Pedagogická fakulta
Faculty of Education

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Fakulta pedagogická

Katedra aplikované fyziky a techniky

Bakalářská práce na téma

Vývoj hybridních mobilních aplikací pro operační systémy Android a iOS

Vypracoval: Jiří Vávrů, DiS.

Vedoucí práce: Ing. Michal Šerý, Ph.D.

České Budějovice 2016

Abstrakt

Cílem této práce je zmapovat důvody a výhody vývoje hybridních mobilních aplikací za pomoci moderních frameworků a technologií jako například Cordova či AngularJS, které umožňují vývoje aplikací běžících na dnešních majoritních mobilních platformách jako Android, iOS, nebo Windows Phone.

Díky všeobecně dobré podpoře standardu HTML5 je dnes možné vytvářet univerzální aplikace pro mobilní zařízení s vlastnostmi a funkcionalitou aplikací nativních. Práce se tedy zaměřuje na zmapování možností v oblasti vývoje hybridních mobilních aplikací s výčtem výhod a nevýhod hybridního mobilního vývoje jako podkladu pro kvalitní analýzu aplikace před začátkem samostatného vývojového cyklu. Práce je doplněna o praktickou ukázku vývoje hybridní mobilní aplikace, která má detailně popsat proces vývoje od návrhu aplikace až po její realizaci.

Abstract

The main goal of this work is to explore the reasons and benefits of the development of hybrid mobile applications using modern technologies and frameworks, such as Cordova and AngularJS that enable the development of applications running on the majority of today's mobile platforms such as Android, iOS, or Windows Phone.

Thanks generally good support HTML5 is now possible to create a universal app for mobile devices with the features and functionality of native applications. The thesis is focused on mapping the possibilities in the development of hybrid mobile applications with a list of advantages and disadvantages of hybrid mobile development as a basis for quality analysis application before the independent development cycle. The work is complemented by a practical demonstration of the development of hybrid mobile application that has to describe in detail the development process from application design to its realization.

Prohlášení

Prohlašuji, že jsem svoji bakalářskou práci vypracoval(a) samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě – v úpravě vzniklé vypuštěním vyznačených částí archivovaných fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejich internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne (datum) .....

(jméno a příjmení)

Poděkování

Chtěl bych poděkovat Ing. Michalovi Šerému, Ph.D. za vedení mé bakalářské práce, cenné rady a odborný dohled.

Obsah

1 Úvod

Mobilní zařízení s trvalým připojením k internetu, tzv. „connected devices“, jsou obrovským hitem několika posledních let. Tato zařízení dokáží pro běžné použití nahradit několik samostatných zařízení a v budoucnosti bude jejich spektrum využití dále narůstat. Své obrovské popularitě vděčí právě tomu, že je pro jejich operační systémy možné vyvíjet aplikace s možností globální distribuce.

Nativní mobilní aplikace mají své výhody i nevýhody. Určitě je velmi nepraktické, když naprogramovaná aplikace funguje pouze na jedné platformě. Pokud tedy chceme, aby byla kompatibilní se všemi nejpoužívanějšími mobilními platformami, musíme napsat kód pro každou zvlášť.

Webové mobilní aplikace jsou obvykle naprogramovány do formy internetové stránky. Spouštějí se tudíž v internetovém prohlížeči, takže je třeba k jejich užívání přístup k internetovému připojení. Také je nemůžeme distribuovat přes prodejní platformy mobilních aplikací. Mají ovšem několik výhod – jsou přístupné všem mobilním platformám, přizpůsobují se zařízením, je snazší psát jejich kód i provádět aktualizace.

Hybridní aplikace jsou, jak jejich název napovídá, takovým mixem výše popsaných aplikací. Jejich kód se píše webovými technologiemi (HTML5, CSS3 a JavaScript) a je plně funkční pro všechny mobilní platformy. Zároveň se ale chová jako nativní mobilní aplikace a má skrze rozhraní přístup ke stejným hardwarovým komponentám a funkcím operačního systému jako aplikace.

Tato práce si klade za cíl provést čtenáře základními poznatky ohledně vývoje hybridních mobilních aplikací, které jsem získal během své dosavadní praxe a práce na projektech.

Publikace se tak snaží objektivně popsat současnou situaci v oblasti vývoje hybridních mobilních aplikací, zmínit klady, a skrze praktickou ukázku ukázat vývoj kompletní hybridní aplikace, včetně její finalizace a publishingu.

2 Důvody pro využití vývoje hybridních aplikací a jejich výhody a nevýhody

2.1 Důvody pro využití vývoje hybridních aplikací

Vývoj mobilních aplikací je vzhledem k dnešnímu množství mobilních platforem vysoce časově náročná činnost. Tato časová náročnost je jednak ovlivněna množstvím mobilních platforem. Namátkou uveďme majoritní systémy jako je například firma Apple s operačním systémem iOS, Google s operačním systémem Android, Microsoft se systémem WindowsPhone a v neposlední řadě forma BlackBerry se stejnojmenným operačním systémem. Jako druhý bod zvyšující časovou náročnost při vývoji mobilních aplikací je proprietární programovací jazyk každého z výše uvedených systémů. Kromě operačních systémů BlackBerry a Android (oba systémy využívají jako nativní programovací jazyk Java), využívá každý z uvedených operačních systémů jiný jazyk pro programování nativních aplikací.

Vzhledem k výše uvedenému je tak zcela evidentní, že vývoj aplikace pro jednotlivé platformy znamená, že každou aplikaci by bylo třeba napsat v nativním jazyce daného operačního systému, což vzhledem k časové a finanční náročnosti může v řadě případů představovat problém. Tuto situaci se v posledních několika letech snaží řešit takzvané hybridní mobilní aplikace, které se díky masivní podpoře ze strany velkých hráčů na poli vývoje aplikací (Google, Microsoft, Adobe a další) snaží dosáhnout vlastností nativních mobilních aplikací za použití technologií známých z dřívějších let především ze sféry programování webových aplikací.

Pojďme si nejprve v další kapitole představit hlavní výhody vývoje hybridních mobilních aplikací.

2.2 Výhody vývoje hybridních aplikací

Jak již bylo uvedeno v předchozí podkapitole, vývoj nativních mobilních aplikací pro majoritní platformy je časově náročná operace. Vývoj hybridních mobilních aplikací si klade za cíl snížit celkovou časovou náročnost na vývoj a testování mobilních aplikací. Mezi hlavní výhody tohoto stylu vývoje můžeme zařadit následující:

Jednotný kód – Znamená, že aplikační logika (potažmo i uživatelské rozhraní) aplikace je napsána za pomoci jednoho jazyka. Odpadá tak nutnost každou novou funkcionalitu či opravu chyby aplikace zapracovávat pro všechny platformy, na které je aplikace vyvíjena. Stačí tedy vše upravit na jednom místě a poté kód zkompilovat pro platformy aplikace.

Možnost testování úprav kódu bez nutnosti kompilace aplikace – Pokud je v kódu nativní aplikace provedena změna, je třeba změnu (zejména změny v uživatelském rozhraní) ručně otestovat. To obnáší potřebu aplikaci po provedených úpravách zkompilovat a spustit na zařízení nebo emulátoru. Vývoj hybridních mobilních aplikací umožňuje sledovat a testovat provedené změny v kódu přímo ve webovém prohlížeči za pomoci speciálního emulátoru. Tato možnost poskytuje velice silný nástroj pro ladění změn bez potřeby kompilace a nahrávání aplikace do zařízení. Výše uvedené platí pouze v případě, že aplikace pro svůj chod nepotřebuje v danou chvíli přístup k některým z hardwarových komponent, které z browseru nejsou dostupné, tzn. komponenty jako například NFC čtečka, senzory, či například čtečka otisku prstu.

Jednoduchá portace aplikace na další platformy – Pokud se v rámci vývojového cyklu aplikace stane, že je třeba aplikaci portovat i na některou další platformu, je za pomoci dnešních nástrojů velice jednoduché přidat další platformu, a s několika nezbytnými úpravami vyexportovat aplikaci na platformě, kterou potřebujeme.

2.3 Nevýhody vývoje hybridních aplikací

Vývoj hybridních mobilních aplikací má rovněž také svá úskalí a je třeba podotknout, že ne na každou aplikaci je tento způsob vývoje vhodný. Základem by tak před každým novým projektem měla být důkladná analýza a vyhodnocení potřeb a použití aplikace. Z vlastních zkušeností bych uvedl jako hlavní nedostatky vývoje hybridních mobilních aplikací následující.

Nekonzistentní funkcionalita skrze Web API – Hybridní mobilní aplikace je spuštěna v instanci tzv. webview, což je instance výchozího internetového prohlížeče v operačním systému. Verze tohoto prohlížeče s jádrem postaveném na platformě Webkit mohou být různé a zejména u starších zařízení, která nejsou ze strany výrobce již aktualizována, dochází k nekonzistenci funkcionality skrze Web API. Tento problém se v současné době dá řešit použitím vlastního jádra prohlížeče, které je distribuováno jako součást binárního souboru aplikace, viz například projekt Crosswalk (dostupné na https://crosswalk-project.org/).

Zpracování operací na jednom vlákně – Aplikace psaná v JavaScriptu spuštěná v rámci hybridní aplikace běží na hlavním vlákně, dlouhotrvající operace je tedy nutné provádět na separátním vlákně, což vyžaduje použití nativního programovacího jazyka pro daný operační systém.

Rychlost odezvy uživatelského rozhraní – V rámci vývoje mobilních aplikací je kladen velký důraz na rychlou odezvu uživatelského rozhraní na zadané vstupy od uživatele. Nativní aplikace má vždy grafické komponenty optimalizovány tak, aby odezva byla okamžitá. U hybridních aplikací byl toto dříve základní problém, se kterým jsem se během vývoje potýkal. V současnosti je již situace o mnoho lepší a například UI framework Ionic je důkazem toho, že dnes lze vyvinout hybridní aplikace se stejnými vlastnostmi odezvy UI jako aplikace nativní.

Paměťová náročnost – Hybridní aplikace mají vyšší paměťovou náročnost než aplikace nativní. Je to dáno tím, že místo v operační paměti zabírá vytvořená instance prohlížeče.

3 Současné použité technologie a jejich porovnání

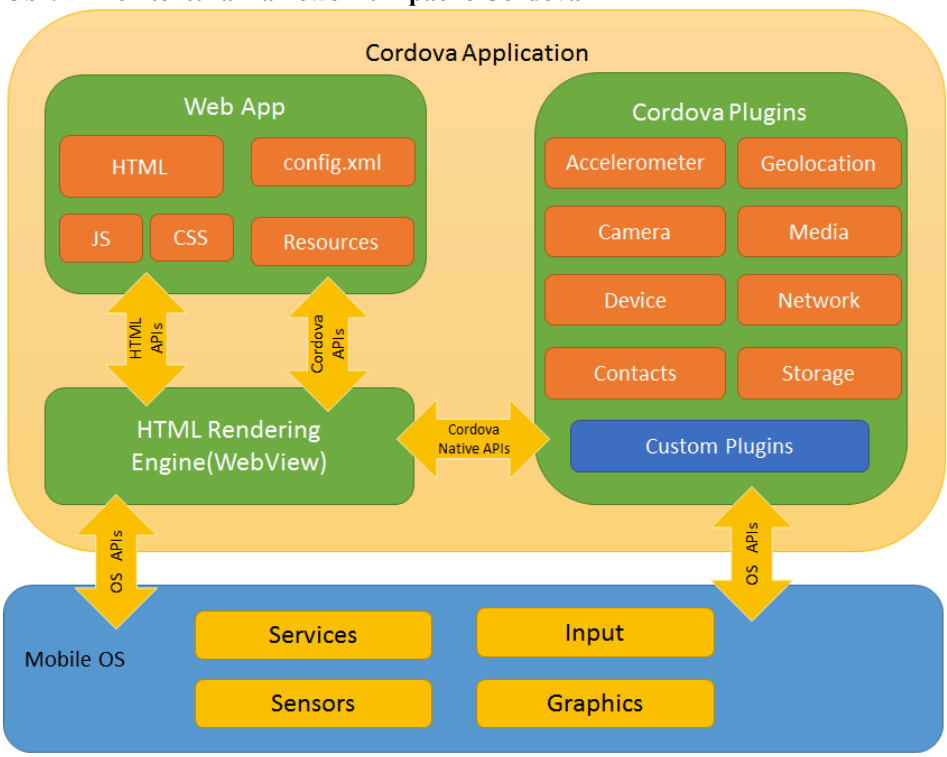
Vývoj hybridních mobilních je v současné době možný za pomoci několika technologií. V této bakalářské práci se budu zabývat vývojem mobilních aplikací za pomoci programovacího jazyka JavaScript, resp. jeho revize ECMAScript v6. Další používané technologie jako například Xamarin, Appcelerator jsou podrobněji popsány v části 3.5.

3.1 Apache Cordova

Tento framework (dostupný na https://cordova.apache.org) vznikl jako odnož frameworku PhoneGap. Jde o rozhraní (tzv. wrapper), které umožňuje přístup k nativním funkcím telefonu jako je například GPS, fotoaparát, senzory (například gyroskop, kompas) a další skrze javascriptové rozhraní (viz obr. 1). Tento framework je tak stěžejní částí pro vývoj hybridních aplikací, které mají stejné možnosti jako aplikace napsané v nativním jazyce. Jelikož se jedná o platformu, stojí určitě za zmínku silná podpora automatizace vývoje hybridních aplikací, což v našem případě znamená například možnost buildování a deploymentu aplikace za pomoci terminálu, nastavovaní konfigurace a instalace jednotlivých modulů.

Architektura frameworku Apache Cordova je podrobně znázorněna na obrázku níže.

Obr. 1 Architektura frameworku Apache Cordova



Stejně tak jako výčet funkcionalit napříč jednotlivými platformami zobrazený na obrázku na následující straně.

Obr. 2 Tabulka podpory přístupu k jednotlivým funkcím telefonu v závislosti na platformě

	android	blackberry10	ios	Ubuntu	wp8 (Windows Phone 8)	windows (8.1, 10, Phone 8.1)
cordova CLI	✔ Mac, Windows, Linux	✔ Mac, Windows	✔ Mac	✔ Ubuntu	✔ Windows	✔
Embedded WebView	✔ (see details)	✘	✔ (see details)	✔	✘	✘
Plugin Interface	✔ (see details)	✔ (see details)	✔ (see details)	✔	✔ (see details)	✔
	Core Plugin APIs					
Accelerometer	✔	✔	✔	✔	✔	✔
BatteryStatus	✔	✔	✔	✘	✔	✔ * Windows Phone 8.1 only
Camera	✔	✔	✔	✔	✔	✔
Capture	✔	✔	✔	✔	✔	✔
Compass	✔	✔	✔ (3GS+)	✔	✔	✔
Connection	✔	✔	✔	✔	✔	✔
Contacts	✔	✔	✔	✔	✔	partially
Device	✔	✔	✔	✔	✔	✔
Events	✔	✔	✔	✔	✔	✔
File	✔	✔	✔	✔	✔	✔
File Transfer	✔	✔ * Do not support onprogress nor abort	✔	✘	✔ * Do not support onprogress nor abort	✔ * Do not support onprogress nor abort
Geolocation	✔	✔	✔	✔	✔	✔
Globalization	✔	✔	✔	✔	✔	✔
InAppBrowser	✔	✔	✔	✔	✔	uses iframe
Media	✔	✔	✔	✔	✔	✔
Notification	✔	✔	✔	✔	✔	✔
Splashscreen	✔	✔	✔	✔	✔	✔
Status Bar	✔	✘	✔	✘	✔	✔ Windows Phone 8.1 only
Storage	✔	✔	✔	✔	✔ localStorage & indexedDB	✔ localStorage & indexedDB
Vibration	✔	✔	✔	✘	✔	✔ * Windows Phone 8.1 only

3.2 Adobe PhoneGap

Framework Adobe PhoneGap (dostupný na https://phonegap.com/) je velmi obdobný výše zmíněnému Apache Cordova. Architektura, API i funkce jsou shodné, rozdíl je v tom, že tento framework umožňuje jako platforma build a deploy aplikace v Cloudu. To v našem případě znamená, že build zdrojového kódu je prováděn na serverech společnosti Adobe a uživateli je vygenerována aplikace včetně podpisu certifikátem pro přímý publishing do některého z aplikačních marketů jako je například Google Play nebo Apple AppStore. Za zmínku určitě stojí i fakt, že u obou zmíněných platforem je možné vytvořit nejen hybridní aplikaci pro telefony, ale rovněž aplikaci pro klasické desktopové operační systémy jako je Windows, Linux nebo Mac OSX.

3.3 AngularJS

AngularJS (dostupný na https://angularjs.org/) je aplikační MVVM (Model View View Model) framework vyvinutý společností Google. Tento framework se stará o celou aplikační logiku aplikace, její strukturu a udržitelnost kódu. JavaScript byl v minulosti vyvinut jako skriptovací jazyk. Jeho nevýhodou byla složitá udržitelnost a přehlednost kódu v rámci projektu. S vývojem JavaScriptu a příchodem moderních javascriptových frameworků, jako je například zmíněný AngularJS, je možné, co se týče udržitelnosti kódu, konkurovat silně objektově orientovaným jazykům jako je například Java. Tento framework je možné samozřejmě použít jak pro vývoj hybridních aplikací, tak i pro vývoj webových aplikací. O jeho popularitě svědčí například fakt, že ho při vývoji webových aplikací používá například NetFlix, Google a další firmy (více ukázek je možné nalézt na https://www.madewithangular.com/). Obdobných frameworků, které je možné použít pro vývoj hybridních aplikací, je samozřejmě více, principiálně jsou si velice podobné. Jako příklad uveďme framework BackBoneJS (dostupný na https://backbonejs.org/).

3.4 Ionic

Framework Ionic (dostupný na http://ionicframework.com/) je primárně vyvinut pro použití při vývoji mobilních hybridních aplikací, ale lze jej použít i pro vývoj webové aplikace pro zobrazení v mobilních prohlížečích. Jedná se primárně o framework pro grafické uživatelské rozhraní. UI frameworků je veliké množství, na základě vlastních vývojářských zkušeností ale mohu framework Ionic prohlásit za jeden z nejlepších co se týče uživatelské komunity, dokumentace a rychlosti. Výběr správného UI frameworku je totiž pro použitelnost zcela zásadní, jelikož uživatel na rozdíl od webových aplikací očekává reakci uživatelského rozhraní ihned. Aplikace, respektive její uživatelské rozhraní a grafické komponenty, musí na vstupu od uživatele reagovat velice rychle. Tento fakt byl vždy jedním z důvodů, proč se k vývoji mobilních hybridních aplikací přistupovalo s jistými obavami.

Ionic je distribuován jako platforma v kombinaci se zmíněnými frameworky Apache Cordova a AngularJS a nabízí širokou řadu nástrojů, které vývojářům ulehčují vývoj mobilní aplikace. Jako příklad uveďme například nástroj pro rychlé prototypování aplikace (dostupný na http://ionic.io/products/creator).

3.5 Další technologie a frameworky (Xamarin, Appcelerator, a další..)

Jak již bylo zmíněno v úvodu, vývoj hybridních mobilních aplikací nemusí být nutně prováděn za použití technologií využívající jako programovací jazyk JavaScript. Existují i další platformy, které umožňují vývoj nativní aplikace. Na závěr této kapitoly bych rád představil některé z nich, se kterými jsem se během své profesní činnosti setkal.

3.5.1 Appcelerator

Tento framework (dostupný na http://www.appcelerator.com/) umožňuje vývoj nativní mobilní aplikace (tzn. aplikace, která je zkompilovaná v jazyce nativním pro danou platformu) za pomoci JavaScriptu. Platforma poskytuje rozhraní pro práci s nativními komponentami daného operačního systému a v rámci kompilace převede kód psaný v JavaScriptu do nativního kódu dané platformy. Kompilace aplikace probíhá v Cloudu služby Appcelerator a výsledkem je zkompilovaná nativní aplikace.

3.5.2 Xamarin

Xamarin (dostupný na https://www.visualstudio.com/cs-cz/features/xamarin-vs.aspx) je platforma pro vývoj mobilních aplikaci pro iOS, Android a WindowsPhone za pomoci kódu psaného v jazyce C# / .NET za dosažení 75 % až 100 % opětovného použití kódu mezi platformami. Aplikace napsané v Xamarinu a C# mají plný přístup k rozhraní API dané platformy a schopnost vytvářet nativní uživatelská rozhraní. Platforma je tak ideální pro vývojáře se zaměřením na .NET platformu, jelikož na základě znalosti jejich programovacího jazyka mohu vyvíjet pro několik dalších platforem naráz.

4 Vývoj ukázkové aplikace

Jedním z cílů této práce je i praktická ukázka vývoje reálné hybridní mobilní aplikace za použití většiny technologií zmíněných v předchozí kapitole. Jako ukázkovou aplikaci jsem zvolil aplikaci, která bude skrze REST API komunikovat se vzdáleným serverem, na který bude odesílat data ke zpracování a po zpracování následné data zobrazí uživateli. Aplikace bude pro svou funkčnost a demonstraci práce s komponentami telefonu pracovat s rozhraním fotoaparátu.

4.1 Definice vyvíjené aplikace a popis jejích vlastností

Cílem aplikace bude z vytvořené fotografie obličeje uživatele odeslané na server zjistit následující údaje.

- Přibližný věk uživatele
- Pohlaví uživatele
- Rasu uživatele
- Zda se uživatel usmívá

Rozpoznávání vlastností obličeje uživatele se odehrává na vzdáleném serveru, který jsem pro účel aplikace vyvinul. Popis funkcionality algoritmu pro detekci není součástí této publikace.

Popis použití aplikace uživatelem je následující:

- Aplikace po stisknutí tlačítka spustí na daném zařízení kameru
- Uživatel vyfotí svůj obličej
- Aplikace zobrazí view s náhledem fotografie
- V případě, že je uživatel s fotografií spokojen, odesílá ji ke zpracovaní na REST API
- V případě, že spokojen s fotografií není, může vyfotit další
- Aplikace po obdržení odpovědi ze serveru zobrazí výstup na samostatném view

4.2 Výběr technologií pro účely aplikace

Na základě technologií popsaných v třetí kapitole této práce budou při vývoji aplikace použity následující technologie a frameworky.

Uživatelské rozhraní: Ionic framework

- Aplikační logika: AngularJS
- Zajištění přístupu k nativním funkcím telefonu: framework Apache Cordova

4.3 Vývoj aplikace

Jelikož v rámci této práce chceme prezentovat celý vývojový cyklus aplikace, je třeba si nejprve třeba zajistit veškeré závislosti pro bezproblémový vývoj aplikace. Jedním z předpokladů je instalace veškerých náležitých softwarových závislostí.

4.3.1 Instalace NodeJS

Většina frameworků zmíněných v předchozích kapitolách se dnes instaluje za pomoci balíčkovacího systému, který můžeme znát například při použití OS Linux. Jelikož většina použitých technologií využívá jako programovací jazyk JavaScript, je jako správce balíčkovacího systému použit jazyk NodeJS (dostupný na https://nodejs.org/en/), který umožňuje běh JavaScriptu na serveru. NodeJS obsahuje správce balíčků s názvem NPM (Node Packaging Manažer) a právě ten pro nás bude stěžejním bodem pro instalaci dalších závislých softwarových komponent.

Pro instalaci NodeJS, postačí stáhnout odpovídající instalační balíček pro vaši platformu z webových stránek uvedených výše a nainstalovat za pomoci průvodce.

4.3.2 Instalace frameworku Ionic

Po instalaci NodeJS je třeba nainstalovat framework Ionic. Jelikož je Ionic distribuován včetně frameworků AngularJS, není třeba kromě frameworku Ionic a Apache Cordova instalovat již nic dalšího. Protože už máme z předchozího kroku nainstalovaný NodeJS a tím pádem dostupný balíčkovací systém NPM, stačí pouze v příkazovém řádku zadat následující příkaz.

```
$ npm install -g cordova ionic
```

Tímto příkazem nainstalujeme globálně do operačního systému frameworky Cordova a Ionic.

Poznámka: Pokud používáte operační systém Windows, je také možné použít IDE Visual Studio, které má NodeJS i Ionic integrované.

Po instalaci a stažení všech potřebných závislostí je možné přistoupit k další části, kterou je je vygenerování základního prázdného projektu v našem workspace tak, abychom mohli projekt otevřít v námi preferovaném IDE a modifikovat.

Pro vytvoření základního projektu se pomocí příkazového řádku přesuneme do adresáře, kde chceme náš nový projekt vytvořit, a spustíme následující příkaz.

```
$ ionic start nazevNasiNoveAplikace sidemenu
```

Tímto příkazem vytvoříme v aktuální cestě nový Ionic projekt s názvem nazevNasiNoveAplikace, na který se aplikuje jedna z projektových šablon, a to šablona s názvem sidemenu. Jak je již z názvy šablony znatelné, jedná se o šablonu, která nám vytvoří základní projekt s vysouvacím bočním menu, jelikož jej naše aplikace bude používat.

Takto vytvořený projekt si poté otevřeme v námi preferovaném IDE, na uvedených příkladech budu používat IDE s názvem WebStorm od společnosti JetBrains. Toto IDE je zaměřené primárně na vývoj javascriptových aplikací a má nativně zabudovanou podporu NodeJS.

4.3.3 Struktura projektu

Po otevření projektu v námi preferovaném IDE by se měl objevit projekt s následující strukturou.

```
├── hooks
├── platforms
├── plugins
├── resources
│   ├── android
│   └── ios
└── www
    ├── css
    ├── img
    ├── js
    ├── lib
    └── templates
```

Pro lepší pochopení struktury projektu a smyslu jednotlivých adresářů uvádím následující základní popis významu jednotlivých adresářů.

Hooks – Tento adresář je spjatý s použitím vlastních skriptů, které se mají provést během procesu buildování aplikace.

Platforms – Adresář obsahuje jednotlivé podadresáře s vygenerovaným nativním kódem pro jednotlivé platformy, pro které uživatel vyvíjí. Ve výchozím stavu je tento adresář prázdný, jelikož nebyla prozatím přidána žádná platforma, pro kterou by bylo třeba aplikaci vygenerovat. V pozdějších kapitolách bude názorně ukázáno, jak jednotlivé platformy přidat.

Plugins – Tento adresář obsahuje jednotlivé nainstalované pluginy platformy Cordova. Tyto pluginy zajišťují přístup k jednotlivým hardwarovým komponentám zařízení. Tyto pluginy jsou buď vyvíjeny komunitou Cordova, případně si každý vývojář může napsat plugin podle svých potřeb. Tyto pluginy se v rámci automatizace instalují pomocí příkazového řádku a jsou standardní součástí platformy Apache Cordova. Repositář jednotlivých pluginů je možno nalézt na webu https://cordova.apache.org/plugins/.

Resources – Obsahuje grafické podklady pro jednotlivé platformy.

www – Obsahuje vlastní aplikační logiku aplikace, jednotlivé views, použité grafické prvky, stylopisy a knihovny třetích stran, které nepracují s hardware na zařízení. Obsah tohoto adresáře obsahuje vygenerovanou šablonu pro aplikaci. Díky tomu je možné jednoduše zobrazit prototyp aplikace přímo v prohlížeči pomocí tzv. live reload serveru, který je součástí Ionic frameworku (v další z kapitol si ukážeme praktické použití). Celý obsah tohoto adresáře je pak při kompilaci zkopírován pro jednotlivé platformy a poté načten do webview v rámci nativní aplikace. Skrze javascriptové API prohlížeče jsou pak volány funkce jednotlivých pluginů psané v nativním kódu a tento kód je poté vykonán.

4.3.4 Spuštění aplikace ve webovém prohlížeči

Jak již bylo zmíněno v předchozích kapitolách, jednou z hlavních výhod vývoje hybridních aplikací za pomoci JavaScriptu je fakt, že aplikace je možno spustit před kompilací a spuštění na samotném zařízení také v emulátoru webového prohlížeče. To vše s okamžitým projevením změn kódu nebo grafických podkladů aplikace bez nutnosti znovu kompilovat celou aplikaci.

Tento způsob má samozřejmě i své limity v podobě, že ne vše se dá v emulátoru nasimulovat, a tak práce s některými hardwarovými komponentami, jako například čtečky otisku prstů, NFC čtečky nebo například s různými senzory, je třeba stále provádět na fyzickém zařízení.

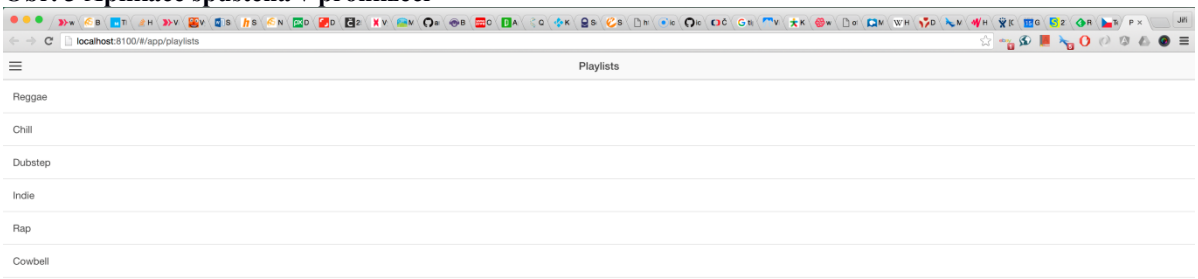
Abychom byli schopni zobrazit vygenerovanou základní aplikaci, zadáme v příkazovém řádku v rootu našeho projektu následující příkaz.

```
$ ionic serve
```

Po exekuci tohoto příkazu bude spuštěn jednoduchý http server, který v našem primárním webovém prohlížeči zobrazí obsah souboru index.html z adresáře www.

Výsledek pak bude vypadat jako na obrázku níže.

Obr. 3 Aplikace spuštěná v prohlížeči

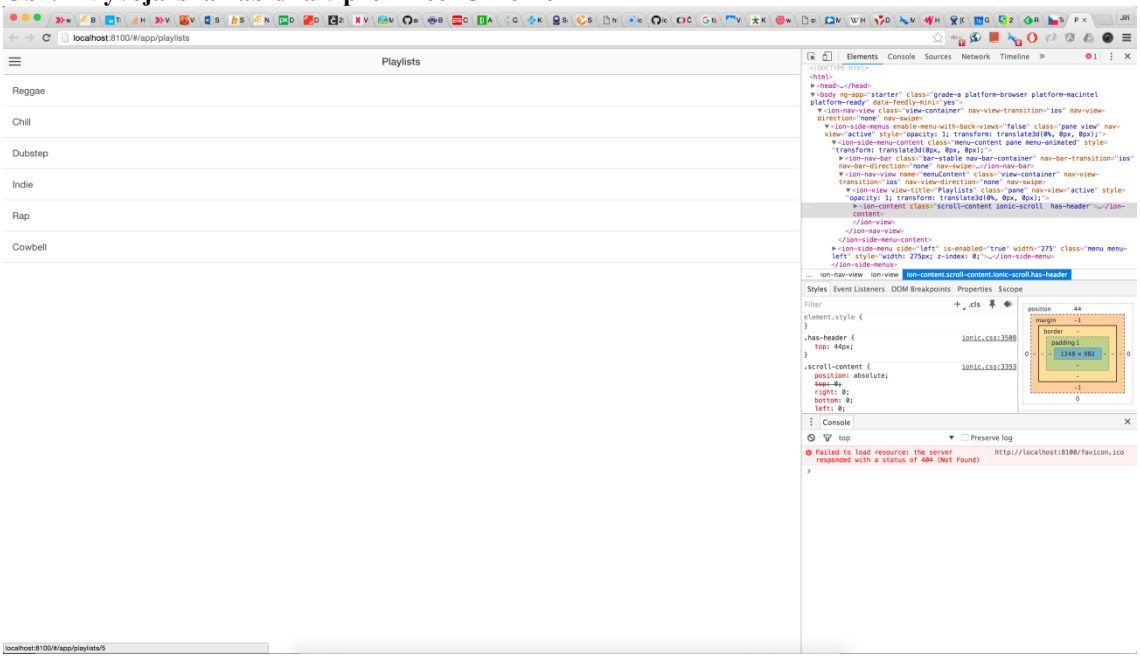


Aplikace je v tuto chvíli ale zobrazena v tzv. landscape režimu, tzn. jako kdyby byla zobrazena na tabletu nebo telefonu, který je položen na šířku. Prohlížeč Chrome, který má silnou podporu pro vývoj a debuggování hybridních mobilních aplikací psaných v JavaScriptu, ale nabízí možnost provést zobrazení, které více odpovídá povaze mobilních zařízení.

Změnu zobrazení jednoduše provedeme kliknutím na pravé tlačítko myši ve stavu, kdy je kurzor myši nad jakýmkoliv prvkem zobrazené aplikace.

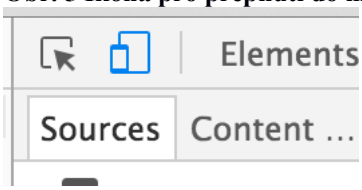
Z nabízených možností poté vybereme možnost „Inspect“, po kliknutí na tuto možnost se nám zobrazí vývojářská nabídka, viz obrázek níže.

Obr. 4 Vývojářská nabídka v prohlížeči Chrome



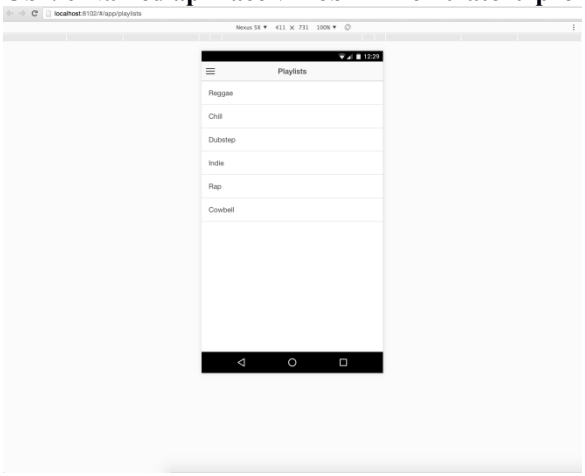
Nyní je třeba kliknout na ikonu v okně inspektoru se symbolem mobilního zařízení. Viz obrázek níže.

Obr. 5 Ikona pro přepnutí do mobilního emulátoru



Po přepnutí zobrazení do mobilního emulátoru uvidíme výsledek v následující podobě.

Obr. 6 Náhled aplikace v mobilním emulátoru prohlížeče Chrome



4.3.5 Úprava úvodní obrazovky

Vygenerovaný projekt obsahuje v rámci úvodní obrazovky seznam položek. Pro naše potřeby je třeba upravit úvodní view do podoby, která odpovídá potřebám naší aplikace. Úvodní obrazovka by měla obsahovat následující položky:

- Logotyp aplikace
- Tlačítko pro spuštění fotoaparátu pro pořízení fotografie obličeje
- Tlačítko pro výběr již existující fotografie z galerie v daném zařízení

Jelikož pracujeme s UI frameworkem Ionic, můžeme využít všech dostupných grafických komponent, které nám nabízí. Jejich podrobný výčet můžete nalézt na adrese http://ionicframework.com/docs/components/.

Pro úpravu view do požadované podoby si otevřeme soubor s obsahem úvodní stránky, který se nachází na cestě:

```
$ www/templates/playlists.html
```

Tento soubor představuje view aplikace použité jako hlavní stránka aplikace.

Obsah souboru nahradíme následujícím kódem:

```
<ion-view title="Face Age">
    <ion-nav-buttons side="left">
        <button menu-toggle="left" class="button button-icon icon ion-
navicon"></button>
    </ion-nav-buttons>
    <ion-content class="has-header">
        // Logotyp
        <img class="custom-centered" src="img/logo4.png">
        // Tlačítko pro spuštení fotoaparátu
        <button ng-click="takePictureUsingCamera()"
                class="button button-large button-full button-custom-blue icon-
left ion-camera">
            Take photo by camera
        </button>
        // Tlačítko pro spuštení galerie
        <button ng-click="takePictureUsingGallery()"
                class="button button-large button-full button-dark icon-left
ion-images">
            Take photo from gallery
        </button>
    </ion-content>
</ion-view>
```

Soubor playlists.html poté přejmenujeme na nový název home_page.html a provedeme rovněž úpravu v souboru na následující cestě:

```
$ js/app.js
```

Tento soubor slouží jako konfigurační soubor pro aplikace využívající aplikační framework AgularJS. Jde o soubor, ve kterém se nastavují jednotlivé url aplikace a k nim odpovídající view a controllery.

V konfiguračním souboru najdeme výskyt našeho předchozího přejmenovaného souboru playlists.html a přepíšeme následujícím kusem kódu.

```
.state('app.home', {
  url: "/home",
  views: {
    'menuContent' :{
      templateUrl: "templates/home_page.html",
      controller: 'HomeCtrl'
    }
  }
})
```

Jakmile bude v url aplikaci zadána cesta /home, bude dosazen view z cesty templates/home_page.html a použit controller s názvem HomeCtrl.

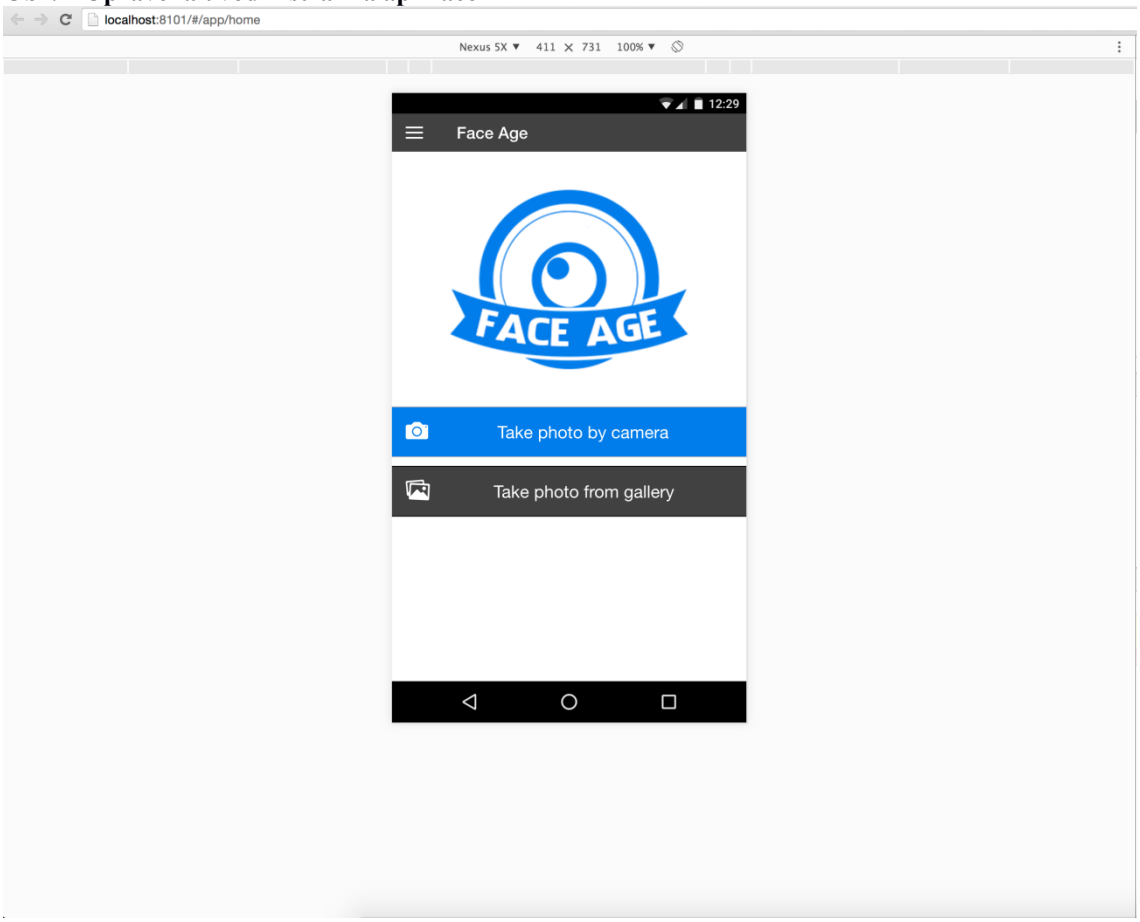
Za zmínku také stojí jednotlivá tlačítka, u prvního tlačítka je uveden následující zápis:

```
$ ng-click=”takePictureUsingCamera”
```

Tento zápis je typický pro aplikace psané za pomoci AngularJS a označuje název metody, která se má zavolat po kliknutí/tapnutí na tlačítko. Jelikož v souboru app.js je tomuto view přiřazen controller s názvem HomeCtrl, je po kliknutí na tlačítko zavolána výše uvedená metoda. Analogicky funguje metoda u druhého tlačítka.

Tímto způsobem jsme upravili úvodní view do požadované podoby a ve spuštěném emulátoru bychom měli vidět následující výsledek.

Obr. 7 Upravená úvodní stránka aplikace



4.3.6 Pořízení fotografie

Jak již bylo uvedeno výše, po kliknutí na tlačítko s názvem „Take photo by camera“ je zavolána odpovídající metoda v přiřazeném controlleru. Nyní se podrobněji podíváme na obsah metody pro pořízení fotografie.

Metoda má následující obsah:

```
$scope.takePictureUsingCamera = function() {
    var cameraOptions = {
        quality: 30,
        destinationType: Camera.DestinationType.DATA_URL,
        sourceType: Camera.PictureSourceType.CAMERA,
        mediaType: Camera.MediaType.PICTURE,
        encodingType : Camera.EncodingType.JPEG,
        saveToPhotoAlbum:false,
        targetWidth:500,
        targetHeight:500,
        correctOrientation:true
    };
    navigator.camera.getPicture( this.takePictureSuccess, this.takePictureError,
cameraOptions );
}
```

Jako první je v metodě definován objekt s názvem cameraOptions, který slouží pro konfiguraci vlastností spuštěného fotoaparátu a pořízené fotografie. Nastavují se zde vlastnosti jako je výstupní formát, kvalita fotografie, atd.

Následuje volání metody frameworku Apache Cordova s názvem navigator.camera.getPicture. Tato metoda má tři následující parametry:

- Callback pro úspěšné pořízení fotografie
- Callback pro chybu během pořizování fotografie
- Objekt s konfigurací pro fotoaparát

Objekt s konfigurací již máme hotový, nyní nás tedy čekají zbývající dva callbacky. Podíváme se nejprve na callback pro úspěšné pořízení fotografie.

```
$scope.takePictureSuccess = function(imageData) {
    $scope.showDetailModal();
    var image = document.getElementById('resultImage');
    // set image data to global scope
    $rootScope.globalImageData = imageData;
    image.src = "" + imageData;
};
```

Callback přijímá jako parametr objekt s výslednými daty z fotoaparátu.

V našem případě bude zpracování probíhat následujícím způsobem.

- Zobrazíme modální okno s vlastním obsahem pomocí metody *showDetailModal()*
- V zobrazeném modálním okně najdeme image tag s ID *resultImage*
- Jako *src* atribut obrázku nastavíme výstup z fotoaparátu ve formátu *base64*

4.4 Testování funkcionality na fyzickém zařízení

Abychom otestovali funkčnost kódu, je třeba nyní aplikace spustit na fyzickém zařízení, jelikož emulátor v rámci Chrome neumožňuje práce s fotoaparátem. Jak již bylo uvedeno dříve, v základu není přidána žádná platforma, pro kterou se má provádět build aplikace.

Pro naše potřeby chceme aplikaci spustit a otestovat na platformách Android a iOS. Je tedy třeba tyto dvě platformy přidat za pomoci následujících příkazů z příkazové řádky.

4.4.1 Přidání platformy Android

Přidání provedeme příkazem:

```
$ cordova platform add android
```

Poté co příkaz proběhne, můžeme zkontrolovat správnost instalace za pomocí příkazu:

```
$ cordova platform ls
```

4.4.2 Přidání platformy iOS

Přidání provedeme příkazem:

```
$ cordova platform add ios
```

Správnost opět můžeme zkontrolovat příkazem:

```
$ cordova platform ls
```

4.4.3 Kompilace a deploy Android

Pokud chceme aplikaci spustit na zařízení nebo emulátoru pro Android, je třeba provést následující příkazy:

```
$ cordova build android
```

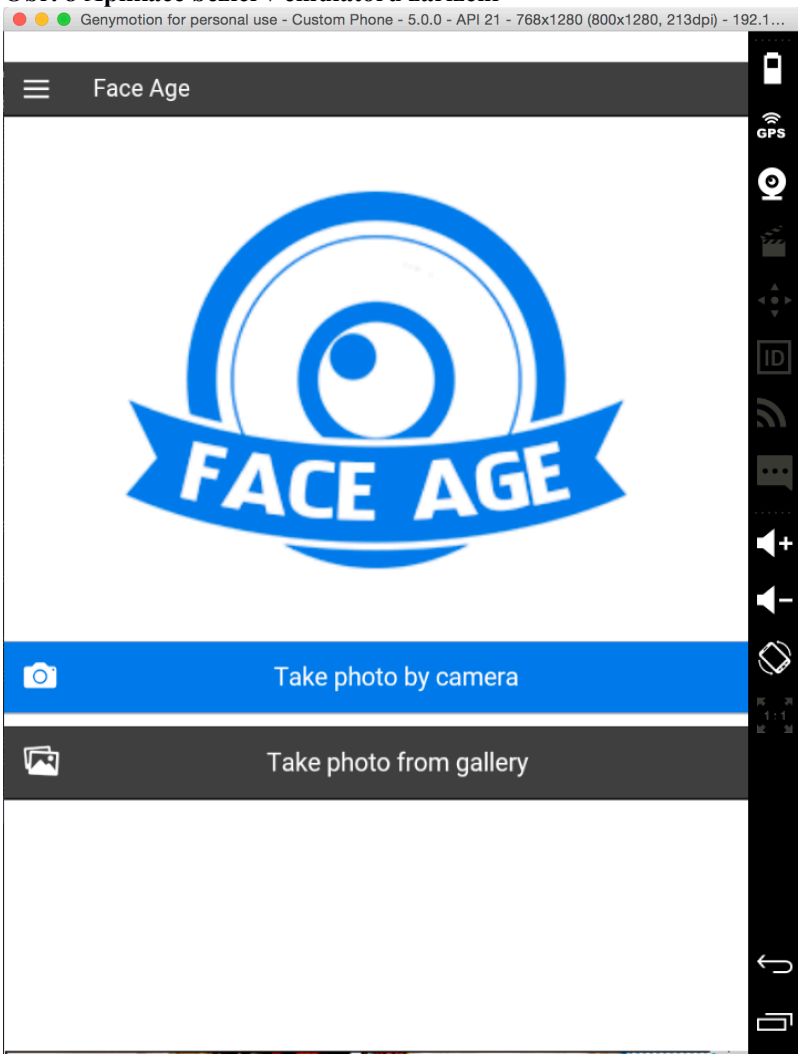
```
$ cordova run android
```

Komplikace proběhne pouze za předpokladu, že uživatel má na svém počítači nainstalován Android SDK tools. Více informací pro korektní instalaci Android SDK tools naleznete na následujícím odkazu (dostupné z http://developer.android.com/sdk/installing/index.html).

Případně je možné nainstalovat kompletní IDE určené pro vývoj aplikací pro platformu Android s názvem Android Studio od společnosti Google. V rámci jeho instalace je obsažena i instalace Android SDK, včetně integrace do použitého operačního systému.

Výsledkem by měla být aplikace spuštěná na emulátoru nebo na fyzickém zařízení za předpokladu, že je připojené fyzické zařízení a zapnutý USB debugging mode. Viz obrázek na další straně:

Obr. 8 Aplikace běžící v emulátoru zařízení



4.4.4 Kompilace a deploy iOS

Pokud chceme aplikaci spustit na zařízení nebo emulátoru pro Android, je třeba provést následující příkazy:

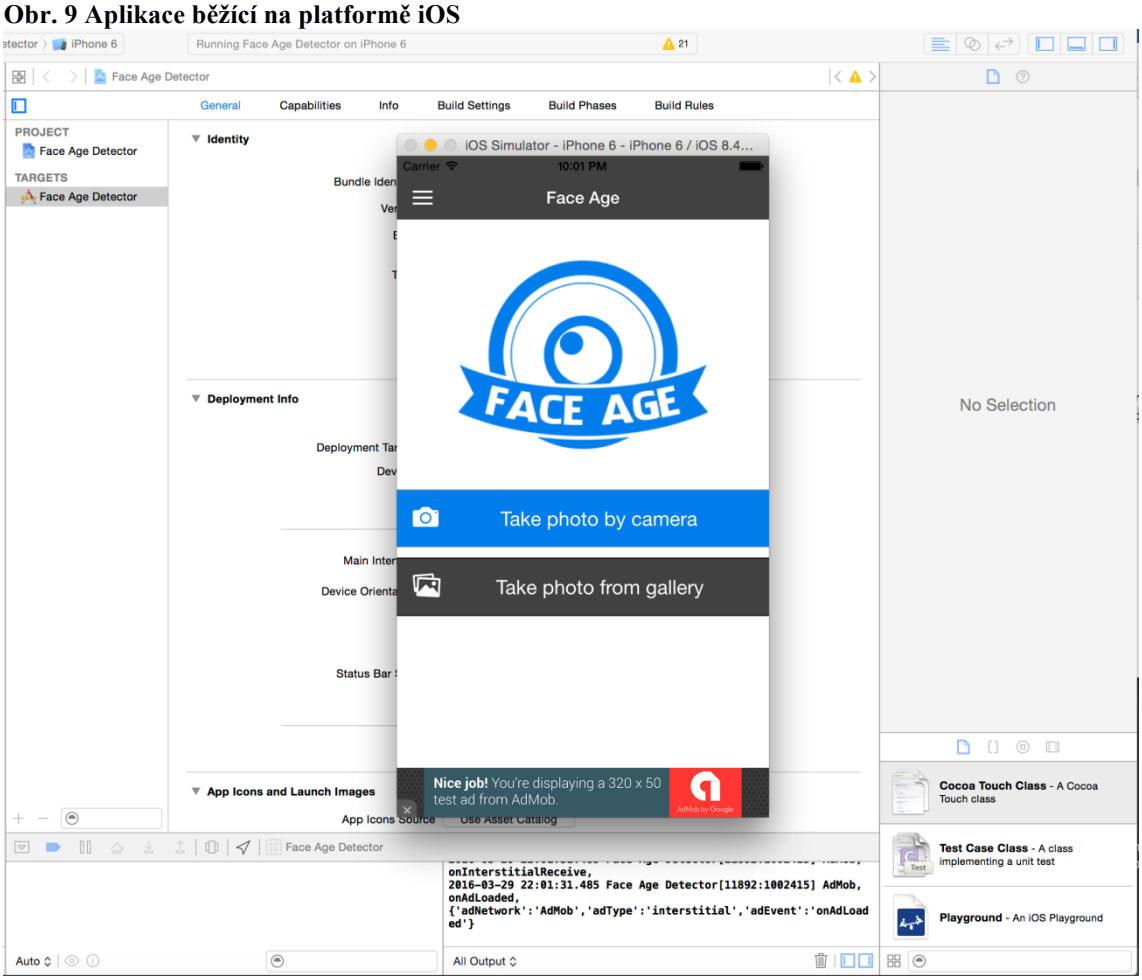
```
$ cordova build ios
```

```
$ cordova run ios
```

Kompilace proběhne pouze v případě, že uživatel spustil příkaz v rámci operačního systému Mac OSX a má nainstalován SDK s názvem X-code (nativní vývojové prostředí pro vývoj aplikací pro platformu iOS).

Výsledkem by měla být aplikace spuštěná na emulátoru, viz obrázek níže:

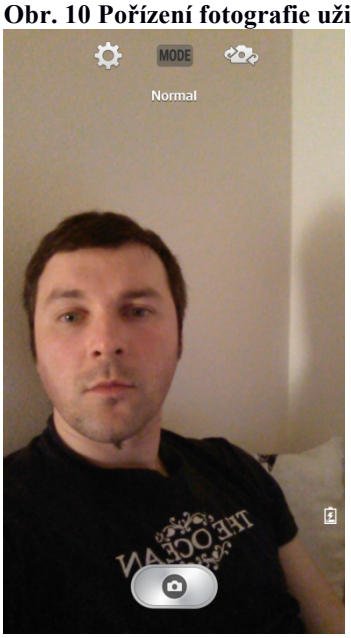
Obr. 9 Aplikace běžící na platformě iOS



4.4.5 Testování funkcionality práce s fotoaparátem

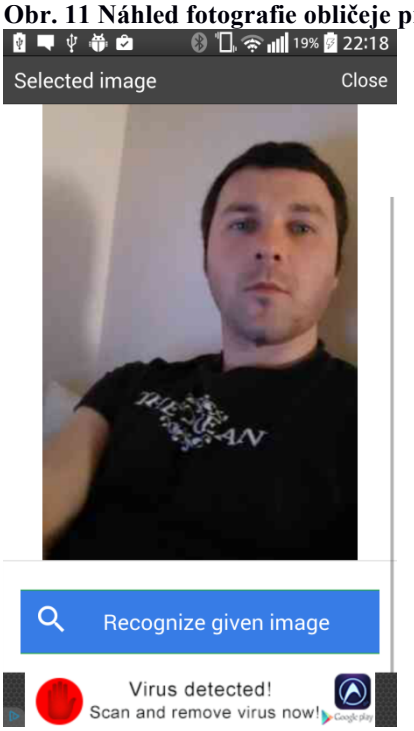
Pokud se nám podařilo aplikaci zkompilovat pro jednu z uvedených platforem, můžeme otestovat funkcionalitu kliknutím na tlačítko „Take photo by camera“ . Výsledkem by měla být spuštěná nativní aplikace fotoaparátu, viz obrázek níže:

Obr. 10 Pořízení fotografie uživatele fotoaparátem telefonu



Po pořízení fotografie se nám zobrazí modální okno s náhledem zvolené fotografie.

Obr. 11 Náhled fotografie obličeje před odesláním na zpracování vzdáleným serverem



Pořízení fotografie fotoaparátem za využití frameworku Apache Cordova nám tedy funguje. Nyní je třeba zapracovat na funkcionalitě, která umožní odeslání pořízené fotografie na vzdálený server, její zpracování a navrácení výsledku k zobrazení v mobilní aplikaci.

4.4.6 Odesílání pořízené fotografie na server a získání výsledků

Abychom mohli zobrazit výsledky analýzy pořízené fotografie, je třeba nejprve odeslat fotografii na vzdálený server, který ji zpracuje. Komunikace s tímto vzdáleným serverem probíhá skrze REST API. V aplikaci je tedy třeba implementovat funkcionalitu pro odesílání http requestu a příjem HTTP response.

AngularJS obsahuje metody umožňující jednoduše pracovat s HTTP protokolem.

Pokud tedy v modálním okně s náhledem fotografie stiskneme tlačítko s textem „Process Image“, vykoná se následující metoda, která provede následující sled operací.

ionicLoading.show: Tato metoda zobrazí tzv. toast message s daným textem a provede ztmavení a deaktivaci grafických prvků na pozadí do chvíle, než server navrátí zpracovaná data nebo do chvíle, kdy request selže.

postData: Proměnná s názvem postData obsahuje JSON objekt, ve kterém definujeme data odesílaná na zpracování. V našem případě odesíláme pouze zvolenou fotografii a to ve formátu base64.

headersData: Tato proměnná obsahuje definici HTTP hlaviček zasílaných společně s requestem.

Poté již zavoláme metodu s názvem *$http*, které předáme námi definované parametry a zvolíme typ HTTP request, který je v našem případě typu POST. V rámci metody poté definujeme dva callbacky, které metoda obsahuje.

Jako obvykle jsou to následující:

success: Definuje callback pro úspěšné provedení requestu a navrácení response ze vzdáleného serveru pro zobrazení v naší aplikaci. Pokud je tedy navrácena odpověď ze serveru během námi definovaného časového limitu (10 sekund), rozparsujeme odpověď a uložíme potřebné proměnné do scope, se kterým budeme dále pracovat ve view pro zobrazení výsledku. Na závěr provedeme skrytí grafického prvku preloaderu a zobrazíme nové modální okno s výsledky navrácenými ze serveru.

error: Definuje callback, který bude zavolán v případě, že dojde k chybě během zpracování požadavku na vzdálený server. Chyba může nastat z několika možných důvodů, tedy že server není dostupný, vypršel časový limit, server vrací chybový HTTP status, atd. Tyto informace můžeme získat z předávaného parametru *data* do callbacku. V našem případě zobrazíme uživateli obecnou alert hlášku o proběhlé chybě.

```
$scope.processImage = function() {

    $ionicLoading.show({
        template: 'Loading...'
    });

    // set post data
    var postData = {
        "img_base64": $rootScope.globalImageData
        };

    var headersData = {
        'Accept':'*/*',
        'Cache-Control':'no-cache',
        'Content-Type':'text/plain',
    };

    // set transfer credentials
    $http({
        method : 'POST',
        url : $scope.remoteUrl,
        data: postData,
        headers: headersData,
        cache: false,
        timeout: 10000
        // success response
    }).success(function(data, status, headers, config) {
        if(data['face'] != "") {
            $rootScope.age = data['face'][0].attribute.age.value;
            $rootScope.genderConfidence =
parseInt(data['face'][0].attribute.gender.confidence, 10);
            $rootScope.genderValue = data['face'][0].attribute.gender.value;
            $rootScope.smiling =
parseInt(data['face'][0].attribute.smiling.value, 10);
            $rootScope.attractivity = $rootScope.genderConfidence -
$rootScope.smiling;
            $rootScope.imageUrl = data.url;
            $scope.closeDetailModal();
            $scope.showResultModal();
        } else {
            var alertPopup = $ionicPopup.alert({
                title: 'Recognize error',
                template: 'Image does not contain human face'
            });
            alertPopup.then(function(res) {
                $scope.closeDetailModal();
            });
        }
        $ionicLoading.hide();
        // error response
    }).error(function(data, status, headers, config) {
        console.log(status);
        console.log(data);
        console.log(headers);
        $ionicLoading.hide();
        var alertPopup = $ionicPopup.alert({
            title: 'Processing error',
            template: 'Image cannot be processed, try it again'
        });
        alertPopup.then(function(res) {
            $scope.closeDetailModal();
        });
    });
};
```

4.4.7 Zobrazení výsledků navrácených ze serveru

V případě, že zaslaná fotografie na vzdálený server obsahuje fotografii lidské tváře v dostatečné fotografické kvalitě, navrácení výsledků je zobrazeno modální s odpovídajícími výsledky.

O toto se stará metoda s názvem *$scope.showResultModal* volaná v rámci *success* callbacku, která zobrazí obsah view s následujícím kódem.

```
<div class="modal">
  <ion-header-bar class="bar-dark">
    <h1 class="title">Result</h1>
    <div class="buttons">
      <button class="button button-clear" ng-
click="closeResultModal()">Close</button>
    </div>
  </ion-header-bar>
  <ion-content>
      <div class="list">
          <div class="item item-divider">
              Basic information
          </div>

          <a class="item item-icon-left" href="#">
              <i class="icon ion-person-stalker"></i>
              Gender
              <span class="badge badge-assertive">{{$root.genderValue}}</span>
          </a>

          <a class="item item-icon-left" href="#">
              <i class="icon ion-thermometer"></i>
              Gender confidence
              <span class="badge badge-assertive">{{$root.genderConfidence}}
%</span>
          </a>

          <a class="item item-icon-left" href="#">
              <i class="icon ion-clock"></i>
              Age
              <span class="badge badge-assertive">{{$root.age}} years</span>
          </a>


          <a class="item item-icon-left" href="#">
              <i class="icon ion-happy"></i>
              Smile
              <span class="badge badge-assertive">{{$root.smiling}} %</span>
          </a>

          <a class="item item-icon-left" href="#">
              <i class="icon ion-heart"></i>
              Attractivity
              <span class="badge badge-assertive">{{$root.attractivity}}
%</span>
          </a>
          <div class="item item-divider">
              Extended information
          </div>

          <a class="item item-icon-left item-icon-right" ng-
click="findSimilarFaces()">
              <i class="icon ion-earth"></i>
              Search similar faces on the internet
```

```
                <i class="icon ion-arrow-right"></i>
            </a>
        </div>
        <div class="button-bar">
            <a ng-click="closeResultModal()" class="button button-large button-
full button-custom-blue icon-left ion-camera">
                New photo
            </a>
            <a ng-click="shareApp()" class="button button-large button-full
button-custom-blue icon-left ion-social-facebook">
                Share
            </a>
        </div>
    </ion-content>
</div>
```

V kódu tohoto view stojí za zmínění princip, jakým AngularJS pracuje s předáváním hodnoty proměnných do view.

Veškeré dynamické hodnoty, se kterými pracujeme, a které jsou navráceny ze serveru, jsou ukládány do tzv. scope. V našem případě se jedná o globální scope, nazývaný také rootScope. AgularJS je MVVM framework, což umožňuje využití možností dvousměrného data-bindingu bez nutnosti využívat separátní aplikační logiku pro předávání dat z controllerů nebo jednotlivých servis do view.

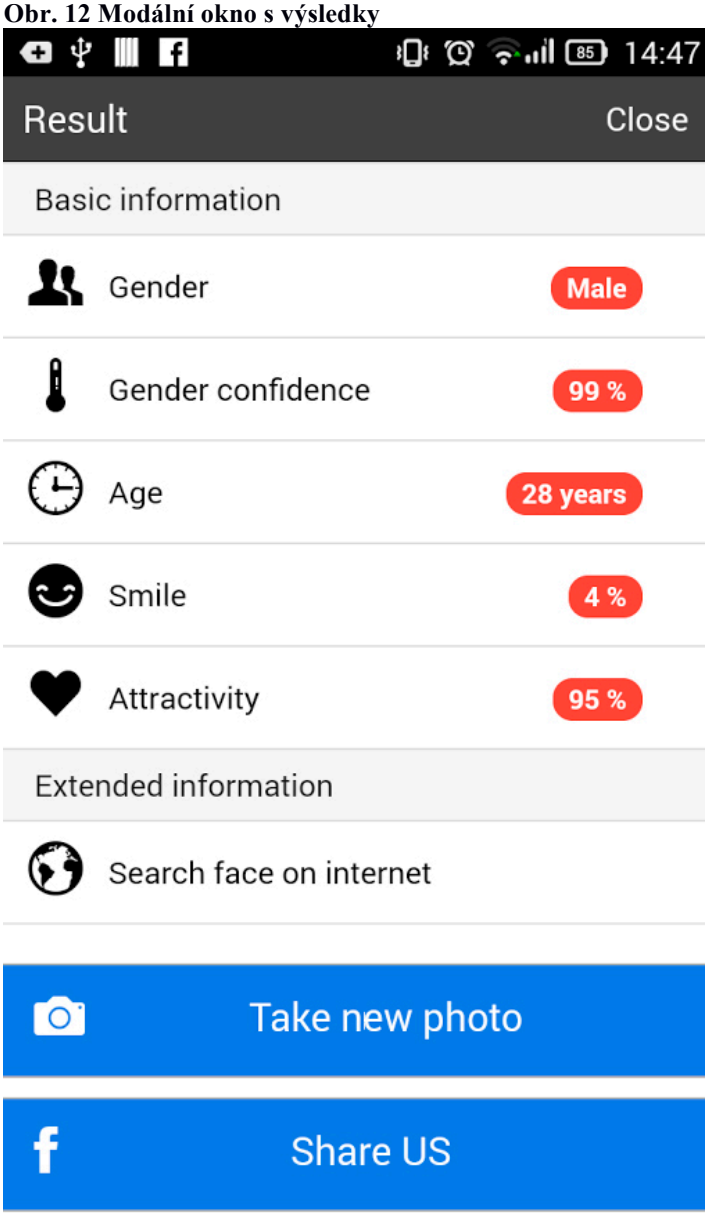
Zjednodušeně řečeno, ve chvíli, kdy přidáme hodnotu definované proměnné ve scope v rámci controlleru, je tato proměnná dostupná ve view. Jako příklad syntaktického zápisu v rámci view můžeme uvést například hodnotu {{$root.age}}, která obsahuje věk uživatele na fotografii navrácený v rámci odpovědi ze serveru. Tato hodnota je nastavena v rámci success callbacku do *rootScope* a v tuto chvíli je již možné hodnotu získat v kterémkoliv view. Ve chvíli, kdy by byla hodnota změněna (například v rámci periodicky se opakující metody), bude změna okamžitě provedena i ve view bez potřeby předávat separátně aktualizovaná data.

Více informací o tzv. dvousměrném data bindingu je možno nalézt na následující adrese: https://docs.angularjs.org/guide/databinding.

Další obsah view již vychází z klasických UX komponent dostupných v rámci Ionic frameworku. Jedná se konkrétně o jednotlivé položky list view s vlastními popisky a ikonami.

Výsledek po spuštění v emulátoru či zařízení bude vypadat jako na obrázku níže.

Obr. 12 Modální okno s výsledky



Po kliknutí na tlačítko „Take new photo“ bude uživatel přesměrován zpět na úvodní view.

4.5 Publikování aplikace

Abychom hotovou aplikaci mohli šířit za pomoci oficiálních distribučních kanálů pro danou platformu, je třeba nejprve aplikaci připravit pro publishing.

Tímto výrazem se v našem případě rozumí upload připravené a otestované aplikace do virtuálního aplikačního marketu dané platformy tak, aby byla po schválení ze strany provozovatele aplikačního portálu dostupná ve výsledcích vyhledávání. Postupy publishingu u jednotlivých platforem jsou si velmi podobné a pro potřeby naší ukázky jsem zvolil publikování aplikace v rámci platformy Android, což představuje upload aplikace na platformu Google Play.

4.5.1 Příprava pro publikování

Přípravou pro publikování se rozumí provedení několika následujících úkonů:

Otestování aplikace na reálném zařízení: Aplikaci je vždy třeba otestovat na reálném fyzickém zařízení, nikoliv jen v emulátoru. V případě Androidu se navíc potýkáme s problémem v podobě velké roztříštěnosti platformy mezi jednotlivými verzemi, takže je více než doporučováno vyzkoušet aplikace na více verzích systému Android (v rámci těch, které jsou podporované). Pro detailní ověření všech požadovaných vlastností aplikace s ohledem na funkcionalitu a uživatelskou přívětivost je možno využít následující odkaz (dostupný na: http://developer.android.com/distribute/tools/launch-checklist.html) s podrobným rozpisem.

Příprava grafických podkladů pro upload a textů: Otestovaná aplikace může být na Google Play nahrána pouze v případě, že jsou k ní přiloženy veškeré potřebné grafické podklady a promo texty. Jedná se primárně o následující položky:

- Ikona aplikace
- Prezentační grafika
- Screenshoty aplikace ze zařízení (telefon či tablet)
- Zkrácený popisek aplikace
- Detailní popisek aplikace

Detailní popis vlastností jednotlivých grafických prvků je k nahlédnutí na následujícím seznamu (dostupný na: https://support.google.com/googleplay/android-developer/answer/1078870?hl=en).

4.5.2 Podpis aplikace certifikátem

V případě, že máme připravené veškeré potřebné podklady pro publikování aplikace uvedené v předchozí kapitole, je třeba aplikaci před samotným uploadem podepsat vygenerovaným podpisovým certifikátem. Bez podpisu tímto certifikátem není možné aplikace do Google play distribuovat. Celou operaci lze při použití frameworku Apache Cordova rozdělit do dvou následujících částí:

Podpis aplikace: Pro podpis aplikace za použití frameworku Apache Cordova je třeba nejprve vytvořit v kořenovém adresáři projektu soubor s názvem *build.json*.

```
$ touch build.json
```

Do tohoto souboru poté vložíme následující konfiguraci pro podpis aplikace:

```
{
  "android": {
    "release": {
      "keystore":absolutni/cesta/k/souboru/keystore.jks",
      "storePassword": "",
      "alias": "keystore",
      "password" : "”,
      "keystoreType": ""
    }
  }
}
```

Jak je vidět z uvedeného příkladu výše, jedná se o konfigurační direktivu pro platformu Android, v níž definujeme absolutní cestu k podpisovému certifikátu, jeho alias a případně je možno vložit i heslo certifikátu. Pokud heslo vyplněno není, bude uživatel dotázán během procesu kompilace a podpisu aplikačního balíčku.

Upravený soubor uložíme a v příkazovém řádku zadáme následující příkaz:

```
$ cordova build android --release
```

Tímto příkazem spustíme proces kompilace aplikace a jejího podpisu. Pokud jsme při generování certifikátu zadali i jeho heslo, je nám zobrazeno modální okno pro zadání tohoto hesla.

Jakmile je kompilace dokončena, nalezneme podepsaný aplikační soubor s koncovkou .apk na následující cestě relativní ke kořenovému adresáři projektu.

```
$ platforms/android/build/outputs/apk
```

Takto vygenerovaný soubor je připraven pro upload do aplikačního marketu Google Play.

4.5.3 Registrace vývojářského účtu

Abychom mohli aplikaci nahrát do aplikačního marketu, je třeba nejprve zaregistrovat tzv. vývojářský účet, který slouží jako prostředek pro nahrání aplikace do aplikačního marketu.

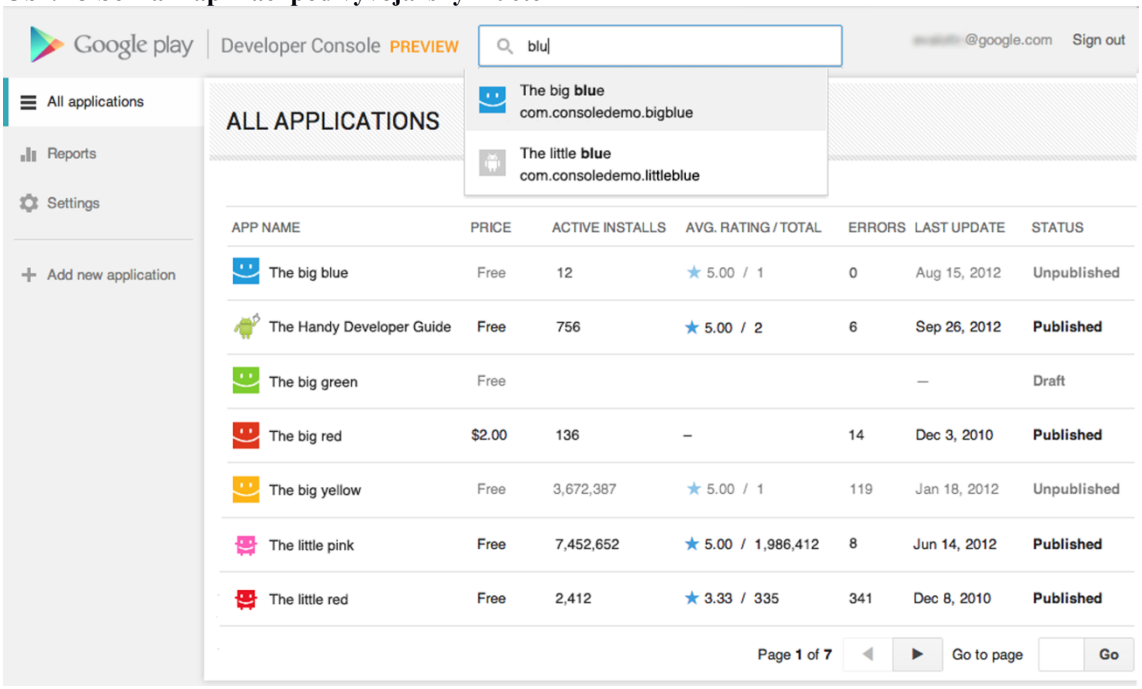
V našem případě nahráváme aplikaci do aplikačního marketu firmy Google s názvem Google Play, proto je třeba nejdříve provést registraci na následující adrese (dostupné na https://developers.google.com), vyplnit požadované údaje a zaplatit registrační poplatek ve výši 25 USD.

Poté je již možné se přihlásit pod nově vytvořený vývojářský účet (dostupné na https://play.google.com/apps/publish/).

4.5.4 Upload aplikace

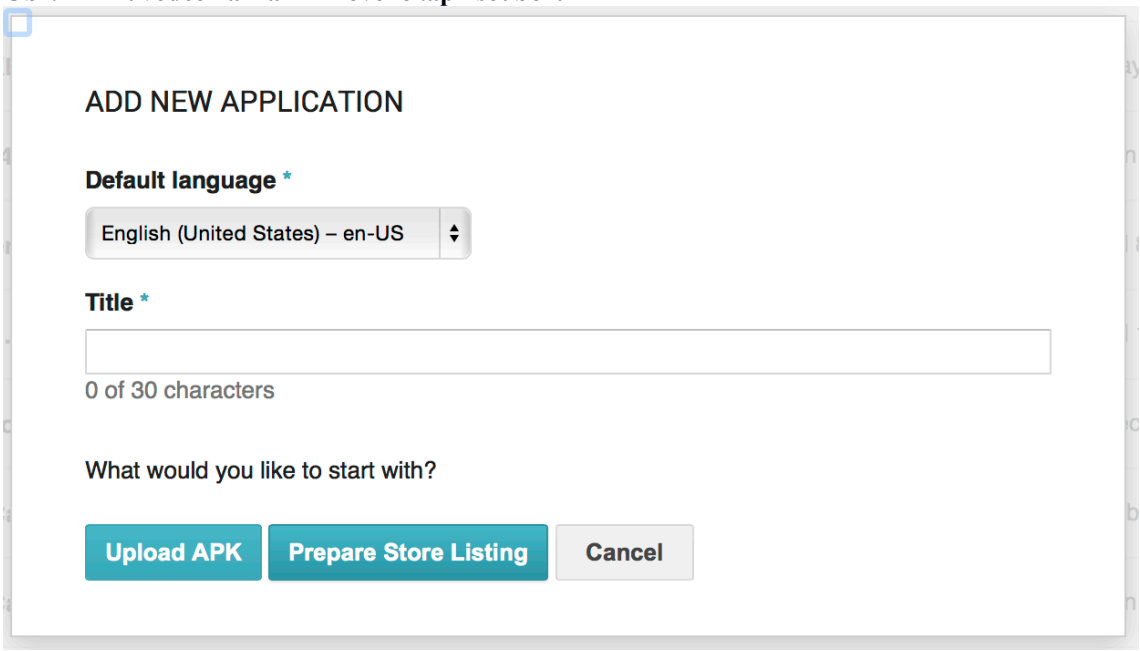
Po přihlášení do našeho vývojářského účtu uvidíme jako první seznam všech našich aplikací, které v rámci Google Play účtu spravujeme. Pokud máme nově založený účet, pak máme tento seznam aplikací prázdný.

Obr. 13 Seznam aplikací pod vývojářským účtem



Pro upload naší aplikace klikneme na tlačítko „Add new application“ v pravém horním rohu. Tím se nám zobrazí průvodce pro nahrání nového .apk souboru aplikace, viz následující obrázek:

Obr. 14 Průvodce nahráním nového .apk souboru



V rámci tohoto průvodce zadáme název aplikace a klikneme na tlačítko s názvem „Upload APK“.

V dalším kroku vybereme náš .apk soubor a počkáme na dokončení uploadu. Pokud soubor splňuje veškeré požadavky pro upload (maximální velikost souboru, vypnutý debuggovací mód, atd.) je zobrazena potvrzovací hláška o úspěšném nahrání .apk souboru a my můžeme úspěšně pokračovat k dalšímu kroku, kterým je upload promo textů a grafických podkladů.

4.5.5 Zadávání popisu aplikace a upload grafických podkladů

Pro zadání textových popisů aplikace a upload grafických podkladů se přepneme na záložku s názvem „Store Listing“ a vyplníme následující položky:

- Titulek aplikace
- Krátký popis aplikace
- Plný popis aplikace
- Screenshoty obrazovek aplikace z telefonu případně tabletu, Android TV nebo Android Wear
- Ikonu aplikace ve vysokém rozlišení
- Propagační a promo grafiku

Následně pak provedeme zařazení naší aplikace do odpovídající kategorie a typů aplikací. Vyplníme formulář pro hodnocení obsahu aplikace a zadáme kontaktní údaje vývojáře.

4.5.6 Nastavení ceny a distribučních rozsahů pro aplikaci

V posledním kroku je třeba definovat, zda aplikace bude na Google Play dostupná zdarma nebo za poplatek, a také v kterých zemích bude aplikace distribuována. Pokud je aplikace jednou zveřejněna jako neplacená aplikace, není ji možné později změnit na placenou a naopak.

Naopak distribuční rozsahy pro aplikaci je možno libovolně editovat. Pokud má být aplikace publikována jako placená, je možné nastavovat rozdílné ceny pro jednotlivé země.

Na závěr je třeba odsouhlasit licenční podmínky služby Google Play pro upload a distribuci aplikací a poté kliknout na tlačítko „Submit app to review“.

Pokud jsou všechny potřebné údaje pro publikování aplikace správně vyplněné, zobrazí se hláška, která nás informuje o tom, že aplikace byla podstoupena na schválení.

Schvalování aplikací v Google Play probíhá v současnosti pověřeným pracovníkem firmy Google a o jeho výsledku jsme informováni e-mailem.

Pokud byla aplikace schválena, je zpravidla do několika hodin dostupná ve vyhledávání aplikační platformy Google Play.

Pokud byl při schvalování naší aplikace nalezen nějaký problém bránící zveřejnění aplikace, jsme na tuto skutečnost opět upozorněni e-mailem a je třeba provést opravu a nahrát novou verzi .apk souboru.

5 Zhodnocení a závěr

Tímto je aplikace kompletně hotová a publikována v aplikačním marketu Google Play. Čtenáři byl v rámci této bakalářské práce vysvětlen princip vývoje hybridních mobilních aplikací, včetně názorné ukázky vývoje reálné aplikace a jejího publishingu do aplikačního marketu. Čtenář také získal přehled o technologiích používaných v dnešní době pro vývoj hybridních mobilních aplikací a měl by tak být schopen se dobře zorientovat v rámci analýzy vyvíjené aplikace pro platformu, která bude vyhovovat potřebám na funkcionalitu aplikace.

5.1 Časová úspora

Jako hlavní benefit hybridního vývoje lze na základě zveřejněných podkladech zmínit časovou úsporu tohoto typu vývoje aplikací pro mobilní platformy. Aplikaci můžeme v současném stavu velice jednoduše zkompilovat a nahrát na další aplikační platformy jako je například AppStore od společnosti Apple či WindowsStore od společnosti Microsoft.

5.2 Udržitelnost kódu

V rámci zhodnocení této práce je třeba vyzdvihnout jednoduchou a dlouhodobou udržitelnost kódu aplikace. V případě že budeme chtít v budoucnu aplikaci upravit a přidat například novou funkcionalitu, upravujeme jednotný zdrojový kód pro všechny platformy, nikoliv několik samostatných a vzájemně nekompatibilních aplikací, které ale ve výsledku provádějí tu samou úlohu.

5.3 Závěr

Vzhledem k rozšířenosti a obrovskému růstu popularity tzv. connected devices, což jsou primárně chytré telefony a tablety, roste poptávka po kvalitním programovém vybavení napříč jednotlivými mobilními platformami.

Svět vývoje software v segmentu pro mobilní platformy je dynamicky se měnící prostředí, ve kterém je kladen důraz zejména na kvalitu, udržitelnost a cenu vývoje.

Vývoj hybridních aplikací pro mobilní platformy má dlouhodobě velký potenciál a jedná se v posledních 3 letech o velmi rychle rostoucí segment trhu, ve kterém jsou zapojeni téměř všichni velcí hráči ze světa vývoje software.

Jako každá platforma či programovací jazyk má vývoj hybridních aplikací své přednosti ale i nedostatky, tato práce se je snaží objektivně porovnat a prezentovat výhody a postupy za použití vývoje hybridních mobilních aplikací pro současné mobilní platformy. Po prostudování této publikace by měl být čtenář schopen porozumět základním principům a použití hybridního mobilního vývoje aplikací.

6 Zdroje

6.1 Seznam literatury

1. Vávrů, Jiří, Programujeme pro Android. Druhé vydání. Praha: Grada Publishing, 2013. ISBN 978-80-247-4863-4

6.2 Internetové zdroje

Oficiální webové stránky frameworku Ionic. ionicframework.com [online]. [vid. 2016-03-30]. Dostupné z: http://www.ionicframework.com/

Oficiální webové stránky frameworku Apache Cordova. cordova.apache.org/ [online]. [vid. 2016-03-30]. Dostupné z: https://cordova.apache.org/

Oficiální webové stránky frameworku PhoneGap. phonegap.com [online]. [vid. 2016-03-30]. Dostupné z: http://phonegap.com/

Oficiální webové stránky vývojářské komunity pro operační systém Android. developers.google.com [online]. [vid. 2016-03-30]. Dostupné z: https://developers.google.com

7 Citace

Obrázek č.1 :
Apache Cordova framework [online]. [cit. 2016-03-30]. Dostupné z:
https://cordova.apache.org/docs/en/latest/guide/overview/

Obrázek č.2 :
Apache Cordova framework [online]. [cit. 2016-03-30]. Dostupné z:
https://cordova.apache.org/docs/en/latest/guide/overview/

Obrázek č.13 :
Google Play Developer Console [online]. [cit. 2016-03-30]. Dostupné z:
https://play.google.com/apps/publish/
Obrázek č.14 :
Google Play Developer Console [online]. [cit. 2016-03-30]. Dostupné z:https://play.google.com/apps/publish/

8 Seznam obrázků a tabulek