Univerzita Hradec Králové

FAKULTA INFORMATIKY A MANAGEMENTU

Katedra informačních technologií

Návrh a realizace řešení pro využití Smart mobilních zařízení jako desktopových počítačů

Diplomová práce

Autor: Bc. Pavel Košťál

Studijní obor: IM2-K

Vedoucí práce: prof. Ing. Ondřej Krejcar, Ph.D.

Hradec Králové Duben 2020

Prohlášení

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

V Hradci Králové dne 20.04.2020 Bc. Pavel Košťál

Poděkování

Děkuji prof. Ing. Ondřeji Krejcarovi Ph.D. za pomoc při vytváření diplomové práce.

Anotace

Anotace

Tato diplomová práce se zabývá návrhem řešení aplikace pro adaptaci mobilního zařízení pro desktopový provoz. Cílem práce bylo vytvořit aplikaci, díky které je možno jakýkoliv telefon s mobilním operačním systémem Android používat jako částečnou náhradu stolního počítače. Funkcionalita spočívá v klonování obrazovky telefonu na monitor nebo televizi a k ovládání je použita bluetooth myš a klávesnice. Přenos obrazu je řešen bezdrátově nebo pomocí videokabelu u telefonů, které tuto technologii podporují. Naprogramovaná aplikace funguje jako launcher, tedy aplikace, která se spustí po startu telefonu nebo při zmáčknutí domovského tlačítka.

Annotation

This graduation thesis seeks to draft an application adapting mobile devices to desktop operation. The aim was to develop an application that enables using any phone with Android mobile operating system as a partial substitute for a desktop computer. Its functionality is based on cloning a phone screen onto a computer or television screen, using a Bluetooth mouse and a keyboard to control these. Images can be transmitted either wirelessly or, if your phone supports the technology, with a video cable. The programmed application works as a launcher, i.e. an application that launches after the phone is switched on or the home button is pressed.

Obsah

Seznam obrázků:

Seznam tabulek:

Seznam zkratek:

ADB	Android Debug Bridge
API	Application programming interface
ART	Android Runtime
DP	Density-independent
DPI	Dots per inch
DSL	Domain Specific Language
DVM	Dalvik Virtual Machine
GPS	Global Positioning System
ORM	Object-relational mapping
JDK	Java Development Kit
JIT	Just-In-Time
JVM	Java Virtual Machine
LDAP	Lightweight Directory Access Protocol
LIFO	Last In First Out
MVC	Model-view-controller
REST	Representational State Transfer
SP	Scale-independent Pixels
SSL	Secure Sockets Layer
TWRP	Team Win Recovery Project
USB	Universal Serial Bus
XML	Extensible Markup Language
WM	Window Manager

1 Úvod

V současné době mají chytré mobilní telefony dostatečný výkon pro provádění velmi složitých operací a zároveň mnoho těchto telefonů má digitální výstup přes HDMI, pomocí něhož lze připojit telefon k monitoru nebo k televizi a klonovat obraz. Klávesnici a myš lze jednoduše připojit k telefonu pomocí bluetooth technologie, zařízení s podporou bluetooth je na trhu již dostatečné množství. Díky velkému množství aplikací dostupných přes aplikaci Google Play je možno v současné době na mobilním telefonu dělat vše možné, například psaní dokumentů ve Wordu, editování tabulek v Excelu, upravování obrázků v Adobe Photoshop Express následovány dalšími běžnými funkcemi, jako jsou poslouchání hudby, sledování filmů, prohlížení internetu, pořizování fotek nebo navigace. V neposlední řadě to je hraní různých her, které je mezi lidmi velmi populární, například hra Candy Crush Saga vydělala v roce 2019 částku 67,58 milionu amerických dolarů (Gough, 2019).

Použít komerčně telefon jako desktopový počítač napadlo jako první využívat firmu Samsung, která představila tento produkt v telefonu Galaxy S8. Řešení se jmenuje Samsung Dex a je použito pouze u nejlepších modelů od firmy Samsung, tedy Galaxy S8, S9, S10, S20, Note 8, Note 9, Note 10 a tablety Galaxy Tab S4, S5 a S6 (Jansen, 2019). Další výrobce, který do svých nejlepších telefonů dal možnost používat desktopové prostředí, je firma Huawei. Toto prostředí se jmenuje Huawei EMUI Desktop a má stejné vlastnosti, jako má Samsung Dex a také je velmi podobné desktopu, který zná většina lidí z operačního systému Windows. Je zde možno mít aplikace spuštěné v okně, toto okno zvětšovat, nebo zmenšovat, mít spuštěných více aplikací najednou, pracovat v Microsoft Word a Excel a mnoho jiného.

Bohužel tuto funkcionalitu mají jen nejlepší telefony od těchto dvou firem, takže pokud člověk nemá dostatek financí nebo nerad používá výrobky firmy Samsung a Huawei, tak nemůže svůj telefon požívat jako náhradu za standardní počítač. Z toho důvodu bude v této diplomové práci představen návrh řešení aplikace pro adaptaci mobilního zařízení pro desktopový provoz, který by měl fungovat na většině mobilních telefonů včetně telefonů, které nemají video výstup, zde bude použit bezdrátový přenos pomocí technologie Google Chrome. Pro lepší přehled se naprogramovaná aplikace jmenuje Revolution Launcher.

Pro manažery by tato aplikace mohla v budoucnosti znamenat, že by vůbec nemuseli používat notebook pro pracovní cesty nebo při přechodu mezi prací a domovem. Výhoda telefonu je, že ho člověk má neustále u sebe prakticky kdekoliv, to o notebooku říci nelze. Aby telefon mohl nahradit notebook, bylo by nutné mít na mnoha místech například v hotelových pokojích něco podobného, jako je dokovací stanice pro notebooky. Z finančního pohledu to není velký

problém, protože při masivnějším využití by univerzální dokovací stanice pro telefony mohla stát několik stovek korun. Do této stanice již lze připojit standardní USB myš a klávesnici, které lze pořídit v současné době za velmi nízkou cenu. Hlavní problém je nízké povědomí lidí o možnosti využití telefonu jako desktopového počítače a také relativně nízká podpora výrobců software pro mobilní telefony, takže mnoho aplikací nemá podporu pro desktopové prostředí.
V následující části je popsáno členění jednotlivých kapitol. Kapitola návrh řešení aplikace pro adaptaci mobilního zařízení pro desktopový provoz je rozdělena do pěti částí. V první části je popsán základní postup programování pro mobilní telefony Android. Další část pojednává o programování v jazyku Java a návaznost tohoto jazyka na programování v aplikaci Android Studiu. Ve třetí části jsou popsány různé funkce nutné pro vytváření programu Revolution Launcher fungující pod operačním systémem Android, jako jsou například xml konfigurace, vytváření vzhledu, co je přesně launcher a jak se nastavuje, co jsou widgety a jak fungují, popis překládání jazyka a jiné. V následující části je uvedeno, jak funguje root oprávnění, jaké jsou úskalí použití root oprávnění a rozdíly různých způsobů root a také srovnání výhod a nevýhod takto upraveného telefonu. V poslední části je uveden přesný popis, jak lze vytvořit z telefonu alternativu desktopového počítače pomocí programů, jenž lze stáhnout v Google Play.
Kapitola nazvaná metodika vytvoření mobilního zařízení pro desktopový provoz je rozdělena do tří částí. V první části je uvedeno, jak se programuje aplikace, která bude přepínat rozlišení a počet bodů a zároveň bude fungovat jako launcher, tedy výchozí aplikace, která je spuštěna po zapnutí telefonu. Tato aplikace je pojmenována jako Revolution Launcher a řídí, jaký launcher bude výchozí, zda to bude Revolution Launcher nebo uživatelův oblíbený launcher. Ve druhé kapitole je vysvětleno, jak se mění rozlišení a jsou zde uvedeny dva možné způsoby používání aplikace Revolution Launcher. První možnost již zde byla zmíněna, tedy přepínání výchozího launcheru včetně rozlišení. Druhá možnost je pro uživatele, kteří nechtějí měnit svůj oblíbený launcher. Aplikace Revolution Launcher slouží pouze pro přepínání rozlišení a počtu bodů. V poslední kapitole je napsáno, jak se aplikace nastavuje, způsob bezdrátového připojení telefonu k televizi nebo monitoru a nakonec jsou uvedeny obrázky aplikací fungujících v desktopovém prostředí.

2 Cíl diplomové práce

Cílem této diplomové práce bylo vytvořit aplikaci pro operační systém Android, díky které bude možno používat mobilní telefon jako osobní počítač. Tato aplikace bude fungovat jako launcher, což v operačním systému Android znamená výchozí aplikaci, která je spuštěna po zapnutí telefonu a zobrazí se vždy po zmáčknutí domovského tlačítka. Při použití telefonu jako desktopový počítač se nejedná o plnohodnotnou náhradu za osobní počítač, ale díky mnoha dostupným aplikacím je možné omezeně používat v tomto režimu telefon pro běžné věci, jako je uživatel zvyklý dělat na svém počítači nebo notebooku. Základní princip spočívá v tom, že je změněn počet bodů na palec (dpi) a je přizpůsobeno rozlišení, které je vhodné pro zobrazení na velké ploše. Jako standard bylo vybráno rozlišení 1920x1080 bodů, většinou známé také jako Full HD rozlišení. Díky aplikacím, které umí změnit svůj vzhled v závislosti na rozlišení a počtu bodů, je poté možno tyto aplikace používat v desktopovém režimu.

Pro vytvoření aplikace byl použit program Android Studio a programovací jazyk Java. Aby telefon fungoval jako desktopový počítač, bylo nutno vytvořit algoritmus, který jedním stisknutím tlačítka přepne počítač do desktopového prostředí a změní rozlišení a počet bodů vhodný pro použití v tomto režimu. Stejně tak bylo důležité vytvořit velmi jednoduchý postup, jak z tohoto režimu se dostat jedním kliknutím zpět do defaultního nastavení telefonu včetně přepnutí na uživatelův oblíbený launcher. Důležité je, aby vše bylo uživatelsky přívětivé a jednoduché na ovládání.

Na začátku této diplomové práce bylo nutno uvést čtenáře do problematiky programování pro operační systém Android včetně představení programovacích jazyků, za pomoci kterých je možno vytvářet aplikace pro operační systém Android včetně jejich krátkého zhodnocení. Následně je potřeba podrobněji vysvětlit programovací jazyk Java, ve kterém je aplikace naprogramována a nakonec je důležité uvést čtenáře do problematiky root oprávnění, jaké jsou výhody a úskalí využití tohoto oprávnění.

3 Teoretický a aplikační rámec vyvíjeného řešení

Následující kapitoly pojednávají o základních informacích týkajících se programování pro operační systém Android. Dále jsou uvedeny relevantní informace o programování v jazyku Java. V další části jsou již specifické informace o funkcionalitách a kódech psaných přímo pro operační systém Android v jazyku Java. V poslední části je rozebrána problematika provozu aplikací s root oprávněním.

3.1 Základní informace o programování pro operační systém Android

Operační systém Android vznikl v roce 2003 firmou Android Inc. V roce 2005 byl tento systém koupen firmou Google. Dne 23. září 2008 byl vydán Android 1.0, který měl verzi API 1. Výraz API pochází z angličtiny a znamená „Application program interface“, jedná se o rozhraní pro programování aplikací, které obsahuje různé procedury, funkce či knihovny. První telefony s operačním systémem Android, které se v České republice prodávaly, měly operační systém Android Android 1.6 Donut - API verze 4 a začaly se prodávat v roce 2009 (Šantora, 2016). V současné době je nejmodernější verze Android 10, jenž má API verzi 28.

Při vytváření nové aplikace si musí programátor zvolit, jakou verzi API chce používat. Při vytvoření nové aplikace je možno v Android Studiu zvolit verzi API 15 (Android 4.0.3 – Ice Cream Sandwich). V případě zvolení nejnižší možné verze bude vytvořená aplikace běžet na všech telefonech, které mají Android ve verzi 4.0.3 nebo vyšší. Nevýhoda zvolení nízké verze API spočívá v nemožnosti využití novějších funkcí v dalších verzích operačního systému, jako je například Android auto, který vyžaduje API verzi 21 (*Android for Cars overview*, 2019) nebo Android Automotive – jedná se o operační systém Android, používaný v osobních automobilech. Tento systém vyžaduje API verzi 28 (Dissanaike, 2019). V současné době pro publikování aplikace v Google Play je nutno mít API verzi 28, tedy Android 9, nižší verze již nelze na Google Play nahrát (Davenport, 2019).

Dne 16. května 2013 byl představen na konferenci Google I/O program Android Studio založený na prostředí IntelliJ IDEA. Tato aplikace je dostupná pro operační systémy Windows, MAC, Linux a Chrome OS a je zdarma (Ducrohet, 2013). Před uvedením této aplikace se programovalo převážně v aplikaci Eclipse s pluginem pro Android.

Pro operační systém Android lze programovat v mnoha jazycích. V současné době je nejvíce používaným jazykem pro programování jazyk Java. Programátory druhým nejvíce používaným jazykem je jazyk Kotlin, který v roce 2017 byl představen na akci Google I/0 jako podporovaný jazyk v aplikaci Android Studio. V květnu 2019 firma Google ohlásila, že jazyk

Kotlin je preferovaným jazykem pro vývoj aplikací pro Android (Lardinois, 2019). Dalším jazykem, který lze použít pro programování, je C++, největší výhoda je stejná nebo větší rychlost běhu aplikací napsaných v jazyku C++, lze použít skoro všechny kódy napsané v jazyku C, obecně ale tento jazyk není k vývoji Android aplikací široce používán. Vývoj v C++ využívají nejčastěji programátoři, kteří mají dlouholeté zkušenosti s tímto jazykem. Největší nevýhoda tohoto jazyka je, že všechny dostupné knihovny pro operační systém Android s tímto jazykem nespolupracují (Bolton, 2016). Dalším populárním jazykem je Flutter, který umožňuje pomocí jednoho kódu vytvářet aplikace jak pro operační systém Android, tak pro iOS, tedy pro mobilní telefony od firmy Apple. Samotná aplikace je napsaná v kódu C a C++. Dle autorů tohoto jazyka lze použít skoro všechny funkcionality určené pro operační systém Android nebo iOS (*Flutter Introduction*, 2019). Dalším programovacím jazykem vhodným pro vytváření programů operační systém Android je C#, který používá platformu nazvanou Xamarian. Nejčastěji se programuje v prostředí Microsoft Visual Studio, ve kterém je možno využít virtuální zařízení Android, tedy virtuální telefon, který běží v prostředí Windows (*Microsoft Xamarian*, 2019).

3.1.1 Základní informace o jazyku Java

Protože programování pro Android se provádí velmi často v jazyku Java, stejně tak i program Revolution Launcher je v tomto jazyku napsán, je potřeba tento jazyk představit. Jazyk Java byl vyvinut firmou Sun Microsystems (v roce 2009 byla tato firma prodána společnosti Oracle America, Inc) a představen v květnu roku 1995. Jednalo se o alternativu k jazykům C a C++, proto mají některé funkce velmi podobné. Největší výhodou tohoto jazyka je jeho přenositelnost díky Java Virtual Machine (dále jen JVM). JVM je sada počítačových programů, který využívá virtuální stroje ke spuštění dalších programů a skriptů vytvořených v jazyce Java. JVM pouze zpracovává kód uložený ve formátu Java bytecode. Díky této technologii je možné napsat jeden program a spustit ho na několika zařízeních, není nutné mít kód již předem zkompilovaný.

Od listopadu 2018 došlo k zásadním změnám u Oracle Javy. S příchodem Java SE 11 se změnily některé licenční podmínky používání. Licenční práva se vztahují na vývojové prostředí JDK – Java Development Kit a na programy vzniklé nad tímto prostředím a využívající dané prostředí ke svému běhu. Cena licence pro desktop vychází na 2,5 $ na měsíc a server na 25 $/měsíc/procesor. Existují i množstevní slevy. Alternativou k Oracle JDK je přechod k Javě, která je zdarma, tzv. OpenJDK uvolněná pod licencí GPL v2, jenž má free JVM (Michalovič, 2019).

V tabulce (viz: Tabulka 1) níže je uveden přehled verzí jazyka Java a nových funkcí.

Verze Java	Datum uvedení (měsíc/rok)	Nejzajímavější nové funkce
JDK 1.0	1/1996	
JDK 1.1	3/1997	JavaBeans, JDBC
J2SE 1.2	12/1998	Swing, JIT compiler
J2SE 1.3	5/2000	Java Sound, Java Platform Debugger Architecture
J2SE 1.4	3/2002	IPv6 support, Security and cryptography
J2SE 5.0	9/2004	Static imports, Enumerations, Scanner class
JDK 6	12/2006	Web Service support
JDK 7	7/2011	Binary integer literals
JDK 8	3/2014	Lambda expressions, Date and Time API
JDK 9	9/2017	Money and Currency API
JDK 10	3/2018	Garbage-Collector Interface
JDK 11	9/2018	Reading/Writing Strings to and from the Files
JDK 12	3/2019	JVM constants API
JDK 13	9/2019	Switch Expressions

Tabulka 1 - Přehled historie jazyku Java

(Minh, 2019) (Java Version History, 2020)

Zajímavé u značení verzí je, že verze 1.0 a 1.1 jsou pojmenované jako JDK (Java Development Kit), od verze 1.2 do verze 1.4 se používalo J2SE (Java 2 Standard Edition). Od verze 1.5 se znovu vrátilo označení JDK a byla představena interní a externí verze. Interní verze pokračovala v číslování ve formátu 1.5, ale externí verze měla už označení 5 (Minh, 2019).

3.1.2 Základní informace o jazyku Kotlin

Protože jazyk Kotlin je hlavním jazykem pro operační systém Android, je nutno zde uvést nějaké informace. Tento jazyk byl představen v roce 2011 firmou JetBrains běžící na Java Virtual Machine, tedy jedná se o stejný virtuální stroj, který používá jazyk Java. Mezi největší výhodu jazyka Kotlin oproti jazyku Java je, že v Kotlinu lze napsat mnohem kratší kód než v jazyku Java, ale přitom je zachována stejná funkčnost. Níže je uvedeno srovnání kódu pro nastavení textu objektu TextView v operačním systému Android. První kód pochází z jazyka Java, druhý z jazyka Kotlin.

```
//jazyk Java:
TextView myTextView = (TextView)
findViewById(R.id.myTextView);
text.setText("Hello World");

//jazyk Kotlin:
myTextView.setText("Hello World")
```

Jak je vidět na kódu výše, v jazyku Kotlin není potřeba složitě definovat, že objekt TextView pojmenovaný myTextView je opravdu TextView. Toto je jeden z důvodů, proč je Kotlin velmi oblíbený mezi profesionálními vývojáři. Úspora kódu se v průměru pohybuje kolem 35 % (Späth, 2018).

Mezi další velmi významnou výhodu patří, že v jazyku Kotlin všechny typy proměnných nemají při vytvoření výchozí hodnotu null, což neplatí pro jazyk Java, kde se velmi často objevuje chyba NullPointerExceptions, která se v jazyku Kotlin prakticky vůbec nevyskytuje. Dále není potřeba v jazyku Kotlin zachytávat do bloku Try-Catch výjimky (anglicky exception). Další výhodou je 100% zaměnitelnost jazyků Java a Kotlin, jinak řečeno lze použít oba jazyky najednou. V jednom programu je možno zavolat kód z jazyka Java vytvořený v jazyku Kotlin a naopak (Thornsby, 2019).

3.1.3 Základní informace o operačním systému Android

Jedná se o mobilní operační systém založený na jádru Linux, který je dostupný jako otevřený software (open source). Tento systém se používá v mobilních telefonech, v tabletech, v chytrých televizích a hodinkách.

V tabulce níže (viz Tabulka 1) je uveden přehled verzí operačního systému Android, datum uvedení API verze a nejzajímavější nové funkce.

Verze Androidu	Datum uvedení (měsíc/rok)	API verze	Nejzajímavější nové funkce
1.0	9/2008	1	Podpora notifikací, widgety, synchronizace s aplikací Gmail
1.1	2/2009	2	Možnost ukládání příloh ze zpráv, zobrazení a skrytí číselníku
1.5 Cupcake	4/2009	3	Možnost přehrávat video, nové animace, klávesnice na obrazovce
1.6 Donut	9/2009	4	Schopnost fungovat na různých rozlišeních obrazovky
2.0/2.1 Eclair	10/2009	5	Podpora více účtů, nativní podpora MS Exchange, vylepšený webový prohlížeč
2.2 Froyo	5/2010	8	Podpora USB a Wi-Fi tetheringu, vylepšení rychlosti, správy paměti a využívání výkonu
2.3 Gingerbread	12/2010	9	Podpora fotoaparátu na čelní straně zařízení, podpora NFC
3.0 Honeycomb	2/2011	11	Verze určená pouze pro tablety, vylepšený multitasking
4.0 Ice Cream Sandwich	10/2011	14	Softwarová tlačítka, zásadní změna vzhledu celého systému, hardwarová akcelerace grafiky
4.1/4.2/4.3 Jelly Bean	7/2012	15	Plynulejší prostředí díky projektu Butter, rozšiřitelné notifikace

4.4 KitKat	10/2013	19	Android Runtime – ART, omezení přístupu aplikací do externího úložiště
5.0/5.1 Lollipop	11/2014	21	Design Material, vylepšená spotřeba baterie – projekt Volta
6.0 Marshmallow	8/2016	22	Možnost práce s více okny, podpora vykreslovacího API Vulkan 3D
7.0 Nougat	8/2016	24	Přímé odpovědi z oznamovací oblasti, vlastní rychlé nastavení, možnost změny velikosti ikon
8.0 Oreo	8/2017	26	Podpora více obrazovek, vylepšení upozornění aplikací
9.0 Pie	8/2018	28	Adaptabilní úprava jasů, změna uživatelského prostředí
10.0 Q	3/2019	29	Tmavé téma, podpora pro telefony, které mají obraz z rohu do rohu

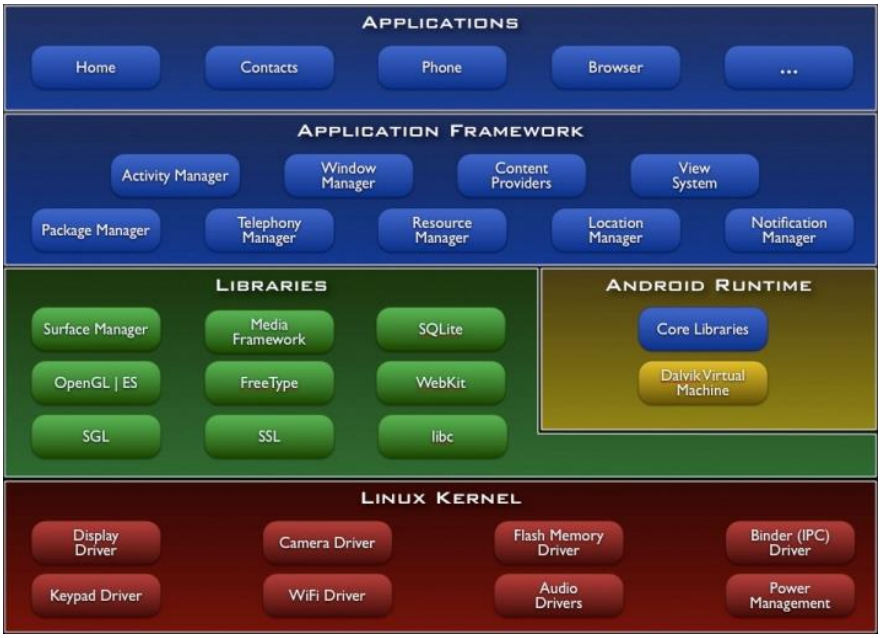
Tabulka 2 - Přehled verzí operačního systému Android

(Raphael, 2019) (Kilián, 2015)

Znalost verzí Androidu je pro každého vývojáře velmi důležitá, protože díky tomu může svoji aplikaci přizpůsobovat novým technologiím. Jediné, na co musí pamatovat, je, aby program byl dostupný pro co nejvíce mobilních telefonů, protože ne všechny telefony mají nejnovější operační systém, protože výrobce telefonu nebo tabletu již přestal vydávat nové aktualizace. Vývojář tedy musí zvolit kompromis mezi nejnovějšími technologiemi a pokrytím co nejvíce možných uživatelů.

V další části bude probrána architektura operačního systému Android. Nejnižší vrstvu představuje Linux Kernel nebo jádro operačního systému. Základní funkce je implementace abstrakce mezi hardware a softwarem ve vyšších verzích. Po startu telefonu je jádro zavedeno do operační paměti a je mu předáno řízení, díky tomu má neustálou kontrolu nad systémem a koordinuje činnosti všech běžících procesů. V další části se nachází Libraries, česky knihovny, které pomocí API nabízejí různé rozhraní, jako je přístup k hodinám, grafice, tlačítkům, 2D grafický engine, SSL šifrování, GPS lokace a mnoho jiného. Android runtime obsahuje virtuální stroj Dalvik Virtual Machine a základní Java knihovny. Virtuální stroj Dalvik (DVM – Dalvik Virtual Machine) byl vyvíjen speciálně pro Android od roku 2005 společností Google. Tento virtuální stroj byl vyvinut, protože programátoři, kteří vyvíjejí aplikace pro operační systém Android, využívá jazyk Java, jehož knihovny jsou licencovány jako open source, ale virtuální stroj JVM – Java virtual machine, který slouží pro překlad programu do spustitelné podoby, již není volně šiřitelný. Dalším důvodem vzniku DVM je optimalizace pro potřeby mobilních zařízení. V současné době DVM je nahrazen virtuálním strojem Android Runtime, zkratka ART. Application Framework je nejdůležitější vrstva pro vývojáře. Aplikační rámec umožňuje přistoupit k nejrůznějším službám, které mohou vývojáři využívat přímo ve

vytvořených aplikacích. Díky nim mohou přistupovat na grafické uživatelské rozhraní, používat hardware zařízení, nastavovat alarmy či spouštět jiné aplikace na pozadí. Další důležité aplikační rámce: Notification Manager – umožňuje zobrazení oznámení, Content Providers – umožňuje přístup k obsahu jiných aplikací, jako jsou například kontakty či kalendář. Activity Manager – ovládá životní cyklus aplikací. Poslední nejvyšší vrstva jsou samotné aplikace, které používají uživatelé, jedná se o předinstalované aplikace nebo aplikace stáhnuté z internetu, nejčastěji z Google Play (Ujbányai, 2012, s. 19–21).



Obrázek 1 - Architektura systému Android

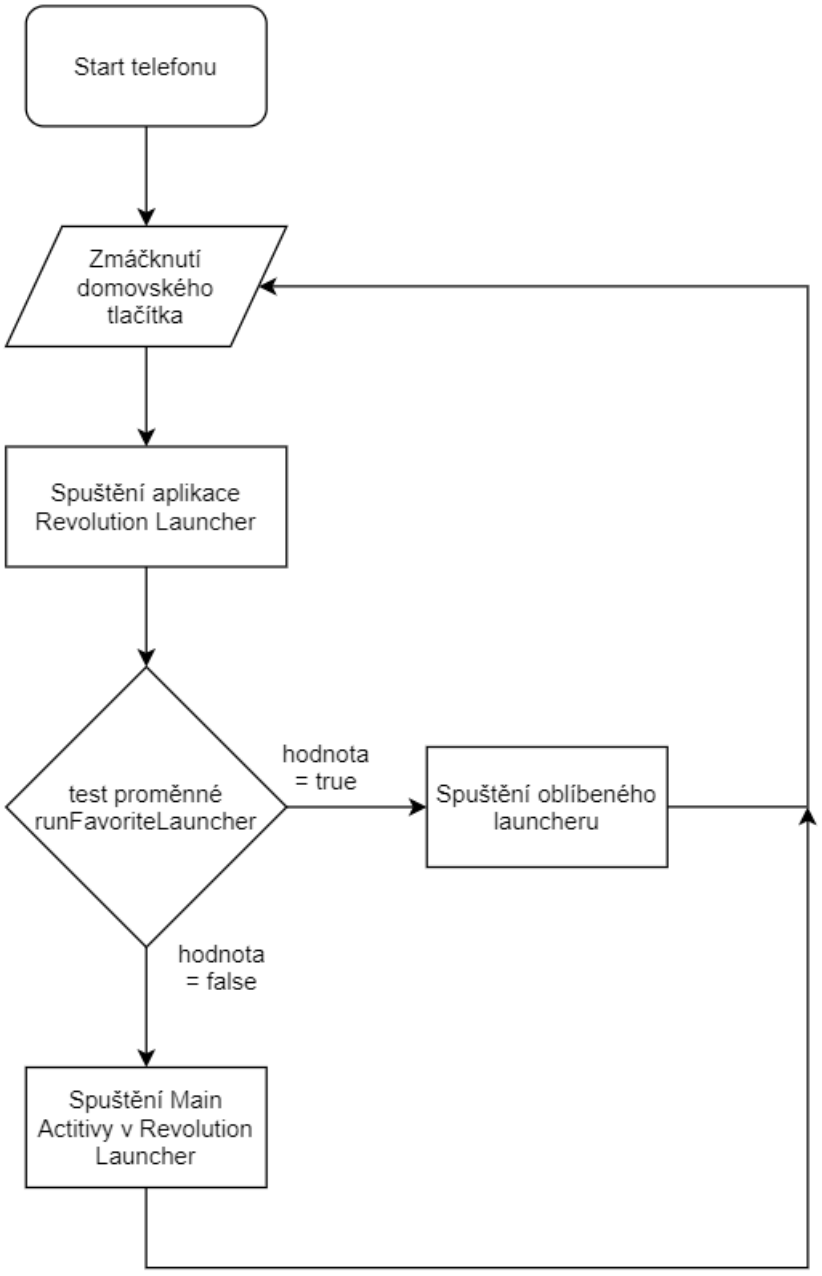
*(*Android Architecture—ELinux.org*, 2011)*

3.1.4 Popis Android programu launcher

Jeden z nejdůležitějších programů běžících na systému Android je launcher, česky také nazývaný jako spouštěč. Launcher je program jako jakýkoliv jiný, a v telefonu jich může být nainstalováno několik alternativ. Tento program má oproti jiným aplikacím jednu výjimku a to je, že se spouští po zapnutí telefonu a zároveň je to jediná aplikace, která se spustí při zmáčknutí domovského tlačítka. Pokud má uživatel nainstalováno více launcherů ve svém telefonu, musí si vybrat, který z nich bude výchozí.

Pochopit funkcionalitu launcheru je velmi důležité, protože cílem této diplomové práce je naprogramovat launcher, který bude výchozí aplikací a pomocí něhož bude velmi snadné měnit rozlišení a přepínat se mezi tímto launcherem a uživatelem oblíbeným spouštěčem. Pro lepší orientaci se tento program jmenuje Revolution Launcher.

Změnit výchozí launcher lze několika způsoby. První může uživatel provést sám v nastavení telefonu, druhá možnost je nainstalování nového launcheru, nebo odinstalování současného launcheru, poté sám systém vyzve uživatele, aby vybral výchozí spouštěč. Další možnost je trochu složitější, funguje tak, že jeden launcher je vždy výchozí a dle předem dané logiky řídí, co se stane po zmáčknutí domácího tlačítka. Vždy se spustí nastavený výchozí launcher, ale ten nemusí nic zobrazovat a rovnou spustí jinou aplikaci, většinou jiný launcher. Pro lepší pochopení slouží vývojový diagram níže (Obrázek 2). Hodnota v proměnné runFavoriteLauncher se nastaví po zmáčknutí widgetu, který mění rozlišení a počet bodů.



Obrázek 2 - základní princip aplikace Revolution Launcher

(vlastní dílo autora)

Tento princip je použit při návrhu aplikace pro adaptaci mobilního zařízení pro desktopový provoz. Revolution Launcher je nastaven jako výchozí aplikace, ale podle nastavení vnitřní logiky tato aplikace rozhoduje, zda se spustí aktivita v aplikaci Revolution nebo uživatelův oblíbený launcher.

3.2 Návrh řešení aplikace – specifické Java funkce

Tato sekce pojednává o funkcionalitách programovacího jazyka JAVA, které je nutné znát v případě, že vývojář chce vyvíjet pro operační systém Android v jazyku Java. Skoro všechny kódy napsané ve standardní verzi jazyku Java, nazývanou také Java SE (Standard Edition) lze použít při programování pro mobilní telefony. Jedinou výjimkou je grafické prostředí, to je specifické pouze pro mobilní telefony a nelze kopírovat ze standardního kódu pro jazyk Java.

3.2.1 Základní funkcionalita jazyka Java

Převádění do strojového jazyka v jazyku Java prochází pěti fázemi, editováním, kompilací, zavedením, ověřováním a prováděním. Fáze ověřování je velmi důležitá, protože umožňuje dosáhnout velmi vysoké bezpečnosti spuštěného programu. Dalším rozdílem oproti jiným programovacím jazykům je, že překlad kódu neprobíhá přímo do spustitelného kódu, tedy do strojového jazyka počítače, ale do pseudo-jazyka nazývaného byte-code. Tento jazyk je nezávislý na cílovém počítači nebo přístroji, což znamená, že programátora nemusí zajímat, na jakém typu počítače a operačního systému bude jeho program běžet (Herout, 2007, s. 21–22). Také je důležité zdůraznit, že jazyk Java je objektově orientovaný jazyk a skládá se z prvků zvaných třídy. Každá třída je umístěna ve svém vlastním souboru, který má stejné jméno jako tato třída. Třída může mít svoje proměnné – atributy a svoje metody. Objektem se nazývají konkrétní prvky – instance nějaké třídy. Třída je obecný popis vlastností a chování, objekt je poté již konkrétní prvek (Roubalová, 2015, s. 57). Pochopení tohoto principu je klíčové, protože stejný postup se používá i při programování pro Android.

3.2.2 Principy objektově orientovaného programování

Objektově orientované programování vychází z myšlenky, že všechny programy jsou simulací skutečného nebo vymyšleného světa. Všechny tyto simulované světy jsou ve skutečnosti světy objektů, které mají různé vlastnosti a schopnosti a navzájem spolu provádějí různé interakce. V běžném životě můžeme za objekty považovat osoby, zvířata a věci. V objektově orientovaných programech lze říci, že za objekt lze považovat vše, co můžeme nazvat

podstatným jménem. Ve většině programů se vyskytují desetitisíce objektů. Pro lepší porozumění je potřeba je nějak roztřídit. Objekty lze roztřídit do skupin s velice podobnými vlastnostmi, tyto skupiny se nazývají třídy. Objekty patřící do dané třídy se nazývají instance této třídy. Třída může mít obecně libovolný počet instancí. Existují i třídy, které dovolí vytvoření pouze omezeného počtu instancí. Metoda je část programu, která definuje, jak bude objekt reagovat na zprávy, programátoři častěji mluví o volání metod (Pecinovský, 2012, s. 54–60).

3.2.3 Nejdůležitější funkce v jazyku Java

V této části bude pojednáno o nejdůležitějších funkcích, které by měl znát každý programátor jazyka Java a také programátoři pro Android.

První funkce je přetěžování metod, která je jedním ze způsobů, jakým Java implementuje polymorfismus. V Javě mohou dvě či více metod v rámci téže třídy sdílet stejný název, pod podmínkou, že jsou jejich parametry odlišné. První metoda akceptuje pouze jednu hodnotu a vrátí text s hodnotou, druhá metoda se stejným názvem akceptuje dvě hodnoty a vrací součet dvou hodnot:

```
int ukazkaPretizeni (int a){
system.out.printl("Jeden parametr: " + a);

int ukazkaPretizeni (int a, int b){
return a + b;
```

Přetěžování metod podporuje polymorfismus, protože se jedná o jeden ze způsobů, jakým Java implementuje paradigma „jedno rozhraní, více metod“. Přetěžování metod lze také provádět pomocí typu proměnné, která je definovaná v metodě, viz příklad níže:

```
Class Pretizeni2{
void f(int x) { //první verze
System.out.println{"Uvnitr f(int): " + x);
}
void f(double x){
System.out.println{"Uvnitr f(double): " + x);
}
```

Volání metody se poté používá následně:

```
Pretizeni2 obj = new Pretizeni2();
```

```
int i = 10;
doulble d = 10.1;
obj.f(i) // zavolá metodu obj.f(int)
obj.f(d) // zavolá metodu obj.d(double)
```

Význam přetěžování spočívá v tom, že umožňuje přistupovat k souvisejícím metodám prostřednictvím stejného jména. Jazyky, které nepodporují přetěžování, musí pro každou funkci stanovit jiný název, což poté stěžuje čitelnost kódu. (Schildt, 2012, s. 222–224)

Další velmi často používanou funkcionalitou je dědičnost, která je jedním ze tří základních principů objektově orientovaného programování. Díky dědičnosti je možno vytvořit obecnou třídu, jež definuje vlastnosti společné nějaké skupině souvisejících prvků. Z této třídy poté mohou dědit další, specifičtější třídy, z nichž každá přidá věci, kterou jsou pro ni jedinečné. V jazyku Java se děděná třída nazývá nadtřída a třída, která dědění provádí, se nazývá podtřída. Podtřída dědí všechny proměnné a metody definované nadtřídou a přidává své vlastní, jedinečné prvky. Java podporuje dědičnost tím, že umožňuje jedné třídě začlenit do své deklarace jinou třídu pomocí klíčového slova extends. Podtřída tedy rozšiřuje nadtřídu. V tabulce níže (viz Tabulka 3) je uveden příklad použití dědičnosti. Tvar2D je nadtřída a obsahuje pouze parametry šířka, výška a zobraz rozměry. Podtřída jménem Trojúhelník rozšiřuje třídu Tvar2D, obsahuje stejné parametry jako nadtřída, ale zároveň přidává parametry styl, plocha a zobrazStyl(). Objekt Tvar 2D je nadtřídou třídy Trojúhelník, ale je také zcela nezávislou, samostatnou třídou. Fungování nadtřída pro nějakou podtřídu neznamená, že tuto nadtřídu nelze používat. Objekt typu Tvard2D neví nic o podtřídách třídy Tvar2D a nemá k nim žádný přístup (Schildt, 2012, s. 255–261).

<table>
<tr><td rowspan="3">Tvar2D</td><td>sirka</td><td rowspan="6">Trojúhelník</td></tr>
<tr><td>vyska</td></tr>
<tr><td>zobrazRozmery()</td></tr>
<tr><td></td><td>styl</td></tr>
<tr><td></td><td>plocha</td></tr>
<tr><td></td><td>zobrazStyl()</td></tr>
</table>

Tabulka 3 - Tabulkové zobrazení dědičnosti

(Schildt, 2012, s. 258)

3.2.4 Tvorba dobrého objektově orientovaného návrhu

Velmi důležité je psát kód podle zavedených konvencí, protože to pomáhá s přehledností zdrojového kódu a ostatní programátoři, kteří pracují na stejném projektu, se mohou mnohem lépe orientovat v napsaném kódu.

Pojmenování balíčků se dělá vše malými písmeny, například com.exmple.deepspace. Jména tříd začínají velkým písmenem, nepoužívá se zde žádná pomlčka, pokud se jméno skládá z více slov, je první písmeno z dalšího slova napsáno velkým písmenem, například ClassOne. Pojmenování metod a lokálních proměnných začíná malým písmenem a další slovo, které tvoří název metody, začíná velkým písmenem, například methodOne. Konstanty, tedy hodnoty, které se nemění, jsou psány velkými písmeny, například NUMBER (*Google Java Style Guide*, 2020).

Knihovny tříd se píší do zvláštního balíčku, kód je potřeba psát tak, aby jej bylo možno znovu využít v jiné aplikaci. Balíček by měl obsahovat pouze třídy s určitým společným zaměřením. Pokud se balíček stane časem velmi rozsáhlý, je dobré udělat další vnořené dílčí balíčky. Skvělou funkcí programu Java je zapouzdření, které slouží k ukrytí datových složek nebo metod uvnitř objektu. Pokud ve třídě jsou datové složky deklarované jako veřejné (public), lze k nim přistupovat z libovolné třídy, například:

```
public class Employee {
public int employeeID;
}

Employee emp = new Emloyee();
emp.employeeID = 123456;
```

Upravovat datové složky tímto způsobem by se nemělo v praxi používat. Hlavní výhodou klíčových objektů je, že lze jejich datovou složku chránit před úpravami. Tato ochrana neznamená, že k nim nelze zvenčí přistupovat, ale tak to není. Zabezpečení datových složek spočívá jen v omezení přímého přístupu. Třída Employee by tedy měla vypadat takto:

```
protected int employeeID;

public int getEmployeeID(){
return employee ID
}
Public void setEmployeeID (int id) {
employeeID = id;
}
```

Díky zapouzdření je kód mnohem přehlednější a navíc zjednodušuje převod existujících tříd na třídy, které jsou základem pro tvorbu distribuovaných objektů (Spell & Kiszka, 2002, s. 23–40).

3.2.5 Nejzajímavější Java frameworky

Framework (česky aplikační rámec) jsou knihovny, které ulehčují práci při programování aplikace, nejčastěji se používá tam, kde je potřeba pořád dokola opakovat jednu funkci, díky tomu je kód přehlednější a vývoj rychlejší.

Jazyk Java je velmi dobrý nástroj pro programování především díky různým frameworkům používaným pro tvorbu podnikových (anglicky Enterprise) aplikací. Mezi nejoblíbenější patří Spring, který díky relativně jednoduchým knihovnám umožňuje programátorům práci s webovým prostředím, anglicky nazývané Model-View-Controller (MVC), se zabezpečením, přístup k adresářům (LDAP) a ve spolupráci s frameworkem Hibernate mnohem jednodušší komunikace s databázemi. Bohužel tyto zmíněné funkce nefungují plně v operačním systému Android, ale existují dva projekty, které se jmenují Spring mobile a Spring for Android. Tyto frameworky umožňují REST operace a komunikaci aplikací vytvořených pro Android s aplikacemi naprogramovaných pomocí Spring frameworku (Walls, 2011, s. 20–27).

3.2.6 Zachytávání výjimek

Další velmi často používanou funkcí je nutnost zpracování výjimek. Výjimkou je myšlena abnormální situace vznikající v nějaké části kódu v době běhu. V Javě výjimku představuje objekt popisující nějaký chybový stav, který nastal v nějaké části kódu. Vznikne-li tento výjimečný stav, běhové prostředí vytvoří objekt představující danou výjimku. Tento objekt je poté vrácen metodě, která chybu způsobila. Pro zpracování výjimek se v Javě používá 5 klíčových slov: try, catch, throw, throws a finally. Ta část kódu, kde je potřeba hlídat výjimky, by měla být obsažena v bloku try, pokud v tomto bloku vznikne výjimka, pak je možno tuto výjimku zachytit v bloku catch a poté ji zpracovat. Pro ručně vyvolané výjimky se použije klíčové slovo throw. Jakákoliv výjimka, která je metodou vyvolána a předána dále, musí být patřičným způsobem specifikována pomocí slova throws. Kód, který musí být bezpodmínečně proveden po dokončení kódu v bloku try, je zapsán v bloku finally (Schildt, 2014, s. 255–257). Zachycení výjimek v aplikaci Revolution Launcher je použito pro příkaz Runtime, který mění rozlišení a počet bodů.

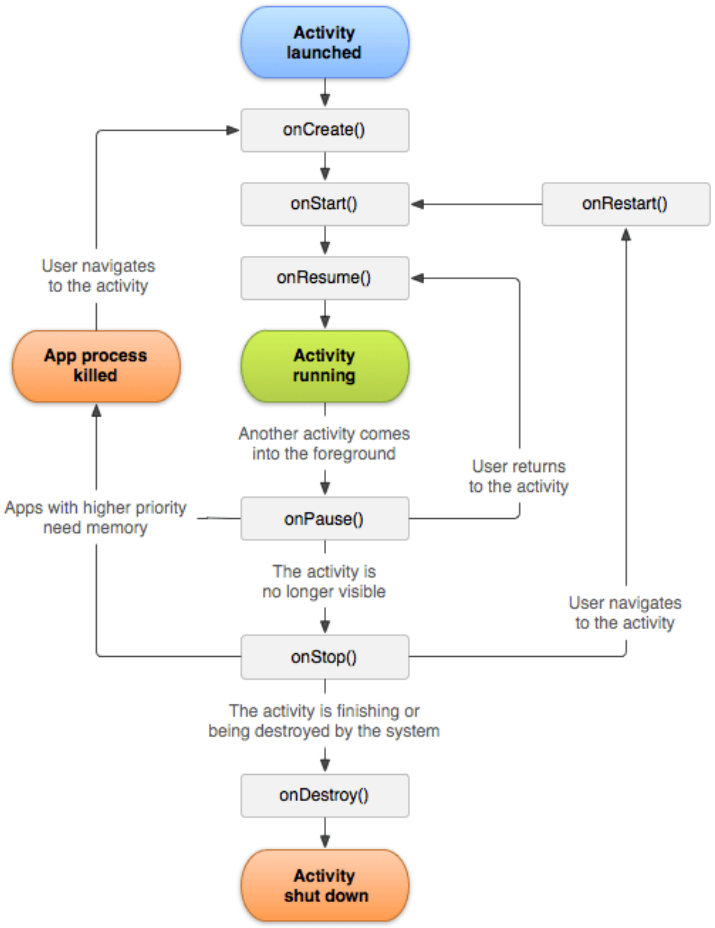
3.3 Návrh řešení aplikace – specifické funkce pro Android Studio

Tato sekce pojednává o jazyku Java a jeho návaznost na programování v jazyku Java v programu Android Studio. Pro operační systém Android lze programovat ve více

programovacích jazycích, protože autor má dobré zkušenosti s jazykem Java, byl jako programovací jazyk vybrán právě jazyk Java. Dále se v této části bude podrobněji pojednávat o funkcích použitých pro návrh řešení aplikace pro adaptaci mobilního zařízení pro desktopový provoz.

3.3.1 Životní cyklus aktivity

Revolution Launcher obsahuje plno aktivit, je proto dobré si říci, jaký je životní cyklus aktivit. Jakákoliv aktivita může spouštět jinou aktivitu. Vždy, když dojde ke spuštění nové aktivity, je předchozí aktivita pozastavena, ale systém ji nechá v zásobníku, který používá systém LIFO (last in, first out – poslední dovnitř, první ven). Zjednodušeně lze říci, že se jedná o klasickou frontu. V případě, že uživatel zmáčkne tlačítko zpět, dojde k vytažení a zobrazení předchozí aktivity ze zásobníku. Protože u většiny mobilních zařízení jsou paměť a zdroj omezeny, stanovuje operační systém Android mechanismy pro zachování těchto zdrojů. Tyto mechanismy jsou patrné v životním cyklu aktivity, který definuje stavy nebo události, jimiž aktivita prochází od svého vzniku do jejího dokončení. Kompletní životní cyklus se odehrává mezi voláním metody onCreate (tedy vytvořením aktivity) a metodou onDestroy (zrušením aktivity), schéma je možno vidět níže (Obrázek 3). Životní cyklus ve stavu viditelnosti se odehrává mezi voláním metody onStart a metody onStop. Během této doby uživatel vidí aktivitu na displeji svého telefonu. Mezi voláním metody onStart a onStop lze udržovat prostředky potřebné k zobrazení aktivity pro uživatele. V případě, že se aktivita dostane do popředí, zavolá se metoda onResume a skončí voláním metody onPause, kdy se aktivita pozastaví (Vávrů & Ujbányai, 2013, s. 42–43).

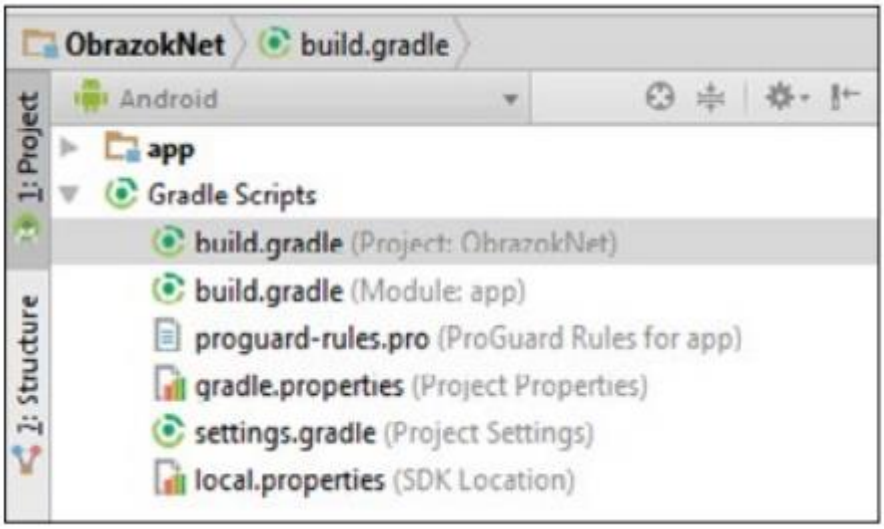


Obrázek 3 - Životní cyklus aktivity

*(*Android Activity Lifecycle—Javatpoint, *2018)*

3.3.2 Kompilování zdrojového kódu

Pro kompilování kódu v aplikaci Android Studio se používá Gradle, který je přímo integrovaný v Android Studiu a slouží k automatizaci procesů, přesněji to je jazyk na automatizaci. Jedná se o jazyk typu Domain Specific Language (DSL), který umožňuje popsat posloupnost úloh, které chce zautomatizovat. Samotný Gradle toho moc neumí, jeho síla spočívá v pluginech. Na sestavování aplikací pro Android se používá Android plugin for Gradle. Android studio umí vytvářet více podprojektů, ze kterých se skládají komplexní aplikace. Každý podprojekt má vlastní build.gradle skript (viz Obrázek 4).



Obrázek 4 - Struktura zobrazení složky Gradle v Android Studiu

(Lacko & Herodek, 2017, s. 49)

Za názvem build.gradle je v závorce uvedeno, že se jedná o build soubor pro celý projekt ve tvaru (Project:<project name>). Skript na té nejvyšší úrovni obsahuje globální konfiguraci pro všechny podprojekty. Zde nejdůležitější je sekce dependencies, která definuje vzájemné závislosti modulů (Lacko & Herodek, 2017, s. 49–52).

3.3.3 Nejzajímavější frameworky pro operační systém Android

Jak již bylo uvedeno výše, samotná Java by nikdy nebyla tak populární, jako je nyní, nebýt různých skvělých frameworků, jako je Spring, který bohužel není plně podporovaný v operačním systému Android, ale díky jeho obrovské popularitě se plno jiných programátorů snaží vytvořit alternativu pro systém Android. Mezi nejzajímavější frameworky patří například ORMLite, který slouží pro komunikaci s databází. Tento framework je alternativou pro framework Hibernate, který používá technologii ORM, jenž mapuje objekty vytvořené v Javě k jednotlivým položkám existujících v databázi (*OrmLite—Lightweight Java ORM Supports Android and SQLite*, 2020).

Další zajímavý framework je OkHttp, který je velmi populární v operačním systému Android pro použití na REST operace, které slouží pro komunikaci mezi aplikacemi přes http protokol. Zde se nejčastěji používají čtyři příkazy, POST pro vytvoření nového záznamu, GET pro čtení již existujících dat, PUT pro změnu dat a DELETE pro výmaz dat (*OkHttp Example REST Client*, 2017).

Oba tyto frameworky jsou v plánu pro přidání funkcí do aplikace Revolution Launcher. ORMLite bude použit na tvorbu uživatelských účtů, kam si lidé budou ukládat nastavení launcheru a OkHttp bude použit například pro stahování dat o počasí. Další framework, který

je použit v aplikaci Revolution Launcher, je PaperDB, aby fungoval, je potřeba do nastavení Gradle napsat tento řádek:

```
implementation 'io.paperdb:paperdb:2.6'
```

Pro ukládání relativně malých hodnot, které je potřeba mít dostupné i po ukončení aplikace, je možno použít SharedPreferences API. SharedPreferences objekt ukazuje na soubor, který obsahuje párové hodnoty, kde jedna reprezentuje název a druhá hodnotu. SharedPreferences umožňuje z tohoto souboru data číst i zapisovat (*Save key-value data*, 2020).
Aplikace Revolution Launcher potřebuje při kliknutí na widget pro změnu nastavení rozlišení a DPI, uložit informaci do proměnné runFavoriteLauncher. Při zmáčknutí domovského tlačítka se načte hodnota z této proměnné a určí, zda se mu spustí hlavní aktivita v aplikaci Revolution Launcher nebo uživatelův oblíbený launcher. Vzhledem k tomu, že SharedPreferences vyžaduje znát kontext, tedy kde se nachází vzhledem k aplikaci, ke které patří, což je velmi složité při použití widgetu, protože ten se nenachází v kontextu aplikace Revolution Launcher, je jednodušší použít knihovnu PaperDB, která umožňuje jednoduše zapsat data, aniž by byl uveden kontext, ve kterém se spouští. Použití PaperDB je uvedeno v kapitole 4.

3.3.4 Informace o AndroidManifest

Stěžejním souborem v každém Android programu je soubor AndroidManifest.xml. Tento soubor definuje všechny základní funkcionality programu, definuje jednotlivé komponenty a oprávnění aplikace, nachází se v hlavní složce projektu. Mezi možnosti nastavení patří minimální verze API, která se nastavuje pomocí „android:minSdkVersion=“, kde za rovná se napíše číslo, které určí minimální verzi operačního systému Android, na kterém bude aplikace fungovat. Nejčastěji se toto nastavení používá v případě, že vývojář chce použít funkce z novější verze systému Android, a tím předejít nefunkčnosti na starších verzích. Dále se zde nastavují vlastnosti každé aktivity, jakou bude mít orientaci, zda na šířku, na výšku nebo automaticky dle polohy telefonu. Dále se zde nastavuje <intent-filter>, který se dá popsat jako záměr abstrakce operace, kterou je potřeba provést. Pomocí záměrů je možné propojit nejen aktivity, ale i aktivity více aplikací (Lacko, 2015, s. 90–91).
Tento <intent-filter> je použit pro specifikování, že aplikace funguje jako launcher, pomocí tohoto kódu:

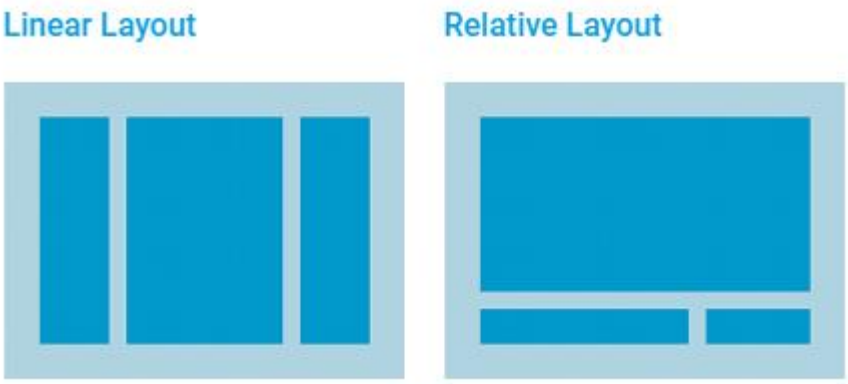
```
<intent-filter>
<category android:name="android.intent.category.HOME" />
```

```
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Zde android.intent.category.HOME říká, že tato aktivita bude fungovat jako launcher, android.intent.category.DEFAULT definuje, že zvolená aktivita bude výchozí aktivitou, která se zobrazí po spuštění aplikace (Sinicki, 2018).

3.3.5 Informace o rozvržení (layout)

Každá aktivita zobrazená na přístroji uživatele má vlastní specifické rozvržení (anglicky layout). Toto rozvržení se nastavuje v xml souboru uloženého v /res/layout. Nastavovat lze pomocí grafického rozhraní nebo přímo pomocí textu. Grafické zobrazení, jak už název napovídá, má výhodu v tom, že je hned vidět, jak budou prvky rozloženy. Nastavení v textovém režimu se používá spíše pro opakující se akce, jako je například vytváření více tlačítkem a jejich pojmenování a přidání jiných funkcí. Mezi hlavní typy layoutů patří Linear Layout, který všechny svoje potomky zachycuje v jednom směru a to buď vertikálním, nebo horizontálním. Relative Layout umisťuje svoje potomky relativně k jiným objektům. Table Layout umožňuje dávat všechny objekty do řádků a sloupců. Zobrazení Linear Layout a Relative Layout je možno vidět níže (Obrázek 5). Jak je na obrázku vidět, každé zobrazení se hodí pro jiné použití (*Layouts*, 2020).

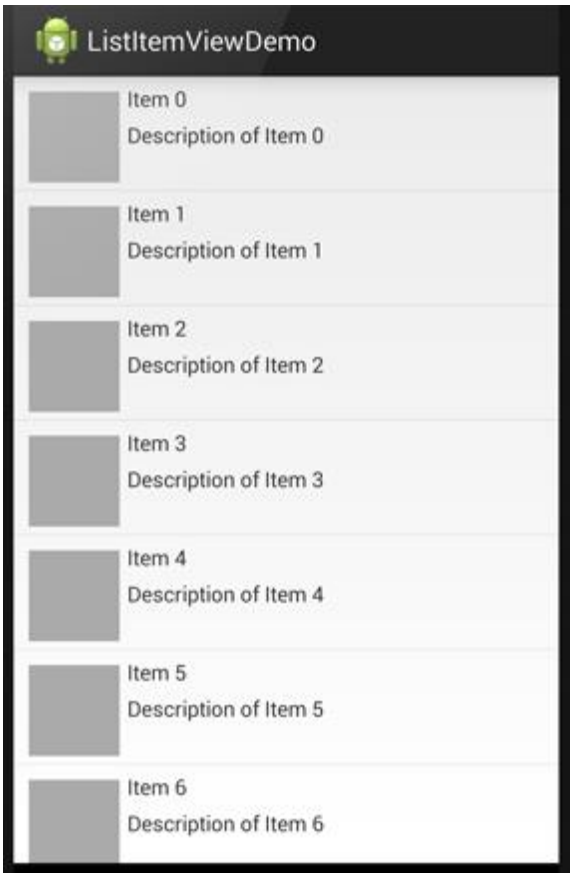


Obrázek 5 - Vzhled Linear Layout a Relative Layout

(Layouts, 2020)

V aplikaci Revolution Launcher jsou nejvíce používána právě tato dvě rozložení. Další použití funkce layout je možné uvnitř jiných layoutů. Tato funkce se používá například pro zobrazení nějaké menu nebo dat uspořádaných do tabulek. V Revolution Launcher se používá objekt ListView pro zobrazení menu a GridView pro zobrazení ikon na ploše.

ListView je objekt, který vkládá do skupiny několik položek a zobrazuje je ve vertikálním seznamu, ve kterém je možno rolovat. Položky jsou vloženy automaticky do seznamu pomocí adaptéru, který vkládá obsah ze zdroje, jakým jsou například pole nebo databáze. Adaptér je ve skutečnosti most mezi uživatelským prostředím komponentů a zdrojem dat, která tyto komponenty naplňují daty. Adaptér udržuje data a posílá je do položky zvané adapter view, který umožňuje data zobrazovat rozdílnými způsoby, například jako klasický seznam, nebo nekonečný seznam či pole. Pomocí xml souboru se dále nastavuje, jak mají jednotlivé pole graficky vypadat, tedy například vlevo bude obrázek a vpravo bude uveden text. Vzhled je možno vidět na obrázku (viz Obrázek 6). V současné době je ListView nahrazován funkcí RecyclerView, který má stejné nebo lepší vlastnosti než ListView (*Android List View—Tutorialspoint*, 2019).

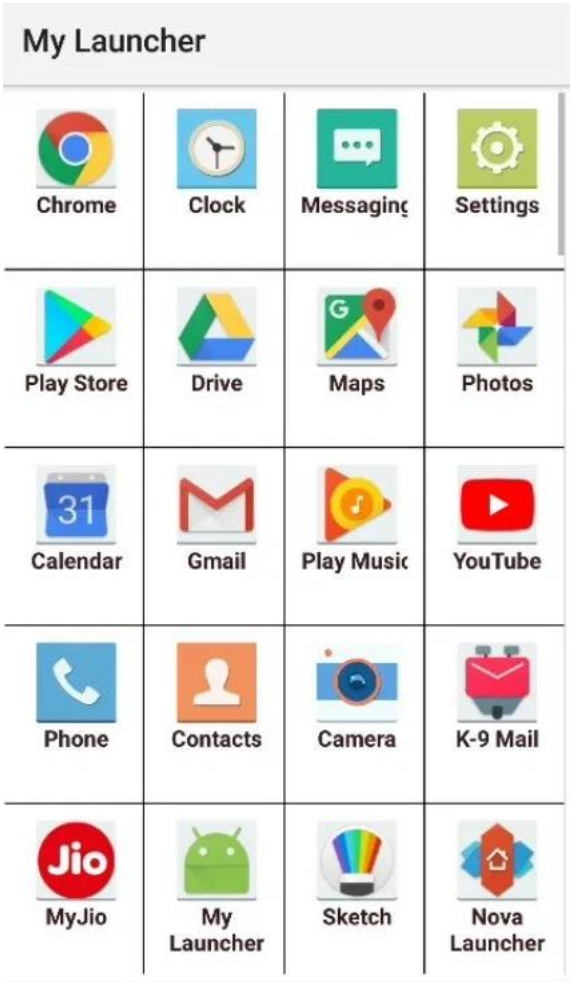


Obrázek 6 - Ukázkový vzhled ListView

(Hardy, 2014)

Dalším důležitým objektem je GridView, který je v aplikaci Revolution Launcher použit pro zobrazení ikon na hlavní stránce aplikace. GridView zobrazuje položky ve dvou dimenzionálních polích (řádky a sloupce), ve kterém je možno rolovat, pokud je položek více, než je velikost pole. Položky jsou vkládány automaticky pomocí funkce ListAdapter. Další

princip je už stejný jako u ListView. Adaptér slouží jako most mezi uloženými daty a polem, které se má zobrazit. AdapterView posílá data a v xml souboru je definováno, jaká data a kde mají být zobrazena, vzhled je možno vidět na obrázku níže (Obrázek 7), ve kterém je zobrazen jednoduchý seznam aplikací. V současné době je GridView nahrazován objektem ConstraintLayout, který nabízí lepší funkce a je méně náročný na paměť (*Android Grid View—Tutorialspoint*, 2020).



Obrázek 7 - Ukázkový vzhled GridView

(Vikani, 2018)

3.3.6 Přizpůsobení vzhledu dle použitého zařízení

Každý programátor musí dříve, nebo později řešit, jak naprogramovat svoji aplikaci, aby fungovala dobře na všech dostupných zařízeních, které mají často velmi rozdílné rozlišení a velikost zobrazovací plochy. V případě, že na to vývojář nebude myslet, může se stát, že některé obrázky budou nepřirozeně roztažené nebo rozmazané. První nástrahou, které je potřeba se vyhnout, je definování velikosti a vzdálenosti na základě počtu pixelů. Zde nastává

hlavní problém, a to je rozdíl počet pixlů, které má každé zařízení jiné, což vyústí v rozdílné rozmístění objektů. Aby se předešlo tomuto problému, je potřeba použít jednotku zvanou pixely nezávislé na hustotě (anglicky Density-independent pixels, zkratka dp). Hodnota jednoho dp virtuálního pixelu odpovídá přibližně jednomu pixelu na zařízení, které má hustotu 160 dpi (160 bodů na palec je považováno za základní hustotu). Operační systém pomocí vzorce px = dp * (dpi / 160) převede zadané hodnoty dp na odpovídající hodnoty opravdových pixelů v zařízení. Pro zadání velikosti textu je nutno používat jednotku nazvanou scalable pixels (zkratka sp). Sp jednotka má ve výchozím nastavení stejnou velikost jako dp, ale jeho velikost se mění na základě uživatele zvolené velikosti textu. V tabulce níže (viz Tabulka 4) je zobrazen přehled hustoty pixelů a jejich odpovídající multiplikátor zadané hodnoty dp (*Support different pixel densities*, 2020).

Hustota obrazovky	Název	Multiplikátor
120 dpi	ldpi – low-density screens	0,75x
160 dpi	mdpi – medium-density screens	1x
240 dpi	hdpi – high-density screens	1,5x
320 dpi	xhdpi – extra-high-density screens	2x
480 dpi	xxhdpi – extra-extra-high-density screens	3x
640 dpi	xxxhdpi – extra-extra-extra-high-density screens	4x

Tabulka 4 - Označení hustoty obrazovky a jejich multiplikátor

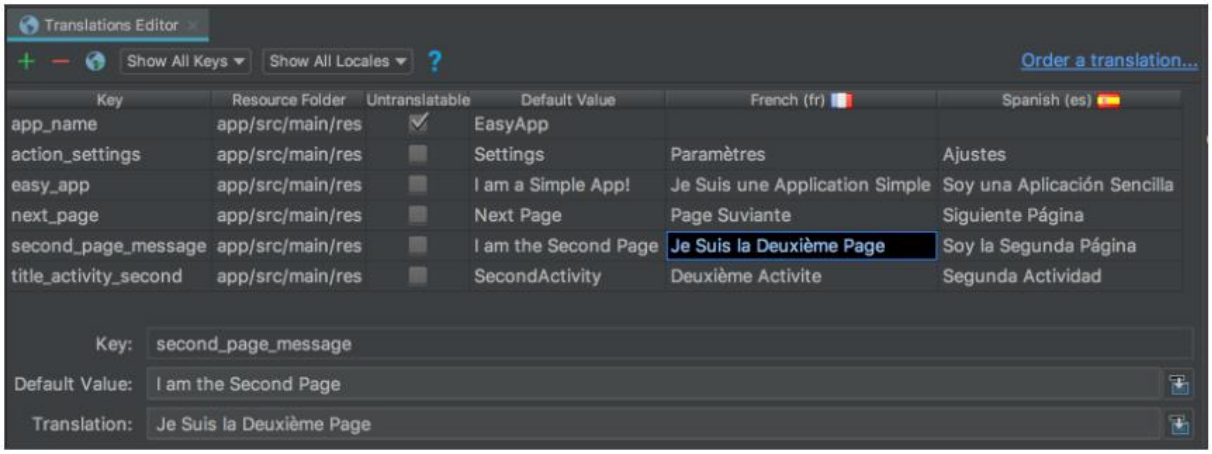
(Support different pixel densities, 2020)

Kromě uvádění velikosti pro vzdálenost a velikosti v pixelech nezávislých na hustotě, je také nutno správně umisťovat prvky při návrhu aktivity. Mezi nejvhodnější layouty pro podporu různých rozlišení patří ConstrainLayout, ve kterém zde umístěné objekty určují svoji polohu vzhledem k jiným objektům nebo k okrajům. Bohužel ani ConstrainLayout nevyřeší všechny možné scénáře, které mohou nastat, proto je velmi důležité, aby velikosti objektů nebyly napevno uvedené. Místo toho je potřeba používat velikost objektu pomocí: „match_parent" – určený pro šířku a výšku komponent, při použití vždy vyplní maximální možnou plochu. Další možnost je použít „wrap_content", který mění svoji velikost podle obsahu, který má v sobě a umí se roztáhnout do své maximální velikosti. V případě, že je potřeba vytvořit jiný vzhled pro případ, kdy je zařízení na výšku a poté, když je zařízení na šířku, je potřeba vytvořit další layout v adresáři res/layout/, a to pro každou obrazovku, která vyžaduje rozdílný vzhled. Další možnost je měnit layout podle šířky měřenou v dp (density-independent pixels). Například pro hlavní aktivitu bude vytvořena cesta k xml souboru pro zařízení s rozlišením 600dp res/layout/main_activity.xml. V případě, že bude použito zařízení, které má šířku více jak 600

dp, bude načtena konfigurace z cesty res/layout-w600dp/main_activity.xml. Dále je možno vytvořit konfiguraci dle orientace zařízení, při použití na šířku je použita cesta res/layout-land/main_activity.xml. Také lze orientaci kombinovat se šířkou zařízení, takže pokud bude 7“ zařízení na výšku, načte se konfigurace z cesty res/layout-sw600dp/main_acitivity.xml, pro stejné zařízení se použije cesta res/layout-sw600dp-land/main-activity.xml. Ještě je nutno vysvětlit, co znamená w a sw, w (anglicky width) je dostupná šířka, která se mění podle toho, zda je zařízení na výšku, nebo na šířku, sw (anglicky smallestWidth) se nemění dle rotace zařízení, jen operační systém zjistí nejmenší šířku zařízení (*Support different screen sizes*, 2020).

3.3.7 Překlad aplikace do jiných jazyků

Operační systém Android je rozšířený po celém světě a také umožňuje provoz v mnoha jazycích. Vydat program s podporou všech možných jazyků umožní vývojáři, aby si jeho produkt stáhlo co nejvíce lidí. Překlad do cizího jazyka není nic extra složitého. První podmínkou je, aby názvy a jiné texty byly napsány nějakým světovým jazykem, jako je angličtina, protože je mnohem jednodušší najít člověka, co umí překládat z angličtiny do svého jazyka než z neznámého jazyka do svého mateřského jazyka. Další podmínkou je, aby všechny texty při tvorbě layoutu byly napsány pomocí anotace @string<klíčová hodnota>, například @string/heading. To zajistí, že textový řetězec nebude napevno zvolen, ale bude se měnit dynamicky. Pro zobrazení správného textu je potřeba se dostat do souboru string.xml, který se nachází v res/values/. Zde lze přímo v textu nebo v textovém editoru napsat, čemu hodnota odpovídá, tvar je následující: <string name=„heading">This is heading</string>. Pro přeložení například do českého jazyka je nutno otevřít editor nacházející se na stejném místě, jako je soubor string.xml, zde přidat jazyk, do kterého se má překládat a vybrat češtinu. Program Android Studio vytvoří nový adresář s cestou /res/values-cs-rCZ a zde vznikne nový soubor nazvaný string.xml. Překládat pak lze přímo psaním do souboru, například napsat <string name=„heading">Toto je nadpis</string>. Nebo použít vestavěný editor (viz Obrázek 8). V případě, že si programátor nepřeje, aby se text překládal, může vložit do pole <string> text translatable=„false" (*Localize the UI with Translations Editor*, 2019).



Obrázek 8 - Vzhled vestavěného editoru pro překlad textu v aplikaci Android Studio

(Localize the UI with Translations Editor, 2019)

3.3.8 Informace o Android Debug Bridge

Nástroj adb (Android Debug Bridge) funguje jako přemostění mezi emulátory nebo zařízeními a zbytkem vývojových nástrojů. Přemostění má podobu procesu démona, který se vytvoří při prvním použití různých nástrojů po posledním restartu systému. Pomocí nástroje adb lze zobrazit seznam rozpoznaných zařízení v emulátoru (příkaz adb devices), získat přístup k příkazové řádce (adb shell), instalovat v telefonu nebo emulátoru aplikace (příkaz adb install), kopírovat soubory do zařízení (adb push) nebo ze zařízení (adb pull) a mnoho jiných funkcí (Allen, 2013, s. 575–576).

Pro nastavení speciálního přístupu pro aplikaci Revolution Launcher se používá příkaz:

```
adb shell pm grant <jméno balíčku>
android.permission.WRITE_SECURE_SETTINGS
```

Secure system setting obsahuje nastavení, které aplikace mohou číst, ale nemohou je měnit. Jedná se o nastavení, které musí sám uživatel vlastnoručně nastavit skrze svůj telefon. Při použití zápisu do secure settings je umožněn zvolené aplikaci přístup k speciálním bezpečnostním nastavením v operačním systému Android (*Settings.Secure*, 2019).

Druhý příkaz, který se používá, je „adb shell settings put secure show_ime_with_hard_keyboard 0“, který nastaví softwarové klávesnici, aby byla schovaná, nebo se zobrazovala jen na jednom řádku, pokud je připojena klasická hardwarová klávesnice.

3.3.9 Widgety

V aplikaci Revolution Launcher widgety představují hlavní nástroj pro změnu rozlišení a dpi včetně přepínání mezi aplikací Revolution Launcher a oblíbeným launcherem.

Widgety jsou malé aplikace, které lze považovat za základní aspekt pro přizpůsobení aplikace fungující jako výchozí obrazovka. Nejčastěji zobrazují jednoduché informace, které lze pochopit při letmém pohledu na takovýto widget. Widgety lze umístit kamkoliv na výchozí obrazovku, lze s nimi pohybovat a měnit jejich velikost. Mezi typické widgety patří například widget pro zobrazení počasí (viz Obrázek 9) nebo pro zobrazení aktuálního data a času (*App Widgets Overview*, 2019).

Obrázek 9 - Widget pro zobrazení počasí

*(*App Widgets Overview*, 2019)*

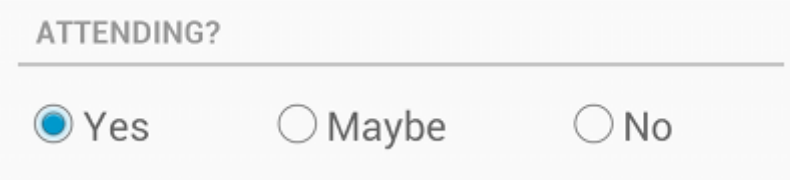
Widgety fungují od verze Androidu 1.5. Pro správnou funkčnost je potřeba vybrat vhodnou velikost widgetu, ta se vypočítá pomocí vzorce 70 x n – 30, takže pro jeden řádek nebo sloupec to je 40dp, pro dva řádky a sloupce to je 110dp, pro tři 180dp atd. V xml konfiguračním souboru se používá pro specifikaci rozměrů u widgetů parametr minWidth (minimální šířka) a minHeight (minimální výška), která je ve skutečnosti výchozí velikost widgetu. Pro nastavení menší velikosti existuje parametr minResizeWidth a minResizeHeigh, ty umožňují widget ještě více zmenšit. Dále je možno nastavit, zda u widgetu lze měnit velikost pouze horizontálně, nebo vertikálně, k tomu slouží parametr resizeMode=„horizontal|vertical". Pro nastavení obrázku, který se zobrazí v seznamu widgetů, se použije parametr previewImage, příklad nastavení v xml je možno vidět níže (*App Widget Design Guidelines*, 2019).

```
android:minHeight="30dp"
android:minWidth="110dp"
android:minResizeWidth="110dp"
android:minResizeHeight="30dp"
android:previewImage="@drawable/widget2x1"
android:resizeMode="horizontal|vertical"
```

3.3.10 Přepínače pro výběr volby

V aplikaci Revolution Launcher je v sekci menu možnost vybrat si oblíbený launcher. Tato funkce je dostupná díky funkci zvané Radio Buttons.

Tato funkce umožňuje uživateli vybrat jednu položku z nabízeného seznamu (viz Obrázek 10).



Obrázek 10 - Vzhled funkce Radio Buttons

*(*Radio Buttons, *2019)*

Pro vytvoření funkce Radio Buttons je potřeba nejdříve vytvořit v layoutu skupinu rádiových tlačítek (anglicky RadioGroup) a poté do této skupiny vložit jednotlivá tlačítka nazvaná RadioButton, a vytvořit tak možnost přepínat mezi nimi, tímto seskupením se sdělí systému, že pouze jedno tlačítko může být vybráno. Pokud uživatel klikne na nějaké RadioButton, objekt obdrží on-click událost, tedy je sděleno systému, že bylo na tento objekt kliknuto. Pro nastavení události kliknutí je potřeba každému RadioButton přidat v xml souboru android:onClick. Jméno musí odpovídat názvu metody, pomocí které chceme na tuto událost reagovat. Níže je uveden příklad použití:

```
<RadioGroup
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical">
     <RadioButton android:id="@+id/radio_pirates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/pirates"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/radio_ninjas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ninjas"
        android:onClick="onRadioButtonClicked"/>
</RadioGroup>
```

Nakonec je potřeba v aktivitě vytvořit metodu, pomocí které chceme reagovat na kliknutí, to je možno udělat například pomocí tohoto kódu:

```
public void onRadioButtonClicked(View view) {
// Is the button now checked?
boolean checked = ((RadioButton) view).isChecked();
```

```
// Check which radio button was clicked
switch(view.getId()) {
case R.id.radio_pirates:
if (checked)
// Pirates are the best
break;
case R.id.radio_ninjas:
if (checked)
// Ninjas rule
break;
}
}
```

Důležité je, aby metoda deklarovaná v android:onClick atributu měla přesný charakteristický rys, jako je zobrazen výše. Především musí tato metoda být veřejná (public), nic nevracet (return Void) a mít definován View jako jediný parametr (*Radio Buttons*, 2019).

3.4 Problematika provozu aplikací s a bez root oprávnění

Tato část se věnuje problematice provozu telefonu s a bez root oprávnění a také ukazuje obecný návod, jak lze získat root práva. Root v překladu z angličtiny znamená kořen, v přeneseném smyslu root oprávnění znamenají přístup do kořenového adresáře s pravomocemi jako super user, což v operačním systému znamená hlavní administrátor, který má oprávnění měnit i věci, které jsou standardně zakázané. V aplikaci Revolution Launcher je možno změnit rozlišení a velikost dpi jen pomocí root oprávnění.

3.4.1 Jak získat root oprávnění pro telefon

Postup, jak získat oprávnění, je někdy velmi jednoduchý a lze jej provést pomocí stažené aplikace, která uživatele navádí krok po kroku. Jindy je potřeba využít Android Debug Bdrige (ADB) nebo jiné metody vyžadující speciální software, který je potřeba si stáhnout do počítače. Například pro udělání administrátorských práv u telefonů Samsung je nutno nejdříve v menu telefonu povolit developer mode (provádí se v menu telefonu, položka o telefonu, softwarové informace a poté 7x kliknout na položku jménem číslo sestavení), poté se zpřístupní položka vývojářské možnosti a zde je nutno povolit ladění USB (anglicky USB debugging). V dalším kroku se do stolního počítače nainstaluje software Odin a stáhnou se soubory, určené pro root telefonu. Stažené soubory se přes aplikaci Odin nahrají do telefonu, který běží v režimu Download Mode (do tohoto módu se lze dostat po restartu telefonu držení tlačítka Volume down, Home Button + Power Key, pokud telefon nemá Home Button, podrží se tlačítko Bixby), následně se nahraje speciální software (například TWRP – zkratka znamená TeamWin Recovery Project), který nainstaluje nový software pro obnovu systému. Po restartu telefonu je potřeba znovu vstoupit do recovery módu a zde nahrát další software, který zajistí root přístup (A Selva, 2019).

3.4.2 Použít root oprávnění v aplikaci Revolution Launcher

Aplikace Revolution Launcher zatím funguje pouze s telefony, které mají root oprávnění. V několika diskuzích bylo zmíněno, že je možné změnit rozlišení i počet bodů bez root práv a dokonce i některé aplikace takto umí fungovat, například Tasker nebo Second Screen. Aplikace Second Screen má svůj zdrojový kód sdílený na serveru GitHub, dle provedeného průzkumu by mělo stačit získat oprávnění do Secure Settings a poté zadat níže uvedený příkaz:

```
Settings.Global.putString(
mContext.getContentResolver(),
```

```
Settings.Global.DISPLAY_SIZE_FORCED, width + "," + height);

Process process = Runtime.getRuntime().exec("wm density 220");
process.waitFor();
```

Bohužel tento kód funguje jen částečně na některých telefonech (změní pouze počet dpi) nebo nefunguje vůbec. Pro změnu rozlišení pomocí root oprávnění je použit kód:

```
process = Runtime.getRuntime().exec("su -c \"\"wm density
240");
process.waitFor();
process = Runtime.getRuntime().exec("su -c \"\"wm size
1080x1920");
process.waitFor();
```

Přesný popis změny nastavení rozlišení v aplikaci Revolution Launcher je popsán v kapitole 4.2.1, včetně nastavení widgetů, pomocí kterých se změna provádí.

3.4.3 Výhody telefonů s root oprávněním

Mezi hlavní výhodu telefonů s root oprávněním patří možnost si cokoliv v telefonu změnit, nejčastěji se využívají programy pro blokování reklamy ve webovém prohlížeči, ale také ve všech programech běžících v telefonu. Existuje i upravená verze programu YouTube, která funguje bez reklam. Další velkou výhodou telefonů s root oprávněním je delší životnost baterie na jedno nabití. Tato funkcionalita je zařízena pomocí odinstalování programů, které standardně odinstalovat není možné a také díky speciálním programům, které umožňují například snížit takt procesoru nebo snížit počet intervalů, kdy se telefon probouzí. Další výhody jsou možnost kompletní zálohy všech programů včetně programů nebo změna bootovací logo („10 Advantages and Disadvantages of Rooting Android devices", 2017).

3.4.4 Problémy s telefony s root oprávněním

První nejdůležitější nevýhodou je to, že pro většinu nezkušených lidí je extrémně složitý samotný proces získání oprávnění, stejně tak nastavení telefonu, aby fungoval bez problému, zabere hodně práce a plno zkoumání.

Mezi další nevýhody patří nefunkčnost streamovací aplikace NetFlix a nemožnost platit pomocí Google Pay (telefon funguje podobně jako debetní platební karta), což sice lze obejít, ale je to velmi rizikové v případě napadení telefonu škodlivou aplikací, která by mohla získat detaily o kreditní kartě a provést platbu, která by pro majitele telefonu mohla znamenat velkou finanční ztrátu. Nakonec existuje možnost, že nezkušený uživatel si telefon poškodí tak, že již

ho nebude možno opravit, bohužel většina výrobců telefonů při takovémto neoprávněném zásahu telefonu neuznává záruku („10 Advantages and Disadvantages of Rooting Android devices", 2017).

3.5 SW řešení pomocí aplikací Tasker, AutoTools, Second Display

Tato kapitola popisuje návod, jak propojit funkcionality několika aplikací stažených z Google Play a zkombinovat je do jednoho funkčního celku, který z telefonu udělá desktopový počítač.

Seznam použitých aplikací:

- **Tasker** – jedná se o placenou aplikaci, která umí automatizovat mnoho funkcí. Například pokud je v kalendáři uvedena schůzka, tak aplikace Tasker vypne v době schůzky všechny zvuky a po ukončení je zase všechny zapne.
- **AutoApps** – jedná se o plugin, tedy o přídavný modul pro aplikaci Tasker. Samotná aplikace pouze spravuje další moduly ze skupiny AutoApps.
- **AutoTools** – placená aplikace ze skupiny AutoApps. Z funkcí bude použita možnost přepínat programově mezi dvěma launchery.
- **Second Display** – aplikace pro změnu rozlišení a hustotu displeje. Lze používat samostatně nebo jako plugin aplikace Tasker.
- **Apex Launcher** – launcher, který lze přizpůsobit pro potřeby domovské stránky, když telefon běží v režimu stolního počítače.
- **Nova Launcher** – launcher, který je určen pro domovskou stránku v případě, kdy telefon běží v režimu běžného telefonu.
- **External Keyboard Helper** – placená aplikace vhodná při připojení externí klávesnice.
- **Keyboard SwiftKey** – softwarová klávesnice, je možno použít jakoukoliv oblíbenou.

Krok 1: Nastavení Secure settings

Po stáhnutí všech aplikací z Google Play je potřeba nastavit povolení Secure Settings k aplikacím Tasker, Second Screen a AutoTools. O funkcích Secure settings je pojednáno v kapitole 3.1.

Postup nastavení Secure settings:

1. Stáhnout aplikaci „Universal ADB Drivers“ ze stránek adb.clockworkmod.com a tuto aplikaci nainstalovat
2. Na mobilním telefonu s operačním systémem Android zapnout Developer Mode

3. Zapnout funkci „Enable USB Debugging“ – toto povolí používat funkci Android Debug Bridge (ADB)
4. Ze stejného adresáře, kde je nainstalována aplikace „Universal ADB Drivers“, spustit příkazový řádek
5. Do příkazového řádku napsat pro aplikaci Tasker:
    - adb shell pm grant net.dinglisch.android.taskerm android.permission.WRITE_SECURE_SETTINGS
6. Pro aplikaci Second Screen napsat:
    - adb shell pm grant com.farmerbb.secondscreen.free android.permission.WRITE_SECURE_SETTINGS
7. Pro aplikaci AutoTools napsat:
    - adb shell pm grant com.joaomgcd.autotools android.permission.WRITE_SECURE_SETTINGS¨

Krok 2: Nastavení aplikace Second Screen

V aplikaci Second Screen je potřeba vytvořit dva profily, jeden s názvem „Desktop resolution“ a druhý s názvem „Phone resolution“

Nastavení profilu „Desktop resolution“:

- Resolution: 1920x1080
- Density: 240dpi
- Screen orientation: Lock orientation - landscape

Nastavení profilu „Phone resolution“:

- Resolution: Device native
- Density: Device native
- Screen orientation: Use system setting

Krok 3: Nastavení aplikace External Keyboard Helper

V této aplikaci je nutno vybrat výchozí rozložení klávesnice, standardně Czech (QWERTZ). Je také možno vybrat i druhou klávesnici, například anglickou, přepínání mezi klávesnicemi se provádí standardně jako v operačním systému Windows, tedy zkratkou levý alt + shift.

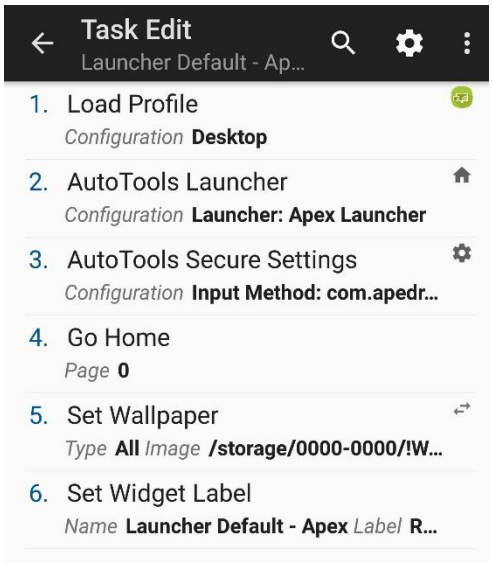
Krok 4: Nastavení aplikace Keyboard SwiftKey

V této aplikaci je potřeba vybrat výchozí rozložení klávesnice, standardně Czech (QWERTZ)

Krok 5: Nastavení aplikace Tasker pro přepnutí z telefonního rozhraní do desktopového vzhledu

Je potřeba vytvořit novou úlohu, v aplikaci Tasker se tato úloha nazývá Task. Tento Task se provede poté, co bude kliknuto na widget, který bude umístěn na ploše. Úloha se bude jmenovat „Launcher Default – Apex Launcher“. Následně jsou popsány kroky vytvoření úlohy, kroky jsou napsány v angličtině. Vysvětlení funkčnosti bude vysvětleno v dalším textu. Konfigurace programu Tasker viz Obrázek 11.

- Plugin Second Screen: Run profile „Desktop resolution “
- Plugin AutoTools set default launcher Apex Launcher
- Change default keyboard to External Keyboard Helper
- Go Home page 0
- Set Wallpaper
- Change Widget name to „Run AL” (změní název widgetu)



Obrázek 11 - Nastavení aplikace Tasker pro přepnutí z telefonního rozhraní do desktopového vzhledu (vlastní dílo autora)

Krok 6: Nastavení aplikace Tasker pro přepnutí z desktopového rozhraní do telefonního vzhledu

Stejně jako v předchozím kroku je potřeba vytvořit novou úlohu, která se bude jmenovat „Launcher Default – Nova Launcher“

- Plugin Second Screen: Run profile „Phone resolution“
- Plugin AutoTools set default launcher Nova Launcher

- Change default keyboard to Keyboard SwiftKey
- Go Home page 0
- Set Wallpaper
- Change Widget name to ,,Run NL”

Krok 7: Nastavení aplikace AutoTools

- V samotné aplikaci není potřeba nic nastavit, ale přímo v telefonu je nutno zvolit aplikaci AutoTools jako výchozí launcher.

Krok 8: Nastavení aplikace Apex Launcher

Nejdříve je nutno nastavit orientaci na šířku (Landscape). V dalším kroku Vypnout dolní umístění ikon v takzvaném docku na domácí stránce. Dále zmenšit velikost ikon z hodnoty 100 na 60, protože po připojení telefonu k monitoru je vhodnější, když ikony mají menší velikost. Nechat pouze jednu hlavní obrazovku, není dobré, pokud by ikony byly na více jak jedné ploše, z důvodu, že ani na osobních počítačích většina lidí nepoužívá více jak jednu plochu. Vložit na plochu widget z aplikace Tasker. Z nabídky je nutno vybrat Task jménem „Launcher Default – Apex Launcher“.

Krok 8: Nastavení aplikace Nova Launcher

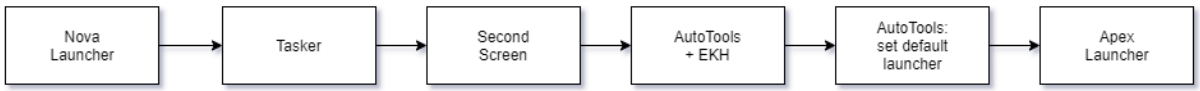
Vložit na plochu widget z aplikace Tasker. Z nabídky je nutno vybrat task jménem „Launcher Default – Nova Launcher“.

3.5.1 Popis funkčnosti přepnutí z telefonního rozhraní do desktopového módu

1. Díky povolení Secure settings mají aplikace přístup ke speciálním funkcím telefonu, a mohou měnit například rozlišení, počet DPI, výchozí launcher atd.
2. V aplikaci Nova Launcher uživatel klikne na widget „Run AL“ (AL je zkratka pro Apex Launcher), touto akcí se spustí task jménem „Launcher Default – Apex Launcher“.
3. Aplikace Tasker zavolá plugin Second Screen, aby spustil profil „Desktop resolution“. Tento plugin nastaví rozlišení na 1920x1080 (Full HD) a změní počet bodů na 240 DPI.
4. Aplikace Tasker pomocí pluginu AutoTools nastaví výchozí klávesnici External Keyboard Helper (zkráceně EKH). Díky tomuto nastavení nebude vidět softwarová klávesnice na obrazovce.

5. Aplikace Tasker spustí plugin AutoTools, který nastaví výchozí launcher aplikaci Apex Launcher.
6. Aplikace Tasker simuluje zmáčknutí tlačítka domů. Tato akce způsobí přepnutí Nova Launcheru na Apex Launcher.
7. Aplikace Tasker nastaví uživatelem vybraný obrázek, který je vhodný při používání launcheru na šířku.

Flow chart běhu programu je popsán níže (Obrázek 12)



Obrázek 12 - Flow chart - přepnutí z normálního prostředí do desktopového módu

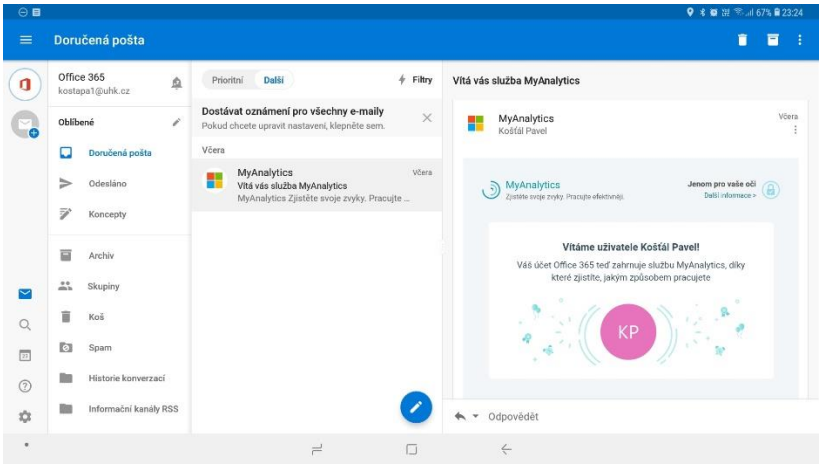
(vlastní dílo autora)

Vzhled Apex Launcheru při používání jako desktopové prostředí (Obrázek 13)

Obrázek 13 - Desktopové prostředí v aplikaci Apex Launcher

(vlastní dílo autora)

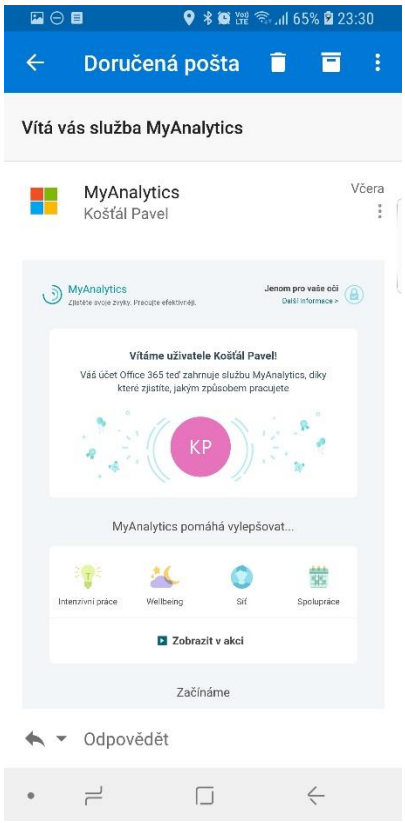
Níže je možno vidět porovnání aplikace Microsoft Outlook spuštěnou v režimu desktopu (Obrázek 13) a poté v režimu telefonu (Obrázek 14).



Obrázek 14 - Microsoft Outlook v desktopovém režimu

(vlastní dílo autora)

Na obrázku výše je vidět, že v desktopové verzi je zobrazeno více informací než v telefonním režimu. Změna uživatelského prostředí v operačním systému Androidu je prováděna, dle velikosti současného rozlišení a počtu bodů (další informace viz kapitola 3.3.5).



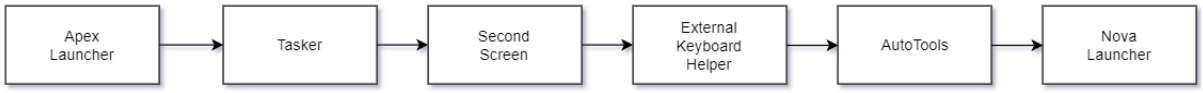
Obrázek 15 - Microsoft Outlook v telefonním režimu

(vlastní dílo autora)

3.5.2 Popis funkčnosti přepnutí z desktopového rozhraní do telefonního módu

1. V aplikaci Apex Launcher uživatel klikne na widget „Run NL“, touto akcí se spustí task v aplikaci Tasker jménem „Launcher Default – Nova Launcher“.
2. Aplikace Tasker zavolá plugin Second Screen, aby spustil profil „Phone resolution“. Tento plugin nastaví rozlišení a počet bodů na výchozí hodnoty.
3. Aplikace Tasker pomocí pluginu AutoTools nastaví výchozí klávesnici SwiftKey. Tímto krokem bude možno zadávat text pomocí softwarové klávesnice.
4. Aplikace Tasker simuluje zmáčknutí tlačítka domů. Tato akce způsobí přepnutí Nova Launcheru na Nova Launcher.
5. Aplikace Tasker simuluje zmáčknutí tlačítka domů. Tato akce způsobí přepnutí Apex Launcheru na Nova Launcher.
6. Aplikace Tasker nastaví uživatelem vybraný obrázek, který je vhodný při používání launcheru na výšku.

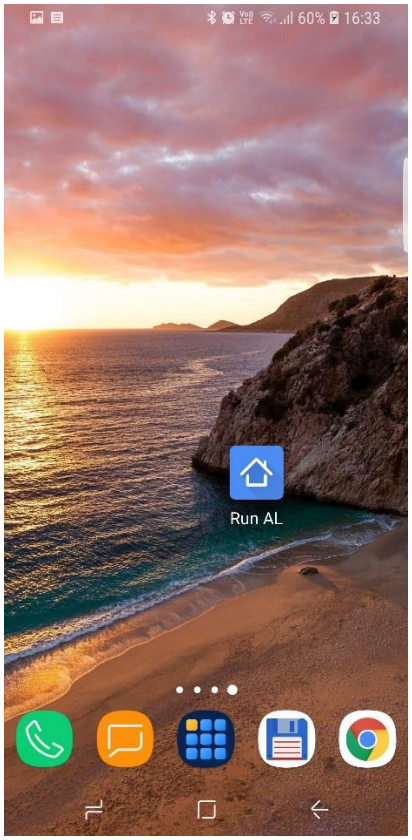
Flow chart běhu programu je popsán níže, viz Obrázek 16.



Obrázek 16 - Flow chart - přepnutí z desktopového prostředí do normálního telefonního vzhledu

(vlastní dílo autora)

Po přepnutí do normálního telefonního zobrazení vypadá prostředí v aplikaci Nova Launcher jako na běžném mobilním telefonu (Obrázek 17).



Obrázek 17 - Vzhled aplikace Nova Launcher při používání telefonního zobrazení

(vlastní dílo autora)

3.5.3 Shrnutí řešení pomocí aplikace Tasker

Mezi hlavní výhody tohoto řešení patří, že lze použít na jakémkoliv telefonu a není potřeba root oprávnění. Díky složitému postupu bude pro většinu uživatelů velmi problematické vše nastavit, bez podrobného návodu, který je potřeba následovat krok po kroku. Další závažný problém je aktualizace pomocných aplikací, ta občas způsobí nefunkčnost celého systému a je potřeba zjistit, kde nastal problém a vše znovu přenastavit. Občas je nutno pouze restartovat telefon. Všechny tyto problémy pro osobní použití nejsou žádný větší problém, pokud si uživatel umí vše opravit svépomocí. V případě, že by se uvažovalo o nasazení tohoto řešení ve větší míře, například v nějaké firmě, kladlo by to velké nároky na technickou podporu, která by velmi často musela řešit nefunkčnost tohoto řešení. Z tohoto důvodu je nejlepší vytvořit aplikaci, která jednoduše umožní používat telefon běžným způsobem a pokud to bude potřeba, udělat z telefonu alternativu osobního počítače. Postup vytvoření takovéto aplikace je popsán v dalších kapitolách.

4 Návrh a implementace řešení aplikace pro adaptaci mobilního zařízení pro desktopový provoz

V dalších kapitolách je popsán vývoj aplikačního řešení, díky kterému lze nastavit telefon tak, aby šel používat jako stolní počítač. Pro vytvoření této aplikace je použita aplikace Android Studio a programovací jazyk Java. Nejdříve je popsán vývoj změny launcheru, dále je uveden postup změny rozlišení a nakonec je popsáno řešení dalších vlastností programu, jako jsou vytvoření manuálu, bezdrátové připojení pomocí Google Chromecast.

4.1 Vývoj aplikačního řešení změny launcheru

Z důvodu, že výše uvedený postup za použití aplikace Tasker nefunguje na 100 %, je nejlepší řešení pro změnu z telefonního prostředí do desktopového vytvoření aplikace v Android Studio. Tato nová aplikace je pojmenovaná Revolution Launcher.

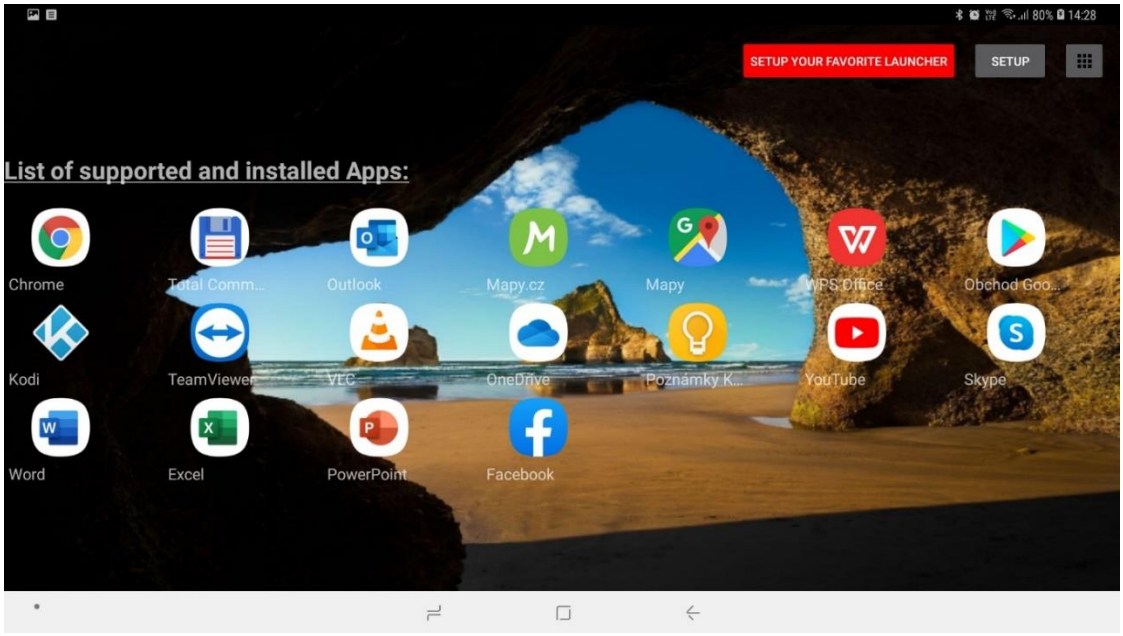
První věc, která musí být vytvořena, je třída MainActivity, tedy hlavní třída, která je spuštěna při startu aplikace. V této části je ale popis vytvoření této třídy přeskočen a bude vytvořena třída pojmenovaná MainActivityReal. Třída MainActivity bude sloužit pouze k přepínání mezi oblíbeným launcherem, který bude použit pro telefonní prostředí a aplikací Revolution Launcher, která bude zobrazovat počítačové prostředí. Postup vytvoření třídy MainActivity bude popsán později.

Pro funkcionalitu programu je nutno dopsat do xml souboru AndroidManifest.xml kód, díky kterému operační systém bude vnímat aplikaci Revolution launcher jako launcher.

```
<activity
android:name=".MainActivityReal"
android:launchMode="singleTask"
android:screenOrientation="landscape"
android:stateNotNeeded="true">
```

4.1.1 Vytvoření třídy MainActivityReal

Pojmenování této třídy s přídavkem Real je zvoleno z důvodu, že toto je opravdová hlavní aktivita, která se zobrazí po spuštění aplikace Revolution Launcher. Vzhled je možno vidět na níže uvedeném obrázku (Obrázek 18).



Obrázek 18 - Úvodní stránka aplikace Revolution Launcher

(vlastní dílo autora)

Při vytvoření této aktivity je nejdříve nutno vytvořit seznam všech aplikací, které jsou vhodné pro běh v desktopovém prostředí. Nikde neexistuje žádný oficiální seznam a zjišťování podporovaných aplikací je zdlouhavý výzkum, který spočívá ve stažení aplikace z Google Play a vyzkoušení této aplikace, zda je vhodná pro desktopové prostředí. Níže je uveden seznam aplikací, které plně podporují desktopové prostředí:

- Google Chrome, Firefox, Microsoft One Note, Total Commander, Microsoft Outlook, Mapy.cz, Google Maps, WPS Office, Google Play, Kodi, TeamViewer, VLC for Android, OneDrive, Google Keep, YouTube, Google Drive, Twitch, Skype

Následující seznam obsahuje aplikace, které lze také použít v desktopovém prostředí, ale nejsou 100 % přizpůsobeny:

- Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Facebook

Seznam aplikací je uložen do pole s názvem: supportedAppPackageNameArray. V dalším kroku je potřeba zjistit, které z výše uvedených aplikací jsou v mobilním telefonu nainstalovány, k tomu slouží tento kód:

```
for (int i = 0; i < supportedAppPackageNameArray.length; i++)
{
PackageManager packageManager = getPackageManager();
try {

packageManager.getPackageInfo(supportedAppPackageNameArray[i],
PackageManager.GET_ACTIVITIES);
```

```
supportedAndInstalledAppArrayList.add(supportedAppPackageNameA
rray[i]);
} catch (Exception e) {
}
}
```

Po provedení tohoto kódu je seznam podporovaných a nainstalovaných aplikací uložen do pole supportedAndInstalledAppArrayList. Pro zobrazení ikon na ploše je potřeba použít funkcionalitu jménem GridView, funkce je detailně popsána v části 3.3. Pro vytvoření GridView je nutno připravit xml soubor pojmenovaný „app_drawer_adapter.xml“ (bude použit i pro funkci App Drawer, viz kapitola 4.1.2), v tomto xml souboru je potřeba nadefinovat vzor pro ikonu včetně umístění textu reprezentující odkaz na nainstalovaný program. Xml konfigurace vypadá následovně:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent">

    <ImageView
        android:id="@+id/icons"
        android:layout_width="70dp"
        android:layout_height="70dp"
        android:layout_alignParentStart="true"
        android:layout_marginStart="30dp"
        android:layout_marginTop="8dp"
        app:srcCompat="@mipmap/ic_launcher_round" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="120dp"
        android:layout_height="25dp"
        android:layout_below="@+id/icons"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_centerHorizontal="true"
        android:layout_marginLeft="5dp"
        android:layout_marginTop="5dp"
        android:ellipsize="end"
        android:maxLines="1"
        android:text="A"
        android:textSize="18sp" />
</RelativeLayout>
```

Defaultní vzhled je potom možno vidět níže (Obrázek 19).

Obrázek 19 - Grafický vzhled předlohy v XML souboru pro zástupce aplikace

(vlastní dílo autora)

V dalším kroku je nutné vytvořit adaptér pro GridView. Adaptér slouží k propojení xml souboru s proměnnými a k vrácení informace o tom, která ikona byla zmáčknuta. Třída je tvořena níže uvedeným kódem:

```
public class AppDrawerAdapter extends BaseAdapter {

private Drawable icons[];
private String letters [];
private Context context;
private LayoutInflater inflater;

public AppDrawerAdapter(Context context, Drawable icons[],
String letters[]) {
this.context = context;
this.icons=icons;
this.letters=letters;
}

@Override
public int getCount() {
return letters.length;
}

@Override
public Object getItem(int i) {
return letters[i];
}

@Override
public long getItemId(int i) {
return i;
}

@Override
public View getView(int i, View view, ViewGroup viewGroup) {
View gridView = view;

if(view==null) {
inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
```

```
gridView = inflater.inflate(R.layout.app_drawer_adapter,
null);
}

ImageView icon = (ImageView)
gridView.findViewById(R.id.icons);
TextView letter = (TextView)
gridView.findViewById(R.id.textView);

icon.setImageDrawable(icons[i]);

letter.setText(letters[i]);

return gridView;
}
}
```

Nakonec je potřeba vložit ikony na plochu, to se udělá pomocí metody nazvané iconSetupGridView, začátek metody je standardní vyhledání prvku GridView, proto je zde uvedena pouze druhá část metody:

```
appNameArrayList.add(packageManager.getApplicationLabel(packag
eManager.getApplicationInfo(supportedAndInstalledAppArrayList.
get(i) + "", PackageManager.GET_META_DATA)));

} catch (PackageManager.NameNotFoundException e) {
e.printStackTrace();
                }
        }

String[] appNameList = appNameArrayList.toArray(new
String[appNameArrayList.size()]);
Drawable[] iconDrawableList =
iconDrawableArrayList.toArray(new
Drawable[iconDrawableArrayList.size()]);

AppDrawerAdapter adapter = new
AppDrawerAdapter(MainActivityReal.this, iconDrawableList,
appNameList);
gridView.setNumColumns(7);
gridView.setAdapter(adapter);
gridView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> adapterView, View view,
int i, long l) {
```

```
Intent intent =
getPackageManager().getLaunchIntentForPackage(supportedAndInst
alledAppArrayList.get(i) + "");
startActivity(intent);

            }
        });
    }
```

V dalším kroku je vytvořeno zobrazení podporovaných a nainstalovaných aplikací na hlavní ploše. Následně je potřeba vytvořit tlačítko, pomocí kterého bude uživatel přepínat z Revolution Launcher na jeho oblíbený launcher, který bude běžet v telefonním rozhraní. Nejdříve je potřeba zjistit, zda je Revolution launcher výchozí, k tomu slouží tato metoda:

```
boolean isMyLauncherDefault() {
final Intent intentCategory = new Intent(Intent.ACTION_MAIN);
intentCategory.addCategory(Intent.CATEGORY_HOME);
final ResolveInfo resolveInfo =
getPackageManager().resolveActivity(intentCategory, 0);
if (resolveInfo.activityInfo != null && getPackageName()
.equals(resolveInfo.activityInfo.packageName)) {
return true;
}
return false;
}
```

Pokud je Revolution launcher již nastaven jako výchozí launcher, tak tato metoda nastaví viditelnost tlačítka jménem buttonFavoriteLauncherSwitch na viditelné. V případě, že Revolution launcher není nastaven jako výchozí, uživateli se zobrazí vyskakovací okno, které mu nabídne, aby si tento launcher nastavil jako výchozí. Pomocí další metody se testuje, zda je již nastaven nějaký oblíbený launcher. Pokud není nastaven, zobrazí se vyskakovací okno, které nabídne nastavení oblíbeného launcheru a také se změní text buttonFavoriteLauncherSwitch na „Setup your favorite launcher“ a pozadí tohoto tlačítka bude červené. V případě, že je oblíbený launcher již nastaven, tlačítku se změní pozadí na zelené a text se změní například na „Switch to Nova Launcher“.

V této chvíli je možné spouštět pouze podporované aplikace pro desktopové rozlišení, ale uživatel potřebuje občas používat i jiné aplikace, které má nainstalované, například pro posílání

SMS nebo telefonování. K tomuto účelu je vytvořeno tlačítko, které po zmáčknutí přepne na aktivitu App Drawer, která vytvoří seznam všech nainstalovaných aplikací.

4.1.2 Nastavení aktivity Application Drawer

Pro zobrazení seznamu nainstalovaných aplikací bude použit stejný adaptér a xml konfigurace, jako je použita pro aplikace zobrazené na hlavní ploše. V tomto případě je ještě potřeba zjistit všechny nainstalované aplikace, které ale nejsou součástí operačního systému. V první části se vyfiltrují všechny uživatelské aplikace a uloží se do pole „listOfAllUserApps“. Kód vypadá následovně:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.app_drawer);

final PackageManager packageManager = getPackageManager();
Intent intent = new Intent(Intent.ACTION_MAIN, null);
intent.addCategory(Intent.CATEGORY_LAUNCHER);
List<ResolveInfo> resInfos =
packageManager.queryIntentActivities(intent, 0);

HashSet<String> packageNames = new HashSet<String>(0);
List<ApplicationInfo> listOfAllUserApps = new
ArrayList<ApplicationInfo>(0);

for(ResolveInfo resolveInfo : resInfos) {

packageNames.add(resolveInfo.activityInfo.packageName);
            }

for(String packageName : packageNames) {
try {

listOfAllUserApps.add(packageManager.getApplicationInfo(packag
eName, PackageManager.GET_META_DATA));
} catch (PackageManager.NameNotFoundException e) {
//Do Nothing
                }
            }
```

V dalším kroku se aplikace zobrazí pomocí adaptéru uživateli na obrazovce:

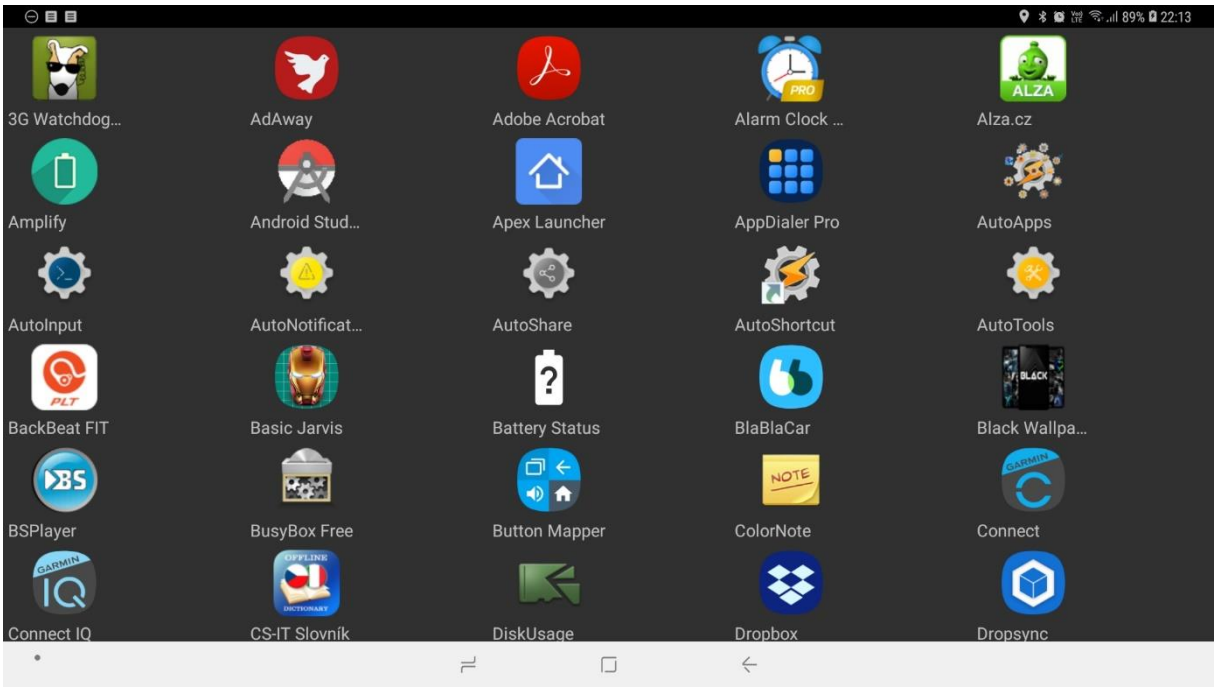
```
Collections.sort(listOfAllUserApps, new
ApplicationInfo.DisplayNameComparator(packageManager));

ArrayList<CharSequence> appNameArrayList = new ArrayList<>();
ArrayList<Drawable> iconDrawableArrayList = new ArrayList<>();
```

```
final ArrayList<String> packageNameArrayList = new
ArrayList<>();

for (int i = 0; i < listOfAllUserApps.size(); i++ ){
                try {

appNameArrayList.add(packageManager.getApplicationLabel(packag
eManager.getApplicationInfo(listOfAllUserApps.get(i).packageNa
me + "", PackageManager.GET_META_DATA)));

iconDrawableArrayList.add(getPackageManager().getApplicationIc
on(listOfAllUserApps.get(i).packageName + ""));

packageNameArrayList.add(listOfAllUserApps.get(i).packageName+
"");
} catch (PackageManager.NameNotFoundException e) {
e.printStackTrace();
                }


            }
String[] appNameListArray = appNameArrayList.toArray(new
String[appNameArrayList.size()]);

Drawable[] iconDrawableArray =
iconDrawableArrayList.toArray(new
Drawable[iconDrawableArrayList.size()]);

gridView = findViewById(R.id.gridView);
AppDrawerAdapter adapter = new
AppDrawerAdapter(AppDrawer.this, iconDrawableArray,
appNameListArray);
gridView.setNumColumns(5);//set number of columns
gridView.setAdapter(adapter);
gridView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

@Override
public void onItemClick(AdapterView<?> adapterView, View view,
int i, long l) {

Intent intent =
getPackageManager().getLaunchIntentForPackage(packageNameArray
List.get(i)+"");
startActivity(intent);

}
});
}
```

V tomto kódu je možno změnit počet řádků, nejvhodnější hodnota byla zvolena 5 sloupců, protože poté je dobře vidět seznam v desktopovém režimu, ale i v případě, že Revolution Launcher není ještě nastaven a rozlišení je stále výchozí, tak se ikony zobrazují bez problému. Nastavení počtu řádků:

```
gridView.setNumColumns(5);
```

Na níže uvedeném obrázku (Obrázek 20) je možno vidět vzhled seznamu aplikací.



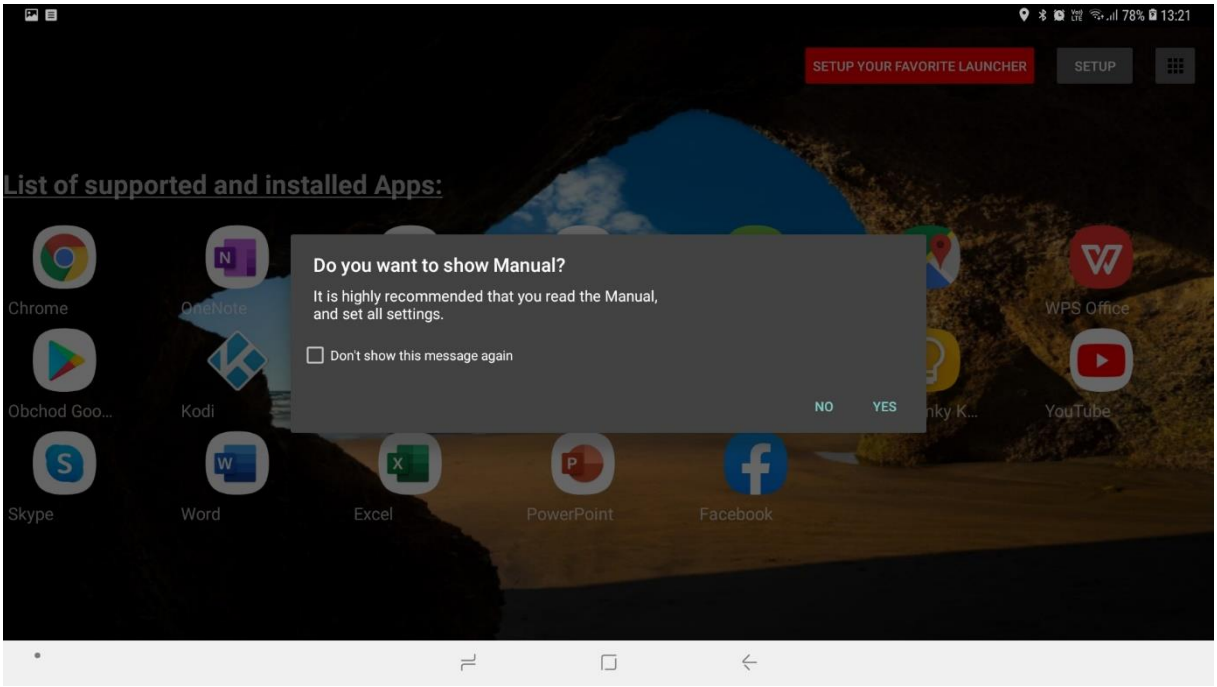
Obrázek 20 - Vzhled seznamu aplikací - App drawer

(vlastní dílo autora)

Aby hlavní stránka byla kompletní, je ještě potřeba vytvořit tlačítko SETUP (česky nastavení), které uživatele přenese do menu, kde si může program přizpůsobit, toto je řešeno dále v kapitole 4.1.5.

4.1.3 Nastavení funkce Dialog alert s možností zaškrtnout volbu již znovu nezobrazovat

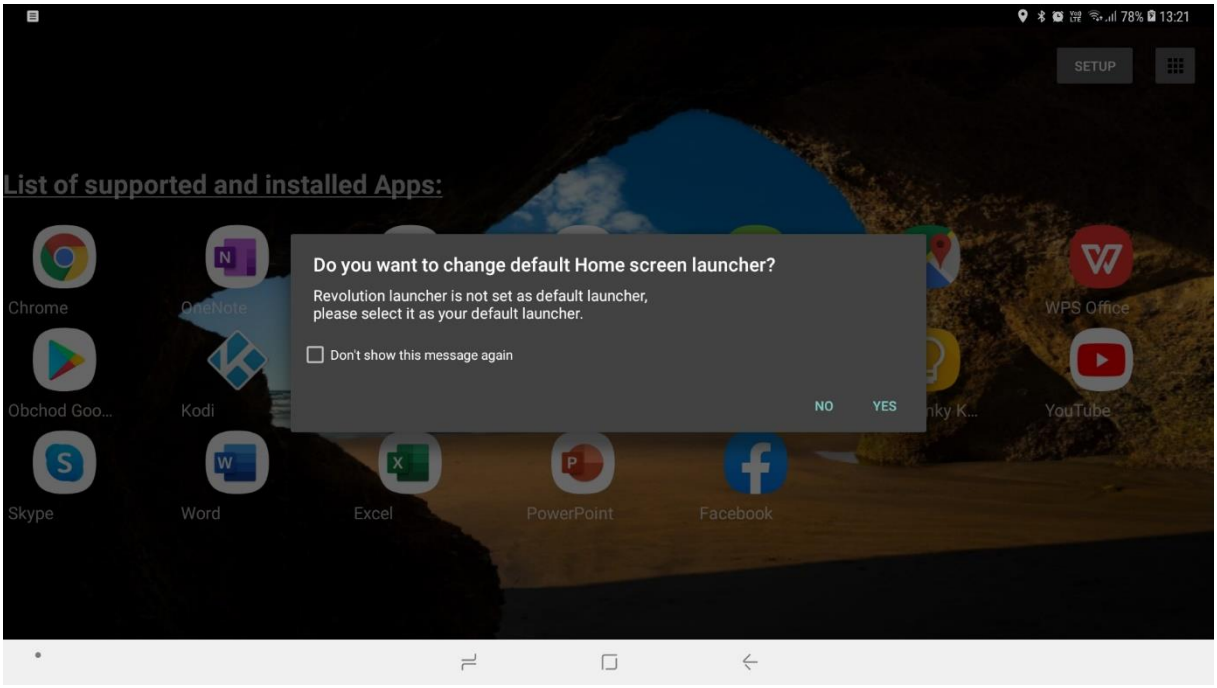
Pro správné použití aplikace Revolution Launcher je nutné nastavit tento launcher jako výchozí a poté jít do nastavení a zde podle návodu nakonfigurovat telefon. Aby uživatel tyto dvě podmínky splnil, je hned po startu aktivity MainActivityReal pokaždé zobrazeno okno, které nejdříve zobrazí dialog pro přechod do manuálu (Obrázek 21), kde je potřeba vše dle návodu nastavit (o funkcích uvedených v manuálu pojednává kapitola 4.3.1).



Obrázek 21 - Dialog zobrazující možnost přejít do manuálu

(vlastní dílo autora)

V dalším kroku je zobrazeno okno, které informuje uživatele, že Revolution Launcher není výchozí launcher (v případě, že je výchozí, tak se dialog nezobrazí). Vzhled je možno vidět níže (Obrázek 22).



Obrázek 22 - Dialog zobrazující možnost nastavit změnu výchozího launcheru

(vlastní dílo autora)

Vytvoření dialogu je relativně jednoduchá věc, pokud není vyžadováno zobrazení zaškrtávacího tlačítka zobrazující text „již znovu nezobrazovat“ (anglicky Don’t show this message again), poté je již postup komplikovanější.

První věc, která je potřeba udělat, je vytvořit vzhled tlačítka „již znovu nezobrazovat“, nastavení je následující:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="10dp">

    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Don't show this message again" />
</LinearLayout>
```

Grafická předloha vypadá takto (Obrázek 23):

Obrázek 23 - Grafická předloha pro volbu „již znovu nezobrazovat"

(vlastní dílo autora)

Zobrazení zaškrtávacího tlačítka se provede tímto kódem:

```
CheckBox checkBox = view.findViewById(R.id.checkBox);

AlertDialog alertDialog = builder.create();
alertDialog.show();
checkBox.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {

@Override
public void onCheckedChanged(CompoundButton compoundButton,
boolean b) {
if (compoundButton.isChecked()) {

setAlertDialogChangeDefaultLauncherToRevolutionLauncher(true);
} else {

setAlertDialogChangeDefaultLauncherToRevolutionLauncher(false)
;
                }
            }
```

```
        });

if (getAlertDialogChangeDefaultLauncherToRevolutionLauncher())
{
alertDialog.hide();
} else {
alertDialog.show();
}
```

Princip spočívá v zobrazení zaškrtávacího tlačítka přímo v dialogu. Po zaškrtnutí se stav tlačítka uloží do proměnné pomocí metody setAlertDialogChangeDefaultLauncherToRevolutionLauncher, která se pak načítá metodou getAlertDialogChangeDefaultLauncherToRevolutionLauncher. Jedná se jen o část kódu, zbytek kódu je běžné zobrazení dialogu s možností ano, nebo ne.

4.1.4 Nastavení aktivity MainActivity

MainActivity je aktivita, která se spustí při zapnutí programu. První krok, který se musí v této aktivitě zjistit, jsou hodnoty pro proměnné runFavoriteLauncher a favoriteLauncherPackageName. Pro ukládání hodnot je použita funkce PaperDB, o které je více napsáno v části 3.3.3. První proměnná říká, zda se má spustit oblíbený launcher nebo Revolution Launcher. Druhá proměnná má v sobě uloženo jméno balíčku oblíbeného launcheru. V případě, že je aplikace spuštěna poprvé, žádné hodnoty nejsou nastaveny (mají výchozí hodnotu null). Pokud je hodnota u runFavoriteLauncher null, je nutno ji přiřadit hodnotu false, to se udělá následujícím kódem:

```
if (runFavoriteLauncher == null) {
runFavoriteLauncher = "false";
Paper.book().write(runFavoriteLauncher, "false");
        }
```

Pokud je hodnota u favoriteLauncherPackageName null, je potřeba nastavit runFavoriteLauncher hodnotu na false, a to z toho důvodu, aby program správně fungoval. Pokud favoriteLauncherPackageName by nebyl nastaven a aplikace se chtěla přepnout na oblíbený launcher, vykonal by se neplatný kód a aplikace by se ukončila. Pro tuto ochranu je tento kód:

```
if (favoriteLauncherPackageName == null) {
runFavoriteLauncher = "false";
Paper.book().write("runFavoriteLauncher", "false");
        }
```

V dalším kroku se zkontroluje hodnota u runFavoriteLauncher, pokud je true, spustí se oblíbený launcher, pokud je false, spustí se Revolution Launcher.

Spuštění oblíbeného launcheru se provede pomocí tohoto kódu:

```
Intent launchFavoriteLauncher =
getPackageManager().getLaunchIntentForPackage(favoriteLauncher
PackageName);
startActivity(launchFavoriteLauncher);
```

Spuštění Revolution launcheru (tedy spuštění MainActivityReal) se provede pomocí tohoto kódu:

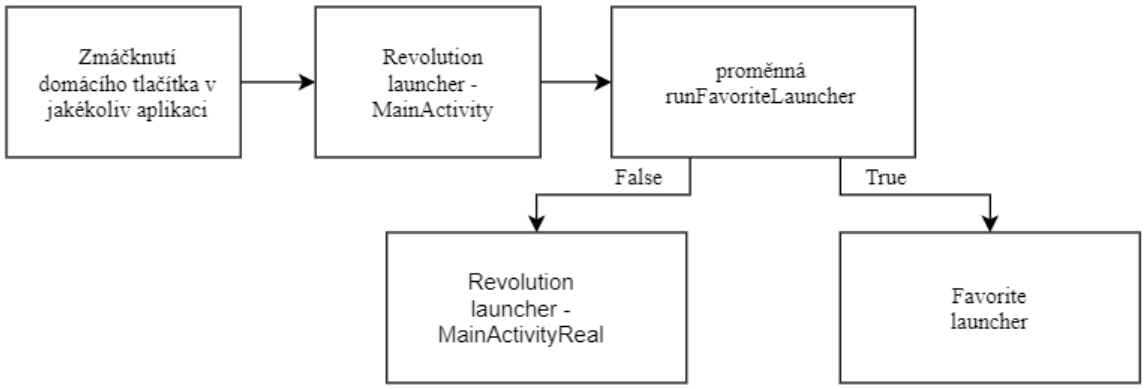
```
Intent intentRunRevolutionDesktop = new Intent(this,
MainActivityReal.class);
startActivity(intentRunRevolutionDesktop);
```

Poslední důležitá věc je nastavení, co se stane při vytvoření této aktivity. Tedy změnit kód, který následuje po onCreate:

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
//setContentView(R.layout.activity_main);
```

Z kódu je patrné, že se nespouští nastavení, které říká, jaký layout (které xml) se má nahrát při spuštění této aktivity. Je to z důvodu zrychlení přepnutí se z MainActivity do oblíbeného launcheru nebo při přechodu z MainActivity do MainActivityReal.

V této části tvorby programu aplikace funguje následovně. Revolution Launcher musí být nastaven jako výchozí launcher. Pokud je spuštěna jakákoliv aplikace, tak při zmáčknutí domácího tlačítka (Home button) se spustí aplikace Revolution Launcher a v aktivitě MainActivity se zkontroluje, jakou hodnotu má proměnná runFavoriteLauncher. Pokud je true, spustí se oblíbený launcher, v případě false se spustí aktivita MainActivityReal v aplikaci Revolution Launcher. Schéma je možno vidět na níže uvedeném diagramu (viz Obrázek 24).

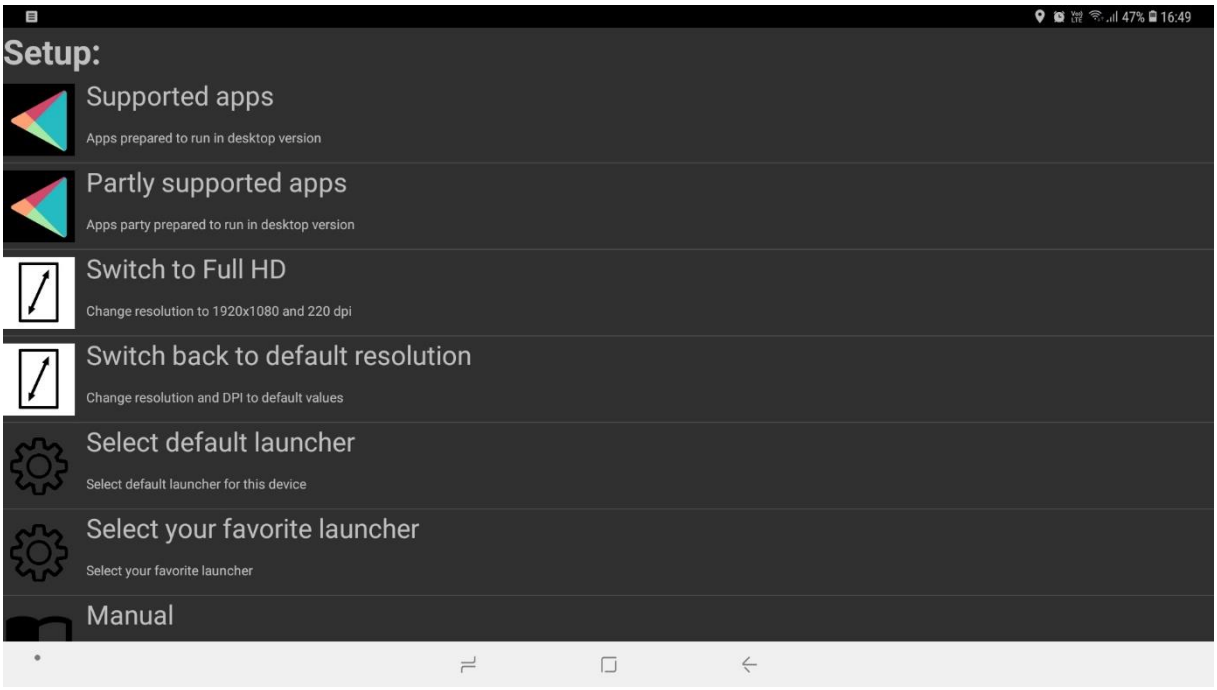


Obrázek 24 - Základní schéma funkcionality aplikace

(vlastní dílo autora)

4.1.5 Nastavení aktivity Setup

Aby si uživatel mohl program nastavit nebo přizpůsobit, byla do aplikace přidána možnost nastavení (anglicky Setup). Menu nastavení vypadá následovně (Obrázek 25)



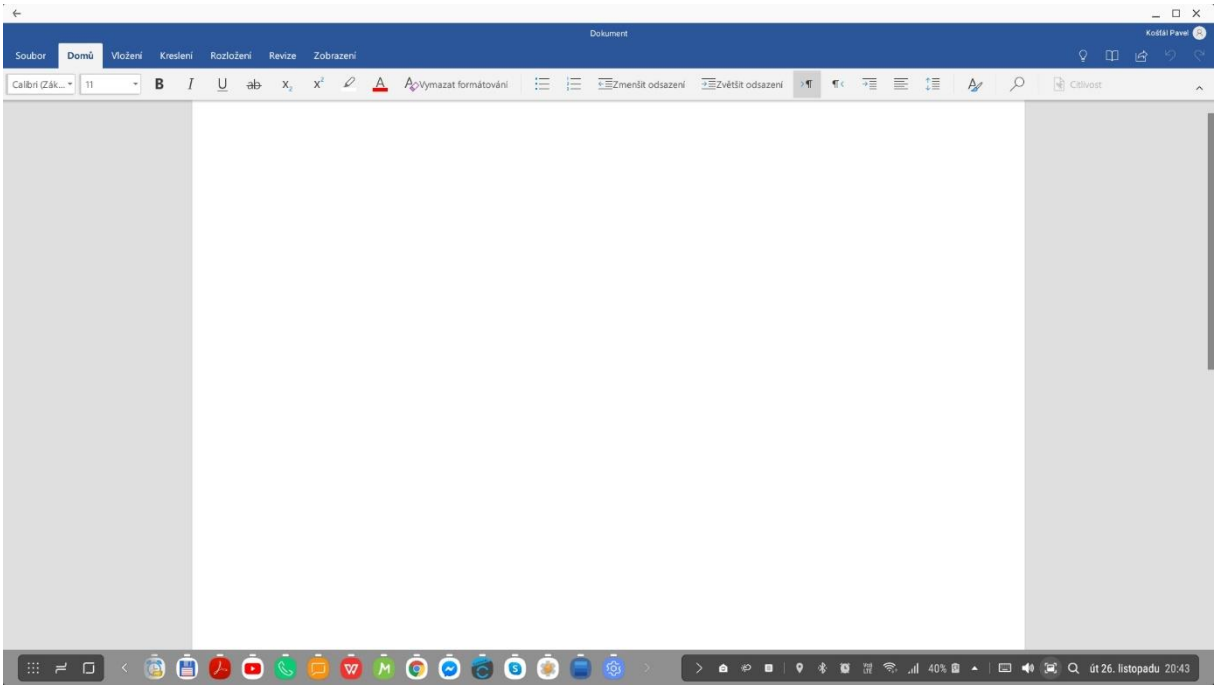
Obrázek 25 - Vzhled menu nastavení

(vlastní dílo autora)

Položky, které lze nastavit:

- Seznam podporovaných aplikací – zde se nachází aplikace plně kompatibilní s desktopovým rozhraním. Seznam nezahrnuje úplně všechny aplikace dostupné na trhu, jsou zde pouze nejpoužívanější programy
- Seznam částečně podporovaných aplikací – jedná se o aplikace, kterou jsou kompatibilní s desktopovým rozhraním, ale mají nějaké nedostatky. Jedná se

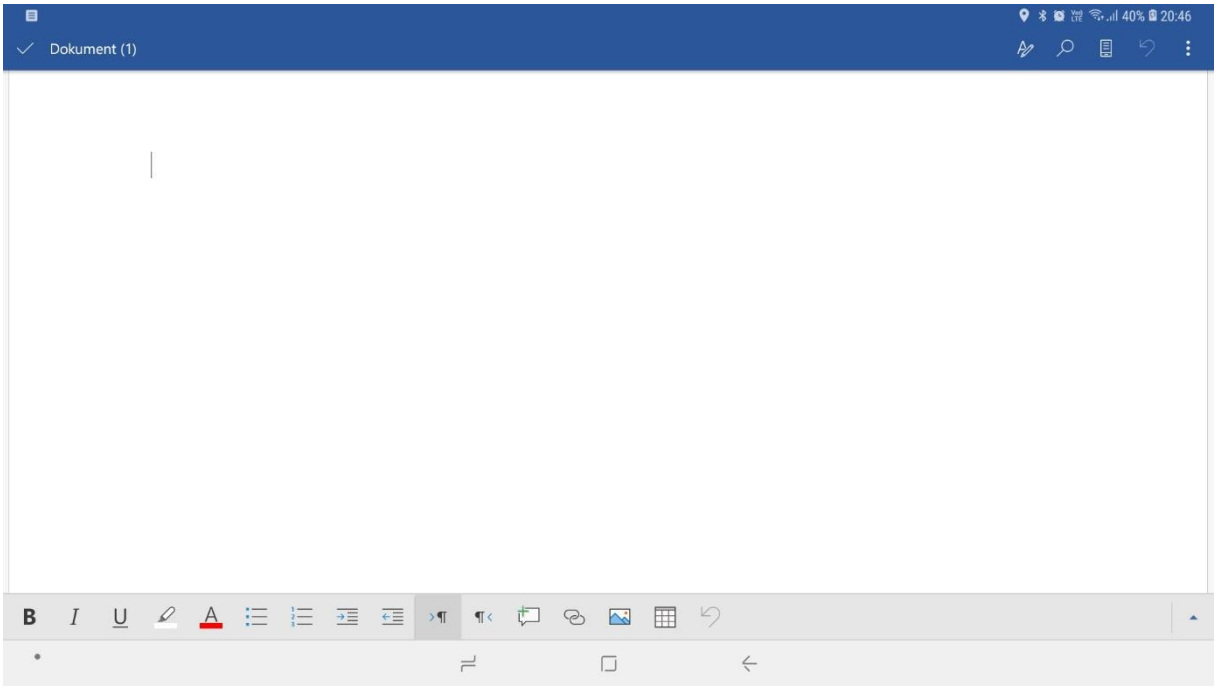
především o produkty Microsoft, které fungují velmi dobře v desktopovém prostředí Samsung Dex (další informace viz kapitola 3.3), ale mají určité nedostatky, způsobené špatnou optimalizací programu. V budoucnu je možno, že aplikace bude plně kompatibilní s desktopovým rozhraním, vytvořeným pomocí aplikace Revolution Launcher.



Obrázek 26 - Microsoft Word v desktopovém prostředí Samsung Dex

(vlastní dílo autora)

V této části je možno vidět rozdíl mezi aplikací Microsoft Word spuštěnou v režimu Samsung Dex (Obrázek 26) a aplikací Microsoft Word spuštěnou pomocí aplikace Revolution Launcher (Obrázek 27). Jak je vidět níže, aplikace se plně nepřepne do desktopového režimu. Toto se může změnit při vydání nové aktualizace.



Obrázek 27 - Microsoft Word v desktopovém prostředí vytvořeného pomocí aplikace Revolution Launcher

(vlastní dílo autora)

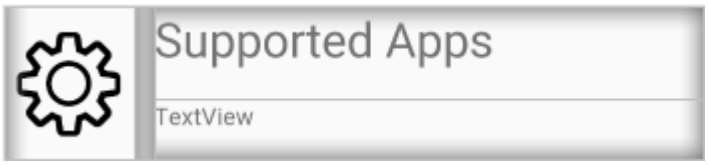
- Možnost ručně změnit rozlišení na Full HD, o této funkci je napsáno v kapitole 4.2.1
- Možnost ručně změnit rozlišení na výchozí
- Vybrání výchozího launcheru
- Vybrání oblíbeného launcheru – zde si uživatel může vybrat svůj oblíbený launcher, který bude fungovat při telefonním rozhraní
- Manuál – o funkcích v manuálu pojednává sekce 4.3.1

Vzhled a funkcionalita menu nastavení je vytvořeno pomocí funkce ListView (podrobnější informace viz kapitola 3.3). Pro vytvoření ListView je potřeba jednotný vzhled pro každý řádek. Tento vzhled se konfiguruje v xml souboru setup_setup_row.xml, XML soubor vypadá následovně:

```
<ImageView
    android:id="@+id/setupImageView"
    android:layout_width="164dp"
    android:layout_height="90dp"
```

```
        android:layout_gravity="center_vertical"
        android:layout_weight="1"
        app:srcCompat="@drawable/setup" />

    <LinearLayout
        android:layout_width="100dp"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="horizontal"></LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="vertical">

        <TextView
            android:id="@+id/setupPart1TextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Supported Apps"
            android:textSize="28sp" />

        <TextView
            android:id="@+id/setupPart2TextView"
            android:layout_width="760dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="TextView" />
    </LinearLayout>
</LinearLayout>
```

Vzhled předlohy v XML souboru vypadá následovně (viz Obrázek 28)



Obrázek 28 - Grafický vzhled předlohy v XML souboru pro ListView Setup

(vlastní dílo autora)

V dalším kroku se nastaví vzhled xml souboru pro zobrazení aktivity nastavení (setup). Kód v xml souboru setup_setup.xml je tento:

```
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".Setup">

    <TextView
```

```
        android:id="@+id/headerTextView2"
        android:layout_width="427dp"
        android:layout_height="52dp"
        android:text="Setup:"
        android:textSize="36sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <ListView
        android:id="@+id/listViewSetup"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/headerTextView2"
        app:layout_constraintVertical_bias="0.0" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Zobrazení funkce ListView a zjištění, na kterou položku se kliknulo, řeší tento kód, který se spouští hned po onCreate:

```
final Context context = this;

SetupAdapter supportedAppsAdapter = new SetupAdapter(this,
setupNameArray, descriptionArray, iconArray);

listView = findViewById(R.id.listViewSetup);
listView.setAdapter(supportedAppsAdapter);

listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
@Override
public void onItemClick(AdapterView<?> adapterView, View view,
int position, long id) {
Intent intent = new Intent(Intent.ACTION_VIEW);

setupItemCliked(position, context);

            }
        });
```

Metodě setupItemCliked je předáno číslo řádku, které bylo zmáčknuto. Tato metoda spouští různé nastavení pomocí Java funkce switch.

4.1.6 Řešení vyvolání dialogu pro změnu výchozího launcheru

Nejdříve je potřeba vytvořit pomocnou třídu jménem FakeHome.class. Tato třída neobsahuje žádné funkce.

V AndroidManifest.xml je nutno nakonfigurovat aktivitu FakeHome.class, tak, aby ji operační systém Android viděl jako defaultní launcher. Kód je následující:

```
<activity
android:name=".FakeHome"
android:enabled="false">
<intent-filter>

<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.HOME" />
<category android:name="android.intent.category.DEFAULT" />

</intent-filter>
</activity>
```

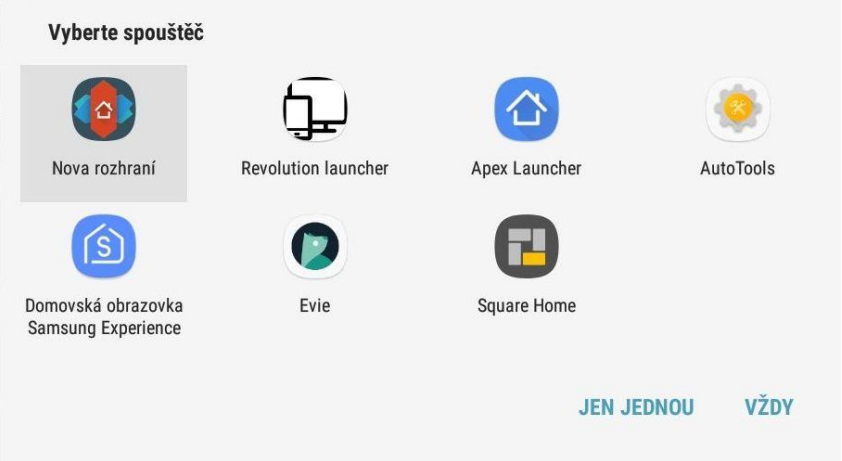
Zde je nejdůležitější kód:

```
android:enabled="false"
```

Díky tomuto kódu je aktivita vypnutá a operační systém poté pomocí níže uvedené metody vyvolá dialog pro změnu launcheru (do češtiny přeloženo jako spouštěč).

```
public static void changeDefaultHomeLauncher(Context c) {
PackageManager p = c.getPackageManager();
ComponentName cN = new ComponentName(c, FakeHome.class);
p.setComponentEnabledSetting(cN,
PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
PackageManager.DONT_KILL_APP);

Intent selector = new Intent(Intent.ACTION_MAIN);
selector.addCategory(Intent.CATEGORY_HOME);
c.startActivity(selector);

p.setComponentEnabledSetting(cN,
PackageManager.COMPONENT_ENABLED_STATE_DISABLED,
PackageManager.DONT_KILL_APP);
}
```

Dialog je poté možno vidět na obrázku níže (viz Obrázek 29)



Obrázek 29 - Dialogy zobrazují možnost změny výchozího launcheru

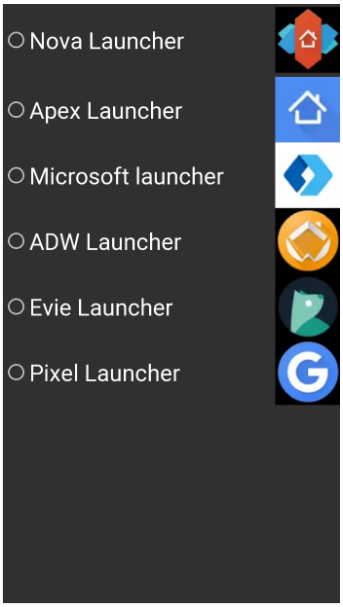
(vlastní dílo autora)

4.1.7 Změna oblíbeného launcheru

Oblíbený launcher je launcher, který se používá v telefonním rozhraní. Seznam byl vybrán pomocí stránky techworm.net (Tyagi, 2019). Níže je uveden seznam launcherů, použitých v seznamu aplikace Revolution launcher:

- Nova Launcher – oblíbený launcher s mnoha možnostmi nastavení, vynikající v rychlosti odezvy
- Apex Launcher – launcher používaný pro tablety, ale i pro mobilní telefony
- Microsoft Launcher – launcher od firmy Microsoft, oblíbený mezi uživateli díky specifickému vzhledu a speciálním funkcím
- ADW Launcher – stabilní, rychlá a jednoduchá aplikace
- Evie Launcher – velmi jednoduchý launcher zaměřený na rychlé odezvy
- Pixel Launcher – launcher nainstalovaný u telefonů od firmy Google

Vzhled je vytvořen pomocí funkce radioGroupSelect a radioGroup (více informací viz kapitola 3.3.10). Grafický vzhled vytvořený pomocí xml je možno vidět níže (viz Obrázek 30).



Obrázek 30 - Grafický vzhled volby oblíbeného launcheru

(vlastní dílo autora)

Pro správnou funkčnost je potřeba zjistit, zda je launcher v telefonu nainstalovaný. K tomu slouží tato metoda (pole nameArray je jmenný seznam launcherů):

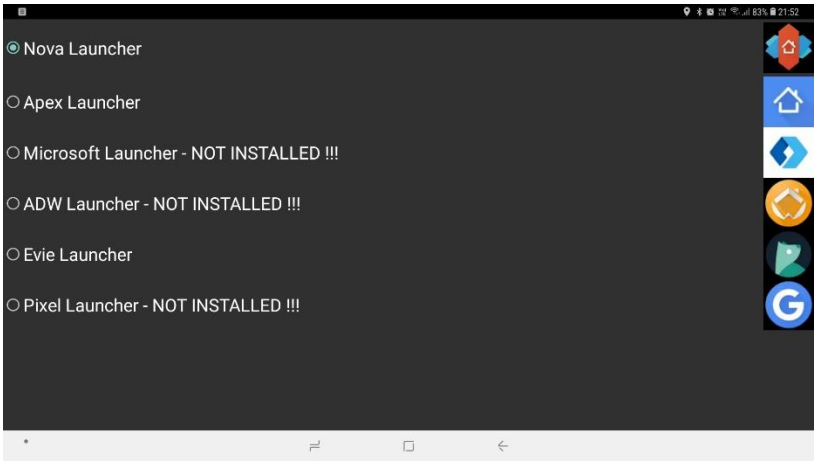
```
Boolean[] isInstalled = new Boolean[nameArray.length];

public void setIsInstalledTest () {

for (int i = 0; i< isInstalled.length; i++) {
PackageManager packageManager = getPackageManager();
try {

packageManager.getPackageInfo(packageNameArray[i],
PackageManager.GET_ACTIVITIES);
isInstalled[i] = true;
} catch (Exception e) {
isInstalled[i] = false;
            }
        }
    }
```

Pokud uživatel klikne na launcher, který není nainstalován, bude přesměrován do aplikace Google Play, kde si bude moci tento launcher nainstalovat. V případě, že je již launcher nainstalován a uživatel si ho vybere, tak se jméno oblíbeného launcheru uloží do proměnné favoriteLauncherName vytvořené pomocí funkce Paper.book a jméno balíčku se uloží do favoriteLauncherPackageName. Kód je vidět níže.

```
Paper.book().write("favoriteLauncherPackageName",
packageNameArray[position] );
```

```
Paper.book().write("favoriteLauncherName", nameArray[position]
);
```

Finální vzhled pro výběr oblíbeného je vidět níže (viz Obrázek 31).



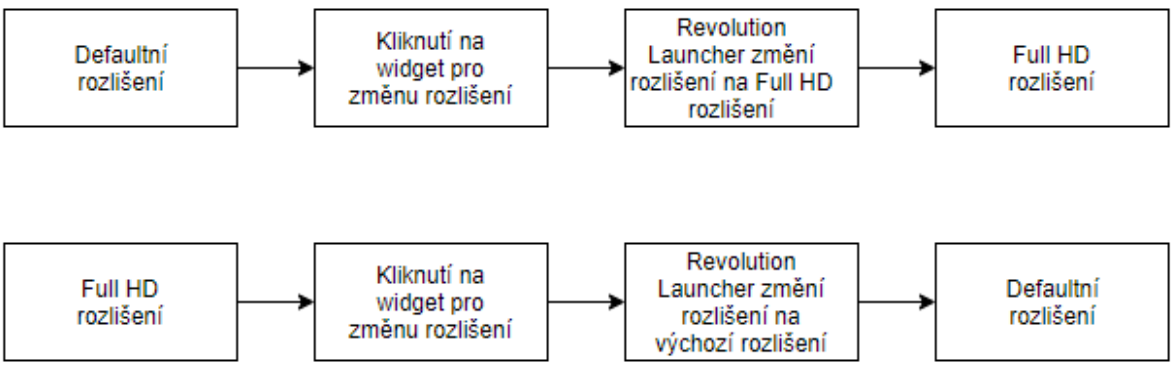
Obrázek 31 - Vzhled menu pro výběr oblíbeného launcheru

(vlastní dílo autora)

4.2 Vývoj aplikačního řešení změny rozlišení displeje

4.2.1 Nastavení programu pro využití pouze oblíbeného launcheru

Aby telefon měl plnohodnotné desktopové rozlišení, je důležité, aby se změnilo rozlišení displeje včetně počtu dpi (bodů na palec). Pro správnou funkčnost jsou zvoleny dva způsoby přepínání rozlišení. První způsob je využití aplikace Revolution launcher pouze pro přepínání rozlišení, nic dalšího není využito. Tato varianta je zvolena z toho důvodu, že někteří uživatelé si přejí používat pouze svůj oblíbený launcher. Princip je následující, uživatel si na svůj oblíbený launcher (který je zároveň výchozím launcherem) vloží widget z aplikace Revolution Launcher na plochu a poté při kliknutí na widget se přepíná rozlišení buď na Full HD nebo na standardní telefonní rozlišení. Přehled funkcionality je možno vidět na níže uvedeném obrázku (viz Obrázek 32).



Obrázek 32 - grafické zobrazení změny rozlišení pomocí widgetu

(vlastní dílo autora)

K vytvoření widgetu je nejdříve nutno nakonfigurovat vlastnosti widgetu. Xml soubor je uložen v adresáři v APP/res/xml. Konfigurace je následující:

```
android:initialKeyguardLayout="@layout/widget_change_resolutio
n"
android:initialLayout="@layout/widget_change_resolution"
android:minWidth="250dp"
android:minHeight="40dp"
android:minResizeWidth="250dp"
android:minResizeHeight="40dp"
android:previewImage="@drawable/widget4x1"
android:resizeMode="horizontal"
android:updatePeriodMillis="86400000"
android:widgetCategory="home_screen">
```

Přesný popis, co znamená které nastavení, je uvedeno v kapitole 3.3.9. Zde je nejdůležitější nastavení úvodního vzhledu, který se zobrazí v menu všech widgetů, to se nastavuje v initialLayout. Vzhled widgetu v menu widgetu je možno vidět na níže uvedeném obrázku (Obrázek 33). Po nastavení bude přesně takto vypadat widget na uživatelské ploše.

Obrázek 33 - Vzhled widgetu pro změnu rozlišení

(vlastní dílo autora)

Další důležité nastavení je minWidth a minHeight, které udává, že widget bude mít velikost 4 buňky na šířku a 1 buňku na výšku. Nastavení minResizeWidth a minResizeHeight udává minimální velikost pro změnu widgetu, zde nastaveno také na velikost 4x1 a to z důvodu, aby celý text byl bez problému vidět.

V následujícím kroku je potřeba nakonfigurovat v xml souboru vzhled widgetu. Konfigurace probíhá stejně jako při nastavování vzhledu aktivity. Uložení xml souboru je v APP/res/layout. Xml nastavení je následující:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_marginLeft="3dp"
android:layout_marginTop="3dp"
android:layout_marginRight="3dp"
android:layout_marginBottom="3dp">>
```

```
<TextView
android:id="@+id/changeResolutionTextView"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_centerInParent="true"
android:layout_marginStart="8dp"
android:layout_marginTop="8dp"
android:layout_marginEnd="8dp"
android:layout_marginBottom="8dp"
android:background="@android:color/darker_gray"
android:contentDescription="@string/appwidget_textReset"
android:gravity="center"
android:text="Initializing Widget"
android:textColor="#ffffff"
android:textSize="24sp"
android:textStyle="bold|italic" />

</RelativeLayout>
```

Grafický vzhled je vidět dále (Obrázek 34), důležité je si všimnout, že TextView zabírá celou plochu, na tuto část bude uživatel klikat. Bílá část jsou okraje. Vzhled widgetu je menší, protože mu je nastavena velikost 4 x 1 buňka.

Obrázek 34 - grafický vzhled widgetu pro změnu rozlišení

(vlastní dílo autora)

Za povšimnutí stojí, že v nastavení je uveden text=„Initializing Widget". Tato hodnota pro text je zvolena z důvodu, kdy telefon musí provádět mnoho operací najednou, například při startu telefonu. Jakmile dojde k nastavení widgetu, tak se text změní na „Click for: FullHD resolution“. V případě, že by zde žádný text nebyl, zobrazoval by se na ploše prázdný widget.

V dalším kroku je potřeba vytvořit třídu pro ovládání widgetu, třída se jmenuje WidgetChangeResolution.class. Nejdříve se nakonfigurují proměnné:

```
private static final String ACTION_WIDGET =
"ACTION_BROADCASTWIDGET";
Process process;
static String PaperdDBName = "resolutionFromPaperDB";
```

Zde nejdůležitější je nastavení proměnné, která reprezentuje nějakou akci. Některé widgety mohou mít například dvě či více různých možností, na které lze kliknout. Proměnná se jmenuje ACTION_WIDGET (jméno může být jakékoliv).
V dalším kroku je potřeba nastavit, co se stane při kliknutí na widget. K tomu je určen tento kód, který se provede při aktualizaci (onUpdate):

```
for (int appWidgetId : appWidgetIds) {
updateAppWidget(context, appWidgetManager, appWidgetId);
        }
```

Tento kód slouží k tomu, aby se aktualizoval widget nebo více widgetů (uživatel může mít na své ploše neomezené množství stejných widgetů). Zde se zavolá metoda updateAppWidget:

```
Paper.init(context);

String resolutionFromPaperDB =
Paper.book().read(PaperdDBName);

//if value from PaperDB is empty:
if(resolutionFromPaperDB == null){
resolutionFromPaperDB="FullHD";
Paper.book().write(PaperdDBName, "FullHD");
        }
```

V další části kódu se volá standardní příkaz pro komunikaci aplikace s widgetem, kde nejdůležitější kód je tento:

```
Intent intent = new Intent(context,
WidgetChangeResolution.class);
intent.setAction(ACTION_WIDGET);
```

Zde se zvolí, pro jakou akci se vybraný úmysl (anglicky intent) má provést.

Nakonec je potřeba změnit text, který je uveden na widgetu:

```
views.setOnClickPendingIntent(R.id.changeResolutionTextView,
pendingIntent);
views.setTextViewText(R.id.changeResolutionTextView, "Click
for: " + resolutionFromPaperDB + " resolution");
appWidgetManager.updateAppWidget(appWidgetId, views);
```

```
    }
```

Dále je potřeba vytvořit PendingIntent (česky by se dalo přeložit jako úmysl, který má přijít na řadu po zmáčknutí widgetu). V kódu se především nastaví, k jakému úmyslu (intent) se tento PendingIntent vztahuje:

```
PendingIntent pendingIntent =
PendingIntent.getBroadcast(context, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
```

Nakonec se přidá kód, který sleduje, zda uživatel klikl na předem definovanou část widgetu.

```
RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.widget_change_resolution);
views.setOnClickPendingIntent(R.id.changeResolutionTextView,
pendingIntent);
```

V tomto bloku kódu setOnClickPendingIntent se odvolává na changeResolutionTextView, které je možno vidět na výše uvedeném obrázku (viz Obrázek 34). Po kliknutí se zavolá metoda onReceive.

Dále se nastaví změna přepínání textu widgetu, který se dělá v části onReceive:

```
if (ACTION_SIMPLEAPPWIDGET.equals(intent.getAction())) {

Paper.init(context);
String resolutionFromPaperDB =
Paper.book().read(PaperdDBName);

if (resolutionFromPaperDB.equals("FullHD")){
Paper.book().write(PaperdDBName, "Reset");
changeToFullHD(context);
} else {
Paper.book().write(PaperdDBName, "FullHD");
changeToDefaultResolution(context);
}

resolutionFromPaperDB = Paper.book().read(PaperdDBName);

RemoteViews views = new RemoteViews(context.getPackageName(),
R.layout.widget_change_resolution);

views.setTextViewText(R.id.changeResolutionTextView, "Click
for: " + resolutionFromPaperDB + " resolution");
ComponentName appWidget = new ComponentName(context,
WidgetChangeResolution.class);
AppWidgetManager appWidgetManager =
AppWidgetManager.getInstance(context);
appWidgetManager.updateAppWidget(appWidget, views);
```

```
}
```

V tomto kódu je nejdůležitější nastavit, při jaké akci chceme provést změnu textu. Je potřeba, aby se kód provedl pouze poté, co bylo kliknuto na widget. Pokud by to nebylo nastaveno, změna by se prováděla při všech aktualizacích widgetu, například po restartování telefonu, což je nežádoucí, k tomu slouží tento kód:

```
if (ACTION_WIDGET.equals(intent.getAction())) {
```

Nakonec je potřeba změnit rozlišení, k tomu slouží dvě metody, první pro změnu na full HD rozlišení nazvanou changeToFullHD:

```
try {

process = Runtime.getRuntime().exec("su -c \"\"wm density
240");
process.waitFor();
process = Runtime.getRuntime().exec("su -c \"\"wm size
1080x1920");
process.waitFor();

Settings.System.putInt(context.getContentResolver(),
Settings.System.ACCELEROMETER_ROTATION, 0);

Settings.System.putInt(context.getContentResolver(),
Settings.System.USER_ROTATION, 1);

} catch (Exception e) {
e.printStackTrace();
}
```

A metoda pro změnu do výchozího rozlišení nazvanou changeToDefaultResolution:

```
try {
process = Runtime.getRuntime().exec("su -c \"\"wm density
reset");
process.waitFor();
process = Runtime.getRuntime().exec("su -c \"\"wm size
reset");
process.waitFor();

Settings.System.putInt(context.getContentResolver(),
Settings.System.ACCELEROMETER_ROTATION, 1);

Settings.System.putInt(context.getContentResolver(),
Settings.System.USER_ROTATION, 0);

} catch (Exception e) {
e.printStackTrace();
```

```
}
```

Dále následuje podrobnější popis kódu. Nejdříve je potřeba změnit hustotu bodů:

```
process = Runtime.getRuntime().exec("su -c \"\"wm density
240");
process = Runtime.getRuntime().exec("su -c \"\"wm density
reset");
```

Zde se provede pomocí funkce Runtime (podrobnější informace v kapitole 3.3) příkaz přímo na úrovni virtuálního stroje. První část příkazu uvedený v závorce „su“ vyžaduje, aby telefon měl administrátorská práva (anglicky super user). V druhé části se zavolá window manager pomocí příkazu „wm density 240“ – nastaví hustotu bodu na 240 dpi a příkaz „wm density reset“ změní hustotu na výchozí hodnotu.

Stejný princip funguje u změny rozlišení, kde se používá tento kód:

```
process = Runtime.getRuntime().exec("su -c \"\"wm size
1080x1920");
process = Runtime.getRuntime().exec("su -c \"\"wm size
reset");
```

Zde window manager změní počet bodů na 1080x1920. Pokud je v příkazu napsáno „wm size reset“, tak se změní rozlišení na výchozí hodnotu.

V následujícím kroku je potřeba vypnout, nebo zapnout automatickou rotaci obrazovky a nastavit rotaci obrazovky na šířku, nebo na výšku. Aby tato funkce fungovala, je nutno, aby telefon měl povolený přístup do secure settings (podrobnější informace viz kapitola 3.3.8). Po stáhnutí Android debug bridge (viz kapitola 3.3.8) je potřeba napsat příkaz „adb shell pm grant com.example.revolutionlauncher android.permission.WRITE_SECURE_SETTINGS“. Díky přístupu aplikace Revolution Launcher do secure settings může tato aplikace měnit například výše zmíněnou rotaci obrazovky.

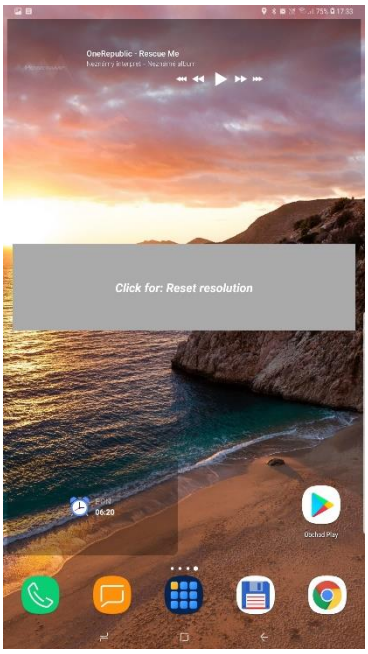
Pro přepnutí do full HD se vypne rotace a nastaví se zobrazení na šířku (anglicky landscape):

```
Settings.System.ACCELEROMETER_ROTATION, 0);
Settings.System.USER_ROTATION, 1);
```

Pro přepnutí do výchozího rozlišení je nutno zapnout rotaci a nastavit zobrazení na výšku (anglicky portrait):

```
Settings.System.ACCELEROMETER_ROTATION, 1);
Settings.System.USER_ROTATION, 0);
```

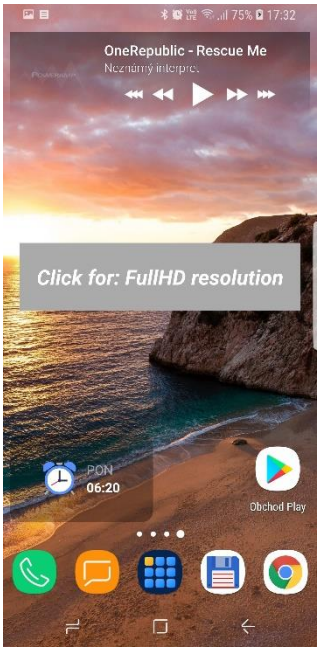
Nevýhodou použití pouze jednoho launcheru je špatné přizpůsobení oblíbeného launcheru na full HD rozlišení (viz Obrázek 35), kde je především text, stejně tak i ikony příliš malé.



Obrázek 35 - Vzhled launcheru Nova Launcher po změně na full HD rozlišení

(vlastní dílo autora)

Níže je možno vidět vzhled ikon a widgetů ve výchozím rozlišení (Obrázek 36). Při porovnání s obrázkem výše (Obrázek 35), kde je zobrazen launcher ve full HD rozlišení, je vidět, že toto rozlišení není ideální.



Obrázek 36 - Vzhled launcheru Nova Launcher ve standardním rozlišení

(vlastní dílo autora)

4.2.2 Nastavení aplikace Revolution Launcher pro desktopové rozlišení a oblíbeného launcheru pro běh v telefonním rozhraní

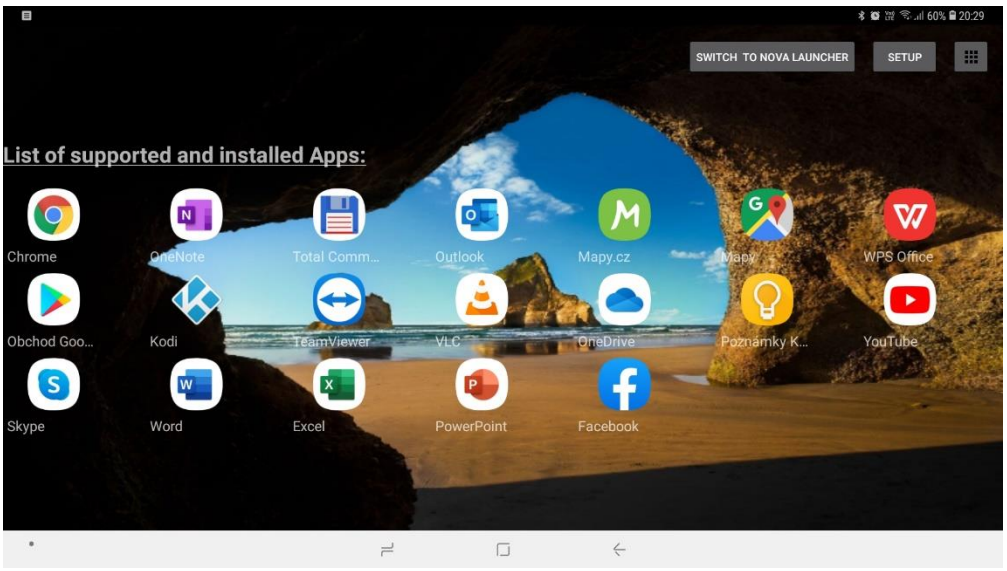
Pokud chce uživatel využít všechny desktopové funkce, které jsou dostupné pro jeho telefon, je pro něho nejlepší používat aplikaci Revolution Launcher ve full HD rozlišení a jeho oblíbený launcher v telefonním prostředí.

Po vložení widgetu na plochu (Obrázek 37) a poté kliknutí na tento widget se spustí aplikace Revolution Launcher, nastaví se hodnota v proměnné runFavoriteLauncher na false a spustí se aktivita MainActivityReal. Pro změnu rozlišení z výchozího rozlišení na full HD se použije stejný kód jako v kapitole 4.2.1. U proměnné runFavoriteLauncher, která je dostupná díky funkci PaperDB pro všechny aktivity, se nastaví hodnota na false.

Obrázek 37 - Grafický vzhled widgetu pro přepnutí z oblíbeného launcheru do Revolution Launcher

(vlastní dílo autora)

Pro přepnutí z aplikace Revolution Launcher do oblíbeného launcheru je potřeba kliknout na tlačítko v této aplikaci. Tlačítko je ve tvaru „Switch to …“ a název oblíbeného launcheru. V níže uvedeném obrázku (Obrázek 38) je název tohoto tlačítka „Switch to Nova Launcher“.

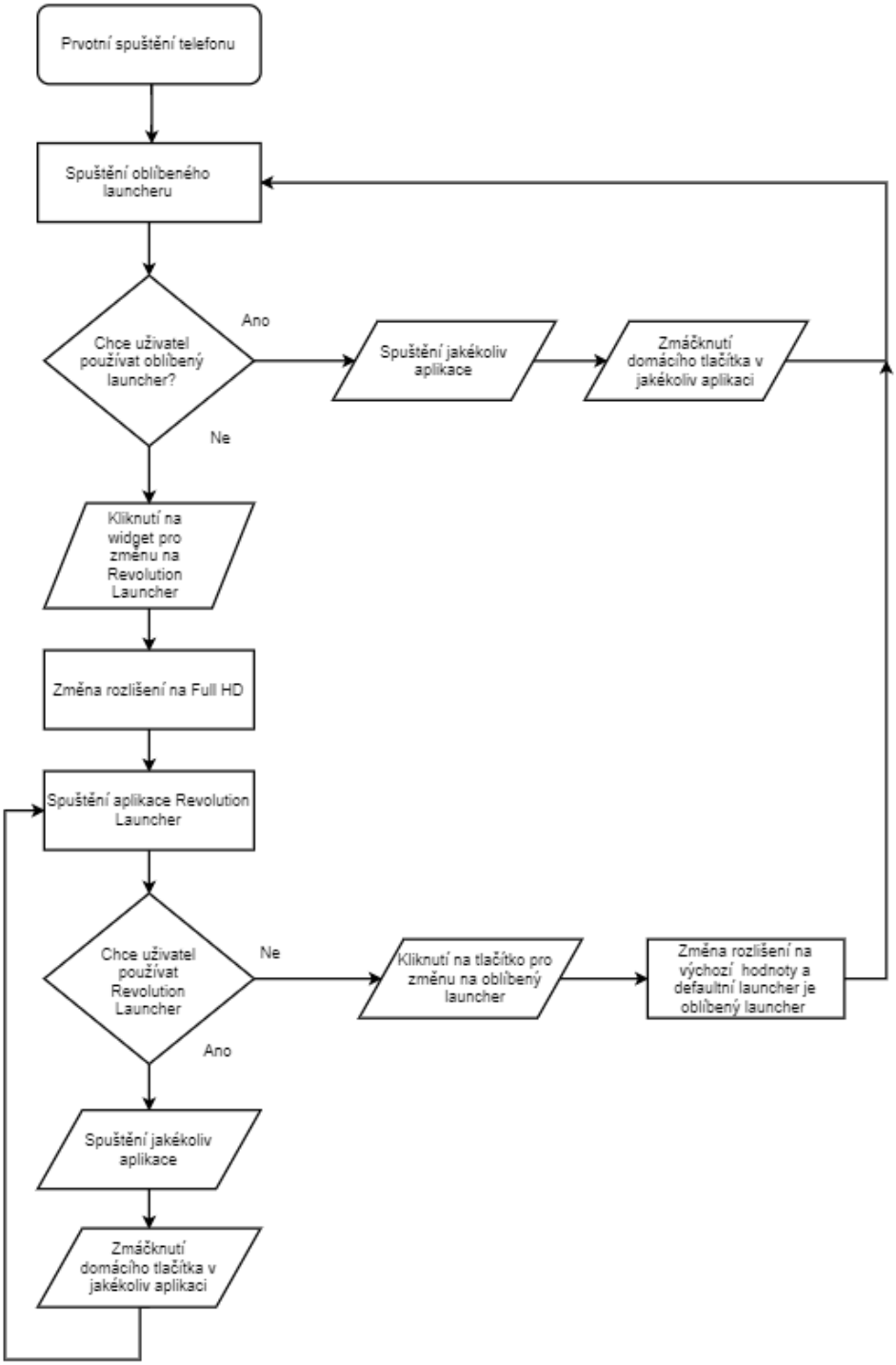


Obrázek 38 - Zobrazení tlačítka pro přepnutí na oblíbený launcher

(vlastní dílo autora)

Po kliknutí na toto tlačítko se proměnná runFavoriteLauncher nastaví hodnota na true, změní se rozlišení a počet bodů na výchozí hodnoty a přepne se telefon do oblíbeného launcheru.

Pro zobrazení toho, jak funguje celý systém při používání aplikace Revolution Launcher, slouží níže uvedený vývojový diagram (Obrázek 39). Zde se po zapnutí telefonu spustí oblíbený launcher a uživatel si může vybrat, zda si chce spustit nějakou aplikaci nebo se přepnout do aplikace Revolution launcher běžící ve full HD rozlišení pomocí kliknutí na widget. V aplikaci Revolution Launcher může uživatel spouštět různé aplikace nebo se rozhodnout vrátit se zpět do svého oblíbeného launcheru včetně změny rozlišení na původní hodnoty.



Obrázek 39 - Vývojový diagram při použití aplikace Revolution Launcher

(vlastní dílo autora)

4.3 Vývoj aplikačního řešení dalších vlastností programu

4.3.1 Vytvoření manuálu

V posledním kroku je nutno vytvořit návod, který pomůže uživateli nastavit vše tak, aby aplikace Revolution Launcher fungovala bez problémů. K tomuto účelu byla v sekci nastavení

vytvořena položka manuál. Zde si uživatel vybere, zda chce zobrazit manuál pro verzi, kdy se používá pouze oblíbený launcher a Revolution Launcher je používán jen pro změnu rozlišení. Druhá alternativa je, že uživatel bude pro telefonní rozhraní používat svůj oblíbený launcher a pro počítačové rozhraní program Revolution Launcher.

4.3.2 Popis manuálu verze 1 – použití pouze oblíbeného launcheru

Jak již bylo uvedeno výše, tato verze manuálu pomáhá nastavit vše tak, aby se používal pouze launcher, který je oblíbený uživatelem. V prvním kroku je potřeba, aby uživatel aplikaci Revolution Launcher dal přístup Secure Settings:

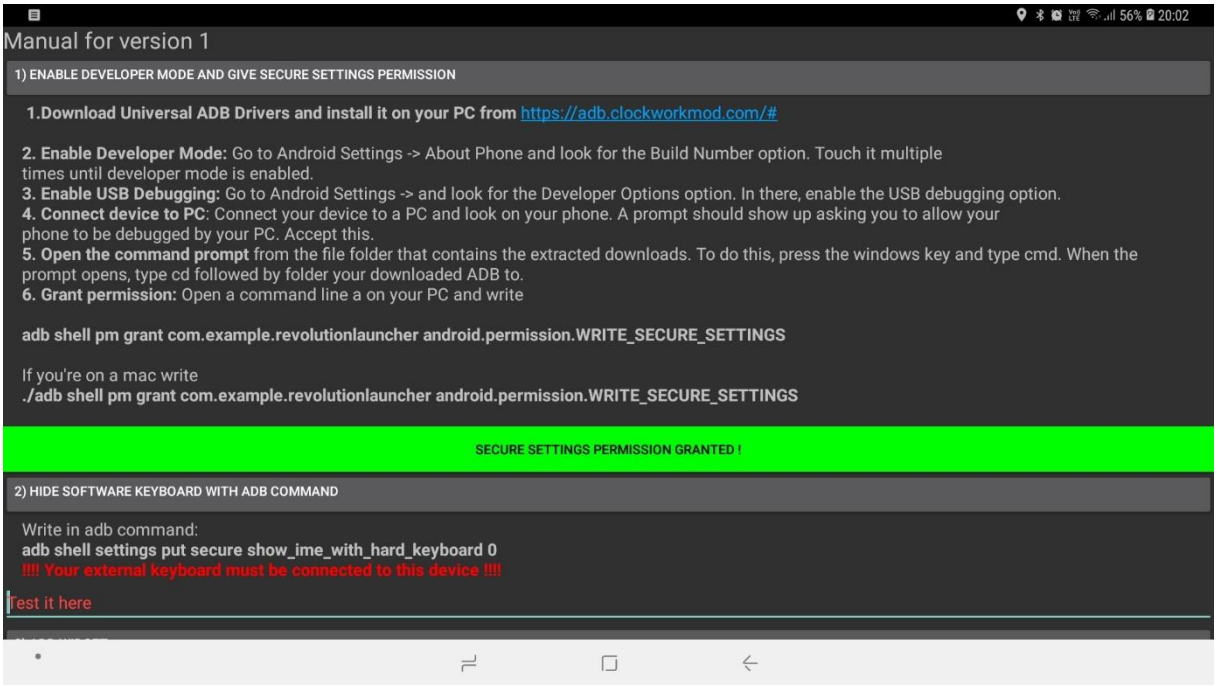
1. Je potřeba stáhnout a nainstalovat ADB program (Android Debug Bridge), ten je možno stáhnout například z https://adb.clockwordmod.com
2. Zapnout vývojářské menu ve svém telefonu – jít do menu telefonu a najít verzi operačního systému. Několikrát se tohoto menu dotknout (na většině telefonů 7krát) a poté se zobrazí vývojářské menu.
3. Přejít do vývojářského menu a zde zaškrtnout „ladění USB“ (anglicky „USB debugging“).
4. Připojit mobilní telefon k počítači. Poté, co se zobrazí dotaz na povolení na ladění přes USB, je nutno toto akceptovat.
5. V dalším kroku je potřeba se dostat do adresáře, kam se nainstaloval ADB a otevřít v tomto místě příkazový řádek.
6. Nakonec se aplikaci Revolution Launcher přidají práva pro Secure Settings, K tomu je nutno do příkazového řádku napsat „adb shell pm grant com.example.revolutionlauncher android.permission.WRITE_SECURE_SETTINGS“

Pro ověření, zda uživatel přidělil oprávnění aplikaci Secure Settings, je v manuálu vytvořeno tlačítko, které po zmáčknutí ověří toto oprávnění, metoda je následující:

```
public boolean checkSecureSettingPermission () {
Button buttonCheckSecureSettingPermission =
findViewById(R.id.buttonCheckSecureSettingPermission);

if (ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_SECURE_SETTINGS) ==
PackageManager.PERMISSION_GRANTED) {

buttonCheckSecureSettingPermission.setBackgroundColor(Color.GR
EEN);

buttonCheckSecureSettingPermission.setTextColor(Color.BLACK);
buttonCheckSecureSettingPermission.setText("Secure Settings
permission GRANTED !");
```

```
return true;
} else {

buttonCheckSecureSettingPermission.setBackgroundColor(Color.RE
D);

buttonCheckSecureSettingPermission.setTextColor(Color.WHITE);
}
return false;
}
```

Pokud je přístup do Secure Settings umožněn, tak se pozadí tlačítka pro ověření přístupu změní na zelenou barvu (Obrázek 40). Test je možno udělat okamžitě po provedení ADB příkazu.



Obrázek 40 - Vzhled manuálu verze 1

(vlastní dílo autora)

V dalším kroku je nutno schovat softwarovou klávesnici, pokud je připojena externí klávesnice. To se udělá pomocí druhého ADB příkazu: „adb shell settings put secure show_ime_with_hard_keyboard 0“. Pokud by na konci tohoto příkazu byla hodnota jednička, znamenalo by to, že při připojení externí klávesnice by se vždy zobrazovala i softwarová klávesnice, což je nežádoucí. Otestování se pak provede kliknutím na textové pole. Pokud se neobjeví softwarová klávesnice, byl ADB příkaz zadán správně.

V posledním kroku je potřeba, aby Revolution Launcher nebyl výchozí launcher, ale aby jím byl uživatelův oblíbený launcher. Poté se už jen uživatel přepne do svého oblíbeného launcheru pomocí tohoto kódu:

```
Intent startMain = new Intent(Intent.ACTION_MAIN);
startMain.addCategory(Intent.CATEGORY_HOME);
startMain.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
startActivity(startMain);
```

Nakonec uživatel musí vložit na svoji plochu widget (Obrázek 41) o velikosti 4x1 buňky, který se nazývá „Click for: Full HD resolution“.

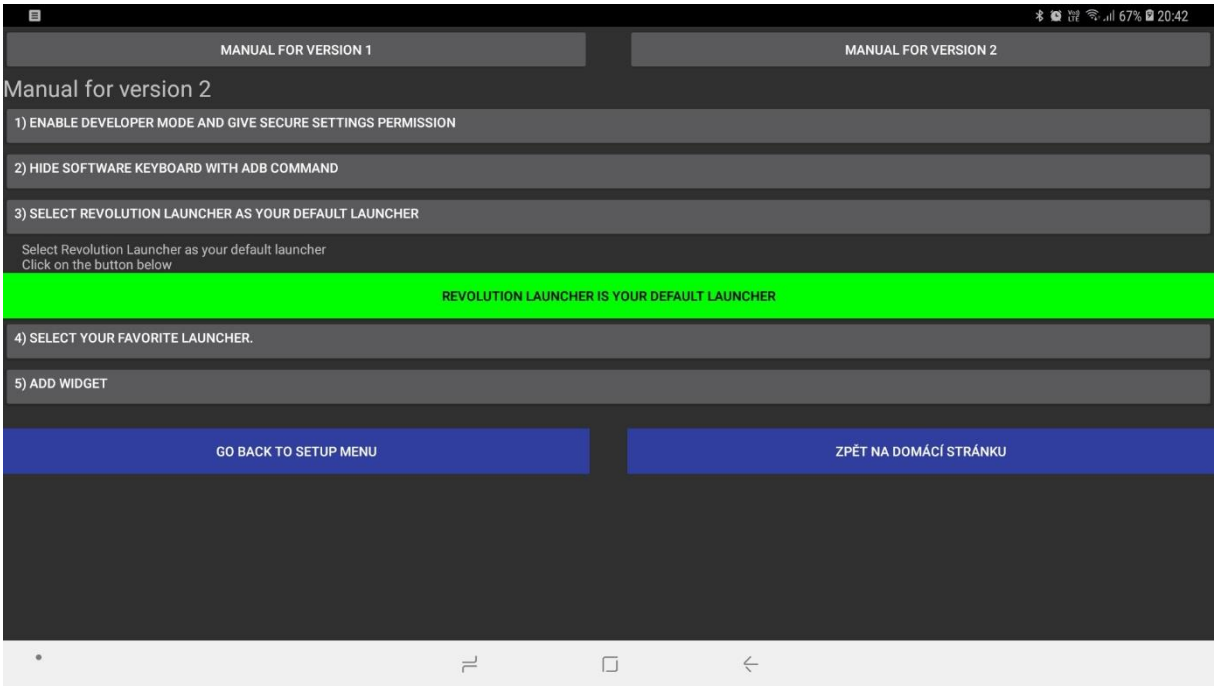
Obrázek 41 - Widget velikosti 4x1 pro přepínání rozlišení

(vlastní dílo autora)

4.3.3 Popis manuálu verze 2 – použití aplikace Revolution Launcheru a oblíbeného launcheru

Tento manuál má některé prvky stejné s verzí 1.

1. Nejdříve je potřeba udělit přístup do Secure Settings (již vysvětleno v kapitole 4.3.2).
2. Dále je nutno schovat softwarovou klávesnici, pokud je připojena externí klávesnice (viz kapitola 4.3.2).
3. Revolution Launcher musí být jako výchozí launcher, ověření probíhá stejně jako v kapitole 4.1.1, vzhled je možno vidět níže (Obrázek 42).



Obrázek 42 - Vzhled manuálu 2 včetně tlačítka pro ověření, zda je Revolution Launcher výchozím launcherem

(vlastní dílo autora)

4. Uživatel si musí vybrat svůj oblíbený launcher (již řešeno v kapitole 4.1.7).
5. Nakonec se uživatel po kliknutí na tlačítko přepne do svého oblíbeného launcheru, kde musí dát na plochu widget o velikosti 2x1 buňka (Obrázek 43), poté po kliknutí na tento widget nebo na domovské tlačítko, se uživatel dostane zpět do aplikace Revolution Launcher.

Obrázek 43 - Widget velikosti 2x1 pro přepnutí do aplikace Revolution Launcher

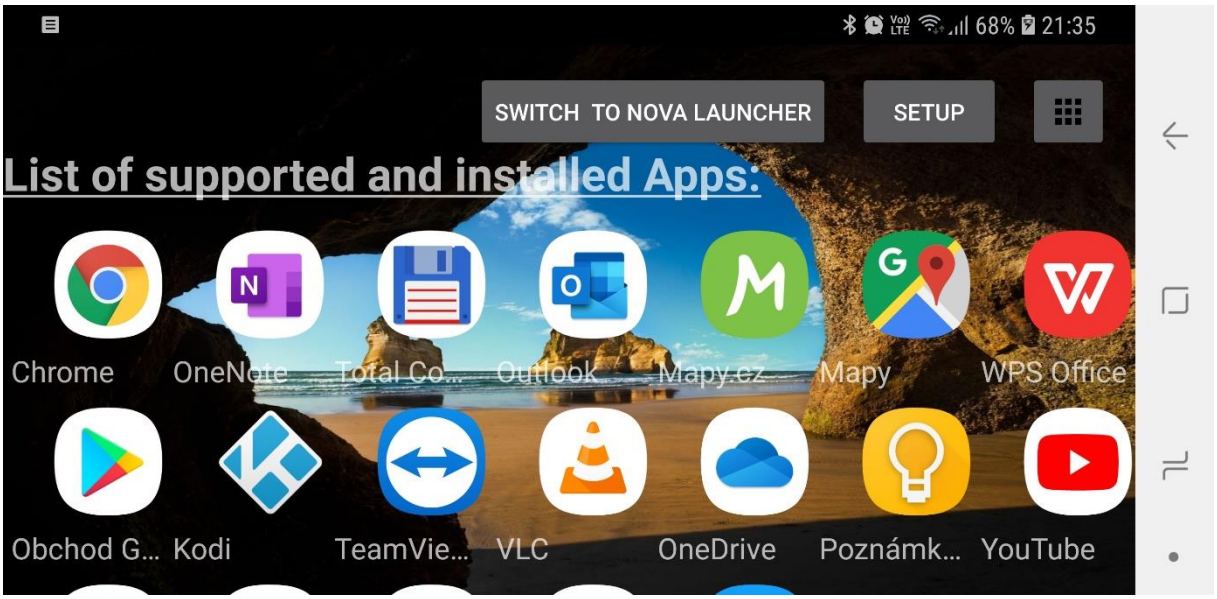
(vlastní dílo autora)

4.3.4 Bezdrátové připojení pomocí Google Chromecast

Většinou jen ty nejdražší a nejlepší telefony mají HDMI výstup, ostatní telefony nemají možnost přes kabel klonovat zobrazení na telefonu. Pro bezdrátové klonování obrazu bude použito zařízení Google Chromecast a aplikace Google Home. Nastavení je velmi jednoduché, nejdříve je potřeba telefon spárovat s Google Chromecast a poté už jen zvolit možnost klonovat obraz. Rychlost odezvy a kvalita obrazu záleží na rychlosti domácí sítě, pokud bude zařízení Chromecast daleko od domácího Wi-Fi, tak bude přenosová rychlost nižší a na obrazovce bude znát menší prodleva.

4.3.5 Popis vzhledu aplikace Revolution v normálním telefonním rozhraní

Ještě předtím, než uživatel vše nastaví, tak aplikace Revolution Launcher běží v telefonním rozhraní, proto je celá aplikace naprogramována tak, aby fungovala bez problému jak v desktopovém rozlišení, tak v rozlišení určeném pro telefonní zobrazení. Vzhled v telefonním rozlišení není ideální, ale i přesto je možno bez potíží aplikaci používat a vše nastavit pro plnohodnotné využití (viz Obrázek 44). Ikony, které se nevejdou na plochu, je možno pomocí klasického gesta na skrolování posunout nahoru.

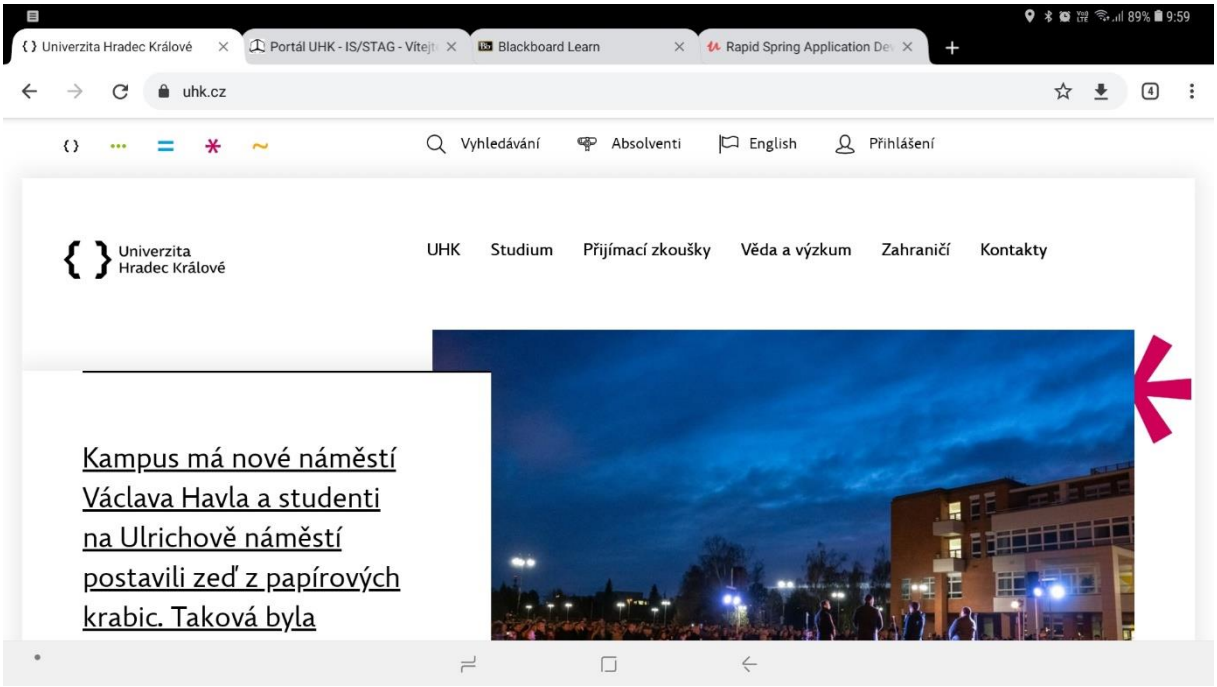


Obrázek 44 - Vzhled úvodní stránky aplikace Revolution Launcher v telefonním rozhraní

(vlastní dílo autora)

4.3.6 Zobrazení nejzajímavějších aplikací v desktopovém režimu

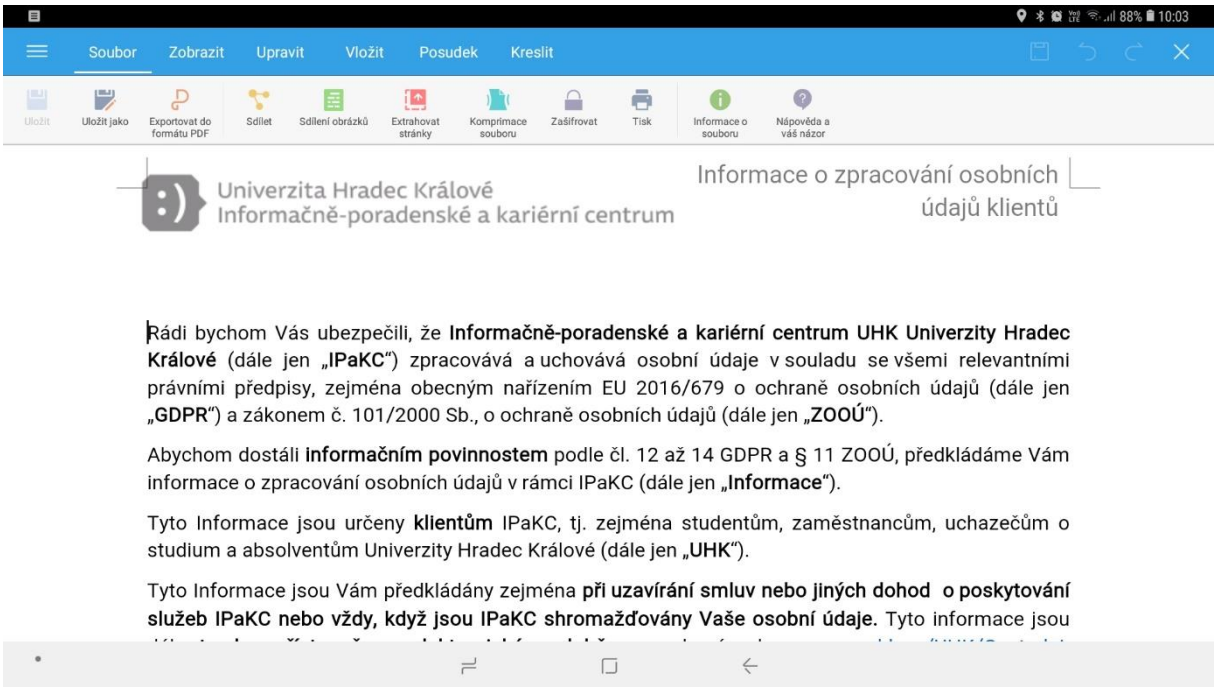
Mezi nejoblíbenější aplikace na osobním počítači patří pro většinu uživatelů prohlížeč webových stránek a kancelářské aplikace Office. Jak je možno vidět na níže uvedeném obrázku (Obrázek 45), tak tato aplikace má velmi podobný vzhled jako na standardním počítači s operačním systémem Windows. Nahoře jsou stejné záložky, které lze přesouvat doleva nebo doprava. Všechny weby se zobrazují v desktopovém režimu.



Obrázek 45 - Vzhled aplikace Google Chrome v desktopovém prostředí

(vlastní dílo autora)

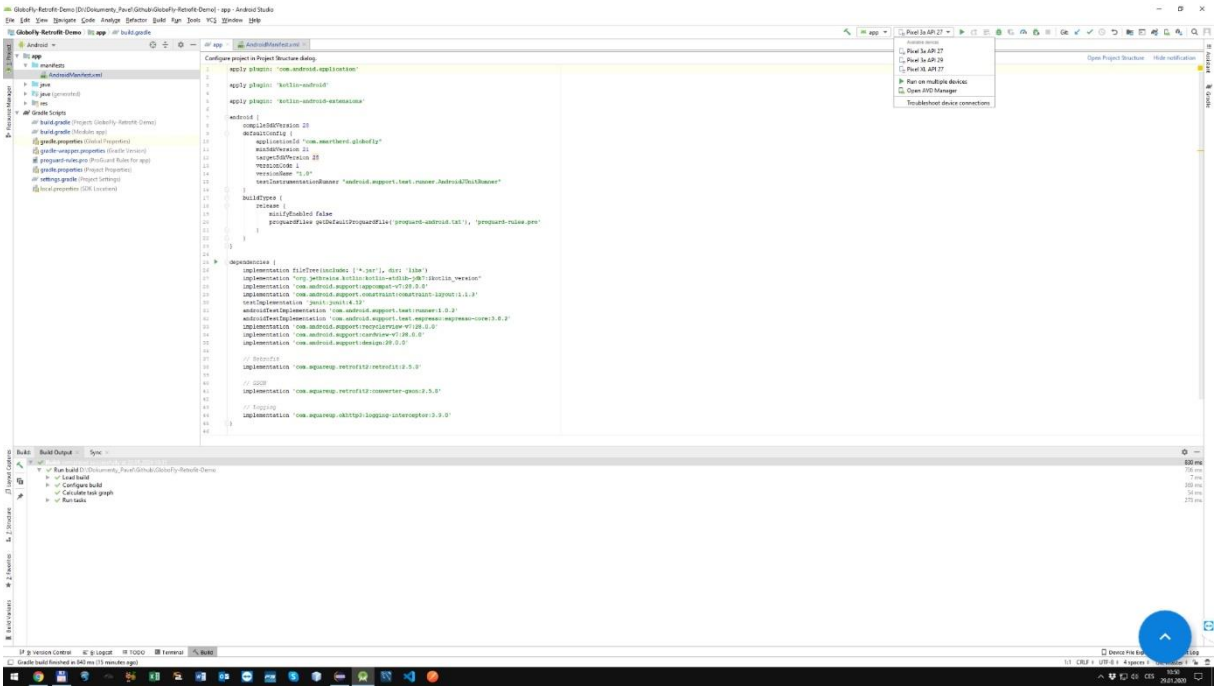
Na dalším obrázku (Obrázek 46) je vidět vzhled aplikace WPS Office, která plně podporuje desktopové prostředí. Také umí zobrazovat pás funkcí, velmi podobný jako má Microsoft Office.



Obrázek 46 - Vzhled aplikace WPS Office v desktopovém prostředí

(vlastní dílo autora)

Pro velmi náročné uživatele existuje možnost použít kombinaci stolního počítače, který bude na nějakém místě běžet nonstop (nebo po dobu nezbytně nutnou) a aplikace TeamViewer, která bude spuštěna na počítači, tak i na mobilním telefonu. Díky tomuto programu lze díky internetovému připojení dělat úplně to samé, co na běžném počítači, například programovat aplikace v Android Studiu (viz Obrázek 47).



Obrázek 47 - Využití aplikace TeamViewer pro možnost programování přímo v telefonu v aplikaci Android Studio (vlastní dílo autora)

5 Diskuze výsledků řešení

Aplikace Revolution Launcher funguje dle zadaného cíle diplomové práce, hlavním nedostatkem je nutnost mít telefon s root oprávněním. Podobnou aplikaci také vytvořila firma Sentio a aplikaci pojmenovala Sentio Desktop, která slouží především pro zobrazení obrazu z telefonu na speciálním notebooku jménem Superbook, který se stále ještě vyvíjí. Tato aplikace nefunguje jako klasický launcher, pouze je spuštěna přes ostatní aplikace, podobně jako například Facebook Messenger, což je pro většinu lidí velký problém, protože telefon se po zmáčknutí domovského tlačítka nepřepne do aplikace Sentio Desktop, ale do výchozího launcheru a teprve z něho je nutno znovu spustit tento program. Nastavení programu jako domovský launcher je sice možné, ale dle hodnocení uživatelů z Google Play tato volba většinou nefunguje. Hlavní výhoda programu Sentio Desktop je grafická stránka, která je propracovanější než u aplikace Revolution Launcher, například umožňuje měnit polohu ikon nebo spustit aplikaci v okně.

Další alternativou k naprogramované aplikaci je desktopové prostředí Samsung Dex a od firmy Huawei jménem EMUI Desktop. Obě aplikace jsou si velmi podobné a jsou profesionálně naprogramované, funguje zde dobře provoz aplikací v okně, jediná nevýhoda je, že tyto aplikace jsou dostupné pouze u nejlepších telefonů od firmy Samsung, respektive od firmy Huawei. Hlavní výhoda těchto dvou desktopových prostředí je plná podpora aplikací od firmy Microsoft, všechny tyto aplikace vypadají velmi podobně jako na klasickém počítači, což bohužel neplatí při spuštění v programu Revolution Launcher, aplikace z balíčku Microsoft Office se plně nepřepnou do desktopového prostředí. Důvod, proč tomu tak je, bude nutno vyzkoumat, bohužel firma Microsoft nesdílí svoje zdrojové kódy, takže není jisté, zda je problém ve zvoleném rozlišení a DPI, nebo jestli to je ovlivněno něčím jiným, co má vliv na vzhled aplikací, více viz kapitola 3.3.6, pojednávající o přizpůsobení vzhledu. Oproti těmto dvěma řešením má aplikace Revolution Launcher jednu velkou výhodu, a to, že není speciálně potřeba, aby vývojáři programů upravovali svoje aplikace tak, aby fungovaly ve speciálním desktopovém prostředí od firmy Samsung nebo Huawei. Bohužel některé aplikace, jako je populární Kodi na sledování filmů, nefungují v Samsung Dex, což hodně lidí odrazuje od pravidelného používání Samsung Dex.

Pro lepší porovnání zmíněných desktopových aplikací je zde uveden v tabulce (Tabulka 5) přehled hlavních výhod a nevýhod řešení, které umožňují používat telefon jako desktopový počítač.

Název řešení	Výhody	Nevýhody
Revolution Launcher	Funguje na jakémkoliv telefonu s operačním systémem Android	Nemá podporu více spuštěných aplikací, běžících ve vlastním okně
	Lze provozovat bezdrátově	Některé aplikace, jako Microsoft Word, zde nejsou zobrazeny v plné desktopové verzi
	Je možno spustit jakoukoliv aplikaci	Nefunguje bez root oprávnění
Samsung Dex a Huawei EMUI Desktop	Vzhled velmi podobný klasickému desktopovému prostředí	Nelze provozovat bezdrátově
	Rozlišení je automaticky nastaveno na rozlišení připojeného zařízení	Funguje pouze na nejlepších telefonech od firmy Samsung nebo Huawei
	Telefon lze použít jako náhradu myši	Některé populární aplikace zde nefungují přes celou obrazovku nebo nejdou vůbec spustit
Sentio Desktop	Vzhled podobný standardnímu desktopovému rozhraní	Nefunguje jako klasický launcher na většině telefonů
	Aplikace lze spouštět v oknu	Dolní lišta je překryta lištou z aplikace Sentio Desktop a nelze tedy použít například tlačítko zpět
	Je možno spustit jakoukoliv aplikaci	Některé aplikace, jako Microsoft Word, zde nejsou zobrazeny v plné desktopové verzi

Tabulka 5 – Přehled hlavních výhod a nevýhod řešení, které umožňují používat telefon jako desktopový počítač

(vlastní dílo autora)

Z výše uvedené tabulky vyplývá, že každé řešení má svoje výhody i nevýhody a že tedy v současné době neexistuje žádné univerzální řešení. Firma Google, která má na starosti vývoj operačního systému Android, by mohla v budoucnu všechny nevýhody zrušit, pokud by vymyslela univerzální řešení implementované přímo v jádru operačního systému.

6 Závěr

Aplikace Revolution Launcher funguje, jak bylo plánováno, tedy je možno pomocí této aplikace použít skoro jakýkoliv telefon jako alternativu k desktopovému počítači, pro nejnáročnější uživatele existuje možnost využití aplikaci TeamViewer, která umožní přístup k domácímu počítači, například z hotelu a spustit v telefonu jakoukoliv aplikaci, určenou pro běžný počítač. Mezi hlavní výhody naprogramované aplikace patří nízká náročnost na paměť, protože se jedná o nenáročný program fungující jako launcher. Také je možno využít aplikaci Revolution Launcher pouze jako přepínač mezi rozlišením a počtem bodů, tuto funkci mohou využívat lidé, kteří z nějakého důvodu chtějí používat jen svůj oblíbený launcher.

V budoucnu má autor aplikace Revolution Launcher v plánu vylepšit některé grafické prvky, podobné, jako má aplikace Sentio Desktop, vytvořit v této aplikaci ještě jeden launcher vhodný pro využití v klasickém telefonním prostředí a nakonec aplikaci publikovat v Google Play. Nakonec by bylo dobré vyzkoumat, jak lze přepínat rozlišení a počet bodů bez nutnosti mít telefon s root oprávněním. Pokud aplikace bude mít vysokou popularitu na Google Play, bylo by dobré vytvořit jednoduchou SQL aplikaci pro ukládání nastavení launcheru na server a také přidat funkci pro stahování informaci o počasí a nakonec přidat přeložení textu do jiných jazyků. Možné je i přeprogramování části aplikace do jazyku Kotlin z důvodu zjednodušení kódu nebo psaní dalšího kódu v tomto jazyku, jak již bylo uvedeno, program může být psán v obou jazycích.

7 Seznam použité literatury

7.1 Knižní publikace

Allen, G. (2013). *Android 4: Průvodce programováním mobilních aplikací*. Computer Press.

Herout, P. (2007). *Učebnice jazyka Java*. Kopp.

Lacko, Ľ. (2015). *Vývoj aplikací pro Android*. Computer Press.

Lacko, Ľ., & Herodek, M. (2017). *Mistrovství—Android*.

Pecinovský, R. (2012). *Java 7: Učebnice objektové architektury pro začátečníky*. Grada.

Roubalová, E. (2015). *Java bez předchozích znalostí*.

Schildt, H. (2012). *Java 7: Výukový kurz*. Computer Press.

Schildt, H. (2014). *Mistrovství—Java*. Computer Press.

Späth, P. (2018). *Pro Android with Kotlin: Developing Modern Mobile Apps*. Apress.

Spell, B., & Kiszka, B. (2002). *Java: Programujeme profesionálně*. Computer Press.

Ujbányai, M. (2012). *Programujeme pro Android*. Grada.

Vávrů, J., & Ujbányai, M. (2013). *Programujeme pro Android*. Grada.

Walls, C. (2011). *Spring in action* (3rd ed). Manning.

7.2 Internetové zdroje

10 Advantages and Disadvantages of Rooting Android devices. (2017, březen 30). *EDUCBA*. https://www.educba.com/rooting-android/

A Selva, K. (2019, duben 21). *Root Samsung Galaxy S8 SM-G950F Pie 9.0 using TWRP*. Android Infotech. https://www.androidinfotech.com/root-samsung-galaxy-s8-sm-g950f-pie/

Android Activity Lifecycle—Javatpoint. (2018). Www.Javatpoint.Com. https://www.javatpoint.com/android-life-cycle-of-activity

Android Architecture—ELinux.org. (2011, červen 13). https://elinux.org/Android_Architecture

Android for Cars overview. (2019, prosinec 27). Android Developers. https://developer.android.com/training/cars?hl=cs

Android Grid View—Tutorialspoint. (2020). https://www.tutorialspoint.com/android/android_grid_view.htm

Android List View—Tutorialspoint. (2019). https://www.tutorialspoint.com/android/android_list_view.htm

App Widget Design Guidelines. (2019, prosinec 27). Android Developers. https://developer.android.com/guide/practices/ui_guidelines/widget_design?hl=cs

App Widgets Overview. (2019, prosinec 27). https://developer.android.com/guide/topics/appwidgets/overview?hl=cs

Bolton, D. (2016, únor 3). *5 Reasons to Use C++ for Android Development*. Dice Insights. https://insights.dice.com/2016/02/03/5-reasons-to-use-c-for-android-development/

Davenport, C. (2019, únor 21). Play Store developers will need to target their apps for Android 9 Pie by this fall. *Android Police*. https://www.androidpolice.com/2019/02/21/play-store-developers-will-need-to-target-their-apps-for-android-9-pie-by-this-fall/

Dissanaike, S. (2019, září 20). *How to use Android Automotive on a 10+ years old Saab 9–5, Part 2/2*. Medium. https://medium.com/himinds/how-to-use-android-automotive-on-a-10-years-old-saab-9-5-part-2-2-95f8d9ff3f3b

Ducrohet, X. (2013, květen 15). Android Studio: An IDE built for Android. *Android Developers Blog*. https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html

Flutter Introduction. (2019). https://flutter.dev/docs/resources/faq

Google Java Style Guide. (2020, únor 7). https://google.github.io/styleguide/javaguide.html

Gough, C. (2019, prosinec 3). *Global top Android games by revenue 2019*. Statista. https://www.statista.com/statistics/690150/leading-mobile-games-google-play-world-revenue/

Hardy, B. (2014, duben 21). Customizing Android ListView Rows by Subclassing. *Digital Product Development Agency | Big Nerd Ranch*. https://www.bignerdranch.com/blog/customizing-android-listview-rows-by-subclassing/

Jansen, M. (2019, listopad 19). *How to Use Samsung's DeX Mode on Your Galaxy Phone*. Digital Trends. https://www.digitaltrends.com/mobile/how-to-use-samsung-dex-mode/

Java Version History. (2020, únor 7). Www.Javatpoint.Com. https://www.javatpoint.com/java-versions

Kilián, K. (2015, červen 24). *Historie Androidu v kostce aneb Od verze 1.0 až po Android M*. Svět Androida. https://www.svetandroida.cz/historie-androidu/

Lardinois, F. (2019, květen 7). Kotlin is now Google's preferred language for Android app development. *TechCrunch*. http://social.techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/

Layouts. (2020, leden 7). Android Developers. https://developer.android.com/guide/topics/ui/declaring-layout?hl=cs

Localize the UI with Translations Editor. (2019, prosinec 27). Android Developers. https://developer.android.com/studio/write/translations-editor?hl=cs

Microsoft Xamarian. (2019). https://docs.microsoft.com/cs-cz/xamarin/android/get-started/installation/windows

Michalovič, R. (2019, únor 6). *Změna licence Java11 JDK a OpenJDK*. https://www.itnetwork.cz/zmena-licence-java11-sdk-a-openjdk

Minh, N. H. (2019, září 26). *Java SE versions history*. https://www.codejava.net/java-se/java-se-versions-history

OkHttp Example REST Client. (2017, březen 21). https://www.stubbornjava.com/posts/okhttp-example-rest-client

OrmLite—Lightweight Java ORM Supports Android and SQLite. (2020). http://ormlite.com/sqlite_java_android_orm.shtml

Radio Buttons. (2019, prosinec 27). Android Developers. https://developer.android.com/guide/topics/ui/controls/radiobutton?hl=cs

Raphael, J. R. (2019, září 13). *Android versions: A living history from 1.0 to 10*. Computerworld. https://www.computerworld.com/article/3235946/android-versions-a-living-history-from-1-0-to-today.html

Save key-value data. (2020). Android Developers. https://developer.android.com/training/data-storage/shared-preferences?hl=cs

Settings.Secure. (2019). Android Developers. https://developer.android.com/reference/android/provider/Settings.Secure?hl=cs

Sinicki, A. (2018, únor 19). *How to build a custom launcher in Android Studio—Part One*. Android Authority. https://www.androidauthority.com/make-a-custom-android-launcher-837342-837342/

Späth, P. (2018). *Pro Android with Kotlin: Developing Modern Mobile Apps*. Apress.

Support different pixel densities. (2020). Android Developers. https://developer.android.com/training/multiscreen/screendensities?hl=cs

Support different screen sizes. (2020, leden 14). Android Developers. https://developer.android.com/training/multiscreen/screensizes?hl=cs

Šantora, D. (2016, srpen 18). *T-Mobile G1: Jaký byl první Android před sedmi lety a jak by obstál dnes? (test)*. https://smartmania.cz/t-mobile-g1-jaky-byl-prvni-android-pred-sedmi-lety-a-jak-by-obstal-dnes-test/

Thornsby, J. (2019, říjen 18). *Kotlin vs Java for Android: Key differences*. Android Authority. https://www.androidauthority.com/kotlin-vs-java-783187/

Tyagi, A. (2019, únor 7). 10 Best Android Launchers To Customize Your Smartphone In 2019. *TechWorm*. https://www.techworm.net/2019/02/best-android-launchers.html

Vikani, H. (2018, leden 28). Android – Creating Simple Launcher in Android Studio. *ParallelCodes*. https://parallelcodes.com/create-android-launcher-program/