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
Kontakt:	storyteller-04@googlegroups.com
Akademický rok:	2016/2017

Obsah

1 Úvod

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

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

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

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

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

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

V piatej kapitole dokumentu je uvedená globálna retrospektíva pre zimný a letný semester. Táto časť zahŕňa postup pri riešení projektu, zoznam činností, ktoré boli vyhodnotené za pozitívne, zoznam problémov, ktoré sa vyskytli počas projektu, spolu s ich riešením a aktuálny stav riešenia projektu.

2 Roly členov tímu a podiel práce

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

2.1 Roly členov tímu

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

Tabuľka 1: Roly členov tímu

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

* člen tímu mal pridelenú rolu do 19. 4. 2017, keďže bol v ten deň vyhodený

2.2 Pridelené zodpovednosti členov tímu

Bc. Jakub Ondik

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

Bc. Patrik Januška

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

Bc. Adam Neupauer

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

Bc. Martin Olejár

- aktualizácia dokumentov na webovej stránke

- vytvorenie šablóny pre písanie dokumentácie a jej údržba
- zodpovednosť za zadeľovanie úloh a sumarizovanie šprintov
- sledovanie termínov
- dohľad nad plnením zadaných úloh

Bc. Miroslav Hurajt

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

Bc. Ondrej Hamara

- zaznamenávanie a vyhodnotenie rizík počas práce na tímovom projekte
- poznámka: člen tímu zastával tieto zodpovednosti do 19. 4. 2017

2.3 Podiel práce

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

Tabuľka 2: Podiel práce na dokumentácii riadenia projektu

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

Tabuľka 3: Podiel práce na dokumentácii inžinierskeho diela

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

3 Aplikácie manažmentov

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

3.1 Manažment dokumentácie

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

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

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

3.2 Manažment komunikácie

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

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

3.3 Manažment plánovania

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

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

3.4 Manažment verziovania

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

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

3.5 Manažment testovania

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

3.6 Manažment kvality písania zdrojového kódu

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

4 Sumarizácie šprintov

4.1 Šprint 1 – Baldr

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

Title	Story Points	State	Assigned To
Registracia prostrednictvom Emailu z webovej 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

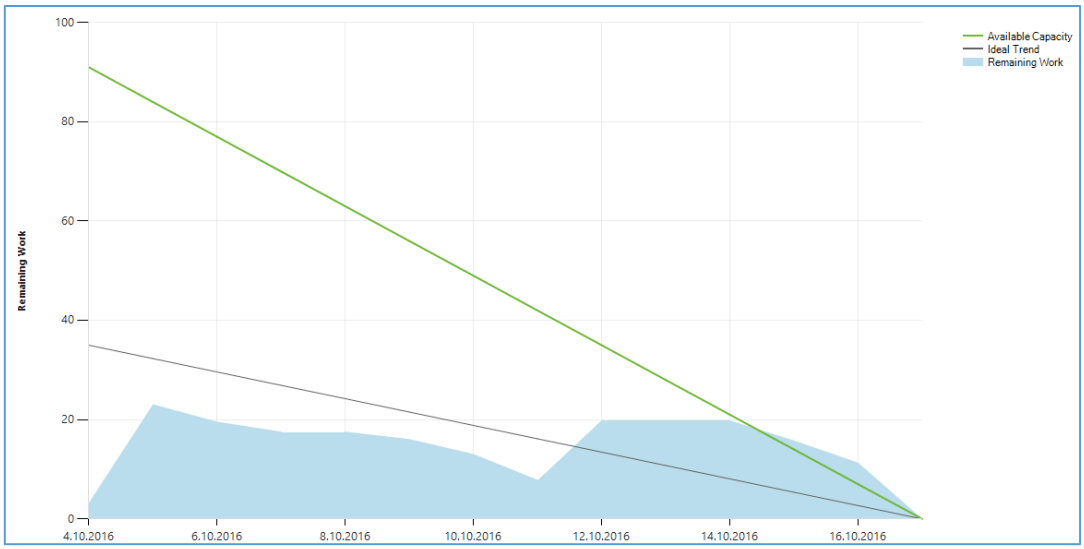
Obrázok 1: Úlohy šprintu Baldr

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

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



Obrázok 3: Burndown chart pre šprint Baldr

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

4.2 Šprint 2 – Fenrir

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

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

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

Obrázok 6: Burndown chart pre šprint Fenrir

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

4.3 Šprint 3 – Freya

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

Výsledkom tohto šprintu bola základná správa projektov – prehľad, zobrazenie, pridávanie používateľov a 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.

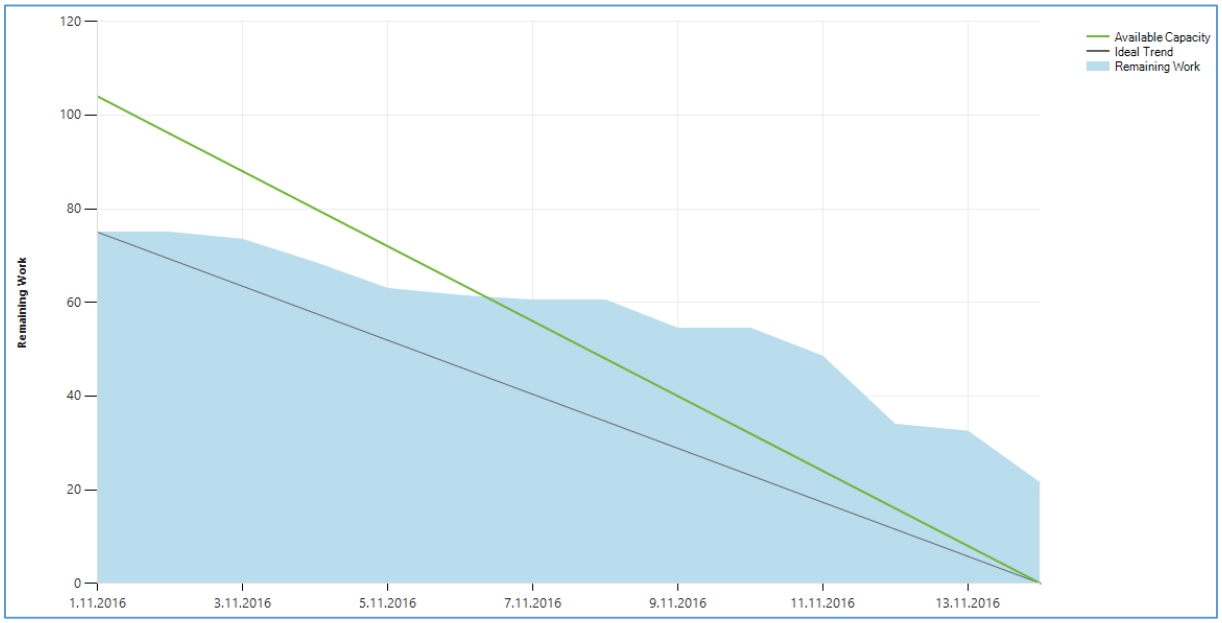
Veľkým negatívom tohto šprintu je nedokončenie pomerne veľkého počtu používateľských príbehov a úloh, čo bolo spôsobené nárazovou prácou tesne pred ukončením šprintu. Taktiež sa nepodarilo dokončiť ani jednu prenesenú úlohu z minulého šprintu, za ktoré bol zodpovedný jednotlivec, ktorý nedokončil (nezačal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

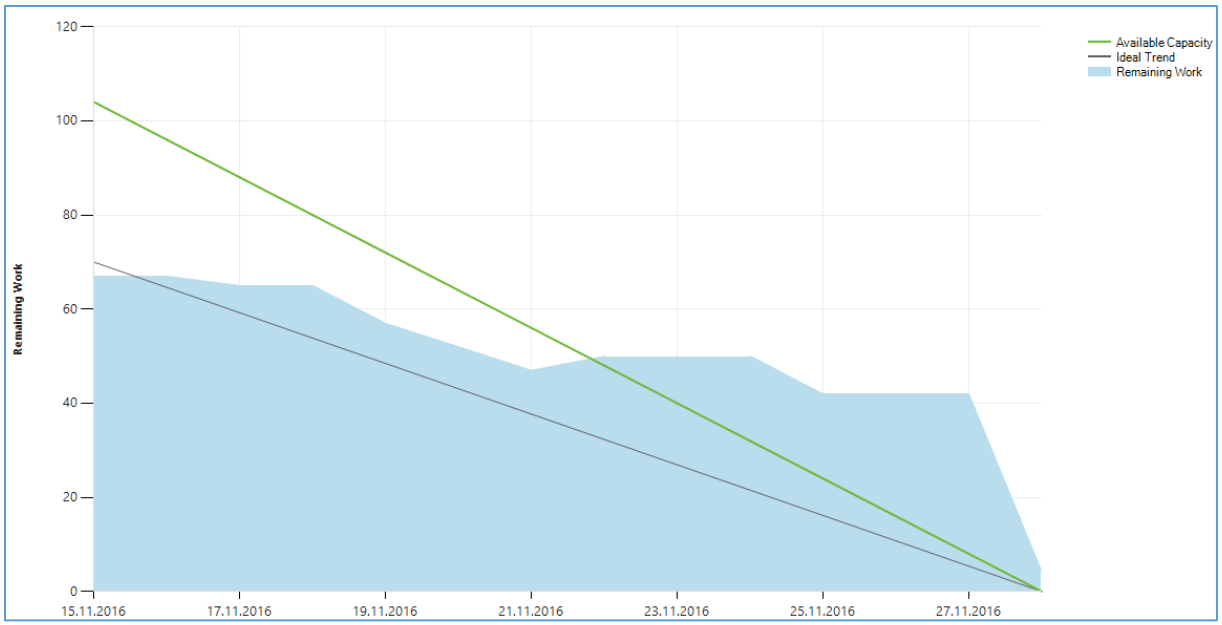
Velocity tohto šprintu sme sa rozhodli zvýšiť na 37 aj z dôvodu prenášaných príbehov. Dosiahnutá velocity bola na úrovni 26, čiže na podobnej úrovni ako predchádzajúce šprinty. Problém znovu nastal na úrovni jednotlivca, ktorému sa nepodarilo dokončiť ani úlohy prenášané z predchádzajúcich šprintov.

4.4 Šprint 4 – Heimdallr

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

Title	Story Points	State	Assigned To
Notifikovanie pouzivatela o diani v StoryTelleri	8	Active	Jakub Ondik
Templaty pre notifikacie		Active	Bc. Miroslav Hurajt
Pozvanie do projektu cez StoryTeller	3	Active	Bc. Miroslav Hurajt
Template na notifikaciu pozvania pouzivatela do projektu		Active	Bc. Miroslav Hurajt
REST endpoint pre príjem priradenia pouzivatela k projektu + sp...		Closed	Bc. Martin Olejar
Definovanie roly zakaznika	3	Active	Bc. Ondrej Hamara
Vytvorenie authorization metod a nastavenie anotacii pre rolu z...		Closed	Bc. Ondrej Hamara
Vytvorenie angularJS modulu specifickeho pre zakaznika		Closed	Bc. Ondrej Hamara
Definovanie roly analytika	3	Active	Bc. Ondrej Hamara
Vytvorenie authorization metod a nastavenie anotacii pre rolu a...		Closed	Bc. Ondrej Hamara
Vytvorenie angularJS modulu specifickeho pre analytika		Closed	Bc. Ondrej Hamara
Pouzivatel ma moznost drag'n'drop tvarov na platno	13	Closed	Bc. Adam Neupauer
Vytvorenie canvasu		Closed	Bc. Adam Neupauer
Vytvorenie panelu s nastrojmi na kreslenie		Closed	Bc. Adam Neupauer
Funkcia drag n drop/pridanie na zaklade kliku		Closed	Bc. Adam Neupauer
Vytvorenie geom. objektov : textove pole		Closed	Jakub Ondik
Vytvorenie wrappera pre RavenJS	3	Active	Jakub Ondik
Wrapper metody pre vsetky typy logovacich sprav, vratane vyni...		Closed	Jakub Ondik
Perzistencia skice	8	Active	Jakub Ondik
Navrh sposobu perzistencie skice		Active	Jakub Ondik
Implementacia perzistencie skice		Active	Jakub Ondik
Registracia resource pre skicu v AngularJS module		New	Bc. Martin Olejar
CRUD REST endpointy		New	Bc. Martin Olejar

Obrázok 10: Úlohy šprintu Heimdallr



Obrázok 11: Burndown chart pre šprint Heimdallr

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

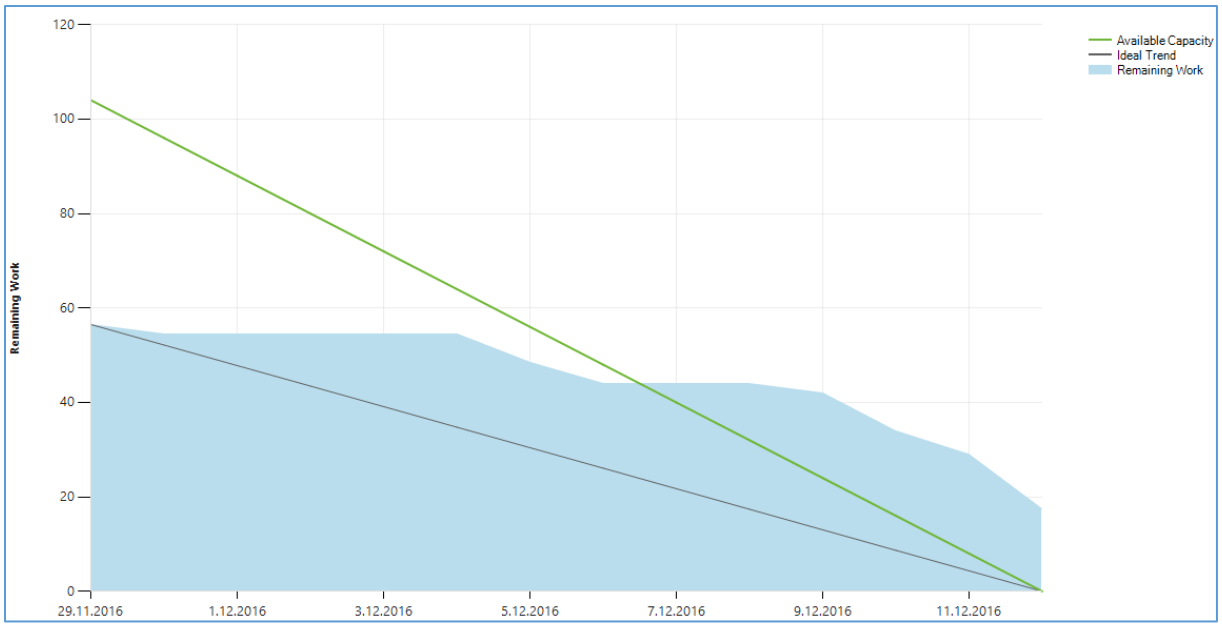
4.5 Šprint 5 – Huginn

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

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

Title	Story Points	State	Assigned To
Notifikovanie pouzivatela o diani v StoryTelleri	8	Active	Jakub Ondik
Templaty pre notifikacie		Active	Bc. Miroslav Hurajt
Pozvanie do projektu cez StoryTeller	3	Active	Bc. Miroslav Hurajt
Template na notifikaciu pozvania pouzivatela do projektu		Active	Bc. Miroslav Hurajt
Pridanie vlastnosti pre komponent v skici	5	Resolved	Bc. Adam Neupauer
Zakladne vlastnosti prvkov		Closed	Bc. Adam Neupauer
Pridanie CSS vlastnosti		Closed	Jakub Ondik
Popup okno		Closed	Bc. Adam Neupauer
Perzistencia skice	8	Resolved	Jakub Ondik
Navrh sposobu perzistencie skice		Closed	Jakub Ondik
Implementacia perzistencie skice		Closed	Jakub Ondik
Registracia resource pre skicu v AngularJS module		Closed	Bc. Martin Olejar
CRUD REST endpointy		Closed	Bc. Martin Olejar
Vytvorenie wrappera pre RavenJS	3	Resolved	Jakub Ondik
Zaregistrovanie watchera pre prihlaseneho usera, ktory nastavi u...		Closed	Bc. Patrik Januska
Pridanie logovania pre jednotlive AngularJS moduly		Closed	Bc. Martin Olejar
Zobrazenie zoznamu skic projektu	5	Active	Bc. Patrik Januska
REST endpoint na ziskanie skic pre projekt		Closed	Bc. Patrik Januska
Vytvorenie AngularJS metody na ziskanie zoznamu skic pre kon...		Closed	Bc. Patrik Januska
Vytvorenie dlazdicoveho panelu so skicami, ktory reaguje na tou...		New	Bc. Miroslav Hurajt
Zobrazenie skice	3	New	Bc. Ondrej Hamara
Zlozenie skice z prijatych objektov		Active	Bc. Ondrej Hamara
Moznost pridat základné komponenty do skice	5	New	Bc. Miroslav Hurajt
Pridanie buttonu		New	Bc. Miroslav Hurajt
Pridanie radiobuttonu		Closed	Bc. Ondrej Hamara
Pridanie labelu		Closed	Bc. Martin Olejar
Pridanie containeru		New	Bc. Patrik Januska
Pridanie checkboxu		New	Bc. Miroslav Hurajt
Pridanie obrazku		New	Bc. Adam Neupauer
Pridanie komponentu pre volny text		Closed	Bc. Ondrej Hamara
Pridanie comboboxu		Closed	Bc. Ondrej Hamara

Obrázok 12: Úlohy šprintu Huginn



Obrázok 13: Burndown chart pre šprint Huginn

4.6 Šprint 6 – Loki

V prvom šprinte letného semestra sme sa snažili dokončiť nedokončené úlohy zo zimného semestra ako pridávanie základných komponentov do skice alebo pozvanie používateľa do projektu. Rozhodli sme sa implementovať pridanie scenára pre skicu. Identifikovali sme viacero implementačných chýb a nedostatkov v našej aplikácii, ktoré boli zaradené do šprintu ako bugy alebo klasické úlohy. Všetky úlohy je možné vidieť na obrázkoch 14 a 15.

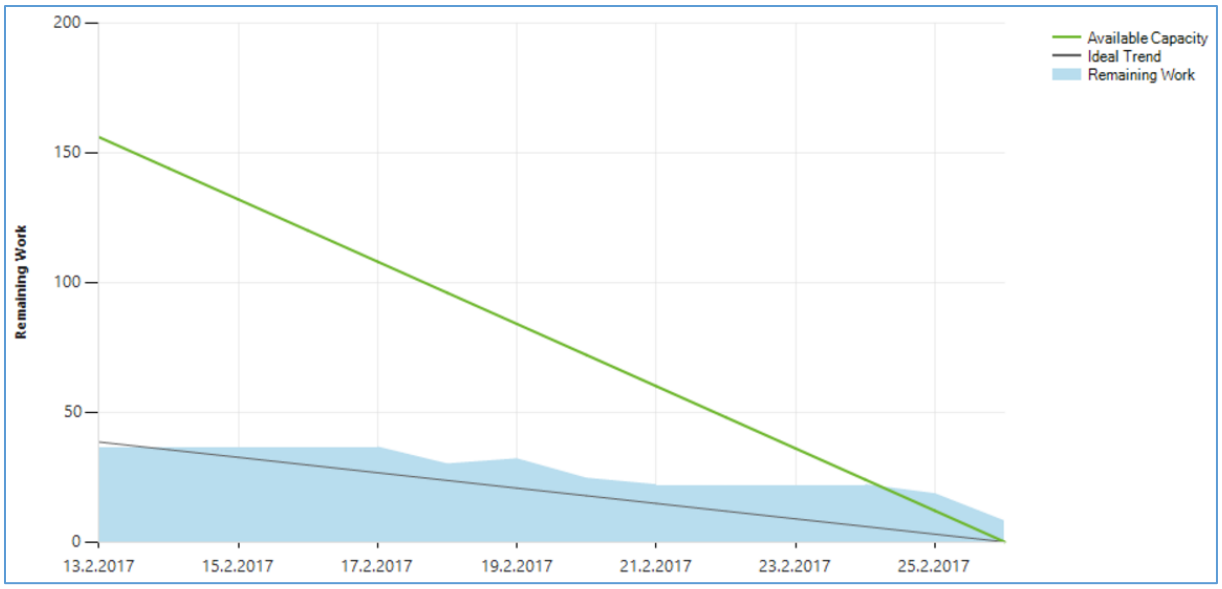
Velocity tohto šprintu bola 36 aj z dôvodu prenášania úloh z predchádzajúcich šprintov. Prenášané úlohy sa nepodarilo dokončiť, nové úlohy v tomto šprinte mali spolu 20 bodov a boli úspešne dokončené. Dosiahnutá velocity tohto šprintu je teda 20. Postup práce na tomto šprinte je zobrazený na obrázku 16.

Title	Story Points	State	Assigned To
Moznost pridat základné komponenty do skice	5	Active	Bc. Adam Neupauer
Pridanie buttonu		Closed	Bc. Miroslav Hurajt
Pridanie radiobuttonu		Active	Bc. Adam Neupauer
Pridanie checkboxu		Active	Bc. Adam Neupauer
Pridanie obrazku		Closed	Bc. Adam Neupauer
Pridanie komponentu pre volny text		Closed	Bc. Ondrej Hamara
Pridanie comboboxu		Active	Bc. Adam Neupauer
Uprava uz existujucej skice	3	Closed	Jakub Ondik
Vytvorenie AngularJS metody na ziskanie skice z BE appky		Closed	Bc. Ondrej Hamara
Namiesto loga je v dashboarde zobrazený len čierny štvorec	1	Closed	Karol Rástočný
Upravit binding na field loga pre projekt		Closed	Bc. Martin Olejar
Kopírovanie emailu	1	Closed	Bc. Patrik Januska
Skopírovanie emailu z prihlásenia do registrácie		Closed	Bc. Patrik Januska
Skopírovanie emailu z prihlásenia do zadudol som heslo		Closed	Bc. Patrik Januska
Skopírovanie emailu z registrácie do opätovne zaslať aktivačný email		Closed	Bc. Patrik Januska
Informácia o odoslaní aktivačného emailu		Closed	Bc. Patrik Januska
Informovanie o dlzke hesla	1	Closed	Bc. Patrik Januska
Pridat hlasku o nespravnom formate hesla		Closed	Bc. Patrik Januska
Jednotne zobrazovať meno usera a jeho avatara	1	Closed	Bc. Adam Neupauer
Zmenit binding na field usera		Closed	Bc. Adam Neupauer

Obrázok 14: Úlohy šprintu Loki

Pridanie nového scenáru pre jednu skicu	8	Closed	Jakub Ondik
Panel s krokmi scenaru		Closed	Jakub Ondik
Pre každý prvok je definovaný template textu kroku scenáru (DSL)		Closed	Jakub Ondik
Pri pridaní prvku je automaticky pridaný jeho krok do scenáru		Closed	Jakub Ondik
Pri zmazaní prvku je zmazaný krok scenáru		Closed	Bc. Martin Olejar
Pri aktualizovaní prvku je aktualizovaný aj krok scenáru (meno, constraint)		Closed	Bc. Martin Olejar
Automaticke spustanie testov		Closed	Jakub Ondik
Zakladny panel pre spustanie akceptacnych testov		Closed	Jakub Ondik
Nahravanie videa z testu		Closed	Jakub Ondik
Okno notifikácií má byť modálne	2	Closed	Bc. Adam Neupauer
Zmenit template pre popup na modal		Closed	Bc. Adam Neupauer
Informácia o korektnom a nekorektnom vyplnení údajpv pri tlačidle	2	Closed	Bc. Adam Neupauer
Presunut hlasky z top formulara k buttonu		Closed	Bc. Adam Neupauer
Logo ma presmeruje na domovskú stránku	1	Closed	Bc. Patrik Januska
Nastavit href podla pripadov		Closed	Bc. Patrik Januska
Notifikovanie pouzivatela o diani v StoryTelleri	8	Active	Jakub Ondik
Templaty pre notifikacie		Closed	Bc. Miroslav Hurajt
Pozvanie do projektu cez StoryTeller	3	Active	Bc. Miroslav Hurajt
Template na notifikaciu pozvania pouzivatela do projektu		Closed	Bc. Miroslav Hurajt

Obrázok 15: Úlohy šprintu Loki - pokračovanie



Obrázok 16: Burndown chart pre šprint Loki

4.7 Šprint 7 – Muninn

V šprinte Muninn sme okrem riešenia ďalších nájdených chýb v aplikácii a vylepšení funkcionality pracovali na generovaní HTML layoutu skice pre programátora. Pustili sme sa do modulárneho testovania, ktoré zahŕňalo spúšťanie testov v docker kontajneri. Ďalej sme navrhli a implementovali používateľské rozhranie pre akceptačné testy, pre ktoré sme vytvorili podporu viacerých rozlíšení. Súčasťou šprintu bol len 1 používateľský príbeh z predchádzajúceho šprintu (Okno notifikácií má byť modálne). Všetky úlohy je možné vidieť na obrázkoch 17 a 18.

Velocity šprintu Muninn bola 24, čo sa nám podarilo splniť. Postup práce je zobrazený na obrázku 19.

Title	Story Points	State	Assigned To
Generovanie pouzitelneho layoutu skice	3	Closed	Bc. Adam Neupauer
Uprava layoutu na pouzitie pre programatora		Closed	Bc. Adam Neupauer
Funkcia na stiahnutie html suboru s upravenym layoutom		Closed	Bc. Adam Neupauer
Okno notifikácií má byť modálne	2	Closed	Bc. Adam Neupauer
Code review		Closed	Jakub Ondik
Pouzivatel nie je informovany o ulozeni skice	1	Closed	Bc. Martin Olejar
Po ulozeni skice sa zobrazi informacia o uspechu		Closed	Bc. Patrik Januska
Redirect pri uspesnom ulozeni		Closed	Bc. Martin Olejar
Code review		Closed	Jakub Ondik
Kopírovanie emailu 2	1	Closed	Bc. Patrik Januska
Skopírovanie emailu z registrácie do prihlásenia		Closed	Bc. Patrik Januska
Skopírovanie emailu z obnovenia reg. emailu do prihlásenia		Closed	Bc. Patrik Januska
Skopírovanie emailu z obnovy hesla na prihlásenie		Closed	Bc. Patrik Januska
Vypísanie hlášky o odoslaní emailu pri obnove hesla		Closed	Bc. Martin Olejar
Code review		Closed	Bc. Miroslav Hurajt
Modularne testovanie	8	Closed	Jakub Ondik
Docker image		Closed	Jakub Ondik
Spustanie testov v docker kontajneri		Closed	Jakub Ondik
Pridat znovuodoslanie emailu na obrazovku s prihlasenim	1	Closed	Jakub Ondik
Prida link na obrazovku znovuodoslania		Closed	Bc. Patrik Januska
Code review		Closed	Bc. Martin Olejar

Obrázok 17: Úlohy šprintu Muninn

Hlasku o zaslani emailu po registracii presunut zo samostanej obrazovky na prihlasovaciu obrazovku	1	Closed	Bc. Miroslav Hurajt
Presun hlasky		Closed	Bc. Patrik Januska
Code review		Closed	Bc. Martin Olejar
V prehľade projektov zobraziť logo	1	Closed	Bc. Martin Olejar
Nastavenie zobrazenia loga pri projekte		Closed	Bc. Martin Olejar
Code review		Closed	Bc. Ondrej Hamara
User interface pre akceptacne testy	3	Closed	Bc. Adam Neupauer
Navrh rozhrania pre akceptacne testy		Closed	Bc. Adam Neupauer
Implemntacia navrhnuteho rozhrania		Closed	Bc. Adam Neupauer
Code review		Closed	Jakub Ondik
Podpora viacerych rozliseni pre akceptacne testy	3	Closed	Bc. Miroslav Hurajt
Uprava TestResult entity		Closed	Bc. Martin Olejar
Uprava REST endpointu		Closed	Bc. Patrik Januska
Uprava skriptov		Closed	Jakub Ondik
Vytvorenie viacerych verzii wallpapperu		Closed	Jakub Ondik
Code review		Closed	Bc. Adam Neupauer
Uprava layoutu pre akceptacne testy		Closed	Bc. Ondrej Hamara
Oprava retardovanych chyb a konfliktov		Closed	Jakub Ondik

Obrázok 18: Úlohy šprintu Muninn - pokračovanie

Obrázok 19: Burndown chart pre šprint Muninn

4.8 Šprint 8 – Odin

V šprinte Odin sme taktiež riešili identifikované chyby a vylepšovali sme naše používateľské rozhrania. Pokračovali sme v implementácii modulárneho testovania a rozhodli sme vytvoriť queue pre akceptačné testy pre lepšiu organizáciu a zobrazenie času štartu testu. Pre lepšiu dokumentáciu nášho kódu sme sa rozhodli prepísať našu API dokumentácie cez Swagger. Z neočakávaných dôvodov sme museli opätovne nastaviť náš server a nainštalovať potrebné časti. Predmetom tohto šprintu bolo aj vytvorenie testovacieho prostredia pre Android aplikácie a generovanie layoutu pre native Android. Ďalšou povinnosťou bola úprava a odovzdanie verzie Camera ready článku na IIT.SRC. Všetky úlohy je možné vidieť na obrázkoch 20 a 21.

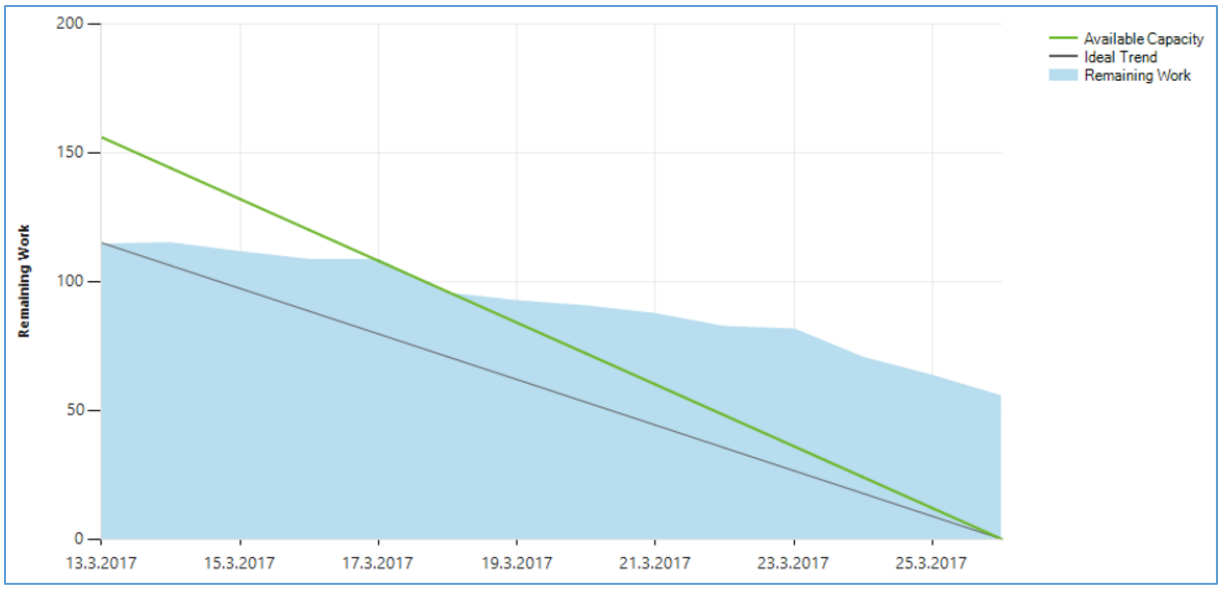
Šprint Odin mal velocity o hodnote 40, pričom dosiahnutá velocity bola 19. Táto malá hodnota velocity bola spôsobená problémami s implementovaním úloh spojených s platformou Android. Postup práce je zobrazený na obrázku 22.

Title	Story Points	State	Assigned To
Zobrazenie zoznamu skic projektu	5	Active	Bc. Miroslav Hurajt
Odstranenie duplicitnej hlasky pri zabudnuti hesla	1	Closed	Bc. Patrik Januska
Odstranenie hlasky o reg. emaili		Closed	Bc. Patrik Januska
Code review		Closed	Bc. Martin Olejar
Modularne testovanie	8	Closed	Jakub Ondik
Vytvorenie API pre docker		Closed	Jakub Ondik
Vytvorenie callback endpointu pre report z vnutra containera		Closed	Jakub Ondik
Code review		Closed	Bc. Adam Neupauer
Emaily chodia iba v slovencine	2	Closed	Bc. Patrik Januska
Nastavit lokalizaciu emailu		Closed	Bc. Patrik Januska
Code review		Closed	Jakub Ondik
IIT.SRC TP CUP Camera ready do 21. 3. 2017	1	Closed	Bc. Martin Olejar
Úprava článku podľa poznámok		Closed	Bc. Martin Olejar
Uprava clanku		Closed	Jakub Ondik
Oprava layoutu pre projekt	2	Closed	Bc. Adam Neupauer
Uprava projekt home, project settings a layout noveho projektu		Closed	Bc. Adam Neupauer
Code review		Closed	Jakub Ondik
Server rebuild	5	Closed	Bc. Adam Neupauer
Instalacia chybajucich casti		Closed	Bc. Adam Neupauer
Instalacia chybajucich casti		Closed	Jakub Ondik
V dialógu nastavenia prvku sú uvedené default hodnoty	1	Closed	Bc. Patrik Januska
Pridanie placeholderu pre kazdy typ atributu		Closed	Bc. Patrik Januska
Code review		Closed	Jakub Ondik

Obrázok 20: Úlohy šprintu Odin

Informovať o chýbajúcom poli pri vytvoreni projektu	1	Closed	Bc. Adam Neupauer
Pridanie hlášky		Closed	Bc. Adam Neupauer
Code review		Closed	Bc. Adam Neupauer
Vytvorenie queue pre testy	5	Closed	Bc. Adam Neupauer
Doplnenie queue mechanizmu do python master node		Closed	Bc. Adam Neupauer
Zobrazenie doby cakania na obrazovke pre testy		Closed	Bc. Adam Neupauer
Testovacie Android prostredie	5	Active	Bc. Martin Olejar
Vytvorit testovacie prostredie pre Android aplikacie		Active	Bc. Martin Olejar
Prepisanie API dokumentacie cez Swagger	1	Closed	Bc. Miroslav Hurajt
Zdokumentovanie API (controllerov) cez Swagger plugin		Closed	Bc. Miroslav Hurajt
Generovanie layoutu pre native Android	3	Active	Jakub Ondik
Vytvorit metody na generovanie prvku podla drawable		Active	Bc. Ondrej Hamara
Zlozenie layoutu z prvkov		Active	Bc. Ondrej Hamara
REST endpoint pre stiahnutie Android layoutu		Closed	Jakub Ondik
Pridat moznost stiahnutia Android layoutu		Closed	Jakub Ondik

Obrázok 21: Úlohy šprintu Odin - pokračovanie



Obrázok 22: Burndown chart pre šprint Odin

4.9 Šprint 9 – Surtr

V šprinte Surtr sme sa venovali hlavne možnosti vytvorenia viacerých scenárov pre skicu. Na stretnutí sme si tento problém rozanalyzovali a následne ho implementovali. Pre podporu testovacieho prostredia pre Android aplikácie sme pokračovali definovaním Android zariadení a vytvorením izolovaného Android prostredia. Medzi dôležité úlohy patrilo aj pozvanie používateľa do projektu prostredníctvom emailu. Predmetom šprintu boli až 4 nájdené bugy a ďalšie vylepšenia. Všetky úlohy je možné vidieť na obrázkoch 23 a 24.

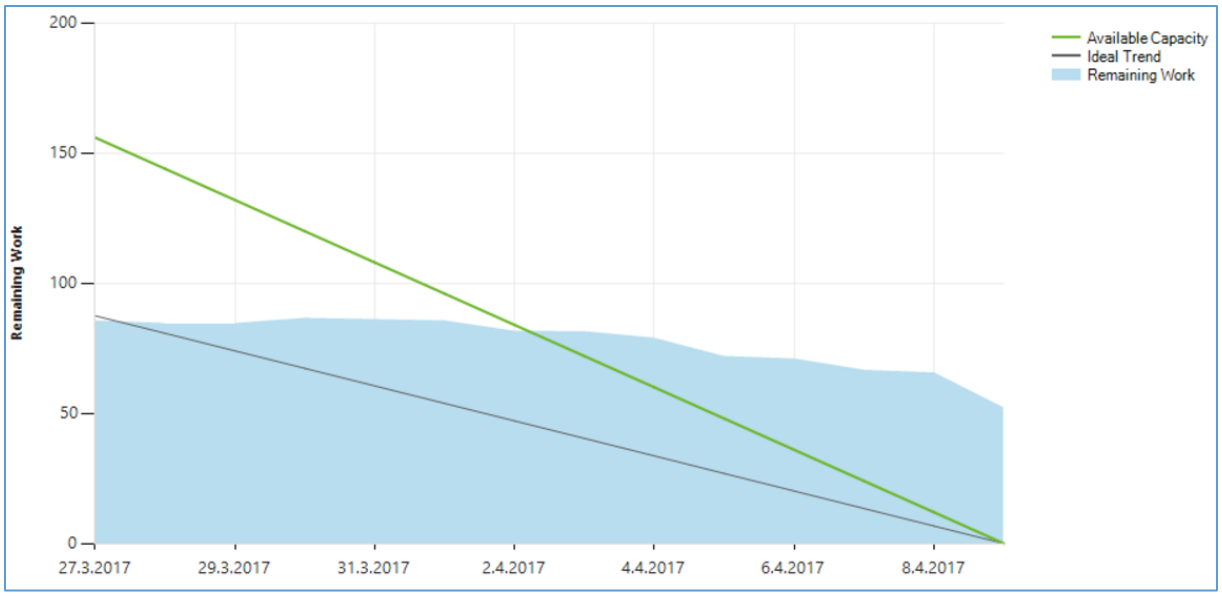
Velocity tohto šprintu bola 32 aj z dôvodu prenášania úloh z predchádzajúcich šprintov. Podarilo sa nám dosiahnuť velocity o hodnote 27, čo je o trochu viac v porovnaní s predchádzajúcimi šprintami. Postup práce je zobrazený na obrázku 25.

Title	Story Points	State	Assigned To
Dialog nastavenia prvku skice je možné presúvať	2	Closed	Bc. Martin Olejar
Pridat moznost presuvania okna		Closed	Bc. Martin Olejar
Code review		Closed	Jakub Ondik
Vytvorenie queue pre testy	5	Closed	Bc. Adam Neupauer
Code review		Closed	Jakub Ondik
Novy vysledok testu nie je pridany do historie	2	Closed	Bc. Adam Neupauer
Pridanie testu do historie po spusteni (po sprave zo servera)		Closed	Bc. Adam Neupauer
Code review		Closed	Jakub Ondik
Po refreshi sa resetne lokalizacia na SK	1	Closed	Bc. Patrik Januska
Nastavenie lokalizacie podla usera		Closed	Bc. Patrik Januska
Code review		Closed	Jakub Ondik
Po prihlaseni zostanu projekty predchadzajuceho pouzivatela	1	Closed	Bc. Patrik Januska
Pridat refresh stavu po prihlaseni		Closed	Bc. Patrik Januska
Pozvanie do projektu prostrednictvom emailu	3	Closed	Bc. Patrik Januska
Vytvorenie tlacidla na pozvanie prostrednictvom emailu		Closed	Bc. Martin Olejar
Odoslanie mailu		Closed	Bc. Patrik Januska
Vytvorit template pre email		Closed	Bc. Miroslav Hurajt
Vytvorenie obrazovky pre potvrdenie		Closed	Bc. Patrik Januska
Po kliknutí na aktivačný email sa zobrazí krajšia stránka	1	Closed	Bc. Martin Olejar
Úprava stránky		Closed	Bc. Martin Olejar
Code review		Closed	Jakub Ondik

Obrázok 23: Úlohy šprintu Surtr

Nefunguje nastavit vlastnosti pre label, placeholdre nemaju skutocne hodnoty	2	Closed	Bc. Patrik Januska
Nastavit realne hodnoty pre placeholdre		Closed	Bc. Patrik Januska
Nastavit binding na CSS vlastnosti pre label		Closed	Bc. Miroslav Hurajt
Izolovane prostredie a definovanie zariadeni	5	Closed	Jakub Ondik
Definovat Android zariadenia		Closed	Jakub Ondik
Vytvorit izolovane Android prostredie		Closed	Bc. Adam Neupauer
Moznost vytvorenia viacerych scenarov pre skicu	8	Closed	Jakub Ondik
Uprava layoutu pre podporu viacerych scenarov		Closed	Jakub Ondik
Pridat inicialny krok		Closed	Jakub Ondik
Pridat stav skice pre kazdy krok		Closed	Bc. Adam Neupauer
Pridat moznost duplikacie scenaru		Closed	Jakub Ondik
Moznost prepnutia medzi scenarmi		Closed	Jakub Ondik
Uprava REST endpointu a servisu		Closed	Bc. Adam Neupauer
Vytvorenie tlacidla na zdielanie skice (url)	2	Closed	Jakub Ondik

Obrázok 24: Úlohy šprintu Surtr - pokračovanie



Obrázok 25: Burndown chart pre šprint Surtr

4.10 Šprint 10 – Thor

Súčasťou šprintu Thor boli všetky nedokončené úlohy z predchádzajúcich šprintov. Našou prioritou bolo dokončiť tieto úlohy, keďže sa blížila konferencia IIT.SRC. Do tohto šprintu sme zaradili len pár nových nájdených chýb. Pre IIT.SRC bolo našou povinnosťou vypracovanie dotazníka, vytvorenie prezentácie na robime.it a nakreslenie plagátu. Všetky úlohy je možné vidieť na obrázkoch 26 a 27.

Keďže sme potrebovali dokončiť dlhodobo nedorobené používateľské príbehy a čo najlepšie sa pripraviť na prezentáciu na IIT.SRC, velocity tohto šprintu bola 58, z čoho sa nám podarilo dosiahnuť velocity o hodnote 1. Bolo to aj spôsobené viacerými povinnosťami z iných predmetov, problémami s úplným dokončením úloh a nečinnosťou jednotlivca, ktorý bol v tomto šprinte vyhodený z tímu (Ondrej Hamara). Po tomto vyhodení si nedokončené úlohy prevzali Jakub Ondik a Adam Neupauer a väčšinu z nich dokončili do konferencie. Postup práce je zobrazený na obrázku 28.

Title	Story Points	State	Assigned To
Generovanie pouzitelneho layoutu skice	3	Closed	Bc. Adam Neupauer
Code review		Closed	Jakub Ondik
Zobrazenie zoznamu skic projektu	5	Active	Bc. Miroslav Hurajt
Vytvorenie dlazdicoveho panelu so skicami s horizontálnym scrollbarom		Active	Bc. Miroslav Hurajt
Code review		New	Jakub Ondik
Oprava "neriesitelnych" problemov		Active	Jakub Ondik
Moznost pridat základné komponenty do skice	5	Active	Bc. Adam Neupauer
Code review		New	Jakub Ondik
Skica ostava v cache	1	Closed	Jakub Ondik
Fix cache skice		Closed	Jakub Ondik
Code review		Closed	Bc. Adam Neupauer
Zobrazenie skice	3	Closed	Jakub Ondik
Zlozenie skice z prijatych objektov		Closed	Jakub Ondik
Code review		Closed	Bc. Adam Neupauer
Testovacie Android prostredie	5	Active	Bc. Martin Olejar
Code review		New	Jakub Ondik
Po prihlaseni zostanu projekty predchadzajuceho pouzivatela	1	Closed	Bc. Patrik Januska
Code review		Closed	Jakub Ondik
Pozvanie do projektu prostrednictvom emailu	3	Closed	Bc. Patrik Januska
Code review		Closed	Jakub Ondik
Prepisanie API dokumentacie cez Swagger	1	Closed	Bc. Miroslav Hurajt
Code review		Closed	Jakub Ondik

Obrázok 26: Úlohy šprintu Thor

Nefunguje nastavit vlastnosti pre label, placeholdre nemaju skutocne hodnoty	2	Closed	Bc. Patrik Januska
Code review		Closed	Jakub Ondik
Izolovane prostredie a definovanie zariadeni	5	Closed	Jakub Ondik
Code review		Closed	Jakub Ondik
Code review		Closed	Bc. Martin Olejar
Notifikovanie pouzivatela o diani v StoryTelleri	8	Active	Jakub Ondik
Code review		Active	Jakub Ondik
Pozvanie do projektu cez StoryTeller	3	Active	Bc. Miroslav Hurajt
Uloženie pozvánky		Closed	Bc. Martin Olejar
Code review		Active	Jakub Ondik
Moznost vytvorenia viacerych scenarov pre skicu	8	Closed	Jakub Ondik
Code review		Closed	Bc. Martin Olejar
Vytvorenie tlacidla na zdielanie skice (url)	2	Closed	Jakub Ondik
Vytvorenie route pre zobrazenie skice v systeme		Closed	Jakub Ondik
Vytvorenie route na zobrazenie nahladu skice		Closed	Jakub Ondik
Vytvorenie controllera s logikou zobrazovania nahladu skice		Closed	Jakub Ondik
Code review		Closed	Bc. Adam Neupauer
Generovanie layoutu pre native Android	3	Active	Jakub Ondik
Code review		New	Jakub Ondik
Code review		New	Bc. Martin Olejar
IITSRC		Closed	Jakub Ondik
Dotaznik		Closed	Bc. Miroslav Hurajt
robime.it		Closed	Bc. Martin Olejar
Plagat		Closed	Bc. Martin Olejar
Plagat		Closed	Bc. Patrik Januska
Plagat		Closed	Jakub Ondik

Obrázok 27: Úlohy šprintu Thor - pokračovanie

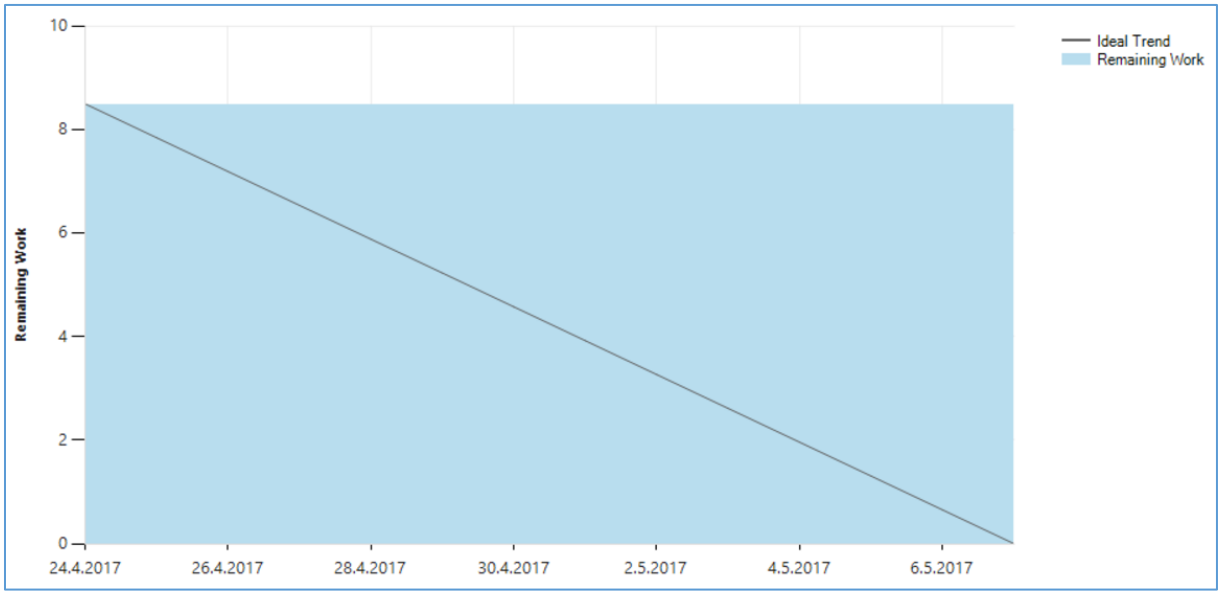
Obrázok 28: Burndown chart pre šprint Thor

4.11 Šprint 11 – Tyr

Náplňou posledného šprintu, ktorého väčšia časť prebiehala po konferencii IIT.SRC, bola finalizácia dokumentácie a refaktoring. Hlavne kvôli dokončeniu projektov na iných predmetoch a odovzdaní prvej časti diplomovej práce sme sa však dohodli, že budeme naplánované úlohy riešiť po spomenutých odovzdaniach. Všetky úlohy je možné vidieť na obrázku 29. Velocity tohto šprintu bola 3. Postup práce je zobrazený na obrázku 30.

Title	Story Points	State	Assigned To
Finalizacia dokumentacie		Active	Bc. Martin Olejar
Spravit protokol o testovani		New	Bc. Miroslav Hurajt
Upravit moduly		New	
Upravit component diagram na formu vrstiev a opisat		New	
Dat dokopy datovy model		Closed	Bc. Martin Olejar
Upravit prirucku na scenarovu formu		Active	Bc. Martin Olejar
Globalna retrospektiva + sprint		New	Bc. Martin Olejar
Refactoring Drawable objektov v JavaScripte	3	New	Bc. Adam Neupauer
Vytvorenie factory		New	Bc. Miroslav Hurajt
Uprava volania funkcie v template		New	Bc. Adam Neupauer
Uprava sketch controllera		New	Bc. Miroslav Hurajt
Code review		New	Jakub Ondik

Obrázok 29: Úlohy šprintu Tyr



Obrázok 30: Burndown chart pre šprint Tyr

5 Používané metodiky

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

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

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

5.1 Metodika plánovania

Metodika hovorí o mikromanažmente úloh, teda o zodpovednosti človeka za používateľský príbeh. Zodpovedný človek za používateľský príbeh by mal po aktivovaní používateľského príbehu zorganizovať stretnutie, na ktorom sa rozoberie obsah jednotlivých úloh, teda vykoná sa detailnejšia analýza problému a jednoduchý návrh riešenia úloh. Ďalej by sa mala definovať priorita úloh a poradie vypracovávania úloh a stanovenie dátumov, v ktorých budú dané úlohy hotové.
Metodika obsahuje aj stavy používateľského príbehu, ktoré je nutné priebežne meniť a ďalšie pravidlá pre riešenie používateľských príbehov.

Odkaz na metodiku: https://1drv.ms/b/s!AnORgXjFAcOEhUAdTdwsVIhiO0Ib

5.2 Metodika konvencií písania zdrojového kódu – Coding Standards

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

Odkaz na metodiku: https://1drv.ms/b/s!AnORgXjFAcOEhEkdTdwsVIhiO0Ib

5.3 Metodika testovania zdrojového kódu

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

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

Odkaz na metodiku: https://1drv.ms/b/s!AnORgXjFAcOEhUIdTdwsVIhiO0Ib

5.4 Metodika prehliadok zdrojového kódu – Code review

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

Odkaz na metodiku: https://1drv.ms/b/s!AnORgXjFAcOEhXbWQvlj1Dmin1uA

5.5 Metodika verziovania zdrojového kódu

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

Odkaz na metodiku: https://1drv.ms/b/s!AnORgXjFAcOEhGUdTdwsVIhiO0Ib

5.6 Metodika písania dokumentácie

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

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

6 Globálna retrospektíva

Táto kapitola obsahuje globálnu retrospektívu pre zimný a letný semester. Počas celého zimného 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.

V nasledujúcich týždňoch zimného semestra a v celom letnom semestri sme pokračovali v implementácii nášho systému. V letnom semestri sme tiež spísali retrospektívu po každom šprinte na tímovom stretnutí.

Na základe retrospektív jednotlivých šprintov v zimnom a letnom semestri sme určili nasledujúce problémy:

- nedostatočné vyjadrovanie sa k dianiu v tíme,
- odkladanie práce na poslednú chvíľu pred koncom šprintu, čoho následkom bolo viackrát preloženie úloh do ďalšieho šprintu, tento problém bol niekedy spojený aj s povinnosťami v iných predmetoch,
- nedostatočný opis a mikromanažment úloh, mikromanažment zahŕňa definovanie závislostí medzi úlohami, ich analýzu a návrh, následkom boli problémy s vyriešením úloh a blokovanie úloh,
- chýbajúca samostatnosť pri štúdiu, používanie stackoverflow namiesto dokumentácie.

Uvedené problémy sme riešili nasledovne. Nedostatočný opis a mikromanažment úloh sme sa snažili zlepšiť hlbšou a dlhšou analýzou úloh na začiatku šprintu na tímovom stretnutí. Počas druhého tímového stretnutia v šprinte sme často riešili problémy, ktoré sa vyskytli pri riešení úloh. Riešenie ostatných problémov bolo ponechané na jednotlivcoch, ktorých sa týkali.

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, zmeny hesla a zobrazenia zariadení,

na ktorých je v systéme prihlásený. Používateľ môže tiež vytvoriť nový projekt a upravovať jeho nastavenia. Systém ponúka možnosť zobrazenia zoznamu dostupných projektov používateľa a zoznamu používateľov, ktorí sú v danom projekte. Ďalšou možnosťou je zmena rolí používateľov v konkrétnom projekte.

Kľúčovou funkcionalitou je možnosť vytvárania skíc k projektu, čo zahŕňa kreslenie prvkov skice na plátno, úprava ich vlastností a ich odstránenie a tvorba scenárov ku skici. Systém tiež umožňuje stiahnutie HTML a Android layoutu skice a zdieľanie skice. Pre každý scenár skice je automaticky vytvorený test, ktorý je možné spustiť a sledovať prostredníctvom videa. V súvislosti s testami systém poskytuje históriu vykonaných testov, ktorá obsahuje ich trvanie a výsledky.

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
Kontakt:	storyteller-04@googlegroups.com
Vypracoval:	Bc. Adam Neupauer

1 Metodika integrácie a nasadzovania

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

Server

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

Vetvy

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

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

Nasadzovací systém - TFS

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

Backend

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

Backend build & deploy

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

Kroky :

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

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

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

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

Client

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

Client deploy

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

Kroky :

- PowerShell script

Nasadenie aplikácie client obsahuje len jeden krok – PowerShell script. Client totižto nepotrebuje zostavovanie, a teda stačí prekopirovať aplikáciu na server. PowerShell script najprv vykoná kompresiu projektu pre rýchlejšie odosielanie, a následne pomocou SCP prekopíruje súbory na server. Pomocou SSH potom spustí na 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
Kontakt:	storyteller-04@googlegroups.com
Vypracoval:	Bc. Miroslav Hurajt

1 Komunikačné nástroje

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

1.1 Slack

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

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

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

1.2 Komunikácia v TFS

Diskutovať o vrátení pull requestu a bližšie objasňovať vrátené časti je možné aj v nástroji Team 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 tímového projektu môžeme diskutovať aj vo vytvorenej diskusnej kupine 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 témam sa môže vyjadriť aj vedúci tímu.

1.4 E-mail

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

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

Tímový projekt

Story Teller

Metodika konvencií písania zdrojového kódu

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
Kontakt:	storyteller-04@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 upper 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,** s výnimkou konštánt. Konštanty musia byť uvedené použitím spôsobu shouting snake case, musia byť uvedené na začiatku triedy a musia obsahovať modifikátory **static** a **final**. Taktiež musí byť dodržané pravidlo jednej premennej na jeden riadok.

Príklad pre metódy:

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

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

Príklad pre atribúty:

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

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

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

1.1.2 Formátovanie kódu

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

Príklad:

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

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

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

Príklad:

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

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

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

Príklad:

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


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

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

V prípade reťazenia volaní metód je nutné prvú reťazenú metódu (t. j. druhú volanú metódu v poradí) umiestniť na nový riadok a vhodne odsadiť. Taktiež by žiaden riadok nemal presahovať dĺžku 80 znakov. V prípade situácie, že niektorý riadok túto 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 anotáciu @RequestMapping s cestou podľa názvu daného zdroja (okrem prípadov controllera, v ktorom sú umiestnené endpointy týkajúce sa rôznych zdrojov) a anotáciu @RestController. Jednotlivé endpointy pre daný zdroj už teda neobsahujú názov zdroja.
Príklad:

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

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

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

Príklad:

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

Tieto repozitáre nie je nutné ďalej implementovať.

1.2 Konvencie písania zdrojového kódu pre jazyk JavaScript

Táto podkapitola uvádza konvencie zdrojového kódu pre jazyk JavaScript, avšak s hlavným zameraním na rámec AngularJS. Konvencie, ktoré tu nie sú 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.

1.3 Štýly názvov

Lower Camel Case – userService, authorityRepository
Upper Camel Case – User, Authority, AuthenticationFilter
Snake Case – buf_ws_msg, msg_template
Shouting Snake Case – REFRESH_INTERVAL, AUTH_HEADER_NAME
Hungarian notation – bActive, lId, iAge

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

Tímový projekt

Story Teller

Metodika plánovania

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

1 Riešenie používateľských príbehov

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

1.1 Mikromanažment používateľského príbehu

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

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

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

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

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

1.3 Ďalšie pravidlá

Pre riešenie používateľských príbehov platia ďalšie pravidlá:

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

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

Tímový projekt

Story Teller

Metodika k prehliadke kódu

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
Kontakt: storyteller-04@googlegroups.com
Vypracoval: Bc. Ondrej Hamara (člen tímu do 19. 4. 2017)

1 Prehliadka kódu

1.1 Čo je to prehliadka kódu?

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

1.2 Prečo robiť prehliadku kódu?

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

1.3 Postup pri prehliadke kódu

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

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

1.4 Pohľad posudzovateľa na kód

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

- Coding conventions
- Testy a Jdoc

- Miesto zmeny
- Neduplikovať kód
- Ošetrenie chybových stavov

1.4.1 Coding conventions

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

1.4.2 Testy a Jdoc

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

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

1.4.3 Miesto zmeny

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

1.4.4 Neduplikovať kód

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

1.4.5 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 pri prehliadke kódu

- 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 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
Kontakt:	storyteller-04@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.

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

1.1.2 Spustenie a vyhodnotenie testu

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

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

1.2 Testovanie frontendu aplikácie

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

1.2.1 Vytvorenie testu

1. V prípade, že ide o otestovanie, či sa v prehliadači zobrazujú informácie z databázy, je potrebné tieto informácie predtým do databázy manuálne pridať.
2. Na strane backendu je následne potrebné v súbore **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. Následne po zadaní konkrétnej URL cesty prejdeme manuálne scenár, ktorého funkcionalitu sme implementovali a teraz ju chceme otestovať. Ak sa pri prechádzaní-testovaní scenára zobrazia relevantné výsledky (napríklad zoznam projektov, ktoré sme v predchádzajúcich krokoch pridali do databázy), test považujeme za úspešný.
7. Ak nastane problém, v ktorom nedostaneme odpoveď akú sme očakávali (napríklad zostane tabuľka projektov prázdna), pozrieme sa, či nie je chyba opísaná v konzole webového prehliadača alebo v konzole servera a odstránime ju.

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

Tímový projekt

Story Teller

Metodika k verziovaniu

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

1 Pravidlá pre vetvenie zdrojového kódu

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

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

1.1 Vytvorenie vetvy pre používateľský príbeh

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

- píše sa v angličtine,
- má tvar "Číslo user story.Krátky popis user story", napr. 4567.Devices,
- v prípade, že sa krátky popis používateľského príbehu skladá z viacerých slov, tieto slová sa začínajú veľkým písmenom a 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"*, kde tento popis môže pre lepšie vyhľadávanie obsahovať kľúčové slová ako napr. added, fixed, removed
- pri viacerých úlohách je tvar popisu nasledujúci:
  "[Tasks number1, number2] Popis pridanej funkcionality a iných dôležitých vecí"
- ak sú vypracované všetky úlohy v rámci používateľského príbehu, pridáme vetvu **dev** do našej pracovnej vetvy (angl. **merge dev into branch**)

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

1.3 Pravidlá pre spájanie vetiev

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

1.4 Vytvorenie požiadavky na kontrolu používateľského príbehu

Po vypracovaní všetkých úloh v rámci používateľského príbehu, čo zahŕňa 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:

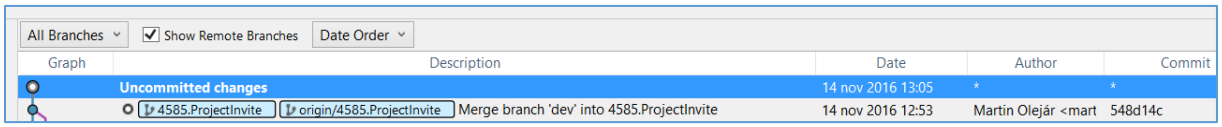
1. v SourceTree klikneme v hornom menu na **Clone / New,**
2. vyplníme **Source Path / URL** nasledujúcim odkazom:
   https://tfs.fiit.stuba.sk:8443/tfs/StudentsProjects/StoryTeller/_git/StoryTeller
3. vyplníme **Destination Path**, t.j. cestu k priečinku, kde bude uložený lokálny repozitár,
4. potvrdíme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

3.2 Práca vo vetve

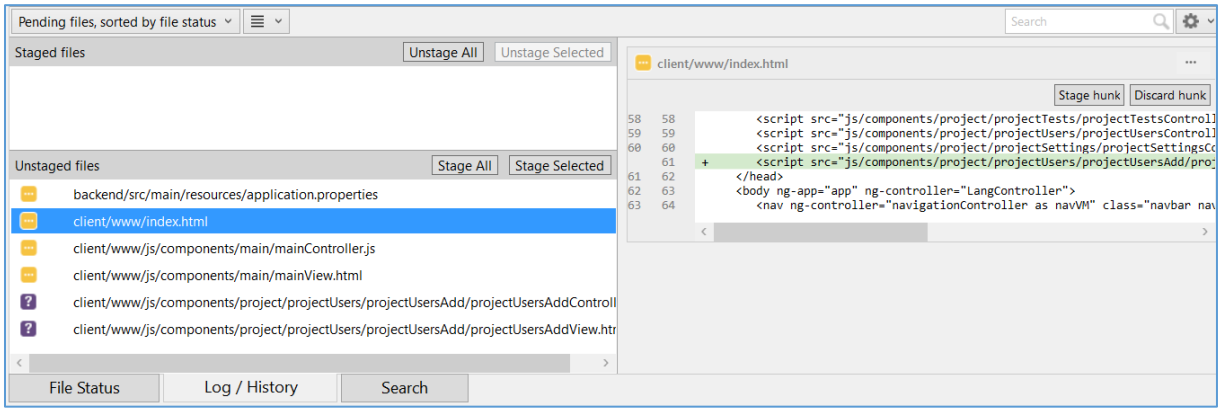
Pracovnú vetvu si otvoríme tak, že v časti BRANCHES klikneme dvakrát na danú vetvu. Pri úplne prvom otváraní vetvy je nutné túto akciu vykonať v časti REMOTES.

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

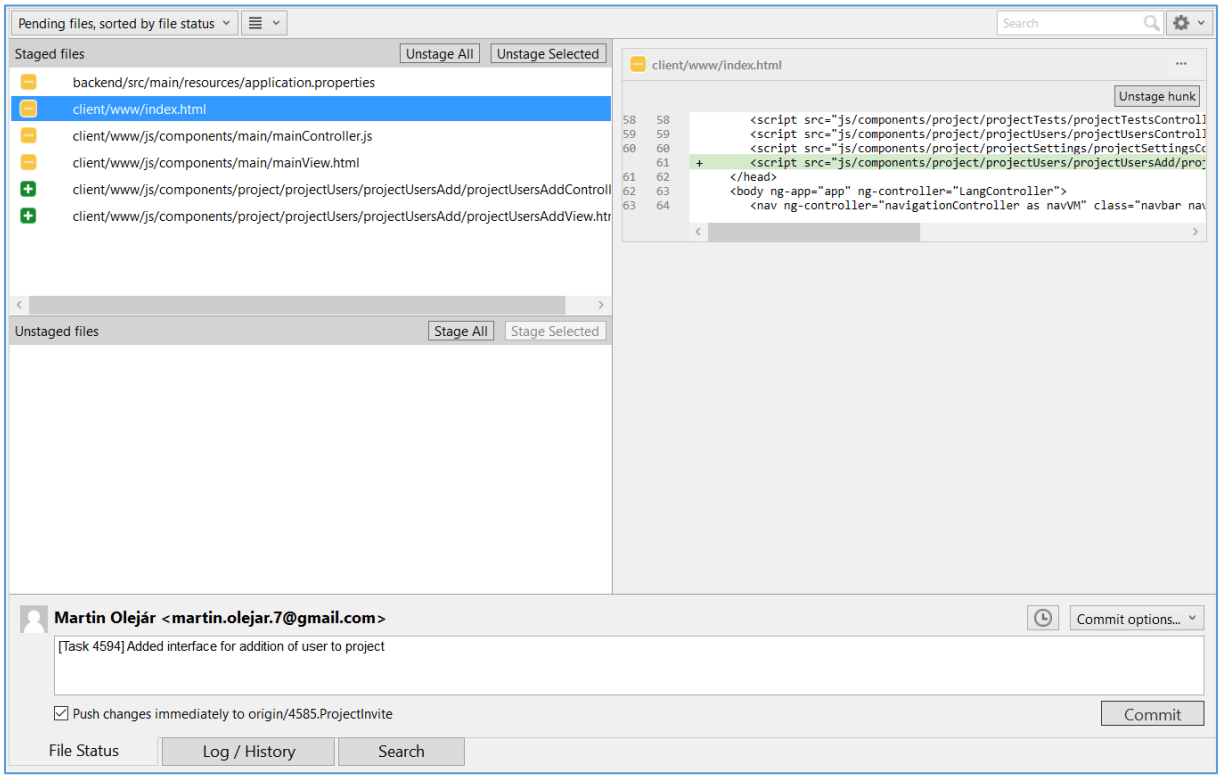


Obrázok 2: Zobrazenie zmien vo vetve

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:

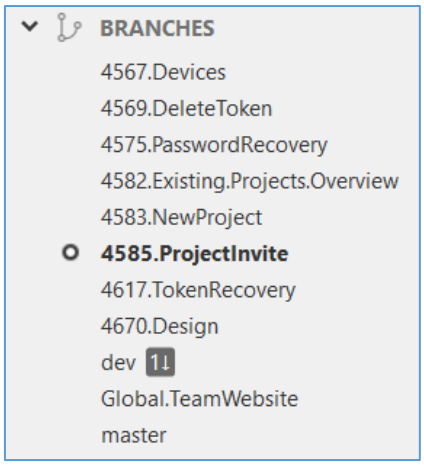
1. klikneme v dolnej časti na **File Status,**
2. v časti **Unstaged files** uvidíme všetky modifikované súbory, klikneme na **Stage All,**
3. v dolnej časti vyplníme popis commitu podľa definovaných pravidiel,
4. zaškrtneme **Push changes immediately to ...,**
5. výsledná situácia je zobrazená na obr. 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

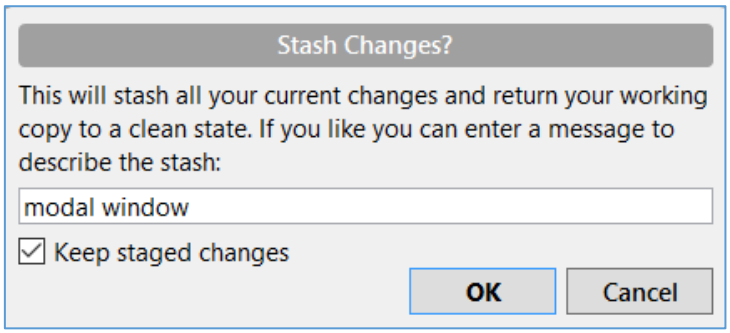
Ak niekto iný vykonal zmeny vo vetvách, je potrebné si tieto zmeny stiahnuť. Najprv klikneme na **Fetch** v hornom menu. Pri každej vetve, ktorá nie je lokálne aktuálna, teda niekto iný ju modifikoval a vykonal commit a jeho zmeny nie sú v lokálnom repozitári, sa zobrazí počet vykonaných commitov a šípka 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

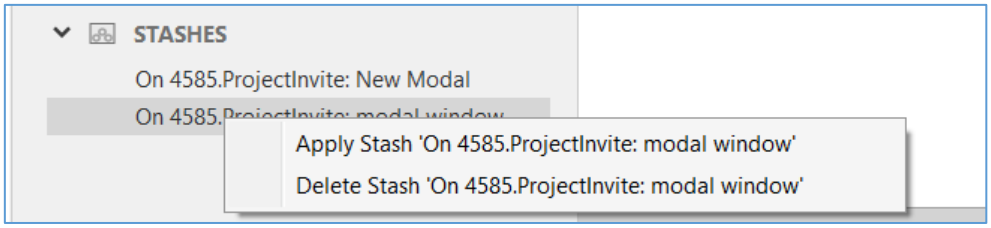
Ak pracujeme v nejakej vetve a potrebujeme vykonať nejaké zmeny v inej vetve, teda musíme sa prepnúť do inej vetvy, je potrebné si predtým vykonané zmeny odložiť (angl. **stash**). V hornom menu klikneme na **Stash**, zobrazí sa modálne okno na obr. 5, v ktorom vyplníme správu pre popis stash-u a zaškrtneme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

Šablóna na dokumentácie

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
Kontakt:	storyteller-04@googlegroups.com
Vypracoval:	Bc. Martin Olejár

Obsah

1 Kapitola číslo 1

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

1.1 Metodika

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

1.1.1 Text – štýl Normálny

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

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

1.1.2 Odrážky a číslovanie

Príklad odrážok:

- sdffsd
    - 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 Paragraph Heading