Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Mobilní aplikace pro podporu dne otevřených dveří na FIM

Bakalářská práce

Autor: Dušan Salay
Studijní obor: Aplikovaná informatika

Vedoucí práce: doc. Ing. Filip Malý, Ph.D.

Hradec Králové srpen 2016

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

……

V Hradci Králové dne 28.8.2016 Dušan Salay

Poděkování:

Děkuji vedoucímu bakalářské práce doc. Ing. Filipu Malému, Ph.D. za cenné rady a připomínky při zpracování této práce.

Anotace

V současné době jsou čím dál tím více populárnější mobilní aplikace pro chytré telefony. Ať už někam jedete nebo jen hledáte nějakou instituci, většinou je vám nabídnuta mobilní aplikace ke stažení. I na Fakultě informatiky a managementu na Univerzitě Hradec Králové je pár takových mobilních aplikací dostupných. Nicméně je zde stále volné místo pro aplikaci na podporu dne otevřených dveří fakulty, která by uchazečům tento den zpříjemnila. Jedna taková aplikace a její vývoj je popsán v následující práci. Aplikace kromě standartních knihoven ještě využívá Google Maps API a YouTube Android API. Součástí práce je rovněž stručné popsání Android OS, pro který byla aplikace vyvíjena.

Annotation

Title: Mobile application for support open days on FIM

Mobile applications for smartphones are nowadays increasingly popular. Even when you are looking for an institution or just going somewhere, there is usually some application available for download. Few similar applications is also on Faculty of Informatics and Management on University of Hradec Kralove. However there is still enough of free space for application, to support Faculty‘s Open day and to make this day more comfortable for visitors. Such an application and its development is described in this Bachelor Thesis. This application uses, besides standard libraries, Google Maps API and Youtube Android API. Part of this work also contains a brief description of Android OS for which was following application developed.

Obsah

Seznam obrázků

Seznam grafů

Seznam výpisů kódu

Terminologický slovník

API.....Application Program Interface - rozhraní k softwarové aplikaci
GPS.....globální polohovací systém určující polohu na Zemi
GUI .....Graphic User Interface – grafické uživatelské rozhraní aplikace
ID .....jednoznačný identifikátor (označení)
JSON.....formát pro výměnu dat vycházející ze syntaxe JavaScriptu
layout.....grafické rozložení stránky (plochy)
LIFO ..... zásobníková paměť, kde se čte poslední zapsaná položka jako první
Nativní kód .....posloupnost strojových instrukcí prováděných procesorem
Open Handset Alliance..... skupina 84 mobilních a technologických společností
open-source ..... licence umožňující šířit celý projekt ke komerčním účelům
OS ..... operační systém
SQL.....Structured Query Language - programovací jazyk pro práci s databází
task-killer ..... aplikace, která hrubou silou ukončuje aktivity
UI ..... uživatelské rozhraní
UML .....Unified Modeling Language - modelovací jazyk
Use Case Model ..... model služeb aplikace pro uživatele
XML.....eXtensible Markup Language – rozšiřitelný značkovací jazyk

1 Úvod

Mobilní aplikace je software vytvořený pro chytré telefony, tablety, chytré hodinky a další mobilní zařízení [21]. Tyto programy zaznamenávají v uplynulých 15 letech čím dál tím větší rozmach. Uživatelé mohou stahovat tyto aplikace do svých zařízení díky dostupných online obchodů. Mezi hlavní distribuční platformy mobilních aplikací patří App Store, Google Play, Windows Phone Store a BlackBerry App World. Aplikace bývají velmi často zdarma, což ještě více umožnuje jejich rozšíření. Vývojáři mobilních aplikací můžou navíc využívat veškerých dostupných prostředků, které umožní vytvořit více praktickou a uživatelsky přívětivou aplikaci. Dnešní mobilní zařízení jsou velmi výkonná, často kompaktní a disponují nepřeberným množstvím funkcí. Smartphony jsou vybaveny nejrůznějšími čidly, senzory, moduly a kvalitními dotykovými displeji, mnohdy i s velmi vysokým rozlišením. Některé modely se navíc vyznačují zvýšenou odolností a dlouhou výdrží na jedno nabytí akumulátoru.

Dnes již existuje nepřeberné množství aplikací všeho druhu. Vzhledem k tomuto stavu je důležité před vývojem vlastní aplikace zvážit k čemu bude aplikace sloužit a zdali už neexistuje řešení pro danou oblast.

Nyní je na Fakultě informatiky a managementu na Univerzitě Hradec Králové pár mobilních aplikací dostupných. Nicméně žádná aplikace se nevěnuje dni otevřených dveří fakulty, která by uchazečům tento den zpříjemnila.

1.1 Cíl práce

Cílem této bakalářské práce je vytvořit mobilní aplikaci pro podporu dne otevřených dveří FIM. Aplikace by pomohla dovést potencionální účastníky dne otevřených dveří z autobusového a vlakového nádraží do školy. Dále zobrazovat události konaných přednášek, základní informace o fakultě a přehled nabízených oborů. Aplikace využívá nový Material UI desing v barvách fakulty. Dále si práce klade za cíl popsat použité technologie, zdůvodnit výběr platformy Android a ostatních zvolených nástrojů.

1.2 Metodika zpracování

Při zpracování popisu systému Android bylo z důvodu aktuálnosti čerpáno převážně z webových zdrojů, dále z odborných publikací a článků. Aby bylo možné získat objektivní pohled na celou problematiku. Bylo také čerpáno z vlastních zkušeností v dané oblasti. Na základě získaných znalostí byla realizována vlastní aplikace a tím byly znalosti ověřeny v praxi.

Programování aplikace probíhalo v programu Android Studio. Programátorské postupy byly čerpány hlavně z oficiální dokumentace a z průvodce programování pro platformu Android. Vše poskytuje Android Developers na svých webových stránkách [22]. Zpracování grafického návrhu uživatelského prostředí probíhalo pomocí náčrtů na papír.

Textový obsah aplikace byl čerpán z oficiálních webových stránek FIM UHK.

2 Android

Android je open-source platforma na bázi Linuxu určená primárně pro mobilní zařízení, tedy smartphony, tablety, navigace, fotoaparáty a chytré hodinky. Dnes se Android objevuje i v televizních přijímačích a dalších domácích spotřebičích jako jsou pračky nebo lednice. Na trhu je 78% mobilních zařízení s Androidem. [2]

Jde o typ operačního systému, který podporuje více platforem a zařízení různých značek s rozdílnou konfigurací hardwaru. To však přináší jednu velkou nevýhodu v optimalizaci systému na konkrétní platformu, což je naopak silná stránka konkurenčního Apple iOS. [1]

Pro Android je k dispozici největší množství aplikací. Spousta z nich však podprůměrné kvality, jelikož proces jejich schvalování není zdaleka tak přísný jako u iOS nebo Windows Phone. [1]

2.1 Historie

Společnost Android Inc. vznikla v roce 2003. O dva roky později, v srpnu 2005, Google odkoupil Android Inc. V listopadu roku 02007 vzniklo pod názvem Open Handset Alliance sdružení společností, které se zabývají vývojem hardwaru a softwaru pro mobilní zařízení, a jehož hlavním smyslem je spolupráce na vytváření standardů. Toto sdružení stojí za vývojem Androidu dodnes.

V roce 2008 přišel na trh první chytrý telefon s Androidem 1.0.

Jednotlivé verze operačního systému Android mají číselné i kódové označení podle názvu sladkostí.

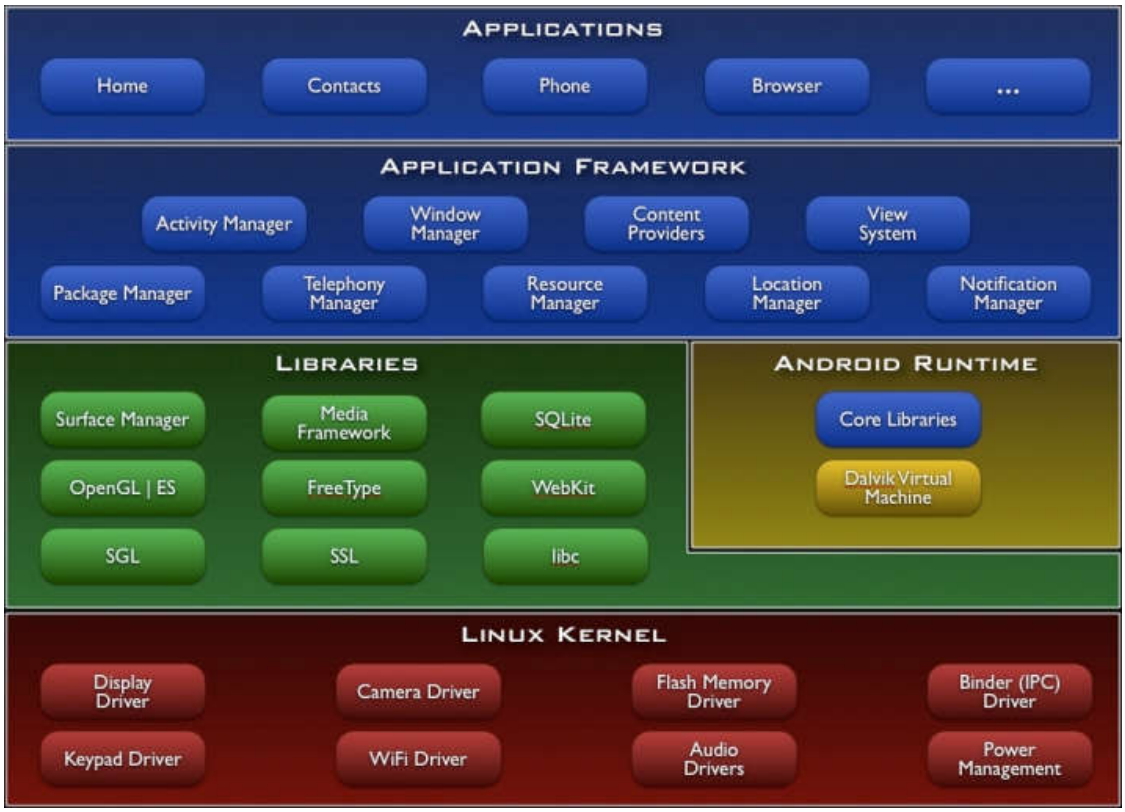
Obr 1: Historie názvů OS Android [20]

2.2 Architektura Androidu

K vývoji aplikací je zapotřebí mít alespoň základní informace o architektuře a o principech, podle kterých spolu jednotlivé vrstvy komunikují.

Operační systém Android je tvořen pěti vrstvami:

- Linux kernel
- Libraries
- Android Runtime
- Application Framework
- Applications



Obr 2: Architektura Androidu [23]

2.2.1 Linux kernel

Srdce systému, nejnižší vrstva architektury Androidu, je upravené jádro populárního operačního systému Linux. Úpravy se týkají redukce funkcí a jejich přizpůsobení možnostem mobilního zařízení. Slouží k přímé komunikaci s hardwarem mobilního zařízení. Například s kamerou, GPS, Wi-Fi, Bluetooth, různými senzory a podobně. Dále se stará o řízení jednotlivých procesů, vláken,

o management paměti, poskytuje služby starající se o zabezpečení systému, správu napájení, vstupně-výstupní operace, služby spojené se souborovým systémem a další. Komunikaci mezi jednotlivými procesy umožnuje ovladač zvaný Blinder.

2.2.2 Libraries

Nad jádrem se nachází vrstva s knihovnami, které komunikují s linuxovým jádrem a ovladači jednotlivých hardwarových modulů. Nativní knihovny jsou napsané v programovacích jazycích C nebo C++. Zde se nachází například Surface Manager pro podporu funkcionality multidotykového displeje, dále knihovna OpenGL pro práci s grafikou, SQLite pro správu databází v Androidu a mnoho dalších užitečných knihoven. Systémová knihovna LibC převzatá z Linuxu je upravena pro mobilní zařízení, takže obsahuje jen části, které jsou potřeba v Androidu.

2.2.3 Android Runtime

Android Runtime obsahuje sadu základních knihoven, které se vyskytují i v klasickém API pro Javu SE. Označují se také jako Core Libraries.

Každá aplikace pro Android je samostatný proces, který vytváří při spuštění aplikace svoji instanci virtuálního stroje Dalvik Virtual Machine (DVM). DVM je obdobou Java Virtual Machine (JVM) na klasických počítačích, ale je optimalizován pro použití na mobilních zařízeních, takže bere v úvahu menší paměť, omezené možnosti napájení a podobně. Dalvik Virtual Machine vykonává programový kód dané aplikace. Při této činnosti pracuje s takzvanými Dalvik Executable soubory, které mají příponu DEX. Soubory DEX vznikají kompilací z klasických souborů CLASS. Při šíření aplikace jsou všechny zdroje náležící aplikaci zabaleny do jediného souboru s příponou APK. [3]

Od Androidu 5.0 se používá dopředná kompilace. To znamená, že aplikace se zkompiluje do nativního kódu zařízení už při samostatné instalaci. Výsledkem je výrazné zrychlení aplikací a úspora energie na úkor jen mírnému zpomalení instalace aplikace. Pro zpětnou kompatibilitu je původní kompilátor Dalvik zachován. [3]

2.2.4 Application framework

Framework obsahuje další knihovny. Je napsán v Javě a je to nejdůležitější vrstva pro vývojáře. Poskytuje aplikacím základní služby systému. Tyto služby mohou zpřístupňovat data (např. kontakty) v jiných aplikacích a prvky GUI (tlačítka, ikony atp.). Dále umožňuje přístup ke stavovému řádku, práci s notifikacemi a správu aplikací běžících na pozadí. Knihovny také zpřístupňují hardware používaného zařízení a mnoho dalších služeb a funkcí.

Obsahuje například: [1]
Package Manager – modul pro správu balíčků, který udržuje seznam všech aplikací nainstalovaných v zařízení
View System – spravuje prvky grafického uživatelského rozhraní
Activity Manager – spravuje životní cyklus aplikace
Notification Manager – umožňuje všem aplikacím zobrazit vlastní upozornění ve stavovém řádku

2.2.5 Applications (Aplikace)

Nejvyšší úroveň architektury operačního systému Android jsou nativní aplikace. Například seznam kontaktů, posílaní SMS, kalendář, kalkulačka, navigace a další.

2.3 Základní součásti aplikace

Aplikace pro Android jsou stavěné na čtyřech základních pilířích realizovaných jako třídy. [1]

2.3.1 Activity (Aktivity)

Aktivita je hlavní třída, která se zobrazí po spuštění aplikace. [1] Aktivity umožnují přes grafické uživatelské rozhraní (GUI) ovládat aplikaci (vytočit číslo, poslat email, vyplnit formulář, vybrat si položku ze seznamu atp.) nebo sdělovat výstup uživateli.

Aplikace se obvykle skládá z více aktivit, které si můžou mezi sebou vzájemně odevzdávat údaje. Když na dané aktivitě provedeme činnost, která vyvolá aktivitu novou, tak se předešlá aktivita zastaví a uloží do zásobníku Back Stack (LIFO). [4]

Aktivitu lze spustit pouze parametrem typu *Bundle* v metodě *onCreate*, protože nemá veřejný konstruktor. [4]

Aktivita by měla být navržena tak, aby umožnila uživateli soustředit se na jednu věc, kterou potřebuje momentálně realizovat, například zadat kontaktní údaje, napsat a odeslat textovou zprávu. [1]

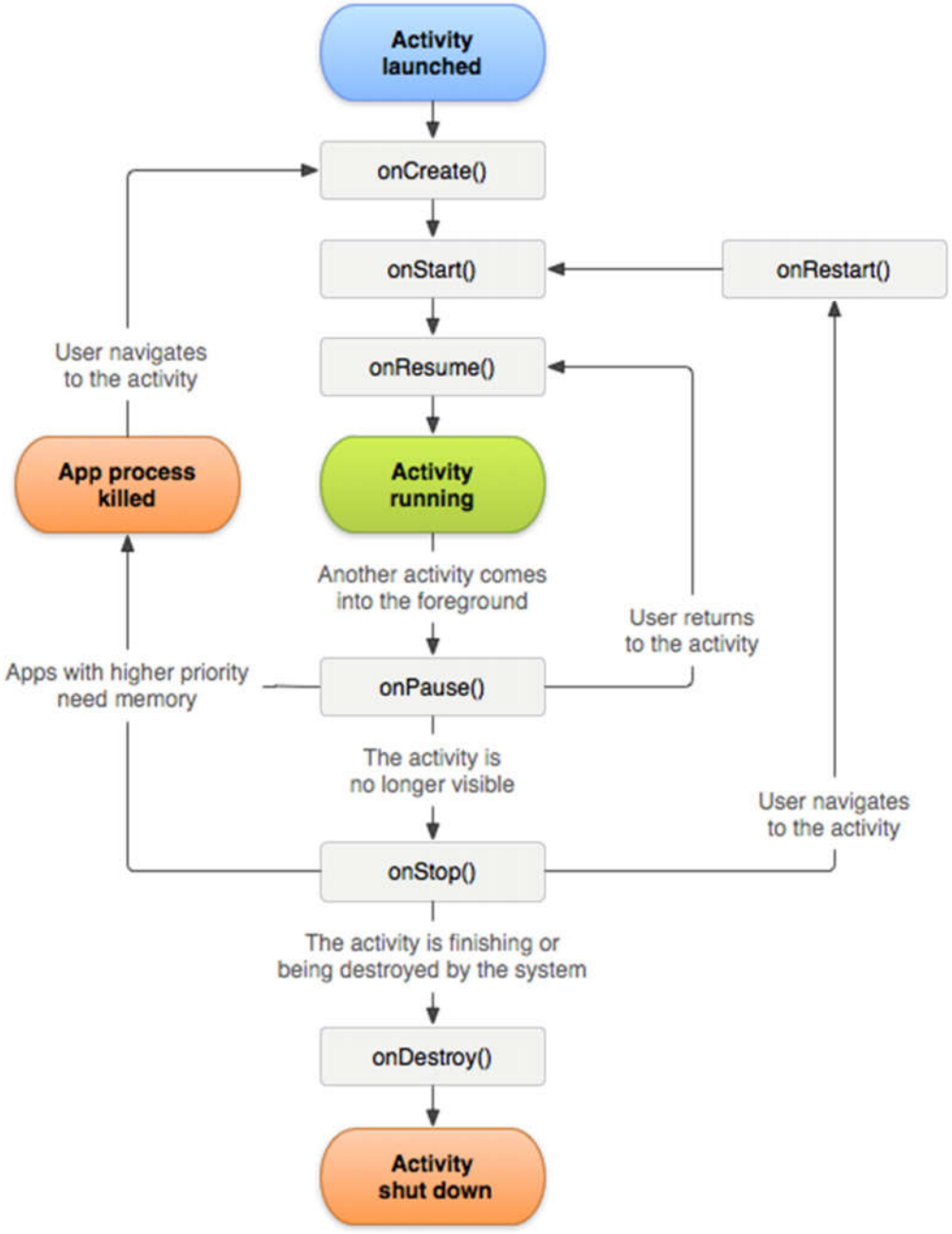
Další tři části nemají GUI.

2.3.1.1 Životní cyklus aktivity

Aplikace pro Android se skládají z více na sobě nezávislých komponent (aktivit a služeb). Operační systém sám rozhoduje, kdy budou instance aktivit vytvořeny, kdy budou odsunuty na pozadí či zničeny v důsledku uvolnění paměti pro jiné aktivity [1]. Z toho plyne, že takzvané *task-killery*, které napevno ničí aktivity, aby uvolnily paměť RAM, jsou zbytečné. Naopak při obnově více mrtvých procesů naráz, dochází k většímu zatížení výkonu zařízení. [5]

Životní cyklus definuje, jaké metody se mají spouštět v přesně daných situacích v určeném pořadí. Má čtyři hlavní fáze (čerpáno z [4]):

- Je-li aktivita v popředí, a má takzvaný fokus, to znamená, že uživatel může aktivitu ovládat.
- Pokud aktivita ztratila fokus, ale je stále vidět. Například překrytí dialogovým oknem. Pokud je extrémní nedostatek paměti může systém aplikaci v tomto stavu odstranit. Je pozastavena.
- Pokud je aktivita zcela zakryta jinou aktivitou. Aktivita je však stále v paměti v oblasti Back Stacku, takže se k ní můžeme bez problému vrátit. Nicméně v případě nedostatku paměti systém tyto aktivity odstraní. Je zastavena.
- Pokud je aktivita pozastavena nebo zastavena a systém jí odstraní, tak při opětovném zobrazení uživateli je aktivita zcela restartována a obnovena do původního stavu.



Obr 3: Schéma životního cyklu aktivity. [4]

Celý životní cyklus je definován následujícími metodami aktivity.

OnCreate() – Aktivuje se po spuštění aktivity. Vytváří GUI a konfiguruje proměnné a objekty. Obvykle řeší zavedení layoutu a prvků *view*. Stále ještě neviditelná a nekomunikuje s uživatelem. Vždy následuje OnStart().

OnStart() – Aktivita přechází do popředí a je viditelná pro uživatele. Volá se při spuštění po předchozím zastavení aktivity. Obvykle následuje metoda OnResume(), která se volá, když aktivita přechází z pozadí do popředí.

OnPause() – Volá se, když se má spustit nová aktivita a ta aktuální přechází do pozadí. Vhodné pro uložení změny údajů, se kterými aktivita pracovala. Metoda se také volá při zobrazení dialogového okna nebo stisknutí tlačítka Home. Po doběhnutí metody může být proces zlikvidován. Operace v této metodě by měly být časově nenáročné.

2.3.2 Services (Služby)

Služby realizují dlouhotrvající operace v pozadí. Také umožnují spolupráci se vzdálenými procesy a neposkytují uživatelské rozhraní. Můžete spustit službu, která bude běžet na pozadí, i když se uživatel přepne do jiné aplikace, například přehrávání hudby [6]. Služby umožňují asynchronně, paralelně s hlavním vláknem, provádět operace, jejichž realizace trvá déle. [1]

2.3.3 Broadcast Receivers (Přijímače)

Jsou objekty na vysílání a přijímání na pozadí, které reagují na události uskutečněné na zařízení. Jako příklad se může uvést přijetí SMS zprávy. Když zpráva přijde do mobilu, Android tuto skutečnost uživateli oznámí. Operační systém neví, kdy SMS přijde, a proto právě neustále naslouchá na pozadí a čeká na přijetí zprávy. [8]

2.3.4 Content providers (Poskytovatelé obsahu)

Poskytovatelé obsahu umožnují ukládání a sdílení dat mezi více aplikacemi a procesy. Aplikace tedy mohou přistupovat k údajům ostatních aplikací, které vystupují také jako poskytovatelé obsahu [1]. Android obsahuje poskytovatele obsahu pro běžné typy dat, například audio, video, obrázky, kontaktní informace a

podobně. Pokud potřebujeme zveřejnit vlastní data a sdílet je, lze vytvořit vlastní poskytovatele obsahu. [8]

2.4 Bezpečnost

Android jako moderní mobilní a plně otevřená platforma musí mít robustní bezpečnost a přísnou bezpečnostní politiku. Proto byl operační systém navržen s vícevrstvou bezpečností. [9]

Hlavním bezpečnostním mechanismem je takzvaný Sandbox, ve kterém běží aplikace izolovaně. Každá aplikace dostane přidělenou část souborového systému, kde může zapisovat soukromá data. Souborový systém je na rozdíl od verze 3.0 zašifrován. [1]

2.4.1 Oprávnění

Mezi další funkce zabezpečení patří mechanismus pro přidělování oprávnění. Tím můžeme aplikaci zabránit přístupu k některým komponentám a údajům. Hlavním důvodem je zabránění aplikaci v dělání úkonů bez vědomí uživatele [9]. Například vytáčení telefonních čísel, zasílání drahých prémiových SMS, pořizování fotografií, zapnutí mikrofonu či odeslání GPS souřadnic, je velký zásah do soukromí.

Přidělená oprávnění se definují v manifestu aplikace. Manifest je nutná součást aplikace pro Android, kde vývojář definuje prostředky, které bude aplikace používat.

Uživatel schvaluje přiřazená oprávnění k aplikaci právě při její instalaci. Pokud je vydána aktualizace, která přidává dodatečná oprávnění, uživatel s nimi musí opět souhlasit. V praxi uživatel bohužel ani kolikrát dlouhý seznam oprávnění nečte nebo po delší době zapomene. Naštěstí v nové verzi Android 6.0 Marshmallow je systém oprávnění zcela přepracovaný. V novém systému bude uživatel jednotlivá oprávnění schvalovat, až když budou opravdu potřeba [10]. Tím se systém přiblíží konkurenčním platformám iOS a Windows Phone.

2.5 Ukládání dat

Ať už jednoduché nebo komplikované mobilní aplikace často potřebují uložit nějaký typ dat. A na to potřebují patřičný úložný prostor. Může se jednat o pouhé uložení

nastavení aplikace nebo záznamy velkého objemu dat. Proto Android OS nabízí několik různých způsobů ukládání dat.

2.5.1 Shared Preferences

Shared Preferences je rozhraní, které umožnuje aplikaci rychle a efektivně ukládat data do perzistentní mapy pod definovaným klíčem. Data jsou uložena v XML souboru, který se nachází v zařízení. Tato metoda se nejvíce používá k ukládání nastavení aplikace, přihlašovacích údajů k umožnění automatického přihlášení nebo dosaženého skóre ve hře. K těmto datům může přistupovat jakákoliv komponenta aplikace.

2.5.2 SQLite Databáze

SQLite je malá a rychlá relační databáze umožňující ukládat složitější strukturovaná data. Oproti jiným databázím využívá pouze klientskou část. Databáze podporuje veškeré známé SQL dotazy. Aplikace si uchovává vlastní instanci k připojení databáze. Databáze je uložená v zařízení ve složce */data/data/<nazev balicku>/databases*. Pomocí content providerů je možné data z databází sdílet mezi aplikacemi. SQLite podporuje jen základní datové typy pro definování tabulek a umožňuje práci s více tabulkami, indexací, triggery a pohledy.

2.5.3 Ukládání do souboru

Pro načítání a ukládání do souboru využívá Android OS známou knihovnu Javy *java.io.File*. Je možné volit mezi dvěma úložnými prostory, interním a externím. Do interního úložiště by měla aplikace ukládat malé a středně velké objemy dat, privátní nebo dočasné údaje. Externí úložiště je nejčastěji namapováno na vyměnitelnou paměťovou kartu. Na toto úložiště je nejvhodnější ukládat velké objemy dat (obrázky, audio soubory, dokumenty, textury k 3D aplikacím). Každá aplikace si vytvoří na kartě vlastní složku s cestou */sdcard/Android/data/<nazev balicku >/files/*.

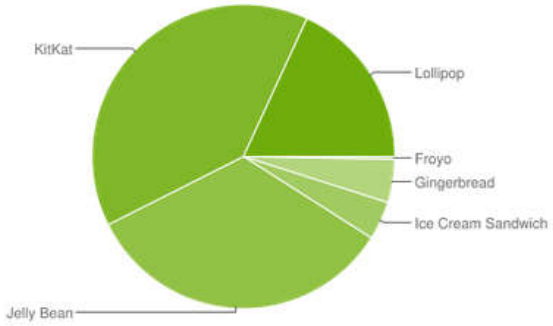
2.5.4 Dočasné soubory (cache)

Občas aplikace potřebuje uložit soubory, které nebude při příštím spuštění potřebovat. Takovéto soubory se ukládají do speciální složky */cache/*. Systém sám rozhodne podle dostupných prostředků, jestli zvolí interní nebo externí úložiště. Ve chvíli, kdy bude mít systém nedostatek úložného prostoru, může dočasné soubory v této složce odstranit.

2.6 Výběr úrovně API (2015)

Před začátkem programování mobilní aplikace je důležité, vybrat cílové verze systému Android, pro které bude aplikace vyvíjena. Příliš staré API omezuje programátora o nové moderní funkce. Naopak příliš nové, zatím nerozšířené API, nemůže využít mnoho potencionálních zákazníků.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	4.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	4.1%
4.1.x	Jelly Bean	16	13.0%
4.2.x		17	15.9%
4.3		18	4.7%
4.4	KitKat	19	39.3%
5.0	Lollipop	21	15.5%
5.1		22	2.6%



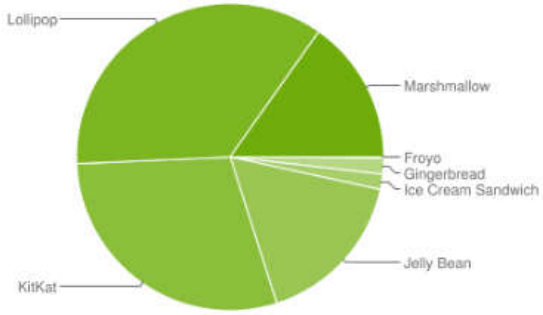
Obr 4: Podíl zařízení s jednotlivými verzemi Androidu. V Srpnu 2015. [11]

Za rok klesl podíl Androidu Gingerbread z 16.2% na 4.6% [11], proto ho už nemá moc smysl podporovat. Ani verze Ice Cream Sandwich se 4.1% nevypadá moc atraktivně, ale u této verze lze již bez omezení používat nové rysy uživatelského rozhraní. Nové aplikace stačí vyvíjet pro verze Ice Cream Sandwich (API 15) nebo Jelly Bean (API 16).

2.6.1 Výběr úrovně API o rok později (2016)

Za pouhý rok vývoje této aplikace vydal Google dvě nová API (API 23 Marshmallow a API 24 Nougat), což dokazuje jak moc je Andriod OS se rychle vyvíjející systém. Příchody nových API pozměnily i podíl zařízení s jednotlivými verzemi Androidu na trhu.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.6%
4.1.x	Jelly Bean	16	6.0%
4.2.x		17	8.3%
4.3		18	2.4%
4.4	KitKat	19	29.2%
5.0	Lollipop	21	14.1%
5.1		22	21.4%
6.0	Marshmallow	23	15.2%



Obr 5: Podíl zařízení s jednotlivými verzemi Androidu. V Srpnu 2016. [11]

Podle obrázku výše můžete vidět, že verze Gingerbread a Ice Cream Sandwich se přiblížili k 1.5% a nemá je vůbec smysl nadále podporovat. Nejstarší podporované API by mělo být API 16 Jelly Bean. Dalo by se uvažovat i o minimálním API 19 (KitKat), kterým pokryjeme 80% zařízení s Androidem.

2.6.2 Support Library

Google garantuje takzvanou dopřednou kompatibilitu. To znamená, že aplikace vytvořené pro starší verzi systému, budou vždy fungovat na nejnovějších verzích. Google dokonce poskytuje pro starší aplikace speciální režim kompatibility, který přizpůsobuje uživatelské prostředí aplikace nové platformě. Tento režim může vývojář zakázat.

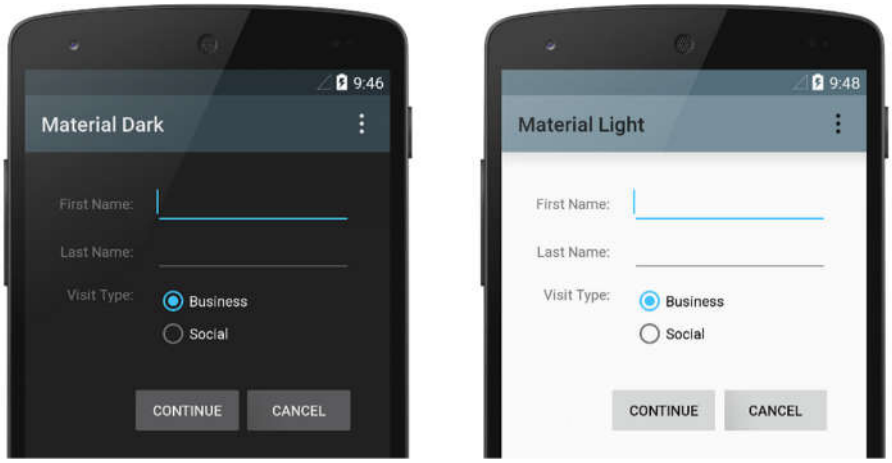
Opakem je takzvaná částečná zpětná kompatibilita. Aby bylo možné využívat nové moderní funkce i na starších zařízeních, musí se do projektu přidat knihovny Android Support Library. Každá knihovna je zpětně kompatibilní na určitou úroveň

API [12]. Například aplikace využívající v13 support library je možné spouštět od verze API 13 (Android 3.2 Honeycomb).
Vývojář se musí rozhodnout, zda příslušnou moderní funkcionalitu implementuje do aplikace přímo, nebo přes knihovnu Android Support Library. [1]

Příklady knihoven (čerpáno z [12]):
Annotations Support Library – podpora přidávání anotací
Design Support Library – podpora přidávání prvků Material Desingu aplikace
Percent Support Library – podporuje přidávání a správu procentuální rozměrů, využívá se v layoutech
App Recommendation Support Library for TV – podpora přidávání obsahu pro aplikace běžící na televizních zařízeních (Android TV).

2.7 Material Desing

Material Desing je koncept nového vzhledu, který přišel s verzí Android 5.0 Lollipop. Je přehlednější a pomocí různých efektů, hlavně stínu, se snaží vyvolat dojem 3D zobrazení. Pro správné dodržování pravidel Material Desingu, společnost Google sepsala příručku. Hodně změn prodělal notifikační panel, který zobrazuje oznámení podle významu a důležitosti. Notifikace je možné zobrazit i na uzamykací obrazovce. Material Theme poskytuje nový styl desingu pro aplikace, systémové prvky a widgety, nové výchozí animace pro zpětnou vazbu a nové přechody mezi aktivitami.[13]



Obr 6: Material Theme. [13]

Přibyly také dvě nové komponenty pro zobrazování karet a seznamů. Jsou to třídy *CardView* a *RecyclerView*, který je obdobou *ListView*, ale podporuje různé typy rozvržení a poskytuje zlepšení výkonu. [13]

Další velkou změnou je udávání souřadnic u *View* komponent. Kromě X a Y souřadnic mají nyní i Z souřadnici. Tato nová vlastnost přibyla kvůli určení velikosti stínu a pořadí zobrazení komponent. Vyšší hodnota znamená zobrazení v popředí.

Nové API pro animace umožňuje vytváření vlastních animací. [13]

2.7.1 Ikony

S novým vzhledem vydal Google i novou sadu ikon. Každá ikona je tvořena podle daných pravidel. Vyznačují se hlavně minimalistickou a jednoduchou formou. Je zakázáno používat trojrozměrný obraz a stínování. Ikony se dělí na systémové a produktové. Systémové ikony nebo ikony uživatelského rozhraní představují typy souborů, adresářů, nástrojů a zařízení. Systémové ikony jsou také používány k reprezentování běžných akcí, jako jsou například tisk nebo ukládání. Naopak produktové ikony, jak už název napovídá, vizuálně představují značku produktů služeb. [15]

3 Vývoj aplikace

V následující kapitole je popsán postup, jakým probíhal vývoj aplikace, vyhodnocení dotazníkového šetření a analýza aplikace, dále použité vývojové prostředí, grafické rozložení v jednotlivých aktivitách a detailnější rozbor použitých API. U návrhu aktivit jsou doplněny snímky obrazovek pro lepší vizuální představu. Pro ukázku jsou zařazeny i zajímavé části kódu z aplikace, odlehčené o některé řádky, aby byl kód snadněji pochopitelný.

3.1 Analýza dotazníkovým šetřením

Dotazníkové šetření vzniklo za účelem zjistit mínění uchazečů, kteří mají zájem studovat na FIM, o mobilní aplikaci, která by jim pomohla při dnu otevřených dveří FIMky. Byla použita metoda osobního dotazování. Výzkum probíhal 15. 1. 2016 přímo na dni otevřených v budově J, kde byli respondenti náhodně vybíráni. Hlavním cílem dotazníku bylo prozkoumat, zda by takovou aplikaci uchazeči využili a jejich názor na připravované funkce, jež bude aplikace obsahovat. Dále byli uchazeči dotazováni na případné nesnáze, se kterými se během dne otevřených dveří potýkali, a u kterých by jim aplikace mohla pomoci problémy vyřešit. Dotazník se skládal z celkem 9 otázek, jak uzavřených (pohlaví, věk, druh operačního systému), tak i otevřených (nápady na další funkce v aplikaci).

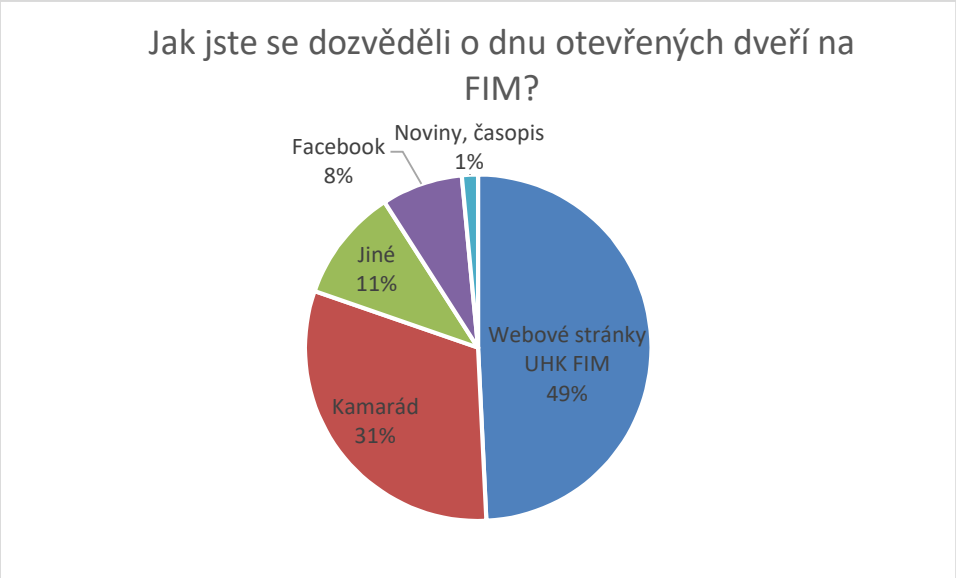
3.1.1 Vyhodnocení dotazníku

Celkový počet respondentů, kteří odpověděli na všechny otázky, byl 115.

První dotaz se týkal pohlaví respondentů, kde výsledky vypovídaly o 54% zastoupení žen a 46% mužů, což vypovídá o téměř rovnocenném rozdělení pohlaví respondentů.

Druhý dotaz prověřoval, zda uchazeč pochází přímo z Hradce Králové nebo odjinud. Zde se podle očekávání jasně ukázalo, že 90 % kandidátů na den otevřených dveří dojíždí a 10 % je místních. Ale i z této malé části místních uchazečů, jich 91% o aplikaci projevilo zájem.

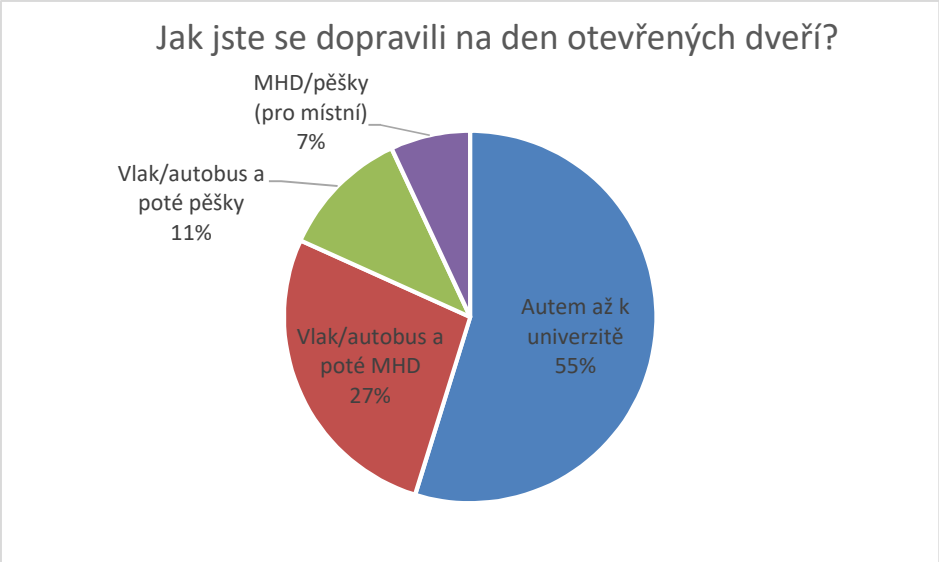
Další dotaz zkoumal způsob, jakým se uchazeči o dnu otevřených dveří dozvěděli. Výsledky měly hlavně zjistit, na jakých místech aplikaci co nejvíce propagovat.



Graf 1: Dotazníkové šetření, Jak jste se dozvěděli o dnu otevřených dveří na FIM [vlastní zpracování]

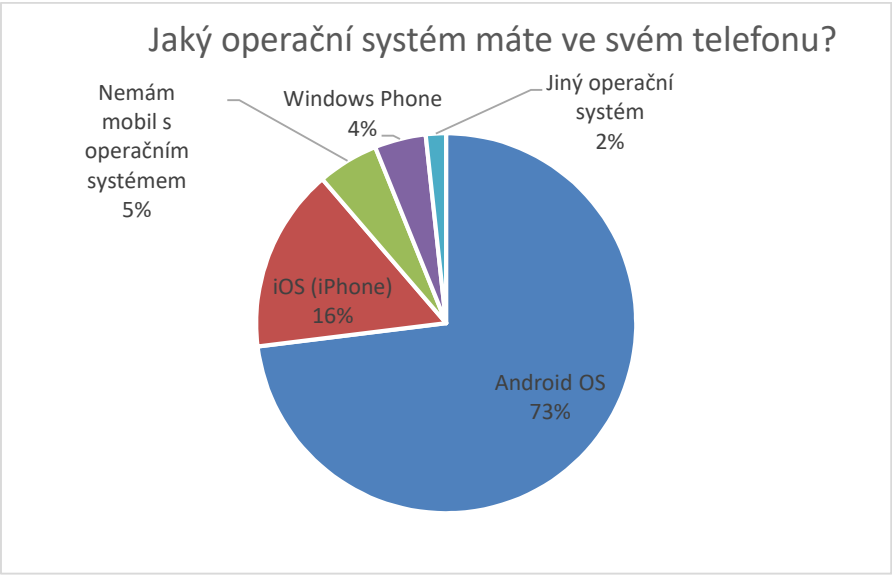
Z odpovědí vyplývá, že polovina uchazečů vyhledávala informace převážně na školním webu univerzity, tudíž by zde mělo být umístěno nejvíce reklam na propagaci mobilní aplikace. Přesto by se určitě neměla zavrhnout možnost šíření po sociálních sítích – 8 %. Respondenti, kteří označili možnost „jiné“, převážně zmínili veletrh vysokých škol Gaudeamus. Proto by kupříkladu u stánku FIM na tomto veletrhu mohl být transparent s QR kódem ke stažení aplikace. V ideálním případě si zprávu o existenci aplikace budou předávat potencionální studenti, kteří mezi sebou sdílejí a vyměňují si informace – 31 %.

Následující dotaz analyzoval, jakým způsobem se respondenti dopravili na den otevřených dveří. Aplikace by měla pomoci navigovat potencionální studenty přímo do školy, takže určitě musí obsahovat GPS souřadnice budovy FIM a integrovat Google Maps API. Jelikož 55 % uchazečů zvolilo odpověď „autem až k univerzitě“ a 11 % se prošlo od hlavního vlakového nebo autobusového nádraží až k budově FIM pěšky. Zbylých 27 % dojíždějících použilo k přepravě hradecké MHD. Z toho plyne, že by v aplikaci měl určitě být implementován „návod“, jakými autobusy MHD je možné se dopravit k univerzitě.



Graf 2: Dotazníkové šetření, Jak jste se dopravili na den otevřených dveří? [vlastní zpracování]

Navazovala důležitá otázka „Jaký operační systém máte ve svém telefonu?“. Je podstatné zjistit, pro jakou platformu by měla být aplikace vyvíjena jako první. Podle odpovědí dotazovaných jasně dominuje platforma Android OS se 73 %. Na druhé pozici se umístil s velkým odstupem iOS, jenž vlastnilo 18 respondentů. Windows Phone nebo jiný operační systém má dohromady zanedbatelných 6 %. Zbylých 5 % respondentů nevlastní mobil s operačním systémem, tudíž se jich jakákoliv aplikace netýká.

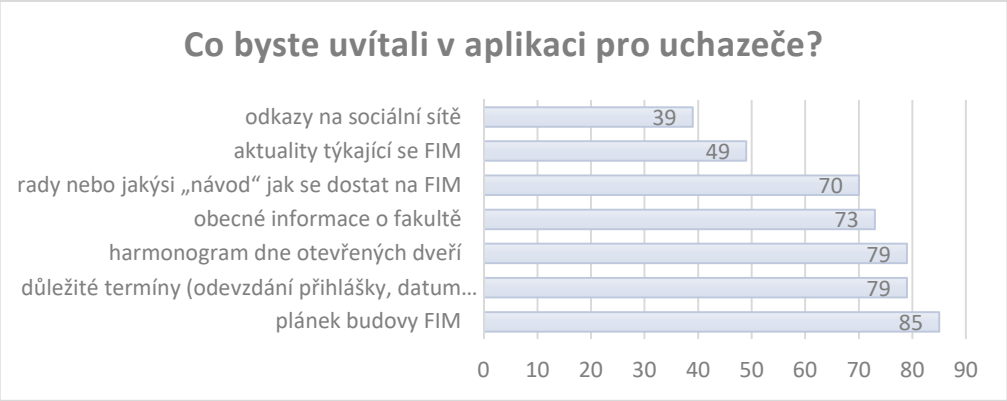


Graf 3: Dotazníkové šetření, Jakou používáte platformu? [vlastní zpracování]

Další dotaz zjišťoval, zda uživatele napadlo vyhledávat v obchodu s aplikacemi aplikaci týkající se UHK. Naprostá většina řekla, že nikoliv – 97 %. Což znamená, že uchazeči nejsou zvyklí vyhledávat takovýto typ aplikací a pro samotnou aplikaci na podporu dne otevřených dveří se bude muset poskytnout co největší podpora při její propagaci. Nato respondentům byla položena další otázka, zda by o takovou aplikaci měli vůbec zájem. Tři čtvrtiny všech dotázaných odpovědělo, že by jim aplikace pomohla, takže má smysl takovou aplikaci vyvíjet a poté se snažit dostat do podvědomí uchazečů, že FIMka takovou aplikací disponuje.
V předposledním dotazu byly respondentům vyjmenovány a popsány následující funkce, které by se v aplikaci mohly vyskytnout:

- *obecné informace o fakultě* – informace vhodné pro uchazeče, popis jednotlivých oborů a možnost uplatnění, popis, co vše lze při studiu získat (druhy certifikátů, které lze při studiu získat), odkaz na první kroky na FIM
- *harmonogram dne otevřených dveří* – rozpis konajících se přednášek
- *rady nebo jakýsi „návod", jak se dostat na FIM* – popis cesty z vlakového/autobusového nádraží, mapy, GPS souřadnice budovy
- *plánek budovy FIM* – půdorys budovy s popisem jednotlivých přednáškových, seminárních a počítačových učeben
- *aktuality týkající se FIM* – informace o aktuálním dění a novinkách na FIM
- *odkazy na sociální sítě* – propojení se sociálními stránkami FIM
- *důležité termíny* – termín odevzdání přihlášky, datum zápisu, příští den otevřených dveří

Respondenti mohli vybrat všechny funkce, které se jim líbili a rádi by je viděli v aplikaci. Největší zájem byl o plánek budovy FIM, dále jen o 6 hlasů méně měly možnosti „harmonogram dne otevřených dveří" a „důležité termíny pro uchazeče". Obecné informace o fakultě a rady, jak se dostat na FIM, také volila většina dotázaných, takže by je měla aplikace určitě obsahovat. Nejmenší zájem byl naopak o aktuality a odkazy na sociální sítě s nejčastějším odůvodněním, že takové informace získají pohodlněji přímo na webových stránkách fakulty.



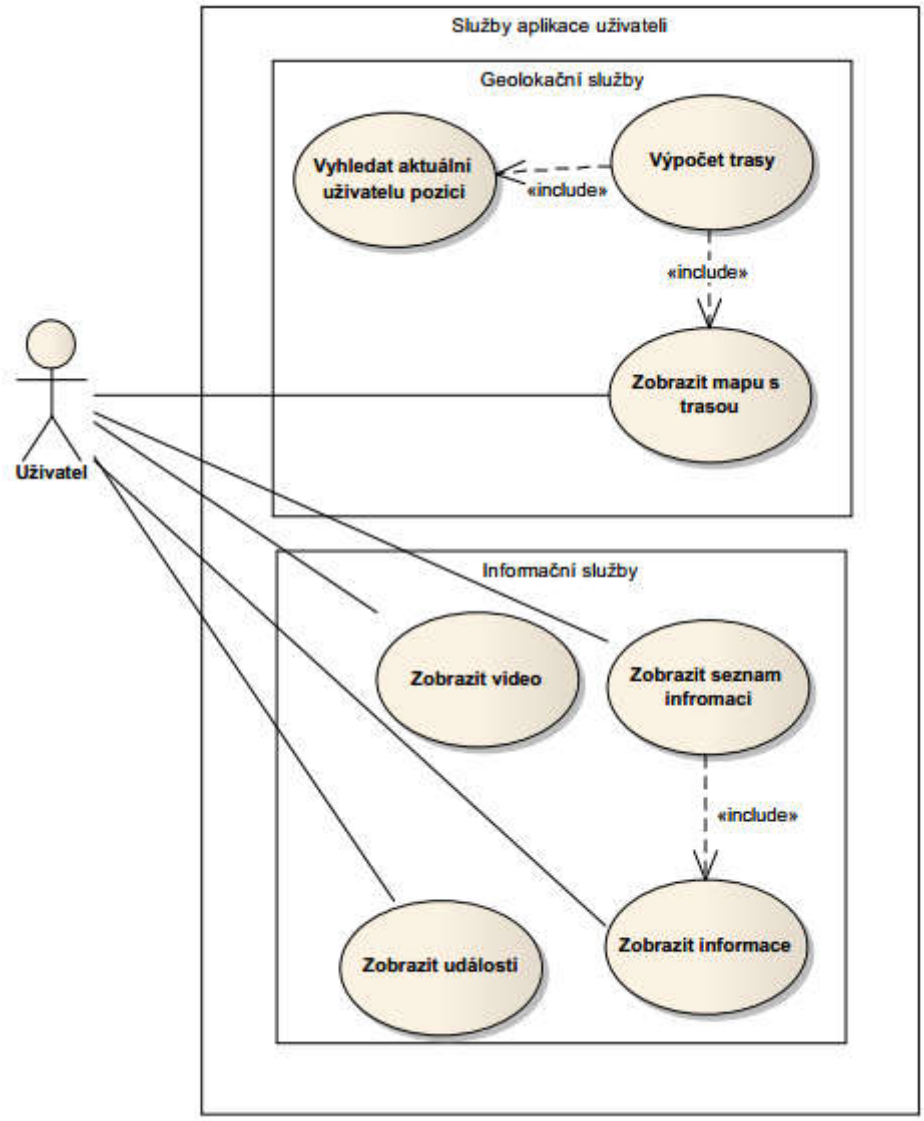
Graf 4: Dotazníkové šetření, Co byste uvítali v aplikaci pro uchazeče? [vlastní tvorba]

Poslední položená otázka uchazečům byla otevřeného typu. Dotaz analyzoval, jaké další funkce by v aplikaci uvítali nebo s jakými potížemi se potýkali při návštěvě fakulty. Nejčastějšími nápady bylo přidat názory a zkušenosti již přijatých studentů, ukázky rozvrhů nebo informace o bydlení. Další návrhy se týkaly ukázek testů v přijímacím řízení. Velký zájem byl i o informace o okolí univerzity. Například, kam je možné se dojít najíst nebo jaké zajímavosti jsou v Hradci Králové k vidění. V aplikaci by se tedy mohla objevit mapka se zajímavými místy nebo přímo i integrace takových míst pomocí GPS souřadnic.

3.2 Funkční analýza

Pro vizualizaci funkčních požadavků aplikace lze použít modelovací jazyk UML, který poskytuje nástroj pro modelování typových úloh (angl. „Use Case Model“). V modelu lze reprezentovat vzájemnou interakci mezi uživatelem a aplikací. Každý případ užití (neboli model typových úloh) je složen z aktérů, činností a vztahů mezi nimi. Aktér je spojen s takovou typovou úlohou, kterou má právo využívat. Pokud jedna úloha ke svému správnému provedení činností potřebuje úlohu druhou, je vztah mezi úlohy označen speciálním indikátorem <<*include*>>.

Následující obrázek byl vytvořen v aplikaci Enterprise Architect a reprezentuje model typových úloh aplikace.



Obr 7: Use Case Model aplikace [autor práce]

Jednotlivé úlohy poskytované uživateli jsou graficky rozděleny do dvou skupin, které jsou popsány v následujících kapitolách.

3.2.1 Geolokační služby

Tato služba řeší lokalizační funkce aplikace za pomocí senzorů (zejména GPS) pro určení polohy uživatele. Má na starosti zejména výpočet trasy mezi aktuální polohou a zadanou polohou fakulty a její následné zobrazení na mapových podkladech.

3.2.2 Informační služby

Uživatel má možnost vyhledávat a zobrazit informace o fakultě, nabízených oborech a konaných přednášek během dne otevřených dveří. Jednotlivé přednášky jsou rozděleny do tří kategorií podle druhu. Do první kategorie patří úvodní přednáška a přednášky o jednotlivých oborech, které se zpravidla opakují každých 30 minut. Další jsou doprovodné akce, které povětšinou probíhají v kuse po celý den otevřených dveří. V poslední kategorii jsou zařazeny workshopy katedry informatiky a kvantitativních metod, jež mají daný přesný časový harmonogram. Každý workshop většinou proběhne jen jednou.

3.3 Vývojové prostředí

Pro vývoj mobilní aplikace bylo zvoleno vývojové prostředí Android Studio. Tento programovací nástroj má na starosti přímo Google Inc. Je založen na známém prostředí IntelliJ IDEA od české firmy JetBrains. Vyniká hlavně velice dobře zpracovaným uživatelským rozhraním pro tvorbu grafického návrhu aplikace. Ten lze vytvářet v grafickém režimu pomocí „přetahování“ jednotlivých komponent do view nebo pomocí psaní XML kódu. Při změně XML kódu ihned vidíme změny v náhledu. Další výhoda tohoto prostředí je v možnosti vybrat si libovolné zařízení s jakýmikoliv parametry pro zobrazování náhledů. Součástí Android Studia jsou i virtuální emulátory, tudíž pro testování běhu aplikace není teoreticky třeba vlastnit reálné zařízení s Android OS. Podporuje operační systémy Windows, Linux a MAC OS.

3.3.1 Android emulátor

Výše zmíněný emulátor integrovaný přímo v Android Studiu byl při testování nahrazen emulátorem Genymotion, jelikož emulátor v Android studiu byl pomalý nebo byl problém s jeho spuštěním. Genymotion je postaven na osvědčeném VirtualBoxu. Využívá x86 virtualizaci a s využitím hardwarové akcelerace OpenGL umožní testovat i 3D aplikace s dostatečným výkonem. Při testování aplikace je možné simulovat senzory, stav baterie, GPS, otočení displeje a další parametry. Při vytváření virtuálního stroje je možné zvolit jakékoliv zařízení s libovolným

systémem a rozlišením displeje. Rovněž podporuje operační systémy Windows, Linux a MAC OS.

3.4 Nastavení manifestu

Jak již bylo řečeno v kapitole 2.4.1, každá aplikace pro Android musí mít nastavena oprávnění k systémovým funkcím a modulům zařízení. To se provádí přidáním řádku *<uses-permission android:name="nazev.package.NAZEV_OPRAVNENI" />* v souboru AndroidManifest.xml.

V aplikaci jsou vyžadována tato oprávnění (čerpáno z [14]):

android.permission.ACCESS_NETWORK_STATE – Umožňuje zjistit stav internetového připojení, zda je uživatel připojen či nikoliv nebo zjistit typ připojení k internetu (Wi-Fi, 3G, LTE).

android.permission.INTERNET – Povoluje aplikaci používat síťové sokety (internet).

com.google.android.providers.gsf.permission.READ_GSERVICE – Povoluje číst z Google Play services konfigurační data. Od verze API 23 a Google Play services 8.1.0 a vyšší není vyžadováno. Zachováno kvůli zpětné kompatibilitě.

android.permission.ACCESS_FINE_LOCATION – Povoluje API používat Wi-Fi nebo mobilní data k určení polohy zařízení. Přesnost v rozmezí jednoho městského bloku.

android.permission.ACCESS_COARSE_LOCATION – Umožňuje co nejpřesněji určit polohu pomocí GPS modulu. Možnost kombinace s dostupnými Wi-Fi a mobilními daty.

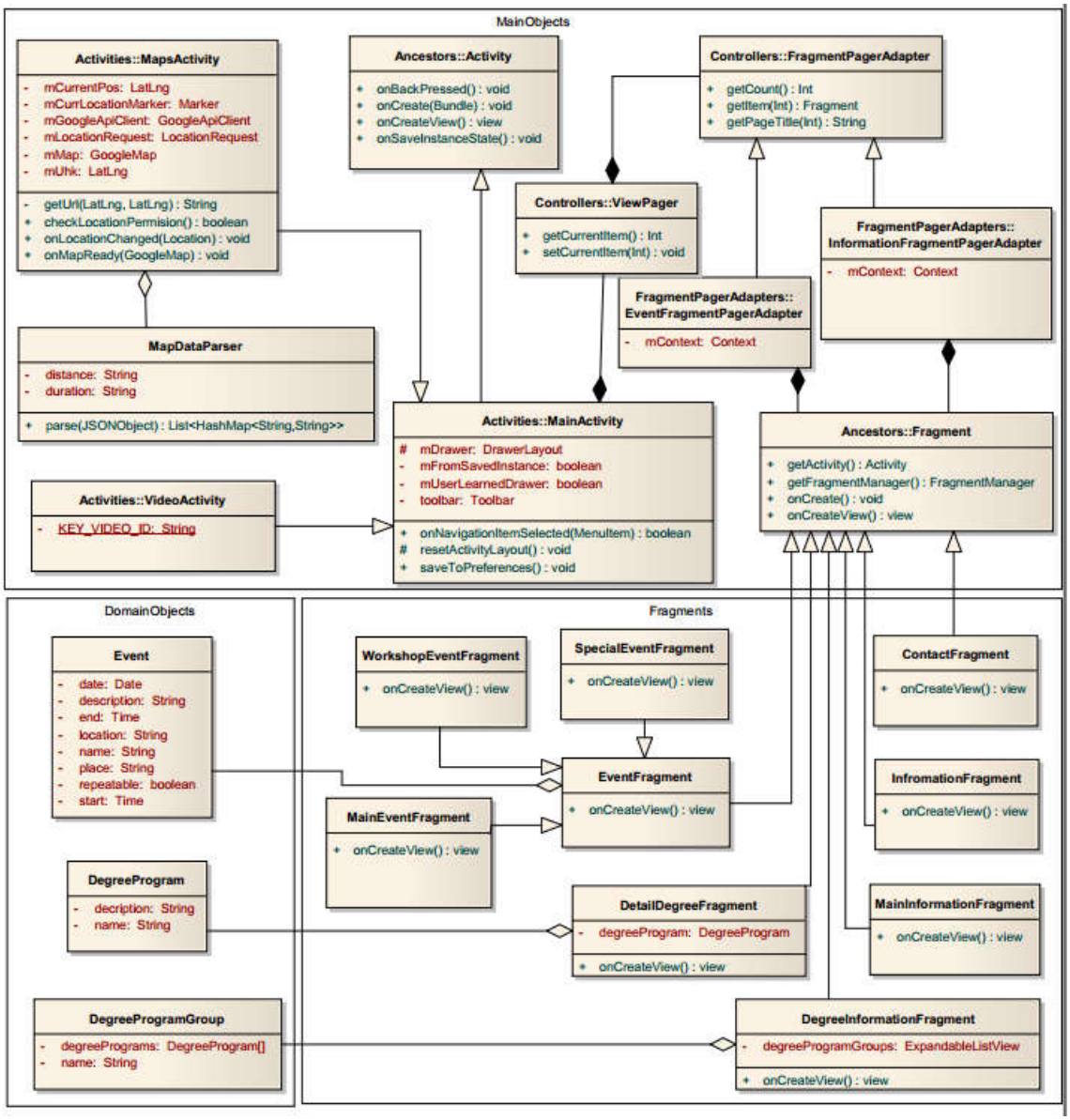
android.permission.WRITE_EXTERNAL_STORAGE – Povoluje zápis do externího úložiště. V aplikaci použito z důvodu využití Google Maps API a jeho zachování zpětné kompatibility pro zařízení s API úrovní 18 a nižší.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cz.uhk.fim.denotevrenychdveri">

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

    <meta-data
        android:name="com.google.android.gms.version"
        android:value="9256000" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Open days on FIM"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/DEVELOPER_KEY" />
```

Výpis 1: Android manifest s vloženými oprávněními [autor práce]

3.5 Logický model aplikace

Při spuštění aplikace dochází k inicializaci hlavní aktivity - *MainActivity*. Jakmile je instance aktivity připravena, začnou se vytvářet objekty tvořící funkce aplikace. První se vytvářejí jednotlivé fragmenty, přičemž dojde ke zpracování a vykreslení vlastních layoutů. Následně se propojí adaptéry s pohledy (view) seznamů. Dále se vytvoří veškeré datové objekty, které už si mohou načíst například jednotlivé adaptéry a následně naplnit data do připravených fragmentů. Každý objekt v aplikaci komunikuje skrze hlavní aktivitu. Níže na obrázku 8 je vyjádřena logická struktura a dědičnost mezi jednotlivými třídami v aplikaci.



Obr 8: Class Diagram [autor práce]

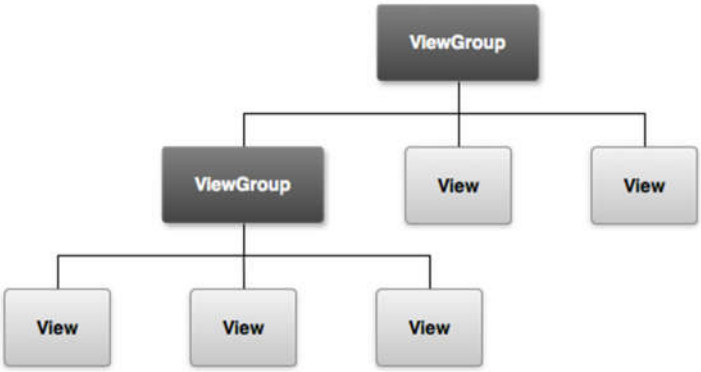
3.6 Grafický návrh jednotlivých obrazovek aplikace

V této kapitole jsou popsány jednotlivé části aplikace z pohledu tvorby uživatelského rozhraní. Každá obrazovka má nadefinovanou vlastní podobu v XML souboru, který je v rámci projektu umístěn ve složce Layouts. Před kódováním konkrétních obrazovek si vývojář většinou vytvoří grafický náčrt. Buď pomocí grafických programů k tomu určených, jako je například volně dostupný Pencil [17], nebo jednodušeji ručně na papír. Výhoda v použití softwaru je ta, že rovnou máte přehled o grafických prvcích, které lze v Androidu použít, což je vhodné zejména pro začínající vývojáře.

Musíme brát v potaz, že zařízení, na kterých bude aplikace spuštěna, mají různou velikost obrazovky a různé rozlišení. Proto při vývoji aplikace je důležité dbát na správné zobrazení uživatelského rozhraní na libovolném displeji. V žádném případě nepoužíváme definování konkrétních souřadnic polohy a rozměrů ovládacích prvků. Místo toho používáme k uspořádání grafických prvků definici vzájemných poloh. Dobré je také zvážit, jestli pro různou orientaci zařízení nezvolit rozdílné uspořádání prvků, abychom využili co nejlépe plochu obrazovky v daném režimu. Základem tvorby uživatelského prostředí jsou pohledy a skupiny pohledů [1].

View (pohled) je základní třída pro tvorbu vizuálních prvků uživatelského rozhraní. Všechny grafické prvky uživatelského rozhraní jsou odvozeny od pohledů.

ViewGroup (skupina pohledů) se používají pro tvorbu složených prvků, které se skládají z více pohledů.



Obr 9: Hierarchie ViewGroup [24]

3.6.1 Přizpůsobení různým velikostem a rozlišením obrazovky

Jak již bylo řečeno, aplikace může být naistalována na zařízeních s různými displeji, které mají odlišnou hustotu pixelů. Na to je důležité myslet, zejména při používání obrázků nebo návrhu ikon aplikace. Jednoduše obrázek určený pro malý displej se bude jevit na displeji s Full HD nebo dokonce 4K rozlišením neostře („kostičkovaně“). Android u spuštěné aplikace kontroluje vlastnosti displeje daného zařízení a načítá z resources (zdrojů) obrázky nebo ikonu v rozlišení odpovídající vlastnostem displeje. Rozlišení se uvádí v jednotce dpi (dots per pinch), tj. počet bodů na palec. Od toho je odvozená jednotka dp (density independent pixel), která se používá například při udávání rozměrů okrajů nebo rozestupů mezi prvky.

Přepočet pixelů na dp je dán vztahem ***px = dp * (dpi / 160)***. Jednotka dp tedy představuje jakousi fyzickou velikost displeje.

Název	Zkratka	Jemnost displeje	Poměr	Měřítko
Low	ldpi	120 dpi	3	0,75
Medium	mdpi	160 dpi	4	1,00
TV	tvdpi	213 dpi	5,325	1,33
Hight	hdpi	240 dpi	6	1,50
Extra Hight	xhdpi	320 dpi	8	2,00
Extra Extra Hight	xxhdpi	480 dpi	12	3,00
Extra Extra Extra Hight	xxxhdpi	640 dpi	16	4,00

Obr 10: Rozdělení displejů podle dpi (vlastní zpracování podle [25])

3.6.2 Rozmisťování prvků

K nadefinování rozmístění prvků uživatelského rozhraní slouží takzvané layouty (vizuální kontejnery), které jsou odvozeny od *ViewGroup*. Nejčastěji se k rozestavění prvků používá LinearLayout a RelativeLayout.

3.6.2.1 LinearLayout

Umožňuje uspořádat všechny vnořené prvky (*View*) v jednom směru, horizontálně do jednoho řádku nebo vertikálně do jednoho sloupce. Důležité parametry pro nastavení LinerLayoutu jsou:

orinentation – nastaví směr uspořádání vnořených prvků (*horizontal* nebo *vertical*)

layout_gravity – definuje umístění centrálního bodu, ke kterému se zarovnávají všechna vnořená *View*

layout_weight – vyjadřuje „významnost“ vnořeného *View*, která určuje, kolik místa může na obrazovce zabrat. Čím větší hodnota, tím více prostoru může vyplnit. Defaultní hodnota je nula.

3.6.2.2 RelativeLayout

V relativních layoutech je možné nastavit vzájemné poziční vazby mezi jednotlivými vizuálními prvky uživatelského rozhraní (View) nebo vůči rodičovskému layoutu. RelativeLayout je efektivní nástroj pro tvorbu uživatelského rozhraní, protože při složitějších layoutech, kde máme mnoho do sebe vnořených LinearLayoutů, je většinou možné je nahradit jedním RelativeLayoutem, což je důležité hlavně

z hlediska výkonu, jelikož mnoho vnořených layoutů může být velice výpočetně náročné a na slabších zařízeních může dojít k poklesu plynulosti aplikace. Aby bylo možné definovat vzájemné vazby, každý prvek musí mít jednoznačný identifikátor – id.

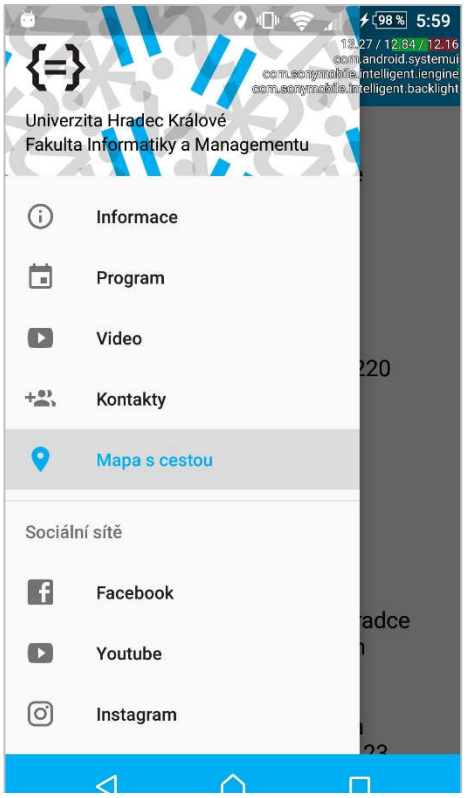
3.6.3 Záložkové rozložení (TabLayout)

Záložkové rozložení je vhodné použít, pokud víme, že mezi obrazovkami budeme často přepínat a nechceme čekat na pomalejší načítání celých nových aktivit se všemi vloženými komponentami [1]. Záložky se implementují pomocí předpřipraveného layoutu ViewPager. K tomuto „zobrazovači" je připojený adaptér, ve kterém nastavíme jednotlivým záložkám předpřipravené layouty jednotlivých stránek. ViewPager už si sám obstará načítání obsahu záložek a jejich přepínání. Výhodou je, že se načtou všechny záložky najednou při spuštění aktivity. Mezi záložkami lze přepínat pomocí kliknutí na název v záhlaví nebo pomocí gest tažením prstu od stran obrazovky ke středu.

3.7 Menu aplikace

Pro hlavní navigaci v aplikaci je použito postranní vysouvací menu. Při prvním spuštění aplikace je uživateli dána najevo existenci postranního menu jeho automatickým vysunutím. Menu je možné zobrazit kliknutím na známou „hamburgerovou" ikonku v levém horním rohu *Toolbaru* nebo táhlým gestem z kraje do středu aplikace. Všechny aktivity mají přístup do hlavního menu.

Pro implementaci menu bylo použito pro vývojáře již připravené *View* zvané DrawerLayout, které se skládá ze dvou částí. První část obsahuje navigační panel s jednotlivými položkami menu, který se



Obr 11: Postranní vysouvací menu [autor práce]

připojuje ke druhému panelu, jenž představuje veškerý obsah aplikace.

Seznam položek v menu aplikace se dělí na dvě skupiny. První částí se pohybujeme uvnitř dané aplikace. Pod ní se nachází druhý oddíl zvaný „Sociální sítě“, jenž odkazuje na externí sociální aplikace, které má uživatel nainstalované ve svém zařízení. Pokud uživatel tyto aplikace nemá, tak se daná sociální síť otevře ve webovém prohlížeči.

3.8 Informace o fakultě

Při spuštění aplikace je zobrazena úvodní obrazovka se základními informacemi o dnu otevřených dveřích, fakultě a přehled studijních oborů.

Aktivita používá již zmíněné záložkové rozdělení a je rozdělena do dvou tabů. Každý tab obsahuje jeden fragment.

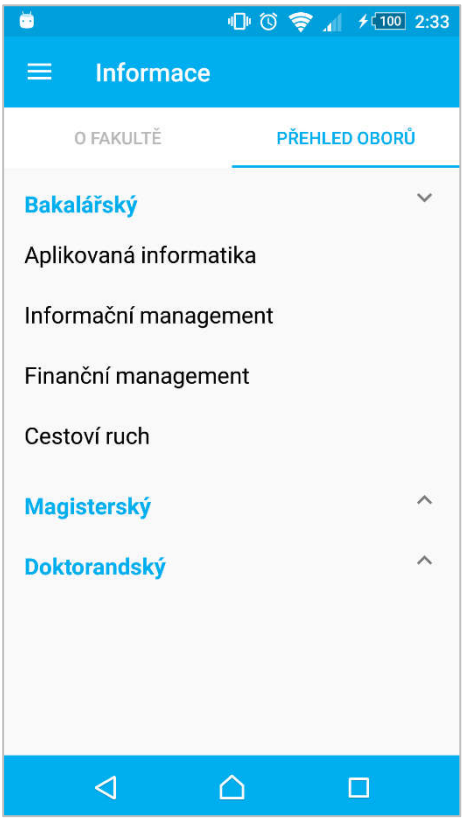
Na první záložce jsou zobrazené obecné informace o fakultě. Celý obsah je zabalen do *ScrollView* pro umožnění vertikálního posouvání v rámci fragmentu, aby byly všechny informace čitelné na všech typech obrazovek zařízení, bez ohledu na velikost a rozlišení displeje.

Druhý tab obsahuje přehled studijních oborů rozdělených hierarchicky podle typu studia. Seznam oborů je vytvořen za pomoci *ExpandableListView*. Jedná se o rozšířené adaptérové *View* pro zobrazení několika položkového seznamu, které dědí od *ListView*. Skládá se z více vnořených listů, které lze rozbalovat a skrývat. V tomto případě jsou zobrazeny tři typy listů, každý pro jeden typ studia. U každého oboru je zobrazen jeho název a případně jazykové zaměření. Po kliknutí na vybraný obor se zobrazí jeho detail.

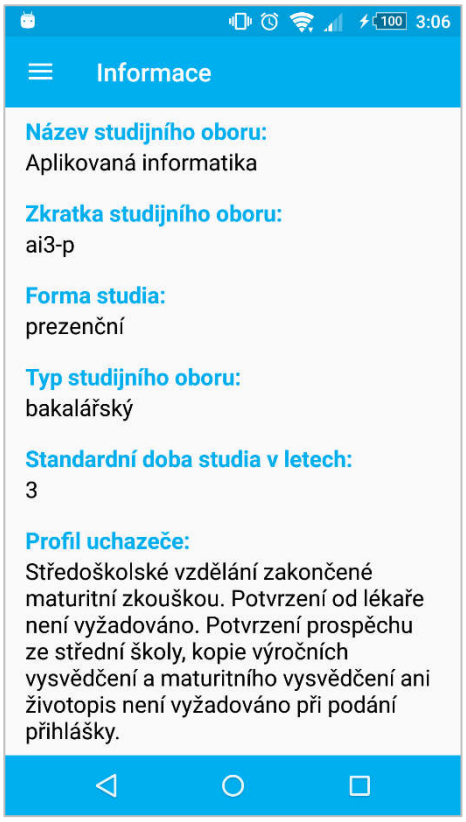
```
<?xml version="1.0" encoding="utf-8"?>
<ExpandableListView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/degree_list"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:groupIndicator="@null"
    android:divider="@android:color/transparent"
    android:childDivider="@android:color/transparent"
    />
```

Výpis 2: ExpandableListView v XML layoutu [autor práce]

V detailu je zobrazen výpis s informacemi o konkrétním oboru. Layout detailu je opět zabalen do *ScrollView* pro svislé posouvání.



Obr 12: Přehled oborů za použití ExpandableListView [autor práce]

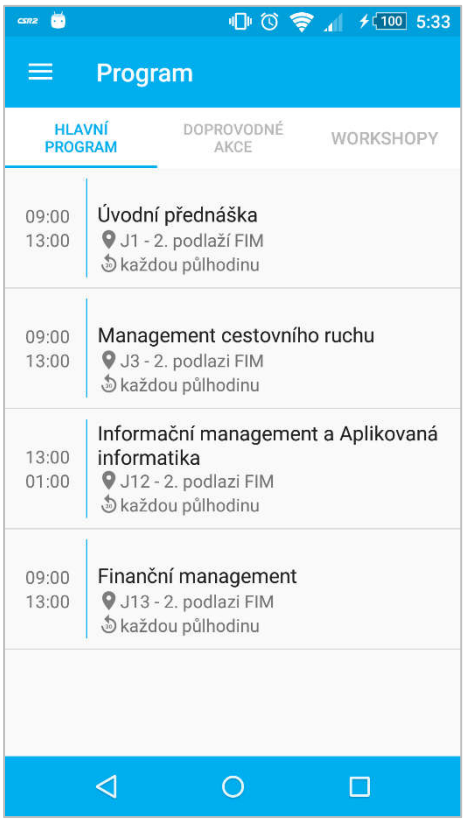


Obr 13: Detail oboru [autor práce]

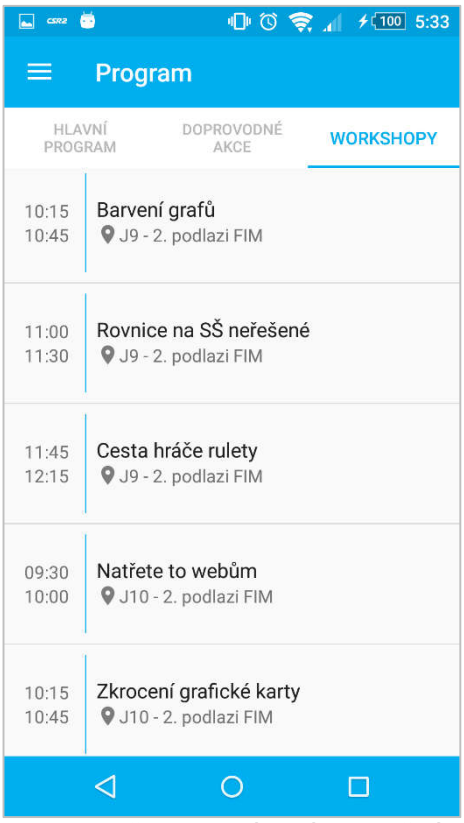
3.9 Program

Obrazovka program slouží k zobrazení přehledu o všech konaných událostech v průběhu dne otevřených dveří. Skládá se z *TabLayoutu* o třech tabech. V první záložce jsou zobrazeny hlavní přednášky, pro které je specifické pravidelné opakování po 30 minutách. Pravidelně se opakující události jsou označeny ikonou a popiskem. Ve druhém a třetím tabu jsou zobrazeny další doprovodné akce a workshopy.

Každý tab obsahuje komponentu *ListView* se seznamem událostí. Pro grafické rozložení každého itemu listu je přes adapter přiřazen specifický XML layout, který se skládá z několika vnořených *ViewGroup*. Každá položka seznamu má uveden čas rozmezí události, název události, místo konání a vyznačenou případnou periodičnost.



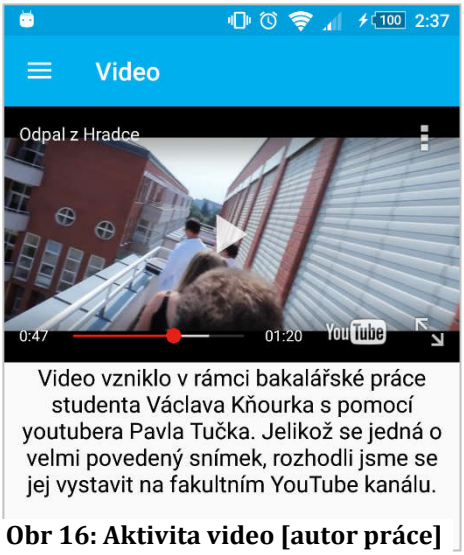
Obr 14: Přehled událostí za použití ListView [autor práce]



Obr 15: Přehled událostí za použití ListView [autor práce]

3.10 Video

Aktivita video zobrazuje přehrávač s propagačním videoklipem FIMky nahraný na serveru YouTube. Pod videem je doplněn krátký popisek. K zobrazení přehrávače byl použit YouTube Android Player API. K používání přehrávače musí mít uživatel v mobilu nainstalovanou YouTube aplikaci z Google Play Store. Minimální verze podporované aplikace je 4.2.16. Pokud má uživatel starší verzi nebo aplikaci nemá vůbec, API mu to oznámí a přesměruje do obchodu s aplikacemi, kde si ji může bezplatně stáhnout. [16]



Obr 16: Aktivita video [autor práce]

3.10.1 YouTube Android Player API

YouTube Android Player API umožňuje integrovat všechny funkce pro přehrávání videa přímo do vaší aplikace. API definuje metody pro načítání a přehrávání YouTube videí (a video playlistů) a také pro přizpůsobení a ovládání přehrávače videa. Například můžete spustit přehrávání, pozastavit video nebo nastavit přehrávač na daný čas videa pomocí vlastních tlačítek. API také samozřejmě podporuje aplikování posluchačů událostí („listenerů“) k nastavení zpětných volání („callbacků“), jako je třeba kontrola načtení videa nebo změna statusu přehrávače. API má také v sobě již implementovány funkcionality na podporu přetočení videa vlivem změny orientace zařízení nebo přepnutí videa do celoobrazovkového režimu (fullscreen).

Velice důležité je také zaregistrovat aplikaci ve vývojářské konzoli (Google Developers Console), kde se vygeneruje unikátní API klíč. Vygenerovaný klíč se vloží do manifestu aplikace. Bez zaregistrování a vložení API klíče nelze YouTube Android API v aplikaci použít.

3.11 Mapa s cestou do FIM

Tato aktivita načítá aktuální polohu zařízení a vypočítá nejkratší cestu do budovy FIM. Cestu zobrazí na mapových podkladech. Největší část obrazovky překrývá právě mapa, která je doplněna o údaje vzdálenosti a odhadovaného času do cílového místa.

K výpočtu trasy mezi aktuální polohou a budovou FIM se používá Google Maps Directions API. Celé to funguje na systému HTTP(S) požadavků, které se odesílají přes sestavenou URL adresu na server Googlu a ten zpět odešle vypočítanou trasu ve formátu XML nebo JSON. API podporuje odesílání požadavků jak přes nešifrovaný (http) protokol, tak i šifrovaný (https). Jelikož se jedná o práci s citlivými daty uživatelů (jejich aktuální poloha), mělo by se výhradně používat jen šifrované spojení. [18] Základní adresa URL vypadá takto:

*http://maps.googleapis.com/maps/api/directions/**outputFormat?parameters***

Za *outputFormat* se zvolí výstupní formát. V aplikaci byl zvolen na základně osobních preferencí JSON, jelikož autor má více zkušeností s jeho parsováním. Pro parsování JSONu byla vytvořena speciální třída.

```
public List<List<HashMap<String,String>>> parse(JSONObject jObject){

    List<List<HashMap<String, String>>> routes = new ArrayList<>() ;
    JSONArray jRoutes;
    JSONArray jLegs;
    JSONArray jSteps;
    try {
        jRoutes = jObject.getJSONArray("routes");

        /** Traversing all routes */
        for(int i=0;i<jRoutes.length();i++){
            jLegs = ( (JSONObject)jRoutes.get(i)).getJSONArray("legs");
            List path = new ArrayList<>();

            /** Traversing all legs */
            for(int j=0;j<jLegs.length();j++){
                jSteps = ( (JSONObject)jLegs.get(j)).getJSONArray("steps");
                distance = (String)((JSONObject)((JSONObject)jLegs.get(j)).get("distance")).get("text");
                duration = (String)((JSONObject)((JSONObject)jLegs.get(j)).get("duration")).get("text");

                /** Traversing all steps */
                for(int k=0;k<jSteps.length();k++){
                    String polyline = "";
                    polyline = (String)((JSONObject)((JSONObject)jSteps.get(k)).get("polyline")).get("points");
                    List<LatLng> list = decodePoly(polyline);

                    /** Traversing all points */
                    for(int l=0;l<list.size();l++){
                        HashMap<String, String> hm = new HashMap<>();
                        hm.put("lat", Double.toString((list.get(l)).latitude) );
                        hm.put("lng", Double.toString((list.get(l)).longitude) );
                        path.add(hm);
                    }
                }
                routes.add(path);
```

Výpis 3: Algoritmus na procházení JSON objektů [autor práce]

Základní parametry Google Maps Directions API (čerpáno z [18]):

origin – Počáteční bod trasy, může být zadán buď souřadnicemi zeměpisné šířky a délky nebo tzv. place ID. Place ID je unikátní kód, který identifikuje umístění v Google Places databázi a na Google mapapách. Place ID pro konkrétní adresu lze zjistit na této adrese: https://developers.google.com/places/place-id.

destination – Konečný bod trasy, stejné podmínky pro zadávání jako parametr origin.

key – Vygenerovaný API klíč.

mode – Nastavuje typ dopravního prostředku, pro který má vyhledávat trasy.

waypoints – Body přes, které bude generována trasa.

traffic_model – Určuje, jakým způsobem se bude odhadovat čas do cíle.

3.11.1 Lokalizační služby

Většina přístrojů s Androidem může přijímat signál GPS z družic, případně určit aktuální polohu pomocí triangulace GSM stanic či z Wi-Fi přístupových bodů. Proto Android nabízí třídu *Location*, která reprezentuje polohu. Třída obsahuje data o zeměpisné šířce a délce, aktuálním čase, přesnosti polohy, rychlosti, nadmořské výšce a podobně. Aplikace využívající lokalizační služby musí požádat o povolení poskytnutí údajů, viz kapitola 3.3 o nastavení manifestu. Další důležitá třída je *LocationManager*. Jedná se o systémovou službu, která umožnuje určit poslední známou polohu uživatele nebo jestli se zařízení vzdaluje nebo blíží k dané geografické oblasti. Android také poskytuje *LocationListener*, který volá callback metody například při změně stavu objektů z třídy Location. Nejvíce používaná metoda je *onLocationChanged()*, která se volá v případě změny polohy. Může například zjistit vzdálenost od jiné lokace. [1]

3.11.2 Google Maps Android API

K zobrazení polohy na mapě poskytuje Google mapovou službu Google Maps Android API. Služba poskytuje několik druhů map. Klasické cestovní mapy, satelitní mapy se snímky zemského povrchu nebo jejich kombinaci.

API umožňuje přidat ikony, tzv. markery, na konkrétní místa na mapě nebo kreslit na popředí, přičemž mapový podklad je na pozadí. Může se využít například pro zobrazení trasy na mapě. Mapa také reaguje na známá gesta, nejčastěji pro přiblížení, pootočení nebo změnu meřítka. API také dokáže zobrazit aktuální polohu zařízení. [1][19]

Pro práci s mapami poskytuje API tyto základní třídy:

GoogleMap – třída reprezentuje mapu a umožnuje její správu

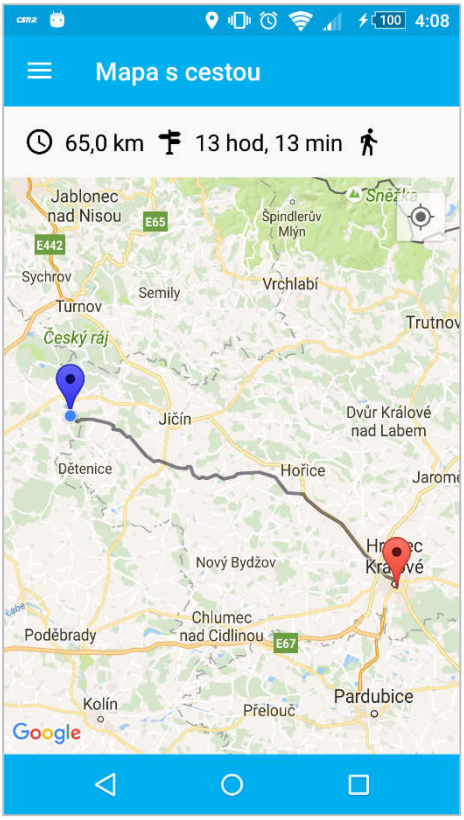
MapFragment – zobrazení mapy ve fragmentu

Marker – třída zastupuje ikony nebo vyskakovací okna, která ukazují na daná místa

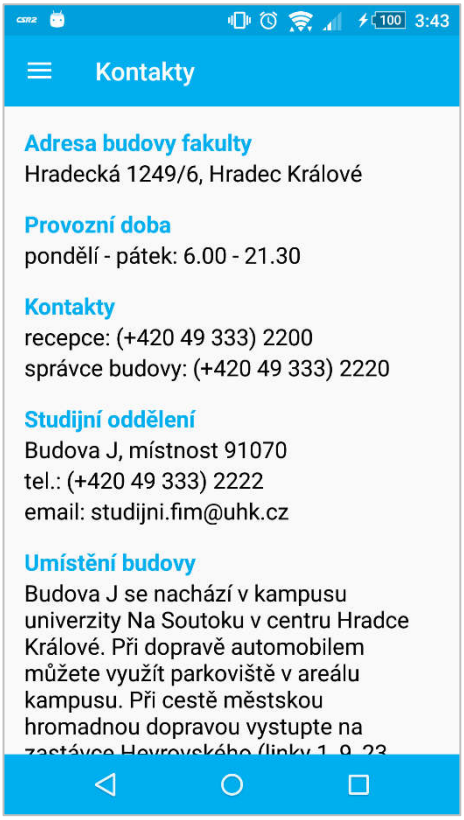
Camera – definuje tu část mapy, která se právě zobrazuje na obrazovce zařízení

Pro umožnění používání mapového API se také musí aplikace zaregistrovat přes vývojářskou konzoli od Googlu a vygenerovat API klíč, který se umístí do manifestu aplikace. API je poskytováno bezplatně, avšak má svá omezení. V základním bezplatném balíčku „*standart*“ je nastaveno 2500 požadavků o výpočet

trasy za den, maximálně zadaných 23 bodů pro výpočet trasy a omezení na maximum odeslaných 50 požadavků za sekundu na server. [18]



Obr 17: Aktivita s mapou [autor práce]



Obr 18: Obrazovka kontakty [autor práce]

4 Testování aplikace

Aplikace byla testována na třech zařízeních. Na Sony Xperia Tipo s Androidem 4.0.4 IceCream Sandwich s mdpi rozlišením displeje, dále na Sony Ericsson Xperia Ray s Andoidem 4.3 Jelly Bean reprezentující hdpi displej a třetí testovací zařízení nese označení Sony Xperia Z2 s nejnovějším Androidem 6.0.1 Marshmallow představující kategorii xxhdpi obrazovek.

K testování během vývoje byly použity dvě základní třídy, Log a Toust. Log se používá k vypisování zpráv přímo do konzole vývojového prostředí. Naopak třída Toust umožňuje na daný časový interval zobrazit jakoukoliv zprávu přímo na displej. Běžně se i používá k zobrazení informačních zpráv uživateli.

4.1 Výsledky testování

Na všech zařízeních se při prvním spuštění správně automaticky vysunulo navigační menu. Dále bylo kontrolováno, jestli jsou korektně zobrazeny všechny komponenty a zda jsou informace plně čitelné na všech typech velikostí displeje. Zde se vyskytl problém na zařízení s menším displejem a ne všechen text v komponentách ListView byl čitelný. Následně tedy proběhly úpravy v rozložení layoutu jednotlivých položek listu a chyba se zobrazením opravena.

Dále bylo testováno, co se stane na aktivitě s vloženým videem, když zařízení nemá nainstalovanou YouTube aplikaci. Zde nedošlo k žádnému překvapení, jelikož plnou funkcionalitu zde ověřuje samotné API a uživateli byla zobrazena zpráva o chybějící aplikaci a byl odkázán ke stažení aplikace z Google Play Storu. Dalším testům se podrobily odkazy v menu na sociální sítě. Zde nenastal žádný problém a při chybějící aplikaci pro podporu dané sociální sítě byla stránka otevřena v klasickém webovém prohlížeči.

Během testování aplikace běžela plynule na všech zařízeních, nebylo ani zaznamenáno zpomalení systému po nainstalování aplikace.

4.2 Testování na subjektech

Po dokončení implementace a poctivém závěrečném testování bylo zařízení s aplikací předáno 8 lidem, kteří aplikaci nikdy předtím neviděli. Byla testována reakce na vzhled aplikace a zároveň, zda se dokáží v aplikaci správně zorientovat.

Testovacím subjektům byly zadávány úkoly, díky kterým postupně prošli celou aplikaci. Vlivem jednoduchého menu nebyl pro nikoho problém se v aplikaci pohybovat. Po grafické stránce byli uživatelé převážně spokojeni. Jeden respondent pouze kritizoval příliš světlý design.

5 Závěr

Během vypracování bakalářské práce si autor značně prohloubil znalosti a nasbíral spoustu zkušeností s programováním mobilních aplikací pro operační systém Android. Úspěšně si osvojil a aplikoval nové moderní grafické uživatelské rozhraní s různými ovládacími prvky ve formulářích podle zásad Material Desingu. Aplikace byla za pomoci rozšířených vlastností *resources* úspěšně optimalizována pro displeje všech velikostí. Byly vloženy obrázky a ikony pro všechny možné typy obrazovek s různou hustotou pixelů.

Do aplikace bylo úspěšně implementováno YouTube Android API pro přehrávání videí. Také se podařilo integrovat Google Maps Android API pro práci s mapovými podklady a využít Directions API pro sestavení trasy mezi dvěma body. Podařilo se pracovat s lokalizačními moduly mobilního zařízení pro zjišťování aktuální polohy.

5.1 Shrnutí výsledků

Výsledkem této práce je základní popis systému Android a jeho součástí, dále rozebrání jednotlivých kroků a postupů při vývoji mobilní aplikace. Práce zahrnuje rady a typy pro používání přídavných API (Google Maps a YouTube Player).

K aplikaci bylo vytvořeno grafické uživatelské rozhraní založené na barvách fakulty a s využitím prvků moderního Material Desingu. Aplikace byla navržena tak, aby se správně zobrazovala na různých velikostech displeje (viz kapitola 4).

5.2 Doporučení

Pokud bude koncept aplikace vhodný, bude v budoucnu pro pohodlnější správu událostí a dat příhodné, vytvořit k aplikaci serverovou část s uživatelským rozhraním pro vkládání a editaci dat. Aplikace si poté bude stahovat ze serveru data ve formátu JSON a má již v sobě připravené doménové třídy, které usnadní případnou integraci parserů a databáze. Aplikace je také připravena pro anglickou lokalizaci. Jazykové mutace se dají jednoduše vytvořit přímo přes Android Studio. Užitečnou funkcí by také bylo vyznačení dalších vhodných míst v okolí univerzity (vědecká knihovna, vysokoškolské koleje, menza atp.) přímo na mapových podkladech.

6 Seznam použité literatury

[1] LACKO, Ľuboslav. *Vývoj aplikací pro Android*. 1. vyd. Brno: Computer Press, 2015, 472 s. ISBN 978-80-251-4347-6.

[2] Smartphone OS Market Share, Q1 2015. In: IDC [online]. [cit. 28.08.2015]. Dostupné z WWW: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[3] ART and Dalvik. In: Android Developers [online]. [cit. 28.08.2015]. Dostupné z WWW: https://source.android.com/devices/tech/dalvik/index.html

[4] Activities. In: Android Developers [online]. [cit. 28.08.2015]. Dostupné z WWW: http://developer.android.com/guide/components/activities.html

[5] „Task killery" jsou zbytečné, tvrdí i Google a Cyanogen. In: Svět Androida [online]. [cit. 28.08.2015]. Dostupné z WWW: http://www.svetandroida.cz/task-killery-201501

[6] Services. In: Android Developers [online]. [cit. 28.08.2015]. Dostupné z WWW: http://developer.android.com/guide/components/services.html

[7] Vyvíjíme pro Android: Notifikace, broadcast receivery a Internet. In: Zdroják.cz [online]. [cit. 28.08.2015]. Dostupné z WWW: http://www.zdrojak.cz/clanky/vyvijime-pro-android-notifikace-broadcast-receivery-a-internet/

[8] UJBÁNYAI, Ing. Miroslav. *Programujeme pro Android*. 1. vyd. Praha: Grada Publishing, a.s., 2012, 192 s. ISBN 978-80-274-3995-3.

[9] Security. In: Android Developers [online]. [cit. 28.08.2015]. Dostupné z WWW: https://source.android.com/devices/tech/security/

[10] Android 6.0 Marshmallow: everything you need to know. In: AndroidPIT [online]. [cit. 28.08.2015]. Dostupné z WWW: https://www.androidpit.com/android-m-release-date-news-features-name

[11] Dashboards. In: Android Developers [online]. [cit. 28.08.2015]. Dostupné z WWW: http://developer.android.com/about/dashboards/index.html

[12] Support Library. In: Android Developers [online]. [cit. 28.08.2015]. Dostupné z WWW: http://developer.android.com/tools/support-library/index.html#overview

[13] Material desing. In: Google [online]. [cit. 28.08.2015]. Dostupné z WWW: http://www.google.com/design/spec/material-design/

[14] Manifest.permission. In: Android Developers [online]. [cit. 18.07.2016]. Dostupné z WWW: http://developer.android.com/reference/android/Manifest.permission.html

[15] Icons. In: Google Material Desing [online]. [cit. 18.07.2016]. Dostupné z WWW: https://material.google.com/style/icons.html

[16] YouTube Android Player API. In: Google Developers [online]. [cit. 18.07.2016]. Dostupné z WWW: https://developers.google.com/youtube/android/player/

[17] Pencil. In: GitHub [online]. [cit. 18.07.2016]. Dostupné z WWW: https://github.com/prikhi/pencil

[18] Directions API. In: Google Maps API [online]. [cit. 18.07.2016]. Dostupné z WWW: https://developers.google.com/maps/documentation/directions/usage-limits

[19] Google Maps API. In: Google Developers [online]. [cit. 18.07.2016]. Dostupné z WWW: https://developers.google.com/maps/

[20] Android Nougat Features. In: Compareraja [online]. [cit. 18.07.2016]. Dostupné z WWW: http://www.compareraja.in/blog/android-nougat-features/

[21] Mobilní aplikace. In: Wikipedia: the free encyclopedia. [online]. 2001- [cit. 2016-08-14]. Dostupné z WWW: https://cs.wikipedia.org/wiki/Mobiln%C3%AD_aplikace

[22] Android Developers. [online]. [cit. 28.08.2015]. Dostupné z WWW: http://developer.android.com/index.html

[23] Android (operating system). In: Wikipedia: the free encyclopedia. [online]. 2001- [cit. 2016-08-14]. Dostupné z WWW: https://en.wikipedia.org/wiki/Android_(operating_system)

[24] Rozšíření informačního systému EPI. In: Kiwiki. [online]. [cit. 2016-08-14]. Dostupné z WWW: http://www.kiwiki.info/index.php/Roz%C5%A1%C3%AD%C5%99en%C3%AD_informa%C4%8Dn%C3%ADho_syst%C3%A9mu_EPI,_s.r.o._na_mobiln%C3%AD_platformu_Android

[25] Supporting Multiple Screens. In: Android Developers [online]. [cit. 18.07.2016]. Dostupné z WWW: https://developer.android.com/guide/practices/screens_support.html

7 Přílohy

K práci je přiložen kompaktní disk, na kterém jsou uloženy všechny zdrojové kódy.

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2015/2016

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Salay Dušan	Zahrádky 318, Dolní Bousov	I1301330

TÉMA ČESKY:

Mobilní aplikace pro podporu dne otevřených dveří na FIM

TÉMA ANGLICKY:

Mobile application for support open days on FIM

VEDOUCÍ PRÁCE:

doc. Ing. Filip Malý, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl práce:
Cílem práce je vytvořit mobilní aplikaci pro podporu dne otevřených dveřích na FIM a popsat použité technologie.

Osnova práce:
1. Úvod
2. OS Android
3. Vývoj aplikace
4. Testování aplikace
5. Závěr
6. Literatura

SEZNAM DOPORUČENÉ LITERATURY:

LACKO, Ľuboslav. Vývoj aplikací pro Android. 1. vyd. Brno: Computer Press, 2015, 472 s. ISBN 978-80-251-4347-6.

SCHILDT, Herbert. Mistrovství - Java. 1. vyd. Brno: Computer Press, 2014, 1224 s. Mistrovství. ISBN 978-80-251-4145-8.

Podpis studenta: ..... **Datum:** 13.10.2015

Podpis vedoucího práce: ..... **Datum:** 13.10.2015