Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Dav proti vizuálnemu smogu

Dokument riadenia

Vedúci tímu: Ing. Jakub Šimko, PhD.
Členovia tímu: Bc. Jana Egriová, Bc. Alexander Ferenčík, Bc. Richard Filipčík, Bc. Tomáš Melicher, Bc. Juraj Slavíček, Bc. Jaroslav Zigo
Akademický rok: 2014/2015

OBSAH

1 Úvod

Dokument zaznamenáva riadenie projektu s názvom Dav proti vizuálnemu smogu pre účely predmetu Tímový projekt na Fakulte informatiky a informačných technológií na Slovenskej technickej univerzite v Bratislave.

Druhá kapitola stručne predstavuje členov a samotný tím Včeličky a krátku motiváciu pre vypracovanie projektu Dav proti vizuálnemu smogu. Kapitola taktiež zahŕňa pridelené úlohy členom tímu a odpracovaný podiel práce.

Kapitoly 3 až 10 sa venujú samotnému manažmentu projektu. Opisujú vykonávané činnosti v manažmente z hľadiska ich vývoja v čase. Ako sa tieto činnosti vykonávali na začiatku, dôvody a motiváciu zmeny týchto činností doplnené o metodické príručky pre vybrané činnosti.

Prílohy obsahujú zápisnice zaznamenané na všetkých oficiálnych tímových stretnutiach.

2 Tím Včeličky

Náš tím sa skladá zo šikovných ľudí, kde časť má väčšie skúsenosti s webovými technológiami, čo naplno využijeme pri tvorbe serverovej časti aplikácie. Druhá časť tímu má skúsenosti s vývojom Android aplikácií. Tieto skúsenosti sa nám hodia pre potreby vývoja mobilnej aplikácie, ktorá bude primárnym nástrojom na zber dát v našom projekte. Nájdu sa medzi nami aj takí, ktorí už majú skúsenosti s počítačovým videním, ktoré sú pre potreby projektu Dav proti vizuálnemu smogu nemenej dôležité. Okrem iného sa nebojíme komunikácie s vyššími autoritami a nájde sa v nás aj kus aktivizmu.

2.1 Zoznam kompetencií

2.1.1 Jana Egriová

Preferujem programovací jazyk Java a jej najrôznejšie technológie. Skrz toho by som si rada vyskúšala vývoj Android aplikácií. Pracovala som však aj v jazyku C#. Čo sa týka webu, tak s HTML/ CSS/ Javascript/ PHP mám základné skúsenosti. Vzhľadom k tomu, že jednou z našich tém, o ktoré sa usilujeme je Dav proti Vizuálnemu Smogu, by som rada spomenula záujem o ekológiu, rôzne dobrovoľnícke aktivity a angažovanie v miestnych novinách a fórach, skrz ktorých som mala možnosť zistiť, aká je komunikácia so samosprávou náročná. Získané kontakty na ľudí v samospráve tiež môžu byť výhodou.

2.1.2 Alexander Ferenčík

Medzi moje najvýraznejšie skúsenosti patrí vývoj aplikácii v C,C++, ale ani Java pre mňa nie je problém. Z webových technológii mám skúsenosti s HTML a Javascript. Vo svojej bakalárskej práci som sa venoval tématike hernej umelej inteligencie, ktorou sa chcem zaoberať aj naďalej. V súčasnom zamestnaní každodenne využívam Javascript a SQL, ktoré ovládam bez problémov. Mám skúsenosti s technológiami JDBC, Hibernate, JPA, Java2D, JLog, MPI, OpenMP, POSIX Threads, CUDA a OpenGL. Veľmi ma zaujíma vývoj aplikácii pre Android, čo je momentálne mojím najväčším cieľom, ktorý sa chcem naučiť.

2.1.3 Richard Filipčík

Praktické skúsenosti mám najmä s webovými technológiami, ktorým sa venujem od konca základnej školy, a to či už zo strany klienta, alebo zo strany servera. Okrem kombinácie PHP/Apache/MySQL mám isté skúsenosti aj s Pythonom, na bakalárskom projekte som pracoval aj s Ruby/RoR. Zaujímala by ma aj práca s C# a .NET technológiami. Okrem toho, mám rád našu krajinu a nepáčia sa mi množstvá bilbordov napr. aj u nás doma, preto by som rád pracoval na niečom, v čom vidím potenciál a možnosť pomôcť aj mimo fakulty.

2.1.4 Tomáš Melicher

Medzi svoje popredné skúsenosti v oblasti IT by som uviedol webové technológie. S front-endovými technológiami som sa stretol už počas strednej školy, keď som si privyrábal tvorbou jednoduchých statických stránok. Nadobudnuté skúsenosti som si potvrdil MTA HTML5 certifikátom. Neskôr som sa preorientoval skôr na back-endové technológie, konkrétne PHP, CFML a sčasti aj .NET. Som otvorený možnostiam a mám záujem sa učiť nové technológie, čiže prácu s frameworkom RoR by som veľmi rád uvítal.

2.1.5 Juraj Slavíček

Medzi moje prednosti patrí určite pomerne veľké skúsenosti s prácou v tíme a z technológii Java a frameworky s ňou spojené. O čo som sa vždy zaujímal bol javovský (ale aj iný) back-end a server side technológie. Tieto technológie využívam aj v súčasnej práci a teda postaviť webovú aplikáciu na Springu Hibernate, prepojiť ju web servicami s inými aplikáciami a tým pripraviť pôdu pre front endistov mi problém nerobí. Pokiaľ sa však ukáže nutnosť naučiť sa nové technológie rád sa do toho pustím.

2.1.6 JAROSLAV ZIGO

Mojím najobľúbenejším programovacím jazykom je Java. Pracujem s ním prakticky denne, využívam ho najmä na vývoj Android aplikácií či už vlastných alebo pracovných. Nejaké tie riadky kódu som však napísal aj v jazykoch ako C, PHP, JavaScript/jQuery, SQL (MySQL) či HTML/CSS. Skúsenosti mám tiež s prácou s GitHubom, JDBC API a Hibernate. Som otvorený poznávaniu nových technológií, najmä javovských (Spring, Grails, JSF...), ale tiež by som si rád vyskúšal RoR a celkovo prácu na strane servera.

2.2 Motivácia

Téma Dav proti Vizuálnemu Smogu náš tím zaujala predovšetkým svojou myšlienkou. Našim cieľom je ponúknuť každému občanovi možnosť zaangažovania sa do skrášlenia svojho okolia pomerne jednoduchým spôsobom. Keďže téma zlepšovania verejného priestoru momentálne v spoločnosti rezonuje, chceli by sme tento trend využiť a za pomoci v začiatkoch menších odmien a neskôr výhod v rámci samosprávy občanov dostatočne motivovať k odstraňovaniu nelegálnych a neestetických reklám.

Vo fáze realizácie chceme vypracovať nielen samotnú mobilnú aplikáciu, ale aj prepojenie na webovú aplikáciu s potrebným back-endom, ktorý bude pri vyhodnocovaní snímok využívať automatizáciu. Obe aplikácie budú komunikovať prostredníctvom RESTful webových služieb. Táto téma sa nám ako tímu javí nadmieru atraktívna vzhľadom na to, že náš tím disponuje ľuďmi, ktorí sa zameriavajú na vývoj pre Android platformu, ako aj ľudia so skúsenosťami z oblasti vývoja webových stránok a aplikácií. Nebojíme sa komunikácie s vyššími autoritami a nájde sa v nás aj kus aktivizmu.

2.3 Manažérske úlohy

Členom tímu boli priradené nasledovné manažérske roly:

- Jana Egriová – manažérka dokumentácie
- Alexander Ferenčík – manažér rizík
- Richard Filipčík – manažér podporných prostriedkov
- Tomáš Melicher – manažér plánovania, manažér integrácie
- Juraj Slavíček – vedúci tímu, manažér komunikácie
- Jaroslav Zigo – manažér kvality

2.4 Podiel práce

2.4.1 DOKUMENT INŽINIERSKEHO DIELA

Časť dokumentu	Podieľajúci sa členovia tímu
Úvod	Juraj Slavíček
Ciele	Jana Egriová
Funkcionálne požiadavky	Juraj Slavíček
Architektúra	Tomáš Melicher, Richard Filipčík, Alexander Ferenčík
Dátový model	Richard Filipčík
Modul registrácia a prihlasovanie	Juraj Slavíček
Modul mapa	Tomáš Melicher
Modul upload obrázka	Tomáš Melicher
Modul mobilnej aplikácie	Jaroslav Zigo, Alexander Ferenčík

2.4.2 DOKUMENT RIADENIA

Časť dokumentu	Podieľajúci sa členovia tímu
Úvod	Jana Egriová
Tím, kompetencie a motivácia	celý tím
Manažment komunikácie	Juraj Slavíček
Manažment plánovania	Jana Egriová
Metodika opisu úloh v systéme JIRA	Richard Filipčík
Metodika plánovania šprintov	Jaroslav Zigo

Manažment kvality	Jaroslav Zigo
Metodika písania zdrojových kódov	Tomáš Melicher
Metodika testovania	Alexander Ferenčík
Manažment rizík	Alexander Ferenčík
Manažment integrácie	Tomáš Melicher
Manažment podporných prostriedkov	Richard Filipčík
Metodika používania verziovacieho systému Git	Juraj Slavíček
Manažment dokumentácie	Jana Egriová
Metodika tvorby dokumentácie	Jana Egriová

3 MANAŽMENT KOMUNIKÁCIE

Na začiatku projektu bolo nutnosťou zaviesť štandardné formy komunikácie pre riešenie každodenných vecí ale aj kľúčových aspektov v projekte. Ako základný komunikačný kanál bola zvolená sociálna sieť Facebook keďže je na nej každý člen tímu registrovaný a je jej každodenným používateľom . V rámci nej sú vytvorené 3 konverzácie. Prvou z nich je konverzácia spoločná pre všetkých členov tímu kde sa dohadujú základné organizačné veci a práca na častiach projektu o ktorých musí vedieť každý jeden člen. Keďže aplikácia je logicky zložená z dvoch častí mobilnej a webovej a každá z nich je implementovaná v inom programovacom jazyku za použitia rôznych vývojových nástrojov obe zostávajúce konverzácie združujú každá polovicu tímu podľa ich zaradenia.

Ďalším komunikačným kanálom vytvoreným hlavne za účelom komunikácie tímu a vedúceho sú tzv. GoogleGroups. Táto aplikácia slúži na rýchlu komunikáciu celého tímu pomocou mailov a je integrovaná s nástrojom GoogleDocs takže každý člen tímu je rýchlo informovaný o nových vypracovaných veciach a reakciách kolegov na ne.

Okrem komunikačných nástrojov sa manažér komunikácie v našom tíme zaoberá aj vedením “zamrznutej” diskusie na tímových stretnutiach ako aj riešeniu vznikajúcich problémov spôsobených inými vlastnosťami, predstavami a plánmi členov tímu ku ktorým prirodzene dochádza vzhľadom na našu odlišnosť.

4 MANAŽMENT PLÁNOVANIA

Pre účely plánovania šprintov bola vytvorená fyzická tabuľa s plánovanými user stories a ich úlohami. Po každom tímovom stretnutí sme vytvárali digitálnu kópiu tabule v podobe fotografie.

Takáto evidencia úloh sa ukázala ako nedostatočná, preto sme začali evidovať úlohy pre jednotlivé úlohy v systéme Jira. Úlohy sme do systému zadávali len pomocou ich názvu, kde chýbal podrobnejší opis tejto úlohy. Dôsledkom toho bola situácia, kedy sme neskôr podľa názvu už nevedeli identifikovať presný cieľ úlohy a z toho dôvodu sme do procesu plánovania zaviedli aj povinnosť podrobného opisu náplne úlohy.

Úlohou manažéra plánovania je vytváranie nových šprintov, priraďovanie úloh do šprintov, kontrola dodržiavania opisov úloh a taktiež kontrola stavu riešenia jednotlivých úloh. Aktuálne kombinujeme evidenciu úloh v systéme Jira spoločne s evidenciou pomocou pôvodnej fyzickej tabule.

S manažmentom plánovania súvisia dve metodiky, ktoré sme vytvorili.

- Metodika opisu úloh v systéme JIRA
- Metodika plánovania šprintov

5 MANAŽMENT KVALITY

Keďže miera kvality akéhokoľvek produktu výrazným spôsobom určuje jeho postavenie na trhu, resp. medzi inými produktami, je potrebné manažovať proces jej vytvárania a zabezpečovania. Výber nástrojov použitých na implementáciu a spôsob vykonávania určitých činností v rámci celého vývoja napomáha zvyšovať mieru kvality. Za týmto účelom sme sa rozhodli využívať nasledovné nástroje / metódy:

- Android Studio - ide o vývojové prostredie, ktoré je veľmi populárne a na vývoj ho využíva aj samotný Google. Umožňuje Gradle build aplikácií, čo má oproti Ant-u (využívanom napr. v Eclipse, IntelliJ Idea…) viaceré výhody (jednou z nich je napr. veľmi jednoduchý import knižníc).
- GitHub - veľmi dôležitý nástroj pre vývoj akéhokoľvek softvéru. Umožňuje komukoľvek s prístupom nazerať do kódu, stiahnuť si ho k sebe, kedykoľvek commitovať a v prípade chybných verzií sa vrátiť k predošlým. Predstavuje štandard kvality pri akomkoľvek softvérovom projekte.

- SCRUM - agilný vývoj. Zvyšuje sa transparentnosť procesu, pričom každý člen tímu má prehľad o tom, v akom stave sa projekt nachádza, všetko má svoju štruktúru a znižuje sa pravdepodobnosť výskytu chybových miest v projekte.

Ľudský kapitál je pri vývoji najdôležitejším faktorom. Keďže každý člen tímu potrebuje zdokonaľovať svoje zručnosti a nie všetci sú na tom čo sa odbornosti týka rovnako, zaviedli sme v našom tíme stretnutia (časovo nie striktne určené) v menších skupinkách, kde členovia s väčšou znalosťou v danej oblasti vyučujú a vysvetľujú svoje poznatky menej zdatným kolegom. Takisto sa menej skúsení členovia vzdelávajú prostredníctvom video kurzov a materiálu dostupného na internete. Takýmto spôsobom sa zvyšuje kvalita nášho tímu ako celku, čo sa prenáša do kvality výsledného produktu.

S manažmentom kvality súvisia dve metodiky, ktoré sme vytvorili.

- Metodika písania zdrojových kódov
- Metodika testovania

6 MANAŽMENT RIZÍK

Na zamedzenie rizík v našom tíme sa snažíme aplikovať pravidelné stretnutia opakujúce sa v týždňových intervaloch. Na stretnutiach si hneď úvodom vzájomne referujeme odvedenú prácu za posledný týždeň, pričom sledujeme najmä prínos projektu a zároveň časovú náročnosť. Týmto spôsobom pozorujeme, či bola pracovná záťaž adekvátne rozdelená medzi členov tímu, nakoľko zlé rozdelenie práce je jedným z možných rizík projektu. Získané postrehy sa snažíme aplikovať pri nasledovnom prideľovaní úloh. Nepodceňujeme pri tom situáciu a všetko zaznamenávame v nástroji JIRA aj s dôkladnými popismi úloh.

Identifikovali sme tiež riziko zlej komunikácie, kde môže dôjsť k nedorozumeniu niektorých členov tímu čo sa môže prejaviť neskôr. Okrem pravidelných stretnutí udržujeme systematickú správu našich informácií a dát na verejne dostupnom úložisku od Google – Google Drive. Spoločne máme prístup k všetkým zozbieraným dokumentom, ktoré sem pravidelne, zvyčajne hneď po stretnutí, uploadujeme. Taktiež komunikujeme prostredníctvom internetového chatu a denne sa oboznamujeme s priebežným stavom práce.

Taktiež sme sa snažili predísť situácii aby niektorý člen musel zisťovať poznatky pre nás vhodné, ak už v minulosti iný člen mal tieto poznatky zaobstarané. Na zamedzenie tohto rizika využívame našu vlastnú wikimedia stránku, na ktorej zdieľame náučné materiály potenciálne vhodné aj pre ostatných členov, respektíve návody na vyriešenie problémov súvisiacich s naším projektom.

Ak sa niektoré z rizík javí ako zvýšené, manažér rizík na to upovedomí ostatných členov tímu, ktorí ho rešpektujú a snažia sa vyzdvihnuté riziko zamedziť.

7 MANAŽMENT INTEGRÁCIE

Keďže naša aplikácia pozostáva z dvoch samostatných modulov, vyskytol sa problém, že navzájom nekooperovali. Náš tím pozostáva z ľudí venujúcich sa čisto webovej aplikácii a ľudí venujúcich sa iba Android aplikácii. V tíme chýbala medzivrstva, ktorá by riešila integráciu medzi obidvoma modulmi. V našom tíme preto bola vytvorená funkcia manažér integrácie, pričom jeho primárnou úlohou bola integrácia práve spomínaných dvoch modulov.

Po prvom šprinte dokázala naša mobilná aplikácia odoslať na server fotografiu o zachytenom billboarde, server ho však nedokázal spracovať nakoľko očakával dáta iného formátu než boli zasielané. Na serveri bolo totižto taktiež možné uploadovať odfotený billboardu s tým, že sa uploadované dáta spracovali inak, než sa očakávalo z mobilnej aplikácie. Prvotnou úlohou do druhého šprintu bolo teda zintegrovať obidva moduly, čo sa nám aj nakoniec podarilo.

Prototyp našej aplikácie počítal iba s natvrdo zadanými dátami, pretože zatiaľ nebol hotový dátový model. Po jeho implementácii bolo opäť potrebné zintegrovať existujúce moduly našej aplikácie s databázou. Integrácia webovej aplikácie s vytvorenou databázou bola pomerne jednoduchá vzhľadom na fakt, že databázový server sa fyzicky nachádzal na tom istom mieste ako náš webový server. Problémom však bola integrácia s mobilnou časťou našej aplikácie, bolo teda potrebné vytvoriť funkcionalitu pre odosielanie dát na mobilné zariadenia v požadovanom formáte. To sa nakoniec taktiež úspešne podarilo, čo bolo aj overené sériou unit testov. Dáta sú zo servera posielané v JSON formáte, pričom odpovedi na požiadavku predchádza POST request.

V treťom šprinte mal manažér integrácie za úlohu kontrolovať vývoj modulu pre spracovanie GPS súradníc zasielaných z mobilnej aplikácie. Momentálne má manažér integrácie úlohu pravidelne kontrolovať kód, ktorý zasahuje do mobilnej aj webovej aplikácie, aby nevznikol problém, ktorý sa nám vyskytol v prvom šprinte.

8 MANAŽMENT PODPORNÝCH PROSTRIEDKOV

Manažér podporných prostriedkov má za úlohu zabezpečovať predovšetkým nasadzovanie a funkčnosť riešení určených pre podporu práci na projekte. Ide napríklad o architektúru webového servera, ktorý zabezpečuje chod webovej aplikácie, databázového servera, či interpreta scriptovacieho jazyka. Zároveň sa stará o spravovanie používateľských účtov,

manažment práv pre otváranie, zapisovanie a spúšťanie súborov a priečinkov, či riadenie a plánovanie automatického spúšťania úloh.

S manažmentom podporných prostriedkov tiež súvisí starostlivosť o projekt v prostredí verziovacieho systému. S tým sa spája napríklad vytvorenie samotného repozitára, kontrola správneho vetvenia a zlučovania už vytvorených vetiev, prípadne riešenie konfliktov vzniknutých pri zlučovaní.

Pre účely archivácie dôležitých dokumentov a propagáciu projektu je dôležité udržiavať aj webovú prezentáciu projektu. Vytvorenie tejto prezentácie, jej priebežnú aktualizáciu, správu používateľských účtov pre ostatných členov tímu a ďalšie prípadné úlohy súvisiace s webovou prezentáciou má na starosti tiež manažér podporných prostriedkov.

S manažmentom podporných prostriedkov súvisí jedna metodika, ktorú sme vytvorili.

- Metodika používania verziovacieho systému Git

9 MANAŽMENT DOKUMENTÁCIE

Pre účel tvorby dokumentácie k tímovému projektu bol vytvorený zdieľaný priečinok v službe Google Drive. K spomínanému priečinku majú prístup všetci členovia tímu a slúži ako úložisko pre všetky vytvárané dokumenty.

Za účelom technickej dokumentácie pre samotných členov tímu, akými sú napríklad rôzne inštalačné príručky, návody a rady, ako postupovať pri práci na rôznych častiach aplikácie v rôznych prostrediach bola vytvorená wiki prístupná všetkým členom tímu prostredníctvom webového sídla.

Úloha manažéra dokumentácie je dohliadať na pravidelnú tvorbu a aktualizovanie dokumentov všetkými členmi tímu. Udržiavať aktuálnu štruktúru spomínaných úložísk a vytvárať šablóny pre jednotlivé dokumenty.

Šablónu pre zápisnicu zo stretnutí sme používali od začiatku tímových stretnutí, avšak pri prvej retrospektíve šprintu vznikla potreba vytvorenia zvlášť dokumentu. V súčasnosti používame zvlášť šablónu na zápisnicu zo stretnutí a zvlášť dokument na retrospektívu šprintu.

S manažmentom dokumentácie súvisí jedna metodika, ktorú sme vytvorili.

- Metodika tvorby dokumentácie

10 PRÍLOHY

10.1 Zápisnice zo stretnutí

- Zápisnica zo stretnutia č. 1
- Zápisnica zo stretnutia č. 2
- Zápisnica zo stretnutia č. 3
- Zápisnica zo stretnutia č. 4
- Zápisnica zo stretnutia č. 5
- Zápisnica zo stretnutia č. 6
- Zápisnica zo stretnutia č. 7
- Zápisnica zo stretnutia č. 8

10.2 Metodiky

- Metodika opisu úloh v systéme JIRA
- Metodika plánovania šprintov
- Metodika písania zdrojových kódov
- Metodika testovania
- Metodika používania verziovacieho systému Git
- Metodika tvorby dokumentácie

ZÁPISNICA ZO STRETNUTIA ČÍSLO 1

Úvodné stretnutie

Dátum:	2.10.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00
Účastníci:	Jana Egriová
*chýbajúcich preškrtnúť	Alexander Ferenčík
	Richard Filipčík
	Tomáš Melicher
	Juraj Slavíček
	Jaroslav Zigo
Vedúci stretnutia:	Jakub Šimko
Zapisovateľ:	Tomáš Melicher

Zaznamenaný priebeh stretnutia

* hviezdickou oznacujem poznamku pod ciarou, ciselne su oznacene ulohy

1, urcit si system na evidenciu stavu projektu (Gira,...) - Juro

* ideme vyvijat agilnou metodou

HAPPYDAY SCENAR - pouzivatel ide po meste so svojim smartfonom, narazi na nelegalnu reklamu, odfoti ju, ulozi sa na server - po dokonceni happyday scenara by uz mala byt hotova cela infrastruktura

FEATURES:

- pri foteni sa zobrazi predikcia polohy billboardu - upozornenie pouzivatela, ze uz som raz dany billboard fotil (vramci gemifikacie, nestupal by mu ranking ak by stale fotil jednu a tu istu) - po odfoteni ak nieje aktualne online, zariadenie pocka kym sa user pripoji na wifi a az tak to odosle

2, treba zriadit google doc, kde budu spisane featurky, happyday scenar atd...ja zriadim

3, zavieze si slovnik pojmov - "spina" - nelegalna reklama, atd... "podnet/ulovok" - odfotenie reklamy

USER STORIES

* je ich treba zapisovat v tvare ze "KTO", "CO ROBI", "PRECO"

- user story template - pouzivatel v telefone odfit reklamy aby ocistil mesto od nelegalneho vizualneho smogu - uradnik si pozrie databazu nasich reklam, vyfiltruje si nelegalne aby mohol zacat prislusne konanie ... - spisat ich co najviac, idealne do tabulky (spominal, ze ich raz bude mozno 50-100)

EPIC STORIES

* tu treba nadefinovat nejake rozsirene user stories - kategorie, bude treba rozpisat konkretne stories

- gemifikacia
- validacia dat - to zahnurje aj nejake to spajanie ulovkov
- vystavenie dat v strojovej podobe
- vodiči
- podpora pre samosprávy

4, rozdelit si roly v time - done

5, tato uloha je pre kazdeho a do buduceho tyzdna - identifikovat nejakych "early adopterov" (idealne 2-3), ktory maju zaujem takto pomoct peknemu zivotnemu prostrediu, predstavit im nasu appku zopar vetami a popytat sa ich, ako by mala vyzerat nasa appka, lebo ak sa to nebude pacit im, tak potom nikomu

* do jedneho dokumentu user stories a do druheho datove entity, do tretieho dokumentu zbierat spatnu vazbu od ludi, idealne zalozit folder na google docs

* treba robit code review, aby mali vsetci prehlad o tom, ako to funguje, opravovali chyby, optimalizovali...prispieva to ku kvalitnejsiemu kodu

* vramci menezmentu rizik, som sa rozhodol ze stranku nasho timu robit nebudem :) robil som ich 10tky, nic nove by som sa nenaucil... ochotne ale pomozem hocikomu, kto ju bude chciet robit

6, my traja webisti by sme mali zhodnotit (primarne)skusenosti a vyuzitie danych technologii a urcit ake konkretne technologie pouzijeme, framework, db, ceknut OpenCV, brat do uvahy ze ideme pracovat hlavne s geografickymi udajmi - bude tam Debian, cize zvazit to

* vsetko pokryvam testami, hlavne na backende, treba sa zoznamit s testovacimi frameworkami

ROLY

- veduci / menezer komunikacie - musi vediet nakopnut tim, ak to niekde stoji, tak to rozbehat
- menezer planovania - musi vediet odhadnut co bude kolko trvat, zistovat, musi vediet urcit nejaky casovy plan, vyhodnocovat progres
- menezer podpornych prostriedkov - konfiguracie serverov, konfiguracie gitu...robi support
- menezer rizik - to by mali robit vsetci, jeden vsak primarne, ktory dokaze mysliet aj za druhych, co sa tyka odhadnutia rizik • menezer kvality - musi dbat na kvalitu kodu, na testovanie, kontroluje code review
- menezer dokumentacie - musi vediet rozoznat co v dokumentacii musi byt, co tam je zbytocne...dba na to, aby kazdy zdokumentoval postup, ked nieco dokaze rozbehat (detekciu billboardu)

Vyplývajúce úlohy

	Úloha	**Kto**
1	urcit si system na evidenciu stavu projektu	Juro
2	zriadit google doc, kde budu spisane featurky, happyday scenar atd	Tomas
3	identifikovat nejakych "early adopterov" (idealne 2-3) a ziskat od nich spatnu vazbu	Kazdy 3 ludi

4	zhodnotit (primarne)skusenosti a vyuzitie danych technologii a urcit ake konkretne technologie pouzijeme, framework, db, ceknut OpenCV	Riso, Juro, Tomas
5	Rozbehat prostredie pre vyvoj android app	Sano, Janka

ZÁPISNICA ZO STRETNUTIA ČÍSLO 2

Inicializácia šprintu 1-1Z

Dátum:	9.10.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00

Účastníci:	Jana Egriová
*chýbajúcich preškrtnúť	Alexander Ferenčík
	Richard Filipčík
	Tomáš Melicher
	Juraj Slavíček
	~~Jaroslav Zigo~~

Vedúci stretnutia:	Tomáš Melicher
Zapisovateľ:	Jana Egriová

Zaznamenaný priebeh stretnutia

Splnenie min. týždňových úloh
Na konci sprintu – retrospektíva.
Tomáš rozbehal server – všetkým treba zriadiť kontá. Vytvoril dokumenty a folder
Rišo pracuje na webe, problém s Jirou ohľadom licencie, existuje školská verzia, na najbližšom stretnutí budeme o tom rozprávať.
Janka nainštalovať prostredie, je to beta verzia – na prediskutovanie.
Juro založil google group.

Spoločné úlohy
User stories 20ks
Spätná väzba splnená úloha bez väčších problémov.

Diskusia
Párové programovanie.
~~Papierové prototypy.~~

Vyplniť lean canvas.

Zadefinovať používateľov (používateľ v teréne, vo webovom rozhraní...)
Rozhodnúť sa, na koho zamerať aplikáciu, aktivista alebo bežný pasívny človek. Rozdiel je v motivácii.
Early adopters – aktivisti
Vygenerovať podanie pre samosprávu na základe zozbieraných dát.

Plánovanie sprintu
Vyberú sa user stories, ktoré zvládneme. Hotové.
Hotové – zadefinovať si, čo to znamená.

User stories

1. Používateľ webového rozhrania – Domased si otvorí mapu mesta, uvidí všetky úlovky, aby mal prehľad, kde je mesto zamorené.
2. Domased uploadne fotku a zapichne marker, aby pridal nový úlovok – prispel k mapovaniu reklamy v meste.
3. Používateľ v teréne odfotí reklamu, aby pridal nový úlovok – prispel k mapovaniu reklamy v meste.

Čo znamená, že je niečo hotové:

1. Určiť pravidlá, ako commitovať do repozitára, ak sa niečo naprogramuje
2. Napísať unit testy k funkcionalite – všetky budú zelené
3. Prebehol code review
4. Hotovo.

Bugy
Ak vznikne nejaký bug, ale nie je to jeho chyba, tak treba vytvoriť task typu bug v Jire. Ten, koho sú to kompetencie to začne riešiť, ak to je rozsiahlejší problém a jeho riešenie ohrozí ďalšie úlohy, tak sa v ďalšom sprinte naplánuje riešenie bugu, ostatné veci sa posúvajú.
Na základe feedbacku diskusia o odozve pre používateľov

Budeme musieť vysvetliť používateľom účel projektu – primárne mapovať reklamy. Odozvu na úlovky vyriešiť iným spôsobom (pochvala, progress bar). Zavedenie hejtov – koľko ľudí hejtuje úlovok.

Nabudúce sa pobaviť o dokumentácii

Vyplývajúce úlohy

Evidovať si, koľko sme robili (pri všetkom, aj ako prisediaci) ku každej úlohe

Vysvetlivky:
* prisediaci človek, android Janka, web Juro
tučným písmom je uvádzaný zodpovedný človek

	Úloha	**Kto**
1	Vytvoriť prístupy na server	Tomáš
2	Rozbehať Jiru	Juro
3	Rozbehať GitHub a zintegrovať s vývojovým prostredím	Juro
4	Zosumarizovať feedback do odporúčaní.	Janka
5	Vytvorenie personas (prečítať si k tomu 1-2 články)	Juro
6	Vytvorenie webu	Rišo
7	Založiť knowledge base (na wiki) a vytvoriť štruktúru	Janka
8	Zaeviovanie úloh a user stories v Jire tasky majú mať opisy	User stories **Tomáš** + tasky každý svoje

9	Vytvoriť lean canvas	Juro
10	Vytvoriť dátový model	**Rišo** + všetci
11	Vytvorenie databázy	Rišo
12	Kostra serverovej časti + Rozhranie	**Rišo** Tomáš
13	Zobrazenie mapy s bodmi – reklamy	Tomáš *
14	Pridanie bodu do mapy – drag and drop	Tomáš *
15	Kostra android aplikácie	Šaňo **Jaro ***
16	Odfotiť obrázok	**Šaňo** Jaro *
17	Upload obrázka	**Šaňo** Jaro *
18	Offline režim aplikácie (pošle, keď je zapnutá a je online)	Šaňo **Jaro ***

ZÁPISNICA ZO STRETNUTIA ČÍSLO 3

Téma stretnutia

Dátum:	16.10.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00
Účastníci:	Jana Egriová
*chýbajúcich preškrtnúť	Alexander Ferenčík
	Richard Filipčík
	Tomáš Melicher
	Juraj Slavíček
	Jaroslav Zigo
Vedúci stretnutia:	Jana Egriová
Zapisovateľ:	Richard Filipčík

Zaznamenaný priebeh stretnutia

Diskusia o fotení bilbordov

- algoritmus rozpoznávania reklám, diskusia o tom, ako zvýšiť úspešnosť
- ako šetriť používateľské dáta
    - odosielanie fotografií na základe typu pripojenia
- aplikácia
    - čo možno najjednoduchšia, bez zbytočného množstva nastavení

Podpora pre vodičov

- zamerania viac na GPS lokáciu, než samotné fotenie(?)

Spätná väzba od samosprávy

- nie je to najvyššia priorita, budeme riešiť zrejme neskôr

Krátka diskusia o čase strávenom nad projektom

- odhad času do ďalšieho týždňa
- „brainstorming“ nad manažovaním vlastného času

Verziovanie databázy

- navrhnúť flexibilný dátový model
- viac vetiev databáz (produkčná, testovacia)
- „verziovanie“ (zaznamenávanie úprav v modeli) nie je nevyhnutné

Zvyšok stretnutia prebiehal "praktickou formou", čas sme využili na diskusiu o technických záležitostiach, frontendisti aj backendisti rozoberali rôzne problémy týkajúce sa zostávajúcich úloh tohto šprintu.

Vyplývajúce úlohy

	Úloha	**Kto**
1	Rozbehať Jiru	Juro
2	Zosumarizovať feedback do odporúčaní.	Janka
3	Vytvorenie personas (prečítať si k tomu 1-2 články)	Juro
4	Zaeviovanie úloh a user stories v Jire tasky majú mať opisy	User stories **Tomáš** + tasky každý svoje
5	Vytvoriť lean canvas	Juro
6	Vytvoriť dátový model	**Rišo** + všetci
7	Vytvorenie databázy	Rišo
8	Kostra serverovej časti + Rozhranie	**Rišo** Tomáš
9	Zobrazenie mapy s bodmi – reklamy	Tomáš *
10	Pridanie bodu do mapy – drag and drop	Tomáš *
11	Kostra android aplikácie	Šaňo **Jaro ***

12	Odfotiť obrázok	**Šaňo** Jaro *
13	Upload obrázka	**Šaňo** Jaro *
14	Offline režim aplikácie (pošle, keď je zapnutá a je online)	Šaňo **Jaro ***

ZÁPISNICA ZO STRETNUTIA ČÍSLO 4

Inicializácia druhého šprintu

Dátum:	23.10.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00
Účastníci:	Jana Egriová
*chýbajúcich preškrtnúť	Alexander Ferenčík
	Richard Filipčík
	Tomáš Melicher
	Juraj Slavíček
	Jaroslav Zigo
Vedúci stretnutia:	Richard Filipčík
Zapisovateľ:	Alexander Ferenčík

Zaznamenaný priebeh stretnutia

Výsledky šprintu:

Personas – bude ich treba prerobiť/spraviť lepšie

Pridanie bodu do mapy vo webapp – funguje, bez ohľadu na grafické rozhrania, user môže pridať obrázok a kliknutím do mapy ho vložiť na presnú pozíciu

Offline režim pre mobileapp – aplikácia reaguje na zmeny stavu pripojenia k sieti, posiela notifikáciu, keď sa užívateľ opäť pripojí – bude upozornený, že už môže uploadnuť zaznamenané úlovky

Vznikla diskusia medzi webapp(wa) a mobileapp(ma) členmi tímu na tému akým spôsobom/v akom tvare posielať úlovky na server. Jedná sa najmä o pripojenie textovej časti(užívateľský komentár) a ostatných údajov.

Retrospektívne hodnotenie šprintu:

Pozitívne hodnotíme, že každý si plní svoje úlohy, snaží sa pracovať na tom, čo má pridelené a tasky postupne odbúdajú. Kanály komunikácie máme už zaužívané a špecifikované, každý vie, čo má akým spôsobom riešiť. Individuálna zodpovednosť za pridelené úlohy je v tíme motiváciou pre každého člena. Väčsina supportných záležitostí je už vyriešená. Napriek tomu očakávame samozrejme ešte niekoľko vzniknutých problémov aj v oblasti supportu. Dôležité je mať zavedené pravidlá, ktoré sa budú dodržiavať.

Negatívne hodnotíme, že dochádza k nezhodám – napr medzi čenmi wa/ma tímov. Na toto sa však dá pripraviť a predísť tomu. Týmto problémom s integráciou sa však budeme snažiť predísť ešte lepšou komunikáciou. Nepodarilo sa splniť všetky zadané úlohy v šprinte, mali by končiť ako DONE a nie v IN PROGRESS, toto je však pre zatiaľ akceptovateľné, nakoľko sme na začiatku projektu. Vzniká otázka: Ako zlepšiť problémy z integráciou? – Diskutovaním medzi skupinami navzájom. Vznikol návrh istého dôstojníka za každú skupinku, ktorý komunikuje vzniknuté otázky s dôstojníkom z druhej skupinky.

Človek by sa mal postupom času venovať najmä svojej časti. Mali by sme viac brať ohľad na nájdené biznis scenáre – na tieto by sme v budúcnosti mali viac prihliadať a orientovať sa viac na USER stories ako na funkcionalitu.

Zvolili sme si novú funkciu, manažér integrácie – Tomáš

Nevie, či bude stíhať... aj komunikácia (maily) je tiež práca

Manažér komunikácie by to mohol zrátať, bolo by zaujímavé poznať ten údaj.

Obava: veľa vecí na Tomáša s novou rolou

- delegovať veci na iných ľudí
- lepšie plánovať – brať na to ohľad a počítať s tým

Treba si uchovať, čo sme si naplánovali – zápis/nahrávka.

Skúsiť vždy dopredu odhadnúť, že čo všetko potrebujem (aj s kým komunikovať) k riešeniu svojich táskov.

Pracnosti (koľko hodín kto pracoval na tímovom projekte za posledný týždeň):

Janka – 8,5

Rišo – 11-12

Tomáš – 10-11

Juro – 6-7

Jaro – 4-5

Šaňo – 6

Do budúcna, vždy keď stojí niečo za pozornosť, treba posielať mail s touto info.

Vyplývajúce úlohy

	Úloha	**Odhad času [h]**	**Zodpovedný**
1	Doplnenie dátového modelu	3	Rišo
2	Update kostry serverovej časti	6	Rišo
3	Vytvorenie webového rozhrania	8	Juro
4	Zobrazenie mapy s bodmi	2,5	Tomáš
5	Pridanie bodu do mapy (drag&drop)	4	Tomáš
6	Integrácia medzi front-endom a back-endom	5	Tomáš
7	Upload obrázka prostredníctvom android app	2	Jaro

8	Kostra android app	2	Šaňo
9	Offline režim pre android app	4	Jaro
10	Android: odoslanie voliteľných dát k úlovku	2	Šaňo
11	Update front-endu a databázy	3	Juro
12	Vytvoriť plán na semester	2	Janka
13	Vytvoriť lean canvas	1	Juro
14	Vytvoriť personas	2	Juro

ZÁPISNICA ZO STRETNUTIA ČÍSLO 5

Priebeh druhého šprintu, low fidelity návrhy

Dátum:	30.10.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00
Účastníci:	Jana Egriová
*chýbajúcich preškrtnúť	Alexander Ferenčík
	Richard Filipčík
	~~Tomáš Melicher~~
	Juraj Slavíček
	Jaroslav Zigo
Vedúci stretnutia:	Alexander Ferenčík
Zapisovateľ:	Juraj Slavíček

Zaznamenaný priebeh stretnutia

Prítomní členovia sa postavili a prediskutovali prácu vykonanú v minulom týždni. Šaňo pracoval na kostre android aplikácie ktorá je skoro hotová. Potrebuje prediskutovať meta informácie ktoré sa budú posielať s úlovkom na server ako aj informácie ktoré zadá používateľ. Juro pracoval na low fidelity modeli a spolu s Jankou pracovali na prihláške na TP cup. Rišo skúmal možnosti využitia frameworkov na backendovej strane.

Diskutoval sa dátový model jeho pripadne updaty. Viac rovnakých reklám na rovnakom mieste na základe GPS následne by sa žiaden nosič z databázy neodstránil ale zlúčili by sa do jedného. K tomuto potrebujeme pomoc od používateľov.

Pri určovní presnej polohy dvoch rovnakej reklamy na dvoch rôznych úlovkoch budeme vyžadovať dvojúrovňovú kontrolu. Jeden user určí polohu a privilegovaní useri to budú určovať. Bolo dohodnuté že v budúcnosti sa spraví jednotný logger pre celý projekt.

Boli diskutované meta dáta odosielané spolu s úlovkom. Mohli by to byt majitelia nosičov,čísla nosičov, typ nosiča....V základnom režime sa používateľovi zobrazí či chce odoslať alebo doplniť informácie a následne ak zvolí doplniť bude mať ešte možnosť otvoriť advanced informácie.

Meta informácie by mohli byť GPS lokalita,model telefónu.

Bol diskutovaný low fidelitz návrh aplikácie. Bolo dohodnuté že je nutné vytvoriť high fidelity návrh pre landing page. Návrh je dobrý ale záložka review ads sa nebude riešiť tabuľkovo a jej layout bude diskutovaný neskôr (pravdepodobne cez fotky.) Na landing page veľké upozornenia na možnosť pozrieť si mapu a zapojiť sa do akcie. Slide show hovoriaca o počte nosičov. Informácie o projekte ľuďoch mestách.

Tím diskutoval vytváranie dokumentácie k projektu. Prešli sa jednotlivé časti a ku každej sa povedali veci ktoré v nej nutne musia byť vypracované.

Šaňo začal kresliť návrh užívateľského rozhrania pre mobilnú aplikáciu ostatní sa zapájali svojimi návrhmi.

Vyplývajúce úlohy

	Úloha	**Odhad času [h]**	**Zodpovedný**
1	Doplnenie dátového modelu	3	Rišo
2	Update kostry serverovej časti	6	Rišo
3	Vytvorenie webového rozhrania	8	Juro
4	Zobrazenie mapy s bodmi	2,5	Tomáš
5	Pridanie bodu do mapy (drag&drop)	4	Tomáš
6	Integrácia medzi front-endom a back-endom	5	Tomáš
7	Upload obrázka prostredníctvom android app	2	Jaro

8	Kostra android app	2	Šaňo
9	Offline režim pre android app	4	Jaro
10	Android: odoslanie voliteľných dát k úlovku	2	Šaňo
11	Update front-endu a databázy	3	Juro
12	Vytvoriť plán na semester	2	Janka
13	Vytvoriť lean canvas	1	Juro
14	Vytvoriť personas	2	Juro

ZÁPISNICA ZO STRETNUTIA ČÍSLO 6

Ukončenie 2. šprintu

Dátum:	6.11.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00
Účastníci:	Jana Egriová Alexander Ferenčík Richard Filipčík Tomáš Melicher Juraj Slavíček Jaroslav Zigo
Vedúci stretnutia:	Jaroslav Zigo
Zapisovateľ:	Jana Egriová

Zaznamenaný priebeh stretnutia

Jaro 6
Tomáš 12
Rišo 11
Janka 7,5
Šaňo 8

Metodiky:
Tomáš – zdrojové kódy, štábna kultúra
Jaro – štábna kultúra XML
Šaňo – metodika testovania
Juro – metodika používania gitu
Rišo – metodika integrácie projektu
Janka – metodika dokumentovania

Diskusia k dátovému modelu

Dva nosiče ich duplicita sa dá vyriešiť iba ich prepojením, ale potom je náročné dopytovanie

Presmerovať z jedného nosiča úlovky na druhý, duplicitný nosič nebudeme mazať, chceme ho ale uchovať, preto duplicitný dostane flag a všetky prepojenia ostávajú

Tabuľka mapovania identít (z opisu v jire) – N nosičov má N úlovkov potreba väzobnej entity

12.11. streda po MSI v estévéčke cca o 18:00 team building

Zvyšok času bol venovaný retrospektíve – zvlášť dokument

Vyplývajúce úlohy

Key	Summary	Assignee
ADHUNTER-39	Obrazovka pre prihlasovanie android +integracia	Jaroslav Zigo
ADHUNTER-38	Definovat kompetencie prihlaseneho a neprihlaseneho	Richard Filipčík
ADHUNTER-37	Ziskat data z databazy o vlastnikovi a odoslat	Richard Filipčík
ADHUNTER-36	integracia web rozhrania s backendom	Juraj Slavicek
ADHUNTER-35	Spracovat data o vlastnikoch pre obrazovku	Tomas Melicher
ADHUNTER-34	WEB vytvorit obrazovku pre volitelne info	Jana Egriova
ADHUNTER-33	Zobrazenie detailu ulovku	Jana Egriova
ADHUNTER-32	Prijat a spracovat data o vlastnikoch	Alexander Ferencik

ADHUNTER-31	Vytvaranie dokumentacie	Jana Egriova
ADHUNTER-30	Backend prihlasenie + registracia	Juraj Slavicek
ADHUNTER-29	Zachytit GPS suradnice	Jaroslav Zigo
ADHUNTER-28	Obrazovka pre zadavanie rozsirenych info o ulovkoch	Alexander Ferencik
ADHUNTER-27	Naplnit databazu datami	Jana Egriova
ADHUNTER-26	Offline rezim	Jaroslav Zigo
ADHUNTER-25	Prerobit personas	Juraj Slavicek
ADHUNTER-24	Vytvrenie planu na semester	Jana Egriova
ADHUNTER-23	Update front end a databazy	Juraj Slavicek
ADHUNTER-22	Odoslanie volitelnych dat v mobilnej aplikacii	Alexander Ferencik
ADHUNTER-13	Doplnit datovy model	Richard Filipčík

ZÁPISNICA ZO STRETNUTIA ČÍSLO 7

Priebeh tretieho šprintu, práca na implementácii

Dátum:	13.11.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00

Účastníci:	Jana Egriová
*chýbajúcich preškrtnúť	Alexander Ferenčík
	Richard Filipčík
	Tomáš Melicher
	Juraj Slavíček
	Jaroslav Zigo

Vedúci stretnutia: Juraj Slavíček
Zapisovateľ: Jaroslav Zigo

Zaznamenaný priebeh stretnutia

Stretnutie prebehlo pomerne rýchlo. Každý člen tímu skonštatoval, ako je na tom s odvedenou prácou a koľko ho ešte čaká, čo treba urobiť do konca šprintu. Rozoberala sa funkcia výberu typu nosičov v aplikácii, z čoho vyplynula potreba prekresliť nosiče aj s panáčikmi, ktorá pripadla Janke. Takisto vyvstala úloha nájsť typických predstaviteľov nosičov, aby užívateľ mohol úlovok priradiť k nejakému predstaviteľovi. Dostali sme sa tiež ku téme „code review“. Zatiaľ sme ho príliš neaplikovali. Preto sme sa rozhodli využiť fakultný projekt PerConIK, vďaka ktorému je možné v spojení s GitHubom code review vykonávať. Zvyšok stretnutia sa venoval práci na implementácii projektu. Každý člen využil zvyšný čas na prácu na svojich projektových úlohách.

Vyplývajúce úlohy

Key	Summary	Assignee
ADHUNTER-39	Obrazovka pre prihlasovanie android +integracia	Jaroslav Zigo
ADHUNTER-38	Definovat kompetencie prihlaseneho a neprihlaseneho	Richard Filipčík
ADHUNTER-37	Ziskat data z databazy o vlastnikovi a odoslat	Richard Filipčík
ADHUNTER-36	integracia web rozhrania s backendom	Juraj Slavicek
ADHUNTER-35	Spracovat data o vlastnikoch pre obrazovku	Tomas Melicher
ADHUNTER-34	WEB vytvorit obrazovku pre volitelne info	Jana Egriova
ADHUNTER-33	Zobrazenie detailu ulovku	Jana Egriova
ADHUNTER-32	Prijat a spracovat data o vlastnikoch	Alexander Ferencik
ADHUNTER-31	Vytvaranie dokumentacie	Jana Egriova
ADHUNTER-30	Backend prihlasenie + registracia	Juraj Slavicek
ADHUNTER-29	Zachytit GPS suradnice	Jaroslav Zigo
ADHUNTER-28	Obrazovka pre zadavanie rozsirenych info o ulovkoch	Alexander Ferencik
ADHUNTER-27	Naplnit databazu datami	Jana Egriova

ADHUNTER-26	Offline rezim	Jaroslav Zigo
ADHUNTER-25	Prerobit personas	Juraj Slavicek
ADHUNTER-24	Vytvrenie planu na semester	Jana Egriova
ADHUNTER-23	Update front end a databazy	Juraj Slavicek
ADHUNTER-22	Odoslanie volitelnych dat v mobilnej aplikacii	Alexander Ferencik
ADHUNTER-13	Doplnit datovy model	Richard Filipčík

ZÁPISNICA ZO STRETNUTIA ČÍSLO 8

Inicializácia 4. šprintu

Dátum:	20.11.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00
Účastníci:	Jana Egriová
Alexander Ferenčík	
	Richard Filipčík
	Tomáš Melicher
	Juraj Slavíček
	Jaroslav Zigo
Vedúci stretnutia:	Tomáš Melicher
Zapisovateľ:	Juraj Slavíček

Zaznamenaný priebeh stretnutia

Na začiatku stretnutia sme vykonali stand up a zosumarizovali sme výsledky šprintu.

Šaňo dorobil obrazovky a rozpracoval upload. Juro dokončil refistráciu a prihlasovanie a integroval ju s webovým rozhraním.

Jaro dokončil lokalizáciu pomocou GPS.

Janka dokončila detail úlovku vo webovom rozhraní a skripty na napĺňanie databáz.

Rišo definoval kompetencie používateľa a upravil dátový model podľa súčasného stavu.

Tomáš zakomponoval výber vlastníkov do procesu pridania billboardov.

Následne každý čln tímu predviedol ostatným prácu na ktorej za posledný šprint pracoval.

Diskusia

Janka nenakreslila miniatúry billboardov vzhľadom na časovú tieseň a táto úloha sa posúva do ďalšieho šprintu. Backendový tím diskutoval samotnú realizáciu unit testovania v jazyku PHP. Jaro a Juro diskutovali o postupe hešovania používateľských hesiel vzhľadom na nautnosť implementácie tejto featury v mobilnej verzii.

Vyplývajúce úlohy

Key	Summary	Assignee
ADHUNTER-47	Nakreslit ikony nosicov	Jana Egriova
ADHUNTER-46	Galeria na mobilnej aplikacii	Jaroslav Zigo
ADHUNTER-45	Vytvorit ikonu pre google play	Jaroslav Zigo
ADHUNTER-44	Vyhladavanie ulovkov na mape	Tomas Melicher
ADHUNTER-43	Editacia ulovkov	Richard Filipcik
ADHUNTER-42	Vytvorit galeriu ulovkov	Juraj Slavicek
ADHUNTER-41	Web App opravenie pozicie billboardu	Tomas Melicher
ADHUNTER-40	Refaktoring kodu	Richard Filipcik
ADHUNTER-39	Obrazovka pre prihlasovanie android +integracia	Jaroslav Zigo
ADHUNTER-36	integracia web rozhrania s backendom	Juraj Slavicek
ADHUNTER-	WEB vytvorit obrazovku pre volitelne info	Jana Egriova

34		
ADHUNTER-32	Prijat a spracovat data o vlastnikoch	Alexander Ferencik
ADHUNTER-26	Offline rezim	Jaroslav Zigo
ADHUNTER-22	Odoslanie volitelnych dat v mobilnej aplikacii	Alexander Ferencik

METODIKA OPISU ÚLOH V SYSTÉME JIRA

Úvod

Táto metodika určuje spôsob a štruktúru opisu úlohy pridelenej členovi tímu v rámci šprintu TP v systéme JIRA. Rovnako určuje štruktúru súhrnu úlohy a komentára pridávanom po úspešnom vyriešení a ukončení úlohy.

Slovník použitých skratiek

- IT - informačné technológie
- TP - tímový projekt

Slovník použitých pojmov

- dashboard - predvolená úvodná stránka po prihlásení do systému JIRA, ktorá formou nástenky zobrazuje súhrn najdôležitejších údajov o projektoch
- JIRA - softvérový nástroj od spoločnosti Atlassian určený pre uľahčenie procesu riadenia projektov
- (textové) pole - v spojení s formulárom predstavuje prvok určený pre vkladanie textu
- úloha - tiež *task*, je úloha zverená konkrétnemu členovi tímu a pridelená v systéme JIRA
- vyriešená úloha - za vyriešenú úlohu sa pokladá úloha, ktorá je okrem samotného vyriešenia problému pokrytá i vhodnou dokumentáciou, prípadne testami

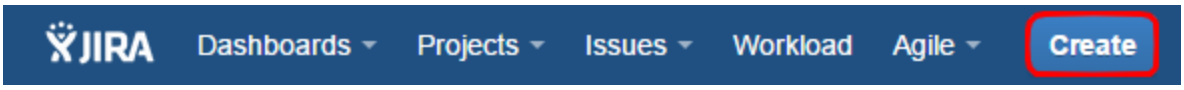
Štruktúra opisu úlohy a súhrnu úlohy

Opísať úlohu a jej súhrn je možné pri vytváraní úlohy, ale i pri úprave informácií o už existujúcej úlohe.

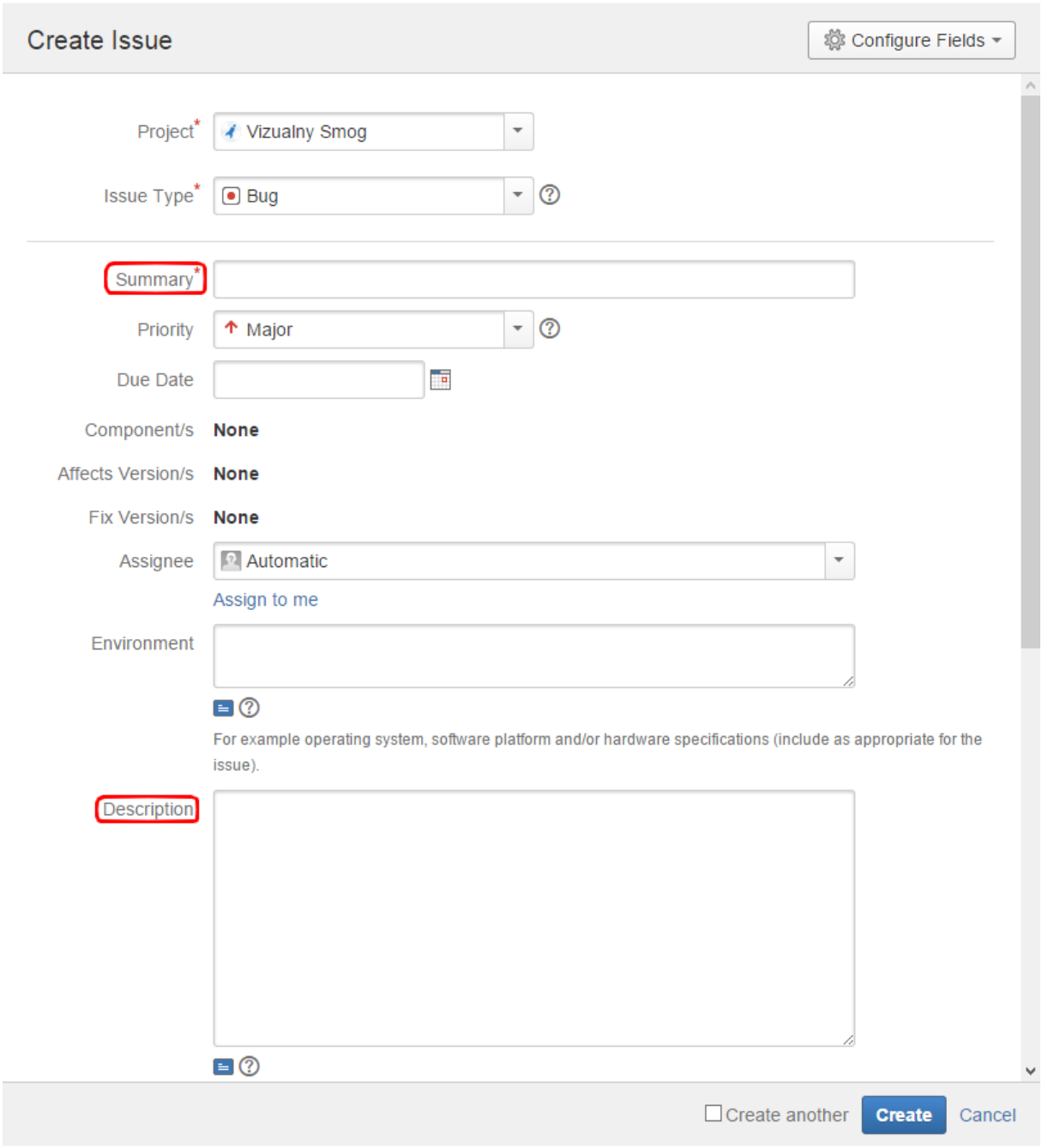
VYTVORENIE NOVEJ ÚLOHY

Po prihlásení do systému JIRA[1] sa (ako predvolená) stránka zobrazí dashboard. V hornej časti stránky je zobrazené základné menu, kde je pomocou tlačidla *Create* možné zobraziť formulár pre vytvorenie úlohy.

[1] http://jira.fiit.stuba.sk/



Vyplnenie väčšiny polí vo formulári je zrejmé zo samotného názvu poľa, preto sa táto metodika zaoberá obsahom tých polí, ktoré predstavujú vlastný opis samotnej úlohy a ktorých štruktúra je dôležitá pre jednoznačnosť a prehľadnosť všetkých detailov vzťahujúcich sa k danej úlohe. Na obrázku nižšie sú vyznačené dve takéto polia (zakrúžkované červenou).



OPIS ÚLOHY

Opis úlohy predstavuje hlavný záchytný bod pri získavaní informácií o úlohe. Vo formulári je reprezentované poľom *Description*. Môže mať rôznu mieru podrobnosti, v ideálnom prípade by mal byť stručný, ale zároveň poskytovať všetky dôležité detaily týkajúce sa danej úlohy. Uprednostňujú sa jednoduché vety s čo najmenším počtom prídavných mien a prívlastkov pred súvetiami. Všetky opisy sa píšu v slovenskom jazyku, používa sa diakritika a dodržiavajú pravidlá slovenského pravopisu.

Opis úlohy sa rozdeľuje na tri časti.

- Úvod opisu poskytuje náhľad do problému naviazanom na danú úlohu. Informuje o probléme, prípadne inom dôvode vzniku úlohy. Môže sa v ňom spomenúť i prípadná nadväznosť na inú úlohu. Nemal by obsahovať viac než 3 vety a nemali by sa v ňom vyskytovať technické detaily úlohy.
- Jadro opisu vysvetľuje celkovú problematiku úlohy, detailnejšie rozvádza úvod, ale opisuje i ďalšie detaily úlohy. Pokiaľ sa úloha skladá z viacerých podúloh, mal by byť ich opis štruktúrovaný v takom chronologickom poradí, v akom sa očakáva ich riešenie. Pre prehľadnosť opisu je vhodné použiť formátovacie možnosti textu systému JIRA, pozri Formátovacia príručka. V jadre opisu sa môžu spomínať aj technické detaily úlohy, nemá však slúžiť ako náhrada dokumentácie.
- Záver opisu nie je povinná súčasť opisu, je však vhodný ako sumarizácia jadra opisu. V závere opisu je tiež vhodné spomenúť všetky časové súvislosti úlohy, ktoré vo formuláre nie je možné uviesť dostatočne detailne - kedy je na úlohe možné začať pracovať, odhadovaný čas potrebný pre jej riešenie, prípadne závislosť na ukončení inej úlohy.

Opis úlohy má slúžiť iba pre informáciu o samotnej úlohe a súvisiacom probléme, nie pre opis spôsobu riešenia!

Jednotlivé časti opisu, predovšetkým jeho jadro, je vhodné členiť na odseky. Ak sa úloha skladá z viacerých podúloh, prípadne je v opise rozvinutých viacero myšlienok, každá by mala byť umiestnená vo svojom vlastnom odseku.

Pri opise úloh, ktoré zaberajú viac priestoru (jadro je členené na 2 a viac odsekov), sa odporúča použiť formátovanie textu. Konkrétne postupy pri formátovaní sú prenechané na intuícii a pocite toho člena tímu, ktorý má na starosti danú úlohu, nižšie je však uvedených niekoľko tipov, ktoré by bolo vhodné dodržiavať.

- Úvod, jadro a záver opisu uviesť pomocou nadpisov (nie však pomocou týchto generických názvov, ale pomocou názvov vzťahujúcim sa k danej úlohe).

- Dôležité údaje v opise (dôležité číselné údaje, dátumy,) písať tučným textom.
- Kľúčové pojmy vzťahujúce sa k TP, ktoré by mohli byť inak chápané aj v inom zmysle (napr. *úlovok*), názvy softvéru, či iné výrazy charakteristické pre oblasť IT písať kurzívou.
- Všetky externé odkazy vkladať pomocou hypertextového odkazu aj s popisom.
- Časti kódov formátovať neproporčným písmom.
- Zoznam podúloh vypisovať pomocou číslovaného zoznamu v poradí, v akom sa očakáva ich plnenie.

Spôsob ako jednotlivé formátovacie efekty dosiahnuť je uvedený v tabuľke *Formátovacie príručka* na konci tohto dokumentu.

SÚHRN ÚLOHY

Súhrn úlohy je vhodné napísať až po dopísaní opisu úlohy. Vo formulári je reprezentovaný poľom *Summary*. Mal by pozostávať predovšetkým z informácií spomenutých v úvode opisu, doplnených o prípadné dôležité informácie spomenuté v závere, či jadre opisu. Odporúčaná dĺžka súhrnu sú 1 až 2 vety.

ÚPRAVA EXISTUJÚCEJ ÚLOHY

V systéme je možné upravovať aj už vytvorenú úlohu. To môže byť užitočné v prípade opravy nájdených chýb, či aktualizácii a doplnení údajov o úlohe.

Upraviť úlohu je možné v projektovej stránke danej úlohy pomocou tlačidla *Edit* v hornej lište. Vo formuláre identickom s tým predošlým vypĺňame polia rovnakým spôsobom, ako pri vytváraní novej úlohy.

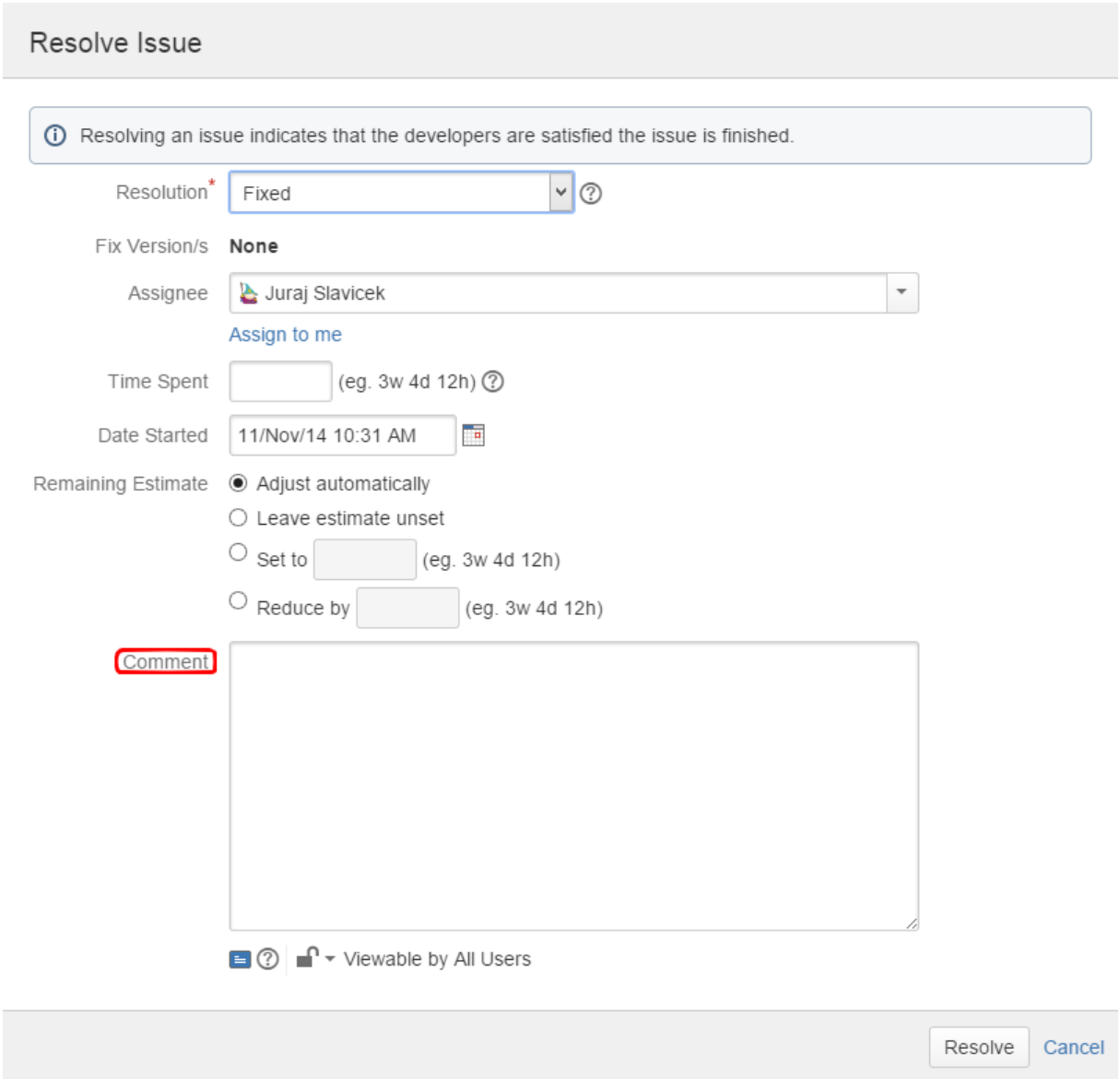
Štruktúra komentára o vyriešení úlohy

Po úplnom vyriešení úlohy je dôležité tento fakt zaznačiť aj v systéme JIRA. Nižšie sa nachádza spôsob písania komentárov o vyriešení úlohy.

VYRIEŠENIE ÚLOHY

Úloha sa v systéme eviduje ako vyriešená pomocou podobného formuláru, aký sa používa pri jej vytváraní. Formulár je možné zobraziť pomocou tlačidla *Resolve Issue* v hornej lište stránky úlohy.

V otvorenom formulári sa okrem ďalších polí, ktorých obsah je opäť zrejmý už len vďaka ich popisu, vyplňuje i pole *Comment*, ktoré má obsahovať komentár detailnejšie informujúci o vyriešení úlohy.



ŠTRUKTÚRA KOMENTÁRA

Účelom komentáre je poskytnutie informácie k stavu úlohy po jej vyriešení. Rovnako ako v predošlej časti platí, že komentár by nemal informovať o spôsobe riešenia a nahrádzať tak iné dokumenty. Obsah komentára možno rozdeliť na dve časti.

Prvá časť informuje o tom, že úloha bola úspešne vyriešená a rozoberá, prečo tak tomu je, teda čo konkrétne sa pri jej riešení podarilo opraviť, vylepšiť, alebo vytvoriť.

Druhá časť informuje o prípadných ďalších nie priamo súvisiacich problémoch objavených počas riešenia, a teda možných nových úlohách. Môže sa tiež spomenúť nadväznosť na nové úlohy vyplývajúce z aktuálnej úlohy.

Komentár by mal byť stručný a uvádzať iba tie najdôležitejšie informácie k vyššie spomenutým častiam. Dĺžka komentára by nemala byť väčšia ako 5 viet. V prípade potreby je opäť možné použiť formátovanie textu, niektoré tipy k jeho používaniu sú uvedené v časti Štruktúra opisu úlohy a súhrnu úlohy - Opis úlohy. Pri písaní komentára platia všetky pravidlá týkajúce sa jazyka a gramatiky, ktoré boli spomenuté pri časti o opise úloh.

Formátovacia príručka

V tejto príručke sú uvedené iba tie najdôležitejšie zápisy pre formátovanie textu, ktoré môžu byť užitočné pri vytváraní opisov úloh a komentárov. Odporúčaný spôsob ich používania je spomenutý v časti venovanej opisu úloh.

Zápis	Efekt	Opis
tučný text	**tučný text**	Vytvorí text písaný tučným písmom.
kurzíva	*kurzíva*	Vytvorí text písaný šikmým písmom.
{{monospace}}	`monospace`	Vytvorí text písaný neproporčným fontom vhodným pre písanie úryvkov zo zdrojových kódov.
h1. Nadpis	Nadpis	Vytvorí nadpis. Možné hodnoty h1 až h6.
[http://fiit.sk]	http://fiit.sk	Vytvorí hypertextový odkaz.
[FIIT|http://fiit.sk]	FIIT	Vytvorí hypertextový odkaz s popisom.

* zoznam
* zoznam
** zoznam
** zoznam
	▪ zoznam
	▪ zoznam
	  ▪ zoznam
	  ▪ zoznam
	▪ zoznam
		Vytvorí nečíslovaný zoznam.

* zoznam		
# zoznam # zoznam # zoznam	1. zoznam 2. zoznam 3. zoznam	Vytvorí číslovaný zoznam.
||hlavička||hlavička|| |bunka|bunka| |bunka|bunka|	**hlavička** **hlavička** bunka bunka bunka bunka	Vytvorí tabuľku.

METODIKA PLÁNOVANIA ŠPRINTOV

Úvod

Cieľom tejto metodiky je definovať postup pri plánovaní šprintu v rámci agilnej metodiky SCRUM. Opisuje činnosti, ktoré s plánovaním projektu súvisia, a to najmä prípravu a rozdeľovanie úloh medzi jednotlivých členov tímu, priebeh stretnutí v polovici aj na konci šprintov a taktiež priebeh kontroly stavu projektu a celkovú rekapituláciu splnených, resp. nesplnených úloh na konci šprintu.

DEDIKÁCIA METODIKY

Keďže SCRUM využíva celý tím, táto metodika je určená pre všetkých členov tímu. Jej praktické využitie je najmä pri šprintových stretnutiach, no jej použitie sa odráža v celom životnom cykle projektu.

NADVÄZUJÚCA METODIKA

Metodika opisu úloh v systéme JIRA - nakoľko je potrebné všetky úlohy, ktoré vznikli na stretnutí, spolu s časom ich trvania evidovať v systéme (každým členom samostatne), možno túto metodiku označiť za nadväzujúcu.

ZOZNAM POJMOV

- *SCRUM* – iteratívna a inkrementálna agilná metóda vývoja softvéru / systému.
- *SCRUM master* – vedúci tímu, dozerá na priebeh vývoja a usmerňuje ho.
- *Šprint* – taktiež *iterácia.* Základná zložka v SCRUM vývoji. Ide o časovo ohraničené obdobie (napr. dva týždne), počas ktorého každý člen tímu pracuje na svojich úlohách a na konci ktorého by mali byť všetky zadané úlohy splnené.
- *Produktový backlog* – Zoznam požiadaviek pre produkt. Zvyčajne písané vo forme „user stories“.
- *Šprintový backlog* – Zoznam úloh, na ktorých tím počas trvania šprintu pracuje.
- *User story* – požiadavka na produkt zapísaná rečou bežného používateľa (teda netechnickým jazykom).
- *Kanban tabuľa* – tabuľa s malými papierikmi / kartičkami prerozdelenými v štyroch častiach: User Stories, To Do, In Progress, Done.

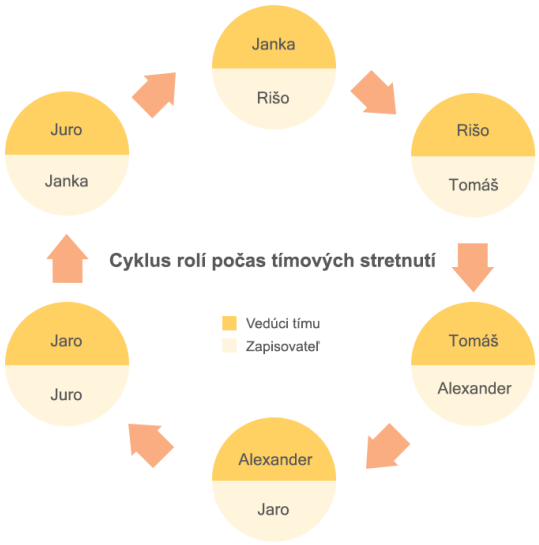
Priebeh stretnutí

Tímové stretnutia trvajú 160 minút a konajú sa každý týždeň vo štvrtok, pričom začínajú **o 8:00 hod**. Ide o dva typy strenutí: **šprintové** a **medzišprintové** stretnutia, ktoré sa navzájom striedajú; jeden týždeň prebieha medzišprintové stretnutie, druhý šprintové. Pred začatím tímového stretnutia je potrebné, aby si každý člen tímu vopred pripravil informácie o tom:

- na akých úlohách pracoval,
- koľko času musel vynaložiť na vykonanie týchto úloh,
- celkové zhodnotenie vykonanej práce – ktoré úlohy zvládal ľahko, prípadne na aké problémy natrafil a ako mieni pri ich výskyte v budúcnosti postupovať.

Samotné stretnutie prebieha formou diskusie, ktorú vedie vždy jeden určený člen tímu, ktorý bol na poslednom stretnutí zapisovateľom. Každý týždeň sa jeden člen tímu dobrovoľne prihlási a prevezme na seba túto úlohu zapisovateľa na ďalší týždeň. Pre úlohu vedúceho stretnutia a zapisovateľa platia tieto dve pravidlá:

1. *zapisovateľom* musí byť počas šiestich týždňov pri šiestich členoch tímu každý člen **práve raz,**
2. *vedúcim tímu* musí byť počas šiestich týždňov pri šiestich členoch tímu každý člen **práve raz.**



Obrázok č. 1: Cyklus rolí počas tímových stretnutí

ŠPRINTOVÉ STRETNUTIE

Šprintové stretnutie je pre členov záväznejšie ako medzišprintové, pretože členovia musia vykázať svoju vykonanú prácu SCRUM masterovi predvedením funkčnosti aplikácie. Na týchto stretnutiach sa hodnotí priebeh šprintu a zadeľujú sa úlohy pre ďalšie dva týždne.

Proces zadeľovania úloh prebieha nasledovne:

- na základe diskusie SCRUM master vypíše na kartičky User Stories, ktoré nalepí na Kanban tabuľu,
- členovia tímu z nich vytvoria úlohy,
- každý člen tímu si zoberie 2-3 úlohy, ktoré najviac súvisia s jeho zameraním.

Na rozdiel od medzišprintového stretnutia, kde SCRUM master zriedkavejšie zasahuje do jeho chodu, počas šprintového stretnutia kladie SCRUM master veľa otázok týkajúcich sa splnených/nesplnených úloh, v prípade nesplnených úloh hľadá dôvody ich nesplnenia a aktívne sa zapája do návrhu nových funkcionalít.

Priebeh šprintového stretnutia		
1.	**Vedúci tímového stretnutia**	Vyzve všetkých členov k rekapitulácii šprintového obdobia.
2.	**Členovia tímu**	*Zhodnotia:*
		- na akých úlohách pracovali
		- koľko času museli vynaložiť na vykonanie týchto úloh
		- celkovú vykonanú prácu; ak nesplnili niektoré úlohy, zdôvodnia prečo.
3.	**Vedúci tímového stretnutia**	Vyzve členov k upraveniu Kanban tabule - splnené úlohy presunúť do časti DONE, prípadne aj User Stories (ak ich naozaj možno považovať za hotové).
4.	**Vedúci tímového stretnutia**	Začne sa pýtať na čiastkové problémy u jednotlivých členov tímu.
5.	**Členovia tímu**	Snažia sa spoločne nájsť riešenia otvorenou diskusiou, prípadne priniesť do existujúcich riešení vylepšenia.
6.	**Členovia tímu**	Po skončení diskusie vznikne nová diskusia, kde tím spoločne navrhuje nové funkcie do aplikácie, nové UI. Prebieha vo forme brainstormingu, kreslí sa na tabuľu, zapisuje na papier.
7.	**SCRUM master**	Vypíše nové User Stories na papieriky, ktoré následne prilepí na tabuľu.
8.	**Členovia tímu**	Členovia tímu si spoločne vypíšu na papieriky 2-3 nové úlohy, ktoré vznikli z diskusie a novovytvorených User Stories. Ku každej úlohe napíšu približný odhadovaný počet hodín potrebných na ich dokončenie.
9.	**Vedúci tímového stretnutia**	Spýta sa členov tímu na prípadne otázky. Ak nie sú otázky a Kanban tabuľa je patrične upravená, ukončí stretnutie.

Tabuľka č. 1: Priebeh šprintového stretnutia

MEDZIŠPRINTOVÉ STRETNUTIE

Medzišprintové stretnutie sa uskutočňuje v polovici šprintového obdobia. Je orientované viac prakticky (implementačne) ako šprintové stretnutie. Jeho priebeh je podobný šprintovému stretnutiu, avšak s niekoľkými úpravami. Nie je ani natoľko záväzné ako šprintové stretnutie, pri ňom však tiež platí, že členovia tímu musia zrekapitulovať svoju týždňovú prácu na projekte a vytvoriť časový odhad pre ďalší týždeň.

Zložky tímových stretnutí		
	Medzišprintové stretnutie	*Šprintové stretnutie*
Rekapitulácia šprintového obdobia	•	•
Úprava Kanban tabule (TODO -> DONE)	•	•
Diskusia vzniknutých problémov	•	•
Návrh nových funkcií, prípadne UI	**X**	•
Vytváranie nových User Stories	**X**	•
Code review	•	**X**
Pair programming	•	**X**
Programovanie / implementácia	•	**X**

Tabuľka č. 2: Zložky tímových stretnutí

ČASOVÝ ODHAD A PREROZDELENIE ÚLOH

Odporúčaný čas pre vykonávanie projektových úloh je pre každého člena **8 hodín týždenne.** Keďže každý člen je povinný vykonať časový odhad pre svoje úlohy do ďalšieho týždňa, mal by tento odporúčaný čas zohľadniť. Pokiaľ z povahy úloh vyplynie, že jeden člen tímu má práce na nasledujúci týždeň podstatne viac ako druhý, je možné tieto úlohy prerozdeliť tak, aby každý člen mal pridelené približne rovnaké množstvo práce.

ABSENCIA ČLENOV TÍMU NA STRETNUTÍ

V prípade, že sa niektorý člen tímu nemôže na stretnutie dostaviť, je povinný oznámiť svoju neprítomnosť vedúcemu tímového stretnutia, ktorého sa nemôže zúčastniť. Ak ide o člena, ktorý mal v dané stretnutia zastávať rolu zapisovateľa alebo vedúceho, je možné urobiť výnimku a namiesto neho môže danú rolu zastávať ľubovoľný člen (aj v prípade, že v danom šesťtýždňovom cykle už danú rolu zastával). Na ďalšom stretnutí sa vystriedajú.

METODIKA PÍSANIA ZDROJOVÝCH KÓDOV

Základné ustanovenia

Táto metodika slúži na detailný popis písania zdrojových kódov. V tomto dokumente je popísaný projekt „Dav proti vizuálnemu smogu“, ktorý vznikol v rámci predmetu tímový projekt. Projekt pozostáva zo serverovej a mobilnej časti, pričom cieľom tohto dokumentu je zachovať konzistenciu v zdrojových kódoch medzi uvedenými modulmi. V serverovej časti je používaný jazyk php a v mobilnej časti jazyk java, preto budú uvedené príklady popísané v oboch jazykoch. Veci nenachádzajúce sa v tomto dokumente platia podľa všeobecných pravidiel pre štylizáciu kódu, ktoré možno nájsť pre jazyk java na adrese

https://google-styleguide.googlecode.com/svn/trunk/javaguide.html

a pre jazyk php na adrese http://pear.php.net/manual/en/standards.php.

SLOVNÍK POJMOV A SKRATIEK

Nasledujúce tabuľky popisujú význam pojmov a skratiek vyskytujúcich sa v tejto metodike.

Pojem	Význam
lowerCamelCase	Text s malým začiatočným písmenom, vynechanými medzerami a každé slovo okrem prvého je odlíšené veľkým začiatočným písmenom.
UpperCamelCase	Text, v ktorom je každé slovo odlíšené veľkým začiatočným písmenom.
lowercase	Text, pozostávajúci iba z malých písmen.
UPPERCASE	Text, pozostávajúci iba z veľkých písmen.
jednoduché zátvorky	Ide o znaky (a), používajú sa napríklad pri deklarácii funkcie.
hranaté zátvorky	Ide o znaky [a], používajú sa napríklad pri pristupovaní prvku poľa na základe jeho indexu.
zložené zátvorky	Ide o znaky { a }, používajú sa napríklad pri deklarovaní tela funkcie.
lomené zátvorky	Ide o znaky < a >, používajú sa napríklad pri deklarovaní listu v jazyku JAVA.

Skratka	Význam
ID	identifikátor
SQL	Structured Query Language

Štylizácia zdrojových kódov

ZÁPIS DEFINOVANÝCH PREMENNÝCH

a) Premenná je definovaná zápisom lowerCamelCase, pričom skratky ako napríklad ID sú definované zápisom UPPERCASE.

Nesprávne zápisy	Správne zápisy
`$Counter = 0;`	`$counter = 0;`
`int MarkerPosition = null;`	`int markerPosition = null;`

b) Premenná je vždy v anglickom jazyku, pričom špecifické pojmy pre projekt „Dav proti vizuálnemu smogu“ ako napríklad „úlovok“ sa neprekladajú. Píšu sa však bez diakritiky. Všetky výrazy, ktoré sa v zdrojových kódoch neprekladajú sú spísané v dokumente s názvom „Slovník metafor“.

Nesprávne zápisy	Správne zápisy
`$catchID = 0;`	`$ulovokID = 0;`
`int webUserName = "";`	`int domasedName = "";`

ZÁPIS DEFINOVANÝCH PROCEDÚR

Procedúra sa definuje pomocou kľúčového slova, názvu procedúry a zoznamom jej parametrov. Kľúčové slovo pre definovanie procedúry a údajové typu premenných, ktoré boli zadané ako argumenty funkcie majú predpísanú formu v špecifikácii konkrétneho programovacieho jazyka. Táto kapitola popisuje formu zadávania názvu procedúry, forma zadaných argumentov je rovnaká ako pri definovaných premenných, ktoré boli popísané v predchádzajúcej kapitole. Názov funkcie je definovaný zápisom lowerCamelCase.

Nesprávne zápisy	Správne zápisy
`function GetUlovokByName($name)`	`function getUlovokByName($name)`
`int MarkerPosition = null;`	`int markerPosition = null;`

ZÁTVORKOVANIE

Ak telo cyklu, prípadne podmienky má len jeden riadok, zo špecifikácie jazykov PHP a JAVA nie je nevyhnutné používanie zložených zátvoriek, zadávajú sa aj v takomto prípade. V prípade zložených zátvoriek je začiatočná aj koncová zátvorka umiestnená na samostatnom riadku.

Daný zápis sa používa pri deklarácii funkcií a tried a pri zápise podmienok a cyklov. Toto pravidlo neplatí pre ostatné typy zátvoriek.

Nesprávne zápisy	Správne zápisy

function getUlovokByName() {
return "name";
 	
}
	function getUlovokByName()
	{
	return "name";
	
	}

void hello() {
echo "hello";
 	
}
	void hello()
	{
	echo "hello";
	
	}

POUŽÍVANIE MEDZIER

Medzery sa používajú na oddelenie jednotlivých argumentov funkcie, kvôli prehľadnosti a pri priradzovaní hodnoty premennej. Jednotlivé argumenty funkcie sú oddelené čiarkou, za ktorou nasleduje jedna medzera. V prípade cyklu `for` sú jednotlivé časti hlavičky cyklu štruktúrované podobne ako argumenty funkcie, pričom sú oddelené bodkočiarkou a za ňou je jedna medzera. Dané pravidlo platí pre deklaráciu funkcie aj pre volanie funkcie. Medzery sa nepoužívajú za kľúčovými slovami pre definovanie cyklu prípadne podmienky.

Nesprávne zápisy	Správne zápisy
`printSomething($color1,$color2);`	`printSomething($color1, $color2);`
`for (i=0;i<10;i++)`	`for(i = 0; i < 10; i++)`
`if( a == 5 )`	`if(a == 5)`

Medzery v logických výrazoch

Pri priradeniach sú priraďujúce operátory ‚=', ‚+=' a ‚-=' oddelené jednou medzeru vpredu a jednou vzadu. Porovnávacie operátory (‚==', ‚<', ‚<='...) a logické operátory (‚||', ‚&&'...) sa zapisujú rovnako ako priraďujúce, sú teda oddelené jednou medzerou vpredu aj vzadu. Binárne operátory (‚^', ‚&', ‚|') nie sú oddelené medzerou.

Nesprávne zápisy	Správne zápisy
`if((a == 5)||(a == 7))`	`if((a==5)||(a==7))`

ZÁPIS SQL DOPYTOV

Na zápis kľúčových slov SQL dopytov sa používa zápis UPPERCASE. Operátory v SQL dopytoch sú oddelené jednou medzerou vpredu aj vzadu.

Nesprávne zápisy	Správne zápisy
`stmt.executeQuery("select * from reklamy where id = 10");`	`stmt.executeQuery("SELECT * FROM reklamy WHERE id = 10");`
`$sql->query("UPDATE nosice SET nazov='skusobny' WHERE id=10")`	`$sql->query("UPDATE nosice SET nazov = 'skusobny' WHERE id = 10")`

KOMENTÁRE V ZDROJOVÝCH KÓDOCH

Každá funkcia musí byť okomentovaná, čo je jej úlohou, pričom je potrebné dodržať formu daného komentára. Ide o blokový komentár (ohraničený sekvenciami znakov „/* a „*/“), pričom každý riadok začína medzerou, znakom ‚*‘ a ďalšou medzerou. Okrem popisu samotnej funkcie, musia byť uvedené aj vstupné a výstupné parametre. Komentáre sa píšu v slovenskom jazyku bez diakritiky.

Nesprávne zápisy	Správne zápisy

`// Funkcia vypise sucet zadanych cisel`
`// cislo1: prvy scitanec`
 	
`// cislo2: druhy scitanec`
 	
`function getUlovokByName(cislo1, cislo2)`
 	
`{`
 	
`return "name";`
 	
`}`
	`/*`
	
	` * Funkcia vypise sucet zadanych cisel`
	` *`
	
	` * cislo1: prvy scitanec`
	
	` * cislo2: druhy scitanec`
	
	`*/`
	
	`function getUlovokByName(cislo1, cislo2)`
	
	`{`
	
	`return "name";`
	
	`}`

V zdrojovom kóde je možné používať blokové i riadkové komentáre v závislosti od dĺžky komentára. Zápis blokového komentára v zdrojovom kóde je rovnaký ako popis pre funkciu a zápis riadkového komentára je spravidla odsadený potrebným počtom tabulátorov tak, aby bol jasne oddelený od zdrojového kódu.

Nesprávne zápisy	Správne zápisy

`int i = 5;// defaultny pocet nosicov`
 	
`/*`
`Tento cyklus kontroluje vsetky nosice`
`*/`
`for (i=0;i<10;i++)`
	`int i = 5;  // defaultny pocet nosicov`
	
	`/*`
	`  Tento cyklus kontroluje vsetky nosice`
	`*/`
	`for (i=0;i<10;i++)`

ODSADENIE V ZDROJOVÝCH KÓDOCH

Na odsadenie sa používa znak tabulátor namiesto medzier.

Nesprávne zápisy	Správne zápisy

`function getUlovokByName(cislo1, cislo2)`
`{`
`  return "name";`
`}`
	`function getUlovokByName(cislo1, cislo2)`
	`{`
	`  return "name";`
	`}`

METODIKA TESTOVANIA

Úvod

Úvodom si zhrňme fakty o testovaní softvéru ako takého. Definícia pojmu softvérové testovanie znie: „Testovanie softvéru je súbor procesov slúžiacich na kontrolu kvality softvérového produktu, ktorých cieľom je dosiahnutie požadovanej kvality softvéru z hľadiska funkčnosti, spoľahlivosti, výkonnosti, použiteľnosti a podporovateľnosti. Kontrola kvality sa môže uskutočniť či už pre jednotlivé časti informačného systému alebo pre systém ako celok.“ Testovanie vykonávame z dôvodu overenia, či naša odvedená práca neobsahuje funkcionálne chyby (bugy), grafické, rýchlostné alebo iné nedostatky softvéru.

Touto metodikou sa riadi vývojár po implementovaní novej časti aplikácie, ktorú má záujem použiť v projekte. Pokyny tejto metodiky stanovujú kedy je daná časť implementovanej funkcionality otestovaná – pripravená na nasadenie, a kedy naopak otestovaná (hotová) nie je a je potrebná jej ďalšia úprava pred nasadením.

Front-end časť aplikácie

Front-endovú časť aplikácie tvorí časť projektu, s ktorou používateľ prichádza priamo do styku vizuálne. V tejto časti záleží na funkcionalite, ale aj vzhľade, ktorý by mal byť podľa správnosti jednoduchý na manipuláciu, flexibilný a umožňujúci rýchlu manipuláciu.

MOBILNÁ APLIKÁCIA

Táto časť aplikácie je vyvíjaná pre platformu Android v jazyku Java. Pri vyvíjaní novej funkcionality, alebo novej obrazovky sa môže jednoducho stať, že vývojár vo svojom kóde spraví či už algoritmickú, grafickú, alebo formulačnú chybu. Po každom vyprodukovaní novej časti, je teda dôležité pred nasadením danú časť náležite otestovať.

Pridávanie viditeľnej funkcionality

Po implementovaní novej funkcionality ju vývojár otestuje v prvom rade priamo v mobilnej aplikácií. Túto možnosť má prostredníctvom vývojárskeho prostredia Android Studio, kde novú verziu aplikácie má možnosť spustiť priamo v pripojenom mobilnom zariadení. Táto funkcionalita by mala byť viditeľne zobrazená a teda vývojár jasne vidí, či spĺňa požiadavky na svoju funkčnosť. Ak je táto funkčnosť dodržaná, môžeme funkcionalitu považovať za otestovanú. V prípade ak sa nájdu akékoľvek nedostatky, je potrebná ďalšia úprava danej funkcionality.

Pridávanie neviditeľnej funkcionality

Pri vyvíjaní časti mobilnej aplikácie, ktorá nie je priamo viditeľná, je náročnejšie jej otestovanie. Môže sa jednať o určitý výpočet na pozadí, ktorý sa ukladá, alebo o funkcionalitu pri ktorej si vývojár nie je istý, či sa naozaj vykonala. Pri tomto testovaní vývojár po spustení mobilnej aplikácie vykoná akcie, ktoré by mali funkcionalitu aktivovať. Následne však nevidí, či funkcionalita funguje podľa požiadaviek. V tomto prípade, vývojár využije textový súbor s názvom vo forme TEST_*NázovFunkcionality_číslоTestu.txt*. V kóde vývojár zadá v rámci rozsahu testovanej funkcionality isté kontrolné body, v ktorých sa bude do tohto textového súboru zapisovať aktuálny stav prebiehania funkcionality. Tieto priebežné hlásenia sú formulované a v priebehu programu umiestnené tak, že informujú vývojára o hodnotách dôležitých atribútov a o aktuálnej činnosti prebiehajúcej v programe. Do textového súboru sa pridávajú v poradí v akom boli zaznamenané. Vývojár tak odsleduje, čo sa presne v aplikácii udialo a či sa funkcionalita vykonala správne. Toto však treba vykonať pre všetky možné situácie, respektíve pre všetky možné skupiny situácii tak, aby bola skontrolovaná každá možnosť pri ktorej môže dôjsť k chybe. Ak tieto záznamy z priebehov danej funkcionality zodpovedajú požadovanému stavu, môžeme ju považovať za otestovanú. Ak nie, sú potrebné ďalšie úpravy v kóde a následné opätovné pretestovanie.

Pridávanie novej obrazovky

Pre otestovanie novo navrhnutej obrazovky pre mobilnú aplikáciu, je potrebné otestovať najmä všetky nové atribúty tejto obrazovky. Po spustení vývojár teda v prvom rade prepne aplikáciu do danej obrazovky, po čom by mala byť následne zobrazená. Skontroluje, či sa zobrazia všetky komponenty a či sú aj adekvátne rozmiestnené. Ak áno, nasleduje test jednotlivých komponentov. Každý komponent treba odskúšať z hľadiska splnenia jeho funkcionality v rámci aplikácie. Tlačidlá, textfieldy, selectboxy, atď. musia spĺňať zadané požiadavky. V prípade, ak každý komponent na danej obrazovke funguje podľa požiadaviek, môžeme obrazovku považovať za otestovanú. V prípade ak nie, sú potrebné ďalšie úpravy na zabezpečenie požadovaného stavu.

WEBOVÉ ROZHRANIE

Táto časť je vyvíjaná pre používateľa, ktorý má aplikáciu spustenú na domácom osobnom počítači. Je vyvíjaná v jazyku HTML. Opäť je dosť možné vývojárove pochybenie či už algoritmické, grafickej alebo formulačné.

Pridávanie novej obrazovky

Pre webovú aplikáciu je dôležité aby každý novopridaný komponent bol správne umiestnený podľa potrieb a podľa návrhu, vývojár teda po spustení web aplikácie skontroluje najprv umiestnenie nových komponentov. Ak po preskúmaní rozmiestnenia nie je nájdený žiaden konflikt, môže sa pristúpiť k testovaniu funkcionality jednotlivých komponentov, ktoré prebieha podobne, ako v časti mobilnej aplikácie.

Pridávanie nových funkcionalít

Po implementovaní novej funkcionality pre webové rozhranie, ho vývojár otestuje spustením priamo vo webovej aplikácii, podobne ako v mobilnej a následne skontroluje, či sa všetky potrebné dáta správne uložili do databázy. Ak sa jedná o funkcionalitu ktorá je neviditeľná postupuje podobne ako pri mobilnej aplikácii, s následnou kontrolou údajov v databáze.

Back-end časť aplikácie

Back-end je časť aplikácie, ktorá sa v programe nachádza, ale používateľ ju nevidí, respektíve si nie je vedomý tejto časti. Táto časť riadi procesy, ktoré sa starajú o chod webového rozhrania a prepojenie Android časti s databázou. Aby v tejto časti nenastali chyby, vývojár musí svoju odvedenú prácu dôkladne otestovať, čo je pre túto časť ešte dôležitejšie ako pre front-endovú, nakoľko chyby ktoré tu môžu nastať môžu mať oveľa poškodzujúcejšie následky.

SERVEROVÁ ČASŤ

Server pre našu aplikáciu slúži primárne ako sprostredkovateľ databázy pre front-end. Dáta z databázy sú mnohokrát zobrazované, alebo naopak dáta zaznamenané z front-endu sa prostredníctvom serveru ukladajú do databázy.

Ukladanie údajov

Ak vývojár pracuje so zápisom určitých dát do databázy. Ktoré sa z front-endu ukladajú. Je porebné, aby si otestoval správne uložene týchto dát ktoré v aplikácii zadal a či jeho funkcionalita zaznamenávania údajov na server naozaj prebehla. A teda skontroluje v príslušnej aplikácii, či sa dané údaje, ktoré mali byť uložené na server, dajú aj spätne zobraziť. V tomto prípade môžeme ukladanie údajov považovať za otestované.

Update údajov

Podobný princíp platí pri updatovaní uložených údajov. Ak vývojár potrebuje updatovať niektorý údaj v databáze, pre správnosť vždy overí v príslušnej aplikácii, či po pokuse o update údajov a uloženie, sa údaje naozaj uložili na serveri a teda, či je možné updatnuté údaje zobraziť v novej verzii. Ak áno, môžeme update považovať za otestovaný. Ak však nie, je potrebné kód správne upraviť a update znova pretestovať.

Merge údajov

V aplikácii bude funkcionalita, ktorá bude sprostredkovávať mergovanie dvoch záznamov z entity zaznamenaného billboardu. Jedná sa o zneaktívnenie jedného záznamu, a ponechanie len jedného, s príznakom ako aktívny. Toto je špecifická časť aplikácie, nakoľko tu môže nastať veľa rôznych chýb. Testovanie spočíva vo vyhľadaní daných záznamov v aplikácií, kde by mal byť po zmergovaní už vyobrazený len jeden a to ten, ktorý bol označený používateľom ako aktuálny. Test mergovania zaznamenaných billboardov bude potrebné vykonávať vždy po zásahu do tejto funkcionality, nakoľko problémy ktoré by tu mohli vzniknúť, by boli veľmi závažného charakteru pre našu databázu údajov.

Delete údajov

Pri vyvíjaní funkcionality, ktorá z nejakého dôvodu zmazáva záznamy. Je potrebné overovať, ako sa výsledok prejaví v aplikácii. Vývojár teda zapne obe aplikácie, kde vykoná akcie ktoré spustia funkcionalitu, ktorá zmazáva údaje. Následne overí, či sa dané údaje už nezobrazujú v aplikáciách. Následne prostredníctvom databázového klienta overí, či sa dané údaje naďalej vyskytujú v databáze, teraz už však s príznakom: Neaktívny.

Zmeny v databázovom modeli

Ak niektorý člen tímu vykonával zmeny týkajúce sa databázového modelu, môže veľmi ľahko dôjsť k vzniknutiu chybového stavu. Je teda dôležité, aby tieto zmeny boli adekvátne otestované. Po každej zmene je potrebné vykonať otestovanie zmenenej časti, alebo celého modelu, podľa potreby.

PRIDANIE NEZÁVISLÉHO ATRIBÚTU

Po pridaní nového atribútu do databázovej entity, ktorý žiadnym spôsobom neovplyvňuje zvyšok databázového modelu, nie sú potrebné rozsiahle testovania. Vývojár jednoducho spustí príslušnú aplikáciu a pokúsi sa o zápis danej hodnoty a jej uloženie, napríklad pri pridávaní nového atribútu k fotke billboardu, vývojár zaznamená nový záznam a pridá k nemu informáciu,

ktorá prislúcha novému atribútu v databáze. Následne overí, či sa daná hodnota adekvátne uložila. V niektorých prípadoch je možné overenie priamo v aplikácii – napríklad galéria záznamov, avšak toto nie je vždy možné a je potrebné skontrolovať uloženie danej hodnoty priamo v databáze, čo vývojár zrealizuje prostredníctvom databázového klienta a zadaním potrebného príkazu SELECT. V prípade že sa hodnota uložila správne, môžeme atribút považovať za správne pridaný. Ak nie, je nutné doladiť chybu v pridanom atribúte a test opakovať.

PRIDANIE ZÁVISLÉHO ATRIBÚTU

Často krát sa v databázovom modeli vyskytuje atribút, na ktorom závisia aj iné entity, ako samotná entita v ktorej sa atribút nachádza. V tomto prípade je potrebné odskúšať aj vzťahy medzi dotyčnými entitami, či spĺňajú svoju funkciu podľa potreby. Podstatný je hlavne vzťah medzi nimi a test by sa mal zamerať práve naň a na atribút – kľúč, ktorý tento vzťah sprostredkuje. Vývojár teda prostredníctvom aplikácie uloží nový záznam, ktorý nadväzuje na ďalšiu entitu (rodičovskú / dcérsku). Potom overí, či naozaj tieto informácie v aplikácii prislúchajú k sebe tak, ako je potrebné. Toto taktiež však nie je vždy možné prostredníctvom aplikácie a v tom prípade vývojár overí vzťah prostredníctvom databázového klienta. Na toto overenie je potrebný príkaz SELECT a JOIN, kde zobrazí hodnoty obidvoch prislúchajúcich entít, ktoré by mali byť na seba správne naviazané. Ak sú výsledné zobrazené hodnoty zodpovedajúce realite, môžeme atribút považovať za správne pridaný. Ak nie, je nutné doladiť chybu v pridanom atribúte a test opakovať.

PRIDANIE NOVEJ ENTITY

Po pridaní novej entity je potrebné pretestovať uloženie všetkých jej atribútov, či už obyčajných - nezávislých, alebo závislých - kľúčov (privátnych / cudzích), podľa popisu uvedeného vyššie. Po tom ako všetky atribúty splnia svoje požiadavky na funkcionalitu, môžeme považovať entitu za otestovanú. Ak však niektorý z atribútov spôsobuje chyby, je nutná oprava a následné pretestovanie.

ROZSIAHLA ÚPRAVA DATABÁZOVÉHO MODELU

V prípade, ak je potrebná zmena výraznejšia zmena vzťahov, entít atribútov v databázovom modeli, alebo vytvorenie jeho novej verzie, je potrebné do testovania zahrnúť celú pokrytú časť zmien a najlepšie zbežné pretestovanie celej aplikácie. Vzťahy medzi entitami v aplikácii môžu totiž navzájom rôzne súvisieť, čo vývojára nemusí pri priamej manipulácii napadnúť, avšak pri špecifickej situácii môže nastať chybový stav. Preto po vykonaní zmien, ktoré manipulovali vo väčšom rozsahu s databázovým modelom, treba v aplikácii vykonať úkony zahŕňajúce všetky

súvisiace entity, aj keď len okrajovo, aby sa predišlo neskoršiemu zisteniu chýb v modeli. Vývojár teda v aplikácii vytvorí niekoľko nových záznamov, navzájom prepojených vzťahmi odrazenými v dátovom modeli, čo by sa všetko následne malo prejaviť ako funkčné po skontrolovaní v aplikácii a tiež pomocou databázového klienta.

METODIKA POUŽÍVANIA VERZIOVACIEHO SYSTÉMU GIT

Úvod

Nasledujúci dokument pojednáva o metodike používania distribuovaného verziovacieho open source systému Git. Metodika sa aplikuje v rámci predmetu Tímový projekt na projekt „Dav proti vizuálnemu smogu“ .Dokument si kladie za cieľ uviesť potenciálne nového člena tímu do problematiky a tak zaručiť konzistentnosť používania verziovacieho systému naprieč celým tímom. V tíme je momentálne používaný systém GitHub ale metodika je aplikovateľná aj na ostatné podobné systémy ako napríklad BitBucket. Projekt je rozdelený na 2 logické jednotky, mobilnú a webovú aplikáciu ktoré sa nachádzajú v samostatných priečinkoch v spoločnom repozitári. V tomto dokumente nebude rozoberaná samotná používateľská príručka k verziovaciemu systému . Túto je možné nájsť napríklad na adrese https://help.github.com/.

Slovník Pojmov

Nasledujúca kapitola obsahuje slovník pojmov a ich vysvetlení týkajúcich sa verziovacieho systému ktoré sú použité v tomto dokumente.

Pojem	Vysvetlenie
Prispievateľ	Člen tímu s prístupovými právami na pridávanie zdrojových kódov do verziovacieho systému
Repozitár	V drvivej väčšine prípadov sa jedná o online úložisko všetkých zdrojových kódov v rámci projektu chránené heslom.
Vetva	Paralelná verzia zdrojového kódu určená napríklad na vývoj novej funkcie systému alebo prácu špecifického prispievateľa
Master Vetva	Hlavná vetva vyvíjanej aplikácie všetky. Prípadné releasy aplikácie sa vždy robia z tejto vetvy. Z každej inej vetvy sa dá spätne

	dostať do vetvy Master.
Devel Vetva	Vetva odštepené od vetvy master. Je určená na vývoj a prototypovanie nových funkcii. Pri určitých dôležitých míľnikoch(umiestnenie mobilnej aplikácie na Google Play, Prvá stabilná verzia webovej aplikácie) v projekte je spájaná s vetvou master
Commit	Jedna z verzii systému v ktorá je pozmenená oproti verzii predchádzajúcej. Je jednoznačne identifikovateľná podľa hashu. Nesie v sebe informácie o prispievateľovi, čase a dátumu pridania a commit správe.
Commit správa	Krátka správa o poúslednom commite jednoznačne popisujúca vykonané zmeny v repozitári
Merge	Spojenie 2 vetiev do jednej z nich

Metodika písania commit správ

V tejto sekcii je opísaná metodika písania commit správ. Tieto správy sú kľúčové pre dohľadávanie kto kedy a v akej náväznosti zmenil obsah repozitáru a prípadne k akým bugom to viedlo. Všeobecné rady sú spísané na oficálnej stránke Githubu https://github.com/erlang/otp/wiki/Writing-good-commit-messages . Náš tím z nich určitú časť preberá ale obaľuje ich do vlastnej štruktúry vysvetlenej nižšie. V commit správach je dovolené a častokrát pre zostručnenie opisu commitu až nutné používať rôzne skracovacie výrazy a skratky vzhľadom na to že maximálna odporúčaná dĺžka commit správ uvedená napríklad v nástroji GitBash je **64 znakov**.

PRAVIDLÁ PRE PÍSANIE COMMIT SPRÁV

- Jednotlivú commit správu začíname identifikátorom z Issue tracking systému JIRA a číslom issue - pravidlo zvyšuje prehľadnosť a prináša možnosť vyhľadávania commitov na základe úlohy alebo bugu
- Nasleduje identifikátor časti systému ktorej sa daný commit týka. Zvolíme buď MA alebo WA ako akronymy pre webovú resp. mobilnú aplikáciu
- Nasleduje sloveso v minulom čase oddelené od predchádzajúcej častui medzerou a pomlčkou opisujúce vykonanú operáciu na isuue v JIRE - typicky sú to slovesá implementované, opravené, otestované, upravené, doplnené atp. Môže sa použiť aj anglický ekvivalent
- Commit správa končí opisom časti ktorá bola z issue spravená v prípade že bola v rámci jedného commitu splnená celá issue môže to byť aj jej titulok z JIRY

Výsledná ukážková správa pre commit ktorý opravoval registráciu pre issue číslo 30 „Backend prihlasenie + registrácia používateľa“ v projekte AdHunter na webovej časti aplikácie.

[ADHUNTER-26] -WA- Opravená registrácia používateľa

Commit správa pre kompletnú implementáciu issue číslo 39 v mobilnej aplikácii „Obrazovka pre prihlasovanie android“

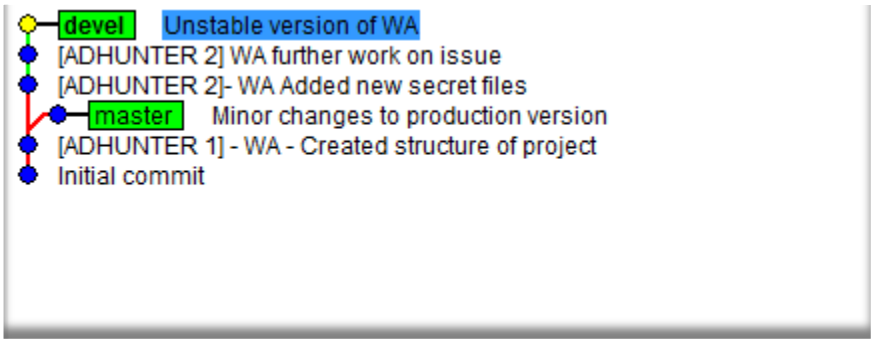
[ADHUNTER-39] -MA- Implementovaná obrazovka pre prihlásenie

Metodika vetvenia vo verziovacom systéme

V nasledujúcej kapitole sa rozoberá problematika vetvenia v tímovom repozitári. Navrhnutá metodika minimalizuje nutnosť riešenia tzv. merge konfliktov ktoré nastávajú pri práci dvoch alebo viacerých prispievateľov na rovnakom súbore. Takisto zvyšuje prehľadnosť pri používaní rôznych GUI nástrojov v Gite napríklad gitk.

ZÁKLADNÝ STAV

Základným stavom ktorý v repozitári je vždy prítomný je prítomnosť dvoch vetiev. Menovite sú to vetvy master a devel. Master ako hlavná vetva celého repozitáru slúži na uchovávanie stabilných verzii prípadne kľúčových tzv milestone bodov aplikácie. Tieto body môžu byť napríklad vydania verzii mobilnej či webovej aplikácie prípadne implementovanie kľúčových čŕt systému. Od určitého momentu sa tu uchováva stav aplikácii ktorá je v súčasnosti v produkcii. Vetva devel slúži na vývoj a prototypovanie sú z nej odvodzované vetvy každého z prispievateľov. Vetvy devel a master sú mergované len po dohode celého tímu a vykonáva ich člen tímu poverený správou verziovacieho systému. Príklad základného stavu je viditeľný na obr. 1. Obrázok je vytvorený pomocou integrovaného nástroja *gitk*.



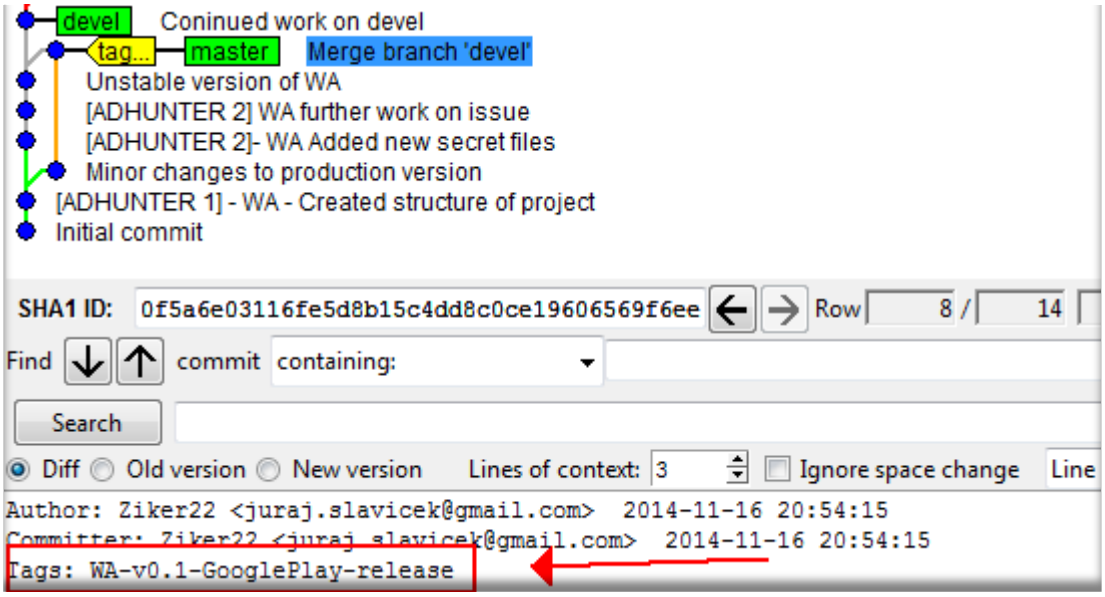
Obr. 1

MERGOVANIE VETIEV DEVEL A MASTER

Ako bolo uvedené vyššie v určitých situáciách je nutné dve základné vetvy spojiť. Popri vyriešení takzvaných merge konfliktov sa k spojenej verzii pridáva **značka(tag**) aby bolo možné hneď a jednoznačne určiť prečo sa tento merge udial a neskôr pri prípadnom znestabilizovaní budúcej verzie. Názov tagu má nasledujúcu štruktúru.

- Skratka ktorej časti (prípadne oboch) sa značka týka v našom prípade WA alebo MA pre webovú resp. mobilnú aplikáciu
- Súčasná verzia označenej aplikácie (v0.1/v3.0 ...)
- Stručný opis stavu aplikácie (Prvá stabilná verzia, Možná platba kartou, Opravená stabilita na platforme Windows)

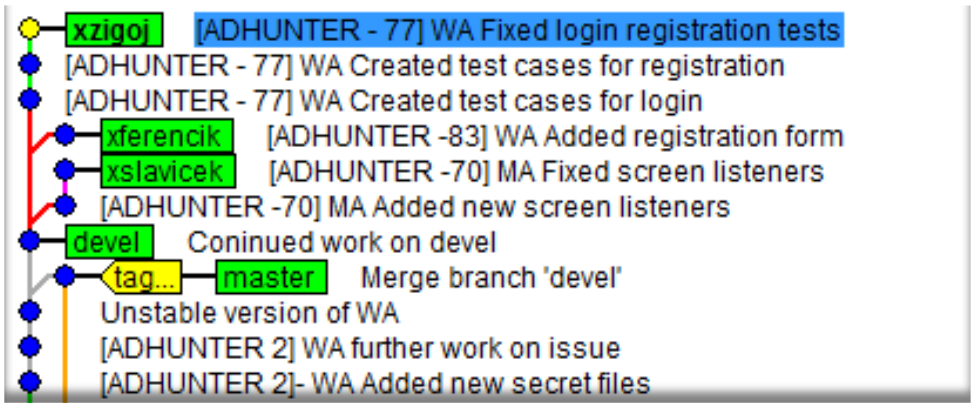
Príklad dobre napísaného tagu a stav repozitára po mergovaní zobrazuje obr. 2.. Plné meno tagu sa zobrazuje len textovo pod týmto oknom.



Obr 2.

VYTVÁRANIE ODVODENÝCH VETIEV Z VETVY DEVEL

Vo vývojových verziách je nutné pri začiatku každej novej úlohy z bug tracking systému JIRA prípadne novej funkcionality v systéme založiť novú vetvu vo verziovaciom systéme. Pomenovanie tejto vetvy sa vytvára podľa osoby ktorá na nej pracuje. Je určené že vetvy sa pomenúvajú podľa prihlasovacích mien do akademického informačného systému. Keďže tieto sú unikátne spolu s pomenovaniami commitov je presne dohľadateľné na akej funkcionalite či úlohe ten ktorý člen tímu práve pracuje. Na obr 3. Sú zobrazení 3 prispievatelia na troch rôznych úlohách.



Obr. 3

MERGOVANIE VETIEV ODVODENÝCH Z VETVY DEVEL NASPÄŤ DO VETVY DEVEL

Pri spájaní týchto druhov vetiev si definujeme pojem splnená úloha. Len vtedy pokiaľ je úloha splnená môžeme pristúpiť k mergovaniu k nej priradenej vetvy a vetvy devel. Úloha je splnená vtedy a len vtedy pokiaľ sú splnené nasledujúce kritéria.

- Úloha je implementovaná a funkčná
- K úlohe sú napísané testy
- Zdrojové testy jadra aj testov prešli code review

Pokiaľ sú vyššie uvedené podmienky splnené vetvy merguje **originálny vlastník vetvy podľa ktorého je vetva nazvaná.**

METODIKA TVORBY DOKUMENTÁCIE

Úvod

Dokument slúži ako metodická príručka pre tvorbu dokumentácie k tímovému projektu pomocou nástrojov Google Documents a MS Word. Opisuje postupy, štruktúru, štábnu kultúru a spôsob ukladania jednotlivých dokumentov.

Táto metodika je určená všetkým členom tímu ako pomôcka pri tvorbe zápisníc z tímových stretnutí, tvorbu dokumentov z retrospektívy šprintov, tvorbu dokumentov, ktoré sú výstupmi z rôznych úloh. Zahŕňa taktiež aj dokumentovanie na wiki tímu, kam sa radia všetky technické dokumenty, inštalačné príručky, návody pre spustenie a integráciu vývojových prostredí.

SKRATKY

Skratka	Význam
MS	Microsoft

POJMY

Pojem	Význam
gDoc	Google dokument
gDrive	Google disk
Wiki	Mediawiki nášho tímu

SÚVISIACE METODIKY

Dokument sa odkazuje na súvisiacu metodiku s názvom Evidencia úloh v systéme Jira, ktorá opisuje ako zapisovať úlohy do systému Jira.

Všeobecné pravidlá

Všetky dokumenty sa píšu v Slovenskom jazyku s použitím diakritiky. Výnimkou môžu byť inštalačné príručky na wiki, kde nie je nutné použiť diakritiku.

Šablóny dokumentov pre zápisnicu zo stretnutí a dokument z retrospektívy šprintu je možné stiahnuť z gDrive, kam má prístup každý člen nášho tímu. Šablóny sa nachádzajú v priečinku s názvom „Zapisy zo stretnuti“.

Štruktúra dokumentov

Táto časť opisuje štruktúru vytváraných dokumentov:

- Zápisnica zo stretnutí
- Retrospektíva
- Dokumentácia na Wiki

ZÁPISNICA ZO STRETNUTÍ

Dokumenty pre zápisnice zo stretnutí majú nasledovnú štruktúru:

Hlavička

1. Nadpis dokumentu má tvar *„Zápisnica zo stretnutia číslo X“*, kde X sa nahradí poradovým číslom aktuálneho stretnutia
2. Téma stretnutia
3. Dátum sa zapisuje vo formáte DD.MM.RRRR
4. Miesto konania stretnutia
5. Čas, kedy stretnutie prebiehalo vo formáte HH:MM – HH:MM
6. Všetci účastníci stretnutia vypísaní pod sebou
7. Vedúci stretnutia
8. Zapisovateľ

Príklad:

Zápisnica zo stretnutia číslo 5

Priebeh druhého šprintu, low fidelity návrhy

Dátum:	23.10.2014
Miesto:	Jobsovo laboratórium
Čas:	8:00 - 11:00
Účastníci:	Jana Egriová
	Alexander Ferenčík
	Richard Filipčík
Vedúci stretnutia:	Richard Filipčík
Zapisovateľ:	Alexander Ferenčík

Priebeh stretnutia

Priebeh stretnutia sa oddeľuje od zvyšku dokumentu rovnomenným nadpisom. Pod nadpis sa zapisuje priebeh diskusie a ľubovoľné ďalšie poznámky týkajúce sa stretnutia formou voľného textu. Zápis priebehu stretnutia je popísaný do takej miery, aby dokumentoval, čo sa dialo na stretnutí aj pre členov, ktorí sa stretnutia nezúčastnili.

Vyplývajúce úlohy

Do tejto časti dokumentu sa zapisujú úlohy vo forme tabuľky z aktuálne prebiehajúceho šprintu. Tabuľka úloh má nasledovnú štruktúru:

1. Číslo úlohy
2. Úloha – jej samo opisný názov, prípadne krátky popis
3. Odhad času – uvádza sa odhadovaný čas potrebný na splnenie úlohy. Údaj je uvedený v hodinách.
4. Zodpovedný – krstné meno člena tímu, ktorý je zodpovedný za splnenie úlohy

Príklad:

	Úloha	**Odhad času [h]**	**Zodpovedný**
5	Pridanie bodu do mapy (drag&drop)	4	Tomáš

Druhou alternatívou zdokumentovania vyplývajúcich úloh je export úloh v šprinte zo systému Jira, pričom zadávanie úloh do systému sa riadi metodikou Evidencia úloh v systéme Jira.

RETROSPEKTÍVA

Dokumenty zo zápisu retrospektívy šprintu majú nasledovnú štruktúru, pričom každá časť je oddelená príslušným nadpisom. Nadpis je rovnomenný s názvom časti.

Hlavička

1. Nadpis dokumentu „Retrospektíva pre X. šprint“, kde X je číslo šprintu
2. Trvanie šprintu v tvare DD.MM.RRRR – DD.MM.RRRR
3. Vedúci tímu
4. Účastníci – členovia tímu, ktorí sa zúčastnili retrospektívy

Príklad:

Retrospektíva pre 2. šprint

Trvanie šprintu: 26.10.2014 – 6.11.2014

Tím: 14 - Včeličky

Vedúci tímu: Ing. Jakub Šimko, PhD.

Účastníci: Jana Egriová

Alexander Ferenčík

Richard Filipčík

Stav vyriešenia úloh

Stav vyriešenia úloh sa dokumentuje formou tabuľky, ktorá má nasledovné stĺpce:

1. User story – príbeh, ku ktorému úloha prislúcha
2. Úloha – samo opisný názov úlohy, prípadne jej krátky popis, ktorý nemá viac ako 50 slov.
3. Stav – aktuálny stav úlohy, spravidla „akceptovaná“ v prípade, že úloha bola dostatočne splnená a „presunutá“ v prípade, že sa vyžaduje ďalšia práca na úlohe v priebehu nasledujúceho šprintu.
4. Poznámky – krátka poznámka k stavu úlohy

Príklad:

User story	**Úloha**	**Stav**	**Poznámky**
Človek v teréne odfotí obrázok	Offline režim pre android app	Presunutá	Vyššiu prioritu získali iné úlohy – úlohy pre integráciu

Hodnotenie šprintu

Kladne a záporne hodnotené javy sa zapisujú formou tabuliek. Prvá tabuľka pre kladne hodnotené javy má dva stĺpce: *pozitívne hodnotené javy šprintu* a *dôvody ich pozitívneho hodnotenia*. Druhá tabuľka obsahuje *stĺpec negatívne hodnotené javy šprintu* a *kroky, ktoré môžu pomôcť javom predísť*.

Opisy v tabuľkách pre stĺpec *pozitívne / negatívne hodnotené javy šprintu* nepresahujú viac, ako 15 slov. Pre stĺpec *dôvody ich pozitívneho hodnotenia* a *kroky, ktoré môžu javom predísť* sa používa maximálne 50 slov.

Príklad:

Pozitívne hodnotené javy šprintu	**Dôvody ich pozitívneho hodnotenia**
Dobre fungujúca integrácia	Zadelenie úloh, ktoré sa týkali integrácie, vytvorenie novej role manažér integrácie pri poslednej retrospektíve

Nové úlohy vyplývajúce z retrospektívy

Úlohy, ktoré vyplynuli z retrospektívy sa zapisujú formou odrážok ako voľný výstižný text, ktorý úlohu opisuje v dostatočnej miere.

Prílohy

Každá príloha sa uvádza na novú stranu s príslušným nadpisom v tvare „Príloha #: Názov“, kde # je veľké písmeno abecedy A-Z.

WIKI

Technické dokumenty sa na wiki rozdeľujú do nasledovnej štruktúry, pričom pre každý dokument sa vytvára nová wiki stránka a jej odkaz sa správne umiestni do štruktúry. V prípade potreby je možné štruktúru vhodne rozšíriť.

- Inštalačné príručky
    - Back-end
    - Front-end
        - Android
        - Webová aplikácia
- Stuff
- Trash

Štábna kultúra

DOKUMENTY

Pre dokumenty zo zápismi zo stretnutí a retrospektívu platí nasledovná štábna kultúra:

Názov

- Font: Trebuchet MS
- Veľkosť: 21

Téma

- Font: Trebuchet MS
- Veľkosť: 13
- Farba: šedá
- Šikmé písmo

Napisy

- Font: Trebuchet MS
- Veľkosť: 13
- Tučné písmo

Text

- Font: Arial
- Veľkosť: 11

Tabuľky

- Font: Arial
- Veľkosť: 11 pre hlavičku, 10 pre text
- Hlavička tučným písmom

Tabuľky a obrázky

- Číslovanie: „**Obrázok č. X:** Stručný názov“, kde X vyjadruje poradové číslo obrázka alebo tabuľky v dokumente
- Font: Arial
- Veľkosť písma: 9
- Zarovnanie: na stred

WIKI

Pre dokumenty na wiki sa používa formátovanie pomocou default wiki syntaxe, ktorá je prístupná na stránke https://www.mediawiki.org/wiki/Help:Formatting/sk.

Úložisko dokumentov

Všetky dokumenty sa ukladajú na gDrive do zdieľaného priečinka s názvom „Timovy projekt“, ku ktorému majú prístup všetci členovia tímu. Zápisnice a dokumenty z retrospektívy šprintu sa ukladajú do priečinka „Zapisy zo stretnuti“ po vytvorení nového podpriečinka s názvom *„Zapisnica X – DD.MM.RRRR“,* kde X je poradové číslo zápisnice.

Úložisko pre Wiki dokumenty sa nachádza na webovom sídle nášho webu, konkrétne na adrese http://team14-14.ucebne.fiit.stuba.sk/wiki/, ktorá je prístupná len po prihlásení. Každý člen tímu sa prihlasuje pomocou vlastného prihlasovacieho mena a hesla.

Formát

Tvorba dokumentov, na ktorých sa podieľa viac členov tímu prebieha spravidla na gDrive vo formáte gDoc. Po vyhotovení sa dokument stiahne vo formáte .docx, podľa potreby upraví v programe MS Word a po uložení sa uploadne späť na zdieľaný gDrive vo formáte .docx.

Zápisnice zo stretnutí a zápisy z retrospektív šprintov sa uverejňujú na webovom sídle tímu (http://team14-14.ucebne.fiit.stuba.sk/dokumenty/) vo formáte .pdf v takej forme, v akej sú uverejnené na gDrive.