Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Visitortrack

Dokumentácia k riadeniu projektu
Tím 14

Vedúci tímu: Ing. Peter Krátky
Členovia: Bc. Tomáš Gaššo, Bc. Peter Paška, Bc. Michal Vantuch, Bc. Jakub Ďaďo, Bc. Dávid Slezák, Bc. Dávid Spišák
Názov tímu: Tím 14
Školský rok: 2015/2016

Základné informácie o dokumente

Názov dokumentu:	VisitorTrack - Dokumentácia k riadeniu projektu
Verzia:	1.0
Stav:	~~Draft~~ / ~~Tímové pripomienkovanie~~ / 1. Odovzdanie
Autor:	Tím 14

Obsah

Zoznam tabuliek

Zoznam obrázkov

1 Úvod

1.1 Účel dokumentu

Dokument vznikol ako doklad manažérskych aktivít nášho tímu pre potreby predmetu Tímový projekt I. Dokument v prvých kapitolách informuje o manažérskych úlohách jednotlivých členov tím, predstavuje risklist so špecifickými rizikami nášho projektu a softvér pre správu úloh. Najväčšiu časť dokumentu tvoria informácie o realizovaných šprintoch, kde dokument popisuje, čo sme v ktorom šprinte riešili a s akým výsledkom, aké vznikli problémy a ako sme sa ich snažili riešiť.
Ďalej dokument ponúka vygenerované výstupy z používaného softvéru na správu úloh a prehľad metodík používaných tímom. Na konci semestra budú doplnené aj časti ako globálna retrospektíva za zimný semester a webové sídlo projektu.

1.2 Systém pre správu úloh

Systém pre správu úloh sme riešili na druhom stretnutí, kedy sme zvolili prvú možnú trojicou riešení a to Trello, Redmine a Jira.
So systémom Jira majú skúsenosti viacerí členovia nášho týmu z pracovného prostredia v rôznych IT firmách.Nastavenie workflow-u a pravidiel pre prácu so zadávaním úloh do systému má veľa funkcií a možností. Zhodli sme sa, že robustnosť tohto systému by pre náš projekt bola zbytočne priveľká, takže sme orientovali na výber zo zvyšných dvoch.
Druhým systémom na správu úloh, ktorému sme sa venovali bol Trello, zapáčila sa nám hlavne jeho jednoduchosť a príjemný vzhľad. Skúšali sme si aj vytvoriť základný backlog, no už v tomto prvom kroku nám chýbali možnosti ako napríklad filtrovanie jednotlivých úloh, logovanie času a podobne. Jedná sa o voľne šíriteľný systém, takže existuje aj mnoho pluginov, ktoré by možno nami zistené nedostatky odstránili. Tie sme už však neinštalovali ale pustili sme sa ešte do analýzy posledného z trojice.

Posledným z analyzovaných systémov pre správu úloh bol Redmine, ktorý nám škola poskytovala zadarmo na školskom serveri. Prihlásenie bolo jednoduché a nebola ani potrebná registrácia, keďže je systém prepojený aj s akademickým informačným systémom a na prihlásenie stačilo to isté meno a heslo. Na začiatku sme si vytvorili náš projekt. Pre zoznamovanie s prostredím sme ponechali defautné nastavenia. Poskytovalo nám množstvo funkcií, ktoré nám chýbali v systéme Trello a zároveň nebolo veľmi zložité na prácu (pridávanie úloh, filtrovanie, zapisovanie času, ...). Redmine v plnej miere splnil naše požiadavky, takže sme si ho zvolili za náš systém na správu úloh.
V priebehu prvého šprintu bol jeden z členov poverený za dôkladnejšiu analýzu systému spolu s vytvorením pravidiel jeho používania. V polovici už mal vytvorené základné pravidlá hlavne pre pridávanie úloh, prvotné prispôsobenie systému a presun dohodnutých úloh do kategórií a prvého šprintu. Z nadobudnutými poznatkami oboznámil aj zvyšok tímu, s ktorým

následne prešiel prvú verziu metodiky a spoločne sa dohodli a upravili navrhnuté pravidlá. Systém doteraz používame podľa týchto pravidiel, ktoré sa veľmi osvedčili, a keď sa vyskytne niečo neočakávané na čo sa pri tvorbe metodiky nemyslelo, operatívne sa to vyrieši a metodika sa následne aktualizuje podľa prijatých rozhodnutí a opatrení.

Viac o práci s nami využívaným systémom na správu úloh Redmine v prílohe v kapitole *7.3Redmine*.

2 Zodpovednosti členov tímu

Okrem manažérskych úloh jednotlivých členov tímu, ktoré sú popísané v nasledujúcej kapitole, je potrebné zastrešiť aj iné, nemanažérskych zodpovedností.

Prehľad zodpovedností členov tímu:

Písanie zápisníc z stretnutí – úloha, ktorej realizátor s mení pri každom stretnutí. Jedinou výnimkou je scrummaster tímu, ktorý zápisnicu nikdy nepíše.

Codereview – vzájomná kontrola implementovaného zdrojového kódu je riešená formou pullrequestov. V tíme nemáme stanovenú osobu, ktorá by bola zodpovedná za validáciu pullrequestov. Každý tvorca pullrequestu si sám určí tímového kolegu, ktorý mu kód skontroluje (pri výbere sa zvažuje, kto ma koľko času, kto bude ďalej s kódom pracovať, resp. kto už s kódom pracoval a pod.).

Písanie dokumentácie – každý člen tímu je povinný zdokumentovať story, ktorú implementoval. Ak by sa dokumentovanie nerealizovalo týmto spôsobom, mohlo by byť náročné dať pri odovzdávaní dokumentáciu do akceptovateľnej podoby.

Testovanie – v tíme ja každý povinný otestovať si svoj kód. Napriek tomu, že sa realizuje aj automatizované testovanie, nemôže sa stať, že niekto dá na pullrequest kód, ktorý nepôjde buildnuť.

2.1 Prehľad manažérskych úloh

Prehľad manažérskych úloh spolu s povinnosťami, ktoré majú členovia tím sa nachádza v tabuľke nižšie.

Manažérska úloha	Člen tímu	Povinnosti
Scrummaster	Jakub Ďaďo	• vedenie tímových stretnutí, • plánovanie agendy stretnutia, • kontrolovanie činnosti členov tímu, • motivovanie členov tímu.
Technologická podpora	Erik Dzurňak	• poskytnúť technologickú podporu v prípade problémov s Gitom a Rubymine.
Správca DB	Erik Dzurňak	• udržovanie konzistentnosti databázy, • kontrola realizovaných migrácii, • aktualizovanie dátového modelu.
Manažér testovania	Michal Vantuch	• vytvorenie a udržovanie metodiky k testovaniu • vytvoriť a realizovať vybrané automatizované testy
Komunikácia s verejnosťou	Peter Paška	• vytvorenie aktualizovanie a spravovanie tímovej webovej stránky • kontrola tímového e-mailu
Manažér kódu	Tomáš Gasso	• nastavenie formálnych pravidiel pre zdrojový kód (odrážkovanie, správne pomenovania tried, metód, premenných) • kontrolovanie dodržiavania nastavených pravidiel
Evidencia úloh	Dávid Slezák	• vytvorenie a udržovanie metodiky k Redmine • exportovanie výsledkov šprintov z Redmine • formálna kontrola vytvorených stories a úloh • vytváranie nových šprintov v Redmine
Manažment rizík	Dávid Spišák	• vytvorenie a spravovanie risklistu • v prípade vzniku rizika informovať tím o navrhnutom krízovom pláne
Release manažér	Dávid Spišák	• pravidelne aktualizovať tímovú webovú stránku • po tímovej dohode, resp. po príkaze od scrummastera vykonať realese aplikácie

Tabuľka 1 - Prehľad manažérskych úloh

2.2 Podiel práce na dokumente k inžinierskemu dielu

Nasledujúca tabuľka uvádza prehľad zapojenia jednotlivých členov tímu pri tvorbe dokumentácie k inžinierskemu dielu nášho projektu.

Číslo kapitoly	Názov kapitoly	Vypracoval
1.1	Účel dokumentu	Dávid Spišák
1.2	Motivácia projektu	Dávid Spišák, Dávid Slezák
1.3	Ciele projektu na zimný semester	Dávid Slezák
2.1	Architektúra systému	Dávid Spišák, Erik Dzurňak, Tomáš Gaššo
2.2	Biznis architektúra	Dávid Spišák, Tomáš Gaššo
2.3	Aplikačná architektúra	Michal Vantuch. Dávid Spišák, Jakub Ďaďo
2.3.1	Klient Visitor Track - komponenty	Michal Vantuch. Dávid Spišák, Jakub Ďaďo
2.3.2	Server Visitor Track - komponenty	Michal Vantuch. Dávid Spišák, Jakub Ďaďo
2.3.3	Klient 3. strana – komponenty	Jakub Ďaďo, Peter Paška
2.4	Dátový model	Erik Dzurňak
2.5	Diagram tried	Dávid Spišák
3.1	Logger	Jakub Ďaďo, Peter Paška
3.2	E-form	David Spišák, Erik Dzurňak
3.3	Graf plotter	Peter Paška
3.4	Request parser a Visitor Track API	Jakub Ďaďo, Dávid Spišák
3.5	Action parser a Histogram	Michal Vantuch, Tomáš Gaššo
3.6	IAM	Erik Dzurňak
3.7	Statistics	Erik Dzurňak
3.8	Tech Filter	Dávid Spišák, Dávid Slezák
3.9	Webpages manger	Dávid Spišák
4	Záver	Dávid Slezák

Tabuľka 2 - Práca členov tímu na inžinierskom diele

3 Manažment rizík

Kapitola obsahuje prehľad rizík, ktoré sú špecifické pre náš projekt. Generické riziká nie sú do tabuľky zahrnuté.

ID	Názov rizika	Popis	Pravdepodobnosť	Stav	Ošetrenie	Krízový scenár	Zodpovedný
RISK_01	Nedostatočný HW serveru	Naša aplikácia pracuje, resp. bude pracovať a ukladať veľké množstvo dát. Pretože máme obmedzené HW zdroje, môže nastať situácia, kedy bude naša databáza príliš veľká a nášmu serveru dôjde fyzická pamäť.	80%	Kontrolované	V pravidelných intervaloch kontrolovať veľkosť dát v serverovej databáze.	V prípade, že sa tímovému serveru zníži kapacita voľného pevného disku pod 20% je potrebné kontaktovať p. Lacka, aby nám pridelil viac pamäte.	Dávid Spišák
RISK_02	Zlá možnosť testovania	Pretože naša aplikácia bude zbierať informácie z iných webových stránok, nie je triviálna záležitosť podobný stav nasimulovať a otestovať. Aj pri vytvorení testovacej vzorky údajov bude komplikované testovať správanie systému pri niekoľkých používateľoch.	85%	Kontrolované	Vynaložiť snahu v nájdení webovej stránky, ktorú by sme mohli monitorovať. Do úvahy prichádzajú aj webové stránky školy.Preventívne optimalizovať requestparser a record.js.	Použiť na testovanie aspoň stránku tímu a vykonať výkonnostné testovanie s čo najväčším počtom návštevníkov (zapojiť pri klikaní rodinu, kamarátov a pod.). Takto získame aspoň nejakú predstavu o tom, koľko používateľov súčasne začne robiť aplikácii problém.	Dávid Spišák
RISK_03	Nedostatočná úspešnosť algoritmu deduplikácie	Náš algoritmus pre deduplikáciu návštevníkov je založený na vzájomnom porovnávaní modelov návštevníkov. Po testovaní sa môže ukázať, že takýto spôsob deduplikácie nedokáže správne párovať návštevníkov a nakoniec budú vznikať duplikácie aj keď by nemali.	50%	Kontrolované	Realizovať dostatočné testovanie algoritmu.	Konzultovať problém a možnosti zlepšenia úspešnosti algoritmu s naším productownerom, ktorý je autorom myšlienky a má o problematike najlepšiu predstavu.	Dávid Spišák

RISK_04	Vypadnutie člena / členov tímu	Všetci členovia projektu sme študenti a preto sa môže stať, že niekto z nás ukončí štúdium a preto nebude môcť pokračovať na projekte.	10%	Kontrolované	Komunikovať s členmi tímu o prípadných študijných problémoch. Ak je niekto nútený odísť zo školy kvôli iným problém, mal by tím o tom informovať v dostatočnom predstihu.	Upraviť výkonnosť tímu a vziať do úvahy pri plánovaní zredukovaný počet členov tímu.	Dávid Spišák
~~RISK_05~~	~~Nedostatočné vzorky dát~~	~~Naša aplikácia pracuje so špecifickými dátami. Aby sme mohli začať s implementáciou, budeme potrebovať vzorku dát, ktorá nemusí byť ľahko získateľná.~~	~~0%~~	~~Nealtuálne~~	~~Konzultovať s našim produkt ownerom, či nemá vzorky dát, ktoré by nám mohol poskytnúť hneď od začiatku projektu.~~	~~Potrebné navrhnúť štruktúru a vytvoriť skript, ktorý naplni databázu testovacími údajmi, ktoré budeme môcť používať.~~	~~Dávid Spišák~~
~~RISK_06~~	~~Nenájdenie vhodnej stránky pre monitorovanie~~	~~Pre testovacie účely musí aplikácia monitorovať nejakú stránku už počas vývoja. Nie je možné vykonať testovanie na localhoste.~~	~~0%~~	~~Nealtuálne~~	~~Hľadať možnosti pre monitorovanie stránky od začiatku projektu, tzn. od prvého šprint.~~	~~Zo začiatku je možné monitorovať tímovú stránku. Ak nebude umožnené v neskorších fázach projektu monitorovať školské stránky, je potrebné vykonať prieskum o možných stránkach, ktoré by sme mohli monitorovať mimo školy.~~	~~Dávid Spišák~~

RISK_07	Record.js nebude postačujúci z výkonnostného hľadiska	Algoritmus bežiaci u klienta (monitorovaný web), nemusí byť dostatočne výkonný pri spracovaní akcií viacerých návštevníkov. Môže dochádzať k chybám pri odosielanírequestov, resp. k odoslaniu chybných dát, alebo zvýšeniu frekvencie odosielania.	50%	Kontrolované	Čo možno najskôr vykonať výkonnostné testy pre record.js. Zo začiatku je postačujúca aj webová stránka tímu, neskôr ale bude nutné zapojiť viac návštevníkov.	Optimalizovať record.js s cieľom zvýšiť jeho výkon.	Dávid Spišák
RISK_08	Parser nebude dostatočne výkonný	Samotné parsovanie dát prijatých z requestu nie je časovo náročné. Problémy môže spôsobiť ukladanie a pristupovanie do databázy. Pretože nemáme možnosť použiť HW riešenie pre odstránenie tohto problému (dva servery a loadbalancer) je potrebné riešiť problém implementačne.	50%	Kontrolované	Čo možno najskôr vykonať výkonnostné testy pre parser. Zo začiatku je postačujúca aj webová stránka tímu, neskôr ale bude nutné zapojiť viac návštevníkov.	Ak naša aplikácia nebude stíhať ukladať dáta to databázy, je potrebne znížiť frekvenciu prístupov do databázy. Z tohto dôvodu bude nutné vykonať analýzu možných spôsobov ako to dosiahnuť, napr. ukladať dáta po dávkach, použiť pomocné úložiská, buffery a pod.	Dávid Spišák
RISK_09	Record.js nebude správne fungovať v niektorých prehliadačoch, resp. inkognito módoch.	Niektoré funkcie použité v record.js na monitorovanie akcií návštevníka nemusia byť funkčné vo všetkých prehliadačoch.	60%	Kontrolované	Vykonať testovanie pre rôzne prehliadače.	Navrhnúť a implementovať spôsob, ktorý umožní prijímať requesty zo všetkých prehliadačov.	Dávid Spišák

RISK_10	Nestíhanie vytvorenia dokumentácie	Pretože realizujem projekt metódou Scrum, sme ako tím zameraný na dodanie funkčného výsledku. Z tohto dôvodu sa môže stať, že bude dokumentovanie zanedbané až do takej miery, že keď budeme musieť dokumenty odovzdať, nebude sa stíhať s ich dokončením.	50	Kontrolované	Pravidelne po dokončení implementácie tasku ju aj zdokumentovať. Taktiež je vhodné určiť človeka v tíme, ktorý bude dohliadať na stav dokumentácie.	Pri nestíhaní vytvárania dokumentácie je potrebné vyčleniť dostatok členov tímu na jej dopísanie.	Dávid Spišák

Tabuľka 3 - Prehľad rizík

4 Šprinty

Spôsob realizácie nášho projektu si vyžaduje pracovať v časovo konštantných etapách – šprintoch. V tejto kapitole sa nachádza stručný popis šprintov, ktoré boli doteraz realizované:

4.1 Šprint č.1

4.1.1 Cieľ šprintu

Hlavný cieľ prvého šprintu je možné rozdeliť do dvoch častí:

1. Realizovanie prepojenia medzi naším serverom a monitorovanými webovými stránkami.
2. Analýza ,vytvorenie a uloženie modelu používateľa (histogramy).

Okrem splnenia týchto dvoch hlavných funkčných cieľov bolo potrebné v prvom šprinte zabezpečiť aj množstvo manažérskych a technických úloh.
Prehľad zvyšných úloh, ktoré bolo potrebné v prvom šprinte realizovať:

1. Vytvoriť webovú stránku tímu.
2. Pripraviť tímový server.
3. Zvoliť nástroj pre správu úloh a vytvoriť k nemu metodiku.
4. Vytvorenie základných šablón pre dokumentáciu (Inžinierske dielo, Manažérsky dokument, všeobecná tímová šablóna).
5. Lokálne sfunkčnenie základných podporných softvérových riešení (Rubymine, Git a pod.).

4.1.2 Stand-up

Realizované pokroky v úlohách reportované počas stand-upu k prvému šprintu:

1. Nainštalovaný server s funkčnou webovou stránkou tímu – frameworkRails nebol počas prvého týždňa prvého šprintu inštalovaný.
2. Vytvorenie šablón pre inžinierske dielo, manažérsky dokument a všeobecnú šablónu.
3. Nástroj pre správu úloh bol zvolený Redmine – bol vytvorený a predstavený draft metodiky k Redmine.
4. Navrhnutá a vytvorená databáza.
5. Napísaná a členom tímu predstavená metodika ku Gitu.
6. Vytvorená REST služba na strane servera, kde sú odosielané dáta z monitorovanej stránky – testované len na localhoste. Z tohto testovanie vznikla vzorka dát, ktorá bola požadovaným vstupom pre analýzu a návrh modelu návštevníka.

4.1.3 Sumarizácia

Prvý šprint nebol z našej strany príliš vydarený. Pre nedostatočnú znalosť a málo skúseností sme pri ohodnocovaním jednotlivých stories používali strašne vysoké čísla, ktoré nám následne skreslili výkonnosť nášho tímu. Tiež sme ešte na začiatku šprintu nemali presne stanovený nástroj pre správu úloh a keďže sme sa rozhodovali medzi Trellom a Redminom,

tak sa nám stalo, že v istej fáze projektu sme mali úlohy rozhádzané v oboch. Následne sme museli vyčleniť jedného člena tímu, ktorý v nich spravil poriadok. Taktiež na začiatku šprintu nebola vytvorená metodika k systému na manažment úloh a preto vznikali zle pomenované úlohy, resp. úlohy, ktoré neboli priradené k žiadnej story a pod.

Z pohľadu splnenia naplánovaných úloh sa nám tiež nedarilo najlepšie, pretože z dvoch hlavných cieľov sa nám podarilo dokončiť len ten prvý, teda komunikáciu servera a monitorovaných webových stránok. Vytvorenie modelu návštevníka nebolo dokončené do zvoleného termínu. Už pri plánovaní sme predpokladali, že táto úloha je najnáročnejšia a podľa toho bola aj ohodnotená. Pretože sme ju chceli dokončiť, vyčlenili sme na jej riešenie až troch členov tímu. Tí ale nemohli pracovať paralelne pretože medzi jednotlivými úlohami tejto story boli závislosti, o ktorých sme počas plánovanie nevedeli. Preto musela byť táto story presunutá do ďalšieho šprintu.
Ostatné menšie naplánované úlohy boli úspešne dokončené v stanovenom termíne.

4.1.4 Retrospektíva

Čo začať robiť:

- Používať nižšie hodnotenia pre stories.
- Priraďovať jednotlivé úlohy pod strories – každá úloha musí mať rodiča.
- Názvy jednotlivých stories písať v Redmine veľkým pre jednoduchšie odlíšenie v zozname.
- Zamyslieť sa pri plánovaní nad možnými závislosťami jednotlivých úloh, aby sa neopakovali problémy z prvého šprintu.
- Začať viac komunikovať.
- Pokúsiť sa nastavovať hodnoty pre čas v Redmine tak, aby nevychádzal nezmyslený burntdownchart.

V čom pokračovať:

- Generovanie sumarizácie šprintov z Redmine-u.

Čo prestať robiť:

- Používať vysoké označenia pre stories.
- Aktualizovať úlohy v Redmine dve hodiny pred tímovým stretnutím.

4.2 Šprint č.2

4.2.1 Cieľ šprintu

V 2. šprinte bolo okrem nových stories potrebné dokončiť aj jednu story, ktorá zostala z 1. Šprintu.Tá momentálne mala najvyššiu prioritu a bolo potrebné ju splniť do stand-upu.
Okrem dokončenia vytvorenia modelu návštevníka bolohlavným cieľom implementovať funkcionality:

1. Vytvorenie identifikačného a autentifikačného komponentu (IAM) – registrácia, prihlásenie a odhlásenie používateľov.
2. Umožnenie prihláseným používateľom registráciu webových stránok do našej aplikácie.
3. Vytvorenie front-endu – zvolenie a implementácia šablóny do našej aplikácie.
4. Implementovanie deduplikačného algoritmu – algoritmus ktorý na základe modelu návštevníka určí, či sa jedná o nového alebo vracajúceho sa návštevníka.

Ďalšie úlohy, ktoré boli v šprinte realizované:

1. Analýza ďalších údajov, ktoré je možné získať od klienta.
2. Analýza možných spôsobov realizovania unite testov najmä pre back-endové výpočty (orientované na model návštevníka).
3. Prihláška na TP CUP.
4. Zriadenie pripojenia k databáze na servery prostredníctvom pgAdmina.
5. Založenie tímového e-mailu

4.2.2 Stand-up

Realizované pokroky v úlohách reportované počas stand-upu k druhému šprintu:

1. Nedokončená story z prvého šprintu bola počas stand-upu už funkčná a práve prebiehalo jej testovanie.
2. Do aplikácia bola doplnená šablóna pre front-end, ktorá dostatočne spĺňa požiadavky našej aplikácie.
3. IAM komponent bol zrealizovaný pomocou gemu Device, na stand-upe bolo možné registrovať, prihlásiť a odhlásiť používateľa. Funkcionalita nebola ešte namapovaná na novú front-endovú šablónu. Tiež nebol dokončený routing.
4. Registrovanie webových stránok bolo s časti implementované - registrovanie, editovanie a deaktivovanie webovej stránky bolo funkčné, pričom zobrazenie a editácia už boli aj v novej front-endovej šablóne. Pre úplne doimplementovanie je ale potrebné mať realizovaný komponent IAM (pre priradenie webových stránok správnemu používateľovi).
5. Bol vytvorený tímový e-mail.
6. Pola predstavená a komentovaná prihláška na TP CUP.
7. Prezentované dostupné možnosti unit testovania.

Zvyšný čas stand-up stretnutia bol venovaný dizajnovému a obsahovému návrhu našej aplikácie.

4.2.3 Sumarizácia

Druhý šprint bol najmä z manažérskeho hľadiska zvládnutý oveľa lepšie ako ten prvý. Počas druhého šprintu sme už mali vhodne nastavený Redmine s podrobnou metodikou, ktorý nám umožnil mať prehľad v úlohách. Tiež sa začal zaznamenávať progres v úlohách priebežne a nie tesne pred stretnutím. Rovnako bol spojazdnený komunikačný nástroj – HipChat, ktorý mal vylepšiť komunikáciu v tíme.

Napriek zlepšeniu v manažérskych aktivitách sa nám znova nepodarilo splniť všetky naplánované stories. V druhom týždni šprintu sa vyskytli značné problémy s registráciou a prihlasovaním používateľom, čo sa podarilo úplne doriešiť až deň pred koncom šprintu. Toto zdržanie spôsobilo, že registráciu webovej stránky prihláseného používateľa sa nepodarilo dokončiť v plánovanom termíne. Vznikol tu podobný problém ako v prvom šprinte, kedy jedna story musela čakať na dokončenie druhej. Pri plánovaní sme o tejto závislosti vedeli a preto sa čo možno najväčšia časť implementovala nezávisle. Dokonca aj počas stand-upu k druhému šprintu bolo možné predpokladať, že budú obe stories úspešne dokončené. Nakoniec sa to nepodarilo a jedna story musela byť dokončená v nasledujúcom šprinte.

4.2.4 Retrospektíva

Čo začať robiť:

- Začať používať HipChat.
- Lepšie rozdeliť čas na plánovanie na začiatku šprintu – snaha zredukovať čas plánovania.
- Úlohy k jednotlivým stories vytvárať v deň naplánovania šprintu.
- Počas plánovania viac komunikovať – nedostatočné argumentovanie členov, ktorí dali najnižšie, resp. najvyššie hodnotenie story (prevažne komunikujú tí istí členovia tímu).
- Na tímových stretnutiach bude používať počítač len ten kto prezentuje a ten kto robí zápis. Nebude sa programovať ani sa riešiť veci mimo stretnutia.
- Po prvom odovzdaní (18.11.2015) naplánovať a zorganizovať oslavný, resp. smútočný (nepredpokladá sa) teambuilding.
- Vytvárať úlohy počas plánovania – doteraz bolo realizované len ústne.

V čom pokračovať:

- Generovanie sumarizácie šprintov z Redmine-u.
- Pokračovať v nasadení ako doteraz. Napriek nedokončeniu jednej story bol výsledok druhého šprintu kladný a aj funkčný a vizuálny prírastok v aplikácii bolo výrazne poznať.
- V priebežnom dopĺňaní dokumentácie.

Čo prestať robiť:

- Prestali sme používať doterajšie komunikačné nástroje - Facebook.
- Nezačínať šprint v stredu ráno, tzn. nevytvárať úlohy až v stredu ráno.
- Pokúsiť sa vyhnúť náväznosti úloh.

4.3 Šprint č.3

4.3.1 Cieľ šprintu

Pretože sa znova nepodarilo dokončiť všetky naplánované stories, tak bolo v tomto šprinte potrebné tieto čo najrýchlejšie dokončiť. Z nových cieľov pre tento šprint bolo naplánované hlavne:

1. Zobrazenie výstupov z databázy vo forme grafov na stránke našej aplikácie.
2. Vytvorenie automatizovaného testovania.

Ďalšie úlohy, ktoré boli v šprinte realizované:

1. Filtrovanie používateľa podľa technických parametrov (prehliadač, platforma a pod.) – cieľom je zredukovať vstupnú množinu pre deduplikačný algoritmus.
2. Vytvoriť vyhodnocovať chybovosti deduplikácie – umožní nám zistiť, s akou pravdepodobnosťou dokážeme správne priradiť návštevníka k už existujúcemu návštevníkovi v databáze aplikácie.
3. Optimalizovať šablónu front-endu – odstrániť nepotrebné grafické prvky.

Na tímovom stretnutí sme boli informovaní, že jednému členovi tímu boli odcudzené osobné veci a to vrátane počítača (nedostavil sa ani na stretnutie). Pretože sme nevedeli či vôbec, a ak áno tak kedy a v akej miere sa bude môcť opäť zapojiť do tímových aktivít, tak sme stories pre tento šprint naplánovali tak, aby sa prípadne dali zvládnuť aj bez jeho pomoci.

4.3.2 Stand-up

Realizované pokroky v úlohách reportované počas stand-upu k tretiemu šprintu:

1. Bolo doplnené a otestované získavanie informácií o prehliadači a jeho verzii od návštevníka.
2. Do front-endu aplikácie bol doplnený Angular (prepínanie tabov bez refreshu).
3. Bola odstránená chyba, ktorú sme objavili v algoritme pre deduplikáciu.
4. Boli vyexportované výsledky druhého šprintu.
5. Dokončená registrácia webových stránok pre prihlásených používateľov.
6. Algoritmus pre filtrovanie návštevníkov bol funkční pre prehliadač a jeho verziu.
7. Aktualizácia výpočtov pri tvorbe modelu používateľa (doplnené dva atribúty).
8. Bola predstavená metodika na testovanie.

Ďalšie výsledky stand-up stretnutia:

- Graf bude zobrazovaný pomocou služby HighCharts.
- Prebrali sa testy pre histogramy.
- Náhradný termín stretnutia bol určený na stredu 18.11.2015 od 14:00 do 17:00 – počas nášho štandardného termínu bolo dekanské voľno.

4.3.3 Sumarizácia

Sumarizácia bola realizovaná na stretnutí, ktoré sa konalo po odovzdaní dokumentu.

4.3.4 Retrospektíva

Retrospektíva bola realizovaná na stretnutí, ktoré sa konalo po odovzdaní dokumentu.

5 Prehľad exportovaných výstupov z Redmine

Prehľad exportovaných výstupov z Redmine pre ukončené šprinty:

5.1 Šprint 1 - Agent Smith

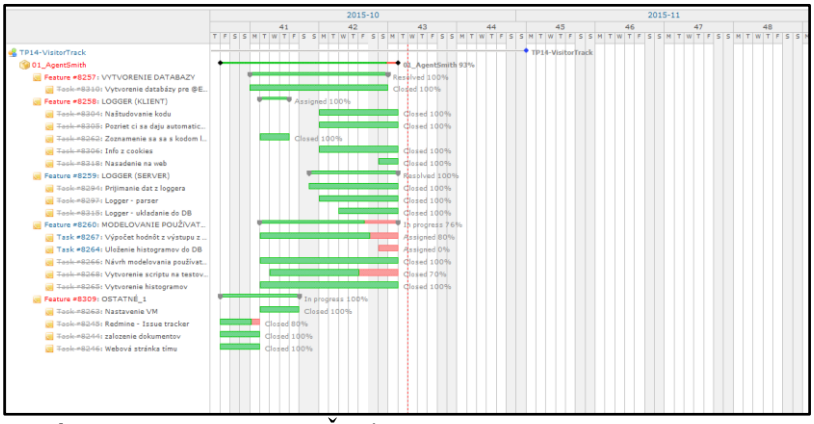
Prehľad úloh a ich priradenie jednotlivým členom tímu:

tasks_01.xlsx

Prehľad timelogov jednotlivých členov tímu:

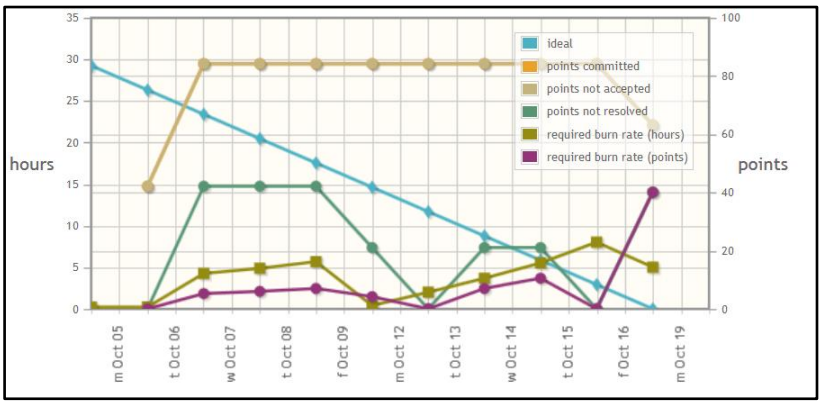
timelog_01.xlsx

Gantt chart šprintu 1



Obrázok 1- Gantt chart - Šprint 1

Burndownchart šprintu 1



Obrázok 2 - Burndown chart - Šprint 1

5.2 Šprint 2 – Bluto

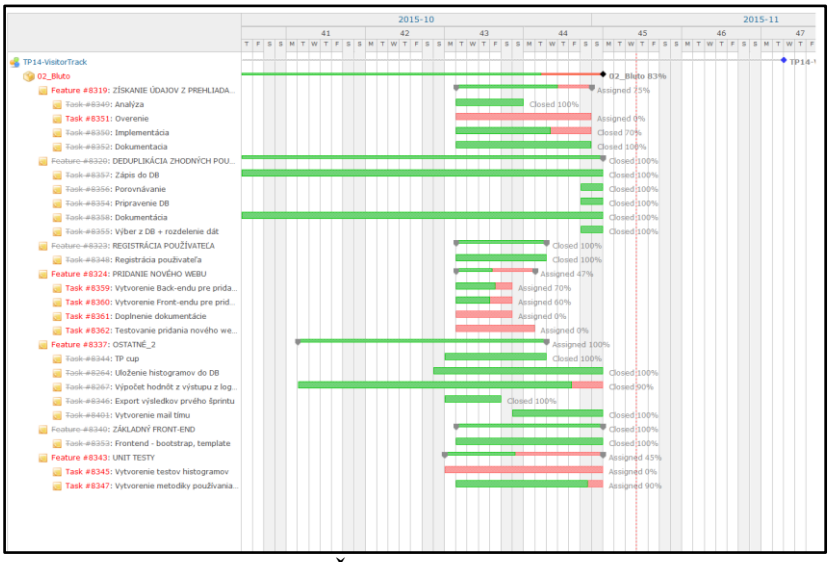
Prehľad úloh a ich priradenie jednotlivým členom tímu:

tasks_02.xlsx

Prehľad timelogov jednotlivých členov tímu:

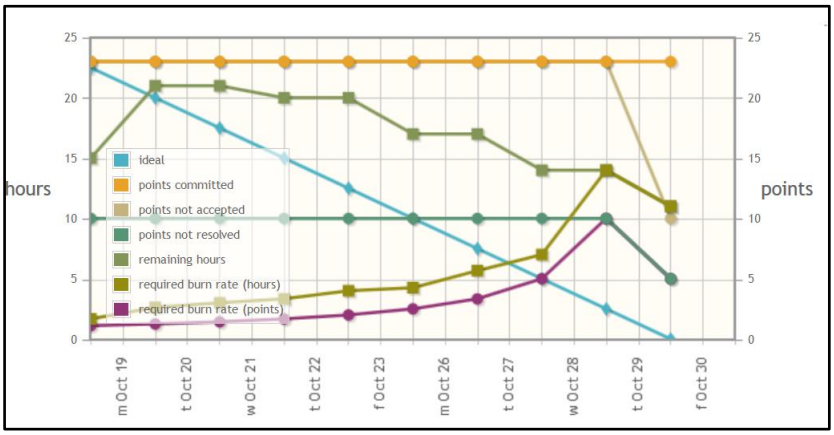
timelog_02.xlsx

Gantt chart šprintu 2



Obrázok 3 - Gantt chart - Šprint 2

Burndown chart šprintu 2



Obrázok 4 - Burndown chart - Šprint 2

6 Globálna retrospektíva

Globálna retrospektíva bude doplnená po skončení zimného semestra.

7 Metoditky

7.1 Git

Ako nastavit git
Stiahnite si git z https://git-scm.com/download/win. Mali by ste nainštalovať git bash a git gui. Po nainštalovaní si vytvorte priečinok, kde chcete mat uložený náš projekt. Ak to robíte cez git gui, tak tam je clone repository. Po vytvorení priečinku napíčte do konzoly "git checkout -b developorigin/develop" čo vám vlastne urobí v PC hlavnú vývojovú vetvu. Túto vetvu si budete vždy aktualizovať a budú sa s ňou mergovať ostatné merge počas vývoja.

EDIT git
Nezabudnite si najskôr naklonovať git repository: https://github.com/jdado/VisitorTrack.git Dá sa to aj zo stránky github. "CLone to desktop" alebo git clone https://github.com/jdado/VisitorTrack.git a až potom použite príkaz "git checkout -b developorigin/develop". Pre spresnenie MASTER branch, z tohto beží web, teda nepushuje sa tu resp. len hotfixy a robia sa z nej releasy DEVELOP branch, toto je vývojová branch, s ktorou sa budú mergovať vaše vývojovébranche. Vyzerá to následovne: https://www.atlassian.com/git/images/tutorials/collaborating/comparing-workflows/gitflow-workflow/05.svg

Git tutorial
https://www.atlassian.com/git/tutorials/Rubymine

7.2 Ruby / Ruby on Rails

Budeme bežať na ruby 2.1.7 - patri medzi novšie, najnovšia nepodporuje všetky gemy, tak radšej staršie. Stiahnite a nainštalujte http://rubyinstaller.org/downloads/ verzia 2.1.7 pozor na 32/64bit systém.
Stiahnite devKit, ktorý budete potrebovať pri bundlovani. (je hneď pod verziami ruby) znova pozor na verziu systému a treba stiahnuť ten, ktoré je "above Ruby 2.0". Rozbaľte devKit, do adresára, ktorý sa vám nepodarí vymazať (teda niekde save). Následne otvorte ruby, ktorý ste nainštalovali v príkazovom riadku (ten, ktorý sa vám nainštaloval inštaláciou nie "cmd"). Teraz nainštalujeme Rails príkazom do konzoly: gem installrails (trvá to asi 3min). Po tomto kroku choďte na stránku https://github.com/oneclick/rubyinstaller/wiki/Development-Kit a vykonajte krok 4. (teda v konzole choďte do priečinku, kde máte extrahovaný devkit) a keď tam budete tak do konzoly zadajte "ruby dk.rbinit" a "ruby dk.rbinstall".

Teraz otvorte Rubymine, otvorte git project, ktorý ste clonovali. Následne choďte do File->Settings->pomcousearchhladaj SDK-> skontrolujte, či je sdk nastavene na ruby 2.1.7 a spustitebundleinstall.

7.3 Redmine

Metodika pre náš softvér na manažment úloh sa nachádza v súbore:

Metodika k Redmine.docx

7.4 Testovanie

Aktuálna metodika pre testovanie sa nachádza v súbore:

Metodika pre testovanie.docx

7.5 Dokumentácia

Metodika pre tvorbu dokumentácie sa nachádza v súbore:

Metodika k dokumentovaniu.doc

8 Webové sídlo projektu

Webové sídlo projektu bude doplnené v ďalších fázach projektu.

Prílohy

Príloha A – Preberacie protokoly

Preberacie protokoly budú doplnené na konci projektu.