Srovnání vývoje a výkonu mobilní aplikace tvořené nativně pro operační systém Android nebo pomocí HTML5

Comparison of the development and performance of native mobile applications created for Android OS or using HTML5.

Bc. Lukáš Horňák

Diplomová práce
2013

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Lukáš HORŇÁK**
Osobní číslo:	**A11434**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Počítačové a komunikační systémy**
Forma studia:	**prezenční**
Téma práce:	**Srovnání vývoje a výkonu mobilní aplikace tvořené nativně pro operační systém Android nebo pomocí HTML5**

Zásady pro vypracování:

1. **Vytvořte rešerši na téma tvorby nativních aplikací pro OS Android a také pomocí HTML5.**
2. **Srovnejte výhody a nevýhody a uveďte specifika obou přístupů pro tvorbu mobilní klient-server aplikace.**
3. **Naprogramujte testovací klient-server aplikaci jako nativní aplikaci OS Android a také pomocí HTML5 a v obou případech srovnejte aspekty samotné publikace.**
4. **Na vytvořených testovacích aplikacích proveďte srovnávací testy výkonu obou vytvořených aplikací při konzumaci dat ze serveru a komunikaci se serverem, ale také samotného běhu aplikace a zpracování na straně klienta.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **HOGAN, Brian P. HTML5 a CSS3: výukový kurz webového vývojáře. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 978-80-251-3576-1.**
2. **LUBBERS, Peter, Brian ALBERS a Frank SALIM. HTML5: programujeme moderní webové aplikace. Vyd. 1. Brno: Computer Press, 2011, 304 s. ISBN 978-80-251-3539-6.**
3. **STARK, Jonathan a Brian JEPSON. Building Android apps with HTML, CSS, and JavaScript. 2nd ed. Sebastopol, CA: O'Reilly, 2012, xiii, 158 p. ISBN 14-493-1641-7.**
4. **WALLACE, Jackson. Android Apps For Absolute Begginers. New York: Apress, 2011. ISBN 978-1-4302-3446-3.**
5. **MURPHY, Mark L. Beginning Android 2. New York: Apress, 2010, xvi, 397 s. Books for profesionals by professionals. ISBN 978-1-4302-2629-1.**
6. **JORDAN, Lucas L a Pieter GREYLING. Practical Android projects. New York: Apress, 2011, xiv, 404 p. ISBN 14-302-3243-9.**

Vedoucí diplomové práce: **Ing. Radek Vala**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **26. února 2013**

Termín odevzdání diplomové práce: **31. května 2013**

Ve Zlíně dne 26. února 2013

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Karel Vlček, CSc.
ředitel ústavu

ABSTRAKT

Táto práca si kladie za cieľ zrovnať teoreticky aj prakticky výhody a nevýhody klient-server natívnych a multiplatformových aplikácii a na vytvorených testovacích aplikácii vykonať porovnávajúce testy výkonu oboch vytvorených aplikácii. Vytvorená multiplatformová aplikácia využíva technológie HTML5, CSS štýly a jQuery Mobile. Natívna aplikácia je určená pre Android zariadenia. Server beží na Java serveri Tomcat 7. Dáta sú serializované pomocou GSON do formátu JSON.

Kľúčové slová: Android OS, HTML5, klient-server, jQuery Mobile, Tomcat, Json, Gson

ABSTRACT

The aim of the thesis is compare theoretical and practical advantages and disadvantages of client-server native and cross-platform application and on created test application perform tests comparing the performance of both developed application. Created a multiplatform application uses technology HTML5, CSS styles and jQuery Mobile. Native app is designed for Android devices. Server is running on Java Tomcat 7 The data is serialized using GSON to JSON.

Keywords: Android OS , HTML5, client-server, jQuery Mobile, Tomcat, Json, Gson

Ďakujem vedúcemu mojej diplomovej práce Ing. Radkovi Valovi za odborné vedenie, pripomienky, cenné rady a čas ktorý mi venoval pri vedení tejto práce. Ďalej sa chcem poďakovať Bc. Tomášovi Sokolovi, za drobné rady pri riešení problémov spojené s touto prácou.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- -beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

Platformy Android a HTML5 sú v dnešnej dobe populárne. Android v mojej práci vystupuje ako zástupca natívnej aplikácie a HTML5 reprezentuje multiplatformovú aplikáciu. Výhodou natívnej aplikácie je možnosť využiť maximum možnosti čo poskytuje daná platforma, avšak na každú platformu je potrebná iná aplikácia napísaná v inom jazyku, čo nám sťažuje vývoj aplikácie aj pre iné platformy. Na rozdiel od natívnej aplikácie nám mobilná multiplatformová ponúka akúsi slobodu naprieč platformami, efektívny a rýchly vývoj aplikácie pre všetky platformy a s použitím vhodných technológii dokáže využívať aj HW súčasti zariadenia ako sú kamera, navigácia, senzory a iné.

Táto práca je rozdelená na dve časti. Teoretická časť sa zaoberá popisom použitých technológii pri vyvíjaní natívnej a multiplatformovej aplikácie. Zahŕňa zrovnanie výhod a nevýhod prístupov pre tvorbu mobilných klient-server aplikácii a typy riešení pre klient-server aplikácie.

Druhá, praktická časť, sa zaoberá konkrétnou tvorbou natívnej a mobilnej aplikácie pomocou spomenutých technológii. Popisuje aj tvorbu serverovej strany a použité technológie. Na vytvorených testovacích aplikáciách sa vykonajú porovnávacie testy výkonu oboch vytvorených aplikácii pri konzumácii dát zo servera a pri komunikácii so serverom. Zároveň sa otestuje aj samotný beh oboch aplikácií a rýchlosť spracovania dát na strane klienta. Výsledky testovania bude možné porovnať v názorných grafoch, kde bude možné pozorovať testy výkonu a rýchlosť behu aplikácii.

Na záver je priložené CD so zdrojovými kódmi oboch aplikácii a aj zdrojové kódy serverovej strany spolu s nameranými výsledkami umiestnené v hárkoch v Exceli.

I. TEORETICKÁ ČASŤ

1 NATÍVNA APLIKÁCIA VERZUS WEBOVÁ APLIKÁCIA

1.1 Webová aplikácia

Typicky sa jedná o webovú stránku optimalizovanú pre použitie na smartfónoch. Obsah takejto webovej stránky môže byť hocičo, štartujúc brožúry firiem až po hypotékovú kalkulačku alebo počítadlo kalórii, takže obsah je irelevantný. Definujúca charakteristika pre webovú aplikáciu je, že používateľské rozhranie (GUI) je vytvorené využitím štandardných webových technológii, je dostupná na URL (verejná, súkromný, alebo za loginom) a je optimalizovaná pre charakteristiky mobilného zariadenia. Webová aplikácia nie je inštalovaná na telefóne, nie je dostupná v Google Play a nie je napísaná v Jave. [1]

Predchádzajúce tvrdenie, že webová aplikácia nie je inštalovaná a nie je v Android Markete sa dá čiastočne vyvrátiť s použitím hybridnej technológie PhoneGap.[11] Takáto aplikácia sa dá nainštalovať na mobilný telefón a umiestniť v Android Markete. Táto možnosť však bola nutná z hľadiska komerčného, aby sa vytvorená webová aplikácia dala prípadne predávať na Markete.

1.1.1 Výhody webových aplikácii [1]

- Programátor môže využiť aktuálne znalosti
- V prípade klient server aplikácie sa pri oprave chýb aktualizácia prejaví v reálnom čase (v opačnom prípade, ak je aplikácia zabalená pomocou PhoneGapu a umiestnená na Google Play sa zmeny prejavia po aktualizácii aplikácie)
- Aplikácia bude bežať na hocijakom zariadení, ktoré má webový prehliadač
- Vývojová fáza je veľmi rýchla

1.1.2 Nevýhody webových aplikácii [1]

- Nemožnosť pristupovať k hardvérovým súčastiam mobilného telefónu (neplatí pri použití PhoneGap-u)
- Potreba vymyslieť platobný systém ak má byť aplikácia spoplatnená (rieši PhoneGap)
- Ťažké dosiahnuť sofistikované GUI efekty
- Častejšie problémy s umiestňovaním a zobrazovaním GUI
- Zložitejšia DOM štruktúra

1.2 Natívna aplikácia

Na rozdiel od typickej webovej aplikácie, natívna aplikácia je nainštalovaná na mobilnom telefóne (Android), má prístup ku hardvéru telefónu (reproduktory, akcelerometer, kamera, atď) a je napísaná v jazyku Java. Definujúca charakteristika je fakt, že aplikácia je umiestnená v Android Markete. [1]

1.2.1 Výhody natívnych aplikácii [1]

- Nespočetné množstvo používateľov so zadanými číslami kreditných kariet v Android Markete – možnosť zárobku
- Možnosť využívať všetky hardvérové súčasti mobilného telefónu (kamera, gps, akcelerometer a iné)
- Rýchlejší a svižnejší chod aplikácii
- Efektívne, dynamické a prehľadné GUI aplikácie využívajúc všetkých možnosti danej platformy

1.2.2 Nevýhody natívnych aplikácii [1]

- Členstvo Android vývojárov je spoplatnené
- Aplikácia beží len na danej platforme (Android)
- Vývoj aplikácie v jazyku Java
- Vývojový cyklus je pomalý (vývoj, kompilácia, nasadenie, opakovanie)

2 ANDROID OS

Obrázok 1 Android OS logo[1]

Android OS je veľmi populárny operačný systém pre mobilné telefóny v dnešnej dobe. Je založený na otvorenej platforme open source. Android OS veľmi rýchlo rastie na trhu, každý deň sa na miliónov nových zariadení aktivuje nová inštalácia tohto operačného systému. Výhodou tohto OS oproti inými OS na trhu s mobilnými telefónmi, je tzv. začlenenie do Google ekosystému. Ide o integráciu väčšiny z Google Apps priamo do mobilného telefónu. Jedná sa hlavne o Gmail, Google+, Google Calendar, Google Contacts, Google Picassa a iné aplikácie od Googlu. Všetky tieto možnosti sa sprístupnia po zadaní prihlasovacích údajov užívateľa Google účtu. Mobilný telefón sa dá využívať aj bez prihlásenia, ale používateľ prichádza o spomenuté výhody.

Vďaka otvorenosti Android OS, teší sa veľkej obľube vývojárov. Každým vydaním novej verzie OS, prináša vývojárom nové možnosti vytvárať výkonné a rôznorodé aplikácie využívajúce najnovšie mobilné technológie. [2]

Android OS umožňuje vývojárovi písať aplikácie pre rôzne veľkosti displejov, rozlíšenia, hustoty bodov na obrazovke, jazyka zariadenia. GUI aplikácie sa dokáže prispôsobiť podľa

[1] Android Logo. *Blogspot* [online]. 2012 [cit. 2013-05-10]. Dostupné z: http://3.bp.blogspot.com/-_rPFRZxAD3o/T_2olBDqpgI/AAAAAAAAAJg/9nsPgBFcqus/s320/Android.jpg

toho, na akom zariadení beží. Pokiaľ vývojár definuje rôzne vzhľady GUI pre tablet a mobilné telefóny, potom Android OS použije práve ten GUI pre konkrétne zariadenie na ktorom aplikácia beží. [2]

Pre distribúciu a predaj aplikácii používa Android OS vlastný marketplace Google Play. Na tomto mieste sa aplikácie distribuujú do zariadení s Android OS. Vývojár si môže zvoliť či bude aplikácia zadarmo alebo speňažená, trh, na ktorý sa má aplikácia distribuovať dokonca môže vývojár zamerať len na špecifický hardvér alebo typ mobilného telefónu. [2]

2.1 Stručná história Android OS

Android OS existuje od Októbra 2009, kde začínal na na verzii 1.5 s kódovým označením CupCake. Túto verziu operačného systému mal smartphone HTC Dream, inak známy pod názvom G1. Bol to jeden z prvých komunikátorov s Android OS. Nasledovala verzia 1.6 s kódovým označením Donut. Ďalej nasledovali verzie 2.1 (Eclair) a 2.2 (Froyo). Až verzia 2.3 (Gingerbread) sa dostala viac do povedomia používateľov. Medzistupňom bola verzia 3.2 (Honeycomb), ktorá bola určená len pre tabletové zariadenia. Najväčšiu zmenu funkčnosti a GUI priniesla až verzia 4.0.3/4.0.4 (Ice Cream Sandwich). Aktuálne najnovšou verziou Android OS je verzia 4.1.x (Jelly Bean), ktorá je v každom novom zariadení na trhu a 4.2.x (Jelly Bean), ktorá je používaná hlavne na Nexus (smartphone priamo od spoločnosti Google s vždy aktuálnymi verziami OS) zariadeniach. [3]

Najnovšia oznámená verzia je 5.0 (Key Lime), ktorú údajne ale budú odkladať, a príde ešte medziverzia Jelly Beanu 4.3. [4]

Obrázok 2 História verzii Android OS s logami verzii[2]

[2] Veriqual. *History of Android OS, from Cupcake to Jelly* [online]. 2013 [cit. 2013-05-10]. Dostupné z: http://www.veriqual.com/wp-content/uploads/2013/03/androidVersions.jpg

3 JAVA (PROGRAMOVACÍ JAZYK)

Obrázok 3: Logo programovacieho jazyka JAVA [5]

Java je konkurencieschopný objektovo orientovaný jazyk, ktorý sa s obmenami používa aj pri programovaní Android OS. Umožňuje vývojárom programovať v duchu „napíš raz, pusti hocikde“ (WORA – write once, run anywhere). To znamená, že kód fungujúci na jednej platforme nemusí byť prekompilovaný aby pracoval na inej platforme. Java aplikácie sú skompilované a bežia na Java virtuálnej mašine (JVM) bez ohľadu na architektúru alebo platformu. Java je aj v roku 2013 jeden z najpopulárnejších jazykov (druhý najpopulárnejší[3], po klasickom jazyku C). Syntax jazyka Java vychádza z jazykov C a C++, takže pre vývojárov je ľahko naučiteľný pri prechode z týchto jazykov. Prvé vydanie Java sa datuje k roku 1995, v roku 2007 Java prešla pod licenciu GNU. [5]

Medzi hlavné ciele pri vytváraní jazyka Java patrí[4]:

- Jednoduchý, objektovo-orientovaný a známy.
- Robustný a bezpečný
- Architektovo neutrálny a prenosný.
- Spustiteľný s vysokým výkonom.
- Interpretovateľný, viacvláknový a dynamický.

Jedným z kľúčových parametrov Javy je prenositeľnosť čo znamená, že programy napísane v Jave musia fungovať rovnako bez ohľadu na hardvér alebo operačný systém. Táto

[3] Programming Community Index for April 2013. *Tiobe software* [online]. 2013 [cit. 2013-05-08]. Dostupné z:http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[4] The Java Language Environment. In: *Oracle* [online]. 1997 [cit. 2013-05-08]. Dostupné z: http://www.oracle.com/technetwork/java/intro-141325.html

vlastnosť je dosiahnutá kompilovaním to tzv. Java bytekódu, na rozdiel od kompilovania priamo do špecifického zdrojového kódu použitej platformy. Kód v Java bytekóde sa spúšťa na virtuálnej mašine (VM) napísanej priamo pre hardvér na ktorom beží. Na koncových staniciach (počítačoch) sa bežne používa Java Runtime Enviroment (JRE), ktorý je nainštalovaný pre spustenie štandardných Java aplikácii. [5]

Programy napísane v jazyku Java majú reputáciu pomalých a pamäťovo náročných aplikácii ako programy napísane v jazyku C++. Tento neduh Javy bol napravený v rokoch 1997/1998 vo verzii Java 1.1 zavedením tzv. Just-in-time (JIT) kompilácie, zavedením vylepšení jazyka na lepšiu analýzu kódu a optimalizáciou Java virtuálnych mašín. Testy výkonov v roku 2012 ukazujú, že Java 7 je len priemerne 44% pomalšia než jazyk C++. [5]

Java používa tzv. Automatic garbage collector pre správu pamäte pri životnom cykle objektov. Programátor určuje, kedy sú vytvorené objekty a Java prostredie je zodpovedné za uvoľnenie pamäte po objektoch, ktoré už nie sú používané. Spustenie Garbage collector-u je garantované akonáhle je nedostatok pamäte pre alokovanie miesta pre nový objekt. [5]

Pri programovaní pomocou akéhokoľvek jazyka musí programátor zvoliť vývojové prostredie. Jazyk Java podporujú známe prostredia, často používaný Eclipse alebo NetBeans. Menej známe vývojové prostredia sú napr. IntelliJ IDEA, JDeveloper a BlueJ. [5]

4 HTML 5

Obrázok 4 HTML 5 logo[5]

HTML (HyperText Markup Language) je značkovací jazyk na vytváranie webových stránok a iných foriem informácii zobrazených vo webových prehliadačoch. Umožňuje vytvárať textové, multimediálne a hypertextové elementy. Vo verzii HTML 4.0 podporuje viac multimediálnych možností, podpora skriptovacích jazykov, šablóny štýlov, lepšie tlačové možnosti. HTML 4.0 prináša lepšiu podporu internacionalizácie dokumentov tak aby bol Web naozaj World Wide (aby bol web svetovo rozsiahly). [6]

HTML 5 je najnovšou verziou tohto jazyka vytvorený v decembri 2012. Stále je avšak v štádiu draft, vo verzii 5.1. Plánuje sa aj verzia 5.2, ktorá má byť k dispozicii v roku 2015. HTML 5 prináša hlavne nové elementy, ako sú *<video>* *<audio>* a *<canvas>*. Niektoré elementy a parametre odstraňuje, niektoré redefinuje a štandardizuje. Podlieha štandartu W3C. [7,8]

Jazyk HTML sa skladá z viacerých elementov, ako sú tagy, atribúty tagov. HTML tag sa skladá z dvoch častí. Z otváracieho tagu a uzavierajúceho tagu. Každý tag sa skladá z názvu tagu uzavretého medzi lomenými (zátvorky „<“ a „>“). Uzavierajúci tag má navyše pred názvom tagu lomítko („/“). Príklad oboch tagov: *<B>Hello World</B>*. [7]

[5] Best UI Psd. *HTML 5 logo* [online]. 2011 [cit. 2013-05-10]. Dostupné z: http://2.bp.blogspot.com/_UZImdYAiry8/TT_dddlfzJI/AAAAAAAAVak/zyoqeywfxn0/s1600/html5_logo.jpg

Tagy väčšinou nestačia, a preto ich HTML jazyk rozširuje o atribúty. Medzi príklad atribútu patrí napr. atribút *width*, ktorý určuje šírku elementu. Zapisuje sa napr. takto: *<td witdh=”100%”></td>*. [10].

5 PHONEGAP

Obrázok 5 Logo PhoneGap[6]

Budovanie softvéru pre rôzne mobilné platformy je ťažké a časovo náročné, pretože každá platforma má svoj špecifický programovací jazyk. Konkurenčný iPhone používa Objective-C, Android OS používa jazyk Java, Windows Phone podporuje hlavne C# a kedysi populárny Symbian používal prevažne Python alebo Symbian C++. PhoneGap umožňuje pomocou HTML5, CSS a JavaScriptu (jQueryMobile, viď. 4. kapitola) vyvíjať mobilné aplikácie naprieč mobilným platformám. [11]

PhoneGap podporuje všetky populárne mobilné platformy, medzi ktoré patrí iPhone, Android, Windows Mobile, BlackBerry, Bada, Symbian a aj menej známe webOS a Tizen. [11]

PhoneGap vznikol 4. Októbra 2011 čomu predchádzala výhra ocenenia na výstave Web 2.0 Expo v roku 2009. PhoneGap je vytvorený spoločnosťou Nitobi Software a odkúpená spoločnosťou Adobe Systems. Názov PhoneGap ponechali napriek tomu, že po odkúpení dostalo názov Adobe PhoneGap alebo Adobe PhoneGap Build. Po odkúpení PhoneGap spoločnosťou Adobe vznikla open source varianta vedená pod názvom Apache Cordova. [12]

Webové aplikácie sa dajú písať aj čisto v HTML 5, s použitím jQuery (viď. 4. kapitola) a CSS štýlov ale táto aplikácia nedokáže používať hardvér mobilného telefónu, jeho senzory a iné. Práve toto umožňuje PhoneGap. Pridáva podporu pre používanie akcelerometru telefónu (pohyb v troch osách), kameru, kompas, pripojenie telefónu,

[6] Devgirl's webblog. *PhoneGap logo* [online]. 2012 [cit. 2013-05-10]. Dostupné z: http://devgirl.org/wp-content/uploads/2012/05/phonegaplogo.png

prístup ku kontaktom telefónu, informácie o zariadení, obsluha udalostí, prístup k súborovému systému mobilného telefónu, geolokácia (použitie GPS zariadenia), nahrávanie a prehrávanie audio súborov, využitie notifikácii mobilného zariadenia a iné. [13].

Samotný projekt PhoneGap nerieši vzhľad mobilnej aplikácie. Programátor preto musí dizajn naprogramovať úplne sám, čo však je veľmi časovo náročné alebo použiť už nejaké hotové riešenie v podobe frameworku. Najznámejšou dizajnovou nadstavbou je jQueryMobile, ktorú rozpíšem viac v nasledujúcej kapitole. Jedná sa ale o známy framework, ľahko použiteľný a ľahko naučiteľný. Existujú aj iné nadstavby ako sú SenchaTouch (kompletné riešenie pre robustné JavaScript aplikácie) alebo KendoUI postavené práve na jQueryMobile. [14]

6 JQUERY MOBILE

Obrázok 6 jQueryMobile logo [15]

jQueryMobile je jednotný systém pre užívateľské rozhranie pre všetky populárne mobilné a tabletové platformy. Využíva tzv. motto spoločnosti: „napíš menej, sprav viac“ a vyzdvihuje ho na novú úroveň. Namiesto programovania jednej mobilnej aplikácie pre každý mobilný OS prichádza z možnosťou dizajnovať webové aplikácie pre všetky populárne mobilné a talbetové platformy len vďaka HTML 5 a jQuery nadácie (JavaScrip). jQuery Mobile podporuje všetky známe populárne mobilné OS, ako sú iOs, Android, BlackBerry, Bada, Windows Phone, Symbian, MeeGo a palm webOS. [15]

Tento framework je ľahko použiteľný, k čomu nasvedčuje aj drag-and-drop editor používateľského rozhrania, kde si môže vývojár vyskúšať čo tento framework dokáže a ihneď začať s vyvíjaním aplikácie. Tento editor dokáže vygenerovať priamo HTML 5 kód, ktorý je už použiteľný v projekte vývojára. [15]

jQuery Mobile ide ruka v ruke s filozofiou PhoneGapu (viď. 3. kapitola), takže sa často využíva v spojení s týmto frameworkom, kde sa dokážu výrazne znásobiť možnosti oboch technológii a využiť maximum.

7 JSP A JAVA SERVLETY

7.1 Java Servlety

Servlety sú odpoveďou Javy na otázku CGI programovania. Sú to programy, ktoré bežia na webovom servery a dynamicky generujú obsah webových stránkach. Táto metóda programovania webových stránok sa používa hlavne preto, pretože stránky sú odrazom dát zadanými užívateľmi. Typicky ide o vyhľadávajúce služby, kde stránka s výsledkami je generovaná po hľadajúcom dotaze užívateľa alebo pri elektronických obchodoch, kde celý obchod je dynamická stránka. Ďalšou potrebou generovať webové stránky pomocou servera je, že dáta na internete sa často menia a treba ich na stránke aktualizovať. Typický príklad tohto použitia sú stránky s počasím, kde sa dáta menia z hodiny na hodinu a stránky. Hlavným účelom serverovo generovanými stránkami je potreba pracovať s dátami, ktoré sú uložené v databázach alebo iných podobných zdrojoch. [16]

Medzi výhody Java Servletov patrí väčšia efektivita, ľahká použiteľnosť, lacnejšie náklady pri vývoji ako pri konkurenčných CGI technológiách. Ďalšou výhodou je rýchle spracovávanie http požiadaviek. Rýchlejšia odozva na tieto požiadavky. Nemenej dôležitá výhoda tejto technológie je, keď už vývojár ovláda jazyk Java, a chce vyvíjať dynamické weby pomocou serveru, tak sa už nemusí učiť iné konkurenčné jazyky (napr. Perl) ale rovno vyvíjať servlety, ktoré už majú bohatú infraštruktúru pre spracovanie HTML formulárových dát, zadávanie http hlavičiek, podpora cookies, trackovanie relácii a iné možnosti. [16]

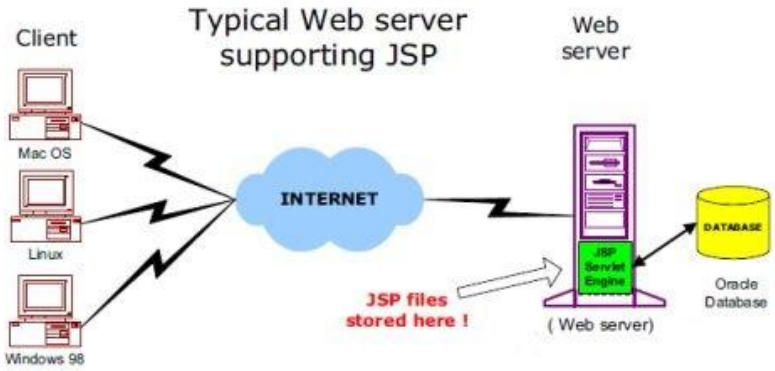
7.2 JSP

JSP (Java Server Pages) sú Java serverové stránky, ktoré umožňujú vytvárať statický HTML obsah dynamicky generovaným HTML kódom na serveri. [16]

Oproti podobnej technológie od Microsoftu s názvom ASP (Active Server Pages) má JSP tri podstatné výhody. Dynamické časti JSP stránok sú písane v Jave a nie špecifickom jazyku od Microsoftu, JSP stránky sú prenosné naprieč operačnými systémami (použiteľné ako na Unixe tak aj na Microsofte). JSP oddeľuje dizajnovú časť od dynamickej dátovej časti. Webový dizajn navrhne dizajnér a vývojár JSP stránok naprogramuje servlet, ktorý bude dynamicky generovať obsah stránky. [16]

7.2.1 Architektúra JSP stránok

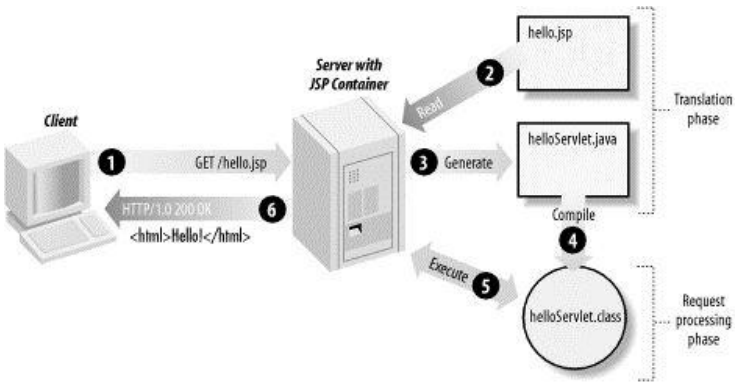
Webový server potrebuje JSP prostriedok alebo kontajner na vykonávanie JSP stránok. JSP engine je zodpovedný za zachytenie požiadaviek na JSP. JSP kontajner pracuje s webovým serverom aby poskytovalo runtime prostredie a iné služby, ktoré JSP potrebuje. [17]



Obrázok 7 Architektúra JSP stránok [17]

7.2.1.1 Spracovanie JSP stránok

Webová stránka pošle HTTP požiadavky webovému serveru, server zistí, že je určená pre JSP stránku a presmeruje túto požiadavku na JSP server. JSP server stránku spracuje a servlet odošle odpoveď v HMTL formáte webovému prehliadaču, ktorý ju následne vykreslí. [17]



Obrázok 8 Diagram spracovania JSP stránok [17]

7.2.2 Životný cyklus JSP stránok

Z hľadiska funkčnosti JSP stránok, je dôležité pochopenie životného cyklu JSP stránok.

Životný cyklus je tvorený 4-mi fázami: [18]

- Kompilácia

- Inicializácia
- Vykonávanie
- Čistenie

7.2.2.1 JSP Kompilácia

Ak požadovaná stránka nebola nikdy kompilovaná, alebo ju JSP modifikovalo od poslednej kompilácie, JSP engine skompiluje stránku.

Kompilácia pozostáva z troch krokov, a to:

- Parsovanie JSP kódu
- Premena JSP na servlet
- Kompilácia servletu [18]

7.2.2.2 JSP Inicializácia

Po načítaní JSP serverom, sa vykoná metóda pred spracovaním akéhokoľvek požiadavku. Inicializácia sa vykonáva pomocou tohto kódu:

Typicky sa jedná o úkony, ktoré sa vykonajú len raz, napr. otvorenie potrebných súborov alebo otvorenie databázových spojení. [18]

7.2.2.3 JSP Vykonávanie

Hlavná fáza životného cyklu JSP, pokiaľ nie je JSP zničený. Až v tejto fáze sa vykonáva hlavná časť servletu.

Metóda *_jspService()* je vykonaná vždy raz za požiadavku a je zodpovedná za generovanie odpovede pomocou HTTP metód odpovedi ako je GET, DELETE, POST a iné. [18]

7.2.2.4 JSP Čistenie

Deštrukčná fáza životného cyklu JSP stránok. JSP stránka sa odstráni z používania v JSP engine. Dá sa povedať, že je to opačná fáza incializačnej fázy, v ktorej uzavrieme vytvorené databázové spojenia alebo uzavrieme otvorené súbory. Vykonávame pomocou metódy *_jspDestroy()*. [18]

8 WEBOVÁ SLUŽBA

Definícia webovej služby je v mnoho prípadoch odlišná. Rôzne knihy a organizácie uvádzajú rôzne definície. Zopár definícii sú uvedené nižšie a každá definícia webových služieb je správna: [19]

- Webové služby sú XML informatické výmenné systémy, ktoré používa internet na priame interakcie medzi aplikáciami. Tieto systémy môžu zahrňovať programy, objekty, správy alebo dokumenty.
- Webová služba je každá časť softvéru, ktorá robí sama seba prístupnú naprieč internetom a používa štandardizovaný XML systém správ. XML je použitý na zakódovanie všetkej komunikácie do webovej služby. Napríklad, klient vyvolá webovú službu poslaním XML správy, potom čaká na XML odpoveď. Pretože je všetka komunikácia v XML, webové služby nie sú viazané na operačný systéme alebo programovací jazyk. Java dokáže komunikovať s Perl, Windowsové aplikácie dokážu komunikovať s Unixovými aplikáciami.
- Webová služba je kolekcia voľne dostupných protokolov a štandardov používaných pre výmenu dát medzi aplikáciami alebo systémami. Aplikácie napísané v rôznych programovacích jazykoch, bežiacimi pod rôznymi operačnými systémami môžu použiť webové služby na výmenu dát naprieč počítačovej siete ako je internet, ako aj komunikácia vrátane samotného počítača. Táto otvorenosť naprieč programovacími jazykmi a operačnými systémami je dosiahnutá vďaka použitiu otvorených štandardov.

V skratke o webových službách platí nasledujúce: [19]

- Je dostupná na internete ako aj na súkromných sietiach.
- Používa štandardizovaný XML komunikačný systém.
- Nie je viazaná na operačný systém alebo programovací jazyk.
- Je samo popisujúca vďaka známemu XML slovníku.
- Je objaviteľný vďaka jednoduchému vyhľadávajúcemu mechanizmu.

8.1 Komponenty webových služieb

Základná platforma webových služieb je XML + HTTP. Všetky štandardné webové služby používajú nasledujúce komponenty [19]:

- SOAP (protokol založený na XML, umožňuje komunikáciu pomocou HTTP)
- UDDI (adresárová služba, kde sa môžu firmy registrovať a vyhľadávať webové služby)
- WSDL (jazyk založený na XML pre lokalizáciu a popis webových služieb)

8.2 JSON

JSON (JavaScript Object Notation) je odľahčený formát pre výmenu dát. Pre človeka je jednoducho čitateľný a zapisovateľný. Pre počítače je ľahko generovateľný a analyzovateľný. JSON je textový, na použitom jazyku nezávislý formát, využívajúci konvencie známych programovateľných jazykov C (C++, C#, Java, JavaScript a iné) a práve preto je obľúbený pre výmenu dát. [20]

JSON má dve hlavné štruktúry: [20]

- Kolekcia párov názov/hodnota. Vo väčšine jazykov, je toto realizované objektami, záznamami, štruktúrami, hash tabuľkami a iné.
- Triedený zoznam hodnôt. Známe ako vektor, pole, zoznam alebo postupnosť.

Vyššie vymenované dátové štruktúry sú univerzálne. Všetky moderné programovacie jazyk ich v nejakej forme podporujú. Preto je práve na nich založený aj tento nezávislý výmenný formát. [20]

Základná štruktúra JSON je nasledujúca:

Objekt je netriedená množina párov názov/hodnota. Objekt začína znakom *„{“* a končí *„}“*. Každý pár názov/hodnota má medzi sebou znak *„:“* (dvojbodka) a každý tento pár je oddelený *„,“* (čiarkou). [20]

Príklad zápis dát vo formáte JSON:

```
{
      “Meno” : “Lukas”,
      “Priezvisko” : “Hornak”,
      “Adresa” : {
            “Ulica” : “Antoninova”,
            “Mesto” : “Zlin”
      },
      “Telefonne cislo” : 773123456
}
```

9 GITHUB

Obrázok 9 Logo GitHub-u [21]

GitHub je obdoba sociálnej siete pre programátorov. Umožňuje zverejňovať zdrojový kód projektov, spolupracovať viac ľudí na jednom projekte, hodnotiť zdrojový kód otvorenou diskusiou a verzovanie zdrojového kódu. [21]

Projekt GitHub bol spustený 17. Apríla 2008 a funguje doteraz s veľa vylepšeniami. [22]

Pre používanie GitHub-u je potrebná bezplatná registrácia na ich webových stránkach. Podmienky bezplatného používania GitHub-u vyžadujú, aby zverejnené knižnice so zdrojovým kódom boli verejné. Ak chce používateľ spolupracovať s niekým súkromne (kód nebude viditeľný verejne), musí si užívateľ priplatiť za vytvorenie súkromných knižníc. [21]

Pre potreby súkromných knižníc so zdrojovým kódom, je tu možnosť prepojenia GitHub-u s cloudovým riešením Dropbox. Pomocou Git rozšírenia je možné vytvoriť centrálnu knižnicu kódu na Dropbox-e a následne otvoriť túto knižnicu na inom PC s Dropboxom.

10 VÝHODY, NEVÝHODY, PRÍSTUPY PRI TVORBE KLIENT-SERVER APLIKÁCII

Pri tvorbe klient-server aplikácii môžeme pristupovať viacerými prístupmi, ktoré si popíšeme v tejto kapitole.

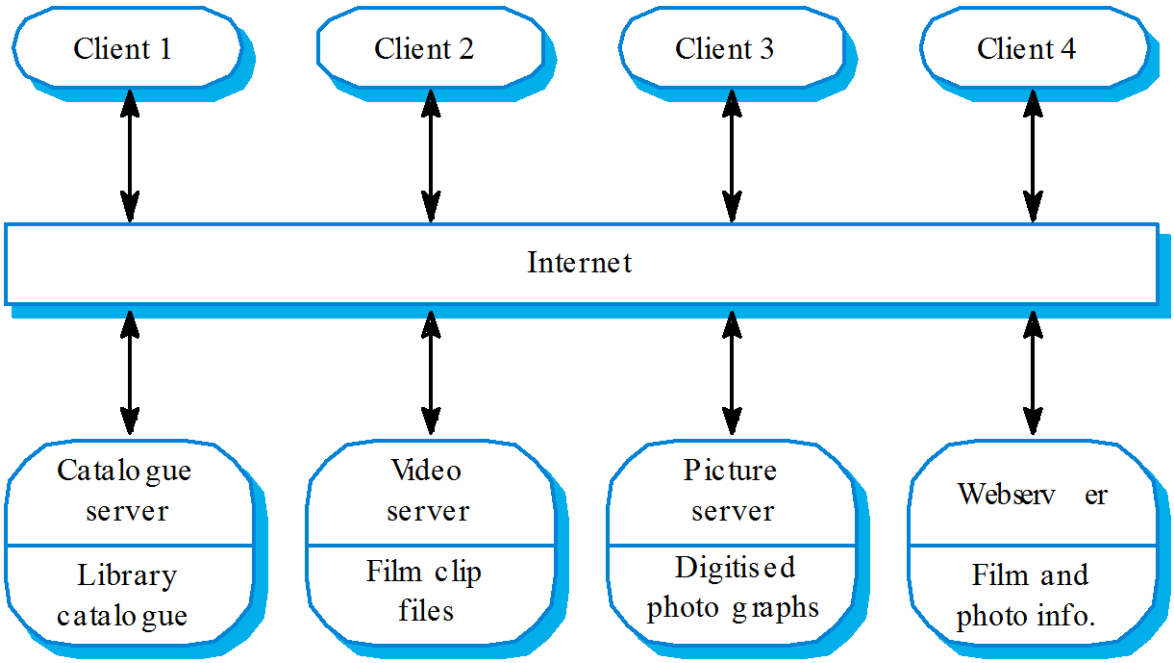
10.1 Výhody a nevýhody klient-server aplikácii

Architektúra klient-server sa skladá z troch častí:

- Server
- Sieť
- Klient

Hlavnou výhodou tohto riešenia sieťové prepojenie s možnosťou zdieľaného prístupu k údajom a výkon serverového PC (podľa typu klienta, viď. nasledujúca kapitola).

Každý server môže poskytovať špecifické funkcie (viď. Obrázok 11). Počítače (klienti) sú prepojené pomocou siete LAN a komunikujú so servermi. Dôležitá je aj rýchlosť siete.



Obrázok 10: Ukážka klient-server distribuovaný model. [23]

10.1.1 Výhody klient-server aplikácii [23]

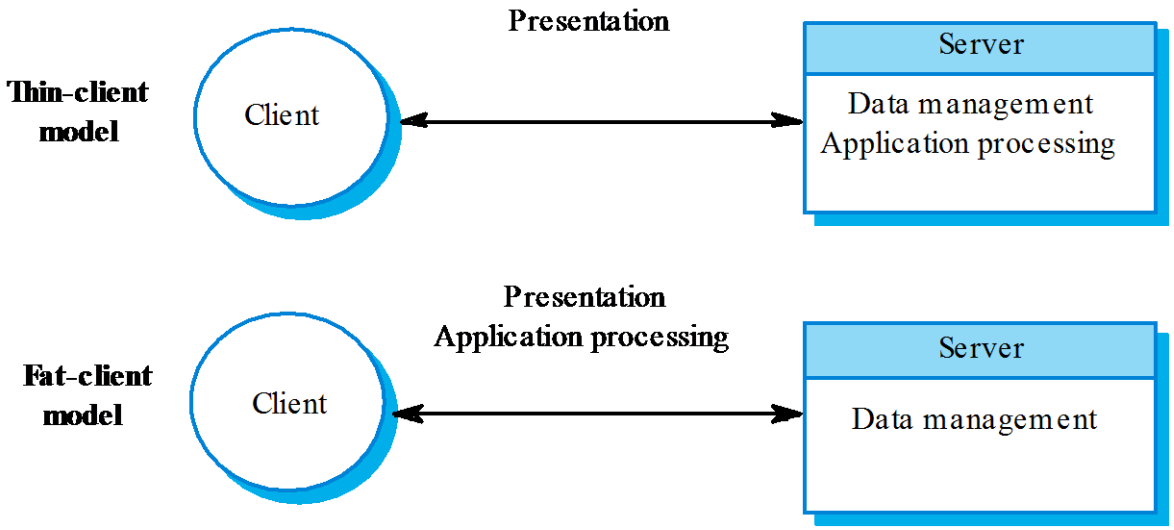
- Rozdelenie dát v sieti má jasnú štruktúru.

- Efektívnejšie pridávanie komponentov a upgrade komponentov.
- Efektívnejšie riešenie, využitie siete s lacnejšími sieťovými prvkami.

10.1.2 Nevýhody klient-server aplikácii [23]

- Náročnejšia správa servera – problém zálohovania
- Neexistujúci centrálny register služieb v systéme

Grafické porovnanie klient-server aplikácii, kde je znázornený postup spracovania dát pri použití tučného a tenkého klienta:



Obrázok 11: Porovnanie klientov. [23]

10.2 Tenký klient

Klient označený ako tenký, je taký, že klient obsahuje len zobrazovaciu časť aplikácie, pritom. Typicky ide o aplikáciu, ktorá sa spúšťa na vzdialenom systéme (Windows, Linux, Android, iOS a iné).Vzdialený konečný systém môže byť navrhnutý priamo na fungovanie klientskej aplikácie alebo lacný osobný počítač, keďže nemusí spracovávať ale len zobrazovať údaje. Klient posiela serveru požiadavky a server posiela naspäť zobrazovacie údaje. [24]

Tento prístup ku klient-server sa používa väčšinou pri modernizácii zastaraných systémov, kde sa doprogramuje klientska zobrazovacia časť a server sa len dodatočné upraví na spracovanie a posielanie dát. [23]

Výkon takejto aplikácie potom závisí na použitej platforme na servery, rýchlosti použitej siete. Hlavne použitá sieť môže byť pri výkone tohto prístupu kľúčová, pretože takáto aplikácia veľmi zaťažuje sieť. [23,24]

Výhodou tohto riešenia je aj fakt, že všetky dáta sú na jednom servery, tzn. sú centralizované, čo vedie ku ľahkej správe servera a zálohovaní systému. [23]

10.3 Tučný klient

Tučný klient je opakom tenkého klienta. Server je zodpovedný za dáta, väčšinou sa jedná o databázový server a spracovanie týchto údajov ma za úlohu klientska časť aplikácie. Toto riešenie sa považuje za zložitejšie, pretože je vyžadovaný kvalitnejší a výkonnejší HW na strane klienta. Napriek tomu toto riešenie je najpoužívanejší typ aplikácie. [23]

Výhodami tohto riešenia sú menšie nároky na server, miestami možnosť pracovať offline, väčšie možnosti multimediálneho prezentovania dát, väčšia flexibilita naprieč platformám, využívanie existujúcich infraštruktúr, zvýšená serverová kapacita. [25]

II. PRAKTICKÁ ČASŤ

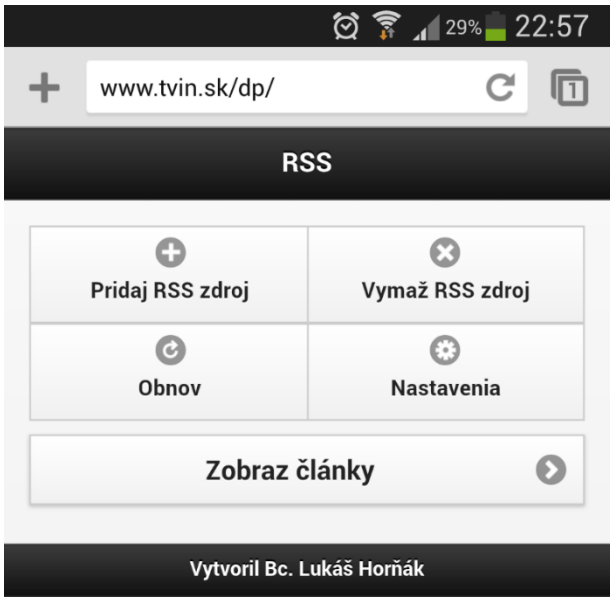
11 MULTIPLATFORMOVÁ WEBOVÁ HTML5 APLIKÁCIA

Pri vývoji multiplatformovej webovej aplikácie som používal známe vývojové prostredie NetBeans IDE verzie 7.3. Táto aplikácia využíva technológie HTML5, JavaScript/jQuery/jQueryMobile a CSS kaskádové štýly pre dizajn aplikácie. Dizajn aplikácie má na starosti framework jQueryMobile, ktorý ponúka CSS štýly priamo prispôsobené layoutu pre mobilné zariadenia.

Skripty obsluhujúce samotnú webovú aplikáciu sú umiestnené v samostatnom súbore a tento súbor je v hlavičke stránky importovaný do projektu.

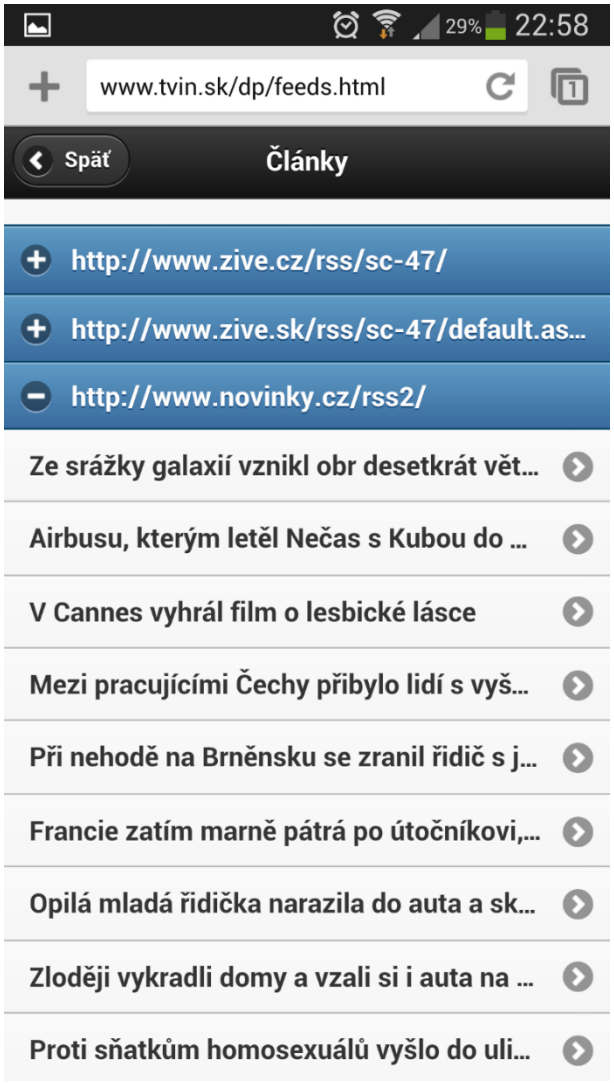
11.1 Usporiadanie webovej aplikácie

Úvodná obrazovka obsahuje tlačidlá na ovládanie celej aplikácie. Je tu možnosť zobraziť články, pridať a vymazať RSS zdroj, obnoviť dáta a nastavenia aplikácie.



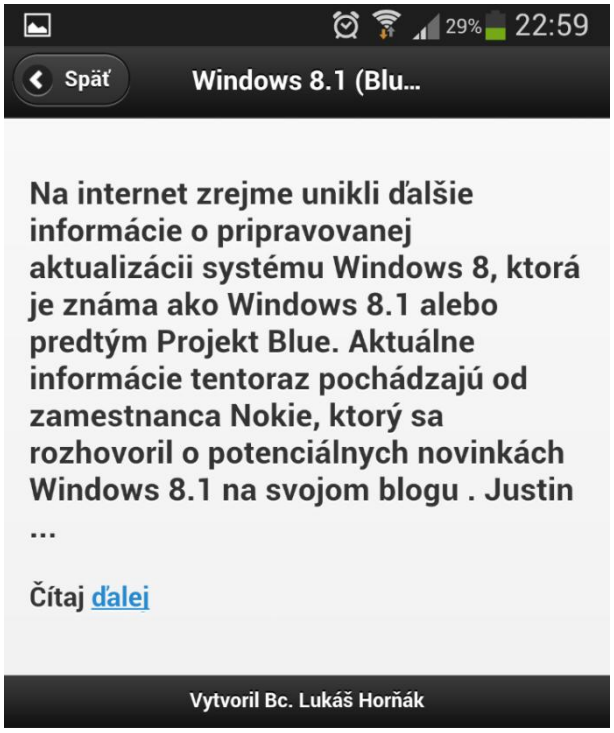
Obrázok 12 Úvodná obrazovka (Web app)

Po kliknutí na *Zobraz články* sa na obrazovke zobrazia RSS články. Načítanie článkov je indikované kruhovým progres barom, ktorý ukazuje užívateľovi, že aplikácia načítava dáta na pozadí.



Obrázok 13 Zobrazenie zoznamu zdrojov a článkov (Web app)

Po kliknutí na článok aplikácie zobrazí obrazovku s názvom a popisom článku s možnosťou prechodu na URL adresu samotného článku, ktorý sa zobrazí v novom okne na mobilnom zariadení. Po stlačení kontextového tlačidla späť, sa vo väčšine prípadoch okno zatvorí a užívateľ pokračuje v práci s aplikáciou kde prestal.

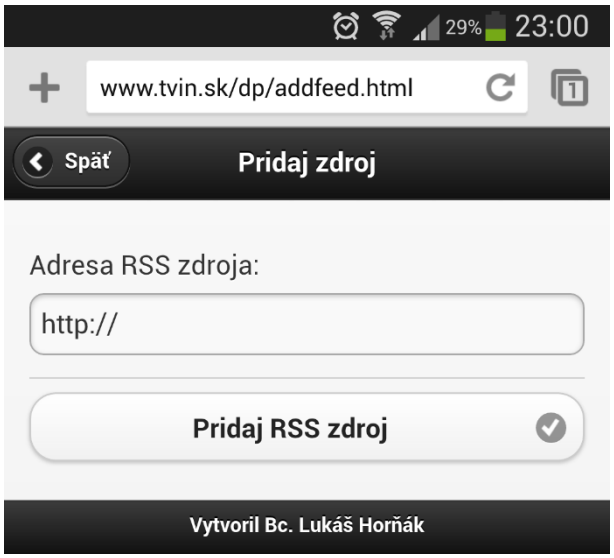


Obrázok 14 Zobrazenie článku (Web app)

Na obrazovke pre vymazanie RSS zdroja sa vypíše zoznam RSS zdrojov a po kliknutí na ľubovoľný zdroj sa tento zdroj vymaže zo zoznamu na serveri. Pri načítavaní zdrojov pre vymazanie, sa taktiež zobrazuje indikátor načítavania.

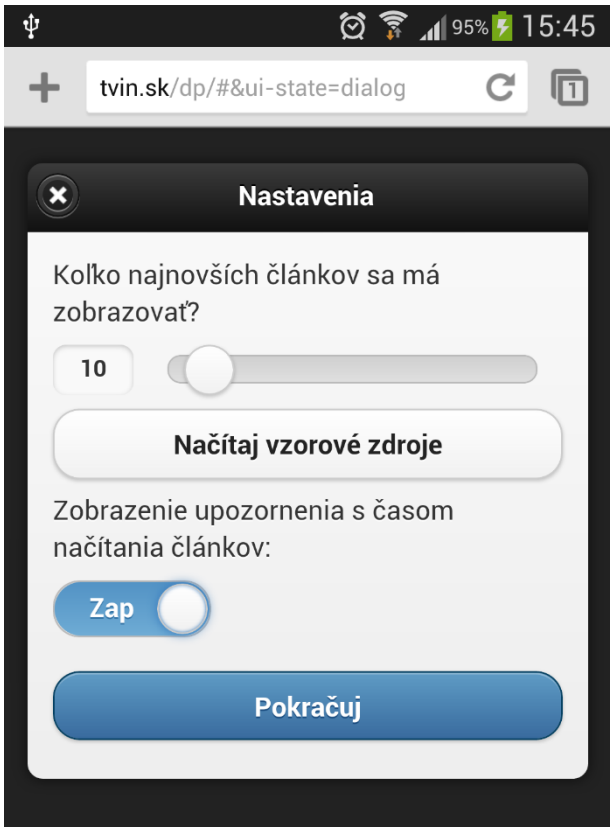
Obrázok 15 Vymazanie RSS zdroja (Web app)

Po zadaní URL adresy a potvrdení, sa požadovaný platný RSS zdroj pridá do zoznamu zdrojov na serveri. Ak je adresa RSS zdroja neplatná, server zdroj nepridá.



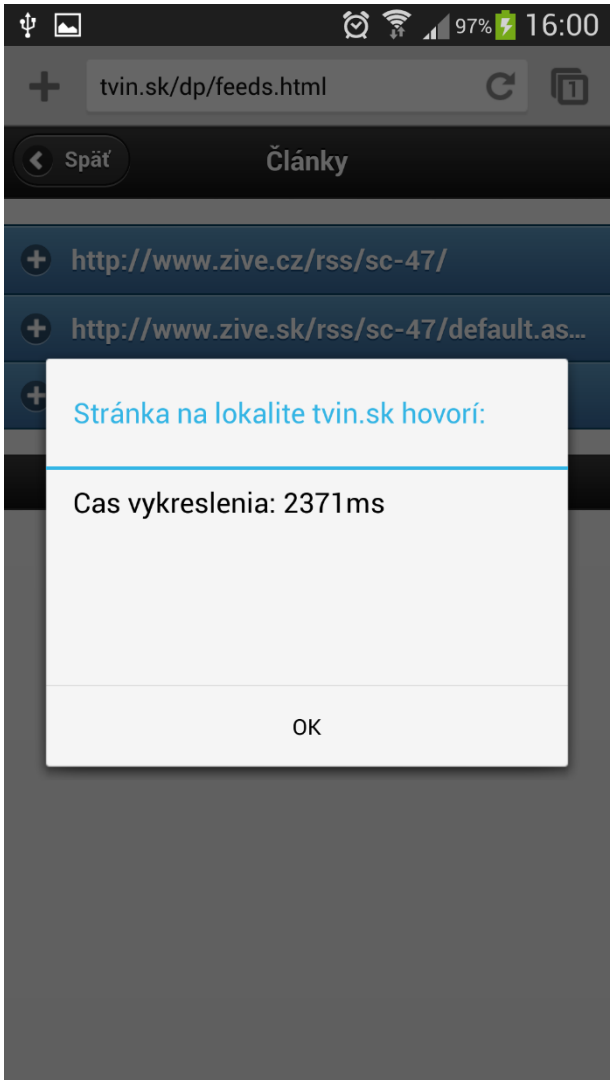
Obrázok 16 Pridanie RSS zdroja (Web app)

Dialógové okno s nastavením webovej aplikácie. Užívateľ má možnosť nastaviť počet nových článkov, ktoré sa majú zobraziť, načítať vzorové RSS zdroje, zapnúť zobrazovanie času vykreslenia článkov v dialógovom okne.



Obrázok 17 Nastavenia aplikácie (Web app)

Ak je funkcia zobrazenia časov zapnutá, dialógové okno zobrazí čas načítania a vykreslenia požadovaných článkov. Čas je udávaný v milisekundách. Časuje sa od kliknutia na zobrazenie článkov po zobrazenie článkov na obrazovku.



Obrázok 18 Zobrazenie času vykreslenia dát (Web app)

11.2 Spojenie webovej aplikácie s webovým serverom

Spojenie aplikácie so serverom je riešené asynchrónnym Ajaxovým volaním typu GET. Spojenie má na starosti funkcia s názvom *connectServlet()*, ktorá ma na starosti celú komunikáciu webovej aplikácie so serverom. Prijaté dáta sú v JSON formáte. Po úspešnom prijatí dát zo serveru, funkcia rozhodne podľa parametru, ktorý je volaný, nasledujúce vykreslenie dát. V prvom prípade sa vykreslí zoznam zdrojov a článkov, v druhom sa

vykreslí zoznam zdrojov pre vymazanie zdroja v poslednom prípade sa len uložia novo načítané dáta do globálnej premennej.

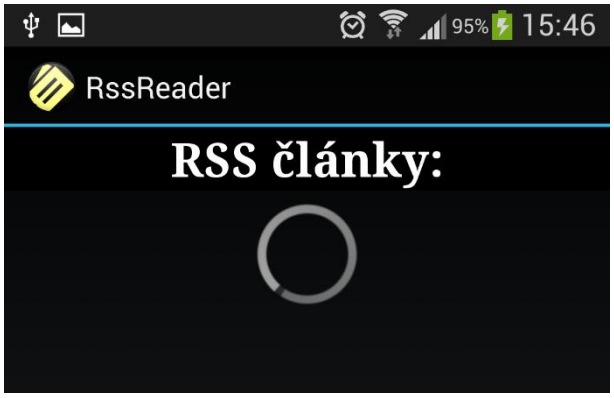
```
function connectServlet(text) {
        setTimeout(function(){
            $.ajax({
                beforeSend: function(){$.mobile.loading('show');},
                type: 'GET',
                url: myUrl,
                async: true,
                dataType: 'jsonp',
                data : {rss : "reach"},
                crossDomain: true,
                contentType: 'application/json',
                success: function (data){
                    data1 = data;
                    if(text == "generate"){
                        generateArticles();
                        $.mobile.loading('hide');
                        if(showTimeAlert){
                            stop = new Date();
                            var time = stop - start;
                            alert("Cas vykreslenia: " + time + "ms");
                        }
                    } else if(text == "delete"){
                        generateDeleteList();
                        $.mobile.loading('hide');
                    } else if(text == "refresh"){
                    //nevykoná sa nič, len sa po úspešnom pripojený
uložia nové dáta
                    }
                },
                error: function(tst, textStatus, errorThrown) {
                alert('Error Message: '+textStatus);
                alert('HTTP Error: '+errorThrown);
                }
            });
        },1);
    }
```

12 NATÍVNA APLIKÁCIA PRE ANDROID OS

Pre programovanie a ladenie androidnej aplikácie som využíval vývojové prostredie Eclipse verzie 8 zahrnuté v balíčku Android SDK os spoločnosti Google pre rýchly a ľahký štart vyvíjania aplikácii pre Android.

12.1 Usporiadanie Android aplikácie

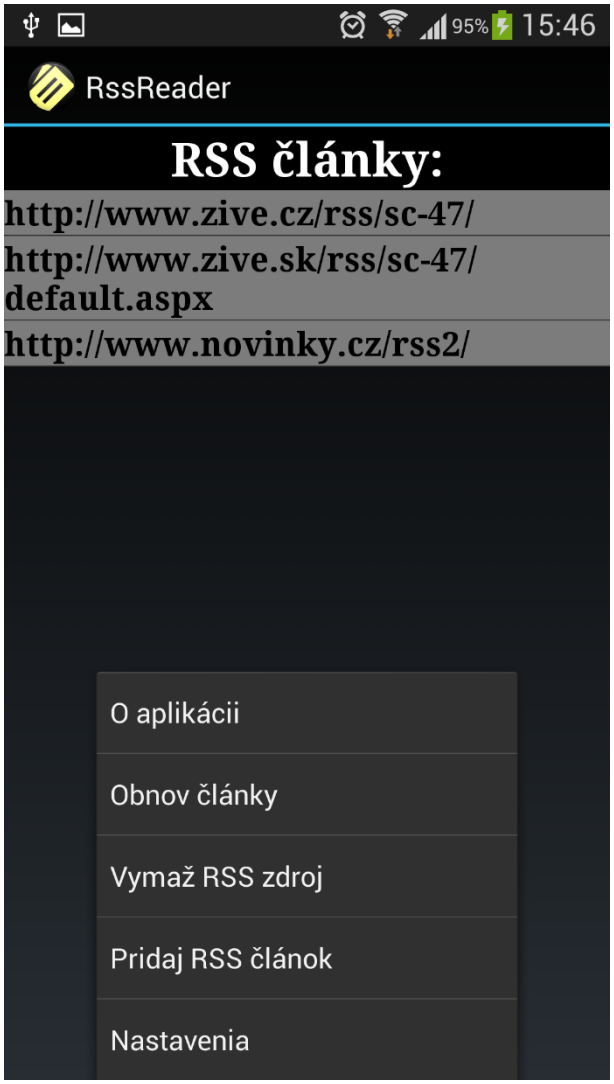
Pri spustení Android aplikácie sa zobrazí hlavná obrazovka aplikácie. Pri každom načítaní dát zo servera sa zobrazuje progres bar.



Obrázok 19 Úvodná obrazovka pred načítaním dát (Android)

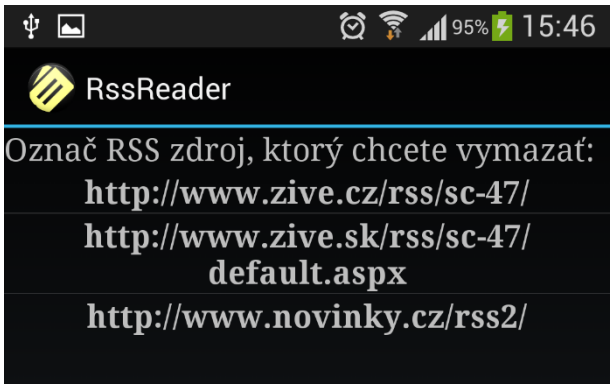
Po načítaní dát zo servera sa zobrazí zoznam RSS zdrojov a článkov. Zoznam zdrojov je riešený pomocou roztváracieho zoznamu. Po kliknutí na zdroj, sa roztvorí zoznam článkov daného zdroja:

Po stlačení kontextového tlačidla na mobilnom telefóne s Android OS, sa zobrazí kontextové menu aplikácie. Sú tu možnosti na pridanie a vymazanie RSS zdroja, obnovenie dát, nastavenie aplikácie a zobrazenie informácii o aplikácii.



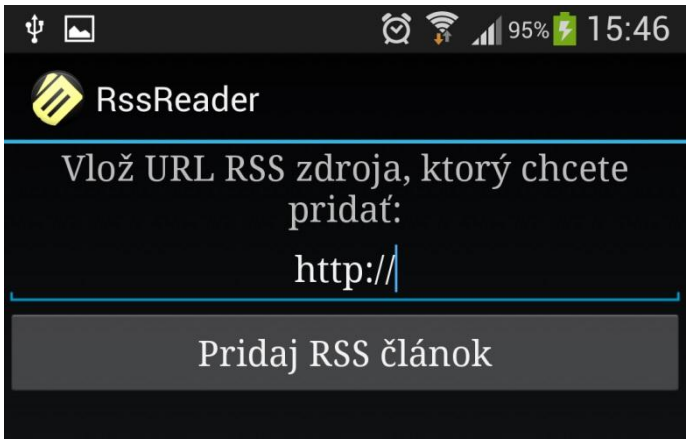
Obrázok 20 Úvodná obrazovka so zobrazením zoznamu zdrojov a článkov (Android)

Na obrazovke pre zmazanie RSS zdroja sa zobrazí zoznam aktuálnych RSS zdrojov. Po kliknutí na zdroj sa zvolený zdroj vymaže zo zoznamu RSS zdrojov.



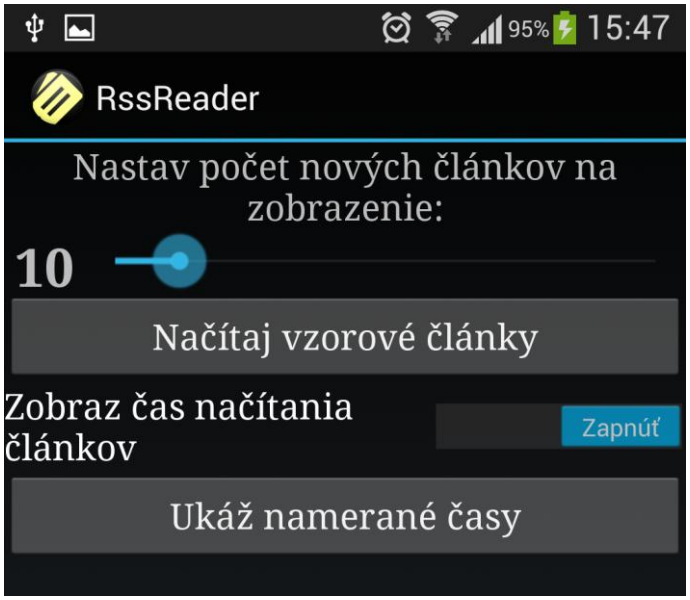
Obrázok 21 Vymazanie RSS zdroja (Android)

Pre pridanie RSS zdroja, slúži táto obrazovka. Po zadaní URL adresy platného zdroja, sa tento zdroj pridá do zoznam RSS zdrojov v aplikácii. Ak je adresa zdroja neplatná, zdroj sa nepridá a aplikácia pokračuje v chode. Aplikácia predvyplní políčko textom *http://*, pre jednoduchšie zadávanie celej adresy zdroja.



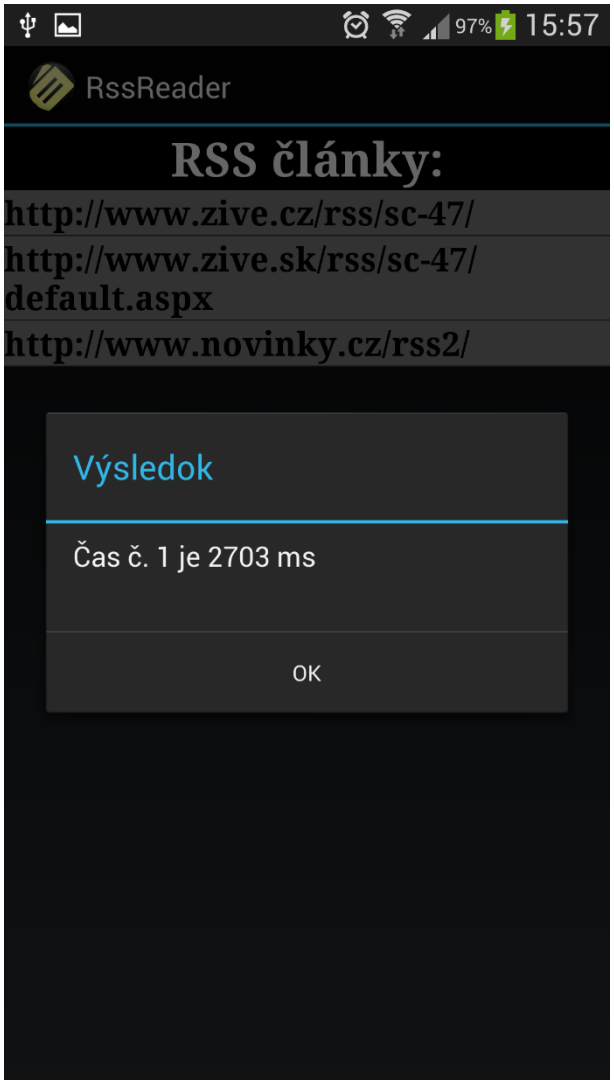
Obrázok 22 Pridanie RSS zdroja (Android)

V nastaveniach aplikácie je možnosť zmeniť maximálny počet najnovších článkov, ktoré má aplikácia zobraziť. Volenie požadovanej hodnoty je riešené posúvnikom, ktorý nastavuje hodnoty v rozmedzí 5 až 50 článkov. Po nastavení sa zobrazí malé upozornenie, že požadovaná hodnota bola uložená. Ďalej je tu možnosť načítať vzorové články, zapnúť zobrazovanie časov načítania článkov a na rozdiel od webovej aplikácie je tu možnosť zobraziť všetky namerané časy v dialógovom okne.



Obrázok 23 Nastavenie aplikácie (Android)

Ak je funkcia pre zobrazenie časov vykreslenia článkov zapnutá, aplikácia vykreslí nameraný čas v dialógovom okne v milisekundách. Ak užívateľ požaduje zobraziť všetky namerané časy, zobrazia sa mu tiež v dialógovom okne.



Obrázok 24 Zobrazenie času vykreslenia dát (Android)

12.2 Spojenie Android aplikácie so serverom.

Pre komunikáciu Android aplikácie so serverom bola na tento účel vytvorená samostatná trieda s názvom *ConnectServlet.java*. Táto trieda má na starosti spojenie so serverom a následné. Trieda obsahuje tri metódy. Prvou z nich je *OpenHttpGETConnectio()*:

```
private static InputStream OpenHttpGETConnection(String url){
            InputStream inputStream = null;
             try {
```

```
                    HttpClient httpclient = new DefaultHttpClient();
                    HttpResponse httpResponse = httpclient.execute(new
HttpGet(url));
                    inputStream = httpResponse.getEntity().getContent();
            } catch (Exception e) {
                  Log.d("InpustStream",e.getLocalizedMessage());
            }
            return inputStream;
      }
```

Táto metóda sa pomocou spojenia typu GET spojí so serverom, ktorý jej následne vráti požadované dáta vo formáte JSON v objekte inputStream. Tento objekt je potrebné previesť do textovej premennej. Pre tento účel je tu druhá metóda s názvom *DownloadData()*:

```
public String DownloadData(String URL){
            InputStream in = null;
            try{
                  in = OpenHttpGETConnection(URL);
            } catch (Exception e){
                  Log.d("DownloadData", e.getLocalizedMessage());
                  return "";
            }

            String jsonData = "";
            try {
                  jsonData = IOUtils.toString(in);
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            return jsonData;
      }
```

Táto metóda sa pripojí na server pomocou prvej spomenutej metódy *OpenHttpGETConnectio()*, a následne získaný objekt typu *InputStream* prevedie do textovej premennej. Pre tento účel sa využíva externá knižnica *Commons IO* od firmy Apache, ktorá pomocou funkcie *IOUtils.toString()* prevedie *InputStream* do *String* premennej. Tento prevod je veľmi efektívny než prechádzať prijatý objekt po znakoch a ukladať ho do textovej premennej.

13 JAVA SERVER

Server beží v rámci localhostu na notebooku, využíva Java server Apache Tomcat verzie 7.0.34. Pre potreby testovania a ukážky aplikácii je využitá aj cloudová služba typu PaaS OpenShift, takže na server sa dá dostať aj mimo localhostu. Načítanie a parsovanie RSS zdrojov

Na spracovanie jednotlivých RSS zdrojov a článkov je v mojom projekte vytvorená trieda s názvom *RssCore.java*. Metódou *getFeedObject()* trieda vracia objekt typu *RssChannel*, ktorým je vlastne vytvorená trieda s požadovanou štruktúrou pre uloženie dát. Táto trieda obsahuje i ďalšie menej dôležité metódy, ktoré sú dostupné v zdrojových kódoch na priloženom CD.

Trieda *RssCore.java* si priebežne ukladá zoznam zdrojov do súboru. Uložené dáta sú využitím knižnice GSON serializované do typu JSON.

Samotné spracovanie RSS zdrojov je riešené využitím externej knižnice feed4j[7], ktorá sa stará o spracovanie zdroja.

13.1 Obsluha požiadavky GET serverom

Pre obsluhu požiadavky GET má Java Server vlastnú metódu *doGet()*, ktorá prijíma v parametre metódy objekt pre požiadavku a objekt pre odpoveď servera. Pomocou objektu požiadavky server vie čo má vykonať a pomocou objektu odpovede dokáže server reagovať na požiadavku od klienta.

Obsluha požiadavky na serveri je riešená nasledujúcim spôsobom:

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {
    String clbck = "", output = "", feedOut;
    rss = new RssCore();
    File f = new File("file.json");
    if(!f.exists()) {
        rss.RssAdd("http://www.zive.cz/rss/sc-47/");
        rss.RssAdd("http://www.zive.sk/rss/sc-47/default.aspx");
```

[7] Feed4j. *Sauron Software.* [online]. 2007 [cit. 2013-05-30]. Dostupné z: http://www.sauronsoftware.it/projects/feed4j/

```
        rss.RssAdd("http://www.novinky.cz/rss2/");
    }
    String parRss = request.getParameter("rss").toString();
    if(new String("reach").equals(parRss)){
        rss.RssRead();
        feedOut = new Gson().toJson(rss.getFeedObject());
        clbck = request.getParameter("callback");
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        output = clbck + "(" + feedOut + ");";
        response.getWriter().write(output);
    } else if(new String("reachand").equals(parRss)){
        rss.RssRead();
        feedOut = new Gson().toJson(rss.getFeedObject());
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        response.getWriter().write(feedOut);
    } else if(new String("add").equals(parRss)){
        rss.RssAdd(request.getParameter("link").toString());
    } else if(new String("delete").equals(parRss)){
        rss.RssDelete(request.getParameter("link").toString());
    } else if(new String("defaults").equals(parRss)){
        rss.DeleteFile();
    } else if(new String("test").equals(parRss)){
        TimmingTest test1 = new TimmingTest();
        test1.startTimingTest(response);
    }
}
```

Na začiatku metódy testujem existenciu potrebného súboru so zoznamom zdrojov. Ak súbor neexistuje, do inštancie *rss* sú pomocou metódy *RssAdd* pridané vzorové zdroje.

Následne pomocou funkcie *getParameter()* zisťujem parameter požiadavky. Na základe hodnoty tohto parametra rozhodujem, akú činnosť server vykoná. Server rozlišuje odkiaľ prišla požiadavka. Ak požiadavka prišla z webovej aplikácie, načíta sa navyše hodnota parametra *callback* a táto hodnota sa spolu s JSON dátami pošle naspäť klientskej aplikácii. Ak prišla požiadavka z Android zariadenia, pošlú sa späť len JSON dáta.

Server obsluhuje aj iné metódy triedy *RssCore*, a to metódy na pridanie alebo odstránenie zdroja. Tieto metódy prijímajú parameter *link*, ktorý poukazuje na zdroj, ktorý sa má pridať alebo zmazať.

Poslednou obsluhou servera je spustenie testovacej sekvencie servera, ktorá vykoná časové a dátové testy na serveri a výsledky zapíše do súborov na serveri.

14 ZROVNÁVACIE TESTY VÝKONU APLIKÁCII

Pre zrovnanie výkonu oboch aplikácii som zvolil dve typy meraní. Časové a pamäťové merania. Pre porovnanie výkonu som zvolil aj testy v rámci Java Servera, kde som tiež meral veľkosť odoslaných dát a čas, za ktorý boli spracované.

Metodika testovania je nasledovná. Testy sa budú opakovať 30 krát po sebe, pre rôzny počet spracovaných RSS zdrojov. Počet spracovaných RSS zdrojov bude 1 RSS zdroj, 10 RSS zdrojov a 100 RSS zdrojov. Výsledné merania vykreslím do grafov.

Počet článkov, ktoré sa budú na serveri spracovávať, nebol obmedzovaný, takže sa bude spracovávať toľko článkov, koľko každý RSS zdroj obsahuje.

Získané časové výsledky budú uvádzane v milisekundách a výsledky konzumácie dát zo servera a veľkosti odoslaných v kilobajtoch.

Testovanie serverovej časti bude prebiehať v rámci localhosta na vlastnom notebooku. Testovanie oboch klientskych aplikácii bude prebiehať na rovnakom mobilnom zariadení s Android OS, pripojený do lokálnej siete pomocou bezdrôtového pripojenia

Jednotlivé merania sú ovplyvnené výkonom notebooku, na ktorom beží Java Server a odozvou bezdrôtového pripojenia mobilného telefónu do lokálnej siete s notebookom, na ktorom bude bežať server.

14.1 Časové testy klientskych aplikácii

14.1.1 Spôsob testovania rýchlosti klientskych aplikácii

Testovanie oboch klientskych aplikácii bolo rovnaké. Testoval sa čas od spustenia načítavania dát zo servera po vykreslenie dát samotnou aplikáciou.

14.1.1.1 Android aplikácia

Pre časové testovania rýchlosti vykreslenia prijatých dát na Android aplikácii som vytvoril triedu *Timming.java*. Táto trieda obsahuje tri hlavné metódy pre časové testovanie aplikácie.

```
public void startTimer(){
      startTime = System.currentTimeMillis();
      running = true;
}
```

```
public void stopTimer(Context context){
      stopTime = System.currentTimeMillis();
      running = false;
      measureData.add(String.valueOf(getTime()));
    countOfTests++;
}
public long getTime(){
      long elapsed;
    if (running) {
      elapsed = ((System.currentTimeMillis() - startTime));
    } else {
      elapsed = ((stopTime - startTime));
    }
    return elapsed;
}
```

Trieda obsahuje aj iné metódy na zobrazenie výsledkov pomocou dialógov, ale sú už obsiahlejšie a zdrojové kódy aplikácii sú umiestnené na priloženom CD.

Výsledky boli po 30tich opakovaniach zobrazené v dialógovom okne a zapísané do hárku v Exceli, kde boli následne spracované.

14.1.1.2 Webová aplikácia

Vo webovej aplikácii testujem výsledný čas podobným spôsobom. Skript bol napísaný v jazyku *Javascript.* Zapnutie časovača:

```
$( document ).delegate("#feeds", "pageinit", function() {
    if(showTimeAlert){
        start = new Date();
    }
    connectServlet("generate");
});
```

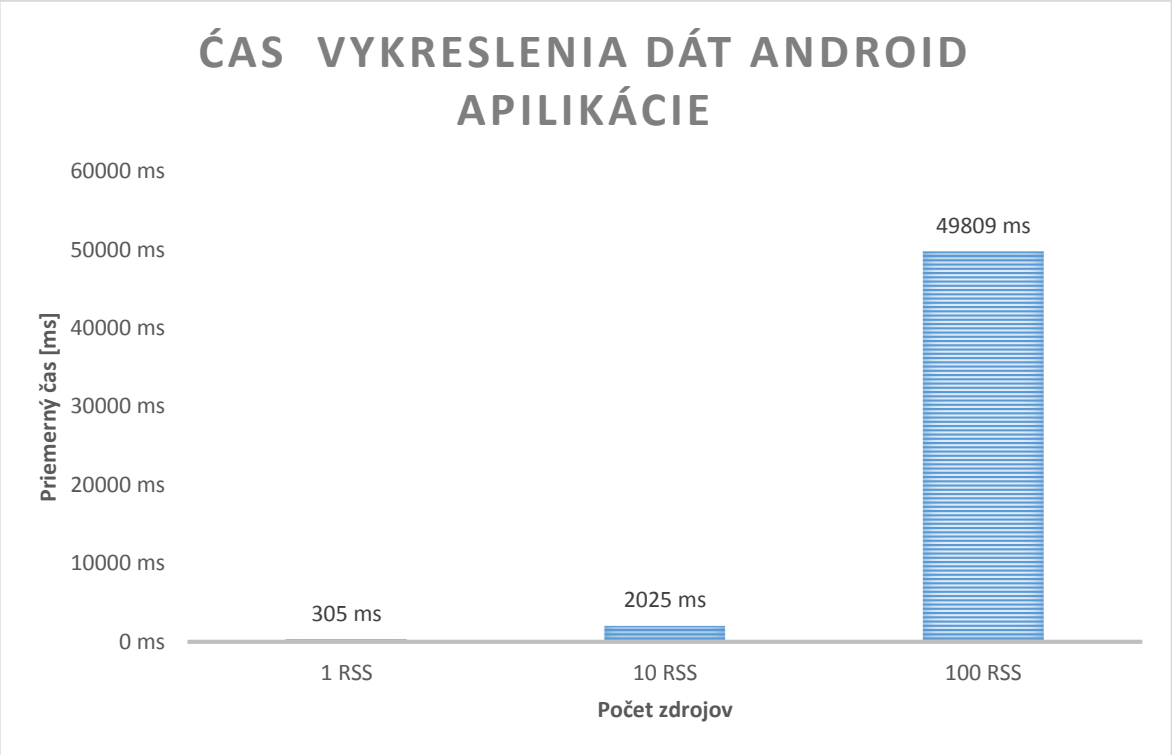
Stopnutie časovača a zobrazenie času pomocou alertu:

```
if(showTimeAlert){
    stop = new Date();
    var time = stop - start;
    alert("Čas vykreslenia: " + time + "ms");
}
```

Výsledky boli zakaždým po zobrazení zapísané do hárku v Exceli, kde boli následne spracované.

14.1.2 Natívna Android aplikácia

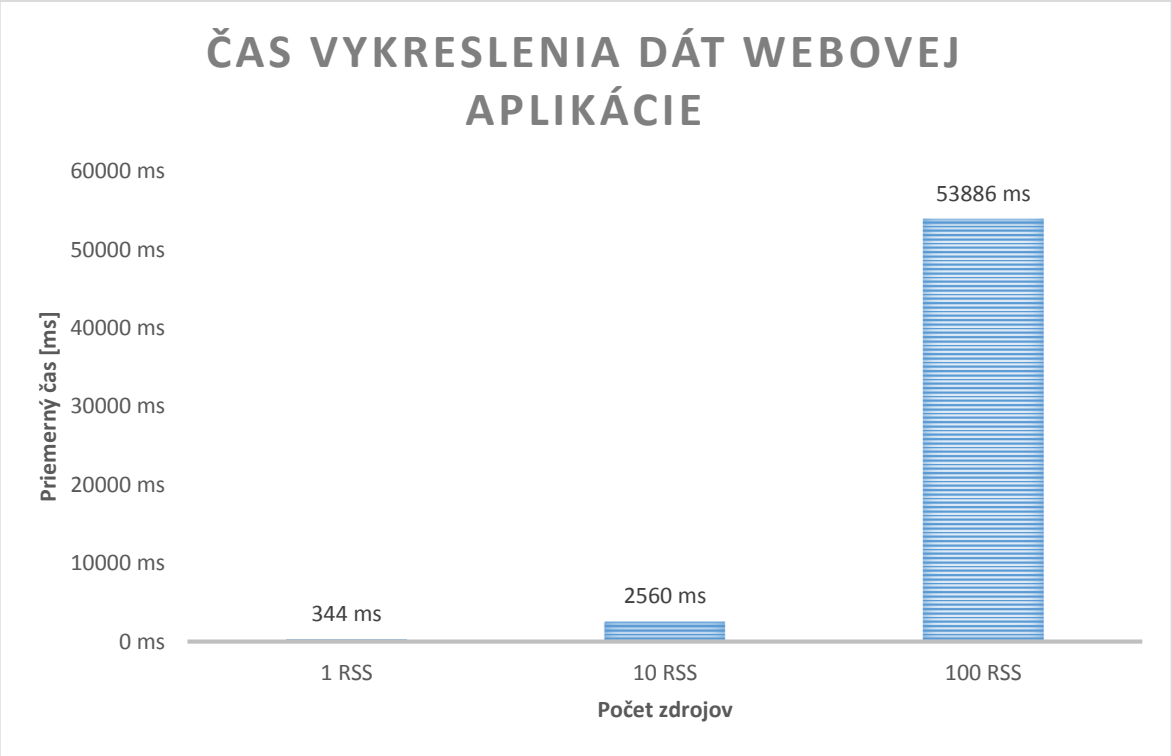
V natívnej Android aplikácie som čas meral od začiatku načítavania dát zo servera po zobrazenie dát na displej zariadenia. Pri prvom spustení aplikácie sa dáta hneď načítavajú zo servera a aj po následnom obnovení dát z kontextového menu. Časovač sa zastaví akonáhle sa dáta zobrazia a aplikácia zobrazí výsledok.



Graf 1 Zobrazenie času vykreslenia dát Android aplikácie

14.1.3 Multiplatformová webová aplikácia

Meranie rýchlosti natívnej aplikácie prebiehalo podobne ako v Android aplikácii avšak spustenie časového testu začínalo po stlačení tlačidla *Zobraz články* a zastavenie časovača po načítaní a zobrazení dát na displej zariadení.



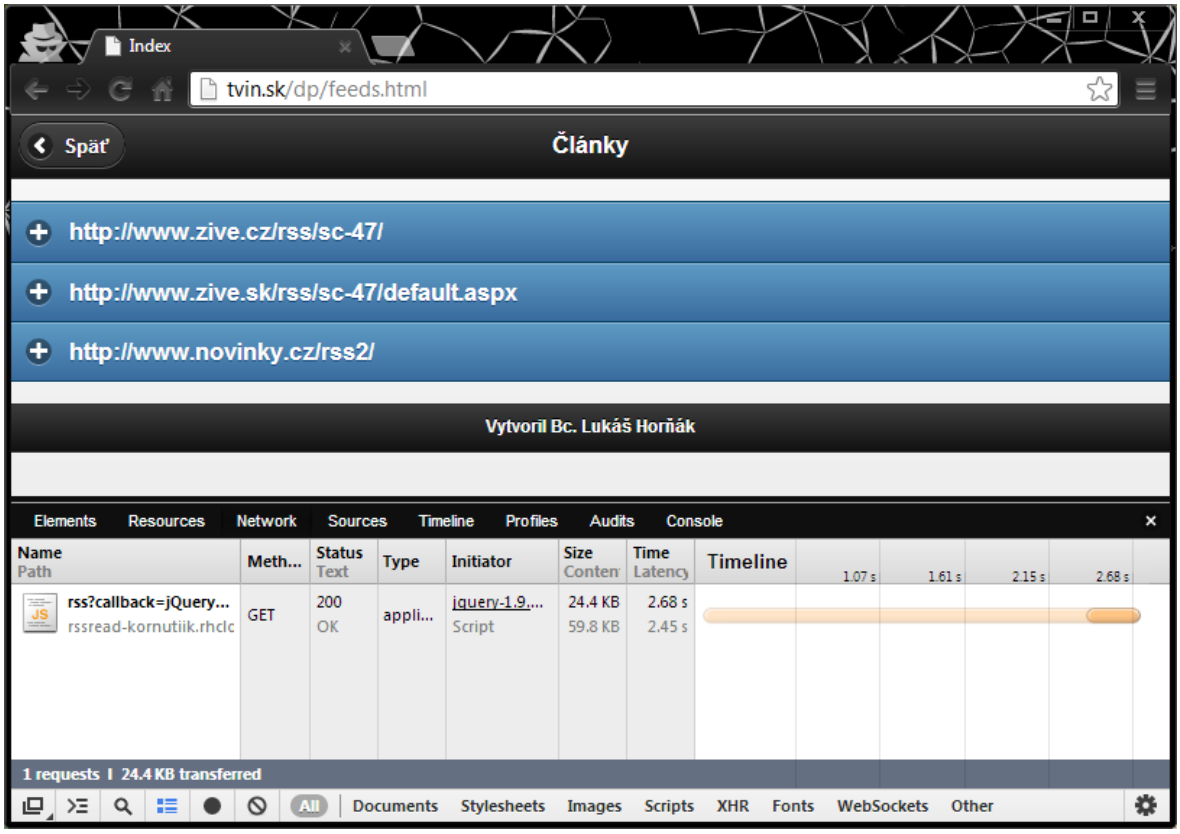
Graf 2 Zobrazenie času vykreslenia dát webovej aplikácie

14.2 Testy výkonu servera

Pre porovnanie výkonu oboch klientskych aplikácii s výkonom servera som vykonal testy aj na serveri. Meral som veľkosti odosielaných dát a čas, za ktorý server spracuje požiadavku na spracovanie zdroja.

Dátové testy boli vykonané len v rámci servera, pretože odosielané dáta sa zhodujú s prijatými dátami na klientskych aplikáciách. Odosielané dáta obsahujú hlavičku odozvy servera a samotné posielané dáta.

Skutočnosť, že odosielané dáta sa zhodujú s dátami som overil pomocou doplnku DevTools vo webovom prehliadači Google Chrome.



Obrázok 25 Ukážka overovania prijatých dát doplnkom DevTools

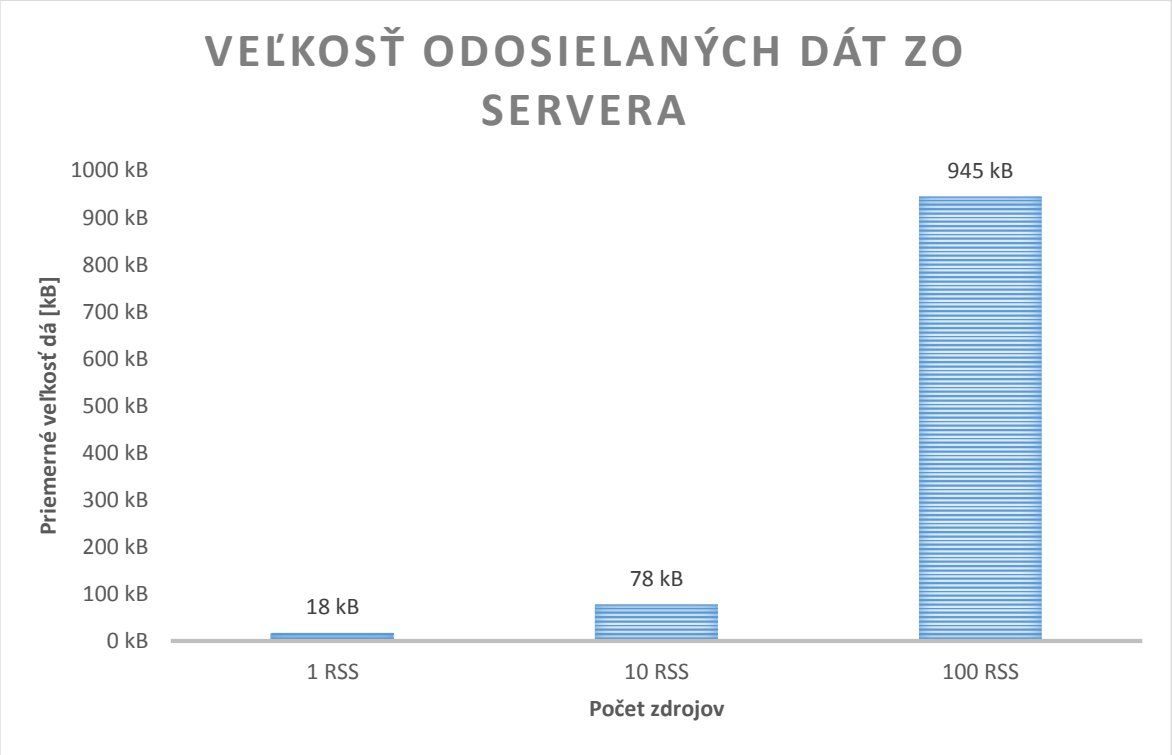
Pre samotné testovanie servera, som vytvoril triedu na to určenú s názvom *TimmingTest.java*. Táto trieda automaticky opakuje testovanie čas a dát 30 krát po sebe pre 1, 10 a 100 RSS zdrojov.

```
public void startTimingTest(HttpServletResponse resp) throws IOException{
    for(int i = 0;i < countOfRepeating;i++){ //30x 1RSS
        startStopwatch();
        rssFeedLoader = new RssCore();
        response = resp;
        rssFeedLoader.setCountOfRssSources(1);
        rssFeedLoader.RssRead();
        feedOut = new Gson().toJson(rssFeedLoader.getFeedObject());
        stopStopwatch(1);
        measureData.add(String.valueOf(feedOut.getBytes().length));
        System.out.println("Test 1RSS number: " + i);
    }
    for(int i = 0;i < countOfRepeating;i++){ //30x 10RSS
        startStopwatch();
        rssFeedLoader = new RssCore();
        response = resp;
```

```
        rssFeedLoader.setCountOfRssSources(10);
        rssFeedLoader.RssRead();
        feedOut = new Gson().toJson(rssFeedLoader.getFeedObject());
        stopStopwatch(10);
        measureData.add(String.valueOf(feedOut.getBytes().length));
        System.out.println("Test 10RSS number: " + i);
    }
    for(int i = 0;i < countOfRepeating;i++){ //30x 100RSS
        startStopwatch();
        rssFeedLoader = new RssCore();
        response = resp;
        rssFeedLoader.setCountOfRssSources(100);
        rssFeedLoader.RssRead();
        feedOut = new Gson().toJson(rssFeedLoader.getFeedObject());
        stopStopwatch(100);
        measureData.add(String.valueOf(feedOut.getBytes().length));
        System.out.println("Test 100RSS number: " + i);
    }
    response.getWriter().write("TESTING END");
    FileWriter fstreamdata = new FileWriter("data1.txt");
    BufferedWriter out1 = new BufferedWriter(fstreamdata);
    for(int i = 0; i < measureData.size();i++){
        out1.write(measureData.get(i) + ";");
    }
    out1.close();
    writeToFile();
}
```

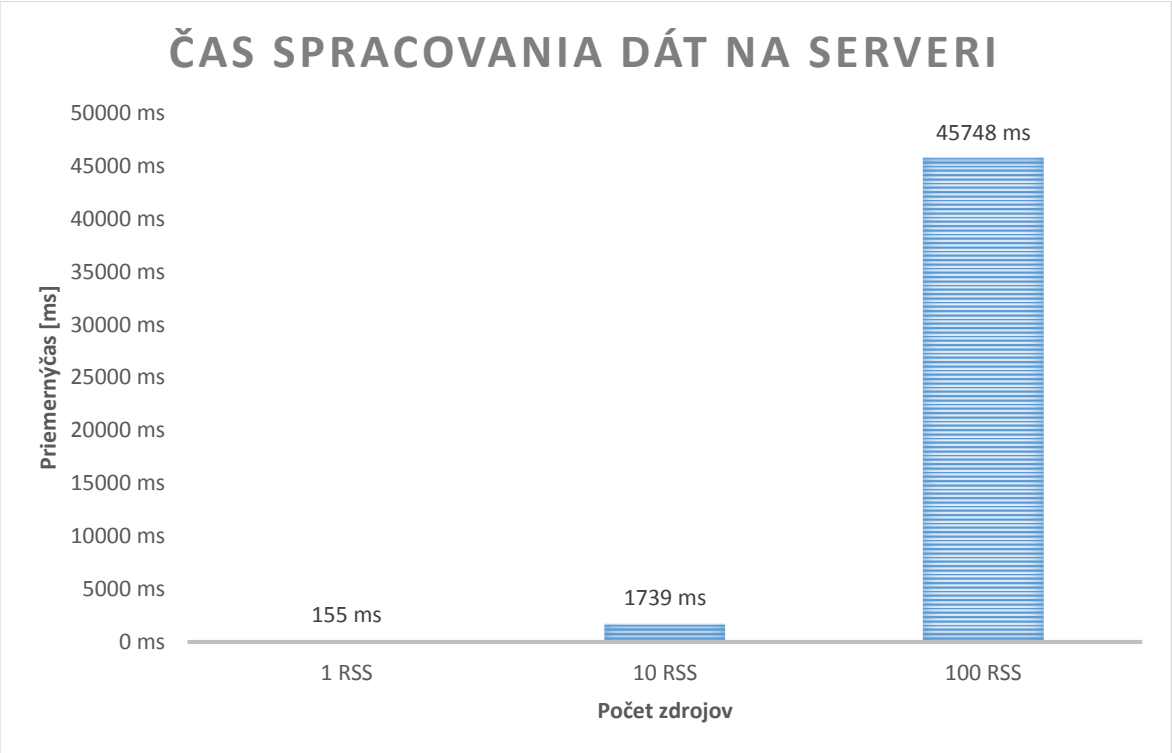
Namerané dáta sa na konci testovania zapíšu do textových súborov umiestnené v koreňovom adresári Java servera.

14.2.1 Test veľkosti odoslaných dát



Graf 3 Veľkosť odosielaných dát zo servera pre klientské aplikácie

14.2.2 Test časov spracovania dát na serveri



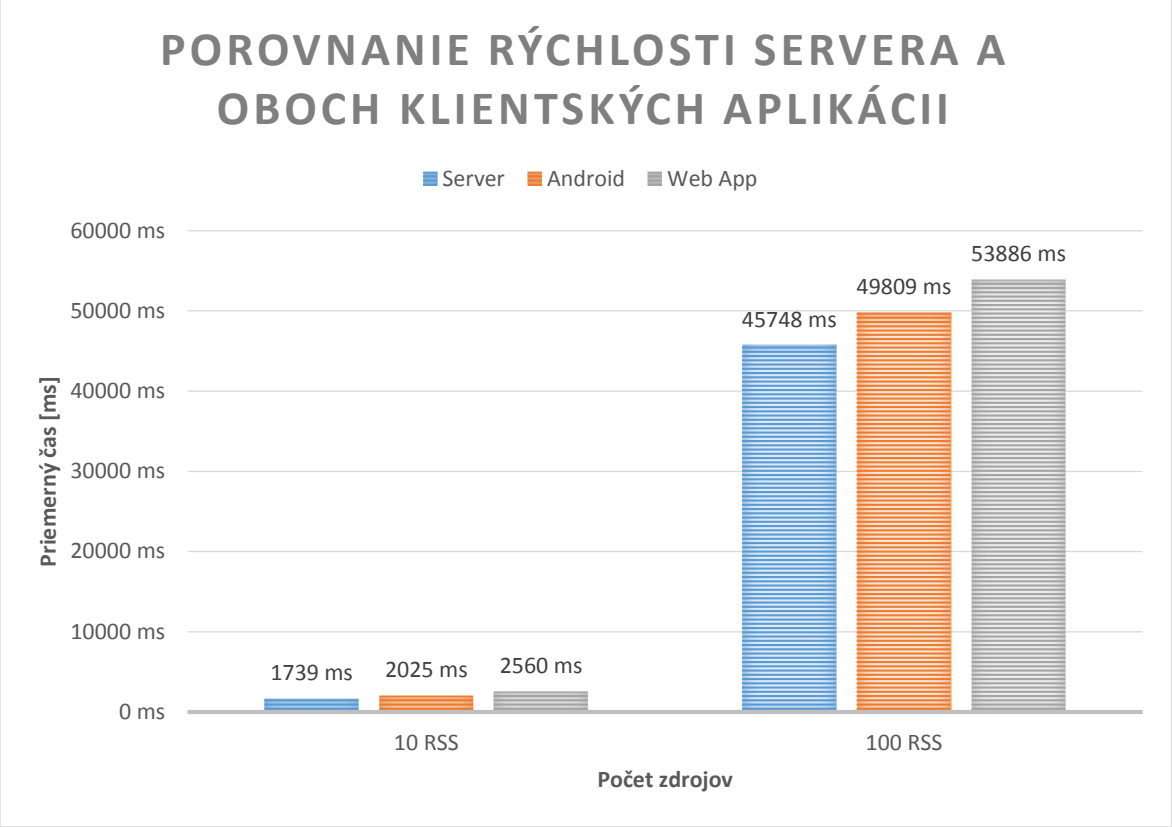
Graf 4 Ubehnutý čas na serveri od požiadavky po spracovanie dát

14.3 Zhrnutie výsledkov testovania

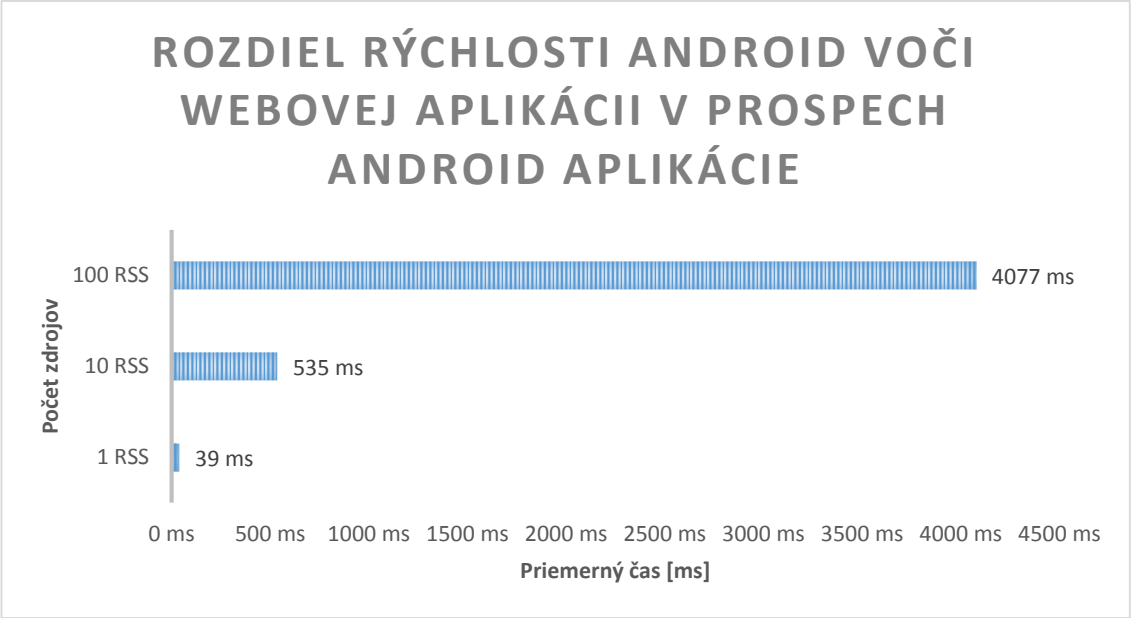
Dátové a časové testy boli vykonané v rámci serverovej strany. V prípade klientskych aplikácii boli vykonané časové testy rýchlosti vykreslenia dát.

Časy v rámci servera ukazujú, za aký čas server spracuje dáta od požiadavky po výstup dát na odoslanie pre klientske aplikácie. Avšak časy rýchlostí klientskych aplikácii zahrňujú jak čas spracovania servera, čas doručenia dát bezdrôtovou LAN sieťou a čas, za ktorý sa prijaté dáta zobrazia na mobilnom zariadení.

Na grafe č.6 môžeme pozorovať, že server je očakávane najrýchlejší, pretože má za úlohu len spracovanie dát. Na tomto grafe je možné vidieť, že webová aplikácia je pomalšia než Android aplikácia.



Graf 5 Porovnanie časov spracovania na serveri a časov vykreslenia na klietnských aplikáciách



Graf 6 Porovnanie rýchlostí klientských aplikácii. Rozdiel uvádzaný v prospech android aplikácie

Z grafu je vidno, že Android aplikácia je pri 100 zdrojoch rýchlejšia oproti mobilnej aplikácii v priemere o 4 sekundy. Pri menšom počte spracovaných a vykreslených zdrojov sú rozdiely zanedbateľné, avšak pri narastajúcom počte spracovaných zdrojov sa rozdiel rýchlosti spracovania markantne navýši v prospech Android aplikácie.

15 ASPEKTY KLIENTSKÝCH APLIKÁCII

Obidve klientske aplikácie sú typu tenký klient. To znamená, že aplikácie sa starajú len o vykresľovanie dát na displej mobilného zariadenia, v prípade HTML5 mobilnej aplikácie aj na obrazovku počítača. Pri vývoji oboch aplikácii som sa stretol s rôznymi problémami a čas potrebný pre vývoj aplikácii bol rôzny.

15.1 Webová aplikácia

Vývoj webovej aplikácie bol oproti vývoju natívnej Android aplikácie jednoduchší. Čas potrebný pre vývoj klientskej aplikácie v porovnaní s Android aplikáciou bol v mojom prípade oveľa kratší a z časti jednoduchší. Je to vďaka programovacím jazykom, ktoré ovláda bežný webový vývojár.

Výhodou webovej aplikácie je univerzálnosť naprieč platformami. Aplikáciu je možné spustiť na veľkej väčšine mobilných zariadeniach. Jedinými podmienkami chodu aplikácie je nutnosť webového prehliadača na zariadení a internetové spojenie v prípade klientskych aplikácii.

Nevýhodou webovej aplikácie vidím v nemožnosti inštalácie aplikácie na mobilné zariadenia, pretože aplikácia je dostupná z webového hostingu po zadaní url adresy. Avšak túto nevýhodu plne rieši PhoneGap, ktorý zaobalí vytvorenú aplikáciu do inštalačného súboru, v našom prípade Androida do **.apk* balíčka, ktorého inštalácia na Androidnom zariadení je ľahkou záležitosťou. Vďaka inštalačnému balíčku nie je následne problém umiestniť na nejaký mobilný market, prípade Androida do Google Play a nechať si za inštaláciu aplikácie aj platiť.

Ak vývojár z nejakých dôvodov nechce použiť PhoneGap na zabalenie aplikácie do inštalovateľného balíčka, užívateľ mobilného zariadenia má možnosť na mobilných telefónoch vytvoriť odkaz stránky na pracovnej ploche telefónu. V prípade operačného systému iOS na mobilných telefónoch značky Apple je možné tento odkaz nastaviť tak, že sa skryje riadok s url adresou aplikácie a užívateľ má pocit, že pracuje s plnohodnotnou aplikáciou.

15.2 Natívna Android aplikácia

Vývoj natívnej Android aplikácie bol pre mňa náročnejší aj napriek tomu, že som sa už tejto platforme v minulosti venoval. Je to spôsobené komplikovanejším programovacím

jazykom s viacerými možnosťami ako jazyka tak aj platformy. Natívna aplikácia beží na mobilnom zariadení oveľa plynulejšie, funkčnosť aplikácie je takmer neobmedzená, obmedzenie existuje len vo verzii použitého API pri vývoji aplikácie. Natívna platforma ponúka aj lepšie možnosti dizajnu a navigácie v aplikácii, použitím rôznych widgetov či prispôsobením rozhrania aj pre tablet zariadenie.

Výhodou pri zložitých natívnych aplikáciách oproti mobilným je rýchlosť behu samotnej aplikácie, spracovania dát v rámci aplikácie a možnosť vyvíjať aj zložité hry.

Oproti webovej aplikácii som zaznamenal výhodu ľahkej lokalizácie aplikácie do viacerých jazykov, ktorá sa prepína na základe nastaveného jazyka na mobilnom zariadení Android.

Nevýhoda natívnej aplikácie je vývoj len pre konkrétnu platformu. Každá platforma používa iný programovací jazyk, takže vývojár musí disponovať znalosťami aj iných jazykov, ak chce vyvíjať aplikácie pre iné mobilné platformy ako pre tú, ktorú vyvíja.

15.2.1 Google Play

Keďže pri vývoji natívnej aplikácie vzniká rovno inštalovateľný balíček, nie je problém umiestniť aplikáciu do marketu. V prípade Androidu ide o market s názvom Google Play. Možnosť umiestniť aplikáciu na tento market predpokladá vlastnenie developerského účtu na Google Play. Tento účet je spoplatnený jednorazovým poplatkom vo výške 25$. Týmto poplatkom si spoločnosť Google sľubuje vyššiu kvalitu aplikácii na Google Play (tj. menej spamových aplikácii)

ZÁVER

Cieľom tejto práce bolo vytvoriť rešerš na tému tvorby natívnych aplikácii pre operačný systém Android ako aj multiplatformových aplikácii v jazyku HTML5, vytvoriť porovnanie výhod a nevýhod klient-server aplikácii a uviesť ich špecifiká, naprogramovať testovací klient-server aplikáciu natívne pre Android OS a pomocou HTML5 a porovnať aspekty týchto aplikácii. Na vytvorených aplikáciách vykonať porovnávajúce testy výkonu aplikácii.

Teoretická časť práce popisuje technológie používané pri vývoji aplikácii pre OS Android a tiež pomocou HTML5, porovnanie výhod a nevýhod natívnych a multiplatformových aplikácii, popisuje podrobnejšie prístup pri tvorbe klient-server aplikácii.

Praktická časť práce popisuje obidve vytvorené aplikácie, približuje používateľské prostredie aplikácií, poukazuje na úryvky zdrojového kódu pri spojeniach aplikácii so serverom a iné, zaujímavé kúsky zdrojového kódu. V tejto časti sú ďalej zahrnuté porovnávajúce testy výkonu oboch vytvorených aplikácii ako aj serverovej strany, výsledky sú znázornené grafmi. Na konci praktickej časti je porovnanie aspektov samotnej publikácie oboch aplikácii.

Všetky zadania diplomovej práce sa mi podarilo spustiť, natívna a webová aplikácia je plne funkčná, ako aj serverová časť. Na základe výsledkov nameraných na oboch vytvorených aplikáciách a na serveri je jasné, že natívna aplikácia oproti multiplatformovej je na rovnakom mobilnom zariadení plynulejšia a rýchlejšia avšak vývoj natívnej aplikácie je zložitejší a zaberie viac času.

Príloha tejto práce je CD, ktoré obsahuje zdrojové kódy oboch klientskych aplikácii ako aj kódy serverovej strany. CD dopĺňa Excel dokument, s nameranými hodnotami testov aplikácii a servera.

ZÁVER V ANGLIČTINE

The aim of this work was to create a research on the topic of native applications for the Android OS as well as cross-platform applications in HTML5, create a comparison of the advantages and disadvantages of client-server application and specify their specifications, create testing client-server application for native Android OS and using HTML5 and compare these aspects of the applications. On created applications perform comparative benchmark tests.

The theoretical part describes the technology used to develop applications for the Android OS and also using HTML5, compare the advantages and disadvantages of native and cross-platform application, and describes a detailed approach in developing client-server applications.

The practical part describes both created application, presents graphical user interface of the applications, shows snippets of source code for connections of the applications to the server and other interesting pieces of source code. This section also includes comparing the performance of both created applications and also server side. The results are shown in graphs. At the end of the practical part, is comparison of aspects of the actual publication both applications.

All objectives of the final thesis I was able to achieve, native and web application is fully working also server part. Based on the results measured on both of developed applications and on the server, it is clear that native applications compared with cross-platform application on the same mobile device is smoother and faster, but the development of native applications is more complex and takes more time.

The attachment to this work is a CD, which contains a source code of both client applications and server-side. CD includes an Excel document, with measured values of tests both applications and server side.

ZOZNAM POUŽITEJ LITERATÚRY

[1] WALLACE, Jackson. *Android Apps For Absolute Begginers*. New York: Apress, 2011. ISBN 978-1-4302-3446-3. Dostupné z: http://books.google.sk/books?id=gSdFq2ype_sC&lpg=PR16&dq=android%20for%20beginners&hl=sk&pg=PR16#v=onepage&q=android%20for%20beginners&f=false

[2] Android, the world's most popular mobile platform. *Android Developers* [online]. 2013 [cit. 2013-05-05]. Dostupné z: http://developer.android.com/about/index.html

[3] Dashboards. *Android Developers* [online]. 2013 [cit. 2013-05-05]. Dostupné z: http://developer.android.com/about/dashboards/index.html

[4] Dashboards. *Android Developers* [online]. 2013 [cit. 2013-05-05]. Dostupné z: http://developer.android.com/about/dashboards/index.html

[5] Java (programming language). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2013-05-08]. Dostupné z: http://en.wikipedia.org/wiki/Java_(programming_language)

[6] Java (programming language). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2013-05-08]. Dostupné z: http://en.wikipedia.org/wiki/Java_(programming_language)

[7] HTML5 — Smile, it's a Snapshot!. *W3C Blog* [online]. 2012 [cit. 2013-05-03]. Dostupné z: http://www.w3.org/QA/2012/12/html5_smile_its_a_snapshot.html

[8] HTML5 differences from HTML4. *W3C Blog* [online]. 2012 [cit. 2013-05-03]. Dostupné z: http://www.w3.org/TR/html5-diff/

[9] SCHURMAN, Eric M. Dynamické HTML v akci: HTML, DHTML a XML, kaskádní styly (CCS), skriptování, kompatibilita s různými prohlížeči, design interaktivních stránek. 1. vyd. Praha: Computer Press, 2000, 421 s. ISBN 80-722-6401-X.

[10] DELLWING, Ingo. *HTML 4: příručka tvůrce webu.* 1. vyd. Praha: Grada, 2002, 272 s. ISBN 80-247-0297-5.

[11] PhoneGap Seeks to Bridge the Gap Between Mobile App Platforms. *Gigaom* [online]. 2009 [cit. 2013-05-05]. Dostupné z: http://gigaom.com/2009/04/05/phonegap-seeks-to-bridge-the-gap-between-mobile-app-platforms/

[12] Apache Cordova gets a new look. *H online* [online]. 2012 [cit. 2013-05-05]. Dostupné z: http://www.h-online.com/open/news/item/Apache-Cordova-gets-a-new-look-1440114.html

[13] PhoneGap API Reference. *PhoneGap Documentation* [online]. 2013 [cit. 2013-05-03]. Dostupné z: http://docs.phonegap.com/en/2.7.0/index.html

[14] PhoneGap – nejrychlejší řešení pro vývoj aplikací na chytrotelefonech. *Karel Mašát* [online]. 2012 [cit. 2013-05-03]. Dostupné z: http://www.karelmasat.cz/phonegap-nejrychlejsi-res.aspx

[15] JQuery Mobile. *JQueryMobile* [online]. 2013 [cit. 2013-05-03]. Dostupné z: http://www.jquerymobile.com/

[16] Servlets and JSP: An Overview. *Kossiakoff Computer Center Facility* [online]. 2008 [cit. 2013-05-05]. Dostupné z: http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/

[17] JSP - Architecture. *Tutorialspoint* [online]. 2012 [cit. 2013-05-05]. Dostupné z: http://www.tutorialspoint.com/jsp/jsp_architecture.htm

[18] JSP - Life Cycle. *Tutorialspoint* [online]. 2012 [cit. 2013-05-05]. Dostupné z: http://www.tutorialspoint.com/jsp/jsp_life_cycle.htm

[19] What are Web Services. *Tutorialspoint* [online]. 2013 [cit. 2013-05-10]. Dostupné z: http://www.tutorialspoint.com/webservices/what_are_web_services.htm

[20] Introducing JSON. *Json* [online]. 2013 [cit. 2013-05-10]. Dostupné z: http://www.json.org/

[21] GitHub. GitHub [online]. 2013 [cit. 2013-05-05]. Dostupné z: https://github.com/

[22] We Launched. GitHub [online]. 2013 [cit. 2013-05-05]. Dostupné z: https://github.com/blog/40-we-launched

[23] Učebné texty, prednášky z predmetu Softvérové inžinierstvo, Ing. Radek Šilhavý, Ph.D.

[24] *A Comparison of Thin-Client Computing Architectures*. 2000, 16 s. Dostupné z: http://www.nomachine.com/documentation/pdf/cucs-022-00.pdf

[25] Fat client. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2013-05-07]. Dostupné z: http://en.wikipedia.org/wiki/Fat_client

[26] HOGAN, Brian P. HTML5 a CSS3: výukový kurz webového vývojáře. Vyd. 1. Brno: Computer Press, 2011, 272 s. ISBN 978-80-251-3576-1.

[27] LUBBERS, Peter, Brian ALBERS a Frank SALIM. HTML5: programujeme moderní webové aplikace. Vyd. 1. Brno: Computer Press, 2011, 304 s. ISBN 978-80-251-3539-6.

[28] STARK, Jonathan a Brian JEPSON. *Building Android apps with HTML, CSS, and JavaScript*. 2nd ed. Sebastopol, CA: O'Reilly, 2012, xiii, 158 p. ISBN 14-493-1641-7. Dostupné z: http://books.google.sk/books?id=zWndz5OpUzgC&lpg=PP1&hl=sk&pg=PP1#v=onepage&q&f=false

[29] MURPHY, Mark L. *Beginning Android 2*. New York: Apress, 2010, xvi, 397 s. Books for profesionals by professionals. ISBN 978-1-4302-2629-1. Dostupné z: http://books.google.sk/books?id=2XeNswkT_2YC&lpg=PP1&dq=android&hl=sk&pg=PP1#v=onepage&q=android&f=false

[30] JORDAN, Lucas L a Pieter GREYLING. *Practical Android projects*. New York: Apress, 2011, xiv, 404 p. ISBN 14-302-3243-9. Dostupné z: http://books.google.sk/books?id=2v55tfq9rosC&lpg=PP1&hl=sk&pg=PP1#v=onepage&q&f=false

ZOZNAM POUŽITÝCH SYMBOLOV A ZKRATIEK

OS Operačný systém je softvér, ktorý spravuje zdroje počítača alebo iného zariadenia.

CSS Cascading Style Sheets, kaskádové štýly, jednoduchý mechanizmus na vizuálne formátovanie dokumentov.

HTML HyperText markup Language, značkovací jazyk určený na vytváranie webových stránok a iných informácii.

GNU GNU's Not Unix, slobodný operačný systém v neustálom vývoji.

GPS Global Positioning System, satelitný navigačný systém na zistenie presnej pozície.

GUI Graphical User Interface, interaktivné grafické používateľské rozhranie, ktoré je k dispozícii používateľovi na ovládanie strojov, zariadení alebo počítačových programov.

PaaS Platform as a service, je to druh cloudových počítačových služieb, ktoré poskytujú výpočtovú platformu pre webovú službu.sl

RSS Rich Site Summary, je obdoba XML formátu určená pre čítanie noviniek na webových stránkach.

AJAX Asynchronus JavaScript and XML, technológia pre vývoj interaktívnych webových aplikácii, ktoré menia obsah stránok bez nutnosti ich znovunačítania.

GSON Google Gson, Java knižnica na serializáciu a deserializáciu Java objektov do a z formátu JSON.

API Aplication programming interface, rozhranie pre programovanie aplikácii, ide o zbierku funkcií a tried, ktoré určujú akým spôsobom sa majú funkcie knižníc volať zo zdrojového kódu programu.

JSON JavaScript object Notation, spôsob zápisu dát (dátový formát) nezávislý na počítačovej platforme, určený na prenos dát.

DOM Document Object Model, objektovo orientovaná reprezentácia XML alebo HTML dokumentu. Umožňuje prístup a modifikáciu obsahu, štruktúry alebo štýlu dokumentu.

ZOZNAM OBRAZKOV

ZOZNAM GRAFOV

ZOZNAM PRÍLOH

Príloha P I: CD so zdrojovými kódmi klientskych aplikácii a serverovej strany, dokument v Exceli s výsledkami testov výkonu.