

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.

december 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.

December, 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 a novým trendom v oblasti interakcie. V práci vychádzame z techniky „Mobile first progressive enhancement“, 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 a nové možnosti interakcie webovej aplikácie založených na rozpoznaní reči, pohybu alebo rotácie zariadenia. 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, December

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 and new interaction methods. In our work, we follow „Mobile First progressive enhancement” 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 and new possibilities of interaction with web application based on speech recognition, motion or device rotation. The evaluation of the proposed method of adaptation shows slightly better results than original approach.

### **Čestné prehlásenie**

Čestne prehlasujem, že záverečnú prácu som vypracoval samostatne, s použitím uvedenej literatúry a na základe svojich vedomostí a znalostí.

Bratislava, december 2013

.....

*Vlastnoručný podpis*

## **Podakovanie**

Týmto sa chcem podakovať najmä vedúcemu práce, doc. Ing. Michalovi Čerňanskému, PhD., za jeho cenné rady, pripomienky a venovaný čas pri konzultáciách. Vďaka patrí taktiež mojej rodine a všetkým kontribútorom podiaľajúcich sa na zlepšení vytvoreného softvéru.

# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
<b>2</b>	<b>Adaptácia</b>	<b>2</b>
2.1	Možnosti prispôsobenia . . . . .	2
2.1.1	Veľkosť a rozlíšenie . . . . .	2
2.1.2	Interakčné prostriedky . . . . .	4
2.1.3	Pripojenie . . . . .	8
2.1.4	Platforma . . . . .	10
2.2	Adaptačné techniky . . . . .	10
2.2.1	Responsive design . . . . .	10
2.2.2	Mobile First . . . . .	11
2.2.3	Progressive enhancement . . . . .	12
2.2.4	Graceful degradation . . . . .	12
2.2.5	Server-side Adaptation . . . . .	13
2.2.6	RESS (Responsive Design + Server Side Components) . . . . .	14
<b>3</b>	<b>Nástroj na overenie</b>	<b>15</b>
3.1	Alternatívne spôsoby ovládania . . . . .	16
3.1.1	Reč . . . . .	16
3.1.2	Pohyb . . . . .	19
3.1.3	Gyroskop . . . . .	23
3.2	Komponenty . . . . .	29
3.2.1	Video . . . . .	29
3.2.2	Mapy . . . . .	32
3.2.3	Lightboxy . . . . .	32
3.2.4	Otvorenie v aplikácii / Stiahnutie aplikácie . . . . .	33
3.3	Nástroje . . . . .	34
3.3.1	Detekcia . . . . .	34
<b>4</b>	<b>Overenie</b>	<b>35</b>
<b>5</b>	<b>Záver</b>	<b>35</b>
	<b>Príloha A Obsah CD nosiča</b>	<b>39</b>

## Zoznam tabuliek

1	Alpha rotácia gyroskopu . . . . .	25
2	Beta rotácia gyroskopu . . . . .	26
3	Gamma rotácia gyroskopu . . . . .	27

## Zoznam obrázkov

1	Porovnanie zariadení vzhľadom na veľkosť displeja . . . . .	3
2	Schopnosť ovládania mobilných telefónov . . . . .	5
3	Schopnosť ovládania tabletov . . . . .	5
4	Schopnosť ovládania hybridných počítačov . . . . .	6
5	Accelerometer a gyroskop . . . . .	7
6	Fázy spracovania požiadavky na server . . . . .	8
7	Čas potrebný na spracovanie požiadavky na server . . . . .	9
8	Progressive enhancement vs Graceful degradation . . . . .	13
9	Webová stránka vytvoreného projektu angular-adaptive . . . . .	15
10	Algoritmus ovládania aplikácie pomocou reči . . . . .	18
11	Vizualizácia videa zachyteného web kamerou . . . . .	19
12	Vizualizácia videa po aplikovaní filtra detekcie kože . . . . .	20
13	Vizualizácia videa po aplikovaní filtra detekcie hrán . . . . .	20
14	Algoritmus ovládania aplikácie pomocou pohybu . . . . .	22
15	Algoritmus ovládania aplikácie pomocou gyroskopu . . . . .	24
16	Gyrocopter - gyroskop simulátor . . . . .	28
17	Prenášané dáta pri požiadavke na video zo servera YouTube . . . . .	29
18	Prenášané dáta pri požiadavke na video zo servera Vimeo . . . . .	30
19	Chybová hláška po otvorení custom URL schémy, pokiaľ aplikácia neexistuje . . . . .	33
20	Otvorenie natívnej aplikácie v iOS 6 . . . . .	34



# 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 [31]. Mobilné telefóny a tablety sa stali ešte viac personálnymi a používajú sa neustále počas celého dňa pri rôznych činnostiach [7, 3, 23]. 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 prispôbiť 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 spoločnosť 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 prispôbiť zobrazenie webového obsahu na ich malé displeje [13, 34].

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 [30] a k podobnému trendu prístupujú aj iní výrobcovia.

Optimalizácia webového obsahu zariadeniam však neznamená 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, rozširiteľnosť a priateľ-

skosť k budúcnosti [11]. 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 [34].  
 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 [28] existujú rôzne iné bežne používané jednotky „CSS pixel”, „device pixel” a „density-independent pixel” [20]. 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.

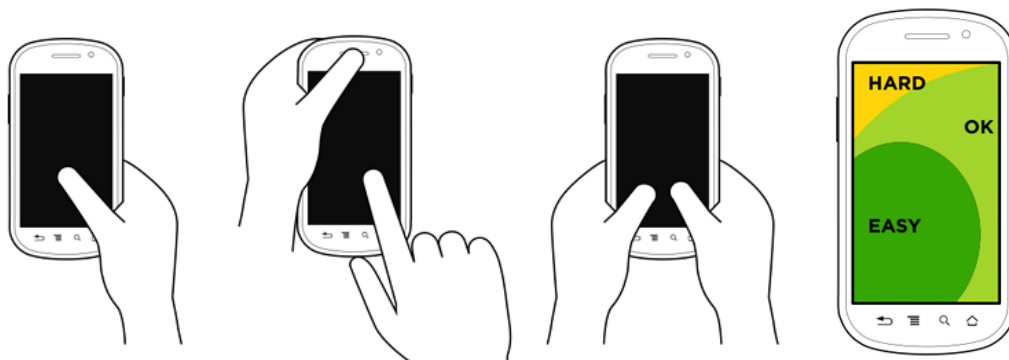
Viewport je celkové miesto potrebné na zobrazenie webovej stránky. Na mobilných zariadeniach je situácia komplikovanejšia, pretože stále existuje množstvo stránok, ktoré nie sú na ne optimalizované. Preto ho výrobcovia mobilných prehliadačov rozdelili na „layout viewport” a „visual viewport”. [20] Layout 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 množstva senzorov, ale hlavne s dotykovou plochou. 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 pomocou web kamery, natočanie a posúvanie zariadenia získané pomocou gyroskopu a accelerometra či počúvanie hlasových povelov zaznamenaných cez mikrofón. Je tak potrebné brať do úvahy aj ďalšie možnosti.

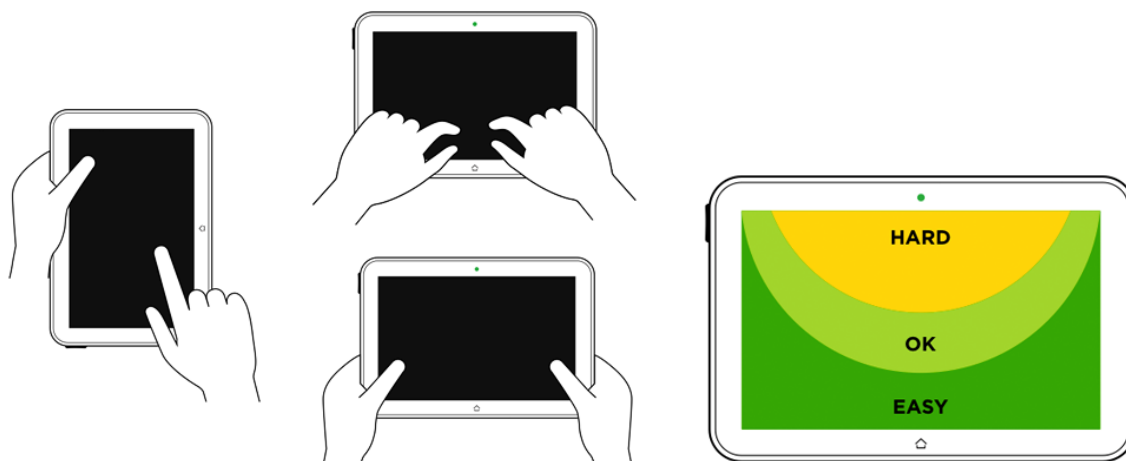
### **Dotyk**

Dôležitým prvkom v prípade mobilných zariadení je možnosť ovládania aplikácii 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 [8].



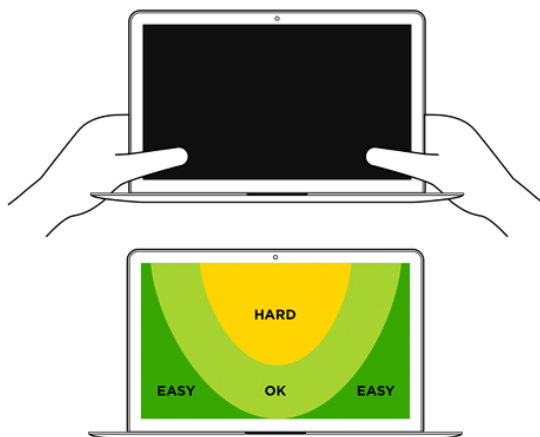
Obr. 2: Schopnosť ovládania mobilných telefónov [34].  
 Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

Väčšie mobilné zariadenia alebo tablety už nie je možné pohodlne udržať v jednej ruke a tak je dôležité prispôbiť ovládanie na dve ruky. Tablety sú držané v dvoch rukách za hranu a tak najlepšie dosiahnuteľné miesta sú na jeho okrajoch [8].



Obr. 3: Schopnosť ovládania tabletov [34].  
 Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

Podobné výsledky [8] sú aj pri novej kategórii zariadení, tzv. hybridných notebookov, ktoré okrem klávesnice obsahujú aj dotykový displej.



Obr. 4: Schopnosť ovládania hybridných počítačov [34].  
Prevzaté z <http://www.lukew.com/ff/entry.asp?1649>

## Reč

Ovládanie aplikácií a zariadení pomocou reči môže byť užitočné pre prípady spojené s indispozíciou používateľov najmä v spojení so slabším zrakom či fyzickým postihnutím [4].

Možnosť rozpoznania reči vo webovej aplikácii je v súčasnosti experimentálna novinka a jej štandardizácia je zatiaľ veľmi ďaleko. Používateľ samozrejme musí explicitne povoliť použitie mikrofónu pre webovú aplikáciu. Špecifikácia je zatiaľ len vo forme návrhu a nie je ani zaradená do W3C štandardu HTML5 [29]. Podpora v prehliadačoch s enginom Blink sa však už nachádza od začiatku roku 2013 a základná funkčnosť bola odprezentovaná v rámci konferencie Google IO 2013. Rozpoznanie reči prebieha vzdialene na serveroch patriacich Googlu a zatiaľ neexistuje možnosť rozpoznania lokálne [5].

S rozpoznávaním reči je spojená aj jeho syntéza, alebo preklad textu na reč. Tá je na tom v súčasnosti čo sa týka podpory a implementácie v prehliadačoch ešte horšie. Experimentálna funkčnosť existuje len v posledných buildoch prehliadačov. Našťastie sa táto funkcionála dá čiastočne nahradiť volaniami vzdialených webových služieb ako je napríklad „Google Translate“. Takáto syntéza však už neprebieha na zariadení, ale len sa zo serveru prijíma zvuková nahrávka, ktorá sa následne prehrá.

## Pohyb

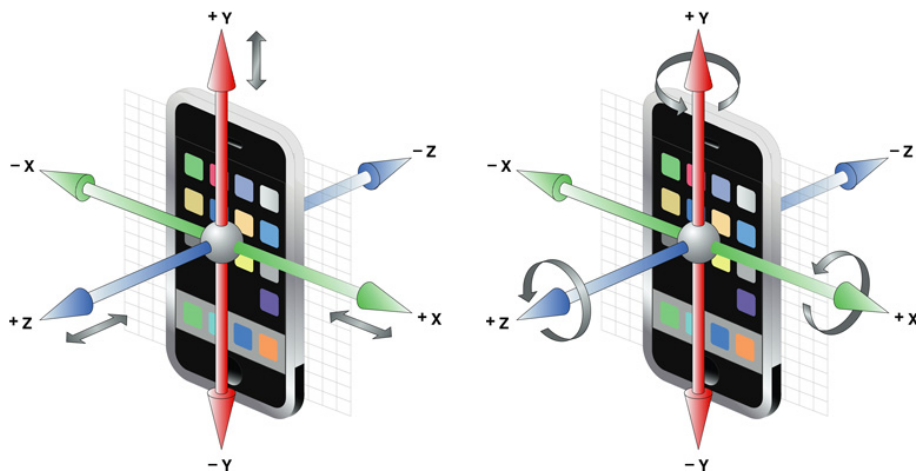
Rozpoznanie pohybu v reálnom čase je možné určiť podľa udalostí uskutočnených používateľom. Jednou z možností je použiť algoritmus na základe zmeny osvetlenia objektu [2] a tým určiť smer jeho pohybu.

Moderné webové prehliadače umožňujú získať prístup aj k video streamu z web kamery používateľa. Na prístup je rovnako potrebné povolenie od používateľa. Tento video stream prebieha vo zvolenej frekvencii a sa dá odchytiť a následne uložiť do elementu canvas, kde už sa k jednotlivým vzorkám pristupuje ako k obrázku. Je možné odfiltrovať okolie a zachovať len podstatné informácie na základe ktorých sa získa pohyb, respektíve gesto od používateľa.

## Accelerometer a Gyroskop

Medzi dlhodobo podporované senzory najmä v mobilných zariadeniach patria accelerometer a gyroskop. Je k nim umožnený prístup priamo z webovej aplikácie aj bez priameho povolenia od používateľa. Dáta z nich získané je zároveň možné použiť na vytvorenie a rozpoznanie giest použitých na ovládanie aplikácie [21].

Accelerometer slúži na získanie zrýchlenia zariadenia v osiach x, y a z, gyroskop na získanie uhlového zrýchlenia okolo týchto osí. Ich rozdiel je znázornený na nasledujúcom obrázku:



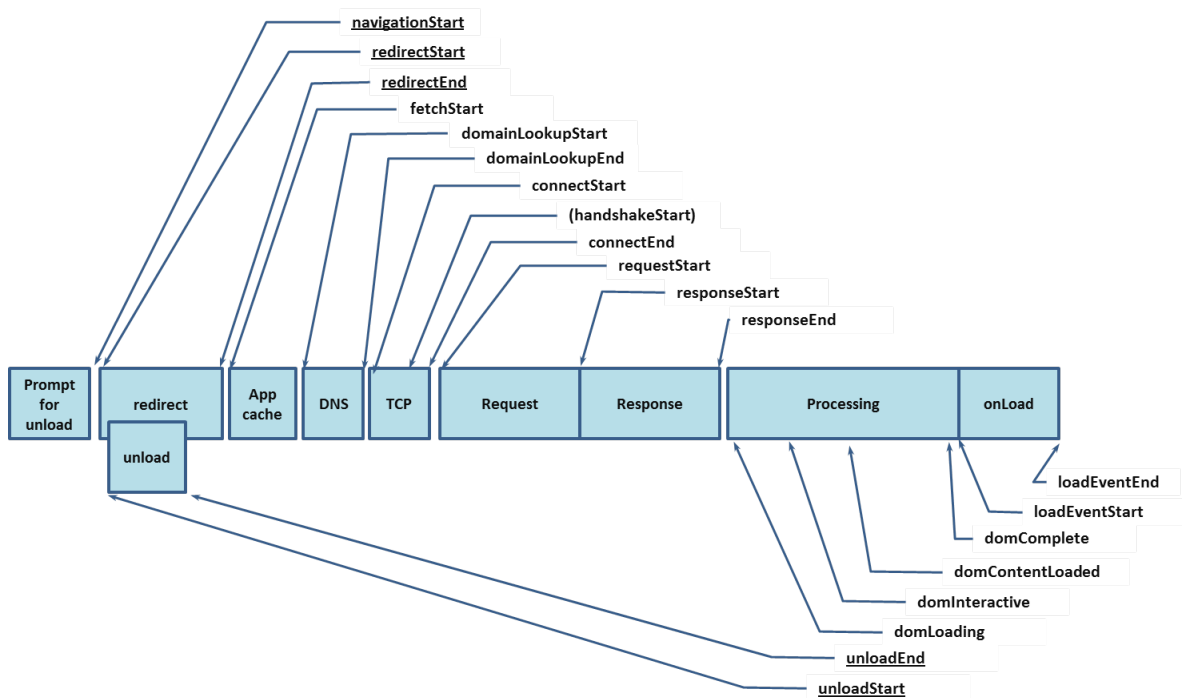
Obr. 5: Zaznamenávané údaje pomocou accelerometera a gyroskopu

### 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 načítaní, tak musíme šetriť množstvom prenášaných dát. Takéto šetrenie dát zároveň šetrí aj peňaženky používateľov [10], 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.



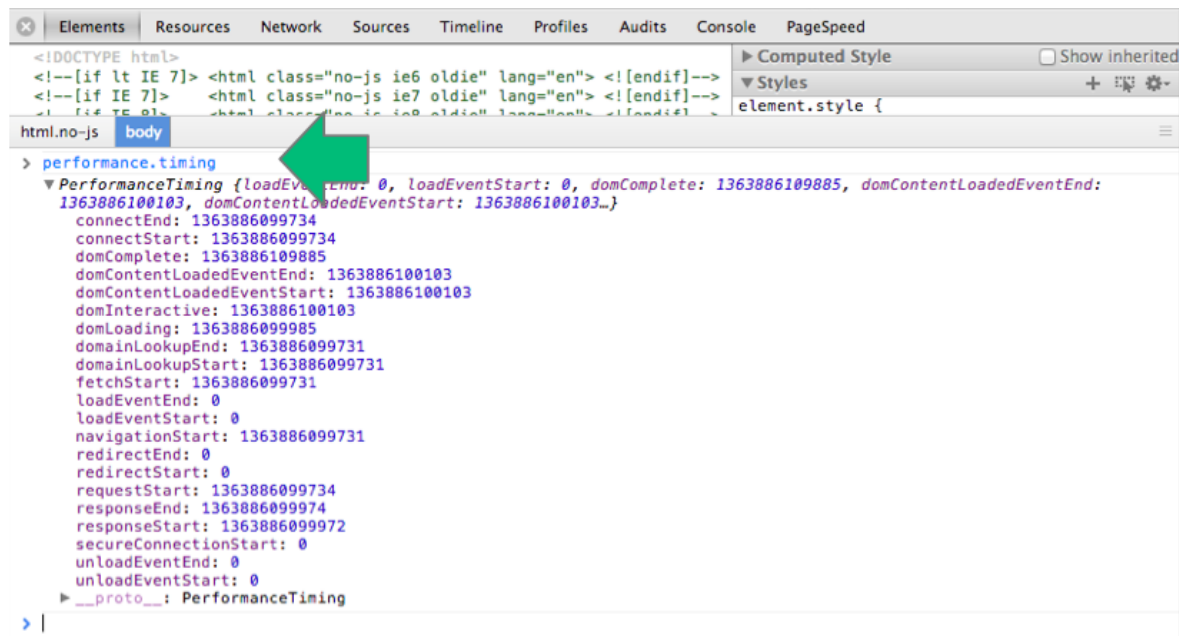
Obr. 6: Fázy spracovania požiadavky na server [27].

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 [14, 15]. Vďaka tomu máme dokonalejší prehľad o používatelovom pripojení a vieme mu tak prispôbiť jednotlivé komponenty stránky. W3C špecifikácia je vo fáze „Recommendation” [27], ale stále hlavnou nevýhodou je chýbajúca podpora v nie-



ktorých prehliadačov. Čiastočnou náhradou je meranie rozdielov dvoch časov, ale tak získame dáta len zo spracovania požiadavky na strane klienta.



Obr. 7: 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 [26]. 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 cieľenú 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 <sup>1</sup>

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 [17]*

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 [9], 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. [22, 25] Až neskôr bol označený ako metóda na dosiahnutie výsledku.

<sup>1</sup>Tejto 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 [33].

#### **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:**

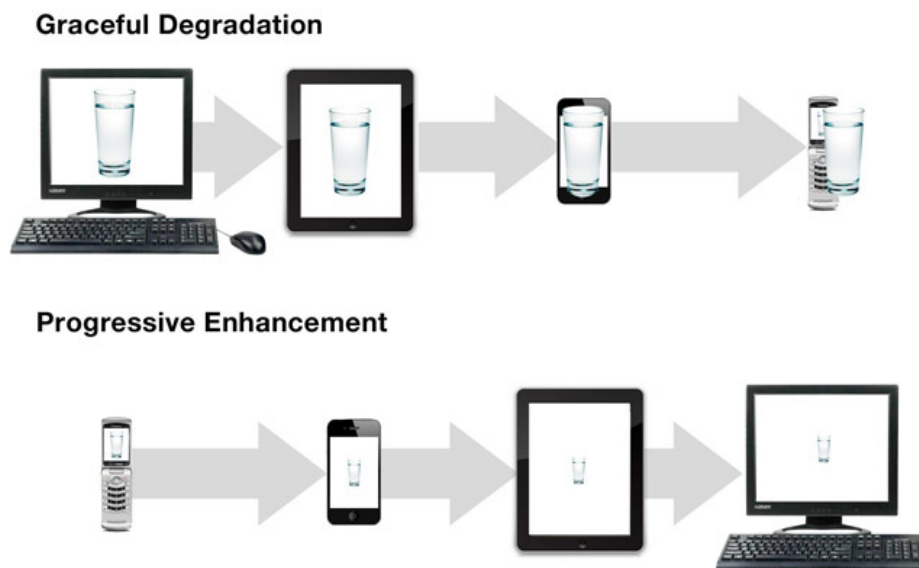
- úplne prispôbenie obsahu a vzhľadu, minimálne alebo žiadne dátové prenosy

**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. 8: Progressive enhancement vs Graceful degradation [12].

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 detekovanie<sup>2</sup>. 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.

<sup>2</sup>napríklad 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:**

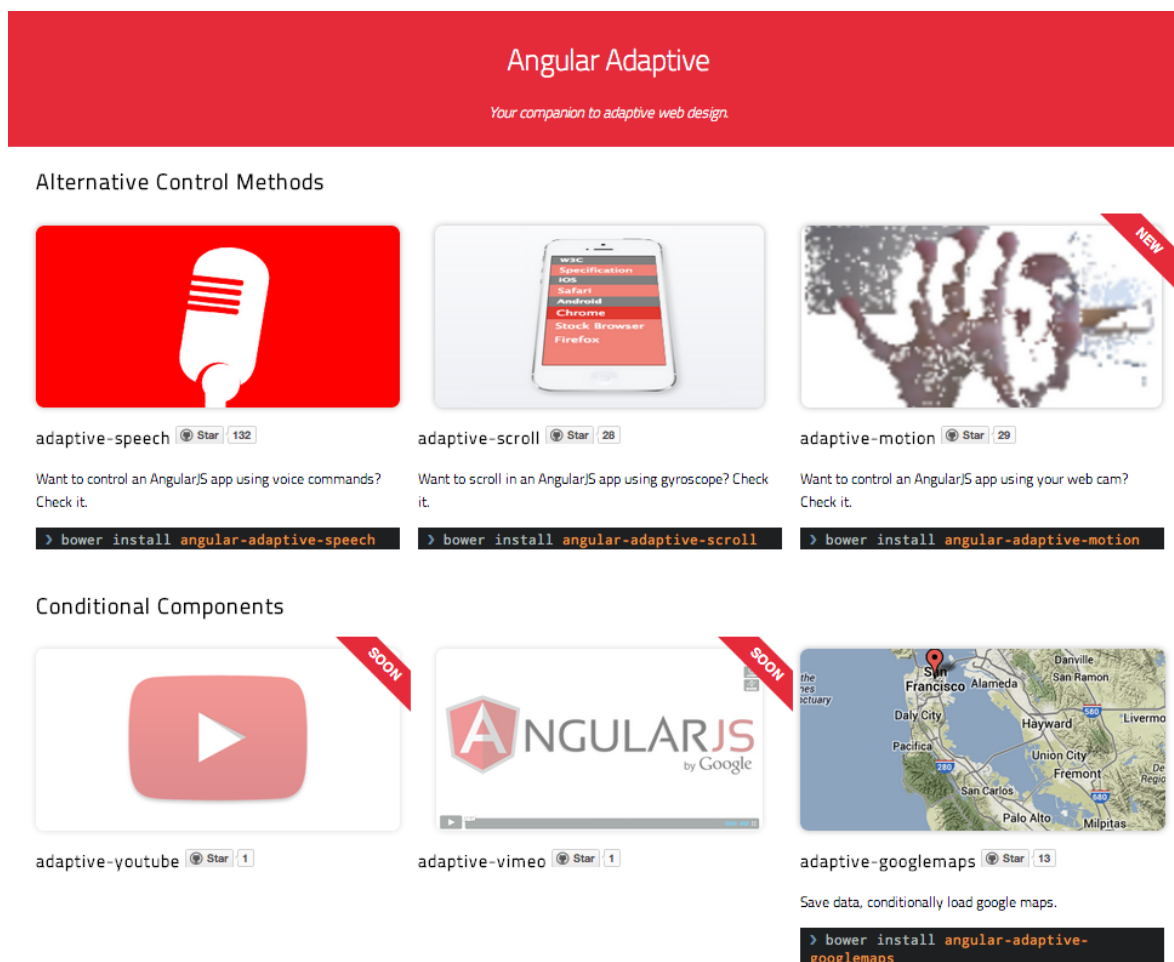
- jednoduchšia udržiavateľnosť, neexistujú rôzne verzie stránky ale len jedna

**Nevýhody:**

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

### 3 Nástroj na overenie

V tejto časti práce sa venujem návrhu jednotlivých súčastí systému. Ako nástroj na overenie analyzovaných metód sa vytvorila knižnica s názvom Angular-Adaptive <sup>3</sup> do populárneho JavaScriptového frameworku AngularJS <sup>4</sup>, kde sa následne skúmali možnosti adaptácie komponentov pomocou metódy progressive enhancement. Hlavným dôvodom výberu AngularJS bola možnosť vytvárania modulov, ktoré sa dajú následne jednoducho používať vo webových projektoch. Takáto podpora vytvárania externých modulov bude už o pár rokov natívna v prehliadačoch vďaka webovým komponentom [6]. Samotná distribúcia modulov prebieha pomocou balíčkovacieho nástroja Bower <sup>5</sup>, ktorý je v súčasnosti štandardným distribučným miestom webových komponentov.



Obr. 9: Webová stránka vytvoreného projektu angular-adaptive

<sup>3</sup>Angular-Adaptive <http://angular-adaptive.github.io/> je sprievodca vo svete adaptívneho web dizajnu.

<sup>4</sup>AngularJS <http://angularjs.org/> je populárny JavaScriptový framework, ktorý rozširuje HTML o nové možnosti pre webové aplikácie.

<sup>5</sup>Bower <http://bower.io/> je balíčkovací manažér pre web.

Celý systém je možné rozdeliť do troch na sebe navzájom nezávislých logických celkov skladajúcich sa z nasledujúcich častí:

1. alternatívne spôsoby ovládania webovej aplikácie, kde sa vytvorili knižnice na ovládanie pomocou reči, pohybu a gyroskopu
2. adaptácie webových komponentov použitých v aplikáciách, kde sa skúmali hlavne ich možnosti prispôbenia sa jednotlivým zariadeniam a internetovému pripojeniu používateľa
3. nástroje použité na implementáciu aplikácie

V nasledujúcich častiach kapitoly opisujem návrh implementácie jednotlivých súčastí systému.

### **3.1 Alternatívne spôsoby ovládania**

Príchod mobilných zariadení priniesol a spopularizoval nový spôsob interakcie - ovládanie zariadenia pomocou dotyku. Ten však vždy nemusí byť ten najvýhodnejší najmä v prostredí webových aplikácií. Súčasné moderné zariadenia obsahujú množstvo senzorov. Najmä mobilné zariadenia sú vybavené gyroskopom ktorý môže uľahčiť ovládanie aplikácie len vďaka nakláňaniu zariadenia do strán. Podobne môže reagovať zariadenie sledovaním okolia vďaka vstupu z webovej kamery, ktorá je vstavaná v množstve zariadení. Typickým senzorom v zariadeniach je mikrofón umožňujúci zaznamenať zvuk, z ktorého sa analyzujú rečové povely a tie vedú zefektívniť ovládanie aplikácie.

#### **3.1.1 Reč**

Ovládanie mobilných zariadení pomocou rečových povelov sa v posledných rokoch teší narastajúcej popularite. Rozpoznávanie je ale špecifické pre jednotlivé mobilné platformy. Javascriptové Web Speech API však umožňuje pridať rozpoznania reči aj do webovej aplikácie.

Cieľom je umožniť ovládanie aplikácie zadávaním statických povelov, ale aj ovládať dynamicky vytvárané entity v aplikácii nezávislé od zvoleného jazyka používateľa. Ovládanie aplikácie by malo prebiehať kontinuálne bez akejkoľvek ďalšej potrebnej interakcie používateľa. Na ich základe bol vytvorený modul ktorý umožňuje pridať kompletne ovládanie webovej aplikácie. Používateľ však musí povoliť prístup aplikácie k mikrofónu. Po jeho potvrdení už prebieha kontinuálne rozpoznávanie reči.

Po pridaní modulu do webovej aplikácie je potrebné si ho nakonfigurovať. Musia sa vytvoriť úlohy, ktoré definujú správanie aplikácie a prípadne aj reč, v ktorej prebieha rozpoznávanie. Implementované rozpoznávanie reči v prehliadači Google Chrome podporuje 32 rôznych jazykov a ďalšie prízvuky vrátane slovenčiny.



Rozpoznanie úloh prebieha na základe regulárneho výrazu definovaného v konfigurácii danej úlohy. Vďaka tomu je zaručené, že ako rozpoznaný text sa získa dynamicky reťazec s ktorým je možné ďalej v aplikácii pracovať.

Nakoľko je v aplikácii dôležitá viacjazyčná podpora je potrebné ju správne definovať rozlíšením kódu jazyka úlohy. Keď sa aktuálne nastavený jazyk rozpoznávania zhoduje s jazykom v úlohe a rozpoznaný text spĺňa podmienku regulárneho výrazu, tak sa zavolá asociovaná callback funkcia. Tá v parametri obsahuje rozpoznaný text v podobe textového reťazca, s ktorým sa dá ďalej pracovať. Príklad konfigurácie na pridanie novej úlohy má nasledujúci tvar:

---

```
1 'someTask': {
2   'regex': /^do .+/gi,
3   'lang': 'en-US',
4   'call': function(utterance){
5     // do something with utterance
6   }
7 }
```

---

Takýchto konfigurácií je možné pridať ľubovoľný počet. Takto navrhnutú konfiguráciu je možné zavolať globálne v rámci aplikácie alebo dynamicky vďaka podpore regulárneho výrazu a frameworku AngularJS pre dynamicky vytvárané elementy. V prípade globálneho volania stačí použiť samotnú konfiguráciu:

---

```
1 $speechRecognition.listenUtterance(
2   $scope.recognition['en-US']['someTask']
3 );
```

---

Pri dynamicky vytváraných elementoch je okrem samotnej konfigurácie s regulárnym výrazom potrebné pridať aj vstupný prvok elementu voči ktorému sa bude rozpoznávanie vykonávať. Takto sa vzájomne rozlíšia jednotlivé dynamické elementy a asociovaná úloha sa vykoná len pre jeden prvok.

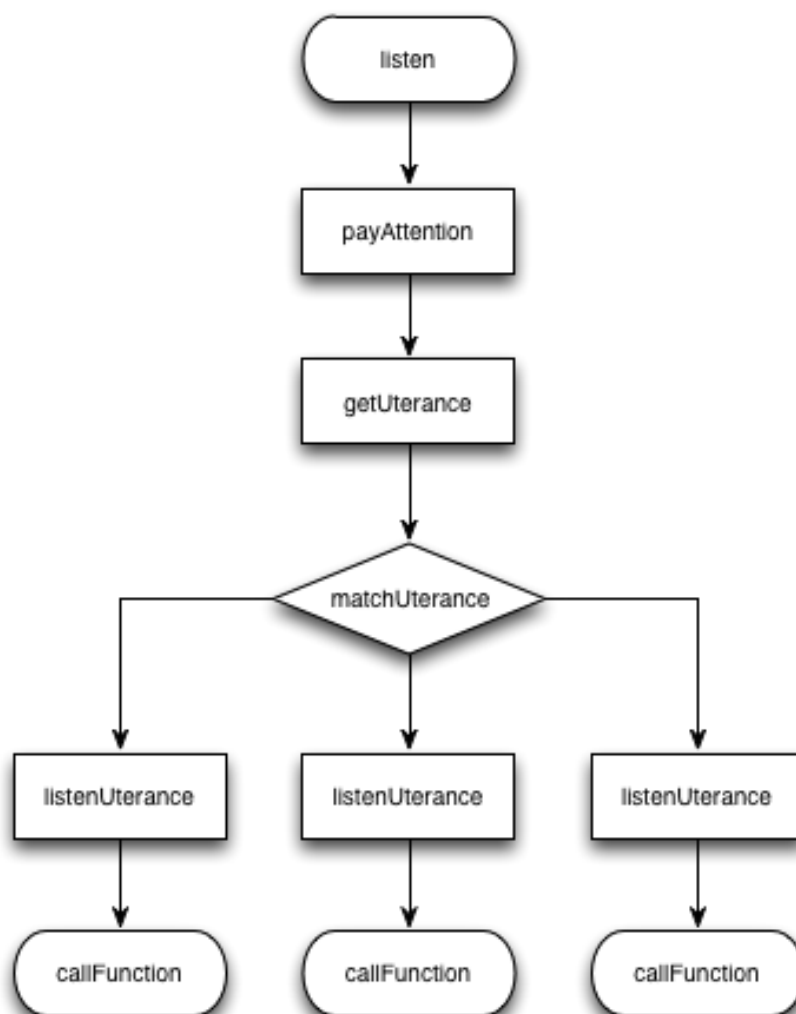
---

```
1 $scope.todos = ['buy milk', 'write essay'];
2
3 <li ng-repeat="todo in todos">
4   <div speechrecognition="{ 'tasks': recognition['en-US']['someTask'],
5     'input': todo }">{{todo}}</div>
6 </li>
```

---

Napríklad ak sa v aplikácii pracuje so zoznamom úloh, tak globálne je možné obsluhovať prídanie novej úlohy a dynamicky vytvárané elementy môžu mať vlastnú obsluhu, ktorá počúva na ich meno.

Niekedy je potrebné v aplikácii na istý čas ignorovať ovládanie pomocou reči. Kvôli takýmto prípadom sa pridala možnosť jeho nastavenia v podobe metódy `payAttention`. Diagram navrhnutého algoritmu ovládania webovej aplikácie pomocou reči sa nachádza na nasledujúcom obrázku:



Obr. 10: Algoritmus ovládania aplikácie pomocou reči

Použitie regulárneho výrazu pri konfigurácii úlohy má samozrejme aj nevýhodu. Najväčšou je potrebnosť presného určenia výrazu, nakoľko v opačnom prípade sa môžu detekovať aj nesprávne príkazy. Nevyhnutnosťou je aj stále pripojenie na internet kvôli vzdialenému rozpoznaniu reči, pričom samotné rozpoznanie nemusí byť vždy správne.

### 3.1.2 Pohyb

Ovládanie webovej aplikácie pohybom prebieha vďaka webovej kamere. Cieľom riešenia je rozpoznať základné smery pohybov používateľa a umožniť na ne namapovať funkcionality aplikácie, ktorá sa vykoná po ich správnom rozpoznaní.

Video sa sníma kontinuálne a zachytený video stream z kamery je s frekvenciou 60 fps. Každý jeden snímok je následne vykreslený do elementu canvas, v ktorom je možné s ním pracovať. Ten je tvorený množinou pixelov s rgba hodnotami.



Obr. 11: Vizualizácia videa zachyteného web kamerou

Získaný obrázok sa následne upraví pomocou hsv filtra aplikovaného na všetky pixely aby sa detekovali oblasti s kožou používateľa, pričom hsv filter je možné nakonfigurovať na vlastné hodnoty. Vďaka týmto oblastiam je rozpoznaný používateľ a je možné spracovávať jednotlivé smery pohybov. Upravený snímok je tvorený dvomi farbami: tmavou znamenajúcou kožu a bielou pre ostatné oblasti. Rozpoznanie prebieha nasledovne:

---

```
1  if ( (
2      (hsv[0] > hsvFilter.huemin && hsv[0] < hsvFilter.huemax) ||
3      (hsv[0] > hsvFilter.losemin && hsv[0] < losemax)
4  ) &&
5      (hsv[1] > hsvFilter.satmin && hsv[1] < hsvFilter.satmax) &&
6      (hsv[2] > hsvFilter.valmin && hsv[2] < hsvFilter.valmax)
7  ) {
8      skinFilter[pix] = r;
9      skinFilter[pix+1] = g;
10     skinFilter[pix+2] = b;
11     skinFilter[pix+3] = a;
12 }
```

---



Obr. 12: Vizualizácia videa po aplikovaní filtra detekcie kože

Ako je možné vidieť, detekcia kože nie je úplne správna a na základe hsv sa detekuju aj iné oblasti. Následne prebieha ďalšie upravovanie pri ktorom sa detekuju len samotné hrany. Ich detekcia prebieha na základe rozdielov farebnosti pixelov posledných dvoch snímok [2] s aplikovaným filtrom kože. Zistí sa celkový rozdiel v rgba hodnote každého pixelu a pokiaľ nastala výrazná zmena, tak sa detekuje ako hrana a zafarbí sa:

---

```

1 var rgbaDelta = Math.abs(draw.data[pix] - lastDraw.data[pix]) +
2   Math.abs(draw.data[pix+1] - lastDraw.data[pix+1]) +
3   Math.abs(draw.data[pix+2] - lastDraw.data[pix+2]);
4
5 if (rgbaDelta > treshold.rgb){
6   edge.data[pix] = 0;
7   edge.data[pix+1] = 0;
8   edge.data[pix+2] = 0;
9   edge.data[pix+3] = 255;
10 }
```

---

Opäť vznikne snímok tvorený dvomi farbami, ale tentokrát tmavá farba sa nachádza len na hranách a biela na ostatných častiach obrázku. Obrázok definuje miesta, na ktorých sa medzi snímkami vykonával pohyb. Na základe sekvencie takýchto obrázkov hrán vieme presnejšie určiť smer pohybu používateľa.



Obr. 13: Vizualizácia videa po aplikovaní filtra detekcie hrán

Keď je obrázok v už takomto stave tak je možné začať rozpoznávať predvádzané gesto. Spočíta sa celkový počet zmenených bodov a pomer zmenených bodov v osiach x a y obrázka. Postupným hromadením snímkov a spočítavaním všetkých zmenených bodov sa určí celkový smer pohybu. Celkovo sa rozpoznávajú štyri základné gestá, kde po úspešnom rozpoznaní sa vykonajú priradené funkcie:

- `onSwipeLeft` - pohyb zprava vľavo
- `onSwipeRight` - pohyb zľava vpravo
- `onSwipeUp` - pohyb zdola hore
- `onSwipeDown` - pohyb zhora dole

Príklad nastavenia funkcie na odchytenie gesta má nasledujúci tvar, kde sa asociuje callback funkcia, ktorá sa má následne v programe vykonať.

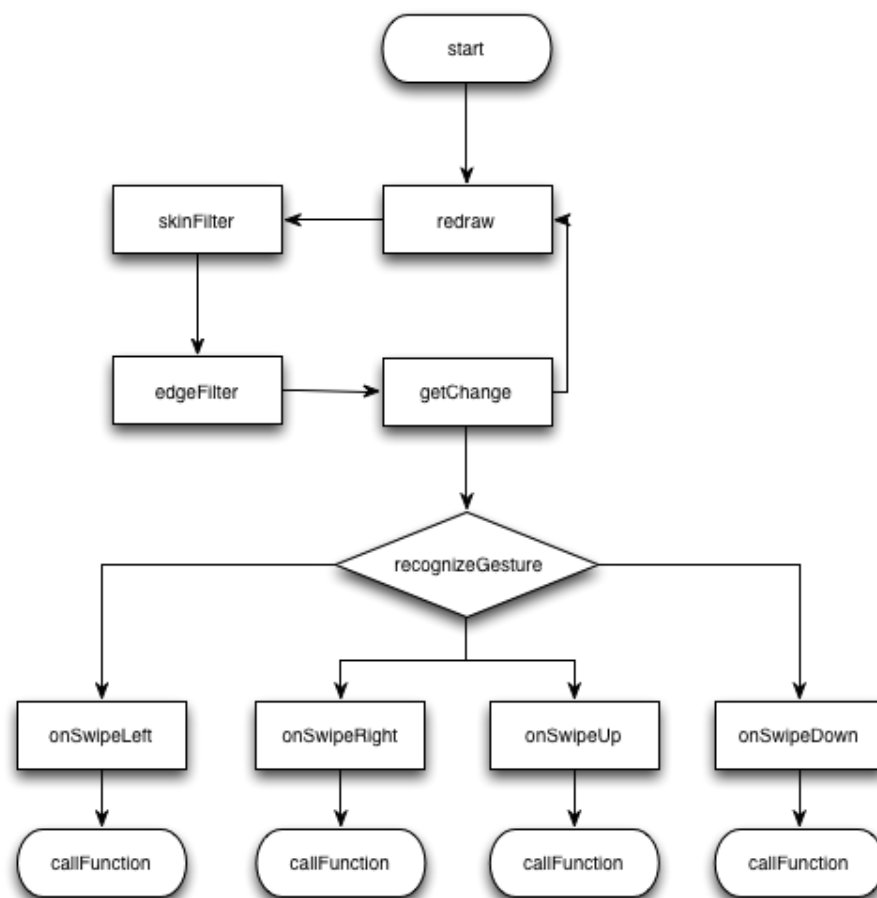
---

```
1 $motion.onSwipeRight(function(){  
2     // DO SOMETHING  
3 });
```

---

Citlivosť aplikovania filtrov a rozpoznávania gesta je samozrejme možné konfigurovať nastavením threshodu a úpravou hsv filtra. Problémy nastávajú najmä pri extrémnych vonkajších podmienkach, ktorými sú nízke alebo príliš vysoké osvetlenie. Tie sa však dajú eliminovať jeho zmenou.

Vytvorený modul podporuje aj ďalšie udalosti na ktoré je možné viazať funkcie, akými sú začiatok a koniec snímania, pretože na prístup k web kamere je potrebné explicitné povolenie od používateľa. Diagram navrhnutého algoritmu ovládania webovej aplikácie pomoc pohybu sa nachádza na nasledujúcom obrázku:



Obr. 14: Algoritmus ovládania aplikácie pomocou pohybu

### Vizualizácia

V aplikácii je možné pridať aj zobrazenie aktuálneho náhľadu z webovej kamery a to použitím vytvorenej direktívy pridaním atribútu „adaptive-motion” ku canvas elementu. Atribút môže nadobúdať nasledujúce hodnoty:

- video - zobrazí aktuálny náhľad z web kamery zariadenia, ktorého detail je zobrazený na obrázku 11
- skin - zobrazí video stream po aplikovaní filtra rozpoznania kože, ktorého detail je zobrazený na obrázku 12
- edge - zobrazí video stream po aplikovaní filtra rozpoznania hrán, ktorého detail je zobrazený na obrázku 13

### 3.1.3 Gyroskop

Jednou z možností ovládania webovej aplikácie je použitie gyroskopu. Na tento prípad som vytvoril modul pomocou ktorého je možné vo webovej aplikácii skrolovať len pomocou natáčania zariadenia do strán bez ďalšej potrebnej interakcie používateľa. Cieľom je zároveň umožniť ovplyvniť samotnú rýchlosť skrolovania na základe veľkosti natočenia oproti počiatočnej pozícii zariadenia.

Modul sa inicializuje pridaním atribútu „adaptivescroll” k elementu, v ktorom má prebiehať skrolovanie. Tým môže byť napríklad element „body”, ktorý zohľadňuje skrolovanie celej webovej stránky, alebo aj nejaký iný vnorený element. V takom prípade však skrolovanie prebieha len v rámci daného elementu.

---

```
1 <body adaptivescroll> </body>
```

---

Následne stačí zavaľať metódu so začatím sledovania polohy zariadenia, v ktorej sa môže upraviť ignorovaná hranica natočenia v uhloch a skrolovanie prebieha automaticky. V prípade potreby je toto sledovanie možné aj ukončiť.

Po začatí sledovania aktuálneho natočenia zariadenia sa uloží jeho počiatočná hodnota, ktorá sa používa ako relatívna hranica medzi stranami na ktoré prebieha skrolovanie. Postupným natáčaním zariadenia v smere hodinových ručičiek okolo osi beta od počiatočnej hodnoty k používateľovi sa vykonáva skrolovanie elementu smerom dolu, natáčaním na opačnú stranu od počiatočnej hodnoty prebieha skrolovanie hore.

Do úvahy pri zohľadňovaní rýchlosti skrolovania sa berie aj absolútna hodnota natočenia zariadenia od počiatočnej hodnoty zariadenia. Čím je táto hodnota väčšia, tým rýchlejšie prebieha skrolovanie. Výpočet rozdielu natočenie zohľadňuje smer rotácie a je určený na základe počiatočnej rotácie, aktuálneho natočenia a celkovej veľkosti rotácie pre danú os:

---

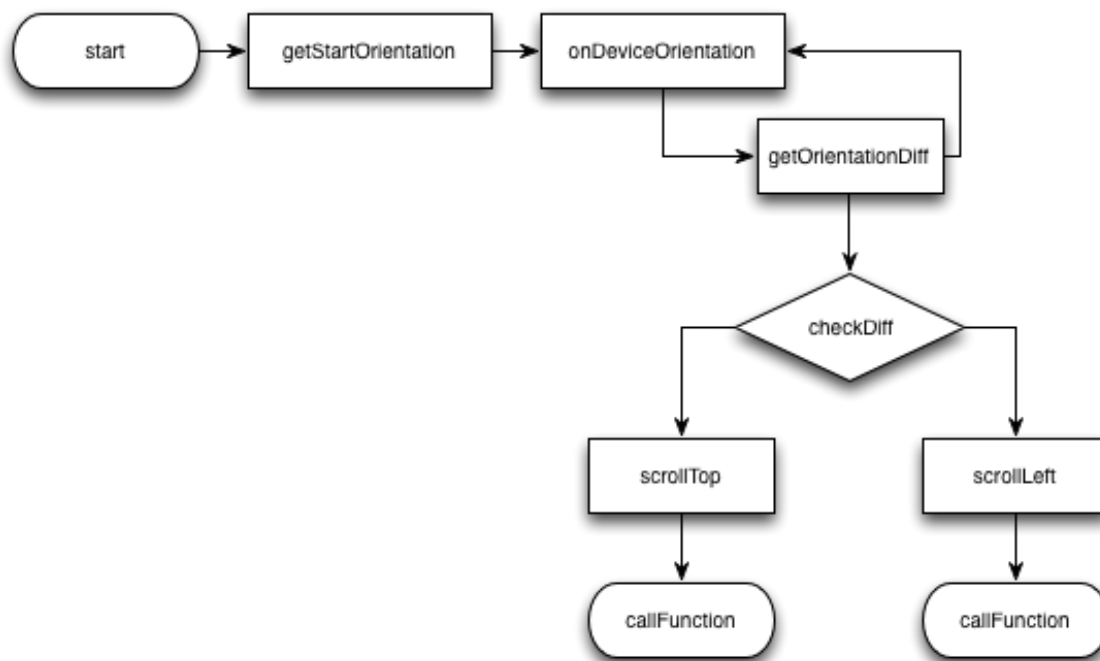
```
1 var getDiff = function(startOrientation, actualOrientation, interval){
2   var d1 = startOrientation - actualOrientation;
3   var d2 = startOrientation - actualOrientation - interval;
4   return (Math.abs(d1) < Math.abs(d2)) ? d1 : d2;
5 };
```

---

Samotný proces skrolovania prebieha vďaka číselnej zmene offsetu elementu. Plynulosť skrolovania je zaručená použitím animácii pomocou requestAnimationFrame. Tá prebieha s frekvenciou 60 fps a je časovaná priamo prehliadačom. Tej však chýba podpora v straších verziách, v takom prípade musí byť použitý manuálny časovač aplikácie.

Okrem skrolovania modul obsahuje aj ďalšie metódy onalpha, onbeta a ongamma, ktorých návratová funkcia sa zavolá vždy po zmene rotácie zariadenia a je ich možné použiť

na iné potreby v aplikácii. Funkcia obsahuje parameter absolútnu zmenu pozície v uhloch od počiatočnej k aktuálnej, ktorá sa používa aj pri skrolovaní v aplikácii. Diagram navrhnutého algoritmu ovládania webovej aplikácie pomocou gyroskopu sa nachádza na nasledujúcom obrázku:



Obr. 15: Algoritmus ovládania aplikácie pomocou gyroskopu

Po vytvorení modulu a testovaní na zariadeniach sa zistili problémy v skutočnej implementácii udalostí natáčania zariadenia v rôznych prehliadačoch. W3C špecifikácia orientácie zariadenia uvádza nasledovné:

- os X je rovnobežná na rovinu zeme, kladné hodnoty má v smere na východ, záporné na západ
- os Y je rovnobežná na rovinu zeme, kladné hodnoty má v smere na sever, záporné na juh
- os Z je kolmá na rovinu zeme, kladné hodnoty má v smere od zeme, záporné smerom k zemi

Rotácia by mala byť popísaná pravidlom pravej ruky, teda kladné hodnoty rotácie sú v smere hodinových ručičiek okolo osí.

Nasledujúce tabuľky ukazujú, kde sa nachádzajú nulové body pre jednotlivé osi, aké hodnoty nadobúda rozsah rotácie a či je dodržané pravidlo pravej ruky.



Pravidlo pravej ruky (PPR) hovorí:

*„Kladné hodnoty rotácie sa zvyšujú pri rotovaní okolo osi v smere hodinových ručičiek pri ukazovaní smerom na narastajúcu kladnú hodnotu smerovej osi.” W3C [24]*

Táto definícia je však protichodná s hodnotami kompasu. Pri rotácii okolo osi Z v smere rotácie kompasu sa hodnoty rotácie znižujú a nie narastajú. To je však spôsobené pohľadom do záporných hodnôt osi Z.

## Alpha

Rotácia okolo osi Z nadobúda hodnoty rotácie alpha. V špecifikácii je nejasne definované aké by mali byť východiskové hodnoty, čo vo výsledku prináša rôzne implementácie rotácie v prehliadačoch. To spôsobuje zmätok nakoľko zo špecifikácie nie je jasné, na akú stranu by mal ukazovať nulový bod. Dá sa odpozorovať podľa pravidla pravej ruky, že 0 stupňov ukazuje smerom na sever, pretože implementovaných 90 stupňov ukazuje na východ.

Rozsah hodnôt bol testovaný pri držaní zariadenia v horizontálnej polohe a vycentrovaní smerom k nulovému bodu, keď hodnota rotácie bola 360 stupňov. Pozorované boli nasledujúce hodnoty:

	Nulový bod	PPR	Rozsah
Reference	Sever (0)	A	[0, 360]
iOS Chrome	Východ (90)	A	[0, 360]
iOS Safari	Východ (90)	A	[0, 360]
Android			
Chrome	Sever (0)	A	[0, 360]
Stock	Západ (270)	A	[0, 360]
Firefox	Sever (0)	N	[0, 360]

Tabuľka 1: Alpha rotácia gyroskopu

## Beta

Rotácia okolo osi X nadobúda hodnoty rotácie beta. Špecifikácia definuje nulový bod ako horizontálnu polohu. Všetky testované prehliadače majú implementovaný smer osi rotácie správne, problémy však nastávajú pri rozsahu rotačných hodnôt.

Rozsah bol testovaný pri držaní zariadenia v horizontálnej polohe, čo je nulový bod. Zariadenie bolo potom otáčané okolo osi X o 90 stupňov tak, že displej zariadenia smeroval k pozorovateľovi. Potom nasledovalo otáčanie o ďalších 90 stupňov tak, že displej bol smerom k zemi. Na koniec nasledovala rotácia zvyšných 180 stupňov smerom do počiatočnej polohy, čím sa rotácia dokončila.

	Nulový bod	PPR	Rozsah	Poznámky
Reference	Horizontálne	A	[0, -180 180]	
iOS Chrome	Horizontálne	A	[-90, 90]	Plný rozsah rotácie nie je podporovaný.
iOS Safari	Horizontálne	A	[-90, 90]	Plný rozsah rotácie nie je podporovaný.
Android				
Chrome	Horizontálne	A	[-90, 90]	Plný rozsah rotácie nie je podporovaný.
Stock	Horizontálne	A	[-90, 90]	Plný rozsah rotácie nie je podporovaný.
Firefox	Horizontálne	N	[0, 180 -180]	Rotácie pokračuje naspäť na začiatok

Tabuľka 2: Beta rotácia gyroskopu

V iOS zariadeniach rovnako ako v Android prehliadači a Chrome pre Android hodnoty rotácie od počiatku stúpajú správne. Ale keď rotácia nadobudne svoje maximum, čo je 90 a -90 stupňov, tak jej hodnoty začnú postupne klesať. Nie je tak podporovaný celý rozsah rotácie.

Vo Firefoxe pre Android je rotácia implementovaná opačným smerom a nie je dodržané špecifikované pravidlo pravej ruky. Je však podporovaný celý rozsah rotácie. Najskôr klesne k hodnote -180 stupňov, čo sa pri pokračujúcej rotácii zmení na kladnú hodnotu a potom klesá k nule.

## Gamma

Rotácia okolo osi Y nadobúda hodnoty rotácie gamma. Špecifikácia definuje nulový bod v horizontálnej polohe. Ale aj v tomto smere rotácie sú problémy s rozsahom rotácie, ale to je spôsobené najmä samotnou špecifikáciou zo strany W3C, nakoľko v nej sa nachádza rozsah len od -90 po 90 stupňov, čo je celkovo 180 a nie je tak pokryté celé otočenie.

Rozsah hodnôt bol testovaný pri držaní zariadenia v horizontálnej polohe a natočením na nulový bod. Zariadenie bolo potom otáčané okolo osi Y o 90 stupňov v smere hodinových ručičiek tak, že jeho displej smeroval vpravo. Potom nasledovalo otočenie o ďalších 90 stupňov tak, že zariadenie smerovalo dolu. Na koniec sa dokočilo zostávajúcich 180 stupňov rotácie do pôvodného stavu.

	Nulový bod	PPR	Rozsah	Poznámky
Reference	Horizontálne	A	[0, 90 -90]	Zlá definícia od W3C
iOS Chrome	Horizontálne	A	[0, 180 -180]	Plný rozsah rotácie nie je podporovaný
iOS Safari	Horizontálne	A	[0, 180 -180]	Plný rozsah rotácie nie je podporovaný.
Android				
Chrome	Horizontálne	A	[0, 270 -90]	Upravený rozsah rotácie definuje natočenie
Stock	Horizontálne	A	[0, 270 -90]	Upravený rozsah rotácie definuje natočenie
Firefox	Horizontálne	N	[0, -90 90]	Rozsah rotácie pokračuje naspäť na začiatok

Tabuľka 3: Gamma rotácia gyroskopu

Zlá definícia rotácie od W3C špecifikuje rotáciu len po hodnotu 90 stupňov z horizontálnej polohy, čo spôsobuje problémy pri väčšom natočení zariadenia.

V iOS rotácia nadobúda nulový bod v horizontálnej podobe s displejom smerujúcim hore. Otočením zariadenia okolo osi Y tak, že obrazovka bude smerovať smerom dole bude aktuálne gamma natočenie nadobúdať hodnotu 180, respektíve -180 stupňov. Ak sa rotácia uskutočňuje v smere hodinových ručičiek, tak hodnoty rotácie sa postupne zvyšujú v kladných číslach. Ak rotácia ide opačným smerom proti smeru hodinových ručičiek, tak hodnoty rotácie sa zmenšujú. Pri natočení zariadenia smerom na bok lavou strano hore je hodnota rotácia 90 stupňov, na opačnej strane pri smerovaní pravej hrany smerom hore je to -90 stupňov.

Chrome pre Android a vstavaný Android prehliadač majú správne hodnoty rotácie od -90 po +90 stupňov. Ale po natočení zariadenia o viac ako 90 stupňov v smere hodinových ručičiek hodnota rotácie naďalej stúpa až po hodnotu 270 stupňov, čo je zároveň aj hodnota -90. Toto poskytuje možnosť presne vedieť veľkosť natočenia zariadenia.

Firefox má aj pri rotácii okolo osi Y podobný problém, a to je že hodnoty sú opačné a nedodržiava špecifikované pravidlo pravej ruky. Rozsah je správny, ale rotácie v smere hodinových ručičiek nadobúdajú záporné hodnoty a naopak.

## Gyrocopter

Na vyriešenie problémov pri vývoji webových aplikácií spôsobených rôznymi implementáciami rotácii v prehliadačoch som vytvoril rozšírenie Gyrocopter pre prehliadač Chrome <sup>6</sup>. Hlavným cieľom rozšírenia je umožniť zvoliť si simuláciu konkrétneho prehliadača a dynamicky meniť rotáciu zariadenia bez potreby testovania funkcionality na reálnom zariadení. Podobná funkcionality sa už nachádza v developerských nástrojoch prehliadača Chrome, ale tam je možné len manuálne zadať hodnoty rotácie bez znalosti skutočných rozsahov a náhľadu natočenia zariadenia.

<sup>6</sup>Dostupné na stiahnutie z Chrome Web Store <https://chrome.google.com/webstore/detail/gyrocopter/oooalfgemajfc1liinfcdkifafmcfjop>

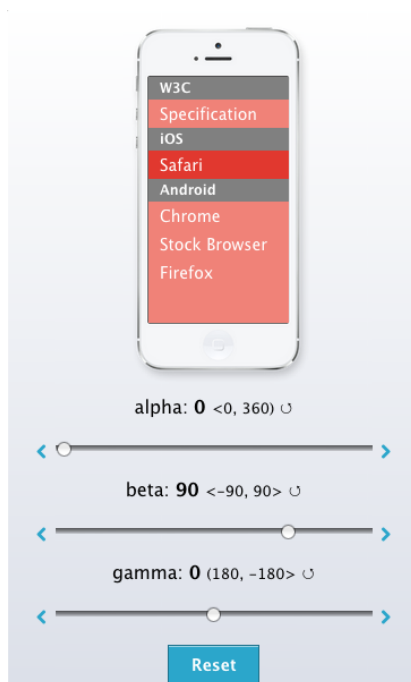
Toto rozšírenie umožňuje simulovať udalosti ktoré nastanú pri rotácii zariadenia, pričom používateľ môže natáčať zariadenie vo všetkých troch osiach. Rozšírenie je implementované pre vývojárov priamo do prostredia „Chrome Developer Tools”.

Nakoľko hodnoty rotácii sú implementované v prehliadačoch rôzne, tak používateľ má na výber zvoliť si podľa akej implementácie chce vyvolávať udalosti rotácie. Simulovanými prehliadačmi na výber sú:

- W3C špecifikácia
- iOS Safari (Chrome používa web view)
- Android Chrome
- Android prehliadač
- Firefox pre Android

Samotné používateľské rozhranie rozšírenia obsahuje náhľad aktuálnej rotácie zariadenia na ktorom je možné zmeniť aktuálne simulovaný prehliadač. Pod ním sa nachádzajú uhly natočenia alpha, beta a gamma, ktoré je možné meniť. Tieto obsahujú pre jednotlivé prehliadače rozsahy a smer natočenia. Pri zmene rotácie na jednotlivých osiach sa zmena automaticky prejaví aj na náhlade zariadenia.

Toto rozhranie navrhnutého rozšírenia je znázornené na nasledujúcom obrázku:



Obr. 16: Gyrocopter - gyroskop simulátor

Po výbere, respektíve zmene prehliadača, ktorý sa má simulovať, sa automaticky prepočítajú rozsahy pre jednotlivé uhly otáčania. Zároveň sa upraví na nich aj aktuálna hodnota natočenia bez toho, aby sa musela znázornená rotácia zariadenia meniť.

## 3.2 Komponenty







Pri adaptívnom dizajne sú dôležité aj webové komponenty s ktorými používateľ webovej aplikácie priamo interaguje. Tieto boli skúmané najmä z hľadiska optimalizácie na rôzne zariadenia a rýchlosti pripojenia.

### 3.2.1 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:

 <b>kxopViU98Xo?autoplay=0</b> www.youtube.com/embed	GET	200 OK	text/html	Other	3.2 KB 5.2 KB	404 ms 400 ms
 <b>www-embed-vflidKrj4Q.css</b> s.ytimg.com/yts/cssbin	GET	200 OK	text/css	<a href="#">kxopViU98Xo:7</a> Parser	26.7 KB 145 KB	124 ms 58 ms
 <b>www-embed-vflDpvlm5.js</b> s.ytimg.com/yts/jsbin	GET	200 OK	text/javascr...	<a href="#">kxopViU98Xo:20</a> Parser	32.4 KB 84.5 KB	156 ms 81 ms
 <b>watch_as3-vflFZFBB0.swf</b> s.ytimg.com/yts/swfbin	GET	200 OK	application...	Other	282 KB 282 KB	530 ms 56 ms
 <b>crossdomain.xml</b> i4.ytimg.com	GET	200 OK	text/x-cros...	<a href="#">www-embed-vflDpvlm5</a> Script	478 B 102 B	90 ms 89 ms
 <b>hqdefault.jpg</b> i4.ytimg.com/vi/kxopViU98Xo	GET	200 OK	image/jpeg	Other	10.8 KB 10.5 KB	68 ms 52 ms

Obr. 17: 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.

	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. 18: 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 Modernizr<sup>7</sup> 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.

---

<sup>7</sup>Modernizr <http://modernizr.com/> je JavaScriptová knižnica, ktorá detekuje dostupnosť natívnej implementácie nových technológií v prehliadači.

### 3.2.2 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 maps<sup>8</sup>, 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/`

### 3.2.3 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.

---

<sup>8</sup><https://developers.google.com/maps/documentation/staticmaps/>



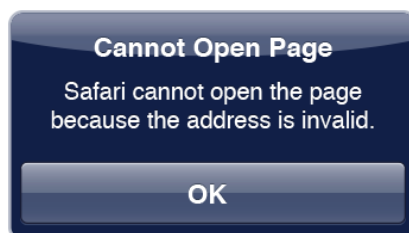
### 3.2.4 Otvorenie v aplikácii / Stiahnutie aplikácie

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 [16]*

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.

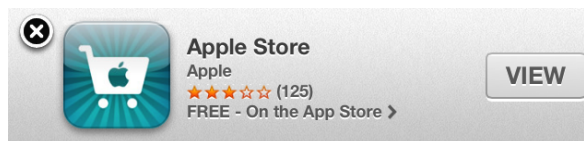
Prepájanie aplikácii sa uskutočňuje pomocou „custom url schémy” [19], ktorá je špecifická pre jednotlivé operačné systémy zariadení. Nevýhodou vlastných odkazov je, že pokiaľ používateľ nemá nainštalovanú aplikáciu, tak po kliknutí sa objaví celkom nepekná chybová hláška.



Obr. 19: Chybová hláška po otvorení custom URL schémy, pokiaľ aplikácia neexistuje [32].  
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. 20: Otvorenie natívnej aplikácie v iOS 6 [18].  
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, prípadne volať viacero natívnych aplikácií, tak toto riešenie je nepostačujúce. Nemôže byť upravované 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 programátorský 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 internetovej 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.

### 3.3 Nástroje

#### 3.3.1 Detekcia

Pre potreby rozpoznávania platformy v aplikácii bol vytvorený modul detekcie podľa jedinečného „user agent” textu. Modul umožňuje detekovať iOS a Android zariadenia. Rozpoznávanie prebieha na základe regulárneho výrazu, kde pri metóde detekcie iOS zariadenia sa kontroluje výskyt slov „iPhone”, „iPad” alebo „iPod”, ktoré jedinečne definujú platformu. Pri Android zariadeniach je detekcia jednoduchšia, pretože stačí ak sa v identifikačnom texte nachádza slovo „Android”. Príklad metódy detekcie má nasledujúci tvar:

---

```
1 var isAndroid = function(userAgent){  
2   return /(Android)/gi.test(userAgent);  
3 }
```

---

Pri konfigurácii webovej aplikácie je možné aj aktuálny „user agent” text nahradiť vlastnou hodnotou, takže prehliadač sa vždy môže správať ako požadované zariadenie. Problém pri detekcii môže nastať, pokiaľ by user agent už bol raz nahradený používateľom, respektíve by sa prestali používať súčasný identifikátory. Potom by sa detekovala nesprávna platforma.

**4 Overenie**

**5 Záver**

## Literatúra

- [1] ANTALA, J. Tvorba bohatých internetových aplikácií pre mobilné zariadenia. Bakalárska práca, FIIT STU, Bratislava, 2012.
- [2] AVGERINAKIS, K. – BRIASSOULI, A. – KOMPATSIARIS, I. Real Time Illumination Invariant Motion Change Detection. In *Proceedings of the First ACM International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Streams*, ARTEMIS '10, pp. 75–80, New York, NY, USA, 2010. ACM. doi: 10.1145/1877868.1877887. Dostupné z: <http://doi.acm.org/10.1145/1877868.1877887>. ISBN 978-1-4503-0163-3.
- [3] 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.
- [4] BASSON, S. – FAIRWEATHER, P. G. – HANSON, V. L. Speech Recognition and Alternative Interfaces for Older Users. *interactions*. Júl 2007, 14, 4, pp. 26–29. ISSN 1072-5520. doi: 10.1145/1273961.1273980. Dostupné z: <http://doi.acm.org/10.1145/1273961.1273980>.
- [5] BIDELMAN, E. A More Awesome Web. In *Google IO 2013 conference*, Moscone Center, San Francisco, CA, USA, 2013.
- [6] BIDELMAN, E. Web Components: A Tectonic Shift for Web Development. In *Google IO 2013 conference*, Moscone Center, San Francisco, CA, USA, 2013.
- [7] 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.
- [8] CLARK, J. Designing For Touch. In *The Mobile Book*, Freiburg, Germany, 2012. Smashing Magazine. ISBN 9783943075502.
- [9] 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>.
- [10] 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>.

- [11] FROST, B. Beyond Media Queries: Anatomy of an Adaptive Web Design. In *An Event Apart conference*, Washington DC, USA, August 6, 2012.
- [12] FROST, B. Beyond Squishy: The Principles of Adaptive Design. In *SXSW conference*, Austin, TX, USA, March 9, 2013.
- [13] FROST, B. Responsive Design Patterns. In *The Mobile Book*, Freiburg, Germany, 2012. Smashing Magazine. ISBN 9783943075502.
- [14] GRIGORIK, I. Breaking the 1000 ms Time to Glass Mobile Barrier. In *SF HTML5*, San Francisco, CA, USA, Mar 22, 2013.
- [15] GRIGORIK, I. High Performance Browser Networking. O'Reilly Media, Incorporated, 2013. ISBN 9781449344764.
- [16] 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/>.
- [17] 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/>.
- [18] 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>.
- [19] iOS Developer Library. Implementing Custom URL Schemes. Technical report, Safari Web Content Guide, . Dostupné z: [http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/AdvancedAppTricks/AdvancedAppTricks.html#//apple\\_ref/doc/uid/TP40007072-CH7-SW50](http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/AdvancedAppTricks/AdvancedAppTricks.html#//apple_ref/doc/uid/TP40007072-CH7-SW50).
- [20] KOCH, P.-P. A pixel is not a pixel. In *Frontiers conference*, Amsterdam, Netherlands, Oct 4, 2012.
- [21] KRATZ, S. – ROHS, M. – ESSL, G. Combining Acceleration and Gyroscope Data for Motion Gesture Recognition Using Classifiers with Dimensionality Constraints. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces, IUI '13*, pp. 173–178, New York, NY, USA, 2013. ACM. doi: 10.1145/2449396.2449419. Dostupné z: <http://doi.acm.org/10.1145/2449396.2449419>. ISBN 978-1-4503-1965-2.
- [22] 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.

- [23] 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.
- [24] W3C. DeviceOrientation Event Specification. In *Editor's Draft*, 13 June 2012. Dostupné z: <http://dev.w3.org/geo/api/spec-source-orientation.html>.
- [25] W3C. Media Queries. In *W3C Recommendation*, 19 June 2012. Dostupné z: <http://www.w3.org/TR/css3-mediaqueries/>.
- [26] W3C. The Network Information API. In *W3C Working Draft*, 29 November 2012. Dostupné z: <http://www.w3.org/TR/netinfo-api/>.
- [27] W3C. Navigation Timing. In *W3C Recommendation*, 17 December 2012. Dostupné z: <http://www.w3.org/TR/navigation-timing/>.
- [28] W3C. CSS Values and Units Module Level 3. In *W3C Candidate Recommendation*, 4 April 2013. Dostupné z: <http://www.w3.org/TR/css3-values/>.
- [29] W3C. Web Speech API Specification. In *Specification Draft*, 19 June 2012. Dostupné z: <https://dvcs.w3.org/hg/speech-api/raw-file/tip/speechapi.html>.
- [30] 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>.
- [31] 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>.
- [32] 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>.
- [33] 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.
- [34] 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>.

## A Obsah CD nosiča

K práci je priložené CD, na ktorom sa nachádzajú implementované moduly spolu s demonstračnými aplikáciami a dokumentácia. Jeho súborová štruktúra je nasledujúca:

- **/angular-adaptive/**  
implementované moduly
- **/angular-adaptive/adaptive-speech/**  
modul na ovládanie aplikácie pomocou reči  
<https://github.com/angular-adaptive/adaptive-speech>
- **/angular-adaptive/adaptive-scroll/**  
modul na ovládanie aplikácie pomocou gyroskopu  
<https://github.com/angular-adaptive/adaptive-scroll>
- **/angular-adaptive/adaptive-motion/**  
modul na ovládanie aplikácie pomocou pohybu  
<https://github.com/angular-adaptive/adaptive-motion>
- **/angular-adaptive/adaptive-detection/**  
modul na detekovanie zariadenia  
<https://github.com/angular-adaptive/adaptive-detection>
- **/angular-adaptive/adaptive-googlemaps/**  
webový komponent google máp  
<https://github.com/angular-adaptive/adaptive-googlemaps>
- **/angular-adaptive/angular-adaptive.github.io/**  
webová stránka projektu  
<https://github.com/angular-adaptive/angular-adaptive.github.io>
- **/Gyrocopter/**  
extension na emulovanie gyroskopu do Chrome Developer Tools  
<https://github.com/janantala/Gyrocopter>
- **/Dokumentacia/**  
dokument a anotácie v slovenskom a anglickom jazyku