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@googlegroups.com
Akademický rok: 2016/2017
Dátum odovzdania: 16. 11. 2016

Obsah

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.

1 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.

1.1 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 prideľovaní.

Tabuľka 1: Roly členov tímu

Zodpovedný člen tímu	Manažérska rola
Bc. Jakub Ondik	Manažér vývoja a kvality
Bc. Patrik Januška	Manažér testovania a kvality
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

1.2 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
- dohľad nad kvalitou systému

Bc. Adam Neupauer

- vytvorenie, správa a štýlovanie tímovej webovej stránky
- správa tímového servera
- kontinuálna integrácia
- návrh dizajnu systému

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

1.3 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	

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

2 Aplikácie manažmentov

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

2.1 Manažment dokumentácie

Dokumentácia k nášmu vyvíjanému projektu je písaná pravidelne v rámci jednotlivých šprintov. Po dokončení taskov, aby bola práca 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.

2.2 Manažment komunikácie

Komunikácia mimo tímové stretnutia prebieha prevažne cez komunikačný kanál Slack. Ktorý umožňuje pohodlné vyhľadávanie vo veľkom počte správ a rôzne témy sú dobre rozdelené do kanálov. Cez tento komunikačný môžeme pohodlne kontaktovať všetkých členov tímu a taktiež diskutovať o vzniknutých problémoch, nápadoch, návrhoch.

Komunikácie 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 vedúci tímu a každý člen sa vyjadrí čo sa mu páčilo a nepáčilo a následne sa spoločne hľadajú riešenia k týmto problémom.

2.3 Manažment plánovania

Na management naplánovaných úloh používame nástroj Team Fundational Server (tfs), v ktorom naplníme backlog úlohami, ktoré si potom jednotliví členovia tímu vyberajú.

Práca v tíme je organizovaná do 2 týždňových šprintov. Na začiatku každého šprintu sa vytvárajú úlohy, ktoré sa dokončia daný šprint. Pri vytváraní úloh vzniká diskusia aby každému členu tímu bolo jasné čo sa ide implementovať. Následne scrum master organizuje scrum poker aby sa úlohy ohodnotili story pointami.

2.4 Manažment verziovania

Po dokončení ucelených funkcionalít sa tieto funkcionality spájajú do novej verzie projektu s novou funkcionalitou. Na management verziovania používame nástroj GitHub a prácu vo vetvách. Každý člen tímu pracuje vo svojej vetve vytvorenej podľa user story. 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.

2.5 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). Ako pomôcku pre vytváranie týchto testov, manager testovania vytvoril prehľadnú metodiku testovania v ktorej detailne popisuje ako vytvoriť tieto testy.

2.6 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 zdrojový kód bol kvalitný a prehľadný. Napísaný zdrojový kód sa po pull requeste kontroluje aj druhým členom tímu programátora v tíme s cieľom nájsť nedostatky v kóde a upovedomiť ho prostredníctvom komentáru, aby tieto nedostatky opravil. Tento postup ako postupovať pri kontrolovaní zdrojového kódu a celkový postup ku kontrole práce iného člena tímu je uvedená v metodike ku code review.

3 Sumarizácie šprintov

3.1 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 stranky	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 prisluchajuceho ...		Closed	Bc. Martin Olejar
Nastavenie zakladych 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 základov práce s git	2	Closed	Bc. Martin Olejar
Definovanie základov (pull, commit)		Closed	Bc. Martin Olejar
Definovanie práce s vetvami		Closed	Bc. Martin Olejar
Prihlasenie na Web stranke	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

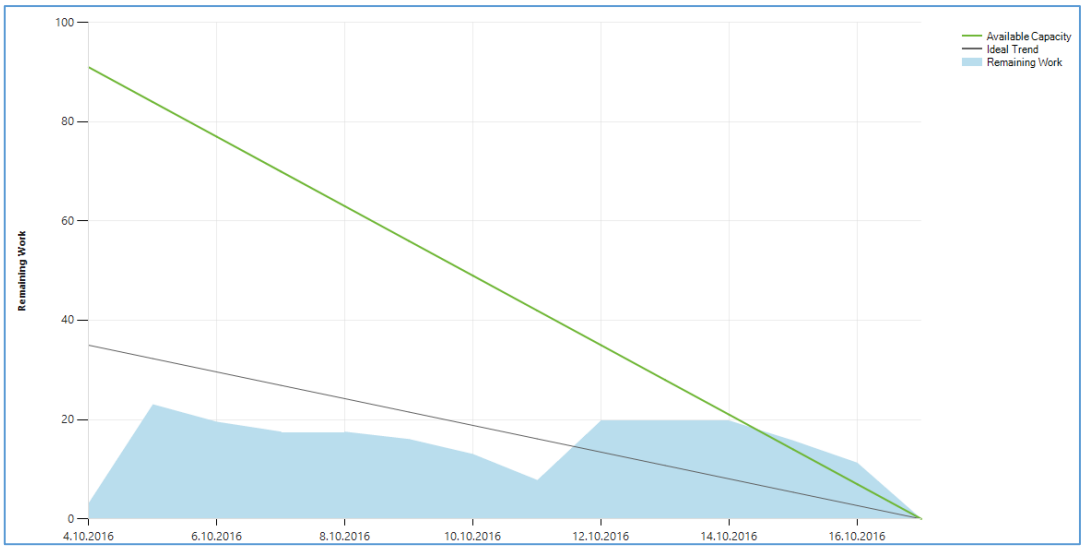
Obrázok 1: Úlohy šprintu Baldr

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 endoint 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
Priebeh 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 product ownerom) 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 mali pocit, že tento šprint obsahoval príliš málo úloh. To bolo spôsobené nesprávnym odhadom počtu bodov z dôvodu oboznamovania sa s novými technológiami pre niektorých členov tímu.

V tomto šprinte prebehlo nastavenie Continuous integration, 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

3.2 Šprint 2 – Fenrir

Vzhľadom na úspešné oboznámenie sa s technológiami sme sa rozhodli v druhom šprinte zvýšiť počet story 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.

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 Tellera	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
Odstranenie tokenu	2	Active	Bc. Ondrej Hamara
Vytvorenie REST endpointu pre zrusenie tokenu		Closed	Bc. Patrik Januska
Doplnenie hlasok pre angular		Active	Bc. Ondrej Hamara
Obnovenie hesla cez email	3	Closed	Bc. Miroslav Hurajt
Doplnenie emailovej sablony 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

Pouzivatel ma zakladne zobrazenie po prihlaseni, ktore mu bude p...	5	Closed	Bc. Adam Neupauer
Layout rozmiestnenia jednotlivych funkcnych prvkov		Closed	Bc. Adam Neupauer
Dodefinovat CSS		Closed	Bc. Adam Neupauer
Hlavna obrazovka s odnavigovanim na sucasny user config		Closed	Bc. Adam Neupauer
Nastylovanie User Configu		Closed	Bc. Adam Neupauer
Pouzivatel ma responzivne pouzivatelske rozhranie pri prihlaseni	5	Closed	Bc. Adam Neupauer
Framework pre responzivny dizajn		Closed	Bc. Adam Neupauer
CSS so spoločnými štýlmi		Closed	Bc. Adam Neupauer
Aplikovanie stylov na login a register screen		Closed	Bc. Adam Neupauer
Vytvorenie noveho projektu	5	Closed	Bc. Patrik Januska
Vytvorenie entity pre projekt		Closed	Bc. Patrik Januska
Formular pre zakladne detaily projektu		Closed	Bc. Patrik Januska
REST endpoint		Closed	Bc. Patrik Januska
Layout rozhrania projektu +CSS + HTML + JS		Closed	Jakub Ondik
Obnovenie aktivacneho tokenu	3	Closed	Bc. Martin Olejar
Vytvorenie REST endpointu pre znovuposlanie emailu		Closed	Bc. Martin Olejar
Vytvorenie AngularJS modulu pre znovuposlanie aktivacneho to...		Closed	Bc. Martin Olejar
Code Review a support pre sprint Fenrir		Closed	Jakub Ondik
CR a support - Jakub		Closed	Jakub Ondik
Prihláška	1	Closed	Jakub Ondik
Napisat prihlasku do TP cupu		Closed	Jakub Ondik
Odovzdat analogovu verziu		Closed	Bc. Martin Olejar

Obrázok 5: Úlohy šprintu Fenrir – pokračovanie

Výsledkom tohto šprintu bol kompletný používateľský modul vrátane bezpečnostných prvkov (odstránenie autentifikačného tokenu v prípade odcudzenia zariadenia). Taktiež sme sa rozhodli zúčastniť sa TP cupu.

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 product ownerom. Rast v začiatočnej časti grafu je spôsobený odhadovaním trvania úloh členmi až po začiatku šprintu.

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

Obrázok 6: Burndown chart pre šprint Fenrir

3.3 Šprint 3 – Freya

V tomto šprinte sme sa rozhodli zobrať vyšší počet story pointov, 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 remote 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 priraďovanie používateľských rolí. Taktiež sa úspešne podarilo nasadiť systémy SonarQube a Sentry a bol implementovaný systém notifikácií pre používateľov. Postup práce v tomto šprinte je možné vidieť na obrázku 9.

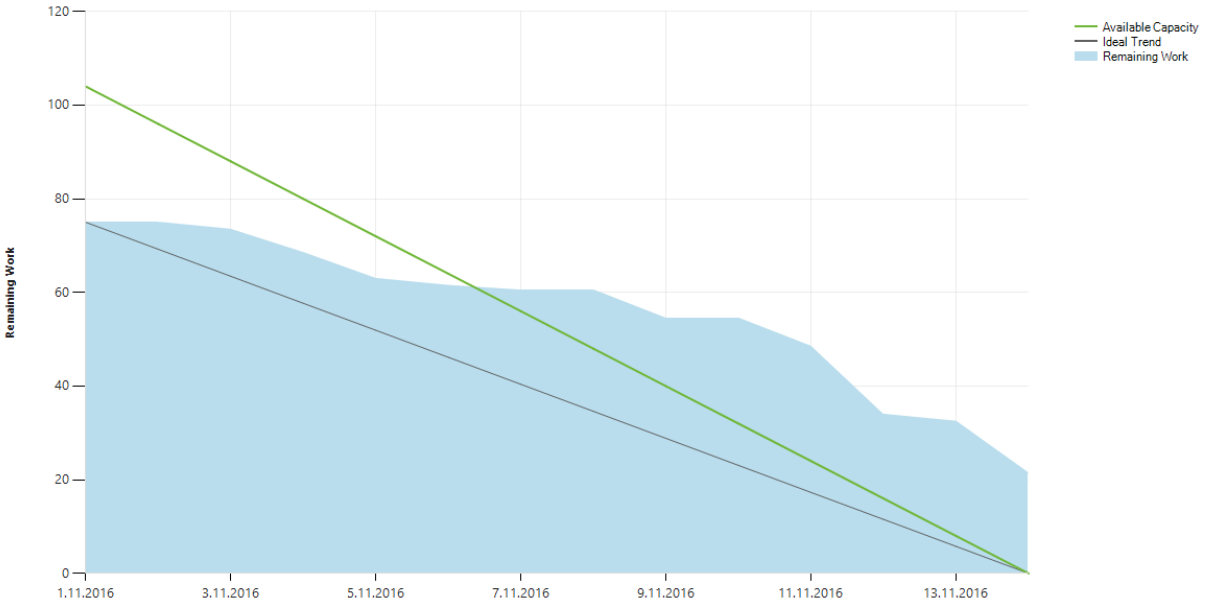
Veľkým negatívom tohto šprintu je nedokončenie pomerne veľkého počtu používateľských príbehov a úloh, čo bolo spôsobené nárazovou prácou tesne pred ukončením šprintu. Taktiež sa nepodarilo dokončiť ani jednu prenesenú úlohu z minulého šprintu, za ktoré bol zodpovedný jednotlivec, ktorý nedokončil (nezačal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

Title	Story Points	State	Assigned To
Odstranenie tokenu	2	Active	Bc. Ondrej Hamara
Doplnenie hlasok pre angular		Active	Bc. Ondrej Hamara
Prehlad uz existujucich projektov	5	Closed	Bc. Patrik Januska
Zmeny nad DB		Closed	Bc. Martin Olejar
REST endpoint na ziskanie zoradenych projektov pre prihlasene...		Closed	Bc. Patrik Januska
Zobrazenie zoznamu projektov		Closed	Bc. Patrik Januska
REST endpoint pre nacitanie informacii o konkretnom projekte		Closed	Jakub Ondik
Zobrazenie zoznamu používateľov v projekte	2	Closed	Bc. Patrik Januska
Zobrazenie zoznamu		Closed	Bc. Patrik Januska
REST endpoint na nacitanie zoznamu pouzivatelov v projekte		Closed	Bc. Patrik Januska
Zobrazenie zariadení, z ktorých bol používateľ prihlásený	3	Active	Bc. Martin Olejar
Vytvorenie komponentu pre AngularJS		Closed	Bc. Ondrej Hamara
Priradenie pouzivatela k role v ramci 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 notifikacii o prideleni role		Closed	Jakub Ondik
Definovanie roly administratora	3	Resolved	Bc. Adam Neupauer
Vytvorenie authorization metod a nastavenie anotacii pre rolu a...		Closed	Bc. Adam Neupauer
Vytvorenie angularJS modulu specifickeho pre administratora		Closed	Bc. Adam Neupauer
Notifikovanie pouzivatela o diani v StoryTelleri	8	Active	Jakub Ondik
DBO notifikacie		Closed	Bc. Miroslav Hurajt
Websocket modul pre Java na publish subscribe		Closed	Jakub Ondik
Templaty pre notifikacie		Active	Bc. Miroslav Hurajt
Angular modul na pripojenie na websockety ktory detekuje notif...		Closed	Jakub Ondik
Pozvanie do projektu cez StoryTeller	3	Active	Bc. Miroslav Hurajt
Rozhranie pre pridanie pouzivatela		Closed	Bc. Martin Olejar
REST endpoint pre príjem priradenia pouzivatela k projektu + sp...		New	Bc. Martin Olejar
Template na notifikaciu pozvania pouzivatela do projektu		Active	Bc. Miroslav Hurajt
Definovanie roly analytika	3	New	Bc. Ondrej Hamara
Vytvorenie authorization metod a nastavenie anotacii pre rolu a...		New	Bc. Ondrej Hamara
Vytvorenie angularJS modulu specifickeho pre analytika		New	Bc. Ondrej Hamara
Definovanie roly zakaznika	3	New	Bc. Ondrej Hamara
Vytvorenie authorization metod a nastavenie anotacii pre rolu z...		New	Bc. Ondrej Hamara
Vytvorenie angularJS modulu specifickeho pre zakaznika		New	Bc. Ondrej Hamara
Finalizacia inzinierskeho diela k prvemu kontrolnemu bodu	2	Closed	Jakub Ondik
Uvod		Closed	Bc. Martin Olejar
Globalne ciele projektu		Closed	Jakub Ondik
Celkovy pohlad na system		Closed	Jakub Ondik

Obrázok 7: Úlohy šprintu Freya

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
Konfigurácia SonarQube	3	Closed	Jakub Ondik
Konfigurácia serverovej casti		Closed	Jakub Ondik
Nastavit pom.xml		Closed	Jakub Ondik
Konfiguracia Sentry	5	Closed	Jakub Ondik
Nastavit bridge na urovni 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

4 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.

4.1 Metodika plánovania

Metodika hovorí o mikromenežmente úloh, teda o zodpovednosti človeka za používateľský príbeh. Zodpovedný človek by mal po aktivovaní používateľského príbehu zorganizovať stretnutie, na ktorom sa rozoberie obsah jednotlivých úloh, teda vykoná sa 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!AnORgXjFAcOEhUAdTdwsVIhiO0Ib

4.2 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!AnORgXjFAcOEhEkdTdwsVIhiO0Ib

4.3 Metodika testovania zdrojového kódu

Metodika hovorí pravidlách testovania zdrojového kódu. Každý človek zodpovedný za úlohu vytvára a vykonáva k svojej úlohe príslušné testy. A práve ich vytváranie a 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!AnORgXjFAcOEhUIdTdwsVIhiO0Ib

4.4 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 code review, prečo ho robiť, cez moderné techniky až po samotný postup jeho 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

4.5 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!AnORgXjFAcOEhGUdTdwsVIhiO0Ib

4.6 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, číslovania a poznámok. Uvedená je tiež vzorová tabuľka.

Odkaz na metodiku: https://1drv.ms/b/s!AnORgXjFAcOEhGYdTdwsVIhiO0Ib

5 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 funkcionalita 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 dianiu v tíme,
- odkladanie práce na poslednú chvíľu pred koncom šprintu,
- nedostatočný opis a mikromanažment úloh, mikromanažment zahŕňa definovanie závislostí medzi úlohami, ich analýzu a návrh,
- 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.

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Motivačný dokument

Tím číslo 4

Jakub Ondik
Patrik Januška
Adam Neupauer
Martin Olejár
Ondrej Hamara
Miroslav Hurajt
Pavol Čurilla

tim_4@googlegroups.com
20. 9. 2016

Predstavenie tímu

Náš tím je zložený zo 6 absolventov bakalárskeho štúdia na STU FIIT a 1 absolventa bakalárskeho štúdia na TUKE FEI. My, absolventi bakalárskeho štúdia na STU FIIT, sa veľmi dobre poznáme už od 1. ročníka a v rámci štúdia sme spolupracovali na viacerých projektoch, čím sme si spoločne vyskúšali prácu v tíme. Taktiež sa spolu stretávame a radi si navzájom pomáhame s lepším pochopením učiva alebo vyriešením problémov týkajúcich sa zadaní a projektov. Medzi silné stránky nášho tímu patria určite komunikatívnosť a skúsenosti nadobudnuté dlhoročnou spoluprácou. Patríme medzi študentov, ktorí sa snažia venovať študentským povinnostiam čo najviac času a riešiť zadania a projekty v dostatočnom predstihu pred ich odovzdaním v nadpriemernej kvalite. Počas predchádzajúceho štúdia sme sa snažili nielen vypracovať projekty, ale získať pri tom čo najviac nových poznatkov, ktoré môžeme v tímovom projekte využiť.

Takmer všetci sme sa rozhodli pokračovať v štúdiu v odbore Softvérové inžinierstvo. Medzi naše vybrané predmety inžinierskeho štúdia, ktoré súvisia s preferovanými témami, patria Architektúra softvérových systémov, Objektovo orientovaná analýza a návrh softvéru, Pokročilé databázové technológie a taktiež Objavovanie znalostí.

V rámci nášho štúdia, ale aj mimo neho sme používali vo viacerých projektoch programovacie jazyky C, Python a v prvom rade Javu. Ovládame prácu s databázami PostgreSQL, MySQL a MongoDB a niektorí členovia tímu aj webový framework AngularJS. Niektorí z nás majú skúsenosti s jazykom C# a architektúrou .NET, ktoré využili pri vypracovaní bakalárskej práce. V tímovom projekte sme ochotní získať mnoho nových vedomostí, tvorivo pracovať a vytvoriť čo najlepší výsledok. Ako tím sme plní očakávaní a nebojíme sa nových výziev.

1. Zber a vyhodnocovanie požiadaviek (Story Teller)

Väčšina nášho tímu má aj z praxe skúsenosti, ako dokážu nepresne definované požiadavky komplikovať samostatnú implementáciu daného systému, čo v dôsledku znamená, že budú premrhané zdroje určené na projekt z pohľadu všetkých zainteresovaných strán. Dá sa teda povedať, že nepresné požiadavky sú jedným z najzávažnejších faktorov, ktoré spôsobujú neúspech pri projektoch.

Skicovanie a komentovanie funkcionálnych požiadaviek výrazne podporuje vyjadrovaciu schopnosť zákazníka. Na druhej strane, vygenerované automatizované akceptačné testy jednoznačne určujú podmienky akceptovania systému zákazníkom a zjednodušujú sledovateľnosť progresu projektu dodávateľom.

Keďže takmer všetci v našom tíme študujeme odbor Softvérové inžinierstvo, vidíme v tomto projekte veľa možností, prostredníctvom ktorých môžeme získať nové poznatky a skúsenosti v tomto odbore a radi by sme pracovali na projekte, ktorý sa pokúša o progres v tejto oblasti. Časť z nás sa okrem samotnej implementácie softvéru zaujíma o návrh architektúr softvérových systémov, čím dokážeme pokryť všetky fázy vývoja softvéru.

Prvotný návrh

Backend časť systému by sme chceli riešiť v Jave nad rámcom Spring Boot ako REST webovú službu. Toto umožňuje flexibilitu z pohľadu klientských aplikácií, či to bude webová aplikácia, desktopový tenký klient, rozšírenie do CASE nástrojov alebo mobilná aplikácia. Taktiež to umožňuje potenciálnu integráciu do alebo s inými nástrojmi na podporu vývoja softvéru.

Webový klient plánujeme implementovať ako JavaScriptovú single-page aplikáciu, čo umožní vysokú interaktivitu s používateľom. V neskorších etapách plánujeme podporu paralelnej editácie a spolupráce na skiciach implementované pomocou websocketov alebo protobufferového (gRPC) rámca v prípade využitia HTTP2, prípadne samostatnej optimalizovanej aplikácie pre tablety a mobilné zariadenia.

Detekciu komentárov a poznámok v skiciach chceme riešiť pomocou OCR technológií, prípadne poskytneme nástroj na vpísanie komentárov a popisov priamo pomocou virtuálnej (alebo aj hardvérovej) klávesnice. Konverziu skíc obrazoviek chceme riešiť pomocou nástroja na rozpoznanie geometrických tvarov, ktorý chceme upraviť priamo pre naše potreby. Taktiež plánujeme použiť už existujúce nástroje, ktoré sú uvedené ako odporúčané technológie v zadaní projektu - Cucumber a Selenium, ktoré sú kompatibilné s Javou (Spring rámcom).

2. Inteligentný sklad (SmartStore)

Hlavnou motiváciou tejto témy je zrýchlenie doby dodania tovaru, čo je kľúčovým aspektom obchodnej sféry. Uvedomujeme si, že je v tejto oblasti veľmi dôležité rýchlo vybavovať objednávky zákazníkov. V opačnom prípade je dnes realitou, že zákazníci odchádzajú ku konkurenčným firmám, čo znižuje zisk firmy. Zavedenie inteligentného skladu poskytne nielen prehľadné usporiadanie tovaru, ale aj jeho efektívne vyhľadávanie. Hlavným dôvodom, prečo nás táto téma zaujala, je možnosť riešiť problém, ktorý sa reálne vyskytuje v praxi, a možnosť spolupráce s firmou Martinus.sk. Projekt by uľahčil aj prácu spojenú s redistribúciou tovarov medzi skladmi.

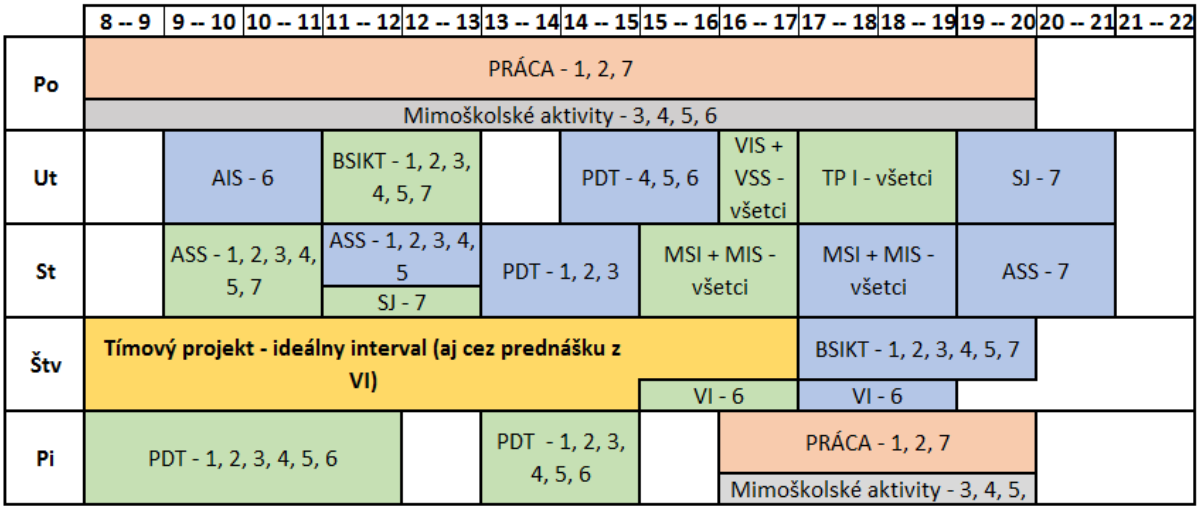
Členovia nášho tímu disponujú v dostatočnej miere znalosťami webových technológií, ktoré by sme chceli v rámci tohto projektu rozšíriť. V tomto projekte by sme vedeli využiť široké vedomosti nadobudnuté v bakalárskom štúdiu, najmä v oblasti návrhu softvéru a zefektívňovania algoritmov.

Našou ambíciou v tomto projekte je okrem základných požiadaviek vyriešiť aj náročnejšie problémy spojené s organizáciou skladu, čím by sme sa mohli priblížiť vysoko inteligentnému skladu, ktorého príkladom je Amazon.

Príloha A: Zoradenie tém podľa priority

1. Zber a vyhodnocovanie požiadaviek [Story Teller]
2. Inteligentný sklad [SmartStore]
3. Pomôcky pre aktívnych programátorov [CodeCrutches]
4. Servisný modul pre stratosférický balón [StratosFIIT]
5. Manažment zdravotného stavu pacienta prostredníctvom monitoringu emócií [eMotion]
6. Nový návrh systému pre MOD [Future MOD]
7. Simulácia správania dronov v roji [DronSim V2]
8. Prepájanie dát o vývoji softvéru [TRACKS]
9. Extrakcia dát z webu [WebExtraction]
10. Tvorba vzdelávacích simulácií [EduSim]
11. Vyhľadávanie so sémantikou [DeepSearch]
12. 3D UML, improved version [3D UML]
13. Monitorovanie a vyhodnocovanie fyziologických procesov človeka [StresMonitor]
14. Monitorovanie bezpečnostných udalostí [SecMon]
15. Vizualizácia informácií v obohatenej realite [AugReality]
16. Rekonštrukcia 3D scény [3D-Recon V2]
17. 3D simulovaný robotický futbal [3D futbal]
18. Aplikácia deterministického ethernetu pre distribuovaný vnorený systém [Slovak TTTech]
19. Navigácia v budove [VirtualFEI]
20. Kam na obed 2.0 [FoodCourt]
21. Automatická podpora moderácie diskusií [ModerateIT]

Príloha B: Rozvrh tímu



1. Jakub
2. Adam
3. Patrik
4. Martin
5. Ondrej
6. Miroslav
7. Pavol

Cvičenie

Prednáška

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@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 úloh

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 osobné stretnutie. Priebeh stretnutia je napísaný v nasledujúcich odsekoch.

Na stretnutí je potrebné rozobrať si 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 Stavy 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 – dáva sa vtedy, keď sa začne pracovať na používateľskom príbehu
- CODE REVIEW – dáva sa vtedy, keď sú dokončené všetky úlohy a bol vytvorený pull request
- RESOLVED – dáva sa vtedy, keď bol pull request schválený
- CLOSED – nedáva sa, tento stav môže určiť len product owner

1.3 Ďalšie pravidlá

Pre riešenie používateľských príbehov platia ďalšie pravidlá:

- 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,
- úlohu netreba dávať do stavu *Active,* ak sa na nej nepracuje.

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

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
Vypracoval: 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 camel case. Tieto názvy musia byť výstižné a nesmú obsahovať nejednoznačné a nevšeobecne 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 nevšeobecne 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,** pokiaľ na použitie iného modifikátora prístupu nemá autor dobrý dôvod, ako je to v prípade 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 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 dlžku 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 isTokenValid(String token, UserDetails userDetails) {
  ...
}
```

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

Príklad:

```
// jednoriadkovy komentar
```

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

Príklad:

```
/* viac
 * riadkovy
 * komentar
 */
```

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 logovanie 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 loggera 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 notáciu @RequestMapping s cestou podľa názvu daného zdroja 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 explicitné 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 jendnotlivé 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

Odchytávanie 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 nofitications", e);
}
```

Objekt loggera (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.

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

Tímový projekt

Story Teller

Metodika testovania zdrojového kódu

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
Vypracoval: 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 správnemu 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.

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 **aplication.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. Ak sa nám po zadaní konkrétnej URL cesty 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 ku code review

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
Vypracoval: 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 jeho čitateľnosti. Pripomienky často obsahujú vylepšenia, ako efektívnejšie napísať kód, prípadne poznamenať na chýbajúce komentáre, použitie implementovanej funkcie a pod. Hlavnou výhodou prehliadky kódu je efektívne zastúpenie programátora pri jeho absencii, človekom, ktorý mu robil prehliadku kódu.

1.3 Postup pri prehliadke kódu

V prehliadke kódu vystupujú dve roly – autor zmeny a človek, ktorý vykonáva prehliadku kódu. Všeobecný postup pri prehliadke kódu je nasledovný: Autor upraví kód, ktorý skontroluje človek vykonávajúci prehliadku kódu, ak je zmena prijateľná, autor je šťastný a má pokoj, inak musí nájdené nedostatky opraviť. Cyklus pokračuje dovtedy, kým nie je človek vykonávajúci prehliadku kódu spokojný so zmenami.

1.3.1 Ako poznať, či je zmena pripravená na prehliadku kódu?

Prvom rade musí byť zmena izolovaná, to znamená, že zmena musí fungovať bez akýchkoľvek zmien v pracovnej vetve. To znamená, že pred začatím programovania je potrebné pre danú Story vytvoriť vetvu a zmeny vykonávať v nej. Zmena musí byť patrične otestovaná unit testami (na beckende) a tiež časti kódu zdokumentované (javadoc). Pokiaľ niečo chýba nedávať commit. Zbytočne si autor bude myslieť, že je problém vyriešený aj keď v skutočnosti nie je, čo vyústi, že pull request pri spájaní vetiev bude zamietnutý. V danej vetve robíme malé a časte comitty, aby sme predošli zbytočným kolíziám. Taktiež je ľahšie kontrolovať kód s menším počtom zmien, pri väčšom počte zmien vzniká riziko prehliadnutia chyby.

1.3.2 Ako si nechať skontrolovať kód?

Pred začatím vykonávania zmien je potrebné pulnuť všetky aktuálne zmeny. Za každú story zodpovedá jeden človek. Jeho úlohou je stanovenie závislosti taskov a povedať, ktoré úlohy je potrebné uprednostniť. Ak pracuje na jednej story viacero ľudí je potrebné dohodnúť sa na pridelení taskov a vysvetlenia si, čo sa ide robiť, aký je cieľ. Po ukončení práce na danej vetve, zodpovedný človek za vetvu skontroluje kód, poprípade da vedieť dotyčnej osobe pracujúcej

na vetve, aby ho zmenila. Následne vykoná pull request na spojenie danej vetvy s vetvou dev. Pull request vykonáva kontrolór kvality kódu, v našom prípade Jakub Ondik, ktorý vykoná prehliadku kódu a okomentuje nedostatky, poprípade môže napísať aj iné vhodné riešenia. Osoba, ktorá požiadala o pull request (väčšinou osoba zodpovedajúca za story) za pomoci ľudí pracujúcich na vetve, opraví nedostatky. Potom znovu vykoná pull request. To sa opakuje pokým nie sú odstránené všetky nedostatky a kontrolór kvality kódu nespojí danú vetvu s dev. Spojenie vetvy dev s masterom vykoná vždy na konci šprintu kontrolór kvality kódu (Jakub) alebo osoba ním poverená.

1.4 Pohľad človeka vykonávajúceho prehliadku kódu.

Prvom rade je rozumieť kódu. Ak človek vykonávajúci prehliadku kódu 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.

Zoznam vecí, ktoré kontrolujeme:

- Coding conventions
- Testy a Jdoc
- Miesto zmeny
- Don’t repeat yourself
- Ošetrenie chybových stavov

1.4.1 Coding conventions

Prvou vecou, ktorú kontrolujeme je dodržiavanie coding conventions. Úprava kódu je prvou vecou, ktorú človek vidí. Pokiaľ každý súbor vyzerá inak alebo jedna vec v súbore je napísaná tromi rozdielnymi spôsobmi, nepokračovať ďalej v prehliadke kódu a okamžite ho vrátiť autorovi. Podrobne opísané coding conventions sa nachádzajú v súbore Coding Standards.pdf.

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 by bolo testovanie náročné, tak to značí problém v navrhnutí kódu.
Pokiaľ zmeny obsahujú testy, pozeráme, či naozaj testujú, čo majú. Tiež sledujeme ako budú testy odolné voči budúcim zmenám, pretože veľa testov závisí od hardvérových parametrov počítača.
Bez patričnej dokumentácie sa ťažko zisťuje, čo autor myslel. Tiež sledujeme syntax zapisovania, správne odsadenie a opísanie atribútov. V prípade nedostatkoch v testoch alebo dokumentácií ďalej nepokračovať a vrátiť 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 Don’t repeat yourself (DRY)

Takmer poslednou vecou kontrolujem, či som podobný kus kódu už predtým nevidel. Ak áno, tak je duplicitný a zhoršuje údržbu. Riešením je presunutie zdieľanej časti do metódy, triedy alebo komponentu.

1.4.5 Ošetrenie 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

- Komunikácia
    - dôležité je výsledné dielo, nie dokazovať kto je lepší
    - ku kritike písať aj to, ako by sa zmena dala urobiť lepšie
- Nebuďte osobný
    - neriešte schopnosti iného členu tímu
    - 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 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@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 píšu 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
    - 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"*
- 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 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 funkcionalitu, 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.fiit.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 zobraziť 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:

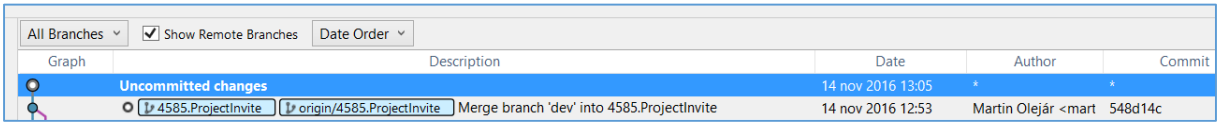
1. v SourceTree klikneme v hornom menu na **Clone / New,**
2. vyplníme **Source Path / URL** nasledujúcim odkazom:
   https://tfs.fiit.stuba.sk:8443/tfs/StudentsProjects/StoryTeller/_git/StoryTeller
3. vyplníme **Destination Path**, t.j. cestu k priečinku, kde bude uložený lokálny repozitár,
4. potvrdíme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

3.2 Práca vo vetve

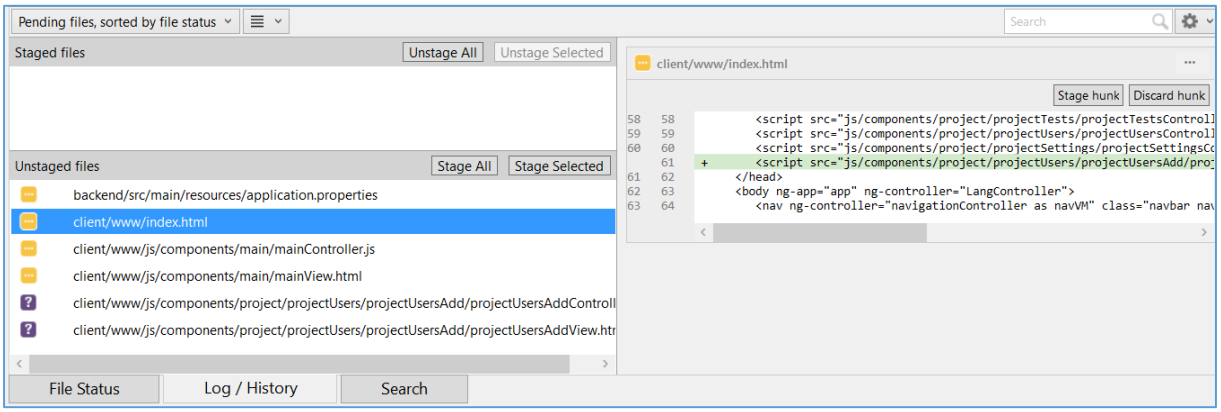
Pracovnú vetvu si otvoríme tak, že v časti BRANCHES klikneme dvakrát na danú vetvu. Pri úplne prvom otváraní vetvy je nutné túto akciu vykonať v časti REMOTES.

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

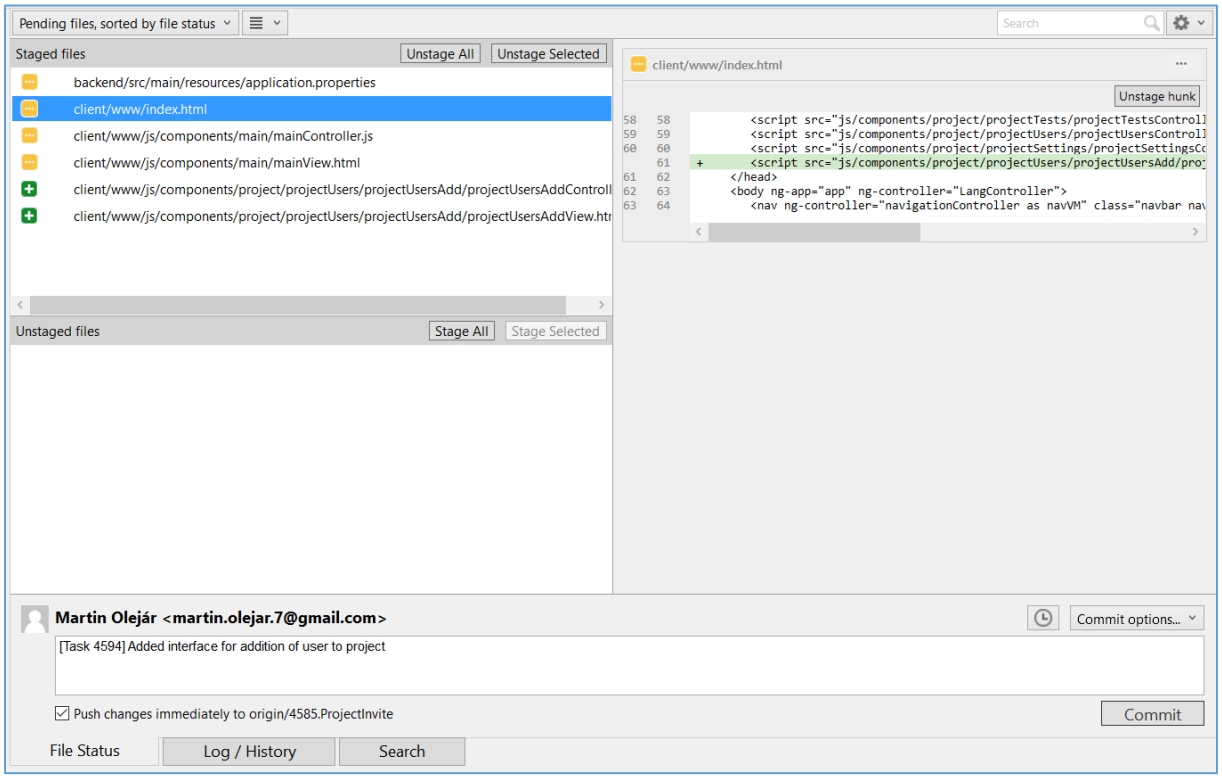


Obrázok 2: Zobrazenie zmien vo vetve

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:

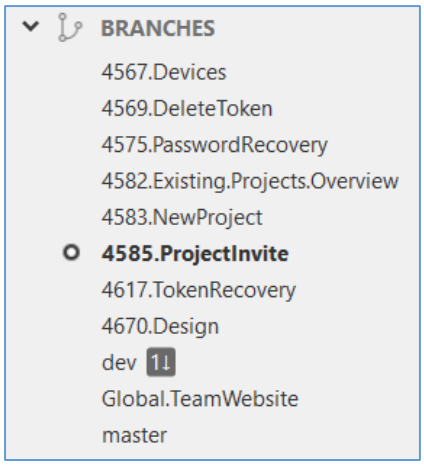
1. klikneme v dolnej časti na **File Status,**
2. v časti **Unstaged files** uvidíme všetky modifikované súbory, klikneme na **Stage All,**
3. v dolnej časti vyplníme popis commitu podľa definovaných pravidiel,
4. zaškrtneme **Push changes immediately to ...,**
5. výsledná situácia je zobrazená na obr.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



Obrázok 3: Odovzdanie zmien

3.4 Stiahnutie aktuálnej verzie vetvy

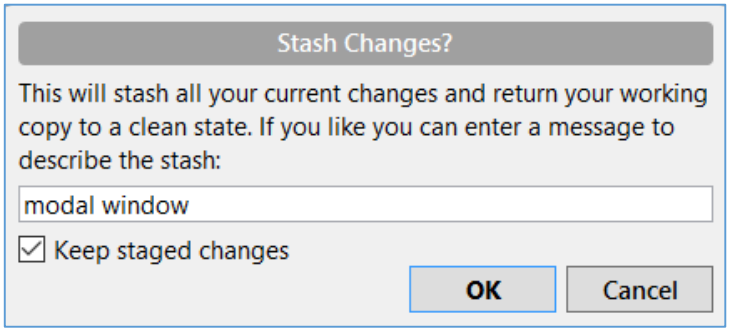
Ak niekto iný vykonal zmeny vo vetvách, je potrebné si tieto zmeny stiahnuť. Najprv klikneme na **Fetch** v hornom menu. Pri každej vetve, ktorá nie je lokálne aktuálna, teda niekto iný ju modifikoval a vykonal commit a jeho zmeny nie sú v lokálnom repozitári, sa zobrazí počet vykonaných commitov a šípka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



Obrázok 4: Zobrazenie aktuálnych vetiev

3.5 Odloženie vykonaných zmien vo vetve

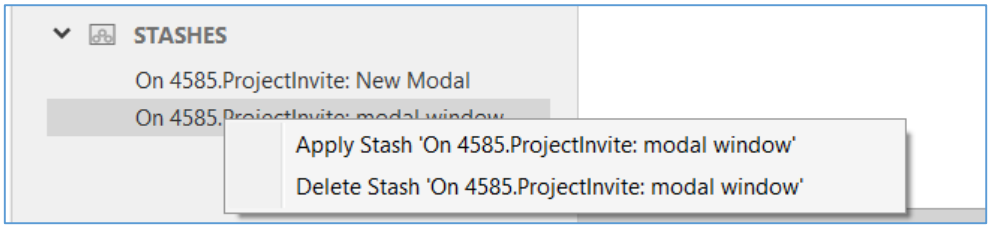
Ak pracujeme v nejakej vetve a potrebujeme vykonať nejaké zmeny v inej vetve, teda musíme sa prepnúť do inej vetvy, je potrebné si predtým vykonané zmeny odložiť (angl. **stash**). V hornom menu klikneme na **Stash**, zobrazí sa modálne okno na obr. 5, v ktorom vyplníme správu pre popis stash-u a zaškrtneme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

Metodika k dokumentácii

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
Vypracoval: Bc. Martin Olejár

Obsah

1 Kapitola číslo 1

1.1 Metodika

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
    - fsdfsd
        - fsdfsdf

Príklad číslovania:

1. fgffdd
2. gfdgfg
    a. fgdfgfdf
    b. fdgfdgdfgfd

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 Bez riadkovania

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@googlegroups.com
Vypracoval: Bc. Adam Neupauer

1 Metodika integrácie a nasadzovania

Tento dokument opisu akým spôsobom sú nasadzované a integrované aplikácie vyvýjané v tímovom projete.

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(branch), 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 Continuous Integration. 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 servery 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@googlegroups.com
Vypracoval: Bc. Miroslav Hurajt

1 Komunikačné kanály

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

1.1 Slack

Najfrekventovanejšie využívaný komunikačný kanál. Pokrýva najväčšiu časť komunikácie v tíme. Obsahuje rôzne menšie kanály (channels) podľa obsahu komunikácie:

- **#announcements** – do tohto kanálu píšeme oznámenia pre všetkých členov tímu napr. o povinnostiach, ktoré ešte treba dokončiť alebo informovať tím o pridaní do rôznych skupín používaných technológií,
- **#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,
- **#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,
- **#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,
- **#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 alebo chodu tímu,
- **#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,
- **#tp_cup** – tu sa vkladajú informácie spojené so súťažou Tp-cup - 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 Fundation Server v časti „pull request“ sekcie „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 tímové projektu môžeme diskutovať aj vo vytvorenej chatovacej 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 použime ako komunikačný kanál Slack a časť #offtopic hlavne z dôvodu, že sa v Slacku ľahšie vyhľadáva a k diskutovaným veciam 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 na jeho uvedenej emailovej školskej adrese.

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@googlegroups.com
Akademický rok: 2016/2017
Dátum odovzdania: 16. 11. 2016

Obsah

Úvod

Tento dokument predstavuje dokumentáciu inžinierskeho 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 funkcionalitu 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 funkcionalitou 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.

1 Globálne ciele projektu

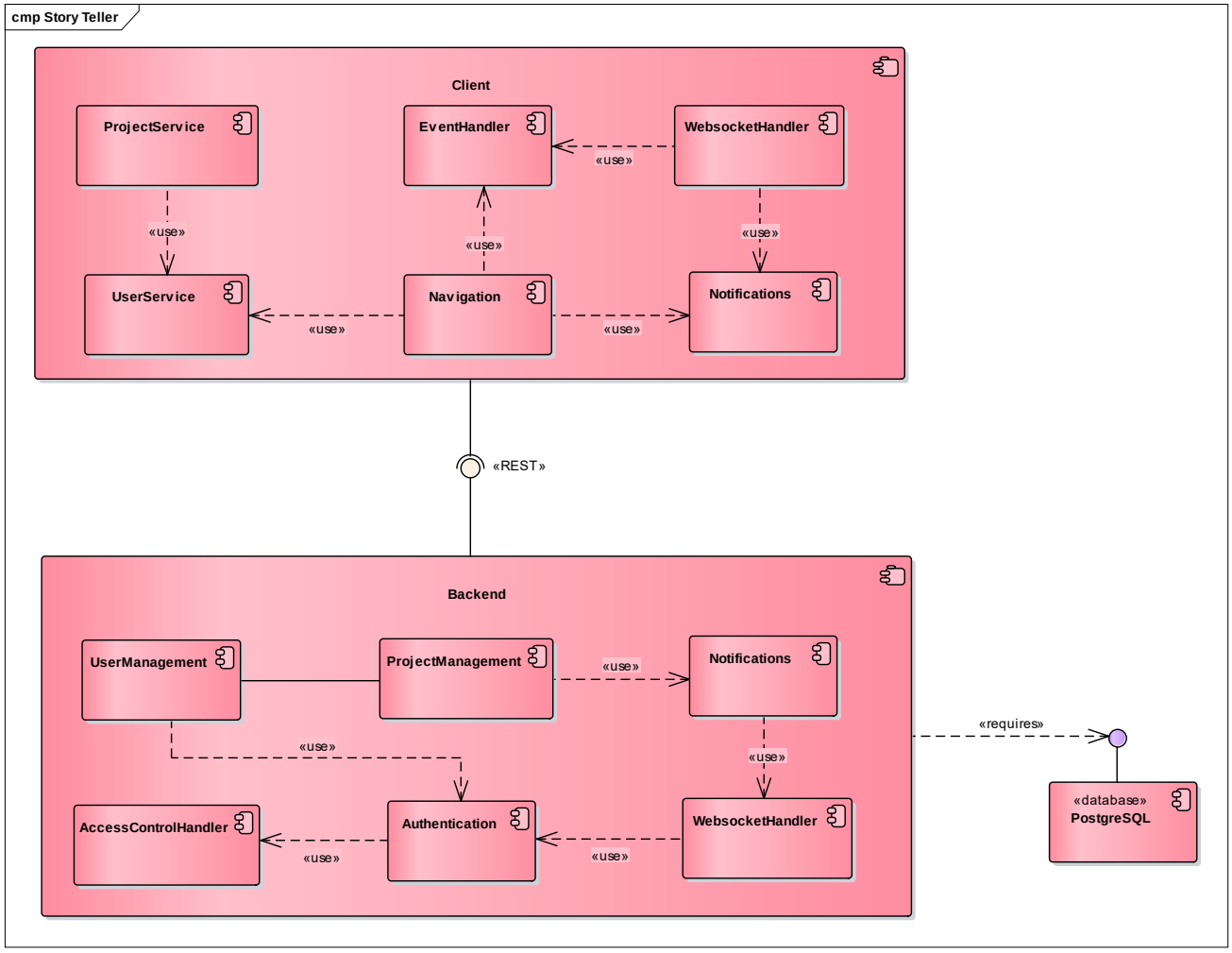
Naším cieľom pre zimný semester je vytvorenie prototypu obsahujúceho základnu logiku pre vytváranie projektov a skíc, vrátane prvotného generovania testov.

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. Centrálne vykonávanie akceptačných testov aspoň pre jednu platformu
8. Možnosť sledovania vykonávania (výsledkov) akceptačných testov

2 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

2.1 Backend

2.1.1 UserManagement

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

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

2.1.2 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

2.1.3 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

2.1.4 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

2.1.5 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

2.1.6 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

2.1.7 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.

2.2 Client

2.2.1 UserService

Komponent UserService zabezpečuje registráciu a prihlásenie, zaslanie požiadavky na obnovu hesla a aktivačných, resp. deaktivačných 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

2.2.2 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

2.2.3 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

2.2.4 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.

2.2.5 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.

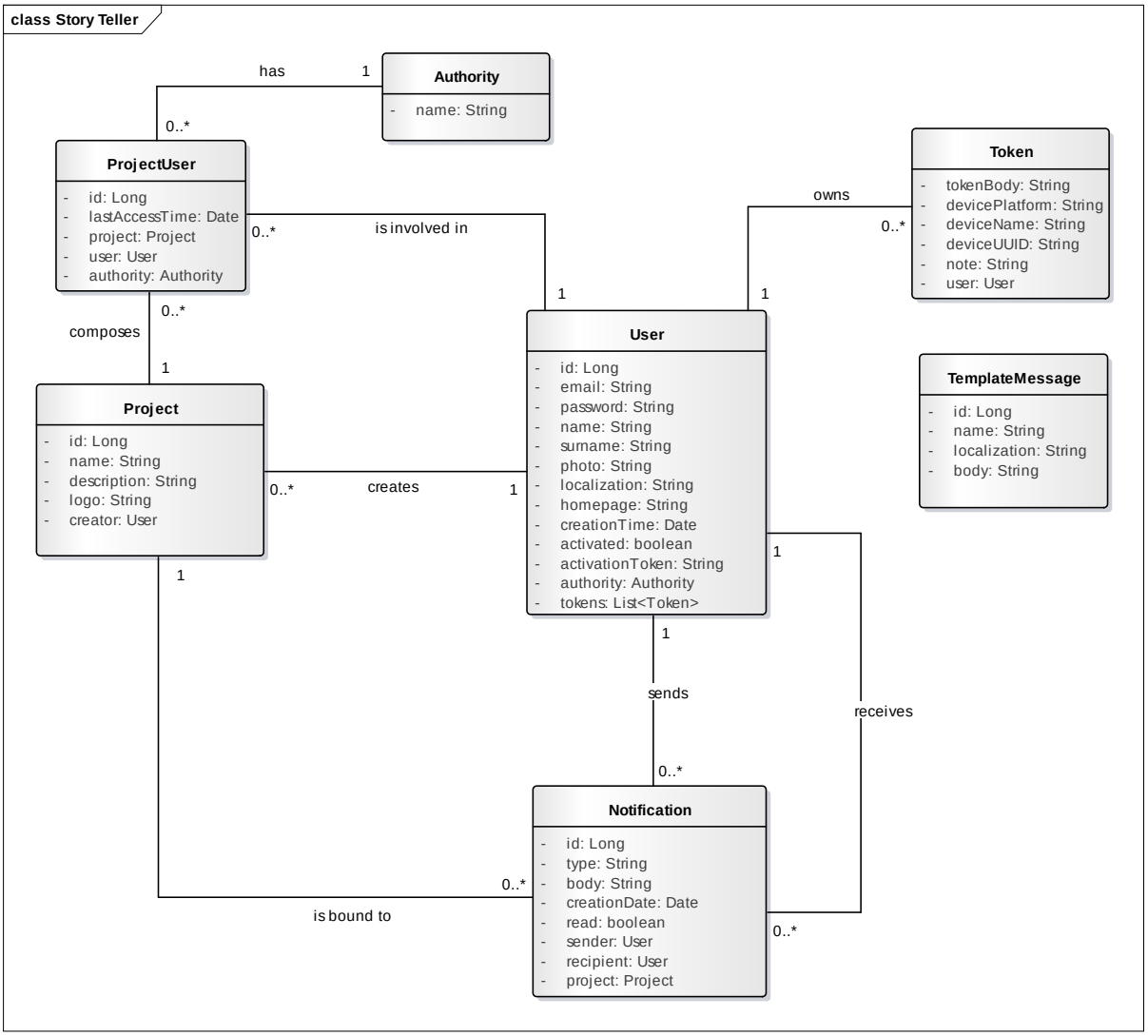
2.2.6 Notifications

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

2.3 Dátový model



Obrázok 2: Dátový model aplikácie

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 spájaciu 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.

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@googlegroups.com
Akademický rok: 2016/2017
Dátum odovzdania: 16. 11. 2016

Obsah

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.

1 Modul – Používateľ

1.1 Analýza

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.

1.2 Návrh

Na realizáciu všetkých opísaných bodov je nutné navrhnúť 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 deaktivačný 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.

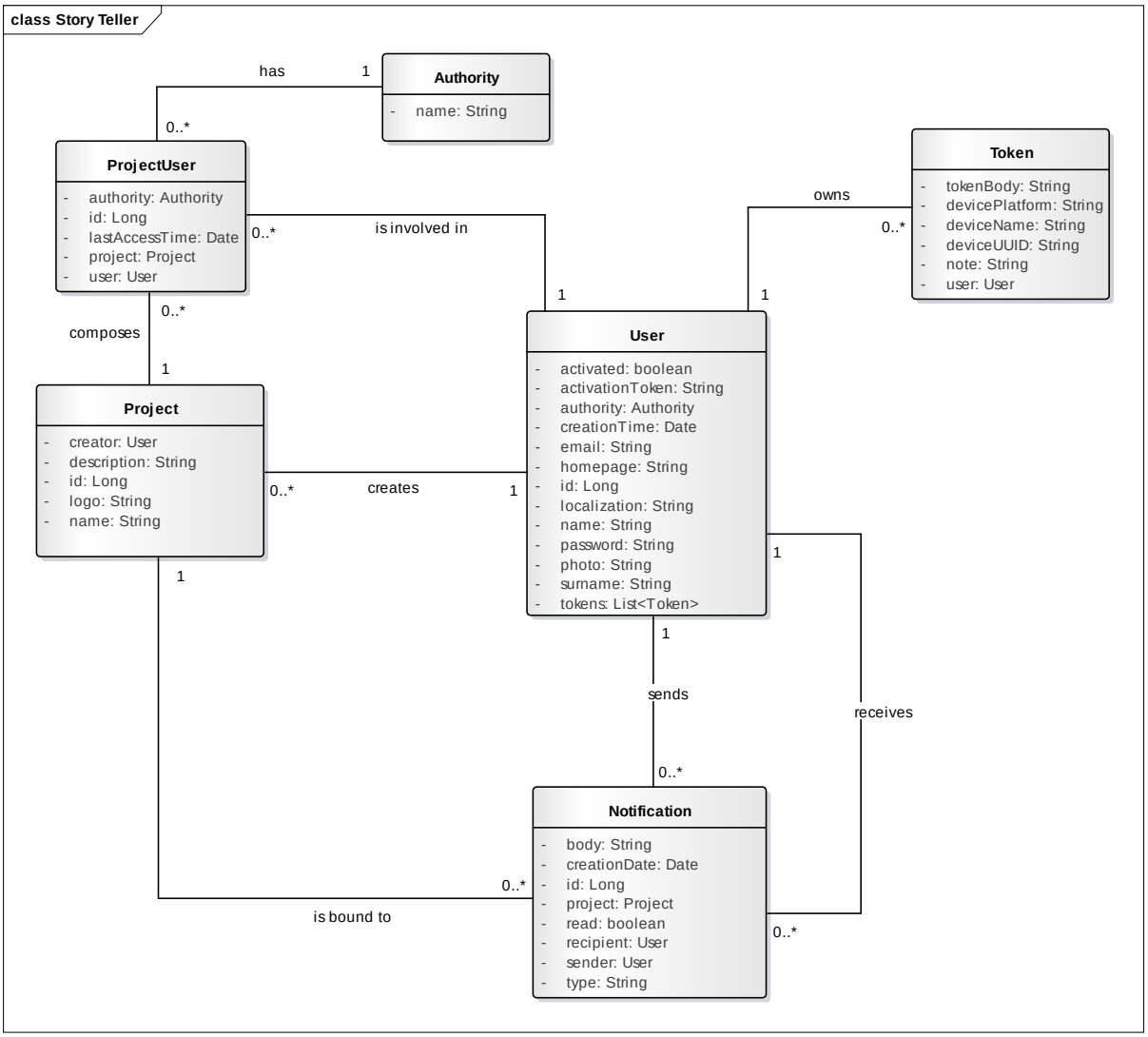
Používateľ bude mať možnosť vyplniť základné informácie o sebe vo svojom profile – meno, priezvisko, homepage a fotografiu a bude mať možnosť zmeniť si heslo. V prípade straty alebo zabudnutia hesla bude mať možnosť odoslania nového hesla na emailovú adresu, pomocou ktorej sa zaregistroval.

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.

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

1.3.1 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)** ošetrujúcou minimálnu a maximálnu dĺžku reťazcov.

Po úspešnej registrácii je používateľovi zaslaný email prostredníctvom **zoho.com** z adresy **noreply@story-teller.xyz** obsahujúci aktivačný a deaktivačný 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}**.

1.3.2 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.

1.3.3 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.

1.3.4 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@story-teller.xyz pomocou metódy **sendPasswordRecoveryEmail(User user, String password)** v triede **MailService**.

1.3.5 Obnova registračného emailu

Systém poskytuje taktiež možnosť opätovného zaslania emailu s aktivačným a deaktivačným odkazom. Táto funkcionalita 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**.

1.3.6 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**.

1.3.7 Riadenie prístupu

Riadenie prístupu k údajom v databáze je realizované pomocou prístupových anotácií **@PreAuthorize.** Tie sú použité pri každom REST endpointe, ktorý pristupuje ku kritickým dátam, špecifickým pre konkrétnych používateľov. Autorizačné anotácie majú tvar : **@PreAuthorize("@roleSercurityService.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.

1.3.8 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**.

1.3.9 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 U**serService** 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**.

2 Modul – Lokalizácia

2.1 Analýza

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

2.2 Návrh

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.

2.3 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**, viď ukážka 1) do objektu typu **Properties**, ktorý je následne serializovaný na JSON a spracovaný modulom ng-translate v klientskej AngularJS aplikácií.

```
myprofile.success.login=You were logged in successfuly
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

3 Konfigurácia - Server a nasadzovanie

3.1 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 vykreslovanie 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**

3.2 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.

3.3 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.

4 Modul – Projekt

4.1 Analýza

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

4.2 Návrh

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

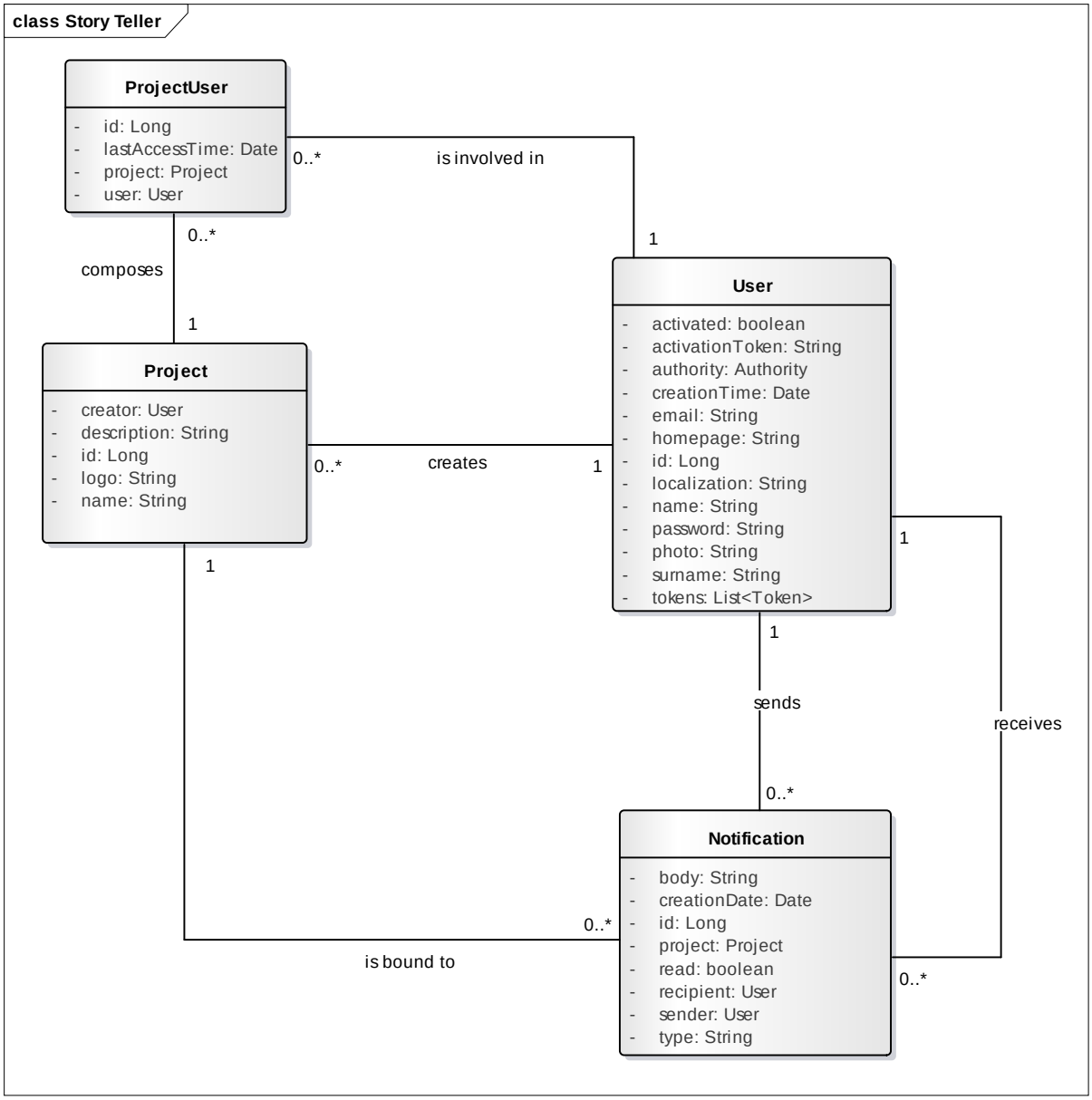
Možnosť vytvárať projekty bude sprístupnená prihláseným používateľom. Pri vytváraní projektu bude používateľ vypĺňať príslušný formulár. Vo formulári bude potrebné zadať názov projektu, opis projektu a prípadne vlastné logo projektu. Následne po odoslaní formulára na spracovanie sa k projektu priradí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

4.3 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 nepristupuje priamo cez repozitáre, ale cez servisy, ktoré následne v transakciách pristupujú k dátam. Týmto je dodržaný princíp viacvrstvovej architektúry.



Obrázok 2: Dátový model projektu

4.3.1 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**.

4.3.2 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**.

4.3.3 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 sú roly úspešne načítané. Na základe filtrov Roly sú z databázy vyťahované prostredníctvom triedy **AuthorityRepository**. Tá pristupuje k entite **Authority.**

4.3.4 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.

5 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 elemtontov 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/

6 Modul – Notifikácie

6.1 Analýza

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

6.2 Návrh

Na prenos notifikácií je možné použiť websockety s modelom publish – subscribe. Z dôvodu viacerých typov notifikácií je nutné navrhnúť 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.

6.3 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**.

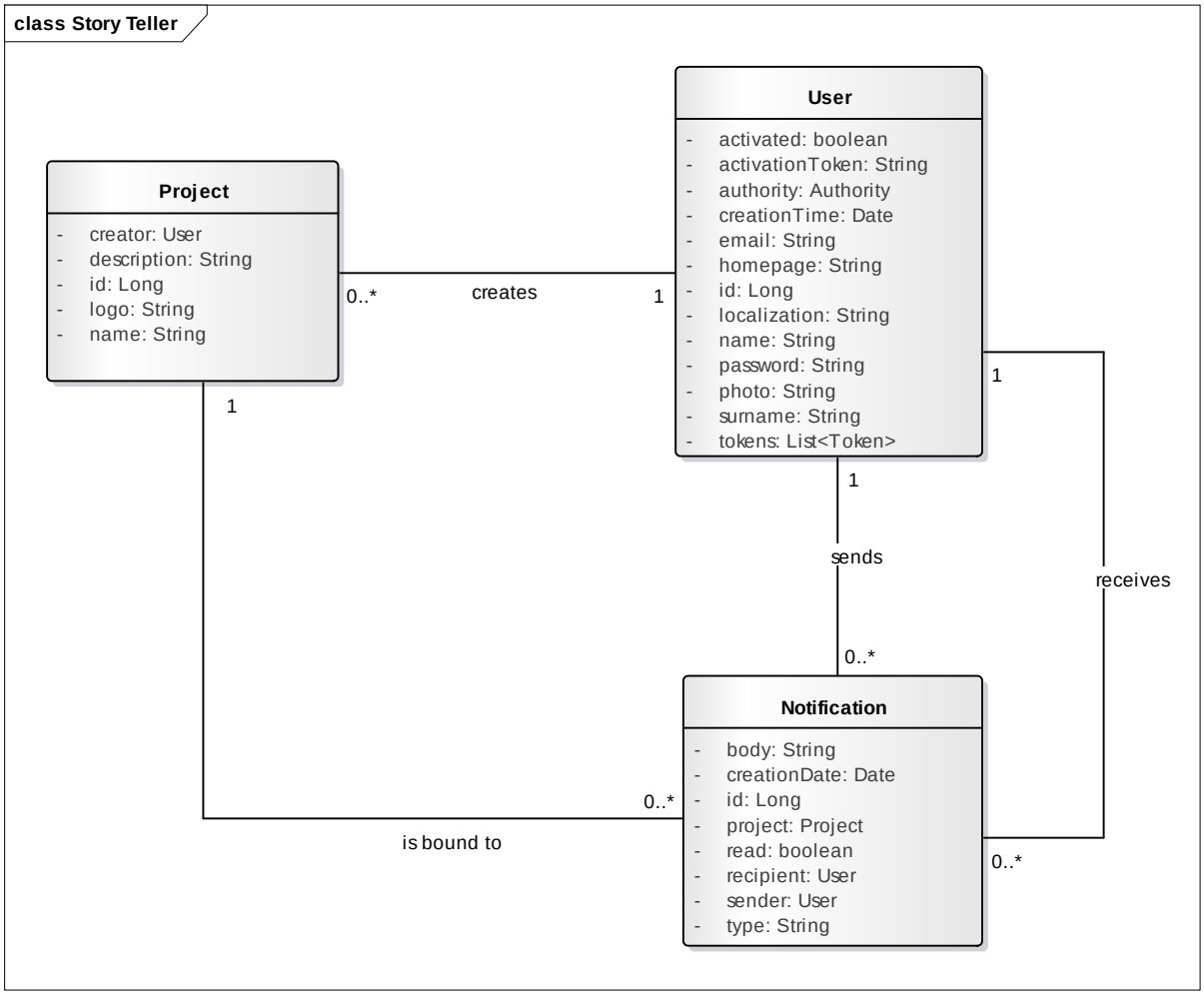
6.3.1 Odosielanie notifikácií

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.

6.3.2 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 panely nachádzajúcom sa na vrchu aplikácie a indikuje novú notifikáciu animáciou zatrasenia ikony notifikácií.

6.3.3 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 prezerať notifikácie cez otváracie popup okno, ktoré sa zobrazí kliknutím na ikonu notifikácií v navigačnom panely – 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 zobraziť.

Označovanie notifikácií za videné prebieha tak, že notifikácie, ktoré boli zobrazené cez otváracie popup okno sú odoslané backendovej aplikácií, 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.

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

Tímový projekt

Story Teller

Projektová dokumentácia – používateľská príručka

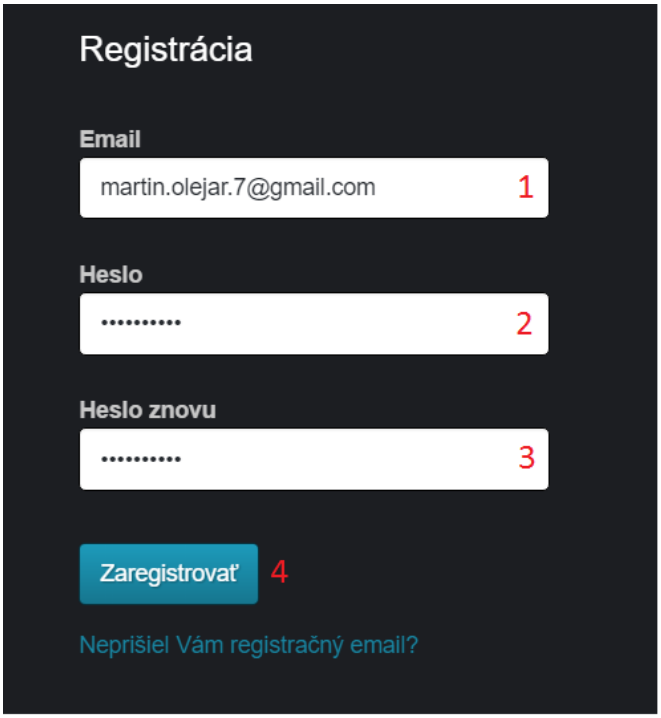
Vedúci projektu: Ing. Karol Rástočný, PhD.
Názov tímu: CoolStoryBro
Členovia tímu: Bc. Jakub Ondik
Bc. Patrik Januška
Bc. Adam Neupauer
Bc. Martin Olejár
Bc. Miroslav Hurajt
Bc. Ondrej Hamara
Kontakt: storyteller@googlegroups.com
Akademický rok: 2016/2017
Dátum odovzdania: 16. 11. 2016

Obsah

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).



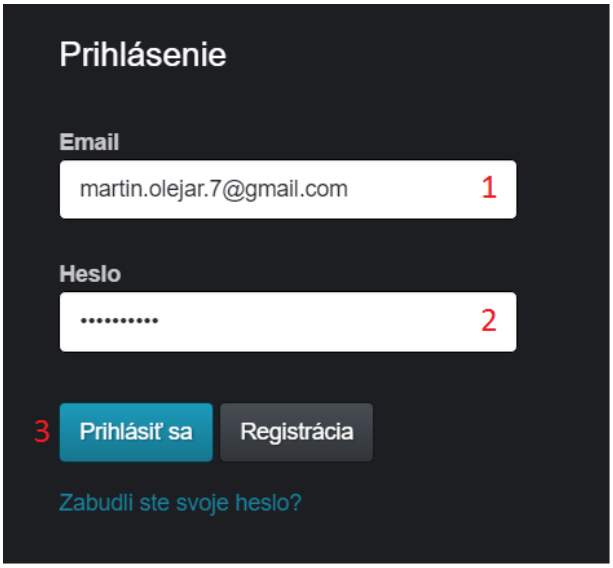
Obrázok 1: Registračný formulár

Po zadaní platných údajov je poslaný registračný email na zadanú emailovú adresu. Obsah tohto emailu je na obr. 2. V tomto emaile 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

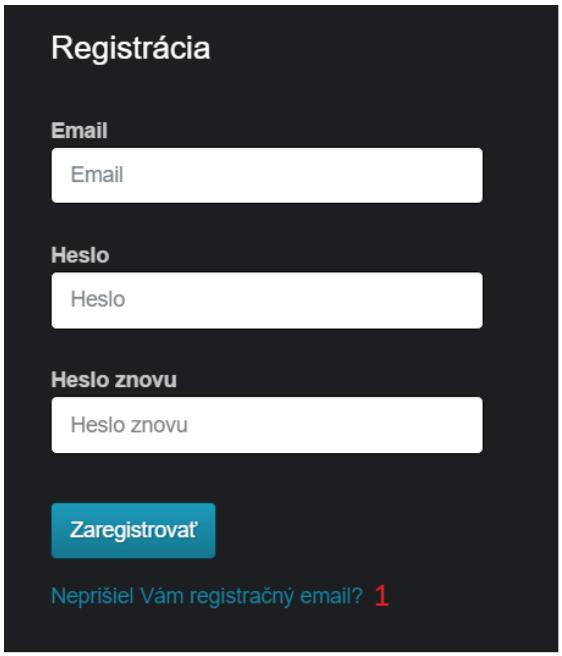
Zaregistrovaný používateľ sa môže prihlásiť do systému prostredníctvom prihlasovacieho formulára. Pre úspešné prihlásenie vyplňte v tomto formulári emailovú adresu (1) a heslo (2) zadané pri registrácii. Potom kliknite na tlačidlo **Prihlásiť sa** (3).



Obrázok 3: Prihlasovací formulár

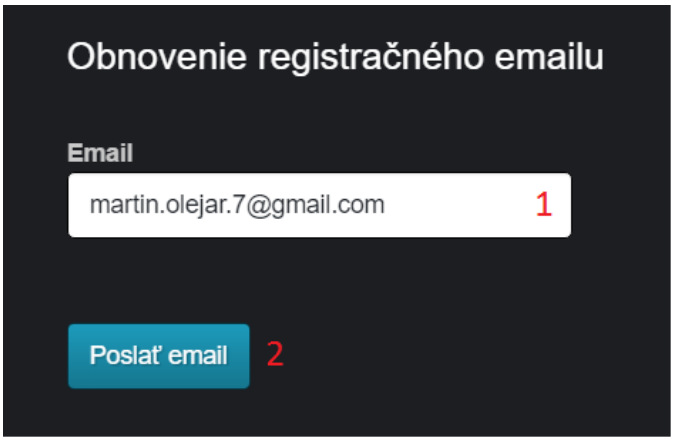
1.3 Obnovenie registračného emailu

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.



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.



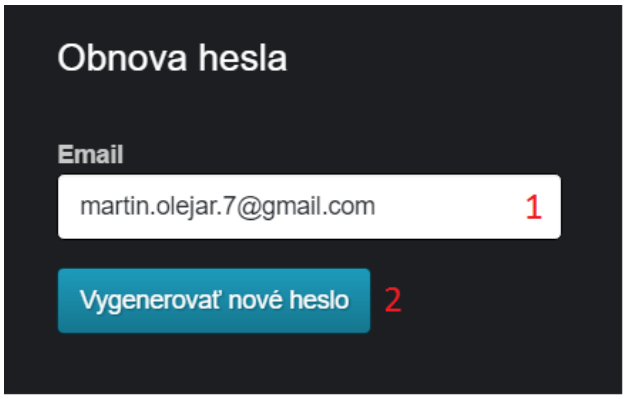
Obrázok 5: Formulár pre obnovenie registračného emailu

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.

Obrázok 6: Prihlasovací formulár

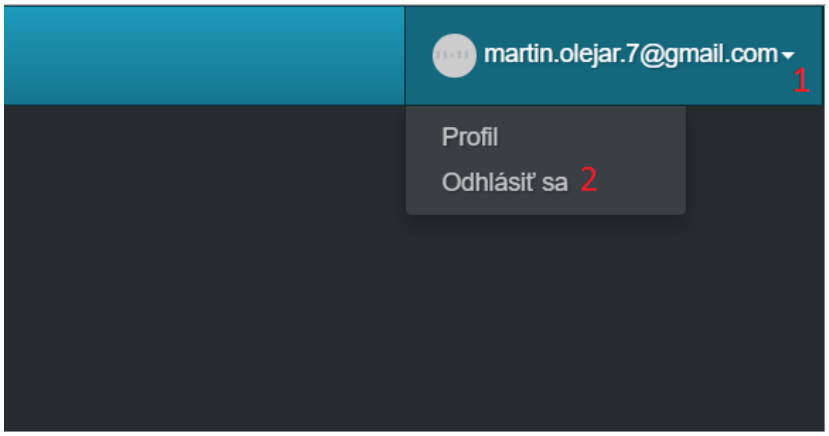
Následne sa zobrazí formulár pre obnovenie hesla na obr. 7, v ktorom vyplňte svoju emailovú adresu (1) potrebnú na prihlásenie. Potom kliknite na tlačidlo **Vygenerovať nové heslo** (2). Na vašu emailovú adresu vám príde email, v ktorom bude vaše nové vygenerované heslo pre vstup do systému.



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).



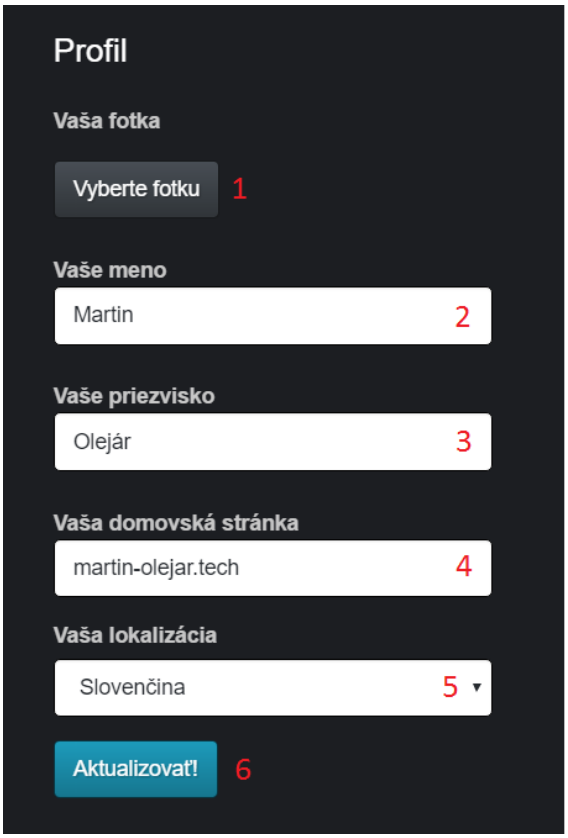
Obrázok 8: Odhlásenie

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).

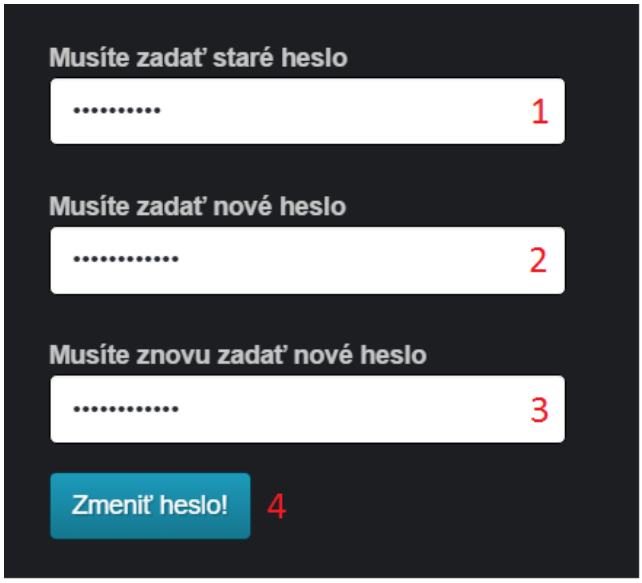
Obrázok 9: Možnosti prihláseného používateľa

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 potvrďte tlačidlom **Zmeniť heslo!** (4).

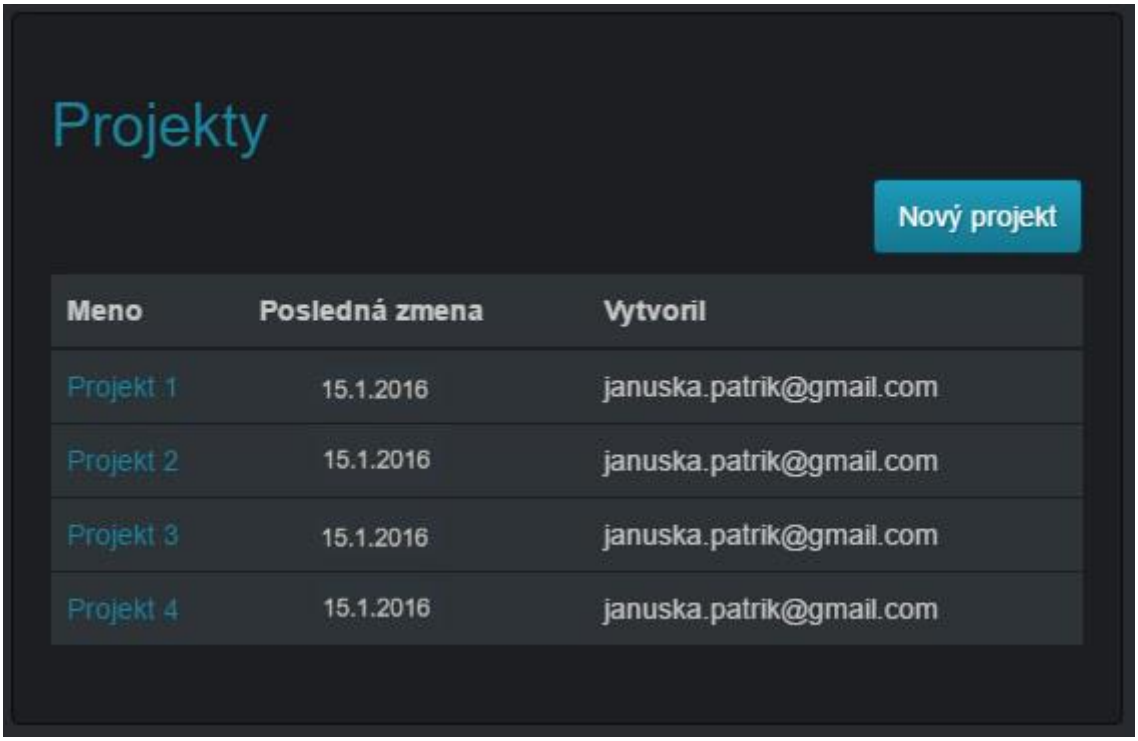


Obrázok 11: Formulár pre zmenu hesla

3 Vytvorenie a správa projektov

3.1 Zoznam projektov

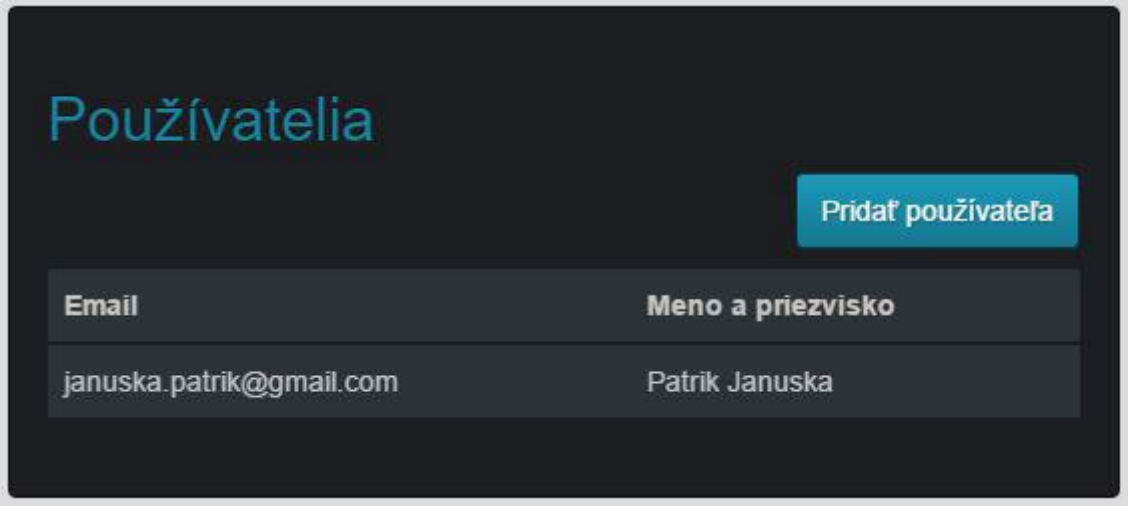
Po prihlásení používateľa je možné prezerať zoznam aktuálnych projektov v časti Projekty ako môžeme vidieť na obrázku 12. Ku každému projektu je uvedené jeho meno, dátum poslednej zmeny a používateľ, ktorý ho vytvoril.



Obrázok 12: Zoznam aktuálnych projektov

3.2 Zoznam používateľov v projekte

Po kliknutí na niektorý z projektov sa dostávame do hlavného menu vybratého projektu. V tomto menu sa po kliknutí na kartu "Správa používateľov" zobrazí zoznam používateľov priradených k danému projektu, ako je vidieť na obrázku 13. Tento zoznam obsahuje email používateľa a tiež jeho meno a priezvisko.

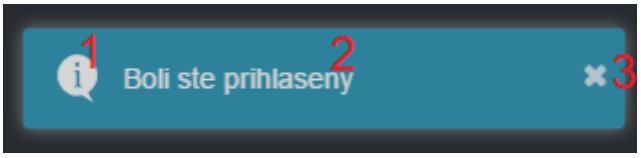


Obrázok 13: Zoznam používateľov v projekte

4 Notifikácie

4.1 Zobrazenie notifikácie – toast popup

Po prijatí notifikácie je používateľovi zobrazený toast popup v pravom dolnom rohu obrazovky (obr. 13).



Obrázok 13: Toast popup notifikácia

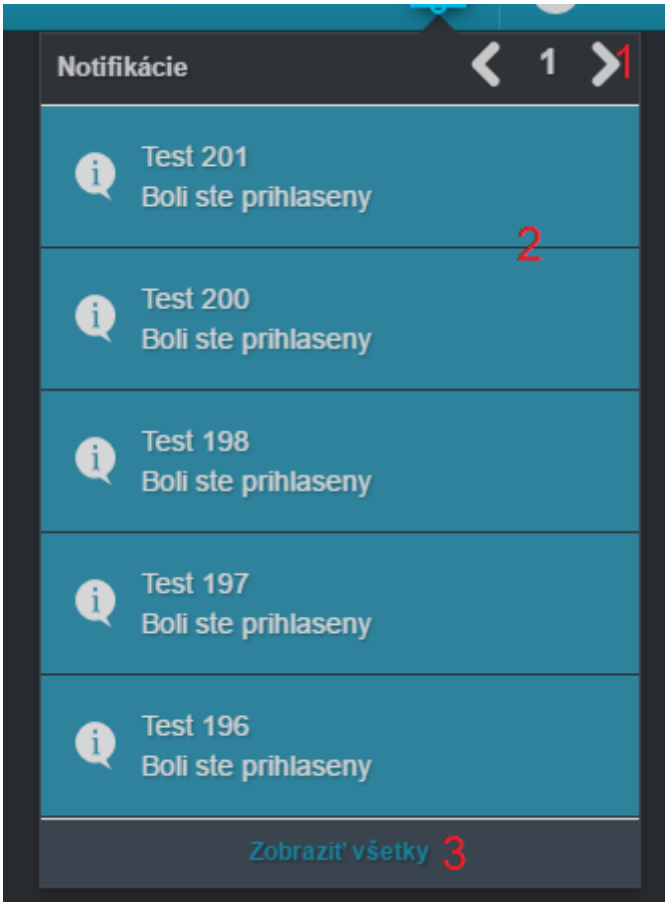
Typ notifikácie je znázornený informačnou ikonkou (1) a farbou pozadia. Samotný text notifikácie je znázornený v strednej časti notifikácie (2). Notifikáciu je možné zavrieť kliknutím na tlačidlo zatvorenia (3) alebo kliknutím na samotnú notifikáciu.

4.2 Zobrazenie notifikácie – navigačný panel

K notifikáciám je možné pristupovať pomocou tlačidla notifikácií v navigačnom panely, ktoré môžeme vidieť na obrázku 14. Súčasťou tohto tlačidla je indikátor počtu nevidených notifikácií (1).

Obrázok 144: Indikátor notifikácií

Po kliknutí na toto tlačidlo sa zobrazia samotné notifikácie (obr. 15). 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 155: Indikátor notifikácií