Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Aplikace Android – teorie a praxe

Karel Doležal

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Karel Doležal

Informatika

Název práce

Aplikace Android – teorie a praxe

Název anglicky

Android Applications – Theory and Practice

Cíle práce

Diplomová práce je tématicky zaměřena na problematiku aplikací pro operační systém Android od společnosti Google. Hlavním cílem práce je charakteristika vývoje aplikací, jejich distribuce a uplatnění na trhu.
Dílčí cíle diplomové práce jsou:
- analyzovat obecné požadavky na aplikace pro operační systém Android
- charakterizovat různe pohledy na využití operačního systému Android a vyzkoušet některé aplikace v praxi

Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní řešení je realizováno formou návrhu a implementace vlastní jednoduché aplikace pro operační systém Android. Na základě syntézy teoretických poznatků a výsledků vlastního řešení budou formulovány závěry diplomové práce.

Doporučený rozsah práce

60-80 stran

Klíčová slova

Android, mobilní operační systém, Java

Doporučené zdroje informací

Android http://developer.android.com/guide/index.htm
Android SDK for Windows – http://code.google.com/android/documentation.html
MEIER R. Professional Android Application Development, 2. vydání. Indianapolis: Wrox, 2010, 576 str., ISBN 0470565527.
MURPHY, Mark L. Beginning Android 2. 1. vydání. New York: Springer Verlag 2010. 416s. ISBN 978-1-4302-2629-1.
ROGERS, R., LOMBARDO, J. Android Application Development, 1st Edition. Cambridge: O'Reilly Media, Inc. 2009, 336s., ISBN 978-0-596-52147-9.
Symbian Developer: home – http://www.symbian.com/developer/index.html
TOPLEY, Kim. J2ME v kostce – Pohotová referenční příručka. Praha: GRADA, 2004,536s., ISBN 80-247-0426-9.
Windows Mobile Developer Center – http://msdn2.microsoft.com/en-us/windowsmobile/default.aspx

---

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Čestmír Halbich, CSc.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 18. 10. 2016

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 24. 10. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 13. 03. 2017

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Aplikace Android – teorie a praxe" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30. 3. 2017 _____

Poděkování

Rád bych touto cestou poděkoval panu Ing. Čestmírovi Halbichovi, CSc. za odborné vedení a cenné připomínky k dokončení diplomové práce.

Aplikace Android – teorie a praxe

Souhrn

Diplomová práce se zabývá charakteristikou vývoje aplikací pro operační systém Android, distribucí a uplatněním různých Android verzí na trhu. V první části jsou analyzovány požadavky na kvalitní aplikaci Android a souhrn různých pohledů na ně. Další část je zaměřena na podrobnou charakteristiku architektury a vývoj aplikací pro operační systém Android. Praktická část se zaměřuje na tvorbu aplikace pro chytrou domácnost, která je členěna na návrh a implementaci klienta a serveru. V této kapitole je charakterizována komunikační metoda TCP/IP, sériová komunikace přes bluetooth, stream videa, snímání teploty a regulace spínání konstrukčních prvků. Na závěr jsou shrnuty teoretické poznatky, doporučení a výsledky práce.

Klíčová slova: Android, mobilní operační systém, Java, chytrá domácnost, Raspberry Pi, Arduino, TCP/IP, bluetooth

Android Applications – Theory and Practice

Summary

This diploma thesis characterizes development of application for the Android operating system, distribution and application of various Android versions to the market. The first part describes of analysis requirements for high quality Android application and summary of the various views of them. The next part includes a detailed description of the architecture and development of applications for the Android operating system. The practical part is contains a description of the creation of applications for smart home, which is divided on the design and implementation of client and server. In this chapter is described communication method TCP/IP, serial communication via Bluetooth, video stream, temperature sensing and control switching components. The conclusion summarizes theoretical findings, recommendations and results thesis.

Keywords: Android, mobile operating system, Java, smart home, Raspberry Pi, Arduino, TCP/IP, bluetooth

Obsah

Seznam obrázků

Seznam tabulek

Seznam zkratek

UI – User Interface

GPS – Global Positioning System

OS – Operating System

NFC – Near Field Communication

VPN – Virtual Protocol Network

WVGA – Wide Video Graphics Array

JIT – Just In Time

HTML – HyperText Markup Language

VoIP – Voice over Internet Protocol

API – Application Programming Interface

MIDI – Musical Instrument Digital Interface

AAC – Advanced Audio Coding

GUI – Graphical User Interface

IPC – Inter-Process Communication

CMS – Content Management System

XML – Extensible Markup Language

JDK – Java Development Kit

SSH – Secure Shell

JSON – JavaScript Object Notation

UUID – Universally Unique Identifier

SDK – Software Development Kit

Úvod

Operační systém Android dosud prochází jedním z nejrychleji rostoucích systémů na trhu v oblasti chytrých telefonů a tabletů. V současnosti životní styl hlavně mladých lidí, kteří používají své smartphony každým dnem, určuje trend ve využívání mobilních aplikací. Tomuto relativně mladému operačnímu systému se podařilo v prvních letech vývoje nahradit slavný operační systém Symbian a stal se tak přímým konkurentem iOS vyvíjeným společností Apple.

Počínaje srpnem 2016 si Android drží až 87 procentní podíl na trhu v oblasti implementovaných operačních systémů v mobilních zařízeních. Do svých výrobků jej instalují giganti, jako jsou Samsung, Huawei, Lenovo, Asus atd. Uživatelé Androidu si tento systém velice oblíbili a to především z důvodu jednoduchosti uživatelského rozhraní, standardní správy souborů, rychlosti systému nebo pro jeho nižší cenu na trhu.

Dokumentace vytvářená společností Google je velice kvalitní a přehledná. Díky tomu je Android velice oblíbený také mezi vývojáři pro tvorbu jeho aplikací. Zdrojový kód je volně dostupný a šířený jako open source. Výrobci mobilních zařízení využívají této skutečnosti, která jím dovoluje instalovat systém bez licenčních poplatků. Proto mohou firmy nabízet nižší ceny a jejich výrobky tak nabídnout i nižší vrstvě obyvatelstva. Protože je systém šířený jako open source mohou ho využívat i lidé s vyššími ambicemi, kteří chtějí řešit svoje požadavky a nápady individuálně a chtějí ho realizovat formou mobilní aplikace. Poté mohou jednoduše využít distribučního kanálu Google Play pro jeho publikaci.

Hlavním přínosem práce je soustředění se na tvorbu aplikace pro skupinu lidí, kteří si chtějí zdokonalit svoje bydlení. V praktické části je hlavně charakterizován vývoj tvorby takovéto aplikace, a co vše je potřebné k její realizaci.

1 Cíl práce a metodika

1.1 Cíl práce

Diplomová práce je tematicky zaměřena na problematiku aplikací systému Android od společnosti Google. Hlavním cílem je charakteristika vývoje aplikací, jejich distribuce a uplatnění na trhu.

Dílčí cíle diplomové práce jsou:

- Analyzovat obecné požadavky na aplikace pro operační systém Android
- Charakterizovat různé pohledy na využití operačního systému Android a vyzkoušet některé aplikace v praxi

1.2 Metodika

Metodika řešené problematiky diplomové práce je založena na studiu a analýze odborných informačních zdrojů. Vlastní řešení je realizováno formou návrhu a implementace vlastní jednoduché aplikace pro operační systém Android. Na základě syntézy teoretických poznatků a výsledků vlastního řešení budou formulovány závěry diplomové práce.

2 Obecné požadavky na aplikace Android

Uživatelé operačního systému Android očekávají od vývojářů vysoce kvalitní aplikaci, která má přímý vliv na dlouhodobý úspěch. Po instalaci má uživatel právo aplikaci hodnotit nebo komentovat a to při nesplnění požadavků může aplikaci zcela podhodnotit. Z tohoto důvodu, je nutné se této oblasti dostatečně věnovat a neopomenout ji.

Android Developers věnující se této problematice vymezil soubor kritérií kvality, podle níž hodnotíme a testujeme jak je naše aplikace kvalitní z hlediska klíčových činností. Testování by mělo probíhat na mnoha zařízeních s různými verzemi systému, kde zjišťujeme, zda aplikace splňuje Android normy pro navigaci a design. Účelem je stanovit základní charakteristiky kvality, které bychom měli zahrnout ve svých testovacích plánech. Všechny aplikace by měly splňovat tyto kritéria.

2.1 Vizuální design a interakce s uživatelem

Tato kritéria zajistí, že vaše aplikace bude poskytovat standardní Android vizuální design a interakce pro konzistentní a intuitivní běh s uživatelskými schopnostmi.

Vizuální design a interakce s uživatelem		
Oblast	**Popis**	**Testy**
Běžný design	Aplikace dodržuje zásady designu pro Android a využívá běžné vzory uživatelského rozhraní (dále UI) a ikony: a) aplikace nemění očekávané funkce systémových ikon b) aplikace není nahrazena úplně jinou systémovou ikonou, když se spouští standardní UI c) pokud aplikace nabízí upravené verze standardních systémových ikon, ikona se silně podobá systémové ikoně a spouští očekávané operace d) aplikace nemění nebo nezneužívá běžné vzory UI takovým způsobem, že ikony nebo chování by mohlo být pro	CR-all

	uživatele matoucí.	
Navigace	Aplikace podporuje standardní systém zpětné navigace a nepodněcuje k používání vlastních návratových tlačítek na obrazovce.	CR-3
	Všechny dialogy se dají zrušit použitím tlačítka „zpět“.	CR-3
	Po stisknutí tlačítka „domů“ vždy uživatele vrátí na domovskou obrazovku zařízení.	CR-1
Upozornění	Upozornění dodržující zásady designu pro Android. Konkrétně: a) pokud je to možné, jsou upozornění shrnuta do jediného upozornění b) upozornění vytrvávají, jen pokud souvisí s probíhající událostí (jako je hudební přehrávač nebo telefonní hovor) c) upozornění neobsahují reklamu nebo obsah nesouvisející s jádrem funkcí aplikace, pokud to uživatel nepožaduje.	CR-11
	Aplikace využívá upozornění pouze: a) na indikace změny v souvislosti týkající se přímo uživatele (například příchozí zpráva) nebo b) na zobrazení informace v souvislosti s probíhající událostí (jako je přehrávání hudby nebo telefonní hovor)	CR-11

Tab. 1 – Testování vizuálního designu a interakce s uživatelem[17]

2.2 Funkčnost

Tato kritéria zajistí, že aplikace poskytuje očekávané funkcionální chování s odpovídající úrovní oprávnění.

Funkčnost		
Oblast	**Popis**	**Testy**
Oprávnění	Aplikace požaduje takové minimum oprávnění, které je potřeba pro podporu základní funkce.	SC-4
	Aplikace nežádá přístup k citlivým údajům (jako jsou kontakty nebo systémové záznamy) nebo k službám, které mohou uživatele stát peníze (např. vytáčení, SMS), pokud není nezbytně nutné pro smysl	

	aplikace.	
Umístění aplikace	Aplikace funguje normálně, když je nainstalována na SD kartě (pokud to aplikace podporuje). Podpora instalace na SD kartu je doporučována pro všechny větší aplikace (více jak 10MB).	SD-1
Zvuk	Zvuk se nepřehrává, když je vypnutá obrazovka, pokud se nejedná o klíčovou funkci. (např. hudební přehrávač).	CR-7
	Zvuk se nepřehrává při zamknuté obrazovce, pokud se nejedná o klíčovou funkci.	CR-8
	Zvuk se nepřehrává na domovské obrazovce nebo přes jinou používanou aplikaci, pokud to není klíčová funkce.	CR-1
	Zvuk se obnoví, když je aplikace vrácena do popředí nebo indikuje uživateli, že přehrávání je v pozastaveném stavu.	CR-1 CR-8
UI a grafika	Aplikace podporuje horizontální i vertikální orientaci (je-li to možné). Při změně orientace zůstávají stejné prvky, činnosti a je zachována funkčnost. Menší změny v obsahu nebo pohledu jsou přijatelné.	CR-5
	Aplikace používá celou obrazovku v obou orientacích a není přidáváno ohraničení. Menší ohraničení pro kompenzaci malé odchylky v geometrii obrazovky je přijatelné.	CR-5
	Aplikace bez problému zpracovává rychlé přechody mezi orientací displeje.	CR-5
Stav uživatele a aplikace	Aplikace by neměla ponechávat spuštěné služby na pozadí, pokud to nesouvisí s klíčovou funkcí. Například by aplikace neměla ponechat běžet služby pro síťové připojení kvůli upozorněním nebo udržovat spojení přes Bluetooth nebo udržovat zapnuté GPS.	CR-6
	Aplikace správně udržuje a obnovuje svůj stav. Aplikace správně zachovává svůj stav, když uživatel odchází z popředí a zabraňuje ztrátě dat, kvůli změně stavu. Když se vrátí do popředí, aplikace musí obnovit svůj zachovalý stav a obnovit všechny významné změny, které čekaly na zpracování, jako je	CR-1 CR-3 CR-5

	změna upravitelných polí, postup ve hře, videí, menu a jiných částech aplikace. a) když aplikace je obnovena ze seznamu nedávných aktivit, vrátí se uživateli přesný stav, v jakém byl naposledy užíván b) když se aplikace obnoví po probuzení ze spánku (zamčení), vrátí se uživateli v přesném stavu, v jakém byl naposledy užíván c) když je aplikace obnovena z domovské obrazovky nebo seznamu aplikací, vrátí se uživateli do stavu, který nejpřesněji odpovídá původnímu stavu d) po stisknutí klávesy „zpět“ aplikace dá uživateli možnost uložení jakékoliv aplikace nebo stavu, které by jinak byly ztraceny	

Tab. 2 – Testování funkčnosti[17]

2.3 Kompatibilita, výkonnost a stabilita

Tato kritéria zajistí, aby aplikace poskytovala kompatibilitu, výkon, stabilitu a schopnost reakce očekávané uživatelem.

.

Kompatibilita, výkonnost a stabilita		
Oblast	**Popis**	**Testy**
Stabilita	Aplikace nepadá, sama se nevypíná, nezamrzá nebo se nechová abnormálně na jiných cílených zařízeních.	CR-all, SD-1, HA-1
Výkon	Aplikace se načítá rychle nebo poskytuje uživateli informace o načítání (např. indikátor průběhu) při delším spouštění než dvě sekundy.	CR-all, SD-1
	Při aktivaci striktního módu, není viditelné žádné červené blesknutí (výkonnostní varování ze striktního režimu) při běhu aplikace, hry, animacích, přechodech UI a jiných částech aplikace.	PM-1
SDK	Aplikace funguje na nejnovější veřejné verzi Android bez padání nebo ztráty klíčových funkcí.	CR-0

	Aplikace se zaměřuje na nejnovější SDK s nastavenou targetSdk hodnotou pro minimální používanou verzi Android.	SP-1
	Aplikace je postavena na nejnovější SDK s nastavenou compileSdk hodnotou.	SD-1
Baterie	Aplikace patřičně podporuje funkce správy napájení ve verzích Android 6.0 a vyšší. V případě, že je základní funkce narušena správou napájení, mohou aplikace požádat o výjimku.	BA-1
Media	Přehrávání videa a hudby je plynulé, bez praskání nebo bez jiných chyb při běžném používání.	CR-all, SD-1, HA-1
Vizuální kvalita	Aplikace zobrazuje grafiku, text, obrázky a další UI prvky bez viditelných narušení, rozmazání nebo viditelnosti pixelů. a) Aplikace poskytuje vysokou kvalitu zobrazení pro všechny cílové obrazovky a velké obrazovky jako jsou tablety. b) Menu, tlačítka a další UI prvky jsou na okrajích viditelné a nedochází k tzv. „aliasingu“.	CR-all
	Aplikace zobrazuje text a textové bloky přijatelným způsobem. a) Kompozice je přijatelná ve všech podporovaných zařízeních pro velkou obrazovku, jako jsou například tablety. b) Nejsou viditelná žádná „uříznutá“ písmena. c) Nejsou viditelná žádná nesprávná zalomení slov na tlačítkách a ikonách. d) Dostatek místa mezi textem a okolními prvky.	

Tab. 3 – Testování kompatibility, výkonosti a stability[17]

2.4 Testovací postupy

Tyto testovací postupy pomáhají objevit různé typy problémů v oblasti kvality vaší aplikace. Testy můžeme kombinovat nebo integrovat do skupiny testů dohromady ve vlastních zkušebních plánech. V tabulkách výše jsou uvedené kódy testu, které spojují určitá kritéria se specifickými testy.

Typ	Test	Popis
Klíčové	CR-0	Procházení všech částí aplikace – obrazovky, hlášení, nastavení a všech uživatelských prvků. a) V případě, že aplikace umožňuje editaci nebo tvorbu obsahu, hraní her nebo přehrávání médií, vyzkoušejte všechny tyto prvky. b) Při testování aplikace vyzkoušejte chování přechodných změn v připojení k síti, funkčnost baterie, GPS nebo jiné určení polohy, načítání systému apod.
	CR-1	Na každé aplikační obrazovce stiskněte na zařízení tlačítko „home“ a vraťte se do aplikace ze seznamu všech aplikací.
	CR-2	Na každé obrazovce přepněte na jinou aplikaci a vraťte se do aplikace přes seznam naposledy použitých aplikací.
	CR-3	Z každé obrazovky aplikace a z každého hlášení stiskněte tlačítko „zpět“.
	CR-5	Na každé obrazovce změňte alespoň třikrát horizontální a vertikální zobrazení.
	CR-6	Přepněte na jinou aplikaci pro odeslání testovací aplikace na pozadí. Přejděte do nastavení a zkontrolujte, zda má aplikace při běhu na pozadí spuštěné nějaké služby. Ve verzi Android 4.0 a vyšší, jděte do nastavení aplikací a najděte ji v „Running“ položce.
	CR-7	Stiskněte tlačítko napájení pro přepnutí přístroje to režimu spánku a poté stiskněte znovu pro jeho probuzení.
	CR-8	Nastavte zařízení tak, aby se zablokovalo při stisku napájecího tlačítka. Stiskem uspěte, poté probuďte a odblokujte.
	CR-9	Zařízení, která mají vysouvací klávesnici, vysuňte alespoň jednou. U zařízení s přídavnou klávesnicí ji připojte.
	CR-10	Zařízení, která mají externí obrazovku, připojte.
	CR-11	Spusťte a pozorujte všechna oznámení, která aplikace může zobrazit. Kde je to možné aktivujte oznámení a vyzkoušejte jejich nabízenou činnost. (možné od verze Android 4.0)

Instalace na SD kartu	SD-1	Zopakujte klíčové testy, bude-li aplikace umístěna na SD kartě zařízení (pokud je podporována aplikací). Přemístěte aplikaci do SD karty přes kroky Nastavení > Aplikace Info > přesunout na SD kartu.
Hardwarová akcelerace	HA-1	Zopakujte klíčové testy s povolenou hardwarovou akcelerací. Chcete-li povolit hardwarovou akceleraci (pokud to zařízení podporuje), přidejte do manifestu aplikace harware-accelerated="true" a překompilujte.
Výkon a stabilita	SP-1	Prozkoumejte soubor manifest a vytvořte konfiguraci, aby byla postavena proti nejnovějším dostupným SDK.
Monitorování výkonu	PM-1	Zopakujte klíčové testy s povoleným striktním módem.

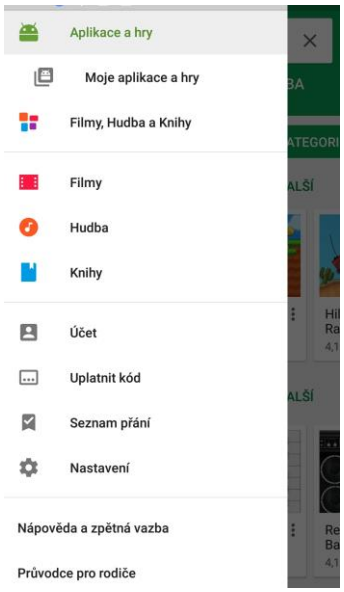
Tab. 4 – Testovací postupy[17]

2.5 Distribuce aplikací

Distribuční kanál aplikací byl zpřístupněn společností Google v roce 2008. Jako první vznikla služba pod názvem Android Market a o několik let později vznikla doplňková služba Google Music. Poté se společnost rozhodla pro sloučení těchto dvou distribučních kanálů a uvedla ji pod názvem Google Play, kterou dále vyvíjí.

V současnosti, jsou aplikace pro operační systém Android dostupné ke stažení na Google Play Store, který je přímým nástupcem Android Marketu a je otevřen pro vývojáře třetích stran. Google Play je online distribuční služba, která nabízí různé druhy zábavy, ke kterým lze přistupovat pomocí počítače nebo jiných zařízení. V druhém případě musí být zařízení vybaveno operačním systémem Android. Ke službě lze také přistupovat prostřednictvím Google TV nebo jinak nazývané jako Android TV. Primárně je kanál specializován pro mobilní telefony a tablety. Play Store umí filtrovat aplikace podle dat zadaných v manifestu. Například aplikaci nezobrazí zařízením, která mají verzi OS Android nižší než je min SDK version. Některé elementy, se kterými může Play Store pracovat uvádím v kapitole 2.5.2.

Google Play umožňuje nákup filmů, knih, hodnocení aplikací, editaci vlastního profilu, včetně sestavování svého seznamu přání, vyhledávání podle vlastního klíče a dalších jiných možností pro požadavky uživatele na jednom místě.



Obr. 1 – Google Play v mobilu

Abychom mohli nabízet aplikaci veřejnosti, je nutné se zaregistrovat a zaplatit poplatek ve výši 25 dolarů (uvedeno k březnu 2017). Šířit aplikaci můžeme za námi stanovenou cenu nebo zdarma. V prvním případě si Google účtuje 30% z ceny, za každý nákup. Rozhodneme-li se aplikaci prodávat, je nutné si vytvořit účet služby Google Wallet Merchant Center, kde stanovujeme ceny.

2.5.1 Finanční modely

Google Play nabízí několik způsobů na zpeněžení aplikace. Za prvé můžeme nabízet aplikaci, kterou je možné bezplatně stahovat, kde výdělek těžíme z reklamy zakomponované v aplikaci. Dalším způsobem je prodej aplikace za předem známou cenu, kdy uživatel nejdříve zaplatí a poté se uživateli zpřístupní možnost stažení. Třetí možností je nákup v aplikaci, kde lze zakoupit obsah pro rozšíření přístupů v dané aplikaci a podobně. Posledním způsobem jsou odběry, kde vám bude zpřístupňován příslušný obsah. Odběry se automaticky obnovují a na začátku každého období odběru vám bude automaticky stržena platba.

2.5.2 Publikace

Před publikováním je důležité aplikaci otestovat na hardwarovém zařízení podle kapitoly 2.4. Níže je uvedeno jen několik obecných příkladů:

- Změnit orientaci displeje.
- Vypnout aplikaci tlačítkem zpět.
- Celou aplikaci proklikat a schválně se chovat neobvykle.
- Aplikaci ukončit tlačítkem *domů* a pak se na ni vrátit.
- Vypnout bluetooth nebo internet (vyžaduje-li to aplikace).

Dále stanovit si a odpovědět na různé otázky typu:

- Jsou v seznamu všechny ikony správně?
- Nejsou názvy příliš dlouhé?
- Nedělá widget problémy, když se zvětší?

Aplikaci je také potřeba vyzkoušet na všech podporovaných verzích Android. Testování by mělo proběhnout nejen na mobilním zařízení ale také na tabletu. Důležitá je také použitelnost aplikace, aby byla uživatelsky přístupná.

Aby bylo možné aplikaci volně šířit přes Play Store, je nutné splnit několik základních podmínek. Samotný element <manifest> má dva důležité atributy android:versionCode a android:versionName. První z atributů musí být celé číslo, které uvádí interní identifikaci aplikace. Obvykle s každou novou verzí aplikace zvýšíme toto číslo o jedničku. Play Store toho využívá pro kontrolu nové verze. Druhý z uvedených atributů identifikuje aktuální verzi a je to jakýkoli řetězec, který se ukáže jako verze uživateli. Důležité je, aby nová verze měla vyšší versionCode než verze předchozí.

Pokud máme vytvořeny všechny ikony, aplikaci pořádně otestovanou a nastavené všechny manifestové elementy, můžeme zkompilovanou aplikaci zabalit, podepsat a zveřejnit.

2.6 Různé pohledy na operační systém Android

Aplikace pro operační systém Android jsou velice různorodé a mohou být kategorizovány podle jejich smyslu využití. Můžeme je dělit do kategorií, jako jsou vzdělávání, hry, sociální sítě, finanční, byznys, hudba, oblast automobilů a mnoho dalších. Z toho vyplývá, že aplikace mohou být specializovány pro potřeby jednotlivých lidí, domácností až po využití malých či velkých společností.

Jednotliví lidé mohou mít potřeby osobního rozvoje, kde jim aplikace pomáhá například ve vzdělávání. Pomocí jednoduchého upozorňování na krátké vykonání nějaké vzdělávací činnosti, může uživateli něco připomenout, co si kdysi stanovil. Vše probíhá prostřednictvím mobilního zařízení, které většinou nosíme neustále u sebe. Příklad této aplikace je v kapitole dále popisován.

Z pohledu zaměstnance trávícího mnoho času na cestách, může aplikace, jako je navigace velice pomoci při jeho výkonu práce. Ovšem musíme počítat, že navigace orientované na specifický druh práce mohou být placené. Pro standardního uživatele jsou tyto navigace většinou zdarma, ale jen za předpokladu, že je využíváme offline. Je-li navigace připojena k internetové síti, lze aplikaci využívat plnohodnotněji. V tomto případě máme přístup k aktualizovaným datům, která vytváří jiní uživatelé na cestách a tak upozorňují na stav momentální silniční dopravy. Dle toho lze vyhodnocovat objížďky a tak ušetřit jinému uživateli čas, který by jinak strávil v dopravní zácpě. Další vychytávkou mohou být aktualizované informace o různých stálých omezeních v dopravě či probíhajících měření rychlosti na daném úseku.

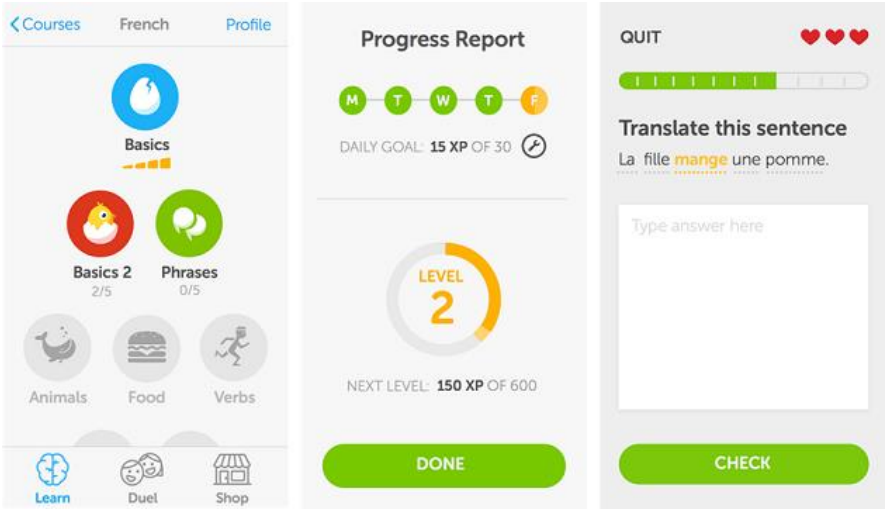
Aplikace mohou být přínosné také pro maloobchody, které nedisponují velkými finančními prostředky a mohou jim pomoci v činnosti podnikání. Tento přístup může být pro podnikatele ušetřením nákladů na provoz, které by jinak vynaložili na nový systém. Operační systém Android je velice rozšířený a implementace aplikace do prostředí podniku je jednoduchá, méně nákladová a pro zaměstnance, který používá tento systém ve svém zařízení srozumitelnější. Jako příklad takového systému můžeme uvést aplikaci na elektronickou evidenci tržeb, která je dále v kapitole popisována.

Google Play (podrobněji v kapitole 2.5) nabízí mnoho sekcí, které rozdělují aplikace podle jejich zaměření.

2.6.1 Sekce vzdělávání

Jak již je zmiňováno na začátku kapitoly, existují aplikace, které umožňují vhodnou formou uživatele vzdělávat, aniž by ho to stálo mnoho času a úsilí. Jednou z takových aplikací je Duolingo, která se specializuje na výuku cizích řečí. K datu 11. 3. 2017 je to nejstahovanější aplikace v této sekci a uživateli velice kladně hodnocena.

Duolingo v současnosti nabízí mnoho kurzů včetně češtiny, kde jsme ale omezeni jen na anglický překlad. Pro výuku ostatních jazyků je zapotřebí znalosti angličtiny. Účelem je věnovat aplikaci alespoň 10 minut denně a procházet jakýsi „jazykový strom“, který je tvořen různými dovednostmi a tématy. Pokroky ve výuce lze sdílet na sociálních sítích a navzájem je porovnávat či komentovat se svými přáteli přímo v aplikaci.



Obr. 2 – Aplikace Dualingo[20]

2.6.2 Sekce mapy a doprava

Ze sekce mapy a doprava můžeme stáhnout například aplikaci Aponia GPS Navigation, která je úzce specializována pro kamionovou dopravu. Aplikace je vytvořena tak, aby počítala nejlepší trasu podle parametrů vozidla, zvolených parametrů trasy, typu zatížení a je schopna zohlednit i nebezpečné materiály. Logistické firmy si jsou vědomy,

že výběr špatné trasy mnohdy vede k plýtvání paliva, časové ztrátě a v některých případech i k bezpečnostním problémům.

Aplikaci lze stáhnout zdarma, ale chceme-li využívat podpory a aktualizovaných map je nutné zaplatit licenci. Abychom mohli využívat online službu, která zahrnuje aktuální dopravní informace, zobrazení pozic přátel na mapě a tak dále, musí být aplikace postavena tak, aby umožňovala připojení k databázové správě dané firmy.

Data jsou do Aponia GPS navigace distribuována pomocí GPRS přenosu a uživatel má možnost definovat interval aktualizací těchto dat v rozmezí 2 až 30 minut. Architektura Aponia dopravních informací byla koncipována tak, aby minimalizovala velikost toku dat přes GPRS [15]. Uživatelé zařízení umožňující mobilní datové přenosy (E/3G/4G/LTE apod.) si tak vystačí i se základní nabídkou připojení k internetu svého operátora.

Obr. 3 – Aplikace Aponia GPS Navigation[15]

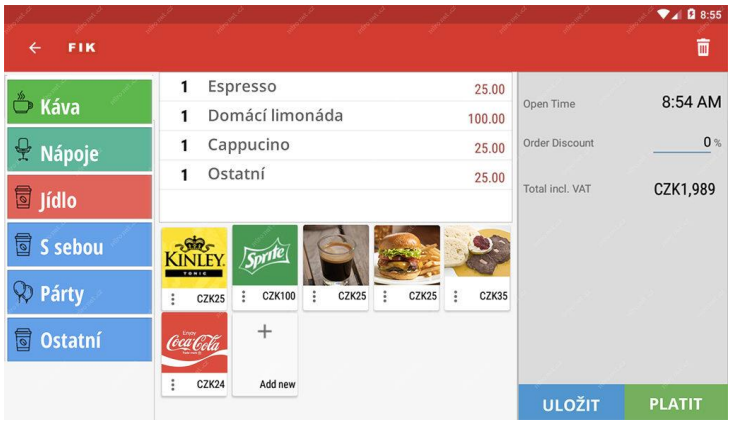
Obecně pro aktuální zobrazení naší pozice v Android aplikaci stačí, aby dané zařízení disponovalo modulem GPS a běželo na systému Android. Výpočet trasy tak probíhá v samotné logice postavené aplikace i bez připojení k internetové síti, máme-li staženy aktuální mapy.

2.6.3 Sekce byznys

V této sekci nalezneme aplikace pro účely podnikatelské činnosti, jako je například aplikace pro evidenci elektronické tržby s názvem KASA Fik. Použití této aplikace v maloobchodech je velmi výhodné z toho důvodu, že není zapotřebí kupovat drahé pokladny. Jednoduše ji lze nainstalovat na tablet nebo na mobilní zařízení běžící na

operačním systému Android. Tato aplikace musí být postavena tak, aby uměla komunikovat se servery finanční správy. Uživatel při platbě zasílá pomocí internetového připojení zprávu Finanční správě, která platbu eviduje. Poté aplikace přijímá unikátní kód a může tisknout účtenku.

Při prvním spuštění je nutné se zaregistrovat a udělat základní nastavení, kde zadáváme údaje o firmě a podnikatelské osobě. Po těchto krocích je podnikatel identifikován a je možné v aplikaci jednoduše vytvářet účtenky. Pro tisk účtenek je možné v aplikaci nastavit tiskárnu, se kterou dále komunikujeme pomocí bluetooth, WiFi nebo usb kabelu.



Obr. 4 – Aplikace KASA FIK[19]

Aplikace můžeme dále rozlišovat podle toho, na jakém typu zařízení budou fungovat. Existují specifické druhy zařízení, kam lze aplikace běžící na operačním systému Android implementovat. Vždy jsou na prvním místě požadavky uživatele, které se mohou lišit podle použitelnosti v dané oblasti, jako je například automobil, televize, chytré hodinky nebo třeba Google Glass.

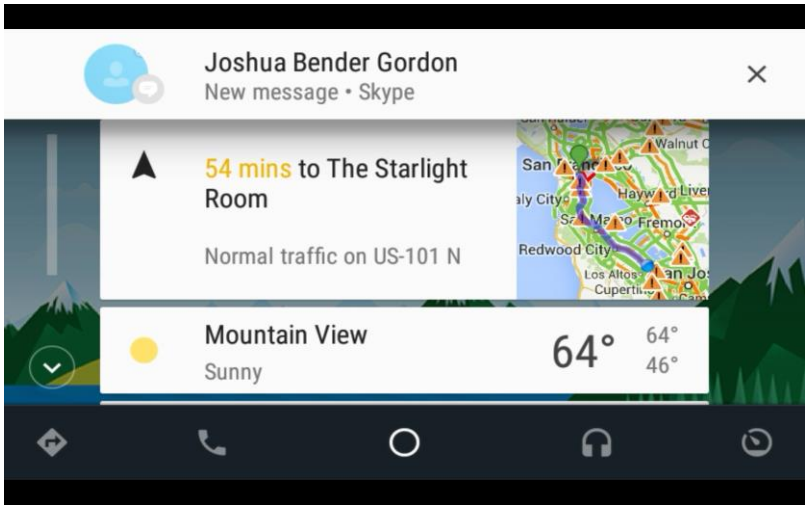
2.6.4 Android auto

Google v roce 2014 představil novou softwarovou platformu určenou pro automobily. Nabízí aplikaci Android Auto, která není v současnosti dostupná v českém obchodě Google Play, ale nalezneme ji na jiných portálech, které instalační balíček apk nabízí. Je prezentována tak, že funguje s jakýmkoli autem. Stačí se v autě připojit telefonem pomocí USB rozhraní a začít ji používat přes zabudovaný palubní počítač. Nedisponuje-li auto tímto palubním počítačem, můžeme aplikaci ovládat na svém telefonu, která je plně kompatibilní s možnostmi auta, jako je například bluetooth, handsfree, přehrávání hudby

prostřednictvím reproduktorů a mnoho dalších. Aplikace je navržena s jednoduchým uživatelským rozhraním, s velkými tlačítky a možností hlasové komunikace, aby uživatel i při jízdě mohl aplikaci snadněji ovládat [24]. Nabízí využití navigace Google Maps, poslouchání vaší oblíbené hudby, posílání zpráv pomocí hlasu (možno jen na odpověď) a mnoho dalšího. V současnosti není aplikace podporována v Českém jazyce a tak je nutné si vystačit například s Angličtinou.

Aplikace je dostupná pro verze Android 5.0 nebo vyšší. Pro plnohodnotné využívání je zapotřebí připojení k síti a mít aktualizované některé stávající aplikace v mobilu, jako je například Google Maps, Google Play Music nebo Google Search.

Android Auto je plně kompatibilní s vozy Škoda a mnoho dalšími nejnovějšími automobily. V případě Škoda auto plnohodnotně funguje se službou SmartLink. Pro spuštění aplikace, se připojíme v autě microUSB kabelem a poté v dialogovém okně povolíme přístup k datům zařízení. Následně displej telefonu zčerná a zobrazí se logo Android Auto. V tuto chvíli telefon přestane reagovat na vstupy od uživatele a je dostupná jen na palubním počítači. Samotná platforma vypadá jednoduše a přehledně. Na úvodní stránce nalezneme přehled důležitých informací, kde vidíme aktuální počasí, poslední uskutečněné hovory, nadcházející schůzky z kalendáře, navigaci na uložená místa nebo widget aktuálně probíhající navigace. V systému se pak můžeme pohybovat ve spodní černé liště, která umožňuje přepínání mezi jednotlivými obrazovkami a aplikacemi. Zde je povoleno jen 5 funkcí a to navigace, volání, domovská obrazovka se všemi relevantními notifikacemi najednou, hudba a servisní tlačítko.



Obr. 5 – Aplikace Android Auto běžící na palubním počítači[21]

Úplně vlevo se nachází tlačítko navigace, která otevře mapy Googlu. Ovšem nejedná se o duplicitu aplikace té, kterou máme v telefonu, ale jedná se o tu samou, jen je zobrazována na jiném displeji s trochu jiným rozložením. Na rozdíl od zobrazení na telefonu postrádá některé funkce, které nejsou v autě z hlediska významu potřebné. Naopak disponuje těmi, které využijeme. Kdykoliv během jízdy můžeme snadno najít pomocí hlasové komunikace nebo ručním zadáním blízké čerpací stanice nebo třeba restaurace. V aplikaci lze také využít tzv. „hamburger“ menu, kde nalezneme například poslední hledané místa. Protože aplikace Mapy Google podporuje Český jazyk, bude rozumět i českým hlasovým povelům. Ovšem je vždy potřeba datového připojení.

Jako druhou položku ve spodní liště nalezneme aplikaci pro hovory. Ta nám na své úvodní stránce ukáže výpis posledních hovorů a oblíbené kontakty. Samozřejmě nechybí možnost zadat telefonní číslo ručně. Tuto možnost pak naleznete opět v „hamburger“ menu. Dalším tlačítkem na liště je kolečko, které vrací uživatele na domovskou obrazovku.

Další ikonou je tlačítko pro hudbu. Zde máme na výběr z aplikací, které v mobilu pro přehrávání hudby používáme, ale nejsou všechny podporované. Používáme-li Google Music Play zobrazí se nám přehrávač automaticky. Pokud používáme jinou, nastavíme ji stisknutím tlačítka pro hudbu přes dialogové okno, ve kterém si můžeme mezi aplikacemi vybírat. Máme-li nastaveno, je možné přehrávač ovládat hlasem – například stylem „Přehraj mi Michaela Jacksona“. Začne se nám přehrávat lokální obsah od tohoto interpreta. Dalším příkladem může být výraz „Najdi písničku od Ed Sheeran Shape of you“, zde je ale již vyžadováno datového připojení a využívání Google Music Play aplikace, kdy se připojujeme na playlisty a stream serveru Google.

Posledním tlačítkem na spodní liště je tzv. servisní tlačítko, které má zobrazit přesnější informace o voze. Při testování tohoto tlačítka na voze Škoda nefungovalo, protože staví Android Auto nad svůj vlastní uzavřený operační systém. Vyřeší-li Google tento problém měli bychom se dočkat například informací o stavu vozidla, množství paliva v nádrži nebo třeba informací o tlaku v pneumatikách.

2.6.5 Android TV

Android TV je smart TV platforma, založena na operačním systému Android 5.0 nebo vyšší. Poprvé byla představena 25. června 2014 na Google I/O jako následovník Google TV. Tato platforma může běžet v televizních přijímačích nebo prostřednictvím zapojeného set-top boxu, na kterém systém běží. Jako uživatel máte přístup k Google Play pro stahování android aplikací včetně streamových služeb Netflix nebo Hulu [26]. Umožňuje také přehrávat HDTV, nastavovat hudební přehrávač, sledování videí na internetu, hraní her a mnoho dalších funkcí. Pro plné využití všech funkcí je potřeba na svůj Android telefon či tablet nainstalovat aplikaci "Android TV Remote Control", která slouží především jako klávesnice pro vyhledávání, bez nutnosti použití externí klávesnice. Android TV se umí chovat i jako "Chromecast" přijímač, který umožňuje bezdrátově přehrávat video či hudbu.

2.6.6 Android wear

Android wear nyní nejnovější verze 2.0 je určena pro chytré hodinky. Umožňuje NFC platby, přesnou GPS, senzor tepové frekvence a mobilní konektivitu. Je navíc vybaven speciálními tlačítky pro Google Fit a Android Pay. Funkce vždy zapnutého displeje umožní přidávání zkratek rovnou na obrazovku, potom pomocí jednoho stisknutí můžete například objednat jízdu Uberu. Ciferníky se mohou měnit zrovna podle toho, co děláte, měnit se dají jednoduchým potáhnutím prstu po displeji. Upravit se dá barva, pozadí, zobrazené aplikace apod.

Google Fit je předinstalovaný na většině hodinek s Android Wear. Ten umožňuje nově například při běhu měřit rychlost, vzdálenost, spálené kalorie nebo tep. Měřit se dá také opakované vzpírání, kliky, dřepy apod. Dále máte možnost si vybrat, které aplikace chcete na svých hodinkách mít a to ze speciální sekce Google Play Store. Pokud hodinky podporují mobilní sítě, můžete s nimi volat, posílat zprávy, komunikovat přes sociální sítě, a to i když telefon nemáte momentálně u sebe.

2.6.7 Google Glass

Velmi zajímavým zařízením využívajícím platformu Android jsou brýle Google Glass. Jsou vybaveny procesorem, fotoaparátem a různými senzory, se kterými se setkáváme i u dnešních inteligentních telefonů. Disponují základním uživatelským

rozhraním, které je jednoduché a dá se intuitivně ovládat pomocí hlasu nebo dotykové plochy na rámu brýlí. Zařízení je umístěno na rámu brýlí nad zorným polem uživatele.

Baterie Google Glass při šikovném používání vydrží větší část dne. V předvoleném případě se vypnou, když si je sundáte z hlavy a zapnou se, když se přes ně podíváte pod úhlem asi 30 stupňů nahoru. K ovládání se používá hlasový povel „OK Glass“ následovaný možnostmi z nabídky nebo dotyková plocha [3]. Google Glass je možné spárovat s chytrým telefonem, tabletem disponujícím operačním systémem Android, ale i s notebookem s Windows.

3 Operační systém Android

Android je open-source platforma na bázi Linuxu určená hlavně pro mobilní zařízení, tedy chytré telefony, tablety, fotoaparáty nebo například navigace [3]. V současnosti se stále více přidávají i konvertibilní zařízení, tj. tablety s odpojitelnou klávesnicí, které se hodí i k méně náročnému podnikovému nasazení.

3.1 Vývoj

První zmínka o operačním systému Android, se zrodila v roce 2003 v Kalifornii, kdy společnost Android Inc. založili Andy Rubin, Rich Miner, Nick Sears a Chris White. Společnost se orientovala na chytřejší mobilní přístroje, které měly brát v úvahu nároky uživatelů a jejich polohu. Firma v prvních letech činnosti nebyla příliš známá. Byla brána jen jako jeden z dalších vývojářů softwaru pro mobilní telefony. Do povědomí uživatelů se dostala až v roce 2005, kdy byla společnost odkoupena gigantem Google. I přes tuto událost zůstali klíčoví zaměstnanci na svých postech. Andy Rubin se stal viceprezidentem mobilní divize Googlu, kdy vývoj nabral rychlejší tempo.

V prosinci 2006 se konečně vývojovému týmu podařilo dostat do světa první prototyp svého operačního systému. Ten se jmenoval Sooner a hodně připomínal operační systém Blackberry OS – nepodporoval ještě dotekovou obrazovku a měl naopak fyzickou klávesnici [1]. O skoro rok později vzniklo konsorcium Open Handset Alliance, jehož součástí se stal jak Google, tak velké množství jiných společností zabývajících se výrobou mobilních technologií včetně Intelu, Samsungu, NVIDIE, HTC, Qualcommu, LG a dalších. Google pak oznámil, že jeho cílem je vytvořit systém, jež bude fungovat na tisícovkách různých modelů.

5. listopadu 2007 přišla na svět už konečně beta verze prvního Androidu, jak ho známe – s otevřenou platformou, neomezenou na konkrétního výrobce hardwaru a s ovládáním primárně přes dotykovou obrazovku [1]. Proto se tento den dodnes slaví jako narozeniny tohoto operačního systému. Prvním komerčně dostupným telefonem s Androidem se pak stal 23. listopadu 2008 HTC Dream.

3.1.1 Historie verzí

Od první verze bylo vydáno několik aktualizací, které opravují chyby a přidávají nové funkce. Jednotlivé verze systému se jmenují abecedně podle sladkostí Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow a dosud poslední Nougar.

Android 1.5 Cupcake *(muffinek - 30. 4. 2009*)

První aktualizace umožňovala nahrávat a sledovat videa z kamery, nahrávat videa na YouTube a vkládat fotografie do Picasy přímo z telefonu. Byla uvedena s novou softwarovou klávesnicí s automatickým dokončováním slov, bluetooth s podporou headset, nové widgety a složky. Další vychytávkou byly animace při přechodu mezi obrazovkami nebo rozšířená funkce kopírovat a vkládat.

Android 1.6 Donut (*koblih s otvorem – 1. 9. 2009*)

Změny nastaly ve vylepšení Android Marketu, v novém prostředí fotoaparátu, kamery či galerie. Galerie v této verzi už umožňují označit více fotografií k vymazání, vyhledávat hlasem, vyhledávat záložky, historii, kontakty na webu z domovské obrazovky. Podporuje technologie CDMA/EV-DO, 802.1x, VPN, gesta a syntézu řeči, WVGA.

.

Android 2.0/2.1 Eclair *(podlouhlý zákusek s náplní a čokoládou na povrchu - 26. 10. 2009*)

Do změn se řadí optimalizace rychlosti hardwaru, podpora pro více velikostí rozlišení displeje, zdokonalené uživatelské prostředí, nové prostředí prohlížeče či seznam kontaktů. Mapy Google se aktualizovaly na verzi 3.1.2, přišla podpora pro Microsoft Exchange, Bluetooth 2.1, HTML5. Vylepšená je také softwarová klávesnice nebo animované tapety na domovské stránce.

Android 2.2 Froyo (*šlehačková špička obložená ovocem – 20. 5. 2010*)

Přidává nové technologie a funkce uživatelského prostředí, kde umožňuje instalovat aplikace na paměťovou kartu, vytvořit WiFi hotspot nebo sdílet internetové připojení přes kabel USB. Obsahuje webový prohlížeč s optimalizací JavaScriptu a podporou Flash 10. Díky kompilátoru JIT (Just-in-time) se podařilo zvýšit rychlost systému o 2× až 5× na

různých benchmarcích. Dále je vylepšena správa paměti RAM, přidána podpora pro OpenGL ES 2.0, vícebarevný trackball, vylepšená podpora pro Exchange, Bluetooth a přidána další vrstva vývojářského API.

Android 2.3 Gingerbread (*perníček – 6. 12. 2010*)

S verzí 2.3 přibyla podpora video formátu WebM pro HTML5 video, Near Field Communication standard, VoIP, nová vylepšená správa prostředků, upravená virtuální klávesnice, zlepšená funkce kopírovat či vkládat, rozšíření podpory nativního kódu, podpora více kamer a nových senzorů nebo nové mapy Google Maps 5 s 3D přístupem.

Android 3.0 Honeycomb (*medová plástev – 22. 2. 2011*)

S příchodem verze Honeycomb, bylo nutné upravit systém pro velké obrazovky tabletů, změnit desing nebo upravit multitasking. Google přichází s eBooks, který je zde již přístupný. Dále má vylepšený prohlížeč, umožňuje video hovory přes Google Talk nebo také disponuje USB přislušenstvím.

Android 4.0 IceCream Sandwich (*„ruská“ zmrzlina – 19. 10. 2011*)

Tato aktualizace umožňuje odemčení telefonu obličejem, má přepracovaný launcher, vylepšený je správce kontaktů, ukazatel přenesených dat a umí zachycovat panoramata nebo rozpoznávat hlasy.

Android 4.2 Jelly Bean *(barevné želatinové bonbóny – 9. 7. 2012)*

Jelly Bean přináší rozpoznávání hlasu offline, vylepšenou aplikaci fotoaparátu, vylepšenou informační lištu, možnost využití více uživatelských účtů, dělit účty na správcovské či omezené a nakonec podporuje OpenGL ES 3.0.

Android 4.4 KitKat *(čokoládové tyčinky – 3. 9. 2013)*

Poskytuje vyšší výkon, vylepšenou podporu zařízení s více jádrovými procesory, umožňuje stažení notifikační lišty v aplikacích s celoobrazovým režimem a je optimalizován pro telefony s menší pamětí RAM.

Android 5.0 Lollipop *(sladké lízátko – 25. 6. 2014)*

Zde můžeme mezi změny zařadit nový vzhled uživatelského rozhraní, efektivnější AndroidRunTime (ART), nový systém správy baterií, nový notifikační systém, 32 i 64 bitový systém, project Volta snižující energetickou náročnost systému a přidána funkce smart Lock. Dále systém umožňuje zapnout úsporný režim, prodlužující pohotovostní dobu telefonu, více uživatelů na jednom telefonu a byl přidán mód Guest. Disponuje funkcí Tap & go umožňující přenesení nastavení z jiného telefonu pomocí technologie NFC.

Android 5.1 Lollipop *(9. 3. 2015)*

Podporuje používání dvou a více SIM najednou, obsahuje Kill switch pro případ odcizení, má lepší zvuk (HDvoice), je výkonnější a bezpečnější.

Android 6.0 Marsmallow *(5. 10. 2015)*

Prodlužuje výdrž baterie o dvojnásobek, podporuje USB typu C, rozpoznává otisky prstů nebo lze definovat skupiny oprávnění pro aplikace.

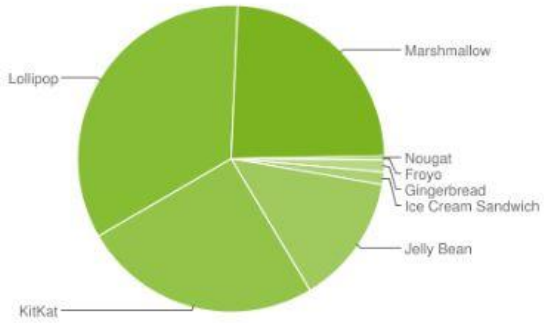
Android 7.0 Nougat *(22. 8. 2016)*

V dosud nejnovějším operačním systému Androidu můžeme využívat podpory více oken na obrazovce. Pro efektivnější práci se systémem slouží rychlé přepínání mezi naposledy použitými aplikacemi pomocí dvojího poklepání na tlačítko multitaskingu nebo také možnosti nastavení přepínačů v horní liště či rychlé odpovědi v chatovacích aplikacích. Google se zaměřil také na podporu Virtuální reality, v čemž pomáhá grafické API Vulkan.

3.1.2 Zastoupení na trhu

Na Obr. 6 jsou uvedené statistiky využívaných verzí operačního systému Android. Ze statistik vyplývá, z kolika procent jsou jednotlivé verze systému mezi uživateli využívány. Verze od 1.0 do 2.2 zde už nejsou uváděny z toho důvodu, že v zařízeních se již už převážně nepoužívají.

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	1.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.3%
4.1.x	Jelly Bean	16	4.9%
4.2.x		17	6.8%
4.3		18	2.0%
4.4	KitKat	19	25.2%
5.0	Lollipop	21	11.3%
5.1		22	22.8%
6.0	Marshmallow	23	24.0%
7.0	Nougat	24	0.3%



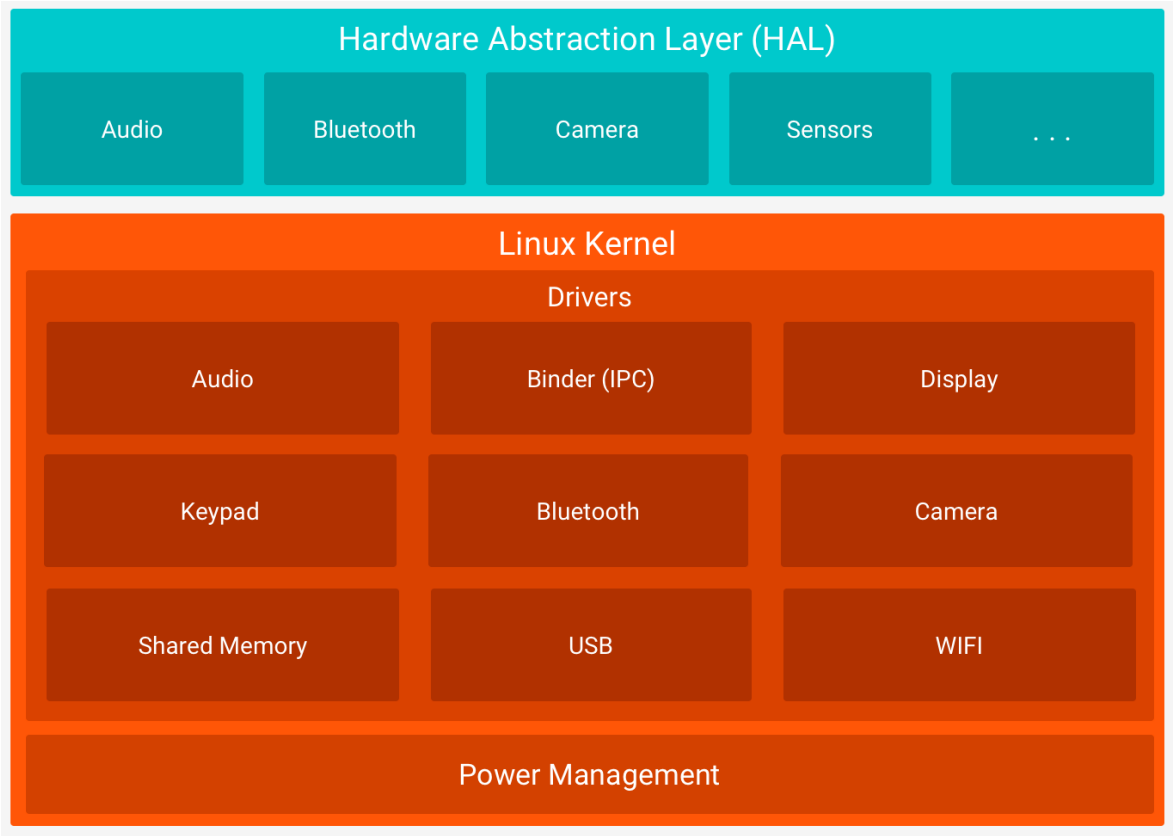
Obr. 6 - Statistiky využití verzí Android (listopad 2016)[12]

3.2 Architektura

Operační systém Android je tvořen softwarovými komponenty, které se dělí do 5 sekcí a 4 hlavních vrstev. Každá vrstva provádí různé operace, ale všechny sekce architektury dohromady spolupracují. Tímto nejsou od sebe odděleny.

3.2.1 Linux kernel

Linux kernel je nejnižší vrstva architektury Android. Má upravené jádro populárního operačního systému Linux. Úpravy se týkají redukce funkcí a jejich přizpůsobení možnostem mobilních zařízení. Vývojáři běžných aplikací se s jádrem do přímého kontaktu nedostávají. Jádro slouží přímé interakci s hardwarem mobilního zařízení, čímž zabezpečuje úplnou abstrakci od hardwaru pro vyšší softwarové vrstvy. Při použití Android Debug Bridge získáte přístup k příkazovému rozhraní Linux Shell. Prostřednictvím tohoto rozhraní můžete zadávat příkazy, které zpracovává jádro operačního systému. Linux Kernel všeobecně zajišťuje hlavní komunikaci s hardwarem a také zabezpečuje základní systémové služby jako je bezpečnost, správa paměti a procesů, správa napájení nebo například síťové připojení.



Obr. 7 – Architektura Linux Kernel a HAL[4]

Android běží na Linuxovém jádře, které se liší od počítačové verze některými úpravami. Hlavní úprava se týká redukce funkcí pro přizpůsobení mobilního zařízení. S příchodem ohromného množství zařízení byla v Androidu vytvořena hardwarová abstraktní vrstva ve zkratce HAL. Díky této vrstvě, vývojáři při programování nemusí znát všechna specifika hardwarových zařízení.

Bezpečnost

Jako bezpečnostní riziko lze označit přímou meziprocesorovou komunikaci. Aplikace a služby jsou spouštěny v oddělených procesech a často potřebují komunikovat mezi sebou. Android tento problém vyřešil vlastním meziprocesorovým ovladačem komunikace a voláním metod zvaných Binder, které vycházejí z projektu OpenBinder. Binder zajišťuje vysoký výkon pomocí sdílené paměti, přičemž data jsou předávána jako jednotlivé balíčky. Binder počítá a mapuje vzájemné reference skrze systém, takže objekty

přesouvané mezi systémem mohou být vysledovány a zrušeny v případě potřeby. Tímto se zajišťuje kontrola komunikačních procesů.

Správa paměti

Hlavním požadavkem na správu paměti je dynamické a efektivní využívání paměti běžícího programu. Z důvodu omezené paměti zařízení jsou implementovány různé nástroje pro správu paměti. Jeden z nich je Physical Memory (PMEM), který spravuje souvislou fyzickou paměť sdílenou mezi procesy. Pro práci s virtuální pamětí sdílenou mezi procesy, systém využívá nástroj Anonymous SHared MEMory (ASHEM). Dalším nástrojem je LowMemoryKiller, který je spouštěný z uživatelského rozhraní v případě kritického nedostatku paměti, kde ukončí procesy při dodržení stanovených podmínek.

Správa napájení

U operačního systému Android je vyžadováno co nejefektivnější šetření a přizpůsobení baterií mobilních zařízení. Při vývoji systému bylo nutné se zaměřit na nejnáročnější energetické části, jako jsou procesor a obrazovka. Při běhu těchto komponentů nastává velké zatížení baterie a tak vývojáři operačního systému byli nuceni vytvořit režim spánku, kdy jsou při nečinnosti uživatele tyto komponenty dočasně vypínány.

U některých aplikací nemůžeme režimu spánku využívat a to z důvodu neustálé činnosti aplikace, která je zapotřebí. Například navigace bychom bez neustálého přísunu energie nemohli používat. Soustavný chod aplikace zajišťuje systémový zámek nazývaný Wakelocks, který nechá zařízení aktivní. Existují celkem 4 úrovně Wakelocks, které se dají aktivovat podle potřeby.

Implementace Alarm Timers slouží k tomu, aby se služby jako budík či alarm, mohly vzbudit a uvést zařízení z režimu spánku, je-li to potřeba [4].

3.2.2 Knihovny

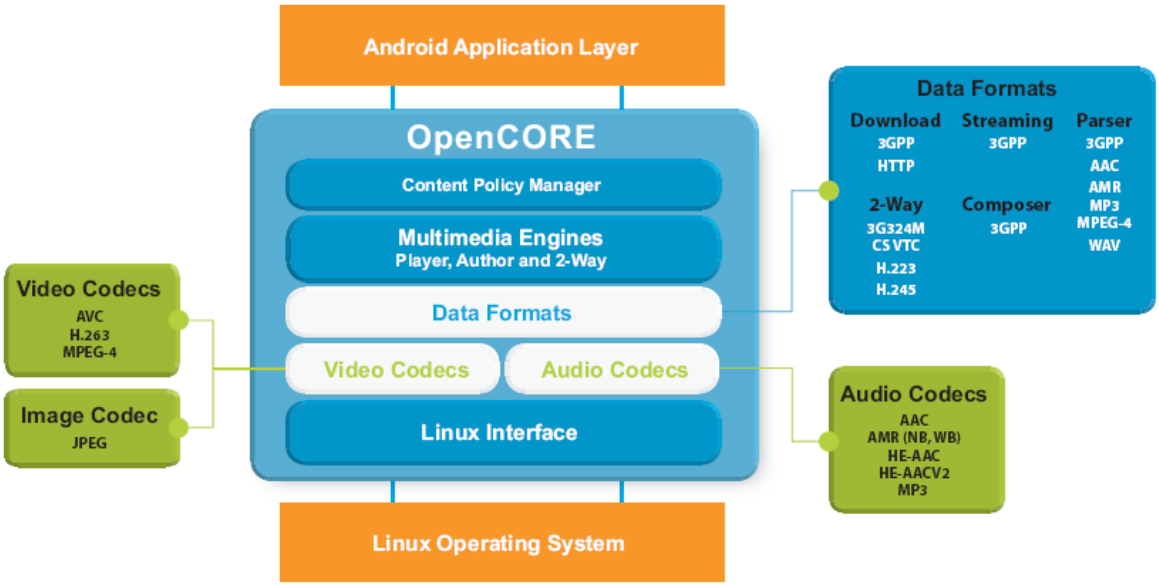
Druhá vrstva obsahuje knihovny napsané v jazycích C a C++. Tím umožnuje mobilním zařízením pracovat s různými typy dat. Tyto knihovny jsou využívány z více částí a komponentů systému. Knihovny poskytují funkce a rozšíření vlastností operačního systému Android, které jsou pro jeho běh nevyhnutelné. Tyto knihovny jsou většinou obsluhované z Java rozhraní z vyšších vrstev systému.

Obr. 8 – Využití knihoven a Android Runtime[4]

V této vrstvě nalezneme knihovny, jako jsou LibWebCore, SGL, FreeType, SQLite, Media Framework, Media Librares, WebKit, Surface Manager atd. Velmi důležitá je systémová C knihovna pod názvem Bionic LibC, která je optimalizována pro mobilní zařízení. Z důvodu dosáhnutí menší kapacitní velikosti, nenabízí možnosti klasické C knihovny, ale obsahuje jen části, které jsou nutné pro potřeby operačního systému Android. Knihovna zabírá přibližně 200 kB a není kompatibilní s ostatními klasickými Linuxovými knihovnami. Byla optimalizována pro použití s ARM architekturou.

Na podporu prohlížení webových stránek slouží knihovna LibWebCore, která je založená na WebKit rámci. Nabízí i možnost plného vykreslení stránek.

Pro multimédia využíváme subsystém Media Framework založený na PacketVideo OpenCore, který poskytuje většinou běžně používané kodeky pro přehrávání audia či videa. Mezi podporovanými kodeky jsou H.264, H.263, MPEG-4, MP3, MIDI, Ogg Vorbis, AMR, AAC, AACv1, AACv2. Dále také podporuje funkci media stream a funkci video hovorů.



Obr. 9 – Media Framework knihovna[13]

Grafický systém Surface Flinger je náhradou klasického Linuxového grafického systému. Tento systém zabezpečuje konečnou kompozici grafického výstupu více aplikací do souvislého toku dat, který směřuje do grafické vyrovnávací paměti, ze které probíhá vykreslování obrázků.

Další oblastí, která bude zmiňována, jsou zvukové výstupy. Tuto část pokrývá Audio Flinger, který přijímá audio výstup ze všech aplikací a zabezpečuje slučování nebo přesměrování zvuku na požadovaný výstup. Poskytuje jednotné rozhraní aplikačního rámce, čím mu umožňuje využit sadu aplikačních rozhraní, které zabezpečují přehrávání a záznam zvuku.

Pro ukládání strukturovaných dat lze využít SQLite. Je to v současnosti pravděpodobně nejrozšířenější neserverový databázový systém, hlavně díky svým nízkým nárokům [4]. V systému Android je implementovaný nástroj SQLite3 Database Tool, který umožňuje jednoduchou práci s obsahem databází a poskytuje podporu SQL příkazem.

3.2.3 Android runtime

Třetí vrstva Android runtime, obsahuje sadu základních runtime knihoven poskytující většinu funkcionalit v jazyce Java. V této vrstvě se nachází základní knihovny Java, virtuální stroj Dalvik nebo nový ART (Android RunTime). Dalvik Virtual Machine vyvíjený společností Google byla vytvořena jako náhrada za Java Virtual Machine, u které je z důvodu licenčních podmínek zakázáno volné šíření. Jedná se o JIT (Just-In-Time) kompilátor interpretující bytekód, jež nejdříve musí přeložit do strojového kódu, a pak spustit. DVM byl vytvořen pro lepší optimalizaci mobilního zařízení. Tento kompilátor nyní nahrazuje nejnovější Android RunTime (ART), který se implementuje v operačních systémech Android od verze 5.0 (API 21).

ART jako runtime provádí formát Dalvik spustitelný soubor a specifikaci DEX bytového kódu určený speciálně pro Android, který je optimalizovaný pro minimální nároky na paměť. Dalvik a ART jsou kompatibilní se systémem runtimes Dex bytecode, takže aplikace vyvinuté pro Dalvik by měly fungovat při běhu s ART [5]. Nicméně některé techniky, které fungují na Dalvik nefungují na ART.

Jednou z novinek v ART je kompilátor AOT (Ahead-Of-Time). Pomocí tohoto systému je předáván bytekód na strojový kód do zařízení ve chvíli, kdy je aplikace instalována. Výhodou je rychlost spuštění aplikací, menší náročnost, tudíž větší výdrž baterie, lepší multitasking a celkově plynulejší prostředí. Na druhou stranu probíhá instalace delší dobu, právě kvůli kompilování a je kladena větší náročnost na paměť.

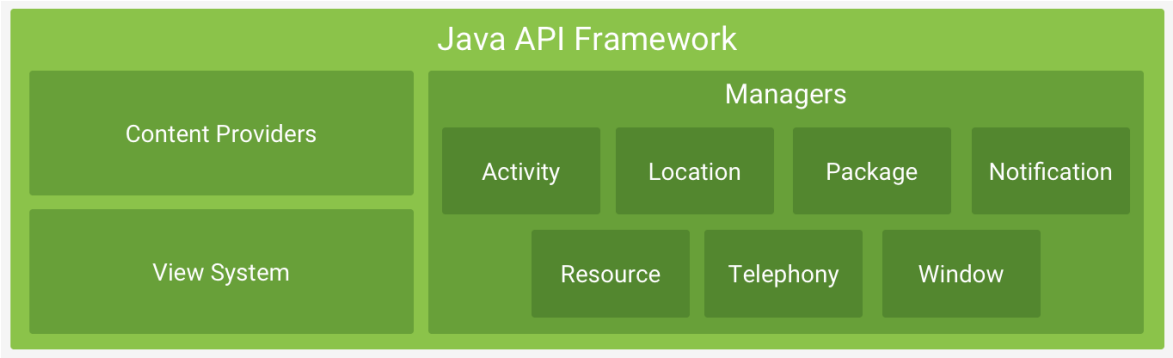
Další součástí ART musí být automatická správa paměti, což zabezpečuje Garbage collection (GC). Ovšem tento speciální algoritmus může zhoršovat aplikační výkon, způsobit sekání obrazovky, špatnou reakci UI a jiné problémy. ART zlepšuje GC několika způsoby [5]:

- Jedna GC pauza na místo dvou.
- Paralelizování procesů během trvání GC pauzy.
- Kolektor s nižším celkovým časem GC pro zvláštní případ vyčištění nedávno přidělených krátko trvajících objektů.

- Vylepšená GC ergonomie, tvoření příhodnějšího souběžného GC, což tvoří GC_FOR_ALLOC události extrémně vzácnými v klasických případech užití.
- Kompaktní GC pro snížení využití paměti na pozadí a fragmentaci.

Dále ART nabízí celou řadu funkcí, které zlepšují vývoj aplikací a ladění.

3.2.4 Aplikační rámec



Obr. 10 – Aplikační rámec Android[4]

Aplikační rámec, anglicky Android Framework je sada rozhraní API, která umožňují vývojářům rychle a snadno psát aplikace pro Android zařízení. Skládá se z nástrojů pro vytváření uživatelského rozhraní, jako jsou tlačítka, textové pole, prvky pro zobrazování obrázků a jiné. Obsahuje systémové nástroje, jako jsou například záměry definující vstupy a výstupy aktivit, ovládací prvky telefonu, přehrávače multimédií, atd.

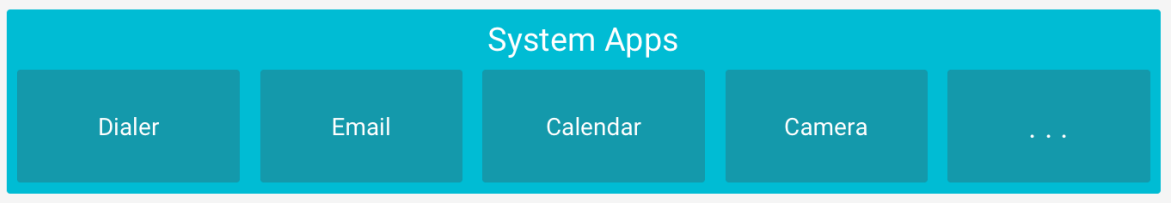
Aplikace jsou v podstatě postavené na čtyřech základních pilířích realizovaných jako třídy Activity, Services, Broadcast receivers, Content providers.

První pilíř jsou aktivity, se kterými uživatel pracuje přes grafické rozhraní (GUI). Potom existují služby, které na rozdíl od aktivit běží na pozadí, například přehrávání hudby nebo poskytují nějakou funkci dalším aplikacím. Důležité informace pro vaši aplikaci jsou získávány třídou Broadcast Receivers. Pomocí poskytovatelů obsahu (Content providers) mohou aplikace přistupovat k údajům ostatních aplikací.

V rámci oblastí Managers, které lze při vývoji aplikace využívat jsou například:

- Activity – stará se o správný běh aplikace podle životního cyklu. Životní cyklus je popisován v kapitole 3.3.1.
- Notification – slouží pro správu upozorňování na příchozí událost.
- Location – pro získávání údajů o poloze pomocí GPS modulů či internetové sítě.
- Package – správce balíčků, která obsahuje seznam nainstalovaných aplikací.
- Window – správa oken.

3.2.5 Aplikační vrstva



Obr. 11 – Aplikační vrstva Android[4]

Aplikační vrstva je nejvyšší vrstvou architektury operačního systému Android. Na této vrstvě jsou aplikace, které využívá koncový uživatel. Aplikace si může uživatel instalovat nebo je má již předinstalované. Jedná se o kalendáře, kontakty, prohlížeč, kameru atd.

V Androidu jsou všechny aplikace rovnocenné. Aplikace třetích stran a nativní aplikace jsou napsány pomocí stejných nástrojů a spouštěny ve stejném prostředí [6]. Uživatelé mohou odstranit nebo nahradit jakoukoliv nativní aplikaci alternativou od vývojáře třetí strany.

3.3 Vývoj aplikací pro Android

V této kapitole jsou charakterizovány některé prvky využívané při vývoji Android aplikací. Aplikace se skládá z komponent (Activity), záměrů (Intents), Android manifestu a zdrojů. Základní třídou můžeme označit Activity, která slouží pro zobrazení obrazovky s uživatelským rozhraním s využíváním životního cyklu Activity. Pro propojení jednotlivých komponentů využíváme záměry neboli Intents. Services běží na pozadí aplikace a nepotřebuje uživatelské rozhraní. Velmi důležitou částí aplikace je Android manifest, který definuje jednotlivé komponenty, oprávnění aplikace a nastavení konfigurace. Toto je základní přehled prvků, se kterými se nejčastěji při vývoji setkáváme.

3.3.1 Activity

Jak již bylo naznačeno, Activity je důležitá část programu bez níž by se aplikace nemohla spustit. Slouží k tomu, aby se všechna data, jež získá od nižších vrstev, správně zobrazila uživateli. Při používání slova s velkým písmenem Activity je myšleno třídy, která obsahuje podtřídy nebo konkrétní objekty. S použitím malého písmena aktivity, mluvím o obecném stavebním bloku návrhového vzoru. Aplikace může být tvořena několika aktivitami, ale vždy může být spuštěna jen jedna. Hlavní aktivita, která je uživatelem spouštěna jako první se uvádí v AndroidManifestu syntaxí Launcher v sekci intent-filter. Po vykonání tohoto filtru, lze spouštět i ostatní aktivity.

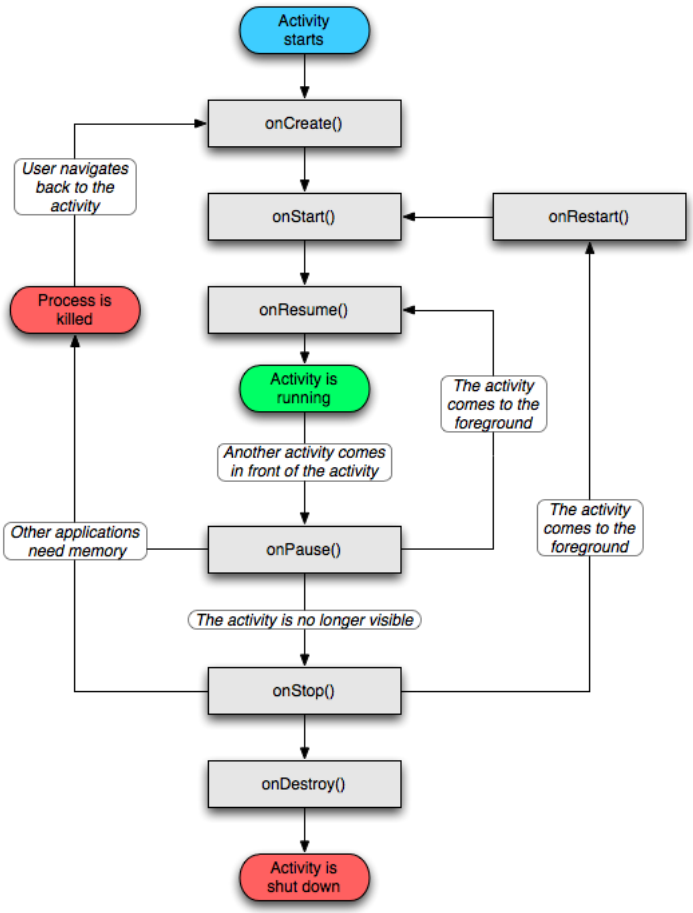
Vždy, když dojde ke spuštění nové aktivity, je předchozí aktivita pozastavena, ale systém ji zachovává v zásobníku. Zásobník disponuje řídícím mechanismem typu LIFO (last in, first out – poslední dovnitř, první ven). Jedná se tedy o klasickou frontu. Pokud uživatel stiskne v rámci obrazovky tlačítko *Zpět,* dojde k vytažení a zobrazení předchozí aktivity ze zásobníku [7]. OS Android vyznává filozofii, že u většiny mobilních zařízení jsou zdroje (pamět, baterie atd.) omezeny a stanovuje mechanismy pro zachování těchto zdrojů. Tyto mechanismy jsou patrné v životním cyklu aktivity.

Pomocí této komponenty můžeme provádět různé akce jako je psaní SMS, vytáčet telefonní číslo, spravovat kalendář a mnoho dalších úkonů, se kterými se uživatel denně stýká. Aktivity nejsou zodpovědné jen za globální funkčnost zařízení, ale jsou obsaženy

v jednotlivých aplikacích, které umožňují v rámci aplikace provádět určité změny. Vezmeme-li si na například kontakty, tak zde můžeme považovat za aktivitu přidání uživatele do seznamu, vymazání uživatele nebo třeba doplňování informací atp.

Životní cyklus Activity

Základní a zároveň velmi důležitou metodou životního cyklu je onCreate(), která se spouští vždy, když chceme do aktivity vstoupit. Aktivita se do popředí dostává voláním metody onResume (pokračování běhu aktivity) a skončí voláním metody onPause (pozastavení aktivity). Během této doby se aktivita zobrazí nad všemi ostatními aktivitami a je připravena komunikovat s uživatelem. Aktivita může pokračovat v běhu, či se pozastavit v cyklu několikrát za sebou podle interakce s uživatelem.



Obr. 12 – Životní cyklus Activity[14]

onCreate()

Spuštěním aktivity se Android postará o základní činnost jako vytvoření objektu a spuštění procesu. Poté zavolá metodu onCreate() a v ní nadefinuje vše potřebné pro první spuštění aktivity [7]. Určuje například jaké grafické rozhraní se má zobrazit, nebo jak se bude zobrazovat, jestli se mají vytvořit globální proměnné nebo zda budeme globálně přistupovat k objektu TextView atd.

Při vytvoření projektu nám vývojářské prostření vygeneruje následující kód:

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

Activity v metodě onCreate() nastaví layout pro svůj vzhled (GUI) řádkem:

```
setContentView(R.layout.activity_main);
```

Uvedený kód odkazuje na layout activity_main, který je v projektu ve složce res/layout. R. poukazuje na automaticky vygenerovaný kód R.java, kde jsou uloženy zdroje.

onStart()

Volá se po spuštění onCreate() nebo může být aktivována po svém skrytí (příchozí SMS, systémový dialog například o nabití baterie nebo jiný dialog jiné aktivity). Poté následuje onResume, pokud činnost spadá do popředí nebo onStop, má-li zůstat skryté.

onResume()

Metoda onResume() se volá těsně předtím, než se aktivita přesune do popředí - po prvním spuštění, restartu ze zastavení nebo po uvolnění dialogu (například dialogu

příchozího hovoru). Tato metoda je skvělým místem pro aktualizaci uživatelského rozhraní na základě událostí, ke kterým mohlo dojít od posledního zobrazení uživatelského rozhraní aktivity uživateli [10]. Vyžaduje-li například od nějaké služby změnu nějakých informací (například nových záznamů z informačního kanálu), představuje metoda onResume() výbornou příležitost k obnovení aktuálního zobrazení a – pokud je to relevantní – k zahájení aktualizace zobrazení ve vlákně na pozadí (například prostřednictvím objektu typu Handler).

onPause()

Výsledkem přesunutí aktivity na pozadí, obvykle kvůli aktivaci jiné aktivity se využívá volání metody onPause(). Zde byste měli zrušit vše, co jste provedli v metodě onResume() – například zastavit provádění vláken na pozadí, uvolnit jakékoliv prostředky s exkluzivním přístupem, které jste si vyžádali (například fotoaparát), apod.

Když se zavolá metoda onPause(), systém Android si vyhradí právo kdykoliv násilně ukončit proces aktivity [10]. Proto byste se po jejím zavolání neměli spoléhat na to, že vás systém upozorní na výskyt jakýchkoliv dalších událostí.

onStop()

Tato metoda je volána má-li se aktivita zastavit. Není viditelná pro uživatele a to z důvodu pokračování komunikace jiné aktivity, jež se kryje s aktivitou původní.

onDestroy()

Na druhém konci životního cyklu může být při ukončování aktivity zavolána metoda onDestroy() – ať už proto, že aktivita zavolala metodu finnish(), která aktivitu „ukončí“ nebo proto, že systém potřebuje více paměti RAM a aktivitu ukončí násilně. K volání této metody při násilném ukončení aktivity však nemusí dojít, pokud je uvolnění paměti urgentní (například při detekci příchozího hovoru) a aktivita přesto bude ukončena [10]. Proto se tato metoda nejlépe hodí k čistému uvolnění prostředků, které jste získali v metodě onCreate() (pokud jste to udělali).

onRestart()

Z obrázku životního cyklu vyplývá, že pokud byla zavolána metoda onStop a aktivita se restartuje, volá se onRestart, která se provede před onStart. Tím se dostáváme na začátek cyklu.

Při programování nemusíme vždy využívat všechny zmiňované metody. Záleží na programátorovi, které metody jsou pro aplikaci nezbytné. Podmínkou je použití metody onCreate(), která aktivitu vytváří.

3.3.2 Services

Služby realizují déle trvající operace a operace na pozadí. Také umožňují spolupráci se vzdálenými procesy. Na rozdíl od aktivit běží služby na pozadí a nepotřebují uživatelské rozhraní [3]. Služby umožňují provádět operace asynchronně paralelně s hlavním vláknem, jejichž realizace trvá déle. Také umožňují požádat různé procesy o provedení operace a sdílení údajů.

Services může mít tři různé typy:

- Scheduled
- Started
- Bound

Scheduled

Android 5.0 Lollipop (API 21) verze přináší plánovač úloh API přes třídu JobScheduler. Toto rozhraní umožňuje dávkovat úlohy, když má zařízení k dispozici více zdrojů [8]. Obecně tento API může být použit k plánování všeho, co není časově rozhodující pro uživatele.

Jestliže nastane opakující se úkol ve vaší aplikaci pro Android, potřebujete vzít v úvahu, že aktivity a služby mohou být ukončeny operačním systémem, aby se uvolnily zdroje. Proto nelze spoléhat na standardní Java plánovač TimerTask.

Started

Service formou "*started*" označujeme v případě, je-li spouštěn komponentou aplikace voláním startService(). Po startu může běžet na pozadí neurčitou dobu i v případě, že je komponenta zničena. Obvykle spuštěný Service vykonává jednu samostatnou operaci a nevrací výsledek volajícím [9]. Například by mohl stáhnout nebo nahrát soubor přes síť. Potom co je operace dokončena by se měl Service zastavit.

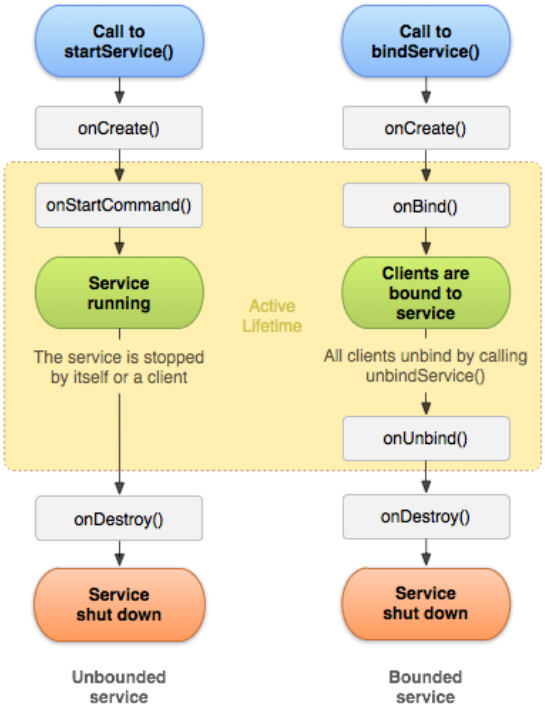
Bound

Service je "*bound*", když komponenta aplikace se k němu váže voláním bindService(). Vázaný Service poskytuje rozhraní klient-server, který umožňuje komponentům interakci se Service, což znamená odesílat požadavky, získávat výsledky a to dokonce i přes procesy s interprocesní komunikací (IPC). Vázaný Service běží pouze tak dlouho, dokud je k němu vázán další komponent aplikace [9]. K Service se může vázat více komponentů najednou, ale když se všechny komponenty rozvážou, je Service zničen.

Obecně jsou tyto formy popisovány samostatně, ale Service může fungovat dvěma směry, kdy je ve stavu "*started*" běžící po dobu neurčitou a současně umožňovat vazby. Je to jen otázka zda implementujeme současně pár metod zpětného volání onStartCommand() pro umožnění komponentům, aby jej mohly spouštět a onBind pro umožnění vazby. Bez ohledu na to, zda je aplikace ve stavu started, bound nebo obojí, Service může užívat jakékoliv aplikační komponenty, stejným způsobem, jako každá komponenta může využít jednotlivé Activity tak, že spustí Intent. Nicméně, můžete deklarovat Service jako soukromý v souboru manifest a blokovat přístup z jiných aplikací.

Životní cyklus Services

Stejně jako Activity mají svůj životní cyklus také služby. Je důležité věnovat pozornost životnímu cyklu provozovaných služeb, jelikož běží na pozadí bez vědomí uživatele. Životní cyklus služby a aktivity je velmi podobný.



Obr. 13 – Životní cyklus Services[9]

Každá služba reaguje na tyto metody [11]:

onCreate() – metoda je volána při prvním vytvoření služby. Pokud již běží, volání se neprovede.

onBind() – metoda je volána, pokud se chtějí některé komponenty k dané službě připojit a komunikovat s ní.

onStartCommand – metoda je volána, pokud některá z komponent (např. aktivita) požaduje spuštění služby voláním metody startService(). Jakmile se tato metoda provede, služba je spuštěna a běží na pozadí. Pro její zastavení je potřeba implementovat metody selfService() nebo stopService().

onUnbind – metoda je volána, pokud dojde k odpojení všech komponent, které se službou komunikovaly.

onDestroy – metoda je volána v případě, že se služba již nevyužívá a bude zrušena. Služba by v tomto případě měla uvolnit všechny využívané prostředky.

3.3.3 Broadcast receivers

Broadcast receivers je komponenta, která umožňuje reagovat na nějakou událost. Funguje na principu publish/subscribe. Události jsou zastoupeny objektu Intent(záměr). Vydavatelé vytvářejí záměry a následně je přes Broadcast směrují do vysílání [3]. Zachytávají je přijímače, které mají příslušné záměry objednané nebo registrované.

Události mohou vzniknout v rámci spuštěné aplikace nebo mohou být vyvolány systémem, kdy čeká například na přijetí SMS zprávy. Jakmile je SMS zpráva přijata, Android na tuto událost zareaguje a upozorní uživatele na přijetí nové zprávy. Následně si ji uživatel prostřednictvím vhodné aplikace může zobrazit.

3.3.4 Content providers

Content providers řídí společnou sadu dat aplikací, kterou lze ukládat do souborového systému v databázi SQLite, cloudu nebo na jiném trvalém uložišti, ke kterému mohou aplikace přistupovat. Umožňuje aplikacím a procesům mezi sebou sdílet nebo ukládat data. Bude-li chtít aplikace přistupovat ke sdíleným údajům, využije se třídy ContentResolver, což umožní aplikacím číst a zapisovat data, která jsou uložena v objektech ContentResolver. Jako příklad, kde je Content providers nejčastěji využíván, můžeme uvést správu informací uživatelských kontaktů.

3.3.5 SQLite

SQLite je malá, ale výkonná relační databáze, která pracuje bez serveru, nemusí se instalovat a nemá žádné konfigurační soubory. Na rozdíl od SQL serveru, Oraclu, MySQL a jiných řešení klient-server používá SQLite pouze klientskou část. Data jsou uložena v souboru, ke kterému přistupuje přímo klientská aplikace. Zjednodušeně řečeno samotná aplikace zastává funkci klienta i serveru současně. Jinak řečeno: server je součástí klienta. Ikdyž SQLite vyřizuje vždy pouze jeden požadavek, samotný výkon databáze je relativně velmi vysoký [3]. Čtení dat má přitom prioritu před zápisem. Proto se používá například pro CMS weby či jako připojená databáze pro desktopové nebo mobilní aplikace.

3.3.6 Android manifest

Android manifest je základní částí programu každé aplikace a nachází se v kořenovém souboru zdrojů zvaný AndroidManifest.xml. Podle koncovky je zřejmé, že se jedná o XML soubor a je v něm deklarován obsah naší aplikace. Zároveň je zde uvedeno, jakým způsobem se části aplikace propojí s operačním systémem. Určuje, například jaká část kódu se má spustit resp. jaká aktivita se provede prvotně atd. Pod odstavcem můžeme vidět, jak vypadá struktura kódu v souboru AndroidManifest.xml pro tuto diplomovou práci.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="diploma.eu.cz.karel.smarthome">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <application
            android:allowBackup="true"
            android:icon="@mipmap/smarthome"
            android:label="@string/app_name"
            android:supportsRtl="true"
            android:theme="@style/AppTheme">
        <activity
                android:name=".NetworkService"
                android:label="@string/network"
                android:theme="@style/AppTheme.NoActionBar"
                android:noHistory = "true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".WatchCamera"
            android:label="@string/camera"
            android:parentActivityName=".NetworkService"
            android:noHistory = "true"/>
        <activity
                android:name=".BluetoothService"
                android:label="@string/bluetooth"
                android:theme="@style/AppTheme.NoActionBar"
                android:noHistory = "true"/>
        <activity
            android:name=".ChartActivity"
            android:label="Graph temperature"
            android:parentActivityName=".NetworkService"
            android:noHistory = "true"/>
    </application>
</manifest>
```

Popis základních elementů v Android manifest:

- **<manifest>** je kořenový prvek souboru. Musí obsahovat <application> element a určit xmlns:android, který definuje jmenný prostor Android a název package pro aplikaci.
- **<uses-permission>** požádá o povolení, která aplikace potřebuje, aby mohla správně fungovat. Například v kódu výše je uvedeno *android.permission.BLUETOOTH,* což povoluje aplikaci připojit se k zařízení bluetooth.
- **<application>** tento element obsahuje dílčí prvky, které deklarují všechny aplikační komponenty a má atributy, které mohou mít na ně vliv.
- **<support-screen>** tento element umožní definovat požadavky aplikace na displej uživatelova telefonu.
- **<uses-configuration>** slouží k definování některých potřebných hardwarových vlastností telefonu, jako je typ hardwarové klávesnice, dotykový displej či trackpoint. Těchto elementů můžete mít v manifestu více. Telefon vyhovuje požadavkům aplikace, pokud splňuje všechny požadavky uvedené (nejméně) na jednom z nich.
- **<uses-feature>** můžete definovat, které vlastnosti telefonu aplikace používá či přímo vyžaduje. Například pokud nemá smysl, aby byla aplikace nainstalována na telefonu bez bluetooth.

```
<uses-feature android:name="android.hardware.bluetooth" android:required="true" />
```

4 Aplikace Android pro chytrou domácnost

V praktické části diplomové práce, byla vyvíjena aplikace pro chytrou domácnost. V kapitole je analyzován samotný proces tvorby aplikace od návrhu až po implementaci. Na závěr jsou vyhodnocovány výsledky a poznatky práce.

4.1 Charakteristika problematiky

Chytrá domácnost je velmi oblíbenou oblastí pro kutily, kteří si chtějí vylepšit svůj obytný prostor. Takové vylepšení může spočívat ve vzdálené regulaci světel, tepla, vypínání elektrických obvodů, zamykání, aktivování zabezpečovacích systémů, sledování kamer a tak podobně.

Je řada možností, jak si takovou domácnost udělat. Při tvorbě je nutné se zabývat nejen programem, ale i konstrukčními prvky. Těmito prvky jsou myšleny různé spínací obvody, senzory, snímače, kamery apod. Důležitou částí je také řídící jednotka, na kterou připojujeme konstrukční prvky, a se kterými dále pracujeme. V této práci bylo využito řídících prvků Raspberry Pi a Arduino.

Smyslem chytré domácnosti je vzdálené ovládání a tak je potřeba, abychom vytvořili klientskou část, v tomto případě Android aplikaci. Serverová část je tvořena řídícím prvkem. Vzdálená komunikaci mezi klientem a serverem je v této práci řešena prostřednictvím internetu nebo bluetooth.

K vyřešení ukládání dat, které server sbírá při jeho aktivitě, je potřeba k němu připojit databázový systém. Ten nám může zabezpečit například aktualizaci grafu na základě snímané teploty po určitou dobu. Klient se pak už jen dotazuje na tato data a server nám je vrací. Spínání konstrukčních prvků je řešeno binární logikou změny stavu z 0 na 1. Pro funkčnost tohoto procesu je potřeba, aby klient posílal data na server a ten je zpracovával. Poté provede změnu stavu napětí na patřičných pinech řídícího obvodu. Výsledkem může být například zapnutí či vypnutí světla. Jako součástí každé chytré domácnosti by mělo být také, sledování bezpečnostních kamer prostřednictvím aplikace. Zde je potřeba, aby kamera byla připojena k řídícímu prvku, na kterém běží software pro streamování videa a umožňuje připojení klienta. V práci se těmito součástmi dále zabýváme.

4.2 Návrh vlastní aplikace

Pro tvorbu aplikace bylo zvoleno vývojové prostředí Android Studio vyvíjeno společností Google, které bylo vytvořeno speciálně pro rozvoj operačního systému Android. Instalace je velice jednoduchá, jelikož vše se instaluje ze staženého balíčku Studia a není již nutné nic víc doplňovat. Protože pracujeme s programovacím jazykem Java, je potřeba mít v počítači nainstalováno JDK, které připojíme při samotné instalaci Studia.

Při tvorbě projektu nám Android Studio automaticky vygeneruje adresáře manifest, java, res a mnoho dalších. Uvedeme zde jen ty adresáře a soubory, ke kterým se budeme v práci vracet:

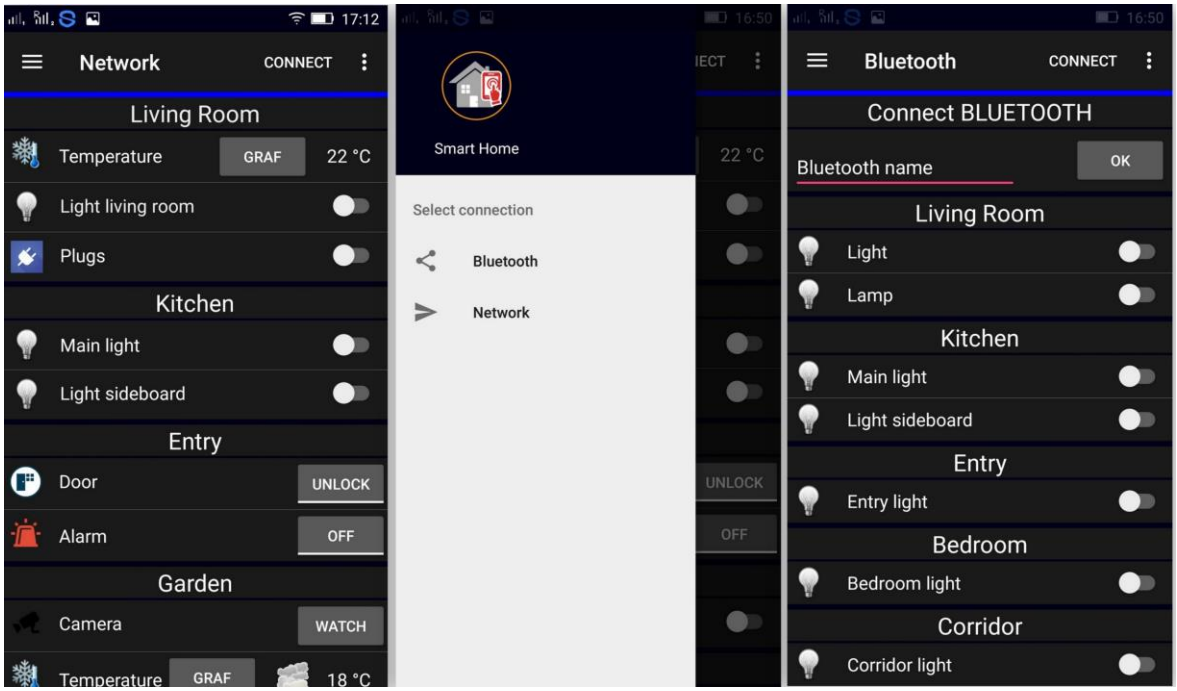
- Manifest – adresář pro soubor manifest.xml.
- Java – adresář pro zdrojové kódy Javy.
- Res – tento adresář se dělí dále na podadresáře.
    - Res/layout – pro uživatelské rozhraní.
    - Res/drawable – pro obrázky.
    - Res/menu – pro vytvoření menu.
    - Res/values – zde jsou proměnné ukládané do souborů string.xml, styles.sml, color.xml. První zmiňovaný soubor obsahuje řetězce. Druhý vytváří styl aplikace a color.xml barvy.

4.2.1 Tvorba layoutu

Rozvržení uživatelského rozhraní aplikace je tvořeno ve vývojovém prostředí v adresáři res/layout/ ve formátu XML. Definice rozložení ve formátu XML je specifikace vzájemných vztahů widgetů a jejich vztahů k obsahujícímu kontejneru ve formátu XML. Každý soubor XML obsahuje strom elementů, které definují rozložení widgetů a kontejnerů, které tvoří jednu instanci třídy View. Atributy jednotlivých elementů XML jsou vlastnosti popisující vzhled widgetu nebo chování kontejneru. Sada SDK systému Android se dodává s nástrojem (aapt), který tato rozložení využívá [16]. Tento nástroj by měl být automaticky spuštěn v rámci řetězu nástrojů systému Android a vygeneruje

v adresáři gen vašeho projektu soubor prostředku R.java. Díky tomu můžeme přistupovat k rozvržením a widgetům definovaným v těchto rozvrženích přímo z kódu jazyka Java.

V práci je aplikace vytvářena pro dva různé styly komunikace se serverem. V prvním případě je to komunikace přes internetovou síť a ve druhém přes bluetooth. Z tohoto důvodu bylo potřeba vytvořit dvě různá specifická rozhraní, která můžeme vidět na Obr. 14. Abychom mohli mezi těmito uživatelskými rozhraními přecházet, bylo vytvořeno menu, které je zabudované v hlavičce aplikace. Jednoduchým stiskem na tzv. „hamburger“ menu se otevírá posuvná nabídka, kde můžeme následně vybírat, jakým stylem budeme s daným serverem komunikovat. Dále je v aplikaci zabudované tlačítko „connect“ a schované „disconnect“ v boční časti hlavičky. Totéž je přidané v aktivitě bluetooth. Uživatelské rozhraní je dále tvořeno obsahem, kde jsou různé prvky pro regulaci. Tyto prvky jsou dále popisovány.



Obr. 14 – Uživatelské rozhraní aplikace

Uživatelské rozhraní se skládá z následujících vytvořených layoutů (umístění res/layout):

- activity_bluetooth.xml – soubor pro tvorbu vysunovací nabídky aktivity bluetooth,
- activity_network.xml – soubor pro tvorbu vysunovací nabídky aktivity network,
- app_bar_bluetooth.xml – soubor pro tvorbu toolbaru aktivity bluetooth,

- app_bar_network.xml – soubor pro tvorbu toolbaru aktivity network,
- content_network.xml – soubor tvořící obsah aktivity network,
- content_bluetooth.xml – soubor tvořící obsah aktivity bluetooth,
- content_watch.xml – soubor tvořící obsah pro stream camery,
- content_chart.xml – soubor tvořící obsah grafu,
- nav_header.xml – soubor pro tvorbu hlavičky ve vysouvacím menu.

Pro zabudování tlačítek connect, disconnect a vytvoření obsahu posuvného menu, bylo nutné vytvořit následující soubory (umístění res/menu):

- activity_drawer.xml – obsah posuvného menu,
- menu_blue.xml – tlačítka connect a disconnect pro aktivitu bluetooth,
- menu_net.xml – tlačítka connect a diconnect pro aktivitu network,
- menu_watch.xml – pro tvorbu tlačítka zpět v aktivitě watch i graph.

Níže pod odstavcem, můžeme vidět část kódu tvorby widgetů v položce Kitchen. Ve vývojovém prostředí Android Studio je možnost využití tzv. design nástroje pro rozmístění jednotlivých widgetů v grafickém režimu. Toto nám ulehčí práci, kdy není zapotřebí psát text kódu a je tak vytvářen automaticky. Ovšem atributy jako jsou identifikátory widgetů, různé názvy, barvu písma a tak podobně je nutné samostatně doplnit. Identifikátory jsou velice důležité pro následné přistupování v Java zdrojovém kódu, ale ne vždy je potřeba k danému widgetu přistupovat.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="@string/kuchyn"
    android:id="@+id/kuchyn" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:background="@color/nav_back" />
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">
    <ImageView
```

```
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:scaleType="fitCenter"
            android:src="@drawable/light_off"
            android:id="@+id/imageView5" />
        <Switch
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/switch1"
            android:layout_centerVertical="true"
            android:layout_alignParentEnd="true"
            android:layout_toEndOf="@+id/imageView5"
            android:checked="false"
            android:text="@string/main_light"
            android:textSize="17dp"
            android:layout_marginLeft="@dimen/fab_margin" />
    </RelativeLayout>
```

V kódu jsou uvedeny widgety:

- TextView – pro tvorbu textového pole.
- ImageView – pro vložení a manipulaci s obrázkem.
- Switch – pro tvorbu přepínacího tlačítka.

Každý widget je tvořen atributy. Pro příklad si uvedeme některé z nich:

- android:text="@string/kuchyn" představuje text, kde jehož obsahem je hodnota uložená v souboru string.xml pod názvem kuchyn.
- android:id="@+id/switch1" představuje identifikátor, ke kterému můžeme dále přistupovat ve zdrojovém kódu Javy nebo může vypomáhat při rozmisťování widgetů.
- android:src="@drawable/light_off" představuje obrázek, který máme uložený v adresáři drawable.
- android:layout_width a android:layout_height pro šířku a výšku widgetu může obsahovat kromě zadání přesné pevné velikosti také dvě vlastnosti a to buď wrap_content nebo match_parent. První z nich znamená, že například TextView bude zabírat v pohledu tolik místa, kolik je potřeba k vypsání textu a pokud chceme, aby text zabíral celý řádek, změníme hodnotu atributu na match_parrent (dostupný od verze 8) nebo fill_parent.

Android Studio standardně používá element <RelativeLayout> pro možnost rozmisťování komponent pomocí grafického režimu. Díky tomu můžeme ručně přemisťovat widget bez nutnosti psaní kódu. Dále Android nabízí element <LinearLayout>, který umožňuje umisťovat prvky postupně za sebou. V práci využíváme tohoto elementu s atributem android:orientation s hodnotou vertical, čímž zařídíme, že se komponenty budou skládat pod sebe. Pro zobrazení pohledu většího než jsou rozměry displeje, používáme element ScrollView, který umožňuje aplikaci tzv. skrolovat.

Obr. 15 – Aplikace v horizontální poloze

Při programování aplikace je také velice důležité dbát na horizontální položení aplikace. Lze vidět na Obr. 15. Při otočení obrazovky se může stát, že některé součásti nebudou uspořádány podle našich očekávání. Může se překrývat text, rozložení tlačítek může být proházené, jakožto i obrázky atd. Aby k tomuto nedocházelo, bylo nutné se co nejvíce vyvarovat používání pevného nastavování velikostí v jednotkách délek a správně používat elementy.

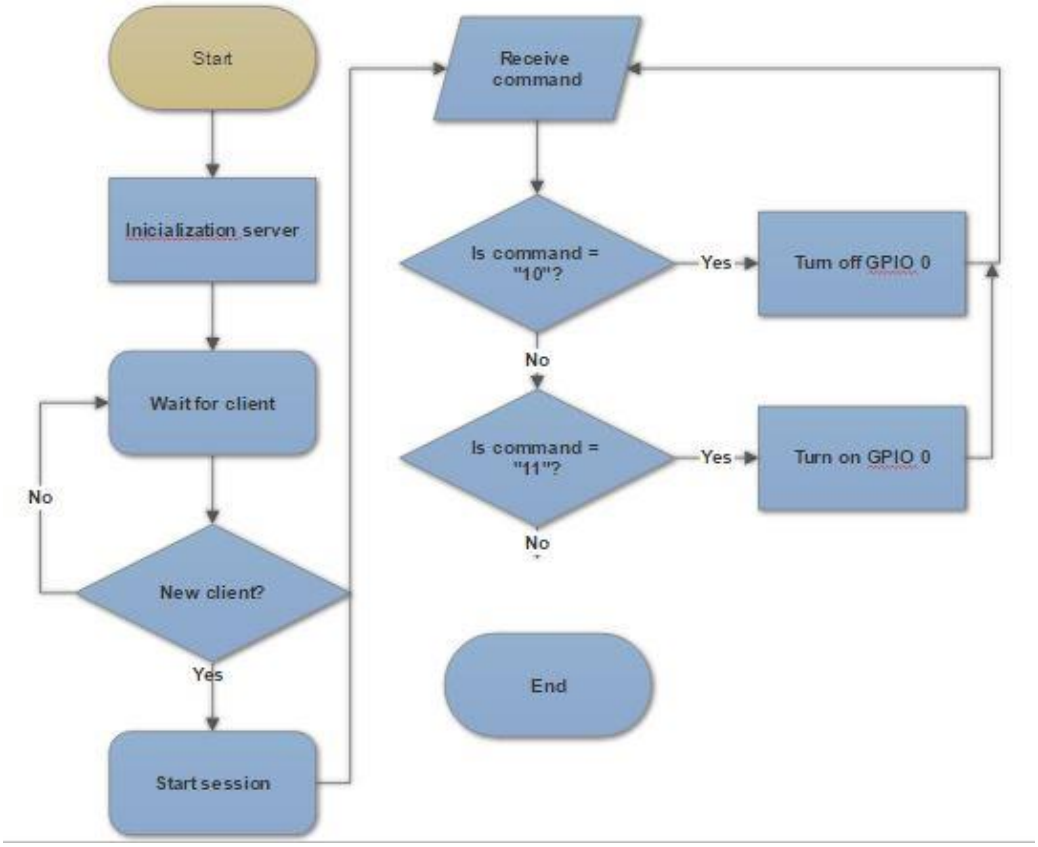
4.3 Server chytré domácnosti

V problematice chytré domácnosti je nutné se zabývat také serverovou částí systému. Tato část může být tvořena jakýmkoliv řídícím obvodem, který je schopný komunikovat s konstrukčními prvky. V práci bylo pro ověření funkčnosti využíváno Raspberry Pi a Arduino.

4.3.1 Raspberry Pi

Raspberry PI je jednočipový počítač velikosti zhruba kreditní karty, do které lze připojit monitor s klávesnicí pro jeho obsluhu. Je-li připojen do internetové sítě, je možnost s Raspberry Pi komunikovat také vzdáleně přes SSH nebo například přes vzdálenou plochu. Byl vyvinut britskou nadací Raspberry Pi Foundantion s cílem podpořit výuku informatiky ve školách a seznámit studenty s možnostmi řídit různá zařízení, která připojujeme k jeho tzv. GPIO pinům. Můžeme ho využívat pro různé projekty z oblasti elektroniky, pro domácí kino a mnoho dalšího. Primárním operačním systémem je Raspbian, ale lze využívat i jiných OS.

V práci je tento počítač používán pro síťovou komunikaci, která může být veřejná nebo jen jako lokální. Budeme-li chtít chytrou domácnost regulovat ze vzdálenějších míst než je dosah naší domácí wifi sítě, je zapotřebí, aby byl server dostupný na veřejné IP adrese. V rámci lokální sítě nám stačí jeho privátní IP adresa. Na Obr. 16 můžeme vidět vývojový diagram průběhu komunikace na serveru od jeho připojení s klientem až po samotnou regulaci spínání pinů.



Obr. 16 – Komunikace server

Typické chování serveru je, že čeká na připojení klienta. Jakmile se připojí, server ho obslouží. Struktura programu je taková, že si vytvoříme instanci třídy ServerSocket a metodou accept() naslouchám na příslušném portu. Jakmile se klient připojí, metoda vrátí instanci třídy Socket. Z ní získáme vstupně výstupní stream, přes který dále komunikujeme.
Program je napsán v programovacím jazyce Java a bylo nutné využít Java knihovny Pi4j, která umožnuje komunikaci s GPIO piny. Tuto knihovnu je nutné samostatně nainstalovat na Raspberry Pi. Pro práci s piny je nutné je nejdříve tzv. otevřít, což děláme v našem vytvářeném zdrojovém kódu. Jednou z důležitých funkcí této knihovny jsou metody HIGH a LOW. Použijeme-li HIGH, bude to znamenat, že při provedení nějaké operace se přivede napětí na příslušný pin. Při použití metody LOW se naopak pin od napětí odpojí.

Aby server mohl komunikovat i s dalšími klienty, je nutné zajistit čekání na připojení během obsluhy klienta. Z tohoto důvodu je potřeba využít více vláknového procesu a vytvářet je samostatně pro každého nového klienta. Původní vlákno opět zavolá metodu accept() a čeká na další připojení. Viz zdrojový kód zde.

```
public void runServer() {
        try {
            server = new ServerSocket(12345);
            System.out.println("Server is running on port 12345:");
            while (true) {
                new Controller(server.accept()).start();
                System.out.println("Hello client!!!");
            }
        } catch (IOException ioException) {
            ioException.printStackTrace();
        }
    }
private class Controller extends Thread {
        private Socket socket;
        private ObjectInputStream input;
        private ObjectOutputStream output;
        private String in;
        public Controller(Socket socket) {
            this.socket = socket;
            System.out.println("New client at " +
socket.getRemoteSocketAddress());
        }
        @Override
```

```
        public void run() {
            try {
                getStreams();
                output.writeObject("Hello, Welcome to Raspberry PI");
                output.flush();
                while (!(in = (String)
input.readObject()).equals("close")) {
        //program body
}
⋮
}
```

4.3.2 Arduino

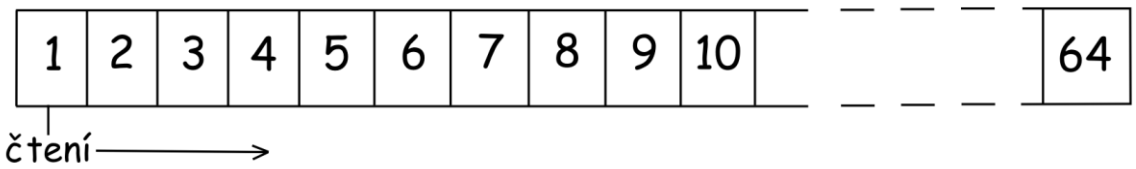
Arduino je open-source elektronická platforma založena na snadném použití hardwaru a softwaru. Mikroprocesor na desce Arduina, se programuje ve vlastním Arduino vývojovém prostření. Arduino deska je schopna číst vstupy a na základě toho provádí změnu na výstupu. Po připojení konstrukčních prvků může deska s příslušným softwarem například spínat relé, ovládat ústřední topení domu, detekovat pohyb a mnoho dalšího. Zařízení je především určeno pro umělce, designéry, fandy a zájemce o vytváření interaktivních objektů nebo prostředí. Arduino je flexibilní, snadno ovladatelný, levný a velmi dostupný pro různé typy součástek. Proto je také velmi oblíbeným nástrojem pro tvorbu chytré domácnosti.

S chytrou domácností můžeme komunikovat i prostřednictvím bluetooth modulu. Zde bylo zvoleno zařízení Arduino. Pro krátký dosah bluetooth je spíše využitelný na menší prostory. Tato metoda není velice vhodná pro sledování kamer nebo například pro detekci pohybu, protože uživatel bude chtít mít možnost kontrolovat stav kolem jeho domu, i když se v něm momentálně nenachází. Z tohoto důvodu se v práci zabýváme jen regulací světel a problematikou připojení na takovýto systém.

Pod textem můžeme vidět část zdrojového kódu v programovacím jazyce C pro serverovou část na Arduinu. Nejprve bylo nutné otevřít sériový port na pinech 0 a 1, které jsou na desce popsané jako Rx a Tx, viz první řádek kódu. Na tyto piny připojujeme bluetooth modul, v tomto případě modul HC-05, který disponuje sériovým výstupem. Výchozí přenosová rychlost sériové komunikace modulu je 9600 Bd. Následně otevřeme piny pro

možnost komunikace s konstrukčními prvky. Poté se dostáváme do smyčky, která se vykonává podle toho, co přichází na vstup Arduina.

Čtení dat na nejnižší úrovni probíhá tak, že se přijímané informace nezpracovávají hned, ale uchovávají v „zásobníku“ anglicky buffer. Ten dokáže uchovat až 64 bytů. Procesor zpracovává jeden byte po druhém, který po přečtení uvolní jeho místo a posune uchované byty dopředu.



Obr. 17 – Schéma zásobníku[23]

V této problematice regulace stavů na pinech, Arduinu odesíláme ASCII hodnotu znaku. Každý znak i čísla mají vlastní byte paměť. Funkce Bluetooth.available() vrátí počet bytů dostupných v zásobníku. Následně funkce Bluetooth.read() vezme první byte ze zásobníku a přečte ho. Zároveň dochází k vyjmutí bytu ze zásobníku, což se projeví snížením hodnoty Bluetooth.available() při dalším čtení [23]. Při odeslání čísla 1 z klientského zařízení do Arduina bude v zásobníku obsazen 1 byte a dále už není potřeba pokračovat ve čtení. Poté se na základě toho rozhoduje co s daným obvodem má stát. V případě přijetí znaku 1 Arduino přivede napětí na pin 2. Viz podmínka if.

```
SoftwareSerial Bluetooth =  SoftwareSerial(0, 1);
long m;
void setup() {
Bluetooth.begin(9600);
delay(500);
  pinMode(2, OUTPUT);//Light1 pin
  pinMode(3, OUTPUT);//Light2 pin
  pinMode(4, OUTPUT);//Light3 pin
  pinMode(5, OUTPUT);//AC pin
  pinMode(6, OUTPUT);//Door Lock
}
void loop() {
  if ( Bluetooth.available())
  {
    char znak = Bluetooth.read();
    if (znak == '1')
```

```
    {
      digitalWrite(2, HIGH);
      Bluetooth.print("OK ");
    }
⋮
}
```

Funkce Bluetooth.print slouží pro odeslání hodnoty z Arduina do klientského zařízení. V praxi to funguje tak, že se odesílá ASCII kód znaků, který se na klientském zařízení interpretuje. V ASCII tabulce má velké „O“ v desítkové soustavě hodnotu 79 a „K“ hodnotu 75. Při volání funkce Bluetooth.print("OK"); se tedy odesílá číslo 79 a 75 za sebou v tomto pořadí, které klientské zařízení přeloží. Tato funkce má také jeden nepovinný parametr, který udává typ odeslaných dat. Mohou to být desítkové soustavy DEC, osmičkové OCT, šestnáctkové HEX a dvojkové BIN. Zadáme-li Bluetooth.print("a", HEX); funkce vrátí číslo v šestnáctkové soustavě 61.

4.4 Klient chytré domácnosti

Každá chytrá domácnost musí mít vytvořenou i klientskou část, která umožní uživateli odesílat požadavky na server. Obecně nelze říci, jestli se jedná o stranu vysílače nebo přijímače, jelikož tuto funkci zastupuje jak server, tak klient současně při jejich komunikaci. Pro naše účely se bude tato část označovat za stranu vysílače.

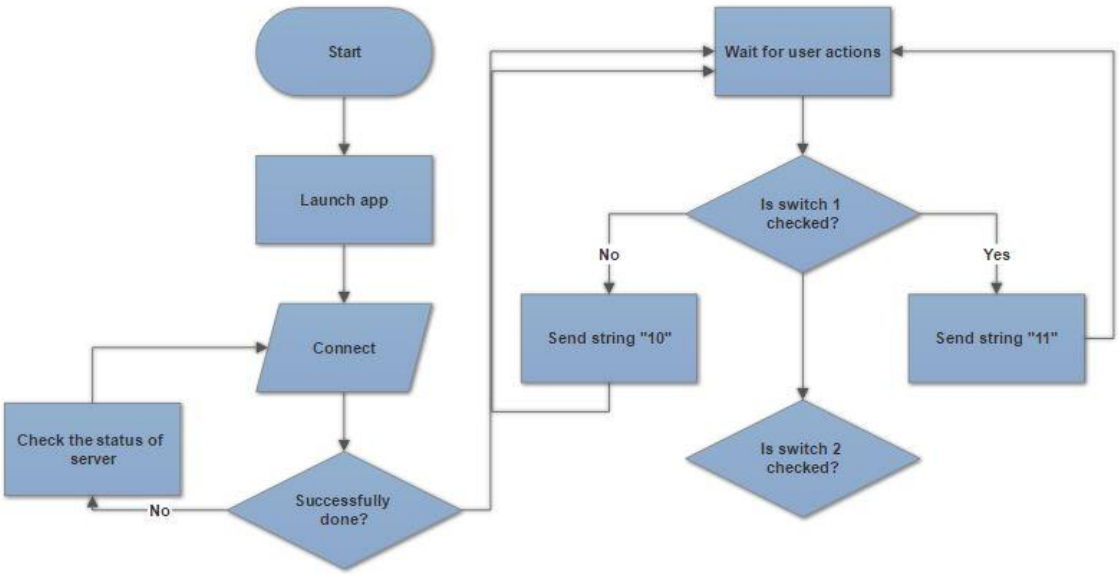
V práci je klient vytvářen jako aplikace pro operační systém Android. Při programování bylo vytvořeno několik tříd:

- NetworkService – hlavní třída pro regulaci domácnosti přes internet.
- BluetoothService – třída pro regulaci domácnosti přes bluetooth.
- Config – třída pro ukládání současných stavů.
- WatchCamera – třída pro stream videa.
- ChartActivity – třída pro graf.

4.4.1 Komunikace TCP/IP

Na Obr. 14 v kapitole 4.2.1 můžeme vidět uživatelské rozhraní v aktivitě network, která představuje regulaci vybraných prvků přes internet. Spojení se navazuje automaticky při spuštění Activity. Toto nám ulehčuje práci, kdy není potřeba opětovně zadávat IP adresu a port při spuštění aplikace. V případě, že je server nedostupný, klient čeká, kdy bude server opět spuštěn. Jakmile je server dostupný, lze provést připojení přes tlačítko connect. Proces připojení k serveru a regulace světelných obvodů je zřejmý z Obr. 16.

Po připojení, server čeká na uživatelský vstup. Server nejprve rozpozná, zda je obvod v sepnutém či vypnutém stavu a vrátí uživateli momentální aktuální stav konstrukčních prvků. Poté dá uživateli možnost odeslat v tomto případě hodnotu string, kterou na druhé straně přijímač zpracuje a vykoná daný úkon. Rozpoznávání aktuálního stavu je nutné, aby nedocházelo k neúmyslnému vypnutí či zapnutí světla při spuštění aplikace. Průběh komunikace klienta můžeme vidět na vývojovém diagramu z Obr. 18. Pro úplné znázornění procesu komunikace je nutné si spojit oba vývojové diagramy.



Obr. 18 – Komunikace klient

Průběh spojení je následující. Na serverové části běží program pro připojení klienta a po spuštění aplikace se okamžitě navazuje spojení. Jelikož probíhající spojení je dlouhotrvající akce, bylo využito asynchronního programování a tento proces přesunut na

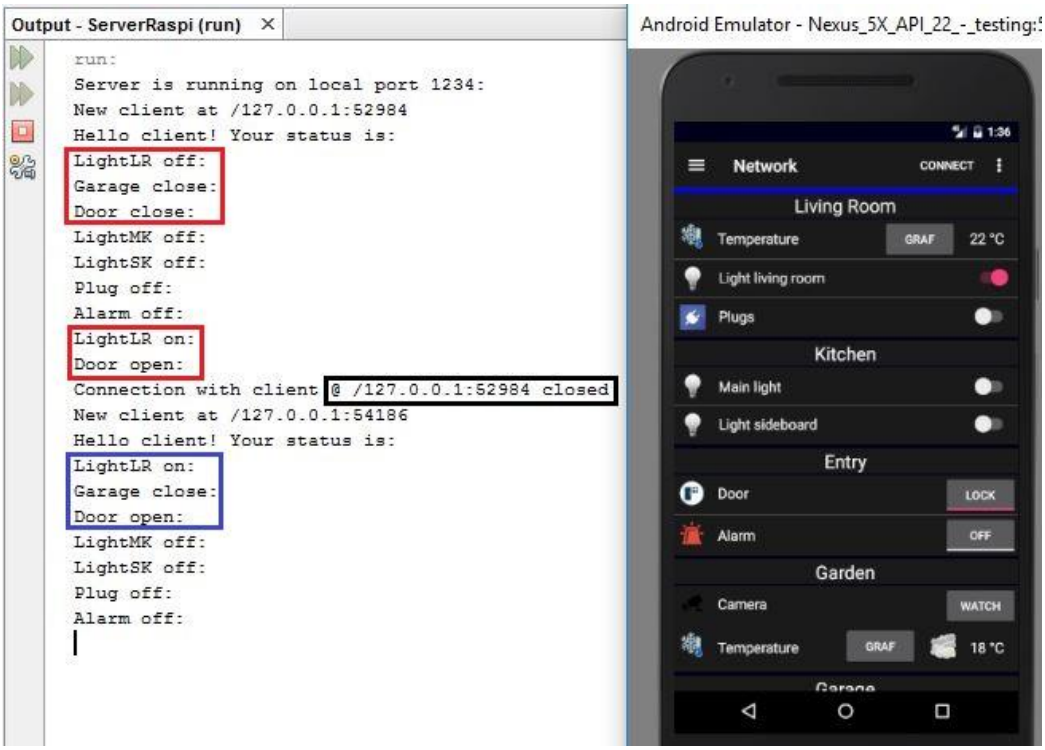
jiné vlákno. Děláme to proto, aby nedocházelo k zahlcení hlavního vlákna, které slouží k interakci s uživatelem. Pod odstavcem je zobrazený část zdrojového kódu, kde lze jednoduše měnit nastavení IP adresy a port řídící jednotky, na kterém je v síti dostupná. Spojení probíhá obdobně jako u serverové části. Na začátku vytváříme soket a vstupně výstupní stream, který se naváže na server.

```
private Connection() {
    IP = "10.0.2.2";
    Port = 1234;
}
@Override
protected String doInBackground(String... Command) {
    try {
        socket = new Socket(IP,Port);
        out = new ObjectOutputStream(socket.getOutputStream());
        out.flush();
        in = new ObjectInputStream(socket.getInputStream());
        checkSwitchStatus();
        Status = "Successfully connect";
}
⋮
}
```

Pro demonstraci byl v programu vytvořen textový výstupu o prováděné aktuální činnosti klienta. Viz Obr. 19. Jsme-li připojeni, vypíše se na výstup IP adresa a port, na kterém klient vysílá. Poté je server seznámen se současným stavem widgetů v aplikaci. Na základě této informace musí server vrátit stavy widgetů do správné polohy, aby nedocházelo k neúmyslnému vypínání obvodů.

V červeně označeném poli na obrázku můžeme vidět provedenou změnu stavu přepnutím světla obývacího pokoje a dveří do aktivního stavu. V tomto případě se na výstupu okamžitě provádí změna ze stavu off na on. Vystoupíme-li z aplikace tlačítkem „home“, komunikace mezi klientem a serverem se automaticky ukončí. Můžeme to vidět v černě označeném poli a znamená to, že se soket uzavírá. Jakmile se vrátíme zpět do aplikace, je opět navázáno spojení se serverem. V modře označeném poli vidíme aktualizovaný stav našich předchozích činů, které jsme se server učinili, před odchodem z aplikace. Výsledkem je to, že server je seznámen se současným stavem widgetů v aplikaci a musí

vrátit aktuální reálný stav konstrukčních prvků, aby nedocházelo k neúmyslnému vypínání elektronických obvodů při návratu zpět.



Obr. 19 – Výstup komunikace klient-server

4.4.2 Využití životního cyklu

V kapitole 3.3.1 je charakterizován životní cyklus aplikace. Podle něj můžeme popsat hierarchii aplikace chytré domácnosti. Na začátku je vždy metoda onCreate(), která spouští uživatelské rozhraní dané Activity. Dále jsou v ní umístěny všechny operace, které metoda vykonává při uživatelském vstupu. Nyní přecházíme k metodám onStart() a onResume, které zapříčiní okamžité připojení k serveru při návratu do aplikace. Je-li aktivita pozastavena, zavolá se metoda onPause() nebo onStop() a uzavřou socket komunikaci, aby nedocházelo k chybám při opětovném navázání spojení. Na Obr. 12 je charakterizováno, kdy se jaká metoda při pozastavení zavolá. Poté může být aktivita ukončena uživatelem nebo systémem metodou onDestroy(), která také ukončí spojení s daným serverem.

4.4.3 Stream

Každá chytrá domácnost by měla disponovat zabezpečením svého domu prostřednictvím kamerového systému. K tomu poslouží IP kamery, které je možno nainstalovat k zařízení Raspberry pi. Poté můžeme využít například VLC multimediálního přehrávače, který umožňuje zobrazit stream z Raspberry pi na klienstkém zařízení. VLC přehrávač je open-source software a tak ho lze jednoduše nainstalovat na Raspberry pi příkazem „sudo apt-get install vlc“. Potom vytvoříme soubor s příponou .sh a do něj vložíme kód, zobrazený pod tímto textem. Jakmile soubor spustíme příkazem ./nazevSouboru.sh bude stream zpřístupněn.

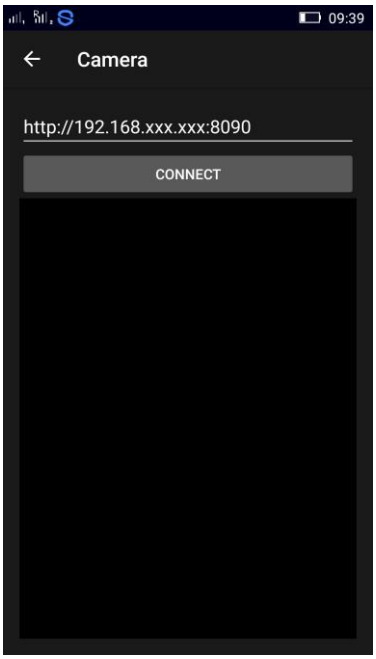
```
raspivid -o - -t 99999 -w 640 -h 360 -fps 25|cvlc stream:///dev/stdin --sout '#standard{access=http,mux=ts,dst=:8090}' :demux=h264
```

Kód je specifikován argumenty:

- „-o“ – vypíše na standardní výstup, který zpracuje rozhraní v aplikaci VideoView
- „-t“ – nastavíme dobu streamování videa, 0 značí stream do nekonečna
- „-w“ „-h“ – značí šířku a výšku videa
- „-sout“ – určuje, kde je výstup streamu
- „-fps“ – určuje, kolik bude snímků za sekundu video zobrazovat

Video je v tomto případě získáváno pomocí protokolu http. Díky tomu je video dostupné na IP adrese Raspberry Pi a portu, v tomto případě 8090.

Do správy kamery se v aplikaci dostaneme přes tlačítko *watch*. Zde můžeme vidět uživatelské rozhraní na Obr. 20, kde je editační pole pro zadání IP adresy a portu kamery, na kterém je dostupná. Zadáme-li správné údaje a stiskneme-li tlačítko *connect*, aplikace informuje hláškou o správném připojení. Po několika sekundách můžeme sledovat video stream v kontejneru „VideoView“ zobrazeném pod tlačítkem *connect*. V opačném případě je aplikace ošetřena o výjimku „Nelze se k videu připojit“. Pro návrat do hlavní nabídky chytrého domu využijeme kontextového menu šipkou zpět.



Obr. 20 – Aktivita stream

Pod odstavcem můžeme vidět část zdrojového kódu pro fungování stream videa. Pro zobrazení videa v rámci aplikace používáme třídy VideoView. Jedná se o vizuální prvek, který přidáme-li do layoutu aktivity, poskytne plochu, na které může být video přehráváno. Zde je pojmenován jako streamView. Tato třída využívá metody setVideoURI, setMediaController a start. Bude-li video přehráváno jednoduše třídou VideoView, uživatel nebude mít kontrolu nad přehráváním. Tento problém lze vyřešit připojením instance třídy MediaController k instanci VideoView. MediaController pak poskytne sadu ovládacích prvků, které umožnují uživateli ovládat přehrávání jako například pozastavení nebo přetáčení. Metodou start se spustí video.

```
private void playStream(String src){
    Uri UriSrc = Uri.parse(src);
        streamView.setVideoURI(UriSrc);
        mediaController = new MediaController(this);
        streamView.setMediaController(mediaController);
        mediaController.setAnchorView(streamView);
        streamView.start();
        Toast.makeText(WatchCamera.this, "Connect: " + src,
Toast.LENGTH_LONG).show();
    }
}
```

4.4.4 Android manifest

Pro správnou funkčnost aplikace je nutné se také zabývat částí tvorby souboru AndroidManifest.xml. V této části jsou nastaveny důležité prvky, jako je oprávnění, ikona, názvy, styl uživatelského rozhraní, aktivity, filtry, ukládání historie aktivity, podpora nastavení hlavičky a mnoho dalšího. Kód je podrobněji uveden v kapitole 3.3.6.

Vzhled uživatelského rozhraní bylo nastaveno v adresáři style pod názvem AppTheme, kde jejím rodičem je „Theme.AppCompat.Light.DarkActionBar“. Tuto cestu uvádíme v AndroidManifestu v atributu „android:theme“. Jelikož aplikace pro chytrou domácnost je ovládána pomocí internetové sítě nebo bluetooth modulu, bylo zapotřebí zde vytvořit několik oprávnění. Za prvé, musíme přidat do souboru element <uses-permission> a atribut s hodnotou „android.permission.INTERNET“, pro možnost připojení k síti. V případě bluetooth je to atribut s hodnotou „android.permission.BLUETOOTH“.

Protože aplikace obsahuje několik aktivit, mezi kterými přecházíme, bylo využito atributu „android:noHistory“, což má za úkol, neukládat naši historii přechodu mezi aktivitami. Ve výsledku to znamená, že při stisku tlačítka „zpět“ na zařízení, nedochází k návratu do předchozí aktivity, ale rovnou vrací do místa, kde jsme aplikaci spouštěli.

Další důležitou částí v souboru AndroidManifest.xml je určení, která aktivita se zobrazí jako první při spuštění aplikace. Třídu NetworkService považujeme za hlavní. Z tohoto důvodu bylo přidáno k této aktivitě element <intent-filter> s hodnotami action.MAIN a category.LAUNCHER.

4.4.5 Měření teploty

Způsobů jak zobrazit aktuální teplotu přímo v aplikaci je mnoho. Zde v kapitole jsou uvedené dva způsoby, kde je řešeno načítání dat v aplikaci formou JSON.
První z možností je automaticky ukládat data z teplotního čidla do databáze MySQL pomocí PHP skriptu. Budeme-li chtít data zobrazovat na webu, lze využít open-source webový server Apache. Abychom data z teplotního čidla získávali v pravidelném intervalu, nastavíme na Raspberry Pi v CRONu automatické spouštění našeho PHP skriptu. Pro zobrazení dat v aplikaci Android, bychom v tomto případě mohli využít extrakci dat

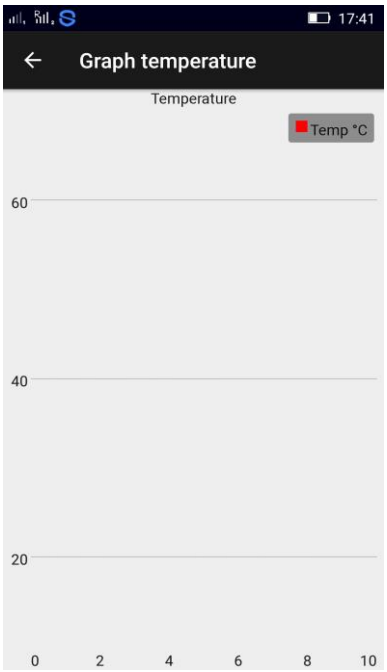
z MySQL za použití PHP skriptu, který bude vracet údaje ve formátu JSON. Aplikace by musela být vytvořena tak, aby uměla načítat data ve formátu JSON a získávat je z webového serveru. Pod odstavcem je ukázka kódu, kde se s aplikací připojujeme na web a voláme skript android.php, který vrací data získané z teplotního čidla.

```
public void getData(){
        String result = "";
        InputStream isr = null;
//get data from HTTP
        try{
            HttpClient httpclient = new DefaultHttpClient();
            HttpPost httppost = new
HttpPost("http://localhost/android.php");
            HttpResponse response = httpclient.execute(httppost);
            HttpEntity entity = response.getEntity();
            isr = entity.getContent();
        }catch(Exception e) {
            Log.e("log_tag","Error on HTTP connection " + e.toString());
            resultView.setText("Couldnt connect to DB");
        }
```

Další možností, která je zároveň pro tuto diplomovou práci nejlepší cestou je nepoužívat Apache webový server, jelikož chytrá domácnost tady není vytvářena na webovém prostoru, ale v aplikaci pro Android. Pro naše účely můžeme využít open-source databáze H2 dostupný na oficiálních stránkách [25]. Je to relační databázový systém napsaný v programovacím jazyce Java, který může být zakomponován do serverové části v jazyce Java a podporuje komunikaci klient-server. V této diplomové práci se pracuje s Raspberry Pi jako serverem, který je pro TCP připojení vytvářen v jazyce Java, takže je databáze H2 pro systém ideální volbou. Pro ukládání dat z teplotního čidla je nutné vytvořit databázi, kam budou data odesílána v určitých časových intervalech. Proto musí být vytvořen skript, který bude nastavený v tzv. CRONu na Raspberry PI. Server si tyto data načítá a posílá přes TCP komunikaci ve formě JSON do aplikace chytré domácnosti. Aplikace tyto data vyhodnotí a může je také zobrazit v grafu. Na další straně můžeme vidět kód, implementovaný v aplikaci pro získávání JSON dat.

```
private JSONObject getJsonData() throws JSONException {
    JSONObject tcpData = new JSONObject();
    tcpData.put("ID", "TEMP");
    tcpData.put("COMMAND", "CURRENTTEMP");
    JSONObject response = new JSONObject();
    return response;
}
```

Pro zobrazení údajů v grafu je potřeba využívat knihovny podporující tvorbu grafu. Na Obr. 21 můžeme vidět aktivitu Graph temperature, kde využíváme open-source knihovny GraphView dostupnou na oficiálních stránkách [22].



Obr. 21 – Aktivita Graph temperature

4.4.6 Sériová komunikace

Sériová komunikace mezi bluetooth modulem a aplikací Android probíhá podobně jako v případě internetového připojení. Na Obr. 14 můžeme vidět uživatelské rozhraní bluetooth aktivity, kde je editační pole pro zadání názvu modulu a tlačítko pro připojení. Provést připojení touto cestou nebude možné, připojujeme-li se poprvé. Tímto zároveň řešíme bezpečnost před připojením nežádoucího klienta. Abychom se mohli prvotně spárovat s bluetooth modulem, je potřeba ho vyhledat v dostupných zařízeních, což nalezneme v nastavení bluetooth klientského zařízení, například v mobilu. Při párování se s

modulem, vyskočí dialogové okno se zadáním hesla. Po zadání správného hesla už nemusíme opakovat tento krok a je možné se připojit jednoduše přes editační pole vytvořené v rozhraní bluetooth aktivity. Každý takovýto modul by měl být zabezpečen dostatečně kvalitním heslem, aby nedocházelo k nežádanému párování jiných uživatelů.

Pod textem níže můžeme vidět vytvořenou metodu findBT(), která vytváří spojení s bluetooth modulem. Třída je ošetřena podmínkou, která rozhoduje, jestli je na klientském zařízení zapnuté bluetooth. Není-li, vyskočí dialogové okno s hláškou pro zapnutí bluetooth. Aby však bylo možné navázat spojení s jiným zařízení bluetooth, musíme nejprve znát informace o zařízení a budeme také potřebovat UUID. UUID je zkratkou pro jednoznačné identifikování objektu nebo subjektu v síti. V případě bluetooth připojení, UUID se musí shodovat na obou koncích komunikace [18]. Pro připojení k Arduino bluetooth modulu je použito následující UUID: 00001101-0000-1000-8000-00805F9B34FB. Potom je možné navázat spojení prostřednictvím třídy BluetoothSocket.

```
protected void findBT() throws IOException {
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
    if (!mBluetoothAdapter.isEnabled()) {
        Intent enableBluetooth = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        startActivityForResult(enableBluetooth, 1);
    }
    Set<BluetoothDevice> pairedDevices =
mBluetoothAdapter.getBondedDevices();
    if (pairedDevices.size() > 0)
    {
        for(BluetoothDevice device : pairedDevices) {
            if (device.getName().equals(blname))
            {
                mmDevice = device;
                break;
            }}}
    UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805f9b34fb");
    mmSocket = mmDevice.createRfcommSocketToServiceRecord(uuid);
    mmSocket.connect();
    mmOutputStream = mmSocket.getOutputStream();
    mmInputStream = mmSocket.getInputStream();
}
```

Níže můžeme vidět část programu, který proběhne při stisku tlačítka nazvaného *bl_swtch_LR_light*. Výsledkem je zapínání či vypínání světla. Podrobněji je proces zřejmý z kapitoly 4.3.2 a vývojového diagramu na Obr. 18.

```
final Switch light1 = (Switch) findViewById(R.id.bl_swtch_LR_ligth);
light1.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if (light1.isChecked()) {
            if (stat == 1) {
                try {
                    mmOutputStream.write('1');
                } catch (IOException e) {
                    Toast.makeText(BluetoothService.this, "Connection
not established with your home", Toast.LENGTH_LONG).show();
                    e.printStackTrace();
                }
            } else
                Toast.makeText(BluetoothService.this, "Connection not
established with your home", Toast.LENGTH_LONG).show();
        } else {
            if (stat == 1) {
                try {
                    mmOutputStream.write('2');
                } catch (IOException e) {
                    Toast.makeText(BluetoothService.this,
                            "Connection not established with your home",
                            Toast.LENGTH_LONG).show();
                    e.printStackTrace();
                }
            } else
                Toast.makeText(BluetoothService.this, "Connection not
established with your home", Toast.LENGTH_LONG).show();
        }
    }
});
```

5 Zhodnocení výsledků a doporučení

V této kapitole jsou charakterizovány nejdůležitější teoretické poznatky, které byly zjištěny při tvorbě diplomové práce.

Při testování aplikace Android Auto bylo zjištěno několik nedostatků a částí bez, kterých se aplikace neobejde. Bez použití datového připojení v aplikaci není možné používat navigaci ani hlasové ovládání. Neumí komunikovat s výbavou auta, jako je vyhřívání sedadel, klimatizace nebo parkovací kamera. Nepodporuje velmi oblíbenou navigaci Waze a pokud nepoužíváme přehrávač Google Play budeme těžko hledat alternativní přehrávač, který je podporován. Aplikace má, ale i mnoho pozitivních poznatků jako je kompatibilita s téměř libovolnými chytrými telefony, telefon se při jízdě nabíjí, telefonování, v noci dokáže měnit barevné schéma, na palubním počítači vozu lze neustále vidět velikost mobilního signálu, stav baterie nebo jaké probíhá připojení.

Vytvořená aplikace pro chytrou domácnost nepotřebuje webový server jako je Apache a lze ho nahradit například Raspberry Pi, Arduinem nebo jiným zařízením. Vždy je nutné vytvářet software na straně serveru i klienta a to hlavně tak, aby klient při manipulaci měl pokaždé aplikaci aktualizovanou na současný stav v domácnosti. Budeme-li se chtít připojit vzdáleně mimo lokální síť je zapotřebí veřejné IP adresy. Vždy při tvorbě je nutné dbát na bezpečnost, aby server na daném zařízení nebyl přístupný pro neautorizované uživatele a byl dostatečně zabezpečený silným heslem. Dalším bezpečnostním prvkem by mělo být povolené používání aplikace jen pro dané zařízení klienta. Databázi lze řešit využitím open-source a tak není nutné za ni platit. Aplikace byla vytvořena tak, aby splňovala aspoň obecné požadavky charakterizované v kapitole 2.5.2.

Aplikace není dále publikována v obchodě Google Play. Práce má být nápomocna pro zájemce o vylepšení své domácnosti, kteří se chtějí vydat vlastní cestou podle svých specifických potřeb a neplatit za komerční řešení.

6 Závěr

Tato práce se z teoretického hlediska zabývá charakteristikou vývoje tvorby aplikace, distribucí a uplatněním různých Android verzí na trhu. Dále byla věnována různému pohledu na aplikace, které mohou být rozděleny do sekcí podle oblasti využití nebo podle zaměření na různé typy zařízení. Zabývá se také popisem různorodosti nezbytného využívání technologií pro správnou funkčnost určitých aplikací. Jednou z novinek na trhu Google Play je právě aplikace Android Auto, která je s větší části popisována na základě prováděného testování.

Jádrem praktické části práce byla tvorba aplikace pro chytrou domácnost. Kapitola je věnována problematice procesu vývoje od tvorby uživatelského rozhraní až po jádro programu. Popisuje technologické postupy, bez kterých se chytrá domácnost neobejde. Poté je část práce věnována grafickému návrhu rozhraní aplikace. Zde jsou uvedeny soubory, se kterými se pracovalo a které byly vytvářeny. Následně jsou uváděny některé widgety a popis jejich atributů. Aplikace konkrétně splňuje obecný požadavek podle testování funkčnosti kódu CR-5, kde při změně orientace zůstávají stejné prvky, činnosti a je zachována funkčnost. Pro dosažení tohoto výsledku je navržena sktruktura využívání elementů a správné používání atributů. Další část je věnovaná komunikaci klient-server.

Diplomová práce je zaměřena spíše na klienta, a proto je zde i více funkčních prvků. Pro účely chytré domácnosti se aspoň z části zabývá serverem. V tomto případě byly využity dva řídící obvody, na kterých bylo demonstrováno různé využití. Prvním z nich je Raspberry Pi, který disponuje ethernetovým portem, a proto byl využit pro komunikaci prostřednictvím internetové sítě. Arduino tímto portem nedisponuje, a tak byl využit přídavný modul pro komunikaci přes bluetooth. Byly vytvořeny dva serverové programy podle toho, o jakou komunikaci se jednalo. V případě síťové komunikace je využívána transportní vrstvy TCP/IP. V případě bluetooth je využívána sériová komunikace.

Na straně klienta byla naprogramována aplikace, která dovede se serverem komunikovat. V kapitole klient jsou charakterizovány některé důležité části zdrojového

kódu pro samotnou funkčnost aplikace. Na základě interakce uživatele dovede odesílat nebo přijímat data. Aplikace zvládá připojení k serveru prostřednictvím internetové sítě, kde umožňuje regulovat osvětlení, zamykat dveře, aktivovat alarm, otevírat či zavírat garážová vrata, měřit teplotu nebo sledovat bezpečnostní kamery. V další aktivitě se umí aplikace připojit k bluetooth modulu, kde je možnost regulace osvětlení. Funkčnost měření teploty a sledování kamer na serverové části je popsána jen teoreticky.

Dalším námětem diplomové práce může být doplnění do aplikace možnost nastavování ústředního topení, tvorby režimů, kontrolování odběru elektrického proudu nebo třeba měření odběru spotřebované vody. Nebo se podrobněji věnovat serverové části tvorby programu s použitím vhodných konstrukčních prvků.

7 Seznam použitých zdrojů

[1]. **Tomáš Xaver.** *Jak šel čas s operačním systémem Android* [online]. [cit. 5-11-2016]. Dostupné z: http://www.androidtip.cz/jak-sel-cas-s-operacnim-systemem-android/

[2]. **IDC Analyze the Future**. *Smartphone OS Market Share, 2016 Q2* [online]. [cit. 6-11-2016]. Dostupné z: http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[3]. **Ľuboslav Lacko.** *Vývoj aplikací pro Android.* Brno, 2015, 463 s. ISBN 978-80-251-4347-6.

[4]. **Android Developers**: *Platform Architecture* [online] [cit. 8-11-2016]. Dostupné z: https://developer.android.com/guide/platform/index.html

[5]. **Android**: *ART and Dalvik* [online] [cit. 13-11-2016]. Dostupné z: http://source.android.com/devices/tech/dalvik/index.html#features

[6]. **Meier, Reto**. *Professional Android application development.* Wiley Publishing, Inc., Indianapolis, 2009, 409 s.. ISBN 978-0-470-34471-2.

[7]. **Jiří Vávrů, Miroslav Ujbányai.** *Programujeme pro Android 2., rozšířené vydání.* Praha, Grada Publishing a.s., 2013, 256 s. ISBN 978-80-247-4863-4

[8]. **Vogella**. *Scheduling of tasks with the Android JobScheduler*. Dostupné z: http://www.vogella.com/tutorials/AndroidTaskScheduling/article.html#schedulingtasks

[9]. **Android Developers**. *Services.* [online] [cit. 17-11-2016]. Dostupné z: https://developer.android.com/guide/components/services.html

[10]. **Grant Allen**. *Android 4: průvodce programováním mobilních aplikací*. Computer Press, Brno, 2013, 656 s., ISBN 978-80-251-3782-6.

[11]. **Miroslav Ujbányai**. *Programujeme pro Android.* Grada Publishing a.s., Praha, 2012, 192 s. ISBN 978-80-3995-3.

[12]. **Android Developer**. *Platform Versions*. [online] [cit. 5-11-2016]. Dostupné z: https://developer.android.com/about/dashboards/index.html#Screens

[13]. **Android librairies**. *Media framework.* [online] [cit. 6-11-2016]. Dostupné z: http://www-igm.univ-mlv.fr/~dr/XPOSE2008/android/archi_lib.html

[14]. **Vyvíjíme pro Android – tvoříme aktivity.** *Životní cyklus*. [online] [cit. 12-11-2016]. Dostupné z: http://www.abclinuxu.cz/clanky/vyvijime-pro-android-tvorime-aktivity#!/-1/

[15]. **Aponia GPS Navigation.** *Product.* [online] [cit. 11-3-2017]. Dostupné z: http://www.aponia.com/cs/products/traffic-information

[16]. **MURPHY, Mark L**. *Beginning Android 2*. 1. vydání. New York: Springer Verlag 2010. 416s. ISBN 978-1-4302-2629-1.

[17]. **Core App Quality**. *Android Developers* [online]. 2010 [cit. 10-03-2017]. Dostupné z: http://developer.android.com/distribute/essentials/quality/core.html

[18]. **Andreas Göransson, David Cuartielles Ruiz.** *Professional Android OpenAccessory Programming with Arduino*. Vydání: John Wiley &Sons, Inc., Indianapolis, Indiana, 2013. 408 s. ISBN: 978-1-118-45476-3.

[19]. **Mironet.** Pokladní set KASA FIK plus. [online] [cit. 15-03-2017]. Dostupné z: https://images.mironet.cz/foto/3/97201196/kasa-fik-plus-plocha.jpg

[20]. **Lagardere Active ČR.** *Duolingo.* [online] [cit. 15-03-2017]. Dostupné z: http://www.koule.cz/cs/clanky/3-aplikace-se-kterymi-zvladnete-cizi-jazyky-supito-presto--52326.shtml

[21]. **Android Developers Blog.** *Message notifications on the car's display.* [online] [cit. 21-03-2017]. Dostupné z: https://android-developers.googleblog.com/2015/04/enable-your-messaging-app-for-android.html

[22]. **Graph View**. *Open source graph plotting library for Android*. [online] [cit 21-03-2017]. Dostupné z: http://www.android-graphview.org

[23]. **Arduino.** *Sériová komunikace a cykly.* [online] [cit. 22-03-2017]. Dostupné z: https://arduino.cz/seriova-komunikace-a-cykly/

[24]. **Google Play.** *Android Auto.* [online] [cit. 20-03-2017]. Dostupné z: https://play.google.com/store/apps/details?id=com.google.android.projection.gearhead&hl=cs

[25]. **H2 Database.** *H2 Database Engine.* [online] [cit. 22-03-2017]. Dostupné z: http://www.h2database.com/html/main.html

[26]. **Václav Nývlt, Pavel Kasík.** *Android TV.* [online] [cit. 13-03-2017]. Dostupné z: http://technet.idnes.cz/recenze-android-tv-0n4-/tec_video.aspx?c=A141203_083154_tec_video_nyv

8 Přílohy

Příloha A Kompletní projekt klient vytvořený v Android Studiu:

- SmartHome

Příloha B Kompletní projekt server vytvořený v NetBeans IDE:

- ServerRaspi

Příloha C Kompletní projekt server vytvořený v Arduino IDE:

- ServerArduino