VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

ANALÝZA TRHU MOBILNÍCH APLIKACÍ NA PLATFORMĚ ANDROID A NÁVRH VLASTNÍ APLIKACE

ANALYSIS OF MOBILE APPLICATIONS MARKET ON ANDROID PLATFORM AND DESIGN OF OWN APPLICATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jaroslav Beneš**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Dydowicz, Ph.D.**
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2015/2016
Ústav informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Beneš Jaroslav, Bc.

Informační management (6209T015)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává diplomovou práci s názvem:

Analýza trhu mobilních aplikací na platformě Android a návrh vlastní aplikace

v anglickém jazyce:

Analysis of Mobile Applications Market on Android Platform and Design of Own Application

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrh řešení, přínos práce
Závěr
Seznam použité literatury

Seznam odborné literatury:

GARGENTA, M. Learning Android. 1. vyd. Sebastopol, Calif.: O'Reilly, 2011. 245 s. ISBN 14-493-9050-1.
LEE, W. Beginning Android application development. Indianapolis, IN: Wiley Pub., 2011. 428 s. Wrox beginning guides. ISBN 978-111-8087-800.
MARTIŠEK, D. Algoritmizace a programování v Delphi. 1. vyd. Brno: Littera, 2007. 230 s. ISBN 978-80-85763-37-9.
UJBÁNYAI, M. Programujeme pro Android. 1. vyd. Praha: Grada, 2012. 187 s. Průvodce Grada. ISBN 978-80-247-3995-3.
VELTE, A., T. VELTE a R. ELSENPETER. Cloud Computing: praktický průvodce. 1. vyd. Brno: Computer Press, 2011. 344 s. ISBN 978-80-251-3333-0.

Vedoucí diplomové práce: Ing. Petr Dydowicz, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2015/2016.

L.S.

doc. RNDr. Bedřich Půža, CSc.	doc. Ing. et Ing. Stanislav Škapa, Ph.D.
Ředitel ústavu	Děkan fakulty

V Brně, dne 29.2.2016

Abstrakt

Cílem práce je sumarizovat charakteristické vlastnosti úspěšných a ziskových aplikací na trhu mobilních aplikací s platformou Android a vytvořit tak odrazový bod pro začínající vývojáře. Trh mobilních aplikací je dnes otevřený a má vlastnosti jako kterýkoli jiný, tudíž na něm nelze být úspěšný bez jeho znalosti. Táto práce se proto zabývá jeho analýzou a zjištěné poznatky budou výchozím bodem pro návrh a implementaci nové aplikace.

Abstract

The aim of the thesis is to summarize characteristics of successful and profitable applications on Android market and create a starting point for beginner developers. The market for mobile applications behaves like any other market, so people cannot be successful without its knowledge. That´s why this thesis is primarily focused on analyzing the market. Learned information will be later used for the design and implementation of new application.

Klíčová slova

Mobilní aplikace, Java, Jsoup, Android, analýza trhu, SWOT analýza

Keywords

Mobile application, Java, Jsoup, Android, market analysis, SWOT analysis

Bibliografická citace

BENEŠ, J. *Analýza trhu mobilních aplikací na platformě Android a návrh vlastní aplikace*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2016. 72 str. Vedoucí diplomové práce Ing. Petr Dydowicz, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva. (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 15. května 2016

……

Jaroslav Beneš

Obsah

Úvod

Mobilní telefony a přenosná zařízení jsou v dnešní době běžnými společníky u více jak poloviny celosvětové populace. Ukrývají v sobě výpočetní výkon, který je v mnohých případech srovnatelný se stolními počítači a zároveň svými rozměry nepřesahují velikost knihy a člověk je tak může mít neustále při sobě. Není proto divu, že spousta operací, kvůli kterým musel dříve uživatel použít stolní počítač nebo laptop, dnes může v klidu vykonat na svém telefonu nebo tabletu nezávisle na tom, v jakém prostředí se právě nachází. Mimo to přišli vývojáři se spoustou dalších způsobů jak efektivně tato zařízení využít a udělat z nich pro zákazníky nepostradatelné pomocníky.

S tímto rozvojem se také výrazně rozšířilo odvětví softwarového vývoje pro mobilní zařízení. Naštěstí se na těchto přístrojích nakonec ustálily pouze dva hlavní operační systémy, který lze na těchto zařízeních nalézt, což usnadňuje vývoj aplikací pro zařízení napříč různými výrobci. Jednoznačně největší výhoda u těchto platforem však spočívá v distribuci aplikací od vývojářů k zákazníkům. Ta probíhá prostřednictvím systému, který je integrován na všech zařízeních a téměř každý má možnost stát se vývojářem nebo zákazníkem. Díky tomu vzniká neustále spousta atraktivních aplikací a platforma zůstává pro zákazníky neustále zajímavá.

Mimo kvalitních aplikací však na trhu existuje spousty dalších, které úspěšné nejsou i přes výborný původní záměr vývojáře. Právě tento problém mě vedl ke zvolení tématu této práce. V rámci ní budu analyzovat současný trh a extrahovat pravidla, jejichž dodržení je nezbytné k vývoji úspěšné aplikace. Data získaná analýzou mohou sloužit jako odrazový můstek pro všechny začínající vývojáře a ušetřit jim tak spoustu času a pár neúspěšných projektů. Na základě zjištěných poznatků, poté ve čtvrté kapitole provedu návrh vlastní aplikace a popis následné implementace společně s hodnocením přínosů práce.

1 Vymezení problému a cíle práce

Tato práce vznikla s cílem analyzovat současný trh aplikací pro platformu Android a následně aplikovat získané poznatky při vývoji nové aplikace. Trh s aplikacemi dnes obsahuje nepřeberné množství aplikací různých kvalit, přičemž o jejich úspěšnosti rozhodují faktory, které nejsou vždy zřetelné a které se zdají mnoha začínajícím vývojářům zanedbatelné. To bývá často důvodem neúspěchu u aplikací s dobrou počáteční myšlenkou. V této práci se pokusím tyto faktory společně s obecně kvalitními postupy charakterizovat a vytvořit tak odrazový můstek pro všechny začínající vývojáře na trhu.

V první řadě použiji metody k provedení důkladné analýzy Android trhu, ve které je na trh nahlíženo z více pohledů s cílem získat velké množství informací. Bude například zkoumáno který způsob výnosu aplikace je vhodný ke kterým aplikacím a jak správně aplikaci prezentovat, či způsob komunikace s uživateli. Dále bude provedena analýza mé pozice vývojáře na trhu za pomoci metody SWOT. Získané informace budou zpracovány do stručného přehledu. Následný návrh aplikace bude vycházet právě z této analýzy, kde budou nejvýznamnější faktory předány ve formě požadavků na aplikaci. Program implementuji agilní metodou a výslednou aplikaci popíši na konci této práce.

2 Teoretická východiska

V této kapitole jsou teoreticky popsány základy, jejichž znalost je nezbytná k pochopení dalších částí práce. Nejprve je zde obecně popsán trh mobilních aplikací a zařízení. Po té se zaměřím více na platformu Android a vývoj aplikací a nakonec představení modelovacího jazyka UML a SWOT analýzy.

2.1 Vývoj mobilních zařízení

Poměr chytrých zařízení na trhu mobilních telefonů je neustále v růstu. Na trhu se objevují stále noví výrobci a i poptávka stoupá nejen díky dorůstající generaci. Přestože je skupina mobilních zařízení převážně tvořena tablety a chytrými telefony, lze o ní hovořit obecně, jelikož obě skupiny používají operační systémy a obě jsou i dlouhodobě v růstu.

Stále více operací, ke kterým dříve lidé potřebovali osobní počítač nebo notebook, lze nyní vykonat prostřednictvím mobilního zařízení, které v současnosti dosahují podobných výkonů. V roce 2013 dokonce překonal počet vlastníků chytrých mobilních zařízení počet vlastníků osobních počítačů.

I přes nepřeberné množství různých výrobců jsou však většinou zařízení vybavena operačním systémem od jednoho ze tří hlavních gigantů, přičemž nejmenší zastoupení z nich má platforma Windows mobile. Největší konkurenční boje probíhají mezi platformami iOS a Android a to nejen na poli samotných vývojářů ale i uživatelů. Přestože iOS se vyskytuje pouze na zařízeních od firmy Apple, jedná se o druhý nejrozšířenější operační systém hned po Androidu, který běží napříč zařízeními od různých výrobců. Nejvíce rozšířený je iOS především v Severní Americe.

2.2 Platforma Android

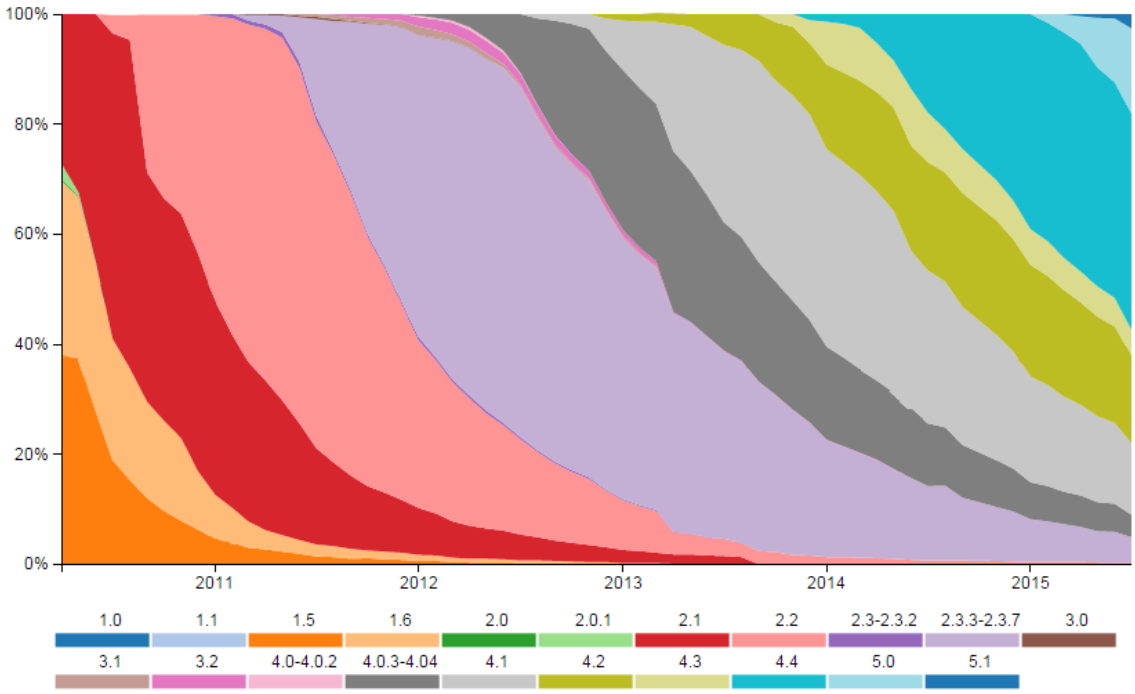
Android je operační systém založený pod Open-Source licencí společností Google. Open source znamená, že každý uživatel může systém využívat zadarmo při splnění jistých podmínek. Největší výhoda této licence však spočívá v dostupnosti zdrojových kódů, které si může kdokoli stáhnout k vlastní potřebě či komerčnímu užití. Základem Androidu je Linuxové jádro, které obstarává správu paměti, procesů a synchronizaci komunikace všech vnitřních komponent a senzorů a zajišťuje tak zabezpečení systému jako celku. Jednotlivé aplikace k jádru nepřistupují přímo, ale prostřednictvím aplikačního rozhraní Android API. (1)

Android je tedy progresivní operační systém primárně vyvíjen jako platforma převážně pro PDA, tablety a tzv. chytré telefony. Byl postaven od základu, který umožní vývojářům vytvářet působivé mobilní aplikace, jež mohou plně využívat všech vlastností, které telefon nabízí, jako např. základní funkce telefonu (obsluha telefonních hovorů, posílání textových zpráv (SMS), nebo využívání fotoaparátu). Takto vybudovaný systém umožňuje vývojářům vytvářet bohatší a soudržnější zážitky pro uživatele. Android je postaven na otevřeném jádře Linux a používá vlastní virtuální stroj, který byl navržen tak, aby optimalizoval paměť a hardwarové prostředky v mobilním prostředí. Tato platforma se bude dále vyvíjet, protože vývojářská komunita pracuje společně na vytváření inovativních mobilních aplikací. (2)

2.2.1 Historie

Za úplný počátek historie platformy Android lze považovat založení společnosti Android Inc. v roce 2003. To měli na svědomí Andy Rubin, Rich Miner, Nick Sears a Christ White, kteří začali vyvíjet aplikace pro mobilní zařízení se sídlem v Palo Altu v Kalifornii v USA. V roce 2005 společnost koupila firma Google a stala se tak její dceřinou firmou se zachováním původních zakladatelů na klíčových pozicích ve firmě. Pod vedením Andyho Rubina navíc vznikl nový tým, který začal vyvíjet zcela novou platformu pro mobilní zařízení, založenou právě na linuxovém jádru. Již v roce 2007 získali řadu patentů v dané oblasti a listopadu téhož roku došlo k představení nové platformy Android. (3)

Ve stejný den rovněž vznikla skupina OHA (Open Handset Alliance), která sdružovala přední společnosti z oboru mobilních zařízení. (3) V době založení šlo o více než 30 firem, jako jsou Intel, HTC, LG, Motorola, Samsung apod. a v dnešní době obsahuje více než 80 společností. Jedním z hlavních důvodů vzniku této skupiny bylo vytvořit otevřené standardy pro mobilní zařízení. V roce 2008 pak byla představena první oficiální verze operačního systému Android 1.0 na telefonu HTC G1 a s potěšením mohu říci, že jsem byl jistou chvíli jedním z jeho hrdých majitelů. Od té doby se systém rozšířil do zařízení většiny výrobců mobilních telefonů a v současnosti je nejnovější verzí 5.1, která se ovšem dostane pouze do nejmodernějších telefonů.

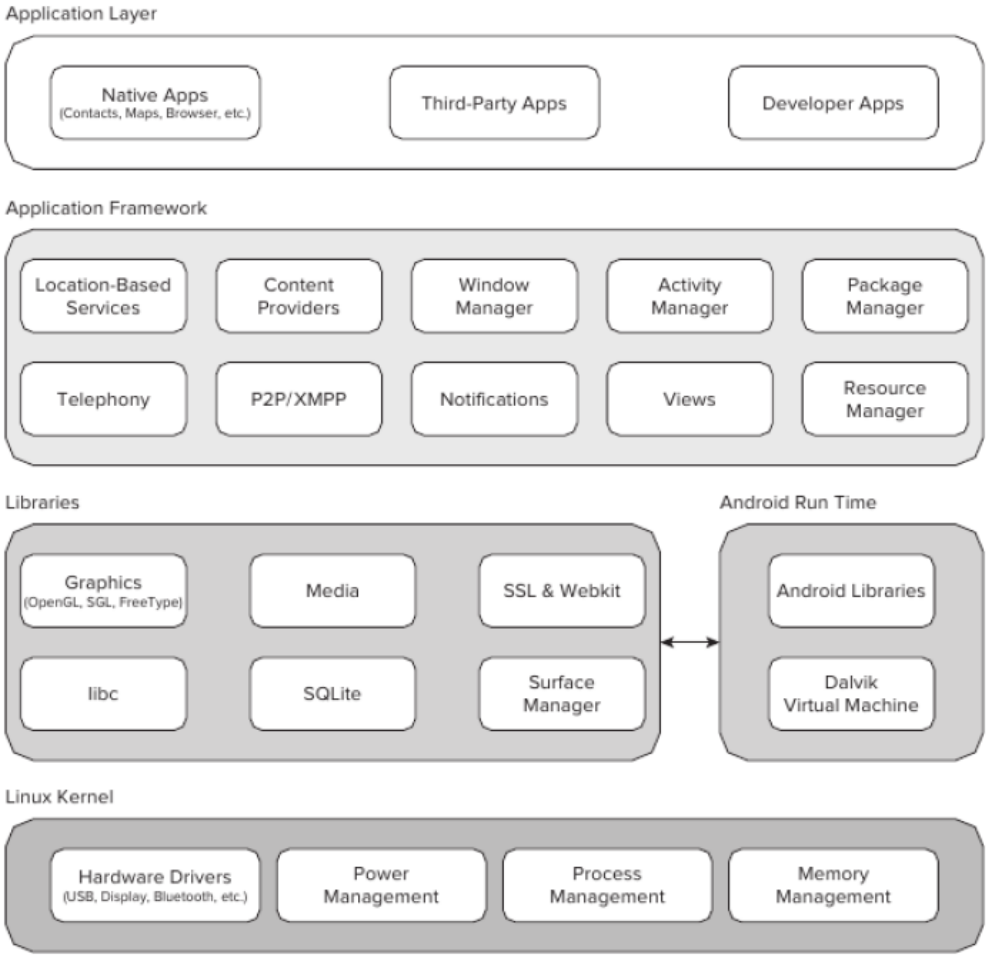


Graf 1 Vývoj fragmentace verzí operačního systému (4)

2.2.2 Architektura OS Android

Architektura operačního systému android je složena z pěti vrstev. Každá z nich vystupuje odděleně, ale v praxi dochází ke spolupráci jednotlivých částí, přičemž jejich komunikace probíhá skrze stanovená rozhraní. (3)

- Linuxové jádro - Jedná se o nejnižší vrstvu, která zprostředkovává komunikaci mezi hardwarovými prvky zařízení a softwarem na vyšších vrstvách.
- Knihovny - Jak jsem uvedl již dříve, aplikace přistupují k jádru skrze aplikační rozhraní, které je představováno právě touto vrstvou knihoven. Po prostudování všech dostupných knihoven má programátor přehled jaké funkce má k dispozici pro všechna zařízení Android.



Obrázek 1 Android architektura (5)

- Android Runtime – Jedná se o běhové prostředí pro spouštěné aplikace. Každá z aplikací je naprogramována v jazyce Java, následně přeložena do Java Byte kódu a nakonec do speciální mezikódu. (6) Ten je pak spouštěn na DVM (Dalvik Virtual Machine), který byl vyvinut speciálně pro tento účel a spouští veškeré aplikace v samostatných procesech.
- Aplikační Framework - Tato vrstva umožňuje přistupovat vývojářům ze svých aplikací ke stejnému rozhraní jako systémové aplikace a tudíž je pro ně jednou z nejdůležitějších.
- Aplikace – Nejvyšší vrstva architektury představuje aplikace, které uživatel stahuje z obchodu Google nebo ty, které jsou předem nainstalovány v zařízeních.

2.3 Životní cyklus vývoje SW

Tak jako ve všech odvětvích, i v informatice existují obecně platné postupy, které umožnují snadnější cestu k cíli. Postup pro vývoj počítačového programu se nazývá Životní cyklus vývoje softwaru (Software deleopment life cycle) a udává, v jakých krocích by měl vývoj probíhat a co se ve které etapě provádí. Jedná se tedy o logicky navazující činnosti, jejichž dodržení napomáhá k úspěšnému vytvoření produktu na trhu informačních technologií. (7)

2.3.1 Specifikace požadavků

Úkolem této části je zjištění a formulace požadavků na vyvíjený software. Požadavek si můžeme představit jako vlastnost, kterou by měla výsledná aplikace obsahovat. Požadavky jsou rozděleny na funkční a nefunkční, přičemž funkční požadavky stanovují nároky na aplikaci ve formě konkrétní funkcionality. Požadavky jsou většinou uváděny ve formě diagramu případů užití. Nefunkční požadavky potom udávají vlastnosti, které musí být splněny ve spojitosti s provozováním aplikace. Takovým požadavkem mohou být například kapacitní nároky nebo nároky na spolehlivost.

2.3.2 Analýza a návrh

Jak vyplývá z názvu analýza, výstupem tohoto kroku je rozbor stávajícího prostředí většinou se zaměřením stávajícího problému který má vyřešit naše aplikace. Nejdůležitějším krokem celého procesu je ale nepochybně návrh. Přestože bývá zejména u mladších vývojářů podceňován, jedná se o část, které by měl programátor věnovat nejvíce času. Kvalitní návrh dokáže velmi urychlit samotnou implementační část a naopak žádný či nekvalitní návrh implementaci výrazně prodloužit a v některých případech dokonce zcela znemožnit.

2.3.3 Implementace

Tato fáze vychází z návrhu aplikace a jejím úkolem je převést navržený model do vybraného programovacího jazyka. V této fázi tedy vzniká samotný produkt, který je cílem celého procesu. Před koncem tohoto kroku však začíná i následující fáze a testování a tyto činnosti mezi sebou vzájemně kooperují.

2.3.4 Testování

Tato fáze většinou začíná při dokončení určitého modulu z fáze předchozí a posílá zpětnou vazbu na funkčnost aplikace. Testování provádí jednak samotný vývojář, na kterém je odhalení kritických chyb, ale mnohdy i samotní uživatelé, kteří využijí aplikaci v plném provozu. V praxi se můžeme setkat i s mezikrokem v podobě najatého testovacího týmu, jehož úkolem je chyby odhalit. Testování tedy dokáže prokázat, že je aplikace chybná, ale nikdy ne opak.

2.4 Vývojové nástroje

Vývoj mobilních aplikacích nezbytně probíhá v odlišném prostředí, než pro které je ve výsledku určen. Programování většinou probíhá na počítači prostřednictvím vývojového prostředí a do telefonu je aplikace přenášena pouze pro účely testování. Právě tyto nástroje a praktiky, které je nezbytné při vývoji znát jsou uvedeny zde.

2.4.1 Android Studio

Android Studio je oficiální vývojové prostředí pro platformu Android. Poprvé bylo představeno světu v květnu roku 2013 na konferenci Google I/O avšak pouze ve vývojové verzi 0.1. První beta verze následovala v červnu následujícího roku a finální stabilní produkt byl vypuštěn v prosinci roku 2014 a je možné si jej bezplatně stáhnout pod licencí Apache 2.0. (8)

Program je určen čistě na vývoj aplikací pro tuto platformu a byl tvořen jako náhrada za původní Eclipse Android Development Tools, jakožto původní způsob vývoje. Jelikož je založen na IntelliJ IDEA, je určen pro systémy Windows, Linux i Mac OS X. IntelliJ IDEA je komerční vývojové prostředí pro programování v jazycích Java, Groovy a dalších. (8) Díky tomu studio obsahuje prvky, které jsou pro již zaběhlá vývojová prostředí standardem, jako jsou navigace v kódu, refaktoring, našeptávání, debugování či analýza.

2.4.2 Java

Vývoj aplikací pro platformu Android probíhá za pomoci jazyku Java. Jedná se nejmodernější podobu programovacího jazyka, který v sobě spojuje výhodu kompilovaného a interpretovaného jazyka. Java je jazyk třetí generace a je objektově

orientovaný, což znamená, že prvky z reálného světa jsou v implementovaném kódu abstrahovány do entit (objektů).

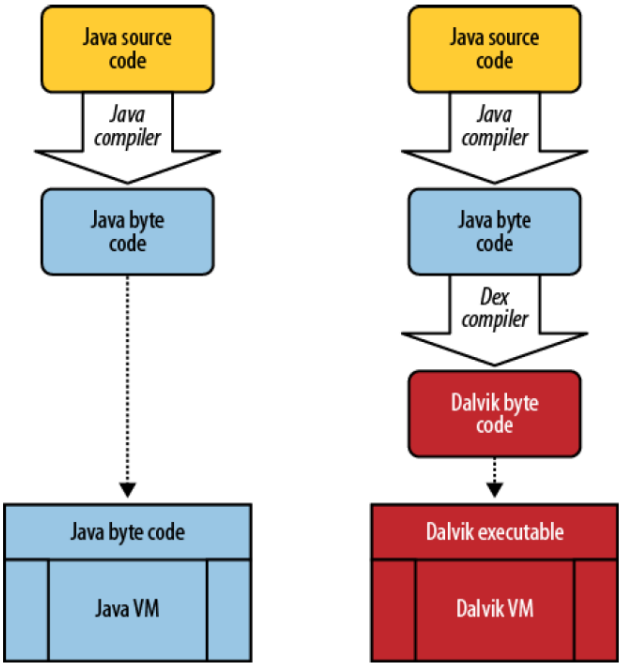
Ideovým předchůdcem je jazyk C++ a v současné době je největším konkurentem C#. Velkou výhodou Javy je přenositelnost napsaných programů napříč různými platformami díky běhu na virtuálním stroji a velice rozšířená komunita vývojářů s čímž souvisí i nepřeberné množství knihoven s volnou licencí.

2.4.3 Dalvik

Dalvik je účelně vyvinutý virtuální stroj speciálně pro Android v Google pod týmovým vedením Dana Bornsteina. Původní Java Byte kód, který byl známý díky své použitelnosti na všech platformách se jeho týmu nezdál dostatečný a rozhodli se vytvořit nový, zaměřený čistě na Android, neboť věřili, že takto dosáhnou lepších výsledků. Dalším velmi důležitým faktorem při výběru VM byly licence. Zatímco nástroje i knihovny na začátku vývoje byly zadarmo, Java VM nikoli. Vytvoření Dalvik VM jako open source velmi přispělo k rozšíření celého Android jazyka do nejrůznějších zařízení u většiny výrobců.

2.4.4 Android vs Java

Pokud píšeme Java aplikaci, zdrojový soubor s kódem se přeloží do Java Byte kódu, který je následně implementován virtuálním strojem Java VM (virtual machine). Jak jsem již zmínil u architektury, překlad zde probíhá odlišně, přestože se programuje ve stejném jazyku. Zdrojový kód přeložíme rovněž do Java Byte kódu, který opět znovu zkompilujeme Dalvikovým kompilátorem a dostaneme Dalvik Byte kód. Ten je následně obdobně jako obyčejná Java aplikace spouštěn za pomocí virtuálního stroje a u Android aplikací jde konkrétně o Dalvik VM.



Obrázek 2 Java vs Dalvik (3)

Na první pohled vyvstává otázka, proč zdrojový kód nepřekládat bezprostředně do Dalvik Byte kódu, avšak stávající způsob má dobré důvody. Hlavním z nich je stabilita Java Byte kódu. U jazyka Java jsou časté aktualizace a probíhají u něj časté změny. Java Byte se ve větší míře nemění a proto se Bornstein rozhodl při kompilaci do Dalvik Byte kódu vycházet právě z něj. Navíc tak lze teoreticky aplikace vyvíjet pod libovolným jazykem, který jde přeložit do Java Byte kód, jako je například jazyk Python nebo Ruby. (3)

2.4.5 Stavební bloky aplikace

Stavební bloky jsou prvky, za pomoci kterých vývojář skládá aplikaci.

Aktivita

Aktivita reprezentuje jednu obrazovku uživatelského rozhraní. (9) Uvažujme například aplikaci emailového klienta - jedna aktivita by zobrazovala seznam přijatých emailů, další aktivita by sloužila k vytvoření nového emailu a další například pro čtení daného emailu. Přestože aktivity společně tvoří komplexní uživatelské prostředí, nejsou na sobě vzájemně závislé. Aktivity mohou být navíc spuštěny i jakoukoli jinou aplikací, tedy pokud programátor tuto možnost povolil. Kupříkladu, aplikace Fotoaparát může po pořízení snímku vyvolat emailovou aplikaci za účelem odeslání pořízeného obrázku emailem.

Fragment

Fragment reprezentuje chování určité části uživatelského rozhraní aktivity. V rámci jedné aktivity lze zobrazit několik fragmentů a vytvořit tak uživatelské rozhraní s několika panely, zároveň lze jeden fragment použít ve více aktivitách. Fragment si lze představit jako modulární sekci aktivity, která má svůj vlastní životní cyklus, dokáže zpracovávat své vlastní vstupní události a může být přidána či odebrána za běhu aktivity. Fragment musí být vždy přiřazen k aktivitě, jeho životní cyklus úzce závisí na životním cyklu hostované aktivity. Pokud je například aktivita pozastavena (je volána metoda onPause()), jsou pozastaveny i všechny její fragmenty. Pokud je však aktivita v běžícím stavu, s každým jejím fragmentem lze manipulovat nezávisle.

Služby

Služba je komponenta běžící v pozadí aplikace, sloužící pro zpracování náročnějších, dlouho trvajících operací (long runtime), nebo zpracování úkolů vzdálených procesů. Služba nemá žádné uživatelské rozhraní. (9)

Základním příkladem může být služba pro přehrávání hudby na pozadí, zatímco uživatel používá jinou aplikaci. Typický způsob použití je také získávání dat z internetu, bez toho aniž by bylo zabráněno uživateli v interakci s právě spuštěnou aktivitou. Služba může být spuštěna komponentou, jako je například aktivita. Služba pak bude provádět svoji činnost, pro kterou byla navržena nebo může být s aktivitou svázána a provádět určitou interakci.

Broadcast receiver

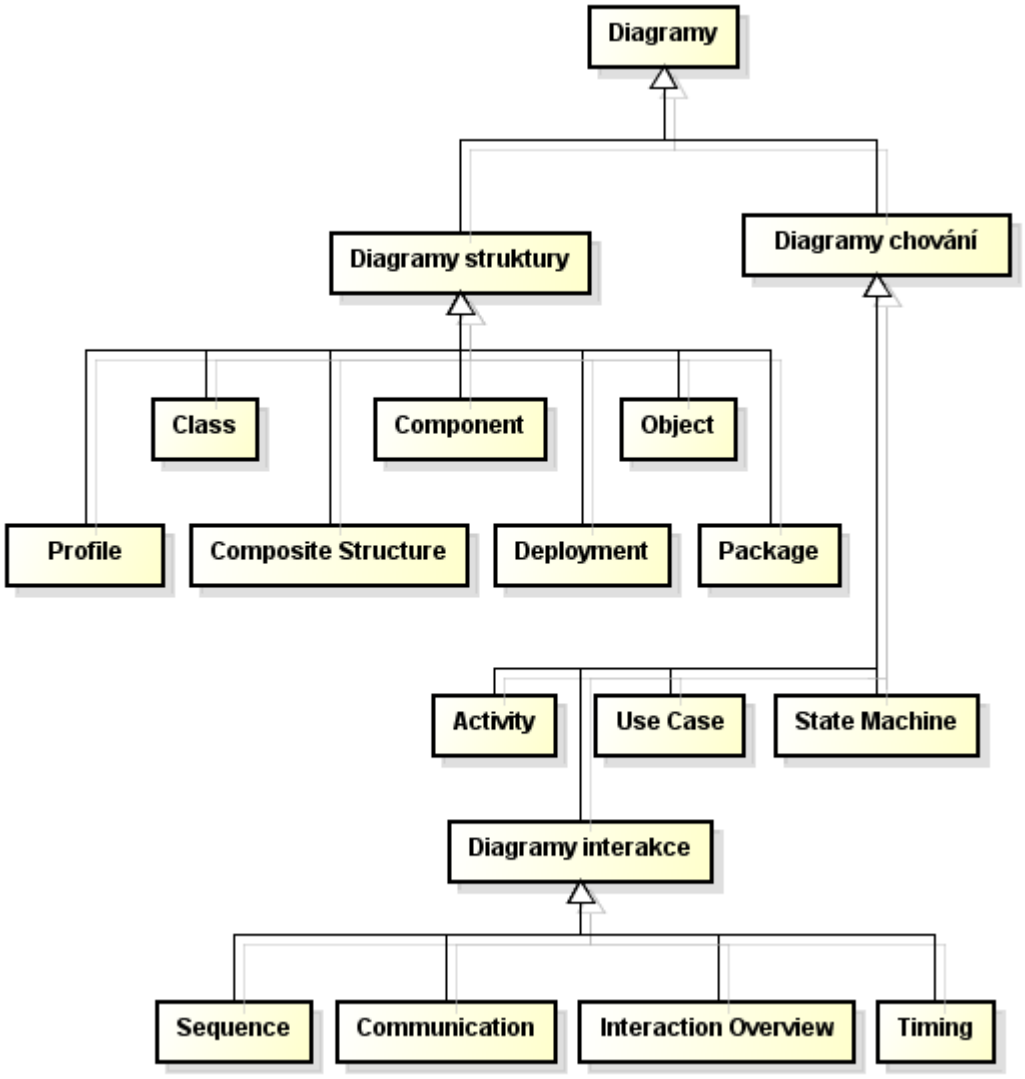
Broadcast receiver je komponenta, která odpovídá na systémové broadcasty. Tyto broadcasty mohou být šířeny mnoha komponentami, jejich velké množství generuje samotný systém – například broadcast oznamující vypnutí obrazovky zařízení nebo slabou baterii.

Aplikace mohou vytvářet svoje vlastní broadcasty, například oznámení o úspěšném stažení požadovaných dat a jejich dostupnosti k dalšímu zpracování. Ačkoli Broadcast receiver nedisponuje uživatelským rozhraním, může vytvořit notifikaci v oznamovací oblasti a informovat tak uživatele o přechodu do určitého stavu. Ve většině případů je Broadcast receiver použit pouze na iniciování další akce, jako je například spuštění služby.

2.5 UML

Jazyk UML (Unified Modeling Language) je soubor grafických předpisů, které lze využít při vývoji softwaru. Je hojně využíván ve fázi analýzy a návrhu. Jelikož je i zavedeným standardem, je nezbytné aby se v něm vývojář perfektně orientoval. Celkově tento jazyk vede k usnadnění návrhu a následného vývoje především u objektové orientovaných softwarových produktů. Můžeme si jej tedy představit jako vizuální model programu.

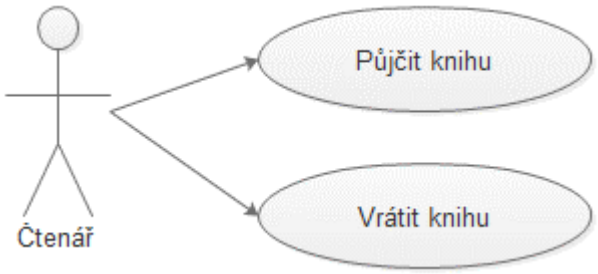
Tento jazyk vznikl s cílem sjednotit stávající metody modelování při zachování maximální vypovídající hodnoty a srozumitelnosti. Jedná se o otevřený standard, který je založen na potřebách komunity uživatelů a podporuje celou řadu nástrojů. Jeho podoba prochází neustále úpravami, i když jeho počátek je datován ke konci 80. let. (10)



Obrázek 3 Rozdělení jazyka UML (11)

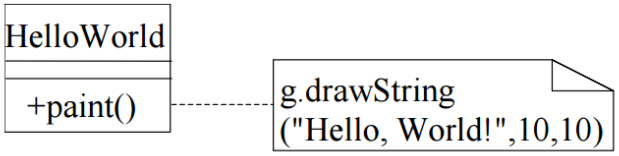
V současné době je UML jazyk složen ze 14 ti různých diagramů které se liší nejen účelem použití ale i syntaxí. Diagramy struktury popisují strukturu systému, tedy to z čeho je celek složen. Diagramy chování popisuje, jakým způsobem se systém chovám, avšak diagramy chování obsahují ještě jednu podskupinu, která popisuje interakci mezi jednotlivými částmi systému. Každý diagram má svůj speciální účel, přičemž nemá smysl dělat všechny ale pouze ty, které mají největší přidanou hodnotu k našemu účelu. (12)

V této práci jsou použity celkem tři modely. Prvním z nich bude diagram případů užití (use case diagram), který reprezentuje funkčnost systému, kterou musí výsledný produkt splňovat. Základním prvkem tohoto diagramu jsou aktéři, kteří se systémem budou přicházet do styku, konkrétní případy použití, a přiřazení jednotlivých případů k aktérům. Nemělo by docházet ke stavu kdy je jeden případ přiřazen více lidem. Vznik takového stavu se obvykle řeší zobecněním, což v podstatě znamená podědění funkcionality od jiné role.



Obrázek 4 Diagram případu užití

Dalším z rodiny UML je diagram tříd. Tento diagram musí obsahovat veškeré třídy, které má mít i aplikace, včetně veškerých atributů a metod. Měl by být koncipován tak, aby po převedení do programového kódu nemuselo být již nic doplněno. Nemusí však obsahovat implementaci, které může být doplněna následně a teprve po té implementována. Ukázku takového procesu lze vidět na další straně.




```
import java.awt.Graphics;
class HelloWord extends java.applet.Applet {
   public void paint (Graphics g) {
      g.drawstring ("Hello, World!", 10, 10);
   }
}
```

Obrázek 5 Ukázka diagramu tříd (13)

Jelikož je ale diagram tříd pro rozsáhlejší aplikace nepřehledný, dalším použitým modelem v této práci bude Doménový (objektový) model. Ten se používá v prvotní fázi vývoje softwaru společně s diagramem případu užití. Model umožňuje navrhovat aplikaci za pomocí tříd (entity), což jsou základní stavební kameny objektově orientovaného programování. Jedná se tedy o zjednodušenou formu diagramu tříd, přičemž třídy jsou v doménovém modelu výrazně zjednodušeny. (14)

Třídy například neobsahují metody, ale pouze důležité atributy. Přestože je možné názvy tříd a atributů psát s diakritikou, názvy v modelu musí korespondovat se jmény objektů v implementovaném kódu a model je tak nutné sestavovat ve stejném jazyce. Model je zároveň nezávislý na platformě díky tomu, že atributy nemají datové typy a lze jej tak realizovat v libovolném objektově orientovaném jazyce.

Při tvorbě doménového modelu se standardně vychází ze zadání a jeho cílem je identifikovat klíčové entity a vztahy mezi nimi. Třída je v modelu reprezentována obdélníkem, kde je v horní části umístěn její název a pod ním příslušné atributy. Třídy jsou mezi sebou propojeny za pomoci vztahů, které mohou nabývat různých forem. (14)

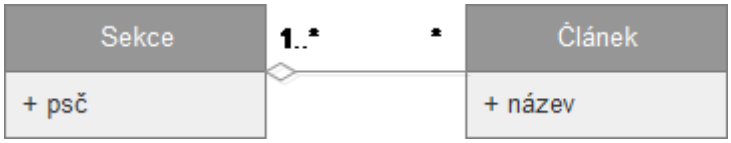
Asociace

Jedná se o základní vztah mezi dvěma objekty, který je značen jednoduchou plnou čárou. Spojení touto vazbou udává existenci odkazu v entitě na jinou entitu. Která entita uchovává odkaz je znázorněn jednoduchou šipkou na konci čáry. Odkaz je uložen v objektu, na který šipka nemíří. V případě, že šipka chybí, jsou odkazy u obou entit.

Obrázek 6 Ukázka vztahu typu asociace

Agregace

Agregace umožňuje modelovat vztah typu část – celek. Značí se rovněž jako jednoduchá plná čára, s rozdílem ukončení na jedné straně prázdným kosočtvercem. Strana, kde je kosočtverec umístěn, je zvolena u té entity, která reprezentuje celek. U vztahu typu agregace také platí, že jednotlivé třídy mohou existovat nezávisle na sobě. V implementaci to může například značit třídu, která obsahuje list objektů.



Obrázek 7 Ukázka vztahu typu agregace

Kompozice

Kompozice je jistá obdoba agregace s tím rozdílem, že reprezentuje silnější vztah. Opět slouží k modelování vztahu mezi částí a celkem, avšak v tomto případě nemůže část bez celku existovat. V případě zániku celku zanikají i jeho součásti. Na rozdíl od agregace se značí plným kosočtvercem.

Obrázek 8 Ukázka vztahu typu multiplicita

Čísla uvedená nad vazbami u jednotlivých entit udávají multiplicitu, která se pro zpřesnění modelu může uvádět u vazeb typu asociace, agregace a kompozice.

Multiplicita

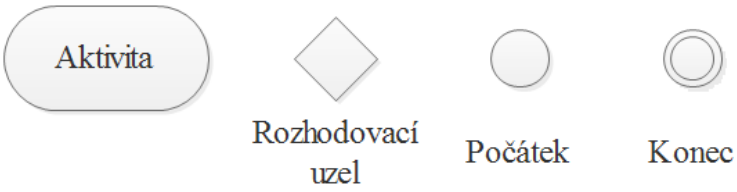
Multiplicita (násobnost) udává, v jakém počtu do vztahu mohou vstupovat jednotlivé objekty. Multiplicita může být definována číslem, hvězdičkou nebo intervalem.

- číslo (1) - označuje konkrétní hodnotu

- hvězdička (*) - označuje libovolný počet včetně nuly, může být nahrazena písmenem N
- interval (1..*) - zde je pomocí dvou teček označen interval, jenž musí být ohraničen jedním z výše uvedených symbolů

Multiplicita se nemusí vždy uvádět. Pokud chybí, předpokládá se konkrétní hodnota jedna.

Posledním použitým modelem v práci je diagram aktivit, který se používá pro popis procedurální logiky, pracovních postupů nebo posloupnost činností. Tento diagram popisuje procesy jako aktivity, které se skládají z propojených uzlů. (15) Diagram je podobný diagramu stavů, s tím rozdílem, že se v stavových uzlech zobrazují činnosti (aktivit), které se vykonávají. Mimo uzly aktivit existují speciální uzly pro počátek a konec diagramu a rozhodovací/spojovací uzly. Na rozhodovacích uzlech je dle stanovené podmínky rozhodnuto, jakým směrem se bude tok dále ubírat. Grafickou podobu jednotlivých uzlů lze vidět na následujícím obrázku.



Obrázek 9 Ukázka grafického značení uzlů

2.6 SWOT

Jedná se o nástroj určený pro systematickou analýzu se zaměřením na charakteristiku klíčových faktorů ovlivňujících strategické postavení podniku. Hlavní podstatou je identifikace silných a slabých stránek podniku a porovnává je s hlavními vlivy z okolí podniku. Od těchto vlastností je odvozen také název (16):

- Strenghts – silné stránky
- Weaknesses – slabé stránky
- Opportunities – příležitosti
- Threats – hrozby

SWOT analýza je rozdělena na čtyři oblasti. Dvě jsou slabé a silné stránky, které jsou charakteristiky vnitřní situace podniku a dvě charakteristiky vnějšího okolí, příležitosti a rizika. Velkou nevýhodou tohoto přístupu je nutnost odlišit silné stránky od slabých a příležitosti od hrozeb. V praxi je bohužel často těžké odlišit, zda se jedná o příležitost či hrozbu a zdali je sledovaná vlastnost silnou či slabou stránkou podniku. Uplatnění této analýzy je vedeno především myšlenkou rozvíjet silná stránky a příležitosti a naopak potlačovat hrozby a slabé stránky (17).

Obrázek 10 SWOT tabulka (18)

Silné a slabé stránky se posuzují vzhledem ke konkurenci. Cílem je maximalizovat silné stránky a pokud možno co nejvíce eliminovat slabé stránky. Analýzou vnějšího prostředí se podnik snaží identifikovat rizika, která ohrožují rozvoj firmy a určit příležitosti, které by k rozvoji mohly napomoci (16). Hrozby a příležitosti nemůže podnik příliš ovlivnit. Snaží se pouze maximálně využít dostupné příležitosti a co nejvíce minimalizovat důsledky případných hrozeb.

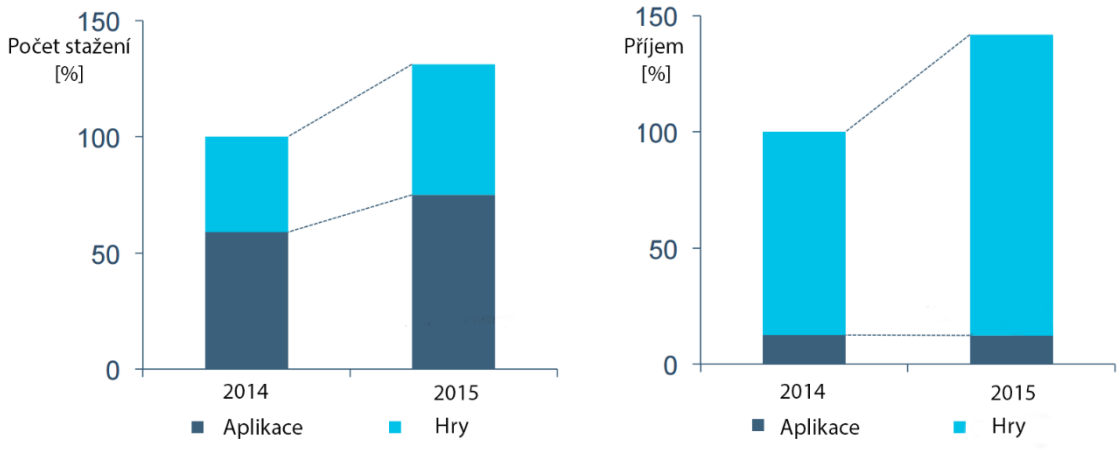
3 Analýza problému a současné situace

Tato kapitola je zaměřená na analýzu současného stavu trhu mobilních aplikací na platformě Android, Google Play. Je zde popsáno, jak se trh vyvíjí, jaké jsou nejužívanější a nejstahovanější aplikace, podrobnosti o době životnosti aplikací a způsoby jejich prezentace na trhu. Dále jsou uvedeny mechanismy, díky kterým může vývojář ze svých aplikací profitovat a způsoby odezvy, které napomáhají reagovat na podněty uživatelů a přispívat tak k jejich spokojenosti. Mimo jiné zde naleznete principy, které vznikly z analýzy současných úspěšných aplikací a jsou nápomocny při zvyšování zisku. Seznámení vývojáře s těmito principy na začátku tvorby napomáhá ke zvyšování úspěšnosti.

3.1 Expanze na trhu

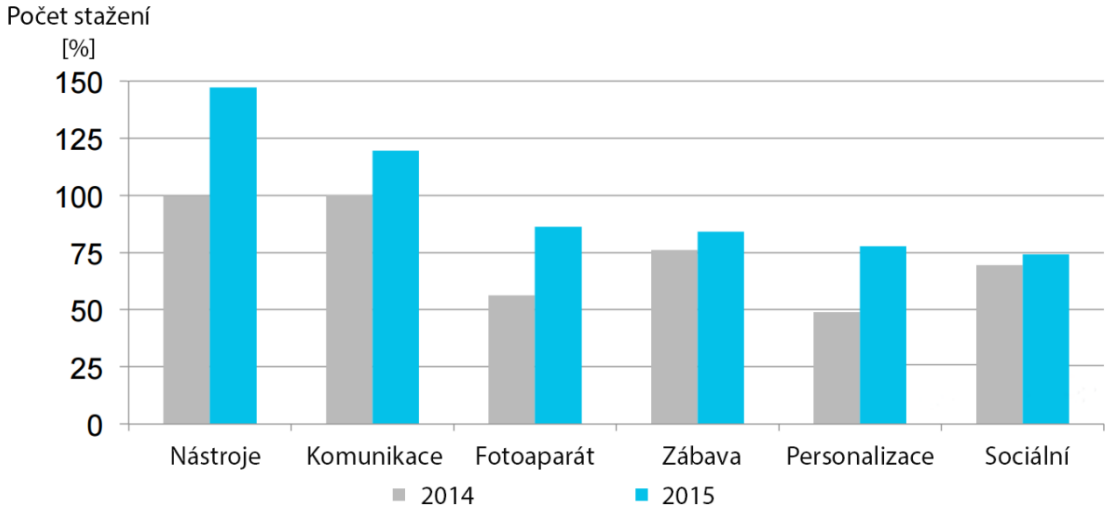
Trh s Android aplikacemi se neustále rozšiřuje a počtem nabízených aplikací již dávno předehnal své konkurenty. Důkazem tohoto růstu je i meziroční rozdíl v počtu stažených aplikací a příjmu z nich. Počet stažených aplikací mezi rokem 2014 a 2015 se navýšil o 30 %. Velký podíl na tomto čísle měly země, ve kterých se trh teprve rozvíjí, jako například Indie či Brazílie. Příjem z aplikací vzrostl dokonce o 40 % s největším růstem tržeb na již stabilních trzích, zejména v USA, Jižní Koreji a Japonsku (19).

Z hlediska kategorie aplikace je růst počtu stažených aplikací vyvážený, zatímco růst zisku byl z majoritní většiny z herních aplikací. Přičemž se udává, že příjem z této kategorie každým rokem stoupá přibližně o 50 %.



Graf 2 Přehled růstu trhu (19)

Pokud bychom se však chtěli zaměřit pouze na aplikace vyjímaje hry, nárůst v počtu stažení byl nejvíce markantní u aplikací v kategorii „Nástrojů“ a „Personalizace“, které dokáží využít plný potenciál chytrého telefonu. Příkladem z kategorie „Personalizace“ jsou takzvané „launchery“, které dokáží optimalizovat chod telefonu a nabízejí širší paletu přizpůsobení uživatelského prostředí.



Graf 3 Nárůst počtu stažení dle kategorií (19)

3.2 Nejpopulárnější aplikace

Podobně jako v Google play jsem zde rozdělil aplikace na herní a ostatní. Následující informace byly sestaveny z celosvětového trhu. V jednotlivých zemí statistiky vycházely v drtivé většině podobně, s výjimkou Japonska, kde první příčky obsadily aplikace s místní tématikou.

V kategorii her jsou mezi deseti nejvíce stahovanými hrami tituly, za kterými stojí velké skupiny vývojářů, jako například Angry Birds od společnosti Rovio, Clash of Clans od Supercell nebo Fruit Ninja od Halfbrick Studios. Není velkým překvapením, že 10 nejvíce vydělávajících her je s 80 % shodný s výše uvedeným i přes různé způsoby zpoplatnění aplikace, přičemž nejúspěšnější aplikace spatřily světlo světa poprvé již před rokem 2013 a stále se řadí mezi nejziskovější. Všechny však mají za sebou řadu vylepšení, na kterých neustále pracují týmy vývojářů a původní podobu aplikace by tak poznal už jen málokdo.

U aplikací je tomu ale jinak, nejstahovanějšími aplikacemi jsou především aplikace sociálních sítí, jako jsou Facebook, Instagram, Twitter či Youtube. V nejziskovějších aplikacích však ani jedna z těchto aplikací není. Navíc jsou zde aplikace nejrůznějšího využití, například Endomondo (monitorování sportovních aktivit), Navigon (GPS navigace) nebo Tinder (seznamovací aplikace). Nejziskovějším druhem aplikací jsou však jednoznačně hry.

3.3 Příjem z aplikace

Jedná se beze sporu o jeden z nejdůležitějších aspektů při tvorbě aplikace. V současné době existuje více způsobů, jak z vlastní aplikace získat peníze. Jednotlivé způsoby se mezi sebou dají různě kombinovat.

Placená aplikace (flat fee, paid downloads)

Jedná se o hojně využívaný způsob. Princip spočívá v tom, že uživatel zaplatí stanovenou částku za aplikaci ještě před jejím stažením. Po té má k dispozici kompletní verzi, která ho v budoucnosti nebude stát již ani korunu. Zisk tedy plyne pouze ze stažení, nikoliv však z používání aplikace. Jedná se o obdobný model, jako u licencí na stolní a přenosné počítače, kde platíte většinou za pořízení programu (Microsoft Office). Při stanovení ceny aplikace není vývojář ničím ovlivněn, avšak musí dbát na zacílení aplikace a na její účel. Pro nové vývojáře se doporučuje začínat s cenou kolem dvou dolarů.

Výhody

Design aplikace není ovlivněn přidanou reklamou a vývojář má peníze k dispozici ihned po stažení zákazníkem.

Nevýhody

Většina zákazníků v České republice není ochotna za aplikace platit, doporučuje se použít pro aplikace určené i pro zahraniční spotřebitele. Větších příjmů při používání tohoto modelu dosahují vývojáři na trhu s aplikacemi pro Apple (v minulosti pro Blackberry). Uživatel si aplikaci nemůže předem vyzkoušet a musí se tak spoléhat na hodnocení ostatních uživatelů. Jedná se o jednorázový příjem.

Reklama (In-app advertise)

V tomto případě si uživatel stáhne rovněž plnou verzi aplikace, avšak nyní zcela zadarmo. Do aplikace je však automaticky vkládána reklama, ze které je následně placen vývojář. Na rozdíl od předchozí metody, příjem je podmíněn užíváním aplikace, což znamená, že by tento způsob měl být využit u aplikací, které uživatelé pravidelně užívají. Tento princip by se dal přirovnat ke komerční televizi: můžeme zdarma sledovat veškerý obsah, ta však má příjem z reklam, které jsou vkládány mezi námi sledovaný obsah. Při tvorbě aplikace zacílené na obyvatele naší země, bych po zkušenosti mého kamaráda vývojáře doporučil právě tento model.

Výhody

Zisk je závislý na užívání aplikace a může tak být dlouhotrvající (20). Neplacené aplikace přitahují více potencionálních uživatelů než placené.

Nevýhody

Na rozdíl od placené aplikace nelze nijak ovlivnit cenu za vloženou reklamu a je tak odkázán pouze na užívání spotřebitelů. Může narušovat design aplikace a v některých případech lze obejít odpojením zařízení od internetového připojení.

Nákupy v aplikaci (In-app purchase)

Tento způsob je dnes ve velké oblibě především u herních aplikací. Uživatel si zdarma stáhne aplikaci, kterou může bezplatně používat. V aplikaci je však spousta neaktivních prvků, které si může dokoupit. Tyto prvky mají sloužit k usnadnění aplikace, získání výhody před protivníkem či jinému vylepšení aplikace. Každý uživatel si tak může vybrat, zdali chce v rámci aplikace investovat peníze. Tento způsob je podobný jako způsob dopravy po velkoměstě. Můžeme se kamkoli dostat pěšky (zdarma), nebo si zaplatit za některou formu dopravy, která nás na místo dostane (MHD, taxi, limuzína….).

Výhody

Neplacené aplikace přitahují více potencionálních uživatelů než placené. Design aplikace není ovlivněn přidanou reklamou a placené prvky lze do libovolně přidávat a zvýšit tak zisk.

Nevýhody

V krajním případě může dojít k nízkým ziskům i při vysokém počtu stažení a častému používání uživatelů. Je potřeba řešit problém přenositelnosti účtů, pokud uživatel využívá stejnou aplikaci na více zařízeních.

Freemium

Jak vypovídá název, jedná se o model založený na spojení aplikace zdarma (Free) a nadstandartní funkcionality či rozšířeného obsahu za příplatek (Premium). Uživatel si tak může zdarma stáhnout aplikaci, kde by měl být funkční alespoň základní princip. Pokud se uživatel rozhodne o přechodu k Premium, budou mu odemčeny všechny výhody. Freemium modely se dají rozdělit do několika kategorií (21):

- **Omezení funkcí** - základní funkce jsou zdarma a uživatel platí za rozšířené funkce či za integraci
- **Časové omezení** - všechny funkce jsou zdarma, ale na omezenou dobu
- **Kapacitní omezení** - základní kapacita (například objem dat) je zdarma, uživatel platí za rozšířenou nebo neomezenou kapacitu (datová úložiště)
- **Zákaznické omezení** - zdarma je pro specifickou skupinu zákazníků (studenti)

Výhody

Uživatel si může nejdřív vyzkoušet bezplatnou verzi a posléze se rozhodnout pro plnou verzi, což napomáhá k většímu rozšíření aplikace. Obdobně jako u předchozího modelu není design ovlivněn reklamou.

Nevýhody

Uživatel si nemůže vybrat pouze požadované aplikace, ale musí zaplatit za celý Premium účet (22). Velká většina uživatelů používá pouze základní verzi, což snižuje příjmy.

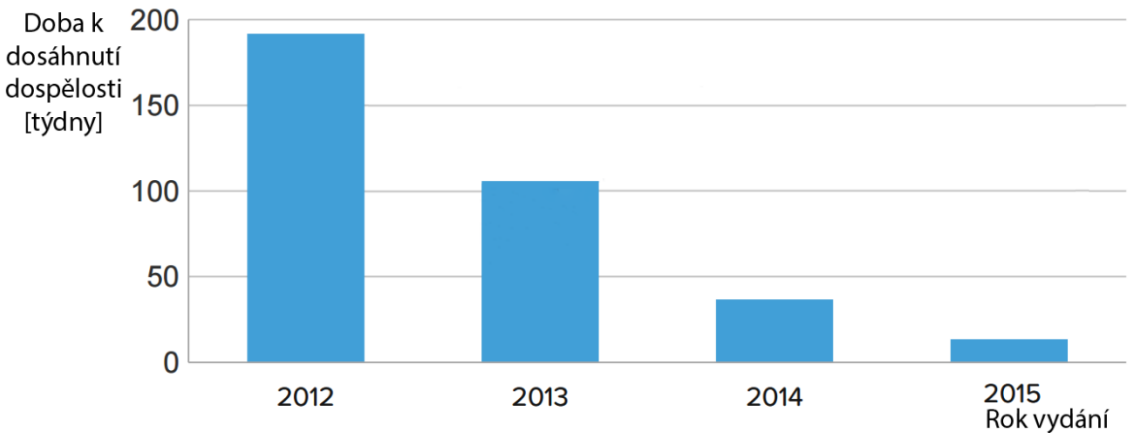
V praxi však lze najít různou kombinací přecházejících modelů. Dost často lze narazit na placenou a neplacenou verzi stejné aplikace. Jedná se vlastně o obdobu modelu Freemium, kterou vývojáři hojně využívají. Dále můžeme narazit na kombinaci reklamy a nákupů v aplikacích a reklamy a Freemium programu.

Dalším důležitým modelem je takzvaná zakázková práce (contract work), avšak tento typ zde nebudu probírat z důvodu zaměření na Android market. I když výsledek programu na zakázku může být následně umístěn na trhu aplikací, jeho princip spočívá v odpovědnosti vývojáře za pouhou implementaci, za což dostane předem stanovený plat. V různých publikacích se můžeme setkat s jiným rozdělením či názvoslovím.

3.4 Životnost aplikace her

Většina vývojářů by nejraději vytvořila takovou aplikaci, která by jim zajistila příjem na několik let. Bohužel, pouze minimum aplikací má tak dlouho dobu životnosti. Jaké jsou typy aplikací, u kterých vydrží uživatelé dlouho a které je naopak rychle omrzí, je popsáno v následujících odstavcích.

Pro jednoduchost si rozdělíme životnost aplikace do tří fází: raná, rostoucí a dospělá fáze. Raná fáze nastává ihned po uvedení na trh a její křivka kumulativního používání (cumulative adoption) pozvolna roste. Nejstrmější je ve fázi růstu a ve fázi dospělosti křivka pozvolna zpomaluje. Cílem každého vývojáře je udržet fázi růstu pokud možno po co nejdelší dobu, což se paradoxně lépe daří věkově starším aplikacím, jak můžeme vidět na následujícím grafu:

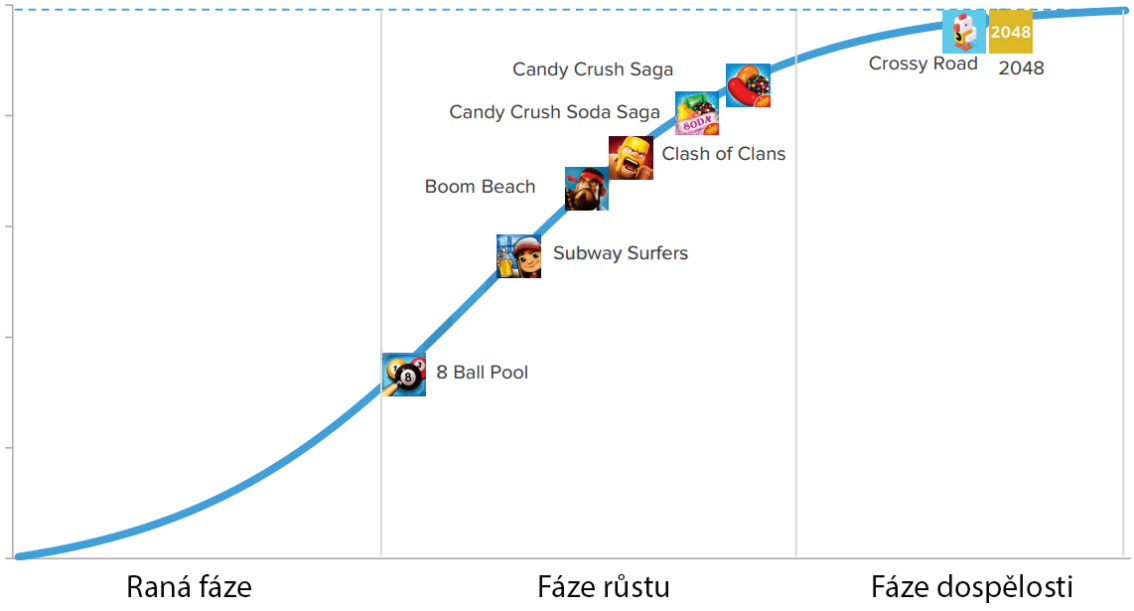


Graf 4 Průměrná doba k dospělosti aplikací dle roku vydání (23)

Vidíme průměrnou dobu aplikace k dosažení fázi dospělosti, která se každým rokem rapidně snižuje. To může být způsobeno jednak zvýšenou konkurencí na trhu, zvýšenou reklamní činností, ale i jinými faktory. Pro vývojáře to znamená především nutnost rozšířit své portfolio aplikací, a nabídnout tak uživatelům novou aplikace, ke které může přejít, až ho původní omrzí. Aplikace by měl uvolňovat postupně tak, aby získal

co nejvíce migrujících uživatelů. Zde vzniká stav, při kterém si konkurují vlastní aplikace, což u kvalitních aplikací nemusí být zas až tak velký problém.

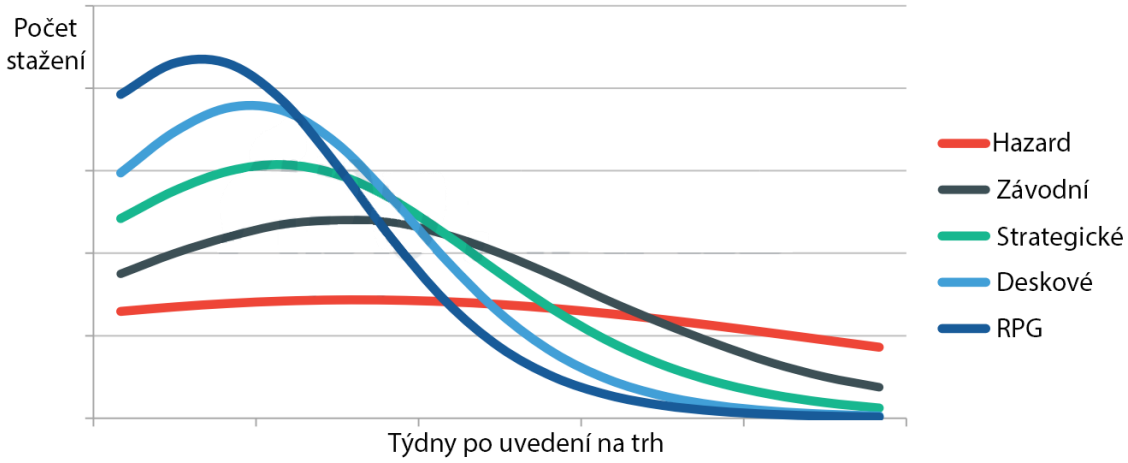
Například aplikace Boom Beach a Candy Crush Soda Saga jsou na téměř stejném bodu životního cyklu aplikace jako jejich předchůdci Clash of Clans a Candy Crush Saqa i přes to, že se jedná o vzájemná pokračování. To může být způsobeno jednak širokou komunitou uživatelů, ale i velkým množstvím bonusových prvků ve hře, což výrazně prodlužuje její životnost. Je zde rovněž vidět různá rychlost postupu jednotlivých titulů. Například Subway Surfers byla vydána dříve než Clash of Clans a Candy Crush Saga, avšak oba tituly ji svým životním cyklem předběhly.



Graf 5 Životní cyklus vybraných aplikací (23)

Stejně tak jako Crossy Road a 2048, které se nacházejí ve finální fázi životního cyklu, přestože jsou jedny z nejmladších aplikací. Hlavním důvodem je povaha aplikace. Jedná se o velmi jednotvárnou aplikaci s velmi jednoduchým ovládáním, která může být z počátku návyková, avšak rychle člověka omrzí. Pro takovéto aplikace je charakteristický životní cyklus s nejvyšším počtem stažení krátce po jejím vydání a rychlý přechod do finální fáze dospělosti. Takovéto aplikace však nevyžadují tolik správy po ukončení vývoje jako jiné tituly.

Nyní se podíváme na délku životního cyklu v závislosti na ostatních žánrech. Pokud vynecháme výše uvedené aplikace, nejkratším životním cyklem jsou RPG tituly.



Graf 6 Průměrný počet stažení od doby vydání (23)

Je to paradox, jelikož na platformě PC zaujímají přesně opačnou pozici. Naopak nejdelší životní cyklus vykazují aplikace z žánru závodních a hazardních her. V případě vývoje více aplikací by měl vývojář vzít v úvahu právě tyto žánry, neboť zajistí stabilitu jeho portfolia.

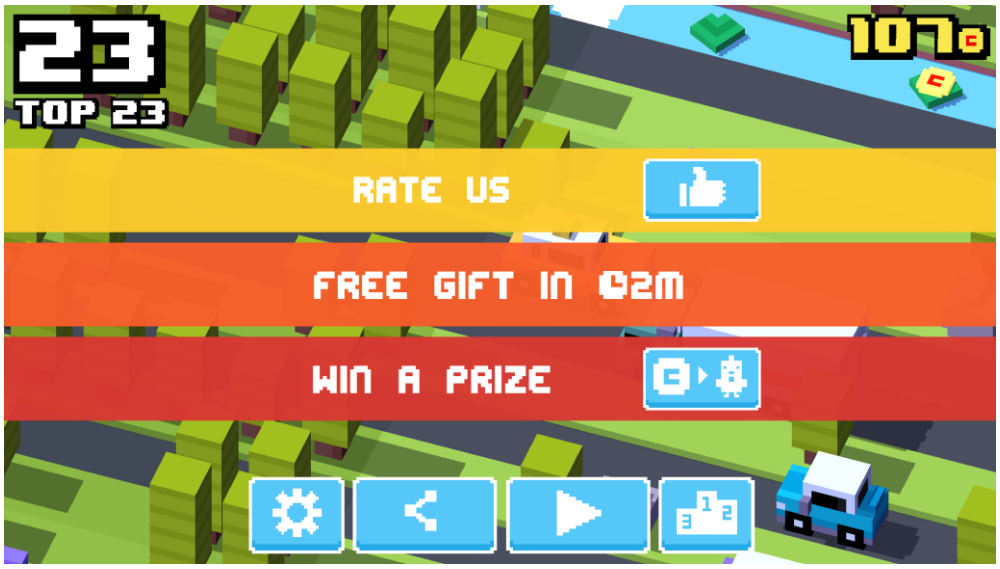
3.5 Udržení pozornosti

Pokud máme aplikaci zdarma ke stažení a karta aplikace vypadá průměrně, uživatel si aplikaci rád stáhne k vyzkoušení. Dost často se bohužel stává, že je aplikace stejně rychle odinstalována, případně není opakovaně spuštěna. Při extrémní situaci může být vysoký počet stažení ale nulový příjem. K příjmu z neplacených aplikací je nezbytné aplikaci používat a to co možná nejčastěji. Jak zaujmout a dlouhodobě udržet uživatele u aplikace můžeme vyčíst u současných titulů. Zde uvedené principy jsou extrahovány z her, některé jsou však přenositelné na veškeré typy aplikací.

Nejdůležitější je zaujmout hned na začátku. Uživatelé, kteří vypnou aplikaci na začátku, se k ní ve většině případech nikdy nevrátí. Uživatel by měl mít možnost dostat se co možná nejrychleji k jádru věci za pomocí jednoduchých kroků. Pokud je tedy v aplikaci nutná registrace, měli bychom implementovat možnost vyzkoušení bez její nutnosti. Obdobně je to s internetovým připojením. Jak jsem uváděl dříve, velké množství uživatelů stahuje aplikace hromadně a k jejich vyzkoušení se vrací později, kdy už k internetové síti nemusí být připojení. Proto by měl mít uživatel možnost vyzkoušet si aplikaci, pokud je to možné, bez připojení i za cenu snížené funkcionality.

Další oblíbenou praktikou je odměňování uživatelů. To funguje na jednoduchém principu: čím více času stráví uživatel v aplikaci, tím více bonusových prvku mu bude odemčeno. Čím déle ji tedy bude používat, stává se aplikace atraktivnější. Studie Josepha Nunese a Xaviera Dredese ukázala, že iluze postupu v progresu působí na uživatele jako závazek, který chtějí dobrovolně dodržet. Do dvou testových skupin rozdali věrnostní kartičky na stejné zboží zadarmo. U jedné skupiny zákazník obdržel zboží po 8 nákupech. U druhé po 10 nákupech s tím, že na kartičce měl již zaškrtnuty 2 nákupy. I přesto že se jednalo o stejný počet nákupů, který musely obě skupiny provést, ve skupině s předvyplněnými kartičkami byla úspěšnost o 50 % vyšší (24). Proto je vhodné dát tuto iluzi uživateli co možná nejdříve.

Skvělým příkladem je populární hra Crossy Road. U hry, která je založena na jednoduchém principu přecházení přes frekventované jízdní pruhy, je uživatel odměněn již při prvním pokusu bez jakékoli souvislosti s výsledkem. Co je ale hlavní, uživatel je informován o další odměně, která bude odemčena po 3 minutách herního času. Ta může být v podobě odemčeného charakteru, zlaťáků v aplikaci či cokoliv jiného. Po třech minutách již uživatel dosáhl úspěchu a postupu v aplikaci a je motivován u aplikace zůstat.



Obrázek 11 Ukázka odměn v aplikaci Crossy Road

Dalším možností je propojit aplikaci s některou ze sociálních sítí (nejčastěji Facebook), která je založena na soutěživosti lidí a pomáhá zvýšit povědomí o aplikaci. Lidé rádi porovnávají své úspěchy s ostatními a tato vlastnost přispívá ke snaze ostatní překonat

a tudíž v aplikaci strávit více času. V některých případech to může fungovat i naopak, kdy se lidé snaží dosáhnout společného cíle a vzdáleně pracují ve dvojici či skupině. Zde motivuje uživatele závazek ke skupině. Největší výhodou je však fakt, že tito lidé mohou být vzdáleni stovky kilometrů.

Dobré je také probudit zvědavost lidí, která je nutí vrátit se zpět do aplikace. Příkladem toho může aplikace Instagram. V dnešní době prahne spousta lidí po uznání a sociální sítě jako tato jim jej dopřávají ve formě hodnocení. Uživatel vystaví fotografii do aplikace, avšak doba reakce na ni je různá. Zvědavost člověka však nutí se do aplikace vracet ke zjištění odezvy na příspěvek.

V neposlední řadě můžou zvýšit používání aplikace za pomocí upozornění. Chytré telefony jsou přeplněny nejrůznějšími aplikacemi a uživatelé sami ztrácejí přehled. Připomenutí formou upozornění s krátkým slovním obsahem zvyšuje využití aplikace a rovněž napomáhá k větším příjmům. Upozornění nemusí být mnohdy jen obecné, ale může poskytovat informace o speciálních akcích v aplikaci či o ztrátě postupu.

3.6 Prezentace aplikace

Na trhu je spousta aplikací, které svojí funkčností nabízí takřka to samé. Jak je možné, že jsou některé úspěšné a jiné ne? Jedním z důvodů může být právě prezentace aplikace a její údaje (název, popis….). Dále si uvedeme jaké společné rysy mají úspěšné aplikace a jakých chyb se dopouštějí autoři těch méně úspěšných.

Nejdůležitějším prvkem je bezesporu název aplikace, ten by měl pokud možno nejkratším způsobem vyjadřovat, k čemu aplikace slouží. Pokud uživatelé z názvu poznají, k čemu aplikace slouží a jak jim může obohatit život, spíše si ji nainstalují. Skvělým příkladem je aplikace Snapchat. Její název je složen ze dvou vět, které přesně vyjadřují, k čemu slouží. Snap (zachytit) znamená zachycení libovolného okamžiku, prostřednictvím fotografie či videa a použít jej k chatu. Při tvorbě aplikace lze tedy být i vynalézavý, avšak nikdy bychom neměli vynechat jednoduchost.

Další důležitou části při prezentaci aplikace je její ikona. Ikona je první věc co uživatel uvidí v obchodě společně s názvem a hlavní rozpoznávací rys aplikace po stažení do telefonu. Spousta uživatelů stahuje více aplikací zároveň a vrací se k nim posléze a právě kvalitní ikona napomáhá k zapamatování. Ikona by rovněž měla symbolizovat

činnost aplikace, popřípadě jeden z jejích hlavních rysů. V tomhle oboru Snapchat mírně pokulhává. Jeho ikona je sice zapamatovatelná a dobře se hledá, avšak souvislost s její činností je neznámá. Naopak výbornou ikonu zvolil Instagram, který tvorbu fotografií a jejich úpravu pomocí filtrů symbolizuje retro fotoaparátem, nebo Dropbox(úložiště) tvořeno jednoduchou krabicí.

Obrázek 12 Ukázka vhodně zvolených ikon u aplikace Instagram a Dropbox

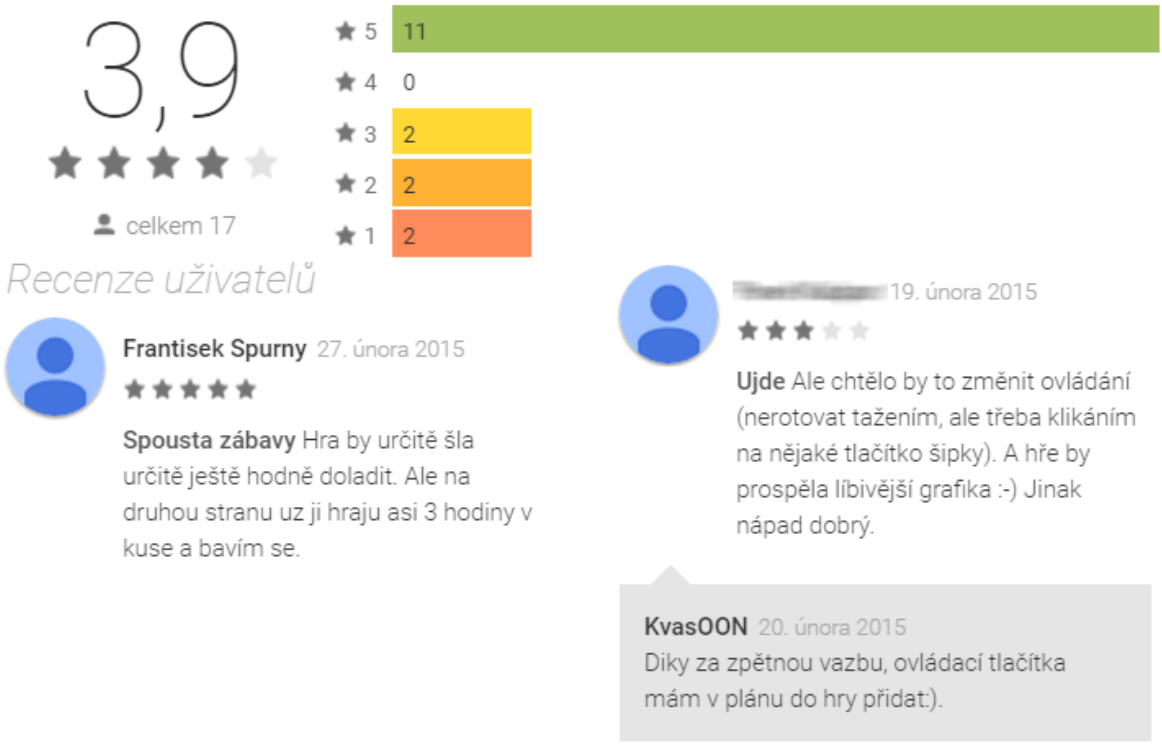
Po té, co si uživatel vybere ze seznamu aplikací v obchodě a otevře se mu její karta, je na stránce další prvek nastavený vývojářem – snímky obrazovky. Ty by měly opět reflektovat, co a jak aplikace provádí a vyzdvihnout její design a eleganci. Jejich volba se trochu komplikuje z důvodu výběru cílového trhu. Uživatelé z různých zemí světa mají odlišnou představu, jak by takové snímky měly vypadat. Zatím co trh Spojených států a Evropy preferuje jednoduchý snímek z aplikace (někdy s mírným popisem), Asijské země vyžadují kreslený až komiksový styl celého snímku. Naštěstí je ale možné vybrat jaké obrázky se mají objevovat na základě země, ze které se uživatel připojí. Příkladem toho může být aplikace Jelly Splash.

Obrázek 13 Snímky obrazovky pro Evropský a Asijský trh

Posledními volenými prvky jsou popis a klíčová slova. Popis by měl být stručný a shrnout základní principy a přednosti aplikace. Klíčová slova jsou slova, podle kterých lze aplikaci v obchodě najít. Při jejich volbě je nejlepší metodou vžít se do role uživatele a situací, ve kterých bude potřebovat právě takovou aplikaci a zjistit jakými kombinacemi ji bude hledat.

Recenze a hodnocení aplikace

Po té co je aplikace umístěna na trh pro rozšíření mezi veřejnost, ji uživatelé můžou hodnotit počtem hvězd a komentářem. I přesto, že se u některých začínajících vývojářů jedná o opomíjený prvek, hodnocení aplikace je veřejně přístupné všem potencionálním uživatelům a právě na něm mnohdy závisí, zdali si jej stáhne do svého zařízení či nikoli.



Obrázek 14 Ukázka přínosného hodnocení u aplikace Puzzle Brain Stimulator (25)

Hodnocení je přirozená cesta ke zvyšování postavení aplikace, což vede k většímu počtu stažení. Jak to ale celé funguje? Jak již zbývá zvykem, žádný nově uvedený produkt nebývá zcela bez chyb, což mnohdy vede ke špatnému počátečnímu hodnocení. Vývojář se ale nesmí odradit počáteční kritikou a naopak z ní vytěžit co nejvíce informací a co nejdříve je zaimplementovat do aplikace, což vede k pozvolnému přelomu hodnocení

na kladné. Pokud však vývojář připomínky nerespektuje, aplikace nemá vysokou šanci na úspěch. Jakmile ale postavení aplikace na trhu stoupá, zvyšuje se i počet uživatelů, kteří se dostanou na její profilovou stránku a po té si ji i stáhnou.

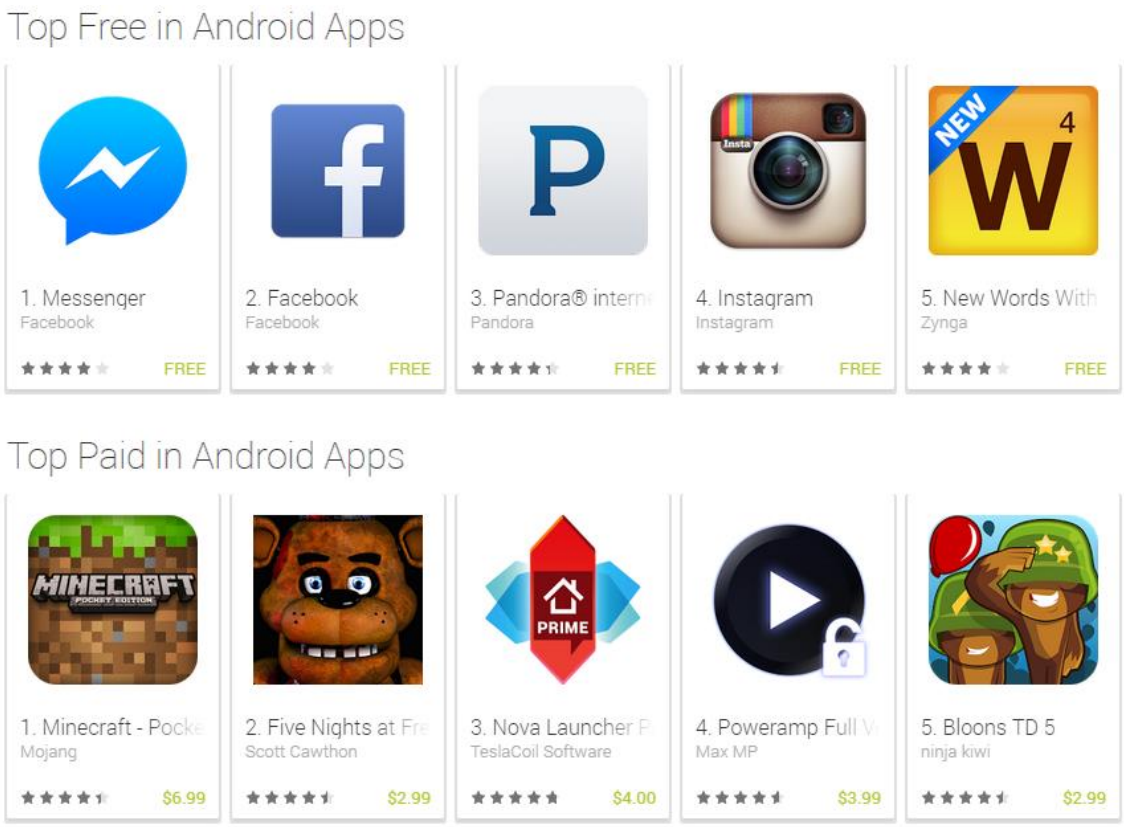
Existují ale také uživatelé, kteří kvalitní aplikaci při spokojenosti neohodnotí, ať už proto, že s hodnocením nemá předchozí zkušenosti nebo v tom nevidí vlastní užitek. Proto se vývojáři snaží tento krok co nejvíce usnadnit a do programu vhodně implementují odkaz přímo na stránku hodnocení. U některých typu aplikací to lze vyřešit výhodou pro uživatele, například u her, kde je odemčen bonusový obsah za udělení hodnocení. Další možností je nabízet hodnocení uživateli na konci významného kroku v aplikaci. U seznamovací aplikace to tak může být při nalezení vhodného protějšku či po ukončení konverzace.

I když hodnocení nepomůže stávajícím uživatelům, napomáhá velmi uživatelům potencionálním. Právě tady z komentářů zjistí největší klady a zápory které aplikace nabízí a podle toho se následně rozhodnout. Co může být ještě horší než špatná recenze je však recenze žádná, uživatel se může dokonce domnívat, že se jedná o škodlivý program a aplikaci si raději nestáhne. Proto je dobré, pokud máme možnost, požádat o upřímné prvotní hodnocení lidi ze svého okolí, kteří spadají do cílové kategorie.

Hodnocení lze získat i jinou cestou, kterou bych však nikomu nedoporučoval. Tento způsob spočívá v zaplacení třetí straně za předem domluvený počet a výši hodnocení. I přes to že se může zdát, že jde o dobrý způsob, většinou se jedná pouze o zvýšení výdajů na aplikaci. Tento způsob je však již dobře znatelný i pro obyčejné uživatele a ubírá na důvěryhodnosti.

Cizí recenze

Pokud ještě nemáme hodnocení u vlastní aplikace, můžeme se poučit z hodnocení jiných, konkrétně z negativních, tak jako to udělal Roland Burrows ve svém článku Lessons Learned From 100 Negative App Reviews (26). Přestože analýzu provedl již na konci roku 2014, z ní plynoucí výsledky jsou stále aktuální. Rozhodl se vybrat tři nejstahovanější aplikace podle typu příjmů a to aplikaci Messenger z kategorie zdarma, Pandora z kategorie nákupy v aplikaci a Minecraft Pocket Edition z placených aplikací.



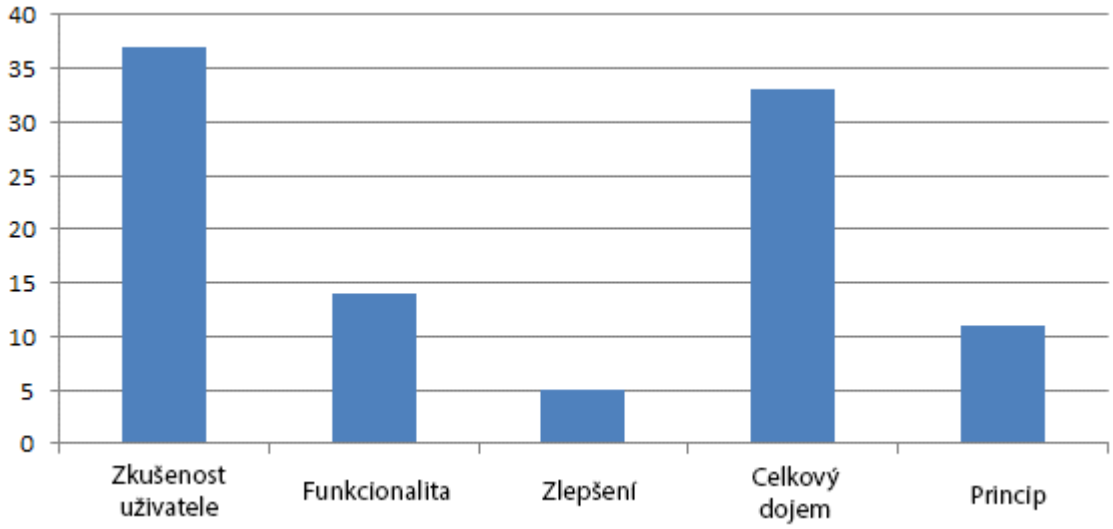
Obrázek 15 Top aplikace na konci roku 2014 (26)

Jak jsem zmiňoval výše, v obchodě funguje systém pěti hvězdového hodnocení. Pokud chceme získat prospěšné údaje z hodnocení zaběhlých aplikací, obecně platí, že jednobodové hodnocení můžeme zanedbat. Většinou se jedná o přehnaně kritická hodnocení a přínosnější jsou dvouhvězdové hodnocení. Při jejich analýze Burrows zjistil, že je lze rozdělit do pěti následujících kategorií:

- Zkušenost uživatele – „Aplikace již nefunguje tak jak jsem byl zvyklý“
- Funkcionality – „Nefunguje po přesunu na SD kartu“
- Zlepšení – „Ocenil bych větší možnost úpravy vzhledu“
- Celkový dojem – „Aplikace je pomalá“
- Princip – „Aplikace neodpovídá popisu a uráží jistý druh lidí.“

První vybranou aplikací je obecně známý Messenger, který umožňuje zasílání zpráv mezi uživateli sítě Facebook. Aplikace založená na jednoduchém principu, jejíž hlavní problém je násilný přechod z původní aplikace Facebook, prostřednictvím které zasílání zpráv fungovalo dříve. Jednoho dne však rozhodli, že pokud chce uživatel zasílat zprávy,

musí si stáhnout další aplikaci. Posledních 100 negativních recenzí díky tomu bylo rozděleno dle kategorií následovně:



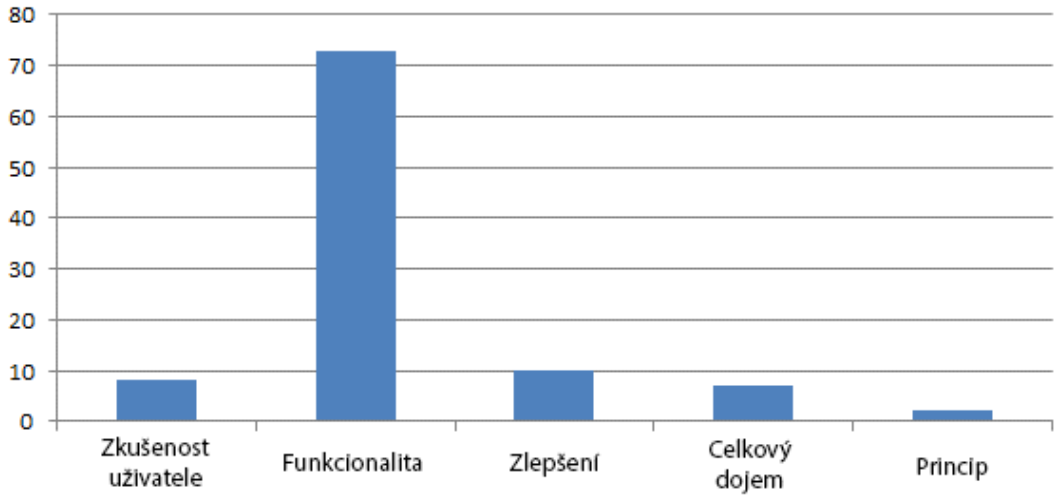
Graf 7 Rozdělení negativních hodnocení u aplikace Messenger (26)

Všech 37 hodnocení uživatelského přístupu souviselo s přesunem zpráv do zvláštní aplikace. Překvapivá stížnost byla ale v kategorii principu, kde uživatelé vyjadřovali znepokojení nad počtem vyžadovaných povolení k používání aplikace. V celkovém dojmu pak byly nejčastěji zmiňovány právě tyto dva faktory nebo obecné stížnosti.

Poznatky:

- Nenutit stahovat uživatele další aplikaci
- Vyžadovat minimum povolení k přístupu

Další vybranou aplikací je Pandora Internet Radio, známá jako Pandora. Jedná se aplikaci pro poslouchání hudby prostřednictvím streamování obsahu do uživatelova zařízení. Jedná se o velmi podobný princip jako u aplikace Spotify. Aplikace vznikla jako rozšíření služby, která fungovala nejdříve na PC a posléze se jí na platformě Android dostalo ještě většího uznání.



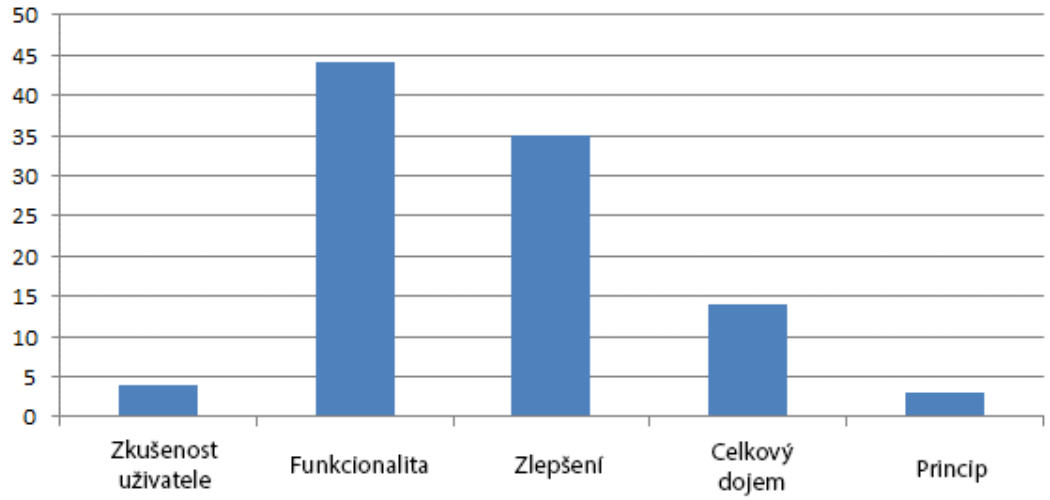
Graf 8 Rozdělení negativních hodnocení u aplikace Pandora (26)

Jednoznačně největší podíl zde mají hodnocení týkající se funkcionality. Uživatelé reportují například chyby, které mohou vznikat pouze na některých zařízeních či komentují změny provedené ve funkcionalitě a hodnotí jejich správnost.

Poznatky:

- Neodebírat žádanou funkcionalitu
- Testovat pro různou konfiguraci

Poslední aplikací je Minecraft: Pocket Edition, což je mobilní verzí počítačové hry Minecraft, která se stala jednou z nejrozšířenějších Indie her současnosti a získala kladné hodnocení kritiků. Zde je uvedeno rozdělení hodnocení tři roky po jejím vydání.



Graf 9 Rozdělení negativních hodnocení u aplikace Minecraft (26)

Vedle funkcionality uživatelé nejčastěji navrhují možná zlepšení. Aplikace byla vydána v pracovní verzi a interval mezi jednotlivými aktualizacemi je příliš dlouhý. Uživatelé tak hlásí neustálé větší množství chyb a požadují, aby se možnosti aplikace podobaly jejímu předchůdci na jiné platformě.

Poznatky:

- Nevydávat pracovní verzi pokud neplánujeme časté aktualizace
- Uživatelé porovnávají mobilní verzi s původní

Poznatky plynoucí z tohoto hodnocení jsou tedy nenutit uživatele k dalšímu využívání stávající funkčnosti stahovat další aplikaci a nevyžadovat povolení přístupu k funkcionalitám telefonu, pokud se nejedná o nezbytné prvky k chodu aplikace. Dále také nikdy nevydávat aplikace v pracovní verzi dříve než je zcela hotová, pokud neplánujete její brzké dokončení a intenzivní aktualizace. Při tvorbě aplikace na základě počítačového předchůdce je také nutno počítat s neustálým porovnáváním a požadováním rovnocenné funkcionality. Před uvedením na trh je nezbytné testovat aplikaci na různých zařízeních k odhalení co nejvíce potencionálních chyb a nikdy neodstraňovat přebytečnou funkčnost bez průzkumu uživatelů.

3.7 Globální přístup

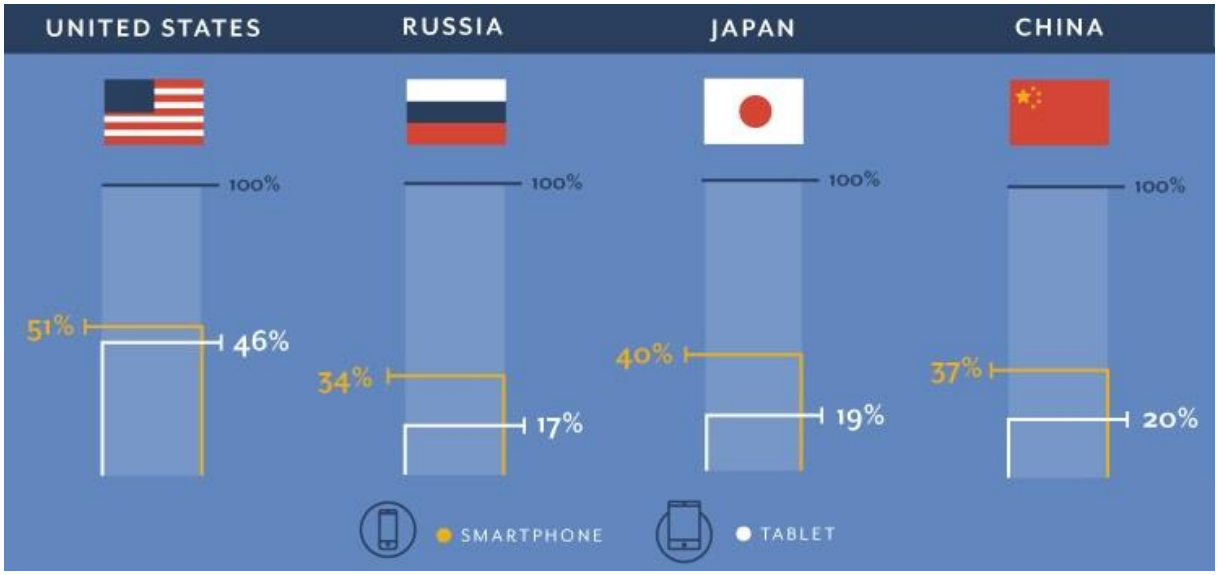
Vytvořit aplikaci pro domácí je u většiny vývojářů první krok. Znají dobře místní podmínky a dokáží se nejlépe vcítit do potřeb lidí. Bohužel počet uživatelů chytrých telefonů je v České republice pouze v pár milionů. Na světě jsou však miliardy lidí ze 190 zemí světa kteří denně stahují nové aplikace. Pokud začne vývojář myslet tímto způsobem, má mnohem vyšší počet potencionálních uživatelů a tudíž větší šanci uspět. Příjem vývojářů v Severní Americe je z 50 % tvořen ziskem z jiných zemí. U developerů v Asii je to 60 % a u Evropských zemí dokonce 85 %.

Hlavním problém je rozdílnost trhů. Pokud zanedbáme základní funkce telefonu, jako jsou volání a SMS, je zařízení v závislosti na trzích využíváno následovně. Pokud začneme v Americe, nejvíce majitelů využívají svá chytrá zařízení k obsluze emailu a zjišťování údajů o počasí. O něco méně k fotografování a sociálním sítím a hraní her a vůbec nejméně k zasílání zpráv skrze online sítí. K zobrazení preferují reklamy

v aplikaci zobrazované dle jejich zájmů. Většina uživatelů používá každý den 1-10 aplikací. Chytrá zařízení jsou zde běžným vybavením zhruba v 50 % domácností.

V Číně, která dominuje ve výrobě těchto zařízení, je počet domácností menší. Zařízení jsou nejvíce využívána k zjišťování informací a fotografování. Dále komunikují pomocí online zpráv a nejméně posílají emaily. Pro tento trh není zcela vhodná placená aplikace, jelikož třetina stahuje pouze aplikace zdarma, ale naopak 90 % uživatelů přecházejí na stránky inzerované prostřednictvím reklam.

V Rusku se daří placeným aplikacím, pokud existuje i trial verze aplikace. Alespoň to tvrdí 59 % místních uživatelů. Nevadí jim prý zaplatit menší poplatek za aplikaci, kterou si mohou předem vyzkoušet v kompletním provedení. Co ale odmítají, je kupovat zajíce v pytli. Počet vlastníků zařízení je obdobný jako v Číně a způsob využívání aplikací je podobný jako v Americe, s rozdílem vyššího využívání online zpráv. Ze všech uživatelů prý 79 % ocení i reklamy v aplikaci, pokud jsou odpovídající jejich potřebám.



Obrázek 16-Počet obyvatel vlastnící chytrá zařízení dle zemí (27)

Poslední analyzovanou zemí je Japonsko, kde chytré telefony vlastní přibližně 40 % obyvatel. Zde preferují individuální reklamy, avšak v mnohem vyšší míře. Ideálně prý nejen podle aplikací, ale i podle věku a pohlaví. I přesto však na stránky s jejich obsahem přechází pouze 43 % lidí. Lze ale říci, že Rusko i Japonsko jsou země vhodné na aplikace s příjmem za reklamy. Zařízení jsou převážně využívána k vyřizování emailů a získávání informací.

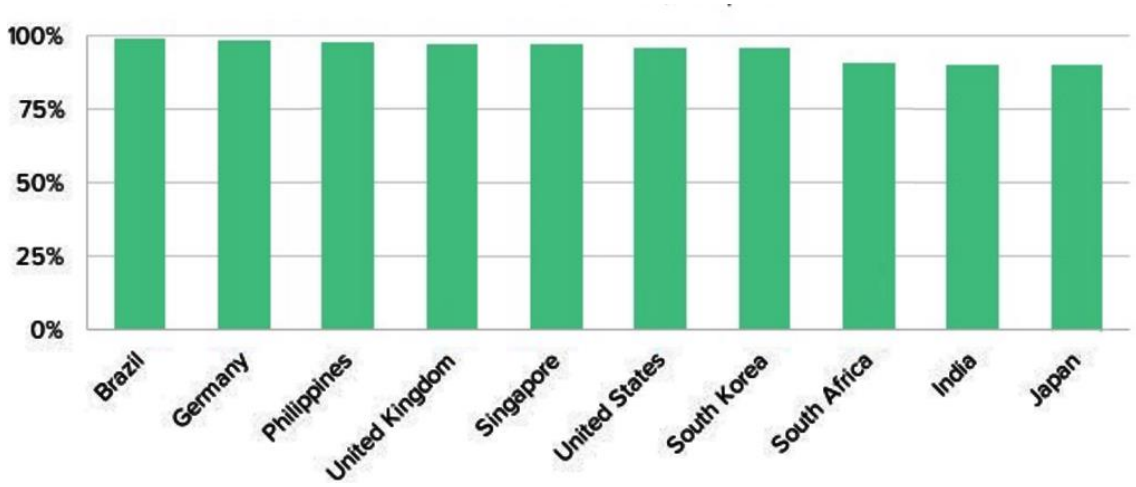
S globalizací aplikací se pojí i čas, který je potřeba strávit nad tvorbou lokalizace. Vývojář jej může jistě zanedbat, což ale vede ke ztrátě uživatelů. Pouze v zemích jako je Čína a Brazílie lidé ochotně stahují i takové aplikace, nikoliv však v Japonsku a USA. Globalizace je ale v každém případě vřele doporučována.

3.8 Užívání internetového připojení

Využití dat v telefonu neustále roste v závislosti na počtu vlastníků chytrých zařízení společně s náročností na webový obsah optimalizovaný pro tyto přístroje. Pro poskytovatele internetového připojení a výrobce poptávaných zařízení je to potěšující fakt, který jim umožňuje zainteresovat tuto skutečnost do jejich plánu a přilákat tak více potencionálních zákazníků.

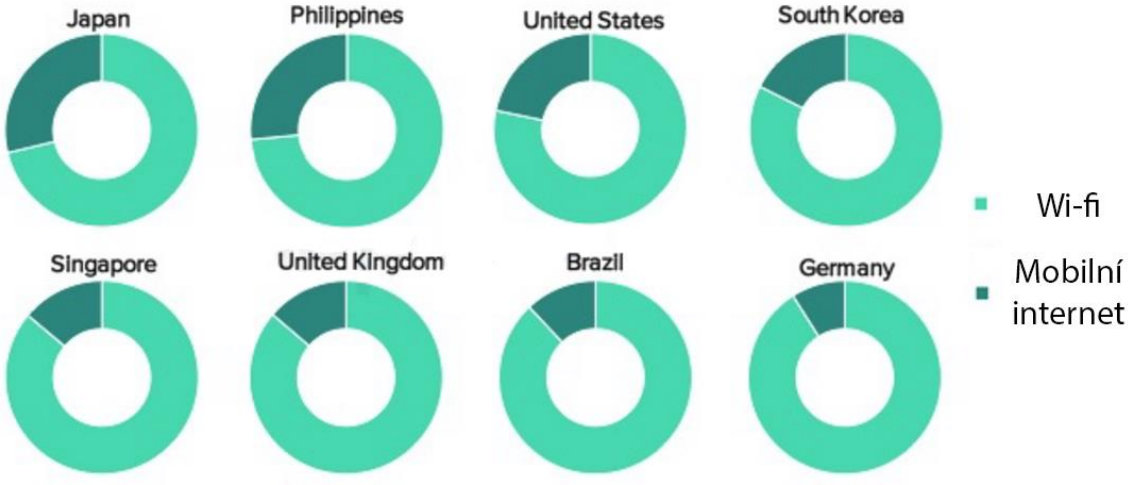
Další otázkou při návrhu aplikace je tedy pochopitelně to, zdali by měla využívat připojení k internetu. Berme v potaz nejen účel a potřeby našeho návrhu, ale i způsob příjmu z aplikace, jelikož reklamy v aplikaci jsou stahovány právě pomocí připojení. V neposlední řadě v jakých mezích má aplikace data sdílet, s ohledem na FUP limit u mobilního připojení. Tato analýza byla provedena se zaměřením na celosvětový trh, konkrétně na země, jejichž příslušníci internet v telefonu využívají nejvíce.

V současné době se nejvíce mluví o mobilních sítích jako jsou 3G a LTE, až se může zdát, že úloha wifi připojení je již historií, kvůli jejímu pevnému zdroji signálu s velmi omezeným dosahem. Wi-fi je však technologie, která bezdrátový internet v telefonu zpopularizovala a dodnes je u mnoha uživatelů preferovaným způsobem. Z celkového počtu majitelů chytrých zařízení je více než polovina aktivními spotřebiteli internetu připojena skrze Wi-fi, jak je vidět na obrázku.



Graf 10 Počet majitelů chytrých zařízení aktivně využívající WI-FI (28)

Uživatelé rádi používají oba typy připojení, vzhledem k jejich ceně. Zatím co Wi-fi připojení má neomezenou kapacitu přenesených dat, mobilní připojení nikoli. Uživatelé si tak díky Wi-fi můžou snížit průměrné náklady na připojení a mobilní data využívat pouze v případě kdy jsou mimo dosah. Dnes proto platí, že 9 z 10 uživatelů využívající mobilní signál přepíná mezi oběma typy připojení, což dnes inteligentně provádí sami přístroje. Míra, do jaké lidé jednotlivé připojení konzumují, se liší dle jednotlivých zemí. To může být způsobeno jednak cenou mobilního internetu, ale i počtu veřejných Wi-fi připojení napříč zeměmi. Jak důležité jsou obě varianty je vidět na následujícím grafu.



Graf 11 Poměr využívání mobilních dat a WI-FI v různých zemích (28)

Tento graf se však během jednotlivých let velmi měnil, například v Brazílii se během posledních dvou let zvedl počet veřejných přístupových míst o 83 %, přičemž ve větších

městech je jejich průměrný počet okolo 100 (29). Vedle toho nikterak nezahálí ani poskytovatelé mobilního internetové připojení, kteří vedle rychlosti zlepšují také pokrytí a velikost FUP limitu.

Z hlediska použití dat z internetu se tedy vývojář nemusí nijak omezovat. Reklamy v aplikaci či menší množství vyžadovaných dat není problémem pro širokou část zákazníků a při větší náročnosti na data musíme počítat se stabilitou zařízení. Absence internetu tedy může napomáhat k větší rozšiřitelnosti aplikace, avšak jeho využití může nabízet mnoho výhod a volba tak závisí pouze na návrháři.

3.9 SWOT analýza

Tato SWOT analýza se bude zabývat postavením začínajícího vývojáře na trhu mobilních aplikací.

	+	-
Vnitřní	**Silné stránky**	**Slabé stránky**
	Dlouholeté zkušenosti s vývojem aplikací v jazyku Java Schopnost plánování a řízení projektu Znalost oborových informací	Omezené lidské zdroje Nedostatek času Neznámé jméno Nízký rozpočet
Vnější	**Příležitosti**	**Hrozby**
	Možnost rychlého rozšíření mezi uživateli Díky centralizovanosti trhu lze najít snadno mezeru na trhu	Vstup konkurenčního programu na trh Prolomení bezpečnosti aplikace třetí stranou

3.10 Shrnutí analýzy

Z analýzy bohužel vyplývá, že u většiny faktorů nelze přesně určit univerzální přístup. Například u rozhodování příjmu z aplikace je potřeba předem dobře zvážit o jakou aplikaci se jedná, kolik nabízí funkcionalit a jak často k ní bude uživatel přistupovat. V rámci toho se může teprve rozhodnout, zdali a jaký typ zpoplatnění aplikace vybrat. Co se týká užívání aplikace, zejména u her je nezbytné udržet uživatelovu pozornost a při delší neaktivitě samovolně upozorňovat a zajistit tak její užívání. Obecně platí, že delší životnost aplikace vede k většímu rozšíření aplikace, což pro vývojáře znamená vyšší zisky.

Díky analýze jsme ale rovněž zjistili důležitost prezentace aplikace, která bývá mnohdy podceňována. Každý prvek na trhu musí mít adekvátní ikonu a název vzhledem ke své činnosti a rovněž musí být název snadno zapamatovatelný a ikona na první pohled dobře rozeznatelná. Vývojář by měl rovněž reagovat na požadavky a připomínky uživatelů formou recenzí, což vede ke zvýšení průměrného hodnocení, které může ovlivnit další potencionální uživatele. Přínosné mohou být recenze i cizích aplikací. Díky těm jsem charakterizoval další praktiky, které by měl vývojář dodržovat. Jsou to například nikdy nevydávat nedokončenou verzi, provést řádné testování, nenutit stahovat další aplikace a vyžadovat minimum povolení k přístupu do funkcí telefonu.

V poslední části analýzy se zabývám globalizací aplikací a internetovým připojením. Pokud to aplikace dovoluje, je vhodné necílit s aplikací na konkrétní trh ale nabídnout ji celému světu. V takovém případě je však nezbytné dát si pozor na správnou lokalizaci a uzpůsobit prezentaci aplikace jednotlivým trhům. Co se týká internetového připojení, vývojář v dnešní době není limitován jeho absencí, avšak v závislost na jeho využití musí brát v potaz při rozhodování o příjmu z aplikace. Zároveň by měl dbát na jeho efektivním využití, neboť většina uživatelů používá mobilní připojení, které bývá často omezeno.

Konkrétní nastavení jednotlivých faktorů uvedu v následující kapitole. Ty budu vhodně vybírat za pomoci výše uvedených znalostí a dle požadavků na mnou navrhovanou aplikaci.

4 Vlastní návrh řešení, přínos práce

V předchozí kapitole jsem uvedl, jaké jsou hlavní rysy úspěšných aplikací. Zde je na základě těchto údajů proveden návrh vlastní aplikace a následně popsán způsob implementace a prezentace na Android trhu.

Dle informací z analýzy jsem vybral tři vlastní nápady na aplikace a nechal vybrat nejatraktivnější z nich lidmi ve svém okolí formou dotazníku. Dle výsledků jsem se rozhodl pro aplikaci sloužící k hlídání volných míst ve spojích společnosti Student Agency a následné automatické rezervaci. Pro cestování u této společnosti je nezbytné si místo předem zarezervovat, avšak na hlavních trasách jsou místa často zabrána několik týdnů dopředu. Vzhledem k tomu, že společnost nabízí bezplatné odhlášení z jízdy nejpozději půl hodiny před termínem odjezdu, dochází často k blokování místa cestujícími, kteří se ze spoje postupně odhlašují. Přestože existuje, obdobný program pro platformu Windows, většina uživatelů není ochotná nechat běžet počítač bez dozoru celý den jen kvůli rezervaci lístku. Naopak by ale uvítali aplikaci pro mobilní zařízení, jelikož (jak vyplývá i z analýzy) počet lidí s chytrým telefonem připojeným skrze mobilní data k internetu stále narůstá.

Aplikace bude splňovat i spoustu faktorů z předchozí kapitoly i přesto že se nebude jednat o herní aplikaci. Vzhledem k tomu, že se bude jednat o aplikaci žádanou uživateli, nemusíme se obávat o udržení pozornosti či době životnosti aplikace. Předpokládá se, že uživatelé budou neustále vyžadovat rezervaci na vytížených spojích a největším rizikem je tak krach firmy. Vzhledem k nutnosti mít zapnutá data při používání aplikace, také nebudu limitován výběrem zpoplatnění, jelikož vím, že uživatel bude vždy připojen k internetu a aplikaci bude pravidelně navštěvovat.

4.1 Návrh řešení

Poznatky z kapitoly analýzy však nestačí k zahájení implementace aplikace. V této kapitole jsou proto rozebrány uživatelské požadavky na aplikace včetně mých poznatků a následně proveden návrh za pomoci doménového modelu.

4.1.1 Analýza požadavků

Základním stavebním prvkem veškerého vývoje softwaru je bezesporu analýza požadavků. Zanedbání této části se při dalším vývoji projevuje negativně nejen na času

dokončení, ale dokonce může vést i k nedokončení projektu či implementaci nevhodné aplikace.

Primárním cílem aplikace by mělo být hlídání volných spojů u společnosti Student Agency. S tímto úkolem je spojeno mnoho dalších požadavků na aplikaci nejen pro její základní funkčnost, ale i pro zajištění příjemné práce s aplikací.

V první řadě by měl uživatel mít možnost vyhledat libovolný spoj, který firma nabízí. Aplikace bude podporovat pouze tuzemské spoje. Vyhledání by mělo probíhat obdobným způsobem, jako je na oficiálních stránkách společnosti a na který jsou uživatelé dobře zvyklí. Konkrétně je to tedy tak, že zadá destinaci pro odjezd a cíl a vybere datum odjezdu. Po té by se měly zobrazit spoje pro konkrétní den se základními informacemi a možnost přidání spoje ke hlídání. Uživatel by měl mít možnost hlídat více spojů současně. Všechny hlídané spoje by měly být zobrazené v jednoduché tabulce spolu s možností spoj kdykoli odebrat. V případě uvolnění místa na hlídaném spoji, by měl být uživatel informován formou upozornění v notifikační liště.

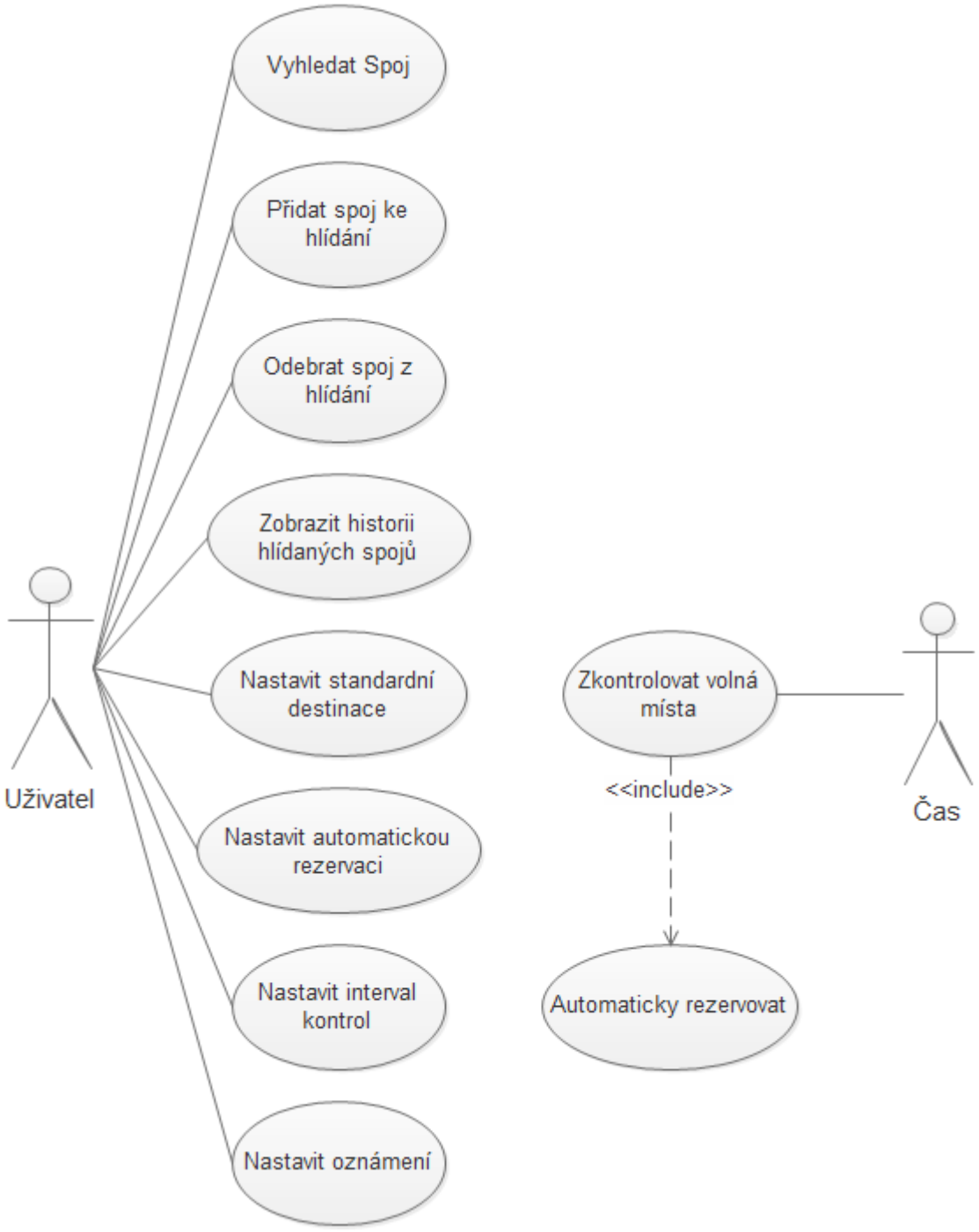
Aplikace by měla rovněž podporovat možnost automatické rezervace jízdenek a umožnit uživateli nastavit, v rámci kterého tarifu má jízdenky rezervovat. Uživatel tak nemusí mít telefon neustále na očích a zároveň bude mít vyšší šanci na získání jízdenky v případě jejího uvolnění.

Ke zlepšení uživatelského pohodlí by měla aplikace mít možnost definovat, jaké destinace budou standardně nastaveny po spuštění aplikace a umožnit uživateli přizpůsobit si notifikace vyskakující aplikací.

Vzhledem k tomu, že aplikace bude využívat při každé kontrole míst internetového připojení, může být pro uživatele s menším datovým FUP limitem nevhodná. Toto riziko lze eliminovat možností pro uživatele vybrat si v jakém intervalu má aplikace volné místo kontrolovat.

Pravidelná kontrola spojů by měla probíhat, aniž by se uživatel nacházel v aplikaci. Po přidání spoje ke hlídání by měl proces kontroly běžet na pozadí a neomezovat uživatele žádným způsobem od standardního používání zařízení.

Diagram případů užití



Obrázek 17 Diagram případů užití

4.1.2 Diagram aktivity

Obrázek na další straně znázorňuje diagram aktivit hlavního procesu aplikace, tedy hlídání spoje. V první řadě je uživatel nucen vyhledat příslušný spoj. To bude ve skutečnosti představovat zvolení data odjezdu a výběr místa výjezdu a cíle spoje. Jakmile spoj nalezne, může jej přidat mezi hlídané spoje.

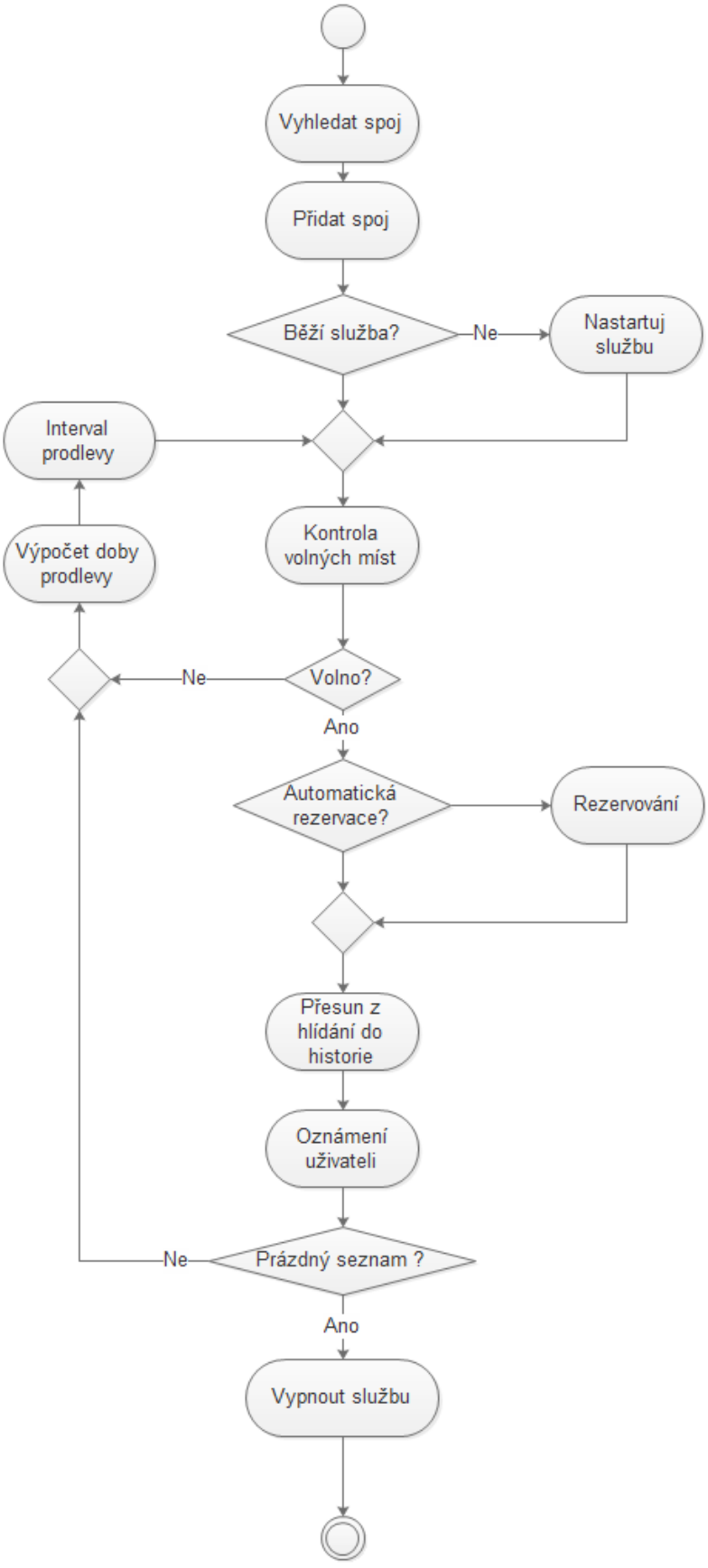
V případě že se jedná o první spoj v seznamu, znamená to, že proces kontroly neběží a je nutné jej spustit. V opačném případě dojde rovnou ke kontrole volného místa u všech spojů v seznamu. Pokud se žádné místo neuvolnilo, vypočítá se nový čekací interval a po jeho uplynutí se znovu provede kontrola míst. Pokud se v hlídaném spoji uvolní místo, aplikace zkontroluje, zdali uživatel požaduje automatické přihlášení a zadal všechny příslušné údaje. Pokud je vše v pořádku, aplikace pokračuje tak, že přesune spoj ze seznamu hlídaných do historie a upozorní uživatele, že se uvolnilo místo, případně informuje o stavu rezervace.

Po modifikaci seznamu hlídaných spojů je nezbytné dotázat se, zdali jsou v seznamu ještě některé položky. Pokud ano, pokračuje se znovu k výpočtu doby prodlevy a cyklus se opakuje. Pokud se v seznamu již nenachází žádné další položky, můžeme proces ukončit.

4.1.3 Doménový model

Jak jsem uvedl již v teoretickém úvodu, společně s diagramem užití se v počáteční fázi vývoje programu vytváří také doménový model, který následně slouží jako obecný návrh tříd při implementaci a znázorňuje jejich nejdůležitější atributy. Domény v modelu a jejich atributy jsou pojmenovány anglicky, z důvodu následné analogie se skutečnými třídami.

I přesto že se nacházíme v České republice, a u většiny vývojářů je mateřský jazyk čeština, zdrojové kódy jsou psané v angličtině a to i tehdy pokud programy vznikají výhradně pro český trh. Neexistuje sice předpis, který toto udává, ale obecně platí, že anglicky napsaný kód je srozumitelnější pro širší spektrum programátorů a při použití knihoven, které jsou všechny psané v angličtině, nedochází ke vzniku dvojjazyčných kódů.



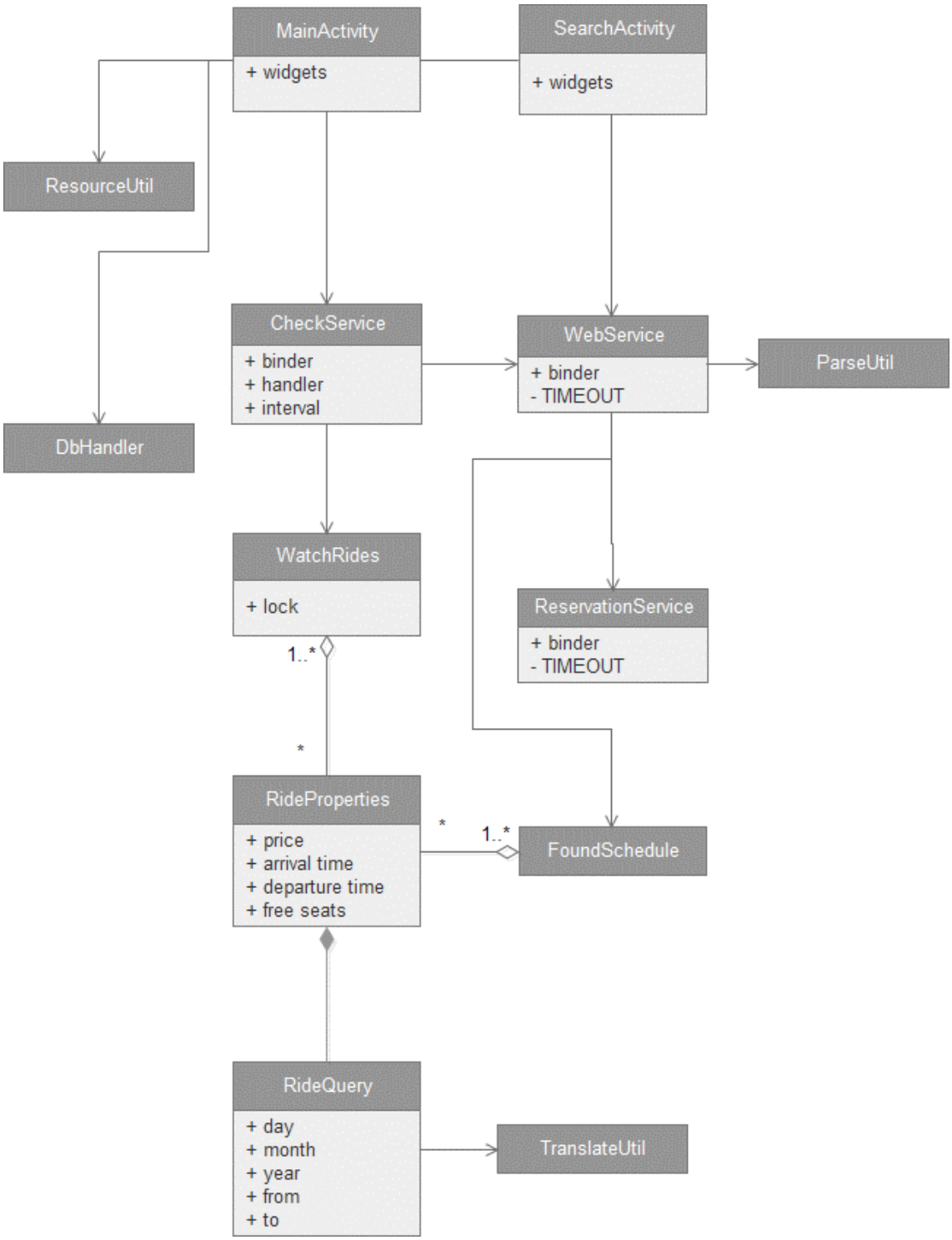
Obrázek 18 Diagram aktivity hlavního procesu aplikace

Víme, že základním stavebním prvkem Android aplikací je aktivita, která reprezentuje jednu obrazovku aplikace. Vzhledem k tomu že se v našem případě jedná o velmi jednoduchou aplikaci, budeme mít pouze dvě třídy aktivit. Třídu **MainActivity**, která bude představovat hlavní obrazovku aplikace a **SearchActivity** ve které bude pouze tabulka s jednotlivými spoji. Mým zvykem je do aktivity jako její atributy nahrát všechny její grafické prvky z designu, což jsem v modelu reprezentoval jedním atributem widgets.

O hlavní procesy na pozadí v aplikaci by se měly starat tři služby, které se budou moci volat navzájem. Hlavní služba, která bude vyvolána z hlavní aktivity, ponese název **CheckService**. Ta bude běžet nezávisle na celé aplikaci a bude inicializátorem pravidelných kontrol a v případě nalezení volného místa také vykonavatelem upozornění uživatele.

Samotnou internetovou komunikaci a přístup k webu Student Agency bude realizovat služba **WebService**. Ta bude volána aktivity SearchActivity při hledání určitých spojů dle zadání uživatele, ale hlavně z druhé služby CheckService při pravidelné kontrole volných míst u vybraných spojů. V případě úspěšného nalezení a požadování automatické rezervace bude zavolána poslední služba **ReservationService**, která bude zajišťovat samotný proces rezervace. Na první pohled se zdá, že tento proces mohl být rovněž obsažen v WebService, jelikož bude ale proces rezervace náročným prvkem na implementaci a současný návrh počítá s odlišným způsobem kontroly a rezervace, budou tyto funkce v oddělených službách. Při přistupování k vzdáleným serverům je rovněž potřeba určit maximální dobu, kterou má služba čekat na výsledek. Tato doba bude v obou třídách určena parametrem **TIMEOUT**.

Nyní něco k datovým objektům aplikace. Základním prvkem je bezesporu spoj, který bude reprezentován třídou **RideProperties**. Tato třída bude obsahovat všechny důležité informace o spoji nezbytné k činnosti aplikace, jako jsou informace o času odjezdu, příjezdu, ceně či počtu volných míst. Další data budou uložena ve třídě **RideQuery**, přičemž RideProperties bude obsahovat instanci této třídy. V té bude uložen datum a informace o místě odjezdu a příjezdu. Toto rozdělení dat jsem navrhl z důvodu následné implementace HTML dotazování. Třída RideQuery bude rovněž obsahovat funkce, které umožní z uložených dat generovat URL pro HTML dotazy.



Obrázek 19 Doménový model

RideProperties může být následně součástí dvou tříd, jednou bude **WatchRides**, která bude obsahovat spoje určené ke hlídání a podpůrné operace nad nimi. K této třídě se bude následně přistupovat ve službě CheckService ale i z hlavní aktivity. Jelikož

se bude jednat o dva asynchronní procesy, je nezbytné implementovat zámek, aby nedocházelo k nekonzistentním stavům dat.

Dále lze najít RideProperties ve třídě **FoundSchedule**, která bude reprezentovat seznam spojů při vyhledávání dle zadání uživatele. Přestože tato třída bude obsahovat pouze list jednotlivých spojů, plánuji rozličné přístupy k seznamu, které mi knihovní funkcionalita neumožňuje, a proto jej hodlám implementovat zde.

Poslední kategorií budou podpůrné statické třídy, které pracovně nazývám utility, z důvodů jejich použití bez nutnosti instanciace. První z nich bude **TranslateUtil**, která bude realizovat některé datové překlady, jako například název stanice na zkratku kterou je reprezentována HTML žádosti a podobně. Data u Android aplikací jsou ukládány do takzvaných zdrojů a přístup k nim bude realizován za pomocí další třídy s názvem **ResourceUtil**. Mimo bude třída realizovat i případné modifikace a dodávat tak data vhodná pro konkrétní činnost.

Jelikož jako odpověď na HTML dotaz je webová stránka, je nezbytné ji analyzovat a najít v ní patřičná data. Tato část práce bude prováděna ve třídě **ParseUtil**, jejíž funkce budou volány přímo ze služby WebService, přičemž služba vždy předá stažený dokument a následně dostane instance třídy FoundSchedule.

Pro dlouhodobé ukládání dat v Android aplikací slouží takzvaná databáze. Ta bude definovaná v třídě DbHandler, a skrze ni budou probíhat i veškeré operace s daty. Do databáze se bude ukládat především historie hlídání. Ostatní data jako například informace o nastavení uživatele či jeho údaje pro automatickou rezervaci budou ukládány automaticky do takzvaného Shared Preferences třídou SettingsActivity. Tu v doménovém modelu nenajdete z důvodu přehlednosti, jelikož je automaticky generována programem Android Studio a váže na sebe několik dalších tříd.

4.2 Výsledná aplikace

Při implementaci mobilní aplikace jsem se držel návrhu řešení z předchozí podkapitoly. Struktura tříd aplikace zcela odpovídá doménovému modelu, s výjimkou přidání některých tříd zajišťující podpůrnou činnost.

4.2.1 Rezervační systém

Rezervační systém společnosti Student Agency je umístěn pouze na webu a bohužel neposkytuje žádné aplikační rozhraní, skrze které bychom k němu mohli přistupovat. Web byl také v minulosti několikrát změněn, přičemž poslední verze, která s drobnými úpravami funguje až do dnes, je založena na technologii Javascript a Ajax.

To přineslo velké problémy, jelikož pro komunikaci ze serverem je při těchto podmínkách nezbytné generovat proměnné, které se dynamicky mění a většinou je obstarává samotný prohlížeč. Typickým příkladem může být počet přístupů na stránku, kterou musí uživatel do požadavků na server také vložit. Rovněž i získání dat z přijatých HTML souborů nebylo zcela triviální.

Přechod na tuto technologii, by se dal chápat jako snaha o zamezení přístupu na stránky prostřednictvím obdobných robotů, jelikož dříve takové programy existovaly a bohužel způsobovaly přetíženost serverů. Můžeme tedy předpokládat, že se jednalo o jeden z mnoha kroků jak těmto nepříjemnostem předcházet.

4.2.2 Interval kontrol

Pokud by byla předchozí domněnka pravdivá, dá se očekávat i další obranné mechanizmy proti periodickému přístupu k systému. Proto bylo nezbytné navrhnout kontroly tak, aby byl přístup co možná nejméně podezřelý. Z toho důvodu jsem se rozhodl interval mezi jednotlivými kontrolami volit následovně.

Celkovou velikost intervalu získáme odečtením 15 vteřin od standardního intervalu voleného uživatelem, k němuž nakonec přičteme náhodně generovaný počet vteřin generovaný z intervalu 0 a 30.

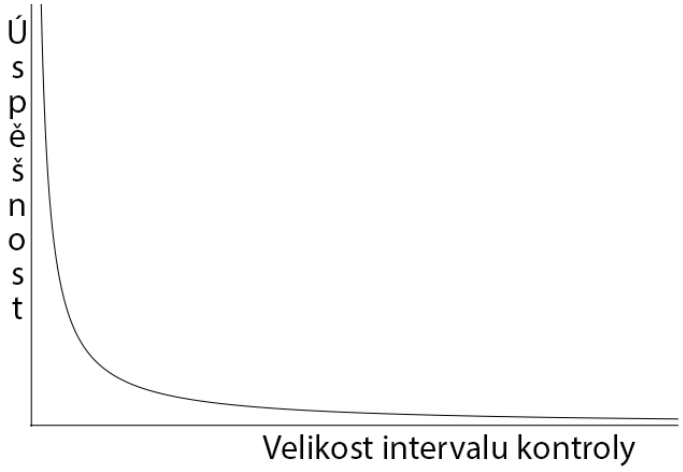
$$t = T - 15 + ran(0{,}30)$$

Ve výsledku tak pro nastavení 1 minuta může být interval mezi dvěma kontrolami proveden nejdříve po 45 vteřinách a nejpozději po 75 vteřinách. Při takovémto dotazování nemůže server najít pravidelný vzorec v přístupu a odhalit tak periodickou kontrolu.

Předpokládal jsem také, že uživatelé budou aplikaci využívat mimo své domovy a k serveru se připojovat prostřednictvím mobilní sítě, která je omezena FUP limitem.

Proto jsem se rozhodl do aplikace implementovat možnost zvolit si interval kontroly T, o kterém jsem již zmiňoval při předchozím vysvětlování.

Změna velikosti intervalu bude mít však za následek změnu pravděpodobnosti nalezení volného místa. V případě že uživatel zvolí nejkratší interval mezi jednotlivými kontrolami, bude mít úspěšnost při kontrole volných míst větší než uživatelé, kteří zvolí interval větší, podle následujícího grafu.



Graf 12 Závislost úspěchu hlídání na délce intervalu

Aplikace umožňuje nastavit jako nejkratší dobu 1 minutu a jako nejdelší 15 minut, přičemž pravděpodobnost při maximální době je velmi nízká, a doporučuje se tak využít opravdu pouze jen u spojů s menším zájmem.

4.2.3 Kontrola volných míst

Ve chvíli kdy, je do tabulky hlídaných spojů přidána první položka, aplikace zavolá službu, ve které je spuštěn asynchronní proces, kde probíhá cyklická kontrola míst. Po přidání dalších spojů k hlídání je pouze upraven seznam spojů v asynchronním procesu.

Jelikož proces běží v samostatném vlákně, mohlo by docházet k situacím, kdy dva procesy přistupují ke sdíleným datům ve stejnou dobu a způsobí tak nežádoucí chování programu, které by mohlo vézt k chybě. Jako eliminaci tohoto problému jsem proto implementoval zámky nad těmito sdílenými daty, které v jednom okamžiku zaručují výlučný přístup.

V případě úspěšného nalezení místa je místo zarezervováno a zároveň je o výsledku informován uživatel upozorněním. Pokud bylo nalezené místo nalezeno u posledního spoje v tabulce, asynchronní proces je ukončen.

4.2.4 Automatická rezervace

Automatická rezervace u této aplikace je nezbytná. Původním záměrem bylo nabídnout tuto možnost uživatelům pouze za příplatek, ale bohužel jsem však po provedené anketě přímo na stanovištích Student Agency zjistil, že drtivá většina dotazovaných není navyklá za aplikace platit a aplikace bez automatické rezervace by si nestáhli. Jako důvod uvedli většinou neochotu pro instalování dalších aplikací pro rezervaci a strach z pozdního zareagování.

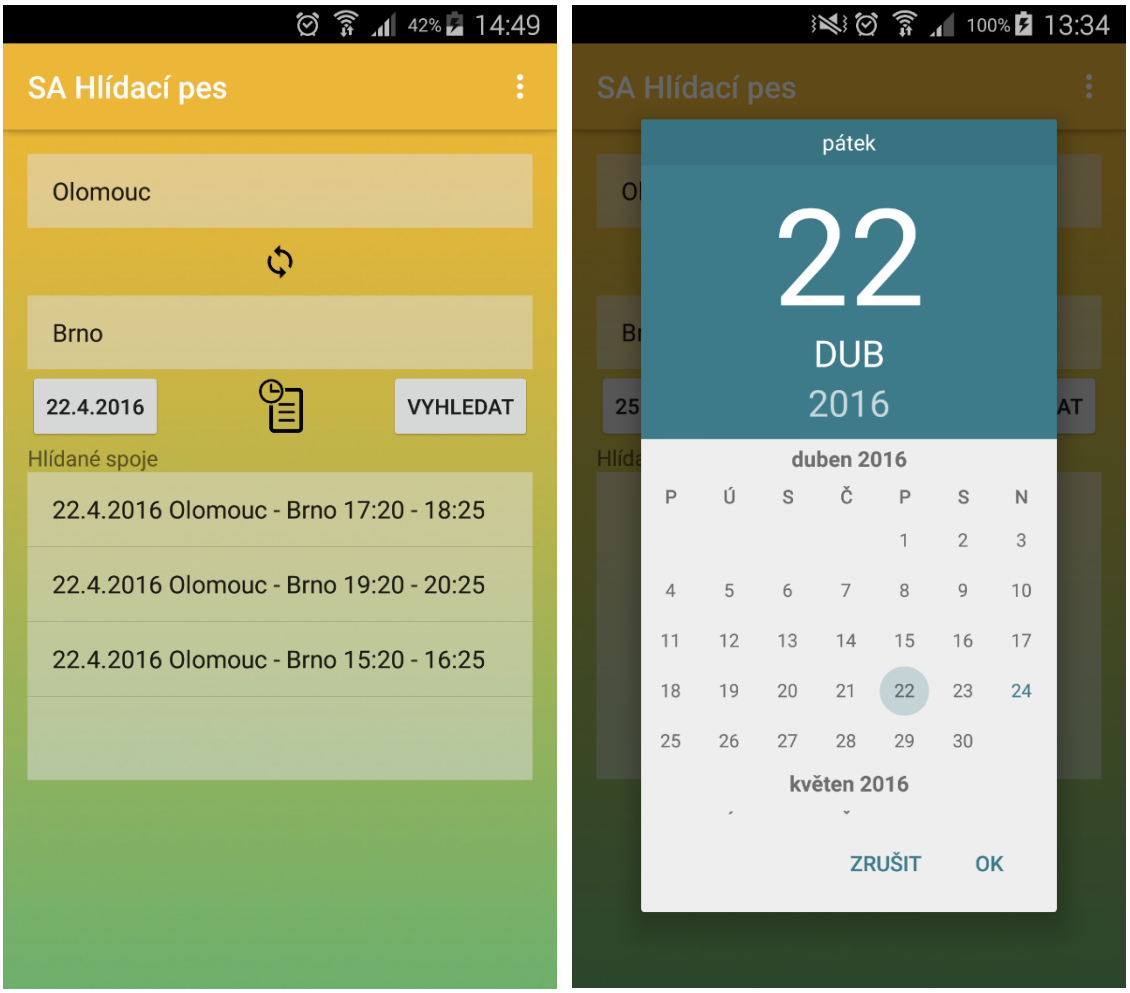
Automatická rezervace je tedy dostupná pro všechny uživatele po zadání čísla kreditové jízdenky, hesla a tarifu, v rámci kterého chtějí jízdenky rezervovat. Tyto informace jsou nezbytné pro to, aby mohla aplikace sama místo zarezervovat.

4.2.5 Grafické rozhraní

Jak jsem předeslal již v návrhu, aplikace se bude sestávat z dvou hlavních aktivit. První bude hlavní okno, které se otevře ihned po spuštění a na kterém se nachází většina funkcionalit. V horní části můžeme najít název aplikace v levé části a v pravé části rozbalovací menu, které se ukrývá pod symbolem svislých teček.

Níže je umístěn výběr místa odjezdu a cíle s možností vzájemné výměny. Výběr místa je realizován za pomoci rozbalovacího seznamu a hodnoty, které jsou standardně zobrazovány po spouštění aplikace, lze uživatelsky nastavit a zajistit tak usnadnění pro uživatele. V praxi totiž většina uživatelů cestuje neustále mezi stejnými městy.

Pod výběrem měst se nachází řádek s třemi funkčními prvky. V levé části je tlačítko, které umožňuje výběr data pro hledání spoje, přičemž aktuální vybrané datum je uvedeno na tlačítku. V případě spuštění aplikace je vždy nastaveno datum následujícího dne. Po té co uživatel stiskne tlačítko, se otevře kalendář s možností výběru konkrétního data cesty, jak lze vidět na následujícím obrázku.

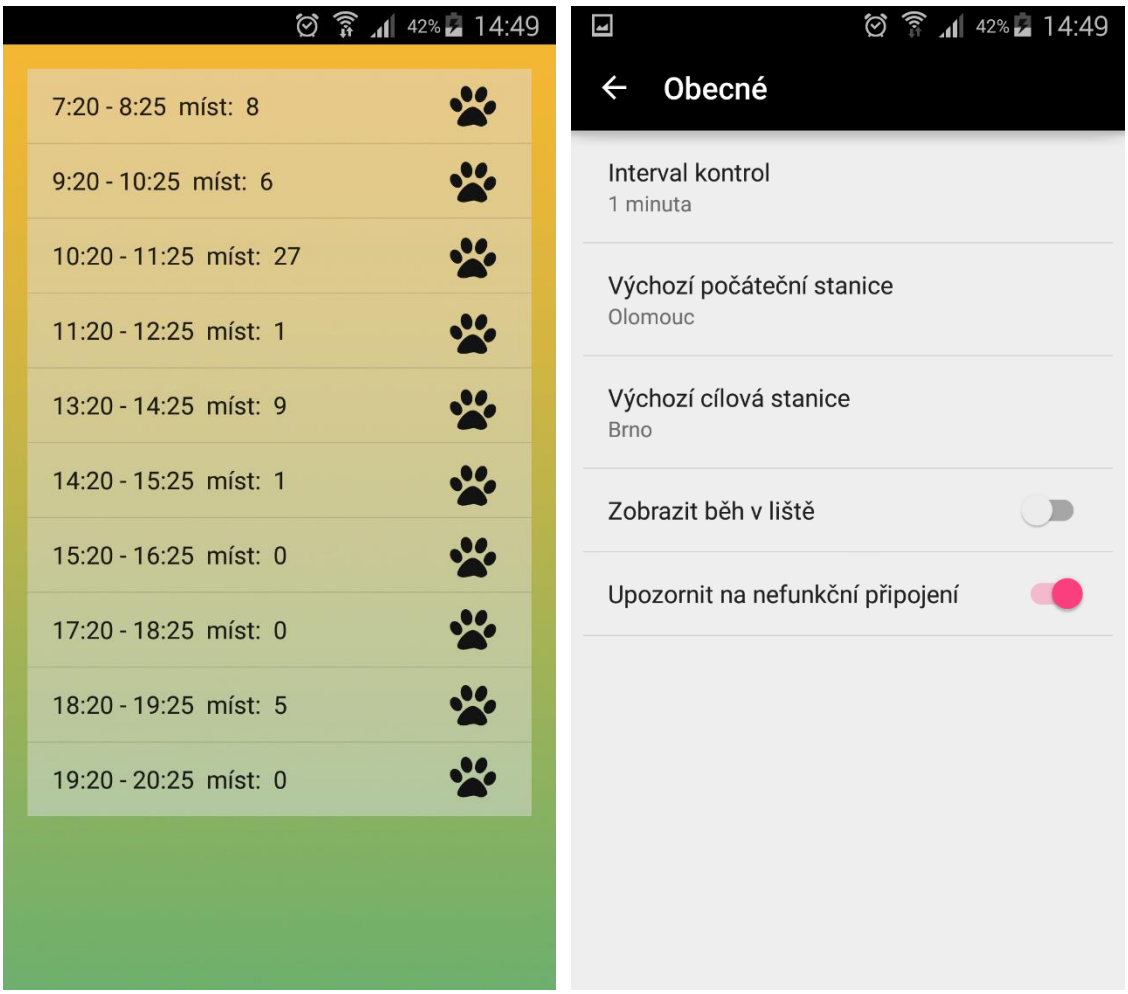


Obrázek 20 Grafické rozhraní hlavní obrazovky a výběru data

Vedle výběru data pak vidíme ikonu listu s hodiny, prostřednictvím které se dostaneme do historie hlídaných spojů. Zde lze najít pouze spoje, které jsou z tabulky hlídaných jízd odebrány programem a to buď úspěšným nalezením místa, nebo automatickým ukončení hledání 20 min před dobou odjezdu, neboť se nepředpokládá, že by některý uživatel stornoval jízdenku v době, kdy ji musí zaplatit. V historii tak nelze nalézt spoje, které byly z hlídání odebrány ručně uživatelem. Úplně vpravo, se na stejném řádku nachází tlačítko, které pro stávající konfiguraci vyhledá příslušné spoje, ty se zobrazí na druhé aktivitě, která bude popsána později.

Po výběru konkrétního spoje ze seznamu na druhé obrazovce je uživatel opět přesměrován zpět, kde se spoj přidá do tabulky hlídaných spojů. Zde má uživatel přehled o všech hlídaných spojích dle pořadí, v jakém je přidal. Ke každému spoji jsou zobrazeny pouze základní informace a to konkrétně datum, destinace a čas jízdy. Pokud se uživatel

rozhodne přestat hlídat některý spoj, docílí toho snadno obyčejným poklepáním na položku v seznamu.



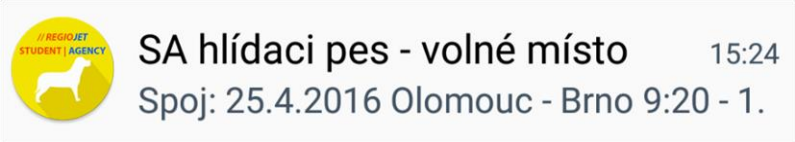
Obrázek 21 Grafické rozhraní seznamu spojů a nastavení

Na levém obrázku výše je zobrazeno druhé hlavní okno aplikace, které se zobrazí při vyhledání spojů. Pro každý spoj jsou opět uvedeny pouze základní informace a to konkrétně čas odjezdu a příjezdu a aktuální počet volných míst ve spoji. Přidání spoje ke hlídání pak uživatel může provést kliknutím na značku tlapy u každé položky v seznamu. Pokud se chce uživatel vrátit zpět na předchozí obrazovku bez výběru spoje, provede tak stisknutí tlačítka zpět ve svém telefonu. Volné místo dole u obou hlavních obrazovek aplikace je v rozložení umístěno záměrně. Právě zde se bude následně po stažení uživatelem zobrazovat reklamní sdělení.

Dalšími obrazovkami grafického rozhraní v aplikaci jsou stránky nastavení aplikace, ty jsou tvořeny automaticky programem Android studio podle programátorem dané

struktury. Na obrázku č. 21 můžeme například vidět stránku obecného nastavení, kde uživatel volí velikost intervalu kontrol, výchozí stanice či zda zobrazovat pomocná upozornění.

Upozornění uživatele probíhá standardním způsobem, jak jsou uživatelé Android zařízení zvyklí. Je doprovázeno zvukovým signálem a rovněž je nastavena informační dioda telefonu. Tyto dva prvky si může uživatel přizpůsobit dle svého, tedy zvolit si tón a barvu diody, která má být použita. Při rozbalení notifikační lišty jsou zobrazeny základní informace o spoji spolu s ikonou a názvem aplikace. Příklad lze vidět na následujícím obrázku.

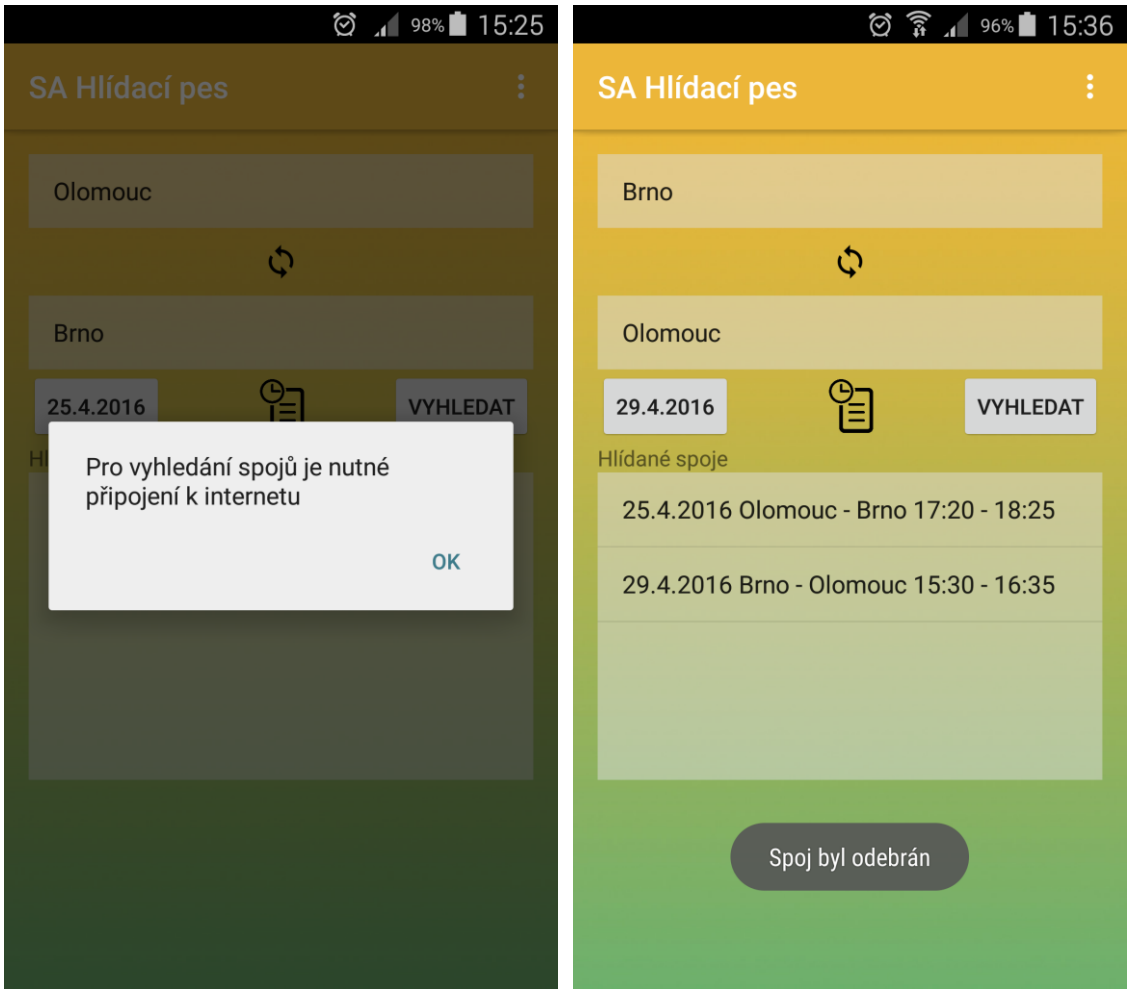


Obrázek 22 Upozornění aplikace

Další způsoby předávání informací uživatelům jsou znázorněny na následujících obrázcích. V případě, že se jedná o informaci, chybě či stavu, na který musí uživatel reagovat a vyřešit, jsou informace předávány formou vyskakovacího dialogového okna. Uživateli je tak zamezena veškerá činnost aplikace a je nucen si zprávu přečíst a potvrdit. Teprve poté může pokračovat v práci. Tento způsob je využit například při nefunkčním připojení k internetu či špatném zadání vstupních údajů pro vyhledání spojů.

Dalším způsobem, velmi rozšířeným na chytrých zařízeních, je takzvaný toast. Ten umožňuje zobrazit na displeji informaci po dobu několika sekund a zároveň nijak neovlivňuje funkcionalitu aplikace. Nevýhodou těchto zpráv je vysoká pravděpodobnost, že uživatel zprávu přehlédne, proto se toast používá převážně pro informativní účely, na které uživatel nemusí nijak reagovat. V mém případě za jejich pomoci například oznamuji uživateli, že spoj, který si on sám vybral, byl přidán či odebrán ze seznamu hlídání a při podobných operacích.

Grafické rozhraní jsem se snažil vytvořit tak, aby bylo intuitivní pro uživatele, a aby splňovalo základní předpoklady, na které jsou zvyklí z ostatních aplikací. Zároveň jsem se snažil o zachování jednoduchosti. Po doporučení profesionálních vývojářů jsem rovněž aplikaci navrhl pouze v portrait rozložení tedy na výšku.



Obrázek 23 Ukázka zpráv pro uživatele

4.2.6 Příjem

Původním záměrem bylo rozdělit funkcionalitu do dvou verzí. V první verzi, která by byla k dispozici zadarmo, by bylo možné pouze upozornit na uvolněné místo, nikoli jej však zarezervovat. Rezervace by byla poté možná pouze v aplikaci za poplatek. S ohledem na informace získané z průzkumu, jsem se nakonec rozhodl veškerou funkcionalitu zpřístupnit zdarma a do aplikace umístit reklamy.

Toto řešení se zdá vhodné i z toho důvodu, že k funkci aplikace je nezbytné připojení k internetu, což je nutné pro korektní zobrazování reklamy v aplikaci.

4.2.7 Prezentace aplikace

V kapitole analýzy jsem rovněž zmiňoval důležitost prezentace aplikace tudíž i jejích hlavních poznávacích znaků. Právě na ty se zaměřím v této části.

Název

Jak již vyplynulo z analýzy, volba názvu aplikace je jedním z klíčových prvků, které mají vliv na její úspěšnost na trhu. Moje aplikace se zabývá samostatnou kontrolou volných spojů, tedy hlídáním volných míst. Z toho důvodu jsem se rozhodl pro název aplikace Hlídací pes. Tento pojem je mezi uživateli již dobře znám a moje aplikace splňuje jeho základní myšlenky, tedy že pracuje samostatně a v případě objevení hlídané události upozorní majitele.

Dále jsem chtěl do názvu zakomponovat název dopravce, jehož spoje uživatelé využívají a poznají tak, že je aplikace určená přímo jim. V tomto bodu jsem narazil na menší zádrhel, jelikož společnost již od svého počátku vystupuje pod názvem Student Agency, webový rezervační formulář lze najít pod doménou www.studentagency.cz a autobusy jsou označeny stejným názvem. Nicméně vlaková spojení jsou provozována pod názvem RegioJet a společnost nyní oznámila, že autobusy ponesou stejné jméno.

Otázkou pro mne tedy bylo, zdali do názvu vložit zkratku SA nebo RJ. Proto jsem se rozhodl udělat menší průzkum, co si představí uživatelé pod těmito dvěma zkratkami a došel k závěru, že více lidí má spojeno SA se Student Agency než RJ s RegioJet. Celý název aplikace tedy bude SA hlídací pes.

Ikona

Ikonu jsem se rozhodl vytvořit vlastní a při jejím návrhu jsem chtěl respektovat znalosti získané v kapitole analýzy, tedy to, že by aplikace měla být dobře zapamatovatelná, reprezentovat smysl aplikace a v menu by měla být dobře viditelná.

Jako první věc jsem chtěl použít poznávací barvu společnosti žlutou. Proto je pozadí kolorováno právě do této barvy. V horní části jsem dále umístil názvy společnosti, jejichž spoje aplikace hlídá. Jak jsem zmínil u názvu aplikace, společnosti v současné době využívá oba názvy, tedy Student Agency i RegioJet. V ikoně jsem se rozhodl oba tyto názvy zobrazit a oslovit tak více uživatelů na trhu.

Pod tyto názvy jsem chtěl umístit něco, co by symbolizovalo smysl aplikace. Nakonec jsem se rozhodl použít stejnou symboliku jako v názvu aplikace, psa. Do dolní části jsem tedy umístil stínovanou siluetu hlídacího psa, která symbolizuje proces hlídání. Výslednou ikonu lze vidět na dalším straně.

Obrázek 24 Ikona aplikace

4.3 Testování

Jelikož jsem pravidelným cestujícím společnosti na pravidelné autobusové lince Brno – Olomouc, která je z důvodu absence vlakových spojů velmi vytížená, testování aplikace probíhalo od počátku vývoje pravidelně každý týden. Díky tomu byla odhalena velká spousta skrytých chyb, které by za normálních okolností číhaly na první uživatele a já jako vývojář bych musel spoléhat na jejich nahlášení.

Jako hlavní zařízení pro testování byl používán Samsung Galaxy S5 s Android verzí 5.0 jakožto zástupce současných moderních telefonů s Full HD rozlišením a Samsung Fame s Android verzí 4.1.2, který odpovídá zařízení s nízkým výpočetním výkonem a rozlišením displeje.

Finální verzi aplikace jsem rovněž testoval na širším spektru zařízeních dostupných v mém okolí, které pomohly odhalit další chyby týkající se převážně grafického rozhraní aplikace.

4.4 Ekonomické zhodnocení

Aplikace SA hlídací pes nevzešla z požadavků žádné organizace. Byla mým vlastním projektem a její implementaci tudíž nikdo finančně nepodporoval. Z mé strany jsem do aplikace investoval pouze svůj čas, od kterého se můžeme odrazit. Odhadem jsem pouze nad implementací a testováním strávil přibližně 60 pracovních hodin. V případě, že bych tento čas chtěl přepočítat na částku pomocí stejného platu, který mám jako SW vývojář junior v soukromém sektoru, dostal bych se na částku kolem 7 000 Kč. Aby však aplikace byla rentabilní, musím ji umístit na trh. Jelikož jsem začínající vývojář, musel jsem si založit vývojářský účet na Google Play a zaplatit registrační poplatek, který činí přibližně 600 Kč (25 dolarů).

V současné době bohužel ještě nemohu objektivně zhodnotit finanční přínos aplikace, z důvodu nedostatečně dlouhé doby na trhu. Nicméně jsem si však jistý návratností registračního poplatku a drobným pravidelným příjmem ze zobrazované reklamy v aplikaci. Vzhledem k nízkým nárokům na údržbu jsem si jist, že se takto zhodnotí i můj investovaný čas do implementace.

4.5 Přínosy práce

Jelikož se práce zabývá dvěma tématy, rozhodl jsem se provést zhodnocení přínosu rovněž ze dvou různých pohledů. V první řadě zde rozeberu přínosy vyplývající z analýzy trhu aplikací Android a následně přínosy vývoje vlastní aplikace.

Přínosy vyplývající z analýzy byly sice využity při návrhu vlastní aplikace, ale jejich využití pro další vývojáře je neomezená. Poskytuje dobrý odrazový můstek pro všechny začínající programátory a upozorňuje na prvky, které bývají problematické a mohou tak vézt k neúspěchu. Čtenáři této práce se mohou rovněž seznámit s nejčastěji zákazníky vyžadovanými standardy, díky analýze recenzí u již zaběhlých aplikací.

Přínos samotné aplikace je pak zřejmý nejen na straně uživatelů, ale i vývojáře. Jak jsem již zmínil dříve, aplikace napomáhá řešit problémy s rezervací spojů, což je přínosem nejen pro cestující, ale i pro společnost jako takovou, neboť aplikace napomáhá snižovat riziko neobsazení všech míst ve spoji. Z mé strany jsou největším přínosem této práce získané zkušenosti při vývoji a přehled o praktikách, které vedou k zvýšení úspěšnosti aplikace. V budoucnu (po rozšíření mezi uživatele) bude dalším přínosem aplikace zisk ze zobrazovaných reklam.

4.6 Možnosti rozšíření

Při vývoji jsem na aplikaci pohlížel s jasným cílem zaměřit se na splnění hlavních požadavků uživatelů a zvládnutí klíčových oblastí se zajištěním celkové stability. Z toho důvodu je zde spoustu prvků, na které se lze v rámci budoucího vývoje zaměřit a posunout tak aplikaci na vyšší úroveň.

Jako další funkcionalitu by u aplikace uživatelé uvítali možnost prohlížet si již zarezervované spoje. Naše aplikace sice nabízí zobrazení historie rezervací, ale ta nemusí vždy korespondovat s aktuálním stavem na příslušném účtu. Proto by bylo vhodné v budoucnu přidat možnost nahlížet do aktuálních rezervací, které by byly vždy

stahovány z účtu. Bohužel i tento způsob by musel být realizován skrze parsování HTML kódu.

Nákladově náročnější změnou však bude realizace myšlenky optimalizace kontroly hlídání spojů za pomoci vlastního serveru. Jsem si vědom, že by při masovém rozšíření aplikace mohlo docházet k nežádoucímu přetěžování serverů Student Agency, které by mohlo vést k nefunkčnosti aplikace. Tento fakt by mohl být snadno vyřešen díky vlastnímu aplikačnímu serveru, který by jako jediný kontroly prováděl. V případě, že by uživatel zadal spoj ke hlídání, vyslal by požadavek na server, který by tento spoj začal periodicky kontrolovat. Aplikace v chytrých zařízeních by se následně dotazovala vlastního serveru a pro jeden spoj by tak docházelo ke kontrole pouze z jednoho serveru. Po odhalení volného místa serverem bude pozitivní výsledek vrácen prvnímu dotazovanému zařízení. Následná registrace by pak probíhala stejným způsobem jako u původní aplikace, tedy přímo ze zařízení. Uživatel by se tak nemusel bát ukládání citlivých údajů na cizím serveru, kde by mohlo dojít ke zneužití. Tímto krokem by došlo nejen ke zmírnění zatížení serveru Student Agency a zároveň k snížení objemu přenášených dat do telefonu, což může být pro majitele mobilního připojení dobrou zprávou.

V neposlední řade bych se chtěl v budoucnu zaměřit na grafické rozhraní aplikace. Během implementace bylo primárním cílem odladit funkcionalitu aplikace a grafická stránka bohužel nebyla dotažena k dokonalosti. Částečně k tomu i přispěla absence mých zkušeností z vývojem front-endu a práci v grafických programech.

V každém případu je nezbytné počítat se slabinami, které provází parsování HTML obsahu. V případě sebemenších úprav na webu společnosti může dojít k ohrožení funkčnosti aplikace. Proto by bylo vhodné implementovat určitý systém, který na změnu samovolně upozorní a bude tak možné reagovat co nejdříve a zamezit tak ztrátě zákazníků.

Závěr

Cílem této práce bylo analyzovat trh mobilních aplikací na platformě Android a poté vypracovat návrh a implementovat vlastní aplikaci. Myšlenkou pro vznik aplikace bylo automatizované hlídání volných míst v dopravních spojích u společnosti Student Agency, a jejich následná rezervace.

Po seznámí se základními teoretickými poznatky, jsem provedl důkladnou analýzu trhu mobilních aplikací. Na data jsem nahlížel z více pohledů a snažil jsem se objasnit všechny témata, která stejně jako mě, vyvstávají v hlavě začínajícím vývojářům při tvorbě prvních aplikací. U většiny témat bohužel neexistuje pouze jedno správné řešení. V práci proto popisuji, do jakých situací se nejlépe hodí dané postupy. Tato část byla velmi časově náročná, z důvodů získávání relevantních podkladů a statistik o trhu, které většinou nebývají veřejně přístupné.

V další části práce se potom zabývám samotným vývojem aplikace. Nejprve jsem provedl samotný návrh, který vycházel nejen z poznatků zjištěných při analýze, ale rovněž dle požadavků potencionálních uživatelů v mém okolí. Jako hlavní formu návrhu jsem zvolil diagramy modelovacího jazyka UML a jejich slovní popis. Samotná implementace následně probíhala v Android Studiu, což je oficiální vývojové prostředí pro vývoj těchto aplikací, za použití jazyka Java. Nejtěžší částí implementace byla jednoznačně komunikace s webovým portálem společnosti bez použití API. Spojení bylo nezbytné realizovat jako klasický přístup skrze grafické rozhraní, což přineslo spoustu nadbytečného práce v podobě parsování stránek, které jsou běžně uživateli zobrazeny.

Všechny komplikace vzniklé během vývoje se však podařilo vyřešit a vznikla tak aplikace, která může zbavit spoustu uživatelů starostí spojených s organizováním jejich cest. Zároveň byla práce velkým přínosem i pro mne, jelikož mi pomohla v prohloubení znalostí vývoje mobilních aplikací a jazyku Java. V budoucnu bych se rád zaměřil na rozšíření aplikace mezi uživatele a na dalším vývoji pro mobilní zařízení.

Literatura

1. ČÁPKA, D. Úvod do jazyka Java. *IT network.* [Online] 21. 1 2014. [Citace: 27. 1 2016.] http://www.itnetwork.cz/java/zaklady/java-tutorial-uvod-do-jazyka-java/.

2. VÁVRŮ, J. a UJBÁNYAI, M. *Programujeme pro Android 2., rozšířené vydání.* Praha : Grada, 2013. str. 256. ISBN 978-80-247-4863-4.

3. GARGENTA, M. *Learning Android.* Sebastopol : O'Reilly, 2011. str. 245. ISBN 14-493-9050-1.

4. Android Fragmentation Visualized. *OpenSignal.* [Online] 1. 8 2015. [Citace: 17. 2 2016.] http://opensignal.com/reports/2015/08/android-fragmentation/.

5. MEIER, R. *Professional Android 2 Application Development.* Indianapolis : Wiley Publishing, Inc.,, 2010. str. 576. ISBN 978-0-470-56552-0.

6. SMYTH, N. *Android Studio Development Essentials.* místo neznámé : eBookFrenzy, 2015. str. 614. ISBN 978-09-86-0273-76.

7. VONDRÁK, I. Úvod do softwarového inženýrství. *Technická Univerzita Ostrava.* [Online] 2002. [Citace: 17. 2 2016.] http://vondrak.cs.vsb.cz/download/Uvod_do_softwaroveho_inzenyrstvi.pdf.

8. SEMECKÝ, V. Android Studio – nové vývojové prostředí. *Zdroják.* [Online] 4. 11 2013. [Citace: 1. 2 2016.] https://www.zdrojak.cz/clanky/android-studio-nove-vyvojove-prostredi/.

9. Application Fundamentals. *Developers.* [Online] [Citace: 1. 3 2016.] http://developer.android.com/guide/components/fundamentals.html.

10. ARLOW, J. *UML a unifikovaný proces vývoje aplikací: průvodce analýzou a návrhem objektově orientovaného softwaru.* Brno : Computer Press, 2003. str. 387. ISBN 80-7226-947-X.

11. Úvod do UML. *IT Network.* [Online] [Citace: 3. 3 2016.] http://www.itnetwork.cz/navrhove-vzory/uml/uml-uvod-historie-vyznam-a-diagramy/.

12. KANISOVÁ, H. *UML srozumitelně.* Brno : Computer Press, 2004. str. 158. ISBN 80-251-0231-9.

13. ZENDULKA, J. *Jazyk UML (Unified Modeling Language).* Brno : VUT. Projektování programových systémů.

14. ČÁPKA, D. UML - Doménový model. *IT Network.* [Online] 20. 10 2012. [Citace: 2. 2 2016.] http://www.itnetwork.cz/navrhove-vzory/uml/uml-domenovy-model-diagram.

15. REJNKOVÁ, P. Diagram aktivit. *UML.* [Online] 2009. [Citace: 3. 2 2016.] http://uml.czweb.org/diagram_aktivit.htm.

16. GRASSEOVÁ, M. *Analýza podniku v rukou manažera: 33 nejpoužívanějších metod strategického řízení.* Brno : Computer Press, a.s., 2010. str. 325. ISBN 978-80-251-2621-9.

17. SEDLÁČKOVÁ, H. *Strategická analýza.* Praha : C. H. Beck, 2006. str. 121. ISBN 80-7179-367-1.

18. MUZIKANT, V. Podnikatelský záměr pro sociální firmu. *Docplayer.* [Online] [Citace: 1. 3 2016.] http://docplayer.cz/1371787-Podnikatelsky-zamer-pro-socialni-firmu.html.

19. Google I/O 2015 Special Report: Google Play's Unstoppable Growth. *App Annie.* [Online] 9. 1 2016. [Citace: 19. 1 2016.] http://blog.appannie.com/google-io-2015-special-report/.

20. Wilcox, M. Are you using the right app revenue model? *Developer Economics.* [Online] 19. 11 2013. [Citace: 15. 12 2015.] http://www.developereconomics.com/apps-using-right-revenue-models/.

21. Freemium. *Management Mania.* [Online] 20. 5 2013. https://managementmania.com/cs/freemium.

22. TRIGGS, ROB. Freemium models and in-app ads spur growth in mobile app revenue. *Android Authority.* [Online] 6. 4 2015. [Citace: 15. 12 2015.] http://www.androidauthority.com/freemium-app-growth-idc-598851/.

23. Mobile Games: Now You Can Predict the Future. *App Annie.* [Online] 27. 10 2015. [Citace: 8. 1 2016.] http://blog.appannie.com/app-adoption-cycle/.

24. Joseph C. Nunes, Xavier Dreze. The Endowed Progress Effect: How Artificial. *Journal of consumer Research, Inc.* 1. 3 2006, 32.

25. Puzzle Brain Stimulator. *Google Play*. [Online] [Citace: 6. 1 2016.] https://play.google.com/store/apps/details?id=cz.kvasoon.brainstimulator.android.

26. Burrows, Roland. Lessons Learned From 100 Negative App Reviews. *App Annie.* [Online] 24. 11 2014. [Citace: 6. 12 2015.] http://blog.appannie.com/lessons-learned-from-100-negative-app-reviews/.

27. Bartov, S. Why Going Global Can Do Wonders for Your App. *Inside AdMob.* [Online] 10. 6 2015. [Citace: 8. 1 2016.] http://admob.blogspot.cz/2015/06/infographic-going-global.html.

28. Global Mobile Data Consumption Trends: A Special Report for Operators & OEMs. *App Annie.* [Online] 10. 11 2015. [Citace: 8. 1 2016.] http://blog.appannie.com/global-mobile-data-consumption-trends-special-report-operators-oems/.

29. Mari, A. Public Wi-Fi provision in Brazil takes off. *ZD Net.* [Online] 31. 8 2015. [Citace: 9. 1 2016.] http://www.zdnet.com/article/public-wi-fi-provision-in-brazil-takes-off/.

Seznam obrázků

Seznam grafů