Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Dokumentácia k riadeniu projektu

Bc. Michal Cihák

Bc. Michal Gajdoš

Bc. Lukáš Masár

Bc. Pavol Michálek

Bc. Vladimír Osvald

Bc. Matúš Pikuliak

Bc. Tomáš Sýkora

Tím (č. 12): Cats Can Code

Predmet: Tímový projekt I

Ročník: 1.

Akademický rok: 2014/2015

Vedúci práce: Ing. Lukáš Turský

Obsah

1 Úvod

Tento dokument obsahuje podrobnú dokumentáciu k riadeniu tímového projektu. Tím podieľajúci sa na tomto projekte je zložený zo siedmich študentov a nesie názov Cats Can Code.

Cieľom nášho projektu je vytvoriť webovú službu, ktorá používateľom umožní čo najjednoduchšie plánovanie stretnutí za pomoci atraktívneho grafického používateľského rozhrania a bez nutnosti registrácie. Plánovanie stretnutia bude veľmi flexibilné. Zakladateľ udalosti môže definovať miesto a termín stretnutia, vytvoriť hlasovanie za miesto a termín alebo jednoducho nechá pozvaných, aby navrhli čo vyhovuje im. Termín udalosti môže byť konkrétny čas alebo časový úsek. Samozrejmosťou je aj možnosť zahlasovať za viac miest či termínov, pričom bude jasne viditeľné kto za čo hlasoval a ktorý termín, respektíve miesto má aktuálne najviac hlasov.

Druhá a tretia kapitola sa zaoberá kompetenciami členov nášho tímu a ich pridelenými manažérskymi úlohami.

Štvrtá kapitola obsahuje metodiky používané v riadení tímového projektu. Každá metodika tvorí jednu podkapitolu. Obsahom tejto kapitoly sú nasledujúce metodiky: metodika testovania, metodika evidencie úloh, metodika verziovania, metodika dokumentovania, metodika revízie kódu, metodika komunikácie, metodika písania kódu

Prílohami dokumentácie k riadeniu sú zápisnice zo všetkých stretnutí, ktoré opisujú ich priebeh vrátane zoznamu prítomných členov tímu, tiež aj jednotlivé metodiky a výstupy z nástroja JIRA.

2 Role členov tímu

2.1 Manažérske úlohy

Rozdelenie manažérskych úloh v našom tíme je nasledovné:

- Matúš Pikuliak - manažér verziovania, front-end
- Michal Gajdoš - manažér kvality
- Lukáš Masár - manažér integrácie projektu, aplikačný server
- Vladimír Osvald - manažér dokumentovania
- Michal Cihák - manažér testovania, jadro aplikácie
- Pavol Michálek - manažér komunikácie, tímová stránka, front-end
- Tomáš Sýkora - manažér plánovania, rizík

2.2 Aplikácia manažérskych úloh

2.2.1 Manažment verziovania

Pre správu verzií kódu nášho projektu používame distribuovaný verziovací systém Git. Na webovej službe bitbucket sme si zriadili vzdialený repozitár, na ktorom udržiavame verzie našej webovej aplikácie v rozličných gitovských vetvách. Verziovanie v Git používame aj pri udržiavaní našej tímovej webovej stránky.

V súčasnosti pri počiatočnom vývoji používame jednu development vetvu. Dôležité milníky vo vývoji označujeme tagmi, pomocou ktorých dokážeme potom jednotlivé verize odsledovať. V ďalších fázach sa chystáme vytvoriť ďalšie vetvy, staging a release. Tieto budú slúžiť pri vytváraní nasadzovaných releasov do prevádzky.

Takisto budeme podľa potreby vytvárať tzv. *features* vetvy pre skúšanie nových technológií či prebudovanie rozsiahlych častí systému. Ako nástroj pre prácu s repozitármi používame integráciu Git v IDE Rubymine, ktoré v tíme používame. Pre verziovanie sme si špecifikovali metodiku verziovania, podľa ktorej postupujeme, resp. pri niektorých častiach len budeme postupovať. Táto metodika sa nachádza v prílohe B.5.

2.2.2 Manažment kvality

V rámci vývoja produktu je potrebné zabezpečiť kvalitu nielen výsledného produktu, ale aj jeho jednotlivých častí, počas jeho vývoja. Je nutné, aby jednotlivé úlohy boli vykonávané s vysokou mierou kvality od začiatku práce na nich, aby sa tak zabezpečilo, že náklady na ich neskoršie úpravy ako aj údržbu budú minimálne. Produkt, ktorý vyvíjame nie je na trhu nový. Existujúce riešenia však nie sú veľmi populárne. Je to možno práve z dôvodu ich nedostatočnej kvality. Preto je potrebné, aby náš produktu bol na vysokej úrovni, aby sme tak neodradili našich potenciálnych zákazníkov. Keďže samotný produkt bude vytváraný postupne, a jednotlivé moduly nebudú vždy kompletné, je potrebné dohliadnuť na kvalitu dodávanej verzie a teda jednotlivých častí.

Na vytvorenie kvalitného produktu je však potrebné aby aj jednotlivé procesy v tíme boli vykonávané s zodpovedne a s veľkou mierou kvality. Jednotliví členovia tímu sa preto majú možnosť vyjadriť ku stavu projektu v podstate kedykoľvek, avšak hlavne na tímovom stretnutí počas retrospektívy uplynulého šprintu. Práve na základe retrospektívy dokážeme resp. sme dokázali jednotlivé procesy v tíme značne zlepšiť. Dokázali sme zlepšiť hlavne komunikáciu a využívanie zdrojov, vďaka čomu je náš produkt resp. jeho jednotlivé prototypy vyvíjané kvalitne. Zároveň sa ta pridávanie ďalších modulov a funkcionalít stáva jednoduchšími.

2.2.3 Manažment integrácie projektu

Na vývoji produktu sa podieľajú viacerí developeri, ktorí pracujú na svojich lokálnych strojoch. Výstupy ich práce sa koordinujú pomocou systému Git. Pre zabezpečenie aktuálnej verzie produktu dostupnej cez internet, beží na serveri skript, ktorý pravidelne každých 5 minút aktualizuje verziu produktu na serveri. Skript sťahuje najnovšiu verziu produktu z git repozitáru pre development aj release vetvu. Zároveň, ak je to potrebné, inštaluje doplňujúce gems a vykonáva úpravy v databáze. Obdobným spôsobom fungujú aj skripty, ktoré zabezpečujú aktuálnosť produkčného prostredia, teda release vetvy, aj tímovej stránky.

2.2.4 Manažment dokumentovania

V rámci tímového projektu vnikajú rôzne dokumentácie. Úlohou manažmentu dokumentovania bolo určiť nástroje, pomocou ktorých sa budú dokumentácie vytvárať a stanoviť miesto, kde sa budú dokumenty nachádzať. Vznikli postupy, ktoré je potrebné dodržiavať pri písaní dokumentov a bola jasne stanovená výsledná forma dokumentácie. Podrobný proces písania dokumentácie k inžinierskemu dielu bol spísaný v príslušnej metodike.

2.2.5 Manažment testovania

Pri vyvíjaní softvéru v tímovom projekte je potrebné zachovať integritu vyvíjaného softvéru počas celého vývoja a zabezpečiť aby jednotlivé časti softvéru vyvíjané rôznymi vývojármi správne fungovali pri vytváraní nových verzií. Preto je vhodné vyvíjaný softvér pravidelne testovať a vytvárať potrebné testy. Úlohou manažmentu testovania bolo určiť vhodné nástroje na testovanie a čo a kedy sa bude primárne testovať. Na základe analýzy manažmentom testovania a po dohode s tímom boli zvolené testovacie nástroje, ktoré sú spísané v prislúchajúcej metodike k testovaniu nachádzajúcej sa v prílohe B.3. V tejto metodike sú ďalej popísané spôsoby vytvárania základných testov vhodných pre náš projekt. Okrem testovania pomocou takto vytvorených testov prebiehajú testy v našom tíme aj samotnými členmi - v tomto prípade testermi. A to vždy pri každej splnenej úlohe, ktorá je posunutá na prehliadku. Ďalšie testovanie prebieha vždy aj pred každým koncom šprintu, kedy sa prezentuje momentálna verzia aplikácie (demo), kde sa kontroluje či demo spĺňa každú požadovanú funkcionalitu.

2.2.6 Manažment komunikácie

Bolo potrebné zabezpečiť, aby všetci členovia tímu komunikovali pomocou jednotných komunikačných nástrojov. Po analýze a sumarizácii vhodných nástrojov bolo tímu predložených niekoľko návrhov. Pod vedením manažmentu komunikácie, ktorý dohliadal na správny počet komunikačných kanálov pre všetky situácie, boli v tíme schválené nasledujúce komunikačné nástroje:

- **HipChat** - štandardná online textová komunikácia a rýchle reporty o posledných aktivitách v ostatných používaných nástrojoch. Manažment komunikácie dohliada na dodržiavaní pravidiel stanovených v metodike a stanovuje a usmerňuje nové témy.
- **Google Hangouts** - hovory a videohovory, bez zásahu manažmentu
- **Jira** - evidencia úloh. V tomto nástroji je potrebný dohľad manažmentu komunikácie nad tvorbou a správou zadefinovaných úloh, aby boli všetkým členom zrozumiteľné a jasné.
- **Google Drive** - zdieľanie súborov. Manažment zadefinoval pravidlá práce s týmto úložiskom.
- **Webové sídlo tímu** - externá komunikácia. Manažment schválil formu webovej prezentácie tímu, a dohliada na aktualizovanie obsahu webu.

2.2.7 Manažment plánovania

Táto oblasť manažmentu bola zabezpečená pomocou nástroja JIRA. Pred začiatkom každého šprintu si definujeme úlohy, ktoré chceme stihnúť do konca nadchádzajúceho šprintu. Tieto úlohy následne ohodnotíme relevantným počtom story points a to tak, že každý člen estimuje daný task podľa seba. Potom podľa strednej hodnoty každý člen, ktorý oestimoval vyššie, tak odôvodní svoj počet a tak isto aj člen z nižším počtom ako stredná hodnota ohodnotí tento svoj počet. Následne sa hodnotí znova a takto to iteruje až kým sa nezhodneme. Keď máme oestimované úlohy, tak ich pridávame do nástroja JIRA podľa metodiky evidencie úloh, ktorá je uvedená v B.2. V metodike je uvedený aj postup pomenovania úlohy.

Keď už máme úlohy uložené v nástroji, vytvorí sa šprint, presunú sa do neho úlohy na ktorých sme sa dohodli a následne sa spustí šprint. Na stretnutí sme si spoločne rozdelili úlohy a vždy mal niekto čo robiť. Každý člen si potom priradí v nástroji JIRA svoju úlohu, a začne na nej pracovať.

Ak už je task hotový, presunie sa do stĺpcu *Review*. Potom daný task iný člen tímu skontroluje a zvaliduje. Táto kontrola sa robí podľa metodiky code review, ktorá je uvedená v B.4. Ak je v úlohe chyba, vytvorí sa relevantná úloha typu *Bug*, taktiež podľa metodiky B.2.

Na konci šprintu sa potom jednotlivé vypracované úlohy prezentujú vlastníkovi produktu, ktorý zhodnotí náš pokrok, spravíme retrospektívu za šprint a potom sa začína príprava na ďalší šprint.

2.2.8 Manažment rizík

Počas vývoja produktu vznikajú rôzne riziká a preto je ich identifikácia a následne aj zamedzenie výskytu dôležité. Každé riziko ma definovanú úroveň rozsahu škôd:

- Nízky - riziko je minimálne, netreba vynaložiť veľkú námahu na jeho vyriešenie,
- Stredný - riziko je stredné, je potrebné ho riešiť,
- Vysoký - riziko je vysoké, je potrebné ho okamžite riešiť.

Za doterajší priebeh vzniklo zopár rizík. Napríklad viaznuca komunikácia medzi členmi tímu, čo je vysoké riziko, keďže pre správny postup vývoja je komunikácia kritická. Toto riziko sme vyriešili použitím nástroja HipChat, ktorý je spomenutý vyššie.

Ďalšie riziko vzniklo pri zistení, že dvaja kolegovia odchádzajú na Erasmus. Ako riešenie tohto problému sme navrhli zopár kolaboračných nástrojov, potrebných pri vývoji, ďalší nástroj na zdieľanie obrazovky a videohovory. Toto riziko je taktiež vysoké, keďže by výrazne ovplyvnilo rýchlosť vývoja produktu.

2.3 Podiel členov na dokumentácií

Nasledujúca tabuľka zobrazuje podiel členov tímu na tvorbe tohto dokumentu.

Úvod	Tomáš Sýkora
Zoznam kompetencií	Každý za seba
Úlohy členov tímu	Vladimír Osvald
Manažment verziovania	Matúš Pikuliak
Manažment kvality	Michal Gajdoš
Manažment integrácie projektu	Lukáš Masár
Manažment dokumentovania	Vladimír Osvald

Manažment testovania	Michal Cihák
Manažment komunikácie	Pavol Michálek
Manažment rizík	Tomáš Sýkora
Metodika testovania	Michal Cihák
Metodika evidencie úloh	Tomáš Sýkora
Metodika verziovania	Matúš Pikuliak
Metodika dokumentovania	Vladimír Osvald
Metodika revízie kódu	Lukáš Masár
Metodika komunikácie	Pavol Michálek
Metodika komentovania zdrojového kódu	Michal Gajdoš

Tabuľka 1. Podiel členov na dokumentácií

3 Sumarizácia šprintov

V kapitole sa nachádzajú sumarizácie šprintov.

3.1 Prvý šprint

Navrhli sme architektúru aplikácie a databázový model a vytvorili sme jadro aplikácie. Jednotlivé úlohy vypracované v rámci šprintu je možné vidieť v prílohe D.1.

3.2 Druhý šprint

Vytvorili sme prvú verziu grafického používateľského rozhrania. Jednotlivé úlohy vypracované v rámci šprintu je možné vidieť v prílohe D.2.

3.3 Tretí šprint

V šprinte sa kompletizovali dokumenty potrebné pre prvý kontrolný bod. Jednotlivé úlohy vypracované v rámci šprintu je možné vidieť v prílohe D.3.

3.4 Štvrtý šprint

V tomto šprinte došlo k "zahodeniu" časti doterajšieho produktu, hlavne v stránke dizajnu, do ktorého bola implementovaná časová os. Taktiež prebehlo rozsiahle refaktorovanie. Jednotlivé úlohy je možné vidieť v prílohe D.4.

3.5 Piaty šprint

Ako je možné vidieť v prílohe D.5, nastal posun v oblasti týkajúcej sa samotných používateľov nášho produktu. Bola vytvorená profilová stránka pre používateľov, ako aj implementované automatické notifikácie posielane na mailovú adresu používateľa.

4 Používané metodiky

V tejto časti dokumentu sú stručne popísané jednotlivé metodiky, ktorými sa náš tím riadi:

- Metodika testovania
- Metodika evidencie úloh
- Metodika verziovania
- Metodika dokumentovania
- Metodika revízie kódu
- Metodika komunikácie
- Metodika komentovania kódu

4.1 Metodika testovania

Metodika určuje testovacie nástroje, ktoré sa používajú v rámci nášho projektu. Ďalej obsahuje popísané spôsoby vytvárania základných testov vhodných pre náš projekt.

Kompletná metodika je priložená v prílohe B.3.

4.2 Metodika evidencie úloh

Metodika definuje prácu s úlohami v nástroji JIRA, aké typy úloh používame, ktorý typ a kedy použijeme a postup pri vytváraní úlohy.

Kompletná metodika je priložená v prílohe B.2.

4.3 Metodika verziovania

Metodika pojednáva o procesoch pri verziovaní kódu a súborov všeobecne pri vyvíjaní implementácie nášho projektu. Procesy slúžia programátorom pri písaní kódu a jeho synchronizácii v rámci tímu.

Kompletná metodika je priložená v prílohe B.5.

4.4 Metodika dokumentovania

Metodika obsahuje pracovné procesy a postupy potrebné pri písaní dokumentácie k inžinierskemu dielu pomocou nástrojov Wiki a Microsoft Word. Obsahuje pokyny pre editáciu a finalizáciu dokumentu.

Kompletná metodika je priložená v prílohe B.7.

4.5 Metodika revízie kódu

Účelom tejto metodiky je stanoviť postup, ktorý dokáže zachytiť chyby v kóde pred tým, ako sa dostanú k používateľovi. Je to základná činnosť vykonávaná za účelom zabezpečenia kvality kódu. Jej zanedbanie môže viesť k zbytočne zvýšeným nákladom v ďalších etapách projektu.

Kompletná metodika je priložená v prílohe B.4.

4.6 Metodika komunikácie

Metodika komunikácie v tíme popisu postupy používané pri riadení nášho projektu. Ďalej definuje použité nástroje a spôsob ich použitia v konkrétnych prípadoch.

Kompletná metodika je priložená v prílohe B.6.

4.7 Metodika komentovania kódu

Táto metodika obsahuje postup a spôsob písania komentárov zdrojového kódu, vo vývojovom prostredí RubyMine, ako aj následné vytvorenie dokumentácie pomocou aplikácie Rdoc.

Kompletná metodika je priložená s prílohe B.1.

5 Globálna retrospektívna

V rámci nášho projektu sa snažíme pracovať čo najefektívnejšie. Ako tím sme však vznikli len pred pár týždňami, a preto neustále zlepšujeme naše postupy, aby tak vyhovovali schopnostiam a znalostiam jednotlivcov a aby sme ako celok dosahovali čo možno najlepšie výsledky. Po každom šprinte preto máme retrospektívu. Práve na základe nich sa nám výrazne podarilo zlepšiť prípadne odstrániť veľké množstvo nedostatkov. Aj napriek tomu je stále možnosť našu prácu v tíme vykonávať efektívnejšie aby naše výsledky boli ešte lepšie.

Začiatky bývajú ťažké, a teda ani ten náš nebol jednoduchý. Problémové boli hlavne prvé šprinty. Jedným z dôvodov bola potreba zvyknúť si na scrum. Ďalším a možno závažnejším problémom boli samotné úlohy, ktoré sme si definovali. Keďže sme začínali na úplne novom projekte z hľadiska vývoja (písania kódu) nebolo vždy možné pracovať paralelne. Práve na základe retrospektív po šprintoch, sa nám podarilo eliminovať tento problém pomocou zadefinovania úloh s interným označením “investigate”, aby tak vznikli aj neimplementačné úlohy a tak sme mohli pracovať paralelne na viacerých úlohách. Pri paralelnej práci vznikali v prvých šprintoch problémy pri zlučovaní lokálnych verzií, ktoré sme ku koncu semestra pociťovali stále menej.

Ďalším problémom je pre nás komunikácia. Aj tá sa každým šprintom zlepšuje, avšak stále nie je na úrovni, ktorú by mohla dosiahnuť. Tento problém si však uvedomujeme, a preto sa snažíme pracovať na jeho odstránení. Ide o informovanie ostatných členov tímu o stave resp. aktuálnych činnostiach. Keďže ako tím (čiže všetci) sa stretávame 1-2 krát týždenne, komunikáciu v tíme riešime v prvom rade zodpovedným písaním komentárov k daným úlohám v nástroji JIRA, aby tak bol jasný pokrok na jednotlivých úlohách. Čo sa týka ostatnej komunikácie, tá prebieha v nástroji HipChat.

Od začiatku semestra sme sa určite posunuli správnym smerom a procesy a postupy sa nám v tíme reálne podarilo (v niektorých smeroch niekoľkonásobne) zlepšiť. Rezervy však stále máme vo využívaní zdrojov - nie všetci členovia tímu pracujú tak, aby plne využili svoj potenciál (aj keď by možno chceli). Tento fakt je spôsobený aktivitami, ktoré sú často spojené s inými študijnými povinnosťami. Jednotlivým členom tímu tak neostáva dostatočné množstvo času na to, aby prejavili svoje schopnosti.

Do budúcna by sme teda chceli zlepšiť komunikáciu a teda aj spôsob a frekvenciu stretnutí. Nakoľko sa v našom tíme nachádzajú dvaja študenti, ktorí idú na výmenný pobyt (Erasmus) zmena resp. úprava komunikácie medzi členmi tímu sa bude musieť zmeniť - bude nutné rozšíriť online komunikáciu, aby tak nebola ohrozená výkonnosť nášho tímu.

A Zoznam kompetencií tímu

Michal Cihák

Záujmová oblasť: moderné webové technológie, mobilné technológie (ios), grafika

Znalosti: Ruby (Ruby on Rails), HTML5, CSS3, Sass, JavaScript, D3.js, SOAP, webové služby, OpenGL4.x

Doterajšie skúsenosti: Vypracovanie BP ako Ruby on Rails webovej aplikácie, kde som získal bohaté skúsenosti s komunikáciou s webovými službami s následnou vizualizáciou týchto dát klientovi v prehliadači. Tvorba webstránok - získanie skúseností s najnovšími CSS3 technológiami, jazykom Sass a JavaScriptom.

Michal Gajdoš

Záujmová oblasť: mobilné technológie, umelá inteligencia, webové technológie

Znalosti: C#, Java, MySQL, PHP, HTML5, CSS3

Doterajšie skúsenosti: Bakalárska práca, ktorej výsledkom je program v jazyku C# (WPF Aplikácia + MVVM), používaný dopravným podnikom mesta Bratislava.

Lukáš Masár

Záujmová oblasť: umelá inteligencia, distribuované systémy, webové technológie, sieťové služby

Znalosti: Erlang, Java, HTML, CSS, PHP, MySQL

Doterajšie skúsenosti: Bakalárska práca, ktorej témou bol distribuovaný rámec pre tvorbu škálovatelných webových aplikácii v jazyku Erlang. Momentálne pracujem v spoločnosti NASES na pozícii špecialistu na prevádzku sieťových služieb.

Pavol Michálek

záujmová oblasť: rozsiahle integrované webové systémy

Znalosti: PHP, HTML5, CSS3, Less, JavaScript (jQuery), D3.js, SOA technológie, Java,

Doterajšie skúsenosti: 1.5 roku web frontend developer a biznis analytik na projekte Multichannel pre banku VÚB (internet/mobile banking), 0.5 roku web developer (e-shopy, backend, administrácia), bakalárka - interaktívna webová vizualizácia

Vladimír Osvald

Záujmová oblasť: webové technológie, mobilné technológie (Android), aplikácie pre MS

Windows

Znalosti: HTML5, CSS3, PHP, C++, C#, Java, MySQL

Doterajšie skúsenosti: Vytvorenie niekoľkých jednoduhších webových stránok, vypracovanie bakalárskej práce v jazyku C#, kde som pracoval najmä s databázou MySQL, pričom som riešil komunikáciu medzi klientom a serverom, čo je využiteľné aj vo webových systémoch a pri mobilných technológiách.

Matúš Pikuliak

záujmová oblasť: webové technológie, umelá inteligencia

Znalosti: PHP, CodeIgniter FW, HTML5, CSS3, LESS, JavaScript, MySQL

Doterajšie skúsenosti: Mám skúsenosti s tvorením webových stránok nad MVC frameworkom CodeIgniter, ktoré spočívali v kompletnom spracovaní od grafického návrhu, cez implementáciu až po nasadenie na server. V jazyku PHP som programoval aj svoju bakalársku prácu týkajúcu sa evolučných algoritmov.

Tomáš Sýkora

záujmová oblasť: databázové technológie, webové technológie, siete, distribuované systémy

Znalosti: Java, MySQL, C, XML, UML, HTML, CSS

Doterajšie skúsenosti: Paralelná práca s tímovým projektom na prototype 3D UML, kde som vytvoril aplikáciu serializácie UML diagramov kvôli kompatibilite s inými CASE nástrojmi (výsledok BP). Ročná skúsenosť na pozícii IT Support v advokátskej firme Cechová & Partners.

B Metodiky

Táto príloha obsahuje vypracované metodiky od jednotlivých členov tímu.

B.1 Komentovanie zdrojového kódu

Autor: Michal Gajdoš

Nasledujúci materiál obsahuje postup a spôsob písania komentárov zdrojového kódu, vo vývojovom prostredí RubyMine, ako aj následné vytvorenie dokumentácie pomocou aplikácie **Rdoc**.

B.1.1 Povolenie komentárov slúžiacich na dokumentáciu

Pre správne fungovanie komentárov je potrebné povoliť dokumentačné komentáre zaškrtnutím políčka ***"Insert documentation comment stub"***. K políčku sa možno dostať pomocou ***"File">"Settings">"Editor">"Smart Keys"***, v časti ***"Enter".***

B.1.2 Všeobecné pokyny k písaniu komentárov

Pri komentovaní kódu je potrebné dodržiavať nasledovné všeobecné zásady:

- komentáre sú písané v anglickom jazyku,
- komentáre sú stručné a výstižné
- používať komentáre len na miestach, kde sú potrebné, aby sa zabránilo neprehľadnosti kódu.

Komentáre je nutné použiť na komentovanie každej:

- triedy,
- metódy a jej vstupných parametrov,
- zložitejšej prípadne nejasnej časti kódu.

B.1.3 Komentovanie v Ruby

Na komentovanie kódu v Ruby sa používa jednoriadková notácia, začínajúca znakom *"#"* a medzerou. Na takýto zápis možno použiť klávesovú skratku ***Ctrl + "/"***, ktorá celý riadok, v ktorom sa nachádza kurzor, označí ako komentár. (Komentár je automaticky ukončený na konci konkrétneho riadka.)

V prípade, že komentár presiahne dĺžku jedného riadka, respektíve sa nachádza vo viacerých riadkoch, začiatok každého riadku je označený "**#** ".

B.1.4 Komentovanie v JavaScript

Pri komentovaní zdrojového kódu písaného v jazyku JavaScript (*.js* tried) je možné použiť dva typy komentárov. V prípade jednoriadkového komentáru je možné použiť znaky "//", ktoré celý riadok resp. zvyšok riadka, nachádzajúci sa za týmito znakmi, označia ako komentár. Takýto komentár je možné použiť aj v prípade že je potrebné označiť ako komentár viacero riadkov. Označenie "//" je však potrebné uviesť na začiatku každého riadku.

V prípade viacriadkového komentáru je taktiež použiť notáciu "/*" na začiatku a "*/" na konci komentáru. Takéto označenie vytvorí komentár z obsahu nachádzajúceho sa medzi uvedenými ohraničeniami. Z dôvodu lepšej prehľadnosti je vhodné každý nový riadok (okrem prvého a posledného) komentáru označiť symbolom "*". Takýto komentár je taktiež možné použiť v prípade že je žiadúce označiť za komentár len časť riadku.

V ďalších častiach tejto metodiky sa uvádza postup pri komentovaní zdrojového kódu v jazyku Ruby. Rovnaké pravidlá však platia aj pri komentovaní kódu v jazyku JavaScript, pričom je potrebná len zámena notácií označujúcich komentár.

B.1.5 Komentovanie tried

Každá trieda obsahuje komentár, ktorý danú metódu opisuje. Takýto komentár sa nachádza respektíve začína na prvom riadku konkrétnej metódy. Tento komentár je potrebné pridať pri vytvorení samotnej triedy. V prípade správne nastaveného prostredia RubyMine, je takýto komentár pridaný automaticky pri vytvorení. Pri manuálnom pridaný komentáru je potrebné zachovať jeho formát – meno člena tímu, ktorý triedu vytvoril, ako aj dátum vytvorenia.

Príklad komentáru v *.js* súboroch:

```
/**
* Created by Piko on 29. 11. 2014.
*/
```

B.1.6 Komentovanie metód

Rovnako ako všetky triedy, tak aj všetky metódy sú uvádzané komentárom. Tento komentár sa nachádza resp. končí o riadok vyššie, ako je riadok, v ktorom je daná metóda definovaná (riadok, v ktorom sa daná metóda začína). Začiatkom respektíve prvou časťou komentáru je opis metódy, ktorej komentár prislúcha (ktorá sa nachádza pod komentárom). Pod týmto opisom sa nachádza prázdny zakomentovaný riadok (riadok obsahujúci len znak "**#**" a **medzeru** na začiatku).

V prípade, že sa v komentári uvádzajú vstupné premenné funkcie (ktoré je nutné uviesť, pokiaľ daná metóda nejaké vstupné parametre obsahuje), sú tieto uvedené samostatne v riadkoch, pričom názov premennej je uvedený medzi dvomi znakmi "**+**". Takto uvedenú premennú zároveň predchádza znak "*****" a **medzera** (takýto riadok sa vo vygenerovanej dokumentácií stáva položkou nečíslovanom zozname, so zvýraznením názvu premennej). Blok premenných je zároveň uvedený zakomentovaným riadkom, obsahujúcim "**=== Parameters**" (slúži na vytvorenie nadpisu tretej úrovne, v dokumentácií).

Pokiaľ sa v komentári nachádza aj návratová hodnota metódy (ktorú je nutné uviesť v prípade, že metóda návratovú hodnotu obsahuje), je uvedená za prípadnými parametrami respektíve na konci komentára samotnej metódy. Formát takéhoto komentára je "**# +return+**" a **popis návratovej hodnoty**.

Príklad:

```
# === Parameters
# * +a+ - first number
# * +b+ - second number
#
# +return+ -sum of numbers
```

Takýto komentár je potrebné pridať vždy pri vytvorení konkrétnej metódy, pričom prvotný tvar komentáru sa môže v závislosti od aktualizácie metódy zmeniť. Výnimkou, kedy komentár nie je

potrebný sú metódy písané (pomenované) podľa konvencie (napr. metódy controllerov typu *new* alebo *create* v jazyku Ruby)

B.1.7 Odstránenie komentáru z dokumentácie

V prípade, že komentár nemá byť súčasťou dokumentácie – komentáre typu TODO a FIXME, ako aj poznámky autora slúžiace pre jeho vlastné účely, prípadne časti kódu, ktoré chce vývojár z určitého dôvodu zachovať, obsahuje tento komentár špeciálne označenie. Toto označenie pozostáva zo symbolov "**#--**" a "**#++**". Obe tieto skupiny znakov sa musia nachádzať na začiatku riadka, pričom ich poradie je taktiež zachované. "**#--**" sa musí v kóde nachádzať vyššie, ako "**#++**". Medzi dva takto označené riadky, je možné pridať ľubovoľný počet riadkov komentárov, pričom tieto, sú pri automatickej tvorbe dokumentácie ignorované.

Príklad:

```
# This method returns sum of two numbers.
#--
# This comment will not be in the documentation.
# And this one as well
#++
# Works only with numbers greater than 0
```

B.1.8 Komentovanie premenných

V prípade, že sa v zdrojovom kóde nachádza premenná, ktorej význam a obsah nie je z jej názvu jasný, sa komentár k takejto premennej nachádza v riadku, v ktorom bola premenná prvý krát uvedená, pričom komentár je vo formáte "**#** " a popis premennej. Pokiaľ by sa na danom riadku nachádzalo viacero premenných, ktoré je potrebné komentovať, pred popisom premennej je uvedený jej názov, a jednotlivé premenné s popismi sú od seba oddelené čiarkou.

B.1.9 Ostatné typy komentárov

V prípade potreby je nutné, aby autor použil špecifické typy komentárov:

- "TODO",

- “FIXME”.

Tieto komentáre sú vývojovým prostredím RubyMine implicitne podporované. Implicitná je taktiež podpora písania kľúčových slov bez rozlíšenia veľkých a malých písmen.

B.1.10 “TODO” komentáre

V prípade, že autor pri písaní kódu neimplementuje celú požadovanú respektíve potrebnú funkcionalitu, prípadne niektorý z autorov zistí túto skutočnosť, miesto v kóde, na ktoré treba funkcionalitu doplniť je potrebné označiť. Na označenie chýbajúcej časti použije komentár typu “TODO”. Komentár začína znakom “**#**”, za ktorým nasleduje **medzera** a kľúčové slovo “**TODO**”. Nasleduje krátky popis chýbajúcej funkcionality, ktorú je potrebné dorobiť.

Príklad:

```
# TODO: implement date check (starting_time<ending_time?)
```

B.1.11 “FIXME” komentáre

V prípade, že autor pri písaní kódu zistí, že program nefunguje dostatočne, prípadne niektorý z autorov zistí túto skutočnosť, miesto v kóde, na ktorom je potrebné vykonať zmeny je potrebné označiť. Na označenie chýbajúcej časti použije komentár typu “FIXME”. Komentár začína znakom “#”, za ktorým nasleduje medzera a kľúčové slovo “FIXME”. Nasleduje samotný stručný popis chyby prípadne udalosti, pri ktorej chyba nastala.

Príklad:

```
# FIXME: fails if starting_date is 29th October
```

B.1.12 Vygenerovanie dokumentácie

Dokumentácia sa generuje pomocou aplikácie RDoc.

Predpoklady na vygenerovanie dokumentácie

Na vygenerovanie dokumentácie je potrebný nainštalovaný RDoc gem. Tento je možné nainštalovať pomocou príkazu “gem install rdoc”, ktorý možno zadať do konzoly vo vývojovom prostredí RubyMine.

Vygenerovanie dokumentácie

Samotné vygenerovanie dokumentácie sa uskutočňuje pomocou príkazu “rdoc”. Tento v adresári projektu vygeneruje požadovanú dokumentáciu. Dokumentáciu je možné zobraziť vo webovom prehliadači otvorením súboru .\doc\index.html.

B.2 Evidencia úloh v nástroji JIRA

Autor: Tomáš Sýkora

Táto časť definuje prácu s úlohami v nástroji JIRA, aké typy úloh používame, ktorý typ a kedy použijeme a postup pri vytváraní úlohy. Pre všetky úlohy je potrebné byť prihlásený pomocou AIS loginu v nástroji JIRA a byť na domovskej stránke.

B.2.1 Roly a zodpovednosti účastníkov

Jednotlivé roly a zodpovednosti účastníkov sú uvedené v tabuľke 2.1.

Rola	Zodpovednosť
Vlastník produktu	Zástupca zákazníka Schválenie zadania úlohy Schválenie vypracovania úlohy Schválenie prezentácie úlohy
Scrum master	Sledovanie stavu úloh v šprinte Report stavu úloh vlastníkovi
Riešiteľ	Riešenie priradenej úlohy Odovzdanie vypracovanej úlohy na testovanie
Tester	Testovanie vypracovanej úlohy Potvrdenie správnosti úlohy Vytvorenie úlohy typu Bug, ak boli zistené chyby vo vypracovaní

Tabuľka 2.1 Roly a zodpovednosti účastníkov

B.2.2 Typy úloh

V našom projekte používame dva typy úloh a to:

- Task
- Bug

B.2.3 Pomenovanie úloh

Názov úlohy pozostáva z počiatočného tagu a zo stručného a výstižného názvu. Tagy sú uzavreté v hranatých zátvorkách. Používame nižšie uvedené tagy:

- [V] – View, úloha týkajúca sa zobrazenia na klientovi,
- [C] – Controller, úloha týkajúca sa serverovej časti produktu,
- [C/V] – Komplexná úloha, ktorá zahrňuje prácu aj na klientskej, aj na serverovej časti produktu,
- [B] – Bug, slúži na identifikáciu chyby v úlohe,
- [?] – Výskumná úloha, slúži na preskúmanie nástrojov alebo technológií, či sú vhodné na použitie vo vývoji produktu,
- [!] – Dôležitá úloha, napríklad odovzdanie časti produktu, dokumentácie.

Potom sú pomocou týchto tagov pomenované konkrétne úlohy. Príklady pomenovania úloh:

- Pre view, controller, komplexnú, výskumnú a dôležitú úlohu je formát názvu rovnaký – "[Tag] (stručný názov úlohy)"
- Pre bug sa názov líši – "[B] Fix bug in (názov pôvodnej úlohy)".

B.2.4 Vytvorenie úlohy typu Task

Tento typ používame vtedy, ak sme sa dohodli na jednotlivých úlohách, ktoré chceme robiť v ďalšom šprinte. Vytvorené úlohy idú do backlogu. Pre vytvorenie postupujeme nasledovne:

1. Kliknúť na tlačidlo Create na hornej lište,
   Alternatívne: Stlačiť klávesu c,
2. Zvoliť projekt CatchMe,
3. Issue type je v tomto prípade Task,
4. Do políčka Summary vpísať stručný a výstižný názov úlohy podľa návodu uvedeného v 2.3 Pomenovanie úloh,
5. Prioritu nezadávame,
6. Due date primárne nezadávať, podľa potreby vložiť,
7. Assignee nastaviť na Unassigned, ak je nastavené inak
8. Do poľa Description vpísať podrobný popis úlohy a akceptačné kritériá,
9. Časové odhady nevypĺňať,
10. Pri políčku Attachment kliknúť na Choose Files a načítať relevantný dokument, obrázok,
11. Labels nevypĺňať,

12. Do políčka Units vložiť odhad, ktorý bol odhadnutý na stretnutí členmi tímu,
13. Položky PercentDone, DueTime, Epic Link a Sprint nevypĺňať,
14. Kliknúť na tlačidlo Create.
    Alternatívne: Zrušiť kliknutím na Cancel.

B.2.5 Vytvorenie úlohy typu Bug

Používame vtedy, keď tester našiel pri testovaní chybu v úlohe. Pre vytvorenie takéhoto typu úlohy je postup nasledovný:

1. Kliknúť na tlačidlo Create na hornej lište,
   Alternatívne: Stlačiť klávesu c,
2. Zvoliť projekt CatchMe,
3. Issue type je v tomto prípade Bug,
4. Do políčka Summary vpísať stručný a výstižný názov úlohy podľa návodu uvedeného v 2.3 Pomenovanie úloh pre typ úlohy Bug,
5. Prioritu nezadávame,
6. Due date primárne nezadávať, podľa potreby vložiť,
7. Assignee nastaviť na riešiteľa pôvodnej úlohy,
8. Do poľa Description vpísať podrobný popis chyby a akceptačné kritériá,
9. Časové odhady nevypĺňať,
10. Pri políčku Attachment kliknúť na Choose Files a načítať relevantný dokument, obrázok,
11. Položky Labels, Units, PercentDone, DueTime, Epic Link a Sprint nevypĺňať,
12. Kliknúť na tlačidlo Create.
    Alternatívne: Zrušiť kliknutím na Cancel.

B.2.6 Zmena názvu úlohy

Zmena názvu úlohy môže byť vhodná pri neúplnom, nepresnom zadaní úlohy alebo aj pri preklepe.

1. Kliknúť na Agile > Scrum board,

2. V zobrazení šprintu zvoliť požadovanú úlohu (Zobrazenie znázornené po kliknutí klávesy 2)

   Alternatívne: V zobrazení backlogu zvoliť požadovanú úlohu (Zobrazenie znázornené po kliknutí klávesy 1)
3. Na pravom boku sa zobrazí panel, ktorý obsahuje atribúty úlohy, tu kliknúť na názov úlohy,

   Alternatívne: Stlačiť klávesu e > kliknúť do políčka za položku Summary,
4. Upraviť existujúci názov na požadovaný názov,
5. Stlačiť klávesu Enter pre potvrdenie zmeny názvu.

   Alternatívne: Stlačiť klávesu Esc pre zrušenie zmeny názvu.

B.2.7 Zmena popisu alebo akceptačných kritérií úlohy

Postup pre zmeny v popise alebo v akceptačných kritériách úlohy sa použije vtedy, keď nebol dostatočne popísaný problém, alebo došlo k (dohodnutým)zmenám v akceptačných kritériách.

1. Kliknúť na Agile > Scrum board,
2. V zobrazení šprintu zvoliť požadovanú úlohu (Zobrazenie znázornené po kliknutí klávesy 2)

   Alternatívne: V zobrazení backlogu zvoliť požadovanú úlohu (Zobrazenie znázornené po kliknutí klávesy 1)
3. Na pravom boku sa zobrazí panel, ktorý obsahuje atribúty úlohy, tu kliknúť do poľa pod Description,

   Alternatívne: Stlačiť klávesu e > kliknúť do poľa pod Description,
4. Upraviť na požadovaný obsah,
5. Stlačiť klávesu Enter pre potvrdenie zmien.

   Alternatívne: Stlačiť klávesu Esc pre zrušenie zmien.

B.2.8 Priradenie úlohy riešiteľovi / zmena riešiteľa

Riešiteľ je ten člen tímu, ktorí rieši konkrétnu úlohu. Pre priradenie úlohy postupujeme nasledovne:

1. Kliknúť na Agile > Scrum board,
2. V zobrazení šprintu zvoliť požadovanú úlohu (Zobrazenie znázornené po kliknutí klávesy 2)

   Alternatívne: V zobrazení backlogu zvoliť požadovanú úlohu (Zobrazenie znázornené po kliknutí klávesy 1)
3. Na pravom boku sa zobrazí panel, ktorý obsahuje atribúty úlohy, tu kliknúť na ikonu troch bodiek v pravom hornom rohu panelu > Assign,
4. Za položkou Assignee vybrať riešiteľa,

   Alternatívne: Kliknúť na Assign to me,
5. Potvrdiť kliknutím na Assign.

   Alternatívne: Zrušiť kliknutím na Cancel.

Tento postup je funkčný aj pre zmenu riešiteľa.

B.2.9 Pridanie komentáru k úlohe

Ak bola úloha vyriešená, tak sa pridá komentár do akého bodu je úloha vyriešená a čaká na revíziu kódu (Metodika revízie kódu od kolegu Bc. Lukáša Masára). Po revízií kódu sa tiež pridá komentár o možnej existujúcej chybe kde sa tiež pridá odkaz do položky Issue Links na úlohu riešiacu túto chybu. Pridávanie komentáru sa spraví nasledovne:

1. Kliknúť na Agile > Scrum board,
2. V zobrazení šprintu zvoliť požadovanú úlohu (Zobrazenie znázornené po kliknutí klávesy 2)

   Alternatívne: V zobrazení backlogu zvoliť požadovanú úlohu (Zobrazenie znázornené po kliknutí klávesy 1)
3. Na pravom boku sa zobrazí panel, ktorý obsahuje atribúty úlohy, tu kliknúť na tlačidlo Comment,

   Alternatívne: Stlačiť klávesu m,
4. Vložiť text komentáru,
5. Potvrdiť kliknutím na Add.

   Alternatívne: Potvrdiť stlačením klávesy Enter.

   Alternatívne: Zrušiť stlačením klávesy Esc.

B.3 Metodika testovania

Autor: Michal Cihák

Táto časť sa zaoberá procesmi vytvárania testov a priebehom testovania.

Projekt tohto tímu je vyvíjaný v jazyku *Ruby* pod rámcom *Ruby on Rails.* Na základe toho boli zvolené tieto testovacie nástroje:

- RSpec - Testovací nástroj pre jazyk ruby, ktorý zabezpečí lepšiu čitateľnosť pre človeka a ľahké porozumenie toho čo daný test testuje. Preto bol uprednostnený tento nástroj oproti Test::Unit, ktorý je síce súčasťou Ruby, no nie je tak prehľadný.
  DOC: http://relishapp.com/rspec/rspec-rails/v/3-1/doc
- Capybara - Tento nástroj pomáha pri testovaní simulovať interakciu skutočného používateľa s webovou aplikáciou.
  DOC: http://www.rubydoc.info/github/jnicklas/capybara/master
- Selenium - Tento nástroj je potrebný na testovanie časti aplikácie, ktorá prebieha u klienta, čiže všetky zmeny, ktoré nastanú pomocou JavaScriptu, či CSS.
  DOC: http://www.rubydoc.info/github/jnicklas/capybara/master/Capybara/Selenium
- Shoulda - Uľahčuje testovanie nad modelmi, vzťahmi medzi nimi a validáciami.
  DOC: http://thoughtbot.github.io/shoulda-matchers/v2.7.0/

B.3.1 Vytvorenie nového testu

Tento proces všeobecne opisuje vytvorenie testu akéhokoľvek typu. Hlavné typy, ktoré využíva tento projekt budú popísané ďalej.

1. Vytvorenie súboru v /spec/TYPE/FOLDER/NAME_spec.rb
   - TYPE - aký typ testu vytvárame
   - FOLDER - v prípade feature testu vytvárame aj priečinok, inak nie.
   - NAME - názov testu sa líši vždy podľa typu testu (upresnené neskôr)

   Príklad cesty: /spec/features/events/edit_event_spec.rb

2. Vloženie require 'rails_helper' do prvého riadku.

B.3.2 Vytvorenie nového testu typu Feature

Tento typ testu sa vytvára, keď je potrebné vytvoriť vysokoúrovňový test napodobňujúci používateľa, ktorý prechádza celou aplikáciou. Takýto test zabezpečuje, že jednotlivé časti a komponenty aplikácie správne spolu fungujú.

Tento proces použijeme vždy, keď doplníme do aplikácie funkcionalitu, ktorá je uskutočniteľná z pohľadu používateľa a prepája všetky vrstvy aplikácie (MVC).

1. Vytvorenie súboru podľa B.3.1. v /spec/features/FOLDER/METHOD_spec.rb
   - FOLDER - zodpovedá tomu, nad ktorým modelom (množné číslo) sa tento test vykonáva, resp. ku ktorému controlleru, ktorý spracováva daný model, test patrí
   - METHOD - konkrétna metóda v controller
2. Pri vytváraní testu sa používajú primárne funkcie, ktoré poskytuje knižnica Capybara
3. Do describe sa vpisuje vždy v angličtine aká funkcionalita sa testuje

   Príklad: describe 'Creating events'
4. Pri vytváraní testov sa často opakujú časti kódu - tomuto je potrebné sa vyhnúť a vytvárať pomocné metódy
5. Do it sa píše v angličtine zrozumiteľným spôsobom čo sa stane keď test prejde

   Príklad: it 'displays an error when new event name has less than 3 characters'
6. Pri vypĺňaní foriem a vytváraní objektov sa používajú názvy relevantné k danému objektu a prislúchajúcich atribútov. Tzn. nepoužívajú sa názvy typu 'Lorem Ipsum', náhodné reťazce znakov, ani žiadne názvy neinformujúce o tom, k čomu prísluchajú. Výnimkou sú názvy keď testujeme napríklad validáciu a je potrebné použiť špecifické názvy.
   Príklad: Event.create(name: "My event name", description: "My long event description")
   Príklad: Event.update(name: "My new event name", description: "My new long event description")

Príklad feature testu:

```
describe 'Editing events' do
  let!(:event) { Event.create(name: "My name", description: “My long description.")}
  def update_event(options={})
    event = options[:event]
    event.reload
    visit "/events/#{event.id}"
    click_link "Edit"
    fill_in "Name", with: options[:name]
    fill_in "Description", with: options[:description]
    click_button "Update Event"
    event.reload
  end
  it 'should updates the event information with correct info' do
    update_event event: event, name: "New name",
                 description: "New long description"
    expect(page).to have_content("Event was successfully updated")
    expect(event.name).to eq("New name")
    expect(event.description).to eq("New long description")
  end
  it 'displays error when the name is too short' do
    update_event event: event, name: "xy"
    name = event.name
    event.reload
    expect(event.name).to eq(name)
    expect(page).to have_content("error")
  end
end
```

B.3.3 Vytvorenie nového testu typu Feature s použitím Selenium

Testovanie pomocou Selenium sa používa keď nenastáva AJAXová komunikácia alebo hocijaká komunikácia medzi vrstvami a daná akcia sa deje len u klienta (Views v MVC).

Tento proces použijeme vždy keď sa doplní Javascriptová funkcionalita, ktorá sa má aktivovať na základe používateľovej interakcie.

1. Vytvorenie súboru podľa B.3.1. v /spec/features/FOLDER/METHOD_spec.rb
   - FOLDER - v tomto prípade zodpovedá Javascriptovému súboru, nad ktorým sa test vykonáva
   - METHOD - konkrétna metóda v súbore
2. Pre písanie testov platia rovnaké pravidlá ako v B.3.2.
3. Používa sa tento kód na aktivovanie Selenium ovládača:

```
before(:all) do
    Capybara.current_driver = :selenium
End
```

4. Na deaktiváciu:

```
after(:all) do
    Capybara.use_default_driver
end
```

Príklad feature testu s použitím Selenium:

```
describe “Event actions" do
  before(:all) do
    Capybara.current_driver = :selenium
  end
  it "pops up a confirm dialog when we click delete" do
    page.click_link("Delete")
    dialog = page.driver.browser.switch_to.alert
    dialog.text.should == “Are you sure?"
    dialog.dismiss
  end
  after(:all) do
    Capybara.use_default_driver
  end
end
```

B.3.4 Vytvorenie nového testu typu Model

Tento typ testu sa používa, keď je potrebné spraviť nízkoúrovňové testy nad modelmi. Ako napríklad testovanie asociačných vzťahov medzi modelmi, testovanie validácií nad modelmi alebo či modely skutočne reprezentujú údaje v databáze.

Tento proces použijeme vždy keď sa vykoná zmena v modeli, alebo medzi modelmi. Tzn. vytvoria sa nové asociačné vzťahy, nové validačné pravidlá, nove atribúty modelu alebo sa čokovek z tohto zmení.

1. Vytvorenie súboru podľa 2.1. v /spec/models/MODEL_spec.rb
   - MODEL - nad ktorým modelom tento test vykonávame
2. Pri vytváraní testu sa používa primárne knižnica shoulda-matchers

Príklad model testu:

```ruby
describe Participant do
  it { should have_many(:time_votes) }
  it { should have_many(:meeting_times).through(:time_votes) }
  it { should accept_nested_attributes_for(:time_votes) }
 end
```

B.3.5 Manuálna kontrola testov

Testy je potrebné skontrolovať po každej zmene v kóde, ktorá zasahuje do nejakej funkcionality, ktoré testy kontrolujú. Takisto po vytváraní testu alebo po jeho zmene je potrebné aby tento test prešiel. Testuje sa viacerými spôsobmi:

Spustenie jediného testu:

- kliknutie pravým tlačidlom na daný test -> 'Run Spec' (v RubyMine)
- alebo cez terminal rspec /spec/features/edit_event_spec.rb

Spustenie skupiny testov určitého typu:

- rake spec:models

Spustenie všetkých testov:

- rake spec

V prípade, že niektorý test neprejde je potrebné vykonať niektoré z nasledujúcich:

- upraviť kód tak aby bol test splňený a test prešiel
- ak test už viac nie je relevantný a nie je potrebné aby bol viac splnený -> upraviť test alebo ho úplne zmazať
- ak si to aby test prešiel žiada väčšie zmeny alebo nie je jasné či je stále test vhodný -> vytvoriť úlohu (Metodika evidencie úloh v nástroji JIRA), alebo odkomunikovať s ostatnými členmi tímu (Metodika komunikácie)

B.4 Metodika – Code Review

Autor: Lukáš Masár

Účelom prehliadky kódu je zachytiť chyby v kóde pred tým, ako sa dostanú k používateľovi. Je to základná činnosť vykonávaná za účelom zabezpečenia kvality kódu. Jej zanedbanie môže viesť k zbytočne zvýšeným nákladom v ďalších etapách projektu.

Pri prehliadke kódu sa kontroluje:

- **vizuálna kvalita kódu** – formátovanie, komentovanie kódu
- **logická kvalita kódu** – či kód zabezpečuje funckionalitu, ktorú má implementovať

B.4.1 Súhrn rol a zodpovednosti účastníkov

Rola	Zoznam zodpovednosti
Programátor	• napísanie kódu, ktorý sa bude kontrolovať • sprístupnenie kódu určeného na prehliadku cez internet • úprava kontrolovaného kódu, podľa pripomienok, ktoré vzniknu pri prehliadke kódu
Inšpektor	• prehliadka kódu • spätná väzba pre programátora • nájdenie chýb v kóde • návrhy na úpravy kódu • kontrola funkčnosti • kontrola formátu

B.4.2 Kedy sa vykonáva prehliadka kódu

Keď je kód pripravený na testovanie a nasadenie do produkčného prostredia, čiže keď je otestovaný a funkčný na lokálnom stroji programátora, ktorý ho napísal. Programátor vtedy pushne svoj kód na development branchu a sprístupní tak kód cez internet (viď dokument "Verziovanie – Git a jeho integrácia v Rubymine").

B.4.3 Kto vykonáva prehliadku kódu

Iný člen tímu, ako ten ktorý kód napísal. Prehliadku každého kódu vykonávajú minimálne dvaja členovia tímu, a to buď spoločne alebo samostatne. Člen tímu, ktorý vykonal prehliadku kódu túto aktivitu zaznamená v JIRE v issue, ktorá sa daného kódu týka (viď. dokument „Metodika evidencie úloh v nástroji JIRA") .

B.4.4 Náhľad metodiky

Postup krokov:

- programátor/i napíše kód implementujúci konkrétnu issue
- programátor sprístupní kód cez internet
- inšpektor/i vykoná prehliadku kódu
- zostaví sa zoznam pripomienok, problémov pre danú prehliadku
- programátor vykoná prípadné úpravy kódu pre opravu nájdených chýb, resp. vyhovenie pripomienkam inšpektora
- kontrolovaný kód je schválený a pripravený pre testovanie a nasadenie v produkčnom prostredí

B.4.5 Postup metodiky

A) Napísať kód a požiadať o jeho prehliadku
   - programátor/i si vyberie issue, ktorú chce implementovať
   - programátor implementuje a lokálne otestuje danú issue

- keď je kód lokálne funkčný, programátor kód sprístupní cez internet na development vetve vzdialeného repozitára (viď dokument "Verziovanie – Git a jeho integrácia v Rubymine")
- programátor požiada o prehliadku kódu pomocou komentáru v danej issue v JIRE, alebo pomocou správy v HipChat-e

B) Prehliadka kódu a spätná väzba

- inšpektor/i prechádza kód prístupný cez internet riadok po riadku a pridáva doňho komentáre, krátke a zrozumitelné, ako spätnú väzbu
- komentáre, obsahujúce pripomienky na úpravu kódu musia obsahovať na začiatku značkou "TODO"
- "TODO" pripomienky, ktoré majú byť spracované pred schválením kódu, musia byť spísané do zoznamu taskov, ktorý je poskytnutý programátorovi
- inšpektor kontroluje funkčnosť, teda či vlastnosť, ktorú kód implementuje, alebo chyba, ktorú fixuje, funguje
- inšpektor kontroluje, či je kód riadne zdokumentovaný a okomentovaný
- inšpektor kotnroluje, či kód dodržiava formátovanie

C) Opraviť objavené chyby a dokončiť prehliadku

- programátor/i, ktorý kód napísal, upraví kód podľa pripomienok, ktoré vznikli pri prehliadke kódu a opraví všetky chyby, ktoré sa našli
- upravení kód sprístupní cez internet na development vetve vzdialeného repozitára a požiada o opätovnú prehliadku
- cyklus sa opakuje, kým kód nie je schválený inšpektormi (minimálne dvomi) bez ďalších pripomienok
- keď je kód schválený, môže byť testovaný pre produkčné prostredie

B.4.6 Ďoplňujúce informácie a požiadavky

- prehliadka kódu sa musí vykonať pred tým ako sa kód testuje v produkčnom prostredí
- o prehliadku žiada ten/tí, kto kód napísal
- všetci členovia tímu by sa minimálne raz mali zúčastniť na prehliadke nejakého kódu
- kód, ktorý je predkladaný na prehliadku musí byť riadne zdokumentovaný a okomentovaný
- kód ktorý je predkladaný na prehliadku, musí dodržiavať formátovanie v súlade so zaužívanými programovacími konvenciami
- ak programátor žiada o prehliadku kódu, ktorý opravuje nejakú chybu, treba podložiť výsledky testov, ako dôkaz, že je chyba opravená
- programátor musí poskytnúť krátky zmysluplný opis toho, čo kód implementuje
- odporúča sa robiť prehliadku najviac na 400 riadkoch kódu a dlhšie kódy rozdeliť na viac prehliadok

B.5 Metodika verziovania v Git a RubyMine

Autor: Matúš Pikuliak

Časť dokumentu pojednáva o procesoch pri verziovaní kódu a súborov všeobecne pri vyvíjaní implementácie nášho projektu. Procesy slúžia programátorom pri písaní kódu a jeho synchronizácii v rámci tímu.

Náš tím používa ako verziovací systém Git. Máme zriadený vzdialený repozitár, kde si tím udržiava kópie projektu. Náš projekt vyvíjame v jazyku Ruby na frameworku Ruby on Rails. Keďže náš tím používa IDE Rubymine, jednotlivé kroky sú opísané práve pre tento program.

Prvá časť tohto dokumentu, Manažment verziovania Git v Rubymine, pojednáva o využívaní nástroja Rubymine pri práci s repozitármi. Druhá časť, Správa vetiev, opisuje náš model vetvenia a procesy, ktoré súvisia s vytváraním, tagovaním a celkovo prácou s vetvami počas vývoja nášho projektu.

B.5.1 Vytvorenie nového lokálneho projektu

Tento proces sa vykonáva, keď chce vývojár vytvoriť zo vzdialeného repozitára lokálny a začať na ňom pracovať. Predpokladá sa, že má prístup do nášho repozitára na Bitbucket. V opačnom prípade kontaktujte vedúceho tímu.

1. VCS > Checkout from Version Control > Git
2. Do prvého poľa skopírovať .git odkaz zo stránky projektu na bitbuckete, napr: https://mpikuliak@bitbucket.org/ltursky/catchandmatch.git
3. Nastaviť adresáre do ktorých sa stiahnu súbory z repozitára.
4. Potvrdiť voľby tlačidlom Clone.
5. Zadať overovacie údaje.
6. Súbory sa stiahnu do lokálneho repozitára.
7. V prípade nutnosti inštalácie nových gemov Rubymine upozorňuje na túto skutočnosť. Kliknutím na ponúkanú popup správy a gemy sa nainštalujú.
   Alternatívne: Tools > Bundler > Install a kliknúť na Install.
8. Nastaviť súbor *config/database.yml* podľa vzoru na bitbuckete

9. Vytvoriť databázu:

   Ctrl+Alt+R > db:create

   Ctrl+Alt+R > db:migrate

Po tomto postupe by mal byť obsah vzdialeného repozitára skopírovaný do žiadaného priečinku a projekt by mal byť naviazaný na tento repozitár. Malo by byť takisto možné spustiť rails server.

B.5.2 Aktualizácia pracovnej kópie súborov

Tento proces sa vykonáva, keď sa zo vzdialeného repozitára sťahuje do lokálneho aktuálna verzia súborov. Toto sa robí na začiatku každej programovacej *session*, ako aj pred odosielaním lokálnej verzie späť na server.

1. Vybrať správnu vetvu, viď. *Zmena pracovnej vetvy.*
2. VCS > Update Project

   Alternatívne: Ctrl + T

3. Vybrať *Update type* Merge.
4. Kliknúť na OK.
5. Zadať overovacie údaje.
6. V prípade rozporov vykonať v ponúnkanom rozhraní merge nad súbormi, pri ktorých nastali konflikty.
7. V prípade nutnosti inštalácie nových gemov Rubymine upozoňuje na túto skutočnosť. Kliknutím na ponúkanú popup správu sa gemy nainštalujú

   Alternatívne: Tools > Bundler > Install a kliknúť na Install.

8. Do databázy natiahnite nové migrácie:

   Ctrl+Alt+R > db:migrate

Pracovná kópia by mala byť aktualizovaná a je možné spustiť rails server.

B.5.3 Commitovanie

Tento proces sa vykonáva, keď sa vytvára lokálna snímka stavu súborov. Táto sa robí, keď sa ukončí práca na jednej malej logickej časti systému.

V jednom commite sa nachádzajú len súbory týkajúce sa zmeny jednej malej logickej časti systému. Do commitu sa neprikladajú drobné zmeny z častí, ktoré s commitom priamo nesúvisia. Týmto drobným zmenám sa vytvára samostatný commit. Do jedného commitu však možno zaradiť viacero podobných drobných zmien, ktoré priamo nesúvisia s chovaním systému, napr. zmeny v syntaxi. Necommitujú sa ani konfiguračné súbory (napr. datbase.yml), ktoré by mohli narušiť prácu ostatným developerom.

O správu súborov, ktoré budú commitované sa automaticky stará Rubymine. Vytváranie, mazanie, presúvanie, premenovanie či modifikácia súborov sú v prostredí Rubymine automaticky zaznamenávané. Zmeny možno sledovať v záložke Changes.

Postup samotného commitovania je nasledovný:

1. VCS > Commit changes / Ctrl + K
2. V strome súborov vyberať, ktoré zo zmien sa commitnú.
3. Vyplniť commit správu. Jej formát je opísaný nižšie.
4. Ostatné položky nechať nezaškrtnuté
5. Kliknúť Commit.

Formát commit správ je nasledovný:

```
[<Issue ID>] <Issue Name>

<subor 1> - <zmena v subore 1>
<subor 2> - <zmena v subore 2>
…
<subor N> - <zmena v subore N>
```

Issue ID a Issue Name sú jedinečný identifikátor a meno issue z projektového issue trackera v systéme Jira (napr. [CATCHME-23] Improve Dates). Každý commit sa týka práve jednej issue. Drobný refactoring, ku ktorému sa nebude vyrábať samostatná issue v Jire označíme tagom [R].

Nasleduje opis zmien v jednotlivých súboroch. Opis zmien sa píše v anglickom jazyku v minulom čase (napr. something was changed, something was implemented). Opis zmien by mal stručne, ale úplne informovať o zmenách v daných súboroch a ich motivácii.

Príklad commit správy:

```
[CATCHME-26] [C] Link Generation

app/models/event.rb - Implemented token generator in Event model. This
token is now used to generate URLs.
app/controllers/events_controller.rb - Controllers refactored to find
event objects by tokens and not by ids.
```

B.5.4 Aktualizácia vetvy vzdialeného repozitára

Tento proces sa vykonáva, keď sa zasiela lokálna verziu súborov na server. Súbory sa aktualizujú na konci každej programátorskej *session*. Aktualizuje sa aj keď sa ukončí práca na jednej väčšej časti systému a presúva sa na inú.

Pred aktualizáciou sa overujú tieto náležitosti:

1. Je vybraná správna vetva, ktorá sa bude aktualizovať, viď. *Zmena pracovne vetvy.*
2. Sú stiahnuté aktuálne súbory z danej vetvy, viď. *Aktualizácia pracovnej kópie súborov.*
3. Súbory tak ako sa budú pushovať prejdú všetkými aktívnymi automatickými testami

Samotná aktualizácia prebieha nasledovne:

1. VCS > Git > Push
   Alternatívne: Ctrl + Shift + K
2. Po prezretí zmien v commitoch kliknúť Push.
3. Pri probléme so synchornizáciou súborov sa celá aktualizácia ruší a súbory sa aktualizujú zo vzdialeného repozitára, viď. *Aktualizácia pracovnej kópie súborov.*
4. Pri vetvách, nad ktorými funguje Continuous Integration (development a production) overiť či aktualizácia nespôsobila chybu. V prípade že áno postupovať podľa príslušných procesov.

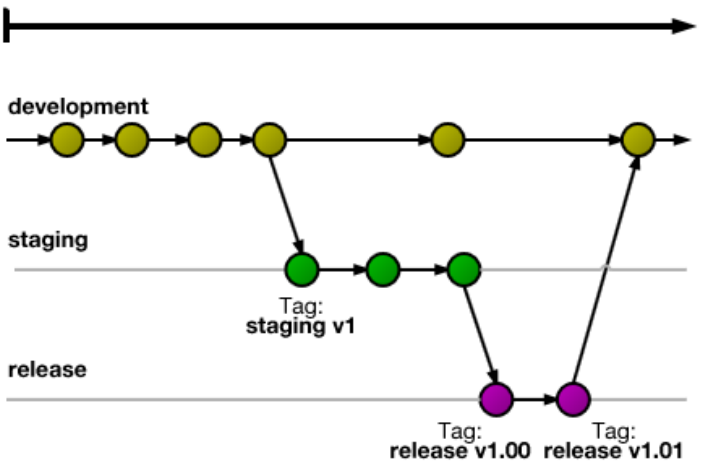
B.5.5 Správa vetiev

Pri vývoji projektu sa používajú tri základné vetvy:

- *development* - vetva, na ktorej prebieha každodenný vývoj projektu. Z tejto vetvy ťahajú a do tejto vetvy odovzdávajú kód vývojari pri bežnej práci. Nad touto vetvou sa vykonáva Continuous Integration a je prístupná na internete.

- *staging* – vetva, ktorá sa obnovuje vždy pre vydaním nového produkčného releasu. Táto vetva slúži ako miesto kde sa aktuálna developerská vetva upravuje do prezentovateľného výsledku v dohodnutej funkcionalite. Úlohou práce na tejto vetve je úprava kódu pre potreby releasu a mazanie nadbytočnej funkcionality.

- *release* – vetva, v ktorej sa nachádzajú funkčné verzie produktu vhodné pre produkciu. Nad touto vetvou sa vykonáva Continuous Integration a je prístupná na internete. Nad touto vetvou sa vykonáva minimum práce. Verzia nasadená na tejto vetve je prezentovaná ako aktuálny stav nášho produktu. Pokiaľ sa nedohodne inak, číslo každého releasu je o jedna väčšie ako číslo predchádzajúceho a začína sa od verzie 1.



Obr. B.5.1 – Model vetvenia pri vytváraní releasu

Okrem týchto troch základných vetiev sa v projekte nachádzajú aj tzv. *features* vetvy, v ktorých sa vyvíjajú nové časti systému, či v ktorých sa skúšajú nové technológie. Tieto vetvy sú

od *developmentu* oddelené vtedy, keď hrozí že by plánované zmeny v kóde mohli narušiť jeho štruktúru a preto sa radšej vytvorí nová vetva, na ktorej sa paralelne vyvíja.

B.5.6 Vytváranie releasu

Tento proces sa uskutočňuje vždy, keď na tímovom stretnutí padne rozhodnutie vytvoriť nový release pre potreby dema či nasadenia produktu. Na mítingu sa špecifikuje funkcionalita daného releasu, aj vo vzťahu k potencionálnym *features* vetvám. Na tomto mítingu je vybraný člen tímu, ktorý zodpovedá za manažment verziovania a vetvenia počas vytvárania tohto releasu. Tento má na starosti aj kroky opisovaná v tomto procese.

1. Vetva *development* sa mergne do vetvy *staging.*
2. Tento mergovací commit sa otaguje číslom príslušnej verzie: `staging v<číslo>`

   napr.: `staging v1`

Nad týmto commitom začnú následne pracovať všetci developeri, ktorí upravujú kód tak, aby vyhovoval špecifikácii určenej na stretnutí. Poverený človek sleduje priebeh týhto zmien a je informovaný o stave vetvy. Napokon poverený človek overí u ostatných členov tímu, či ukončili prácu na vetve a či je v stave vhodnom pre release.

3. Vetva *staging* sa mergne do vetvy *release.* Do vetvy staging sa ďalej neprispieva.
4. Tento mergovací commit sa otaguje číslom príslušnej verzie: `release v<číslo>`

   napr.: `release v1.00`

   V správe k danému tagu treba v odrážkach spísať hlavné zmeny vo funkcionalite oproti predošlému releasu.

Nad týmto commitom ešte ďalej prebieha krátky čas vývoj, ktorý má za účel odchytiť drobné chyby. Poverený človek má v tomto prípade povinnosť tagovať nové commity pokračujúcimi číslami verzie, napr.: `release v1.01`. Do správ týchto tagov uvádza zoznam zmien z používateľského hľadiska.

Po ukončení vývoja na pre aktuálny release poverený človek mergne t´túto vetvu späť do development, aby sa uchovali prípadne opravy bugov a podobne. Rozhodnutia o pokračovaní či ukončení vývoja na release vetve sa dejú na tímovom mítingu.

B.5.7 Features vetvy

Features vetvy vytvárajú developeri keď chcú v kóde urobiť zmeny, ktoré by mohli narušiť jeho štruktúru či funkčnosť, keď chcú vyskúšať nový prístup či technológiu. Pri vytváraní feature vetvy vývojári nasledujú tento proces:

1. Vytvoriť novú vetvu s menom `feature-<názov feature>`, napr. feature-angular alebo feature-user_roles
2. Prvému commitu nastaviť tag s menom opisujúcim názov danej feature. V popise tagu uviesť podrobný opis plánovaných zmien a dôvod prečo bolo potrebné vytvoriť novú vetvu.

Vývoj na feature vetve prebieha podobne ako na development s tým rozdielom, že tieto vetvy nemajú nad sebou spustený proces Continuous Integration. Ktokoľvek môže modifikovať a upravovť kód, či ho ďalej vetviť podľa uváženia. Podľa potreby možno takisto feature vetvu mergovať spať do *development* vetvy.

B.5.8 Zmena pracovnej vetvy

Tento a nasledovné opísané podprocesy slúžia len ako pomôcka pri vykonávaní procesov opísaných vyššie, týkajúcich sa práce s vetvami pri vývoji projektu.

Názov aktuálnej vetvy sa zobrazuje vpravo dole v stavovom riadku v okne Git. Okrem názvu sa tu po kliknutí dá dostať aj k ďalším vetvám z projektu. Pracovná vetva sa mení nasledovne:

1. Kliknúť na Git okno v stavovom riadku
2. Kliknúť na názov požadovanej vetvy.
3. Zvoliť Checkout.
4. V prípade problémov so synchronizáciou zvoliť možnost Smart checkout a mergnúť súbory

B.5.9 Vytvorenie novej vetvy

1. Kliknúť na Git okno v stavovom riadku.
2. Zvoliť možnosť New Branch.
3. Zadať názov branche.

4. Kliknúť OK

B.5.10 Mergovanie vetiev

1. Prepnúť sa do vetvy, do ktorej chceme mergovať
2. Kliknúť na Git okno v stavovom riadku
3. Kliknúť na vetvu, ktorú chceme do aktuálnej mergovať
4. Kliknuť Merge

B.5.11 Tagovanie commitov

1. Okienko Changes v dolnom panely > Log
2. Kliknúť na vybraný commit a okopírovať jeho hash.
3. VCS > Git > Tag files
4. Do okienka commit nakopírujte hash commitu a prípadne validujte.
5. Do okienka Tag name uveďte samotný tag.
6. Do okienka Message uveďte prípadne ďalšie informácie o commite.
7. Kliknite OK.

B.6 Metodika komunikácie

Autor: Pavol Michálek

B.6.1 Kompetencie členov tímu

Pomocou neskôr vymedzených komunikačných nástrojov sa členovia tímu CatsCanCode môžu okrem tímových stretnutí prednostne obrátiť s dotazmi na členov tímu:

- Ing. Lukáš Turský - *email: xtursky@gmail.com, tel.č.: 0918219725*
  vlastník produktu, smerovanie vývoja,
- Michal Cihák - *mxcihak@gmail.com, tel.č.: 0918966744*
  manažér testovania, jadro aplikácie,
- Michal Gajdoš - *michalgajdo@gmail.com, tel.č.: 091892237*
  manažér rizík
- Lukáš Masár - *lukas.masar@gmail.com, tel.č.: 0904111729*
  aplikačný server, nasadzovanie nových verzií
- Pavol Michálek - *pali.michalek@gmail.com, tel.č.: 0902682727*
  tímová stránka, front-end aplikácie
- Vladimír Osvald - *osvald.v@gmail.com*
  dokumentácia,
- Matúš Pikuliak - *matus.pikuliak@gmail.com*
  front-end aplikácie
- Tomáš Sýkora - *sykora.tomino@gmail.com, tel.č.: 0905384406*
  zápisnice

B.6.2 Interná tímová komunikácia - nástroj HipChat

V prípade komunikácie v tíme a s vlastníkom produktu sa používa chatovací nástroj HipChat. Je nakonfigurovaný na konkrétne potreby tímu CatsCanCode. Každý člen tímu má v pracovnej dobe tento komunikačný nástroj zapnutý a v čo najkratšom čase reaguje na nové správy, ktoré sa ho týkajú. Je teda potrebné mať zapnuté notifikácie o nových správach. Preddefinované je nastavenie zasielania notifikačných emailov v čase, keď je člen offline.

Internetová komunikácia členov tímu sa musí v čo najväčšej miere sústrediť v nástroji HipChat. Výnimkou je

- nástroj **Google Hangouts** používaný na zdieľanie obrazovky a internetové (video)hovory,
- nástroj **Jira** používaný na komunikáciu (komentovanie) spojenú s konkrétnymi úlohami ("task") a chybami ("bug").

B.6.3 Miestnosti

Na základnej obrazovke (Lobby) je zoznam miestností. Boli vytvorené na špecifické prípady komunikácie:

- *Team Page* - slúži na informovanie ohľadom nových skutočností spojených s tvorbou, prevádzkou a aktualizáciou tímovej stránky (umiestnenej na adrese http://team12-14.ucebne.fiit.stuba.sk/). Miestnosť má nastavenú integráciu na webový repozitár BitBucket s projektom tejto stránky a je v nej možné sledovať vývoj stránky každým commitom.
- *Rails* - slúži na informovanie ohľadom vývoja aplikácie CatchMe, jej vývojového prostredia, aplikačného servera, … . Miestnosť má nastavenú integráciu na webový repozitár BitBucket s projektom tejto aplikácie a je v nej možné sledovať vývoj projektu každým commitom.
- *Jira* - slúži na sledovanie aktivity ostatných členov tímu v nástroji Jira. Miestnosť má nastavenú integráciu na "Activity Stream" nástroja Jira.
- *CatsCanCode* - sústreďuje každodennú komunikáciu v rámci tímu. Zahŕňa oznamy o najbližších stretnutiach, organizácii predmetu, pripomienky blížiacich sa termínov, … .

Každý člen tímu má rovnaké oprávnenia v týchto chatovacích miestnostiach. Každý člen má možnosť sa **stručne a vecne vyjadriť** v príslušných miestnostiach a informovať tak ostatných členov o nových skutočnostiach. Nesmú zasielať správy nesúvisiace s témami miestností.

Adresovanie správ v miestnostiach

Pri komunikácií v spoločných miestnostiach je vyžadované špecifikovať na začiatku správy adresáta. Robí sa tak **znakom "@" nasledovaným celým menom adresáta** (mená sa automaticky ponúkajú po napísaní znaku @). Je možné napísať aj viacerým adresátov. Pokiaľ je správa smerovaná na každého člena tímu, používa sa reťazec **"@all"**.

HipChat umožňuje **zasielanie správ len jednému členovi**. Uprednostňujte tento spôsob v prípade, že sa jedná o informácie, ktoré by zaťažovali ostatných členov, svojím rozsahom zneprehľadnili komunikáciu v miestnostiach alebo sa jedná o riešenie konkrétnej úlohy, na ktorej títo dvaja spolupracujú.

B.6.4 Špeciálna interná tímová komunikácia - nástroj Google Hangouts

V prípade, že sa jedná o rozsiahlejšiu komunikáciu viacerých členov tímu, nie je možné osobné stretnutie a nepostačuje textová forma HipChat-u, používa sa komunikačný nástroj Google Hangouts. Používa sa na hovory, videohovory a zdieľanie obrazovky.

B.6.5 Webové sídlo tímu

Slúži na informovanie verejnosti o priebehu projektu.

Zápisnica stretnutia je dokončená najneskôr ďalší deň po stretnutí a je uverejnená na webovom sídle tímového projektu. Uverejňuje sa aj **aktualita** (prípadne dve) z posledného obdobia. Na stretnutí sa určia členovia poverení dokončením dokumentu a uverejnením dokumentu a aktuality na webových stránkach tímu. Na proces dohliadajú členovia s pôvodnými kompetenciami (Tomáš Sýkora - dokumentácia projektu, Pavol Michálek - webové sidlo tímu).

Projekt webového sídla je zdieľaný webový repozitárom BitBucket a zmeny sa nasadzujú na projektový server, ku ktorému majú prístup všetci členovia tímu.

B.6.6 Komunikácia o úlohách

V nástroji Jira, ktorý zaznamenáva priebeh vývoja projektu, prebieha komunikácia v komentároch jednotlivých úloh. Je dôležité, aby čo najjemnejšie zaznamenávali prácu na týchto úlohách, ktorá sa týka integrácie, náväzností, spolupráce viacerých členov a finalizácie práce.

Na sprehľadnenie dlhších komentárov je odporúčané použiť možnosti formátovania textu (krátku dokumentáciu je možné nájsť pod textovým poľom nového komentáru po kliknutí na *"?"*).

Adresovanie komentáru sa používa len pri vymenuvávaní konkrétnych členov tímu, ak je potrebné určiť kompetentného člena (a nestačí pridelenie - *Assignee*). Nemení sa prednastavená viditeľnosť komentáru (*Viewable by All Users*).

B.6.7 Zdieľanie súborov - Google Drive

Všetky súbory súvisiace s prácou na tímovom projekte s výnimkou zdrojových kódov sa sústredia na webovom úložisku Google Drive. Všetci členovia majú rovnaké oprávnenia na zdieľaný spoločný priečinok projektu na tomto úložisku.

Samostatné sú podpriečinky s materiálmi z pravidelných tímových stretnutí. Nachádza sa v nich zápisnica a fotografická dokumentácia zo stretnutia.

Pre ďalšie potreby samostatných tém, je možné vytvárať podpriečinok zaradený v koreňovom priečinku alebo na príslušnom mieste v strome priečinkov. Podpriečinky majú svoj názov vo formáte *XX Názov podpriečinku*, pričom XX je dvojciferné inkrementujúce poradové číslo.

B.7 Metodika dokumentovania inžinierskeho diela

Autor: Vladimír Osvald

Tento dokument obsahuje pracovné procesy a postupy potrebné pri písaní dokumentácie k inžinierskemu dielu pomocou nástrojov Wiki a Microsoft Word. Obsahuje pokyny pre editáciu a finalizáciu dokumentu.

Tento dokument je určený všetkým členom tímu, ktorí sa podieľajú na programovaní inžinierskeho diela a jeho následnom dokumentovaní.

B.7.1 Identifikácia rolí

Na tvorbe dokumentácie sa podieľajú dve používateľské role.

Dokumentátor - každý člen tímu, ktorý sa podieľa na tvorbe dokumentácie k inžinierskemu dielu v nástroji Wiki.

Dokončovateľ - určený člen tímu, ktorý vyexportuje dokumentáciu z Wiki do nástoja Word, kde celý dokument naformátuje do finálnej verzie.

B.7.2 Dostupnosť dokumentácie

Dokumentácia k inžinierskemu dielu je dostupná na stránkach Bitbucket.org vo vytvorenom projekte ltursky / CatchAndMatch v sekcii Wiki. Názov súboru je „Dokumentácia k inžinierskemu dielu“.

Výsledný naformátovaný dokument sa nachádza na gDrive v zložke Dokumentácie. Názov súboru je „Dokumentácia k inžinierskemu dielu“.

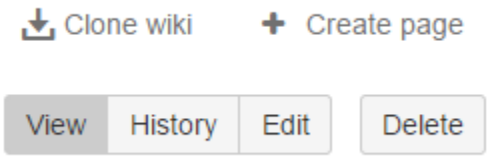
B.7.3 Dokumentovanie

Súbor s dokumentáciou na Wiki (umiestnenie súboru v kapitole 4) obsahuje predpripravenú základnú štruktúru.

B.7.4 Proces editovania

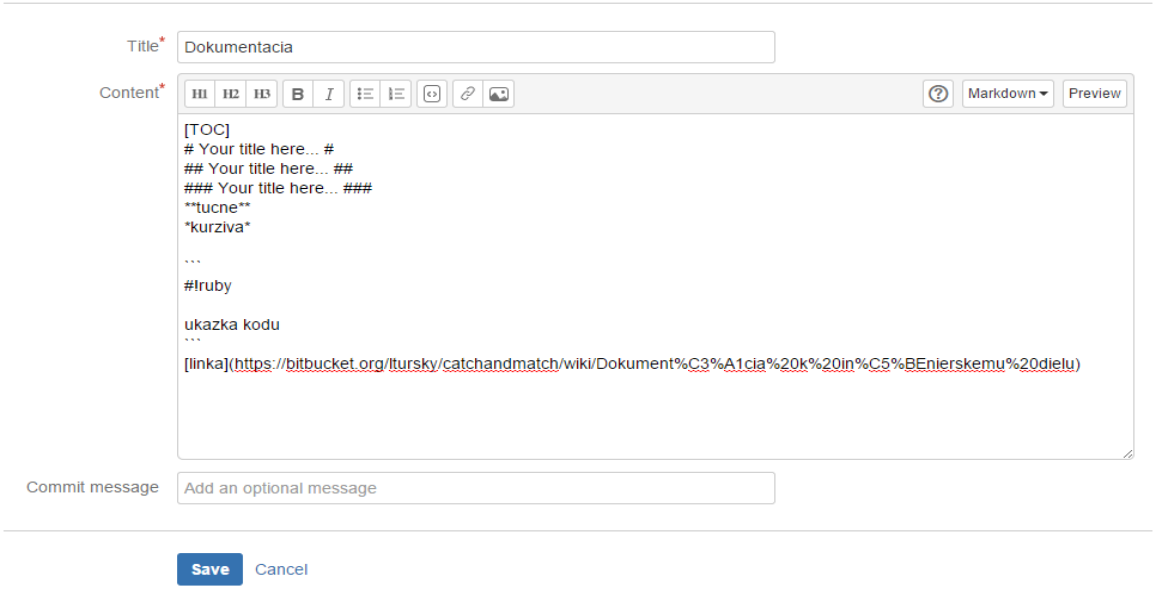
Dokumentátori editujú tento súbor vždy, keď je potrebné doplniť aktuálne informácie do súboru. Súbor s dokumentáciou k inžinierskemu dielu obsahuje prevažne technickú dokumentáciu. Dokumentovanie prebieha bezodkladne po ukončení programovania funkcie alebo dostatočne malého modulu. Dostatočne malým modulom sa rozumie úsek programu, ktorý na seba priveľmi nadväzuje a nie je možné ho rozdeliť na menšie celky.

Po vyhľadní súboru a jeho otvorení sa na obrazovke zobrazí hlavná stránka. V pravom hornom rohu sa náchádzajú následovné ovládacie prvky:



Obr 1. - Hlavná ponuka wiki

Edit – umožní prístup k editovaniu dokumentu. Po zvolení tejto možnosti sa zobrazí nasledujúce okno, v ktorom editujeme dokument:



Obr 2. Editačné okno

Editačné okno je rozdelené do štyroch horizontálnych sekcií a to Title, Content, Commit message a Save/Cancel tlačidlá.

V procese Editovania sa nezasahuje do sekcia Title, ktorá je prednastavená. Dokumentovanie prebieha v sekcii Content podľa pravidiel uvedených nižšie.

Po Ukončení editovania je potrebné vyplniť sekciu Commit message (viď obr 2.). V tejto sekcii sa stručne ale výstižne opíšu vykonané zmeny na dokumente.

Pravidlá editácie:

1. Dodržiava sa syntax určený pre Wiki
2. Používajú sa nadpisy troch úrovní
    - Prvou úrovňou označujeme hlavné kapitoly
- Používame následovnú syntax : # nadpis prvej úrovne #
    - Druhou úrovňou označujeme podkapitoly
- Používame následovnú syntax : ## nadpis druhej úrovne ##
    - Treťou úrovňou označujeme podrobnejšie sekcie podkapitol
- Používame následovnú syntax : ### nadpis tretej úrovne ###

3. Zvýrazňujú sa dôležité pojmy a vety
    - Zvýrazňujeme pomocou tučného a šikmého písma
    - Syntax pre tučné písmo : ** text **
    - Syntax pre šikmé písmo : * text *
4. Vložené obrázky sa vždy popisujú a číslujú
5. Pri dokumentovaní funkcií sa vkladajú ukážky kódu
    - Syntax na vloženie kódu : ``` - začiatok bloku s kódom
- #!ruby - definícia jazyka
- Ukážka kódu - vložený kód
- ``` - koniec bloku s kódom

6. Pri vkladaní linkov používame nasledovný syntax: [linka](url adresa)

7. Nevymazáva sa tag [TOC] na začiatku dokumentu predstavujúci samogenerujúci obsah

Postup editovania dokumentácie:

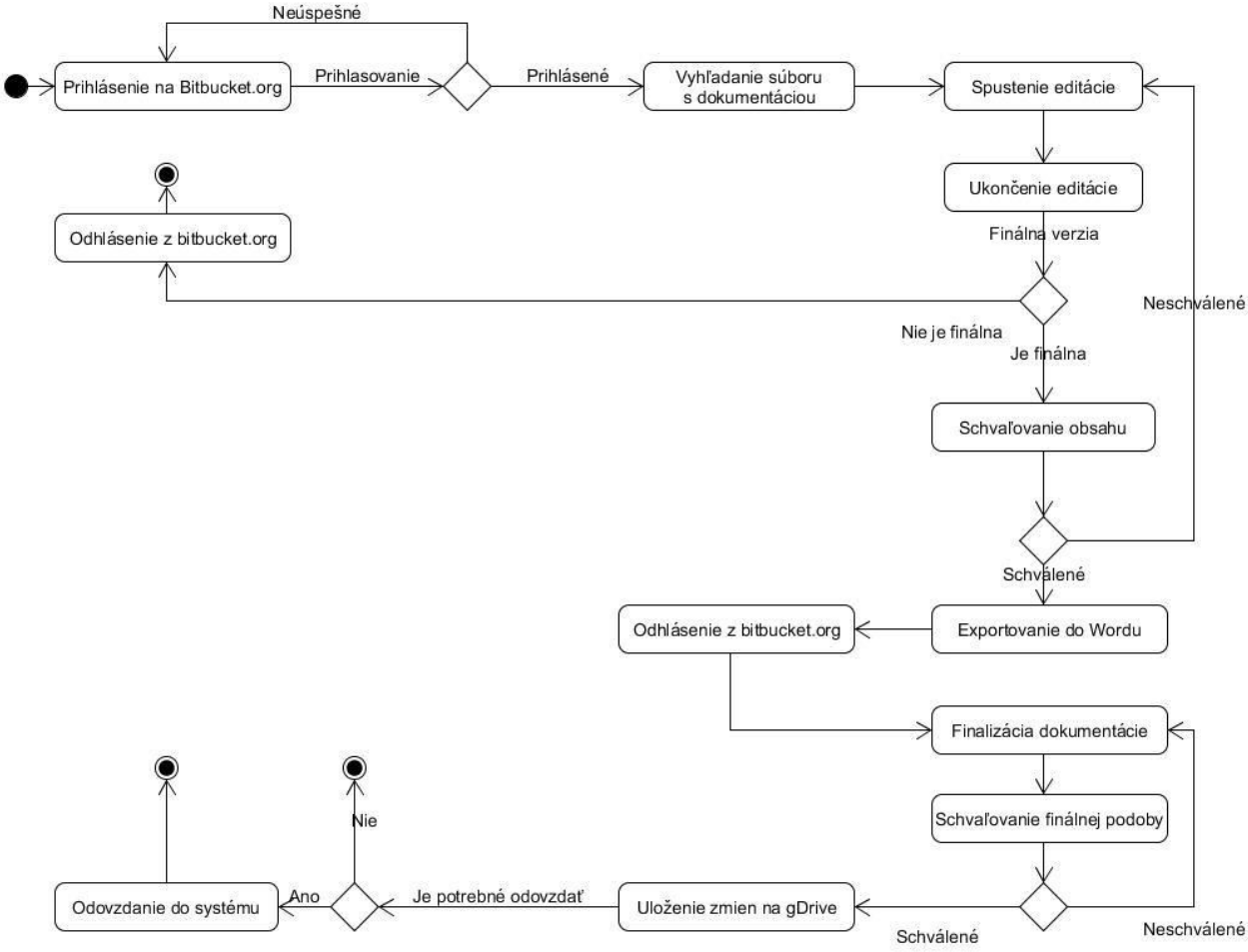
1. Prihlásenie do služby Bitbucket
2. Vyhľadanie projektu CatchAndMatch
3. Vyhľadanie súboru s dokumentáciou k inžinierskemu dielu
4. Zvolenie možnosti “Edit”
5. Vykonanie zmien
6. Okomentovanie zmien
7. Uloženie zmien
8. Kontrola, či boli zmey vykonané
9. Odlhásenie zo služby Bitbucket

B.7.5 Proces finalizácie

Postup finalizácie je nasledovný:

1. Tím schváli obsah dokumentu na wiki
2. Finalizátor príslušný .md súbor s dokumentáciou prekonvertuje na formát .docx
3. Otvorí sa dokumentácia vo Worde
4. Dokumentátor podľa pravidiel uvedených v sekcii 5.4 naformátuje dokument
5. Po formátovaní sa tímom schváli výsledná podoba
6. Dokument sa uloží na gDrive
7. V prípade potreby sa dokument odovzdá do akademického system

1.1 Diagram procesu dokumentovania



Obr. 3 - Diagram procesu dokumentovania

B.7.6 Pravidlá formátovania výsledného dokumentu

1. Dodržiava sa stanovený obsah titulnej stránky
    - V hlavičke názov fakulty, centrované na stred
    - V strede titulnej strany názov projektu a dokumentu
    - V ľavej dolnej časti mená členov tímu, meno vedúceho, akademický rok - centrované naľavo
2. Prvý odstavec sa neodsadzuje, ostatné o tabulátor
3. Používajú sa nadpisy rôznych úrovní totožné s wiki
4. Nadpisy zarovnávame naľavo

- Prvá úroveň - font Calibri, 16, bold
- Druhá úroveň - font Calibri, 14, bold
- Tretia úroveň - font Calibri, 12, bold

5. Text zarovnávame naľavo
    - Font Arial, 12
6. Obrázky zarovnávame na stred
7. K obrázkom sa pridáva popis v tvare „Obr. # - názov“
    - # označuje poradové číslo obrázka v dokumente
    - názov označuje názov obrázku
    - centrované na stred, kurzíva
8. Tabuľky zarovnávame na stred
9. K tabuľkám sa pridáva popis v tvare „Tab. # - názov“
    - # označuje poradové číslo tabuľky v dokumente
    - názov označuje názov tabuľky centrované na stred, kurzíva

C Zápisnice zo stretnutí

V predloženej prílohe C sa nachádzajú zápisnice z jednotlivých tímových stretnutí.

C.1 Zápisnica stretnutia č. 1

Autor Bc. Tomáš Sýkora

Dátum a čas stretnutia 30.09.2014, 13:00 -16:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.1.1 Osnova na stretnutie

- Určiť názov tímu.
- Na základe názvu navrhnúť logo a plagát.
- Špecifikovať požiadavky.

C.1.2 Priebeh stretnutia

- Navrhovali sme základný flow.
- Rozberali sme názov produktu. Všetci sme sa zhodli na tom, že názov bude obsahovať slovo "catch", kedže sa snažíme "pochytať" ľudí na stretnutie. Vznikli dve varianty:
    - Catch me

    - Catch and Match
- Na základe názvu tímu sme sa dohadovali na logu tímu. Najbližšie k tejto tematike nám prišla analógia mačka a myš. Zhodli sme sa na mačacej labke a počítačovej myši.
- Ohľadom plagátu sme vyberali layout. Vybrali sme si layout hollywoodskeho plagátu.
- Ďalej sme navrhovali dizajn produktu, definovali sme si základné požiadavky:
    - Minimalistický dizajn, easy-to-use, quick
    - V hlavnej časti stránky budú výrazné štyri kachlíky:
        - What?
        - When?
        - Where?
        - With?
    - Pre skupiny ľudí
    - Podpora dočasných účtov (generovať každému používateľovi ID, ktorý sa prihlási a napr generovať do 10 000 a potom zmazať), neskôr na registráciu (mail, integrácia so sociálnymi sieťami)
    - Notifikácie:
        - mailové
        - pomocou integracií (???)
        - neskôr pravdepodobne push cez mobilnú appku
    - Módy:
        - neformálne stretnutia (priatelia, kolegovia, spolužiaci, atď.)
        - formálne (projekt vedúci-študent, atď.)
    - Znovupoužitie - pomocou vygenerovaného linku na udalosť, po odohraní udalosti sa udalosť zmaže
    - Vizualizácia stavu (Teplomer, atď.)

C.2 Zápisnica stretnutia č. 2

Autor Bc. Tomáš Sýkora

Dátum a čas stretnutia 07.10.2014, 13:00 - 17:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.2.1 Osnova na stretnutie

- Určiť skupiny ľudí, pre koho bude produkt.
- Definovať vlastnosti produktu.
- Prečo vlastne tento produkt vzniká?
- Určiť technológie, ktoré budú použíté na vývoj.

C.2.2 Priebeh stretnutia

- Určili sme skupiny ľudí, pre ktorích bude, resp. nebude tento produkt:
    - + Vek 15-33
    - + Uponáhľaní
    - + Pohodlní
    - + Komunikujúci cez internet
    - - Firmy
    - - Profesionálne eventy / konferencie
- Ujasnili sme si, prečo vlastne tento produkt vzniká:
    - Uľahčenie dohadovania

        - Na jeden klik
        - Rýchle
    - Matcher
- Definovali sme vlastnosti produktu:
    - Jednoduchý
    - Intuitívny
    - Catch and Match
    - Maching
        - Akceptačné kritériá
            - Automatické
            - Manuálne
        - Priority
    - Účty
        - Dočasné
        - Cez integrácie
    - Eventy
        - Súkromné
        - Verejné
    - Opakovaná návšteva eventu (Revisit), možno z viacerých zariadení?
        - Overenie cez pin alebo nejakú frázu
        - unikátny link na užívateľa
    - Real-time?
    - Vizualizácia stavu
    - Notifikácie
    - Módy
    - Integrácie
    - Mobilná aplikácia
- Určili sme si cieľ produktu:
    - Uľahčiť a zjednodušiť dohadovanie stretnutí na par klikov.
- Rozoberali sme technológie na vývoj:
    - Ruby on Rails - pre naše potreby pokrýva všetko

- Databázový server PostgreSQL

C.3 Zápisnica stretnutia č. 3

Autor Bc. Pavol Michálek

Dátum a čas stretnutia 14.10.2014, 13:00 - 16:30

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.3.1 Osnova na stretnutie

- Nájsť vhodné komunikačné nástroje, version control nástroje, scrum nástroje.
- Určiť "Issues" na prvý šprint.

C.3.2 Priebeh stretnutia

- Našli sme viacero komunikačných, version control a scrum nástrojov ale nakoniec sme sa zhodli na:
    - komunikačný - HipChat
    - version control - BitBucket
    - scrum - Jira
- Nástroje sme hneď aj spojaznili.
- Mišo Cihák každého navigoval, ako si naklonovať repozitár CatchAndMatch do RubyMine.

- Určovali sme "Issues" na 1. šprint - prešli sme každý issue a dopisovali popis. Všetky issues sú v angličine:
    - Impelement event flow for first prototype
    - Design event structure
    - Design link generation
    - Implement link generation
    - Design simple user identification
    - Implement user interface
    - Implement user identification
    - Design user interface
    - Create team page
    - Setup Ruby on Rails project
    - Setup server for team page
    - Setup database
    - Design data model
    - Create project structure
    - Setup private git repository
- estimácia jednotlivých issue

C.4 Zápisnica stretnutia č. 4

Autor Bc. Michal Gajdoš

Dátum a čas stretnutia 21.10.2014, 13:00 - 17:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.4.1 Osnova na stretnutie

- Zistiť ako dokážeme vyriešiť dizajn tak, aby produkt spĺňal základnú podmienku jednoduchosti použitia
- Prejsť navrhnutý základný objektový model
- Koncept architektúry
- DB maintenance
- Technická dokumentácia
- Prebrať usporiadanie stretnutia s potenciálnymi používateľmi (early adopters) pre získanie feedbacku a nápadov na design a používanie CatchAndMatch služby
- Testovanie aplikácie

C.4.2 Priebeh stretnutia

- prebrali a pozmenili sme navrhnutý dátový model, pričom sme sa zhodli, že jeho použitie bude spojené s prototypom, a následne bude upravený

- opätovne sme otvorili otázku UI, pričom sme sa zamerali hlavne na layout stránky - “vertikálny” vs. “kartičky”
- zhodnotili sme doterajší priebeh šprintu

C.5 Zápisnica stretnutia č. 5

Autor Bc. Michal Cihák

Dátum a čas stretnutia 28.10.2014, 13:00 - 17:30

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.5.1 Osnova na stretnutie

- predviesť demo product owner-ovi
- retrospektíva - zhodnotenie prvého šprintu
- defininovať "Issues" na ďaľší šprint
- rozdeliť procesy na MSI/MIS (zadanie 2)

C.5.2 Priebeh stretnutia

- na základe retrospektívy sme odhalili problémy, ktoré vznikli počas prveho šprintu (viď dokument "Retrospektíva")
- dodefinovali sme požiadavky na prvý prototyp
    - Linky/Routing (C)
    - Voting -> List (V,C)
    - Link user to EventOptions (C)
    - "Copy to Clipboard" button (V)

    - Add duration -> Hide DateTo (C,V)
    - Improve (C,V)
        - dates
        - places
        - users - highlight, user selection
- taktiež sme prebrali nadchádzajúce úlohy
    - Event Place/Time representation in DB
    - investigate CI (Jenkins, Bamboo)
    - 17.11. - document
    - Landing page
    - Dev and Prod Enviroment
    - Organize UX session
    - How to finish Event flow?
    - How to represent temporary/logged user in DB

Legenda:

C - controller

V - view

C.6 Zápisnica stretnutia č. 6

Autor Bc. Michal Gajdoš

Dátum a čas stretnutia 4.11.2014 13:00 - 17:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.6.1 Osnova na stretnutie

- user votes
- front-end javascript framework
- CI

C.6.2 Priebeh stretnutia

- prebehla diskusia k prideleným procesom/metodikám
    - testovanie - Gabo
    - issue flow - Lukáš
    - VCS - Matúš
    - komunikácia - Pali
    - technická dokumentácia - Vlado
    - issue creation - Tomáš
- k uvedeným procesom budú vytvorené tasky v Jire

- prebehol daily stand-up meeting
- upravili sme šprint doplnením issues
- dohodli sme sa na definovaní niektorých procesov do konca týždňa - komunikácia, VCS, technická dokumentácia, stav projektu, issue flow

C.7 Zápisnica stretnutia č. 7

Autor Bc. Vladimír Osvald

Dátum a čas stretnutia 11.11.2014, 13:00 - 17:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.7.1 Osnova na stretnutie

- prebrať "investigate tasks" z predchádzajúceho šprintu
- zadefinovať tasky na ďalší šprint
- demo, planning a retrospective meeting
- existujúce riešenie - DOODLE
- "Panely pre tímové projekty 2014-15"

C.7.2 Priebeh stretnutia

- aj napriek predchádzajúcej dohode, bolo z časových dôvodov prezentovane demo až na tomto utorkovom stretnutí
- okrem prezentácie dema, boli prezentované výsledky z "investigate taskov" ako aj vypracovane metodiky k procesom
- následne prebehla retrospektíva, ktorej výsledkom je kladné zhodnotenie práce a pozitívne dojmy z uplynulého šprintu

- zhodli sme sa, že nasledujúci šprint bude trvať len jeden týždeň
- čo sa týka produktu, dohodli sme sa, že pri výbere času je podstatný začiatok udalosti - ostatné informácie (koniec/trvanie) nie sú natoľko podstatne, resp. ich zatiaľ nebudeme špeciálne uvažovať a riešiť

C.8 Zápisnica stretnutia č. 8

Autor Bc. Matúš Pikuliak

Dátum a čas stretnutia 18.11.2014,13:00 - 18:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.8.1 Osnova na stretnutie

- odovzdávaná dokumentácia - kto čo urobil
- angular
- dôležitosť rolí pri eventoch
- long-term plan
- finalize event flow
- retrospektíva

C.8.2 Priebeh stretnutia

- prebrali sme vytvorené metodiky riadenia
- zamerali sme sa na finalizáciu dokumentácie, ktorú je potrebne odovzdať
- nakoľko išlo o týždňový šprint, prebehla retrospektíva, na základe ktorej sme sa rozhodli že aj ďalší šprint bude týždňový, aj napriek tomu, že nie všetky tasky z minulého šprintu boli dokončené

- aj napriek tomu, že zobrazenie stránky nefunguje správne, rozhodli sme sa neopravovať vzniknuté chyby, keďže samoty front-end sa bude ešte meniť
- základná myšlienka a požiadavka na produkt je, aby bol ASAP (as stupid as possible), to znamená, že musí byť v základnej verzii veľmi jednoduchý na ovládanie

C.9 Zápisnica stretnutia č. 9

Autor Bc. Michal Gajdoš

Dátum a čas stretnutia 25.11.2014, 13:00 - 16:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.9.1 Osnova na stretnutie

- demo
- ďalší šprint
- retrospektíva
- prebrať/predniesť modul “rezervácia termínov”

C.9.2 Priebeh stretnutia

- oboznámili sme sa so stavom CI, pričom budeme používať nástroj na to určený (CruiseControl.cb)
- prebehlo demo produktu, v ktorom sme product ownerovi predviedli pokrok a nove funkcionality produktu, ako aj nový layout stránky s časovou osou
- prebehli rozsiahle diskusie o možnostiach autentifikácie používateľov, ako aj o možnostiach a spôsobe ukončenia eventu
- predstavili sme nový potenciál v oblasti využitia stránky - nový modul rezervácie miesta

v podnikoch (miesto pri stole v reštaurácií)

- prebrali a zadefinovali sme si tasky do tohto šprintu, ktorý sme predĺžili na dva týždne

C.10 Zápisnica stretnutia č. 10

Autor Bc. Michal Gajdoš

Dátum a čas stretnutia 2.12.2014, 13:00 - 16:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.10.1 Osnova na stretnutie

- demo
- ďalší šprint
- retrospektíva
- zhodnotiť prezentáciu investorovi, ktorá prebehla v pondelok

C.10.2 Priebeh stretnutia

- odprezentovali sme pokrok na úlohách definovaných v Jire, avšak k odprezentovaniu dema nedošlo
- keďže išlo o náročný týždeň, a tasky po tomto šprinte (ktorý trval len 1 týždeň) neboli splnené v dostačujúcej miere, ako aj v z iných organizačných pohnútok spojených priamo s predmetom *Tímový projekt* sme sa po dohode s vlastníkom produktu (a naším vedúcim) rozhodli predĺžiť aktuálny šprint o jeden týždeň

- na základe predchádzajúceho bodu sme doplnili Jiru (tento šprint) o ďalšie úlohy

C.11 Zápisnica stretnutia č. 11

Autor Bc. Tomáš Sýkora

Dátum a čas stretnutia 9.12.2014, 13:00 - 16:00

Miesto stretnutia JOBSOVO softvérové štúdio (1.31a)

Prítomní

- Ing. Lukáš Turský
- Bc. Michal Cihák
- Bc. Michal Gajdoš
- Bc. Lukáš Masár
- Bc. Pavol Michálek
- Bc. Vladimír Osvald
- Bc. Matúš Pikuliak
- Bc. Tomáš Sýkora

C.11.1 Osnova na stretnutie

- demo po šprinte
- dohodnúť sa na postupe k dokončeniu dokumentácie
- upresniť smer, ktorým sa bude produkt uberať

C.11.2 Priebeh stretnutia

- zastavil sa u nás Alexander Vengrin a dal nám cenné rady k produktu a nasmeroval nás správnym smerom k finálnemu produktu
- pozreli sme videá, ako správne odprezentovať produkt aj bez fungujúcej verzie
- konzultovali sme dôležitosť timeliny
- spresnili sme základné vlastnosti, ktoré bude obsahovať verzia do konca decembra

D Exporty evidencie úloh

V tejto časti príloh sa nachádzajú exporty evidencie úloh k jednotlivým šprintom.

D.1 Prvý šprint

T	Key	Components	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	CATCHME-21		Create application form for TP Cup 2015	Vladimir Osvald	Lukas Tursky		Closed	Fixed	21/Oct/14	28/Oct/14	28/Oct
	CATCHME-20		Design data model	Matus Pikuliak	Michal Cihak		Closed	Fixed	14/Oct/14	28/Oct/14	
	CATCHME-19		Implement User Identification	Lukas Tursky	Michal Cihak		Closed	Fixed	14/Oct/14	28/Oct/14	
	CATCHME-18		Implement Link Generation	*Unassigned*	Michal Cihak		Closed	Fixed	14/Oct/14	28/Oct/14	
	CATCHME-17		Implement User Interface	Matus Pikuliak	Michal Cihak		Closed	Fixed	14/Oct/14	28/Oct/14	
	CATCHME-16		Implement event flow for first prototype	Michal Cihak	Michal Cihak		Closed	Fixed	14/Oct/14	28/Oct/14	
	CATCHME-15		Setup RoR Project	Michal Cihak	Matus Pikuliak		Closed	Fixed	09/Oct/14	25/Oct/14	
	CATCHME-14		Design simple user identification	*Unassigned*	Lukas Tursky		Closed	Fixed	08/Oct/14	28/Oct/14	
	CATCHME-13		Design Link generation	Tomas Sykora	Lukas Tursky		Closed	Fixed	08/Oct/14	28/Oct/14	
	CATCHME-12		Design User Interface	Matus Pikuliak	Lukas Tursky		Closed	Fixed	08/Oct/14	28/Oct/14	
	CATCHME-11		Setup DB	Lukas Masar	Lukas Tursky		Closed	Done	08/Oct/14	25/Oct/14	
	CATCHME-10		Design Event structure	*Unassigned*	Lukas Tursky		Closed	Fixed	08/Oct/14	28/Oct/14	
	CATCHME-8		Create Team page	Pavol Michalek	Lukas Tursky		Closed	Fixed	08/Oct/14	28/Oct/14	
	CATCHME-7		Create project structure	Michal Cihak	Lukas Tursky		Closed	Fixed	08/Oct/14	25/Oct/14	
	CATCHME-4		Setup server for team page	Lukas Masar	Lukas Tursky		Closed	Done	08/Oct/14	19/Oct/14	
	CATCHME-1		Setup private Git repository	Lukas Tursky	Lukas Tursky		Closed	Fixed	08/Oct/14	14/Oct/14	

Obr. D.1. Export evidencie úloh pre prvý šprint

D.2 Druhý šprint

T	Key	Components	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	CATCHME-40		[P] Propose version management process	Matus Pikuliak	Matus Pikuliak		Closed	Fixed	09/Nov/14	12/Nov/14	
	CATCHME-37		[?] Represantion of accounts/permissions/roles	Michal Gajdos	Michal Cihak		Closed	Done	28/Oct/14	12/Nov/14	
	CATCHME-36		[UX] Organize UX session	Lukas Tursky	Michal Cihak		Open	*Unresolved*	28/Oct/14	04/Nov/14	
	CATCHME-35		[S] Setup production environment	Lukas Masar	Michal Cihak		Closed	Incomplete	28/Oct/14	12/Nov/14	
	CATCHME-33		[!] Odovzdanie dokumentacie 17.11.	*Unassigned*	Michal Cihak		Open	*Unresolved*	28/Oct/14	17/Nov/14	17/N
	CATCHME-30		[V] Improve Places	Michal Gajdos	Michal Cihak		Closed	Done	28/Oct/14	12/Nov/14	
	CATCHME-29		[V] Improve Dates	Pavol Michalek	Michal Cihak		Closed	Done	28/Oct/14	12/Nov/14	
	CATCHME-28		[V] Copy to clipboard	Matus Pikuliak	Michal Cihak		Closed	Fixed	28/Oct/14	05/Nov/14	
	CATCHME-27		[V] Prepare few layout options	Michal Gajdos	Michal Cihak		Closed	Incomplete	28/Oct/14	11/Nov/14	
	CATCHME-26		[C] Link Generation	Matus Pikuliak	Michal Cihak		Closed	Fixed	28/Oct/14	10/Nov/14	
	CATCHME-25		[V] Voting	Pavol Michalek	Michal Cihak		Closed	Done	28/Oct/14	11/Nov/14	
	CATCHME-24		[C] Voting	Michal Cihak	Michal Cihak		Closed	Done	28/Oct/14	11/Nov/14	
	CATCHME-23		[V] User Selection	Pavol Michalek	Michal Cihak		Closed	Done	28/Oct/14	11/Nov/14	
	CATCHME-22		[C] Link User to Event Options	Michal Cihak	Michal Cihak		Closed	Done	28/Oct/14	11/Nov/14	
	CATCHME-2		Agree on documentation structure and tool	Vladimir Osvald	Lukas Tursky		Closed	Fixed	08/Oct/14	12/Nov/14	

Obr. D.2. Export evidencie úloh pre druhý šprint

D.3 Tretí šprint

T	Key	Components	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	D
	CATCHME-49		[Proces] Improve Version Management	Matus Pikuliak	Matus Pikuliak		Resolved	Fixed	14/Nov/14	16/Nov/14	
	CATCHME-48		[V] Fix list of event times	*Unassigned*	Michal Cihak		Open	*Unresolved*	11/Nov/14	11/Nov/14	
	CATCHME-47		[C/V] Add Duration	*Unassigned*	Michal Cihak		Open	*Unresolved*	11/Nov/14	11/Nov/14	
	CATCHME-46		[V] Create page	*Unassigned*	Michal Cihak		Open	*Unresolved*	11/Nov/14	11/Nov/14	
	CATCHME-45		[Proces] Code Documentation	Michal Gajdos	Michal Cihak		Resolved	Fixed	11/Nov/14	18/Nov/14	
	CATCHME-44		[Proces] Issue Flow	Tomas Sykora	Michal Cihak		Resolved	Fixed	11/Nov/14	16/Nov/14	
	CATCHME-43		[Proces] Communication	Pavol Michalek	Michal Cihak		Resolved	Done	11/Nov/14	18/Nov/14	
	CATCHME-42		[Proces] Code Review	Lukas Masar	Michal Cihak		Resolved	Done	11/Nov/14	17/Nov/14	
	CATCHME-41		[Proces] Testovanie	Michal Cihak	Michal Cihak		Resolved	Fixed	11/Nov/14	18/Nov/14	
	CATCHME-33		[!] Odovzdanie dokumentacie 17.11.	*Unassigned*	Michal Cihak		Open	*Unresolved*	28/Oct/14	17/Nov/14	1
	CATCHME-32		[S] Implement CI solution	Lukas Masar	Michal Cihak		Resolved	Done	28/Oct/14	17/Nov/14	
	CATCHME-3		[?] Investigate Angular	Matus Pikuliak	Lukas Tursky		Resolved	Fixed	08/Oct/14	17/Nov/14	

Obr. D.3. Export evidencie úloh pre tretí šprint

D.4 Štvrtý šprint

T	Key	Components	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	CATCHME-62		Dokumentacia riadenia a inzinierske dielo na Web z 1. kontrolneho bodu na webe	Lukas Tursky	Lukas Tursky		Closed	Fixed	29/Nov/14	09/Dec/14	
	CATCHME-59		(Think of) Matcher	Vladimir Osvald	Lukas Tursky		Closed	Fixed	27/Nov/14	04/Dec/14	
	CATCHME-58		Decide event lifecycle end options	Lukas Tursky	Lukas Tursky		Closed	Fixed	27/Nov/14	04/Dec/14	
	CATCHME-57		Possibility to update name and description of created event	Pavol Michalek	Lukas Tursky		Closed	Fixed	22/Nov/14	09/Dec/14	
	CATCHME-56		[V] Timeline improvement	Pavol Michalek	Pavol Michalek		Closed	Incomplete	20/Nov/14	10/Dec/14	
	CATCHME-55		[V/C] Refactoring	Matus Pikuliak	Michal Cihak		Closed	Fixed	19/Nov/14	04/Dec/14	
	CATCHME-54		[!] 1.12. Prezentácia	Tomas Sykora	Michal Cihak		Closed	Fixed	19/Nov/14	02/Dec/14	
	CATCHME-53		[!] 24.11. Business Model Canvas & Customer development	Michal Gajdos	Michal Cihak		Closed	Fixed	19/Nov/14	25/Nov/14	
	CATCHME-52		(Think of) Preparation for design session	Lukas Tursky	Lukas Tursky		Closed	Fixed	19/Nov/14	02/Dec/14	
	CATCHME-51		User registration support	Michal Cihak	Lukas Tursky		Closed	Incomplete	19/Nov/14	10/Dec/14	
	CATCHME-50		Add event states to app	Matus Pikuliak	Lukas Tursky		Closed	Fixed	19/Nov/14	04/Dec/14	
	CATCHME-48		[V] Fix showing of first element in list of event times	Pavol Michalek	Michal Cihak		Closed	Fixed	11/Nov/14	04/Dec/14	
	CATCHME-46		[V] Simplify creation page	Pavol Michalek	Michal Cihak		Closed	Done	11/Nov/14	23/Nov/14	
	CATCHME-32		[S] Implement CI solution	Lukas Masar	Michal Cihak		Closed	Done	28/Oct/14	02/Dec/14	

Obr. D.4. Export evidencie úloh pre tretí šprint

D.5 Piaty šprint

T	Key	Components	Summary	Assignee	Reporter	P	Status	Resolution	Created	Updated	Due
	CATCHME-70		Create User profile page	Michal Gajdos	Lukas Tursky		Closed	Incomplete	02/Dec/14	10/Dec/14	
	CATCHME-69		Real time data synchronization on client site	Matus Pikuliak	Lukas Tursky		Closed	Fixed	02/Dec/14	08/Dec/14	
	CATCHME-67		Deploy current functional version to Prod env	Lukas Masar	Lukas Tursky		Closed	Fixed	02/Dec/14	09/Dec/14	
	CATCHME-65		Change Voting to just Yes Selection	Matus Pikuliak	Lukas Tursky		Closed	Fixed	02/Dec/14	08/Dec/14	
	CATCHME-64		Everybody put photo as avatar in Jira	*Unassigned*	Lukas Tursky		Closed	Fixed	02/Dec/14	09/Dec/14	
	CATCHME-63		Introduce versioning for easier deployment of stable version on Prod env	Pavol Michalek	Lukas Tursky		Closed	Fixed	30/Nov/14	09/Dec/14	
	CATCHME-62		Dokumentacia riadenia a inzinierske dielo na Web z 1. kontrolneho bodu na webe	Lukas Tursky	Lukas Tursky		Closed	Fixed	29/Nov/14	09/Dec/14	
	CATCHME-57		Possibility to update name and description of created event	Pavol Michalek	Lukas Tursky		Closed	Fixed	22/Nov/14	09/Dec/14	
	CATCHME-56		[V] Timeline improvement	Pavol Michalek	Pavol Michalek		Closed	Incomplete	20/Nov/14	10/Dec/14	
	CATCHME-51		User registration support	Michal Cihak	Lukas Tursky		Closed	Incomplete	19/Nov/14	10/Dec/14	

Obr. D.5. Export evidencie úloh pre piaty šprint