Adaptívny proxy server

Riadenie

Tím 13 – Old School Brothers

Tímový projekt 2010/2011
Fakulta informatiky a informačných technológií
Slovenská technická univerzita

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Tím 13 – Old School Brothers
email: team013@googlegroups.com

1 Obsah

2 Úvod

Tento dokument slúži ako dokumentácia k predmetu tímový projekt, vedeného na Fakulte informatiky a informačných technológií STU. Opisuje spôsob riadenia tímu na tomto predmete.

V úvode sa nachádzajú dokumenty ponúk k projektom, ktoré sme vytvárali na začiatku semestra, ešte pred pridelením jednotlivých tém. Ponuka na nám pridelenú tému, adaptívny proxy server, sa tu nenachádza, keďže na túto tému sme ponuku nepripravovali.

V dokumente takisto uvádzame dlhodobý plán na zimný semester ako aj krátkodobé, dvojtýždňové plány na jednotlivé šprinty. Taktiež popisujeme priebeh vzniku krátkodobého plánu.

Príspevky jednotlivých členov tímu k tomuto dokumentu, ako aj krátkodobé a dlhodobé úlohy členov tímu sú sumarizované v spoločnej kapitole.

Nakoniec dokument obsahuje štábnu kultúru a metodiku používania nástroja na správu verzií, konfigurácií a zmien.

V prílohách sa nachádzajú záznamy z jednotlivých stretnutí a preberacie protokoly.

3 Ponuky

Zo zoznamu tém, ktoré boli ponúknuté v tomto akademickom roku sme sa rozhodli vypracovať ponuky k týmto témam:

- Portál pre časopis
- Správa študentských projektov na fakulte
- Simulated Car Racing Competition 2011

Tieto témy boli vybrané najmä preto, že väčšina členov nášho tímu už má nejaké skúsenosti s tvorbou webových aplikácií, čo nám dávalo istú konkurenčnú výhodu.

Nakoniec sme však nevyhrali súťaž ani k jednej z nami preferovaných tém a boli sme nútení sa pomerne pružne rozhodnúť pre niektorú zo zostávajúcich tém. Téma adaptívny proxy server sa ukázala ako vhodná alternatíva vzhľadom na naše preferencie a rozloženie záujmu konkurenčných tímov.

Súčasťou tvorby ponuky bolo aj vypracovanie profilov jednotlivých členov tímu, ktoré tu tiež uvádzame.

Kto sme

Bc. Ján Hudek

Bakalárske štúdium absolvoval na FIIT STU, kde sa venoval tvorbe portálu na kolaboratívne vytváranie konceptových máp. Znalosti nadobudnuté v období štúdia mal možnosť zhodnotiť počas ročnej praxe v oblasti tvorby webu. Má bohaté skúsenosti s prácou vo frameworkoch Magento a Typo3. Veľmi dobre ovláda technológie PHP, MySQL, JavaScript, HTML, CSS a Silverlight. Zároveň má skúsenosti s jazykmi Java, C, C#. Medzi jeho najcennejšie vlastnosti patrí komunikatívnosť a schopnosť pracovať v tíme.

Bc. Ivan Pleško

Absolvoval bakalárske štúdium v odbore informatika na FIIT. Vďaka ročnej praxi má bohaté skúsenosti s prácou v tíme a v oblasti moderného webu. Jeho doménou je jazyk PHP, JavaScript (+ jQuery), HTML + CSS, databáza MySQL. Ovláda technológiu AJAX, Flash, pracuje s grafikou v nástroji Adobe Photoshop. Ďalej ovláda jazyky Java, C, C++.

Bc. Michal Valluš

Počas bakalárskeho štúdia na FIIT STU sa zameral na objektovú paradigmu (Java) a na paralelné programovanie (CUDA) v rámci bakalárskeho projektu. Uplynulý rok pracoval na aplikáciách pre Orange Slovensko, kde získal cenné programátorské a komunikačné schopnosti. K jeho všestrannosti prispievajú aj výborné matematické myslenie a pôsobenie v oblasti hudby a fotografie.

Bc. Michal Mrázik

Absolvoval bakalárske štúdium na FIIT v odbore Informatika. V bakalárskej práci sa venoval porovnávaniu významov textov na základe ontologických konceptov. Najviac sa zaoberá objektovo orientovaným programovaním v jazyku JAVA. Ďalej má skúsenosti s jazykom C, C++, SQL. Vie používať nastroj LATEX. Zaujíma ho tvorba zložitejších algoritmov a riešenie matematických problémov. Medzi jeho dôležité vlastnosti patrí súťaživosť a komunikatívnosť.

Bc. Juraj Spusta

V oblasti IT sa zaujíma skôr o tvorbu algoritmov a riešenie matematických problémov. Pri vývoji aplikácií nadobudol skúsenosti hlavne s jazykom Java a preferuje objektové programovanie. Z hľadiska webových aplikácií ovláda základy HTML a CSS, ale viac sa venuje tvorbe RIA (Rich Internet Aplication), kde používa technológiu Adobe Flex spolu s jazykom ActionScript.

Bc. Pavol Sokol

Počas štúdia na FIIT pracoval v niekoľkých firmách, kde sa zaoberal vývojom aplikácií a informačných systémov v jazykoch PHP, Java, C++. V tíme pôsobí ako jednotiaci prvok. V programátorskej praxi obľubuje implementačne náročné problémy a ťažko riešiteľné otázky.

Portál pre časopis

Naša motivácia

Pri výbere témy projektu sme zohľadňovali dve hlavné požiadavky. Prvou bolo, aby naše dielo našlo reálne využitie v praxi a slúžilo svojmu účelu aj po absolvovaní predmetu tímový projekt. Druhou požiadavkou bolo využitie našich doterajších znalostí a získanie nových poznatkov a skúseností v oblasti, ktorej by sme sa chceli v budúcnosti venovať. Tieto požiadavky veľmi dobre spĺňala práve táto téma.

Snahou nášho tímu bude teda vytvoriť portál, ktorý by prezentoval články časopisu ACM a zároveň by uľahčoval procesy schvaľovania a posudzovania článkov. Portál by prehľadnou formou prezentoval slovenskú pobočku a prácu komunity prispievajúcej do časopisu. Obsahoval by teda aj knižnicu všetkých článkov napísaných touto komunitou.

Keďže viacerí členovia majú bohaté praktické skúsenosti v práci na väčších webových projektoch zahŕňajúcich firemné portály, internetové obchody a fakturačné systémy, veríme, že sme schopní dotiahnuť tento projekt do úspešného konca k spokojnosti zadávateľa ale aj samotnej ACM komunity.

Naše riešenie

V tejto stati zhrnieme hlavné požiadavky na portál, opíšeme návrh riešenia a navrhneme aplikovateľné technológie.

Z hľadiska funkcionality by mal portál spĺňať tieto požiadavky:

- prezentácia slovenskej pobočky organizácie ACM s možnosťou editácie obsahu umiestneného na portály (vytvorenie jednoduchého a prehľadného CMS systému na správu portálu),
- umiestňovanie článkov na portál samotnými autormi bez nutnosti registrácie. Zároveň by portál umožňoval registráciu, ktorá by autorovi ponúkala benefity ako sledovanie histórie jeho uverejnených článkov alebo sledovanie požiadaviek na úpravu týkajúcich sa jedného konkrétneho článku,
- podpora procesov schvaľovania a posudzovania článkov administrátorom s možnosťou komunikácie s autorom. V prípade požiadavky na úpravu článku na autora, ktorý nemá vytvorené konto na portály, by sa generovala do e-mailu linka, kde by mohol priamo vložiť upravený článok. Uľahčila by sa tým práca autora ale aj samotného administrátora, ktorý by nebol nútený ručne párovať články k autorom,
- fulltextové vyhľadávanie v článkoch s možnosťou rozšíreného vyhľadávania podľa autora alebo dátumu uverejnenia (pre zvýšenie odozvy by mal portál implementovanú cache, kde by sa ukladali výsledky predošlých vyhľadávaní), umiestnené priamo na portály.

Portál by mal zároveň spĺňať aj ďalšie nefunkcionálne požiadavky:

- portál bude spĺňať tvrdé kritéria na bezpečnosť (ochrana pred útokmi typu SQL Injection, XSS, Remote File Inclusion, ...),
- portál by mal byť rovnako prehľadný a použiteľný pre bežného návštevníka, autora ale aj administrátora,
- modulárnosť a štruktúrnosť kódu obsahujúceho dostatočné množstvo komentáru. Pri návrhu a implementácii sa bude počítať s možnosťou neskoršieho rozšírenia, prípadne upravenia portálu.

Po zvážení požiadaviek by sme navrhli implementáciu použitím technológií PHP a MySQL na serverovej strane. Na klientskej strane by portál ponúkal komfort moderných technológií a prístupov ako AJAX, zároveň by sa však držal štandardov definovaných W3C.

Naším primárnym cieľom je spokojnosť zákazníka a samotných používateľov portálu. Preto sa budeme snažiť čo najbližšie priblížiť jeho predstave, no veríme, že naše skúsenosti a nápady prispejú ku kvalite produktu.

Správa študentských projektov na fakulte

Naša motivácia

Prvým z motívov voľby tejto témy je, že členovia nášho tímu majú bohaté znalosti v oblasti webu, čo nás robí vhodnými kandidátmi na vypracovanie tejto témy. Ďalším, nemenej dôležitým motívom je myšlienka, že vzniknutá aplikácia bude reálne nasadená a používaná. Jedná o veľmi dôležitú aplikáciu,

ktorá bude slúžiť ako študentom, tak aj pedagógom. Poskytne podporu pre mnohé procesy a práve preto musí byť vypracovaná s ohľadom na kvalitu, robustnosť, prehľadnosť rozhrania a bezpečnosť. Veríme, že vďaka naším skúsenostiam v oblasti webových technológií dokážeme vytvoriť aplikáciu, ktorá tieto podmienky splní.

Existujúci systém Yonban je funkčný, dobre vypracovaný a poskytne kvalitnú inšpiráciu pri tvorbe nového systému. My by sme chceli priniesť moderné prístupy. Použiť asynchrónne požiadavky na server (napr. pomocou AJAX), čo by používanie aplikácie výrazne zrýchlilo a sprehľadnilo. Tiež budeme dbať na grafické rozhranie, ktoré by malo byť moderné, no najmä prívetivé a prehľadné pre používateľa.

Táto téma je pre nás veľmi zaujímavá, predstavuje výzvu, s ktorou by sme sa radi popasovali a priniesli tak osoh pre celú fakultu.

Naše riešenie

Prvým krokom riešenia nášho tímu bude podrobenie systému Yonban dôkladnej analýze. Jej úlohou bude získať dostatočné množstvo informácií, ktoré nám pomôžu pri tvorbe nového systému. Budeme vychádzať z jednotlivých prípadov použitia systému, tak, aby spracovanie v systéme pokrývalo všetky procesy pri správe študentských projektov na fakulte. Počas analýzy sa taktiež budeme snažiť odhaliť, aké funkcionality chýbajú v súčasnom systéme, prípadné upravenie a doplnenie budeme konzultovať s kompetentnými osobami a budúcimi používateľmi systému.

Podrobná analýza nám poskytne dobrý základ k vypracovaniu kvalitného návrhu. Našim cieľom bude čo najvhodnejšie použiť známe návrhové vzory, aby sme zabezpečili vysokú modularitu systému a tým umožnili neskoršie pridávanie a modifikovanie funkcionality.

V závislosti od analýzy a návrhu vyberieme najvhodnejšie technické prostriedky na realizáciu projektu. V tíme máme niekoľko odborníkov na tvorbu webov v PHP s niekoľkoročnými skúsenosťami. V prípade potreby vieme použiť niektorý z existujúcich PHP frameworkov. Samozrejmosťou je využitie technológie Ajax na sprehľadnenie a zefektívnenie aplikácie. Pri implementácii budeme dbať na dodržanie webových štandardov W3C, najmä validitu samotného kódu.

Novovytváraný systém bude taktiež potrebné prepojiť s už fungujúcim Akademickým informačným systémom fakulty a to minimálne použitím spoločného repozitára loginov a hesiel.

Najväčšia zmena oproti súčasnému fungovanie systému bude pravdepodobne možnosť využitia systému ako historickej bázy znalostí z rôznych oblastí. Preto chceme zabezpečiť kvalitné triedenie informácií z možnosťou pokročilého vyhľadávania v prácach. Výsledky vyhľadávania bude možné zoradiť podľa rôznych atribútov, v neposlednom rade aj podľa relevancie, získanej na základe rozhodovania ostatných používateľov. Taktiež by sme chceli implementovať možnosť fulltextového vyhľadávania s prípadnou optimalizáciou na najčastejšie vyhľadávané výrazy.

Vzhľadom na to, že sa jedná o aplikáciu ktorá spracováva dôležité informácie o rádovo stovkách študentov, je veľmi dôležité dbať aj na bezpečnosť. Aplikácia bude musieť byť odolná voči základným ale

aj pokročilým útokom, napríklad XSS, SQL injection, remote file inclusion a pod. Všetky tieto vlastnosti vieme zabezpečiť, opierajúc sa o skúsenosti našich vývojárov s tvorbou podobných systémov využívaných v praxi.

Simulated Car Racing Competition 2011

Naša motivácia

V dnešnej dobe sa roboty nachádzajú viac či menej vo všetkých oblastiach ľudskej činnosti. Niektoré dosahujú väčšiu mieru inteligencie, iné robia len pár naprogramovaných krokov.

Veľmi dobrým nápadom je automatické ovládanie dopravných prostriedkov. Je veľmi zaujímavé si predstaviť, že autá by nás vozili z miesta na miesto bez toho, aby sme čo i len trošku prispeli k riadeniu. Čas potrebný na prepravu, by sme mohli využiť omnoho užitočnejšie. V reálnej premávke je tento problém zatiaľ neriešiteľný z dôvodu veľmi rozmanitého prostredia.

Určité zjednodušenie prináša pretekárska dráha. Každá dráha obsahuje samotnú trať s obrubníkmi a zónami mimo trate ktoré sú prázdne pre prípad, že by vozidlo nezvládlo riadenie a skončilo mimo trate. Ďalším nemenej dôležitým bodom je, že na pretekárskej dráhe jazdia autá vždy v jednom smere. Toto všetko prispieva k zjednodušeniu riadenia vozidla na trati na rozdiel od premávky.

Simulovanie pretekárskych áut má veľký význam z pohľadu umelej inteligencie. Ak dokážeme nasimulovať robota, ktorý bezchybne riadi virtuálne auto, potom nám chýba už len malý krôčik k tomu, aby ten istý robot mohol riadiť auto za niekoľko miliónov v reálnych pretekoch. Nás by veľmi zaujímalo ako robot obstojí v boji porovnaní so živou bytosťou na pretekárskej dráhe a preto by sme chceli aspoň malou časťou prispieť ku vytvoreniu obstojného súpera pre človeka.

Naše riešenie

Prvým a základným krokom pre nás by bolo oboznámiť sa s TORCS a spôsobmi ako sa v tomto prostredí dajú riadiť modely. Zistili by sme aké ovládacie prvky sú k dispozícii a akými senzormi auto disponuje.

V ďalšom kroku by bolo potrebné zistiť ako sa auto správa v prostredí simulátora a na základe toho určiť rýchlosti prejazdu zákrutami, nutnosť brzdenia v daných okamihoch, možnosť predbiehania a iné. Všetky tieto vlastnosti sa musia premietnuť do modelu umelej inteligencie, ktorá rozhodne čo je v danej situácii správna urobiť (pridať plyn, zabrzdiť, otočiť volantom ....). Spôsobov akými by sa to dalo realizovať je mnoho. Za pokus by stálo napríklad implementovanie fuzzy logiky.

4 Plány

V nasledujúcej kapitole uvádzame plán jednotlivých šprintov do 12. týždňa zimného semestra.

4. – 5. týždeň	šprint 1 *Eniac*	- oboznámenie sa s projektom - naštudovanie technológie evercookies - odstránenie User Agenta + zavedenie evercookies
6. – 7. týždeň	šprint 2 *MITS Altair 8800*	- vytvorenie nástroja pre deploy proxy servera - presunutie funkcionality z Proxy-Web na Proxy-Server
8. – 9. týždeň	šprint 3	- špecifikácia požiadaviek pre vytvorenie komunikačných fór na stránkach - vytvorenie fór pre navštevované stránky
10. – 11. týždeň	šprint 4	- vytvorenie API pre tvorcov pluginov so zameraním na search engines
12. týždeň	šprint 5	- testovanie - code review - implementovanie menších zadaní

Vytváranie plánov v našom tíme prebieha v niekoľkých na seba nadväzujúcich etapách. Každá etapa má vstupy a výstupy na základe, ktorých sa vždy priblížime k výslednému plánu na daný šprint. Postup pri tvorení plánov je nasledujúci:

1. Prvým krokom je vytýčenie jednotlivých požiadaviek (requirement) na zmenu resp. pridanie novej funkcionality do systému. Tieto sú vybraté buď z product backlog-u (už existujúce požiadavky), alebo sú špecifikované na začiatku nového šprintu. Od výberu týchto požiadaviek potom závisí celá tvorba plánu pre daný šprint.
2. Po definovaní požiadaviek je potrebné špecifikovať, čo má byť výsledkom jednotlivých požiadaviek. Zároveň sú požiadavky rozdelené na jednotlivé príbehy (user stories), ktoré sú vyjadrením logických celkov danej požiadavky. Spolu s vytvorením príbehov sú definované aj väzby a nadväznosti príbehov alebo samotných požiadaviek. Každý príbeh je potom ohodnotený metódou “scrum poker“, čím sa určí náročnosť každého príbehu. Príbehy sú potom pridelené členom tímu, podľa ich záujmu alebo znalostí v danej problematike. Každý vlastník príbehu je potom zodpovedný za vytvorenie úloh (task) a ich priradenie členovi, ktorý bude danú úlohu vypracovávať.

3. Zatiaľ čo prvé dve etapy sa uskutočňujú na tímovom stretnutí a všetci členovia sa do nich aktívne zapájajú, nasledujúca etapa je už zastrešovaná manažérom plánovania. Prvým krokom je výzva plánovača pre všetkých členov tímu, aby mu zaslali čo najskôr svoju časovú dostupnosť na najbližší šprint. Časová dostupnosť je vlastne zoznam dní v šprinte a ku každému dňu je pridelená hodnota v hodinách. Táto hodnota je predpokladom daného člena tímu (istá forma self-estimate), koľko sa bude venovať úlohám na projekte.
4. Poslednou etapou je samotné vytvorenie plánu. Existujúci zoznám úloh je usporiadaný chronologicky podľa väzieb medzi úlohami a následne sú zoskupené podľa členov tímu. Potom je vytvorený plán v ktorom každý člen vidí, koľko hodín a ktoré dni by sa mal venovať svojím úlohám. Prípadné zmeny v dostupnosti je nutné hlásiť manažérovi plánovania, ale aj členom tímu, ktorí spolupracujú na danom príbehu.

V nasledujúcej časti sú zhrnuté vytvorené plány pre jednotlivé šprinty.

Šprint #3 : 3.11.2010 - 16.11.2010

Plán bol vypracovaný 5.11.2010

Názov úlohy	*Zodpovedný*	*Čas, v ktorom sa bude úloha riešiť*
Komunikácia - návrh	Hudek	5.11
Databáza	Hudek	5.11 - 6.11.
Grafický návrh	Pleško	6.11 - 7.11.
Implementácia - klient	Pleško	12.11 - 13.11.
Implementácia - server	Hudek	7.11. - 12.11.
Service na parsovanie HTML do DOM	Sokol	5.11. - 6.11.
SearchEngineService a získavanie výsledkov vyhľadávania	Mrázik	6.11. - 7.11.
SearchEngineService a menenie vysledkov vyhladavania	Valluš	8.11. - 13.11.
Update Proxy-Web do Rails 3	Spusta	5.11. - 7.11.
Konfigurácia loadbalancera pre proxy	Sokol	13.11 - 15.11.
Evercookie pre všetky domény	Spusta	8.11. - 12.11.

Šprint #4 : 17.11.2010 - 30.11.2010

Plán bol vypracovaný 19.11.2010

Názov úlohy	*Zodpovedný*	*Čas, v ktorom sa bude úloha riešiť*
Klient - grafický návrh	Pleško	20.11 - 21.11.
Implementácia - klient - prezeranie	Pleško	21.11 - 24.11.
Implementácia - klient - pridávanie	Pleško	24.11.
Implementácia - klient - mazanie	Pleško	24.11.
Komunikácia - návrh	Hudek	19.11.
Implementácia - server - prezeranie	Hudek	19.11. - 21.11.
Implementácia - server - pridávanie	Hudek	21.11 - 23.11.
Implementácia - server - mazanie	Hudek	23.11. - 24.11.
Dokumentácia	Hudek	24.11. - 25.11.
Implementácia rozhrania na čítanie	Valluš	20.11.
SearchEngineService a menenie výsledkov vyhľadávania z Yahoo!	Valluš	20.11. - 23.11.
SearchEngineService a menenie výsledkov vyhľadávania z Bing	Valluš	23.11. - 26.11.
SearchEngineService pre získanie výsledkov z Yahoo!	Mrázik	20.11. - 24.11.
SearchEngineService pre získanie výsledkov z Bing	Mrázik	24.11. - 27.11.
Nastavenie XMLoutputtera na spracovateľný výstup	Sokol	19.11.
Vytvorenie nového záznamu v access_logs	Spusta	19.11. - 20.11.
Nový activity logging na základe guid	Spusta	20.11. - 21.11.
Konfigurácia loadbalancera pre proxy	Sokol	24.11 - 27.11.
Presunúť extrakciu kľúčových slov z JKeyExtractor na Metall	Sokol	20.11. - 24.11.

Šprint #5 : 1.12.2010 - 14.12.2010

Plán bol vypracovaný 1.12.2010

Názov úlohy	*Zodpovedný*	*Čas, v ktorom sa bude úloha riešiť*
Oprava logovania - do access logu sa loguje retazec "IP"	Hudek	2.12.
Refactoring, prerobenie messageboard + keywords	Hudek	2.12. - 10.12.
Sprístupniť pre vývojára cestu k adresáru s binárkami a scriptami	Pleško	5.12. - 8.12.
Premenovat after:after_deploy task na after:deploy	Sokol	2.12.
Nájsť a odstrániť volanie, v ktorom rakefile volá sám seba	Valluš	3.12.
Na začiatku offline:schedule premazať crontab	Pleško	2.12.
Nastavenie XMLoutputtera na spracovateľný výstup	Sokol	2.12. - 3.12.
Upraviť migrácie	Hudek	11.12. - 13.12.
Servis na ziskavanie posielanych dat z klienta	Spusta	2.12.
Vytvoriť pole, v ktorom bude poradie, v ktorom sa majú kompilovať bundle	Valluš	5.12. - 10.12.
Kompilácia bundlov podľa poradia	Valluš	11.12.
Nastaviť git submoduly a opraviť rakefile tak, aby to fungovalo	Pleško	3.12. - 4.12.
Testovanie deploy scriptov	Pleško	4.12. - 5.12.
Napísať štábnu kultúru	Sokol	3.12. - 5.12.
Napísať o manažmente verzií	Hudek	10.12. - 12.12.
Napísať celkový pohľad na technické riešenie	Pleško	8.12. - 9.12.
Kompletizácia dokumentácie	Valluš	6.12. - 12.12.

Šprinty letného semestra

V letnom semestri sme explicitné plány na jednotlivé šprinty nerobili. Ako referencia naplánovaných úloh nám slúžil nástroj na podporu plánovania RedMine, ktorý sme začali v letnom semestri používať namieso nástroja Agilo.

5 Úlohy členov tímu

Dlhodobé úlohy členov tímu sú prehľadne zobrazené v tabuľke:

Ján Hudek	- manažér vývoja - manažér verzií
Michal Mrázik	- manažér kvality
Ivan Pleško	- manažér podporných prostriedkov - prezentácia na webe
Pavol Sokol	- tím líder - SCRUM master
Juraj Spusta	- manažér plánovania - technická podpora v TRAC
Michal Valluš	- manažér dokumentácie - manažér rizík

Nasledujúca tabuľka zobrazuje jednotlivé používateľské prípehy (*angl. user story*), tak ako boli pridelené členom tímu. Daný člen je zodpovedný za implementáciu a dokumentáciu pridelenej user-story, neznamená to však, že ju nutne musí implementovať; v prípade potreby môže túto úlohu delegovať na iného člena tímu.

user-story	šprint	zodpovedný člen tímu
Inicializácia pomocou evercookie	Eniac	Michal Valluš
Prenos evercookie	Eniac	Pavol Sokol
Prezeranie logov	Eniac	Pavol Sokol
Nastavenie evercookie do response	Eniac	Ivan Pleško
Logovanie času	Eniac	Juraj Spusta
Prístup do logov	Eniac	Ján Hudek
Odstraňovanie evercookie	Eniac	Michal Mrázik
Servovanie skriptov z proxy	MITS Altair 8800	Juraj Spusta
Spracovanie requestov zjavascriptov v proxy	MITS Altair 8800	Juraj Spusta
Offline tásky v ofline adresári	MITS Altair 8800	Ivan Pleško
Folder pre migrácie	MITS Altair 8800	Michal Mrázik
Adresár pre konfikurácie pluginov	MITS Altair 8800	Ján Hudek
Adresár pre statický obsah	MITS Altair 8800	Ján Hudek
Adresár pre knižnice	MITS Altair 8800	Michal Mrázik
Adresár pre zdrojové kódy	MITS Altair 8800	Michal Valluš

Spustenie tásku po deploymente	MITS Altair 8800	Pavol Sokol
Úprava skriptov pre deployment	MITS Altair 8800	Pavol Sokol

Zmeny v dlhodobých úlohách členov tímu

Po druhom šprinte nastala v tíme výmena na pozíciách SCRUM master a manažér podporných prostriedkov. Ostatné pozície členov tímu zostali nezmenené.

Ivan Pleško	- tím líder - SCRUM master - prezentácia na webe
Pavol Sokol	- manažér podporných prostriedkov

Nasledujúca tabuľka zobrazuje používateľské príbehy z ďalších šprintov semestra a k nim aj zodpovedné osoby.

user-story	šprint	zodpovedný člen tímu
Používateľ nechá odkaz na ľubovoľnej stránke	Fairchild Illiac-IV	Ivan Pleško
Proxy-web v Rails 3	Fairchild Illiac-IV	Juraj Spusta
Evercookie pre všetky domény	Fairchild Illiac-IV	Juraj Spusta
SearchEngineService a získavanie výsledkov vyhľadávania	Fairchild Illiac-IV	Michal Mrázik
SearchEngineService a menenie výsledkov vyhľadávania	Fairchild Illiac-IV	Michal Valluš
Plugin pre vytváranie DOM webovej stránky	Fairchild Illiac-IV	Pavol Sokol
Prezeranie metadát k aktuálnej stránke	Commodore	Ivan Pleško
Pridávanie metadát k aktuálne zobrazenej stránke	Commodore	Ivan Pleško
Mazanie metadát k stránke	Commodore	Ivan Pleško
SearchEngineService pre získanie výsledkov z Yahoo a Bing	Commodore	Michal Mrázik
SearchEngineService pre zmenu výsledkov z Yahoo a Bing	Commodore	Michal Valluš
Proxy používa Metall	Commodore	Pavol Sokol
Nahradenie checksum za request ID	Commodore	Juraj Spusta
Load-balancer pre proxy	Commodore	Pavol Sokol
Refactoring, prerobenie messageboard + keywords	ZX Spectrum+	Ján Hudek
Servis na získavanie posielaných dát z klienta	ZX Spectrum+	Juraj Spusta
Kompilácia bundlov podľa poradia	ZX Spectrum+	Michal Valluš
Dostať deploy do funkčného stavu	ZX Spectrum+	Ivan Pleško

Príspevky členov tímu k časti riadenie na konci zimného semestra

Nasledujúca tabuľka prehľadne zobrazuje príspevky jednotlivých členov tímu k dokumentácii, konkrétne k časti riadenie (čiže k tomuto dokumentu).

Kapitola	Autor
2 Úvod	Michal Valluš
3 Ponuky – Kto sme	Pavol Sokol
3 Ponuky – Portál pre časopis	Ján Hudek
3 Ponuky – Správa študentských projektov na fakulte	Ivan Pleško, Michal Valluš
3 ponuky – Simulated Car Racing Competition 2011	Michal Mrázik
4 Plány	Juraj Spusta
5 Úlohy členov tímu	Michal Valluš
6 Štábna kultúra	Pavol Sokol
7 Manažment verzií, konfigurácií a zmien	Ján Hudek
Celkový pohľad na vytvorený prototyp, zimný semester (technická dokumentácia)	*Ivan Pleško*

Príspevky k častiam v technickej dokumentácii sú uvedené pri každej user-story.

Personálne zmeny v tíme

Na začiatku letnného semestra náš tím opustil Bc. Michal Mrázik.

6 Štábna kultúra

Štábna kultúra popisuje formálne náležitosti a štrukturálnu integritu výstupov, najmä zápisy zo stretnutí, technickú dokumentáciu a zdrojový kód.

Zápisy zo stretnutí

Zápisy zo stretnutí sa vytvárajú v prostredí Microsoft Word v kompatibilnom režime 97-2003. Pre formátovanie zápisov je vytvorená šablóna na dokumentovom serveri tímu. Autorom šablóny je Bc. Michal Valluš.

Technická dokumentácia

Technická dokumentácia sa vytvára v prostredí Microsoft Word v kompatibilnom režime 97-2003. Pre formátovanie je vytvorená šablóna na dokumentovom serveri tímu. Autorom šablóny je Bc. Michal Valluš.

Zdrojové kódy

Balíky

Názvoslovie každého balíka má tvar „sk.fiit.rabbit.adaptiveproxy.plugins[.xxx]“, kde „xxx“ sú nahradené nasledovným názvoslovím:

- Pre všetky deklarácie rozhrania služieb je „xxx“ nahradené „servicedefinitions“ .
- Pre všetky definície tried je „xxx“ nahradené „services.[yyy]“, kde „yyy“ je jednoslovný názov výstižne popisujúci obsiahnutú funkcionalitu v balíku.

V balíku musí byť zapuzdrená každá trieda, rozhranie, alebo enumerický typ. Názov balíka do ktorého trieda, rozhranie, alebo enumerický typ patrí sa nachádza v súbore vždy na prvom riadku. Povolené sú iba znaky z ASCII množiny [a-z], teda iba malé znaky anglickej abecedy, a bodka pre oddelenie vnorených štruktúr.

Import

Importy sa nachádzajú iba v súbore v ktorom sú vyžadované, alebo vhodné. Import odkazuje na konkrétnu triedu. Znak hviezdička „*“ a import celého balíka nie je povolený. Každý import sa nachádza na samostatnom riadku. Deklarácie všetkých importov sa v súbore nachádzajú za deklaráciou balíka od ktorého sú oddelené jedným prázdnym riadkom.

Rozhrania

Rozhranie je deklarované kľúčovým slovom „interface“ za ktorým nasleduje názov rozhrania a prípadné dedenia. Názov je vytvorený spojením najviac troch slov, ktoré popisujú funkciu rozhrania v anglickom jazyku a pridaním slova „Service“ na koniec názvu. Povolené sú iba znaky ASCII [a-z] a [A-Z], teda iba veľké a malé písmená anglickej abecedy. Každé slovo začína veľkým písmenom.

Triedy

Trieda je deklarovaná kľúčovým slovom „class“ za ktorým nasleduje názov triedy a prípadné dedenia. Názov je vytvorený spojením najviac troch slov, ktoré popisujú funkciu triedy v anglickom jazyku a pridaním žiadneho, alebo jedného slovného spojenia podľa pravidiel:

- Ak trieda priamo, alebo nepriamo cez dedenie implementuje rozhrania RequestProcessingPlugin, ResponseProcessingPlugin, názov takejto triedy bude ukončený slovným spojením „ProcessingPlugin“.
- Ak trieda priamo, alebo nepriamo cez dedenie implementuje rozhrania RequestServiceModul, ResponseServiceModul, názov takejto triedy bude ukončený slovným spojením „ServiceModul“.
- Ak trieda implementuje rozhranie RequestServiceProvider, alebo ResponseServiceProvider, názov takejto triedy bude ukončený slovným spojením „ServiceProvider“.
- Ak trieda nezodpovedá žiadnemu z vyššie spomínaných pravidiel, jej názov nebude ukončený žiadnym z týchto spojení.

Funkcie a premenné

Názvy funkcií a premenných sú tvorené podľa uváženia programátora, musia však dodržať nasledujúce:

- Názov funkcie sa začína veľkým, alebo malým písmenom.
- Názov premennej sa začína vždy malým písmenom
- Názov funkcie je tvorený tak, aby čo najvýstižnejšie popísal funkčnosť funkcie v anglickom jazyku
- Názov premennej je tvorený tak, aby čo najvýstižnejšie popísal úlohu premennej v anglickom jazyku

Umiestňovanie zátvoriek a odsadzovanie blokov

Otváracie množinové zátvorky sa nikdy nachádzajú na samostatnom riadku. Sú vždy späté s deklaráciou, alebo definíciou rozhrania, triedy, enumerickým typom, funkciou, alebo kľúčovým slovom, ktoré predchádza otvorením bloku. Uzatváracia množinová zátvorka je na samostatnom riadku, odsadená v rovnakej vzdialenosti, ako začína odsadenie, na ktorom je otváracia zátvorka.

Každý vnorený blok je odsadený o šírku jedného tabulátora ďalej od začiatku riadka oproti bloku, v ktorom je vnorený.

Čitateľnosť kódu a komentáre

Funkcia musí byť písaná tak, aby sa jej celé telo zmestilo do maximalizovaného okna so zdrojovým kódom vo vývojovom prostredí Eclipse pri rozlíšení obrazovky 1280x1024.

Komentáre v zdrojovom kóde nie sú vyžadované podľa hesla: „Najlepším popisom je samotný kód.“ Ak však programátor uzná uvedenie komentára za vhodné, používa jednoriadkový komentár uvedený znakmi “//“.

7 Manažment verzií, konfigurácií a zmien

Nástroj manažmentu verzií

Pri práci na tímovom projekte sme na verziovanie zdrojových kódov zvolili nástroj GIT. Prácu s týmto klientom realizujeme cez príkazový riadok. V nasledujúcich kapitolách opíšeme procesy realizované pri manažmente verzií. Táto metodika definuje správny spôsob práce členov tímu s týmto nástrojom počas vývoja softvérového projektu.

Použité pojmy

Repozitár – balíček súborov nad ktorým sa vykonáva verziovanie
Klonovanie repozitára – vytvorenie lokálnej verzie centrálneho repozitára
Commit – potvrdenie zmien súborov a uloženie zmien do histórie
Push – odoslanie commitov do centrálneho repozitára
Pull – stiahnutie a zosynchronizovanie commitov na lokálnej verzii
Ignore list – zoznam súbor označených ako ignorovateľných pri kontrole a úprave verzií
Origin – repozitár z ktorého bola vytvorená lokálna verzia repozitára
Master vetva – označovaná tiež ako hlavná vetva, mala by zodpovedať fungujúcej verzii softvéru ktorá sa nasadzuje na ostrú alebo testovaciu prevádzku.
Branch – alebo vetva, je verzia s konfliktnými zmenami oproti master vetve
GID – ID skupiny (UNIX, LINUX)
Gitweb – nástroj prístupný cez internet umožňujúci sledovať históriu zmien
Verejný kľúč – angl. public key, slúži na autentifikáciu používateľa
Plugin bundle – súbor pluginov

Použité repozitáre

V tejto kapitole je zoznam repozitárov so stručným popisom ich obsahu. Tieto repozitáre bolo možné na začiatku semestra prehliadať cez webové rozhranie nástroja GitWeb umiestnený na adrese *http://peweproxy-staging.fiit.stuba.sk/gitweb/*. Tento nástroj neskúr vystriedal vyspelejší nástroj Gitorious umiestnený na adrese *http://gitbus.fiit.stuba.sk*.

- **adaptive-proxy:** Obsahuje zdrojové súbory samotného adaptívneho proxy-servera
- **adaptive-proxy-plugins-core:** Obsahuje zdrojové súbory pluginov jadra servera
- **adaptive-proxy-plugin-template:** Šablóna pre nový plugin bundle
- **adaptive-proxy-bundle-messageboard:** Zdrojové súbory pre messageboard plugin bundle
- **adaptive-proxy-bundle-search:** Zdrojové súbory plugin bundlu pre API pracujúce s výsledkami vyhľadávačov
- **adaptive-proxy-web:** Webová stránka proxy-servera

- **adaptive-proxy-deploy:** Adresárová štruktúra používaná pre automatizovanom deploy
- **development-support:** Zdrojové súbory podporujúce závislosti medzi viacerými plugin bundlami používané pri vývoji

Role a zodpovednosti

Rola	Zodpovednosť
Vývojár	• Vykonanie zmien • Vytváranie vetiev repozitára
Manažér vývoja	• Zaznamenanie potreby na vytvorenie repozitára • Vypracovanie formálnej požiadavky na vytvorenie repozitára • Sledovanie zmien
Správca servera	• Nasadenie nástroja GIT • Nasadenie a konfigurácia nástroja Gitweb
Správca repozitára	• Vytvorenie a konfigurácia repozitára • Vytváranie vetiev repozitára • Spájanie vetiev repozitára
Tester	• Testovanie funkčnosti zmien
Dokumentarista	• Aktualizácia dokumentácie po vykonaných zmenách

Procesy manažmentu verzií

V tejto kapitole sú opísané procesy manažmentu verzií na vyššej úrovni. Podrobnejšie sú rozpísané procesy *manažment vetiev* a *vytváranie zmien, práca s repozitárom*.

Zaznamenanie požiadavky na vytvorenie repozitára

Vstup: požiadavka na návrh a vytvorenie repozitára
Výstup: nový repozitár
Zodpovedný: Manažér vývoja

Nové repozitáre sa vytvárajú pre každý plugin bundle reprezentujúci ucelenú funkcionalitu. Takúto požiadavku smeruje vývojár, ktorému bola pridelená funkcionalita na starosť.

Testovanie zmien

Vstup: scenáre testov, projektová špecifikácia, zdrojové súbory, testovací systém
Výstup: výsledky testov
Zodpovedný: tester

Testovanie vykonajte podľa scenárov. Je nevyhnutné vykonať všetky testy, nie len testy týkajúce sa zmenených alebo upravených zdrojových kódov. Testovanie sa vykonáva po vykonaní kľúčových zmien. Zdrojové súbory sú prístupné v repozitári.

Dokumentovanie zmien

Vstup: požiadavka na zmenu, špecifikácia zmien, zdrojové súbory
Výstup: aktualizovaná dokumentácia
Zodpovedný: dokumentarista

Zaznamenané zmeny je potrebné zapracovať do projektovej dokumentácie. Zaktualizujte dokumentáciu podľa pokynov. Zdrojové súbory sú prístupné v repozitári.

Manažment verzií

Manažment verzií sa používa na sledovanie zmien dokumentov, programov alebo iných informácií uložených na disku ako súbory. Keďže GIT je distribuovaný systém, nespadá do štandardnej klient/server architektúry. Miesto toho sa toho sa dá rozdeliť na origin repozitár a lokálny repozitár. Origin repozitár umiestnený na serveri ku ktorému sa budú pripájať vývojári budeme v tejto metodike nazývať centrálny repozitár.

Nasledovné kapitoly popisujú postupy pri manažovaní verzií, vytváraní nových vetiev a štandardy dodržované pri odosielaní commitov.

Vytváranie zmien, práca s repozitárom

Vstup: požiadavka na zmenu softvérového projektu, centrálny repozitár
Výstup: centrálny repozitár s vykonanými zmenami softvérového projektu
Zodpovedný: vývojár

	Krok
1.	Vytvorenie lokálnej verzie repozitára
2.	Stiahnutie zmien z centrálneho repozitára do lokálneho
3.	Zobrazenie rozdielov medzi lokálnym repozitárom a centrálnym
4.	Pridanie súborov do indexu commitu
5.	Označenie súborov určených na zmazanie
6.	Premenovanie súborov
7.	Commit zmien
8.	Odoslanie zmien do centrálneho repozitára

Vytvorenie lokálnej verzie repozitára

Vytváranie lokálnej verzie repozitára vykonávajte príkazom *git clone [url centrálneho repozitára]*. Zoznam repozitárov sa nachádza v kapitole 3 „*Použité repozitáre*".

Stiahnutie zmien z centrálneho repozitára do lokálneho

Tento krok sa odporúča vykonávať aspoň dva krát za pracovnú zmenu. Stiahnutie zmien z centrálneho repozitára do lokálneho vykonajte príkazom git pull.

Zobrazenie rozdielov medzi lokálnym repozitárom a centrálnym

Pred odoslaním zmien na server overte rozdiely medzi lokálnym repozitárom a centrálnym. Na overenie rozdielov použite príkaz git status. Výstupom na konzolu je zoznam súborov líšiacich sa od centrálnej verzie a počet commitov čakajúcich na odoslanie, prípadne počet commitov ktoré neboli spojené s lokálnou verziou. V prípade, že na server boli odoslané commity, ktoré neboli zatiaľ spojené s lokálnou verziou je nevyhnutné spustiť príkaz git pull (kapitola 7.1.2), prípadne vytvoriť novú vetvu (kapitola 7.2.2).

Pridanie súborov do indexu commit

Súbory pridajte do indexu príkazom *git add [názov súboru]* alebo interaktívnou formou spustením príkazu *git add –i*. Index commitu je možné sledovať príkazom *git status*.

Označenie súborov určených na zmazanie

Súbory zmazané v lokálnej verzii je nutné označiť k zmazaniu aj v centrálnej verzii *git rm [názov súboru]*. Takto zmazané súbory nie sú zmazané z histórie nástroja GIT.

Premenovanie súborov

Premenované súbory sa vo výpise *git status* zobrazujú ako zmazané a zároveň pridané pod iným názvom. Súbory je nutné označiť na vymazanie a následne pridať pod novým názvom. Nástroj GIT ich rozozná a v statuse označí ako premenované.

Commit zmien

Commit prebieha v lokálnej verzii repozitára. Pre jeho vykonanie spustite príkaz git commit. Po zadaní príkazu vyplňte komentár commitu. Pri komentári dodržujte nasledovný formát. Prvý riadok je názov commitu, nasleduje prázdny riadok a komentár commitu. Commity ktoré nedodržia takýto formát sú považované za chybné.

Odoslanie zmien do centrálneho repozitára

Zmeny odosielajte do centrálneho repozitára príkazom git push. Túto akciu vykonávajte po vytvorení jedného prípadne niekoľkých commitov v lokálnom repozitári.

Manažment vetiev

Vstup: repozitár s dvoma alebo viacerými vetvami, požiadavka na spojenie vetiev
Výstup: repozitár so spojenými vetvami
Zodpovedný: správca repozitára

	Krok
1.	Zobrazenie existujúcich vetiev
2.	Vytvorenie novej vetvy
3.	Zmena aktívnej vetvy
4.	Spájanie vetiev
5.	Sledovanie histórie vetiev
6.	Zmazanie vetvy

Zobrazenie existujúcich vetiev

Zoznam existujúcich vetiev získate spustením príkazu git brach. Výstupom je zoznam vetiev, aktívna vetva je označená znakom *.

Vytvorenie novej vetvy

Novú vetvu vytvorte v prípade potreby práce viacerými vývojármi nad rovnakými súbormi, prípadne pri nutnosti odoslania na server testovacej verzie softvérového produktu.

Novú vetvu vytvorte príkazom git branch [názov vetvy]. Po vytvorení novej vetvy ostáva aktívna pôvodná. Pre prácu nad novo vytvorenou vetvou zmeňte aktívnu vetvu podľa postupu opísaného v kapitole 6.2.3.

Zmena aktívnej vetvy

Zmenu aktívnej vetvy vykonajte príkazom *git checkout [názov vetvy]*. Úspešnosť príkazu je možné overiť príkazom *git brach*, opísaného v kapitole 6.2.1.

Spájanie vetiev

Spájanie vetiev vykonajte príkazom `git merge [názov vetvy]`. Spojená bude aktívna vetva s vetvou posunutou ako parameter príkazu.

Sledovanie histórie vetiev

Pre zobrazenie histórie vetiev použite príkaz `gitk`. Výstupom je grafické zobrazenie vetiev s vyznačenou históriou.

Zmazanie vetvy

Zmazanie vetvy vykonajte príkazom *git branch -d [názov vetvy]*. Tento príkaz overí, či sú zmeny z tejto vetvy už v aktívnej vetve. V prípade nutnosti zmazania vetvy vez synchronizácie zmien použite príkaz *git branch -D [názov vetvy]*.

8 Prílohy

K dokumentu prikladáme ako prílohy zápisnice zo stretnutí a preberacie protokoly.