VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2016 Jiří Vaculík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jiří Vaculík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: AutoCont CZ a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Mgr. Jiří Dvorský, Ph.D.**

Konzultant bakalářské práce: Bc. Jan Mudrák

Datum zadání: 01.09.2015

Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

V Ostravě 29. dubna 2016

.....

AutoCont
AutoCont CZ a.s.
Hornopolní 34, 702 00 Ostrava
Tel.: 910 971 111, fax: 910 970 100
DIČ: CZ47676795

Rád bych na tomto místě poděkoval firmě AutoCont CZ a.s. za umožnění vykonání bakalářské praxe. Také bych chtěl poděkovat svému odbornému vedoucímu, Bc. Janu Mudrákovi a kolegovi Bc. Jakubu Vašicovi, který se mnou na většině zadaných úkolů spolupracoval.

Abstrakt

Tato bakalářská práce popisuje působení ve firmě AutoCont CZ a.s. v rámci odborné praxe na pozici Vývojář mobilních aplikací – trainee. V první části je představena firma a mé pracovní zařazení. V další části jsou popsány úkoly, se kterými jsem se v rámci praxe setkal a také řešení těchto úkolů, na kterých jsem spolupracoval. V závěru práce hodnotím výsledky a celkový přínos absolvované praxe.

Klíčová slova: bakalářská práce, odborná praxe, AutoCont, mobilní aplikace, JavaScript, Ionic, Android, iOS, Cordova, rozšířená realita

Abstract

This bachelor thesis covers my work in the company AutoCont CZ a.s. as a part of my professional practice where I worked as a Trainee mobile application developer. In the first part of this thesis I introduce the company and my work assignment. In the following part there are described the tasks which I have been assigned to and also their solutions on which I have cooperated. In the final part I summarize the results and the overall benefit of the internship.

Key Words: bachelor thesis, professional practice, AutoCont, mobile applications, JavaScript, Ionic, Android, iOS, Cordova, augmented reality

Obsah

Seznam použitých zkratek a symbolů **8**

Seznam obrázků **9**

Seznam tabulek **10**

Seznam výpisů zdrojového kódu **11**

1 Úvod **12**

2 O firmě a pracovním zaměření **13**
2.1 Tým ESA-UX 13
2.2 Pracovní zaměření 13

3 Použité technologie **15**
3.1 Apache Cordova 15
3.2 Ionic 15
3.3 Wikitude SDK 15
3.4 Vuforia SDK 16
3.5 ARToolKit 17

4 Zadané úkoly a jejich řešení **18**
4.1 Aplikace SharePoint čtečka 18
4.2 Vydávání aplikací na Android a iOS 21
4.3 Komponenta pro zobrazování mapových podkladů 24
4.4 Komponenta pro zobrazování formulářů 26
4.5 Rozšířená realita 28

5 Zhodnocení znalostí a dovedností **34**
5.1 Uplatněné znalosti a dovednosti 34
5.2 Chybějící znalosti a dovednosti 34

6 Závěr **35**

Literatura **36**

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
APK	–	Android Package
AR	–	Augmented Reality
CLI	–	Command Line Interface
CMS	–	Content Management System
CSS	–	Cascading Style Sheets
FBX	–	Filmbox
GDAL	–	Geospatial Data Abstraction Library
HTML	–	HyperText Markup Language
JDK	–	Java Development Kit
JDK	–	Java Development Kit
JSON	–	JavaScript Object Notation
JS	–	JavaScript
LESS	–	Dynamický jazyk pro kaskádové styly
OS	–	Operační systém
REST	–	Representational State Transfer
S-JTSK	–	Systém jednotné trigonometrické sítě katastrální
SASS	–	Syntactically Awesome Style Sheets
SDK	–	Software Development Kit
TFS	–	Team Foundation Server
TMS	–	Tile Map Service
UI	–	User Interface
UX	–	User Experience
VRT	–	Virtual Format
WGS	–	World Geodetic System
XML	–	Extensible Markup Language

Seznam obrázků

1 Schéma Wikitude SDK [13] 16
2 Architektura aplikace SharePoint čtečka 18
3 Základní obrazovky aplikace SharePoint čtečka 19
4 Souborová struktura projektu Ionic 22
5 Snímek obrazovky z mapové aplikace 25
6 Snímek obrazovky z formulářové aplikace 27
7 Epson Moverio^{TM} BT-200 [4] 29

Seznam tabulek

1 Časová náročnost zadaných úkolů 18

Seznam výpisů zdrojového kódu

1 Vygenerování soukromého klíče 22
2 Podepsání balíčku APK 23
3 Optimalizace APK nástrojem zipalign 23
4 Ukázka převodu konkrétního rastru 26
5 Ukázka formulářové šablony pro kalendář 28
6 Ukázka jednoduchého zobrazení 3D objektu na vzor 30
7 Ukázka načtení databáze vzorů Vuforia 32
8 Vykreslování objektu na vzor 33

1 Úvod

V rámci své bakalářské praxe ve firmě AutoCont CZ a.s. jsem pracoval na několika úkolech, které se obecně dají rozdělit do dvou kategorií – vývoj hybridních mobilních aplikací ve frameworku Ionic a rozšířená realita.

Hybridní mobilní aplikace jsou v korporátní sféře stále více a více populární, a zdá se, že tento trend bude v roce 2016 dále pokračovat. Hlavní výhodou hybridních aplikací je možnost napsat aplikaci jednou, a nasadit ji na více platforem. To přináší velké úspory oproti nativnímu vývoji, kde bychom danou aplikaci museli napsat pro každou platformu zvlášť. Hybridní aplikace jsou obecně náročnější na systémové prostředky, avšak s dnešními stále výkonnějšími mobilními zařízeními tento problém postupně padá. Dnešní frameworky pro vývoj hybridních aplikací jsou také stále pokročilejší, takže uživatelé často nepoznají, že se nejedná o nativní aplikaci. V rámci bakalářské praxe jsem pracoval na třech úkolech týkajících se hybridních mobilních aplikací, které popíšu v následujících kapitolách.

Rozšířená realita, neboli augmented reality je další moderní technologie, která zejména ve spojení s nositelnou elektronikou (tzv. „wearables“), nachází své aplikace v korporátním prostředí. V této části popíšu práci s několika nástroji určených pro vývoj aplikací pro rozšířenou realitu na mobilní zařízení.

2 O firmě a pracovním zaměření

„AutoCont CZ a.s. je česká soukromá společnost, která přes 25 let na českém a slovenském trhu úspěšně zavádí a provozuje užitečné informační technologie. Zaměřuje se na poskytování komplexních IT řešení a služeb pro firemní klientelu a státní správu.“ [2]

AutoCont CZ je součástí AutoCont Holdingu, do kterého dále patří například slovenská ICT společnost AutoCont SK nebo největší dodavatel CAD řešení Autodesk v České republice firma CAD Studio. AutoCont Holding zaměstnává téměř 1000 lidí, z toho přibližně 700 je zaměstnáno v AutoCont CZ [2].

Pracoviště, které jsem navštěvoval, a zároveň sídlo společnosti se nachází v kancelářském komplexu Orchard – Hornopolní 3322/34, Ostrava. Jelikož moje práce nevyžadovala každodenní přítomnost ve firmě, často jsem využíval možnosti pracovat z domova.

AutoCont se dělí na 4 specializované divize:

- **ITI** – IT infrastruktura
- **PAS** – Podnikové aplikace a služby
- **ITSM** – Outsourcing
- **ESA** – Enterprise Solutions and Applications

2.1 Tým ESA-UX

Tým, ve kterém jsem absolvoval praxi, je součástí divize ESA a vznikl oddělením od týmu ESA-DEV, který vytváří webové aplikace a portály. Současný tým ESA-UX se kromě webových portálů a jejich podpory specializuje na UX[1] a nově také na mobilní aplikace.

Tým má včetně vedoucího 6 členů, součástí týmu jsou: konzultant, programátor C#, UX specialista a 2 vývojáři mobilních aplikací. Tým na projektech spolupracuje ještě s dalšími interními odděleními a některé úkoly předává externistům.

Pro vývoj webových portálů se v týmu používá převážně Kentico CMS, pro správu projektů slouží nástroj JIRA a pro komunikaci pak MS Outlook a Skype pro firmy. Vývoj se řídí agilními metodami, pro správu zdrojových kódů používáme TFS[2].

2.2 Pracovní zaměření

V týmu společně s kolegou pracujeme primárně jako vývojáři mobilních aplikací. Oba se zaměřujeme na vývoj hybridních mobilních aplikací v JS[3] frameworku Ionic, vývoj nativních aplikací pro systém Android a vývoj aplikací pro rozšířenou realitu na mobilní telefony a brýle Epson Moverio™.

[1] **User Experience** – uživatelský prožitek z používání produktu nebo služby
[2] **Team Foundation Server** – nástroj pro správu zdrojových kódů od firmy Microsoft
[3] **JavaScript** – programovací jazyk

Dále jsem se věnoval testováním a vydáváním vytvořených aplikací pro platformu Android do obchodu Google Play a také pro platformu iOS (mimo AppStore).

Kromě vývoje mobilních aplikací jsem se podílel na dvou menších úkolech pro interní potřeby týmu, jednalo se o vyhledání/vývoj nástrojů pro optimalizaci LESS/CSS kódu.

3 Použité technologie

3.1 Apache Cordova

Apache Cordova (dříve PhoneGap) je populární open source framework pro vývoj mobilních aplikací. Apache Cordova umožňuje vývojářům psát multiplatformní aplikace v jazycích HTML5, CSS3 a JavaScript. Umožňuje napsat aplikaci jako webovou stránku a poté ji zabalit a nasadit pro různé platformy. Pro zobrazení grafického rozhraní se použije WebView komponenta, pro přístup k nativním API daných platforem, jako je souborový systém, kamera nebo geolokace, se používají pluginy.

Jelikož pluginy jsou psány nativně a mají pouze zpřístupněné JS API[4], výsledná aplikace není ani čistě webová ani čistě nativní, nýbrž hybridní. Pro nasazování aplikací se používá Cordova CLI[5].

Aktuálně podporované platformy: Android, iOS, Windows Phone, Blackberry OS, Ubuntu Touch, Firefox OS, webOS, FireOS

3.2 Ionic

Ionic je open source SDK[6] pro vývoj hybridních mobilních aplikací, postaveno nad Apache Cordova a AngularJS. Součástí tohoto SDK je např.:

- **Ionic Framework**
- prototypovací nástroj Ionic Creator
- Ionic CLI (podobné Cordova CLI)
- odesílač push notifikací

Ionic má sadu svých UI[7] komponent, s rozdílnými styly pro různé platformy tak, aby se co nejvíce blížily svým nativním předlohám. CSS styly je možno upravovat pomocí SASS[8].

Aktuálně podporované platformy: Android, iOS, pro ostatní platformy pouze neoficiální podpora

3.3 Wikitude SDK

Wikitude SDK je balík nástrojů pro tvorbu nativních i hybridních AR[9] aplikací, který podporuje platformy Android a iOS a také některé chytré brýle, např. Google Glass, Epson Moverio™.

[4] **Application Programming Interface** – rozhraní pro programování aplikací
[5] **Command Line Interface** – rozhraní v příkazovém řádku
[6] **Software Development Kit** – balík nástrojů a knihoven pro tvorbu software
[7] **User Interface** – uživatelské rozhraní
[8] **Syntactically Awesome Style Sheets** – CSS preprocesor
[9] **Augmented Reality** – rozšířená realita – zobrazování virtuálních objektů do reálného světa

Aplikace je možné buď psát přímo nad tímto SDK, nebo pomocí pluginů pro Unity3D, Apache Cordova, Titanium a Xamarin. Hlavní součásti Wikitude SDK:

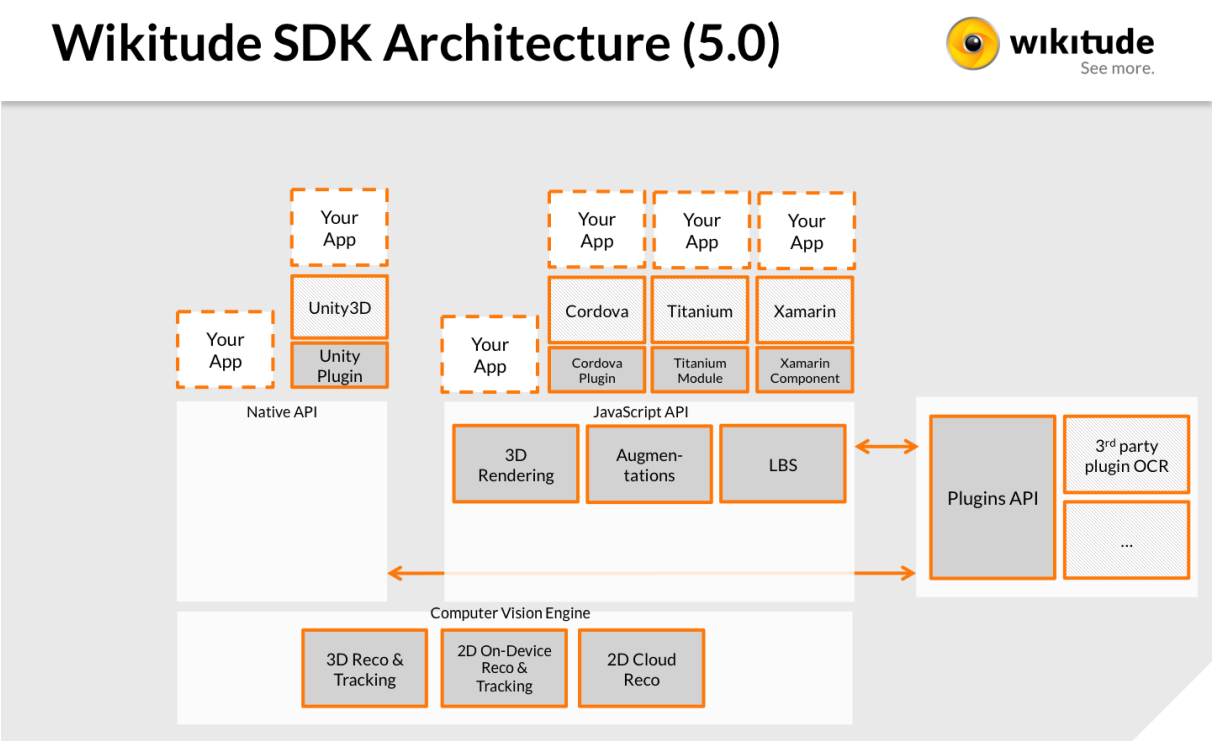
- **Wikitude API**

  JavaScript API – jednoduché, nejvíce funkcí, největší podpora, má svůj jednoduchý renderovací engine

  Native API – nativní API pro dané platformy (Android – Java, iOS – ObjectiveC), nemá svůj renderovací engine

- **Wikitude 3D Encoder** – nástroj pro převod 3D modelů z formátu FBX[10] do vlastního formátu Wikitude

- **Target Manager** – Webový nástroj pro vytváření tzv. „markerů“, které je pak možné v aplikaci detekovat



Obrázek 1: Schéma Wikitude SDK [13]

3.4 Vuforia SDK

Vuforia SDK obsahuje nativní API pro tvorbu AR aplikací, původně vyvíjené společností Qualcomm, dnes jej vlastní společnost PTC. Má podobné funkce jako Wikitude, ale je výkonnější a

[10] **FBX** – Formát souborů pro ukládání 3D modelů od firmy Autodesk

stabilnější. Oproti Wikitude postrádá geolokační funkce a vlastní renderer, má ale levnější licencování.
API je dostupné nativně pro OS Android (Java, C++), iOS (C++) a pro další platformy skrze nástroj Unity.

3.5 ARToolKit

ARToolKit je volně dostupné open source SDK spravované společností DARQI. Oproti komerčním nástrojům zatím postrádá rozpoznávání objektů bez markerů a je také pomalejší. API je dostupné kromě mobilních platforem Android, iOS a Windows Phone také na desktopových systémech Windows a OS X. Součástí je i plugin pro Unity.

4 Zadané úkoly a jejich řešení

V této kapitole je uveden seznam všech zadaných úkolů, časová náročnost a postup řešení těchto úkolů.

Tabulka 1: Časová náročnost zadaných úkolů

Úkol	**Počet hodin**
Aplikace SharePoint čtečka	131
Vydávání aplikací na Android a iOS	62
Komponenta pro zobrazování mapových podkladů	40
Komponenta pro zobrazování formulářů	70
Rozšířená realita	74
Celkem	**377**

4.1 Aplikace SharePoint čtečka

4.1.1 Zadání

Prvním projektem byla aplikace pro synchronizaci dokumentů mezi mobilním zařízením a službou MS SharePoint. Při prvním spuštění aplikace se zobrazí přihlašovací obrazovka, kde uživatel vyplní své uživatelské jméno a heslo. Po přihlášení se zobrazí seznam knihoven služby SharePoint, které je možné synchronizovat. Uživatel vybere knihovnu, kterou si přeje stáhnout a spustí se synchronizace. Po stažení knihovny je možné v aplikaci procházet její obsah jako strukturu souborů a složek. Při kliknutí na soubor aplikace nabídne otevření v nainstalovaných aplikacích na zařízení.

4.1.2 Řešení

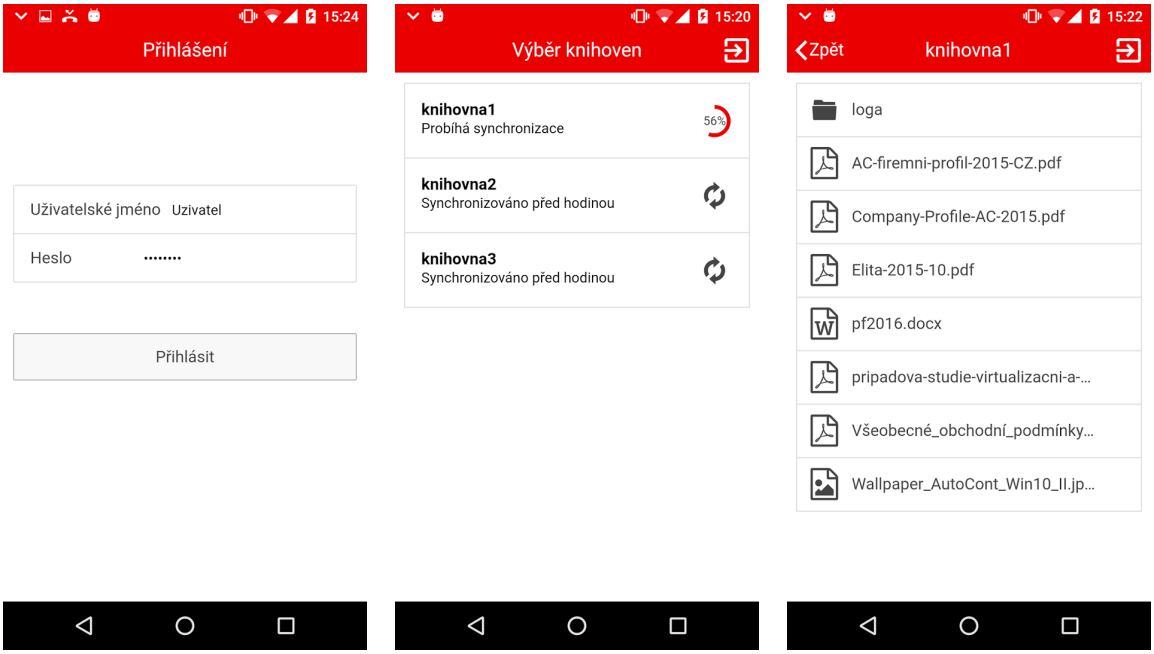
Aplikace je logicky rozdělena do tří hlavních celků:

Obrázek 2: Architektura aplikace SharePoint čtečka

Jednotlivé části nyní postupně popíšu. Jelikož jsem měl na starosti frontend část mobilní aplikace, bude jí věnováno nejvíce prostoru.

4.1.3 Mobilní aplikace

Jelikož už se v týmu pro vývoj multiplatformních aplikací osvědčil Ionic framework, rozhodli jsme se v tomto frameworku vytvořit i aplikaci SharePoint čtečka. Celá aplikace je napsaná v jazyce JavaScript, a její uživatelské rozhraní je vytvořeno v HTML5 a SASS. S kolegou jsme si práci rozdělili dvě části – frontend a backend.



Obrázek 3: Základní obrazovky aplikace SharePoint čtečka

Frontend část se skládá ze tří základních obrazovek – přihlášení, výběr knihoven, procházení souborové struktury dané knihovny. Pro každou obrazovku byla vytvořena zvláštní HTML šablona, ve které se používají převážně základní UI komponenty frameworku Ionic. Každá šablona má také svůj AngularJS controller, kde je napsána logika pro danou obrazovku.

Mějme např. přihlašovací tlačítko, na které je namapována metoda `LoginCtrl.login()`. Tato metoda si z textových polí přečte uživatelské jméno a heslo a tyto údaje pak předá metodě `AuthService.login(user, password)` jako parametry. Metoda se pokusí navázat spojení se serverovou API službou, která v případě úspěšného přihlášení vrátí token, který se uloží na zařízení. Po přihlášení aplikace pokračuje na obrazovce s výběrem knihoven.

Při zobrazení obrazovky s výběrem knihoven daný controller nejprve požádá službu `FileService` o seznam knihoven uložený offline na zařízení. Pokud je vrácený seznam prázdný nebo je dostupné internetové připojení, controller si vyžádá aktuální seznam knihoven ze serveru pomocí služby `NetworkService`. Seznam je poté zobrazen uživateli. Uživatel má možnost vybrat knihovny pro synchronizaci. Při výběru dané knihovny se zkontroluje dostupné místo na disku, pokud je místo dostatečné, zahájí se synchronizace. V případě, že uživatel vybere

další knihovnu, přidá se do synchronizační fronty. O průběhu synchronizace knihovny je uživatel informován ukazatelem stavu, který zprostředkovává knihovna progressbar.js.

Po dokončení synchronizace knihovny je možné prohlížet její obsah jako strukturu souborů a složek. Otevírání souborů je vyřešeno pluginem cordova-file-opener2, který na OS Android zobrazí nativní dialog pro výběr aplikace, která umí daný soubor otevřít (popř. soubor rovnou otevře ve výchozí aplikaci). V systému iOS se soubor pokusí otevřít standardní systémový prohlížeč souborů, kde je následně umožněno sdílení do dalších aplikací.

Pokud zrovna probíhá synchronizace souborů a uživatel spustí jinou aplikaci nebo zhasne displej, bude synchronizace probíhat dále na pozadí.

Backend část se skládá z následujících služeb:

- **FileService** – využívá pluginu cordova-file a zprostředkovává vytváření, zapisování a mazání souborů a složek v paměti zařízení, dále udržuje interní offline databázi o obsahu knihoven
- **NetworkService** – má na starosti síťovou komunikaci se serverovou API službou, stahuje seznam knihoven, obsah knihoven, stahuje soubory po částech, atd. Využívá standardní metody `$http.get` a `$http.post` z AngularJS a Promise API.
- **AuthService** – zajišťuje přihlášení a odhlášení uživatele, používá `localStorage` pro udržení session
- **SyncService** – provádí synchronizaci souborů a zajišťuje, aby se knihovny synchronizovaly ve frontě, pomocí událostí informuje aplikaci o dokončení synchronizace dané knihovny

4.1.4 API služba

Jelikož by pro takto jednoduchou aplikaci bylo zbytečně složité implementovat oficiální SharePoint API, rozhodli jsme si ulehčit práci vytvořením vlastní služby. Služba je spuštěna na externím hostingu a funguje jako prostředník mezi mobilní aplikací a serverem MS SharePoint. SharePoint server je často dostupný pouze z intranetu firmy, naše služba však poskytuje jednoduché REST[11] API pro čtení knihoven SharePoint dostupné z internetu. Pro komunikaci se používá formát JSON[12].

4.1.5 SharePoint server

Jedná se o klasický SharePoint server, kde jsou uložena data pro synchronizaci.

[11] **Representational State Transfer** – způsob manipulace s daty na serveru pomocí jednoduchých HTTP volání

[12] **JavaScript Object Notation** – JavaScriptový objektový zápis, formát pro ukládání dat

4.1.6 Zhodnocení

SharePoint čtečka byla mojí první komplexní aplikací napsanou v Ionic frameworku, tudíž jsem se na ní mnoho naučil. To byl také jeden z důvodů, proč její vytvoření bylo tak časově náročné.

Dalším důvodem je také to, že synchronizace souborů není řešena triviálně – jednotlivé soubory se stahují a zapisují na disk po částech. Museli jsme také ošetřit případy, kdy se např. soubor na vzdáleném úložišti přejmenuje nebo přesune do jiné složky, aby aplikace soubor nestahovala znovu, ale daný soubor jen přejmenovala nebo přesunula. Aplikace si při synchronizaci souborů ukládá i stav synchronizace, aby v ní bylo možné v případě výpadku internetového spojení opět pokračovat.

Dost času také zabralo hledání a testování vhodných Cordova pluginů, kterých aplikace obsahuje více než deset. Mimo jiné např. pluginy pro práci se souborovým systémem, otevírání souborů, přenos souborů, zjištění stavu sítě, manipulaci se stavovým řádkem, běh aplikace na pozadí atd.

Aplikace se navenek může zdát jako velmi jednoduchá, uvnitř je však poměrně komplexní.

4.2 Vydávání aplikací na Android a iOS

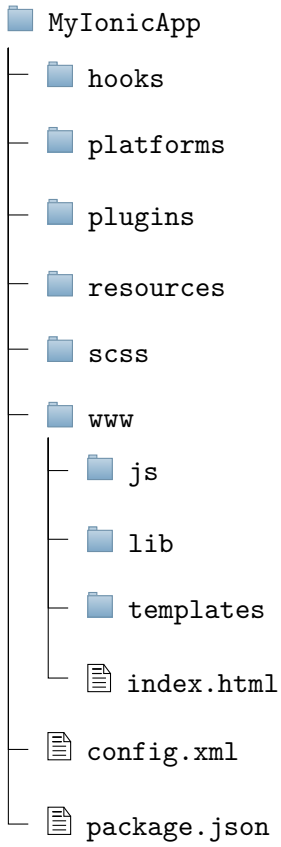
4.2.1 Zadání

V danou chvíli jsme měli tři hotové aplikace (včetně čtečky SharePoint), které bylo potřeba vydat pro systémy Android a iOS. Tento úkol zahrnoval opravy posledních drobných chyb a specifických chyb na daných platformách. V průběhu vývoje byly aplikace testovány převážně na OS Android, takže zvláštní důraz byl kladen na platformu iOS.

4.2.2 Vydávání aplikací v Google Play pro OS Android

Aplikace vyvíjené ve frameworku Ionic spoléhají na systémovou komponentu WebView. Jelikož je Android ekosystém značně fragmentovaný, je v provozu velké množství různých verzí systému s různými verzemi komponenty WebView. Pokud bychom se tedy spoléhali čistě na systémové WebView, naše aplikace by na různých zařízeních mohly vypadat a chovat se jinak. Tento problém řeší projekt Crosswalk, což je běhové prostředí, které můžeme v našich aplikacích použít místo systémového WebView a zajistit tak stejnou kompatibilitu pro všechna zařízení s verzí Android 4.0 a vyšší [3].

Ionic framework přímo podporuje zmiňovaný plugin Crosswalk, instalace se provádí jediným příkazem: `ionic browser add crosswalk`. Před vydáním aplikace je vhodné odstranit pomocné výpisy do konzole, můžeme tedy odstranit výchozí Cordova plugin pro práci s konzolí: `cordova plugin rm cordova-plugin-console`. Před finálním sestavením aplikace je třeba zkontrolovat soubor config.xml (ve složce projektu), ve kterém můžeme upravit název aplikace, číslo verze, oprávnění a další specifická nastavení pro různé platformy.



Obrázek 4: Souborová struktura projektu Ionic

Pokud je vše v pořádku, můžeme aplikaci sestavit příkazem `cordova build --release android`. Výstupem je spustitelný balíček APK[13], který nalezneme ve složce `platforms/android/build/outputs/apk`.

Abychom mohli aplikaci publikovat v obchodu Google Play, musíme ji podepsat svým soukromým klíčem. Tento klíč musí být pro každou novou verzi aplikace stejný, jinak nám Google Play nedovolí nahrát aktualizaci. Pokud svůj soukromý klíč ještě nemáme, je nutné jej vygenerovat nástrojem `keytool`, který je součástí JDK[14].

```
keytool -genkey -v -keystore my-release-key.keystore -alias my_app_key -keyalg
    RSA -keysize 2048 -validity 10000
```

Výpis 1: Vygenerování soukromého klíče

Pokud máme vytvořen soukromý klíč, můžeme aplikaci podepsat nástrojem `jarsigner`, který je opět součástí JDK.

[13] **Android Package** – formát souborů pro distribuci aplikací v OS Android
[14] **Java Development Kit** – sada nástrojů pro vývoj software v jazyce Java

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key
    .keystore MyIonicApp-release-unsigned.apk my_app_key
```

Výpis 2: Podepsání balíčku APK

Instalační balíček máme podepsaný, posledním krokem je jeho optimalizace, která se provádí nástrojem `zipalign`. Tento nástroj je součástí Android SDK a nalezneme ho ve složce `android-sdk/build-tools/VERZE/zipalign`.

```
zipalign -v 4 MyIonicApp-release-unsigned.apk MyIonicApp.apk
```

Výpis 3: Optimalizace APK nástrojem zipalign

Výsledný soubor MyIonicApp.apk nyní můžeme nahrát do obchodu Google Play.

4.2.3 Vydávání aplikací pro iOS

Sestavení a vydání iOS aplikace se provádí v nástroji Xcode, který je dostupný pouze pro operační systém OS X. Pro účely vydávání iOS aplikaci tedy musel být zakoupen počítač typu Mac.

Prvotní postup při sestavení aplikace je podobný jako u OS Android, tedy kontrola souboru config.xml a následné sestavení příkazem `ionic build ios`. U platformy iOS byla ve výchozím stavu zakázána změna orientace, pro povolení bylo nutné do config.xml přidat řádek `<preference name="Orientation"value="all"/>` dovnitř elementu `<platform name="ios">`.

Po sestavení nalezneme ve složce platforms/ios soubor MyIonicApp.xcodeproj, který otevřeme v nástroji Xcode. Z tohoto nástroje můžeme aplikaci spustit v simulátoru nebo na reálném iOS zařízení. Z počátku jsem neměl přímo k dispozici testovací zařízení, takže jsem aplikaci spouštěl a ladil v simulátoru. Zjistili jsme však, že u aplikace spuštěné na fyzickém zařízení vznikaly nové chyby, které se v simulátoru neobjevovaly.

Na fyzickém zařízení se často stávalo, že se aplikace zasekla v průběhu přechodu mezi obrazovkami. To bylo zapříčiněno změnou chování systémové komponenty UIWebView na iOS 9. Tento problém jsme po chvíli hledání opravili záplatou pro AngularJS [6].

Po opravě všech chyb můžeme přistoupit k sestavení výsledného balíčku IPA[15], který můžeme distribuovat sami nebo nahrát do obchodu AppStore. Pro sestavení instalačního balíčku je třeba nejprve vybrat v nástroji Xcode jako cílové zařízení „Generic iOS Device“, poté se odemkne volba v menu Product – Archive.

Jakmile bude archiv sestaven, zobrazí nové okno, ve kterém můžeme podepsat aplikaci pro distribuci. Jelikož chceme aplikaci distribuovat sami, mimo obchod AppStore, není potřeba odesílat aplikaci pro ověření společnosti Apple. Stačí zvolit možnost Export a vybrat jednu ze tří možností pro distribuci mimo AppStore, jelikož jsem aplikaci podepisoval za firmu, zvolil

[15] **IPA** – formát pro distribuci aplikací v systému iOS

jsem možnost „iOS Enterprise Deployment“. Po potvrzení se vygeneruje výsledný IPA balíček pro distribuci. Instalace balíčku na zařízení se provádí v programu iTunes.

4.2.4 Zhodnocení

Nejnáročnější bylo na tomto úkolu samotné hledání a odstraňování platformově specifických chyb. S vydáváním aplikací pro OS Android už jsem měl předchozí zkušenosti, takže v tom nebyl problém.

Časově náročnější bylo seznámení s platformou iOS, pro kterou jsem dříve žádné aplikace nevydával.

4.3 Komponenta pro zobrazování mapových podkladů

4.3.1 Zadání

V několika nabídkách se objevoval požadavek pro zobrazování mapových podkladů a geolokaci, a proto jsme se rozhodli vytvořit jednoduchou komponentu pro zobrazení map. Daná komponenta by měla sloužit zvláště pro navigaci uvnitř budov a průmyslových nebo výstavních areálů ve spojení s navigačními systémy založenými na WiFi, Bluetooth nebo RFID. Proto v základu obsahuje možnost volby patra, resp. přepínání vrstev.

4.3.2 Řešení

Komponenta se skládá z jedné podkladové mapové vrstvy, která se stahuje z internetu, a z několika překryvných vrstev, které jsou uloženy offline na zařízení.

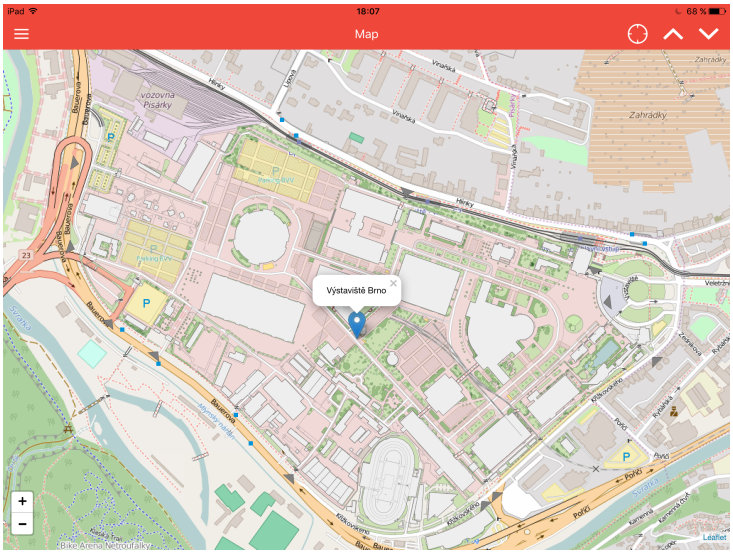
Uživatelské rozhraní se skládá z horního panelu, na kterém jsou tlačítka pro zobrazení aktuální polohy na mapě a výběr patra budovy. Obsahuje také postranní menu s výběrem lokací a hlavně komponentu s mapou, kterou obstarává knihovna Leaflet.

4.3.3 Frontend

Frontend kromě klasických Ionic komponent obsahuje zejména komponentu pro zobrazování map z knihovny Leaflet. Pro tuto knihovnu existuje také direktiva pro AngularJS, takže práce s knihovnou a samotné zobrazení mapy je velice jednoduché. Problém nastává, pokud chceme nějakým způsobem upravovat uživatelské rozhraní, které je součástí Leaflet.

V aplikaci bylo potřeba umožnit výběr patra budovy, což je pouze změna překryvné mapové vrstvy. Knihovna Leaflet automaticky při použití více než jedné vrstvy zobrazuje své uživatelské rozhraní pro výběr vrstev. Tato funkcionalita byla pro mně nežádoucí, jelikož pro jednoduchý výběr patra bylo toto rozhraní příliš složité a nevypadalo podle našich představ.

Hlavním problémem nebylo skrytí výchozího rozhraní, ale samotné přepínání vrstev. Knihovna Leaflet totiž programově neumožňuje přepínat již načtené mapové vrstvy, to musí dělat uživatel



Obrázek 5: Snímek obrazovky z mapové aplikace

přes výchozí rozhraní. Tento problém jsem vyřešil použitím knihovny jQuery, pomocí které jsem programově ovládal přímo UI mapové komponenty Leaflet.

4.3.4 Úprava rastru na dlaždicovou vrstvu

Ačkoliv Leaflet umožňuje do mapy zobrazovat překryvný rastrový i vektorový obraz [7], jedním z požadavků bylo také převést mapový rastr na dlaždicovou strukturu, kterou používají webové mapové služby – pro případ, kdyby měl zákazník k dispozici pouze rastrovou mapu.

Jelikož byla zatím vyvíjena obecná mapová komponenta, nikoliv aplikace pro konkrétního zákazníka, vyžádali jsme si od kolegů z firmy CAD Studio libovolný mapový rastr. Dostalo se nám rastru z Výstaviště města Brna v českém souřadném systému S-JTSK[16] [9].

Pro zpracování rastru do dlaždicové vrstvy jsem použil geografickou knihovnu GDAL[17], konkrétně skript gdal_translate pro převod ze souřadnic S-JTSK do světových souřadnic WGS84[18] [8] a skript gdal2tiles.py, který slouží pro samotné rozřezání rastru na dlaždice.

[16] **Systém jednotné trigonometrické sítě katastrální** – souřadnicová síť používaná v geodézii na území České republiky a Slovenska

[17] **Geospatial Data Abstraction Library** – knihovna pro manipulaci s rastrovými a vektorovými geografickými daty

[18] **World Geodetic System** – světový souřadnicový systém používaný v GPS

Skriptu gdal_translate bylo potřeba předat následující parametry (kromě vstupního a výstupního souboru):

- **-of VRT** – výstupní formát VRT[19]
- **-a_srs „def_str“** – definice souřadného systému (transformace vůči WGS84)
- **-a_ullr lx ly rx ry** – souřadnice levého horního a pravého dolního rohu rastru v daném souřadném systému (S-JTSK)

Výstupní soubor se poté předal jako parametr skriptu gdal2tiles.py, který rozřezal původní rastr na dlaždice a rozmístil je do adresářové struktury TMS[20].

```
gdal_translate -of VRT -a_srs '+proj=krovak +lat_0=49.5 +lon_0
    =24.83333333333333 +alpha=0 +k=0.9999 +x_0=0 +y_0=0 +ellps=bessel +towgs84
    =570.8,85.7,462.8,4.998,1.587,5.261,3.56 +units=m +no_defs' -a_ullr
    -601368.03 -1160665.55 -599474.25 -1162046.81 bvv_areal_1tis.png out.vrt
gdal2tiles.py -z 10-18 out.vrt
```

Výpis 4: Ukázka převodu konkrétního rastru

4.3.5 Zhodnocení

Jelikož jsem měl předchozí zkušenosti z aplikace SharePoint čtečka a měl jsem předpřipravenou šablonu od svého vedoucího, bylo vytvoření této aplikace poměrně rychlé a jednoduché. Nejzajímavější částí pro mě bylo převedení mapového rastru na dlaždice, kde jsem se musel seznámit s používanými souřadnicovými systémy a s knihovnou GDAL.

4.4 Komponenta pro zobrazování formulářů

4.4.1 Zadání

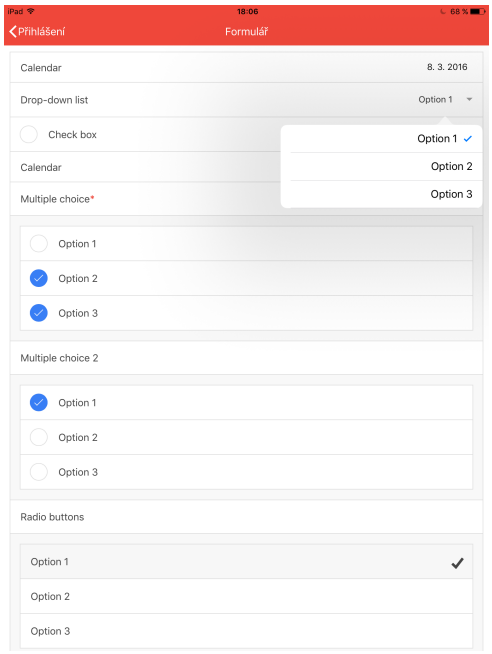
Byl vznesen požadavek pro dynamické generování mobilních formulářů, např. pro využití v aplikacích pro ankety a dotazníky. Správce obsahu webového portálu si v CMS[21] bude moci nadefinovat klasický webový formulář, který se pak zobrazí v aplikaci s přizpůsobeným ovládáním pro mobilní zařízení.

4.4.2 Řešení

Samotná komponenta si nejprve pomocí API stáhne definici formuláře, formulář vygeneruje a zobrazí uživateli. Při vyplňování formuláře uživatelem provádí komponenta základní validaci dat. Komponenta zašle vyplněná data zpět na webový portál.

[19] **Virtual Format** – XML formát pro popis transformace obrázku do cílových souřadnic
[20] **Tile Map Service** – formát používaný zejména webovými mapovými službami, adresářová struktura dlaždic
[21] **Content Management System** – systém pro správu obsahu, např. webového portálu

Komponenta měla být původně součástí aplikace pro vyplňování dotazníků pro interní potřeby firmy. Pro tuto funkcionalitu se nakonec použila webová služba, takže aplikace už nebyla potřeba. Komponentu jsme ale přesto vytvořili a zobecnili tak, aby se dala použít pro zobrazení jakéhokoliv formuláře. Stejně jako ostatní aplikace je i tato napsána ve frameworku Ionic.



Obrázek 6: Snímek obrazovky z formulářové aplikace

Komponenta umožňuje zobrazení formuláře, který si správce systému nadefinuje v administračním prostředí Kentico CMS. Samotná definice formuláře je uložena ve formátu XML[22]. V mobilní aplikaci je pak možné tento formulář zobrazit, vyplnit a odeslat na server.

Pro stažení a odeslání vyplněného formuláře slouží REST API, které poskytuje Kentico CMS [5]. Aplikace s tímto API komunikuje ve formátu JSON.

Komponenta se v zásadě skládá pouze z jednoho AngularJS controlleru a jedné HTML šablony. Metoda buildForm projde staženou XML definici formuláře, vyfiltruje pro aplikaci zbytečná data a sestaví JSON objekt, který obsahuje pouze seznam jednotlivých UI komponent a jejich parametry.

Výsledný objekt už zpracovává HTML šablona, kde je pomocí AngularJS direktiv pro manipulaci DOM[23] popsáno, jak a které UI komponenty zobrazit. AngularJS také obsahuje funkce pro validaci formulářů [1], které provádí validaci dat vždy, když uživatel provede nějakou změnu. Podle toho aplikace umožní nebo neumožní odeslání formuláře na server.

[22] **Extensible Markup Language** – formát pro ukládání dat
[23] **Document Object Model** – způsob reprezentace HTML a XML dokumentů

```
<div ng-repeat="field in survey.form.field track by $index"
   ng-if="field._visible === 'true'">
   <label class='item item-input'
      ng-if="field.settings.controlname === 'CalendarControl'">
      <span class='input-label'>
         {{ field.properties.fieldcaption }}
         <b ng-if="field._allowempty !== 'true'">*</b>
      </span>
       <input type="date" name="{{ field.settings.controlname }}"
         ng-model="field.data"
         ng-required="field._allowempty !== 'true'">
   </label>
   ...
</div>
```

Výpis 5: Ukázka formulářové šablony pro kalendář

4.4.3 Zhodnocení

Z předchozích zkušeností z aplikace pro mapové podklady a SharePoint bylo zobrazování formulářů bezproblémové.

4.5 Rozšířená realita

4.5.1 Zadání

Úkolem bylo zmapovat současnou situaci na trhu komerčních i volně dostupných SDK pro tvorbu aplikací využívající rozšířenou realitu, a seznámit se s možnostmi využití AR u nositelných zařízení a chytrých telefonů.

V době mého nástupu měla firma k dispozici brýle pro rozšířenou realitu Epson Moverio^{TM} BT-200, které jsem si měl možnost vyzkoušet. Mým úkolem z počátku bylo prozkoumat možnosti Wikitude SDK pro rozšířenou realitu na daných brýlích a na chytrém telefonu. Jelikož jsme zjistili, že Wikitude nebude vyhovovat pro všechny případy užití, byl jsem později pověřen hledáním dalších SDK.

4.5.2 Brýle Epson Moverio^{TM} BT-200

Součástí brýlí je náhlavní sada se dvěma průhlednými LCD displeji s rozlišením 960x540 pixelů, která umožňuje zobrazovat kromě 2D i 3D stereoskopický obraz. Náhlavní sada má kameru s rozlišením VGA, mikrofon a senzory – gyroskop, akcelerometr a kompas.

Druhou částí je dotykový ovladač, který je s náhlavní sadou spojen kabelem, a je na něm nainstalován systém Android 4.0.4. Ovladač disponuje dvoujádrovým APU TI OMAP 4460 1.2 GHz, 1 GB RAM, 8 GB interní paměti a GPS. Má navíc stejné senzory jako náhlavní sada. Ovladač je možné připojit k dalším zařízením pomocí WiFi, Bluetooth 3.0 a USB. [4]

Aplikace je do brýlí možno instalovat z online obchodů Google Play, Moverio Apps Market a přes USB.

Obrázek 7: Epson Moverio^{TM} BT-200 [4]

4.5.3 Wikitude SDK

V této sekci popíšu proces vytvoření jednoduché aplikace pro rozšířenou realitu s Wikitude SDK, která bude zobrazovat 3D objekt na daný vzor.

Zobrazení 3D objektu je s Wikitude JavaScript API poměrně jednoduché. Aplikace jako taková se skládá z části napsané v jazyce Java, která obsahuje aktivity a další komponenty typické pro Android aplikace. Druhá část, specifická pro Wikitude, je tzv. **„ARchitectView“**, což je komponenta pro rozšířenou realitu. V ARchitectView je zobrazen obraz z kamery překrytý 2D nebo 3D objekty. Komponenta se dále skládá z HTML šablony a JavaScript kódu, kde je napsána logika aplikace.

Než budeme zobrazovat 3D objekt, je potřeba mít jej na co zobrazovat. Musíme tedy vytvořit tzv. „target image“, neboli vzor, na kterém se budou zobrazovat prvky rozšířené reality. Vzorem může být téměř libovolný obrázek, který není symetrický. Pokud máme vhodný obrázek, je potřeba jej nahrát do online nástroje Wikitude Target Manager, který z něj vytvoří vzor pro naši aplikaci ve formátu WTC. Nástroj pro každý obrázek vypočítá skóre, které udává jak moc je obrázek vhodný pro rozpoznávání. Jeden WTC soubor může obsahovat až 1000 vzorů. [12]

```
var World = {
   loaded: false,
   createOverlays: function createOverlaysFn() {
      this.tracker = new AR.ClientTracker("assets/ac_tracker.wtc", {
         onLoaded: this.loadingStep
      });
      this.modelCar = new AR.Model("assets/car.wt3", {
         onLoaded: this.loadingStep,
         scale: {
            x: 0.2,
            y: 0.2,
            z: 0.2
         },
         rotate: {
            roll: -90
         }
      });
      var trackable = new AR.Trackable2DObject(this.tracker, "nazev", {
         drawables: {
            cam: [this.modelCar]
         }
      });
   }
};
World.createOverlays();
```

Výpis 6: Ukázka jednoduchého zobrazení 3D objektu na vzor

3D model, který chceme zobrazovat ve vykreslovacím enginu Wikitude, musíme také převést do speciálního formátu WT3. Pro převod slouží nástroj Wikitude 3D Encoder, dostupný pro operační systémy Windows a OS X. Nástroj v současné době umí převádět modely a animace pouze z formátu FBX. [11]

Na výpisu č. 6 je vidět část kódu, která zajišťuje zobrazení převedeného 3D modelu na náš vzor. Nejprve se inicializuje `AR.ClientTracker`, který zajišťuje rozpoznávání vzoru. Načtení 3D modelu zajišťuje třída `AR.Model`, kde v konstruktoru můžeme specifikovat i transformace objektu.

Posledním krokem je vytvoření objektu typu `AR.Trackable2DObject`, kterému předáme referenci na `AR.ClientTracker`, zadáme název vzoru a 3D modely nebo obrázky, které jej mají překreslovat.

Při testování Wikitude jsem přišel na poměrně zásadní limitace. První z nich je **nemožnost detekce více vzorů zároveň**, např. pokud je v záběru kamery více než jeden vzor nebo více stejných vzorů, mohu překreslit pouze jeden z nich. Dalším limitem je **absence podpory čárových kódů**, což může být problém zejména u aplikací určených pro korporátní sféru.

4.5.4 Vuforia SDK

Kvůli zmíněným limitacím Wikitude bylo nutné prozkoumat i další SDK pro rozšířenou realitu. Jedním z nich je Vuforia. Na dalších řádcích popíšu tvorbu podobné aplikace jako v předchozí sekci.

Vuforia poskytuje pro OS Android nativní API v jazyce Java, takže tvorba aplikace se nijak zásadně neliší od tvorby ostatních aplikací pro Android. Není tu však žádná komponenta podobná ARchitectView jako ve Wikitude, která by usnadnila práci. Musíme pracovat přímo s komponentou GLSurfaceView z Android API.

Pro správu vzorů opět existuje webová služba – Vuforia Target Manager [10]. Je k dispozici několik typů vzorů:

- **Single Image (obrázek)** – obyčejný obrázek ve formátu PNG nebo JPG
- **Cuboid (kvádr)** – skládá se ze šesti obrázků, které reprezentují jednotlivé stěny kvádru
- **Cylinder (válec)** – celkem tři obrázky – horní a dolní podstava, plášť
- **3D objekt** – jedná se o reálný 3D objekt naskenovaný v aplikaci Vuforia Object Scanner
- **Frame Markers**[24] – sada 512 speciálních čtvercových vzorů, kde je rozpoznáván pouze rámeček, uvnitř může být vlastní logo, optimalizováno pro rozpoznávání mnoha vzorů současně

Ve správci vzorů je nejprve nutné vytvořit databázi vzorů, do této databáze pak můžeme nahrávat samotné vzory. Každý vzor musí mít svůj unikátní název a rozměry v poměru vůči ostatním vzorům v databázi. Po nahrání do databáze jsou obrázky zpracovány a převedeny do formátu Vuforia. Opět se vypočte skóre, které udává jak spolehlivé bude rozpoznávání daného vzoru. Databázi vzorů je možno stáhnout buďto pro použití v SDK (2 soubory DAT a XML) nebo Unity (UnityPackage).

[24] **Frame Markers** se nenahrávají do speciální databáze, jsou součástí Vuforia SDK. V aplikaci se pracuje pouze s jejich ID, tedy číslem 0 až 511.

```
public boolean loadTrackerData(){
    TrackerManager tManager = TrackerManager.getInstance();
    ObjectTracker objectTracker = (ObjectTracker) tManager.getTracker(
        ObjectTracker.getClassType());
    if (mDataSet == null)
        mDataSet = objectTracker.createDataSet();
    if (!mDataSet.load("TargetDB.xml", STORAGE_TYPE.STORAGE_APPRESOURCE))
        return false;
    if (!objectTracker.activateDataSet(mDataSet))
        return false;
    int numTrackables = mDataSet.getNumTrackables();
    for (int count = 0; count < numTrackables; count++) {
        Trackable trackable = mDataSet.getTrackable(count);
        if(isExtendedTrackingActive())
            trackable.startExtendedTracking();
    }
    return true;
}
```

Výpis 7: Ukázka načtení databáze vzorů Vuforia

Na výpisu č. 7 je zjednodušená ukázka načtení databáze vzorů. Metoda načte z disku soubor TargetDB.xml, který vygeneroval webový správce vzorů. V souboru může být uloženo více vzorů, uvedená metoda je všechny načte a inicializuje rozpoznávání.

Jelikož Vuforia nemá svůj vykreslovací engine, musíme vykreslovat sami pomocí OpenGL[25] nebo použít externí engine. Bez ohledu na metodu vykreslování musíme nejprve určit kam přesně objekt vykreslit, tak aby překrýval rozpoznaný vzor. Postup je na výpisu č. 8, kde je část vykreslovací metody. Na začátku vykreslování každého snímku si z objektu typu Renderer (pomocná třída z Vuforia API) uchováme stav všech sledovaných vzorů. Poté cyklem procházíme vzory, které byly na snímku rozpoznány. Cílovou pozici objektu, který chceme vykreslovat udává matice ModelViewProjection, kterou můžeme odeslat shaderu na grafické kartě.

[25] **Brýle Epson Moverio™ BT-200**, které jsme měli k dispozici podporují OpenGL ES 2.0. Dnešní moderní zařízení už však podporují technologie OpenGL ES 3.0 nebo Vulkan.

```
State state = mRenderer.begin();
for (tId = 0; tId < state.getNumTrackableResults(); tId){
   TrackableResult result = state.getTrackableResult(tId);
   Trackable trackable = result.getTrackable();
   Matrix44F modelView_Vuforia = Tool.convertPose2GLMatrix(result.getPose());
   float[] modelViewMatrix = modelView_Vuforia.getData();
   float[] modelViewProjection = new float[16];
   Matrix.multiplyMM(modelViewProjection, 0, vuforiaAppSession.
       getProjectionMatrix().getData(), 0, modelViewMatrix, 0);
   // Následuje samotné vykreslování
}
```

Výpis 8: Vykreslování objektu na vzor

4.5.5 Zhodnocení

Práce s rozšířenou realitou byla sice náročnější, ale o to zajímavější. Dle mého názoru, má tato technologie opravdu velkou budoucnost, nicméně zatím není používání brýlí pro rozšířenou realitu poněkud uživatelsky přívětivé.

Při programování mi velice pomohlo absolvování předmětu Základy počítačové grafiky.

5 Zhodnocení znalostí a dovedností

5.1 Uplatněné znalosti a dovednosti

Jelikož byla mou hlavní pracovní náplní tvorba hybridních mobilních aplikací, uplatnil jsem zejména znalosti z předmětu **Tvorba aplikací pro mobilní zařízení I**, kde jsem se naučil základy JavaScriptu. Pomohly mi také znalosti o operačním systému Google Android, které jsem nabyl v předmětu **Tvorba aplikací pro mobilní zařízení II**. Při tvorbě aplikací pro rozšířenou realitu jsem velmi ocenil základní znalost OpenGL získanou v předmětu **Základy počítačové grafiky** a také znalost programovacího jazyku Java z předmětu **Programovací jazyky II**.

Dostatečným obecným základem pro vývoj software pro mě byly předměty **Algoritmy II** a **Programování II**, kde jsem si osvojil principy objektově orientovaného programování. Dále jsem využil znalosti z předmětu **Vývoj informačních systémů** týkající se zejména návrhových vzorů.

V průběhu praxe jsem občas pro usnadnění práce využil programovací jazyk Python, který jsem se naučil v předmětu **Skriptovací programovací jazyky a jejich aplikace**.

5.2 Chybějící znalosti a dovednosti

Určitě bych ocenil, kdyby existoval předmět více zaměřený na JavaScript a na frameworky typu AngularJS.

Chyběly mi také znalosti týkající se analýzy obrazu, které by mi pomohly obejít limitace některých SDK pro rozšířenou realitu. Předměty, které se analýzou obrazu zabývají jsou v nabídce až v navazujícím studiu.

6 Závěr

Absolvování odborné praxe ve firmě AutoCont CZ a.s. pro mne bylo bezesporu značným přínosem, ať už v rovině osvojených znalostí či nutné spolupráce ve větším týmu.

Naučil jsem se dvě, pro mne velice podstatné věci, a sice: jakým způsobem lze pracovat ve velké IT firmě a rovněž jak co nejefektivněji komunikovat s lidmi v týmu.

Při práci na zadaných úkolech jsem se přesvědčil, že je někdy výhodnější vyvinout hybridní mobilní aplikaci, namísto aplikace nativní. Rovněž jsem se přesvědčil, že i hybridní aplikace může působit velmi dobrým dojmem a být téměř k nerozeznání od aplikace nativní.

Ze všeho nejvíce si velmi cením příležitosti pracovat na aplikacích pro rozšířenou realitu. Ačkoliv jsou nositelná zařízení využívající rozšířenou realitu v současnosti stále poněkud nepraktická, troufnu si říct, že mají před sebou světlou budoucnost, a to nejen v zábavním průmyslu...

Jiří Vaculík

Literatura

1. *AngularJS: Developer Guide: Forms* [online]. [cit. 2016-01-22]. Dostupný z WWW: ⟨https://docs.angularjs.org/guide/forms⟩.
2. *AutoCont CZ a.s. – Profil společnosti* [online]. [cit. 2016-01-04]. Dostupný z WWW: ⟨http://www.autocont.cz/o-spolecnosti/profil-spolecnosti⟩.
3. *Crosswalk – build world class hybrid apps* [online]. [cit. 2016-04-22]. Dostupný z WWW: ⟨https://crosswalk-project.org⟩.
4. *Epson Moverio^{TM} BT-200 Smart Glasses (Developer Version Only) – Product Information* [online]. [cit. 2016-04-10]. Dostupný z WWW: ⟨http://www.epson.com/cgi-bin/Store/jsp/Product.do?sku=V11H560020⟩.
5. *Getting data using REST - Kentico 8.2 Documentation - Kentico Documentation* [online]. [cit. 2016-01-22]. Dostupný z WWW: ⟨https://docs.kentico.com/display/K82/Getting+data+using+REST⟩.
6. *iOS 9 Potential Breaking Change* [online]. 2015 [cit. 2015-12-07]. Dostupný z WWW: ⟨http://blog.ionic.io/ios-9-potential-breaking-change/⟩.
7. *Leaflet 0.7 API Reference* [online]. [cit. 2016-02-02]. Dostupný z WWW: ⟨http://leafletjs.com/reference.html⟩.
8. PŘIDAL, Petr. *[FreeGeoCZ] Konverze z S-JTSK pomocí gdalwarp* [online]. 2009 [cit. 2016-02-07]. Dostupný z WWW: ⟨http://mailman.fsv.cvut.cz/pipermail/freegeocz/2009-April/000414.html⟩.
9. *S-JTSK* [online]. [cit. 2016-02-03]. Dostupný z WWW: ⟨http://freegis.fsv.cvut.cz/gwiki/S-JTSK⟩.
10. *Vuforia Target Manager* [online]. [cit. 2016-01-30]. Dostupný z WWW: ⟨https://developer.vuforia.com/library/articles/Training/Getting-Started-with-the-Vuforia-Target-Manager⟩.
11. *Wikitude 3D Encoder* [online]. [cit. 2016-04-18]. Dostupný z WWW: ⟨http://www.wikitude.com/external/doc/documentation/latest/htmlcss/encoder.html⟩.
12. *Wikitude SDK Android 4.0 Documentation* [online]. [cit. 2016-04-18]. Dostupný z WWW: ⟨http://www.wikitude.com/external/doc/documentation/4.0/android/targetmanagement.html⟩.
13. *Wikitude SDK Documentation for Android* [online]. [cit. 2016-01-03]. Dostupný z WWW: ⟨http://www.wikitude.com/developer/documentation/android⟩.