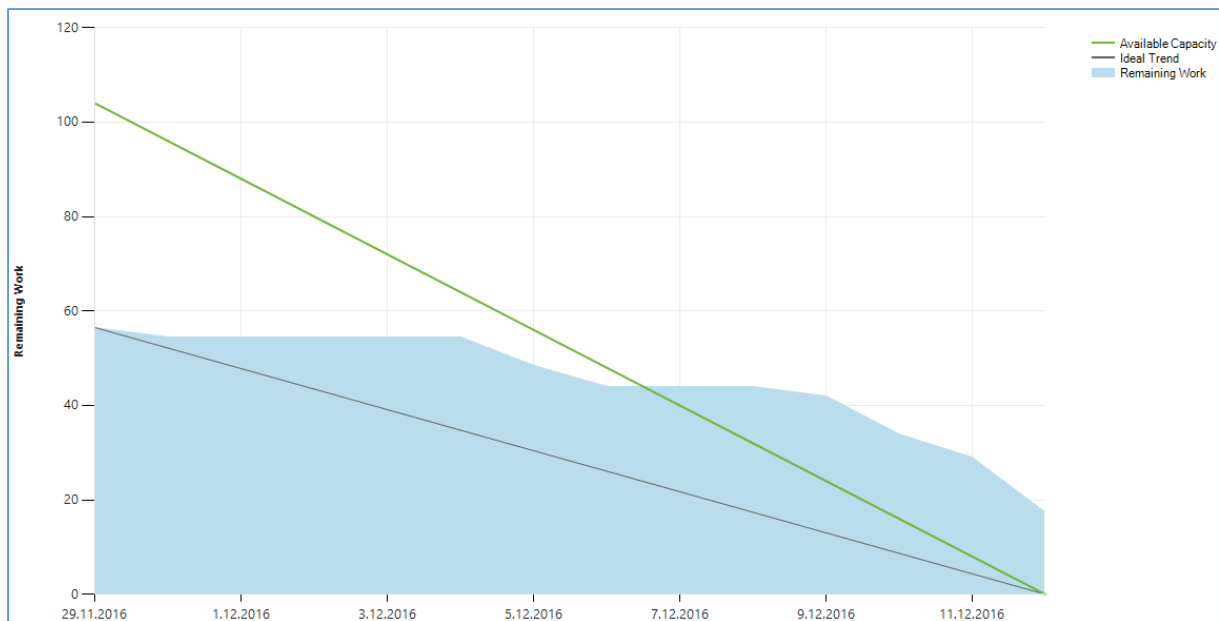
4.5 Šprint 5 – Huginn

V tomto šprinte sme sa okrem prenesených úloh rozhodli navrhnuť a implementovať ďalšie kresliace nástroje. Taktiež sme sa rozhodli vytvoriť zoznam skíc konkrétneho projektu, ktorý reaguje na posúvanie (scrolling) prstom, pričom sú zobrazené generované náhľady vytvorených skíc. Skice sme sa rozhodli obohatiť o nastavenie vlastností štýlov, akými sú napríklad farba, orámovanie a veľkosť písma. Do tohto šprintu sme tiež zahrnuli implementáciu zloženia (vykreslenia) existujúcej skice na plátno, od čoho závisí funkcionálna úprava skíc. Jednotlivé úlohy je možné vidieť na obrázku 12. Postup práce na tomto šprinte je zobrazený na obrázku 13.

Velocity tohto šprintu bola 40 aj z dôvodu prenášania úloh z predchádzajúcich šprintov, avšak podobne ako aj v predchádzajúcich šprintoch sa nám ju nepodarilo dosiahnuť – dosiahli sme velocity 16. Takáto nízka velocity je zapríčinená opakovaným nepracovaním stále tých istých členov tímu, ktorí nie sú schopní dokončiť úlohy ešte z tretieho šprintu. Štandardne, všetky nedokončené úlohy boli presunuté do ďalšieho šprintu.

Title	Story Points	State	Assigned To
<ul style="list-style-type: none"> ▾ ■ Notifikovanie pouzivatela o diani v StoryTelleri <ul style="list-style-type: none"> ■ Templaty pre notifikacie 	8	Active	Jakub Ondik
<ul style="list-style-type: none"> ▾ ■ Pozvanie do projektu cez StoryTeller <ul style="list-style-type: none"> ■ Template na notifikaciju pozvania pouzivatela do projektu 	3	Active	Bc. Miroslav Hurajt
<ul style="list-style-type: none"> ▾ ■ Pridanie vlastnosti pre komponent v skici <ul style="list-style-type: none"> ■ Zakladne vlastnosti prvkov ■ Pridanie CSS vlastnosti ■ Popup okno 	5	Resolved	Bc. Adam Neupauer
<ul style="list-style-type: none"> ▾ ■ Perzistencia skice <ul style="list-style-type: none"> ■ Navrh sposobu perzistencie skice ■ Implementacia perzistencie skice ■ Registracia resource pre skicu v AngularJS module ■ CRUD REST endpointy 	8	Resolved	Jakub Ondik
<ul style="list-style-type: none"> ▾ ■ Vytvorenie wrappera pre RavenJS <ul style="list-style-type: none"> ■ Zaregistrovanie watchera pre prihlaseneho usera, ktorý nastavi u... ■ Pridanie logovania pre jednotlivé AngularJS moduly 	3	Resolved	Jakub Ondik
<ul style="list-style-type: none"> ▾ ■ Zobrazenie zoznamu skic projektu <ul style="list-style-type: none"> ■ REST endpoint na získanie skic pre projekt ■ Vytvorenie AngularJS metody na získanie zoznamu skic pre kon... ■ Vytvorenie dlazdicoveho panelu so skicami, ktorý reaguje na tou... 	5	Active	Bc. Patrik Januska
<ul style="list-style-type: none"> ▾ ■ Zobrazenie skice <ul style="list-style-type: none"> ■ Zloženie skice z prijatých objektov 	3	New	Bc. Ondrej Hamara
<ul style="list-style-type: none"> ▾ ■ Možnosť pridať základné komponenty do skice <ul style="list-style-type: none"> ■ Pridanie buttonu ■ Pridanie radiobuttonu ■ Pridanie labelu ■ Pridanie containeru ■ Pridanie checkboxu ■ Pridanie obrazku ■ Pridanie komponentu pre volný text ■ Pridanie comboboxu 	5	New	Bc. Miroslav Hurajt
		New	Bc. Miroslav Hurajt
		Closed	Bc. Ondrej Hamara
		Closed	Bc. Martin Olejar
		New	Bc. Patrik Januska
		New	Bc. Miroslav Hurajt
		New	Bc. Adam Neupauer
		Closed	Bc. Ondrej Hamara
		Closed	Bc. Ondrej Hamara

Obrázok 12: Úlohy šprintu Huginn



Obrázok 13: Burndown chart pre šprint Huginn

5 Používané metodiky

V tejto kapitole sú zosumarizované metodiky, podľa ktorých sme postupovali pre dosiahnutie konzistentnosti jednotlivých súčastí projektu. S postupujúcou prácou na projekte sa objavovali stále nové podnety, pre ktoré bolo potrebné stanoviť si štandard, akým postupom danú vec – ak sa vyskytne, riešiť.

V projekte sme teda identifikovali nasledovné metodiky:

- *Metodika plánovania*
- *Metodika konvencií písania zdrojového kódu – Coding Standards*
- *Metodika prehliadok zdrojového kódu – Code review*
- *Metodika testovania zdrojového kódu*
- *Metodika verziovania zdrojového kódu*
- *Metodika písania dokumentácie*

Každá z metodík je stručne opísaná spolu s odkazom na koreňový dokument v nasledujúcich kapitolách.

5.1 Metodika plánovania

Metodika hovorí o mikromanažmente úloh, teda o zodpovednosti človeka za používateľský príbeh. Zodpovedný človek za používateľský príbeh by mal po aktivovaní používateľského príbehu zorganizovať stretnutie, na ktorom sa rozoberie obsah jednotlivých úloh, teda vykoná sa detailnejšia analýza problému a jednoduchý návrh riešenia úloh. Ďalej by sa mala definovať priorita úloh a poradie vypracovávania úloh a stanovenie dátumov, v ktorých budú dané úlohy hotové.

Metodika obsahuje aj stavy používateľského príbehu, ktoré je nutné priebežne meniť a ďalšie pravidlá pre riešenie používateľských príbehov.

Odkaz na metodiku: <https://1drv.ms/b/s!AnORgXjFAcOEhUAdTdwsVIhi00Ib>

5.2 Metodika konvencií písania zdrojového kódu – Coding Standards

Metodika opisuje základné konvencie pri písaní zdrojového kódu v jazykoch Java a JavaScript. Tieto konvencie sú upravené a obohatené na základe dobrých návykov (angl. best practices). Jednotlivé konvencie sprevádzajú praktické ukážky kódu, vrátane príkladov s nedodržaním danej konvencie. Konvencie uvedené v tejto metodike majú odporúčací charakter, t. j. autor ich môže porušiť v špecifických prípadoch s odôvodnením.

Odkaz na metodiku: <https://1drv.ms/b/s!AnORgXjFAcOEhEkdTdwsVIhi00Ib>

5.3 Metodika testovania zdrojového kódu

Metodika hovorí o pravidlách testovania zdrojového kódu. Každý človek zodpovedný za úlohu vytvára a vykonáva k svojej úlohe príslušné testy. A práve ich vytváranie, vykonávanie a

vyhodnocovanie je opísané v kapitolách tejto metodiky. Testovanie v tomto projekte pozostáva z dvoch častí a to testovania backendu, teda logiky samotnej aplikácie a testovania frontendu, teda toho čo vidí používateľ aplikácie. Metodika obsahuje podrobné postupy s konkrétnymi príkladmi použitia vybranými z projektu zvlášť pre backend a frontend.

Odkaz na metodiku: <https://1drv.ms/b/s!AnORgXjFACOEhUIdTdwsVIhi00Ib>

5.4 Metodika prehliadok zdrojového kódu – Code review

Metodika opisuje spôsob akým sa má v projekte vykonávať prehliadka zdrojového kódu (angl. Code Review). V jednotlivých kapitolách sa prechádza od toho, čo je prehliadka zdrojového kódu, prečo ju robiť, cez moderné techniky až po samotný postup jej vykonávania. V postupe sa hovorí o tom, že má zväčša dvoch účastníkov – autora zmeny a kontrolóra kódu, pričom ak autor zmení kód prebehne jeho kontrola zo strany kontrolóra a ak je zmena neprijateľná, musí autor nájdené nedostatky odstrániť. Takýto cyklus sa opakuje dovtedy, kým kontrolór neprehlási všetky zmeny za prijateľné.

Odkaz na metodiku: <https://1drv.ms/b/s!AnORgXjFACOEhXbWQvlj1Dmin1uA>

5.5 Metodika verziovania zdrojového kódu

Metodika opisuje pravidlá pre vetvenie zdrojového kódu v rámci hlavného repozitára a vytváranie vetiev pre používateľské príbehy. Hovorí o tom, že na používateľskom príbehu sa pracuje v osobitnej vetve pre daný používateľský príbeh. Vetvu pre používateľský príbeh vytvára ten, kto je zaň zodpovedný. Obsahuje pravidlá pre nazývanie vetiev a takisto postup ich korektného vytvorenia a pracovania s nimi. Súčasťou metodiky je aj opis vytvorenia požiadavky na kontrolu používateľského príbehu a konkrétne príkazy, ktoré treba pri tejto časti práce používať.

Odkaz na metodiku: <https://1drv.ms/b/s!AnORgXjFACOEhGUdTdwsVIhi00Ib>

5.6 Metodika písania dokumentácie

Metodika obsahuje šablónu dokumentu metodiky. Dokument obsahuje titulnú stranu, obsah a samotné kapitoly. Samozrejmosťou sú čísla strán a hlavička. V metodike sú popísané štýly písma, ktoré treba dodržiavať pri nadpisoch a podnadpisoch na rôznych úrovniach. Ďalej sú uvedené príklady odrážok, číslovaní a poznámok. Uvedená je tiež vzorová tabuľka.

Odkaz na metodiku: <https://1drv.ms/b/s!AnORgXjFACOEhGYdTdwsVIhi00Ib>

6 Globálna retrospektíva

Táto kapitola obsahuje globálnu retrospektívu pre celý zimný semester. Počas celého semestra sme sa snažili zaviesť procesy v rozličných oblastiach riadenia projektu, aby sme mohli pracovať viac koordinovane, efektívne a tímovo. Po každom šprinte, ktorý trval vždy 2 týždne, sme na tímovom stretnutí spísali retrospektívu za daný šprint. Retrospektíva prebiehala tak, že každý člen tímu vyjadril činnosti, ktoré sa mu počas šprintu páčili a nepáčili, a taktiež návrhy do budúcnosti. Hlavným cieľom retrospektívy bolo poukázať na procesy, v ktorých používaní by sme mali ďalej pokračovať, ale taktiež nájsť a definovať procesy, ktoré nefungujú správne a mali sme ich v budúcnosti zlepšiť.

V prvých týždňoch semestra sme sa hlavne učili pracovať s novými technológiami a pracovať v tíme, keďže väčšina členov tímu nemala skúsenosti s tímovými projektami. Už v počiatočnej fáze riešenia tímového projektu sa nám podarilo dobre nastaviť tieto procesy:

- verziovanie a kontrola kvality kódu – definovali sme si metodiku verziovania, ktorá obsahuje pravidlá pre prácu s vetvami. Pre každý používateľský príbeh existuje osobitná vetva, čím sa predchádza konfliktom a blokovaniu ostatných členov tímu. Nová funkcionálna je pridaná do hlavnej vetvy až po jej schválení – osoba zodpovedná za používateľský príbeh musí vytvoriť pull request, iný člen tímu skontroluje funkčnosť a dodržanie štandardov pre písanie zdrojového kódu.
- písanie jednotkových testov – pre každú funkciu musí byť vytvorený jednotkový test,
- zavedenie Continuous Integration – hlavná vetva je automaticky nasadzovaná,
- používanie Team Foundation Server pre správu používateľských príbehov a úloh,
- používanie OneDrive ako úložisko všetkých dokumentov projektu,
- používanie Slack-u ako hlavný komunikačný nástroj.

Na základe retrospektív jednotlivých šprintov sme určili problémy, ktoré je potrebné vyriešiť alebo sa im v budúcnosti vyhnúť. Medzi tieto problémy patria:

- nedostatočné vyjadrovanie sa k daniu v tíme,
- odkladanie práce na poslednú chvíľu pred koncom šprintu, čoho následkom bolo viackrát preloženie úloh do ďalšieho šprintu,
- nedostatočný opis a mikromanažment úloh, mikromanažment zahŕňa definovanie závislostí medzi úlohami, ich analýzu a návrh, následkom sú problémy s vyriešením úloh a blokovanie úloh,
- chýbajúca samostatnosť pri štúdiu.

V aktuálnej podobe náš systém poskytuje možnosť registrácie, prihlásenia a odhlásenia používateľa spolu s možnosťou obnovenia registračného emailu a hesla. Po prihlásení má používateľ možnosť úpravy informácií o svojom profile a zobrazenia zariadení, na ktorých je v systéme prihlásený. Používateľ môže tiež vytvoriť nový projekt a poslať notifikáciu iným používateľom pre ich priradenie do projektu. Systém ponúka možnosť zobrazenia zoznamu dostupných projektov používateľa a zoznamu používateľov, ktorí sú v danom projekte. Ďalšou možnosťou je zmena rolí používateľov v konkrétnom projekte a vytvorenie novej skice k projektu.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Projektová dokumentácia – inžinierske dielo

Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller-04@googlegroups.com
Akademický rok: 2016/2017

Obsah

1	Úvod	1-1
2	Globálne ciele projektu	2-1
3	Celkový pohľad na systém	3-1
3.1	Backend	3-2
3.1.1	UserManagement	3-2
3.1.2	ProjectManagement.....	3-2
3.1.3	SketchManagement	3-2
3.1.4	DrawableComponents	3-2
3.1.5	DrawableTemplateProvider	3-2
3.1.6	HtmlRenderer	3-3
3.1.7	Notifications	3-3
3.1.8	WebsocketHandler	3-3
3.1.9	Authentication	3-3
3.1.10	AccessControlHandler	3-3
3.1.11	wkhtmltopdf	3-4
3.1.12	PostgreSQL	3-4
3.2	Client	3-4
3.2.1	UserService	3-4
3.2.2	ProjectService.....	3-4
3.2.3	SketchService	3-4
3.2.4	DrawableComponents	3-5
3.2.5	Navigation	3-5
3.2.6	WebsocketHandler	3-5
3.2.7	EventHandler.....	3-6
3.2.8	Notifications	3-6
3.3	Dátový model	3-6

1 Úvod

Tento dokument predstavuje dokumentáciu inžnierskeho diela, ktoré je vytvárané v rámci predmetu Tímový projekt v akademickom roku 2016/2017. Dokument zahŕňa globálne ciele projektu pre jednotlivé semestre a celkový pohľad na štruktúru a funkcionality systému.

Názov témy nášho projektu je *Story Teller – Zber a vyhodnocovanie požiadaviek* a jeho cieľom je vytvoriť informačný systém, ktorý umožní pohodlné zadávanie a sledovanie splnenia používateľských scenárov (funkcionálnych požiadaviek). Zber požiadaviek bude realizovaný prostredníctvom vizuálneho skicovania obrazoviek scenáru a ich komentovania, na základe čoho môžu byť vygenerované akceptačné testy. Ďalšou funkcionality systému bude vykonávanie akceptačných testov a komentovanie ich výsledkov.

Prvá kapitola dokumentu obsahuje globálne ciele projektu pre zimný semester.

V druhej kapitole dokumentu je bližšie popísaná architektúra systému, jej jednotlivé komponenty a vzťahy medzi nimi. Pri každom komponente je uvedený odkaz na priložené elektronické dokumenty (technická dokumentácia a používateľská príručka) a ich kapitoly. Kapitola obsahuje taktiež dátový model aplikácie.

2 Globálne ciele projektu

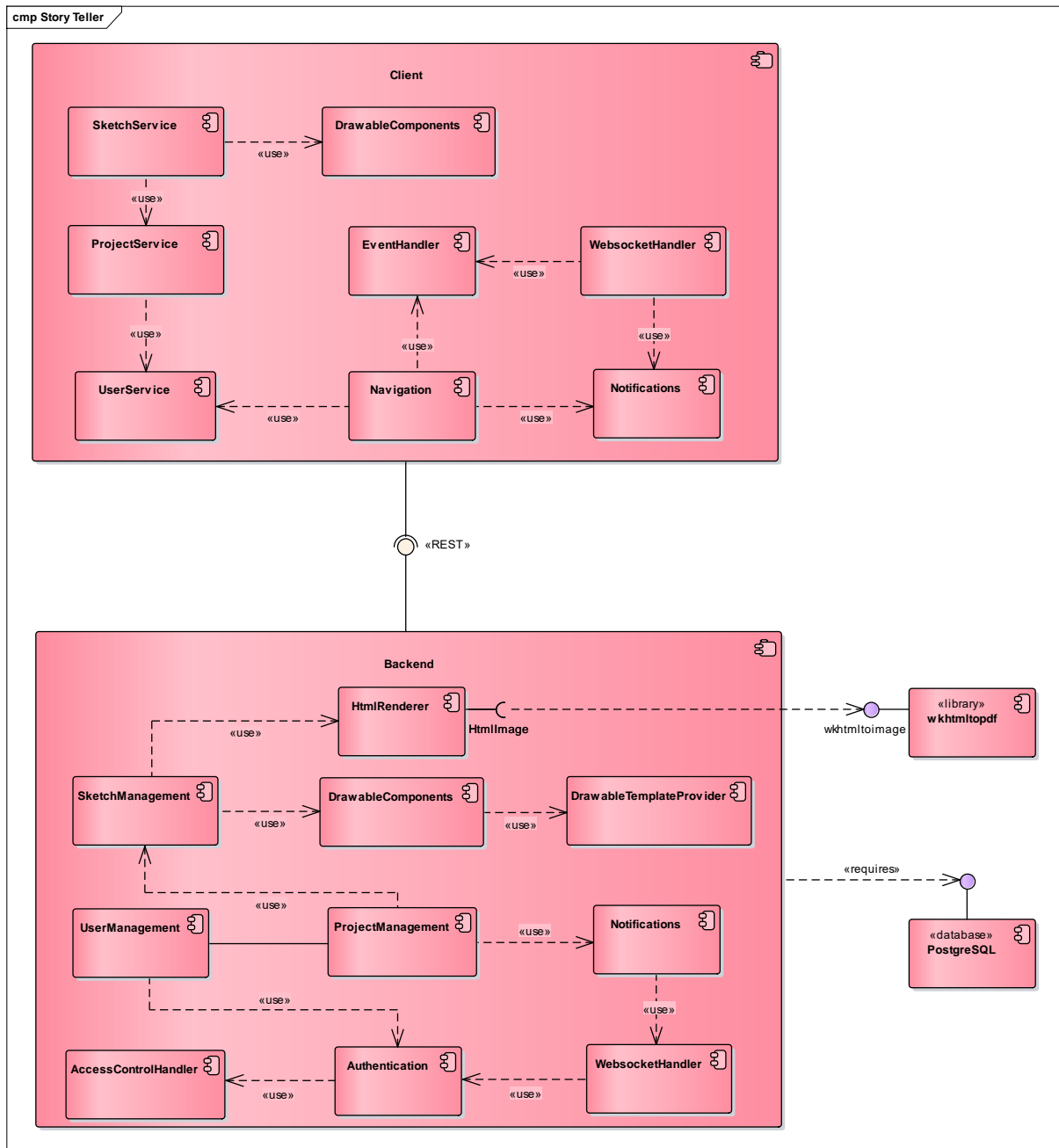
Naším cieľom pre zimný semester je vytvorenie prototypu obsahujúceho základnú logiku pre vytváranie projektov a skíc, vrátane prvotného generovania testov.

Globálne ciele sú nasledovné:

1. Autentifikácia používateľa vrátane mechanizmov na obnovenie prístupu
2. Riadenie prístupu k projektom
3. Vytvorenie a správa projektov
4. Tvorba a modifikácia skíc
5. Zdieľanie projektov a skíc
6. Prvotné generovanie akceptačných testov
7. Centrálné vykonávanie akceptačných testov aspoň pre jednu platformu
8. Možnosť sledovania vykonávania (výsledkov) akceptačných testov

3 Celkový pohľad na systém

Aplikácia je realizovaná klient – server architektúrou, pričom klienti sú tenkí – webové prehliadače a aplikácie pre tablety. Títo klienti komunikujú so serverom pomocou HTTP protokolu cez REST rozhranie. Táto architektúra vrátane závislosti komponentov je zobrazená na obrázku 1.



Obrázok 1: Architektúra systému

3.1 Backend

3.1.1 UserManagement

Komponent UserManagement rieši celkovú správu používateľov, vrátane aktivácie a deaktivácie používateľského účtu. Taktiež rieši zmenu a obnovu hesla, obnovu aktivačných, resp. deaktivovaných tokenov a úpravu informácií o používateľoch.

Technická dokumentácia

Kapitola 1: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.2 ProjectManagement

Komponent ProjectManagement zabezpečuje vytváranie a správu projektov, pre ktoré sa budú vytvárať skice a generovať testy.

Technická dokumentácia

Kapitola 4: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.3 SketchManagement

Komponent SketchManagement zabezpečuje správu skíc. Taktiež zabezpečuje vygenerovanie ich náhľadu ako obrázku.

Technická dokumentácia

Kapitola 8: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.4 DrawableComponents

Komponent DrawableComponents predstavuje komponenty tvoriace skicu. Príkladom takéhoto komponentu je textové pole. Tento komponent taktiež zabezpečuje nastavenie CSS vlastností (štýlov) a konverziu na html kód pred renderovaním skice ako obrázka.

Technická dokumentácia

Kapitola 8: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.5 DrawableTemplateProvider

Komponent DrawableTemplateProvider predstavuje poskytovateľa šablón dostupných komponentov, ktoré ponúka komponent DrawableComponents.

Technická dokumentácia

Kapitola 8: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.6 HtmlRenderer

Komponent HtmlRenderer predstavuje wrapper nad knižnicou wkhtmltopdf. Zabezpečuje vytvorenie procesu, v ktorom je táto knižnica spúšťaná, a presmerovanie deskriptora stdout do bajtového poľa v Jave.

Technická dokumentácia

Kapitola 8: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.7 Notifications

Komponent Notifications zabezpečuje odosielanie notifikácií konkrétnym používateľom, resp. používateľovi. Odosielanie prebieha prostredníctvom websocketov.

Technická dokumentácia

Kapitola 6.3.1: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.8 WebSocketHandler

Komponent WebSocketHandler zabezpečuje vytvorenie STOMP websocketového servera. Taktiež zabezpečuje registráciu kanálov a samotné publish / subscribe funkcie.

Technická dokumentácia

Kapitola 6.3.2: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.9 Authentication

Komponent Authentication zabezpečuje overenie používateľa. Taktiež zabezpečuje vygenerovanie autentifikačných tokenov v prípade správneho zadania prihlasovacích údajov a overenie identity používateľa pre websockety. Tokeny boli zvolené z dôvodu poskytovania možnosti stateless autentifikácie pri prípadnom škálovaní aplikácie.

Technická dokumentácia

Kapitoly 1.3.1 a 1.3.2: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.10 AccessControlHandler

Komponent AccessControlHandler zabezpečuje riadenie prístupu na základe rolí. Roly nie sú globálne, ale pre konkrétny projekt.

Technická dokumentácia

Kapitola 1.3.7: https://1drv.ms/b/s!AnORgXjFACOEhQ_yCmfcF5irje7t

3.1.11 wkhtmltopdf

Komponent wkhtmltopdf predstavuje externú knižnicu zodpovednú za renderovanie html súborov do pdf formátu. Taktiež umožňuje renderovanie do jpg a png formátu, ktoré je použité komponentom HtmlRenderer.

3.1.12 PostgreSQL

Komponent PostgreSQL predstavuje databázový server, ktorý je využitý na perzistenciu údajov. Komunikácia je zabezpečená prostredníctvom JDBC pripojenia, pričom je použitý rámec Spring Data, ktorý používa entitno-relačný mapovač Hibernate.

3.2 Client

3.2.1 UserService

Komponent UserService zabezpečuje registráciu a prihlásenie, zaslanie požiadavky na obnovu hesla a aktivačných, resp. deaktivovaných tokenov, zobrazenie a modifikáciu používateľského profilu vrátane zmeny hesla a nastavenia lokalizácie. Tiež zabezpečuje uloženie autentifikačného tokenu a jeho pripojenie k požiadavkám zasielaných serverovej aplikácií.

Technická dokumentácia

Kapitola 1: https://1drv.ms/b/s!AnORgXjFAcOEhQ_yCmfcF5irje7t

Používateľská príručka

Kapitoly 1 a 2: <https://1drv.ms/b/s!AnORgXjFAcOEhQ7nvJNqzZfLQxQM>

3.2.2 ProjectService

Komponent ProjectService zabezpečuje vytvorenie projektu a modifikáciu jeho informácií, vrátane správy používateľov, ktorý k projektu majú prístup. Taktiež zabezpečuje rozhranie pre projekt a projektovú navigačnú lištu a zobrazenie projektov pre aktuálne prihláseného používateľa.

Technická dokumentácia

Kapitola 4: https://1drv.ms/b/s!AnORgXjFAcOEhQ_yCmfcF5irje7t

Používateľská príručka

Kapitola 3: <https://1drv.ms/b/s!AnORgXjFAcOEhQ7nvJNqzZfLQxQM>

3.2.3 SketchService

Komponent SketchService zabezpečuje vytvorenie skice a v budúcnosti aj ich modifikácie a zdieľanie vrátane zobrazovania. Taktiež zabezpečuje zloženie skice do JSON objektu, ktorý je následne odoslaný na serverovú časť aplikácie na ďalšie spracovanie.

Technická dokumentácia

Kapitola 8: https://1drv.ms/b/s!AnORgXjFAcOEhQ_yCmfcF5irje7t

Používateľská príručka

Kapitola 3.3.1: <https://1drv.ms/b/s!AnORgXjFAcOEhQ7nvJNqzZfLQxQM>

3.2.4 DrawableComponents

Komponent DrawableComponents zabezpečuje inicializáciu dostupných komponentov na základe ich šablón získaných zo serverovej časti aplikácie. Taktiež zabezpečuje nastavenie ich základných vlastností, medzi ktoré patrí napríklad farba.

Technická dokumentácia

Kapitola 8: https://1drv.ms/b/s!AnORgXjFAcOEhQ_yCmfcF5irje7t

Používateľská príručka

Kapitola 3.3.1: <https://1drv.ms/b/s!AnORgXjFAcOEhQ7nvJNqzZfLQxQM>

3.2.5 Navigation

Komponent Navigation zabezpečuje zobrazenie navigačnej lišty v hornej časti aplikácie. Taktiež zabezpečuje zobrazenie počtu aktuálne neprečítaných notifikácií a poskytuje zobrazenie okna s notifikáciami. Tiež zabezpečuje zobrazenie mena aktuálne prihláseného používateľa a zobrazenie používateľskej ponuky.

Technická dokumentácia

Kapitoly 1 a 6: https://1drv.ms/b/s!AnORgXjFAcOEhQ_yCmfcF5irje7t

Používateľská príručka

Kapitoly 1.5, 2 a 4.2: <https://1drv.ms/b/s!AnORgXjFAcOEhQ7nvJNqzZfLQxQM>

3.2.6 WebSocketHandler

Komponent WebSocketHandler zabezpečuje pripojenie na websocketový server a autentifikáciu aktuálne prihláseného používateľa pripojením autentifikačného tokenu do správy. Použitý je protokol STOMP s upraveným SockJS klientom. Tento komponent taktiež zabezpečuje subscribe na konkrétny kanál, odoslanie správ a spracovanie publishnutých dát, napr. zavolaním notifikačného komponentu.

Technická dokumentácia

Kapitola 6.3.2: https://1drv.ms/b/s!AnORgXjFAcOEhQ_yCmfcF5irje7t

Používateľská príručka

Nie je, keďže ide o iba o spracovanie dát.

3.2.7 EventHandler

Komponent EventHandler zabezpečuje broadcast a zachytenie udalostí. Tieto udalosti sú v aplikácii použité na aktualizáciu počítadla neprečítaných notifikácií a na úpravu zoznamu notifikácií.

Technická dokumentácia

Kapitola 6.3.2: https://1drv.ms/b/s!AnORgXjFAcOEhQ_yCmfcF5irje7t

Používateľská príručka

Nie je, keďže ide o iba o spracovanie dát.

3.2.8 Notifications

Komponent Notifications slúži na zobrazenie konkrétnych notifikácií s využitím knižnice ngToaster. Taktiež zabezpečuje načítanie počtu neprečítaných notifikácií, načítanie stránkovaných notifikácií a načítanie prehľadových notifikácií.

Technická dokumentácia

Kapitola 6.3.3: https://1drv.ms/b/s!AnORgXjFAcOEhQ_yCmfcF5irje7t

Používateľská príručka

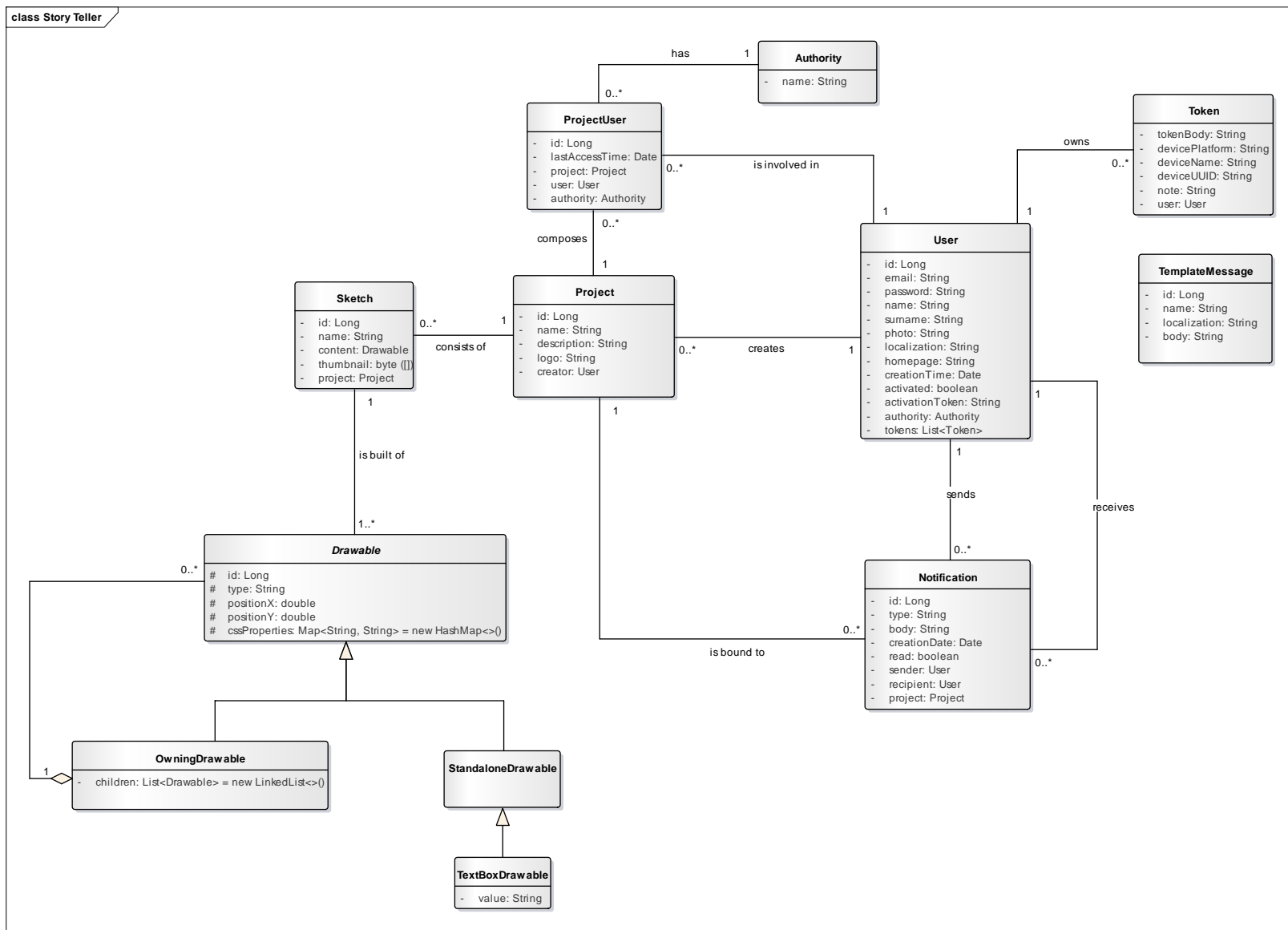
Kapitola 4: <https://1drv.ms/b/s!AnORgXjFAcOEhQ7nvJNqzZfLQxQM>

3.3 Dátový model

Dátový model aplikácie je znázornený na obrázku 2.

Opis jednotlivých tried:

- **Trieda User** predstavuje entitu používateľa vystupujúceho v systéme. Okrem základných samoopisných atribútov obsahuje zoznam autentifikačných tokenov pre všetky zariadenia, z ktorých je používateľ prihlásený.
- **Trieda Token** predstavuje entitu pre autentifikačné tokeny.
- **Trieda Authority** predstavuje konkrétnu rolu používateľa v danom projekte.
- **Trieda ProjectUser** predstavuje spájajúcu entitu obohatenú o konkrétnu rolu používateľa a čas posledného prístupu k projektu.
- **Trieda Project** reprezentuje projekt vytvorený v systéme.
- **Trieda Notification** predstavuje notifikáciu zaslanú používateľovi systémom alebo iným používateľom. Preto môže byť naviazaná vzťahom 'is bound to' s projektom a vzťahom 'sends' s používateľom.
- **Trieda TemplateMessage** predstavuje emailovú šablónu, ktorá je staticky vytvorená a uložená v databáze a pri odoslaní sa vyplní reálnymi dátami.
- **Trieda Sketch** predstavuje entitu pre skice konkrétneho projektu.
- **Trieda Drawable** predstavuje abstraktnú reprezentáciu vykresliteľného komponentu skice. Táto časť je implementovaná pomocou návrhového vzoru Composite.



Obrázok 2: Dátový model aplikácie

- **Trieda OwningDrawable** predstavuje komponent, ktorý môže obsahovať ďalšie komponenty. Príkladom takéhoto komponentu je kontajner.
- **Trieda StandaloneDrawable** predstavuje samostatný komponent, t. j. komponent, ktorý nemôže obsahovať ďalšie komponenty. Príkladom takéhoto komponentu je textové pole.
- **Trieda TextBoxDrawable** predstavuje konkrétny komponent – textové pole.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Projektová dokumentácia – moduly systému

Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller-04@googlegroups.com
Akademický rok: 2016/2017

Obsah

1	Úvod.....	1-1
2	Modul – Používateľ.....	2-1
2.1	Analýza.....	2-1
2.2	Návrh a implementácia.....	2-1
2.2.1	Registrácia.....	2-2
2.2.2	Prihlásenie.....	2-3
2.2.3	Úprava údajov v profile.....	2-3
2.2.4	Obnova hesla.....	2-4
2.2.5	Obnova registračného emailu.....	2-4
2.2.6	Správa prihlásených zariadení.....	2-4
2.2.7	Riadenie prístupu.....	2-5
2.2.8	Načítanie zoznamu používateľov v projekte.....	2-5
2.2.9	Roly.....	2-5
2.2.10	Pridanie používateľskej roly v rámci projektu.....	2-5
3	Modul – Lokalizácia.....	3-1
3.1	Analýza.....	3-1
3.2	Návrh a implementácia.....	3-1
4	Konfigurácia - Server a nasadzovanie.....	4-1
4.1	Konfigurácia.....	4-1
4.2	Backend.....	4-1
4.3	Client.....	4-1
5	Modul – Projekt.....	5-1
5.1	Analýza.....	5-1
5.2	Návrh a implementácia.....	5-1
5.2.1	Vytvorenie projektu.....	5-2
5.2.2	Načítanie zoznamu projektov prihláseného používateľa.....	5-2
5.2.3	Priradenie roly používateľovi projektu.....	5-3
5.2.4	Načítanie informácií o konkrétnom projekte.....	5-3
6	Používateľské rozhranie.....	6-1
7	Modul – Notifikácie.....	7-1
7.1	Analýza.....	7-1

7.2	Návrh a implementácia	7-1
7.2.1	Odosielanie notifikácií	7-1
7.2.2	Prijímanie notifikácií	7-1
7.2.3	Zobrazenie notifikácií	7-2
8	Modul – Skice	8-1
8.1	Analýza	8-1
8.2	Návrh a implementácia	8-1
8.2.1	Perzistencia.....	8-1
8.2.2	Vytvorenie skice.....	8-2
8.2.3	Získanie skice	8-3
8.2.4	Úprava skice	8-3
8.2.5	Odstránenie skice	8-3
8.2.6	Načítanie skíc projektu	8-3

1 Úvod

Tento dokument obsahuje popis všetkých modulov vystupujúcich v systéme. Každý modul obsahuje analýzu, návrh a implementáciu vrátane diagramu tried tam, kde je to vhodné.

2 Modul – Používateľ

Úlohou modulu používateľ má byť riadenie prístupu k jednotlivým častiam a funkciám systému. Zároveň má ponúkať používateľovi možnosť prezentovať sa verejným profilom, obnoviť prístup do systému v prípade zabudnutia alebo straty identifikačných údajov.

2.1 Analýza

Na realizáciu všetkých opísaných bodov je nutné navrhnuť a implementovať jednotlivé funkcie:

1. Registrácia
2. Prihlásenie
3. Úprava údajov v profile
4. Obnova hesla
5. Obnova registračného emailu
6. Správa prihlásených zariadení
7. Riadenie prístupu
8. Načítanie zoznamu používateľov v projekte
9. Pridanie používateľskej roly v rámci projektu

Registrácia bude prebiehať zadaním prihlasovacieho mena – emailu a prihlasovacieho hesla, podobne ako prihlásenie. Po registrácii bude používateľovi odoslaný aktivačný a deaktivovaný odkaz prostredníctvom emailovej správy. Pomocou týchto odkazov si bude môcť svoj účet aktivovať, čím sa mu umožní prihlásenie, ale aj deaktivovať, čím sa jeho účet zo systému odstráni.

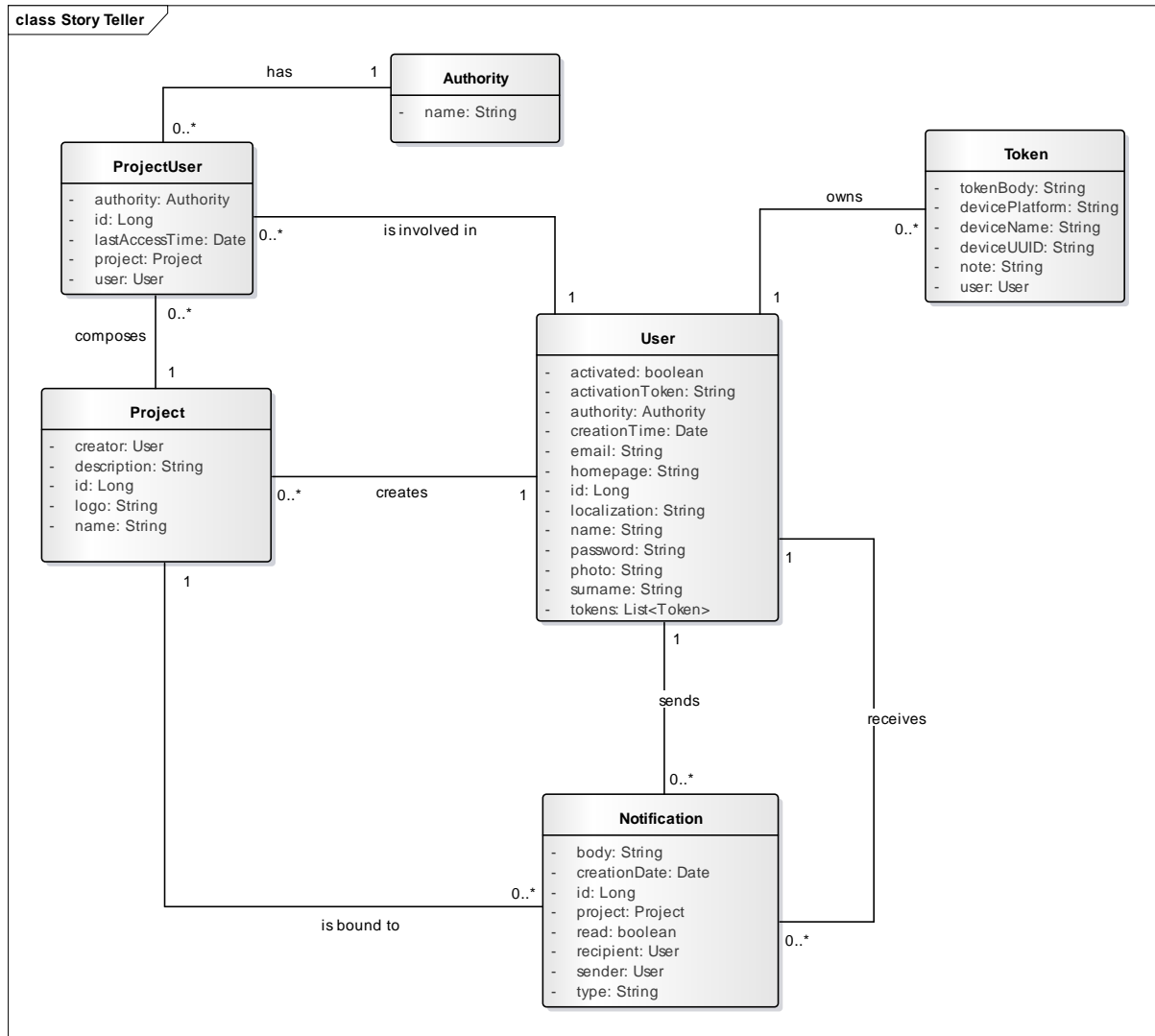
Používateľ bude mať možnosť vyplniť základné informácie o sebe vo svojom profile – meno, priezvisko, homepage a fotografiu a bude mať možnosť zmeniť si heslo. V prípade straty alebo zabudnutia hesla bude mať možnosť odoslania nového hesla na emailovú adresu, pomocou ktorej sa zaregistroval.

Riadenie prístupu bude prebiehať pomocou rolí. Používateľ si bude mať možnosť vybrať hlavnú rolu – rolu analytika alebo zákazníka, v ktorej bude v systéme vystupovať. Používateľ bude mať možnosť túto rolu kedykoľvek upraviť vo svojom profile.

2.2 Návrh a implementácia

Na perzistenciu objektu používateľa reprezentovaného triedou **User** z balíka **com.storyteller.model** bol použitý OR mapovač s použitím modulu Spring Data rovnako ako na perzistenciu zvyšných tried zobrazených v dátovom modeli na obrázku 1 – **Token** a **Authority**. Trieda **Authority** je určená na riadenie prístupu prostredníctvom rolí, v tomto prípade rolí analytika a zákazníka. Trieda **Token** reprezentuje autentifikačný token pre používateľa – teda nepoužívame HTTP relácie (z angl. sessions). Použitie tokenov zjednodušuje škálovanie aplikácie v budúcnosti, keďže nám umožňuje prechod na bezstavovú (z angl. stateless) autentifikáciu.

K základnému prístupu k údajom v databáze boli použité repozitáre (balík **com.storyteller.repository**) rozhrania, ktoré rozširujú rozhrania modulu Spring Data a umožňujú tvorbu tzv. dopytových metód (z angl. query methods), pomocou ktorých modul Spring Data dokáže vygenerovať dopyty do databázy na základe názvu metódy. Repozitáre taktiež podporujú definíciu dopytov pomocou **@Query** anotácií a JPA syntaxe.



Obrázok 1: Dátový model používateľa

Taktiež boli použité servery (z angl. services) obsahujúce biznis logiku a riadenie databázových transakcií. K dátam sa teda neprístupuje priamo cez repozitáre, ale cez servery, ktoré následne v transakciách pristupujú k dátam. Týmto je dodržaný princíp viacvrstvovej architektúry.

2.2.1 Registrácia

Registrácia bola implementovaná ako štandardný REST endpoint **/register** s **POST** metódou, ktorá sa nachádza v triede **AuthenticationController** v balíku **com.storyteller.controller**. Táto metóda využíva návratové hodnoty podľa HTTP špecifikácie so zachovaním sémantiky

a teda v prípade úspešnej registrácie vracia HTTP stav **201 – CREATED** a používateľ je uložený do databázy prostredníctvom metódy **createUser(User user)** triedy **UserService** z balíka **com.storyteller.service**. Taktiež je vykonávaná kontrola na duplicitu emailových adries. V prípade pokusu o registráciu s duplicitnou emailovou adresou je vrátený HTTP stav **400 – BAD REQUEST**. Na validovanie emailu a hesla boli použité JPA anotácie **@Email** a **@NotNull** s kombináciou anotácie **@Size(min, max)** ošetrojúcou minimálnu a maximálnu dĺžku reťazcov.

Po úspešnej registrácii je používateľovi zaslaný email prostredníctvom **zoho.com** z adresy **noreply@story-teller.xyz** obsahujúci aktivačný a deaktivovaný odkaz. Túto konfiguráciu je možné zmeniť upravením konfiguračného súboru **application.properties** v **/src/main/resources/**. Používateľ si svoj účet môže aktivovať do 12 hodín od odoslania odkazu, inak mu bude účet pri nasledujúcom pokuse o aktiváciu, resp. v pravidelnom nastavenom intervale, zrušený. Deaktivácia účtu môže prebehnúť, pokiaľ účet nie je aktivovaný. Obe operácie prebiehajú pomocou požiadavky metódou **GET** na REST endpoint **/activate/{token}**, resp. **/deactivate/{token}**.

2.2.2 Prihlásenie

Prihlásenie bolo implementované s využitím konfigurácie modulu Spring Security a servletových filtrov. Tieto filtre sa nachádzajú v balíku **com.storyteller.security** a zabezpečujú overenie autentifikačného tokenu, ktorý používateľ pri autentifikácii musí priložiť do HTTP hlavičky **X-AUTH-TOKEN** v HTTP požiadavke. Samotná konfigurácia sa nachádza v triede **SecurityConfiguration** v balíku **com.storyteller.config** spolu s triedou reprezentujúcou Spring Security servis používateľa **UserDetailsService**. Po úspešnom prihlásení odoslaním požiadavky metódou **POST** na endpoint **/login** je používateľovi vygenerovaný autentifikačný token reprezentovaný triedou **Token**, ktorý je uložený v databáze. Tento token je v klientskej časti uložený do **local storage** prehliadača, resp. zariadenia a následne pripájaný do HTTP hlavičky **X-AUTH-TOKEN** v HTTP požiadavke pomocou AngularJS interceptora **authInterceptor.js** v **/components/interceptor/**. Pri odhlásení používateľa (REST endpoint **/logout** metódou **GET**) je tento token odstránený z **local storage** ako aj z databázy.

2.2.3 Úprava údajov v profile

Úprava základných informácií prebieha odoslaním celého objektu používateľa **User** do REST endpointu **/users/current** metódou **PUT**. Po odoslaní celého objektu s informáciami sú tieto informácie overované a v prípade úspešného overenia uložené do databázy. Získanie aktuálnych dát o prihlásenom používateľovi prebieha odoslaním požiadavky metódou **GET** na REST endpoint **/users/current**. Zmena hesla prebieha odoslaním iba starého a nového hesla metódou **POST** na endpoint **/users/current/password**, pričom sa overuje zhoda starého hesla s aktuálnym heslom. V prípade zhody je heslo zmenené.

2.2.4 Obnova hesla

Obnova hesla bola implementovaná ako REST endpoint `/passwordrecovery` s POST metódou, ktorá sa nachádza v triede **AuthenticationController** v balíku **com.storyteller.controller**. V tejto metóde je vykonávaná kontrola, či zvolený email na zmenu hesla je v systéme uložený. V prípade pokusu o obnovenie hesla, kedy používateľ so zadanou emailovou adresou neexistuje, je vrátený HTTP stav **400 – BAD REQUEST**. Taktiež tento HTTP stav je vrátený v prípade, že metóda **generateNewPassword(user, email)** v triede **UserService** nevygeneruje nové heslo. Táto metóda generuje nové 10-miestne heslo použitím java **UUID.randomUUID()** utility. Nové autentifikačné údaje sú používateľovi poslané emailom z adresy `noreply@storyteller.xyz` pomocou metódy **sendPasswordRecoveryEmail(User user, String password)** v triede **MailService**.

2.2.5 Obnova registračného emailu

Systém poskytuje taktiež možnosť opätovného zaslania emailu s aktivačným a deaktiváčnym odkazom. Táto funkcionality prebieha odoslaním emailovej adresy, ktorú zadá používateľ, do REST POST metódy `/tokenRecovery` v triede **AuthenticationController**. Táto metóda najprv skontroluje, či existuje používateľ s daným emailom, metóda vracia HTTP stav **401 – UNAUTHORIZED**, ak používateľ neexistuje v databáze. Potom nasleduje kontrola aktivácie účtu. Ak bol účet používateľa s daným emailom aktivovaný, email nie je nutné poslať a metóda vráti HTTP stav **400 – BAD_REQUEST**. Nový aktivačný token sa vygeneruje v metóde **generateActivationToken(user)** v triede **UserService**, nasleduje zmena aktivačného odkazu v entite používateľa pomocou metódy **updateUserActivationToken(String email, String activationToken)** v triede **UserService** a poslanie emailu na emailovú adresu používateľa tak ako pri registrácii pomocou metódy **sendRegistrationEmail(User user)** v triede **MailService**.

2.2.6 Správa prihlásených zariadení

Správa prihlásených zariadení, ktoré sú reprezentované entitou **Token**, bola implementovaná prostredníctvom viacerých REST endpointov. Výber všetkých zariadení, cez ktoré je používateľ prihlásený v aplikácii, je realizovaný ako REST endpoint `/users/current/tokens` metódou GET v triede **UserController** v balíku **com.storyteller.controller**. Pri každom zariadení je možné modifikovať poznámku, čo je implementované ako REST endpoint `/tokens/note` metódou PUT v triede **TokenController** v balíku **com.storyteller.controller**. Táto metóda má na vstupe token zariadenia a zmenenú poznámku a prostredníctvom metódy **updateTokenNote(Token token, String newNote)** v triede **TokenService** v balíku **com.storyteller.service** je nová poznámka pre dané zariadenie uložená do databázy. Používateľ môže tiež odstrániť zariadenie zo zoznamu prihlásených zariadení, čo bolo implementované ako štandardný REST endpoint `/tokens/{token}` s **DELETE** metódou, ktorá sa nachádza v triede **TokenController** v balíku **com.storyteller.controller**. Táto metóda využíva návratové hodnoty podľa HTTP špecifikácie so zachovaním sémantiky. Zariadenie je odstránené prostredníctvom metódy **deleteByTokenBodyAndEmail(String token, String email)** triedy **TokenService** z balíka **com.storyteller.service**. V prípade neúspešného pokusu o odstránenie zariadenia je vrátený HTTP stav **400 – BAD REQUEST**.

2.2.7 Riadenie prístupu

Riadenie prístupu k údajom v databáze je realizované pomocou prístupových anotácií **@PreAuthorize**. Tie sú použité pri každom REST endpointe, ktorý prístupuje ku kritickým dátam, špecifickým pre konkrétnych používateľov. Autorizačné anotácie majú tvar : **@PreAuthorize("@roleSecurityService.authorizationMethod(#arg1,'constant1'))**.

@roleSecurityService reprezentuje triedu, v ktorej sú umiestnené autorizačné metódy. **AuthorizationMethod** reprezentuje metódu, na základe ktorej je rozhodnuté, či používateľ má prístup k daným údajom. Autorizačné metódy sú definované v triede **RoleSecurityService**, v balíku **com.storyteller.security**. Autorizačná metóda môže prijať argumenty ako **#arg1**, čo je premenná ako napríklad id projektu, id používateľa. Konštanty ako **'constant1'** sú využívané najmä pre roly, keď priradíme REST endpointu konkrétnu rolu, pre ktorú je prístupný.

2.2.8 Načítanie zoznamu používateľov v projekte

Získanie zoradených používateľov prislúchajúcich aktuálne prehliadanému projektu bolo implementované ako štandardný REST endpoint **/projects/{projectId}** s **GET** metódou, ktorá sa nachádza v triede **UserController** v balíku **com.storyteller.controller**. Táto metóda využíva návratové hodnoty podľa HTTP špecifikácie so zachovaním sémantiky, a teda vracia HTTP stav **OK**, ak sú používatelia pre aktuálne prehliadaný projekt úspešne načítaní prostredníctvom metódy **getUsersForProjectId(Long projectId)** triedy **UserService** z balíka **com.storyteller.service**. Z databázy je najskôr vybratý projekt pomocou metódy **findOne(Long projectId)**, z ktorého je následne vybratý a zoradený zoznam jeho používateľov. V prípade neúspešného pokusu o načítanie používateľov je vrátený HTTP stav **400 – BAD REQUEST**.

2.2.9 Roly

V rámci projektu boli rozpoznané viaceré typy rolí. Ide o tieto roly: zákazník, používateľ, analytik a administrátor.

- Zákazník – rola zákazníka umožňuje prezeranie akceptačných testov a skíc a pridávanie ľudí do projektu, resp. pridávanie ich rolí
- Analytik – rola analytika umožňuje modifikovanie, prezeranie alebo odstránenie akceptačných testov a skíc a vytváranie nového projektu.
- Používateľ – rola charakteristická pre všeobecného používateľa a opisuje spoločné vlastnosti nadtriedy
- Administrátor – rola administrátora určuje špecifické vlastnosti, ktoré môže vykonávať jedine administrátor

2.2.10 Pridanie používateľskej roly v rámci projektu

Zmena používateľskej roly v rámci projektu bola implementovaná ako štandardný REST endpoint **/users/{userId}/authority** s **POST** metódou, ktorá sa nachádza v triede **UserController** v balíku **com.storyteller.controller**. Táto metóda využíva návratové hodnoty podľa HTTP špecifikácie so zachovaním sémantiky, a teda vracia HTTP stav **OK**, ak je rola príslušného používateľa úspešne aktualizovaná prostredníctvom metódy

updateAuthority(ProjectUser projectUser, String authority) triedy **UserService** z balíka **com.storyteller.service**. Používateľ príslušného projektu je z databázy vybratý pomocou metódy **findByUserIdAndProjectId(Long userId, Long projectId)** triedy **ProjectUserRepository** z balíka **com.storyteller.repository**. V prípade neúspešného pokusu o načítanie používateľa alebo pridania jeho roly je vrátený HTTP stav **400 – BAD REQUEST**.

3 Modul – Lokalizácia

Úlohou modulu lokalizácia je umožnenie nastavenia a vytvorenie prekladu celého systému bez zásahu do funkcionality systému

3.1 Analýza

Na lokalizáciu je možné použiť funkcie jazyka Java a prekladový modul ng-translate pre AngularJS. Samotné preklady je možné ukladať v properties súboroch s názvom v tvare **messages_LOKALIZÁCIA.properties**, ktorý obsahuje preklady v tvare HODNOTA=KĽÚČ.

3.2 Návrh a implementácia

Na implementáciu lokalizácií sme použili prístup popísaný v návrhu. Súbory properties s prekladmi sú umiestnené v **/src/main/resources** a ich obsah si klient preberá cez REST endpoint **/language?lang={localization}** GET metódou, ktorá je definovaná v triede **ResourceBundleController** v balíku **com.storyteller.controller**. Táto metóda na základe vyžiadaného kódu lokalizácie – napr. sk alebo en, načíta obsah daného properties súboru (v prípade sk lokalizácie sa načíta súbor **messages_sk.properties**, vid' ukážka 1) do objektu typu **Properties**, ktorý je následne serializovaný na JSON a spracovaný modulom ng-translate v klientskej AngularJS aplikácii.

```
myprofile.success.login=You were logged in successfully
myprofile.password.old=Old password
myprofile.password.new=New password
myprofile.password.new.confirm=Confirm new password
myprofile.update=Update!
myprofile.password.update=Change password!
myprofile.password.old.required=Old password is required
myprofile.password.new.required=New password is required
```

Ukážka 1: Časť súboru `messages_en.properties`

4 Konfigurácia - Server a nasadzovanie

4.1 Konfigurácia

Náš projekt je založený na architektonickom vzore klient-server, a z toho dôvodu je rozdelený na dva podprojekty. Prvým je projekt **backend**, ktorý sa stará o aplikačnú logiku a perzistenciu. Druhým je projekt **client**, ktorý sa stará o vykresľovanie obsahu a poskytuje užívateľovi grafické rozhranie. Oba projekty sú nasadené na servery **Ubuntu Server 16.04**.

Náš server poskytuje tieto domény :

- story-teller.xyz – **tímová stránka o projekte**
- api.story-teller.xyz – **backend aplikácia**
- app.story-teller.xzy – **client aplikácia**

4.2 Backend

Projekt backend je založený na **Spring framework-u**, a teda je nasadený na aplikačnom serveri **Apache Tomcat**. Tomcat umožňuje nasadzovanie priamo pomocou URL, to znamená, že priamo po vykonaní buildu je nová verzia nasadená.

4.3 Client

Client je založený na frameworku **Ionic**, ktorý umožňuje zostaviť aplikácie pre mobilné zariadenia, rovnako ako umiestnenie priamo na webe. Client je nasadzovaný pomocou **SCP** kopírovania a následného spustenia skriptu. Skript sa postará o rozbalenie a nahradenie už existujúcich súborov. Ionic poskytuje službu, ktorá spustí aplikáciu ako webovú stránku, na ktorú sa dá následne pristupovať na základe zvoleného mena.

5 Modul – Projekt

Úlohou modulu projekt má byť možnosť vytvárania nových projektov a spravovania existujúcich projektov prihláseným používateľom.

5.1 Analýza

Na realizáciu všetkých opísaných bodov je nutné navrhnúť jednotlivé funkcie:

1. Vytvorenie nového projektu
2. Načítanie zoznamu projektov prihláseného používateľa
3. Načítanie informácií o konkrétnom projekte

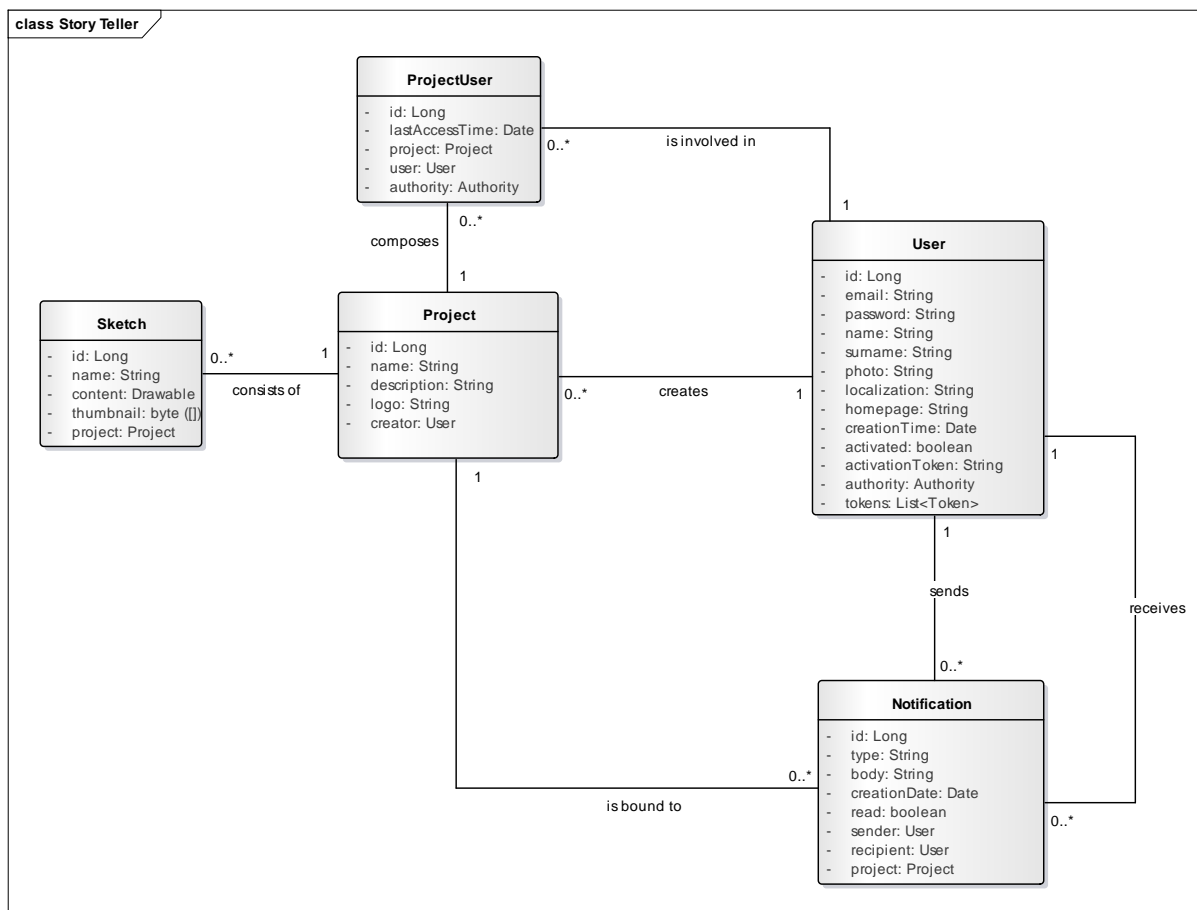
Možnosť vytvárať projekty bude sprístupnená prihláseným používateľom. Pri vytváraní projektu bude používateľ vyplňať príslušný formulár. Vo formulári bude potrebné zadať názov projektu, opis projektu a prípadne vlastné logo projektu. Následne po odoslaní formulára na spracovanie sa k projektu priradí jeho tvorca – čiže automaticky je tomuto projektu priradený ďalší atribút vyjadrujúci údaje o používateľovi, ktorý projekt vytvoril.

5.2 Návrh a implementácia

Na perzistenciu objektu projektu reprezentovaného triedou **Project** z balíka **com.storyteller.model** bol použitý OR mapovač s použitím modulu Spring Data.

K základnému prístupu k údajom v databáze boli použité repozitáre (balík **com.storyteller.repository**) rozhrania, ktoré rozširujú rozhrania modulu Spring Data a umožňujú tvorbu tzv. dopytových metód (z angl. query methods), pomocou ktorých modul Spring Data dokáže vygenerovať dopyty do databázy na základe názvu metódy. Repozitáre taktiež podporujú definíciu dopytov pomocou **@Query** anotácií a JPA syntaxe.

Taktiež boli použité servisy (z angl. services) obsahujúce biznis logiku a riadenie databázových transakcií. K dátam sa teda neprístupuje priamo cez repozitáre, ale cez servisy, ktoré následne v transakciách pristupujú k dátam. Týmto je dodržaný princíp viacvrstvovej architektúry.



Obrázok 2: Dátový model projektu

5.2.1 Vytvorenie projektu

Vytvorenie projektu bolo implementované ako štandardný REST endpoint / s **POST** metódou, ktorá sa nachádza v triede **ProjectController** v balíku **com.storyteller.controller**. Táto metóda využíva návratové hodnoty podľa HTTP špecifikácie so zachovaním sémantiky a teda v prípade úspešného vytvorenia projektu vracia HTTP stav **OK** a projekt je uložený do databázy prostredníctvom metódy **createProject(Project project, String email)** triedy **ProjectService** z balíka **com.storyteller.service**. Projektu sa priradí jeho tvorca na základe emailovej adresy prihláseného používateľa. V prípade neúspešného pokusu o vytvorenie projektu je vrátený HTTP stav **400 – BAD REQUEST**. Na validovanie názvu a opisu projektu bola použitá JPA anotácia **@NotNull**.

5.2.2 Načítanie zoznamu projektov prihláseného používateľa

Získanie zoradených projektov prislúchajúcich aktuálne prihlásenému používateľovi bolo implementované ako štandardný REST endpoint **/projects/{pageIndex}/{pageSize}** s **GET** metódou, ktorá sa nachádza v triede **ProjectController** v balíku **com.storyteller.controller**. Táto metóda využíva návratové hodnoty podľa HTTP špecifikácie so zachovaním sémantiky, a teda vracia HTTP stav **OK**, ak sú projekty pre aktuálne prihláseného používateľa úspešne načítané prostredníctvom metódy **getProjectsForUser(String email, Pageable pageable)** triedy **ProjectService** z balíka **com.storyteller.service**. Projekty sú z databázy vybraté

pomocou metódy **findByUserEmailOrderByProjectName(String email, Pageable pageable)** triedy **ProjectUserRepository** z balíka **com.storyteller.repository**. V prípade neúspešného pokusu o načítanie projektov je vrátený HTTP stav **400 – BAD REQUEST**.

5.2.3 Priradenie roly používateľovi projektu

Riadenie prístupu ku projektu je založené na rolách. Tie sú používateľovi priradené pomocou REST endpointov, ktoré najprv načítajú všetky dostupné roly, a následne vyznačia aktuálne priradenú. Načítanie všetkých rolý je možné cez **GET** endpoint **/authority/all** a načítanie aktuálnej roly cez **GET** endpoint **/authority/{userId}** spolu s parametrom **projectId**. Tieto REST endpointy sú dostupné v triede **AuthorityController** v balíku **com.storyteller.controller**. Tieto metódy využívajú návratové hodnoty podľa HTTP špecifikácie so zachovaním sémantiky, a teda vracia HTTP stav **OK**, ak sú roly úspešne načítané. Na základe filtrov Roly sú z databázy vyťahované prostredníctvom triedy **AuthorityRepository**. Tá pristupuje k entite **Authority**.

5.2.4 Načítanie informácií o konkrétnom projekte

Načítanie konkrétneho projektu je realizované načítaním jeho objektu podľa identifikátora. V triede **ProjectController** v balíku **com.storyteller.controller** je implementovaný štandardný REST endpoint **/projects/{projectId}**, kde **projectId** je identifikátor projektu. Tento endpoint volá metódu **getProjectById(Long projectId)** triedy **ProjectService** z balíka **com.storyteller.service**. Na strane klienta je tento endpoint volaný už pri zmene obrazovky (stavu) na obrazovku s prehľadom informácií o konkrétnom projekte.

6 Používateľské rozhranie

Používateľské rozhranie je realizované v projekte **client**. Ten je založený na technológii **Ionic**, ktorá umožňuje zostavovanie mobilných aplikácií. Framework Ionic ponúka vlastné elementy a štýly na použitie, tie však nie sú príliš vhodné pre webovú stránku. Z toho dôvodu je použitý mobile-first framework **Bootstrap**. V tomto projekte je použitá knižnica **Angular-UI**, nakoľko Ionic využíva AngularJS, ktorý by mohol spôsobovať konflikty s klasickým jQuery.

Bootstrap je responzívny framework, umožňujúci efektívne zobrazenie webovej stránky na rôznych zariadeniach, s rôznym rozlíšením. Dokáže automaticky prispôsobiť rozloženie grafických elementov vzhľadom na aktuálnu veľkosť obrazovky. Bootstrap využíva triedy na špecifikáciu štýlu pre príslušný element. V projekte sú využité len štandardné triedy, ktoré využíva bootstrap. Všetky ostatné štýly ako odsadenie, posunutie, a iné, ktoré sú špecifické pre konkrétny element sú definované priamo v HTML. Bootstrap štýly sú definované v súbore **custom.css** a následne minifikované v súbore **custom.min.css**. Iba minifikovaný súbor je importovaný do projektu, a teda pri každej zmene je nutné súbor znovu minifikovať.

Bootstrap dokumentácia : <http://getbootstrap.com/getting-started/http://getbootstrap.com/getting-started/>

7 Modul – Notifikácie

Úlohou modulu notifikácie je poskytnutie dôležitých informácií používateľovi v reálnom čase, bez nutnosti manuálne vyžiadania.

7.1 Analýza

Na prenos notifikácií je možné použiť websockety s modelom publish – subscribe. Z dôvodu viacerých typov notifikácií je nutné navrhnuť šablónový podmodul, ktorý umožní dynamické nastavenie šablóny konkrétnej notifikácie v závislosti od jej typu, napr. či daná notifikácia bude poskytovať možnosť potvrdenia. Použitie websocketov taktiež umožňuje budúcu implementáciu chatovacieho modulu a modulu paralelného upravovania skíc.

7.2 Návrh a implementácia

Samotná notifikácia je reprezentovaná pomocou triedy **Notification** z balíka **com.storyteller.model**. Databázové operácie nad ňou sú realizované prostredníctvom repozitárov a servisov, podobne ako je to v prípade modulu používateľa. Prístup k notifikáciám z klientskej aplikácie je realizovaný prostredníctvom REST endpointov definovaných v triede **NotificationController** z balíka **com.storyteller.controller**.

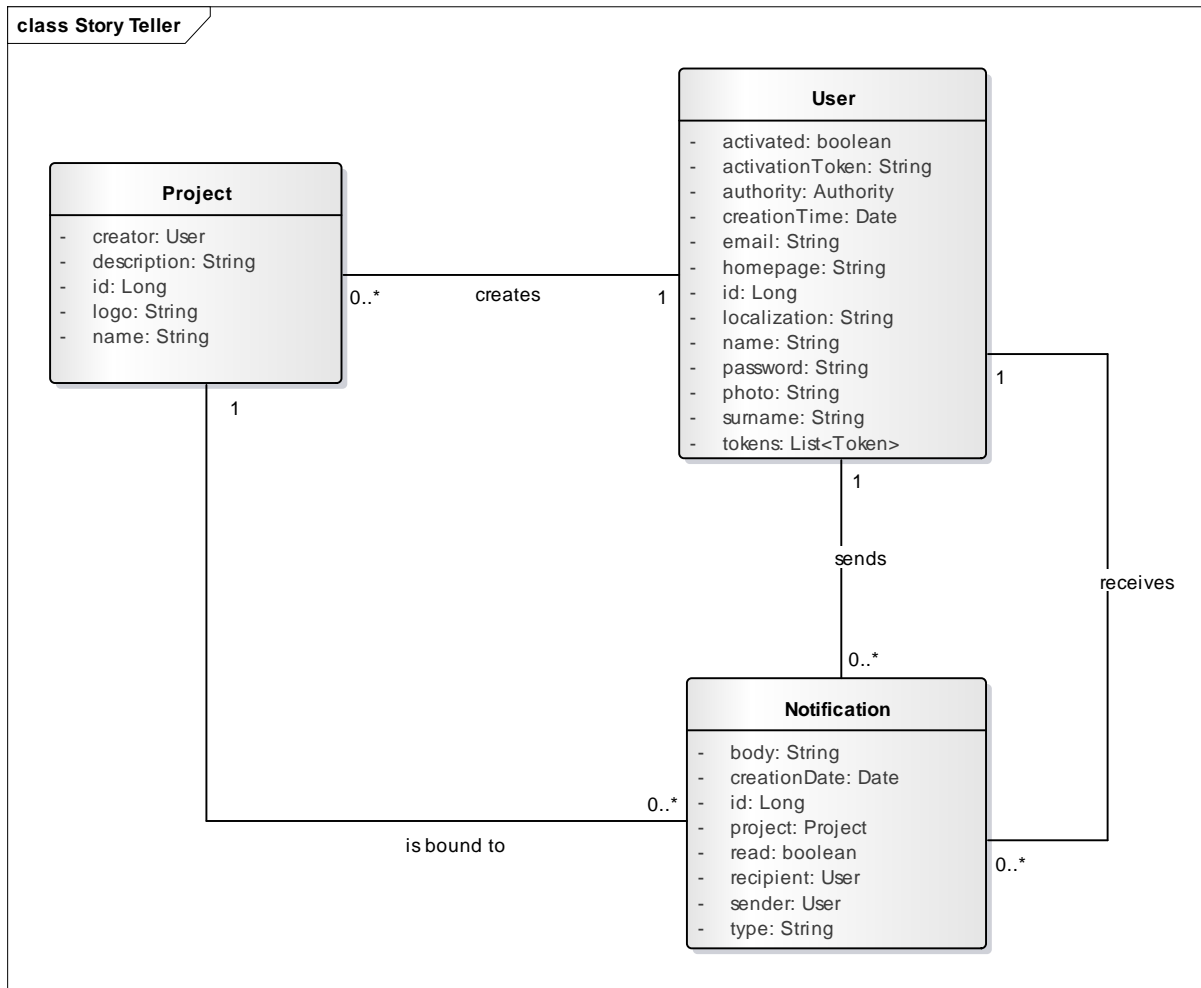
7.2.1 Odosielanie notifikácií

Notifikácie sú odosielané z backendovej strany projektu pomocou protokolu **STOMP**, ktorý je podporovaný rámcom Spring, ktorý používame. Notifikácie sú odosielané prostredníctvom volania metódy **sendNotification(Notification notification)**, do ktorej vstupuje už vyplnený objekt notifikácie vrátane príjemcu. Táto metóda volá Spring metódu **convertAndSendToUser(String user, String destination, Object payload)**. Argument **user** predstavuje identifikátor používateľa, v našom prípade jeho email. Argument **destination** predstavuje cestu websocketového publish kanálu, do ktorého sa má notifikácia zverejniť. Táto cesta musí obsahovať prefix **'/topic'** z dôvodu odlíšenia publish ciest od subscribe. Argument **payload** predstavuje samotný obsah správy, t. j. notifikáciu. Táto metóda je volaná prostredníctvom objektu **SimpMessageSendingOperations**, ktorý je súčasťou Spring modulu pre websockety. Autentifikácia používateľa pre websockety je realizovaná pomocou objektu **AuthenticationPrincipal**, ktorý je vytvorený pred samotným odoslaním správy do konkrétneho kanála, t. j. po prijatí dispatcherom.

7.2.2 Prijímanie notifikácií

Notifikácie sú prijímané s využitím knižnice **SockJS** a javascriptového modulu pre protokol **STOMP**. Knižnicu **SockJS** bolo potrebné upraviť z dôvodu **CORS** mechanizmu – knižnica neposkytuje možnosť vypnúť atribút **withCredentials** pre **XMLHttpRequest** volania a rovnako sa neriadi globálnymi nastaveniami rámca AngularJS. Bolo preto nutné nastaviť atribút **withCredentials** na **false**. Prijímanie notifikácií zabezpečuje subscribe na kanál **'/user/topic/notifications'** v metóde **onConnect(frame)** komponentu **socketService**. Tento

komponent predstavuje wrapper nad knižnicou **SockJS**, ktorý bolo nutné vytvoriť z dôvodu chýbajúcej podpory autentifikácie prostredníctvom tokenu.



Obrázok 3: Dátový model notifikácií

Po prijatí správy je notifikácia pretransformovaná z JSON reťazca na objekt, následne je zobrazená a je broadcastnutý event '**\$newNotification**', čo umožňuje spracovanie notifikácie aj ďalším modulom, ako je napríklad navigačný modul. Tento modul inkrementuje počítadlo zatiaľ nevidených notifikácií v navigačnom paneli nachádzajúcom sa na vrchu aplikácie a indikuje novú notifikáciu animáciou zatrasenia ikony notifikácií.

7.2.3 Zobrazenie notifikácií

Notifikácie sú zobrazované ako **toast popup** prostredníctvom modulu **ngToaster**. Na zobrazenie sa používa metóda **showNotification(notification)** komponentu **notificationService**, do ktorej ako argument prichádza vyplnená notifikácia určená na zobrazenie.

Taktiež je možné prezerat' notifikácie cez otváracie popup okno, ktoré sa zobrazí kliknutím na ikonu notifikácií v navigačnom paneli – volanie metódy **showPreviewNotifications()** z komponentu **navigationController**. Táto ikona taktiež zobrazuje počet neprečítaných notifikácií, čo je realizované volaním metódy

getUnreadNotificationsCount() z komponentu **navigationController**. Po kliknutí na ikonu sa zobrazí päť najnovších notifikácií, ktoré sú načítané volaním metódy **getBriefNotifications()** z komponentu **navigationController**.

K starším notifikáciám je možné pristupovať cez tlačidlá pre stránkovanie. Stránkovanie je realizované volaním metódy **getNotificationPage(page)** z komponentu **navigationController** do ktorej ako argument vstupuje číslo strany, ktorá sa má zobrazit'.

Označovanie notifikácií za videné prebieha tak, že notifikácie, ktoré boli zobrazené cez otváracie popup okno sú odoslané backendovej aplikácii, ktorá im zmení príznak **read** na true. Toto je realizované metódou **setNotificationsAreRead()** z komponentu **navigationController**. Po úspešnom odoslaní je tento príznak nastavený aj daným notifikáciám v klientskej časti a počítadlo neprečítaných notifikácií je aktualizované.

8 Modul – Skice

Možnosť vytvárať skice je jednou z najdôležitejších funkcionalít v tomto projekte. Pomocou skíc je možné prototypovať používateľské rozhranie, na základe ktorého budú neskôr generované akceptačné testy.

Používateľské rozhranie na webových stránkach býva zložené z viacerých komponentov, ktoré slúžia ako ovládacie prvky, alebo ako tzv. kontajnery, do ktorých sa potom vkladajú iné prvky.

Dôležitým aspektom pri skladaní rozhrania pomocou komponentov je určovanie pozícií jednotlivých prvkov. Prvky môžu mať absolútnu pozíciu, kde ich pozícia je počítaná od kraja plátna, na ktorom sú umiestnené, alebo môžu mať relatívne pozície, pri ktorých sa pozície počítajú vzhľadom na 'rodiča'.

8.1 Analýza

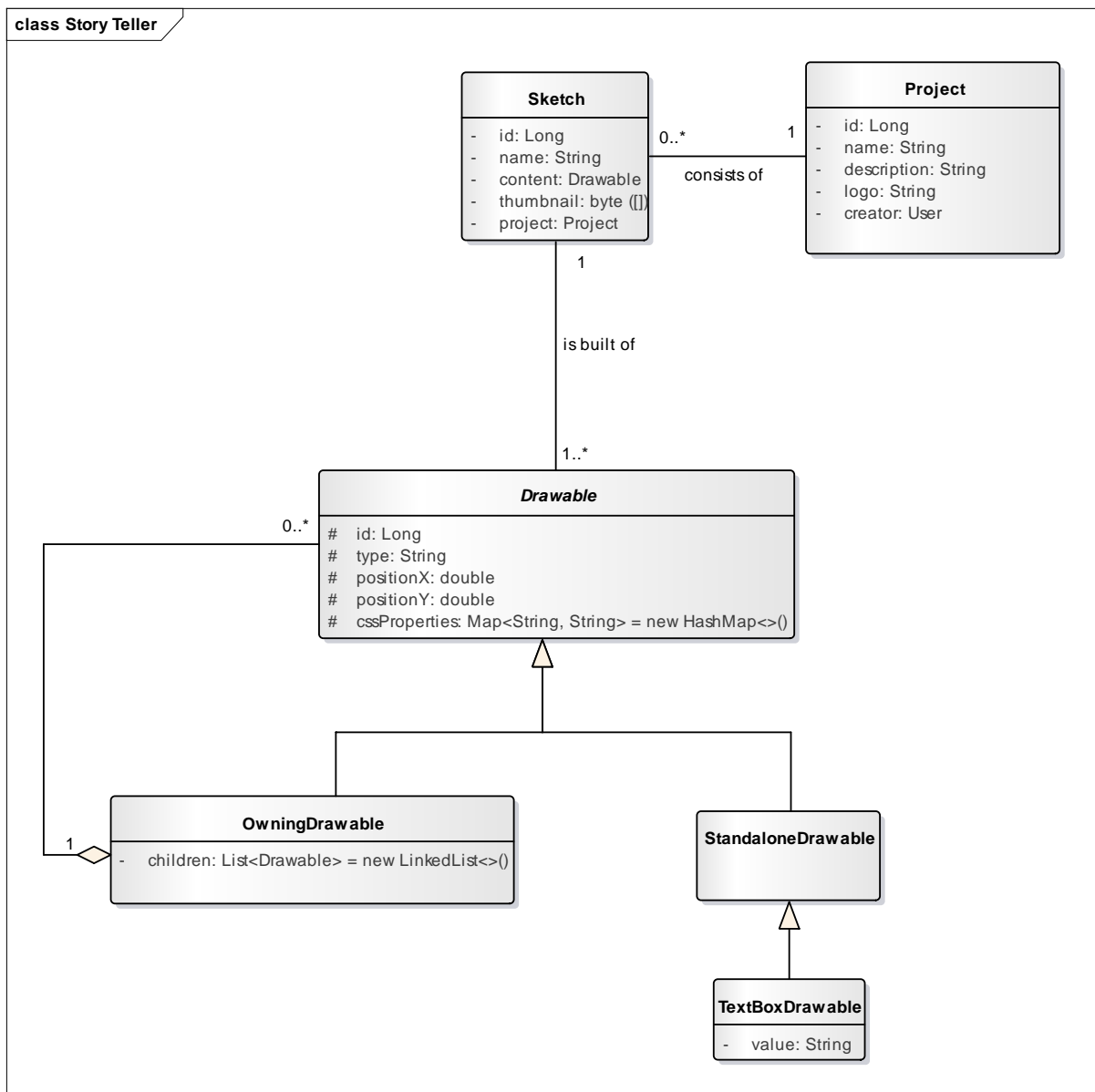
Pre možnosť tvorby skice je nutné vytvoriť používateľské rozhranie, ktoré bude poskytovať plátno a panel nástrojov, z ktorého sa budú môcť pridávať prvky na plátno.

Prvky používateľského rozhrania majú vo všeobecnosti veľkú časť vlastností spoločných. Ide napríklad o ich pozíciu na plátne, farbu, veľkosť, a mnoho iných. Najefektívnejším spôsobom je vytvoriť všeobecnú abstraktnú triedu, ktorá bude tieto vlastnosti obsahovať, a konkrétne prvky používateľského rozhrania budú od tejto abstraktnej triedy dediť, pričom ich vlastné atribúty budú len tie, ktoré sú špecifické pre konkrétny prvok.

8.2 Návrh a implementácia

8.2.1 Perzistencia

Dátový model skice a jej komponentov je zobrazený na obrázku 4. Trieda **Sketch** predstavuje samotnú skicu, pričom atribút **thumbnail** predstavuje generovaný náhľad a atribút **content** jej obsah, t. j. zloženie z komponentov. Komponenty skice sú navrhnuté ako vzor composite, pričom komponent predstavuje abstraktná trieda **Drawable**. Od tejto triedy dedia triedy **OwningDrawable**, ktorá predstavuje komponent, ktorý môže mať ďalšie vnorené prvky, čo v kontexte vzoru composite predstavuje časť composite. Príkladom komponentu, ktorý dedí od tejto triedy, je komponent kontajner. Ďalšou dediacou triedou je trieda **StandaloneDrawable**, ktorá predstavuje samostatný komponent, t. j. komponent, ktorý nemôže obsahovať ďalšie vnorené komponenty. Príkladom je komponent textové pole, ktorý je reprezentovaný triedou **TextBoxDrawable** so špecifickým atribútom **value**, čiže jeho hodnotou. Všetky tieto triedy obsahujú metódu na vygenerovanie inline CSS s názvom **getInlineCss()**, ktorá vracia inline CSS ako reťazec na základe obsahu poľa **cssProperties**. Taktiež obsahujú metódu **toHtml()**, ktorá vracia html reprezentáciu jednotlivého prvku ako reťazec, čo znamená, že v prípade zavolania tejto metódy na inštanciu triedy **Drawable** s typom (pole type) **ROOT_CONTAINER** sa vytvorí html reprezentácia celej skice. Takéto volanie je použité napríklad pri generovaní náhľadu skice.



Obrázok 4: Dátový model skice a komponentov

8.2.2 Vytvorenie skice

Vytvorenie skice bolo implementované ako štandardný REST endpoint `/project/{projectId}/sketches` s **POST** metódou, ktorá sa nachádza v triede **SketchController** v balíku **com.storyteller.controller**. Z klienta sú poslané identifikátor projektu a vytvorená skica so všetkými komponentami, ktorú treba uložiť. V prípade úspešného vytvorenia skice metóda vracia HTTP stav **OK** a vytvorenú skicu. Skica a jej komponenty sú uložené do databázy prostredníctvom metódy **createSketch(Long projectId, Sketch newSketch, boolean createThumbnail)** triedy **SketchService** v balíku **com.storyteller.service** a projektu je priradená vytvorená skica. V prípade neúspešného pokusu o uloženie skice je vrátený HTTP stav **500 – INTERNAL SERVER ERROR**. Súčasťou vytvorenia skice je automatické generovanie náhľadu vo forme obrázku. Na to je použitá knižnica **wkhtmltopdf**, resp. jej časť **wkhtmltoimage**. Keďže táto knižnica nie je priamo kompatibilná s Javou, bol vytvorený (upravený) wrapper, ktorý túto knižnicu volá prostredníctvom vytvorenia nového procesu

a dočasného html súboru so skicou, ktorý je špecifikovaný ako prvý argument volania príkazu **wkhtmltoimage**. Výstup (vyrenderovaný obrázok) je presmerovaný cez deskriptor do bajtového poľa v Jave.

8.2.3 Získanie skice

Získanie skice podľa jej identifikátora bolo implementované ako štandardný REST endpoint `/sketches/{sketchId}` s **GET** metódou, ktorá sa nachádza v triede **SketchController** v balíku **com.storyteller.controller**. Táto metóda vracia HTTP stav **OK** a skicu s daným identifikátorom, ktorú načíta z databázy pomocou metódy **getSketchById(Long sketchId)** triedy **SketchService** v balíku **com.storyteller.service**.

8.2.4 Úprava skice

Úprava skice prebieha odoslaním celého objektu skice a identifikátora skice do REST endpointu `/sketches/{sketchId}` metódou **PUT**. V tejto metóde sa najprv získa skica na základe jej identifikátora pomocou metódy **getSketchById(Long sketchId)** triedy **SketchService** v balíku **com.storyteller.service**. V prípade, že sa nenájde skica s daným identifikátorom, metóda **PUT** vracia HTTP stav **404 - NOT FOUND**. Inak sa informácie o skici upravia v metóde **updateSketch(Long sketchId, Sketch newSketchInfo, boolean createThumbnail)** triedy **SketchService** v balíku **com.storyteller.service** a metóda vracia HTTP stav **OK** a upravenú skicu. V prípade neúspešného pokusu o úpravu skice je vrátený HTTP stav **500 – INTERNAL SERVER ERROR**.

8.2.5 Odstránenie skice

Systém poskytuje tiež možnosť odstránenia skice, čo bolo implementované ako štandardný REST endpoint `/sketches/{sketchId}` s **DELETE** metódou, ktorá sa nachádza v triede **SketchController** v balíku **com.storyteller.controller**. V tejto metóde sa najprv získa skica na základe jej identifikátora pomocou metódy **getSketchById(Long sketchId)** triedy **SketchService** v balíku **com.storyteller.service**. V prípade, že sa nenájde skica s daným identifikátorom, metóda **DELETE** vracia HTTP stav **404 - NOT FOUND**. Inak je skica odstránená prostredníctvom metódy **deleteSketch(Long sketchId)** triedy **SketchService** z balíka **com.storyteller.service**. V prípade neúspešného pokusu o odstránenie skice je vrátený HTTP stav **400 – BAD REQUEST**.

8.2.6 Načítanie skíc projektu

Načítanie skíc projektu bolo implementované ako štandardný REST endpoint `/sketches/{projectId}` s **GET** metódou, ktorá sa nachádza v triede **SketchController** v balíku **com.storyteller.controller**. V tejto metóde sa získa zoznam skíc na základe identifikátora projektu pomocou metódy **getSketchesForProjectId(Long projectId)** triedy **SketchService** v balíku **com.storyteller.service**. V prípade, že sa zoznam skíc pre daný identifikátor projektu nenájde, metóda **GET** vracia http stav **400 – BAD REQUEST**. Inak metóda vráti zoznam načítaných skíc pre vybraný projekt.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Projektová dokumentácia – používateľská príručka

Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller@googlegroups.com
Akademický rok: 2016/2017



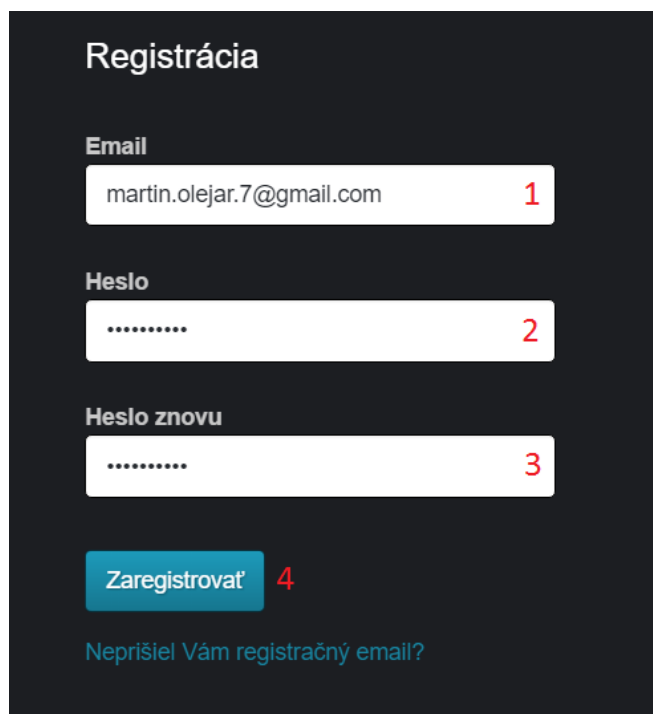
Obsah

1	Registrácia a prihlásenie používateľa.....	1-1
1.1	Registrácia používateľa	1-1
1.2	Prihlásenie používateľa.....	1-2
1.3	Obnovenie registračného emailu	1-2
1.4	Obnovenie hesla	1-3
1.5	Odhlásenie používateľa	1-4
2	Úprava informácií o profile.....	2-1
2.1	Správa zariadení	2-2
3	Vytvorenie a správa projektov	3-1
3.1	Vytvorenie projektu	3-1
3.2	Zoznam projektov	3-2
3.3	Správa projektu.....	3-2
3.3.1	Správa skíc	3-3
3.3.2	Správa používateľov.....	3-4
4	Notifikácie	4-1
4.1	Zobrazenie notifikácie – toast popup.....	4-1
4.2	Zobrazenie notifikácie – navigačný panel	4-1

1 Registrácia a prihlásenie používateľa

1.1 Registrácia používateľa

Každý používateľ sa musí pre prístup do systému zaregistrovať. Pre registráciu zadajte v registračnom formulári na obr. 1 platnú emailovú adresu (1) a heslo (2), ktoré musí mať aspoň 6 znakov. Zvolené heslo treba ešte raz potvrdiť (3) a kliknúť na tlačidlo **Zaregistrovať** (4).



Obrázok 1: Registračný formulár

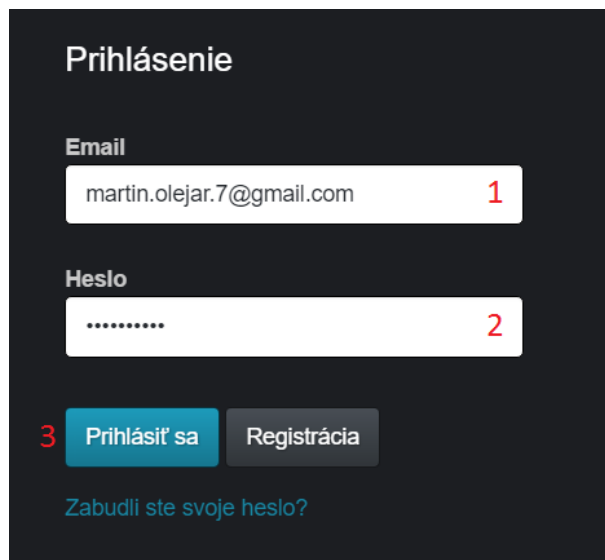
Po zadaní platných údajov je poslaný registračný email na zadanú emailovú adresu. Obsah tohto emailu je na obr. 2. V tomto emailu sa nachádza odkaz na aktiváciu účtu (1) a odkaz na deaktiváciu účtu (2). Pre úspešné ukončenie procesu registrácie kliknite na odkaz na aktiváciu účtu. Odkaz na deaktiváciu účtu slúži na zrušenie vášho profilu.



Obrázok 2: Registračný email

1.2 Prihlásenie používateľa

Zaregistrovaný používateľ sa môže prihlásiť do systému prostredníctvom prihlasovacieho formulára. Pre úspešné prihlásenie vyplňte vo formulári na obr. 3 emailovú adresu (1) a heslo (2) zadané pri registrácii. Potom kliknite na tlačidlo **Prihlásiť sa** (3).

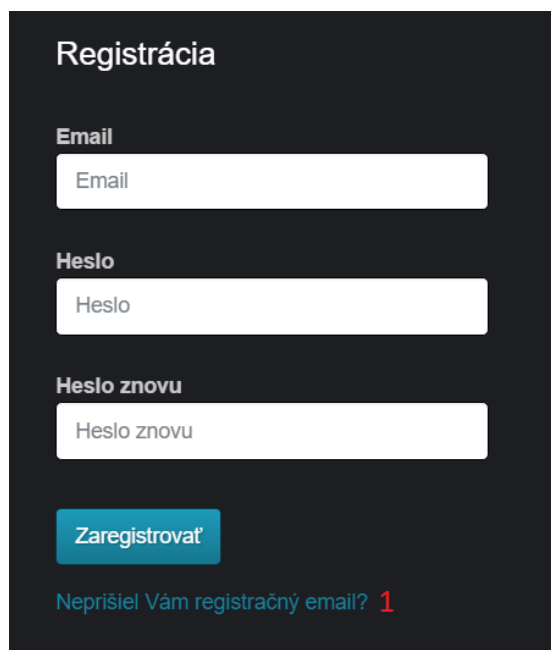


The image shows a login form titled "Prihlásenie" on a dark background. It contains two input fields: "Email" with the value "martin.olejar.7@gmail.com" and "Heslo" with masked characters. Below the fields are two buttons: "Prihlásiť sa" (highlighted with a red '3') and "Registrácia". A link "Zabudli ste svoje heslo?" is located below the buttons. Red numbers 1, 2, and 3 are placed to the right of the email field, the password field, and the "Prihlásiť sa" button respectively.

Obrázok 3: Prihlasovací formulár

1.3 Obnovenie registračného emailu

V prípade, že vám nepríde registračný email na zadanú emailovú adresu pri registrácii, je tu možnosť jeho znovu poslania. Prvým krokom je kliknúť na vyznačený odkaz (1) v časti Registrácia na obr. 4.



The image shows a registration form titled "Registrácia" on a dark background. It contains three input fields: "Email", "Heslo", and "Heslo znovu". Below the fields is a "Zaregistrovať" button. At the bottom, there is a link "Neprišiel Vám registračný email?" with a red number '1' next to it.

Obrázok 4: Registračný formulár



Následne sa zobrazí formulár na obnovenie registračného emailu na obr. 5, v ktorom vyplňte svoju emailovú adresu (1) zadanú pri registrácii. Potom kliknite na tlačidlo **Poslať email** (2). Na vašu emailovú adresu vám príde registračný email ako pri registrácii, v ktorom treba kliknúť na aktivačný odkaz.

Obnovenie registračného emailu

Email

martin.olejar.7@gmail.com 1

Poslať email 2

Obrázok 5: Formulár pre obnovenie registračného emailu

1.4 Obnovenie hesla

Ak ste zabudli svoje heslo do systému, môžeme vám vygenerovať nové heslo, ktoré si môžete v nastavení profilu neskôr zmeniť. Najprv kliknite na odkaz (1) v časti Prihlásenie na obr. 6.

Prihlásenie

Email

Email

Heslo

Heslo

Prihlásiť sa Registrácia

Zabudli ste svoje heslo? 1

Obrázok 6: Prihlasovací formulár

Následne sa zobrazí formulár pre obnovenie hesla na obr. 7, v ktorom vyplňte svoju emailovú adresu (1) potrebnú na prihlásenie. Potom kliknite na tlačidlo **Vygenerovať nové heslo** (2). Na vašu emailovú adresu vám príde email, v ktorom bude vaše nové vygenerované heslo pre vstup do systému.



Obnova hesla

Email

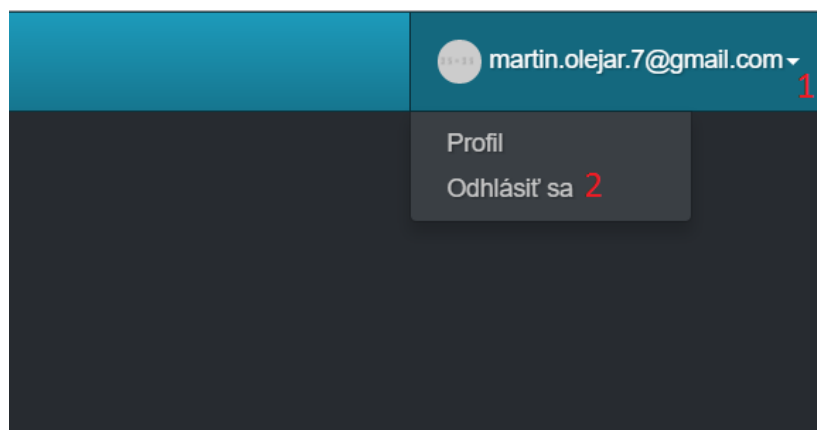
martin.olejar.7@gmail.com 1

Vygenerovať nové heslo 2

Obrázok 7: Formulár pre obnovu hesla

1.5 Odhlásenie používateľa

Pre odhlásenie zo systému kliknite na váš email (1) v hornom menu obrazovky na obr. 8. Potom sa objaví ponuka ďalších možností, v ktorej vyberte možnosť Odhlásiť sa (2).

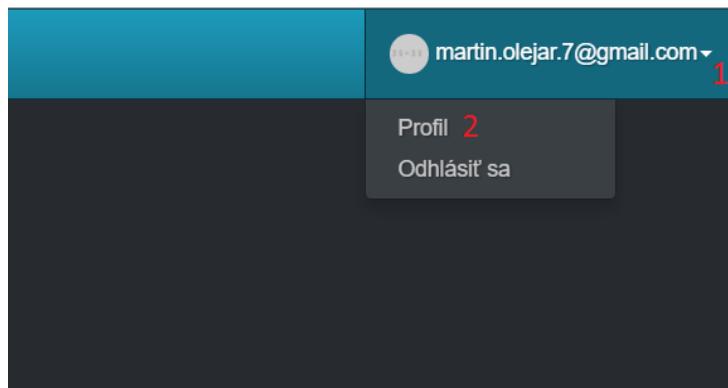


Obrázok 8: Odhlásenie



2 Úprava informácií o profile

Informácie o profile používateľa je možné kedykoľvek upraviť. Po prihlásení kliknite v hornom menu obrazovky na obr. 9 na váš email (1). Potom sa objaví ponuka ďalších možností, v ktorej vyberte možnosť Profil (2).

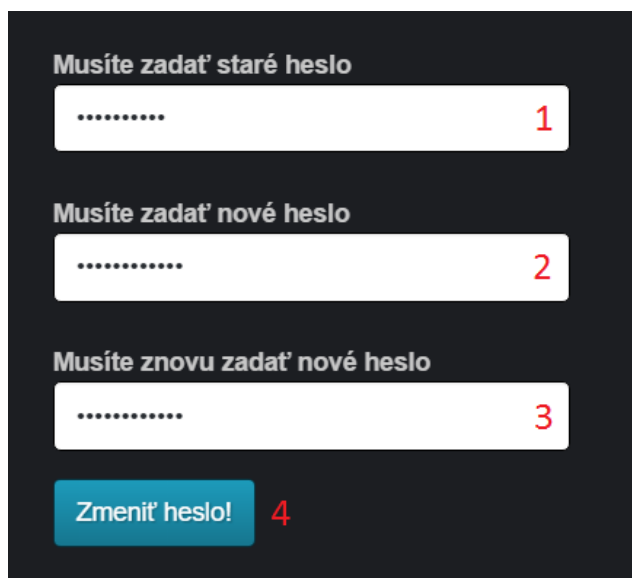


Obrázok 9: Možnosti prihláseného používateľa

Potom sa zobrazí formulár pre úpravu informácií o profile, ktorý je znázornený na obr. 10 a 11. V prvej časti formuláru na obr. 10 môžete vybrať svoju profilovú fotku pomocou tlačidla **Vybrať súbor** (1), upravovať svoje meno (2), priezvisko (3) a domovskú stránku (4). Ďalej je možné vybrať lokalizáciu (5) (Slovenčina alebo Angličtina). Zmenené údaje je nutné potvrdiť kliknutím na tlačidlo **Aktualizovať!** (6).

Obrázok 10: Úprava základných informácií o profile

V druhej časti formuláru na obr. 11 si môžete zmeniť svoje prihlasovacie heslo. Najprv zadajte svoje staré heslo (1), potom zadajte svoje nové heslo do polí (2) a (3) a potvrdíte tlačidlom **Zmeniť heslo!** (4).

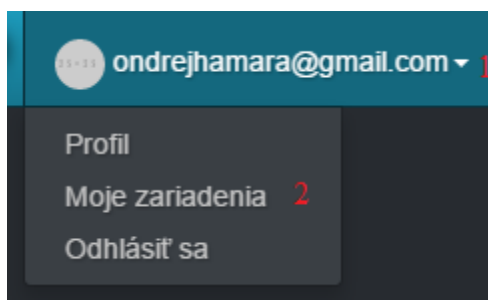


The image shows a dark-themed form for changing a password. It contains three input fields and one button. The first field is labeled 'Musíte zadať staré heslo' and contains a red '1'. The second field is labeled 'Musíte zadať nové heslo' and contains a red '2'. The third field is labeled 'Musíte znovu zadať nové heslo' and contains a red '3'. Below the fields is a blue button labeled 'Zmeniť heslo!' with a red '4' next to it.

Obrázok 11: Formulár pre zmenu hesla

2.1 Správa zariadení

Po prihlásení používateľ má možnosť zobrazenia a úpravy aktuálnych zariadení, z ktorých je aktuálne prihlásený. Kliknite v hornom menu obrazovky na obr. 12 na váš email (1). Potom sa objaví ponuka ďalších možností, v ktorej vyberte možnosť Moje zariadenia (2).



Obrázok 12: Možnosti prihláseného používateľa

Potom sa zobrazí tabuľka s aktuálnymi zariadeniami, ktorá je zobrazená na obr. 13. V tabuľke sa nachádza informácia o názve zariadenia (1), ID zariadenia (2), platforme zariadenia (3), používateľovej poznámke (4) a operáciách, ktoré môže používateľ vykonať (5). V stĺpci operácie môžete uložiť vyplnenú poznámku alebo odstrániť prihlásené zariadenie kliknutím na tlačidlo Uložiť (6) resp. Odstrániť (7).



Moje zariadenia

Názov zariadenia ¹	ID zariadenia ²	Platforma zariadenia ³	Poznámka ⁴	Operácie ⁵
-	-	Browser	<input type="text"/>	<input type="button" value="Uložiť"/> <input type="button" value="Odstrániť"/>

⁶ ⁷

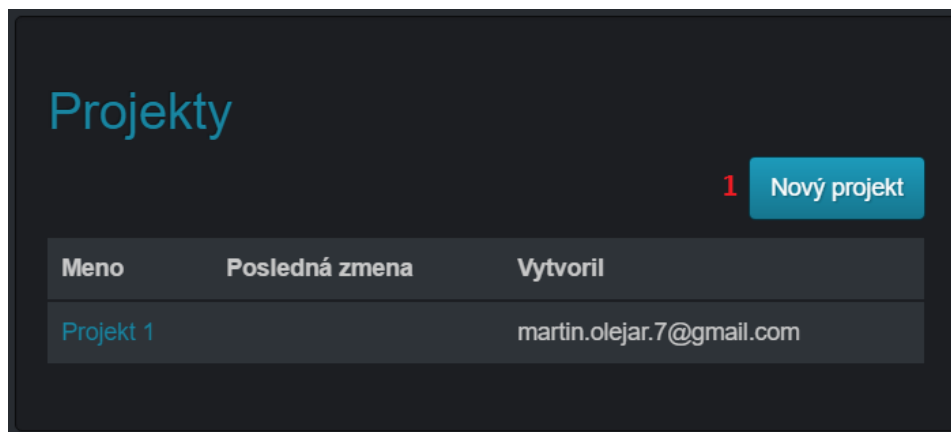
Obrázok 13: Tabuľka zariadení, z ktorých je používateľ prihlásený



3 Vytvorenie a správa projektov

3.1 Vytvorenie projektu

Po prihlásení používateľ môže vytvoriť nový projekt kliknutím na tlačidlo **Nový projekt** (1) v zozname projektov na obr. 14.



Obrázok 14: Zoznam aktuálnych projektov

Po kliknutí sa zobrazí formulár pre vytvorenie nového projektu na obr. 15, v ktorom vyplňte názov projektu (1) a popis projektu (2), môžete vybrať logo pre projekt kliknutím na tlačidlo Logo (3) a vytvorenie nového projektu potvrdíte kliknutím na tlačidlo **Vytvoriť** (4).

Názov projektu

Nový projekt 1

Popis

registrácia a prihlásenie 2

3 Logo

4 Vytvoriť

Obrázok 15: Vytvorenie nového projektu



3.2 Zoznam projektov

Po prihlásení používateľa je možné prezerať zoznam aktuálnych projektov v časti Projekty ako môžeme vidieť na obr. 16. Ku každému projektu je uvedené jeho meno, dátum poslednej zmeny a používateľ, ktorý ho vytvoril.

Meno	Posledná zmena	Vytvoril
Projekt 1	15.1.2016	januska.patrik@gmail.com
Projekt 2	15.1.2016	januska.patrik@gmail.com
Projekt 3	15.1.2016	januska.patrik@gmail.com
Projekt 4	15.1.2016	januska.patrik@gmail.com

Obrázok 16: Zoznam aktuálnych projektov

3.3 Správa projektu

Po kliknutí na niektorý z projektov sa dostávame do hlavného menu vybraného projektu, ktoré je znázornené na obr. 17. V ľavom paneli sa nachádzajú karty pre:

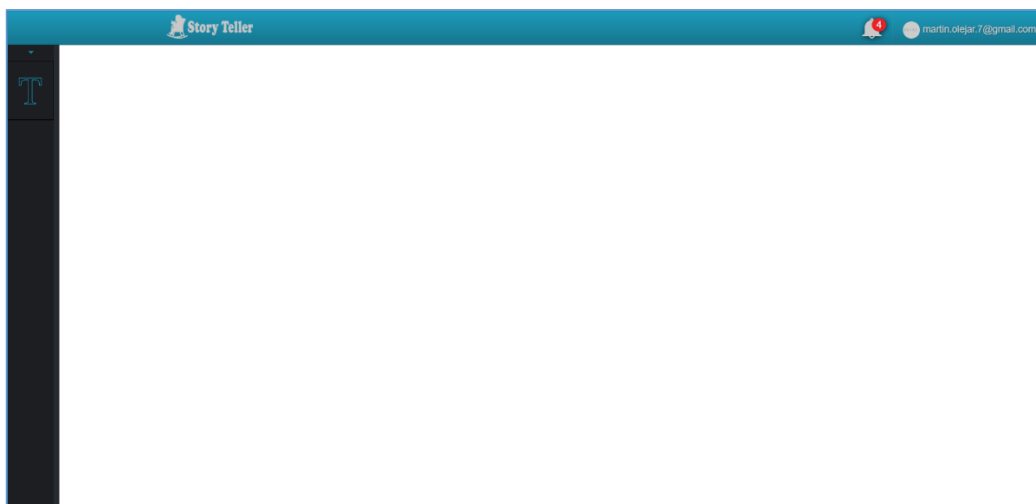
- zobrazenie základných informácií o projekte,
- správu skíc v projekte,
- správu akceptačných testov
- správu používateľov v projekte a
- nastavenia projektu.



Obrázok 17: Správa projektu

3.3.1 Správa skíc

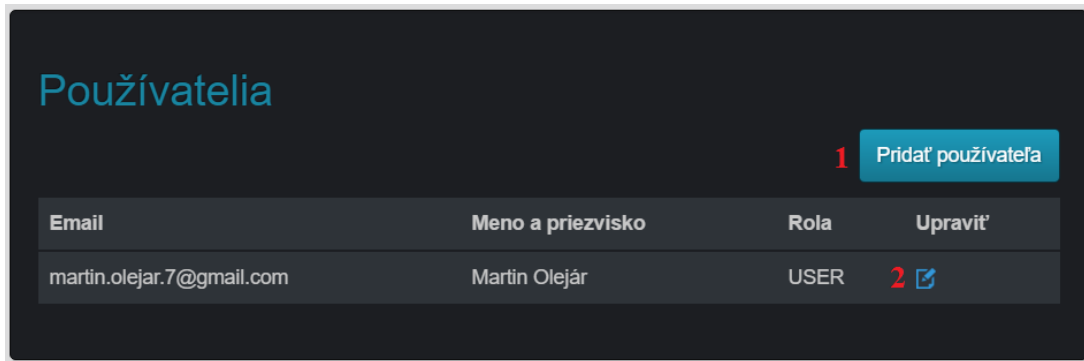
Po kliknutí na kartu "Skice" je možné vytvoriť novú skicu k projektu. Na obrázku 18 môžeme vidieť čisté plátno pre kreslenie skice spolu s komponentami v ľavom paneli, ktoré sa dajú potiahnuť do plátna, čím sa pridajú do skice.



Obrázok 18: Nová skica k projektu

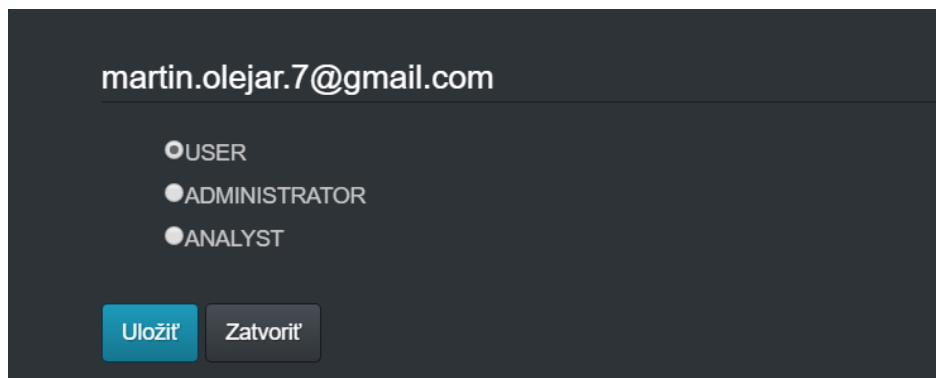
3.3.2 Správa používateľov

Po kliknutí na kartu "Správa používateľov" sa zobrazí zoznam používateľov priradených k danému projektu, ako je vidieť na obrázku 19. Tento zoznam obsahuje email používateľa, jeho meno a priezvisko a rolu v rámci projektu.



Obrázok 19: Zoznam používateľov v projekte

Do projektu je možné pridať používateľa kliknutím na tlačidlo **Pridať používateľa** (1). V projekte je možné zmeniť rolu niektorého z používateľov projektu kliknutím na ikonu (2) v stĺpci Upraviť na obrázku 19. Následne sa zobrazí modálne okno na obrázku 20, v ktorom vyberte novú rolu vybraného používateľa a kliknite na tlačidlo **Uložiť**. Vybranému používateľovi je poslaná notifikácia, čo je opísané v kapitole o notifikáciách.

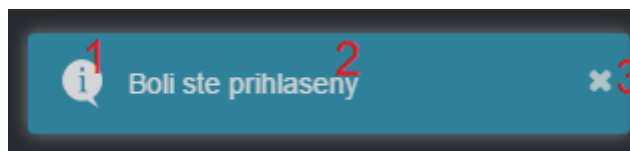


Obrázok 20: Úprava roly používateľa v projekte

4 Notifikácie

4.1 Zobrazenie notifikácie – toast popup

Po prijatí notifikácie je používateľovi zobrazený toast popup v pravom dolnom rohu obrazovky (obr. 21).

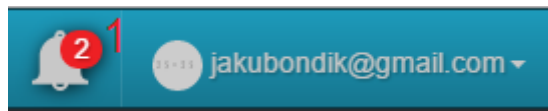


Obrázok 21: Toast popup notifikácia

Typ notifikácie je znázornený informačnou ikonkou (1) a farbou pozadia. Samotný text notifikácie je znázornený v strednej časti notifikácie (2). Notifikáciu je možné zavrieť kliknutím na tlačidlo zatvorenia (3) alebo kliknutím na samotnú notifikáciu.

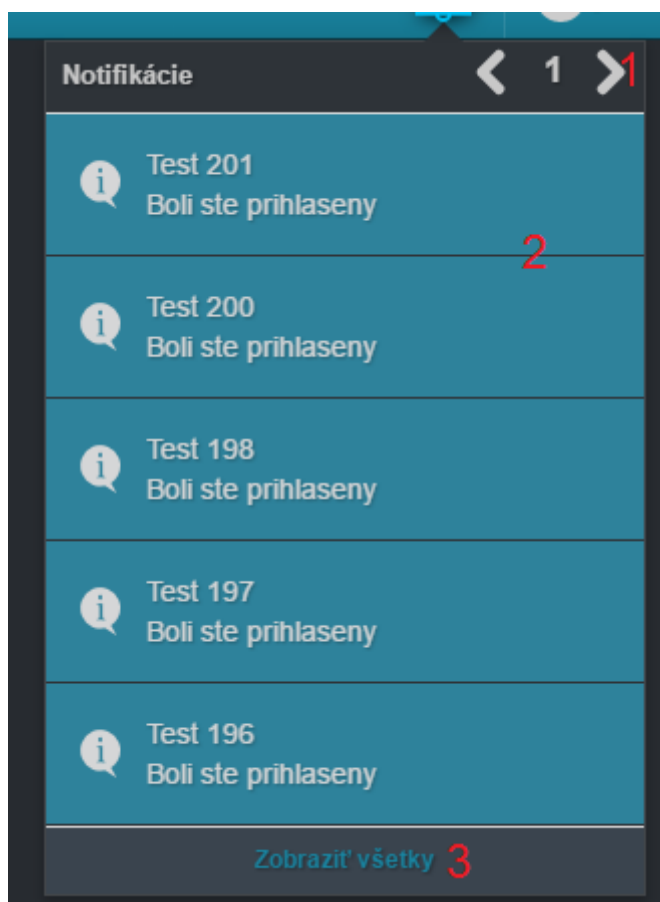
4.2 Zobrazenie notifikácie – navigačný panel

K notifikáciám je možné pristupovať pomocou tlačidla notifikácií v navigačnom paneli, ktoré môžeme vidieť na obrázku 22. Súčasťou tohto tlačidla je indikátor počtu nevidených notifikácií (1).



Obrázok 22: Indikátor notifikácií

Po kliknutí na toto tlačidlo sa zobrazia samotné notifikácie (obr. 23). Notifikácie sú zobrazené v skupinách po päť s ich typom a znením (2). K starším notifikáciám je možné pristupovať pomocou tlačidiel stránkovania (1). K prehľadu notifikácií v skupinách väčších ako päť je možné pristúpiť kliknutím na tlačidlo **Zobraziť všetky** (3).



Obrázok 23: Indikátor notifikácií

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Metodika integrácie a nasadzovania

Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller-04@googlegroups.com
Vypracoval: Bc. Adam Neupauer

1 Metodika integrácie a nasadzovania

Tento dokument opisuje, akým spôsobom sú nasadzované a integrované aplikácie vyvíjané v tímovom projekte.

Server

Aplikácie sú nasadzované na Linux server Ubuntu Server 16.04 LTS. Tento server obsahuje rozhranie príkazového riadku.

Vetvy

Vývoj je rozdelený do viacerých vetiev (branches), pričom pre každú User Story je vytvorená jedna vetva. Okrem nich existujú vetvy *dev* a *master*. Vetva *dev* slúži na mergovanie prijatých pull requestov. Vetva *master* slúži ako hlavná vetva obsahujúca finálny kód, ktorý už bol otestovaný na nasadenej vetve *dev*.

Počas aktívneho vývoju projektu je potrebné, aby bolo možné vidieť a vyskúšať práve prijatú zmenu. Z toho dôvodu sa na server nasadzuje vetva *dev*, do ktorej sú mergované práve prijaté zmeny. To zabezpečí okamžitú dostupnosť najaktuálnejšej verzie aplikácie. V budúcnosti, keď bude existovať funkčná verzia projektu, nasadzovať sa bude vetva *master*.

Nasadzovací systém - TFS

Systém na správu verzií – TFS využívame aj na kontinuálnu integráciu. TFS ponúka systém zostavovacích a nasadzovacích krokov, na základe ktorých je možné zostaviť postupnosť krokov. Výsledkom je zostavený, respektíve nasadený projekt.

Backend

Aplikácia backend je postavená na platforme Java a rámci Spring, s využitím SQL databázy PostgreSQL.

Backend build & deploy

Na zostavenie a nasadenie aplikácie backend je využitý build s názvom backend build & deploy.

Kroky :

- Maven build - system Maven zostaví aplikáciu na základe konfiguračných súborov
- Deploy to Apache Tomcat – aplikácia je nasadená na aplikačný server Tomcat cez URL

Po prijatí zmeny v podobe mergnutej User Story sa aplikácia automaticky zostaví pomocou systému Apache Maven. Systému Maven je pri zostavovaní predaný profil *prod*, ktorý zabezpečí, že budú použité konfiguračné súbory špecifické pre produkčnú verziu.

Aplikácia backend nasadzovaná na aplikačný server Apache Tomcat. Tomcat ponúka možnosť nasadzovania pomocou URL adresy bežiaceho serveru, pričom je poskytnutá cesta, na ktorej bude aplikácia nasadená.

Adresa backend : **api.story-teller.xyz**

Client

Client je aplikácia postavená na rámci Ionic, ktorý využíva AngularJS na vytvorenie aplikácií pre web ale aj mobilné platformy.

Client deploy

Na zostavenie a nasadenie aplikácie client je využitý build s názvom *client deploy*.

Kroky :

- PowerShell script

Nasadenie aplikácie client obsahuje len jeden krok – PowerShell script. Client totižto nepotrebuje zostavovanie, a teda stačí prekopirovať aplikáciu na server. PowerShell script najprv vykoná kompresiu projektu pre rýchlejšie odosielanie, a následne pomocou SCP prekopíruje súbory na server. Pomocou SSH potom spustí na servere script, ktorý sa postará o dekompresiu a nahradenie starej verzie aplikácie.

Adresa client : **app.story-teller.xyz**

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Metodika komunikácie

Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller-04@googlegroups.com
Vypracoval: Bc. Miroslav Hurajt

1 Komunikačné nástroje

Pre efektívnu a účinnú komunikáciu v tíme používame komunikačné nástroje, ktoré sú využívané rôzne podľa obsahu danej komunikácie. Tieto nástroje sú Slack, TFS, Facebook, e-mail.

1.1 Slack

Slack je najfrekventovanejšie využívaný komunikačný kanál. Pokrýva najväčšiu časť komunikácie v tíme. Obsahuje menšie kanály (channels) podľa obsahu komunikácie a taktiež v ňom môžeme diskutovať jednotlivito s členmi tímu v súkromnej diskusii, napr. o riešení spoločného používateľského príbehu alebo závislostiach medzi úlohami a podobne. Kanály v nástroji Slack sú:

- **#announcements** – do tohto kanálu píšeme oznámenia pre všetkých členov tímu o vytvorení nových kanálov a ich popise, na čo slúžia alebo inej zmene, čo sa týka komunikačných nástrojov a o organizačných informáciách, čo sa celého tímu týka – napr. odchod členov tímu, zdôvodnenie vykonanej akcie v rámci zmenu stavu úloh a pod.,
- **#design** – v tomto kanále sú umiestňované všetky návrhy na dizajn, štýly, farebné kombinácie atď. vytváranej aplikácie, ku ktorým sa očakáva, že sa ostatní členovia tímu vyjadria a zhodnotia ich v čo najkratšom možnom čase,
- **#documentation** – do tejto časti vkladáme návrhy na vylepšenie a hodnotenie vytvorenej dokumentácie, prípadne odkazy na novovytvorené dokumenty. Tento kanál ponúka priestor aj na diskusiu, aký nástroj na dokumentáciu je vhodné použiť,
- **#issues** – tento kanál slúži na pridávanie rôznych chýb a problémov, s ktorými sa členovia tímu stretli počas implementácie úloh, a následnú diskusiu o týchto problémoch,
- **#offtopic** – tu môžeme vkladať informácie a postrehy, ktoré sa priamo netýkajú práce na tímovom projekte. Tento kanál slúži aj na riešenie neočakávaných udalostiach, ktoré môžu ovplyvniť prácu v tíme,
- **#organizacia** – do tejto časti sa vkladajú informácie ohľadom organizácie prebiehajúceho šprintu: úlohy, ktoré treba prioritne riešiť, alebo nové nasadené časti a ich používanie. Taktiež tento kanál poskytuje priestor pre nové nápady pre fungovanie a organizáciu šprintov,
- **#sprinty** – tento kanál slúži na podávanie informácií o úlohách v rámci šprintov, prípadne na organizačné informácie o daných šprintoch,
- **#team_web** - do tohto kanálu vkladáme informácie o úprave tímovej stránky alebo prípadne návrhy na jej vylepšenie,
- **#teambuildings** – kanál slúžiaci na dohadovanie termínu najbližšieho neformálneho stretnutia tímu, tu zapisujeme aj vzniknuté nápady na vylepšenie vyvíjanej aplikácie a o iných dôležitých veciach, ktoré sa diskutovali na neformálnom stretnutí,
- **#technologie** – tento kanál slúži na diskusiu o nasadzovaných nových technológiách a vyjadrenie sa k nim,

- **#tfs** – v tomto kanále sú umiestnené informácie o zmenách stavoch taskov, pull requestoch, ku ktorým je možnosť vyjadriť sa,
- **#timove_roly** – v tomto kanále určenom pre všetkých členov tímu sa diskutuje o výbere a zastávaní tímových rolí a aj o prípadnom nedodržiavaní týchto rolí,
- **#tp_cup** – tu sa vkladajú informácie spojené so súťažou TP Cup-e - ako sú dokumenty na odovzdanie, ku ktorým je možné vyjadriť sa.

1.2 Komunikácia v TFS

Diskutovať o vrátení pull requestu a bližšie objasňovať vrátené časti je možné aj v nástroji Team Foundation Server v časti „pull request“ sekcii „code“, kde po kliknutí na tlačidlo „reply“ pod daným komentárom je možné sa k nemu vyjadriť.

1.3 Facebook

Na pripomenutie tímových úloh alebo neformálne diskusie o ďalšom priebehu tímového projektu môžeme diskutovať aj vo vytvorenej diskusnej skupine na sociálnej sieti Facebook. Avšak v tejto skupine by nemalo byť nič dôležité, čo sa týka práce na projekte a čo by sa malo dostať ku všetkým členom tímu.

Aj keď si nie sme istí, či sa vec, o ktorej chceme diskutovať, týka priamo projektu alebo by mala mať neformálny charakter, radšej použijeme ako komunikačný kanál Slack a časť #offtopic hlavne z dôvodu, že sa v Slacku ľahšie vyhľadáva a k diskutovaným témam sa môže vyjadriť aj vedúci tímu.

1.4 E-mail

V nevyhnutných prípadoch, ktoré blokujú ďalšiu prácu na tímovom projekte, napr. pri problémoch s prihlásením do TFS alebo pri výnimočných neočakávaných udalostiach, kontaktujeme vedúceho tímu alebo iného tímového kolegu na jeho emailovej školskej adrese.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Metodika konvencií písania zdrojového kódu

<u>Vedúci projektu:</u>	Ing. Karol Rástočný, PhD.
<u>Názov tímu:</u>	CoolStoryBro
<u>Členovia tímu:</u>	Bc. Jakub Ondik Bc. Patrik Januška Bc. Adam Neupauer Bc. Martin Olejár Bc. Miroslav Hurajt Bc. Ondrej Hamara
<u>Kontakt:</u>	storyteller-04@googlegroups.com
<u>Vypracoval:</u>	Bc. Jakub Ondik

1 Konvencie písania zdrojového kódu

Táto kapitola opisuje základne konvencie pri písaní kódu v jazykoch Java a JavaScript. Tieto konvencie sú upravené a obohatené na základe dobrých návykov (z angl. best practices) špecifických pre rámce, ktoré používame – Spring a Ionic (AngularJS). Jednotlivé konvencie sprevádzajú praktické ukážky kódu, vrátane príkladov s nedodržaním danej konvencie, ak je to vhodné. Konvencie uvedené v tejto kapitole majú odporúčací charakter, t. j. autor ich môže porušiť v špecifických prípadoch s odôvodnením.

1.1 Konvencie písania zdrojového kódu pre jazyk Java

1.1.1 Názvy

Názvy všetkých tried, rozhraní a vymenovaných typov musia byť uvedené použitím spôsobu upper camel case. Tieto názvy musia byť výstižné a nesmú obsahovať nejednoznačné a nevhodne známe skratky. Medzi názvom a začiatkom bloku musí byť vložená medzera.

Príklad:

```
public class myClass{ // nesprávne
    // code
}

public class MyClass { // správne
    // code
}
```

Názvy metód a premenných (atribútov, polí) musia byť uvedené použitím spôsobu lower camel case a podobne ako názvy tried, musia byť výstižné, stručné a nesmú obsahovať nevhodne známe skratky. Metódy nesmú obsahovať medzery medzi ich názvom a parametrami a musia obsahovať medzeru pred začiatkom bloku. Parametre metód musia byť oddelené čiarkou a za ňou nasledujúcou medzerou. Taktiež premenné tried (polia) musia byť definované s modifikátorom prístupu **private**, s výnimkou konštánt. Konštanty musia byť uvedené použitím spôsobu shouting snake case, musia byť uvedené na začiatku triedy a musia obsahovať modifikátory **static** a **final**. Taktiež musí byť dodržané pravidlo jednej premennej na jeden riadok.

Príklad pre metódy:

```
// nesprávne
public List<String> getOrdersForUser (User user,Date date){
    // code
}

// správne
public List<String> getOrdersForUserByDate(User user, Date date) {
    // code
}
```

Príklad pre atribúty:

```
public int AGE; // nesprávne
public int age, salary; // nesprávne
```

```
private int age; // správne
private int salary; // správne

private final int ageLimit = 20; // nesprávne
public static final int AGE_LIMIT = 20; // správne
```

1.1.2 Formátovanie kódu

Každý príkaz musí byť vhodne odsadený tabulátorom podľa úrovne bloku, pričom začiatok daného bloku vyjadrený kučeravou zátvorkou sa musí nachádzať na rovnakom riadku ako príkaz začínajúci blok. Taktiež medzi príkazom začínajúcim blok a zátvorkou sa musí nachádzať medzera.

Príklad:

```
while(true) // nesprávne
{
    ...
    System.out.println(...);
    ...
}

while (true) { // správne
    ...
    System.out.println(...);
    ...
}
```

V prípade podmienok (príkaz if) je nutné tento príkaz udávať s kučeravými zátvorkami aj v prípade, že jeho telo obsahuje len jeden príkaz.

Príklad:

```
if(...) // nesprávne
    return true;

if (...) { // správne
    return true;
}
```

Ak podmienka obsahuje viac vetiev, začiatok príkazu vetvy a jej začiatok musí začínať na rovnakom riadku ako koniec predchádzajúcej vetvy.

Príklad:

```
if (...) { // nesprávne
    return true;
}
else
{
    return false;
}

if (...) { // správne
    return true;
} else {
    return false;
}
```


Medzi jednotlivými definíciami metód a blokov sa musí nachádzať nový riadok. Taktiež každý volaný príkaz musí začínať na samostatnom riadku.

V prípade reťazenia volaní metód je nutné prvú reťazenú metódu (t. j. druhú volanú metódu v poradí) umiestniť na nový riadok a vhodne odsadiť. Taktiež by žiaden riadok nemal presahovať dĺžku 80 znakov. V prípade situácie, že niektorý riadok túto dĺžku prekračuje, je nutné riadok zlomiť buď za čiarkou, pred operátorom alebo pred bodkou a vhodne odsadiť.

1.1.3 Programovanie voči rozhraniam

V prípade použitia triedy, ktorá implementuje rozhranie, je nutné toto rozhranie použiť ako typ definície premennej alebo typ návratovej hodnoty metódy.

Príklad:

```
private ArrayList<String> names; // nesprávne
private List<String> names; // správne
```

1.1.4 Písanie komentárov

Komentáre nad triedami a metódami musia byť uvedené výhradne v tvare dokumentačných komentárov JavaDoc.

Príklad:

```
/**
 * Validates given JWT token
 * @param token given token
 * @param userDetails Spring security object with authed user info
 * @return true if JWT is valid, else false
 */
public boolean isValidToken(String token, UserDetails userDetails) {
    ...
}
```

V prípade jednoriadkových komentárov premenných alebo komentárov v tele metódy je nutné použiť `//`.

Príklad:

```
// jednoriadkový komentár
```

V prípade viacriadkových komentárov je nutné použiť `/* */`, nie `/** */`.

Príklad:

```
/* viac
 * riadkový
 * komentár
 */
```

1.1.5 Výnimky a logovanie

Všetky výnimky je nutné odchytiť a zalogovať alebo znovu vyhodiť, nie zalogovať a vyhodiť zároveň (okrem špecifických prípadov). Taktiež, ak je to vhodné, je ich nutné ošetriť v zachytávacom bloku. V prípade logovania je nutné k logovacej správe pripojiť znenie výnimky.

Príklad:

```
// nesprávne
try {
    ...
} catch (JwtException e) {
    LOGGER.fatal("Could not verify JWT origin");
    throw e;
}

// správne
try {
    ...
} catch (JwtException e) {
    LOGGER.fatal("Could not verify JWT origin", e);
}
}
```

Objekt logger je nutné definovať ako:

```
private static final Logger LOGGER = Logger.getLogger(TokenHandler.class);
z balíka org.apache.log4j.Logger.
```

1.1.6 Konvencie pre Spring

Okrem všeobecných Java konvencií je nutné dodržiavať špecifické konvencie pre rámec Spring.

Príkladom špecifickej konvencie je definícia REST endpointov. Controller, v ktorom sú definované endpointy k špecifickému zdroju (z angl. resource), by mal obsahovať anotáciu `@RequestMapping` s cestou podľa názvu daného zdroja (okrem prípadov controllera, v ktorom sú umiestnené endpointy týkajúce sa rôznych zdrojov) a anotáciu `@RestController`. Jednotlivé endpointy pre daný zdroj už teda neobsahujú názov zdroja.

Príklad:

```
// nesprávne
@RestController
public class UserController {
    @RequestMapping(value = "/users/current",
                    method = RequestMethod.GET)
    public ResponseEntity<?> getUserProfile() {
        ...
    }
}

// správne
@RestController
@RequestMapping(value = "/users",
                produces = MediaType.APPLICATION_JSON_VALUE)
public class UserController {
    @RequestMapping(value = "/current",
                    method = RequestMethod.GET)
    public ResponseEntity<?> getUserProfile() {
        ...
    }
}
```

Pri prístupe k databáze je nutné používať Spring repozitáre. Repozitáre musia byť definované ako rozhrania, ktoré rozširujú rozhranie `JpaRepository<T, ID>` a musia mať anotáciu `@Repository`. Taktiež triedy objektov, ku ktorým sa má pristupovať cez repozitár, musia implementovať rozhranie `Serializable`.

Príklad:

```
public class User implements Serializable { };  
public interface UserRepository extends JpaRepository<User, Long> { };
```

Tieto repozitáre nie je nutné ďalej implementovať.

1.2 Konvencie písania zdrojového kódu pre jazyk JavaScript

Táto podkapitola uvádza konvencie zdrojového kódu pre jazyk JavaScript, avšak s hlavným zameraním na rámec AngularJS. Konvencie, ktoré tu nie sú explicitne uvedené, sa preberajú z konvencií uvedených pre jazyk Java.

1.2.1 Názvy

Názvy jednotlivých komponentov (modulov) musia obsahovať prefix **app** (príklad `app.user`) a musia byť definované spolu s ich závislosťami v súbore **www/js/components.js**.

Samotné komponenty, pokiaľ nie sú globálne, musia byť uvedené použitím spôsobu camel case a umiestnené v ceste **www/js/components/<nazov komponentu>/**, pričom jednotlivé časti komponentov musia byť umiestnené v samostatných súboroch nazvaných s použitím spôsobu camel case.

Názvy a hodnoty konštánt musia byť uvedené v súbore **www/js/constants.js** a musia byť nazvané s použitím spôsobu shouting snake case, ako je to v prípade konvencií pre jazyk Java.

Názvy metód a atribútov a formátovanie kódu sa riadia konvenciami jazyka Java.

1.2.2 Metódy

Všetky metódy musia byť definované ako neanonymné a nesmú byť definované globálne, t. j. cez `$rootScope`. Toto neplatí napríklad pre zachytávanie event signálov.

Metódy pre komponent typu controller musia byť definované cez `ViewModel` na začiatku daného komponentu.

Príklad:

```
var vm = this;  
  
vm.changeInfo = changeInfo;  
vm.changePassword = changePassword;  
  
function changeInfo() {  
    ...  
}  
  
function changePassword() {  
    ...  
}
```

Metódy pre komponent typu factory a service musia byť definované cez objekt na začiatku daného komponentu.

Príklad:

```
return {
  login: login,
  logout: logout
}

function login() {
  ...
}

function logout() {
  ...
}
```

1.2.3 Výnimky a logovanie

Odchyťovanie výnimiek by malo prebiehať rovnakým spôsobom ako v jazyku Java – odchytiť a zalogovať alebo znovu vyhodiť, nie zalogovať a vyhodiť zároveň (okrem špecifických prípadov). Na rozdiel od loggera použitého v jazyku Java, logger v jazyku JavaScript (wrapper nad RavenJS, ktorý komunikuje so systémom Sentry) pozná len tri úrovne logovania – info, warning a error, ktoré sú ale postačujúce

Príklad:

```
try {
  ...
} catch (e) {
  LOGGER.error('Failed to translate notifications', e);
}
```

Objekt logger (wrapper nad RavenJS) sa nachádza v service LOGGER v module app.logger – tento modul je teda nutné definovať ako závislosť pre moduly, v ktorých sa používa.

1.3 Štýly názvov

Lower Camel Case – userService, authorityRepository

Upper Camel Case – User, Authority, AuthenticationFilter

Snake Case – buf_ws_msg, msg_template

Shouting Snake Case – REFRESH_INTERVAL, AUTH_HEADER_NAME

Hungarian notation – bActive, lId, iAge

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Metodika plánovania

Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller-04@googlegroups.com
Vypracoval: Bc. Martin Olejár

1 Riešenie používateľských príbehov

Na začiatku riešenia úloh v používateľskom príbehu je potrebné vykonať mikromanažment úloh s cieľom lepšej analýzy a návrhu úloh v rámci používateľského príbehu.

1.1 Mikromanažment používateľského príbehu

Keď zodpovedný človek za používateľský príbeh aktivuje používateľský príbeh, zorganizuje krátke stretnutie, na ktorom sa zúčastnia všetci riešitelia úloh v danom používateľskom príbehu. Stretnutie môže byť vykonané v škole, na internáte alebo prostredníctvom Slack-u. Odporúča sa ale osobné stretnutie. Príbeh stretnutia je napísaný v nasledujúcich odsekoch.

Na stretnutí je potrebné prediskutovať obsah jednotlivých úloh v používateľskom príbehu. Diskutujeme o tom, čo treba riešiť v úlohách a aké sú závislosti medzi nimi, rozoberáme popis uvedený pri úlohách. Vykonáme analýzu a jednoduchý návrh riešenia úloh, čo môže zahŕňať nakreslenie dátových entít, sekvenčného diagramu alebo súvislostí medzi úlohami.

Ďalšou dôležitou časťou je určenie priority vykonávania jednotlivých úloh, čo sa musí aj zaznačiť v nástroji Team Foundation Server. To znamená, že každá úloha bude mať určeného predchodcu a nasledovníka, čím jasne definujeme poradie vypracovania úloh.

Potom vykonáme plánovania vykonania úloh – určíme si dátumy, v ktorých budú vypracované jednotlivé úlohy. Dôležité je, aby človek, ktorý je zodpovedný za prvú úlohu podľa priority, určil čo najskorší dátum, do ktorého ju spraví, čím sa predíde blokovaniu úloh alebo neskoršej oprave riešenia úloh. Po dokončení prvej úlohy zodpovedná osoba označí úlohu za uzavretú a informuje ostatných o jej dokončení prostredníctvom Slack-u alebo inou formou.

1.2 Stavý používateľského príbehu

Používateľský príbeh prechádza rôznymi stavmi počas svojho vývoja. V nástroji Team Foundation Server je nutné tieto stavy priebežne meniť – je to úlohou osoby, ktorá je zodpovedná za používateľský príbeh. Nasleduje prehľad stavov a situácia, kedy ich treba pre daný používateľský príbeh nastaviť.

- ACTIVE – nastavuje sa vtedy, keď sa začne pracovať na používateľskom príbehu, tzn. v čase prvého stretnutia k používateľskému príbehu; tento stav môže nastaviť aj vlastník produktu, ak neschváli splnenie používateľského príbehu, pričom určí aj nedostatky v jeho splnení
- CODE REVIEW – nastavuje sa vtedy, keď sú dokončené všetky úlohy a bol vytvorený pull request
- RESOLVED – nastavuje sa vtedy, keď bol pull request schválený a kód používateľského príbehu bol pridaný do dev vetvy
- CLOSED – nastavuje sa ho vlastník produktu, ak schváli splnenie používateľského príbehu

1.3 Ďalšie pravidlá

Pre riešenie používateľských príbehov platia ďalšie pravidlá:

TEAM COOLSTORYBRO

- každý riešiteľ úlohy pridáva do opisu používateľského príbehu odkaz na dokument a číslo kapitoly, kde sa nachádza dokumentácia k úlohe,
- osoba, ktorá je zodpovedná za používateľský príbeh, dohliada na to, aby všetky odkazy na dokumentáciu boli uvedené,
- riešiteľ úlohy nastavuje úlohe stav *Active*, ak na nej začne pracovať,
- po ukončení práce na úlohe riešiteľ úlohy zmení jej stav na *Closed*.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Metodika k prehliadke kódu

<u>Vedúci projektu:</u>	Ing. Karol Rástočný, PhD.
<u>Názov tímu:</u>	CoolStoryBro
<u>Členovia tímu:</u>	Bc. Jakub Ondik Bc. Patrik Januška Bc. Adam Neupauer Bc. Martin Olejár Bc. Miroslav Hurajt Bc. Ondrej Hamara
<u>Kontakt:</u>	storyteller-04@googlegroups.com
<u>Vypracoval:</u>	Bc. Ondrej Hamara

1 Prehliadka kódu

1.1 Čo je to prehliadka kódu?

Prehliadka kódu je proces, ktorý zabezpečí vedomé a systematické kontrolovanie kódu iného programátora v tíme s cieľom nájsť nedostatky v kóde a upovedomiť ho prostredníctvom komentáru.

1.2 Prečo robiť prehliadku kódu?

Prehliadka kódu slúži na zlepšenie kvality kódu, predovšetkým zabezpečenia lepšej údržby a identifikovaniu skrytých chýb. Pripomienky obsahujú vylepšenia, ako efektívnejšie napísať kód, prípadne poznamenať na chýbajúce komentáre, použitie implementovanej funkcie a pod.

1.3 Postup pri prehliadke kódu

K prehliadke vypracovaného kódu v rámci používateľského príbehu je možné pristúpiť vtedy, ak sú dokončené všetky úlohy používateľského príbehu podľa ich opisu a k nim je napísaná dokumentácia. V prehliadke kódu vystupujú dve roly – človek zodpovedný za používateľský príbeh a posudzovateľ kódu. Postup prehliadky kódu je nasledovný:

1. Zodpovedný človek za používateľský príbeh vytvorí pull request v TFS na spojenie danej vetvy s vetvou dev (postup pre vytvorenie pull request-u je v metodike verziovania), pri vytváraní pull request-u sa určí meno posudzovateľa – ide o 1 člena tímu, ktorý nepracoval na žiadnej úlohe v danom používateľskom príbehu.
2. Potom určený posudzovateľ vykoná prehliadku kódu, čo zahŕňa označenie časti nesprávne napísaného kódu a komentovanie nájdenných nedostatkov kódu vo vytvorenom pull request-e v TFS (bližšie popísané v kapitole 1.4).
3. Osoba, ktorá požiadala o pull request, zodpovedá za opravu nedostatkov kódu, upozorní osoby, ktoré pracovali na úlohách používateľského príbehu, aby nedostatky opravili, ak sa týkajú ich úloh. Po opravení nedostatkov a ich odovzdaní sa aktualizuje pull request.
4. Posudzovateľ skontroluje opravené nedostatky a ak kód nemá ďalšie nedostatky, schváli pull request. V prípade, že kód má naďalej nedostatky, vyznačí ich a pokračuje sa krokom č. 3.
5. Osoba, ktorá požiadala o pull request, vyrieši konflikty pre spojenie vetvy s dev vetvou v prípade, že existujú, a skompletizuje pull request kliknutím na tlačidlo **Complete pull request** v TFS, čím sa spojí daná vetva s dev.

1.4 Pohľad posudzovateľa na kód

V prvom rade posudzovateľ musí rozumieť kódu. Ak posudzovateľ nerozumie tomu, čo kód robí, tak chyba je na strane autora a vzniká dôvod na prepracovanie kódu. Aby sme zabezpečili efektívnu a konzistentnú prehliadku, je dobré postupovať podľa nasledujúceho zoznamu vecí, ktoré kontrolujeme:

- Coding conventions
- Testy a Jdoc

- Miesto zmeny
- Neduplikovať kód
- Ošetrovanie chybových stavov

1.4.1 Coding conventions

Prvou vecou, ktorú kontrolujeme, je dodržiavanie konvencií zdrojového kódu. Ak sú tieto konvencie hrubo porušené, nepokračujeme ďalej v prehliadke kódu a okamžite ho vrátime autorovi. Podrobne opísané konvencie zdrojového kódu sa nachádzajú v metodike konvencií písania zdrojového kódu.

1.4.2 Testy a Jdoc

Pri prehliadke kódu pokračujeme tým, či zmena má testy. Ak nemá, tak uvažujeme, či pre kontrolovanú časť je zmysluplné vytvárať testy. Pokiaľ zmeny obsahujú testy, pozeráme, či naozaj testujú to, čo majú.

Bez patričnej dokumentácie sa ťažko zisťuje, čo autor myslel. V prípade nedostatkov v testoch alebo dokumentácii ďalej nepokračujeme a vrátime kód.

1.4.3 Miesto zmeny

Tento bod súvisí s tým, či zmena je vykonaná na správnom mieste (v správnom balíku, resp. triede). Nesprávne umiestnená zmena zhoršuje udržiavateľnosť, sťažuje budúce zmeny a zhoršuje čitateľnosť kódu.

1.4.4 Neduplikovať kód

Takmer poslednou vecou, ktorú kontrolujeme, je to, či sme podobný kus kódu už predtým nevideli. Ak áno, tak je duplicitný a zhoršuje údržbu. Riešením je presunutie zdieľanej časti do metódy, triedy alebo komponentu.

1.4.5 Ošetrovanie chybových stavov

Skúmame, či kód je správne napísaný z hľadiska riešenia chybových stavov. V jazykoch bez výnimiek kontrolujeme, či sú ošetrené všetky návratové hodnoty oznamujúce chyby. U výnimkách je situácia jednoduchšia, ale je potrebné popremýšľať, kde presne výnimky zachytávať a riešiť.

1.5 Iné odporúčania pri prehliadke kódu

- Komunikácia
 - o dôležité je výsledné dielo, nie dokazovať, kto je lepší
 - o ku kritike písať aj to, ako by sa zmena dala urobiť lepšie
- Nebuďte osobný
 - o neriešajte schopnosti iného člena tímu
 - o najprv sa opýtať, až tak kritizovať
- Nebojte sa pochváliť

- chváliť a oceňovať dobré myšlienky

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Metodika testovania zdrojového kódu

<u>Vedúci projektu:</u>	Ing. Karol Rástočný, PhD.
<u>Názov tímu:</u>	CoolStoryBro
<u>Členovia tímu:</u>	Bc. Jakub Ondik Bc. Patrik Januška Bc. Adam Neupauer Bc. Martin Olejár Bc. Miroslav Hurajt Bc. Ondrej Hamara
<u>Kontakt:</u>	storyteller-04@googlegroups.com
<u>Vypracoval:</u>	Bc. Patrik Januška

1 Pravidlá testovania zdrojového kódu

V rámci projektu je testovanie zložené z dvoch častí:

- **testovanie backendu** – prostredníctvom jednotkových testov (angl. Unit test) pre otestovanie správneho fungovania vytvorených REST endpoitov
- **testovanie frontendu** – testovanie AngularJS komponentov, kvôli funkčnej komunikácii s backendom, prípadne správne zobrazovaniu výsledkov

Každý človek zodpovedný za úlohu vytvára a vykonáva k svojej úlohe príslušné testy. Spôsob vytvárania testov a ich spúšťania je opísaný v ďalších kapitolách zvlášť pre backend a frontend. Riešenie danej úlohy je z hľadiska funkčnosti správne, ak všetky testy úspešne prejdú.

1.1 Testovanie backendu aplikácie

Testovanie je založené na vytváraní jednotkových testov pre jednotlivé ucelené časti kódu. Po úprave kódu alebo pridaní súborov do projektu skontrolujeme funkčnosť projektu a spustíme všetky jednotkové testy.

1.1.1 Vytvorenie testu

Testy sa vytvárajú v príslušných triedach (angl. Class) určených na testovanie. Tieto triedy sa nachádzajú v balíku **com.storyteller** v zložke s cetou **src/test/java**. V tomto balíku sa nachádzajú triedy k testovaniu tried obsahujúcich služby (angl. service).

Ak chceme vytvoriť korektný test, postup je nasledovný (príklad vytvorenia testu pre pridanie používateľa):

1. Pozrieme sa v akej triede je implementovaná logika pridania používateľa
 - a. V našom prípade vidíme, že ide o triedu **UserService.java** balíka **com.storyteller.service**.
2. Prejdeme teda do príslušnej testovacej triedy k nájdenej **service** triede
 - b. V našom prípade **UserServiceTests.java**
3. V tejto testovacej triede vytvoríme novú metódu, pričom jej názov bude začínať kľúčovým slovom „**test**“ a pridáme k nej anotáciu **@Test**
 - c. V našom prípade to vyzerá nasledovne:

```
@Test
public void testCreateUser() {
}
```

4. Po takejto inicializácii testovacej metódy môžeme prejsť k vytvoreniu samotnej logiky testu
 - d. v našom prípade vytvoríme objekt nového používateľa a nastavíme mu príslušné údaje. Následne voláme metódu **createUser(User user)** triedy **UserService**, v ktorej je logika vytvorenia používateľa implementovaná, a ktorú chceme otestovať.
5. Do logiky testu pridávame kľúčové metódy ako **assertNotNull()** – ak chceme zistiť, či sa hodnota parametra objektu nerovná Null, **assertEquals** – ak chceme zistiť, či sa rovnajú dve hodnoty a mnohé ďalšie podľa potreby.

6. Po takto vytvorenom teste môžeme prejsť k spusteniu samotného testu, opísanému v nasledujúcej kapitole.

V prípade, že chceme otestovať funkčnosť metód implementovaných v controlleroch, vytvárame tzv. mocky. V tomto prípade píšeme testy do tried, ktoré vo svojom názve obsahujú namiesto kľúčového slova "Service" kľúčové slovo "Controller", napríklad trieda `SketchControllerTest.java`. V tomto prípade sledujeme, aký `HttpStatus` nám metódy vracajú. Ak je test úspešný, metóda vráti `HttpStatus.OK`, v opačnom prípade to môžu byť stavy `HttpStatus.BAD_REQUEST` alebo `HttpStatus.NOT_FOUND`, kedy treba funkcionality opraviť.

1.1.2 Spustenie a vyhodnotenie testu

Postup spustenia testu je nasledovný (príklad spustenia testu pre pridanie používateľa):

1. Pravým tlačidlom myši klikneme na „testovaciu triedu“, v ktorej sme test implementovali
 - a. V našom prípade pravý klik na triedu `UserServiceTests.java`.
2. Z ponúknutých možností vyberieme „**Run as**“
3. Z ďalších ponúknutých možností vyberieme „**JUnit Test**“
4. Následne sa spustí testovanie metód implementovaných v danej triede
 - a. V našom prípade sa spustí testovanie metód v triede `UserServiceTests.java`
5. V spodnej časti nášho IDE môžeme v karte „**JUnit**“ sledovať priebeh úspešnosti testov. Ak je test konkrétnej metódy úspešný, bude označený zelenou farbou, v opačnom prípade je farba testu červená.
 - a. V našom prípade je test „`testCreateUser`“ označený zelenou farbou, a teda prebehol úspešne.
6. Ak je test neúspešný, chyba je opísaná vedľa zoznamu testov v stĺpci „**Failure Trace**“ v karte „**JUnit**“ v spodnej časti IDE.
7. Rada: Ak sú neúspešné len niektoré testy, nie je potrebné spúšťanie všetkých testov za účelom šetrenia času. Ak je neúspešný napríklad len jeden test, klikneme naň pravým tlačidlom myši a z ponuky vyberieme „**Run**“. Následne prebehne len tento jediný test.

1.2 Testovanie frontendu aplikácie

Pri testovaní frontendu sa testy simulujú už priamo cez rozhranie webového prehliadača.

1.2.1 Vytvorenie testu

1. V prípade, že ide o otestovanie, či sa v prehliadači zobrazujú informácie z databázy, je potrebné tieto informácie predtým do databázy manuálne pridať.
2. Na strane backendu je následne potrebné v súbore `application.properties` v zložke s cestou `src/main/resources` zmeniť hodnoty:
 - a. `spring.datasource.initialize = true` na `spring.datasource.initialize = false`
 - b. `spring.jpa.hibernate.ddl-auto = create` na `spring.jpa.hibernate.ddl-auto = update`

3. Následne !SPUSTÍME SERVER!

1.2.2 Spustenie a vyhodnotenie testu

Postup spustenia samotného frontendu je nasledovný:

1. Spustíme **node.js** príkazový riadok
2. V otvorenom príkazovom riadku prejdeme do zložky **StoryTeller\client**
3. Zadáme príkaz **ionic serve** a stlačíme ENTER
4. Otvorí sa nám webová stránka projektu StoryTeller a s použitím príslušných URL ciest, sa dostaneme k tej časti aplikácie, ktorú chceme otestovať.
5. URL cesty pre konkrétne implementácie sú uvedené v zložke **client\www\js\components** pri príslušných komponentoch v súboroch s príponou **.js** obsahujúcich kľúčové slovo **Routes** – napríklad **projectRoutes.js** v zložke **client\www\js\components\project**
6. Následne po zadaní konkrétnej URL cesty prejdeme manuálne scenár, ktorého funkcionality sme implementovali a teraz ju chceme otestovať. Ak sa pri prechádzaní-testovaní scenára zobrazia relevantné výsledky (napríklad zoznam projektov, ktoré sme v predchádzajúcich krokoch pridali do databázy), test považujeme za úspešný.
7. Ak nastane problém, v ktorom nedostaneme odpoveď akú sme očakávali (napríklad zostane tabuľka projektov prázdna), pozrieme sa, či nie je chyba opísaná v konzole webového prehliadača alebo v konzole servera a odstránime ju.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Metodika k verziovaniu

Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller-04@googlegroups.com
Vypracoval: Bc. Martin Olejár

1 Pravidlá pre vetvenie zdrojového kódu

V rámci hlavného repozitára projektu rozlišujeme tieto vetvy (angl. branch):

- **master** – produkčná vetva pre zákazníka,
- **dev** – vývojová vetva,
- vetvy pre jednotlivé používateľské príbehy, pre každý použ. príbeh existuje 1 vetva.

1.1 Vytvorenie vetvy pre používateľský príbeh

Na používateľskom príbehu sa pracuje v osobitnej vetve pre daný používateľský príbeh. Vetvu pre používateľský príbeh vytvára ten, kto je zaň zodpovedný. Pre názov tejto vetvy platí:

- píše sa v angličtine,
- má tvar “Číslo user story.Krátky popis user story”, napr. 4567.Devices,
- v prípade, že sa krátky popis používateľského príbehu skladá z viacerých slov, tieto slová sa začínajú veľkým písmenom a píše sa spolu, napr. 4575.PasswordRecovery.

Postup pre vytvorenie vetvy prostredníctvom Team Foundation Server (TFS) je nasledujúci:

1. prihlásime sa do TFS,
2. klikneme na **CODE** v hornom menu,
3. klikneme na názov vetvy,
4. zobrazí sa okno, v ktorom klikneme na možnosť **New branch...**,
5. v ďalšom okne pre vytvorenie vetvy vyplníme jej meno podľa definovaných pravidiel a v časti **Based on** vyberieme vetvu **dev**.

1.2 Pravidlá pre prácu vo vetve

Pre prácu s vetvami platia nasledujúce pravidlá, ktoré treba dodržiavať:

- priamo do vetiev **master** a **dev** sa nikdy nič nepridáva,
- pracuje sa iba v osobitnej vetve pre daný používateľský príbeh,
- pre aktualizáciu webovej stránky sa pracuje vo vetve Global.TeamWebsite,
- pred tým, ako začíname pracovať vo vetve, je potrebné stiahnuť si (angl. **pull**) zmeny, ktoré vykonal niekto iný do tejto vetvy, čím sa predíde neskorším konfliktom,
- pred tým, ako odovzdáme zmeny, resp. dáme **commit** do vetvy, skontrolujeme funkčnosť programu a spustíme všetky jednotkové testy
 - o ak všetky testy prejdú a program funguje bez chýb, je možné dať commit do vetvy, inak treba pred commitom nedostatky opraviť
- je dobré dávať commit do vetvy čo najčastejšie, aby ostatní pracujúci v tej istej vetve mohli použiť pridaný kód a aby nevzniklo veľa konfliktov pri spájaní,
- popis commitu k určitej konkrétnej úlohe píšeme po anglicky v tvare “[Task number] Popis pridanej funkcionality a iných dôležitých vecí“, kde tento popis môže pre lepšie vyhľadávanie obsahovať kľúčové slová ako napr. added, fixed, removed
- pri viacerých úlohách je tvar popisu nasledujúci:
“[Tasks number1, number2] Popis pridanej funkcionality a iných dôležitých vecí“
- ak sú vypracované všetky úlohy v rámci používateľského príbehu, pridáme vetvu **dev** do našej pracovnej vetvy (angl. **merge dev into branch**)

- ak vzniknú konflikty, treba ich vyriešiť, vykonať jednotkové testy pre overenie zachovania funkcionality a potom dať ešte raz commit do pracovnej vetvy
- po vypracovaní všetkých úloh v rámci používateľského príbehu a spojení s **dev** sa riadime podkapitolou 1.4.

1.3 Pravidlá pre spájanie vetiev

Všetky vetvy pre používateľské príbehy sú spájané po ich dokončení s vetvou **dev**. Vetva **dev** sa spája s vetvou **master** po dokončení každého šprintu.

1.4 Vytvorenie požiadavky na kontrolu používateľského príbehu

Po vypracovaní všetkých úloh v rámci používateľského príbehu, čo zahŕňa funkcionality, jednotkové testy, dokumentáciu modulov systému, kódu a doplnenie používateľskej príručky (ak sa dá), je nutné niekomu dať **pull request** podľa metodiky prehliadok kódu. Postup pre pull request je nasledujúci:

1. prihlásime sa do TFS,
2. klikneme na **CODE** v hornom menu,
3. klikneme na **Pull Requests** v menu,
4. klikneme vpravo na **New pull request**,
5. najprv vyberieme názov našej vetvy, ktorú je potrebné pridať do vetvy **dev**, napr.:
Review changes in 4567.Devices relative to **dev**
6. vyplníme názov a v časti **Reviewers** vyberieme niekoho z tímu podľa metodiky prehliadok kódu,
7. klikneme na **New pull request** v dolnej časti.

V prípade, že chceme aktualizovať webovú stránku, používame vetvu **Global.TeamWebsite**, ktorú je potrebné pridať do vetvy **master**.

2 Používanie Git-u na verziovanie

V tímovom projekte je možné používať verziovací systém Git. Pre ďalšie časti tejto metodiky je potrebné si ho nainštalovať.

2.1 Inicializácia lokálneho repozitára

Pre pridávanie kódu a súborov do projektu je potrebné si vytvoriť lokálny repozitár. Postup sa skladá z nasledujúcich krokov:

1. vytvoríme nový priečinok kdekoľvek vo svojom počítači,
2. nastavíme sa v príkazovom riadku do nového priečinka a spustíme príkaz *git init*,
3. pre naklonovanie projektu do priečinka spustíme príkaz

```
git clone https://tfs.fii.stuba.sk:8443/tfs/StudentsProjects/_git/StoryTeller
```

2.2 Odovzdávanie zmien

Po úprave kódu alebo pridaní súborov do projektu skontrolujeme funkčnosť projektu a spustíme všetky jednotkové testy. Ak všetky testy prejdú a projekt je funkčný, môžeme odovzdať (angl. commit) aktuálnu verziu repozitára do vetvy, aby si ostatní členovia tímu mohli stiahnuť pridané časti projektu. Postup sa skladá z nasledujúcich krokov:

1. nastavíme sa v príkazovom riadku do priečinka StoryTeller, ktorý sa nachádza v našom vytvorenom priečinku pre lokálny repozitár,
2. spustíme príkaz *git add*.
3. spustíme príkaz *git commit -m "popis"* – popis odovzdania vyplníme podľa definovaných pravidiel, napríklad:

```
git commit -m "[Task 4330] Added Angular controller for user profile with basic functions, removed useless dependencies"
```

4. spustíme príkaz *git push -u origin <názov vetvy>*.

2.3 Stiahnutie aktuálnej verzie vetvy

Pre stiahnutie aktuálnej verzie vetvy do lokálneho repozitára je postup nasledovný:

1. nastavíme sa v príkazovom riadku do priečinka StoryTeller, ktorý sa nachádza v našom vytvorenom priečinku pre lokálny repozitár,
2. spustíme príkaz *git pull*.

2.4 Zistenie aktuálneho stavu

Git umožňuje zobrazit' stav lokálneho repozitára – 1 príkazom môžeme vidieť, ktoré súbory spolu s ich umiestnením sme pridali, modifikovali alebo odstránili. Postup je nasledovný:

1. nastavíme sa v príkazovom riadku do priečinka StoryTeller, ktorý sa nachádza v našom vytvorenom priečinku pre lokálny repozitár,
2. spustíme príkaz *git status*.

2.5 Riešenie problému konfliktov

Niekedy sa môže stať, že odovzdanie zmien v zdrojovom kóde nie je možné, pretože sa našli konflikty. Vtedy je nutné použiť nasledujúce príkazy:

1. *git stash*
2. *git pull*
3. *git stash apply*

V prípade, že je stále problém, treba vyriešiť priamo v kóde konflikty, ktoré sa do kódu dopísali.

Po ich vyriešení zadáme príkazy:

1. *git add .*
2. *git merge*

V prípade úspechu dáme *git stash drop*.

3 Nástroj na prácu s vetvami

Po úvodných problémoch s Gitom sme začali používať nástroj SourceTree, ktorý ponúka prehľadné zobrazenie vetiev projektu a možnosti pre ich spájanie.

3.1 Inicializácia lokálneho repozitára

Postup pre inicializácia lokálneho repozitára, teda naklonovanie globálneho repozitára do priečinka v počítači, je nasledujúci:

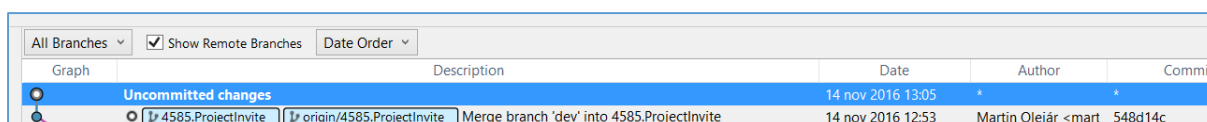
1. v SourceTree klikneme v hornom menu na **Clone / New**,
2. vyplníme **Source Path / URL** nasledujúcim odkazom:
https://tfs.fiiit.stuba.sk:8443/tfs/StudentsProjects/StoryTeller/_git/StoryTeller
3. vyplníme **Destination Path**, t.j. cestu k priečinku, kde bude uložený lokálny repozitár,
4. potvrdíme tlačidlom **Clone**.

3.2 Práca vo vetve

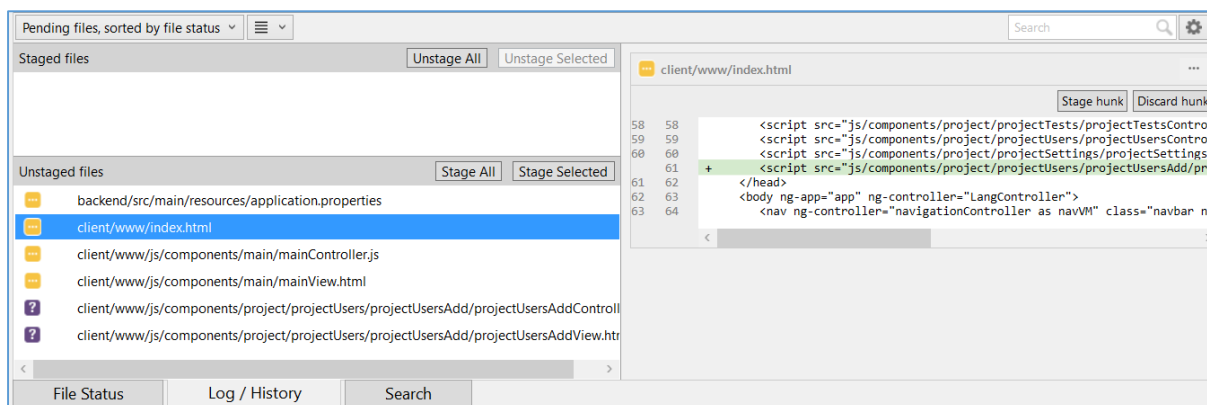
Pracovnú vetvu si otvoríme tak, že v časti BRANCHES klikneme dvakrát na danú vetvu. Pri úplne prvom otváraní vetvy je nutné túto akciu vykonať v časti REMOTES.

Tlačidlo **Fetch** v hornom menu slúži na aktualizáciu zobrazovaných commitov v nástroji. Toto tlačidlo treba stlačiť vždy, keď začíname pracovať, alebo priebežne pre aktuálne zobrazenie. Takto môžeme zistiť, či niekto iný vykonal commit v našej pracovnej vetve.

Všetky zmeny, ktoré vykonáme vo vetve, si môžeme v nástroji pozrieť. Najprv klikneme na **Uncommitted changes** podľa obr. 1. V dolnej časti (na obr. 2) sa nám zobrazia všetky súbory, ktoré sme modifikovali. Po kliknutí na nejaký súbor vidíme vpravo všetky zmeny, ktoré sme v tomto súbore vykonali.



Obrázok 1: Zoznam vykonaných commitov

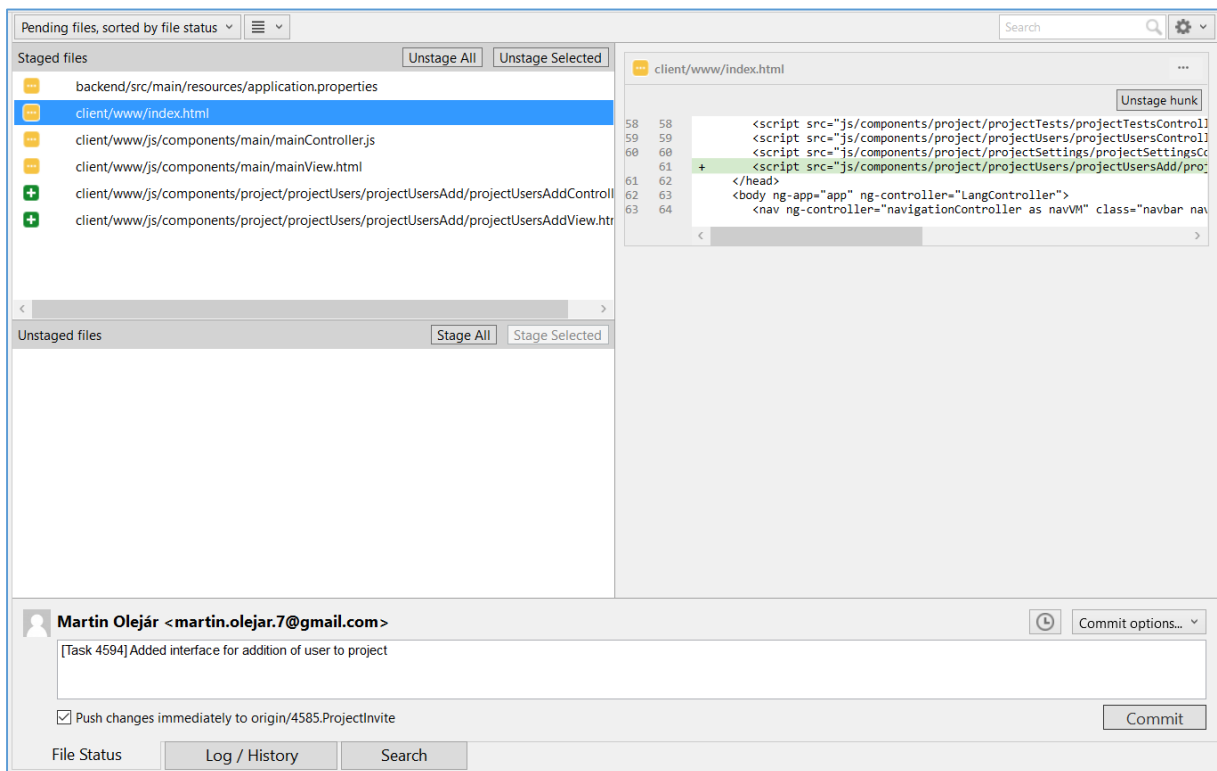


Obrázok 2: Zobrazenie zmien vo vetve

3.3 Odovzdávanie zmien

Po úprave kódu alebo pridaní súborov do projektu skontrolujeme funkčnosť projektu a spustíme všetky jednotkové testy. Ak všetky testy prejdú a projekt je funkčný, môžeme odovzdať (angl. commit) aktuálnu verziu repozitára do vetvy, aby si ostatní členovia tímu mohli stiahnuť pridané časti projektu. Postup sa skladá z nasledujúcich krokov:

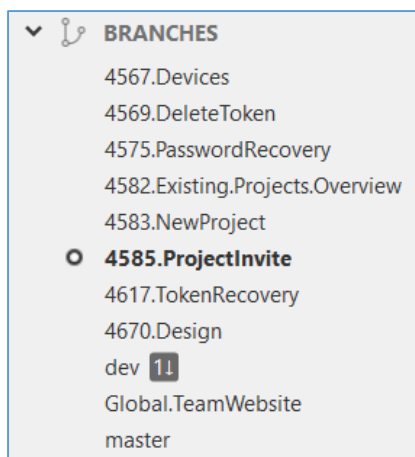
1. klikneme v dolnej časti na **File Status**,
2. v časti **Unstaged files** uvidíme všetky modifikované súbory, klikneme na **Stage All**,
3. v dolnej časti vyplníme popis commitu podľa definovaných pravidiel,
4. zaškrtneme **Push changes immediately to ...**,
5. výsledná situácia je zobrazená na obr. 3, môžeme si tiež prezerat' vykonané zmeny v jednotlivých súboroch
6. pre potvrdenie commitu klikneme na **Commit**.



Obrázok 3: Odovzdanie zmien

3.4 Stiahnutie aktuálnej verzie vetvy

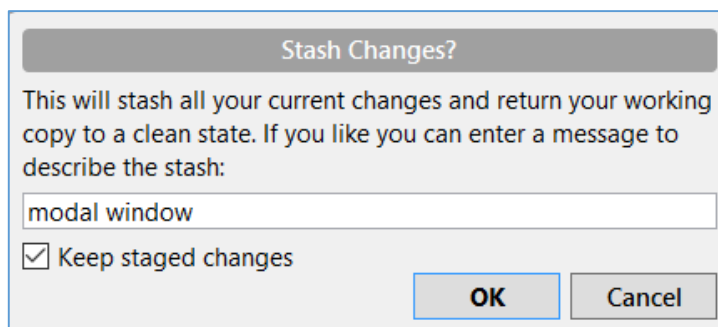
Ak niekto iný vykonal zmeny vo vetvách, je potrebné si tieto zmeny stiahnuť. Najprv klikneme na **Fetch** v hornom menu. Pri každej vetve, ktorá nie je lokálne aktuálna, teda niekto iný ju modifikoval a vykonal commit a jeho zmeny nie sú v lokálnom repozitári, sa zobrazí počet vykonaných commitov a šípka dole ako na obr. 4. Stiahnutie aktuálnej verzie vetvy vykonáme tak, že sa prepne do danej vetvy a klikneme na tlačidlo **Pull** v hornom menu.



Obrázok 4: Zobrazenie aktuálnych vetiev

3.5 Odloženie vykonaných zmien vo vetve

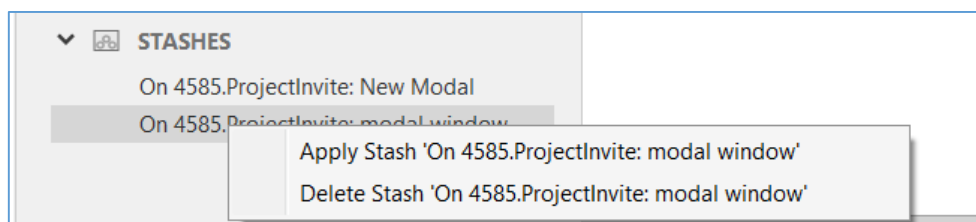
Ak pracujeme v nejakej vetve a potrebujeme vykonať nejaké zmeny v inej vetve, teda musíme sa prepnúť do inej vetvy, je potrebné si predtým vykonané zmeny odložiť (angl. **stash**). V hornom menu klikneme na **Stash**, zobrazí sa modálne okno na obr. 5, v ktorom vyplníme správu pre popis stash-u a zaškrtneme **Keep staged changes**. Potvrdíme kliknutím na **OK**.



Obrázok 5: Odloženie zmien

Potom sa môžeme prepnúť do inej vetvy a vykonávať v nej ľubovoľné zmeny aj robiť commity. Zmeny, ktoré sme si odložili pomocou stash, si môžeme vrátiť naspäť do vetvy. Postup je nasledujúci:

1. prepne sa do danej vetvy, t.j. dvakrát klikneme na názov vetvy,
2. v časti STASHES (na obr. 6) klikneme pravým tlačidlom na názov stash-u (v našom prípade *modal window*) a klikneme na **Apply Stash...**



Obrázok 6: Aplikovanie stash-u

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Tímový projekt



Story Teller

Šablóna na dokumentácie

<u>Vedúci projektu:</u>	Ing. Karol Rástočný, PhD.
<u>Názov tímu:</u>	CoolStoryBro
<u>Členovia tímu:</u>	Bc. Jakub Ondik Bc. Patrik Januška Bc. Adam Neupauer Bc. Martin Olejár Bc. Miroslav Hurajt Bc. Ondrej Hamara
<u>Kontakt:</u>	storyteller-04@googlegroups.com
<u>Vypracoval:</u>	Bc. Martin Olejár

Obsah

1	Kapitola číslo 1.....	1-1
1.1	Metodika.....	1-1
1.1.1	Text – štýl Normálny.....	1-1
1.1.2	Odrážky a číslovanie	1-1
1.1.3	Príklad tabuľky + popis nad ňou (takisto obrázok).....	1-1
2	Kapitola – štýl Nadpis 1	2-1
2.1	Podkapitola – štýl Nadpis 2	2-1
2.1.1	Podpodkapitola – štýl Nadpis 3.....	2-1

1 Kapitola číslo 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

1.1 Metodika

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua

1.1.1 Text – štýl Normálny

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, Sunt in culpa qui officia deserunt mollit anim id est laborum

1.1.2 Odrážky a číslovanie

Príklad odrážok:

- sdffsd
 - o fsdfsd
 - fsdfsd

Príklad číslovania:

1. fgffdd
2. gfdgfg
 - a. fgdfgdf
 - b. fdgfdgdfgd

1.1.3 Príklad tabuľky + popis nad ňou (takisto obrázok)

Tabuľka 1: Popis

Úloha	Zodpovedná osoba	Termín

2 Kapitola – štýl Nadpis 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Poznámky:

- Každá kapitola najvyššej úrovne sa začína na novej strane
- Každá kapitola najvyššej úrovne sa končí zlomom sekcie a zlomom strany
- Všetky strany kapitoly majú v hlavičke popis
- Prvý odsek hocijakej kapitoly nemá odsadenie

2.1 Podkapitola – štýl Nadpis 2

2.1.1 Podpodkapitola – štýl Nadpis 3

Menší nadpis – štýl Paragraph Heading

