Masarykova univerzita
Fakulta informatiky

Vývoj pre inteligentný mobilný telefón pomocou aplikačného rámca PhoneGap (Apache Cordova)

Bakalárska práca

Marek Mihálech

Brno, jar 2015

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Marek Mihálech

Vedúci práce: Mgr. Juraj Michálek

Kľúčové slová

PhoneGap, Cordova, multiplatformný vývoj, hybridné aplikácie, cross–platform, HTML5, CSS, Nofrack, Frack Attack, left–to–right, right–to–left

Poďakovanie

Chcel by som poďakovať vedúcemu práce Mgr. Jurajovi Michálkovi a konzultantovi práce RNDr. Andriymu Stetskovi, Ph.D za odborné rady a spoluprácu.

Zhrnutie

Cieľom práce je vytvoriť aplikáciu pre inteligentné mobilné telefóny pomocou aplikačného rámca PhoneGap a Apache Cordova. Aplikácia musí byť spustiteľná na operačných systémoch Android, iOS a Windows Phone.

Obsah

1 Úvod

V súčasnej dobe sú inteligentné telefóny súčasťou života skoro každého človeka. Neslúžia iba na bežné telefónne rozhovory, ale ponúkaju širokú škálu funkcionalít, ktoré sú sprístupnené pomocou aplikácií. Zo začiatku bol vývoj aplikácií pre inteligentné telefóny možný iba osobitne pre každú platformu, ale s postupom času sa začali objavovať nové možnosti vývoja multiplatformných aplikácií. Medzi ne patrí aj vývoj webových alebo hybridných aplikácií.

Cieľom tejto práce je navrhnúť a vytvoriť mobilnú aplikáciu pomocou aplikačného rámca PhoneGap alebo Apache Crodova. Aplikácia má byť schopná prezentovať funkcionalitu ako zobrazovanie tlačových úloh, sken čiarových kódov, geolokácia a prístup k fotoaparátu a do galérie. V teoretickej časti práce má byť zanalyzovaná interná štruktúra použitého aplikačného rámca.

Práca je rozdelená do troch častí. Prvá časť obsahuje teoretické informácie o jednotlivých typoch aplikácií a prehľad podporovaných funkcionalít, ktoré ponúka aplikačný rámec.

V druhej časti je zanalyzovaná technická stránka aplikačného rámca a analýza požadovaných funkcionalít aplikácie.

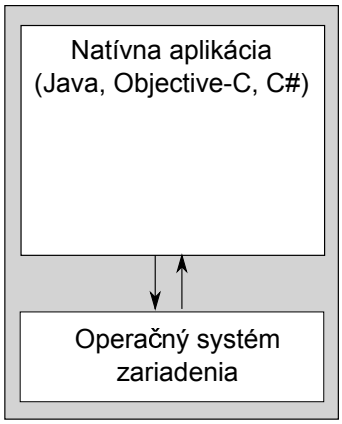
V poslednej časti je popísaný vývoj aplikácie a následné možnosti testovania a validácie.

2 Typy aplikácií a dostupné aplikačné rámce

2.1 Typy aplikácií

Pri vývoji mobilnej aplikácie má vývojár v dnešnej dobe viac možností, ktorou cestou sa vybrať. Z hľadiska tvorby aplikácie spustitelnej na rôznych platformách je možný vývoj natívnej, webovej alebo hybridnej aplikácie [2].

2.1.1 Natívne aplikácie



Obr. 2.1: Architektúra natívnej aplikácie

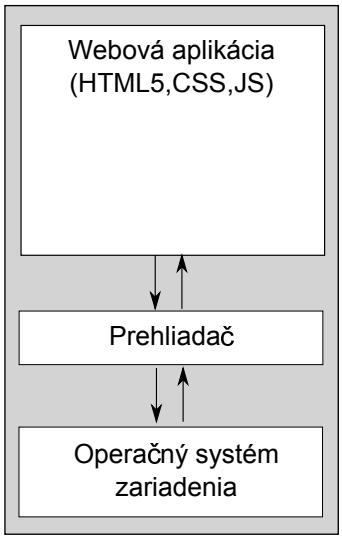
Natívne aplikácie sú špecifické pre každú platformu, používajú prostriedky konkrétneho operačného systému a sú napísané v jazykoch konkrétnych pre každú platformu. Napríklad aplikácie pre systém Android sú programované v jazyku Java, iOS používa Objective–C a Windows Phone používa jazyk C#. Obrázok 2.1 zobrazuje architektúru natívnej aplikácie.

Výhodou natívnych aplikácií je, že poskytujú najlepší prístup ku prostriedkom zariadenia. Taktiež používajú natívne prvky používateľského prostredia, ktoré sú špecifické pre každý operačný systém.

Nevýhodou vývoja natívnych aplikácií je, že jednu mobilnú aplikáciu je nutné programovať a vyvíjať osobitne pre každý operačný systém konkrétnej platformy. Tým pádom je väčšinou potrebné väčšie množstvo vývojárov.

2.1.2 Webové aplikácie

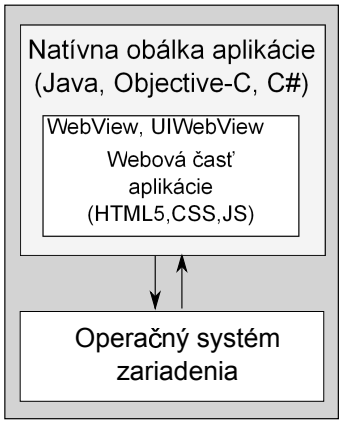
Webové aplikácie používajú štandartné webové technológie ako HTML5 (HyperText Markup Language – značkovací jazyk na vytváranie webových stránok, verzia 5), CSS (Cascading Style Sheets – jazyk pre vizuálne formátovanie HTML elementov) v kombinácii s jazykom JavaScript. Na obrázku 2.2 je možné vidieť, že webové aplikácie sú spúšťané pomocou prehliadača v operačnom systéme mobilnej platformy.



Obr. 2.2: Architekrúra webovej aplikácie

Webová aplikácia je webová stránka, ktorú zobrazí prehliadač po spustení aplikácie, takže nie je nutná znalosť programovacích jazykov pre všetky platformy. Taktiež nie je potrebné také množstvo vývojárov, ako je to u natívnych aplikácií. Dôležitým faktorom pri vývoji webovej aplikácie pre mobilné platformy je optimalizácia stránky pre rôzne veľkosti obrazoviek (responzívnosť).

Medzi nevýhody webových aplikácií patrí obmedzená funkčnost v prípade, že mobilné zariadenie nie je pripojené k internetu. Ďalšou slabinou tohto typu aplikácií je rozličná implementácia HTML5 štandartov v rôznych prehliadačoch platforiem. Taktiež pomocou tohto typu aplikácií nie je možný prístup k prostriedkom a funkcionalite zariadenia ako kontakty alebo notifikácie [3].



Obr. 2.3: Architektúra hybridnej aplikácie

2.1.3 Hybridné aplikácie

Hybridné aplikácie sú kombináciou natívnych a webových aplikácií. Podobne ako webové aplikácie, aj hybridné aplikácie sú vyvýjané použitím technologií ako HTML5, CSS a JavaScript. Narozdiel od webových a natívnych aplikácií sú zabalené do obálky, ktorú vytvorí natívna časť aplikácie. Kód webovej časti aplikácie je spúštaný v špeciálnej inštancii prehliadača zariadenia, ktorá je špecifická pre konkrétnu platformu. Pre Android a Windows Phone je to `WebView` a u iOS je to `UIWebView`. Architekúra hybridnej aplikácie je znázornená na obrázku 2.3.

Vývoj hybridnej aplikácie väčšinou prebieha naraz pre všetky platformy, takže vývoj by mohol zabrať menšie množstvo času. Podobne ako u webových, tak aj u hybridných aplikácií sa s použitím vhodného aplikačného rámca[1] stráca nutnosť znalosti konkrétnych programovacích jazykov pre všetky platformy.

Nevýhodou hybridných aplikácií môže byť ich väčšia spotreba energie v porovnaní s natívnymi aplikáciami. Do istej miery je to spôsobené najmä faktom, že pre komunikáciu s prostriedkami mobilného zariadenia je nutný preklad informácií z webovej časti aplikácie do natívnej časti [1].

2.2 Hybridné aplikačné rámce

Hybridné aplikačné rámce slúžia na tvorbu hybridných aplikácií. Väčšinou implementujú natívnu časť aplikácií a zabezpečujú komunikáciu s operač-

1. Ang. Framework

ným systémom zariadenia. Taktiež podporujú preklad dopytov a informácií z jazyka JavaScript do natívneho jazyka operačného systému. Príkladom takéhoto rámca je PhoneGap resp. Apache Cordova.

Prvotný vývoj aplikačného rámca pre tvorbu hybridných aplikácií pod názvom PhoneGap začala firma Nitobi Software už v roku 2008. Neskôr, v októbri 2011 bola firma pripojená k Adobe Systems [4]. Adobe Systems poskytlo celý aplikačný rámec neziskovej organizácii Apache Software Foundation, ktorá pokračuje vo vývoji tohto projektu pod názvom Apache Cordova. Adobe pokračuje s projektom pod menom PhoneGap, ktorý je založený na Apache Cordova.

PhoneGap podporuje vývoj aplikácií pre najrozšírenejšie mobilné operačné systémy ako Android, iOS, Windows Phone (WP) a BlackBerry. V súčasnosti pribúda podpora jednotlivých funkcionalít pre systémy Ubuntu Touch a Firefox OS:

	iPhone 3G	iPhone 3GS a novší	Android	WP 8	Ubuntu	Firefox OS
Akcelerometer	✓	✓	✓	✓	✓	✓
Kamera	✓	✓	✓	✓	✓	✓
Kontakty	✓	✓	✓	✓	✓	✓
Sieť	✓	✓	✓	✓	✓	✓
Média	✓	✓	✓	✓	✓	X
Notifikácie (upozornenie)	✓	✓	✓	✓	✓	✓
Notifikácie (zvuk)	✓	✓	✓	✓	✓	✓
Notifikácie (vibrácie)	✓	✓	✓	✓	✓	✓
Úložný priestor	✓	✓	✓	✓	✓	✓
Kompas	X	✓	✓	✓	✓	✓

Tabuľka 2.1: Prehľad podporovaných funkcionalít [36]

3 Analýza

Táto kapitola predstavuje analýzu interného fungovania aplikačného rámca PhoneGap a Apache Cordova. Taktiež sú zanalyzované jednotlivé podporované funkcionality, ktoré rámec ponúka.

3.1 Architektúra aplikačného rámca

Hybridný aplikačný rámec PhoneGap a Apache Cordova sa skladá z webovej a natívnej časti. Natívna časť rámca je naprogramovaná v jazyku špecifickom pre každú platformu a webová časť obsahuje knižnicu `Cordova.js`, ktorá poskytuje funkcionalitu pre volanie natívnych častí aplikačného rámca. Jednotlivé funckionality sú zapúzdrené do zásuvnych modulov.

Komunikáciu a preklad informácií medzi webovou a natívnou časťou zabezpečujú mosty aplikačného rámca. Existujú dva druhy mostov:

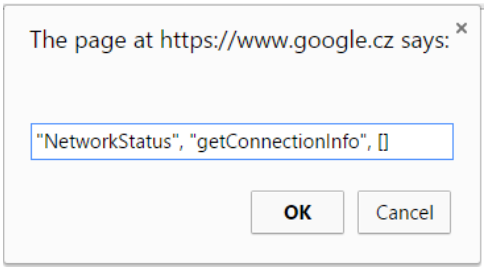
JsToNativeBridgeMode a NativeToJsBridgeMode.

JsToNativeBridgeMode [6] – Tento typ mostu zabezpečuje komunikáciu z webovej časti do natívnej časti. Vo webovej časti aplikácie je dostupný JavaScript objekt `cordova`, na ktorom je možné zavolať funkciu exec, ako je možné vidiet v nasledujúcej ukážke:

```
cordova.exec(
    successCallBack, errorCallBack,
    "NetworkStatus", "getConnectionInfo", []
);
```

Parametre `successCallBack` a `errorCallBack` sú spätné volania funkcí, ktoré sa vykonajú v prípade úspechu alebo neúspechu. Parameter s názvom `NetworkStatus` je názov triedy z natívnej časti a parameter s názvom `getConnectionInfo` je názov akcie (funkcie), ktorá sa má vykonať. Posledným argumentom je pole dodatočných nastaviteľných parametrov.

Knižnica `cordova.js` následne vyvoláva JavaScript dialógové okno typu `prompt`, do ktorého vloží názov triedy, akciu a dodatočné parametre. Obrázok 3.1 znázorňuje klasické JavaScript dialógové okno, avšak v prípade knižnice `cordova.js` sa toto okno vyvoláva na pozadí, bez upozornenia používateľa.



Obr. 3.1: JavaScript dialógové okno

Vyvolanie dialógového okna v prípade systému Android zachytí funkcia `onJsPrompt`. Táto funkcia je dostupná ako súčasť vývojového rámca pre natívne Android aplikácie, ale v natívnej časti rámca je táto funkcia prepísaná. Triede `PluginManager` predá parametre ako názov hľadanej triedy, akciu a dodatočné parametre z dialógového okna. Trieda `PluginManager` následne spustí funkciu `execute` príslušného modulu. Kedže sa jedná o natívnu časť, modul môže pristupovať k prostriedkom zariadenia.

NativeToJsBridgeMode [6] – Typ mostu, ktorý slúži na vrátenie informácií z natívnej časti kódu do webovej časti. Natívna časť posiela získané údaje pomocou inštancie triedy `PluginResult`. Údaje okrem získaných informácií obsahujú aj správu o tom, či bolo volanie natívnych funkcionalít úspešné alebo neúspešné. V triede `CordovaChromeClient` sa následne zachytia vrátené informácie a pomocou inštancie triedy `JsPromptResult` sa údaje vracajú do webovej časti aplikácie. Prí vracaní informácií sa volá funkcia, ktorá bola definovaná pre úspešné (`successCallBack`) alebo neúspešné (`errorCallBack`) volanie, takže vývojár môže adekvátne zareagovať na oba prípady.

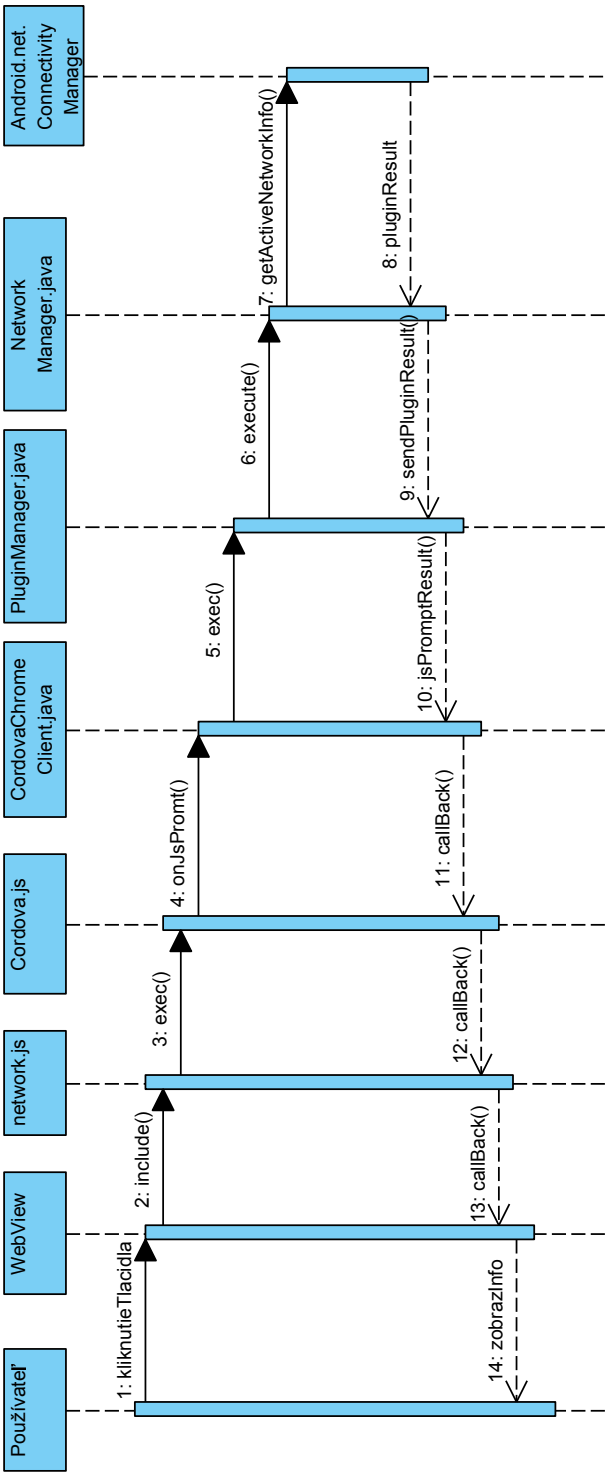
Kompletný priebeh volania natívnych funkcií a návrat informácií do webovej časti je znázornený na obrázku 3.2.

3.2 Proces tvorby aplikácie

V tejto sekcii je zachytený proces inštalácie jednotlivých prerekvizít rámca, ako aj vytvorenie vzorového projektu a preklad na lokálnom stroji alebo pomocou cloud computingu.

3.2.1 Prerekvizity

Na úspešné vytvorenie a preloženie výslednej aplikácie zo zdrojových webových súborov je potrebný nástroj `Cordova Command-Line Interface`. Tento



Obr. 3.2: Priebeh komunikácie medzi webovou a natívnou časťou aplikácie

nástroj sa inštaluje do operačného systému lokálneho stroja ako `npm` (manažér JavaScript balíčkov) balíček. Taktiež je nutné mať nainštalovaný nástroj `NodeJS` (nástroj pre programovanie internetových aplikácií, predovšetkým webových serverov v jazyku JavaScript).

Následne sú potrebné knižnice pre vývoj a preklad na jednotlivých platformách. Na systéme Android je to Android SDK, iOS používa nástroj Xcode a Windows Phone 8 používa Windows Phone SDK. Na vývoj Windows Phone aplikácií je taktiež nutný 64–bitový operačný systém Windows [7].

3.2.2 Vytvorenie projektu

Vytvorenie projektu aplikácie je možné pomocou nástroja `Cordova Command-Line Interface`. Príkazy nutné na vytvorenie projektu a pridanie platforiem sú znázornené na obrázku 3.3.

```
> cordova create hello com.example.hello HelloWorld
> cd hello
> cordova platform add wp8
> cordova platform add android
> cordova platform add ios
```

Obr. 3.3: Vytvorenie projektu a pridanie platforiem

3.2.3 Preklad projektu na lokálnom stroji

Preklad na lokálnom stroji je možný taktiež vďaka nástroju `Cordova Command-Line Interface`. Je potrebné použiť príkaz `cordova build` alebo `cordova build NazovPlatformy`, kde `NazovPlatformy` definuje konkrétnu platformu, ktorá je pridaná v projekte (v prípade nešpecifikovania platformy sa pokúsa o preloženie pre všetky pridané platformy). V prípade, že používame rámec PhoneGap, príkazy sú analogické, avšak namiesto slova `cordova` sa používa slovo `phonegap`.

3.2.4 Preklad projektu v službe PhoneGap Build

Spoločnosť Adobe poskytuje pre preklad PhoneGap aplikácií svoje cloud computing služby pod názvom PhoneGap Build. Používatelia si po registrácii môžu nahrať svoje zabalené aplikácie na stránku portálu PhoneGap Build[1] a následne sa táto služba postará o preloženie pre jednotlivé platformy. Zatiaľ sú podporované iba systémy Android, iOS a Windows Phone.

1. https://build.phonegap.com

Táto služba avšak nie je vo všetkých prípadoch bezplatná. Porovnanie platených a bezplatných plánov je znázornené v tabuľke 3.1.

	Bezplatný plán	Platený plán	Adobe Creative Cloud členstvo
Počet súkromných aplikácií	1	25	
Počet aplikácií s otvoreným zdrojovým kódom	Neobmedzene		
Maximálna veľkosť aplikácie	50 MB	100 MB	1 GB
Cena	Zdarma	od 9,99 $ / mesiac	od 29,99 $ / mesiac

Tabuľka 3.1: Porovnanie plánov Adobe PhoneGap Build služby [8]

PhoneGap Hydration

Výhodou používania služby PhoneGap Build je možnosť využiť popri preklade aplikácie službu PhoneGap Hydration. Bez služby Hydration je nutné po každom preložení aplikáciu znova nainštalovať do zariadenia. S použitím tejto služby táto nutnosť odpadá. PhoneGap Hydration doplní do aplikácie kontrolné nástroje, ktoré budú pri každom spustení monitorovať, či je v službe PhoneGap Build dostupná nová verzia preloženej aplikácie a v prípade že áno, poskytne používateľovi možnosť aktualizácie aplikácie bez nutnosti opätovného inštalovania.

Služba PhoneGap Hydration taktiež znižuje čas potrebný na celkové preloženie aplikácie [9].

3.3 Knižnice pre vývoj webovej časti aplikácie

Kedže kód hybridnej aplikácie je vyvíjaný v jazykoch HTML, CSS a JavaScript, je možný vývoj bez akýchkoľvek prídavných knižníc. Avšak vývoj bez použitia dnes už bežných knižníc by bol časovo náročný.

3.3.1 jQuery

V dnešnej dobe patrí knižnica jQuery medzi najrošírenejšie knižnice [10]. Umožňuje vyhľadávanie a úpravu DOM (Document Object Model - objektový model dokumentu) elementov, vytváranie animácií, správu udalostí a vývoj pomocou interaktívnych AJAX technológií (technológia tvorby dynamických web stránok bez nutnosti opätovného načítavania stránky) [11].

jQuery Mobile

Výhodou použitia knižnice jQuery pri tvorbe hybridnej webovej aplikácie je možnosť použitia dodatočnej knižnice jQuery Mobile, ktorá knižnicu jQuery rozširuje o dodatočnú funkcionalitu potrebnú pre dotykové zariadenia. Obsahuje napríklad niektoré predpripravené animácie pre navigáciu medzi stránkami aplikácie a taktiež ponúka širokú škálu spracovania mobilných udalostí [12]. Patria medzi ne najmä:

- tap – jednoduchý dotyk
- taphold – súvislý dotyk
- swipe – horizotálny ťah väčší ako 30px
- swipeleft / swiperight – vertikálny ťah do ľavej alebo pravej strany

3.3.2 Twitter Bootstrap

Knižnica Twitter Bootstrap poskytuje veľmi širokú škálu predvytvorených prvkov používateľského rozhrania a je optimalizovaná pre mobilné zariadenia. Medzi podporované prvky rozhrania patria najmä: modálne okná, rozbaľovacie listy, záložky a popisy tlačidiel.

Nevýhodou je, že neobsahuje spracovanie dotykových udalostí, narozdiel od knižnice jQuery Mobile. Preto je vhodná ich vzájomná kombinácia. Ďaľšou nevýhodou je, že niektoré prvky k svojej funkčnosti potrebujú knižnicu jQuery.

3.3.3 Hammer JS

Knižnica Hammer JS je vyvynutá hlavne pre poskytovanie podpory pre mobilné udalosti. Neobsahuje podporu pre manipuláciu s DOM elementami. Avšak výhodou tejto knižnice je fakt, že nie je závislá na ďalších, takže by pri vývoji nemusela byť použitá knižnica jQuery. Knižnica Hammer JS

poskytuje podporu pre dotykové udalosti ako: jednoduchý dotyk, dvojitý dotyk, súvislý dotyk, ťahanie, zmena veľkosti objektu, vertikálny ťah, horizontálny ťah a rotácia [13].

3.3.4 AngularJS

AngularJS je knižnica vytvorená spoločnosťou Google. Knižnica slúži na zviazanie (`binding`) a prepojenie HTML elementov a premenných z JavaScript modelu. HTML elementom sa špecifikujú špeciálne atribúty (väčšinou začínajúce spojením `ng-`), podľa ktorých sa určuje spojenie s JavaScript modelom. Obsah JavaScript modelu sa môže nastavovať manuálne v kóde, alebo dynamicky napríklad pomocou AJAX dopytov. Príklad jednoduchého použitia knižnice je dostupný nižšie:

```
<span ng-model="variable"></span>
```

Po takomto zápise sa do obsahu elementu `span` vloží obsah z JavaScript premennej `variable`.

3.4 Podpora LTR / RTL jazykov a fontov

Väčšina najrozšírenejších JavaScript knižníc je vyvýjaná pre použitie s `Left-To-Right` textom a podpora pre `Right-To-Left` písma nie je implementovaná. V nutnosti použitia RTL jazyku je nutné použiť správne písmo, zmeniť tok textu z pravej strany a kompletne horizontálne otočiť elementy webovej aplikácie.

Pre zmenu smeru toku textu je k dispozícii CSS vlastnosť `direction` [14]. Štandartne majú HTML prvky nastavenú túto vlastnosť na hodnotu `ltr` (`Left-To-Right`), avšak je možné ju nastaviť na hodnotu `rtl`.

Pre otočenie prvkov aplikácie je nutné zmeniť všetky CSS ľavé vlastnosti na pravé a naopak. Napríklad ľavé odsadenie textu je nutné zmeniť na pravé odsadenie, ľavý okraj objektu zmeniť na pravý. Toto je možné kompletným prerobením CSS kódu, alebo využitím online nástroja CSSJanus[2]. Tento nástroj dokáže vo vloženom CSS kóde vyhľadať `Left-To-Right` prvky a pretransformovať ich na `Right-To-Left` prvky.

S použitím knižníc ako jQuery Mobile alebo Twitter Bootstrap nastáva problém, pretože nemajú žiadnu oficiálnu podporu pre RTL jazyky [15]. Nezávislí vývojari avšak upravili knižnice pre podporu RTL jazykov a sprístupnili tieto upravené verzie pre verejnosť.

2. http://cssjanus.commoner.com/

Pre správnu optimalizáciu aplikácie pre RTL jazyky je nutné zmeniť taktiež font písma. Zmena fontu je možná pomocou CSS pravidla `font-face`, ktoré vo väčšine prehliadačov akceptuje formáty TTF a WOFF.

3.5 Možnosti integrácie UI s konvenciami platforiem

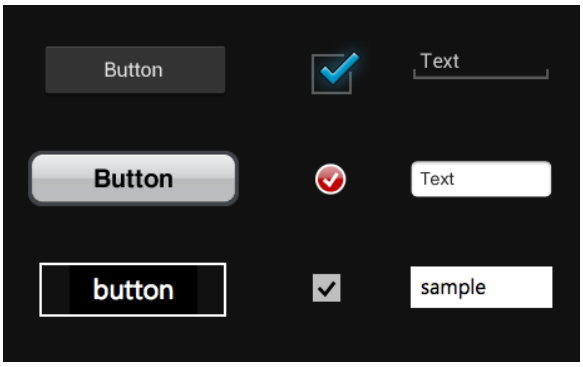
Keďže hybridná aplikácia je vytváraná pomocou jazykov HTML a CSS, je možné nastaviť vzhľad jednotlivých prvkov používateľského rozhrania pre každú platformu. Avšak tento krok už porušuje princíp vývoja aplikácie pomocou rámca PhoneGap / Cordova, pretože je nutné vytvárať rôzne dizajny pre špecifické platformy a nutnosť osobitne prekladať aplikáciu pre každú platformu s rôznym CSS kódom.

Problém osobitného prekladania aplikácie je možné riešiť pomocou JavaScript kódu. Každá web stránka obsahuje JavaScript premennú *navigator*, ktorá obsahuje informácie o aktuálne použitom prehliadači. Objekt *navigator.userAgent* obsahuje okrem iného aj názov používaného operačného systému, platformy alebo zariadenia [16]. Takže je možné načítať rôzne CSS súbory pre rôzne platformy.

Nevýhodou tohto prístupu je, že aplikácia potom obsahuje pribalené CSS súbory pre všetky platformy a zväčšuje sa jej celková veľkosť.

Prehľad konvencií platforiem

UI elementy sú v rôznych platformách dizajnované odlišne. Napríklad tlačidlá majú rôzne farby textu a pozadia, ostré alebo oblé hrany. Taktiež pozadie textu a farba textu je v rôznych farbách. Prehľad základných elementov je zobrazený na obrázku 3.4.

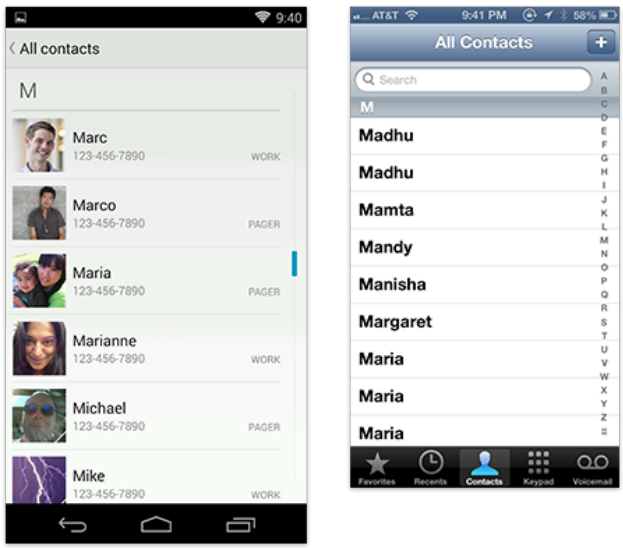


Obr. 3.4: Prehľad elementov pre Android, iOS a Windows Phone [17]

Jednotlivé ikony pre typickú funcionalitu ako zdieľanie, vytváranie a mazanie dokumentov, vyhľadávanie obsahu a pridanie obsahu k obľúbeným položkám sú takisto špecifické pre každú platformu.

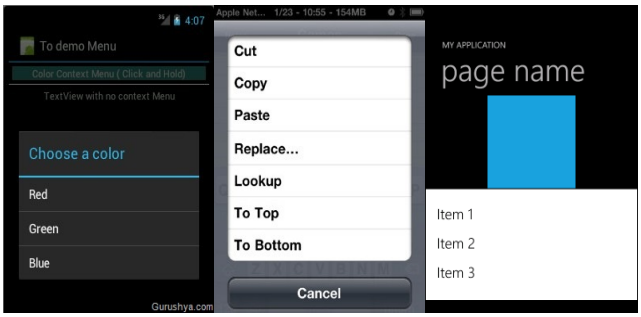
Obr. 3.5: Prehľad ikon pre Android, iOS a Windows Phone [17]

Taktiež navigácia medzi kartami obsahu je rozdielna. Napríklad, Android používa lištu pre navigáciu v hornej časti, narozdiel iOS používa ikony pre navigáciu v spodnej časti aplikácie.



Obr. 3.6: Rozdielne umiestnenie navigácie v systéme Android a iOS [17]

Ďalším prvkom používateľského prostredia je kontextová navigácia, ktorá sa zobrazuje po podržaní dotyku na obrazovke. Jej vzhľad je taktiež rozličný naprieč platformami.



Obr. 3.7: Rozdielne kontextové menu pre Android [18], iOS [19] a Windows Phone [20]

3.6 Prístup k špecifickým funkcionalitám

Niektorá funkcionalita je dostupná už v samotnom jadre aplikačného rámca, dodatočná funkcionalita sa získava pomocou zásuvnych modulov. Služba PhoneGap Build ponúka na svojich stránkach zoznam podporovaných modulov. Zdrojové kódy týchto modulov sú prevažne dostupné na stránke GitHub.

3.6.1 Otočenie obrazovky

Aplikácie vyvynuté pomocou rámca PhoneGap a Cordova majú štandartne zapnutú funkcionalitu pre otáčanie obrazovky. Rotácia obrazovky sa však dá aj obmedziť, použitím príkazu
`<preference name="Orientation" value="landscape|portrait" />`
v konfigurácii aplikácie, kde `portrait` znamená otočenie na výšku a `landscape` na šírku.

3.6.2 Prístup k fotoaparátu, prístup do galérie

Funkcionalitu zachytávania fotografií a prístupu do galérie poskytuje oficiálny zásuvný modul rámca PhoneGap *org.apache.cordova.camera*. V súčasnosti plne podporuje Android, iOS, Windows Phone a i. Nové verzie modulu vychádzajú približne 8 krát za rok. Modul je dostupný s licenciou Apache 2.0. Na platforme Android je nutné pridať do konfigurácie AndroidManifest riadok:
`<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />`
To umožní pri inštalácií povoliť potrebné oprávnenia, ktoré sú na systéme Android nutné pre zápis do externého ukladacieho priestoru.

3.6.3 QR kódy

Funkcionalitu pre prácu s QR kódmi poskytuje hneď niekoľko zásuvnych modulov.

Názov GitHub účtu / názov modulu	Licencia	Podporované platformy
Wildabeast / BarcodeScanner	MIT	Android, iOS, Windows Phone 8
Manateeworks / MWBarcodeScanner	MIT	Android, iOS, Windows Phone 8
Scandit / BarcodeScannerPlugin	Apache 2.0	Android, iOS

Tabuľka 3.2: Prehľad dostupných modulov pre QR kódy

- Wildabeast / BarcodeScanner – Tento modul je podľa služby PhoneGap Build najpoužívanejší, aktuálne ho používa viac ako 200 tisíc aplikácií. V súčasnosti modul nepodporuje na platforme Windows Phone vytváranie čiarových kódov. Ďalším problémom je nefunkčnosť alebo zablokovanie používateľského rozhrania po stlačení tlačidla pre zrušenie skenovania kódu pomocou fotoaparátu [21]. Kvalita dokumentácie je primeraná, sú popísané všetky podporované čiarové kódy naprieč všetkými podporovanými platformami. QR kódy sú podporované na všetkých systémoch. V ďalšom repozitári je dostupná aj ukážková PhoneGap aplikácia so základným použitím modulu, avšak z dokumentácie modulu na túto ukážkovú aplikáciu nevedie žiadny odkaz.

- Manateeworks / MWBarcodeScanner – Podla údajov zo služby PhoneGap Build používa tento plugin približne 600 aplikácií. Kvalita dokumentácie v repozitári modulu je nízka, chýbajú základné ukážky použitia a taktiež aj ukážková aplikácia. V minulých verziách mal modul problém so stlačením tlačidla pre zrušenie skenovania čiarového kódu pomocou fotoaparátu.

- Scandit / BarcodeScannerPluigin – Kvalita dokumentácie je vysoká, avšak nachádza sa na stránkach firmy Scandit. Tento modul je súčasťou väčšieho vývojového nástroja pre skenovanie čiarových kódov od firmy Scandit. Najnižšia cena nástroja je 199 dolárov za mesiac [22].

3.6.4 NFC

Funkcionalitu pre NFC komunikáciu v súčasnosti podľa služby PhoneGap Build podporuje iba jeden modul – *com.chariotsolutions.nfc.* Tento modul aktívne využiva približne 7 tisíc aplikácií preložených službou PhoneGap Build. Modul podporuje platformy Android a Windows Phone 8. Zatiaľ je absencia podpory iOS platformy, nakoľko NFC komunikáciu podporuje iba iPhone 6, ktorý je krátko na trhu. Podpora iOS platformy avšak nie je pravdepodobná ani v blízkej budúcnosti, pretože iOS zatiaľ neposkytuje prístup k funkcionalite NFC vo svojom súbore nástrojov pre vývoj softvéru [23]. Modul je vydaný s licenciou MIT.

3.6.5 Pozdržanie zablokovania obrazovky

Pre pozdržanie zablokovania obrazovky je dostupný modul *nl.x–services.plugins.insomnia,* ktorý aktuálne využíva približne 34 tisíc aplikácií. Modul podporuje platformy Android, iOS a Windows Phone 8 a má licenciu MIT. Je dostupná základná dokumentácia pre pozdržanie zablokovania obrazovky a následného povolenia zablokovania obrazovky. V súčasnosti avšak nie je dostupné riešenie pre okamžité zablokovanie obrazovky.

3.6.6 Geolokácia

Pre sprístupnenie funkcionality geolokácie je dostupný oficiálny zásuvný modul *org.apache.cordova.geolocation,* ktorý je podporovaný na platformách Android, iOS, Windows Phone a i. Kvalita dokumentácie je na dobrej úrovni, nechýbajú príklady použitia. Avšak v oficiálnej dokumentácii modulu nie je žiadna zmienka o tom, že modul už nie je na platforme Android podporovaný [24]. Modul prestal byť podporovaný, pretože funkcionalita geolokácie je implementovaná štandartom HTML5 [25].

3.7 Bezpečnosť

3.7.1 Bezpečnostné zraniteľnosti

Bezpečnostné zraniteľnosti systémov popisujú CVE záznamy [26]. V súčastnosti existuje v National Vulnerability Database (databáza bezpečnostných incidentov vlády Spojených štátov) 12 CVE záznamov pre rámce PhoneGap a Apache Cordova.

Skupina CVE záznamov CVE-2014-3500, CVE-2014-3501, CVE-2014-3502, ktoré popisujú bezpečnostné zraniteľnosti prekladu aplikácií pre platformu

Android. Chyby objavili David Kaplan a Roee Hay z IBM Security Systems.

- CVE-2014-3500 – Android aplikácie vyvynuté pomocou rámca Cordova sa spúšťajú pomocou špeciálnej intent URL. Pomocou špeciálne vytvorenej URL adresy bolo možné pri spustení aplikácie načítať inú HTML stránku, ako zamýšľal vývojár.

- CVE-2014-3501 – Android aplikácie vyvynuté pomocu rámca Cordova používajú objekt WebView na zobrazenie obsahu. Cordova aplikácie môžu špecifikovať zoznam dôveryhodných adries, ktorých obsah bude možné zobraziť v objekte WebView, avšak tento zoznam sa nekontroloval vočí adresám, ktoré používali iný protokol ako HTTP. V tom prípade bolo možné ustanoviť aktívne WebSocket spojenie.

- CVE-2014-3502 – Android aplikácie vyvynuté pomocu rámca Cordova môžu spúšťať iné aplikácie pomocou HTML odkazov alebo presmerovaním intent URL adresy. Útočník, ktorý dokáže pozmeniť HTML obsah aplikácie dokáže spúšťať a odosielať informácie do iných aplikácií.

Časová os záznamov

23.6.2014	Objavené chyby boli oznámené Apache Security oddeleniu
26.6.2014	Chyby potvrdené Apache Security tímom
4.8.2014	Chyby opravené, vydaná aktuálizácia pre Android preklad rámca Cordova. Oznámenie o chybách a aktualizácia boli publikované verejnosti
7.8.2014	Chyby opravené v službe PhoneGap Build. Avšak namiesto použitia opravenej verzie Android prekladu rámca Cordova, vývojari v PhoneGap Build opravili svoju verziu Android prekladu, takže verzia ostala nezmenená [35]. To však malo po čase za následok, že sa niektorým vývojárom, ktorý svoje aplikácie preložili touto verziou pomocou PhoneGap Build služby ozvala spoločnosť Google. Informovala ich o tom, že používajú zraniteľnú verziu rámca Cordova a že aplikácie môžu byť v obchode Google Play zablokované alebo zmazané [36]. Aj napriek tomu, že aplikácie boli preložené opravenou verziou.

Ďalšou skupinou bezpečnostných záznamov v National Vulnerability Database sú: CVE-2014-1881 až CVE-2014-1888. Tieto bezpečnostné incidenty objavili, nahlásili a následne publikovali Martin Georgiev, Suman Jana a Vitaly Shmatikov.

Hybridné aplikačné rámce ako Apache Cordova a PhoneGap by mali zabezpečiť nasledujúcu dôležitú vlasnosť: Webový obsah (napríklad JavaScript kód), pochádzajúci z nedôveryhodných zdrojov, ktorý je zahrňovaný do aplikácie by nemal mať prístup k prostriedkom zariadenia (príklad takéhoto obsahu môže byť reklamný skript, ktorý je do aplikácie zahrňovaný pomocou HTML tagu IFRAME). Aby bola táto požadovaná bezpečnostná funkcionalita implementovaná, je potrebné aby aplikačný rámec skombinoval dve rôzne bezpečnostné politiky. Prvá je politika rovnakého pôvodu – Same-Origin-Policy (d'alej len SOP), ktorá je aplikovaná na webový obsah aplikácie. Druhá je kontrola prístupu k prostriedkom zariadenia operačným systémom platformy. Predovšetkým teda musia korektne propagovať SOP na objekty, ktoré existujú mimo webového obsahu aplikácie (napr. prostriedky zariadenia) a sú prístupné z prehliadača pomocou mostov, ktoré poskytuje aplikačný rámec.

Fracking Attack

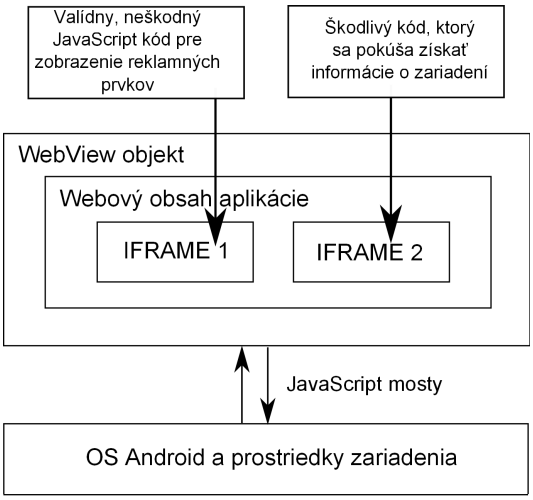
Pojem Fracking Attack označuje útok, ktorý umožňuje vloženým skriptom (napríklad pomocou tagu IFRAME) pristupovať k prostriedkom zariadenia. Prístup je možný, nakoľko sú v aplikácii už zahrnuté Cordova knižnice, ktoré poskytujú funcionalitu volania mostov z webovej časti.

Ako je možné vidieť na obrázku 3.8, do aplikácie je vložený tag IFRAME, ktorý načíta obsah tretej strany. Tvorca aplikácie avšak nemá možnosť vedieť, či sa v budúcnosti server z ktorého je načítaný obsah nezmení na škodlivý server, ktorý bude poskytovať škodlivý kód.

Obrana voči tomuto typu útoku sa dá rozdeliť do dvoch kategórií: No-Load a No-Bridge.

No-Load

Rámce Apache Cordova a PhoneGap v súčasnosti obsahujú implementovanú funkcionalitu whitelist. Rámec sa snaží blokovať obsah tretej strany, pokiaľ sa doména obsahu nenachádza v zozname whitelist. Takže nie je povolené načítanie obsahu z nedôveryhodných zdrojov. Avšak toto nefungovalo korektne na platforme Android vždy. V minulosti obsahovala chybu, keď bola nesprávne spracovávaná URL adresa zo zoznamu whitelist. Na-

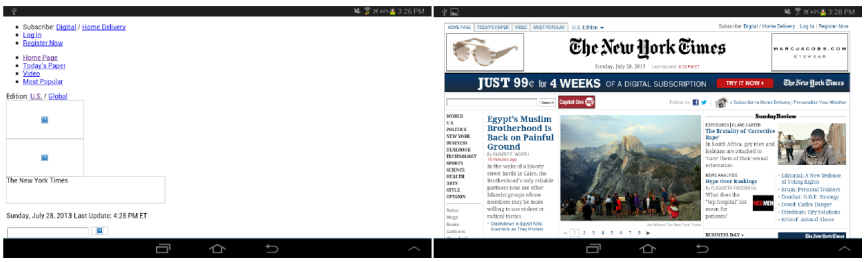


Obr. 3.8: Fracking Attack

príklad, ak bola adresa foo.com označená ako dôveryhodná, obsah bol povolený načítať aj z domény foo.com.example.com.

Implementácia metódy No-Load ale nie je korektná z hľadiska obchodného modelu väčšiny komerčných aplikácií, ktoré používajú reklamné služby. Do zoznamu whitelist sa môžu zapísať iba adresy, ktoré sú známe pred prekladom aplikácie. Adresa serveru, ktorý sprostredkuje reklamu je pred prekladom aplikácie známa, ale tento server vo väčšine prípadov načítava ešte ďalší obsah pomocou svojho kódu zo serverov, ktoré sa môžu meniť docela často a nemusia byť známe pri preklade aplikácie.

Istou možnosťou je aj povolenie všetkého obsahu, tak že do zoznamu whitelist sa pridá hodnota s hiezdičkou, avšak toto nie je bezpečná cesta načítavania obsahu.



Obr. 3.9: Stránka nytimes.com s povolením domény iba nytimes.com a s povolením všetkého obsahu [37]

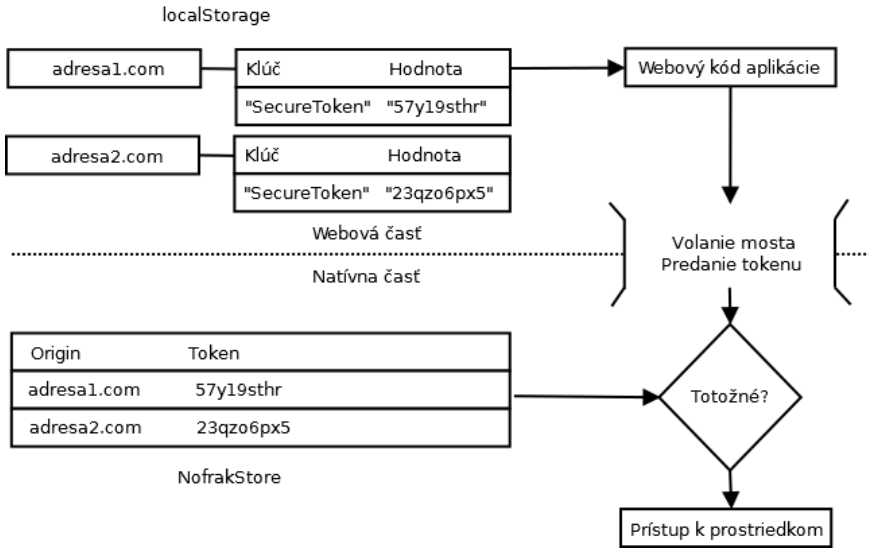
No-Bridge

Metóda No-Bridge povoluje načítanie cudzieho obsahu, avšak povoluje volanie mostov rámca iba obsahu aplikácie a dôveryhodným zdrojom. Takže je kompatibilná s obchodným modelom aplikácií používajúcich reklamu z tretích strán [37].

NoFrak

Autori článku, v ktorom boli tieto chyby publikované tiež naprogramovali implementáciu No-Bridge pod názvom NoFrak pre platformu Android, ktorú neskôr vývojári Apache Cordova zahrnuli do oficiálnej knižnice. Hlavnou ideou nástroja NoFrak je, že každý prístup k mostu rámca musí byť overený charakteristickým znakom (ďalej len token), ktorý je unikátny pre každý pôvod obsahu (`origin`). Nástroj NoFrak zmenil väčšinou iba natívnu časť rámca, takže webová funkcionalita a prístup k mostom z webovej časti ostal rovnaký, takže vývojari aplikácií nemuseli meniť kód svojich aplikácií preložených staršou verziou rámca Cordova.

Vnútorne, nástroj NoFrak pred štartom objektu `WebView` vygeneruje deväť miestny token pomocou Java triedy `SecureRandom` pre každý pôvod obsahu (`origin`), ktorý sa nachádza v zozname `whitelist`. Tieto informácie sú uchované v lokálnej časti aplikácie v `ArrayList` poli nazvanom `NofrakStore`. Následne, po vygenerovaní tokenov pre každý `origin` sa tieto tokeny vložia do `localStorage` `WebView` objektu prehliadača [37].



Obr. 3.10: Ideový náčrt funkcionality NoFrak

`LocalStorage` je interná databáza prehliadača, ktorá umožňuje ukladať informácie pre každý `origin` samostatne, pričom každý `origin` má prístup iba ku svojim dátam a dáta su ukladané vo forme kľúča a hodnoty. Každý token sa ukladá do `localStorage` ku konkrétnemu `origin`, kľúč má názov `SecureToken` a hodnota je vygenerovaný deväťmiestny token. Následne, pri volaní mosta aplikačného rámca, sa tento token vytiahne z `localStorage` prehliadača a vloží sa do dopytu. Keď sa volanie dostane až do natívnej časti, kontroluje sa, či je tento token prítomný v predtým vytvorenom zozname `NofrakStore`, ktorý obsahuje tokeny dôveryhodných adries. V prípade, že k mostu posiela požiadavky nedôveryhodný `origin`, tak sa v `localStorage` nebude nachádzať premenná `SecureToken` a z toho dôvodu nedostane prístup k prostriedkom zariadenia. Prehľad implementácie NoFrak je na obrázku 3.10.

3.7.2 Metódy overenia identity servera a klienta

Overenie identity servera prebieha pomocou kontroly dôvery operačného systému v certifikát servera. Ak však dôjde k chybe a z nejakého dôvodu je certifikát servera nedôveryhodný, na platforme Android nie je možné v súčasnej implementácii rámca túto chybu zachytiť.

Overenie klienta prebieha pomocou klientských certifikátov. V aplikačnom rámci v minulosti chýbala požadovaná funkcionalita na používanie klientských certifikátov. Tento problém by mal byť implementovaný v roku 2015, avšak zatiaľ neexistuje žiadny zásuvny modul, ktorý by funkcionalitu klientských certifikátov podporoval [34]. Implementácia podporuje iba platformu Android vo verzii 5 (Lollipop), nakoľko táto funkcionalita bola pridaná do prostredia operačného systému až s verziou Android 5 (Lollipop). Na platforme Windows Phone nie sú momentálne klientské certifikáty podporované vôbec [35].

4 Vývoj a validácia aplikácie

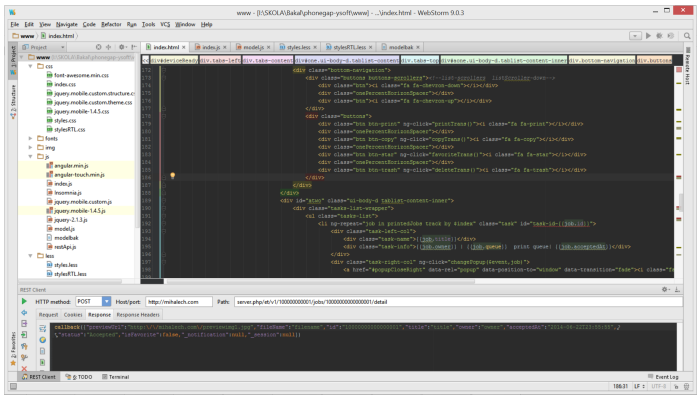
4.1 Vývoj aplikácie

Pred vývojom aplikácie som sa najprv rozhodoval, ktoré vývojové prostredie použijem, následne som si vybral dodatočné knižnice pre jednotlivú funkcionalitu. Neskôr som si rozpracoval prototyp interného modelu aplikácie a následne som spracoval používateľské rozhranie aplikácie. Nakoniec som pridal požadovanú funkcionalitu pomocou zásuvných modulov alebo vlastného riešenia. Náhľad aplikácie spustenej v operačných systémoch Android, iOS a Windows Phone je dostupný v prílohe A. V prílohe sú taktiež dostupné aj náhľady implementovaných funcionalít.

4.1.1 Vývojové prostredie

Keďže vývoj mobilnej aplikácie pomocou rámca PhoneGap prebieha ako vývoj webovej aplikácie, je možné použiť akékoľvek vývojové prostredie, ktoré umožňuje tvorbu webových aplikácií. Medzi populárne editory patria: Sublime, LightTable, WebStorm, Brackets.

Pre vývoj aplikácie som sa rozhodol pre editor WebStorm od firmy JetBrains, nakoľko mám už pozitívne skúsenosti s používaním iných produktov tejto firmy. Editor okrem klasických funkcionalít ako doplňanie kódu, zvýraznenie syntaxe a podpory verzovacích systémov umožňuje aj zasielanie požiadavok k službám typu REST(dopytovanie na server pomocou HTTP volaní). Náhľad vývojového prostredia je na obrázku 4.1.



Obr. 4.1: Vývojové prostredie WebStorm

Ďalší editor, ktorý stojí za zváženie pre vývoj je editor Brackets. Do tohto editoru je možná inštalácia dodatočného modulu, ktorý umožňuje komunikáciu so službou PhoneGap Build a dokáže potrebné súbory nahrávať na server služby a taktiež je možné aplikáciu preložiť. V konečnom dôsledku tak odpadá nutnosť zdrojové súbory aplikácie nahrávať do služby PhoneGap Build pomocou webového rozhrania cez prehliadač. V minulosti bolo vyvíjané aj vývojové prostredie Adobe Edge Code, avšak na konci roka 2014 sa toto prostredie stalo súčasťou editoru Brackets [29].

4.1.2 Použité knižnice

Pri vývoji aplikácie som použil nasledovné knižnice:

- jQuery – pre ľahšiu manipuláciu s HTML elementami, využitie animácií a informačného vyskakovacieho okna
- jQuery Mobile – využitie udalostí typických pre dotykové zariadenia ako `tap` a `taphold`.
- Angular JS – interné spracovanie tlačových úloh a okamžité premietnutie do používateľského rozhrania

Knižnice sa zahrňujú do projektu typicky pomocou HTML tagov `<script>` pre JavaScript funkcionalitu a `<link rel="stylesheet">` pre vzhľadovú funkcionalitu.

4.1.3 Prehľad interného modelu aplikácie

Uloženie interného modelu aplikácie prebieha pomocou knižnice AngularJS. Najprv je nutné špecifikovať atribút `ng-app` a `ng-controller` niektorému HTML tagu:

```
<div ng-app="ysoftApp" ng-controller="AppCntrl">
```

Následne pomocou jazyka JavaScript definujem, ako bude vyzerať interné spracovanie dát v aplikácii.

```
var myApp = angular.module('ysoftApp', []);
myApp.controller('AppCntrl', function ( $scope) {
        $scope.waitingJobs = [];
        $scope.printedJobs = [];
        $scope.favoriteJobs = [];
        $scope.selectedTasks = [];
        $scope.IsRTL = false;

        $scope.getPrintedJobs = function () {...};
        $scope.getWaitingJobs = function () {...};
```

```
        $scope.getJobDetail = function (jobId) {...};
        $scope.fillTasks = function () {...};
        $scope.printTrans = function () {...};
        $scope.copyTrans = function () {...};
        $scope.favoriteTrans = function () {...};
        $scope.deleteTrans = function () {...};
        $scope.changePopup = function () {...};
    }
);
```

Objekt `$scope` poskytuje knižnica AngularJS a je to oblasť, ku ktorej má HTML časť kódu prístup. Pre kategórie čakajúcich, vytlačených a obľúbených úloh som si vytvoril polia `waitingJobs`, `printedJobs` a `favoriteJobs`. Pre úlohy, ktoré budú zvýraznené na základe podržania dotyku som vytvoril pole `selectedTasks`. Poslednou definovanou premennou je premenná `IsRTL`, ktorá obsahuje pravdivostnú hodnotu, či má byť používateľské prostredie vo forme RTL alebo LTR.

Následne definujem funkcie `getPrintedJobs` a `getWaitingJobs`, ktoré pomocou AJAX HTTP dopytov vrátia základné informácie o úlohách v zložkách pre čakajúce a vytlačené úlohy. Ďalšia funkcia `getJobDetail` taktiež pomocou AJAX HTTP dopytu vráti detailné informácie o úlohe, vrátane odkazu na náhľad úlohy, na základe `jobId`, ktoré získavam z dvoch predchádzajucích funkcií. Tieto funkcie využívajú jednoduché REST rozhranie.

Funkcia `fillTasks` následne naplní polia `waitingJobs`, `printedJobs` a `favoriteJobs` na základe získaných informácií o jednotlivých úlohách z predošlých funkcií.

Funkcie `printTrans`, `copyTrans`, `favoriteTrans` a `deleteTrans` slúžia na implementáciu funkcionality vytvárania transakcií pre tlač, kopírovanie, obľúbenie a mazanie úloh.

Posledná funkcia `changePopup` slúži na zmenu obsahu informačného vyskakovacieho okna k jednotlivej úlohe, ktoré zobrazuje dodatočné informácie a náhľad úlohy.

4.1.4 Tvorba používateľského rozhrania podľa predlohy

Keďže kód aplikácie je webová aplikácia, používa sa pre tvorbu používateľského rozhrania jazyk HTML v kombinácii s jazykom kaskádových štýlov CSS. Základným dôležitým prvkom pri tvorbe vzhľadu prispôsobeného pre mobilné zariadenia je jeho responzívnosť. To znamená, že používateľské prostredie by sa malo automaticky adaptovať na rôzne veľkosti displejov a taktiež by malo plynule reagovať na zmenu orientácie zariadenia. Táto funkcionalita sa v jazyku CSS dosahuje pomocou `media` dopytov.

Typickým *media* dopytom môže byť napríklad nastavenie odlišnej farby pozadia elementu *body*, ak sa jedná o element na obrazovke a aktuálna maximálna šírka obrazovky je 300 pixelov:

```
@media screen and (max-width: 300px) {
    body {
            background-color: red;
    }
}
```

Po dlhšom vývoji aplikácie som avšak nadobudol názor, že písať čistý CSS kód nebude pre mňa dostatočne prehladné. Takže som sa rozhodol že použijem jazyk LESS. Pomocou jazyka LESS si môžem definovať vlastné premenné, taktiež môžem zapisovať štýly pre vnorené elementy bez nutnosti opakovania kódu, takže v konečnom dôsledku je LESS kód prehľadnejší a čitateľnejší. LESS kód je možné následne preložiť do jazyka CSS a výstup použiť v aplikácii. Porovnanie CSS a LESS kódu:

```
/*    LESS Code Example  */
@red: #842220;
#element {
    h1 {
        font-size: 3em;
        color: @red;
    }
    p {
        font-size: 2em;
        a {
                text-decoration: none;
                &:hover {
                  border-width: 1px solid black;
                }
        }
    }
}
```

Výsledný CSS kód ktorý vznikne preložením:

```
/*   CSS Code Example   */
#element h1 {
    font-size: 3em;
    color: #842220;
}

#element p {
    font-size: 2em;
}

#element p a {
    text-decoration: none;
}

#element p a:hover {
    border: 1px solid black;
}
```

Ďalšou vlastnosťou jazyka CSS, ktorú je možné pri vývoji responzívneho dizajnu využiť je vlastnosť `webkit-min-device-pixel-ratio`. Táto vlastnosť udáva pomer počtu fyzických pixelov zariadenia a počtu CSS pixelov. Takže udáva hustotu displeja. V tabuľke 4.1 je znázornený prehľad hustoty bodov niektorých zariadení.

Minimálna hustota	Zariadenia
1,0	Apple iPhone 3G, Apple iPhone 3GS, Samsung Galaxy S, Samsung Galaxy Tab 10.1
1,5	Google Nexus S, HTC Desire, Samsung Galaxy S II
2,0	Apple iPhone 4, 5, 6, Apple iPad Air, HTC One X, Google Galaxy Nexus, Samsung Galaxy S III
3,0	Apple iPhone 6 Plus, OnePlus One, Samsung Galaxy Note 4, Sony Xperia Z

Tabuľka 4.1: Prehľad hodnoty hustoty displeja niektorých zariadení.

Použitie vlastnosti `webkit-min-device-pixel-ratio` je podporované na platformách Android a iOS. Pre podporu tejto funkcionality aj na iných platformách je možné využiť CSS vlastnosť `min-resolution`, ktorá v podstate udáva počet pixelov na jednotku palec. A keďže jeden palec má 96 pixelov, `webkit-min-device-pixel-ratio` s hodnotou 1,5 bude nadobúdať hodnotu 144 pre vlastnosť `min-resolution` [30].

Obr. 4.2: Porovnanie používateľského prostredia. Vľavo vzorová ukážka poskytnutá spoločnosťou Y Soft, vpravo výsledná aplikácia

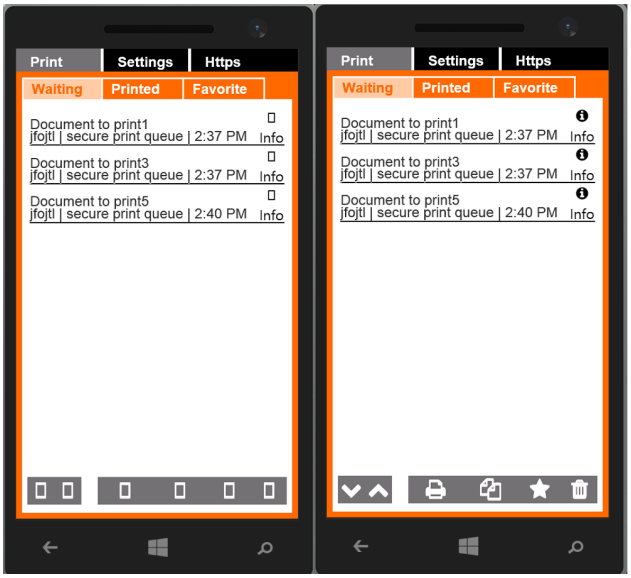
Pre zobrazenie ikon tlačidiel v používateľskom rozhraní som sa rozhodol, že namiesto veľkého množstva obrázkov s rôznym rozlíšením použijem špeciálne písmo. Konkrétne som použil písmo `Fonts Awesome`[1], ktoré

1. http://fortawesome.github.io/Font-Awesome/

obsahuje veľké množstvo predpripravených ikon. Tým pádom odpadá nutnosť tvorby veľkého množstva ikon pre každé rozlíšenie.

Porovnanie vzorovej predlohy používateľského rozhrania a výsledného vytvoreného rozhrania je možné vidieť na obrázku 4.2.

Pri testovaní som ale narazil na problém, že tento font sa nezobrazuje správne na platforme Windows Phone, zatiaľ čo na platformách iOS a Android funguje korektne. Neskôr som zistil, že objekt `WebView` platformy Windows Phone nepodporuje CSS vlastnosť `font-face` (vlastnosť, pomocou ktorej sa definujú fonty), ak sa jedná o font ktorý je uložený lokálne. Preto bolo nutné načítavať font z externej stránky a fungovanie je korektné. Ak však zariadenie nebude pri štarte aplikácie pripojené k internetu, nebude zaručené správne fungovanie fontu. V nasledujúcej ukážke je porovnanie aplikácie, ktorá načítava font z lokálneho súboru a načítanie fontu z externej stránky na operačnom systéme Windows Phone.



Obr. 4.3: Vľavo načítanie fontu z lokálneho súboru, vpravo z externej stránky. Platforma Windows Phone.

4.1.5 Triedenie úloh do zložiek

Na rozdelenie úloh do zložiek v používateľskom prostredí som použil funkcionalitu `tabs` knižnice jQuery Mobile. Pre inicializáciu je potrebné nastaviť atribút `data-role` určitému elementu:
`<div data-role="tabs" > ... </div>`.

Následne je potrebné každú zložku naplniť údajmi. Na to je vhodné použiť funkcionalitu knižnice AngularJS a to konkrétne `ng-repeat`. Výsledný kód pre naplnenie môže vyzerať nasledovne:
`<li ng-repeat="job in printedJobs" id="job-id-{{job.id}}"> ... </li>`.
Tento zápis odkazuje na premennú `printedJobs`, ktorá je v modeli aplikácie a je naplnená informáciami o vytlačených úlohách. Podobný postup sa aplikuje na zložku vytlačených a obľúbených úloh.

4.1.6 Implementácia zvolených funkcionalít

Pre implementovanie zvolených funkcionalít sú do projektu postupne pridávané jednotlivé zásuvné moduly. Vo väčšine prípadov zásuvné moduly fungujú na princípe volania funkcí so spätnými volaniami (callback) v prípade úspechu a neúspechu.

Prístup k fotoaparátu a do galérie

Funkcionalitu kamery a prístupu do galérie poskytuje zásuvný modul `org.apache.cordova.camera`. Pre vyvolanie funkcionality fotoaparátu a zachytenie fotografie je dostupná funkcia `getPicture`:

```
function getPhotoFromCam() {
    navigator.camera.getPicture(onSuccess, onFail, { quality: 50,
            destinationType: Camera.DestinationType.DATA_URL
    });
}
function onSuccess(imageData) {
    var image = document.getElementById('myImage');
    image.src = "" + imageData;
}
function onFail(message) {
    alert('Failed because: ' + message);
}
```

Ukážka práce s kamerou [31]

Funkcie `onSuccess` a `onFail` sú spätné volania (callback). Definujú aké príkazy sa majú vykonať ak dôjde k úspešnému zachyteniu fotografie, alebo z nejakého dôvodu proces zlyhá. Funkcia `alert` vyvolá natívne okno upozornenia operačného systému.

Pre prístup do galérie a zvolenie fotografie z galérie slúži taktiež funkcia `getPicture`, avšak je volaná s odlišnými parametrami:

```
function getPhotoFromGallery() {
    navigator.camera.getPicture(onSuccess, onFail, { quality: 50,
        destinationType: Camera.DestinationType.FILE_URI,
        sourceType: navigator.camera.PictureSourceType.PHOTOLIBRARY
    });
}
```

Spätné volania *onSuccess* a *onFail* majú rovnakú funkcionalitu ako v prípade fotografie z kamery.

Sken QR kódov

Pre sken QR kódov je vhodný modul *com.phonegap.plugins.barcode-scanner*[2], ktorý je dostupný na GitHub účte Wildabeast. Tento modul je podľa služby PhoneGap Build najpoužívanejší. Základné vyvolanie okna pre skenovanie čiarového kódu je možné pomocou nasledovného kódu [32]:

```
cordova.plugins.barcodeScanner.scan(
    onSuccess, onFail
);
function onSuccess(result) {
    alert("We got a barcode\n" +
            "Result: " + result.text + "\n" +
            "Format: " + result.format + "\n" +
            "Cancelled: " + result.cancelled );
}
function onFail(error) {
    alert("Scanning failed: " + error);
}
```

Tento zásuvný modul podporuje aj skenovanie iných formátov čiarových kódov, ako napríklad: DATA_MATRIX, CODE_128, CODE_39, ITF, EAN_8, EAN_13.

Geolokácia

Pre funkcionalitu geolokácie som využil fakt, že má byť implementovaná štandartom HTML5 a nie je potrebný dodatočný zásuvný modul. HTML5 geolokácia má 2 hlavné funkcie pre vrátenie aktuálnej pozície.

- *getCurrentPosition* – funkcia ktorá vráti informácie o polohe iba raz
- *watchPosition* – funkcia, ktorá vracia údaje o polohe nepretržite, takže funkcionalitou pripomína GPS

2. https://github.com/wildabeast/BarcodeScanner

Volanie funkcí prebieha taktiež pomocou spätných volaní (callback):

```
navigator.geolocation.getCurrentPosition(success, error, options);
var options = {
    enableHighAccuracy: true,   timeout: 5000,  maximumAge: 0
};
function success(pos) {
    var crd = pos.coords;
    console.log('Your current position is:');
    console.log('Latitude : ' + crd.latitude);
    console.log('Longitude: ' + crd.longitude);
    console.log('More or less ' + crd.accuracy + ' meters.');
};
function error(err) {
    console.warn('ERROR(' + err.code + '): ' + err.message);
};
```

Ukážka volania funkcie getCurrentPosition [33].

Volanie obsahuje okrem iného aj nastaviteľné možnosti:

- zapnutie vysokej presnosti (*enableHighAccuracy*) - avšak povolenie vysokej presnosti na mobilných zariadeniach môže mať za následok vyššiu spotrebu batérie a taktiež môže viesť k pomalej odozve funkcie.

- maximálny čas, ktorý sa bude zariadenie pokúšať vrátiť svoju polohu (*timeout*)

- maximálny vek vrátenej pozície (*maximumAge*), v prípade že sa jedná o hodnotu z vyrovnávacej pamäte.

Funkcia *watchPosition* má argumenty totožné, avšak jej návratová hodnota je použitá v prípade zrušenia monitorovania polohy. Využíva sa to napríklad pri GPS navigácii, keď sa zariadenie dostane do cieľovej destinácie a nie je potrebné ďalej aktualizovať polohu. V tom prípade sa volá funkcia *clearWatch*.

Odloženie zablokovania obrazovky

Pre implementáciu funkcionality odloženia zablokovania obrazovky som použil modul Insomnia. Použitie je vcelku jednoduché, stačí volanie dvoch funkcí:

```
<button onclick="window.plugins.insomnia.keepAwake()">Keep screen awake</button>
<button onclick="window.plugins.insomnia.allowSleepAgain()">Allow lock</button>
```

Po bližšom študovaní kódu modulu som zistil, že na jednotlivých platformách je táto funkcionalita zaistená rôzne:

- Android – na tejto platforme sa využíva funkcia `addFlags`, ktorá umožňuje nastaviť rôznu funkcionalitu pomocou indikátorov (flag). V tomto prípade sa využíva indikátor `FLAG_KEEP_SCREEN_ON`. Na znovu povolenie zablokovania obrazovky je použitá funkcia `clearFlags` s totožným parametrom, ktorý tento príznak vymaže.
- iOS – využíva sa premenná `isIdleTimerDisabled`, ktorá sa nastavuje pomocou `setIdleTimerDisabled(true/false)`.
- Windows Phone – využíva vlastnosť `UserIdleDetectionMode` ktorej nastavuje `IdleDetectionMode.Disabled` / `IdleDetectionMode.Enabled`.

Vo vyššie uvedených prípadoch sa jedná o nastavenie, či má operačný systém po učitej dobe neaktivity od používateľa uviesť zariadenie do stavu spánku alebo nie.

4.1.7 Overenia identity servera

Pri HTTPS dopytoch sa kontroluje dôvera mobilného operačného systému v certifikát servera. V prípade dôvery v certifikát servera je dopyt úspešný, avšak v prípade nedôvery v certifikát servera sa na platforme Android v inštancii `WebView` nezobrazí žiadne varovanie alebo chyba a nie je vyvolaná žiadna udalosť.

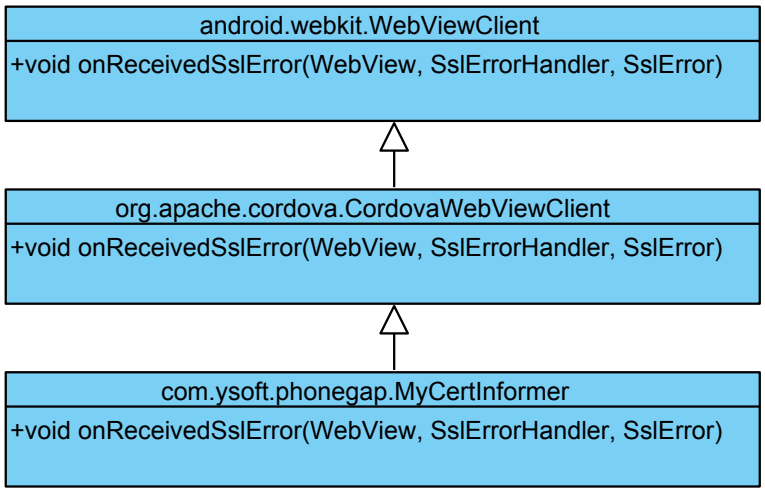
Príklad AJAX GET dopytu pre webovú adresu https://is.muni.cz/

```
function send() {
    jQuery.ajax(
        {
            type: "GET",
            url: "https://is.muni.cz/",
            success: function (data, status, jqXHR) {
                console.log(data);
                alert("Status: "+status);
            },
            error: function (jqXHR, status) {
                alert("Status: "+status);
            }
        }
    );
}
```

V prípade dôvery v certifikát serveru adresy https://is.muni.cz/ sa vykoná funkcia definovaná v parametri success, avšak v prípade nedôvery v certifikát sa nevykoná funkcia definovaná v parametri error.

Operačný systém Android poskytuje funkciu `onReceivedSslError`, ktorá je deklarovaná v triede `WebViewClient`. Túto funkciu prepisuje trieda `CordovaWebViewClient`, ktorá patrí do natívnej časti aplikačného rámca PhoneGap. Avšak v súčasnej implementácii funkcia iba kontroluje, či je aplikácia preložená s parametrom DEBUG (nastavený príznak `FLAG_DEBUGGABLE`), takže všetky SSL chyby ignoruje. Ak príznak nastavený nie je, tak iba volá funkciu z nadradenej triedy pomocou príkazu `super.onReceivedSslError`, takže do webovej časti neposiela žiadne varovanie o vzniknutej chybe.

Toto chovanie je možné zmeniť vlastnou implementáciou funkcie `onReceivedSsslError` (triedu som pomenoval `MyCertInformer`). Ukážka dedičnosti tried, ktoré implementujú funkciu `onReceivedSslError` je zobrazená na obrázku 4.3.



Obr. 4.4: Ukážka dedičnosti tried implementujúcich funkciu onReceivedSslError

V mojej implementácii funkcie `onReceivedSsslError` som využil zavolanie JavaScript udalosti `onSSLError`, na ktorú som si naviazal obsluhovač udalostí (event handler) vo webovej časti kódu aplikácie:

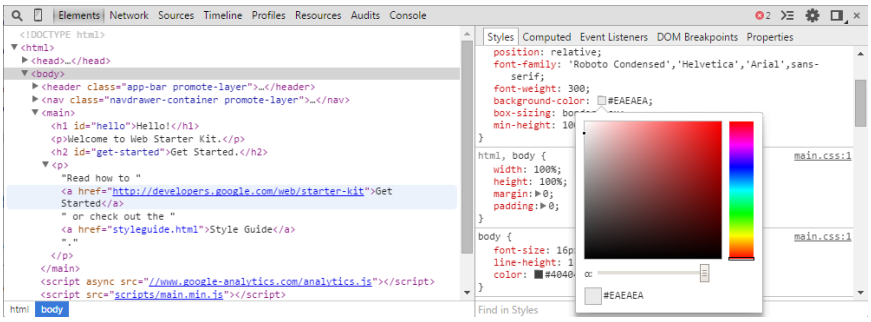
```
onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {
    ...
    String event = String.format("javascript: ",
                        "cordova.fireDocumentEvent('onSSLError');");
    view.loadUrl(event);
}
...
//Web part of application
document.addEventListener("onSSLError", function(){alert("SSL error!");});
```

Toto riešenie umožňuje zobrazenie varovania, v prípade akejkoľvek SSL chyby. V dôkladnej implementácii je možné využiť informácie z parametra typu `SslError`, ktorý obsahuje údaje o certifikáte, adresu servera a typ chyby (expirácia certifikátu, nevalidný dátum platnosti certifikátu, nedôveryhodný certifikát).

V ukážkovej aplikácii je možné túto funkcionalitu testovať v záložke `Https`, kde sú dve tlačidlá pre odosielanie dopytov. Avšak najprv je nutné v operačnom systéme vypnúť dôveru v ich certifikáty v nastaveniach (na platforme Android):
`Settings>Personal>Security>Credential storage>Trusted credentials`
(server seznam.cz používa certifikát od certifikačnej autority `Thawte` a server is.muni.cz používa certifikát od certifikačnej autority `Terena`).

4.2 Validácia a testovanie

Pri testovaní aplikácie som pri vývoji používal hlavne prehliadač Google Chrome. Prehliadač disponuje nástrojom Developer Tools, ktorý umožňuje prehliadanie zdrojového kódu, úpravu DOM elementov a CSS štýlov (pre vyvolanie ponuky Developer Tools stačí stlačiť tlačidlo F12). Taktiež je možné nástroj Developer Tools prepnúť do režimu, ktorý vzhľadovo simuluje mobilné zariadenie. Je k dispozícii výber z niektorých aktuálne dostupných zariadení na trhu. Taktiež je možné si definovať vlastnú výšku a šírku simulovaného zariadenia. Nástroj vie zachytiť a zobraziť prehľad CSS `media` dopytov, takže je vhodný na testovanie responzívneho dizajnu stránky. Súčasťou je aj nástroj na ladenie chýb (debugger) s možnosťou nastavenia podmienečného zastavenia (breakpoint) pre kód v jazyku JavaScript. Podobné nástroje pre vývojarov sú dostupné aj v prehliadačoch Internet Explorer, Firefox a Safari.



Obr. 4.5: Nástroj Developer Tools v prehliadači Google Chrome [38]

Vo fáze, keď je hotový HTML model aplikácie a používateľské prostredie, je nutná validácia funkcionalít typických pre mobilné zariadenia. To poskytujú emulátory pre jednotlivé platformy. Väčšinou sú emulátory súčasťou vývojárskych nástrojov, v ostatných prípadoch je nutná dodatočná inštalácia. Výhodou vyvýjania aplikácie na systéme Microsoft Windows je možnosť, že tento systém podporuje emulátory pre Android aj Windows Phone. Pre emulovanie iOS zariadení je nutný operačný systém Macintosh. Určitou možnosťou vývoja a validácie aplikácie na jednom počítači je použitie nástroja, ktorý umožňuje emuláciu operačných systémov (napríklad nástroj VMWare). Avšak validácia aplikácie iba na emulátoroch má svoje nevýhody v pomalej odozve emulátora. Taktiež na niektorých emulátoroch nie je prístup k funkcionalite ako kamera a bluetooth.

Preto ďalším krokom pri validácii aplikácie je spustenie na reálnych zariadeniach. Tieto zariadenia poskytujú najlepšiu odozvu o stave a funkčnosti aplikácie. Keďže sa jedná o aplikáciu, ktorá pozostáva z HTML elementov a JavaScript kódu, je dôležitý najmä prístup k tejto časti aplikácie. Tento postup sa nazýva Remote Debugging. V súčasnosti je podporované prehliadanie webovej časti kódu aplikácie, ktorá je spustená na zariadení u platforiem Android, iOS aj Windows Phone. Objekt WebView na platforme Android komunikuje s prehliadačom Google Chrome a je možné prezerať a ladiť webový kód spustený v tomto objekte. Podobne to funguje na platforme iOS, ktorá obsahuje objekt UIWebView, ktorý komunikuje s prehliadačom Safari. Na platforme Windows Phone je webový kód aplikácie spúšťaný taktiež v objekte s názvom WebView a tento kód je možné prehliadať a ladiť pomocou nástroja Microsoft Visual Studio.

Ďalšou možnosťou testovania je použitie unit testov. Tieto testy sa taktiež programujú v jazyku JavaScript. Na testovanie je možné použiť aj predpripravené aplikačné rámce ako Unit.js[3], QUnit[4] alebo YUI Test[5].

Výsledná aplikácia bola vyvíjaná a validovaná pomocou prehliadača Google Chrome na operačnom systéme Microsoft Windows. Validácia prebiehala za pomoci emulátorov pre Windows Phone 8 a Android 4. Pre otestovanie iOS prekladu aplikácie bolo nutné preloženie na zariadeni MacBook, s následným testom na iOS iPhone emulátore a zariadeni iPad. Validácia používateľského prostredia a funkcionality aplikácie bola konzultovaná s UX dizajnérom spoločnosti YSoft. Výsledné snímky obrazoviek spustenej aplikácie na platformách sú zachytené v prílohe A.

3. http://unitjs.com/
4. https://qunitjs.com/
5. http://yuilibrary.com/yui/docs/test/

5 Záver

Cieľom práce bolo navrhnúť a vytvoriť aplikáciu pre zobrazovanie tlačových úloh s využitím aplikačného rámca PhoneGap alebo Apache Cordova a demonštrácia vybraných funkcionalít. V teoretickej časti práce je prehľad rôznych typov aplikácií. Ďalej je v práci zahrnutá analýza internej komunikácie aplikačného rámca a analýza požadovaných funkcionalít. V práci sú taktiež rozobrané bezpečnostné trhliny aplikačného rámca, ktoré boli objavené a opravené v minulosti.

V praktickej časti práce je popísaný vývoj aplikácie a implementácia požadovaných funkcionalít. Ďalej je popísaný postup, ktorý bol použitý pre overenie validnosti požadovaných funkcionalít.

Pri vývoji aplikácie sa ukázalo, že vývoj multiplatformných aplikácií pomocou rámca PhoneGap a Apache Cordova môže byť v niektorých prípadoch jednoduchší a rýchlejší ako vývoj osobitných aplikácií pre každú platformu. Avšak použitie neštandartných alebo nových funkcionalít vyžaduje použitie zásuvných modulov, ktoré nemusia v dobe vývoja aplikácie existovať.

Literatúra

[1] RUDOLPH, Patrick. Hybrid Mobile Apps: Providing A Native Experience With Web Technologies. [online]. [cit. 2015-05-03]. Dostupné z: http://www.smashingmagazine.com/2014/10/21/providing-a-native-experience-with-web-technologies/

[2] FREY, Andy. There's More Than One Way to Build Mobile Apps. [online]. [cit. 2015-05-03]. Dostupné z: http://blog.meltmedia.com/2013/05/theres-more-than-one-way-to-build-mobile-apps/

[3] KORF, Mario a OKSMAN. Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options [online]. [cit. 2015-05-03]. Dostupné z: https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development

[4] RAO, Leena. Adobe Acquires Developer Of HTML5 Mobile App Framework PhoneGap Nitobi. [online]. [cit. 2015-05-03]. Dostupné z: http://techcrunch.com/2011/10/03/adobe-acquires-developer-of-\html5-mobile-app-framework-phonegap-nitobi/

[5] Plugin Development Guide. [online]. [cit. 2015-05-03]. Dostupné z: http://docs.phonegap.com/en/4.0.0/guide_hybrid_plugins_index.md.html#Plugin%20Development%20Guide_the_javascript_interface

[6] RANGARAJAN, Santosh. Apache Cordova – Workflow [online]. [cit. 2015-05-03]. Dostupné z: http://abstractlayers.com/2012/10/25/apache-cordova-workflow-part-2/

[7] Windows Phone 8 Platform Guide. [online]. [cit. 2015-05-03]. Dostupné z: https://cordova.apache.org/docs/en/4.0.0/guide_platforms_wp8_index.md.html

[8] Adobe PhoneGap Build Plans [online]. [cit. 2015-05-03]. Dostupné z: https://build.phonegap.com/plans

[9] Hydration [online]. [cit. 2015-05-03]. Dostupné z: http://docs.build.phonegap.com/en_US/tools_hydration.md.html#Hydration

[10] Usage of JavaScript libraries for websites [online]. [cit. 2015-05-03]. Dostupné z: http://w3techs.com/technologies/overview/javascript_library/all

[11] JQuery [online]. [cit. 2015-05-03]. Dostupné z: http://sk.wikipedia.org/wiki/JQuery

[12] JQuery Mobile Docs - Events [online]. [cit. 2015-05-03]. Dostupné z: http://demos.jquerymobile.com/1.0/docs/api/events.html

[13] LIEW, Kevin. 11 Multi-touch and Touch events Javascript libraries [online]. [cit. 2015-05-03]. Dostupné z: http://www.queness.com/post/11755/11-multi-touch-and-touch-events-javascript-libraries

[14] CSS direction Property [online]. [cit. 2015-05-03]. Dostupné z: http://www.w3schools.com/cssref/pr_text_direction.asp

[15] JQuery Mobile in RTL [online]. [cit. 2015-05-03]. Dostupné z: http://rtl-this.com/blog-entry/jquery-mobile-rtl

[16] Android or iPhone or Blackberry – How to detect using JavaScript [online]. [cit. 2015-05-03]. Dostupné z: https://jbkflex.wordpress.com/2012/01/18/android-or-iphone-or-blackberry-how-to-detect-using-javascript/

[17] Pure Android [online]. [cit. 2015-05-03]. Dostupné z: http://developer.android.com/design/patterns/pure-android.html

[18] Android Context Menus [online]. [cit. 2015-05-03]. Dostupné z: http://gurushya.com/android-context-menus/

[19] Positioning The Menu (ContextMenu) [online]. [cit. 2015-05-03]. Dostupné z: http://help.infragistics.com/Help/Doc/WindowsPhone/2012.2/CLR4.0/html/Positioning_The_Menu_(ContextMenu).html

[20] IPhone Context Menu [online]. [cit. 2015-05-03]. Dostupné z: http://www.iphone-tips-and-advice.com/iphone-context-menu.html

[21] Cancel button does not work on IOS 8 [online]. [cit. 2015-05-03]. Dostupné z: https://github.com/wildabeast/BarcodeScanner/issues/237

[22] Barcode Scanner SDK [online]. [cit. 2015-05-03]. Dostupné z: http://www.scandit.com/pricing/

[23] IOS / Apple iPhone 6 support [online]. [cit. 2015-05-03]. Dostupné z: https://github.com/chariotsolutions/phonegap-nfc/issues/139

[24] HTML5 Geolocation [online]. [cit. 2015-05-03]. Dostupné z: http://www.w3schools.com/html/html5_geolocation.asp

[25] Deprecate Android Geolocation plugin [online]. [cit. 2015-05-03]. Dostupné z: https://issues.apache.org/jira/browse/CB-5977

[26] Common Vulnerabilities and Exposures [online]. [cit. 2015-05-03]. Dostupné z: http://en.wikipedia.org/wiki/Common_Vulnerabilities_and_Exposures

[27] WILLOUGHBY, Ryan. Cordova Android 3.5.0 Patched With Security Updates [online]. [cit. 2015-05-03]. Dostupné z: http://phonegap.com/blog/2014/08/07/cordova-android-3_5_0-patched-with-security-fixes/

[28] BECHARD, Chris. Security Alert: Apache Cordova vulnerabilities in your Google Play app [online]. [cit. 2015-05-03]. Dostupné z: http://community.phonegap.com/nitobi/topics/security-alert-apache-\cordova-vulnerabilities-in-your-google-play-app

[29] STEWART, Ryan. Returning to our roots: Edge Code is now Brackets [online]. [cit. 2015-05-03]. Dostupné z: http://blogs.adobe.com/edgecode/returning-to-our-roots-edge-code-is-now-brackets/

[30] HOW TO UNPREFIX -WEBKIT-DEVICE-PIXEL-RATIO [online]. [cit. 2015-05-03]. Dostupné z: http://www.w3.org/blog/CSS/2012/06/14/unprefix-webkit-device-pixel-ratio/

[31] PhoneGap API Documentation: Camera [online]. [cit. 2015-05-03]. Dostupné z: http://docs.phonegap.com/en/edge/cordova_camera_camera.md.html

[32] WILLOUGHBY, Ryan. BarcodeScanner [online]. [cit. 2015-05-03]. Dostupné z: https://github.com/wildabeast/BarcodeScanner/blob/c74e37a/README.md

[33] Geolocation - Web API Interfaces [online]. [cit. 2015-05-03]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation/getCurrentPosition

[34] Add support for certificate challenges into Cordova Android [online]. [cit. 2015-05-03]. Dostupné z: https://issues.apache.org/jira/browse/CB-8328

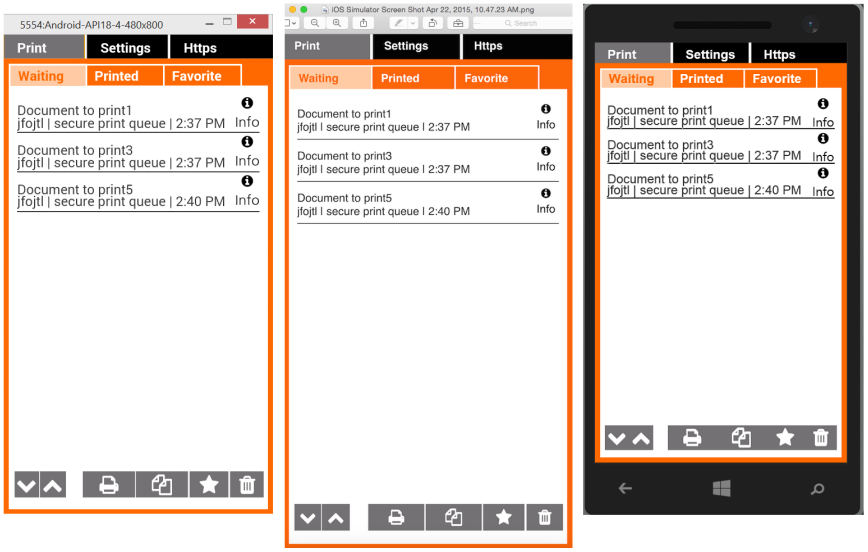
[35] Security for Windows Phone 8 [online]. [cit. 2015-05-03]. Dostupné z: https://msdn.microsoft.com/library/windows/apps/ff402533#BKMK_SecureSocketsLayerSSLcertificates

[36] Supported Features [online]. [cit. 2015-05-03]. Dostupné z: http://phonegap.com/about/feature/

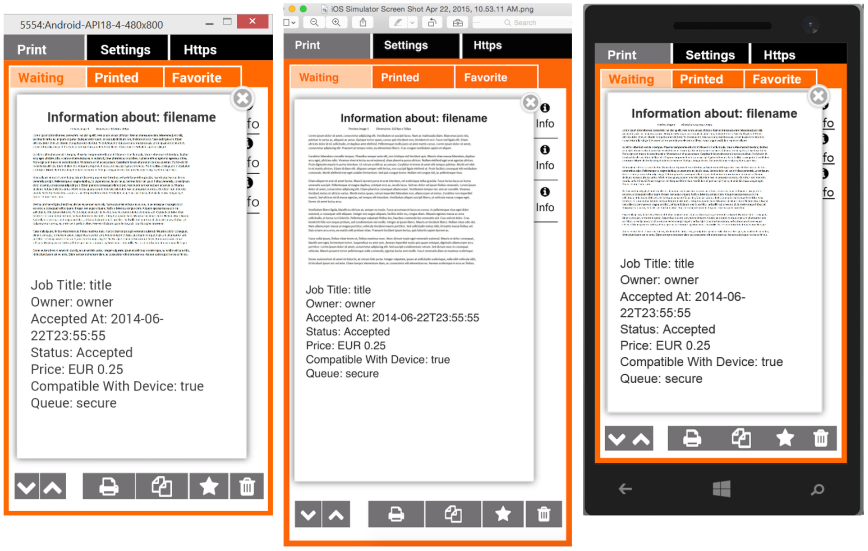
[37] GEORGIEV, Martin, Suman JANA a Vitaly SHMATIKOV. Breaking and Fixing Origin-Based Access Control in Hybrid Web/Mobile Application Frameworks [online]. [cit. 2015-05-03]. Dostupné z: https://www.cs.utexas.edu/~suman/publications/suman_ndss14.pdf

[38] Chrome DevTools Overview [online]. [cit. 2015-05-06]. Dostupné z: https://developer.chrome.com/devtools

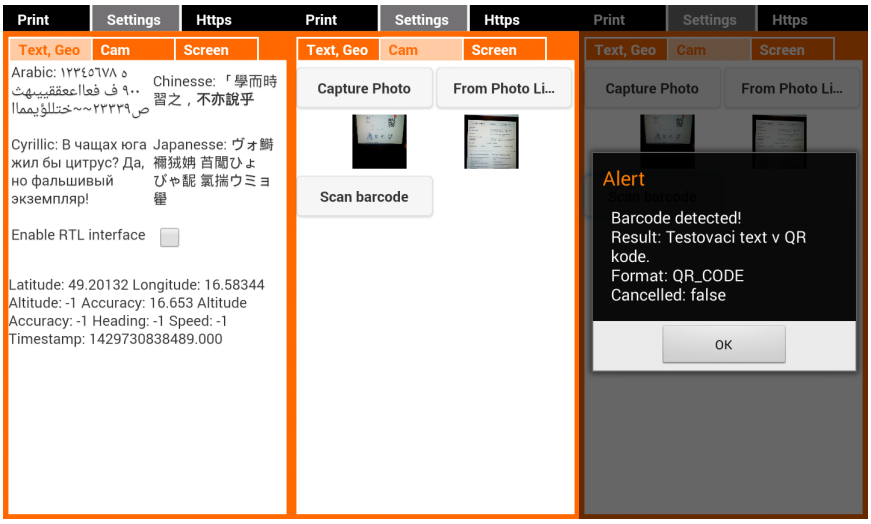
A Snímky obrazoviek



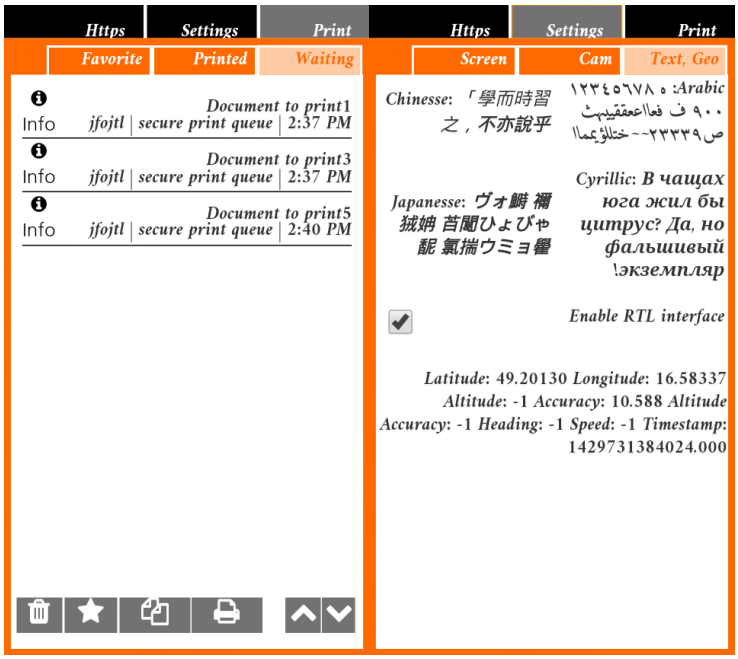
Obr. A.1: Snímky aplikácie. Zľava: Android, iOS, Windows Phone



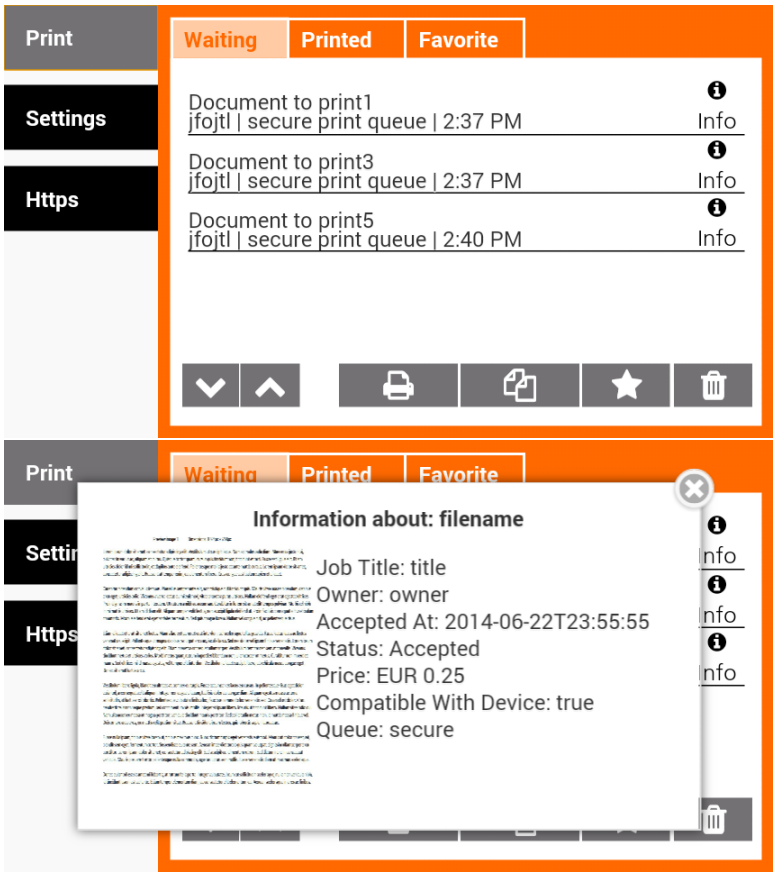
Obr. A.2: Snímky aplikácie, s náhľadom na úlohu. Zľava: Android, iOS, Windows Phone



Obr. A.3: Snímky zachytávajúce použitie fontov, kamery, prístupu do galérie, skenovania QR kódov a geolokácie. Zariadenie Lenovo A328, Android 4.4



Obr. A.4: Snímky zachytávajúce zapnutie funkcionality RTL. Zariadenie Lenovo A328, Android 4.4



Obr. A.5: Otočenie obrazovky na šírku.