Slovenská technická univerzita
Fakulta informatiky a informačných technológií

Testovanie používateľského zážitku pomocou sledovania pohľadu

Dokumentácia riadenia

Vedúci tímu: Ing. Róbert Móro

Členovia tímu: Bc. Peter Kiš, Bc. Matej Kucek, Bc. Viktória Lovasová,
Bc. Peter Kušnír, Bc. Marek Šulek, Bc. Šimon Valíček, Bc. Martin Svrček

Akademický rok: 2014/2015

Obsah

1 Úvod

Tento dokument predstavuje podrobnú dokumentáciu k riadeniu tímového projektu Testovanie používateľského zážitku pomocou sledovania pohľadu, ktorému sa venuje sedemčlenný tím študentov s názvom Team Focus.

Projekt nadväzuje na prácu minuloročného tímu Carrots, ktorý v rámci projektu Sledovanie pohľadu pri používaní aplikácií vytvoril základnú infraštruktúru systému pomenovanom TrackView. Nakoľko sa minuloročnému tímu podarilo vytvoriť prostredia na správu používateľov, projektov a sedení, ako aj infraštruktúru pre zber a spracovanie dát z eyetrackerov, rozhodli sme sa v našom projekte zamerať na problematiku zjednodušovania, vyhodnocovania a vizualizácie zbieraných dát. Rovnako tak sme si dali za cieľ vytvoriť prepojenie nami vyvíjaného systému s existujúcimi webovými stránkami a systémami pomocou nášeho API. Toto prepojenie hodláme demonštrovať na fakultnom edukačnom webovom systéme Alef.

Nasledujúce kapitoly popisujú Team Focus a jeho snahu o prácu v agilnom prístupe vývoja softvéru. Úvodná kapitola popisuje jednotlivých členov tímu, ich manažérske role a postupy, ktorými sa snažia riadiť pri vývoji. Rovnako sú v prvej kapitole popísané jednotlivé šprinty, silné a slabé stránky tímu pri dodržiavaní agilného prístupu, ako aj vízia riešenia akútnych problémov. Druhá kapitola pojednáva o kompetenciách jednotlivých členov tímu, ich programátorských zručnostiach, odborných schopnostiach, ako aj voľnočasových aktivitách. V tretej kapitole sú uvedené všetky metodiky, podľa ktorých sa Team Focus riadi pri vývoji, testovaní a údržbe vyvíjaného softvéru. Posledná kapitola obsahuje vyexportované backlogy tímu pre všetky dokončené šprinty.

1.1 Roly členov a podiel práce

V prvej časti kapitoly sme zaznamenali jednotlivé roly členov nášho tímu, tak ako sme si ich určili na začiatku projektu. V druhej časti sú opísané podiely práce členov tímu na jednotlivých častiach dokumentácie.

1.1.1 Manažérske činnostičlenov tímu

V našom tíme máme jednotlivé činnostirozdelené nasledovne:

- *Peter Kiš* - vedúci tímu, organizácia komunikácie v tíme, informovanie učiteľa o stave projektu
- *Matej Kucek* - zástupca vedúceho, udržiavanie informácií o stave projektu, integrácia softvéru
- *Marek Šulek* - monitorovanie, prehliadky vytváraného výsledku
- *Šimon Valíček* - manažment verzií, organizácia zdrojov, definovanie architektúry a rozsahu projektu
- *Viktória Lovasová* - riadenie procesu dokumentovania, identifikácia a riadenie chýb v softvéri
- *Martin Svrček* - plánovanie pre tím a jednotlivých členov tímu, vyhodnocovanie plnenia plánu a návrh úprav, identifikovanie a riadenie rizík

- *Peter Kušnír* - udržiavanie informácií o stave projektu, evidencia úloh, vyhodnocovanie testov

1.1.2 Podiel práce na dokumentácii

Dokumentácia riadenia

Tabuľka 1 znázorňuje prácu členov na jednotlivých kapitolách v dokumente Dokumácia riadenia.

Tabuľka 1 – Podiel práce na dokumentácii

Úvod	Peter Kiš
Role členov tímu a podiel práce	Každý individuálne
Aplikácia manažmentov	Martin Svrček
Sumarizácia šprintov	Martin Svrček
Použité metodiky	Martin Svrček
Globálna retrospektíva	Martin Svrček
Zoznam kompetencií tímu	Peter Kiš
Metodiky	
Metodika písania zdrojových kódov v C#	PeterKiš
Metodika integrácie	Matej Kucek
Metodika prehliadok kódu	Marek Šulek
Metodika manažmentu verzií	Šimon Valíček
Metodika manažmentu chýb	Viktória Lovasová
Metodika dekompozície používateľských príbehov na úlohy	Martin Svrček
Metodika testovania	Peter Kušnír
Export evidencie úloh	Matej Kucek

Dokumentácia inžinierskeho diela

Tabuľka 2 znázorňuje prácu členov na jednotlivých kapitolách v dokumente Dokumácia riadenia.

Tabuľka 2 – Podiel práce členov tímu na dokumente

Úvod	Marek Šulek
Globálne ciele	Martin Svrček
Celkový pohľad na systém	Šimon Valíček
Modul API	Šimon Valíček
Knižnice JS	Martin Svrček
Agregácie	Peter Kiš, Matej Kucek

1.2 Aplikácia manažmentov

V rámci projektu máme rozdelené jednotlivé role a každý člen má za úlohu vytvoriť metodiky spadajúce do danej oblasti, ktoré nám pomôžu pracovať efektívne a kvalitne. V súčasnom

stave sme fungovali skôr na báze ústnej dohody ako sa budú jednotlivé procesy robiť, keďže ešte neboli spísané všetky metodiky.
V rámci manažmentu kvality bolo našou snahou vytvárať kód, ktorý bude čo najviac prehľadný a čitateľný a hlavne bez zbytočných chýb. V rámci celého projektu sa chceme zamerať práve na kvalitu kódu možno aj na úkor kvantity. V rámci tejto oblasti sa preto snažíme ku každej ucelenej funkcionalite vždy vykonávať aj jednotlivé úlohy zabezpečujúce kvalitu softvéru ako sú prehliadky kódu a testovanie. Toto pomerne jasne dokazujú úlohy zaznamenané v systéme TFS.
V kontexte určitých praktík v písaní kódu sme spoločne určili človeka, ktorý má pomerne dlhé skúsenosti s prácou v jazyku C# aby vypracoval metodiku na písanie kódu, ktorou sa riadime (viď 3.5 Metodika písania kódu v jazyku C#).
Pri práci na projekte využívame verziovací nástroj *Git*. Tento nástroj umožňuje určité špecifické aktivity ako vytváranie rôznych vetiev kódu, *commit* zmien v kóde a podobne, ktoré je však možné robiť rôznymi spôsobmi. Preto je veľmi dôležité, aby sme mali nejaké spoločné postupy, ktoré bude každý uplatňovať (napríklad aj pri názvoch), a bude tak jednoduché orientovať sa v rôznych verziách kódu. V súčasnosti sa snažíme pri spomínaných aktivitách využívať všeobecné postupy a vytvárať názvy, ktoré budú prehľadne opisovať danú vetvu alebo commit. V tomto kontexte už existuje aj metodika, ktorá opisuje všetky tieto podrobnosti (viď 3.6 Metodika manažmentu verzií).

V kontexte plánovania využívame všeobecné postupy (best practices) metodiky SCRUM pre kvalitné plánovanie jednotlivých šprintov. V tíme využívame nástroj TFS, do ktorého okrem iného zaznamenávame jednotlivé úlohy a ich plnenie. Jednou z dôležitých aktivít je dekompozícia používateľských príbehov, pri ktorej ako tím pomerne dobre stanovujeme jednotlivé úlohy. Toto je pekne vidieť aj na exporte úloh z TFS, kde sa často vyskytuje pomerne rovnaký vzor úloh:

- Analýza problematiky
- Implementácie v niekoľkých krokoch
- Prehliadka kódu a testovanie

Aj v tejto oblasti však už existuje metodika (viď 3.2 Metodika dekompozície používateľských príbehov na úlohy).

V rámci testovania musíme priznať, že sa nám príliš nedarí testovať náš kód. Je to hlavne z dôvodu malej skúsenosti s takýmto prístupom. Tento aspekt nám určite pomôžu zlepšiť aj jednotlivé dokumentácie k testovaniu (viď príloha A: Testovanie v jazyku javascript - JASMINE) a metodiky (viď 3.7 Metodika testovania).

Jednou z dôležitých aktivít je aj riešenie chýb, ktoré sa objavia. Je nutné postupovať určitým spôsobom, ktorý bude každý poznať a zabezpečí čo najlepšie vyriešenie daného problému (viď 3.13.1 Manažmentriešenia chýb).

1.3 Sumarizácia šprintov

V tejto kapitole sú opísané a zhodnotené jednotlivé šprinty. Venujeme sa hlavne celkovému procesu práce na nich a spĺňaniu všetkých stanovených úloh.

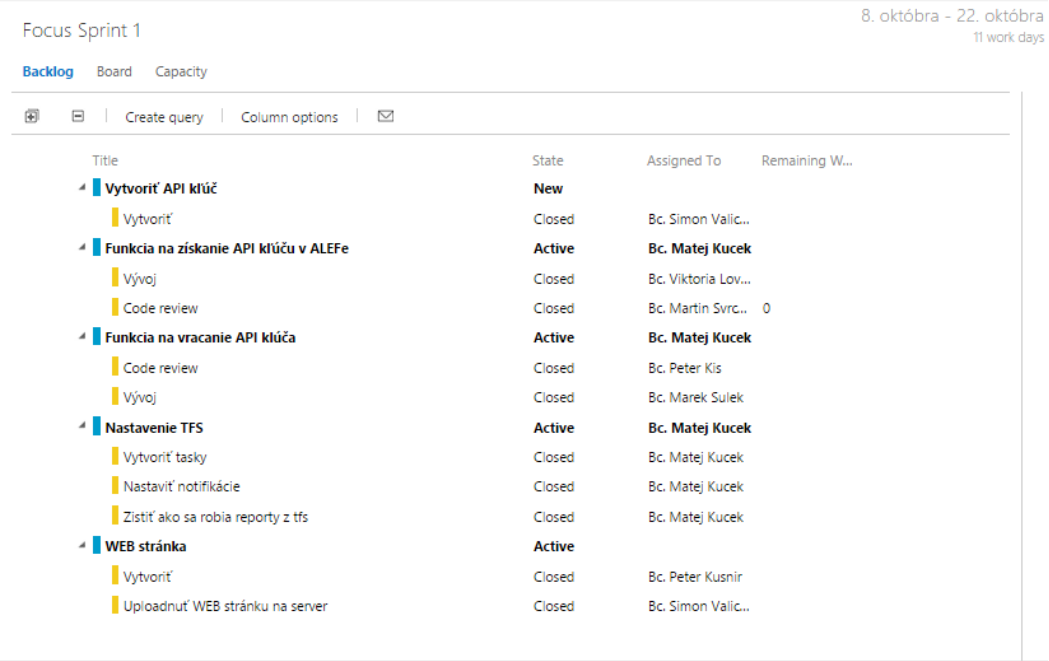
1.3.1 Šprint 1

Prvý šprint bol pre náš tím pomerne špeciálny, keďže pokračujeme v práci z minulého roku a základná infraštruktúra systému na sledovanie pohľadu už existuje.

Na jednej strane bolo preto našou úlohou hlavne pochopiť súvislosti a fungovanie aktuálneho stavu systému tak, aby sme mohli plynule pokračovať v jeho vývoji. Tomuto boli prispôsobené aj jednotlivé používateľské príbehy a súvisiace úlohy.

Na strane druhej bol tento šprint pre nás aj zoznamovacou fázou, keďže sme sa predtým niektorý príliš nepoznali.

Ako sme už spomenuli v rámci šprintu sa riešili základné úlohy na oboznámenie sa so systémom **Error! Reference source not found.**.



Obrázok 1– Sprint 1 backlog

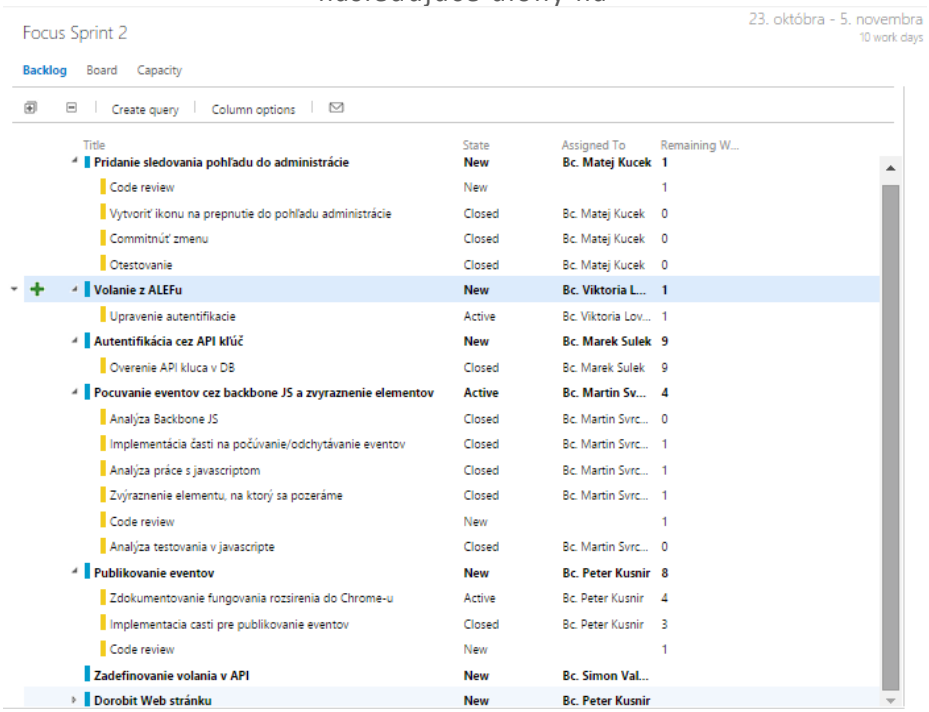
Pretože sme ešte nepoznali nástroj TFS bolo nutné prácu s ním analyzovať a z tohto dôvodu sme ho vlastne pri zaznamenávaní práce na šprinte č.1 ani nepoužili. Všetky úlohy sme pridali do systému až na konci, keď už boli hotové čo zapríčinilo, že náš burndown graf je prázdny a neuvádzali sme ho preto ani v dokumente.V rámci prvého šprintu sa nám však podarilo spoločnými silami spraviť všetky úlohy. Jediným problémom bolo nasadenie stránky v rámci termínu pre predmet Tímový projekt. Tento problém bol však na strane servera a súvisel s presmerovaním stránky tímu z minulého roku.

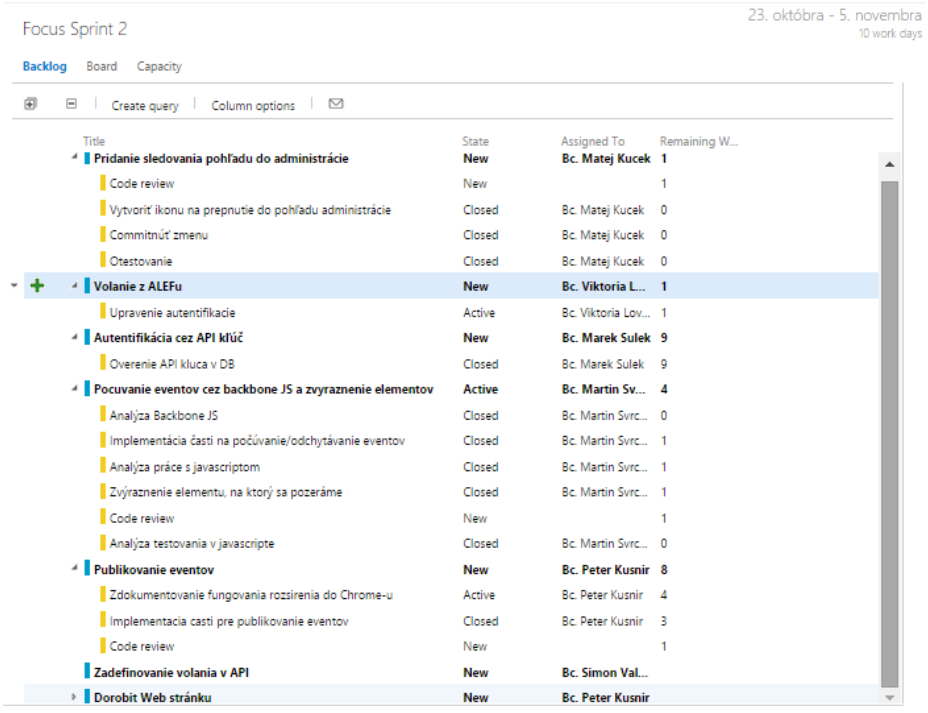
V retrospektíve tohto šprintu sme určili, že bude potrebné hlbšie pochopiť problematiku celého systému, pretože väčšina z nás nemá vôbec predstavu ako to celé funguje vo vnútri. V tomto kontexte sme sa dohodli, že tomuto prispôsobíme aj nasledujúce úlohy.

Na druhej strane sme sa zhodli, že sme sa počas tohto šprintu ako tím pomerne dobre spoznali. Odcenili sme takisto aj vzájomnú pomoc na úlohách.

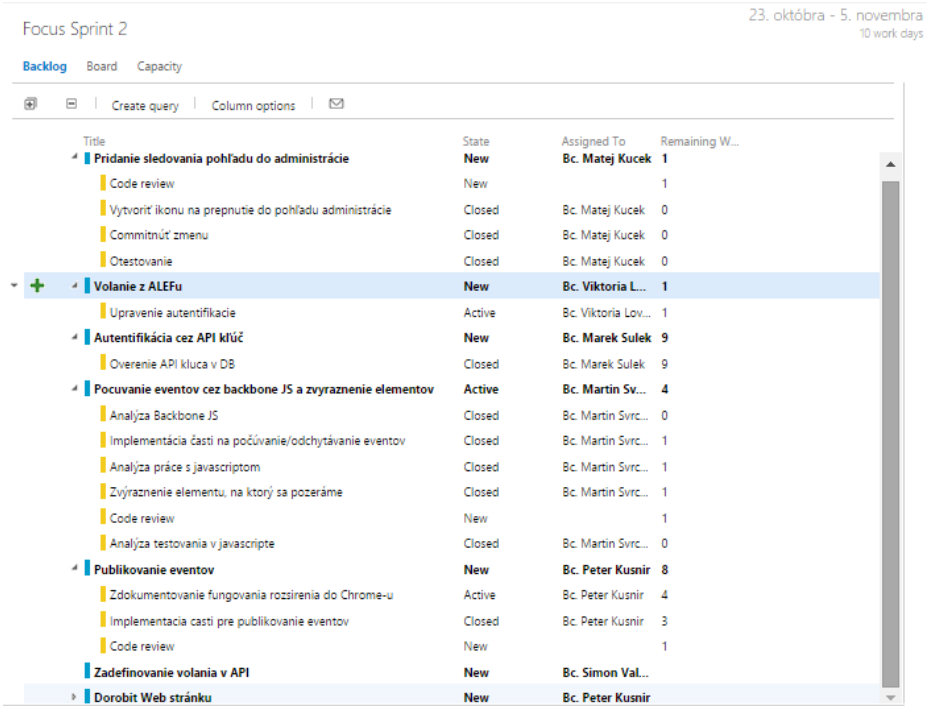
1.3.2 Šprint 2

Na základe zistení zo šprintu 1 sme si stanovili, že vzhľadom na zložitosť systému bude potrebné venovať sa ešte hlbšie jednotlivým častiam systému prostredníctvom implementácií. S ohľadom na zistenia zo stretnutia so zákazníkmi sme si teda stanovili nasledujúce úlohy na





.



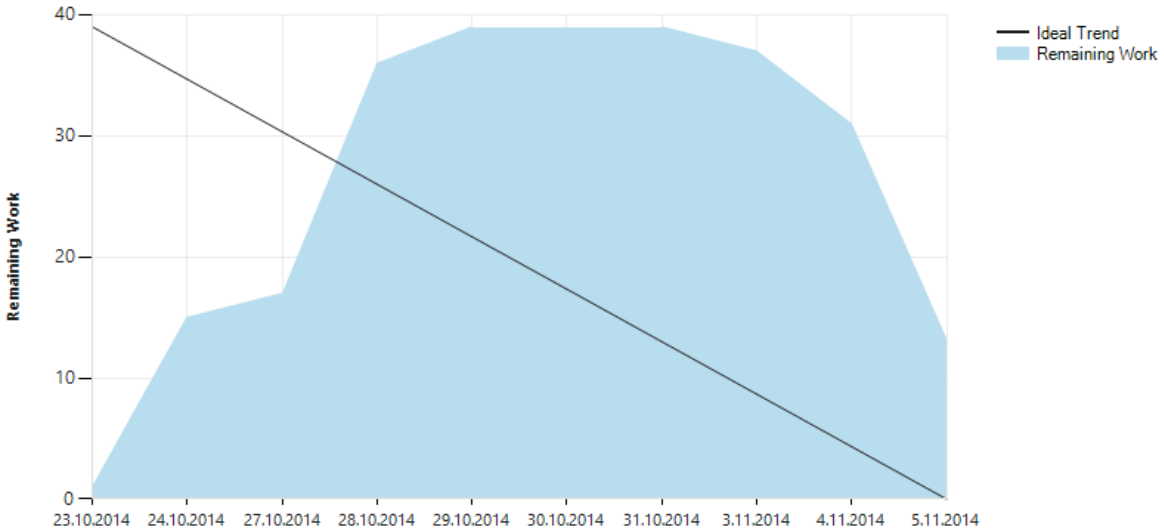
Obrázok 2– Šprint 2 backlog

Počas druhého šprintu sme si zaznamenali všetky používateľské príbehy aj úlohy do systému TFS. Avšak niektorí členovia si nerozdelili im pridelené príbehy na úlohy dostatočne skoro a preto nám graf rástol v priebehu prvého týždňa (Obrázok 3). V rámci zadefinovaných úloh sa nám podarilo spraviť skoro všetky, avšak ukázalo sa, že sme si nechávali úlohy až na koniec šprintu(viď Obrázok 3) a preto sa nestihli dorobiť prehliadky kódu a dokumentácie k jednotlivým problematikám (**Error! Reference source not found.**).
Jediným používateľským príbehom, ktorý ostal z veľkej časti nedorobený a preto musel byť presunutý do ďalšieho šprintu bolo:

- Počítanie agregácií

Dôvodom nedorobenia príbehu bolo zistenie, že je oveľa zložitejší ako sa predpokladalo a bude ho potrebné rozdeliť medzi viac ľudí.

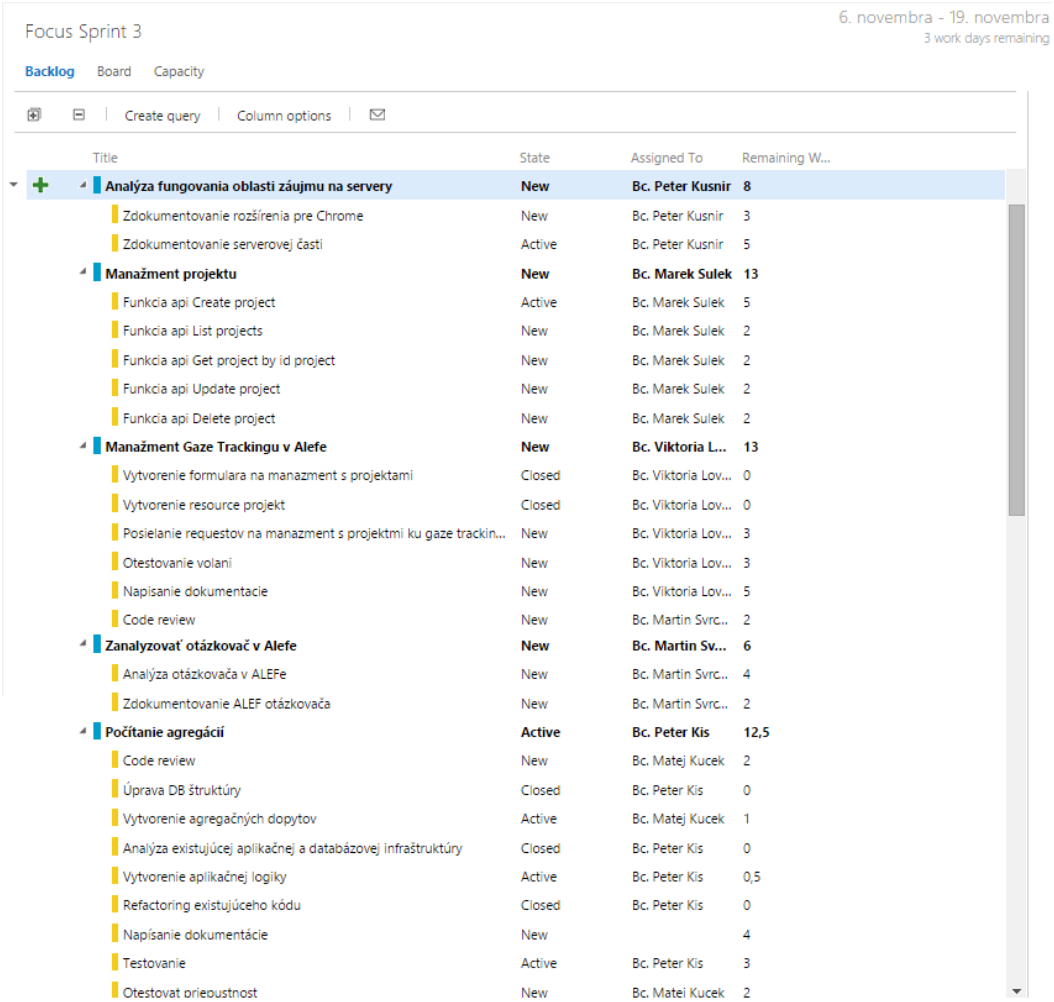
V retrospektíve druhého šprintu sme určili ako najväčší problém nedostatočnú komunikáciu prostredníctvom nástroja HipChat. Určili sme si, že je potrebné aby sme viac komunikovali aby sme vedeli riešiť problémy v tíme čo najskôr. Identifikovali sme aj problém neskorého riešenia úloh a zhodli sa, že v budúcnosti je potrebné venovať sa úlohám priebežne.Výhodou bolo pre nás lepšie pochopenie problemtiky na základe riešenia určených úloh a zorganizovanie stretnutia so zákazníkmi, kde sme si veľa vecí ujasnili alebo, na ktoré sme sa začali pozerať inak.



Obrázok 3 – Šprint 2 Burndown chart

1.3.3 Šprint 3

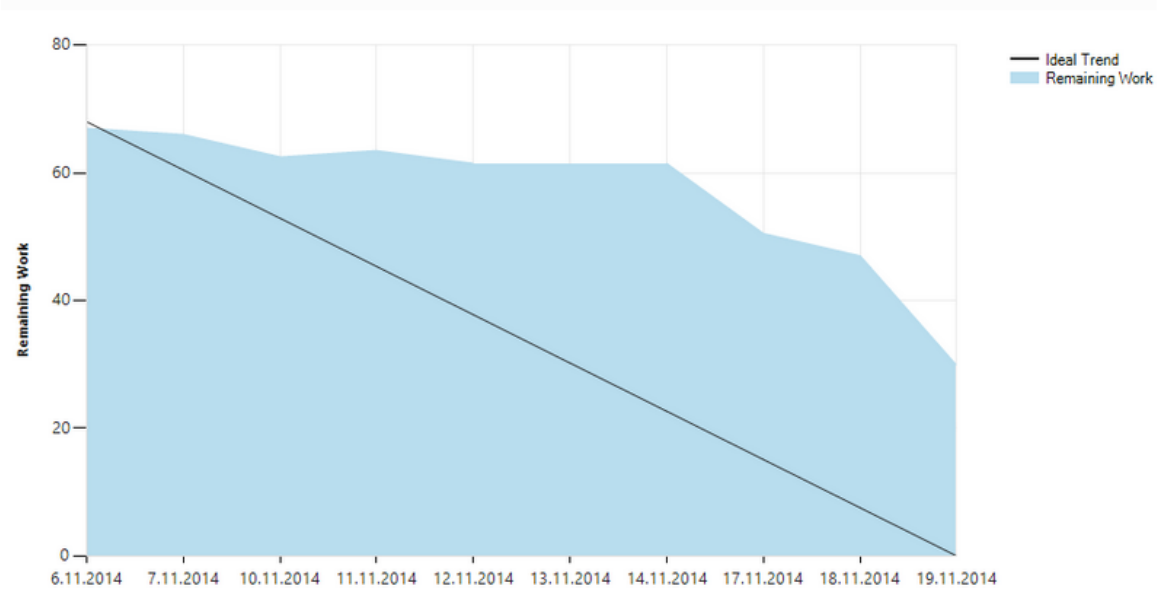
V rámci tretieho šprintu sme si už zadefinovali pomerne reálne používateľské príbehy. V rámci príbehu *Analýza otázkovača v ALEFe* sa však v strede šprintu zistilo, že súčasná implementácia počúvania udalostí (event) prostredníctvom knižnice BackboneJS je z hľadiska prenositeľnosti na rôzne iné aplikácie zlým riešením. Preto bolo potrebné prerobiť implementáciu samotného počúvanie udalostí. Jednotlivé používateľské príbehy sú na Obrázok4.



Obrázok4 – Šprint 3 backlog

V rámci tohto šprintu sa nám podarilo spraviť väčšinu úloh avšak niektoré sa taktiež nestihli čo je možné vidieť aj na burndown grafe (Obrázok 5). Jedným z príčin bola určite potreba práce na dokumentácii, ktorej sme museli venovať nejaký čas.

V retrospektíve tretieho šprintu bolo hlavnou témou ako brať do úvahy rôzne udalosti, ktoré mžu ovplyvniť daný šprint. V tomto šprinte bolo napríklad potrebné pracovať na dokumentácii. Táto úloha pomerne výrazne zasiahla do času, ktorý mal každý člen vymedzený na prácu na daných úlohách a príbehoch. V budúcnosti budeme chcieť takéto udalosti predikovať vopred aby sme sa mohli na ne pripraviť.



Obrázok 5 – Šprint 3 Burndown chart

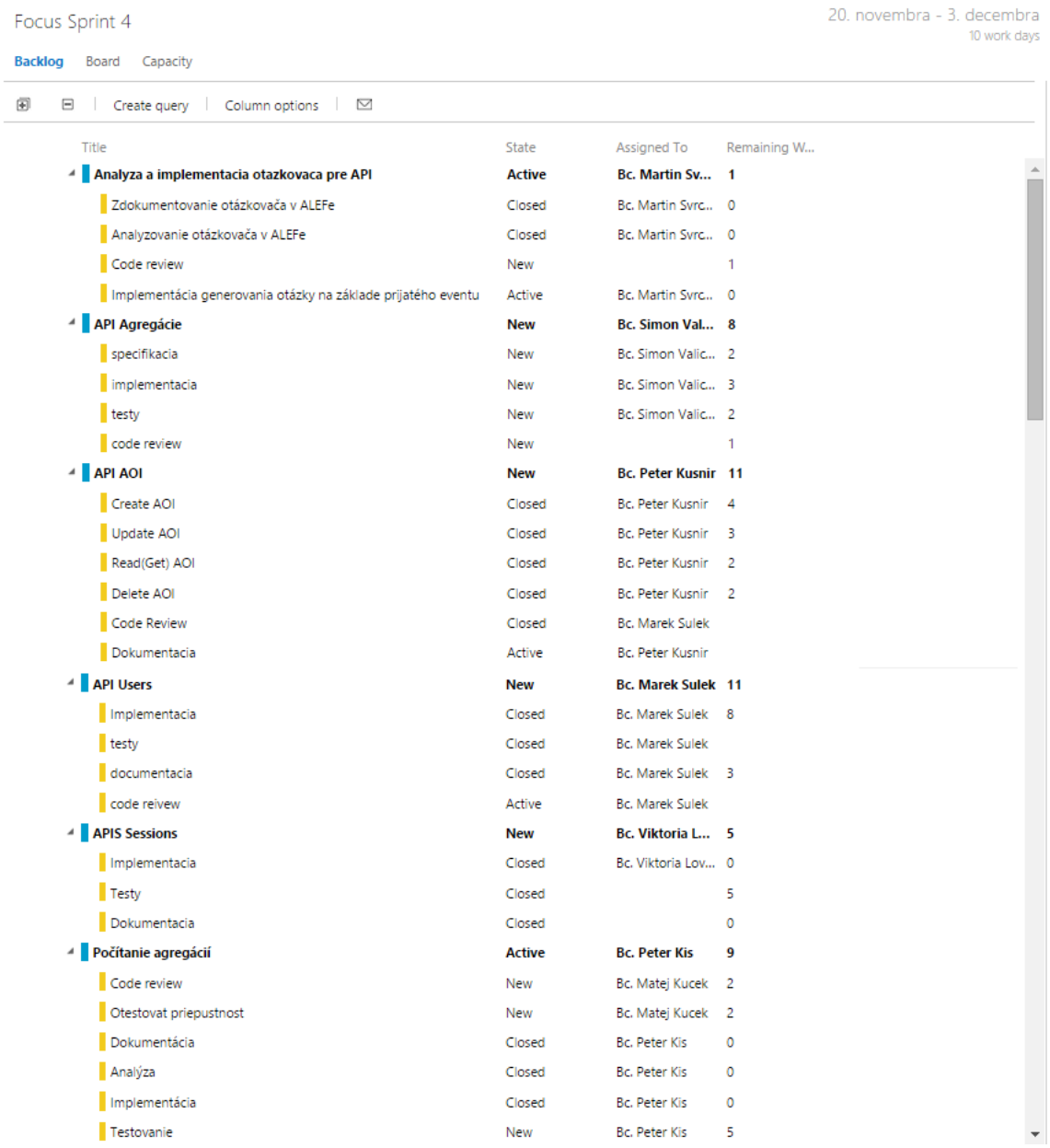
1.3.4 Šprint 4

V rámci štvrtého šprintu sme sa pustili do vytvárania jednotlivých API volaní, ktoré budú tvoriť podstatnú časť našej ďalšej práce.

Takisto sme zistili, že otázkovač v systéme ALEF, ktorý sme analyzovali v treťom šprinte nie je úplne to čo potrebujeme a preto bolo potrebné vykonať analýzu v kontexte pýtača v systéme ALEF (viď príloha C: Analýza pýtača v ALEFe).

V rámci vytvárania agregácií bolo potrebné vykonať posledné úpravy, tak aby bola táto časť úplne dokončená. Na používateľskom príbehu *Počítanie agregácií* je vidieť ako sme zle identifikovali zložitosť. Malo to vplyv na nutnosť prenosu tohto príbehu do nasledujúceho šprintu.

Jednotlivé príbehy a úlohy k nim priradené sú zobrazené na Obrázok 6.

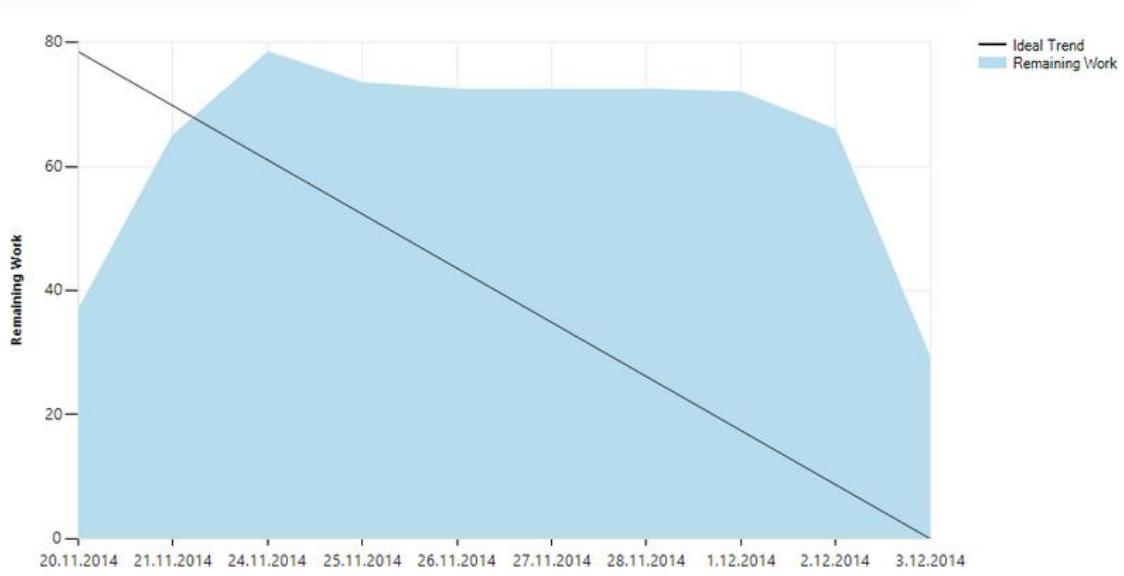


Title	State	Assigned To	Remaining W...
Analyza a implementacia otazkovaca pre API	**Active**	**Bc. Martin Sv...**	**1**
Zdokumentovanie otázkovača v ALEFe	Closed	Bc. Martin Svrc...	0
Analyzovanie otázkovača v ALEFe	Closed	Bc. Martin Svrc...	0
Code review	New		1
Implementácia generovania otázky na základe prijatého eventu	Active	Bc. Martin Svrc...	0
API Agregácie	**New**	**Bc. Simon Val...**	**8**
specifikacia	New	Bc. Simon Valic...	2
implementacia	New	Bc. Simon Valic...	3
testy	New	Bc. Simon Valic...	2
code review	New		1
API AOI	**New**	**Bc. Peter Kusnir**	**11**
Create AOI	Closed	Bc. Peter Kusnir	4
Update AOI	Closed	Bc. Peter Kusnir	3
Read(Get) AOI	Closed	Bc. Peter Kusnir	2
Delete AOI	Closed	Bc. Peter Kusnir	2
Code Review	Closed	Bc. Marek Sulek	
Dokumentacia	Active	Bc. Peter Kusnir	
API Users	**New**	**Bc. Marek Sulek**	**11**
Implementacia	Closed	Bc. Marek Sulek	8
testy	Closed	Bc. Marek Sulek	
documentacia	Closed	Bc. Marek Sulek	3
code reivew	Active	Bc. Marek Sulek	
APIS Sessions	**New**	**Bc. Viktoria L...**	**5**
Implementacia	Closed	Bc. Viktoria Lov...	0
Testy	Closed		5
Dokumentacia	Closed		0
Počítanie agregácií	**Active**	**Bc. Peter Kis**	**9**
Code review	New	Bc. Matej Kucek	2
Otestovat priepustnost	New	Bc. Matej Kucek	2
Dokumentácia	Closed	Bc. Peter Kis	0
Analýza	Closed	Bc. Peter Kis	0
Implementácia	Closed	Bc. Peter Kis	0
Testovanie	New	Bc. Peter Kis	5

Obrázok 6 - Šprint 4 backlog

V rámci štvrtého šprintu sa podarilo spraviť väčšinu úloh. Burndown graf však nevyzerá úplne ideálne (Obrázok 7- Šprint 4 Burndown chartObrázok 7) a to z dôvodu neskorého zaznamenávania práce a ukončenia úloh v systéme TFS. Tento problém nás sprevádza počas všetkých šprintov a preto je potrebné aby sme to v budúcnosti zmenili.

V retrospektíve štvrtého šprintu sme sa rozprávali o dôležitosti zaznamenávania odporacovaného času do systému TFS. Zhodli sme sa taktiež na tom, že jednotlivé používateľské príbehy by bolo dobré dekomponovať na úlohy priamo na stretnutí a nie samostatne. Toto môže vyriešiť problém s nepresným určením zložitosti daného príbehu.



Obrázok 7- Šprint 4 Burndown chart

1.4 Použité metodiky

V rámci práce na projekte sme vypracovali nasledovné metodiky:

- Písanie zdrojových kódov v jazyku C#
  - Metodika zahŕňa pravidlá pre tvorbu názvoslovia, písanie komentárov, formátovanie zdrojových textov a používania dátových typov.
  - Metodika neobsahuje popis syntaxe jazyka C#, návod na písanie optimálneho kódu, ani postupy pri písaní aplikačnej logiky.
  - Popísané pravidlá vychádzajú z oficiálnej príručky písania zdrojových kódov v jazyku C# od spoločnosti Microsoft.
  - Metodika integrácieMetodika opisuje postup ako nasadiť doimplementovanú funkcionalitu, poprípade zmenu, na server. Toto nasadzovanie sa týka iba vývojovej vetvy.
  - Postup je opisovaný v prostredí Visual Studio 2013 za použitia techológie Team Foundation Server.
- Metodika prehliadky kódu
  - Účelom tejto metodiky je definovať postup pri prehliadkach kódu, za cieľom aby zachovania dostatočná kvalita zdrojového kódu.
  - Metodika obsahuje postup, ktorý sa musí dodržiavať pri prehliadke kódu a opis práce s nástrojmi,ktoré sa pri prehliadke musia použiť.
- Metodika používania softvérového nástroja na správuverzií zdrojového kódu
  - Metodický pokyn k používaniu softvérového nástroja na kontrolu verzií zdrojového kódu.
- Manažment riešenia chýb
  - Metodika popisuje proces zaznamenávania a manažovania chýb prostredníctvom nástroja Team Foundation Server.
- Dekompozícia používateľských príbehov na úlohy v nástroji TFS

    - Princípom tejto metodiky je poskytnúť postup alebo metódu na spomínané funkcionálne dekomponovanie používateľských príbehov na menšie časti (úlohy).
    - Táto metodika zahŕňa celý proces dekompozície používateľských príbehov od vytvorenia základne predstavy na tímovom stretnutí až po samotné zadávanie úloh do systému TFS.
    - Metodika sa však nezaoberá vytváraním používateľských príbehov ani správou úloh počas šprintu (napr. aktualizácia času a stavu úloh).
- Metodika testovania
    - Tento dokument popisuje proces vytvárania testovacích projektov a testov pre jednotlivé komponenty aplikácie napísanej v jazyku C#.
    - Zachytáva postup tvorby testu, jeho členenie, konvencie v pomenovávaní jednotlivých testovaných častí a spúšťanie testov.

1.5 Globálna reptrospektíva

V tejto časti bližšie opíšeme určité aspekty projektu a práce na projekte z hľadiska ich prínosu a prípadných zmien, ktoré treba spraviť.

1.5.1 Aspekty projektu, s ktorými sme boli spokojní

- Vzájomná pomoc
    - Jednou z pozitívnych stránok našej doterajšej práce je pomerne dobrá spolupráca na projekte.
    - Jednotlivý členovia si pri ich úlohách často pomáhali, hlavne vtedy ak už jeden mal skúsenosti s danou problematikou a pomohol ďalšiemu členovi s pochopením problému.
    - Tento aspekt je pre celkový úspech projektu nesmierne dôležitý a je potrebné, aby sme si ho uvedomili. Takisto je potrebné aby sme v takomto trende pokračovali, pretože to môže výrazne uľahčiť a aj zrýchliť jednotlivé aktivity.
- Správa úloh v TFS
    - Na začiatku (v priebehu 1 a 2 šprintu) sme si ako tím nie príliš dobre zaznamenávali prácu v systéme TFS a takisto sme mali problém s včasným rozdelením príbehov na úlohy.
    - V priebehu tretieho šprintu sa však toto výrazne zlepšilo. Každý z členov si hneď v deň stretnutia k novému šprintu vytvoril jednotlivé úlohy. Toto je z hľadiska sledovania správneho plnenia cieľov veľmi dôležité a taktiež sa môže pomerne skoro určiť a vyriešiť nejaký problém.
    - Takisto sme pomerne dobre zaznamenávali jednotlivé stavy plnenia úloh. V tomto kontexte je dobre, že každý člen zaznamenáva zmenu stavu danej úlohy čo možno najskôr. Pomôže to lepšej orientácii počas šprintu.

1.5.2 Aspekty projektu, ktoré treba zmeniť

- Rozdelenie práce na šprinte

    - Z burndown grafov je možné vidieť, že sa nám často nepodarilo splniť všetky stanovné úlohy. Z veľkej časti to súviselo hlavne s neskorým začiatkom práce na danej úlohe alebo príbehu.
    - V tomto kontexte je preto potrebné, aby každý jeden člen vykonával všetky úlohy postupne od začiatku šprintu. Pomôže to vo viacerých oblastiach. Na jednej strane bude môcť byť daná úloha konzultovaná dostatočne skoro, ak sa objavia nejaké problémy. Na strane druhej sa zvýši kvalita implementácie, keďže ostane viac času na správne overenie riešenia.
- Špeciálne úlohy
    - Ďalšou oblasťou, kde je potrebné zmeniť prístup je vykonávanie špeciálnych úloh, ktoré v súčasnosti pomerne často vynechávame. Patrí sem:
        - Analýza problematiky
        - Testovanie
        - Prehliadky kódu
        - Dokumentácia k určitým uceleným celkom
    - Na začiatku celého projektu sme si povedali, že sa nechceme zamerať na kvantitu, ale hlavne na kvalitu. Dôležité je preto, aby bola každá jedna funkcionalita kvalitne otestovaná, aby bola spravená prehliadka kódu, a aby bola poprípade zdokumentovaná celá problematika.
- Aktualizácia zmien v kóde do TFS (push)
    - Výrazným problémom je taktiež nedostatočné zdieľanie zmien prostredníctvom nástroja git.
    - Aktuálne toto možno nie je až tak veľký problém, keďže každý pracuje na pomerne oddelených častiach. Avšak ak bude potrebné pracovať a mať k dispozícii niektoré nové zmeny bude dôležité aby sa takéto aktualizácie robili čo najskôr po otestovaní.

1.5.3 Nové a potrebné aspekty projektu

- Plánovanie úloh na stretnutí
    - V rámci plánovania sa nám stalo, že sme zle ohodnotili zložitosť jedného z používateľských príbehov. Konkrétne sa ukázalo, že daný príbeh je podstatne zložitejší ako sme si na začiatku mysleli. Nakoniec sa tento príbeh nepodarilo ukončiť v danom šprinte a musel sa presunúť do nasledujúceho.
    - Aby sa predišlo takýmto problémom bolo by dobré, keby sme si jednotlivé používateľské príbehy rozdeľovali na úlohy priamo na stretnutí a následne spoločne ohodnocovali tieto úlohy z hľadiska zložitosti.
    - Takýto prístup môže pomôcť aj lepšiemu pochopeniu daného príbehu.
- Možnosť simulácie
    - Veľmi dôležitou časťou je potreba vykonávania simulácie komunikácie medzi systémom na sledovania pohľadu a výučbovým systémom ALEF.
    - V tomto kontexte je potrebné nasadenie úprav v rámci ALEF implementácie na nejakú online inštanciu tohto systému.
    - Druhou možnosťou je využitie Sandbox inštancie určenej na podobné testovania.
    - Prepojenie s externou aplikáciu je jedným z našich najdôležitejších cieľov v projekte, a preto je potrebné aby bol tento problém čo najskôr vyriešený.

2 Zoznam kompetencii tímu

Peter Kiš

C, C#.Net, Asp.Net, javascript, xml, MySQL, MSSQL, Oracle DB, Unity3Dniekoľkoročná prax s vývojom hier, skúsenosti s vedením tímu - pol roka 2-4 ľudia

Matej Kucek

C/C++, Java, Ruby, Linux, MySQL, xml, html, android development. Vodcovské schopnosti, zodpovednosť, flexibilita, riešenie konfliktov, vo volnom čase vývoj hier

Marek Šulek

C/C++, Java , Linux, JavaScript, MySQL, XML, HTML, android development, skúsenosti s prácou v tíme.

Šimon Valíček

C/C++, Java, HTML, CSS, JavaScript, PHP, MySQL, PostgreSQL, Android. Inštalácia a konfigurácia serverov na platforme Linux. Skúsenosti s prácou v tíme na malých a stredne veľkých projektoch.

Viktória Lovasová

C, java, ruby on rails, html, javascript, bootstrap, mysql. Zodpovednosť, komunikatívnosť

Martin Svrček

C, java, ruby on rails, HTML, CSS, JavaScript, MySQL, XML Menšie skúsenosti s grafickými návrhmi, záujem o User Experience (UX) a vizuálnu stránku softvéru.

Peter Kušnír

C, Java, HTML, CSS, JavaScript, PHP + Zend Framework, MySQL, PostgreSQL, menšieskúsenosti s knižnicami OpenGL a OpenCV, základy tvorby 3D modelov (Blender), kreslenie ako hobby.

3 Metodiky

V tejto kapitole sú zaznamenané jednotlivé metodiky, ktoré vytvorili členovia nášho tímu s cieľom určenia spoločných princípov a postupov pri jednotlivých činnostiach v rámci práce na projekte. Všetky majú za úlohu poskytnúť čo najlepší pohľad na jednotlivé aktivity tak, aby sa vykonávali správne, kvalitne a hlavne jednotne.

3.1 Manažment riešenia chýb

3.1.1 Úvod

Táto metodika je určená pre tím číslo 16 – Focus a popisuje proces zaznamenávania a manažovania chýb prostredníctvom nástroja Team Foundation Server 2013, ktorý ako tím používame. V prípade, že je v projekte GazeTracking/Focus chyba, riadi sa tím touto metodikou.

3.1.1.1 Vymedzenie skratiek

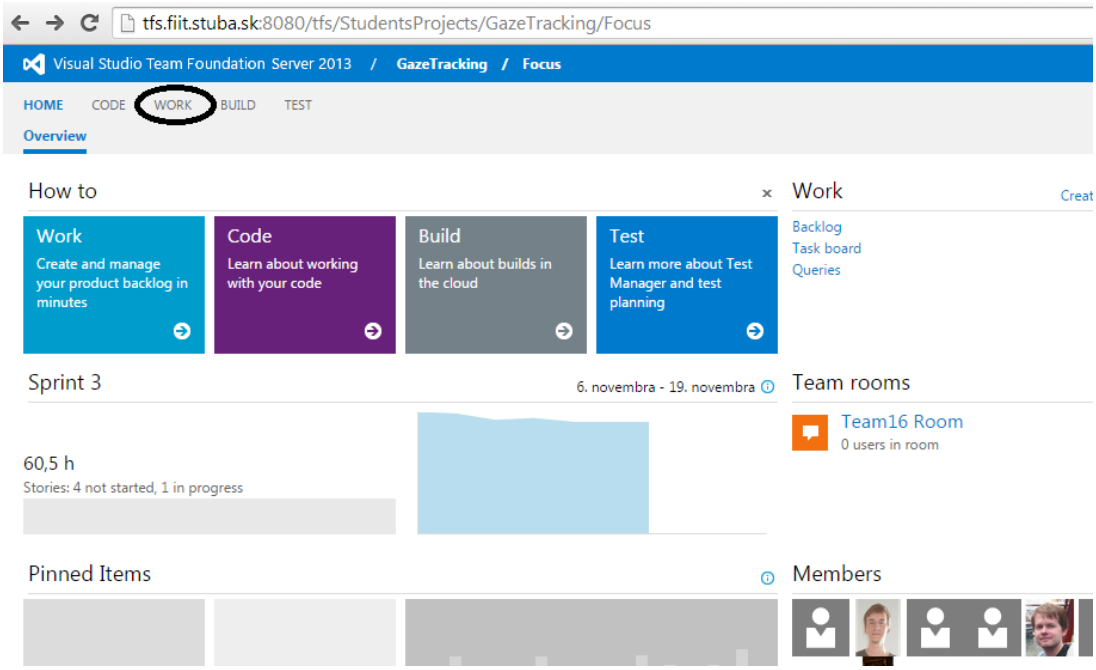
Team Foundation Server 2013 – TFS
Akademický informačný systém – AIS

3.1.1.2 Zoznam nadväzujúcich metodík a dokumentov

- Metodika – Metodika testovania
- Metodika – Manažment verzií
- Metodika – Prehliadky kódu

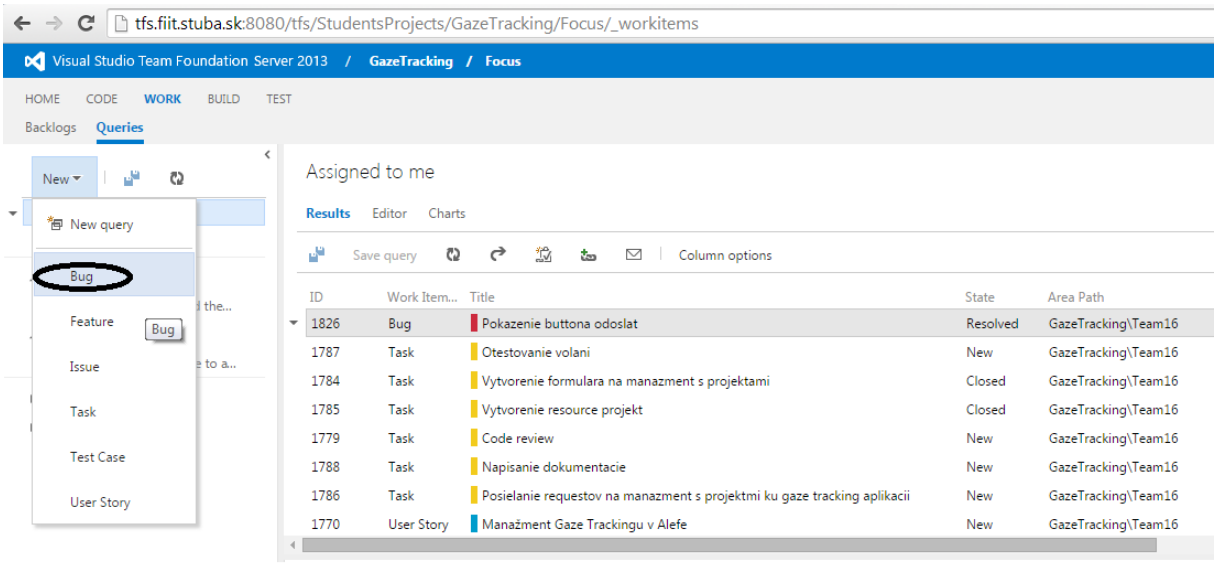
3.1.2 Vytvorenie nového objektu – chyby v TFS

1. Otvorenie internetového prehliadača, napísanie adresy: https://tfs.fiit.stuba.sk/
2. Prihlás sa s prihlasovacími údajmi ako do AIS
3. Klikni na Project collection
4. Vyber projekt GazeTracking/Focus
5. V hornom ľavom menu klikni na Work, pozri Obrázok 8:



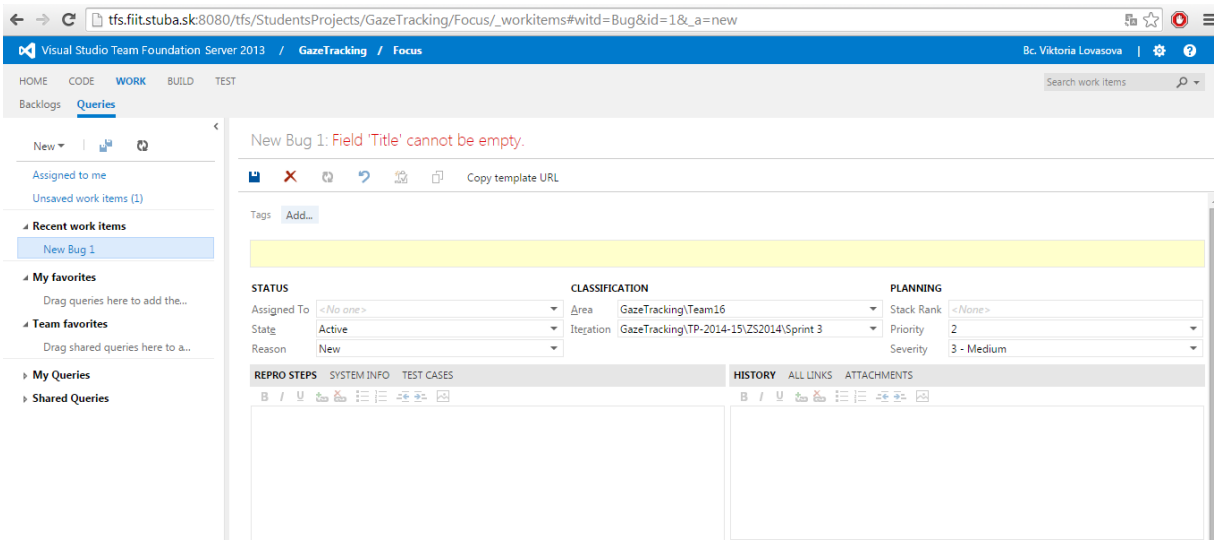
Obrázok 8– Kliknutie na možnosť work.

6. V menu Queries (dopyty) klikni na New (nový) a potom na Bug (chyba), Obrázok 9:



Obrázok 9 – Kliknutie na vytvorenie novej chyby.

7. Zobrazí sa okno na vytvorenie novej chyby, pozri Obrázok 10:



Obrázok 10– Vytvorenie chyby v TFS.

8. Vyplň názov chyby, ktorý ju stručne a jasne vystihuje.
9. V prípade, že chybu budeš opravovať ty, vyber z možností svoje meno, ak nie, tak políčko nechaj prázdne.
10. Stav a príčinu chyby nechaj nezmenené.
11. V prípade, že je chyba závažná zmeň prioritu na 1.
12. Nakoniec dopíš komentár, v ktorom podrobne opíšeš ako chyba nastáva.

Potom ako je chyba vytvorená, príde používateľovi, ktorému bolo vyriešenie chyby pridelené, notifikácia na kontaktný email s informáciami, ktoré boli poskytnuté pri vytváraní chyby. Keď chyba nie je pridelená nikomu, príde notifikácia každému členovi z tímu, pričom si ju musí niekto zobrať na starosti a vyriešiť. V prípade, že si ju nikto nezoberie, pridelí ju niekomu vedúci tímu. Len vedúci tímu môže chybu prideliť inej osobe ako on je sám.

3.1.3 Odstránenie chyby

Aby sme vedeli, či je chyba odstránená a zároveň zvyšovali pokrytie funkcionality projektu testami, najprv si vždy vytvoríme test, ktorý bude chybu simulovať. Následne test spustíme a vidíme, že test padne. Keď chybu opravíme, spustíme test a bude úspešný, budeme vedieť, že sme chybu opravili.

3.1.3.1 Postup odstránenia chyby

1. Vytvorenie testu [1], ktorý simuluje danú chybu
2. Spustenie testu [1], výsledok – neúspešný test.
3. Oprava funkcionality v kóde (odstránenie chyby)
4. Otestovanie, či je chyba odstránená spustením testu, ktorý simuloval chybu, výsledok – úspešný test.
5. Potvrdenie zmien a nasadenie do vlastnej vetvy [2].
6. Nastav stav chyby na vyriešený (resolved).

3.1.4 Zrevidovanie opravenej chyby

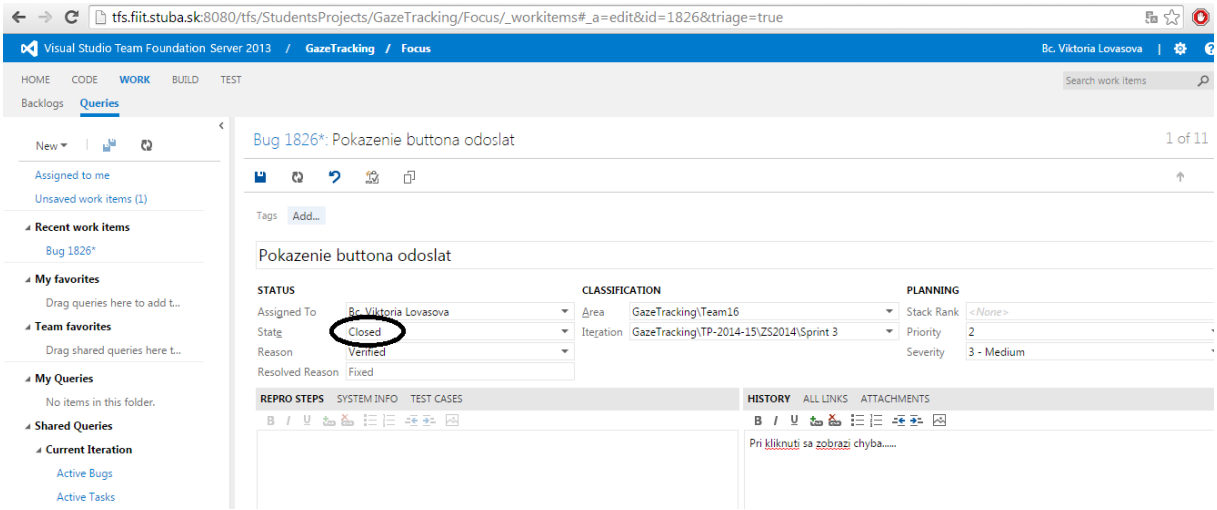
Aby bola daná oprava odsúhlasená aj ostatnými členmi tímu, musí ju prezrieť jeden člen z tímu, ktorý spraví code review, túto metodiku opísal Marek Šukek [3] a následne prezrieť a odsúhlasiť product owner.

3.1.5 Nastavenie stavu chyby ako zatvorený

Ak sa chyba vyrieši a aj tím ju skontroloval, je potrebné zmeniť jej stav na zatvorený.

3.1.5.1 Postup zmeny stavu chyby

1. Klikni na Work, Queries, Assigned to me. Zobrazia sa všetky úlohy, ktoré sú ti pridelené
3. Vyber chybu, ktorej chceš zmeniť stav
4. Otvorí sa nasledujúci formulár, pričom v stave vyber možnosť closed, Obrázok 11:



Obrázok 11 – Zmenastavu chyby na zatvorený.

3.1.5.2 Stavy, v ktorých sa chyba môže nachádzať

Na začiatku, keď sa chyba vytvorí až pokým sa nevyrieši je v stave – aktívny. Do tohto stavu môže prejsť aj chyba, ktorú niekto znovu-otvorí. Keď ju člen tímu opraví, nastaví stav na vyriešený. Aby mohol byť stav chyby zatvorený, musia ju overiť v stave vyriešenom aj ostatní členovia tímu, až potom sa môže nastaviť na zatvorenú.

3.2 Dekompozícia používateľských príbehov na úlohy v nástroji TFS

3.2.1 Úvod

- Proces plánovania v našom tíme
  - Plánovanie práce v našom tíme - agilná metodika SCRUM.
  - Používateľské príbehy (ang. *user story*) si stanovujeme, plánujeme a rozdeľujeme spoločne na začiatku každého šprintu.
  - Ďalšou fázou je rozdelenie pridelených používateľských príbehov na jednotlivé úlohy (ang. *task*).
- Nástroj na manažment v tíme FOCUS
  - Náš tím v rámci agilnej metodiky SCRUM využíva pre manažment v tíme softvérový nástroj *Team Foundation Server* (TFS).
  - Celý proces funkcionálneho dekomponovania jednotlivých používateľských príbehov na úlohy sa vykonáva práve v tomto nástroji.
- Princíp metodiky (komu je určená, čo zahŕňa)
  - Princípom tejto metodiky je poskytnúť postup alebo metódu na spomínané funkcionálne dekomponovanie používateľských príbehov na menšie časti (úlohy).
  - Táto metodika je určená pre všetkých členov tímu FOCUS a budú s ňou pracovať počas celého procesu práce na projekte.
  - Táto metodika zahŕňa celý proces dekompozície používateľských príbehov od vytvorenia základne predstavy na tímovom stretnutí až po samotné zadávanie úloh do systému TFS.
  - Metodika sa však nezaoberá vytváraním používateľských príbehov ani správou úloh počas šprintu (napr. aktualizácia času a stavu úloh).
  - Aby bol celý proces dekompozície kvalitný a hlavne jednotný je potrebné aby sa presne dodržiavali všetky nasledujúce kroky.

3.2.1.1 Prvotná dekompozícia na tímovom stretnutí

- V prvom kroku si na tímovom stretnutí spoločne vytvorte základnú predstavu o aktivitách (úlohách) potrebných pre úspešné dokončenie daného používateľského príbehu.
- Usporiadajte tieto úlohy v čase. Pri ich usporiadaní je potrebné aby ste brali ohľad hlavne na závislosti medzi úlohami, keďže výstup jednej môže byť vstupom pre inú úlohu.
- Následne priraďte ku každej úlohe *story point* na základe princípov metodiky SCRUM.
- Zložité úlohy, ktorým bolo priradené vysoké číslo musíte rozdeliť na menšie časti. Klasickou výnimkou takýchto úloh, ktoré nie je možné rozdeliť sú integračné úlohy.
- V prípade, že sa daná úloha skladá z viacerých aktivít vytvorte druhú úroveň podúloh. Toto však robí každý člen tímu individuálne. Avšak nikdy nerozdeľujte úlohy viac ako na dve úrovne.

Až keď máte takto definovanú a usporiadanú predstavu o tom čo bude potrebné spraviť a kedy to bude potrebné spraviť, môžete pristúpiť k samotnému vytváraniu úloh v TFS.

3.2.1.2 Vysvetlenie použitých pojmov

V rámci metodiky boli použité nasledovné odborné výrazy a skratky:

- Používateľský príbeh (ang. *user story*) – sú základné funkcie, ktoré musí systém poskytovať a predstavujú ucelenú funkcionalitu.
- Úloha (ang. *task*) – je jedna samostatná aktivita, ktorá sa musí vykonať v rámci práce na používateľskom príbehu.
- *Sprint backlog* – zoznam používateľských príbehov a prislúchajúcich úloh, ktoré by mali byť spravené na konci daného šprintu.
- *Story point* – jednotka obtiažnosti, ktorú tím priraďuje k jednotlivým používateľským príbehom alebo úlohám.
- TFS (*Team Foundation Server*) – nástroj pre manažment v tíme používaný aj na zaznamenávanie používateľských príbehov a úloh a ich atribútov.

Celý dokument je rozdelený nasledovne. V kapitole 2 je opísaný celý postup funkcionálnej dekompozície používateľských príbehov na jednotlivé úlohy. V kapitole 3 sú popísané špeciálne udalosti, ktoré môžu nastať pri definovaní úloh. V kapitole 4 sú spomenuté úlohy, ktoré sa často zabúdajú pri dekompozícii uvádzať.

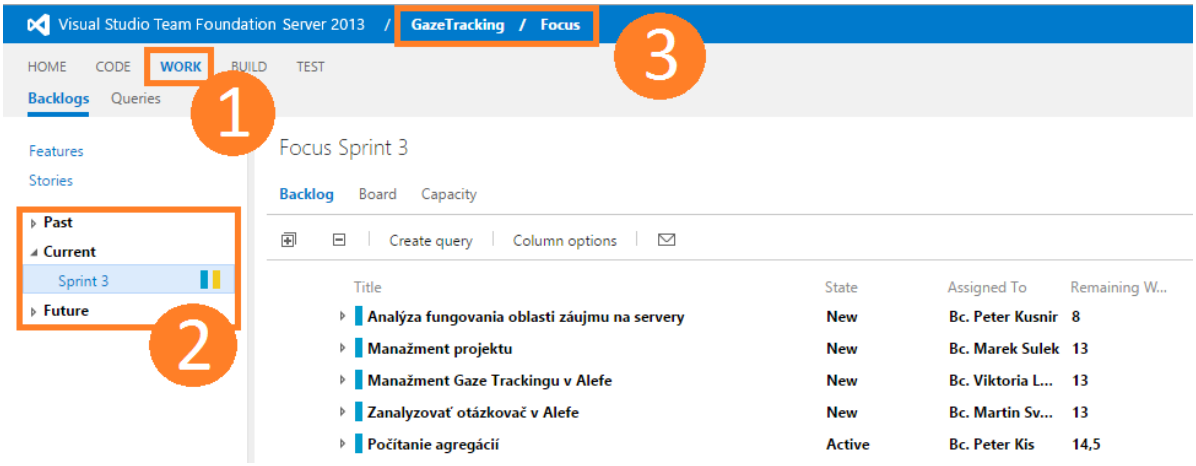
3.2.2 Metóda funkcionálnej dekompozície používateľských príbehov

Táto kapitola bude obsahovať podrobný opis jednotlivých krokov funkcionálnej dekompozície a metód ich vykonania aj s ohľadom na rôzne rozhodovania, ktoré je nutné v tomto kontexte spraviť.

3.2.2.1 Sprint backlog v TFS

V prvej fáze je potrebné mať otvorený adekvátny *sprint backlog* v TFS nástroji (Obrázok 12):

1. Prihláste sa do systému TFS.
2. Kliknite na projekt Gaze Tracking / Focus (bod 3 na Obrázok 12).
3. Kliknite na záložku Work (bod 1 na Obrázok 12) v ľavom hornom rohu obrazovky.
4. Zvoľte si adekvátny šprint, v ktorom chcete vytvárať úlohy. Zoznam šprintov sa nachádza taktiež na ľavej strane obrazovky (bod 2 na Obrázok 12).

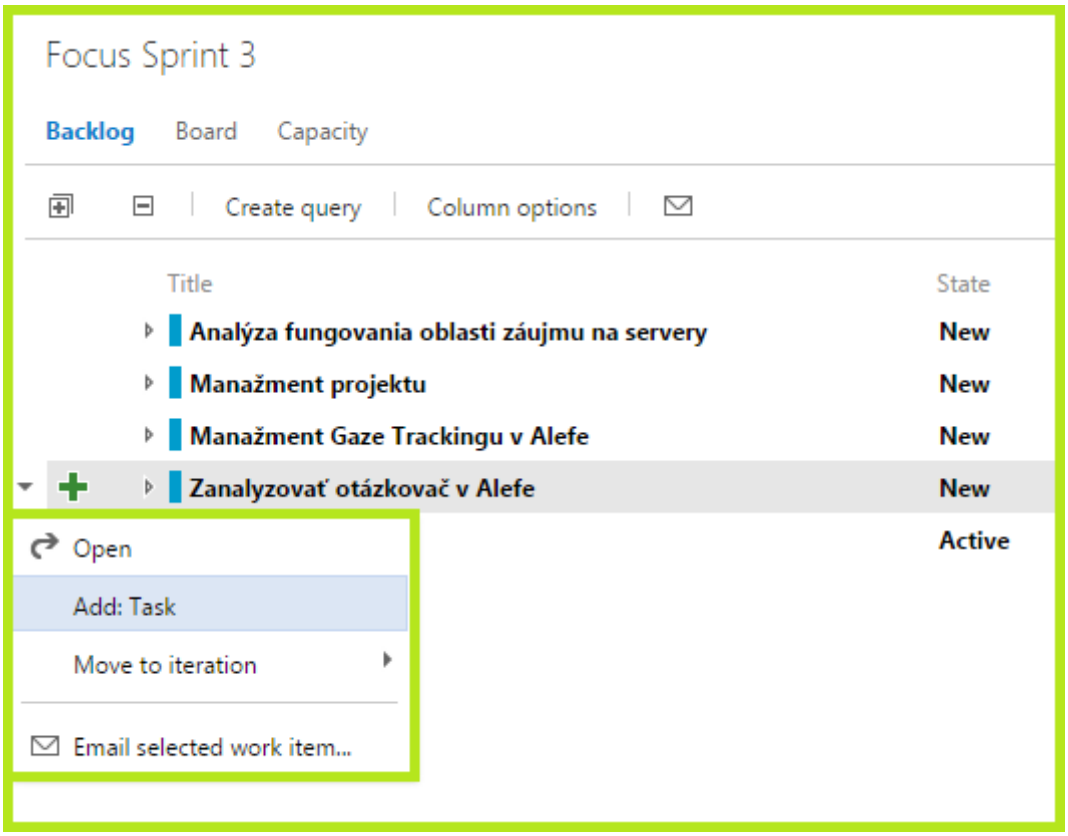


Obrázok 12 - Sprint backlog

3.2.2.2 Prístup formuláru na vytváranie úloh

Pridávanie úloh sa vykonáva prostredníctvom formuláru a preto bude prvým krokom prístup k takémuto formuláru:

1. Označte jeden z používateľských príbehov.
2. Kliknite na tento príbeh pravým tlačítkom myši.
3. Vyberte možnosť Add: Task (Obrázok 13) zo zobrazeného menu (alternatívou pre tento krok je stlačenie zeleného krížiku pri názve používateľského príbehu).



Obrázok 13– Otvorenie formulára

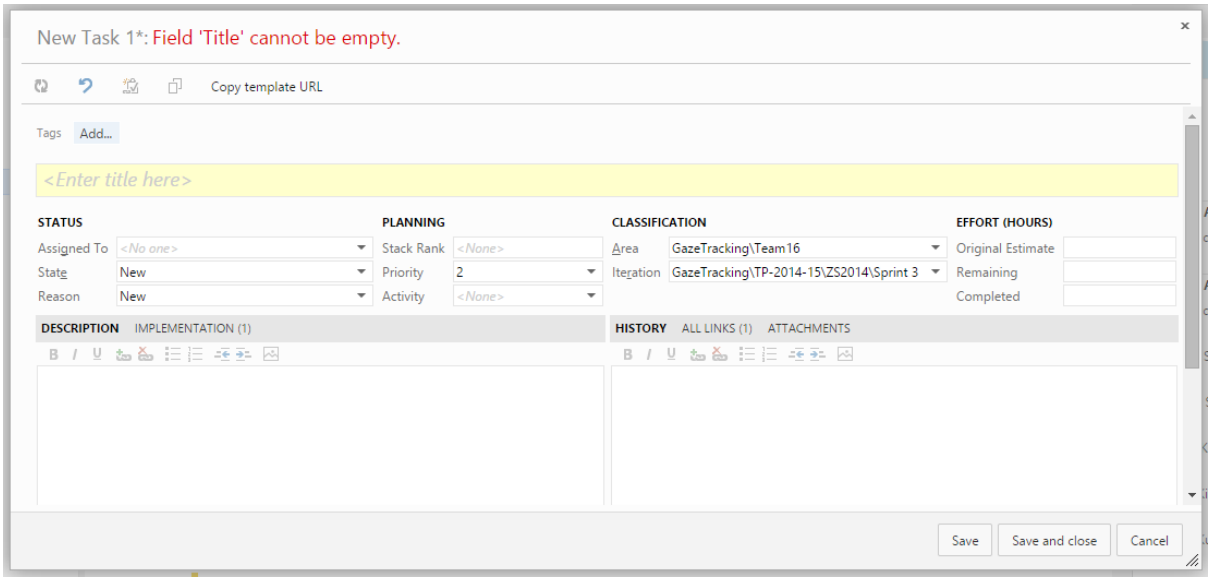
3.2.2.3 Vytváranie úloh

Po úspešnom vykonaní všetkých týchto aktivít sa vám zobrazí formulár pre pridávanie úlohy (Obrázok 14). Vypĺňanie jednotlivých boxov robte v uvedenom poradí aby sa predišlo zlým identifikáciám niektorých atribútov a z dôvodu predchádzaniu chybám, ktoré sa môžu takto včas identifikovať (napr. zistenie, že danú úlohu by bolo dobré rozdeliť na viac častí). Postup:

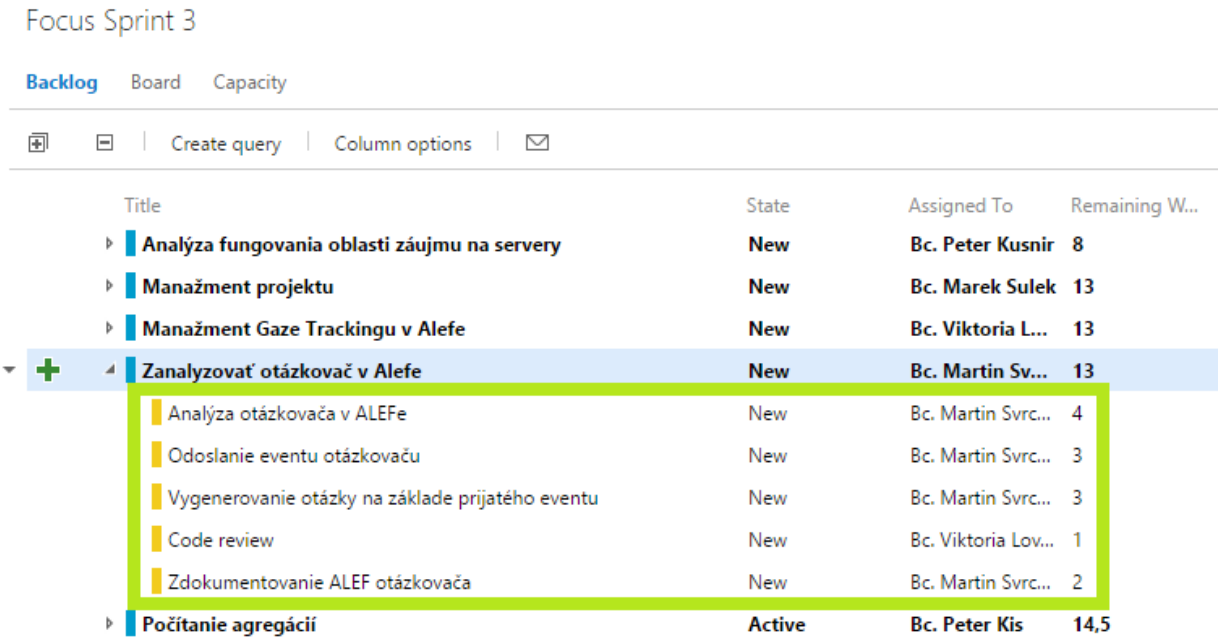
1. Názov (ang. *Title*) – Ako prvé definujte názov úlohy, ktorý sa zadáva do prvého políčka (so žltým pozadím).Tento názov vždy uvádzajte v tvare slovesného podstatného mena (vytvorenie, zdokumentovanie, odoslanie, atď.). Vyhýbajte sa príliš dlhým názvom a názvom so spojkou „a“. Takéto názvy naznačujú, že daná úloha je príliš veľká a musíte ju rozdeliť na menšie úlohy.
2. Tagy (ang. *Tags*) – Stlačte tlačítko Add..., ktoré sa nachádza nad políčkom pre zadávanie názvu úlohy. Pri tagoch vyberte jednu z ponúkaných možností alebo si zadefinujte vlastné tagy:
    a. Klasickým príkladmi tagov, ktoré musí obsahovať každá úloha je typ tejto úlohy (napr. *Analysis, Design, Implementation, Testing*).

   b. Následne pridajte tagy opisujúce technológie, knižnice alebo aktuálne časti softvéru, s ktorými budete pracovať (napr. *coffee script, Backbone JS, database*).
   c. Ďalšími dôležitými tagmi sú tagy predstavujúce kľúčové slová k danej úlohe (napr. *aggregation, AOIs*).
3. Status (ang. *Status*)
   a. Priradený k (ang. *Assigned To*) – Zadajte človeka, ktorý bude pracovať na danej úlohe.V tomto kontexte vždy zadávajte seba samého ako riešiteľa úlohy.
   b. Stav (ang. *State*) – Zadajte stav danej úlohy. Ak úlohu práve vytvárate tak zadajte vždy stav *New*.
   c. Dôvod (ang. *Reason*) – AK vytvárate úlohy tak táto položka bude mať vždy hodnotu *New*. Podobne ako stav má význam hlavne pri správe a aktualizácii úloh.
4. Plánovanie(ang. *Planning*)
   a. *Stack Rank* – Tento atribút podľa dokumentácie[1] predstavuje subjektívne hodnotenie obtiažnosti danej úlohy. Zadefinujte túto položku vo formáte čísla z Fibonacciho postupnosti, ktorá jej podľa vás prislúcha.Je však dôležité aby ste tento atribút zadávali na základe porovnania jednotlivých úloh v danom používateľskom príbehu.Ak do tohto políčka zadáte príliš vysoké číslo, znamená to, že ste zle navrhli danú úlohu a je potrebné ju rozdeliť na menšie časti.
   b. Priorita (ang. *Priority*) – Následne zadajte prioritu danej úlohy vzhľadom na daný používateľský príbeh. Číslo 1 priraďujte najdôležitejším úlohám.
   c. Aktivita (ang. *Activity*)– Pri tejto položke zadajte typ aktivity, ktorá sa bude v rámci danej úlohy vykonávať.
   Podľa typu úlohy vyberte jednu z nasledujúcich možností: Nasadenie - *Deployment*, Návrh - *Design*, Vývoj - *Development*, Dokumentácia - *Documentation*, Požiadavky - *Requirements,* Testovanie – *Testing*.
5. Klasifikácia(ang. *Classification*)
   a. Oblasť (ang. *Area*) – Toto políčko nemusíte zadávať. Tento atribút je preddefinovaný a predstavuje tím, ktorý na danom projekte pracuje.
   b. Iterácia (ang. *Iteration*) – Toto políčko nemusíte zadávať. Tento atribút predstavuje šprint, v ktorom sa daná úloha vykonáva.
6. Úsilie (hodiny)(ang. *Effort (hours)*)
7. Pôvodný odhad (ang. *Original estimate*) – V tomto políčku zadajte pôvodný časový odhad (v hodinách), ktorý bude predstavovať čas potrebný na dokončenie danej úlohy. Ak zadáte pôvodný odhad príliš vysoký, tak je potrebné rozdeliť danú úlohu do menších častí
   a. Zvyšný (ang. *Remaining*) – Následne zadajte zvyšný čas, ktorý vám zostáva aby bola úloha hotová. Pri vytváraní je tento atribút rovnaký ako pôvodný odhad.
   b. Dokončený (ang. *Completed*) – Zadajte koľko času ste už strávili pri práci na danej úlohe. Pri vytváraní úlohy by mal byť tento čas „0“, keďže úlohy sa vytvárajú na začiatku šprintu pred začatím práce.
8. Opis(ang. *Description*) – V tejto časti napíšte stručný opis aktivity, ktorú budete vykonávať a poprípade aj technológie, ktoré využijete v rámci danej úlohy.
   a. Implementácia (ang. *Implementation*) – Túto časť nevypĺňajte pri vytváraní úlohy.
9. História (ang. *History*) – Túto časť pri vytváraní úlohy nevypĺňajte.

10. Po vyplnení všetkých potrebných atribútov skontrolujte všetky informácie a následne kliknite na tlačítko Save and close, ktorým uložíte danú úlohu. Jednotlivé vytvorené úlohy sa zobrazia priamo pod názvom používateľského príbehu (viďObrázok 15).



Obrázok 14– Formulár na pridávanie úloh



Obrázok 15 - Zobrazenie vytvorených úloh v časti Backlog

3.2.3 Riešenie špeciálnych udalostí

3.2.3.1 Odstránenie úlohy

Odstránenie existujúcej úlohy je potrebné hlavne pri zmenách v procese alebo pri nových zisteniach, kedy sa príde na to, že danú úlohu už nie je potrebné riešiť.

1. Kliknite pravým tlačidlom na danú úlohu v časti Backlog.
2. Vyberte možnosť Open(Obrázok 13) v zobrazenom menu.
3. V zobrazenom formulári v časti Status kliknite na položku State a zvoľte možnosť Removed.
4. Do časti History stručne popíšte dôvod prečo bolo potrebné zrušiť danú úlohu.

3.2.3.2 Rozdelenie úlohy na menšie časti

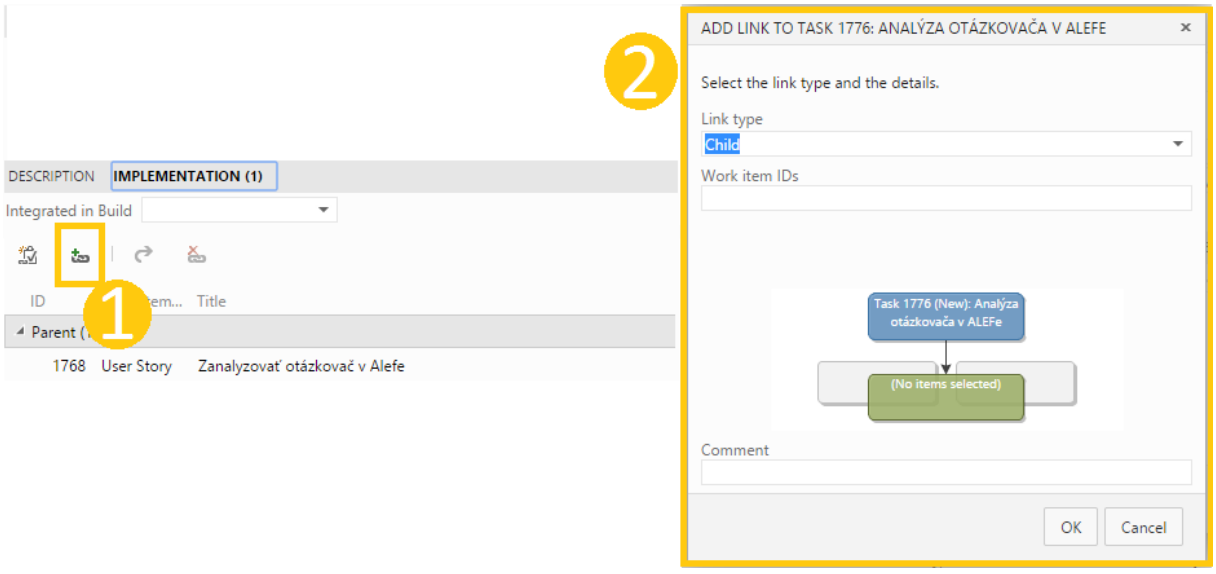
Rozdelenie úlohy je potrebné ak sa v priebehu práce na úlohe zistia nové skutočnosti, ktoré zvýšia zložitosť úlohy.

1. Určite si ako rozdelíte danú úlohu na viacero menších úloh.
2. Vykonajte udalosť *Odstránenie úlohy*.
3. Následne vytvorte novú úlohu podľa postupu v kapitole *2.3 Vytváranie úloh*.

3.2.3.3 Závislosti úloh

Závislosti úloh pridávajte až po vytvorení všetkých úloh v rámci používateľského príbehu.

1. Kliknite pravým tlačidlom na danú úlohu v časti Backlog.
2. Vyberte možnosť Open (Obrázok 13) v zobrazenom menu.
3. V časti Description vyberte záložku Implementation a v nej kliknite na ikonu pridania prepojenia (bod 1 na Obrázok 16).
4. V zobrazenom formulári (bod 2 na Obrázok 16) zadajte do časti Work item IDs identifikačné číslo úlohy, ktorá je závislá na aktuálnej úlohe a pridajte komentár do časti Comment s popisom danej závislosti.



Obrázok 16 - Závislosti úloh

3.2.4 Zoznam bežne chýbajúcich úloh

V tejto časti sú spísané základné úlohy, ktoré často chýbajú pri funkcionálnej dekompozícii používateľského príbehu. Pred tým ako skončíte s dekompozíciou používateľských príbehov tak sa pozrite na tento zoznam a skontrolujte či vám niektorá z nich v zozname úloh nechýba.

1. *Analýza problematiky* – Analýza oblasti, ktorá súvisí s riešením danej úlohy.
2. *Prehliadka kódu* – Kontrola kódu z hľadiska dodržiavania určitých pravidiel programovania.
3. *Testovanie* – Otestovanie vytvorenej funkcionality z hľadiska logiky riešenia.
4. *Integrácia* – Spojenie jednotlivých častí do jedného celku.
5. *Dokumentácia* – Zdokumentovanie práce na danej úlohe alebo používateľskom príbehu.

3.3 Metodika používania softvérového nástroja na správuverzií zdrojového kódu

3.3.1 Vymedzenie pojmov

Softvérový nástroj na kontrolu verzií zdrojového kódu

Týmto pojmom sa myslí taký softvérový nástroj, ktorý umožňuje udržiavať históriu jednotlivých verzií súborov so zdrojovým kódom a zároveň ich synchronizáciu s ostatnými členmi vývojového tímu. V prípade tohoto metodického pokynu bude pojednávané o nástroji GIT (ďalej len "nástroj na kontrolu verzií").

Zdrojový kód

Pod pojmom "zdrojový kód" sa rozumejú súbory, ktoré obsahujú zdrojový kód implementovaného softvérového systému.

Repozitár

Pojmom "repozitár" je označovaná kópia zdrojového kódu, ktorá je spravovaná nástrojom na kontrolu verzií.

Vývojársky tým

Pojem "vývojársky tým" označuje členov tímu "Focus", ktorý spolupracujú na vývoji softvérového systému, ktorého vyhotovenie je cieľom predmetu "Tímový projekt".

3.3.2 Preambula

Tento metodický pokyn je určený na usmernenie práce vo vývojárskom týme a týka sa všetkých jeho členov. Tento dokument určuje pravidlá pri používaní softvérového nástroja na kontrolu verzií zdrojového kódu. Zahrnuté tu sú najmä pokyny k odovzdávaniu a uverejňovaniu zdrojového kódu a manipuláciu s vetvami zdrojového kódu. V rámci tejto metodiky nie sú v žiadnom prípade pokryté pokyny pre vvytváranie a klonovanie repozitárov.

3.3.3 Metodické pokyny k odovzdávaniu zdrojového kódu

Uplatňovanie zmien v nástroji na kontrolu verzií

Zmeny v zdrojovom kóde je nutné priebežne uplatňovať v nástroji na kontrolu verzií, a to aspoň po dokončení implementácie funkcionality alebo odstránení chyby zaznamenanej v nástroji na sledovanie úloh [1], v ideálnom prípade však častejšie.Každá zmena zdrojového kódu uplatnená v nástroji na kontrolu verzií je sprevádzaná informatívnou správou. Táto správa je písaná výlučne v anglickom jazyku a jej maximálna dĺžka je obmedzená na 120 znakov.Sprievodná správa stručne identifikuje dotknutú časť zdrojového kódu a popisuje zmeny na nej vykonané.

Sprievodná správa má nasledovný formát:

[+/-/*] identifikácia dotknutej časti - popis zmien,

kde prvá časť v hranatých zátvorkách značí typ zmeny (+ — pridaná funkcionalita, - — odobraná funkcionalita, * — zmenená funkcionalita), druhá identifikuje dotknutú časť zdrojového kódu a tretia popisuje zmeny na nej vykonané.

Príklad:

[+] auth module - added session timeout check
[-] api module - removed user delete call
[*] api module - improved db connection settings

Uverejňovanie zmien do centrálneho repozitáru

Po každom uplatnení zmien v miestnom repozitári je nutné tieto zmeny uverejniť do vzdialeného spoločného repozitáru.
Zmeny do vzdialeného repozitáru uverejňuje každý člen tímu, prispievajúci do danej vetvy, a to výhradne do vzdialenej vetvy s rovnakým názvom, aký má miestna vetva obsahujúca uplatnené zmeny.

3.3.4 Metodické pokyny k vetveniu zrojového kódu

Vytváranie vetiev

Pred začatím vývoja novej funkcionality je nutné vytvoriť v nástroji na kontrolu verzií novú vetvu zdrojového kódu. Každá vetva zdrojového kódu slúži na vývoj jedného uceleného bloku funkcionality. Do jednej vetvy môžu zasahovať zmenami viacerí členovia tímu, zodpovedný je za ňu však člen, ktorý je zodpovedný za vývoj funkcionality obsiahnutej v danej vetve.
Novú vetvu vytvára člen, ktorý je zodpovedný za implementáciu danej funkcionality a je ju možné vytvoriť výlučne z vetvy zdrojového kódu, v ktorej obsiahnutej funkcionality sa zmeny týkajú. V prípade potreby je možné do takto vytvorenej vetvy prebrať funkcionalitu z ďaľších existujúcich vetiev, je však nutné dbať na stav a funkčnosť zdrojového kódu v požadovanej vetve. Po vytvorení novej vetvy je nutné túto vetvu uverejniť do vzdialeného spoločného repozitáru.
Každá vetva je pomenovaná názvom v anglickom jazyku. Názov čo najvýstižnejšie označuje funkcionalitu, ktorej sa týkajú zmeny zdrojového kódu v danej vetve. Názov je jednoslovný, v prípade potreby je možné medzery v názve nahradiť podtržníkmi. Maximálna dĺžka názvu vetvy je obmedzená na 20 znakov.

Príklad:

auth_module
api_module
frontend

Spájanie vetiev

Po dokonční vývoja funkcionalit v niektorej z vetiev je potrebné tieto zmeny zlúčiť s vetvou, z ktorej bola táto vetva na začiatku vývoja vytvorená. Spájanie jednotlivých vetiev vykonávajú výlučne členovia vývojárskeho tímu, ktorý sú za dotknuté vetvy zodpovední.

Pred samotným spájaním vetiev je nutné prebrať zo vzdialeného repozitáru aktuálne verzie obidvoch dotknutých vetiev.

Pri spájaní vetiev zdrojového kódu môže dôjsť ku konfliktom medzi jednotlivými zmenami vykonanými na zdrojovom kóde. V niektorých prípadoch je tieto konflikty možné odstrániť automaticky za pomoci nástroja na kontrolu verzií. V opačnom prípade je nutné konflikty odstrániť ručne. Pri ručnom odstraňovaní konfliktov je potrebné použiť softvérový nástroj na tvorbu zdrojového kódu [2].
Pri odstraňovaní konfliktov je potrebné dbať na zachovanie funkcionality z obidvoch dotknutých vetiev. O prípadnom vynechaní funkcionality môže rozhodnúť výlučne člen tímu za danú funkcionalitu zodpovedný.
Po dokončení spájania vetiev je nutné na novo vzniknutom zdrojovom kóde vykonať jednotkové testy funkcionality. V prípade úspešného dokončenia testov je potrebné spojenú vetvu uverejniť vo vzdialenom spoločnom repozitári.

3.3.5 Súvisiace zdroje

1. Metodiký pokyn k vytváraniu a správe úloh v softvérovom nástroji na manažment projektov
2. Metodický pokyn k písaniu zdrojového kódu v jazyku C#.

3.4 Metodika prehliadky kódu

3.4.1 Úvod

Účelom tejto metodiky je definovať postup pri prehliadkach kódu, za cieľomaby zachovania dostatočná kvalita zdrojového kódu. Metodika obsahuje postup,ktorý sa musí dodržiavať pri prehliadke kódu a opis práce s nástrojmi,ktoré sa pri prehliadke musia použiť.

3.4.2 Prehliadka kódu

3.4.2.1 Vymedzenie používateľov metodiky

Táto metodika je určená pre všetkých programátorov, ktorý budú písať novýkód alebo budú vykonávať prehliadku zdrojových súborov napísaných v jazykuC# pre project GazeTracking.

3.4.2.2 Súvisiace manuály

Metodika vychádza z nasledujúcich manuálov:

- **CodeReview Používateľská príručka projektu perconik**
- **Používateľské značky pojektu perconik**

3.4.2.3 Použité pojmy

- Perconik - Nástroj na vytváranie a monitorovanie značiek v zdrojovom kóde
- Značka - Príkaz označujúci časť kódu, nesúci v sebe informáciu (z angličtiny tag)

3.4.2.4 Roly zahrnuté do prehliadky kódu

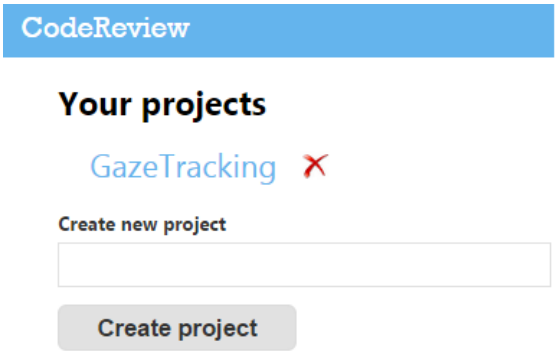
Tabuľka 3 – Roly členov tímu

Rola	Zodpovednosti
Manažér monitorovania projektu	Pridávanie používateľov do projektu v aplikácii perconik Vytýčenie cieľov prehliadok kódu Určenie osôb vykonávajúcich prehliadky kódu Výber používaných nástrojov
Prehliadajúci progrmátor	Identikovanie problému v kóde Zapísanie príslušných značiek do perconic
Autor kódu	Úprava kódu na základe prehliadky Označenie kódu, ktorý je pripravený na prehliadku

3.4.2.5 Prehliadka kódu pomocou nástroja perconic

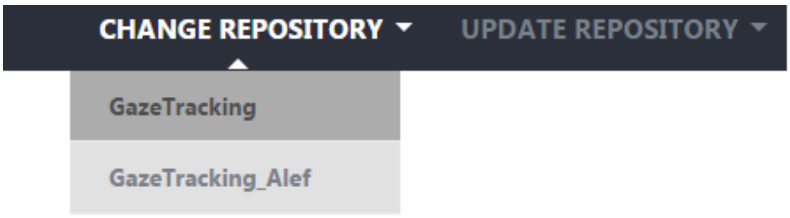
Prehliadka kódu sa uskutočňuje pomocou nástroja perconik, ktorý je dostupný na stránke https://perconik.fiit.stuba.sk/CodeReview/Home/Index. Do systému sa prihlasuje pomocou prihlasovacích údajov akademického informačného systému.

Prvé prihlásenie Po prihlásení kontaktujte Bc. Mareka Šulek pomocou správy v akademickom informačnom systéme na xsulek@fiit.stuba.sk pre pridanie do projektu GazeTracking. Po prihlásení a pridaní vášho účtu do projektu GazeTracking si vyberte projekt zo zoznamu projektov pod nadpisom Your projects (Obrázok 17)

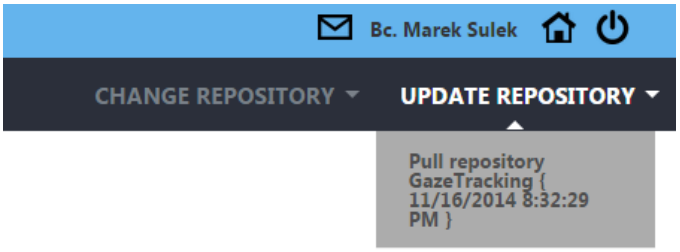


Obrázok 17 - Výber projektu

Výber a aktualizácia repozitára
Po výbere projektu si zvolte repozitár pomocou horného menu. Vyberte CHANGE REPOSITORY a volte GazeTracking (Obrázok 18). V ďalšom kroku si aktualizujte repozitár kliknutím na UPDATE REPOSITORY a následne vyberte Pull repozitory GazeTracking (Obrázok 19).



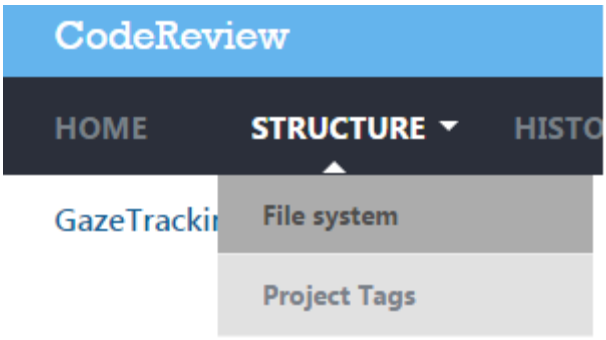
Obrázok 18 - Výber repozitáru



Obrázok 19 - Pull repository

Označenie kódu na prehliadku
Každý kód, ktorý je určený na prehliadku musíte označiť pomocou značiek v nástroji perconik. V menu nástoja perconik vyberte STRUCTURE a následne File system (Obrázok 20). Vyberte vetvu Team16 a kliknite na Change branch!. Pokračujte zobrazením vami vytvoreného kódu, pomocou stromového zoznamu projektu v ľavej časti obrazovky aplikácie

peconik (Obrázok 21). Následne sa zobrazí zdrojový kód. Označte časť kódu, ktorá je určená na prehliadku, zobrazí sa vám kontextová ponuka Add tag. Kliknite na ňu. Z combo boxu vyberte možnosť CODEREVIEW a do popisu napíšte TODO.15



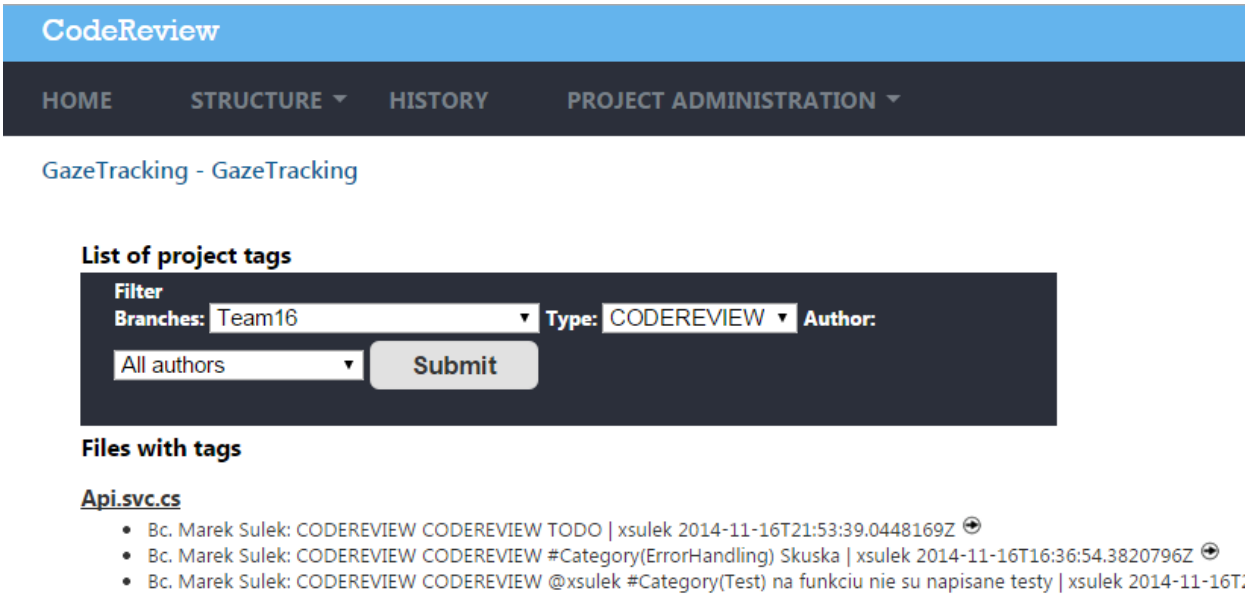
Obrázok 20 - Otvorenie súborov projektu

Vyhladenie kódu na prehliadku

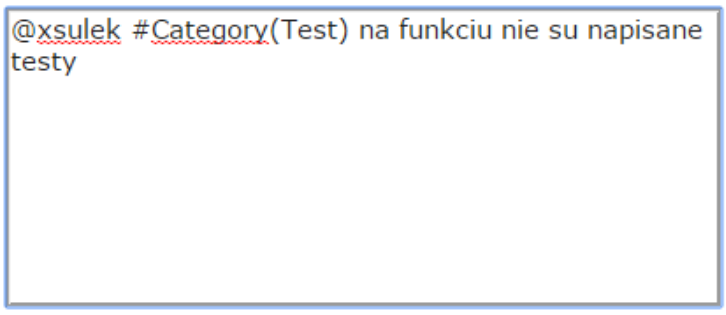
V menu perconik vyberme STRUCTURE a klikneme na Project Tags (Obrázok 20). Vyberieme vetvu Team16 kliknutím na combo box Branches. Následne vyberieme typ značky CODEREVIEW z combo boxu type. Spustíme vyhľadávanie kliknutím na tlačidlo Submit. Zo zoznamu si vyberajte tie, ktoré v sebe obsahuje položku TODO(Obrázok 21). Vyberte z príslušného zoznamu, položku ktorú chcete prehliadnuť. Kliknutím na šípku sa vám zobrazí príslušný kód.

Kontrola kódu - pridanie značiek

Skontrolujte daný kód podľa konvencii ktoré sú v určene v týme. Kliknutím na čierny kruh, ktorý znázorňuje značku a výberom Delete zmažte značku. Ak kód nie je správny, pridajte značku pre označenie nesprávneho kódu. Vyznačte danú časť kódu a kliknite na Add tag. Vyberte možnosť CODEREVIEW. Prvá povinná položka je pre koho je značka určená. Vložte zavináč a vyberte člena týmu, ktorý ma kód opraviť. Napíšte #Category() a do zátvorky vložte atribút, čo je v kóde zlé. Zoznam všetkých atribútov a ich význam je v Obrázok 21. Nasleduje komentár, ktorý podrobnejšie opisuje chybu. V komentári nepoužívajte diakritiku! Príklad je zobrazený na Obrázok 22. Potvrdte Add tag.



Obrázok 21 - Nájdenie súboru v štruktúre



Obrázok 22 - Príklad značky

3.4.2.6 Atribúty

- Category – Kategória pripomienky
- Documentation – dokumentácia; kód je nedostatočne zdokumentovaný
- Name – zlé pomenovanie; premennú/triedu/... je potrebné premenovať
- Threading – porušenie pravidiel práce s vláknami
- Test – ku kódu je potrebné doplniť testy
- Logging – chýbajúce/nevhodné zaznamenávanie udalostí
- Recommendation – odporúčanie na zlepšenie
- Configuration – konštanty by mali byť definované v konfiguračných súboroch
- Logic – chyba v logike riešenia/algoritmu
- Security – porušenie bezpečnostných pravidiel
- API – nevhodné použitie knižnice, služby, …
- DataStructure – nevhodné použitie dátovej štruktúry
- ErrorHandling – poručenie pravidiel spracovania výnimiek a chýb
- Readability – zle čitateľný kód. Napr. z dôvodu použitia príliš dlhýchkonštrukcií, nevhodného formátovania, …

- Style – všeobecné porušenie štýlu písania zdrojového kódu
- Async – zlé spracovanie asynchrónnych udalostí
- Scope – nevhodné použitie modifikátora prístupu
  Perf – možný problém s výkonnosťou
- CVE – typ zraniteľnosti podľa CVE (http://www.cvedetails.com). V prípade,že je známa konkrétna zraniteľnosť, môže byť jej identifikátor uvedený v atribúte CRV.
- DoSo CodeExecution
- Overflow
- MemoryCorruption
- SqlInjection
- XSS
- DirectoryTraversal
- HttpResponseSplitting
- BypassSomething
- GainInformation
- GainPrivilegues
- CSRF
- FileInclusion

3.5 Písanie zdrojových kódov v jazyku C#

3.5.1 Predslov

Táto metodika je určená pre členov tímu Team Focus, predovšetkým však pre tých, ktorí vytvárajú, upravujú a kontrolujú zdrojové kódy v jazyku C#. Metodika zahŕňa pravidlá pre tvorbu názvoslovia, písanie komentárov, formátovanie zdrojových textov a používania dátových typov. Metodika neobsahuje popis syntaxe jazyka C#, návod na písanie optimálneho kódu, ani postupy pri písaní aplikačnej logiky. Popísané pravidlá vychádzajú z oficiálnej príručky písania zdrojových kódov v jazyku C# od spoločnosti Microsoft (C# Coding Conventions). Niektoré pravidlá boli upravené pre potreby tímu Team Focus, nakoľko pri implementácii dorábame do už existujúceho kódu s určitými zažitými pravidlami a nechceme narušiť integritu zdrojových kódov.

3.5.2 Pri tvorbe názvoslovia ...

… sa musí:

- Všetky názvy píšte v angličtine.
- Všetky názvy musia mať samovypovedaciu hodnotu.
- Pre pomenovania tried, metód, konštánt a read only premenných používajte PascalCase notáciu, teda aby všetky slová tvoriace názov objektu začínali veľkým písmenom.
- Pre pomenovania argumentov metód, lokálnych a členských premenných používajte camelCase notáciu, teda aby všetky slová tvoriace názov objektu, okrem prvého, začínali veľkým písmenom.
- Všetky názvy rozhraní píšte s prefixom "I".
  Skratky v názvoch používajte iba ak ide o všeobecne používané skratky, ako sú napr. url, xml, ftp,... alebo ak ide o projektové skratky ako sú napr. aoi, dto.
  Všetky súbory so zdrojovými kódmi pomenúvajte rovnako ako ste v nich pomenovali zadefinovanú verejnú triedu.

… sa nesmie:

- Zásadne nepoužívajte Maďarskú notáciu, teda nedopĺňajte názvy premenných prefixom označujúcim dátový typ.
- Nepoužívajte v názvoch vlastné/vymyslené skratky.
- Neoddeľujte jednotlivé slová tvoriace názov objektu pomlčkami, podčiarkovníkmi

Žiadne názvy nepíšte VEĽKÝMI písmenami.

3.5.3 Pri písaní komentárov

… sa musí:

- Všetky komentáre píšte v angličtine.
- Všetky triedy a ich metódy okomentujte dokumentačnými XML komentármi opisujúcimi ich správanie/funkciu/zmysel.
- Všetky zložitejšie konštrukcie aplikačnej logiky doplňte o komentár, popisujúci funkčnosť danej konštrukcie.

- Všetky zakomentované kusy kódu doplňte o komentár, vysvetľujúci zmysel zakomentovania daného kódu.
- Všetky komentáre obsahujúce značky TODO, HACK, UNDONE doplňte o vysvetľujúci popis.
- Všetky komentáre píšte vždy hneď pred, alebo vedľa zdrojového kódu, ku ktorému sa daný komentár viaže.
- Samotný text komentáru vždy začnite veľkým písmenom

Medzi komentárovú značku a text komentáru vložte vždy jednu medzeruvbvbbvbdf

… sa nesmie:

- Nepoužívajte vhniezdené komentáre.

3.5.4 Formátovanie zdrojových textov ...

… sa musí:

- Na odsadzovanie blokov zdrojových kódov používajte 4 medzery.
- Otváracie a uzatváracie kučeravé zátvorky umiestňujte pri viac riadkových blokoch textu vždy na samostatný nový riadok.
- Deklarácie použitých knižníc musia byť od zvyšku zdrojových kódov oddelené jedným prázdnym riadkom.
- Členské premenné uvádzajte vždy na začiatku tried.
- Členské premenné oddeľujte od metód triedy jedným prázdnym riadkom.
- Jednotlivé metódy oddeľujte od seba presne jedným prázdnym riadkom.
- Jednotlivé bloky kódu obsahujúce ucelenú logiku oddeľujte od ďalšieho kódu jedným prázdnym riadkom.
- Jeden súbor so zdrojovými kódmi musí obsahovať najviac jednu definíciu verejnej triedy.
- Všetky parametre metód oddeľujte od seba čiarkou a medzerou tak, že za čiarka nasleduje vždy za parametrom a medzera nasleduje vždy za čiarkou.
- Vetvenia (if, else if) a cykly (for, while, foreach) oddeľujte od prislúchajúcich guľatých zátvoriek jednou medzerou.
- Každú podmienku pri vetvení obaľujte do guľatých zátvoriek.
- Každý riadok súboru so zdrojovými kódmi musí obsahovať najviac 140 znakov.
- V prípade príkazov presahujúcich 140 znakov rozdeľte tento príkaz do viacerých riadkov tak, že na každý riadok umiestnite čo možno najviac ucelenú časť príkazu.
- Na každý riadok súboru so zdrojovými kódmi píšte najviac jeden príkaz.
- Na každý riadok súboru so zdrojovými kódmi píšte najviac jednu deklaráciu.

… sa nesmie:

- Na odsadzovanie blokov zdrojových kódov nepoužívajte symbol tabulátoru.
- Medzi deklarácie knižníc nepíšte nepoužité/zbytočné knižnice.
- Indexy v hranatých zátvorkách neoddeľujte od hranatých zátvoriek medzerou.
- Guľaté zátvorky neoddeľujte od parametrov medzerou.
- Meno funkcie a k nej prislúchajúce guľaté zátvorky s parametrami neoddeľujte medzerami.

3.5.5 Používanie dátových typov...

… sa musí:

- Používajte preddefinované primitívne dátové typy miesto ich systémových obaľovacích objektov.
- Pri deklarácii lokálnych premenných používajte implicitný dátový typ var.
- Implicitný dátový typ var používajte aj pri deklarácií riadiacich premenných vo vnútri cyklov for a foreach.
- Implicitný dátový typ var používajte aj pre uchovávanie výsledku LINQ dopytov.
- V prípade potreby vrátenia viacerých hodnôt v metódach, používajte zložený dátový typ Tuple.
- V prípade veľkých reťazcov používajte objekt StringBuilder namiesto primitívneho dátového typu string alebo objektu String.

… sa nesmie:

- Nepoužívajte implicitný dátový typ var namiesto primitívnych dátových typov.
- Nepoužívajte implicitný dátový typ var ako dynamický dátový typ.
- Pri deklarácii premenných primitívnych dátových typov nepriradzujte premenným ich prednastavenú hodnotu.
- Zložený dátový typ Tuple nepoužívajte mimo návratovú hodnotu metód.

3.5.6 Ukážkový zdrojový kód

```
using System;
using System.Collections.Generic;
using System.Linq;
using ViewTracking.Infrastructure.Dto.Account;
using ViewTracking.Infrastructure.Dto.Message;
using ViewTracking.Infrastructure.Dto.Project;

namespace ViewTracking.Infrastructure.DataAccessLayer.AggregationRepository
{
    /// <summary>
    /// Contains methods for manipulating aggregations of gaze data
    /// </summary>
    public class AggregationRepository : GlobalRepository, IAggregationRepository, IDisposable
    {
        private readonly ModelContainer AggregationContext;

        /// <summary>
        /// AggregationRepository constructor
        /// </summary>
        public AggregationRepository()
        {
            AggregationContext = new ModelContainer();
        }

        /// <summary>
        /// Get AOI statistics for session between selected frames
        /// </summary>
        /// <param name="AOIId"></param>
        /// <param name="sessionId"></param>
        /// <param name="startFrame"></param>
        /// <param name="endFrame"></param>
        public void GetAOITimeStatistic(int AOIId, int sessionId, int startFrame, int endFrame)
        {
            // LINQ select to get AOI statistics from database
            var statistic = (from aggregation in AggregationContext.Aggregations
                             where aggregation.AreasOfInterestId == AOIId
                             && aggregation.SessionsId == sessionId
                             && startFrame > aggregation.FrameNumber
                             && endFrame < aggregation.FrameNumber
                             group aggregation by aggregation.UsersId into b
                             select new
                             {
                                 userId = b.Key,
                                 duration = b.Sum(d => d.Duration)
                             }).ToList();

            for (int i = 0; i < statistic.Count; i++)
            {
                statistic[i].duration *= 0.9; // TODO optimalisation and refactoring
            }
        }
    }
}
```

Obrázok 23- Vzorová ukážka

3.6 Metodika integrácie

3.6.1 Vymedzenie metodiky

Metodika opisuje postup ako nasadiť doimplementovanú funkcionalitu, poprípade zmenu, na server. Toto nasadzovanie sa týka iba vývojovej vetvy. Je určená pre vývojárov v Týme 16. Postup je opisovaný v prostredí Visual Studio 2013 za použitia techológie Team Foundation Server.

3.6.2 Požiadavky pred začiatkom integrácie

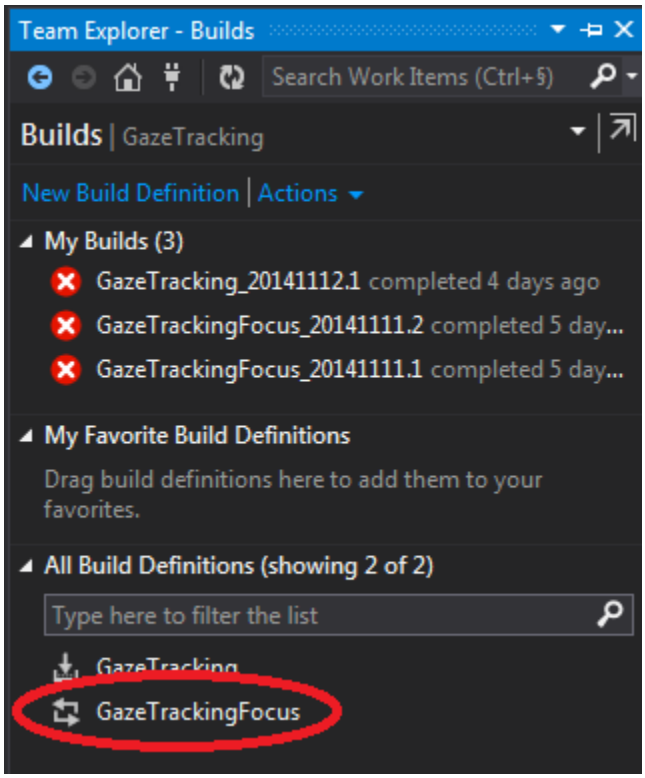
Pred tým, ako začnete integrovať, tak je nutné mať splnené nalsedujúce požiadavky. Ako prvé musíte mať úspešné skompilované lokálne riešenie s najaktuálnejšiou verziou z developerskej vetvy. Ďalej je nevyhnutné mať otestovanú pridávanú funkcionalitu a mať úspšný code review. Správny postup na code review sa nachádza v metodike Focus_Sulek_Metodika_Code_Review.

3.6.3 Postup

1. **Pripojenie vetvy**

   Aby ste mohli napojiť vašu aktuálnu vetvu na jej nad-vetvu, tak musí byť daná vetva najskôr schválená vlastníkom produktu. V našom týme sa vykonáva serverová kompilácia až na hlavnej vývojovej vetve. Vykonáva sa automaticky potom, ako sa na vetve vykoná zmena. Zmena znamená pripojenie vetvy, alebo "commit" zmeny. To znamená, že v prípade, že pripájate vetvu na hlavnú vývojovú vetvu, alebo vykonávate zmenu na hlavnej vetve, tak musíte skontrolovať správnosť a funkčnosť zmeny. Názov našej vývojovej vetvy je TeamFocus.
2. **Kompiláciaa testovanie**

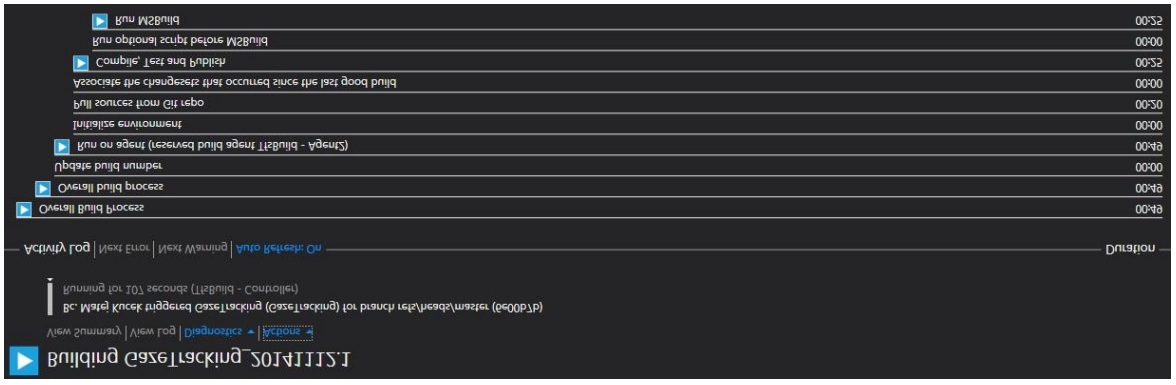
   Ako bolo spomínané, kompilácia sa vytvára automaticky. Taktiež testovanie prebieha automatizovane. Kompilácia sa zaraďuje do poradovníku. V prípade, že chcete kompiláciu spustiť ešte raz tak musíte použiť nasledujúce prednastavené kompilácie vyznačené na *Obrázok 24*.



Obrázok 24 - Prednastavené kompilácie

Stav kompilácie potom sledujte v poradovníku kompilácií, alebo v jej detajlnom popise, ukážka poradovníku sa nachádza na *Obrázok 25*a ukážka priebehu kompilácie na Obrázok 26.

Obrázok 25 - poradovník kompilácií



Obrázok 26 - priebeh kompilácie

Pokiaľ výsledok ukáže, že kompilácia prebehla úspešne, tak je nasadenie novej verzie na vývojovú vetvu serveru úspešné a týmto ste skončili v rámci činnosti. V opačnom prípade pokračujte ďalším bodom.

3. Neúspešnénasadenie

Existujú iba dve možnosti pre neúspešne skompilovanie. Prvou je, že sa nepodarilo skompilovať riešenie a druhou, že nezbehli testy. V oboch prípadoch však platí, že nesmiete odkladať opravenie závady na dlhšiu dobu ako je 1 hodina. V prípade, že to do tejto doby splniť neviete, tak musíte vývojovú vetvu navrátiť späť na poslednú funkčnú veriu.

4. Neúspešné skompilovanie

V prípade neúspešného skompilovania vám server zobrazí nasledujúcu chybovú hlášku (*Obrázok 27*):

Obrázok 27 - oznam o neúspešnej kompilácii

Neúspech môže mať na svedomí viacero príčin:

- Medzi majoritné príčiny patrí chybný kód.
- Ďalšou notorickou chybou môže byť, že pri lokálnej kompilácií boli k dispozícií iné knižnice ako pri serverovej kompilácií. Knižnice môžu okrem toho chýbať, alebo mať inú verziu.
- Lokálna kompilácia prebiehala na inej verzii ako je vývojová. Preto fungovala lokálne, ale nie globálne.
- Rozdielné nastavenia

Ďalšie prípadné chyby sa nachádzjú v tomto odkaze:http://www.timstall.com/2011/12/10-reasons-why-build-works-locally-but.html

Čo robiť po identifikovaní chyby

V prípade, že identifikujete chybu, ktorá nie je charakteru chybného kódu, tak nesmiete samozvane sa snažiť opraviť chybu nahrávaním knižníc, alebo iným zásahom do serverovej časti. Je vyžadované, aby ste v takejto situácií kontaktovali buď vlastníka produktu, alebo manažéra integrácie.

Ak je chyba v zdrojovom kóde tak ju môžete opraviť bez upovedomenia. Odporúča sa však konzultovať s autorom daného kódu, ak ním nie ste vy.

1. **Nespravené testy**

   Globálne najčastejšiou príčinou neúspešných testov býva chybný kód. Okrem toho môže byť problémom nezhoda v lokálnych a serverových testoch. Príčinou môže byť aj niektorá z vyššie uvedených, napríklad, že rôzne verzie knižnice majú obmenenú funkcionalitu rovnakej funkcie. Opäť ako v predošlom kroku, je zakázané zasahovať do knižníc samovolne, ale treba dať upovedomenie kompetentnej osobe, pod ktorou rozumieme vlastníka produktu, alebo manažéra integrácie.

2. **Čo robiť, keď sa chyba objaví až pri behu**

   Častokrát býva situácia, v ktorej kompilácia aj testovanie prebehne v poriadku, napriek tomu, že v kompilácií môže byť chyba. Takáto chyba vypláva na povrch až pri reálnom používaní. Opäť platí podobný postup ako v predošlích prípadoch s oboznámením situácie relevantných osôb. Ak pôjde o častú chybu, ktorá sa už viac krát opakovala, tak to nesmiete nechať len tým, že chybu odstránite, ale musíte buď vy pridať do tejto metodiky postup, ktorý zabezpečí prevenciu danej chyby, alebo vytvoriť test, ktorý by tejto chybe dokázal zamedziť. V prípade, že nie ste kompetentný, alebo z iného dôvodu nemôžete toto vykonať, tak musíte upovedomiť niekoho, kto toto urobí, alebo minimálne takúto prevenciu chyby zdokumentovať v podobe tasku na tfs.

4. Integračné testy

Integračné testy nie su povinnosťou, preto ich zatiaľ ani nemáme vytvorené. Avšak v prípade, že máte pochybnosti o funkčnosti, alebo si myslíte, že integračné testy sú potrebné, prípadne vhodné tak môžete tieto testy navrhnúť a vytvoriť, pod podmienkou, že oboznámite manažéra integrácie, alebo vlastníka produktu o vami pridávaných testoch. Okrem vytvorenia musíte aj zdokumentovať tieto testy do tejto metodiky. Pod dokumentáciou rozumiem opísať možnosť a význam využitia týchto testov.

5. Nasadenie

Nasadenie prebehne automaticky potom ako sa úspešne dokompiluje. Nie je preto nutné vyvíjať žiadnu nadbytočnú aktivitu.

3.7 Metodika testovania

3.7.1 Vymedzenie pojmov

TDD – Vývoj riadený testami (z angl. test driven development) je jeden z prístupov k tvorbe softvéru
AAA – z angl. Arrange-Act-Assert (vo voľnom preklade stanov-konaj-osveč) je vzor bežne používaný na písanie jednotlivých metód unit testov

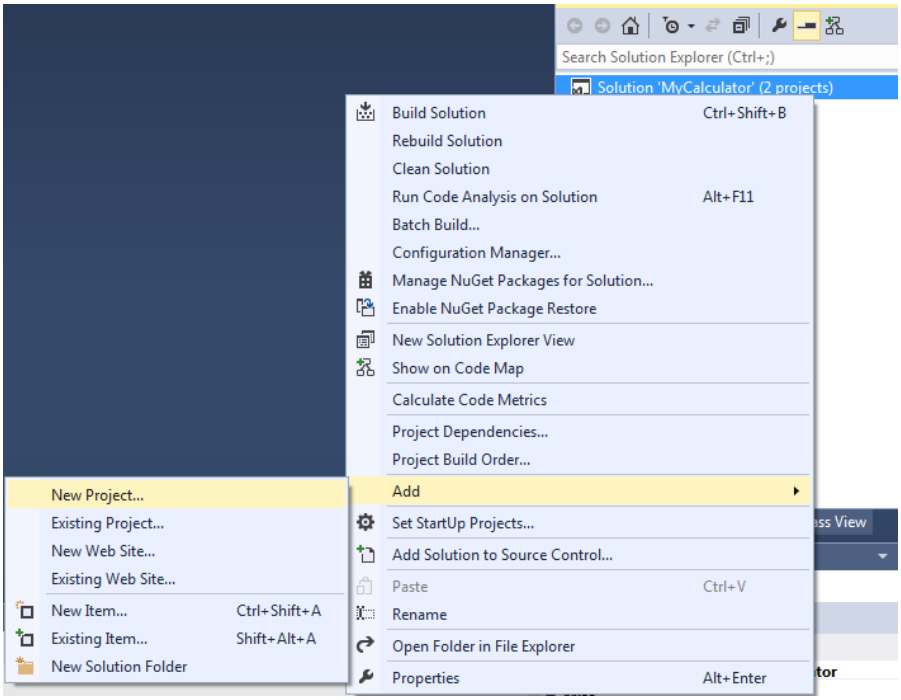
3.7.2 Predslov

Pri vývoji softvéru sa bude tím riadiť metódou TDD, ktorá sa opiera o opakovanie veľmi krátkych vývojových cyklov. Najprv vývojár napíše automatizované testy (na začiatku neúspešné), ktoré definujú potrebu zlepšenia alebo pridania funkcionality. Následne vytvorí minimálne množstvo kódu, ktorým je schopný úspešne zvládnuť vytvorené testy. Tento dokument popisuje proces vytvárania testovacích projektov a testov pre jednotlivé komponenty aplikácie napísanej v jazyku C#. Zachytáva postup tvorby testu, jeho členenie, konvencie v pomenovávaní jednotlivých testovaných častí a spúšťanie testov. Nasledujúce pravidlá sú záväzné pre členov tímu, ktorí sa podieľajú na vývoji častí aplikácie písanej v jazyku C# a využívajú vývojové prostredie Microsoft Visual Studio 2013. Nástrojom na tvorbu testov je Microsoft Unit Testing Framework, ktorý je súčasťou zvoleného vývojového prostredia.

3.7.3 Vytvorenie projektu s testami

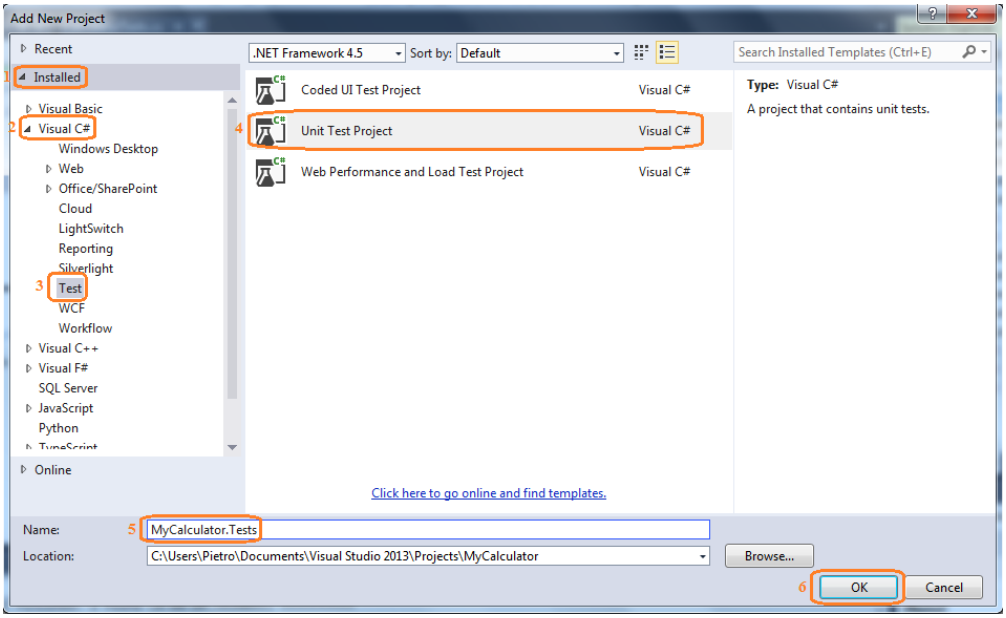
Štruktúra projektu s testami odráža existujúcu štruktúru testovaného projektu. Dbajte na to pri vytváraní hierarchie zdrojových súborov v testovacom projekte.

1. Pre existujúce riešenie (solution), kliknite pravým tlačidlom myši na názov riešenia v prieskumníkovi. V zobrazenej ponuke zvoľte postupne možnosť ***Add***a ***New Project…***



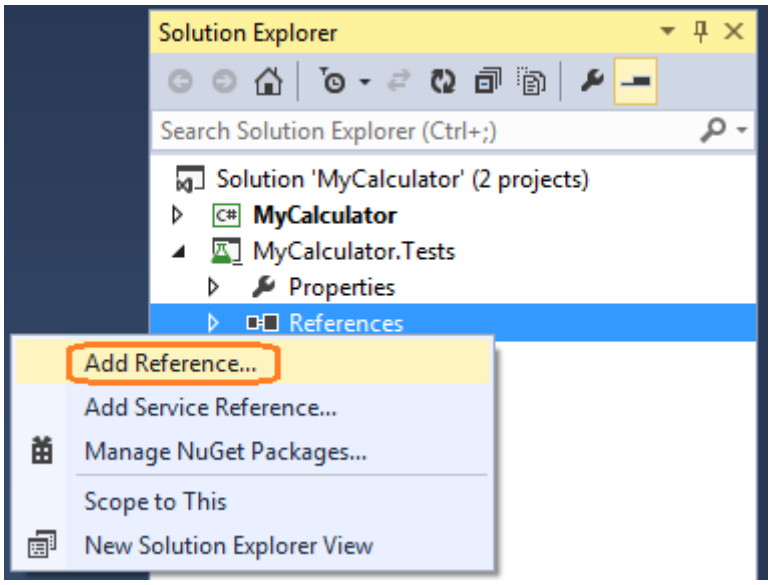
Obrázok 28 – Pridanie nového projektu

2. V zobrazenom okne zvoľte v ľavej časti sekciu ***Installed***. Pokračujte výberom jazyka používaného v testoch, v našom prípade ***Visual C#***.V zvolenej sekcii vyznačte podsekciu ***Test*** a z ponúknutých možností zakliknite ***Unit Test Project***. Pomenujte vytváranýprojekt s testami nasledovne <názov testovaného projektu>.Tests. Dokončite vytváranie projektu kliknutím na tlačidlo ***OK*** v pravom dolnom rohu okna.



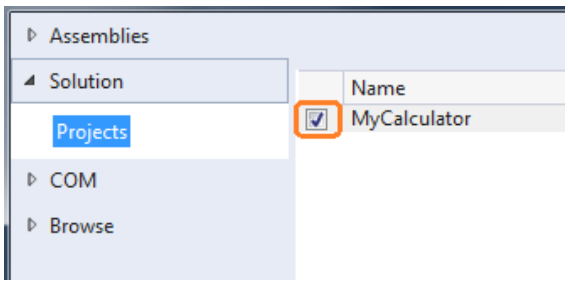
Obrázok 29 – Vytváranie projektu

3. Vo vytvorenom projekte pridajte referenciu na projekt, ktorý chcete testovať. Odkaz na projekt s testami v prieskumníkovi rozkliknite, kliknite pravým tlačidlom na odkaz ***Reference*** a zvoľte možnosť ***Add Reference….***



Obrázok 30 – Pridanie referencie na projekt

4. V ľavej časti zobrazeného okna rozkliknite položku ***Solution*** a zaznačte položku ***Projects***. Z ponúknutých projektov zaškrtnite ten, pre ktorý chcete vytvárať testy. Výber potvrďte kliknutím na tlačidlo ***OK*** v pravom dolnom rohu okna.



Obrázok 31 – Výber projektu na testovanie

3.7.4 Písanie testov

Snažte sa testami pokryť čo najväčšiu časť testovaného projektu, avšak zamerajte sa na fukcie aplikačnej logiky a funkcie nevyhnutné k chodu aplikácie.

1. Súbor s testami pomenujte podľa konvencie <názov testovaného súboru>Tests. Rovnako postupujte aj pri pomenovávaní testovanej triedy, čiže <názov testovanej triedy>Tests. V prípade konkrétnej metódy používajte meno testovanej metódy rozšírené o názov testovanej situácie, teda <názov testovanej metódy><názov testovanej situácie>. V prípade testovania základnej funkcionality metódy použijte ako názov testovanej situácie slovo *Test*.

2. Do testovacieho súboru nezabudnite uviesť direktívu na názvoslovie testovaného súboru použitím kľúčového slova *using*.
3. Nad názov testovacej metódy uveďte atribút metódy *[TestMethod]*. Keďže môže byť na jednu metódu napísaných viac testov, pod atribút [TestMethod] uveďte atribút [TestCategory("<názov testovanej metódy>")] v uvedenom tvare, ktorým budete značiť príslušnosť testovacej metódy k testovanej metóde. V prípade, že výsledkom metódy má byť vyhodenie výnimky uveďte nad funkciu atribút testovacej metódy [ExpectedException(typeof(<názov výnimky s ktorou má skončiť testovaná funkcia>))].

```
[TestMethod]
[TestCategory("Divide")]
[ExpectedException(typeof(DivideByZeroException))]
0 references
public void DivideByZero()
{
    #region Arrange
    var calculator = new Calculator();
    #endregion

    #region Act
    calculator.Divide(10, 0);
    #endregion
}
```

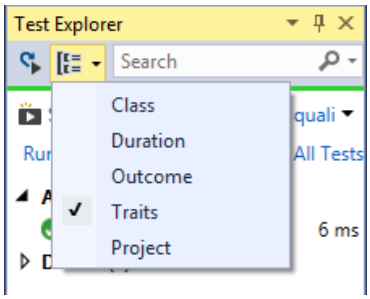
4. Písanie testu na konkrétnu metódu vykonávajte podľa vzoru AAA. Každú metódu rozdeľte do sekcií v poradí *Arrange*, *Act*, *Assert*. Sekcie ohraničte použitím príkazu #region <názov sekcie> a #endregion.
5. V sekcii *Arrange* inicializujte objekty a premenné, s ktorými sa bude v metóde pracovať. Ako prvé dve premenné v sekcii uvádzajte premenné *result* (na konci obsahuje výsledok testu) a *expected* (inicializovaná na očakávaný výstup testu). Vynechajte riadok a uveďte a inicializujte zvyšné objekty a premenné. V prípade, že výsledkom metódy má byť vyhodenie výnimky táto sekcia nebude obsahovať premenné *result* a *expected*.
6. V sekcii *Act* volajte testované metódy so stanovenými parametrami z prvej časti.
7. V sekcii *Assert* overte, či testovaná metóda vracia očakávaný výsledok. Vykonajte porovnanie očakávanej a reálne vrátenej hodnoty pomocou volania *Assert.AreEqual(expected, result);*. V prípade, že výsledkom metódy má byť vyhodenie výnimky táto sekcia ostane prázdna.
8. Pri písaní kódu testov sa riaďte pokynmi a odporúčaniami metodiky Písanie zdrojových kódov v C#[1].

[1]dostupné v kapitole 4 v rámci dokumentu o riadení nachadzajúcom sa na stránke http://labss2.fiit.stuba.sk/TeamProject/2014/team16is-si/ v sekcii Dokumenty.

```
using MyCalculator;

namespace MyCalculator.Tests
{
    [TestClass]
    0 references
    public class CalculatorTests
    {
        [TestMethod]
        [TestCategory("Add")]
        0 references
        public void AddTest()
        {
            #region Arrange
             int result = 0;
             int expected = 3;

             var calculator = new Calculator();
            #endregion

            #region Act
             result = calculator.Add(1, 2);
            #endregion

            #region Assert
             Assert.AreEqual(expected, result);
            #endregion
        }
```

3.7.5 Spúšťanie testov

1. Všetky napísané testy môžete vidieť po skompilovaní v prieskumníkovi testov. V prípade, že prieskumník nemáte zapnutý, kliknite v menu vývojového prostredia Microsoft Visual Studio na položku ***Test***, vyberte možnosť***Windows*** a zvoľte ***Test Explorer***.
2. Keďže boli testy pri vytváraní zaraďované do kategórií, kliknite v prieskumníkovi testov v ľavom hornom rohu na šípku vedľa tlačidla  a zakliknite možnosť ***Traits***. Existujúce testy sa zoskupia podľa priradenej kategórie.



Obrázok 32 – Zoskupenie testov

3. V prieskumníkovi testov kliknite v ľavom hornom rohu na možnosť***Run All***, alebo spustite testy podľa potreby jednotlivo, pravým kliknutím na požadovaný test a výberom možnosti ***Run Selected Tests***.
4. Výsledky testov môžete vidieť po ich vykonanív prieskumníkovi testov. V prípade chyby v niektorom z testov sú podrobnosti o chybe zobrazené v prieskumníkovi testov v časti *Summary*. Ak ste pri tesovaní objavili chybný test k časti kódu ktorej nie ste autorom, vytvorte o tejto chybe záznam. Návod ako postupovať nájdete v metodike Manažment chýb[2].

[2]dostupné v kapitole 4 v rámci dokumentu o riadení nachadzajúcom sa na stránke http://labss2.fiit.stuba.sk/TeamProject/2014/team16is-si/ v sekcii Dokumenty.

4 Export evidencie úloh

1. šprint ukončenie

Tabuľka 4 – Ukončenie 1. šprintu

ID	Work Item Type	Title	Assigned To	State
1520	User Story	WEB stránka		Active
1522	User Story	Funkcia na vracanie API klúča	Bc. Matej Kucek	Active
1523	Task	Code review	Bc. Peter Kis	Active
1524	Task	Vývoj	Bc. Marek Sulek	Active
1525	Task	Vytvoriť	Bc. Peter Kusnir	Active
1526	User Story	Funkcia na získanie API kľúču v ALEFe	Bc. Matej Kucek	Active
1527	Task	Vývoj	Bc. Viktoria Lovasova	Closed
1528	Task	Code review	Bc. Martin Svrcek	New
1529	User Story	Vytvoriť API kľúč		New
1530	Task	Vytvoriť	Bc. Simon Valicek	Closed
1531	Task	Uploadnuť WEB stránku na server	Bc. Simon Valicek	New
1532	Task	Nastaviť notifikácie	Bc. Matej Kucek	Closed
1533	Task	Nastaviť continuous integration	Bc. Matej Kucek	Active
1534	Task	Zistiť ako sa robia reporty z tfs	Bc. Matej Kucek	Active
1621	User Story	Dorobit Web stránku	Bc. Peter Kusnir	New
1622	Task	Dorobit features na vyssej urovni	Bc. Peter Kusnir	New

2. šprint - polovica šprintu

Tabuľka 5–Polovica 2. šprintu

ID	Work Item Type	Title	Assigned To	State

1621	User Story	Dorobit Web stránku	Bc. Peter Kusnir	New
1622	Task	Dorobit features na vyssej urovni	Bc. Peter Kusnir	Closed
1623	User Story	Počítanie agregácií	Bc. Peter Kis	Active
1624	User Story	Zadefinovanie volania v API	Bc. Simon Valicek	New
1625	User Story	Autentifikácia cez API kľúč	Bc. Marek Sulek	New
1626	User Story	Publikovanie eventov	Bc. Peter Kusnir	New
1627	User Story	Odozva ALEFU		Removed
1628	User Story	Pocuvanie eventov cez backbone JS a zvyraznenie elementov	Bc. Martin Svrcek	Active
1629	User Story	Volanie z ALEFu	Bc. Viktoria Lovasova	New
1630	User Story	Pridanie sledovania pohľadu do administrácie	Bc. Matej Kucek	New
1656	Task	Code review		New
1657	Task	Úprava DB štruktúry		New
1658	Task	Vytvorenie agregačných dopytov		New
1659	Task	Analýza existujúcej aplikačnej a databázovej infraštruktúry	Bc. Peter Kis	Active
1660	Task	Vytvorenie aplikačnej logiky		New
1667	Task	Code review		New
1668	Task	Vytvoriť ikonu na prepnutie do pohľadu administrácie	Bc. Matej Kucek	New
1676	Task	Analýza Backbone JS	Bc. Martin Svrcek	Active
1677	Task	Pridanie controllera	Bc. Martin Svrcek	New
1678	Task	Implementácia tela controllera na počúvanie/odchytávanie eventov	Bc. Martin Svrcek	New

1679	Task	Analýza práce s javascriptom	Bc. Martin Svrcek	New
1680	Task	Zvýraznenie elementu, na ktorý sa pozeráme	Bc. Martin Svrcek	New
1681	Task	Code review		New
1682	Task	Napísanie testov	Bc. Martin Svrcek	New
1683	Task	Upravenie autentifikacie	Bc. Viktoria Lovasova	Active
1684	Task	Zdokumentovanie fungovania rozsirenia do Chrome-u	Bc. Peter Kusnir	Active
1685	Task	Implementacia casti pre publikovania eventov	Bc. Peter Kusnir	New
1686	Task	Code review		New
1687	Task	Refactoring existujúceho kódu	Bc. Peter Kis	Closed

2. šprint - ukončenie šprintu

Tabuľka 6 – Ukončenie 2. šprintu

ID	Work Item Type	Title	Assigned To	State
1621	User Story	Dorobit Web stránku	Bc. Peter Kusnir	New
1622	Task	Dorobit features na vyssej urovni	Bc. Peter Kusnir	Closed
1623	User Story	Počítanie agregácií	Bc. Peter Kis	Active
1624	User Story	Zadefinovanie volania v API	Bc. Simon Valicek	New
1625	User Story	Autentifikácia cez API kľúč	Bc. Marek Sulek	New
1626	User Story	Publikovanie eventov	Bc. Peter Kusnir	New
1627	User Story	Odozva ALEFU		Removed
1628	User Story	Pocuvanie eventov cez backbone JS a zvyraznenie elementov	Bc. Martin Svrcek	Active
1629	User Story	Volanie z ALEFu	Bc. Viktoria Lovasova	New

1630	User Story	Pridanie sledovania pohľadu do administrácie	Bc. Matej Kucek	New
1656	Task	Code review		New
1657	Task	Úprava DB štruktúry	Bc. Peter Kis	Active
1658	Task	Vytvorenie agregačných dopytov	Bc. Peter Kis	Active
1659	Task	Analýza existujúcej aplikačnej a databázovej infraštruktúry	Bc. Peter Kis	Closed
1660	Task	Vytvorenie aplikačnej logiky	Bc. Peter Kis	Active
1667	Task	Code review		New
1668	Task	Vytvoriť ikonu na prepnutie do pohľadu administrácie	Bc. Matej Kucek	Closed
1676	Task	Analýza Backbone JS	Bc. Martin Svrcek	Closed
1677	Task	Pridanie controllera	Bc. Martin Svrcek	Removed
1678	Task	Implementácia časti na počúvanie/odchytávanie eventov	Bc. Martin Svrcek	Closed
1679	Task	Analýza práce s javascriptom	Bc. Martin Svrcek	Closed
1680	Task	Zvýraznenie elementu, na ktorý sa pozeráme	Bc. Martin Svrcek	Closed
1681	Task	Code review		New
1682	Task	Analýza testovania v javascripte	Bc. Martin Svrcek	Active
1683	Task	Upravenie autentifikacie	Bc. Viktoria Lovasova	Active
1684	Task	Zdokumentovanie fungovania rozsirenia do Chrome-u	Bc. Peter Kusnir	Active
1685	Task	Implementacia casti pre publikovania eventov	Bc. Peter Kusnir	New
1686	Task	Code review		New
1687	Task	Refactoring existujúceho kódu	Bc. Peter Kis	Closed
1688	Task	Commitnúť zmenu	Bc. Matej Kucek	Closed

1689	Task	Otestovanie	Bc. Matej Kucek	Closed
1690	Task	Overenie API kluca v DB	Bc. Marek Sulek	Active

3. šprint - polovica šprintu

Tabuľka 7 – Polovica 3. šprintu

ID	Work Item Type	Title	Assigned To	State	Tags
1623	User Story	Počítanie agregácií	Bc. Peter Kis	Active	
1656	Task	Code review	Bc. Matej Kucek	New	
1657	Task	Úprava DB štruktúry	Bc. Peter Kis	Closed	
1658	Task	Vytvorenie agregačných dopytov	Bc. Matej Kucek	Active	
1659	Task	Analýza existujúcej aplikačnej a databázovej infraštruktúry	Bc. Peter Kis	Closed	
1660	Task	Vytvorenie aplikačnej logiky	Bc. Peter Kis	Active	
1687	Task	Refactoring existujúceho kódu	Bc. Peter Kis	Closed	
1768	User Story	Zanalyzovať otázkovač v Alefe	Bc. Martin Svrcek	New	
1769	User Story	Manažment projektu	Bc. Marek Sulek	New	
1770	User Story	Manažment Gaze Trackingu v Alefe	Bc. Viktoria Lovasova	New	
1771	User Story	Analýza fungovania oblasti záujmu na servery	Bc. Peter Kusnir	New	
1774	Task	Zdokumentovanie rozšírenia pre Chrome	Bc. Peter Kusnir	New	
1775	Task	Zdokumentovanie serverovej časti	Bc. Peter Kusnir	Active	

1776	Task	Analýza otázkovača v ALEFe	Bc. Martin Svrcek	New	
1777	Task	Odoslanie eventu otázkovaču	Bc. Martin Svrcek	New	
1778	Task	Vygenerovanie otázky na základe prijatého eventu	Bc. Martin Svrcek	New	
1779	Task	Code review	Bc. Viktoria Lovasova	New	
1780	Task	Zdokumentovanie ALEF otázkovača	Bc. Martin Svrcek	New	
1781	Task	Analýza použitia Redis	Bc. Matej Kucek	Removed	
1784	Task	Vytvorenie formulara na manazment s projektami	Bc. Viktoria Lovasova	Closed	
1785	Task	Vytvorenie resource projekt	Bc. Viktoria Lovasova	Closed	
1786	Task	Posielanie requestov na manazment s projektmi ku gaze tracking aplikacii	Bc. Viktoria Lovasova	New	
1787	Task	Otestovanie volani	Bc. Viktoria Lovasova	New	
1788	Task	Napisanie dokumentacie	Bc. Viktoria Lovasova	New	
1789	Task	Code review	Bc. Martin Svrcek	New	
1790	Task	Napísanie dokumentácie		New	
1791	Task	Funkcia api Create project	Bc. Marek Sulek	Active	server
1792	Task	Funkcia api List projects	Bc. Marek Sulek	New	server
1793	Task	Funkcia api Get project by id project	Bc. Marek Sulek	New	
1794	Task	Funkcia api Update project	Bc. Marek Sulek	New	
1795	Task	Funkcia api Delete project	Bc. Marek Sulek	New	

1812	Task	Testovanie	Bc. Peter Kis	New	
1823	Task	Otestovat priepustnost	Bc. Matej Kucek	New	

3. šprint ukončenie

Tabuľka 8 – Ukončenie 3. šprintu

ID	Work Item Type	Title	Assigned To	State	Tags
1623	User Story	Počítanie agregácií	Bc. Peter Kis	Active	
1656	Task	Code review	Bc. Matej Kucek	New	
1657	Task	Úprava DB štruktúry	Bc. Peter Kis	Closed	
1658	Task	Vytvorenie agregačných dopytov	Bc. Matej Kucek	Closed	
1659	Task	Analýza existujúcej aplikačnej a databázovej infraštruktúry	Bc. Peter Kis	Closed	
1660	Task	Vytvorenie aplikačnej logiky	Bc. Peter Kis	Active	
1687	Task	Refactoring existujúceho kódu	Bc. Peter Kis	Closed	
1768	User Story	Zanalyzovať otázkovač v Alefe	Bc. Martin Svrcek	New	
1769	User Story	Manažment projektu	Bc. Marek Sulek	New	
1770	User Story	Manažment Gaze Trackingu v Alefe	Bc. Viktoria Lovasova	New	
1771	User Story	Analýza fungovania oblasti záujmu na servery	Bc. Peter Kusnir	New	
1774	Task	Zdokumentovanie rozšírenia pre Chrome	Bc. Peter Kusnir	New	
1775	Task	Zdokumentovanie serverovej časti	Bc. Peter Kusnir	Active	

1776	Task	Analýza otázkovača v ALEFe	Bc. Martin Svrcek	Closed	
1777	Task	Odoslanie eventu otázkovaču	Bc. Martin Svrcek	Removed	
1778	Task	Vygenerovanie otázky na základe prijatého eventu	Bc. Martin Svrcek	Removed	
1779	Task	Code review	Bc. Viktoria Lovasova	Removed	
1780	Task	Zdokumentovanie ALEF otázkovača	Bc. Martin Svrcek	Closed	
1781	Task	Analýza použitia Redis	Bc. Matej Kucek	Removed	
1784	Task	Vytvorenie formulara na manazment s projektami	Bc. Viktoria Lovasova	Closed	
1785	Task	Vytvorenie resource projekt	Bc. Viktoria Lovasova	Closed	
1786	Task	Posielanie requestov na manazment s projektmi ku gaze tracking aplikacii	Bc. Viktoria Lovasova	New	
1787	Task	Otestovanie volani	Bc. Viktoria Lovasova	New	
1788	Task	Napisanie dokumentacie	Bc. Viktoria Lovasova	New	
1789	Task	Code review	Bc. Martin Svrcek	New	
1790	Task	Napísanie dokumentácie	Bc. Peter Kis	Active	
1791	Task	Funkcia api Create project	Bc. Marek Sulek	Active	server
1792	Task	Funkcia api List projects	Bc. Marek Sulek	New	server
1793	Task	Funkcia api Get project by id project	Bc. Marek Sulek	New	
1794	Task	Funkcia api Update project	Bc. Marek Sulek	New	
1795	Task	Funkcia api Delete project	Bc. Marek Sulek	New	

1812	Task	Testovanie	Bc. Peter Kis	Active	
1823	Task	Otestovat priepustnost	Bc. Matej Kucek	New	
1826	Bug	Pokazenie buttona odoslat	Bc. Viktoria Lovasova	Closed	
1827	Bug	skuska	Bc. Marek Sulek	Active	
1828	Bug	skuska na najdenie chyby		Active	

4. šprint ukončenie

Tabuľka 9-ukončenie 4 šprintu

Project: GazeTracking **Server:** tfs.fiit.stuba.sk\StudentsProjects **Query:** Sprint 4 **List type:** Flat

ID	Work Item Type	Title	Assigned To	State
1656	Task	Code review	Bc. Matej Kucek	New
1823	Task	Otestovat priepustnost	Bc. Matej Kucek	New
1845	User Story	Zobrazenie agregácií v časti Statistics	Bc. Matej Kucek	New
1846	User Story	APIS Sessions	Bc. Viktoria Lovasova	New
1847	User Story	API Users	Bc. Marek Sulek	New
1848	User Story	API AOI	Bc. Peter Kusnir	New
1849	User Story	API Agregácie	Bc. Simon Valicek	New
1851	Task	Analyzovanie kódu	Bc. Matej Kucek	Closed
1852	Task	Implementovanie prepojenia	Bc. Matej Kucek	Active
1853	Task	Implementovanie vizualizácie	Bc. Matej Kucek	New
1854	Task	Testovanie štatistík	Bc. Matej Kucek	New
1855	Task	Prehliadka kodu	Bc. Peter Kis	New
1859	Task	Create AOI	Bc. Peter Kusnir	Active
1860	Task	Update AOI	Bc. Peter Kusnir	Active
1861	Task	Read(Get) AOI	Bc. Peter Kusnir	Active
1862	Task	Delete AOI	Bc. Peter Kusnir	Active
1863	Task	Code Review		New
1864	Task	Dokumentacia	Bc. Peter Kusnir	New
1881	Task	Dokumentácia	Bc. Peter Kis	Closed
1882	Task	Analýza	Bc. Peter Kis	Closed
1883	Task	Implementácia	Bc. Peter Kis	Closed
1884	Task	Testovanie	Bc. Peter Kis	New
1885	User Story	Analyza a implementacia otazkovaca pre API	Bc. Martin Svrcek	Active
1886	Task	Analyzovanie otázkovača v ALEFe	Bc. Martin Svrcek	Closed
1887	Task	Zdokumentovanie otázkovača v ALEFe	Bc. Martin Svrcek	Closed
1888	Task	Implementácia generovania otázky na základe prijatého eventu	Bc. Martin Svrcek	Active
1889	Task	Code review		New
1891	Task	Implementacia	Bc. Viktoria Lovasova	Closed
1892	Task	Testy		Closed
1893	Task	Dokumentacia		Closed
1895	Task	Implementacia	Bc. Marek Sulek	Closed
1896	Task	testy	Bc. Marek Sulek	Closed
1897	Task	documentacia	Bc. Marek Sulek	Closed
1898	Task	code reivew	Bc. Marek Sulek	Active
1907	Task	specifikacia	Bc. Simon Valicek	New
1908	Task	implementacia	Bc. Simon Valicek	New
1909	Task	testy	Bc. Simon Valicek	New
1910	Task	code review		New
1927	Task	Rozbehavanie lokalneho testovania web stranky	Bc. Matej Kucek	Closed

V prílohe B sú umiestnené zápisy z jednotlivých tímových stretnutí.

Príloha A: Testovanie v jazyku javascript - JASMINE

1 Úvod

Čo je JASMIN ?

JASMIN je správaním riadený (behaviour-driven) rámec pre testovanie kódu v javascripte. Je vhodný pre vývoj riadený testami (test-driven development) a má syntax podobnú nástroju RSpec pre testovanie jazyka Ruby. Celý rámec JASMINE je možné stiahnuť na adrese: *https://github.com/pivotal/jasmine/releases*.

2 Základná štruktúra

Predstavme si funkciu v javascripte, ktorá vracia jednoduchý reťazec "Team Focus." :

```
function focusName() {

      return "Team Focus.";
}
```

Ak by sme pre takúto funkciu chceli otestovať jej správnosť test v JASMINE by mohol vyzerať nasledovne:

```
describe("Focus Team Name", function() {
      it("return Team Focus Name", function() {
            expect(focusName()).toEqual("Team Focus.");
      });
});
```

Časť *describe* predstavuje nejaký komponent aplikácie, v našom prípade je to funkcia *focusName()*. "Focus Team Name" je len reťazec a nepredstavuje žiadny kód.

Časť *it* predstavuje špecifikáciu, je to vlastne funkcia, ktorá hovorí čo má daný komponent robiť [1].V rámci časti *describe* môžeme pridať viacero špecifikácií. Časť *expect* predstavuje nejaké očakávanie, ktoré pracuje s hodnotou v zátvorke (v našom príklade návratová hodnota funkcie) a túto hodnotu následne nejakým spôsobom konfrontuje s funkciou nazývanou *matcher* [2] (v našom prípade *.toEqual*) .

3 Matcher

Ako už bolo spomenuté funkcia *matcher* vykonáva boolean porovnávanie medzi aktuálnou hodnotou a očakávanou hodnotou [2]. To znamená, že pred každú takúto funkciu môžeme pridať *.not* ako negáciu (napríklad *.not.toEqual()*) .

Dôležitou črtou týchto funkcií je, že existuje veľa predpísaných funkcií *matcher* ale taktiež si môžeme definovať aj vlastné.

Základnými príkladmi predpísaných funkcií *matcher* sú:

- .toEqual() - porovnanie či sú dva objekty rovnaké.
- .toContain() - porovnanie či jeden objekt obsahuje druhý objekt (reťazce, polia, ...) .
- .toBe() - porovnanie či jeden objekt je ten istý ako druhý objekt.
- .toBeTruthy() - porovnanie či je niečo *true*.
- .toBeFalsy() - porovnanie či je niečo *false*.
- .toBeDefined() - porovnanie či je objekt definovaný.
- .toBeUndefined() - porovnanie či nie je objekt definovaný.
- .toBeNull() - porovnanie či je objekt NULL.
- .toBeNaN() - porovnanie či je objekt NaN (Not a Number).
- .toBeGreaterThan() - porovnanie či je objekt väčší ako druhý objekt.
- .toBeLessThan() - porovnanie či je objekt menši ako druhý objekt.
- .toBeCloseTo() - porovnanie či je číslo blízke inému číslu na stanovený počet desatiných miest.

Vlastné funkcie *matcher* [1]:

```
beforeEach(function() {
     this.addMatchers({
          toBeLarge: function() {
               this.message = function() {
                 return "Expected " + this.actual + " to be
               large";
               };
               return this.actual > 100;
          }
     });
});
```

4 Ďalšie funkcie a črty

Rámec JASMINE má aj veľa ďalších užitočných čŕt, ktoré sa dajú efektívne využiť pri testovaní kódu vzhľadom na špecifikáciu.

4.1 Before and After

Klauzula *beforeEach* sa využíva ak chceme vykonať nejaký kód pred každou špecifikáciou (*it*). Na druhej strane klauzula *afterEach* sa využíva ak chceme nejaký kód vykonať po vykonaní každej špecifikácie.

Príklady [1]:

```
beforeEach(function() {
     employee = new Employee;
});
```

```
afterEach(function() {
     calculator.reset();
});
```

4.2 Zahniezdenie

Jednotlivé bloky *describe(...* môžeme do seba jednoducho zahniezdiť a vytvárať tak rôzne podskupiny. Príklad [2]:

```
describe("A            spec",            function()            {
     var foo;
     describe("nested inside a second describe", function() {
          .   .   .
     });
});
```

4.3 Vynechávanie špecifikácií

Ak z nejakého dôvodu nechceme niektoré špecifikácie testovať, tak ich väčšinou zakomentujeme v kóde. JASMINE však má na tieto prípady vlatné riešenie. Stačí pridať pred každú sekciu *it* alebo *describe* písmeno "x" (napr. *xit* alebo *xdescribe*). Pri testovaní sa táto sekcia automaticky preskočí.

Príkaz *return;* zase preruší testovanie a ďalej už test nebude pokračovať [1]:

```
describe("I'm only going to run SOME of these", function() {
     it("will run this spec", function() {});
     it("will run this spec", function() {});
     return; // This will stop the function
     it("will not run this spec", function() {});
     it("will not run this spec", function() {});
});
```

5 Špióni

Špióni (spies) sú špeciálne črty rámca JASMINE, ktoré dokážu sledovať volania funkcií a aj ich atribúty. Existujú iba v rámci *describe* a *it* blokov a zrušia sa pokaždej ukočenej špecifikácii.

Obsahujú špeciálne funkcie *matcher* ako napríklad:

- .toHaveBeenCalled
- .toHaveBeenCalledWith
- and.callThrough
- and.returnValue
- and.callFake
- and.throwError
- and.stub

Samotného špióna môžeme vytvoriť nasledovne [1]:

```
var distionary = new Distionary;
spyOn(dictionary, "hello");
expect(dictionary.hello).toHaveBeenCalled();
```

V tomto prípade špión nahradí funkciu *hello* triedy *Dictionary*. A následne bude volanie tejto funkcie sledované.

V rámci tohto dokumentu som spísal základné informácie o testovaní javascript jazyka prostredníctvom rámcu JASMINE. Využil som pri tom dva zdroje, kde môžete nájsť viacero ďalších informácií.

6 Zdroje

[1] http://jasmine.github.io/2.0/introduction.html
[2] http://it-ebooks.info/book/2085/

Príloha B: Zápisy zo stretnutí

1 Prvé stretnutie

- Stretnutie otvára Robo a oznamuje nám, že budúci týždeň bude musieť byť stretnutie v iný deň, pretože bude odcestovaný. Skúsi, či si ešte pamätá naše mená.
- Robo chce vidieť náš plagát, Martin Svrček mu ho ukáže na notebooku a Robo je spokojný, hovorí, že môže byť a aj názov tímu Focus sa mu páči.
- Začínajú sa riešiť organizačné veci, Robo navrhuje nech máme nejaké centrálne úložisko, kde si budeme ukladať dokumenty. Šimon odpovedá, že už sme si spravili Google drive. Robo povie, že môže byť a nech ho tam pridáme.
- Robo sa pýta ako okrem emailu medzi sebou komunikujeme, pretože to nestačí. Vie, že máme aj Facebook-skupinu ale radšej by bol keby to bolo niečo ako instant messaging, či niečo také nepoznáme z práce? Šimon navrhuje heapchat.
- Robo nám začína vysvetľovať jednotlive funkcie rolí:
- Náš tím by mal mať obsadene nasledujúce role: Manažér komunikácie, Manažér dokumentácie, Vedúci tímu, Zástupca tímu, Manažér kvality: 2 ľudia môžu byť, jeden bude dozerať nato ako sa budú robiť testy – pomocou tfs-ka, druhý bude mať na starosti code-review, to znamená, že každý kód budú vidieť dvaja ľudia a až potom pôjde na schválenie k Robovi. Tento proces by sa mal zachytiť v nejakej metodike manažérom kvality.
  Manažér pre code versioning – napríklad v Gite, napísať aj postup ako to bude fungovať. Tieto metodiky môžeme využiť potom aj v manažmente.
  Manažér plánovania: v podstate sa plánovanie robí spoločne, najprv sa stanovia featury, tie sa rozbijú na user stories a tie na tasky. Plánovanie sa robí pomoou metódy pokru, kedy spoločne prídeme nato koľko story pointov obsahuje task.
- Manažér rizík a rozsahu – bude robiť aj manažér plánovania.
- Martin Svrček sa opýtal, kto sú naši zákazníci? Robo odpovedal, že máme možnosti:
  - možnosť, že to budú členovia fakulty - napríklad profesorka Bieliková, Michal Barla. Preto sa Robo pýta, či máme záujem o stretnutie s nimi, kde by sme sa dohodli o našich a ich predstavách, aby sme nezačali robiť niečo, čo oni nechceli.
  - plus sú to ľudia, ktorí používajú ux-lab, nejakí diplomanti a mohli by používať náš produkt
  - Robova funkcia je product owner.

- **Rozdelenie rolí**:
  Vedúci tímu – Petet Kiš
  Zástupca – Matej Kucek
  Manažér komunikácie – Peter Kiš
  Manažér kvality a integrácie – kvalita testovania – Peter Kušnír
  - integrácia – Matej Kucek
  - Git- Šimon Váliček
  - code review – Marek Šulek

  Manažér plánovania a rozsahu- Martin Svrček
  Manažér dokumentácie - Viktória Lovasová

- Dohodnutie sa na stretnutí so zákazníkmi, či máme záujem? Súhlasíme, Peter Kiš napíše prvý mail.
- Robo sa pýta, či chceme ísť na TP cup?
  Dve strany treba napísať a poslať do 28.10.2014. Prezentácia bude na iit src – tam nejaké 4 strany po anglicky napísať. Hodnotenie bude na základe prezentácie a aj na základe výstupnej správy. Porota je z praxe a môžeme dostať spätnú väzbu. Záverom je, že sa ešte rozhodneme.

- Prichádza odborný praktikant Mišo a začína nám vysvetľovať projekt, na ktorý my nadväzujeme. Má tri komponenty: informačný systém, chrom-addon, firefox, dve zariadenia na sledovanie pohľadu. Server – obsahuje dve databázy: nosql ravenDB v c# písaná na framework .NET, podporuje JSONy tie sa posielajú, druhá je MSSQL: obsahuje dáta o používateľoch. Eyetracker posiela súradnice očí na server a takisto identifikuje element, ďalej aj focued application, kde sa nachádza kurzor, pracuje sa aj na tom kde sú kliky, takisto aj timestampy. Komunikácia je zabezpečená cez REST operácie. Zvonku je nedostupný gaze data service. Najvyššia úroveň:
  - Prepojenie cez api s aplikáciou ALEF:
    Alef bude komunikovať s našou aplikáciou, bude môcť robiť nejaké akcie: reagovať na základe pohľadu a myši a odporúčať. Toto je najdôležitejšie, lebo to je použiteľné, nahradíme ALEF dáme tam niečo iné a bude to fungovať.
  - Štatistiky a vizualizácia
  - Podpora priebehu samotného testovania
    Zadať pokyny, má nižšiu prioritu ako ostatné dve.

Prepojenie cez api s aplikáciou ALEF

Prebieha diskusia o features. Autentifikácia – posielanie v requeste apikey. ALEF si musí uložiť apikey, aby ho mohol posielať. Potrebné získať aké sú zadefinované objects of interests. Get pre aoi koľkokrát sa tam používateľ pozrel, ako dlho sa tam pozeral, nie len kde sa pozerá, momentálne, xpath tiež môže vracať kam sa pozerá. Šimon to celé zrušil lebo to je zlé pre biznis model. Preto sa bude posielať len Get fix count, fix time -> rel, absolut. Uložiť v alefe apikey treba spraviť, na jeho strane nič nie je. V requeste posielam apikye, project_id, session_id, user_id.

Dohodli sme sa na nasledovných Features:

Features: Ukladanie Apikey v ALEFe
Vytvorenie projektu
Vytvorenie sedenia
Pridanie používateľov k projektu
Ziskavanie dat - aoi agregacie-pohlad,mys
Upozornenie/Prispôsobenie obsahu(odporúčanie)
Štatistika/Vizualizácia

Robo navrhuje aj aby sme to dotiahli až na odporúčania v ALEFe.

Štatistiky a vizualizácia

Features:

Poskytnutie prehľadu – koľko tam je toho nazbieraného, project, session, user,
- počet, čas

Počítanie agregácií – nejaké predrátavanie by tam malo byť

(Vysvetlenie): ak sa niekam pozeral 1 s tak mi vytvor anotáciu(značku) viem si nastaviť aj že 5 sekúnd napríklad dá sa to nastaviť cez rozhranie. Mám objekty A1,A2,A3,…… kam sa pozeral a koľko. Takže ak už robím agregácie tak z týchto anotácií. Takže aj pri prehľade sa dá z anotácií robiť aj agregácie. Priebežne aktualizovať anotácie, zvyšovať počet pozretí aby som mala predpripravené keď príde request. Teda mať tam nejaké countre .

Počet fixácií
Zmena v čase
Trvanie -> relativne
->abolutne
Zachytávanie obrazovky-

(Vysvetlenie): robiť cez screenshot? Alebo video? Alebo stiahnuť stránku obraz(Mirror)

Heatmapy
Graf- poradie kam používateľ šiel
Zoom
Analýza pozerania na video

Šimon navrhuje personalizovanú reklamu – Robo hovorí, že to je super point na prezentáciu do TP cupu, nie len na výučbový systém ale aj na reklamu a bol by tam aj biznis model lebo že aj minulý rok sa sťažovali, že neboli stavané tak, že majú víziu posunúť to ďalej. Minimálne prezentovať aj keď nie realizovať.
Ďalej nasledovalo plánovanie úloh do najbližšieho šprintu. Nabudúce si povieme v akom stave sú úlohy, na konci je retrospektíva, čo by sme mali zmeniť, čo bolo dobre. Potom je plánovanie nového šprintu.

Hotovo je keď:

Ten, kto implementuje hneď spraví aj testy, najprv by mal byť test a až potom implementácia, keď všetky testy prešli, kód spĺňa špecifikáciu, potom to preje k druhej osobe, ktorá bude zodpovedná za code review, dá mu feedback, ak je to ok tak to ide na kontrolu Robovi. Ako sa presne bude robiť code review si povieme nabudúce, podobne ako s gitom to je je tam strom kde viem pridávať komentáre potom do kódu. Peter Kiš sa pýta, či bude code review robiť stále ten istý alebo môžu byť aj viacerí? Robo hovorí, že to môže byť jeden, ktorý bude potom implementovať menej a je dobrý v .NETE alebo budú dvojičky a jeden implementovať a druhý code review robiť alebo pri každej úlohe sa určí kto robí review. Robo hovorí nech je úloh v šprinte menej ale aby sa to stihlo spraviť. Už po prvom týždni to má ísť ku code reviewerovi. K Robovi to môže ísť najneskôr na dokončení šprintu.

Úlohy:

- ALEF- vytvoriť funkciu v rámci gaze api, ktorý si alef vypýta apikey a tá funkcia v api vráti len obyčajné ok, nejaká jednoduchá komunikácia so slovom ok, a v alefe sa zobrazi modálne okno ok. User story: ALEF chce komunikovať cez api.
- Rozdelenie úloh - tasky:
    - ALEF: Forknúť si git repozitár nad našou vlastnou verziou alefu, Rozbehať alef, Volanie cez api, Zobrazenie výsledkov, Testy.

- api: Vytvoriť api kľúč pre ALEF - cez rozhranie spraviť. Vytvoriť volanie
  Support: rozbehať a nastaviť con. integration cez tfs
- stránka tímu- jednoduchá, statická, vždy tam treba dať zápisnicu a dokumenty a plán
- Zápisnica – všetky veci majú byt v tfs, teda aj zápisnica a exportovať sa vždy bude zapisnica z tfs
- Budúce stretnutie bude viesť Viktória Lovasová, pretože písala túto dokumentáciu, povieme si čo sme mali spraviť.
- Štúdium – ako sa pracuje s gitom, .net, prostredie (VS), Ruby on Rails, pozrieť dokumentáciu z min roku ako je zdokumentovane api a databáza. Tieto úlohy spraví každý. Každý, za čo je zodpovedný začať nízko úrovňovo písať metodiku. Testovanie, integrácia a git je dôležité.

- Šimon sa pýta kedy bude budúce stretnutie. Bude vo štvrtok. Presný čas nám ešte Robo dá vedieť. Dohadovanie, kto bude čo robiť:
  stránka – do tfska nahodiť kto je zodpovedný, za task je zodpovedný vždy jeden, stránku robí Martin Svrček a Peter Kušnír.
  Apikey spraviť pre ALEF- Šimon,
- Api vytvoriť funkciu – Marek Šulek, codeReview: Peter Kiš.
- Nastavovanie v tfs – Matej Kucek
- ALEF – Viki a Robo , code review Martin Svrček
- Ukončenie stretnutia.

2 Druhé stretnutie

- Viki spravila funkciu na preposielanie z ALEFu do facebooku, ale nefunguje zatiaľ funkcia na strane Gaze-u.
- Peter Kušnír spravil stránku. Robo ->Big like. Treba prerobiť plán aby tam boli všetky hlavné úlohy a aj plán na semester s termínmi odovzdávania.
- Martin - - Plánovanie sa robí na konci šprintu ako retrospektíva plus ohodnocovanie obtiažnosti úloh.
- Matej - - Tasky v TFS by sa mali podrobnejšie trošku opísať. Nastaviť si notifikácie o úlohách. Ukladať si aj čas, ktorý sme venovali robote. Napísať aj keď sme riešili bugy a nestihli sme spraviť nejaký task.
- Marek - - Komunikácia – ALEF posiela requesty, aby sa identifikovala musí použiť API key. Marekova aplikácia by zachytila volanie a vráti hocijaký text.
  Funkcia už existuje – my potrebujeme aby sa prepojila s nami (vložíme tam URL alef-u). Marekova API musí skontrolovať či ten prichádzajúci API klúč je dobrý. Pri testoch musíme riešiť keď sa neprijme správny API klúč.
  + API key v alefe hodiť do konfiguráku nejakého.
- Peter Kiš – vyriešil komunikačný kanál, dohodol stretnutie so zákazníkmi.
- Šimon pomáhal Marekovi generovať API key --- ďalšie úlohy: nahodiť stránku na web + vyriešiť API.

Ďalšie poznámky
Pozrieť si ako sa testuje v Railsoch aj v C# a RSpec
Po každom šprinte by mala byť každá funkcia otestovaná a nasadená.
Všetko čo robíme by sa malo nasadzovať na našu vetvu a tam to bude bežať a naša práca sa mergne s prácou praktikantov až na konci.
Metodika na integráciu – pozrieť TFS integration.
Treba mať do konca šprintu spravené aj testy a spraviť aj review kódu. Aby sa stihol code review tak musí byť funkcia hotová trošku skorej.
V TFS nový repozitár na ALEF treba vytvoriť.

Code review:

- pozriet aký kod bol pridaný vo Visual(prave tlačitko COMPARE)
- to čo je spravené či to robí to čo má robiť korektnosť a správnosť
- či to je spravené rozumne (konštanty, veci v konfiguráku, premenné) , či by som bol ja spokojný s kódom, zatiaľ to robiť cez komentáre ku tomu čo je zle alebo čo treba zmeniť
- podpora pre codereview v TFS možno vo Visualku , ale zatiaľ iba cez komentáre.
- Ten kto píše kód píše aj testy ale codereviewer následne kontroluje aj testy
- Robiť aj testy na výskyt bugu. – Toto môže robiť code reviewer a potom povie tomu kto to naprogramoval, že na tomto to môže padať.
- Každý by si mal pozrieť ako sa testuje.

Budúce stretnutie – okolo 3 hodiny

- Rozdeliť si nové zadanie úloh pre tím do budúceho cvičenia podľa upravenej prezentácie z prednášky TP.
- Rozhodnúť sa ako to bude s TPcupom.
- Vyhodnotenie šprintu – či to čo sme spravili funguje a či je to správne.
- Povieme čo bolo dobre a čo treba zmeniť.
- Continual Integration – keď ja niečo pushnem tak sa automaticky rozozná zmena a automaticky spustí všetky testy a ak je všetko zelené tak sa to nasadí ale ak nie je všetko zelené tak sa to musí opraviť --- lebo sa môže stať, že našou prácou sme mohli rozbiť alebo ovplyvniť niečo iné a Continual Integration tieto veci odhalí.
- Ak sa odhalí bug v robote predchádzajúceho tímu (lebo ten nemá spravené testy) tak treba spraviť test ktorý bude simulovať tieto bugy.
- Treba robiť refactoring kódu – aj odreportovať, že sme to spravili.
- Vygenerovanie reportu úloh z TFS.
- Product Backlog – treba si ho dopĺňať popri práci – ale nezasahovať do sprint backlogu počas sprintu. Môžeme pridávať úlohy kedykoľvek do product backlogu.

3 Tretie stretnutie

8:30 dostavili sa všetci. Môžme začať.

V TFSku môžme mať viacero repozitárov kvôli ALEFu, dá sa meniť stav tasku commitmi (aj ho zavrieť). O detailoch ešte príde mail.

Zhodnotenie týždňa

Viki: zavolala funkciu z API, ale nemá ešte hotové testy. Robo to kontroluje chmúravým pohľadom (nejak dlhšie to bežalo), ale nakoniec je spokojný.

Mato Svrček: robil code review Viki, kontroloval zdrojáky aj konfigurák, Robo má nejaké výhrady k názvu controlleru (do budúcna funkcionalitu umiestňovať do libky nie do controllera).

Marek: vytvoril nový projekt (solution) na testovanie API, lebo starý kód nebol použiteľný v tomto smere. S pomocou Šimona to nejako spravili, aby to šlo zavolať. Code review -> Peťo Kiš (nebolo veľmi čo opravovať), testy sa nestihli.

Matej: venoval sa nastaveniu continuous integration (Šimon že to donastaví), spravil nastavenie notifikácií, čo sa týka reportov tak k tomu sa mu nepodarilo nič nájsť. Robo k tomu niečo našiel, tak sa na to treba pozrieť. Report -> kto čo robí, v akom stave sú úlohy a čo sa bude riešiť.

Peto Kušnír: dokončenie stránky (plán), task: dorobiť features na vyššej úrovni (termíny odovzdávania, vysokoúrovňový pohľad na veci čo ideme robiť; momentálny stav nie je úplne vyhovujúci, plán je náš backlog, čiže len termíny a fičúrky)

Potrebujeme dohodnúť framework na testovanie pre C# (Šimon navrhuje Nspec, Peťo Kušnír Nunit), Rails je dohodnutý na Rspec.

Došiel Mišo, Šimon sa pýta ohľadom deployovania na server, oni to robia ručne; otázky na API: Šimon rozhodol, že je to nepoužiteľné tak ideme robiť API odznova. Čo sa týka stránky Šimon vybaví novú doménu v priebehu tohto týždňa, aby sa dali stránky odlišovať (stará a nová); chce tam nahodiť ftp.

Záver zo včera: hlavné veci pre tímak sú spraviť základné vizualizácie, začať zbierať viac dát (pohľad, myš, video); chystá sa projekt z UX labu, majú termín, aby bol sfunkčnený lab, možno v budúcnosti by sa dalo zohnať aj lepšie železo; ďalej sa bude riešiť integrácia s ALEFom, tiež boli rôzne nápady, posielalo by to aj nám aj klientovi (APIčku by bolo treba dorobiť server). Posledná vec je poskytovanie údajov učiteľom o tom, ktoré časti sú čítané apod. Mišo odchádza. Ak sa nájde niečo ďalšie, tak máme poslať mail.

Robo navrhol uzavrieť šprint. Každý týždeň by mal byť report a na konci šprintu by mala byť retrospektíva. Celkom zaujímavý spôsob ako to spraviť je, že povieme, čo z toho čo sme spravili už nerobiť, v čom pokračovať a čo začať robiť nové.

Začína Maťo Svrček. Navrhuje, aby sme ako tím lepšie pochopili, ako to celé funguje a navrhuje zaintegrovať tam všetkých členov. Bolo dobré, že sa nejako začalo, a že sa aj niečo spravilo. Navrhuje analýzu kódu, aby sme sa ako členovia lepšie oboznámili s fungovaním kódu.

Viki: nerobili sa testy a malo by sa to zmeniť. Páčilo sa jej, že sme spolupracovali na taskoch.

Peto Kiš: páčilo sa mu, že sme zorganizovali stretnutie so zákazníkmi, Šimon sa ukázal ako "ťahúň".

Maťo navrhuje ešte stanoviť si jasné metodiky.

Matej: páči sa mu atmosféra v tíme, a to že sme sa chválili :). Čo mu vadí, že by mohla byť väčšia aktivita na TFSku, písať si hodiny, čo sa týka vizuálka, tak viac sa oboznámiť s projektom.

Šimon navrhuje kóderské stretnutia, kde by sme spoločne kódili.

Peťo Kušnír: páčila sa mu tímová práca a navrhuje pracovať na reálnych, aj keď jednoduchších taskoch.
Marek: oceňuje prácu v tíme a pomoc od Šimona. Negatívne hodnotí tímovú komunikáciu cez HipChat a odporúča označovať ľudí v komentoch a rýchlejšiu spätnú väzbu. Ešte spomenul, že veľa času strávil na aktuálnom tasku a veľmi mu to nešlo. Navrhuje začať makať na ľahších, ale reálnych veciach.
Šimon: ++ ľudia v tíme a spolupráca, navrhuje začať makať na reálnych veciach.
Robo: ++ že sme dokázali pracovať spolu po stretnutí s prof. Bielikovou, ale máme sa vyvarovať problémom s termínmi (stránka). Treba identifikovať potencionálny problém, aby sme si vedeli vytvoriť časovú rezervu.

Robo navrhuje spraviť lean canvas pre náš tímový projekt do najbližšieho stretnutia a nezabudnúť spraviť veci ešte vrámci tohto šprintu (aktualizovať web,...).

Uzatvorenie šprintu

Metodiky:
Vypracovať a hodiť do google docu kvôli pripomienkovaniu (má to byť tímová verzia, aby s ňou všetci súhlasili)

Založiť dokument (základ pre tímovú dokumentáciu) a na začiatok spraviť analýzu súčasného stavu (tiež ako google doc), aby sa rozobrala súčasná situácia a kvôli lepšiemu pochopeniu fungovania kódu (na úrovni tried)

Rozprava o paneloch: vytvorenie prezentácie o tom ako funguje produkt, ale aj o tíme, ako fungujeme v tíme, ako by to mohlo byť…

Roly:
Plánovanie pre tím a jednotlivých členov tímu: Maťo Svrček (metodika), inak prevažne spoločne v rámci tímu
Vyhodnocovanie plnenia plánu a návrh úprav: v priebehu a na konci šprintu priradiť taskom hodnoty, vytvoriť burndown chart => Maťo Svrček
Organizácia komunikácie v tíme: Peťo Kiš
Udržiavanie informácií o stave projektu: TFSko -> Matej, stránka aktuality -> Peťo Kušnír
Evidencia úloh: TFSko = Matej
Identifikovanie a riadenie rizík: odtrhnuté od reality pri tímakoch; identifikácia rizík, ohodnotenie pravdepodobnosťou, že nastane, cenou škody a navrhnúť riešenia, či im budem predchádzať, alebo ich budem ignorovať; vytvorenie nejakého plánovania akcií tímu (mal by sa na to niekto pozriet a spisať to - prihlásil sa Maťo Svrček)
Zabezpečenie efektívneho znovupoužitia:
Monitorovanie prehliadky vytváraného výsledku: Robo posielal link na code review, Marek má pozrieť prezentáciu z prednášky o code review a vytvoriť projekt v Perconik-u už v tomto šprinte, aby sa robil code review v tomto systéme. Aby to do budúceho týždňa bolo nastavené.
Vyhodnocovanie testov: každý píše testy sám
Identifikácia riadenia chýb v softvéri: ako zalogovať objavený bug, kto ho má riešiť, atd. Spíše Viki

Integrácia softvéru: viac menej hotové
Manažment verzií: Git, ako to používať, ako písať commit správy, ako sa dá automaticky uzatvárať úlohy a priraďovať taskom commity
Požiadavky na zmenu: návrh na zmenu, ako vyhodnotiť požiadavku, ako ju zapracovať,...

TP cup:
národ je rozdelený, prebytočná robota za malý zisk.
Rozhoduje kvalita produktu, kvalita prezentácie, použiteľnosť v reálnom živote, biznis plán.
V zimnom semestri: prihláška, v letnom v marci 4 strany + 2 stranový abstrakt po anglicky, vytvoriť poster na IIT SRC a odprezentovať to, napísať na blog a predstaviť sa ako tím, spraviť video prezentujúce tím (60 sec), finálna prezentácia (povinná aj tak) + ak postúpime do finále tak 1 prezentácia naviac.
Čo nám to dá: skúsenosť súťažiť s niekým, podobné ako startup súťaže, prezentovať nápad a víziu.

Robo by bol za, máme sa rozhodnúť do večera že ako.
Šimon spravil veľmi dobré vysvetlenie aktuálneho stavu projektu s víziou na nasledujúce obdobie.
Budúce stretnutie (backlog rooming).
Rozdelenie úloh na nasledujúci šprint
AGREGÁCIE:

- počítanie agregácií z anotovaných dát -> vytvoriť tabuľku v MS SQL, rozšíriť workera (pri pridaní anotácie nech sa inkrementuje príslušný counter) - [zdokumentovanie fungovania Raven DB a aj workera]

API

- zadefinovanie volaní API -> identifikovanie základných call-ov (cez JSON) [vytvoriť projekt, vytvoriť session {v podstate podobné veci aké sú dostupné cez web}]
- autentifikácia cez API kľúč (cez databázu)
- volanie z ALEFu (skryť do libky), premenovať controller
- administrácia -> sledovanie pohľadu

REAL TIME

- add-on (Chrome) -> Publisher -> Event(x,y) OR Event(XPath)
- ALEF -> Subscriber (JavaScript) [format Publisher-Subscriber] ->handler (odozva ALEFu), AJAX
- Publikovanie Eventov -> JQuery $('event').ontrigger(...); pozrieť BackBoneJS framework; pridanie GazeTrackingController-u, ktorý by obsluhoval eventy ktoré prídu z JQuery
- počúvanie eventov cez backboneJS (spojiť so zvýrazňovaním elementov)

ODOZVA ALEFU

- zvýrazňovanie elementu v ALEFe, na ktorí sa užívateľ pozerá

Ohodnocovanie úloh:
Autentifikácia cez API kľúč (20) - Marek
Volanie z ALEFu (3) - Viki
Administrácia (sledovanie pohľadu) (3) - Matej
Zadefinovanie volaní API (5) - Šimon
Agregácie (20) - Peťo Kiš

Publikovanie eventov (13) - Peťo Kušnír (pozrieť si dokumentáciu k addonu a skúsiť spísať dokumentáciu, aby sa vedelo do budúcna)
Počúvanie eventov + zvýraznenie (13) - Maťo Svrček

Doplniť odhady časov (najlepšie ešte dnes).
Všetci sú už hladní a je tu zima. Je 12:15. Končíme a ideme jesť.

4 Štvrté stretnutie

Peto Kušnír: spravil publikovanie eventov, neporadilo sa nastavit extension. Identifikoval si miesto. Nestihol dokončit úlohu, bol v tom problém. Dnes to dokončí.
Treba spraviť dokumentáciu ku každému tasku, aby ostatný vedeli sa v tom orientovať.
Viktoria – nevedela presne ako na to. Neporadilo sa jej uplne dokončiť task.
Martin Svrcek - Mal robit zvýraznenie eventov. Podarilo sa mu to. Robo mal ešte pripomienky k tomu. Urobil analyzu testovania.
Peto Kiš - nepodarilo sa mu uplne dokončiť task. Odhaduje to ešte na dosť dlho.
Mato Kucek - Podarilo sa mu splniť task. Najviac sa mu zabralo rozbehať ALEF.
Marek Šulek - Air spec treba. Dokoncil task ale nespravil testy.
Šimon - Spolupracoval na tasku s Marekom, čiastočne zadefinoval API, dnes to dorobí.

Zhrnutie šprintu
Peto Kušnír - Navrhuje stretnutia kde sa spoločne programovalo.
Viki - tiež je za spoločné programovanie
Martin - Spokojny.
Peto - Jeho uloha bola, zle odhadnuta, venoval jej vela času, potrebuje spolupracovníka. Jeho uloha bola pre viacerich ludi.
Matej - Robilo sa mu dobre lebo dostal dobry support.
Robo - Vela času stále zaberá informácie, treba si zdielať informácie.
Marek - Nespokojny s efektivitou svojho programovania,

Šimon ukazoval navrhnuté **API. R**eqesty budu vo formate post. Responsy vo forme JSON.
Agregacie - Peťo navrhol ako vylepšiť databázu. Dohadovali sme sa ako bude navrhnutá databáza.
Real time komunikacia -
Alef

Pridelenie uloh
- Marek - Pridanie funkcianality do API, support od Šimona - 13 story points
- Šimon - Dozerat a pomoct nad marekom a Petom.
- Martin - Analyzovanie otazkovaca v ALEFE - 13 points
- Matej - Manazment a vytvorenie gazetracking projektu v ALEFE, volat funckie API
- Peto - Analyza
- Dokumentaciu, musi kazdy spisat.

5 Piate stretnutie

Peter kiš - robili s Matejom dopyty. Vytváral testovacie data, Agregacne dopyty
Robov komentar - otestovat z pohladu priepustnosti, kde su limity riesenia, skusit si cez simulator aspon od jedneho cloveka ci to bude stihat. Musite to niekde nasadit.

Matej Kucek - Nevie sa pripojit, Nedostal sa k analyze redisu, Robil agregacne dopyty
Robov komentar - heslo je nepoviem, spravit si novy ucet cez ktory to budeme testovat, Redis nie je uz relevantny,

Peto Kusmir - dorobit ulohu, zacal so serverovou castou, nestihol vsetko
Robov komentar - ked si nie sme isty treba dat komentar, mozme sa poradit s minulymi vyvojarmi, zakomentovane kody su uz asi nepotrebne, treba ich mazat a komentovat na urovni

Simon Valicek - pokracuje v praci, treba doriesit deploy a stranku, skusi to dnes pripomenut,
Robov komentar - treba to doriesit, spristupnit stranku,

Marek Sulek - dorobil funkcie z minuleho tyzdna, pripravil si funkcie,
Robov komentar - oddelit zobrazenie od kontrolera, pytal sa ohladne code review ci su vsetci v perconiku.

Martin Svrcek - dorobil view, opisal svoje problemy

Viktoria Lovasova - spravila manazovanie projektu gaze tracking, ale este to nekomunikuje a je to nekonzistentne jazykovo.

Robo rozpraval o buducnosti alefu.
Dalej rozprava o obsahu dokumentacie, akym cinnostiam sa manazer venoval, podiel prace vyzera ako jedna tabulka kde su vyjadreny pomer kto sa ktorej kapitole venoval. Aplikácie manažmentu tam opiseme jednotlive cinnosti, v strucnosti zhodnotit a odkazat sa na prislusnu metodiku. Sumarizacia sprintov - burndown chart, okomentovany preco tak vyzera, opisat ak ofungovalo planovanie a priebeh. Dalej zhodnotenie co bolo dobre co by sme chceli stihnut.

Diskusia o metodikach.

6 Šieste stretnutie

Peter Kušnír:
-moderoval stretnutie

Peter Kiš:
-analyzoval, ako aplikacia komunikuje s RavenDB
    -implementacia je neprehladna, nepouziva orm ale linq
    -obsahuje vela zakomentovaneho kodu-nvieme, ci je nefunkcny alebo nepotrebny

Robo:
-v pripade potreby je nutne urobit markup existujuceho kodu

-pri implementacii noveho kodu treba kod komentovat a generovat z neho dokumentaciu
-treba pouzivat db migracie

Matej:
-nastavoval query v tfs
-dorobit do alefu nove view v administracii
    -rozbehal si alef
    -nema v nom testovacie data

Robo:
-treba sa pridat do redminu alefu
-vie pomoct s vysvetlenim implementacie alefu

Marek:
-studoval rest sluzby

Robo:
-treba si pozriet ako sa v sucasnosti realizuju dopyty na db

Viki:
-upravila api call z alefu
-vytvorila testy, ale ich funkcnost je zavisla na implementacii api
-este mala napisat metodiku testovania, co nestihla

Robo:
-testy by nemali priamo volat api ale mali by sa napisat mocky
-toto iste treba aplikovat pri testovani funkcii, ktore vyuzivaju model
-treba si pozret dokumentaciu k Rspec
-pri zisteni chyby pri testovani je dobre vytvorit novy task v tfs

Martin:
-robil analyzu implementacie zachytavania eventov zverejnenych browser extensionom pomocou backbone.js
-pisal metodiku riadenia scrum

Robo:
-treba si pozriet repozitar alefu, kde je mozne vidiet pouzitie backbone.js
    -backbone.js vyuziva model, ktory komunikuje so serverom
    -view, ktory zobrazuje data

Šimon:
-nova domena
-aliasy + deploymenty pre novu domenu
-vytvorenie kopie db
-rozpracovanie api

Robo:
-v db nie su az tak hodnotne data
-specifikaciu api je nutne stihnut dokoncit do konca sprintu
-v 9. tyzdni sa odovzdava priebezna dokumentacia k projektu, riadeniu a big picture

-big picture
-opis povodnej myslienky sluzby
-architektura
-moduly
-prepojenie
-komunikacia
-vystupy analyzy jednotlivych modulov
-sucastou dokumentacie vysledky testov modulov
-dokumentacia riadenia
-big picture
-role clenov a podiel prace
-zodpovednosti na systeme jednotlivych clencov
-aplikacie manazmentov
-opis jednotlivych sprintov
-na testovanie budeme pouzivat Nspec
-v ramci tp cupu budu k dispozicii mentoringy
-prezentacia/business
-treba vytvorit a vytlacit lean canvas
-na code review treba pouzivat perconik
-Robo sa ozve Karolovi Rastocnemu, aby nam v nom vytvoril projekt

7 Siedme stretnutie

Viktoria Lovasova
naimplementovane manazovanie projektov
testy nie su
Peter Kis
stihli sa agregovane dopyty, je to otestovane
zacal sa robit manazer, ktory zbiera data a vytvara agregacie
testovat cez simulator, ale on nie je prepojeny s anotaciami
Peter Kusnir
vytvoril dokumenty na zdokumentovany kod
treba otestovat nejako
Simon
bojoval so serverom, pisal na fakultny mail ale neodpisali
nemal iny task
Marek Sulek
spravil tasky localne
nema testy
ani code review
Martin Svrek
pocuvanie v alefe javasript
otestovanie
analyza otazkovaca

Retrospektiva

Robo

dufas ze, na konci kazdeho sprintu bude vzdy nasadenie testy sa spustia
koniec sprintu a tasky nie su uplne skoncene
snazit sa uzavriet
hlavne testy

Martin Svrcek

rozdelovat tasky uz na stretnuti, nie len user story
nabuduci tyzden v stredu prejdeme si dalsie user stories a ohodnotime to aj na dalsom stretnuti uz len vybrat a rozbit na tasky a takto by sme to stihali

Marek Sulek

viazne komunikacia

Zaciatok 4. sprintu

Planovanie

API

Sessions
Users: previazat usera so systemom druhym, vytvorit usra a zapamat idcko
AOI
Agregacie
get fix count, fix duration

AGREGACIE

prerobit tabulku v casti statistics
a zobrazi pre tychto pouzivatelov taketo pocty
kludne aj viac tabuliek
nemusi to byt user friendly
Vytvaranie agreacii

JS KNIZNICe

evaluation questions
bud volanie z kodu
alebo definovat v administracii custom pravidla
urobit trigger cez tu otazku

AOI

URL
ako to funguje, ignorujem vsetko za / napr ze dam tam * - zanalyzovat ci to tak funguje, ak ano ci to anotacny worker zohladnuje
xpath
nevratit len jeden ale viac

8 ôsme stretnutie

Simon:

dospecifikovanie API, pretrvavajuce problemy s pridelenim domeny

Peter Kusnir:

vytvoril zakladnu kostru API volani pre oblasti zaujmu, otazky ohladom typu navratovych hodnot, potrebuje este dospecifikovat niektore volania

Marek:

Zacal robit na API funkciach pre spravu userov. Upravil celkovu strukturu API a zmenil navratove hodnoty funkcii na DTO prislusneho modelu + http status kod, pomahal Simon Napisal aj testy na niektore vytvorene funkcie.

Viki:

Mala robit API funkcie pre sessiony. Vytvorila zakladne funkcie podla specifikacie, pomahal Marek.

Mato Svrcek:

Robi na analyze a implementacii otazkovaca v Alefe a jeho prepojenie so systemom cez api. Podarilo sa mu zdokumnetovat otazkovac v Alefe, vytvoril o tom dokumentaciu a zacal s implementaciou.

Matej Kucek:

Mal spravit zobrazovanie agregacii vo webovej aplikacii. Mal problemy s testovanim kodu kedze to u seba nevedel rozbehat. Dosiel Miso a pomohol mu s tym a aj rozpraval o tom ako testovat aplikaciu.

Peto Kis:

Podarilo sa mu dokoncit pocitanie agregacii.

9 Deviate stretnutie

Viki:

Dorobila API funkcie pre spravu sedeni, nestihla napisat testy, kod skontroloval Marek vramci pair programming-u

Matej:

Podarilo sa mu rozbehat webovu aplikaciu u seba aj ked musel nasledne riesit nejake problemy s prihlasenim. Ako riesenie si musel rozbehat u seba MySQL databazu v ktorej su zaznamy o uzivateloch a preto sa mu nepodarilo uplne dokoncit vizualizovanie agregacii na webe

Peto Kis:

Dokoncenie prace na pocitani agregacii, spisanie dokumentacie a testovanie priepustnosti

Peto Kusnir:

Doplnil funkcie do API pre spravu oblasti zaujmu, zmenil navratove hodnoty funkcii podla dohodnuteho standardu, zacal na testoch ale nestihol dokoncit

Marek:

Dokoncil API funkcie pre spravu userov, napisal testy a menil celkovu strukturu projektu API. Pomahal Viki s implementaciou.

Mato Svrcek:

Implementacia generovania otazky na zaklade prijateho eventu.

Simon:

Trosku s meskanim ale predsa, vecne problemy so serverom, dospecifikoval API volania, stretol sa s Petrom Lackom kvoli domene, aj tak sa s tym velmi nepodarilo pohnut

Semestralna retrospektiva:
Podarilo sa spravit pocitanie agregacii a zaklad pre API, poopravovali sa niektore chyby v kode, robilo sa na prepojeni s Alefom, od ktoreho sa kvoli buducim zmenam nakoniec upustilo, tim sa uchytil v teme, komunikacia bola na dobrej urovni ale treba sa vyvarovat situaciam kedy sa tyzden kvoli nejakej chybe niekto nevie pohnut a tim sa o tom dozvie az na stretnuti, zlepsit pracu v kolektive, stretnutia aj mimo vyhradeny rozvrh a riesenie problemov spolocne, zlepsit pisanie testov a vykonavanie prehliadok kodu, neriesit ulohy na poslednu chvilu, kladne je hodnotena ustretovost predosleho timu a ochota pomoct a poradit pri vzniknutych chybach alebo nasadzovani a testovani aplikacie.
Planovanie posledneho tyzdnoveho sprintu:
Dokoncenie rozrobenych uloh z predchadzajucich sprintov, doplnenie testov, spisanie dokumentacie riadenia a dokumentacie o inzinierskom diele a nasadenie celeho systemu na server.

10 Stretnutie so zákazníkmi

- Prepojenie s ALEFom , Záznam z kamery – problémom je synchronizácia, spôsob odosielania , Treba dbať na čo sa používa eye-tracker.

- Biznis problém
- MB – Posielať dáta klientovi, (Šimon – „prídeme s tým o biznis“) -> MB - neprídeme s tým o biznis, lebo to môžeme nejako spoplatniť -> ??.
- Ak to budeme posielať na server tak to nebude škálovateľné -> ak bude 200 aplikácií s 200 používateľmi tak všetko bude tiecť na náš server a z neho a bude to problém. Ideou bolo aby tá druhá aplikácia nemusela robiť cez svoju databázu nejaké veci. Skôr investovať do toho aby to mohol klient použiť.
- MLabaj - Ani to nemusí ísť vôbec nikam a môže sa to v javascripte spracovať. Javascript bude čakať na event a ak sa niečo bude sledovať (nejaký pohľad) tak bude reagovať nejako.

SOFTEC-FIIT_projekt

- MBielikova - Pozor na konkurenciu so Softecom -> s nimi sa bude robiť projekt na UX lab – budú tam ľudia aj od nás. Nemôžeme replikovať to čo je v požiadavkách na tento projekt.
- Minulý projekt robil spracovanie s webom -> ak by sme to dotiahli tak sa nebijeme so softecom.
- Pridanie poznámok k priebehu experimentov -> toto sa budfe riešiť so Softecom.
- Kvalitatívne testovanie – napr. zadávanie úloh a pod. – máme toto vôbec riešiť ? – skôr nie, lebo to sa rieši so Softecom. Toto ani nie je potrebné , skôr sa to robí ako súčasť webu.

Vizualizácia

- aj pre jedného aj pre viac používateľov – nie on the fly ale prepočítané vopred

- heat mapy a obrazovka za tým, za grafom
- Vizualizácia pre učiteľa – **dôležité pre zákazníkov !!!** -> na základe toho by vedel že ktoré časti textu sú zlé. – systém by vedel pomôcť učiteľovi na základe získaných údajov zo sledovania – toto by bolo veľmi dôležité.

ALEF integration

- MB: dáta sa dostanú do ALEFu – a priamo v ALefe vymyslieť scenár súvisiaci s pozorovaním
- ML: definovanie pravidiel (serverové, klientské) keď sa človek pozrie na LO tak sa ho spýtame prečo tam pozerá -> toto už funguje !!!
- Zamerať sa na webové aplikácie.
- Analýza portálu -> toto bola 1naj stránka a a na toto sa nepozeralo.

Zhrnutie

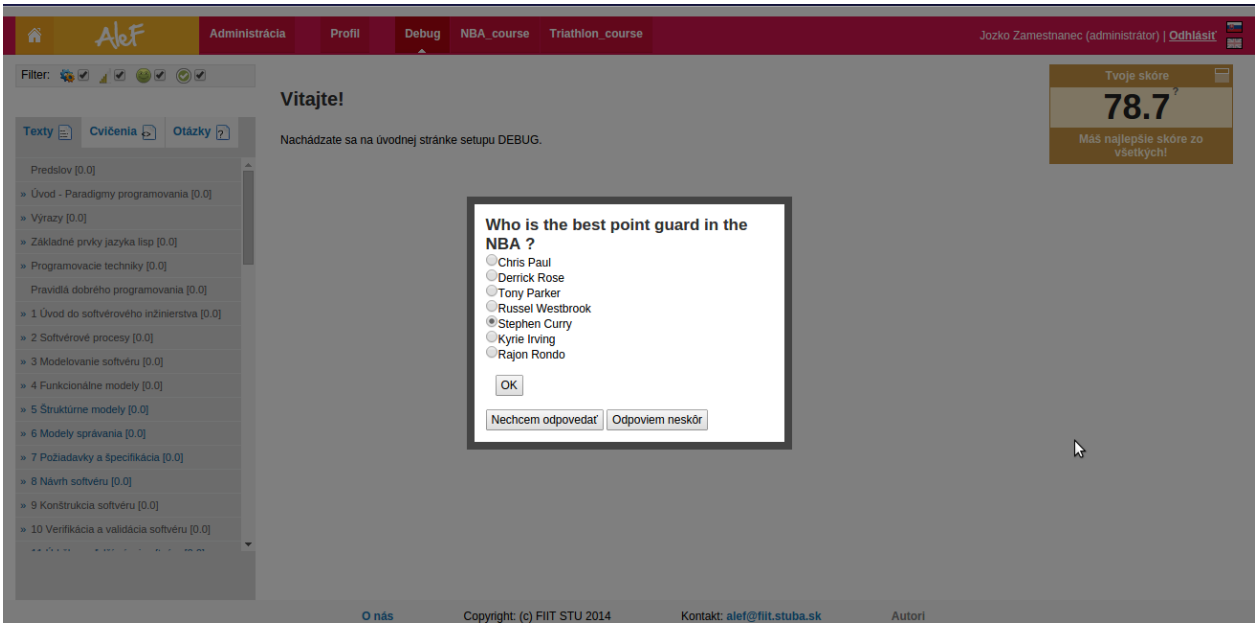
- Ako urobiť vyzbierané dáta užitočné – vizualizovať ich a prepojiť ich s aplikáciami.
- Orientovať sa na pomoc učiteľovi.

Príloha C: Pýtač v systéme ALEF (spätnoväzobné otázky)

1 Čo je to pýtač ? (evaluation_question)

V rámci ALEFu existuje otázkovač a pýtač:

1. otázkovač (CRANE) - Tento otázkovač systému ALEF poskytuje základné rozhranie pre zobrazovanie otázok, odpovedí a ich následné hodnotenie študentami. CRANE tento scenár rozširuje o možnosť odpovedania na otázky. Všetky otázky sú zobrazované ako samostatná html stránka. Bližšie informácie v časti Otázkovač v systéme ALEF (CRANE).
2. pýtač (spätnoväzobné otázky) - Tento pýtač umožňuje tiež zobrazovanie otázok a možnosť odpovdania na tieto otázky. Jeden z rozdielov je však v tom, že jednotlivé otázky sú generované ako samostatný formulár, ktorý sa zobrazí na aktuálnej html stránke (Obrázok C1).



Obrázok C1 - zobrazená otázka

V celom nasledujúcom dokumente sa budeme venovať práve pýtaču zobrazovanému ako prídavný formulár v danom dokumente.

Možnosť takéhoto zobrazovania má jeden dôležitý význam z hľadiska možností v systéme ALEF. V kontexte presonalizovaného webu je takto možné reagovať na individuálne vzory správania sa používateľa a generovať mu na základe týchto vzorov jednotlivé otázky. Otvára sa nám tak možnosť:

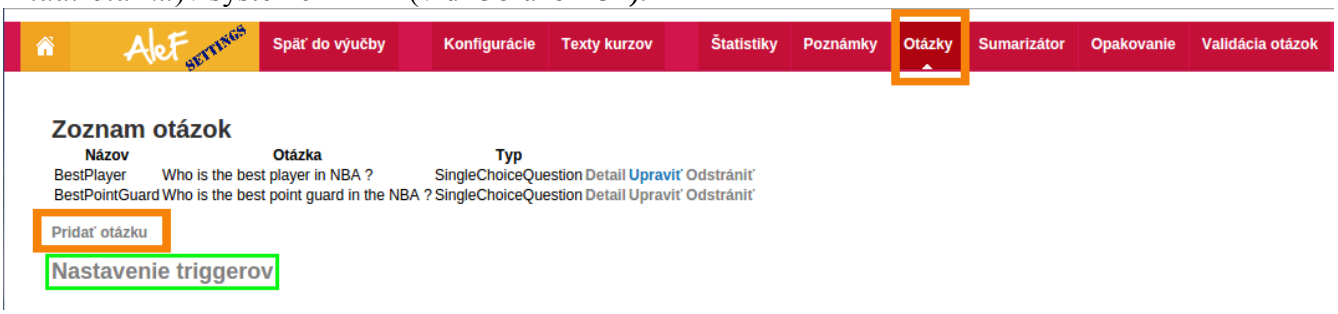
- Pomáhať študentom pri štúdiu
  - Tento scenár súvisí hlavne s možnosťou reagovať na určité akcie používateľa, ako napríklad dlhodobé zlé riešenie cvičení alebo dlhá doba strávená štúdiom jednej problematiky.
- Získavať spätnú väzbu na určité prvky systému
  - Tento scenár je veľmi dôležitý, lebo nám umožňuje získať informácie o niektorých prvkoch systému. Takisto môžeme zisťovať prečo daný používateľ nepoužíva niektoré nástroje v systéme ALEF.

2 Ako funguje takýto pýtač ?

Fungovanie pýtača je pomerne jednoduché. Jednotlivé otázky sú uložené v databáze, ktorá konkrétne obsahuje nasledujúce tabuľky:

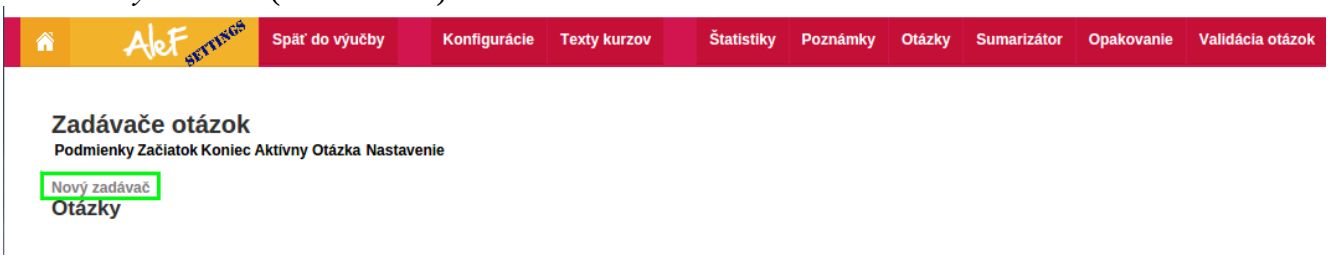
- EvaluationQuestion
  - Základná tabuľka pre otázky, pričom každá otázka sa skladá z: "id", "question_type", "content", "answers", "name", "group", "created_at", "updated_at".
- EvaluationQuestionQueue
  - Tabuľka obsahujúca otázky "čakajúce v rade" na to aby mohli byť zobrazené používateľovi.
- EvaluationQuestionTrigger
  - Tabuľka obsahujúca zavádzače pre jednotlivé otázky. Tieto zavádzače obsahujú pravidlá kedy sa môžu jednotlivé otázky zobraziť.
- UserAnsweredEvaluationQuestionRelation
  - Tabuľka obsahujúca záznam o tom, na ktoré otázky odpovedal daný používateľ.
- UserWasOfferedEvaluationQuestionRelation
  - Tabuľka obsahujúca záznam o tom, že danému používateľovi bola ponúknutá daná otázka.

Jednotlivé otázky sa dajú generovať prostredníctvom záložky *Administrácia->Otázky* (link *Pridať otázku*)v systéme ALEF (viď Obrázok C2).



Obrázok C2 - Pridanie otázky

Jednotlivé zavádzače sa taktiež pridávajú prostredníctvom záložky *Administrácia->Otázky* a následne je potrebné kliknúť na odkaz *Nastavenie triggerov* (Obrázok C2) a v ďalšej časti na odkaz *Nový zadávač* (Obrázok C3).



Obrázok C3 - Pridanie zavádzača (trigger)

V Administrácii je možné upravovať, odstraňovať a pridávať jednotlivé otázky.

Pridanie otázky do "radu na čakanie" sa realizuje v časti *Detail* (Obrázok C2), kde je potrebné kliknúť na odkaz *Testovať*.

Takto pridaná otázka sa následne bude zobrazovať v systéme ALEF (Obrázok C1). A ako bolo spomenuté, podmienky jej zobrazovania môžeme určiť pomocou zavádzačov.

2.1 Vyvolanie otázky priamo z kódu

Ak by sme chceli vyvolať otázku priamo z kódu existujú na to dva spôsoby:

1. Jedna možnosť je vytvoriť *EvaluationQuestionTrigger*, v ňom sú podmienky, ktoré sa vyhodnocujú vtedy, keď sa zisťuje, či nie je pre používateľa nová otázka a ak sa splnia, položí sa otázka.
2. Druhá možnosť je rovno vytvoriť záznam *EvaluationQuestionQueue*, tam sa potom nič nezisťuje, ale pri najbližšej možnosti sa otázka (podľa priority vo fronte ak ich je tam viac) zobrazí.

3 Implementácia

Samotnú implementáciu pýtača môžeme rozdeliť o dvoch častí:

- Časť administrácie
- Časť generovania otázok - zobrazovanie otázok, odpovedanie a následné spracovanie odpovedí

3.1 Administrácia

V súbore *EvaluationQuestionsController* sa v prvej časti nachádzajú funkcie na administráciu otázok (pridávanie, odstraňovanie, upravovanie, a pod.).
V súbore *EvaluationQuestionTriggersController* sa nachádzajú funkcie na administráciu zavádzačov (pridávanie, odstraňovanie, upravovanie, a pod.).
V časti *views->evaluation_questions* a *views->evaluation_question->triggers* sa následne nachádzajú konkrétne GUI pomocou, ktorých je správa realizovaná.

3.2 Generovanie otázok

Veci spojené s generovaním otázok sa nachádzajú v druhej časti súboru *EvaluationQuestionsController*.
S generovaním otázok je spojené aj ajax volanie v súbore *evaluation_questions.js.erb*. O toto sa starajú funkcie *askForQuestions()* a *getEvaluationQuestionHTML()*.

V súbore *EvaluationQuestionsController* sú dôležité najmä metódy:

- question - funkcia, ktorá ako prvé kontroluje či sa už môže generovať ďalšia otázka (kontroluje či už ubehol dostatočný čas od predchádzajúcej otázky). V tomto kontexte sú volané funkcie.
    - topmost_question_in_queue - získanie prvého prvku z radu "čakajúcich".
    - offer_question - ponúknutie otázky.

  Následne sa pozerá či už nejaká otázka "čaká v poradí" na zobrazenie. Treťou časťou je zistenie či si nejaký zavádzač žiada generovanie otázky.
- answer - funkcia na spracovanie odpovede na generovanú otázku.

Zisťovanie v metóde *question* sa vykonáva vždy ak je nanovo nahratá stránka alebo sa na pozadí už nahratej stránky zavolá pravidelný AJAX check.