

Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Ján Antala

ADAPTÍVNY WEB DIZAJN

Diplomová práca

Študijný program: Informačné systémy

Študijný odbor: 9.2.6 Informačné systémy

Miesto vypracovania: Ústav aplikovanej informatiky, FIIT STU, Bratislava

Vedúci práce: doc. Ing. Michal Čerňanský PhD.

máj 2013

ANOTÁCIA

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLOGIÍ

Študijný program: INFORMAČNÉ SYSTÉMY

Autor: Ján Antala

Diplomová práca: Adaptívny web dizajn.

Vedúci diplomovej práce: doc. Ing. Michal Čerňanský PhD.

Máj, 2013

Používanie mobilných zariadení rapídne rastie a tie so sebou prinášajú nie len rôzne veľkosti displejov a nové interakčné spôsoby, ale aj nové druhy pripojenia na internet. Je preto nevyhnutné prispôbiť webové stránky rôznym zariadeniam. V práci vychádzame z techniky „Mobile first“, ktorá kladie dôraz na vytvorenie dizajnu s vysokým výkonom pre mobilné zariadenia a neskorším profitom pri klasických stolových počítačoch. V práci prezentujeme nový prístup adaptácie webových komponentov založenom na vlastnostiach používateľovho zariadenia a externých podmienok. Experimentálnym overením úspešnosti navrhnutých spôsobov adaptácie sme dosiahli lepšie výsledky na testovacej vzorke dát ako boli pôvodne používané.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: INFORMATION SYSTEMS

Author: Ján Antala
Diploma Thesis: Adaptive web design.
Supervisor: doc. Ing. Michal Čerňanský PhD.
2013, May

Mobile device penetration grows rapidly and it brings not only different display sizes with new human interaction methods, but also new kinds of internet connection. So it is necessary to adapt web pages for different devices. In our work, we follow „Mobile First” design, which emphasizes a great performance on mobile devices and later profit on classical desktop computers. We present a new way of web components adaptation based on a device features and external conditions in our work. The evaluation of the proposed method of adaptation shows slightly better results than original approach.

Podakovanie

Týmto sa chcem poďakovať najmä vedúcemu práce, doc. Ing. Michalovi Čerňanskému, PhD., za jeho cenné rady, pripomienky a venovaný čas pri konzultáciách.

Obsah

1	Úvod	1
2	Adaptácia	2
2.1	Možnosti prispôsobenia	2
2.1.1	Veľkosť a rozlíšenie	2
2.1.2	Interakčné prostriedky	4
2.1.3	Pripojenie	5
2.1.4	Platforma	8
2.2	Adaptačné techniky	8
2.2.1	Responsive design	8
2.2.2	Mobile First	9
2.2.3	Progressive enhancement	10
2.2.4	Graceful degradation	10
2.2.5	Server-side Adaptation	11
2.2.6	RESS (Responsive Design + Server Side Components)	12
3	Nástroj na overenie	13
3.1	Komponenty	13
3.1.1	Video	13
3.1.2	Mapy	16
3.1.3	Lightboxy	16
3.1.4	Otvorenie v aplikácii / Stiahnutie aplikácie	17
4	Overenie	18
5	Záver	18

Zoznam obrázkov

1	Porovnanie zariadení vzhľadom na veľkosť displeja	3
2	Schopnosť ovládania mobilných telefónov	4
3	Schopnosť ovládania tabletov	5
4	Schopnosť ovládania hybridných počítačov	5
5	Fázy spracovania požiadavky na server	6
6	Čas potrebný na spracovanie požiadavky na server	7
7	Progressive enhancement vs Graceful degradation	11
8	Prenášané dáta pri požiadavke na video zo servera YouTube	13
9	Prenášané dáta pri požiadavke na video zo servera Vimeo	14
10	Chybová hláška po otvorení custom URL schémy, pokiaľ aplikácia neexistuje	17
11	Otvorenie natívnej aplikácie v iOS 6	18

1 Úvod

S príchodom mobilných zariadení, ktoré postupne nahradzujú klasické počítače, vznikol problém pri správnom zobrazovaní webového obsahu. Web na ne nebol pripravený, a tak sa im ponúkala verzia stránky pre klasické počítače.

Len za posledných pár rokov sa vo svete predalo viac ako 1 bilión mobilných zariadení, 1.038 bilióna celkovo [23]. Mobilné telefóny a tablety sa stali ešte viac personálnymi a používajú sa neustále počas celého dňa pri rôznych činnostiach [3, 2, 17]. Ich predajnosť sa zvyšuje každým dňom. V porovnaní vývoja trhového podielu osobných počítačov typu WINTEL a mobilných zariadení Apple a Android za posledné roky je tento trend ešte viac viditeľný.

Postupom času sa vyvinuli rôzne spôsoby ako prispôsobiť web aj na mobilné zariadenia. Najlepší spôsob prispôsobenia obsahu je taký, ktorý pokrýva všetky zariadenia od najmenších mobilných po veľké televízne obrazovky a nevytvára pre rôzne zariadenia vlastné samostatné stránky.

Ignorovanie webu na mobilných zariadeniach väčšinou spoločnosťí však vedie k vytváraniu samostatných natívnych aplikácií pre každú platformu. Okrem vedúceho postavenia Androidu a iOS existuje množstvo ďalších platforiem, pre ktorú treba vytvoriť vlastnú aplikáciu, a tým sa vývoj predražuje.

Nevýhodou natívnych aplikácií je, že neotvárajú webové odkazy a stále nemáme istotu, že si ich používateľ stiahne a nainštaluje. Taktiež vzniká problém pri aktualizáciách, používateľ ju musí manuálne spustiť. Nestačí len otvoriť aplikáciu, ktorá bude automaticky obsahovať najnovšiu verziu ako web. S tým je spojený aj problém s ich udržiavaním.

Práve tu je neoddeliteľná súčasť webu a mobilných zariadení. V súčasnosti populárne sociálne siete, ale aj emaily či qr kódy obsahujú množstvo odkazov na web. Na rovnaké odkazy je však možné prísť aj z klasických počítačov. Je tak dôležité, aby sa obsah používateľom zobrazil správne bez ohľadu na to, na akom zariadení k nemu prístupujú.

Optimalizácia webového obsahu pre mobilné zariadenia má veľmi krátku históriu. Dnes však už existujú základné vzory, podľa ktorých je možné prispôsobiť zobrazenie webového obsahu na ich malé displeje [9, 26].

Problémom je, že neustále rastú rozmery mobilných zariadení, ale aj veľké televízne obrazovky sa stávajú prístupovým bodom k webovému obsahu. Len za posledné 3 mesiace bolo predaných 29% android zariadení s obrazovkou väčšou ako 4.5 palca [22] a k podobnému trendu prístupujú aj iní výrobcovia.

Optimalizácia webového obsahu zariadeniam však neznamena len jeho vizuálne prispôsobenie rôznym veľkostiam displejom. Nemenej dôležitým prvkom je aj pripojiteľnosť zariadenia na sieť s cieľom čo najrýchlejšieho stiahnutia, zobrazenia stránky a šetrenia používateľových prenášaných dát a financií.

Nielen práve preto je vhodné použiť princípy adaptívneho web dizajnu. Medzi jeho hlavné charakteristiky patria všadeprítomnosť, flexibilita, výkonnosť, rozšíriteľnosť a pria-

teľskosť k budúcnosti [7]. V súčasnosti nevieme povedať aké zariadenia sa budú predávať o pár rokov, aké budú mať vlastnosti, ale s celkom veľkou pravdepodobnosťou budú obsahovať webový prehliadač.

Výzvou sa tak stáva nie len vytvorenie celkového používateľského rozhrania, ale aj jednotlivých prvkov ako je navigácia či ovládacie prvky, ktoré by pomohli správne zobrazíť obsah na malých zariadeniach a zároveň aby sa obsah prispôbil aj tabletom, klasickým počítačom či pre veľké obrazovky televíznych prijímačov. Keďže rozlíšenia mobilných zariadení sa začínajú prelínať s klasickými počítačmi a aj klasické počítače pridávajú nové spôsoby ovládania dotykom, je dôležité rozoznať spôsob ovládania zariadenia a prispôbiť navigáciu a obsah na dotyk alebo klávesnicu a myš. Rovnako je potrebné rozoznať aj internetové pripojenie zariadenia a automaticky mu odovzdať taký obsah, aby sa mu čo najrýchlejšie načítal a aby to používateľa nestálo zbytočný čas a financie za prenášané dáta.

2 Adaptácia

2.1 Možnosti prispôsobenia

Existuje mnoho možností, na základe ktorých môžeme prispôbovať webové aplikácie jednotlivým zariadeniam. Hlavným prvkom adaptácie je veľkosť a rozlíšenie displeja cieľového zariadenia. Ďalšími ale nemej dôležitými sú spôsoby ovládania zariadenia, jeho pripojenie na internet alebo samotná platforma.

2.1.1 Veľkosť a rozlíšenie

Jednou zo základných možností prispôsobenia webových aplikácií je adaptácia na základe zobrazovacieho displeja zariadenia. Displeje môžu mať rozličné fyzické veľkosti ale aj rozlíšenia.

Časy s „rovnakým“ displejom na všetkých zariadeniach a podporou jednotného statického rozlíšenia na webových stránkach sú už dávno preč. S príchodom nových malých mobilných zariadení sa potreba prispôsobenia ešte viac umocnila. Veľkosti a rozlíšenia zariadení sa postupne začínali prelínať, dokonca v súčasnosti sa na trh uvádzajú zariadenia s väčším rozlíšením ako majú monitory stolových počítačov.



Obr. 1: Porovnanie zariadení vzhľadom na veľkosť displeja [26].
 Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

Keďže sa začínajú vyskytovať rovnaké rozlíšenia displeja na rozlične veľkých zariadeniach alebo opačne, je potrebné medzi zariadeniami rozlišovať aj inými spôsobmi. Je potrebné správne porozumieť jednotkám „pixel” a „viewport”.

Na rozdiel od pixelu definovaného W3C pomocou pozorovacieho uhlu a vzdialenosti [21] existujú rôzne iné bežne používané jednotky „CSS pixel”, „device pixel” a „density-independent pixel” [15]. CSS pixel je abstraktný, môže sa zvyšovať alebo znižovať, používa sa

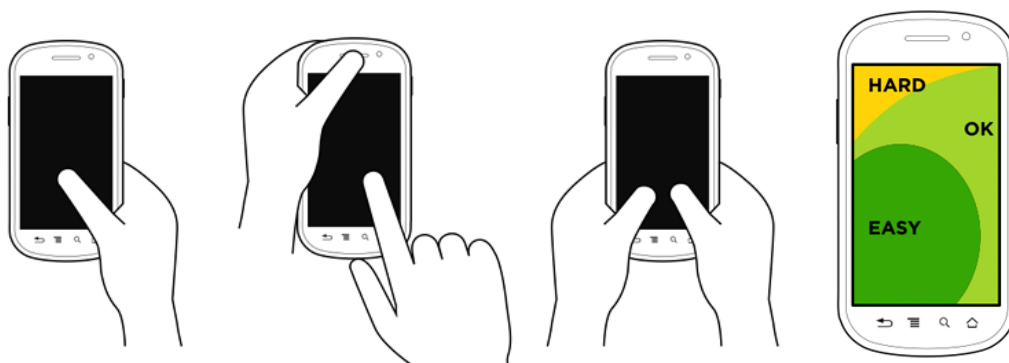
bežne v kóde na definovanie rozmerov elementov. Device pixel je fyzický pixel nachádzajúci sa na zariadení. Pretože zariadenia majú stále viac fyzických pixelov a tým aj ich väčšiu hustotu, zaviedol sa pojem density-independent pixel. Ten je opäť abstraktný a predstavuje počet CSS pixelov optimálnych na prezeranie obsahu. Pokiaľ by nebol zavedený, tak zariadenia s veľkou hustotou pixelov by sa nedali použiť na bežné prezeranie obsahu, nakoľko pixely sú veľmi malé a zobrazený text alebo elementy by tak boli nečitateľné.

Viewport je celkové miesto potrebné na zobrazenie webovej stránky. Na mobilných zariadeniach je situácia komplikovanejšia, pretože stále existuje množstvo stránok, ktoré nie sú na ne optimalizované. Preto ho výrobcovia mobilných prehliadačov rozdelili na „layout viewport” a „visual viewport”. [15] Layout viewport je pre rozmiestnenie elementov celej stránky a visual viewport je definovaný pre elementy po priblížení stránky tak, že sa nezmestila na displej zariadenia.

2.1.2 Interakčné prostriedky

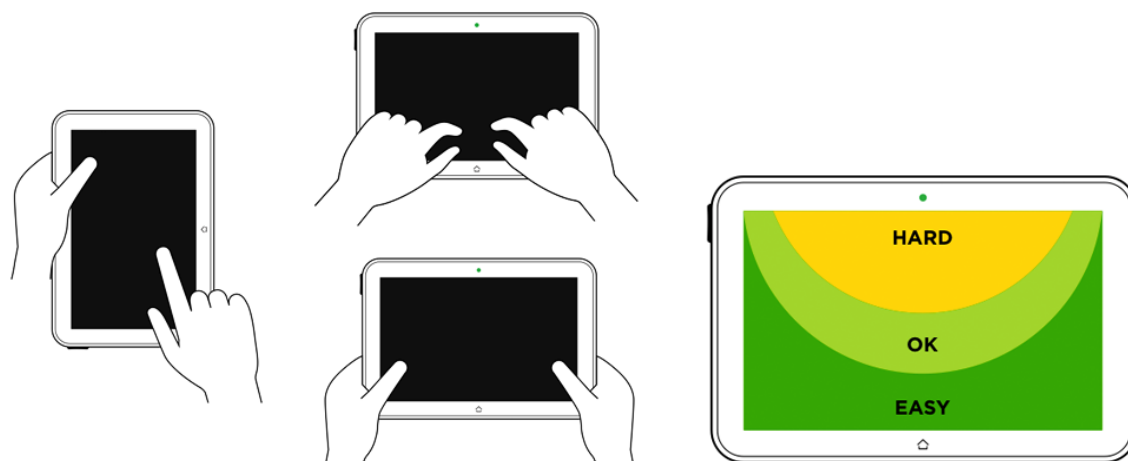
Postupom času začína upadať používanie klasických stolových počítačov ovládaných pomocou klávesnice a myši a presadzujú sa nové druhy mobilných zariadení so vstupným interfejsom v podobe dotykovej plochy. Tá sa ako ovládací prostriedok začína presadzovať okrem mobilných zariadení aj v notebookoch. Pri dizajne aplikácie je preto potrebné myslieť aj na takýchto používateľov. Okrem nich však existujú aj iné možnosti ovládania ako sú napríklad sledovanie pohybu očí, natáčanie zariadenia či počúvanie hlasových povelov. Je tak potrebné brať do úvahy aj ďalšie možnosti.

Dôležitým prvkom v prípade mobilných zariadení je možnosť ovládania aplikácií jednou rukou. Mobilné zariadenia sú využívané takmer pri každej príležitosti či vo vonkajšom alebo vnútornom prostredí, preto je potrebné správne prispôbiť vzhľad a rozmiestnenie ovládacích prvkov. Pre ne platí, že najjednoduchšie dosiahnuteľné časti sú v spodnej strane zariadenia a s postupom k hornému okraju možnosť dosiahnutia klesá [4].



Obr. 2: Schopnosť ovládania mobilných telefónov [26].
Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

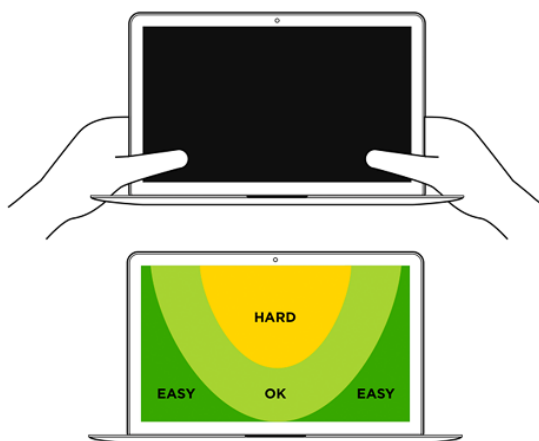
Väčšie mobilné zariadenia alebo tablety už nie je možné pohodlne udržať v jednej ruke a tak je dôležité prispôsobiť ovládanie na dve ruky. Tablety sú držané v dvoch rukách za hranu a tak najlepšie dosiahnuteľné miesta sú na jeho okrajoch [4].



Obr. 3: Schopnosť ovládania tabletov [26].

Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

Podobné výsledky [4] sú aj pri novej kategórii zariadení, tzv. hybridných notebookov, ktoré okrem klávesnice obsahujú aj dotykový displej.



Obr. 4: Schopnosť ovládania hybridných počítačov [26].

Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

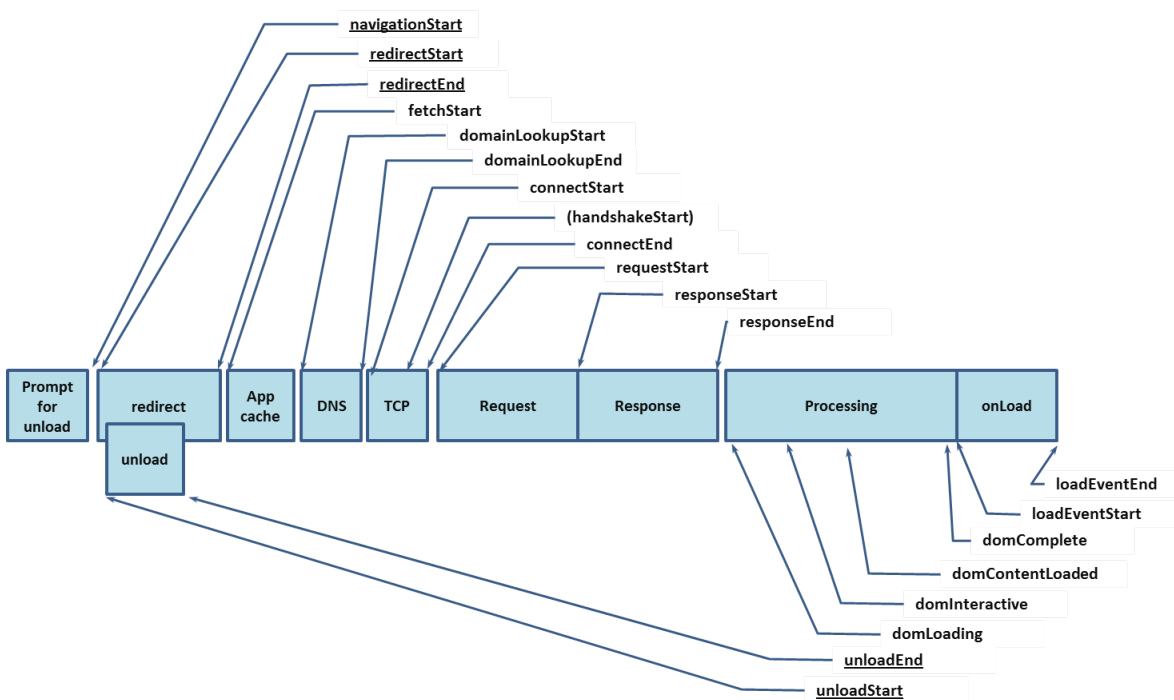
2.1.3 Pripojenie

Spoločnou vlastnosťou webových aplikácií je, že prístupujú k rôznym zdrojom, ktoré sa môžu nachádzať okrem lokálneho úložiska aj na serverom, pomocou internetu. Spôsoby prenosu dát sú rôzne, od pevného pripojenie cez bezdrôtové až po mobilné, a každé z nich má iné

vlastnosti.

V poslednom období sa spolu s mobilnými zariadeniami rozširuje aj používanie mobilného internetu. Keďže našim cieľom je, aby sa webová stránka načítala používateľovi čo najrýchlejšie, respektíve ak chceme aby sa používateľ na našu stránku prišiel a príchod si nerozmyslel pri jej dlhom načítavni, tak musíme šetriť množstvom prenášaných dát. Takéto šetrenie dát zároveň šetrí aj peňaženky používateľov [6], hlavne pokiaľ sa jedná o roamingové dáta v zahraničí.

Načítanie webovej stránky sa skladá z viacerých fáz. Okrem samotnej konektivity používateľa aj spracovanie požiadavky serverom a následne vykonanie akcie v prehliadači. To je jediná fáza, ktorú môžeme ovplyvniť.



Obr. 5: Fázy spracovania požiadavky na server [20].

Prevzaté z <http://www.w3.org/TR/navigation-timing/>

Tieto fázy môžeme aj priamo merať pomocou interfacu `performance.timing`, ktorý ich zobrazuje priamo v podobe času [10]. Vďaka tomu máme dokonalejší prehľad o používateľovom pripojení a vieme mu tak prispôbiť jednotlivé komponenty stránky. W3C špecifikácia je vo fáze „Recommendation” [20], ale stále hlavnou nevýhodou je chýbajúca podpora v niektorých prehliadačoch. Čiastočnou náhradou je meranie rozdielov dvoch časov, ale tak získame dáta len zo spracovania požiadavky na strane klienta.



Obr. 6: Čas potrebný na spracovanie požiadavky na server.

Každá požiadavka na server niečo stojí. Ideálny prípad je taký, že medzi zariadením a serverom sa neprenášajú žiadne dáta a všetky prístupy ku zdrojom sa riešia len z lokálneho úložiska. V prípade internetových aplikácií to však väčšinou nie je úplne možné, pretože používateľ chce pristupovať k čo najčerstvejším dátam. Cieľom je však čo najviac limitovať požiadavky na server.

To, či je vôbec používateľ pripojený na internet vieme zistiť pomocou interfacu `navigator.onLine`, ktorý vráti hodnotu „true” alebo „false” a takisto môžeme počúvať na zmeny pripojenia vďaka „event listenerom” na `window.online` a `window.offline`.

Rýchlosť, akou je používateľ pripojený na internet, je dostupná v objekte `navigator.connection` pod atribútom „bandwidth” charakterizujúcej pripojenie v MB/s [19]. V prípade zmeny rýchlosti je rovnako vyvolaný „event”. Nevýhodou je zatiaľ slabá podpora zo strany prehliadačov.

2.1.4 Platforma

Rozhodovanie sa na základe platformy je taktiež veľmi dôležité. Umožňuje nám jednak zjednotiť dizajn a zmenšiť počet potrebných komponentov na webovej stránke, ale taktiež vytvárať cieleňú reklamu. Rozlišovanie prebieha na základe pola „user agent” špecifickom pre každý prehliadač, respektíve operačný systém.

V prípade zisťovania podpory jednotlivých vlastností je však lepšie priamo zisťovať podporu komponentu zo strany prehliadača ako zisťovaním a porovnaním s platformou. Nemusíme si tak udržiavať databázu a neustále ju aktualizovať. Takéto riešenie je preto z pohľadu vývoja lepšie pre budúcnosť.

2.2 Adaptačné techniky ¹

S príchodom prvých mobilných zariadení existoval rozdiel medzi mobilným webom a webom pre desktopy a tak bolo pomocou servera jednoduché zistiť, aká verzia sa má zariadeniu zobrazit.

Pretože dnes už existuje mnoho zariadení od mobilných cez tablety až po klasické počítače a vzájomne sa prelínajú, je potrebné zabezpečiť, aby sa webový obsah zobrazoval správne na každom z nich.

„There is no Mobile Web. There is only The Web, which we view in different ways. There is also no Desktop Web. Or Tablet Web. Thank you.” Stephen Hay [12]

Tento výrok bol vyslovený už pred niekoľkými rokmi a v súčasnosti pri prelínaní rôznych zariadení sa stále viac potvrdzuje. Pre vývoj webovej stránky alebo aplikácie existuje viacero spôsobov [5], každý má svoje výhody a nevýhody. Správnosť výberu konkrétnej metódy záleží od toho, či ideme upravovať už existujúcu webovú verziu na rôzne zariadenia alebo či vytvárame novú aplikáciu a v neposlednom rade aj od vynaloženého úsilia či financií.

2.2.1 Responsive design

Pojem „Responsive design” bol pôvodne súbor pravidiel tvorby dizajnu pre rôzne rozlíšenia, ktoré definoval Ethan Marcotte v článku Responsive design v roku 2010. Všetkým zariadeniam je posielaný rovnaký HTML a javascript, rozdiel je len v designe. Design sa zakladá na používaní flexibelného vzhľadu stránky, ktorý sa prispôboval rôznym zariadeniam, flexibilných obrázkoch, ktoré sa prispôbujú vzhľadu a CSS media queries. [16, 18] Až neskôr bol označený ako metóda na dosiahnutie výsledku.

¹Tejto kapitole som sa už z časti venoval vo svojej bakalárskej práci Tvorba bohatých internetových aplikácií pre mobilné zariadenia [1]. V tomto dokumente sa nachádza rozšírená verzia doplnená o novo vzniknuté techniky adaptácie.

Tvorba vzhľadu pomocou Responsive design znamená používanie hodnôt v percentuálnom pomere namiesto statických hodnôt, obrázky sú prepojené s elementom stránky, majú nastavené jeho plné rozmery a automaticky sa prispôsobujú jeho zmenám. Vytvára sa stránka pre väčšie rozlíšenie a pomocou CSS media queries sa môžu aplikovať rôzne pravidlá pre jednotlivé elementy na základe rozlíšenia zariadenia, jeho orientácie či pomeru strán. Pri použití takéhoto prístupu však nastávajú problémy na mobilných zariadeniach s menším rozlíšením displeja a na väčších zariadeniach ako sú televízory.

Výhody:

- dobrá metóda na dosiahnutie nezávislosti zobrazenia obsahu pri rôznom rozlíšení zariadení, stránka vyzerá inak v mobilnom zariadení ako v tablete alebo stolnom počítači
- rýchly vývoj aplikácie

Nevýhody:

- neumožňuje prispôbenie obsahu, ale len jeho vzhľad
- mobilné zariadenie sťahuje plnú veľkosť obrázka, ale vidí ho v menšom rozlíšení
- problémy pri zariadeniach s nižším a väčším rozlíšením

2.2.2 Mobile First

Problémy pri správnom zobrazení stránky na zariadeniach s nižším rozlíšením podnietili vznik novej metódy dizajnu - „Mobile First”. Základ tvorí metóda Responsive design, ale pôvodný návrh stránky sa nerobí pre desktopovú verziu ale na malé rozlíšenie. Až pomocou media queries sa pridávajú elementy pre väčšie rozlíšenie. Takto sa pokryju všetky zariadenia od najmenších po najväčšie a webová stránka je pripravená na zariadenia, ktoré vzniknú aj v budúcnosti.

Technika Mobile First však okrem samotného dizajnu zahŕňa aj optimalizáciu webu pre používateľa či už z pohľadu UX alebo výkonu [25].

Výhody:

- dosiahnutie nezávislosti zobrazenia obsahu pri rôznom rozlíšení zariadení
- podporuje všetky zariadenia od najmenších po najväčšie, stránka je pripravená aj na „zariadenia budúcnosti”

Nevýhody:

- stále neumožňuje prispôbenie obsahu, ale len jeho vzhľad
- dizajn stránky musí byť vytvorený od základu, čo však môže byť aj východa

2.2.3 Progressive enhancement

V poslednom období sa stáva veľmi populárnou metódou Progressive Enhancement. Táto metóda je založená na princípe posielania rovnakého html, javascriptu a iných zdrojových súborov všetkým zariadeniam. Následné vykonávanie aplikácie sa presúva zo strany servera a prebieha na klientovi pomocou javascriptu, kde je už možné presne špecifikovať, čo sa má kedy a ako vykonávať. Používateľovi sa sprístupňujú pokročilejšie vlastností aplikácie len ak ich podporuje používaný webový prehliadač, respektíve zariadenie. Taktiež je možné načítavať objekty zo servera len vtedy, keď sú potrebné, a tým sa zabráňuje zbytočným prenosom dát. Pri spojení tejto metódy s HTML5 je dokonca možné tvoriť offline klientské aplikácie.

Pokiaľ chceme pokryť celé spektrum zariadení tak je implementácia pomocou tejto metódy náročnejšia. V spojení s metódou reponsive design je to najlepšie možné riešenie na prispôbenie si obsahu a vzhľadu aplikácie.

Výhody:

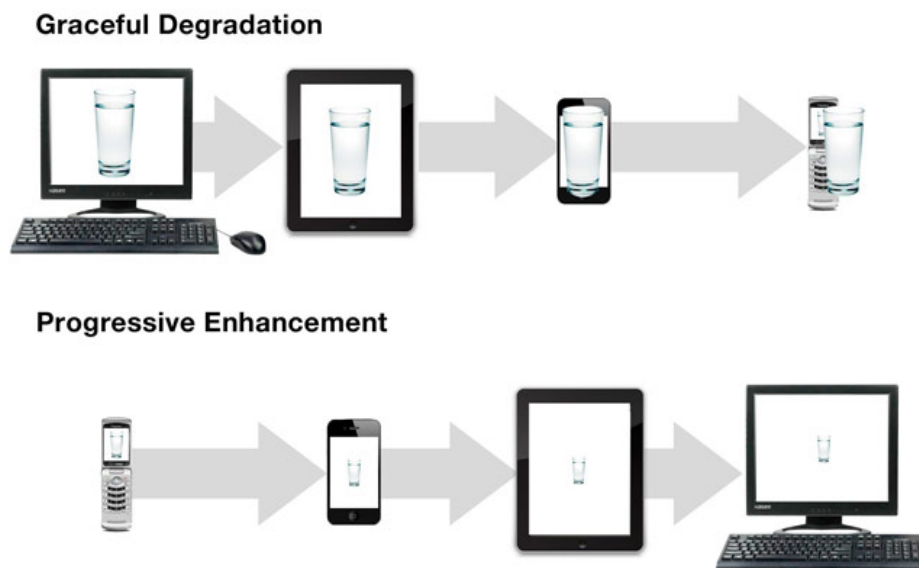
- úplne prispôbenie obsahu a vzhľadu, minimálne alebo žiadne dátové prenosy

Nevýhody:

- náročnejšia implementácia,
- rýchlosť vykonávania aplikácie závisí od výkonu zariadenia

2.2.4 Graceful degradation

Opak metódy Progressive enhancement sa nazýva Graceful degradation. Používateľovi sa predáva plne funkčná aplikácia nadizajnovaná na najlepšie zariadenia, kde sa na jednotlivé pokročilejšie funkcie postupne vypínajú vzhľadom na použité zariadenie. Používa sa často na upravenie súčasnej nasadenej verzie na potreby mobilných zariadení.



Obr. 7: Progressive enhancement vs Graceful degradation [8].

Prevzaté z <http://bradfrostweb.com/blog/web/mobile-first-responsive-web-design/>

Výhody:

- prispôsobenie obsahu a vzhľadu na jednoduchšie zariadenia
- zachovanie súčasnej verzie webovej stránky

Nevýhody:

- nezohľadňuje budúce zariadenia, už v súčasnosti majú niektoré tablety kvalitnejšie displeje ako stolové počítače

2.2.5 Server-side Adaptation

Metóda Server-side Adaptation je používaná väčšinou webových stránok na detekovanie mobilného zariadenia a v súčasnosti patrí už medzi historické techniky. Pri prístupe na stránku sa pomocou servera detekuje zariadenie a je mu ponuknutá vhodná verzia stránky, väčšinou dochádza k presmerovaniu (na mobilnú verziu). Celá logika aplikácie sa nachádza na serveri. Stránka môže byť presne vytvorená pre dané zariadenie, takže nenastávajú problémy pri dizajne, je však potrebné mať na serveri nainštalovanú knižnicu na jeho detekovanie². Detekcia zariadenia je pri priamej návšteve stránky cez prehliadač úspešná, k problémom však prichádza pri návšteve stránky cez iných klientov. Taktiež je dôležité mať databázu zariadení neustále aktualizovanú.

²napríklad DeviceAtlas alebo WURFL založené na detekovaní pola user agent

Výhody:

- zobrazenie vhodnej stránky pre zariadenie, nestahuje sa nepotrebný obsah

Nevýhody:

- potreba knižníc na detekovanie zariadenia, ktorú je nutnú neustále aktualizovať
- dlhšie načítavanie stránky na pomalšom internetovom pripojení, pretože dochádza k presmerovaniu

2.2.6 RESS (Responsive Design + Server Side Components)

Kombináciou techník Responsive Design a Server-side Adaptation vznikla novšia technika adaptácie. Pre každé zariadenie sa na serveri dynamicky vygenerujú pre neho špecifické časti webovej aplikácie, ktoré sa mu následne zobrazia a upraví pomocou responsive dizajnu.

Výhody:

- jednoduchšia udržiavateľnosť, neexistujú rôzne verzie stránky ale len jedna

Nevýhody:

- potreba nainštalovaných knižníc na detekovanie zariadenia
- náročnejšia implementácia na strane servera

3 Nástroj na overenie

Ako nástroj na overenie sa vytvorila knižnica do populárneho JavaScriptového frameworku AngularJS ³, kde sa následne skúmali možnosti adaptácie komponentov.







3.1 Komponenty

3.1.1 Video

Populárnym doplnkom súčasných webových stránok je priložené video, ktoré sa často nachádza na serveroch YouTube alebo Vimeo.

Problémom je, že pri vložení videa pomocou iframe alebo embed api sa uskutočňuje množstvo dopytov na servery a prenášajú sa zbytočné dáta aj keď používateľa video nezaujíma a vôbec si ho neprehrá. Okrem zbytočne prenášaných dát sa aj znižuje výkon, nakoľko určitý čas trvá spracovanie požiadaviek.
















Nasledujúce dáta sa prenesú pri zobrazení stránky, na ktorú bolo vložené video zo servera YouTube pomocou dostupného iframe API:

 kxopViU98Xo?autoplay=0 www.youtube.com/embed	GET	200 OK	text/html	Other	3.2 KB 5.2 KB	404 ms 400 ms
 www-embed-vflDKrJ4Q.css s.ytimg.com/yts/cssbin	GET	200 OK	text/css	kxopViU98Xo:7 Parser	26.7 KB 145 KB	124 ms 58 ms
 www-embed-vflDpvlm5.js s.ytimg.com/yts/jsbin	GET	200 OK	text/javascr...	kxopViU98Xo:20 Parser	32.4 KB 84.5 KB	156 ms 81 ms
 watch_as3-vflFZFBB0.swf s.ytimg.com/yts/swfbin	GET	200 OK	application...	Other	282 KB 282 KB	530 ms 56 ms
 crossdomain.xml i4.ytimg.com	GET	200 OK	text/x-cros...	www-embed-vflDpvlm5 Script	478 B 102 B	90 ms 89 ms
 hqdefault.jpg i4.ytimg.com/vi/kxopViU98Xo	GET	200 OK	image/jpeg	Other	10.8 KB 10.5 KB	68 ms 52 ms

Obr. 8: Prenášané dáta pri požiadavke na video zo servera YouTube

Celkovo sa vykoná 6 požiadaviek a prenesie sa 350 kB dát bez toho, aby používateľ spustil video. Podobná situácia sa opakuje aj pri požiadavke na video zo servera Vimeo pomocou iframe API.

³ AngularJS <http://angularjs.org/> je populárny JavaScriptový framework, ktorý rozširuje HTML o nové možnosti pre webové aplikácie.

	57625268?autoplay=0 player.vimeo.com/video	GET	200 OK	text/html	Other	3.9 KB 10.0 KB	405 ms 401 ms
	player.core.opt.css a.vimeocdn.com/p/1.4.31/css	GET	200 OK	text/css	57625268:1 Parser	2.3 KB 7.5 KB	54 ms 53 ms
	player.core.opt.js a.vimeocdn.com/p/1.4.31/js	GET	200 OK	application...	57625268:1 Parser	14.7 KB 39.1 KB	152 ms 57 ms
	399544952_295.jpg b.vimeocdn.com/ts/399/544	GET	200 OK	image/jpeg	57625268:1 Parser	10.0 KB 9.6 KB	571 ms 359 ms
	ga.js www.google-analytics.com	GET	200 OK	text/javasc...	57625268:1 Script	15.7 KB 38.5 KB	124 ms 64 ms
	3591052_75.jpg b.vimeocdn.com/ps/359/105	GET	200 OK	image/jpeg	57625268:1 Parser	1.8 KB 1.4 KB	49 ms 48 ms
	swfobject.v2.2.js a.vimeocdn.com/p/1.4.31/js	GET	200 OK	application...	player.core.opt.js:6 Script	4.1 KB 9.8 KB	86 ms 81 ms
	__utm.gif?utmwv=5.4.0&utms=1&utmn= www.google-analytics.com	GET	200 OK	image/gif	ga.js:60 Script	376 B 35 B	68 ms 68 ms
	__utm.gif?utmwv=5.4.0&utms=2&utmn= www.google-analytics.com	GET	200 OK	image/gif	ga.js:60 Script	376 B 35 B	123 ms 122 ms
	moogalover.swf?v=1.0.0 a.vimeocdn.com/p/flash/moogalover/1.1.	GET	200 OK	application...	swfobject.v2.2.js:1 Script	33.5 KB 33.2 KB	127 ms 50 ms
	plus_icon.gif a.vimeocdn.com/images_v6	GET	200 OK	image/gif	Other	396 B 89 B	49 ms 48 ms
	crossdomain.xml b.vimeocdn.com	GET	200 OK	application...	Other	582 B 342 B	42 ms 41 ms
	crossdomain.xml player.vimeo.com	GET	200 OK	application...	Other	714 B 342 B	291 ms 291 ms
	3591052_75.jpg b.vimeocdn.com/ps/359/105	GET	200 OK	image/jpeg	Other	1.8 KB 1.4 KB	41 ms 41 ms
	399544952_640.jpg b.vimeocdn.com/ts/399/544	GET	200 OK	image/jpeg	Other	29.9 KB 29.5 KB	149 ms 48 ms

Obr. 9: Prenášané dáta pri požiadavke na video zo servera Vimeo

V tomto prípade sa dokonca vykoná 15 požiadaviek a prenesie sa 120 kB dát. V prípade použitia javascriptového API s HTML 5 prehrávačom videa sa síce nestiahnú flashové komponenty, ale nahradia ich rovnakoveľké javascriptové súbory.

Lepšie riešenie je použiť podmienené načítavanie. Najskôr sa načíta len obrázok videa a pridajú sa jednoduché štylistické prvky aby vytvorený element pripomínal video súbor a až po kliknutí sa urobia dopyty na vzdialený server s automatickým prehratím videa. Výhodou takéhoto riešenia je zmenšenie veľkosti dopytov a s tým spojené šetrenie dát používateľov.

Čo sa týka získania obrázkového náhľadu videa, tak YouTube túto možnosť priamo poskytuje a záleží len od identifikátora videa. K obrázku je tak možné prísť priamo. Navyše po vykonaní požiadaviek na server po kliknutí na obrázok a následnom spustení videa pomocou api sa už daný obrázok nachádza v pamäti, tak sa nemusia robiť ďalšie požiadavky.

Samotný proces automatického spustenia YouTube videa nie je úplne jednoduchý, nakoľko nastavenie hodnoty „autoplay“ pri vložení elementu iframe s videom na stránku nefunguje na mobilnom prehliadači Safari, kde je táto možnosť zakázaná. V takomto prípade by

sa po kliknutí na obrázok videa načítal len element s videom a aby sa samotné video začalo prehrávať očakáva sa ďalšie kliknutie. Takéto správanie je neželené a je proti používateľskému zážitku.

Preto bolo potrebné vymyslieť lepšie riešenie. To spočíva v načítaní scriptu s javascriptovým youtube API a následným vytvorením iframe elementu s videom až pomocou neho. Takto máme prístup k programátorskému ovládaniu prehrávača videa a môžeme ho spustiť keď potrebujeme, teda po kliknutí na obrázok s náhľadom videa. Takéto riešenie už funguje aj na iOS s mobilným prehliadačom Safari.

Stále však máme problém pri staršej verzii iOS 5 a menej, tam nefunguje ani programátorske spustenie videa. V tejto verzii iOS sa však ešte distribuovala predinštalovaná aplikácia na prehrávanie Youtube videí, ktorá bola v neskorších verziách odstránená a je ju možné stiahnuť z iTunes. Výhodou je, že video môžeme otvoriť priamo v nej pomocou youtube url schémy. Musíme však používaný prehliadač správne detekovať a následne sa rozhodnúť, aké prehrávanie zvolíme.

Na detekovanie funkcionality otvorenia youtube videa v aplikácii pomocou systému iOS a jej zisťovaním len z pola „user agent” nie je správne, nakoľko by bolo potrebné získať úplne všetky verzie systému, ktorých je mnoho, a následne porovnať s verziou zariadenia.

Lepším riešením je vytvorenie testu funkčnosti prehrávania videa, ktoré je riešené pomocou nástroja Modernizr⁴ umožňujúcim detekciu základných HTML5 a CSS3 vlastností prehliadača a vytváranie vlastných testov. Následne sa na základe výsledku testu rozhodneme akú akciu vykonať. Problémom je, že testy sa vykonávajú po načítaní stránky a nechceme používateľovi automaticky otvoriť natívnu aplikáciu alebo ho presmerovať na stránky youtube. Preto je zvolená iná varianta a detekuje sa podbora vlastnosti, ktorá bola pridaná až v novšej verzii iOS 6. Konkrétne sa jedná o podboru jednotiek „vh a vw” (viewport height a viewport width) slúžiacich na nastavenie veľkosti DOM elementu. Detekcia či je zariadenie iOS vychádza z pola user agent, ale už sa nezisťuje jej verzia.

Výsledok je taký, že po kliknutí na obrázok s náhľadom videa a vyhodnotení testu prehrávania sa začne prehrávať v prehliadači alebo v natívnej aplikácii.

V prípade Vimea je situácia trochu komplikovanejšia pretože k náhľadovému obrázku sa priamo nevieme dostať a je potrebné urobiť jednu požiadavku na API, ktorá vráti informácie o videu. Vykonanie požiadavky síca nejaký čas trvá, ale aj tak je výsledok z pohľadu prenášaný údajov lepší ako robiť požiadavku priamo na samotné video.

S prehrávaním videa zo služby vimeo je situácia podobná, neexistuje však natívna aplikácia a po kliknutí na náhľad videa je používateľ v starších verziách iOS presmerovaný na stránky vimeo, v novších verziách iOS a v Androide sa mu automaticky prehrá.

⁴Modernizr <http://modernizr.com/> je JavaScriptová knižnica, ktorá detekuje dostupnosť natívnej implementácie nových technológií v prehliadači.

3.1.2 Mapy

Mapy podobne ako videá vytvárajú nechcené dátové prenosy, dokonca ich ešte aj prevyšujú pretože sa neprenášajú len údaje o aktuálne zobrazenej časti mapy ale aj jej okolie. Okrem nich navyše na webe neposkytujú taký plnohodnotný zážitok z prezerania ako v natívnej aplikácii. Nevýhodou je aj nemožnosť posúvania webovej stránky pokiaľ je element s mapou väčší ako je rozlíšenie displeja mobilného zariadenia, lebo eventy sú zachytávané mapou a posúva sa tá.

Používateľ nemusí chcieť okamžite interagovať s mapou a v takomto prípade ho zbytočne zatažuje. Výhodnejšie je tak zobrazíť len náhľad mapy, ktorý sa načíta rýchlejšie pretože šetrí prenášané dáta, a pokiaľ sa používateľ chce dozvedieť viac, tak len jednoducho na mapu klikne. V takomto prípade sa mu automaticky načíta plne interaktívna mapa. Na zobrazenie náhľadu mapy existuje API v službe google maps⁵, takže je možné využiť priamo to. Nevýhodou je, že počet požiadaviek za deň, ktoré sú zadarmo, je obmedzený, po získaní API kľúča je možné ich spraviť 25000, ďalšie sú spoplatňované.

Rozšírením na mobilných zariadeniach iOS je možnosť otvorenia mapy priamo v natívnej aplikácii, ktorá prináša ešte väčší používateľský zážitok ako zobrazenie na webe. To je uskutočňované pomocou url schémy. V starších verziách iOS sa nachádzala natívna aplikácia na Google API a bola vyvolaná otvorením nasledujúceho odkazu, kde bolo možné zadávať parametre zobrazenia.

`http://maps.google.com/`

S príchodom iOS 6 bola aplikácia nahradená vlastnom Apple aplikáciou, na ktorej spustenie sa zmenila aj url schéma a pôvodná otvorí mapu len v prehliadači. Výhodou je, že na starších, respektíve nepodporovaných zariadeniach nastáva presmerovanie na stránky Google a tak na rovnakom odkaze funguje spúšťanie aj pôvodnej aplikácie. Nová url schéma má nasledujúci tvar:

`http://maps.apple.com/`

3.1.3 Lightboxy

Lightbox je technika umožňujúca zobrazovať webový obsah v modálnych oknách nachádzajúcich sa nad úrovňou pôvodnej stránky.

Hlavným problémom použitia „lightboxov“ na mobilných zariadeniach je nesprávne zobrazovanie stránok pokiaľ nové okno je väčšie ako displej. V takomto prípade by bolo vhodnejšie používateľa presmerovať priamo na novú stránku.

⁵<https://developers.google.com/maps/documentation/staticmaps/>

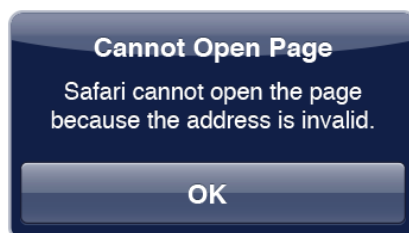
3.1.4 Otvorenie v aplikácii / Stiahnutie aplikácie

V súčasnosti sme opklopovaný množstvom informačných zdrojov, ktoré sa zväčša nachádzajú na internete dostupné na konkrétnej webovej adrese. Na tieto zdroje existuje množstvo odkazov z rôznych webových stránok, sociálnych sietí či mobilných aplikácií, ktoré zobrazia ich obsah v prehliadači.

„Links don't open apps.” Jason Grigsby [11]

Webové odkazy neotvárajú natívne aplikácie, čo je technicky pravda, no realita je trochu odlišná. Prepojenie webovej časti aplikácie s natívnou umožňuje otvárať komplexnejšie úlohy, na ktoré je vo webe nedostatočný výkon, priamo v natívnom kóde. To umožní plynulejší chod aplikácie a lepší používateľský zážitok.

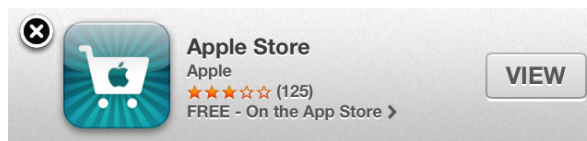
Prepájanie aplikácii sa uskutočňuje pomocou „custom url schémy” [14], ktorá je špecifická pre jednotlivé operačné systémy zariadení. Nevýhodou vlastných odkazov je, že pokiaľ používateľ nemá nainštalovanú aplikáciu, tak po kliknutí sa objaví celkom nepekná chybová hláška.



Obr. 10: Chybová hláška po otvorení custom URL schémy, pokiaľ aplikácia neexistuje [24].
Prevzaté z <http://www.lukew.com/ff/entry.asp?1654>

Lepším riešením je detekovanie, či si používateľ stiahol natívnu aplikáciu a odkaz na otvorenie v nej vytvoriť až potom, čo je overenie úspešné. Pokiaľ by nebolo, tak by sa mohlo zobraziť tlačítko na stiahnutie natívnej aplikácie, ktoré opäť musí byť prispôbené platforme na ktorej na stránku prispôbujeme, pokiaľ nechceme používateľa zahltiť všetkými platformami ktoré podporujeme.

Samotný proces detekcie či má používateľ nainštalovanú našu aplikáciu vôbec nie je triviálny, pretože na webe neexistuje žiadne dostupné API, ktoré by zahrňovalo všetky platformy. Apple síce vydal možnosť otvorenia aplikácie pomocou „Smart Banners”, ale až v iOS 6 a tak táto možnosť nie je úplne použiteľná.



Obr. 11: Otvorenie natívnej aplikácie v iOS 6 [13].
Prevzaté z <http://developer.apple.com/>

Navyše „smart banner” je priamo definovaný v html meta tagu aplikácie. Na stránke existuje len jeden ktorý volá url schému aplikácie a aj ten má prednastavený vzhľad v podobne okna v hornej časti obrazovky. Keby chceme vlastnú natívnu aplikáciu zavolať z viacerých častí webovej, prípadné volať viacero natívnych aplikácií, tak toto riešenie je nepostačujúce. Nemôže byť upravené vlastným potrebám.

Jediné možné riešenie je len v spolupráci s natívnou aplikáciou, aj tak sa však musí vyvolať otvorenie stránky v prehliadači, kde sa dá už do cookies alebo lokálneho úložiska programátorksy zapísať existencia aplikácie. Vytvorenie samotného „webView” komponentu s webovou stránkou vrámci aplikácie a následné zatvorenie nestačí. Prehliadač má oddelené úložiská stránok pre rôzne typy prístupov ako sú s internetový prehliadač, aplikácia a v iOS aj pre otvorenie internetovaj stránky z plochy. Možnosť ako tento proces zamaskovať je skrytá v procese registrácie, keď po otvorení aplikácie a zaregistrovaní pošleme používateľovi e-mail s potvrdzujúcim odkazom, ktorý otvorí webovú stránku v prehliadači.

4 Overenie

5 Záver

Literatúra

- [1] ANTALA, J. Tvorba bohatých internetových aplikácií pre mobilné zariadenia. Bakalárska práca, FIIT STU, Bratislava, 2012.
- [2] BARKHUUS, L. – POLICHAR, V. E. Empowerment through seamfulness: Smart phones in everyday life. In *Personal and Ubiquitous Computing 15*, pp. 629–639. Springer, 2011. ISSN 1617-4909.
- [3] CHUA, A. Y. K. – BALKUNJE, R. S. – GOH, D. H.-L. Fulfilling mobile information needs: a study on the use of mobile phones. In *Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, ICUIMC '11*, pp. 92:1–92:7, New York, NY, USA, 2011. ACM. doi: 10.1145/1968613.1968721. Dostupné z: <http://doi.acm.org/10.1145/1968613.1968721>. ISBN 978-1-4503-0571-6.
- [4] CLARK, J. Designing For Touch. In *The Mobile Book*, Freiburg, Germany, 2012. Smashing Magazine. ISBN 9783943075502.
- [5] CREMIN, R. Mobile web content adaptation techniques. Technical report, mobiForge, November 2, 2011. Dostupné z: <http://mobiforge.com/starting/story/mobile-web-content-adaptation-techniques>.
- [6] CREMIN, R. Performance is money, part 1: the end-user's wallet. Technical report, mobiForge, March 12, 2013. Dostupné z: <http://mobiforge.com/designing/blog/performance-money-part-1-end-users-wallet>.
- [7] FROST, B. Beyond Media Queries: Anatomy of an Adaptive Web Design. In *An Event Apart conference*, Washington DC, USA, August 6, 2012.
- [8] FROST, B. Beyond Squishy: The Principles of Adaptive Design. In *SXSW conference*, Austin, TX, USA, March 9, 2013.
- [9] FROST, B. Responsive Design Patterns. In *The Mobile Book*, Freiburg, Germany, 2012. Smashing Magazine. ISBN 9783943075502.
- [10] GRIGORIK, I. Breaking the 1000 ms Time to Glass Mobile Barrier. In *SF HTML5*, San Francisco, CA, USA, Mar 22, 2013.
- [11] GRIGSBY, J. Links Don't Open Apps. Technical report, Cloud Four, Portland, OR, USA, Mar 16, 2011. Dostupné z: <http://blog.cloudfour.com/links-do-not-open-apps/>.
- [12] HAY, S. There is no Mobile Web, Jan 07, 2011. Dostupné z: <http://www.the-haystack.com/2011/01/07/there-is-no-mobile-web/>.

- [13] iOS Developer Library. Promoting Apps with Smart App Banners. Technical report, Safari Web Content Guide, . Dostupné z: <http://developer.apple.com/library/ios/#documentation/AppleApplications/Reference/SafariWebContent/PromotingApps/AppBanners/PromotingApps/AppBanners.html>.
- [14] iOS Developer Library. Implementing Custom URL Schemes. Technical report, Safari Web Content Guide, . Dostupné z: http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphonesprogrammingguide/AdvancedAppTricks/AdvancedAppTricks.html#//apple_ref/doc/uid/TP40007072-CH7-SW50.
- [15] KOCH, P.-P. A pixel is not a pixel. In *Fronteers conference*, Amsterdam, Netherlands, Oct 4, 2012.
- [16] MARCOTTE, E. Responsive Web Design. In *A List Apart Magazine: Issue 306*, New York, New York, USA, May 25, 2010. A List Apart. Dostupné z: <http://www.alistapart.com/articles/responsive-web-design>. ISSN 1534-0295.
- [17] MUELLER, H. – GOVE, J. L. – WEBB, J. S. Understanding Tablet Use: A Multi-Method Exploration. In *Proceedings of the 14th Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2012)*, 2012.
- [18] W3C. Media Queries. In *W3C Recommendation*, 19 June 2012. Dostupné z: <http://www.w3.org/TR/css3-mediaqueries/>.
- [19] W3C. The Network Information API. In *W3C Working Draft*, 29 November 2012. Dostupné z: <http://www.w3.org/TR/netinfo-api/>.
- [20] W3C. Navigation Timing. In *W3C Recommendation*, 17 December 2012. Dostupné z: <http://www.w3.org/TR/navigation-timing/>.
- [21] W3C. CSS Values and Units Module Level 3. In *W3C Candidate Recommendation*, 4 April 2013. Dostupné z: <http://www.w3.org/TR/css3-values/>.
- [22] WROBLEWSKI, L. Data Monday: Big Screen Smartphones. Technical report, LukeW Ideation + Design, Silicon Valley, CA, USA, October 15, 2012. Dostupné z: <http://www.lukew.com/ff/entry.asp?1644>.
- [23] WROBLEWSKI, L. Data Monday: Can Smartphones Keep Growing? Technical report, LukeW Ideation + Design, Silicon Valley, CA, USA, October 21, 2012. Dostupné z: <http://www.lukew.com/ff/entry.asp?1644>.
- [24] WROBLEWSKI, L. Linking Mobile Web and Native App Experiences. Technical report, LukeW Ideation + Design, Silicon Valley, CA, USA, November 14, 2012. Dostupné z: <http://www.lukew.com/ff/entry.asp?1654>.

- [25] WROBLEWSKI, L. Mobile First. New York, New York, USA : A Book Apart, 1st edition, 2011. Dostupné z: <http://www.abookapart.com/products/mobile-first>. ISBN 9781937557027.
- [26] WROBLEWSKI, L. Responsive Navigation: Optimizing for Touch Across Devices. Technical report, LukeW Ideation + Design, Silicon Valley, CA, USA, November 2, 2012. Dostupné z: <http://www.lukew.com/ff/entry.asp?1649>.