Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Bakalářská práce

Mobilní aplikace osobní trenér pro Android

Ondřej Procházka

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Ondřej Procházka

Informatika

Název práce

Mobilní aplikace osobní trenér pro android

Název anglicky

Personal Trainer mobile application for Android

Cíle práce

Práce je zaměřena na problematiku vývoje aplikací pro OS Android. Cílem je vytvořit podpůrnou mobilní aplikaci pro fitness fungující jako organizátor tréninku, která bude řešit nedostatky a obsahovat funkcionalitu chybějící v
aktuálně existujících aplikacích podobného typu.

Dílčím cílem je popsat, demonstrovat, vývoj software pro OS Android.

Metodika

Metodika řešené bakalářské práce je založena na analýze odborných informačních zdrojů. Na základě syntézy teoretických poznatků budou shrnuty možnosti vývoje aplikací pro OS Android. Zjištěných poznatků bude využito při tvorbě aplikace, která je hlavním cílem práce.

Bude proveden návrh a implementace mobilní aplikace, přičemž návrh bude reflektovat slabá místa a funkční nedostatky zjištěné analýzou již existujících mobilních aplikací určených pro podobné účely.

Aplikace bude dále otestována, postup jejího vývoje popsán a zhodnocen a budou navrženy případné další možnosti jejího rozvoje.

Doporučený rozsah práce

35-40 stran

Klíčová slova

OS Android, Osobní trenér, Mobilní aplikace, Implementace, Vývoj

---

Doporučené zdroje informací

ALLEN, G. *Android 4 : průvodce programováním mobilních aplikací.* Brno: Computer Press, 2013. ISBN 978-80-251-3782-6.
HEROUT, P. *Java a XML.* České Budějovice: Kopp, 2007. ISBN 978-80-7232-307-4.
LACKO, Ľ. *Vývoj aplikací pro Android.* Brno: Computer Press, 2015. ISBN 978-80-251-4347-6.
UJBÁNYAI, M. – VÁVRŮ, J. *Programujeme pro Android.* Praha: Grada, 2013. ISBN 978-80-247-4863-4.

Předběžný termín obhajoby

2016/17 LS – PEF

Vedoucí práce

Ing. Jiří Brožek, Ph.D.

Garantující pracoviště

Katedra informačního inženýrství

Elektronicky schváleno dne 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 1. 11. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 12. 03. 2017

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Mobilní aplikace osobní trenér pro Android" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské 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(ka) uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.3.2017 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu mé bakalářské práce panu Ing. Jiřímu Brožkovi, Ph.D. za jeho rady a čas, který mi věnoval při tvorbě této práce. Dále bych chtěl poděkovat rodině za podporu, kterou mi poskytli během psaní bakalářské práce. V neposlední řadě patří mé díky lidem, kteří se podíleli na otestování výsledné aplikace.

Mobilní aplikace osobní trenér pro Android

Souhrn

Tato bakalářská práce se zabývá vývojem mobilní aplikace pro správu osobního fitness tréninku. Vyvíjená aplikace je určena pro zařízení s operačním systémem Android.

V teoretické části je provedena analýza několika aplikací podobného typu, které jsou v době psaní této práce dostupné na trhu. U každé aplikace jsou vypsány její silné i slabé stránky. Dále je charakterizován OS Android. Je ve zkratce probrána historie OS, verze, které jsou v současné době aktuální a poté jsou nastíněny základní znalosti potřebné pro vývoj aplikace. Je zde také popsáno vývojové prostředí AndroidStudio. Nakonec je krátké seznámení s jazykem Java, XML a s SQLite.

Ve vlastní práci je uveden návrh aplikace a je charakterizován datový model, jež je v aplikaci využit. Dále se zde uvádějí úkázky zdrojových kódů vybraných částí vyvíjené aplikace.

Klíčová slova: Android, vývoj, programování, fitness, XML, Java, mobilní zařízení

Personal Trainer mobile application for Android

Summary

This bachelor thesis deals with the development of mobile applications for managing personal fitness training. The developed application is designed for devices running on OS Android.

In the theoretical part is an analysis of several applications of this type, which are at the time of writing this thesis available in the market. For each application are listed its strengths and weaknesses. The Android OS is also there characterized. It is discussed in brief the history of OS versions that are currently up to date, and then presents the basic knowledge for application development. There is also described android studio development environment. At the end is a brief introduction to Java, XML and SQLite.

The own work includes design of application and it is here characterized a data model, which is used in the application.There are also shown samples of selected parts of the source code developed applications.

Keywords: Android, developement, programing, fitness, XML, Java, mobile device

Obsah

1 Úvod

Vývoj aplikace pro OS Android byl zvolen za téma bakalářské práce z důvodu zájmu autora o tuto oblast informačních technologií a zároveň získání nových znalostí z oblasti vývoje. Tématické zaměření vyvíjené aplikace směřuje na správu fitness tréninku, jelikož je to jedna z autorových oblíbených činností.

Prvotním faktorem, který ovlivnil jakým směrem vyvíjenou aplikaci tématicky směřovat, byla potřeba jednoduché aplikace v českém jazyce, která by uživateli pomohla se správným postupem při cvičení. Požadavkem tedy bylo, aby aplikace poskytovala informace o tom, jak a s čím správně cviky provádět a také aby uživateli co nejvíce usnadnila průběh cvičení.

2 Cíl práce a metodika

2.1 Cíl práce

Cílem bakalářské práce je vytvoření funkční aplikace, která bude sloužit k organizaci fitness tréninku a zároveň zaplní nedostatky nalezené v již existujících aplikacích podobného typu.

2.2 Metodika

Práce je rozdělena do 4 kapitol. První kapitola nese název „Teoretická východiska“, po ní následuje kapitola „Praktická část“ a nakonec kapitoly „Zhodnocení a další možný rozvoj“, a „Závěr a doporučení“.

První kapitola „Teoretická východiska“ se zabývá analýzou a zhodnocením již existujících aplikací podobného typu na trhu. Je představeno 5 aplikací se stručným popisem a zhodnocením jejich výhod a nevýhod. Dále se v této kapitole probírá operační systém Android. Ve stručnosti je vysvětleno co to OS Android je, jaká je jeho historie a s jakými jeho verzemi se můžeme v současné době setkat. Poté následuje popis nástrojů, které jsou pro vývoj aplikací pro OS Android důležité. Na konci je krátké seznámení s jazyky SQL, Java a XML jelikož jsou tyto jazyky použity k vytvoření aplikace, která je cílovým výstupem této práce.

Druhá kapitola „Praktická část“ obsahuje popis vývoje již samotné aplikace včetně návrhu. Z důvodu velikosti samotné aplikace nejsou popsány úplně všechny části. Popis se zaměřuje převážně na části aplikace, které jsou něčím zajímavé nebo důležité pro funkci aplikace. Dalším bodem této kapitoly je také popis datového modelu, který je v aplikaci použit.

V předposlední kapitole „Zhodnocení a další možný rozvoj“ je uvedeno zhodnocení aplikace v praxi a její případný další rozvoj v budoucnosti.

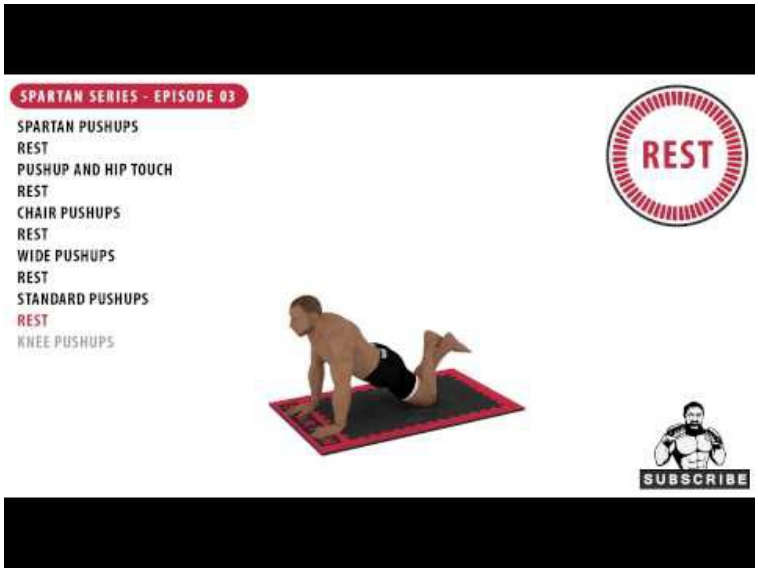
3 Teoretická východiska

3.1 Aplikace na trhu

V současné době existuje velké množství aplikací, které se zaměřují na skoro každé téma. Ani tématika fitness není výjimkou, a tak lze nalézt mnoho variací aplikace pro správu fitness tréninku. Součástí této práce je tedy také analýza vzorku dostupných aplikací na trhu. Z důvodu mnoha programů je vybráno 5 aplikací, které zastupují ostatní díky své rozmanitosti.

3.1.1 Home Workout MMA Spartan PRO

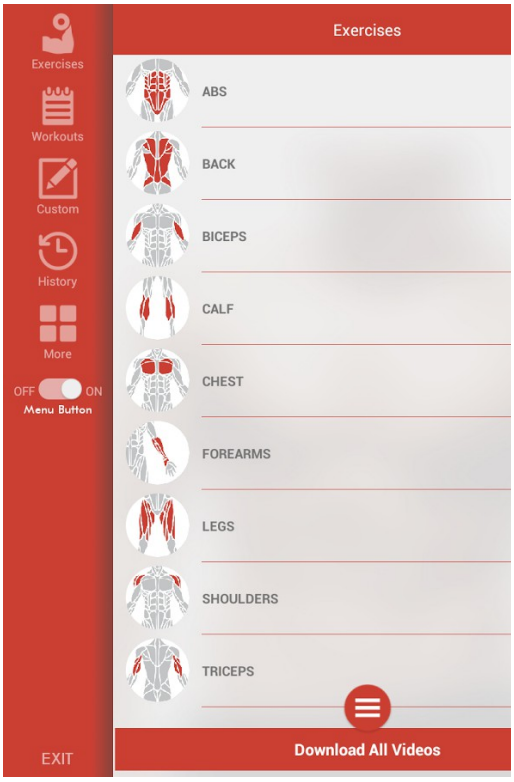
Tato aplikace je vhodná pro ty, kteří chtějí cvičit, ale nemají čas chodit do fitness centra nebo posilovny. Aplikace nabízí řadu cviků, ke kterým není zapotřebí žádných nástrojů. Výhodou je, že není zapotřebí připojení k internetu, aby si uživatel mohl na animovaných obrázcích prohlédnout, jak správně cvik provádět. Bohužel chybí textový popis provedení cviku. Cviky samotné jsou rozděleny dle svalových partií, které procvičují. K měření provedení cviků je k dispozici pouze časové omezení. Aplikace umožňuje přidávat nové cvičební plány, nebo využít již plánů vytvořených. Nevýhodou je, že ve verzi, která je zdarma, není možné plány mazat či upravovat. Pro kompletní funkčnost je nutné aplikaci zakoupit. Nevýhodou je také to, že není možné nastavit český jazyk.



Obrázek 1: Cvik v Home Workout MMA Spartan PRO (vlastní zpracování)

3.1.2 Fitness and Bodybuilding

Jednoduchá a srozumitelná aplikace, která poskytuje cviky ke cvičení doma i ve fitness centru. Cviky jsou rozdělenné dle svalových partií. Každý cvik je popsán textovou formou v jednotlivých krocích. Dále je pak možné si cvik prohlédnout na obrázcích, nebo pokud je k dispozici připojení k internetu shlédnout provedení ve videu. Po stránce plánování cvičení, je možné vytvořit si individuální plán a ten si rozdělit do několika dnů. V průběhu samotného cvičení, aplikace umožňuje nastavení časového odpočtu a po skončení časového intervalu aplikace uloží, že jste v daný den cvik provedli. Tímto poskytuje uživateli zpětnou vazbu. Aplikace je zdarma a pouze v anglickém jazyce.

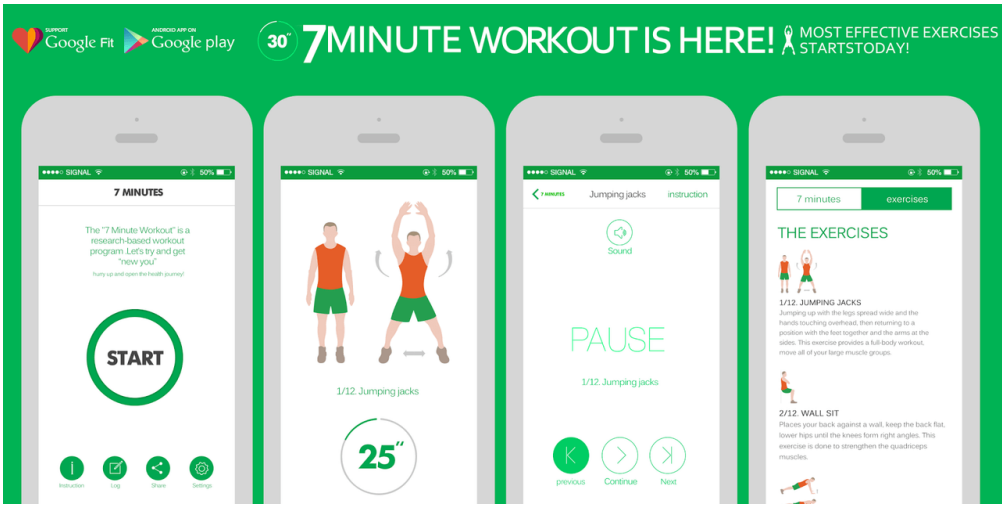


Obrázek 2: Fitness and Bodybuilding (Fitness and Bodybuilding Foto)

3.1.3 7 minute

Tato aplikace umožňuje pouze cvičení dle již zadaných cvičebních plánů. Bohužel zde není jednotná galerie cviků. Cviky si lze prohlédnou v již vytvořených plánech. K měření cvičení slouží časový odpočet, který je již ke každému cviku přednastaven. Celkově je

aplikace jednoduchá, ale také nestabilní a najdou se i jisté nesrovnalosti, jako je například občasný výskyt českého jazyka v jinak anglickém textu. Uživatele také nepotěší neustálé vyskakování reklamních nabídek.



Obrázek 3: 7 minute (7 minute Foto)

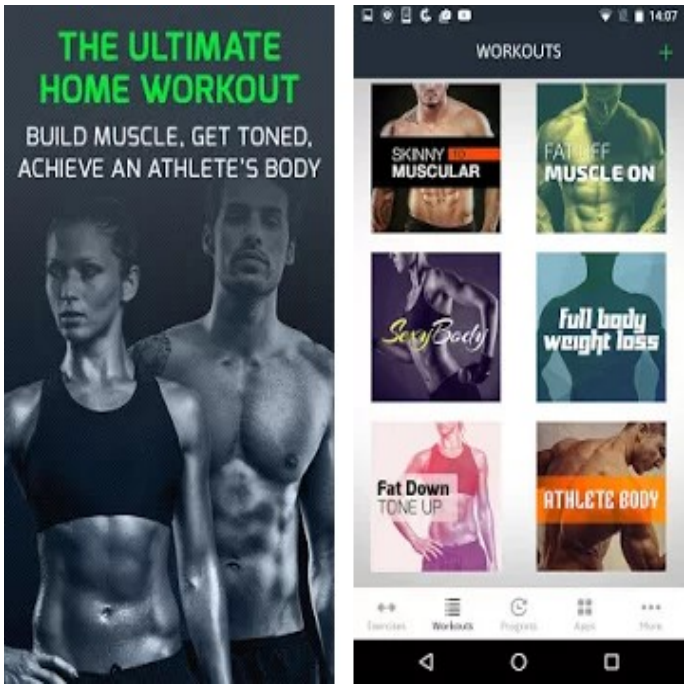
3.1.4 JetFit

JetFit je profesionální aplikace pro správu fitness. Tato aplikace nabízí širokou škálu cviků, řazených dle svalových partií. Cviky jsou popsány textově v jednotlivých krocích, nebo také pomocí animovaných obrázků, ke kterým je ale nutné mít k dispozici připojení k internetu. Cviky je možné také přidávat. Aplikace umožňuje vytvářet vlastní cvičební plány a nebo využít již plány vytvořené. V průběhu cvičení je možné spustit stopky nebo odpočet. Po každém cviku je nutné kliknout na tlačítko, aby se výsledek uložil. Po stisku následuje automaticky pauza, která je nastavena na 1 minutu. Aplikace umožňuje zaznamenávání cvičebních výsledků, a poskytuje tak zpětnou vazbu. Nevýhodou je horší orientace v uživatelském rozhraní. Aplikace je k dispozici v základní verzi zdarma, nebo také ve verzi PRO, která je ovšem placená. Aplikace podporuje několik jazyků. Český jazyk mezi nimi ale bohužel není.

Obrázek 4: JetFit (Jetfit Foto)

3.1.5 Home Workouts Personal Trainer

Přehledná aplikace, která potěší svým příjemným designem. Cviky jsou zde vyhledávány podle svalových partií, a k jejich popisu slouží tři možnosti. Uživatel si tak může vybrat, zda chce vidět jak cvik provést buď na animovaném obrázku, nebo za pomoci videa, ke kterému ale je nutné připojení k internetu. Další možností je pak textový popis. Po stránce plánování si uživatel může vybrat z několika již sestavených cvičebních plánů, nebo si navrhnout vlastní individuální plán. Plány je možné mazat i editovat. K měření cvičení slouží časový odpočet, který si uživatel nastaví při vytváření. V průběhu vytváření plánu uživatel také zadává kolikrát plánuje v jednotlivé sadě cvik provést. Tento údaj je pak využit při skončení cvičebního plánu ke zhodnocení. Aplikace také ukládá informace o tom, kdy jaký cvik uživatel vykonal a kolikrát. Aplikace je zcela zdarma a pouze v anglickém jazyce.



Obrázek 5: Home Workouts Personal Trainer (Home Workouts Foto)

3.2 Android

Jelikož bude cílová aplikace vyvíjena pro OS Android, je dobré vědět něco málo o základech tohoto systému. Proto se tato kapitola věnuje historii OS Android, tomu jak tento software funguje a s jakými verzemi se můžeme v dnešní době setkat. Na konci je také uvedeno s jakými nástroji se bude při vývoji pracovat.

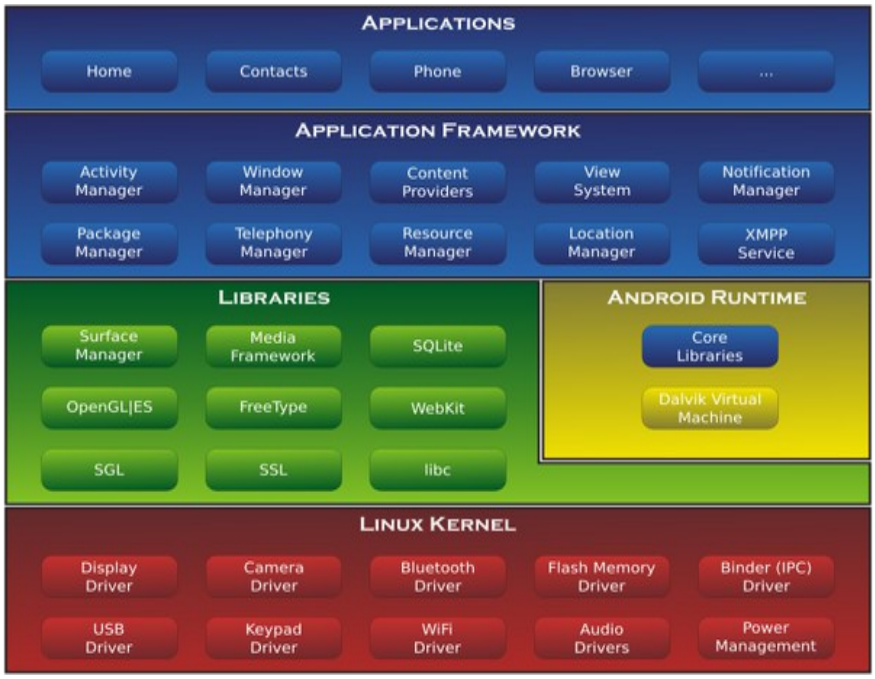
3.3.1 Historie ve zkratce

Android není dílem Google, který ho odkoupil. Společnost Android Inc. vznikla v říjnu roku 2003, kdy ji založili čtyři zakladatelé v kalifornském Palo Alto. O dva roky později, v roce 2005 Google odkoupil Android Inc. zhruba za 50 milionů dolarů. Jeho dnešní hodnota je vyšší o tři řády. V listopadu 2007 vznikla Open Handset Alliance, která stojí za vývojem Androidu dodnes, v té době byl zároveň vydán vývojařský kit (SDK). V září 2008 přišel v USA na trh HTC Dream (G1), první chytrý telefon s Androidem 1.0. který měl na trhu inteligentních telefonů podíl zanedbatelného 0,5% (*Lacko, 2015*).

3.3.2 Co Android je a jak to funguje

Android je open-source platforma postavená na bázi Linuxu a určená hlavně pro mobilní zařízení, tedy chytré telefony, tablety, fotoaparáty a navigace (*Lacko, 2015*).

Pro vývoj je potřeba znát alespoň základy architektury operačního systému, na kterém má vyvíjená aplikace fungovat.



Obrázek 6: Architektura OS Android (API)

Nejvyšší úrovní je Aplikační vrstva, tedy nainstalované aplikace, které uživatel využívá.

Po aplikační vrstvě následuje vrstva Application Framework. Aplikační framework obsahuje v aplikacích opakovaně použitelný software, jako jsou ikony, ovládací prvky a podobně. Framework je vytvořen v programovacím jazyce Java a je to nejdůležitější vrstva pro programátory. Poskytuje aplikacím základní služby systému (*Lacko, 2015*).

Další v pořadí je vrstva Android Runtime. V této vrstvě se nacházejí základní knihovny. Nachází se zde také DVM (Dalvik Virtual Machine), což je virtuální stroj, který umožňuje každé aplikaci běžet jako samostatný proces.

Nad jádrem je umístěna vrstva knihoven, které umožňují přímý přístup aplikací k různým komponentám systému Android. Jsou to nativní knihovny vytvořené v C/C++. Tvoří mezivrstvu mezi různými komponentami vyšších vrstev a linuxovým jádrem (*Lacko, 2015*).

Úplně vespod se nachází Linux Kernel. Toto jádro přímo komunikuje s hardwarem mobilních zařízení.

3.3.3 Verze

V průběhu času vycházeli stále nové verze OS Android, které přinášeli nové možnosti a řešily problémy předchozích verzí. Od verze 1.5 také začala tradice pojmenovávání dle sladkostí. Dále je uveden seznam současných verzí společně s některými novinkami, které sebou přinesly. Informace o jednotlivých verzích byly zjištěny a zpracovány z webových stránek „https://www.svetandroida.cz/historie-androidu-201506“ a z knihy „Vývoj aplikací pro Android“, kterou napsal Luboslav Lacko.

3.3.3.1 Android 4.4

KitKat, verze která vyšla v roce 2013, se vyznačuje vyšším výkonem, vylepšenou podporou zařízení s vícejádrovými procesory a rychlejším multitaskingem.

3.3.3.2 Android 5.0

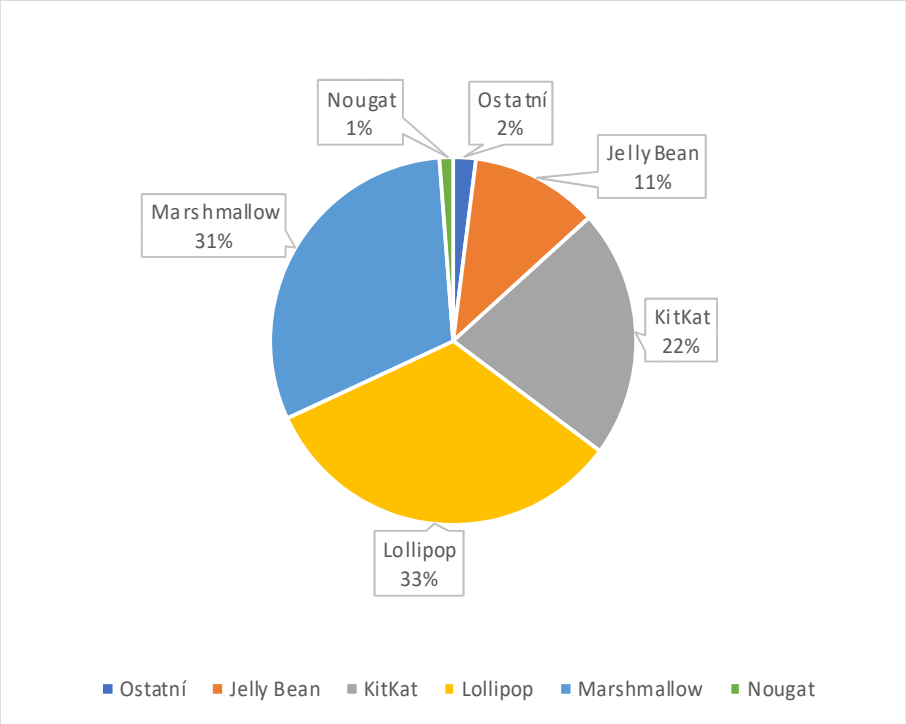
Další verzí je Lollipop. Tato verze přináší podporu náhledu tisku, podporu 64bitových procesorů a design ve stylu „Material“.

3.3.3.3 Android 6.0

Verze Marshmallow. Tato verze přináší nový systém správy baterie, funkci Smart Lock a snižuje energetickou náročnost systému.

3.3.3.4 Android 7.0

Momentálně poslední verzí je Nougat. Mezi novinky v této verzi patří úpravy v blokování čísel, jako například možnost zachování blokovaných čísel i při restartování zařízení, či nastavení do továrního módu. Dále pak umožňuje práci s více okny, kdy dvě aplikační okna mohou sdílet každé jednu polovinu obrazovky. Nakonec také můžeme zmínit podporu API Vulkan pro vykreslování 3D.



Obrázek 7: Graf znázorňující zastupitelnost API na trhu v roce 2017 (vlastní zpracování za využití informací z Android (Market Share))

3.3.4 API

Hodnota API je označení úrovně frameworku aplikačního rozhraní. Jedná se o sbírku funkcí, procedur a tříd nějaké knihovny, které může vývojář použít. Definuje, jak jsou funkce knihovny volány ze zdrojového kódu.

Verze Platformy	API Level	VERSION_CODE
Android 7.1.1 a 7.1	25	N_MR1
Android 7.0	24	N
Android 6.0	23	M
Android 5.1	22	LOLLIPOP_MR1
Android 5.0	21	LOLLIPOP
Android 4.4W	20	KITKAT_WATCH
Android 4.4	19	KITKAT
Android 4.3	18	JELLY_BEAN_MR2
Android 4.2 a 4.2.2	17	JELLY_BEAN_MR1
Android 4.1 a 4.1.1	16	JELLY_BEAN
Android 4.0.3 a 4.0.4	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1 a 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.3 a 2.3.4	10	GINGERBREAD_MR1
Android 2.3, 2.3.1 a 2.3.2	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

Tabulka 1: Tabulka API s odpovídajícími verzemi (vlastní zpracování za využití informací z Android (API))

3.3.5 Základní komponenty

Na rozdíl od konceptu vývoje pro iPhone, kde stavíme aplikaci poměrně striktně jako MVC architekturu, v Androidu si vrstvení aplikace určujeme sami. Leckdy je ale vhodné se u MVC alespoň inspirovat, aby zůstala aplikace čitelná. Aplikace v Androidu sestávají ze čtyř základních typů komponent (*Vývoj pro Android komponenty*).

Dále jsou uvedeny čtyři základní komponenty.

3.3.5.1 Aktivita

Stěžejní částí programu, bez které by aplikace nešla spustit, je takzvaná aktivita. Tu si můžeme velmi zhruba představit jako takový controller či presenter z MVC/P, jelikož stejně jako on má na starosti, aby všechna data, jež získá od nižších vrstev, byla správně zobrazena uživateli. Tato definice má ale problém, že Android nemá view, které je v MVC/P. Přesnější by bylo uvést, že aktivita je prezentační vrstva aplikace. Každá aktivita je potomkem třídy Activity (*Vyvijime pro Android Activity*).

Aktivita je zásadní třída, která je uživateli zobrazena po spuštění aplikace. Aplikace se většinou skládají z většího počtu aktivit, které si navzájem předávají údaje. Z jedné aktivity je možné přepnout se na další, nebo se vrátit na předchozí. Aktivity umožňují uživatelům přes grafické rozhraní (GUI) získávat informace od aplikace a ovládat ji. Dají se představit jako například klasická stránka webové aplikace. Přes aktivitu se většinou implementuje více nebo méně komplexní částečná úloha, kterou má uživatel uskutečnit, jako například vyplnit údaje do formuláře, nastavit parametry, vybrat si položku ze seznamu a podobně. (Allen*, 2013*).

Jelikož aktivitám se víceméně podobají fragmenty, jsou zde uvedeny, i když nepatří mezi základní komponenty. Fragmenty vznikly na základě potřeby optimalizovat uživatelské rozhraní pro funkčnost, jak na mobilních telefoních zařízeních, tak na tabletech.

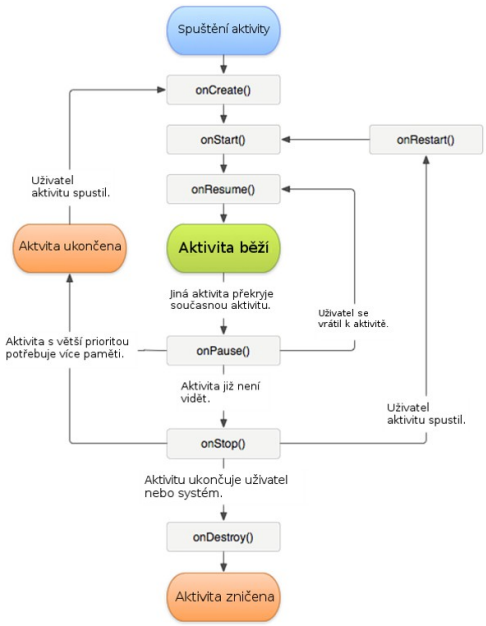
Fragment představuje část nebo celé uživatelské rozhraní nějaké activity i s příslušnými metodami. Na rozdíl od activity ale může být součástí jiné activity nebo fragmentu. Díky

tomu se dají vytvářet flexibilní a znovupoužitelné jednotky uživatelského rozhraní. To umožňuje vyvíjet bez duplikace kódu optimalizovaná rozhraní pro větší displeje, jako jsou tablety a televize (*Vyvíjíme pro Android Fragmenty*).

Pro práci s aktivitami je nutné znát životní cyklus aktivity, tedy to, v jakých situacích se může jednotlivá aktivita ocitnout.

Android, jakožto OS primárně pro chytré telefony, sám zodpovídá za activity a vše co s nimi souvisí. Aby majitel telefonu mohl přijmout příchozí hovor nebo si přečíst SMS, i v okamžiku, kdy pracuje s nějakou aplikací, musí OS rozpoznat co má jakou prioritu a podle toho se zachovat. Může nastat několik situací, jako například, že příchozí hovor pozastavení aplikaci, kterou uživatel zrovna používá.

Z toho plyne, že není vždy jasné, jak dlouho aktivita poběží, a proto existují různé metody, které se volají v různých situacích. Podívejte se na následující schéma (*Android porgramování*).



Obrázek 8: Schéma životního cyklu Aktivity (IT network)

Životní cyklus aktivity má tři fáze. První fází je „Aktivita na popředí“ a to je aktivita, která je zobrazena na displeji a interaguje s uživatelem. Tyto aktivity jsou ve stavech Running

nebo Resumed. Druhou fází je „Pozastavená aktivita“.To je aktivita, která je uložená v paměti, může být částečně viditelná, ale nemá interakci s uživatelem. Poslední je fáze „Zastavená aktivita“. Taková aktivita je stále v paměti, ale je kompletně překrytá jinou aktivitou.

Dále jsou uvedeny metody, které jsou volané pro při přechodu aktivity z jednoho stavu do dalšího. Informace byly přejaty z knihy „*Vývoj aplikací pro Android*“ od Luboslava Lacko.

Metoda onCreate() se aktivuje po prvním spuštění aktivity. V těle metody se zatím na pozadí vytváří uživatelské rozhraní a konfigurují se proměnné a objekty nutné k běhu aktivity. Po zavolání metody onCreate() je aktivita stále zastavená, neviditelná a nekomunikuje s uživatelem. Kód metody onCreate() vloží vývojové prostředí do hlavní aktivity nově vytvořeného projektu (*Lacko, 2015*).

Metody onStart() a onResume() převádí aktivitu do popředí. V metodě onStart() se realizují činnosti potřebné k tomu, aby bylo možné zobrazit uživatelské rozhraní aktivity. Rozdíl mezi metodami je zřejmý z diagramu. OnStart() se volá při spouštěních, která následují po předchozím zastavení aktivity. Po spuštění metody onStart() obvykle následuje spuštění metody onResume(). Metoda onResume() se volá tehdy, když aktivita přechází z pozadí do popředí (*Lacko, 2015*).

Metoda onPause() obstarává to, že v případě, že je spuštěná jiná aktivita, přechází ta, která byla spuštěná předtím na popředí, do pozadí. V této metodě je vhodné automaticky uložit změny údajů, se kterými aktivita pracovala, například do databáze, souborů, dokumentů a podobně (*Lacko, 2015*).

Metoda onStop() se volá při zastavení aktivit z jiného důvodu, než je nedostatek paměti. Aktivita je stále v Back Stacku a uživatel ji může znovu zobrazit na popředí. Tehdy se volá metoda onRestart() (*Lacko, 2015*).

Metoda onDestroy() se volá při ukončování životnosti aktivity, bez ohledu na to, zda se tak stane explicitně nebo implicitně (*Lacko, 2015*).

3.3.5.2 Služby

Služby představují déle trvající operace a operace na pozadí. Také umožňují spolupracovat se vzdálenými procesy. Na rozdíl od aktivit běží služby na pozadí a nepotřebují tak interakci přímo s uživatelem. Služby umožňují pracovat asynchronně s hlavním vláknem, tedy provádět operace, jejichž realizace zabírá více času. Také poskytují možnost požádat různé procesy o provedení operace a sdílení údajů.

Typickým příkladem služby je například přehrávání hudby na pozadí. Přehrávání se nastaví v aplikaci na popředí. V této fázi si uživatel prostřednictvím GUI aktivity zvolí hudbu, kterou chce přehrát. V době kdy hudba začne hrát, se uživatel může přepnout do jiné aplikace a plně ji využívat, aniž by se přehrávání hudby přerušilo (Allen, *2013*).

3.3.5.3 Broadcast Receivers

Objekty na vysílání a přijímání poslouchají na pozadí a reagují na události, které se odehrávají na zařízení. Broadcast Receivers fungují na principu publish/subscribe. Události jsou zastoupeny objekty typu Intent (záměr). Vydavatelé vytvářejí záměry a následně je přes Broadcast směřují do vysílání. Zachytávají je přijímače, které mají příslušné záměry objednané nebo registrované (*Lacko, 2015*).

3.3.5.4 Poskytovatelé obsahu (Content providers)

Poskytovatelé obsahu poskytují možnost ukládat a sdílet data mezi více aplikacemi a procesy. Aplikaci je tak umožněno přistupovat k datům jiných aplikací, které vystupují jako poskytovatelé obsahu. Těmi může být cokoliv, od místních databází přes webové kanály až po složitější varianty (Allen, *2013*).

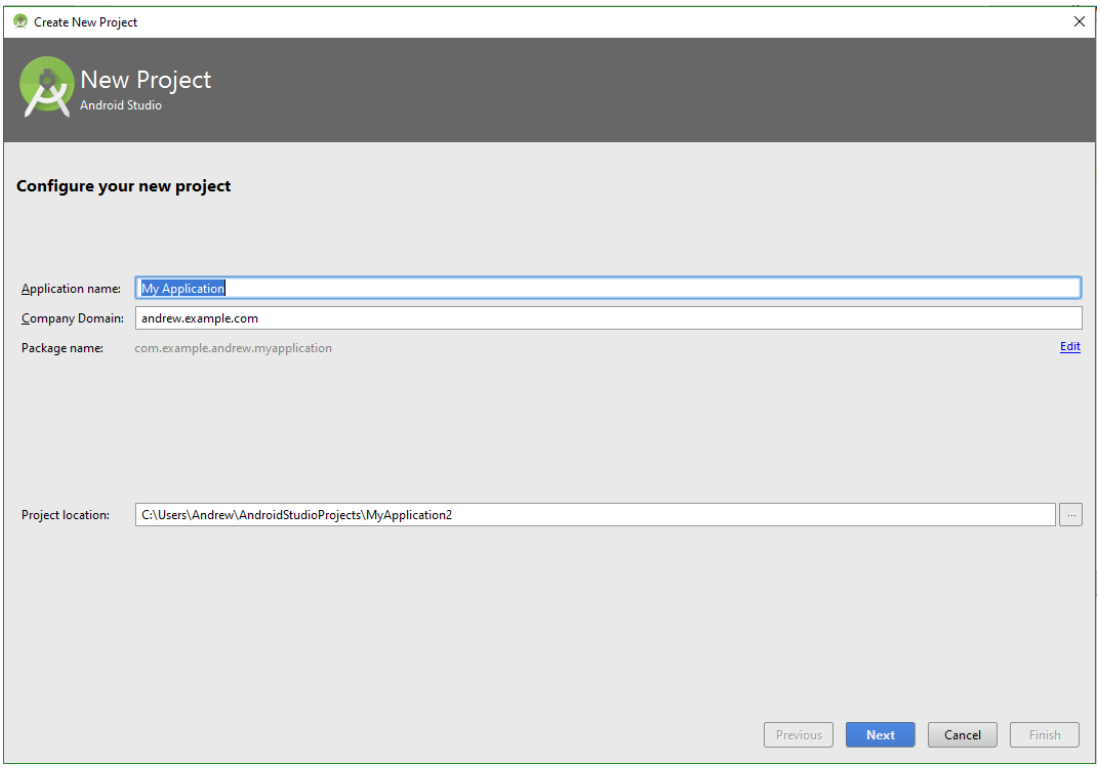
3.3.6 Android studio

Jelikož pro vývoj cílové aplikace bylo vybráno prostředí Android Studio, je tato kapitola věnováná seznámení se s tímto programem.

Android Studio je poměrně nové. Představeno bylo v roce 2013 firmou Google a je k dispozici zdarma.

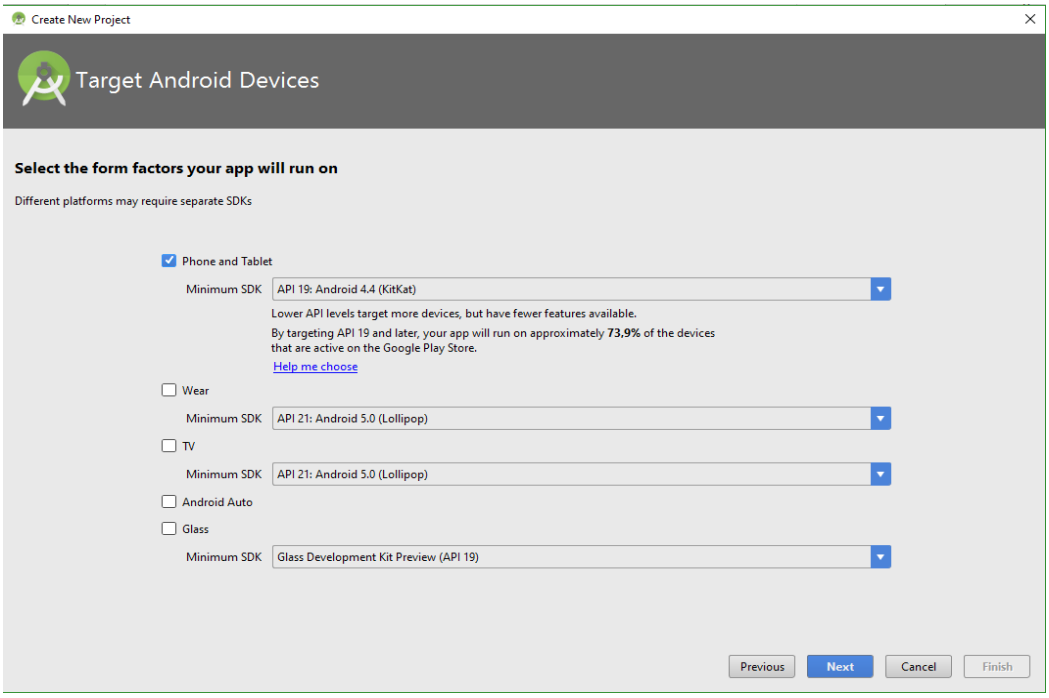
Jeho instalace je poměrně jednoduchá. Stačí si stáhnout SDK a instalační balíček s Android Studiem a spustit instalaci. Balíček obsahuje všechny důležité části. Těmi jsou Android Studio IDE, Android SDK Tools, kompilátor a základní emulátory.

Po instalaci je Android Studio připraveno k okamžitému použití. Stačí prostředí spustit a založit nový projekt. Pro ulehčení se navíc po spuštění uživateli zobrazí okno s nápovědou.



Obrázek 9: Založení nového projektu (vlastní zpracování)

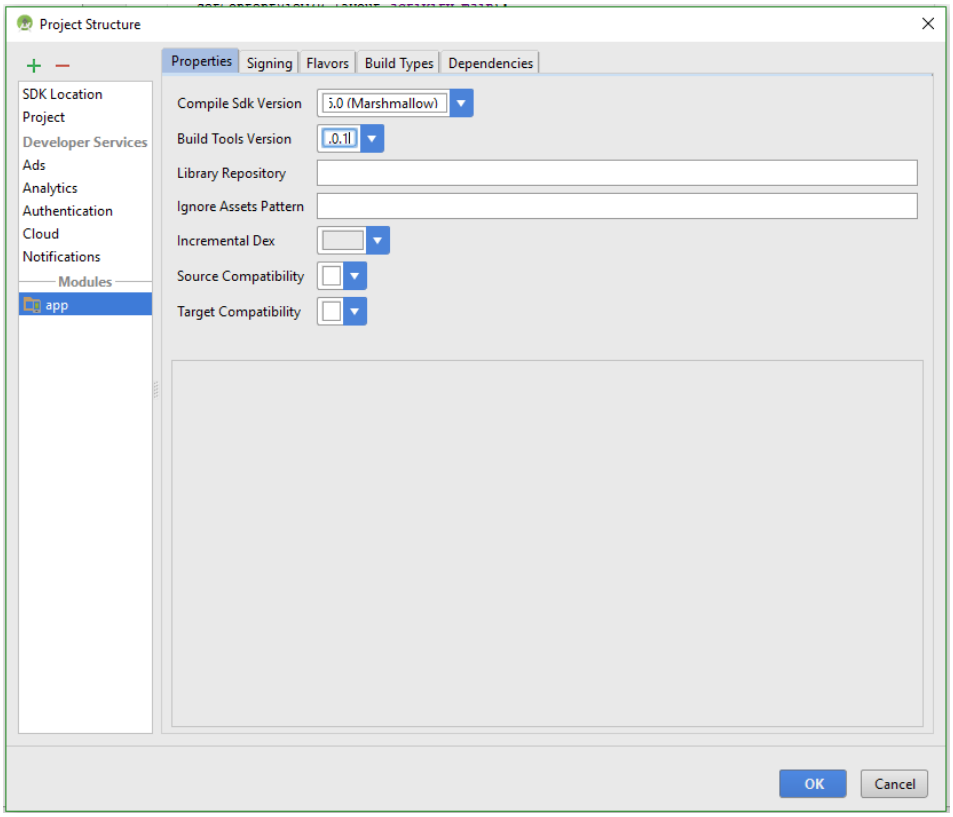
Při zakládání si uživatel určí, pro jaké zařízení a jakou verzi bude aplikaci vytvářet. To je důležité, jak bylo v bodě 3.3.3 uvedeno, jelikož to uživateli určuje, co všechno může využít.



Obrázek 10: Výběr verze a zařízení (vlastní zpracování)

Nakonec si uživatel zvolí, jestli chce nechat automaticky vytvořit první aktivitu, popřípadě jakého typu. Tímto je projekt založen a je možné začít programovat.

K testování funkčnosti aplikace jsou k dispozici dvě metody. První je využití takzvaného emulátoru, který na počítači vytvoří virtuální stroj, reprezentující reálné zařízení. Parametry zařízení si uživatel může nadefinovat sám. Tento způsob je ovšem závislý na hardwaru počítače, na kterém je aplikace vyvíjena. Například na některých počítačích s procesorem AMD nelze virtuální zařízení vůbec spustit. Také je to často vůči druhému způsobu pomalejší. Druhý způsob je využití reálného zařízení. K tomu většinou stačí zařízení připojit pomocí USB, povolit na něm ve složce možnosti pro vývojaře ladění USB, v Android Studiu aplikaci spustit a nakonec vybrat zařízení ze seznamu. Zde bych podotknul, že může dojít k situaci, kdy při použití druhého způsobu nebude aplikace spuštěna. V takovém případě je nutné změnit v File - Project Structure – app Build Tools Version na jinou hodnotu, třeba 23.0.3.



Obrázek 11: Změna Build Tools Version (vlastní zpracování)

3.3 SQLite

SQLite je open source relační databáze. Od ostatních databází se liší tím, že je to pouze malá knihovna, která se připojí k aplikaci a pomocí jednoduchého rozhraní ji lze použít.

Jako i v ostatních databázích jsou zde výjimky v implementaci normy SQL. Klauzule check určující omezení sloupců je vyhodnocena správně, ale využije se jen omezení unique a not null. Vnořené dotazy musí být statické. Cizí klíče jsou k dispozici, ale nejsou vyžadovány výhrady, které by měly být na ně uplatněny. K dispozici není také alter table a změny tabulek se tak musí provádět jejich opětovným založením. Outer join může být použit pouze ve formě left (*SQLite – ultralehké SQL*).

3.4 Java

Jelikož je programovací jazyk Java hojně využíván k tvorbě aplikací pro OS Android, je vhodné si o něm něco málo říci.

Programovací jazyk Java je jazyk vyšší úrovně, který lze popsat slovy jako jednoduchý, objektově orientovaný, přenostitelný, robustní, dynamický a zabezpečený (*Zakhour, 2007*).

To, že je Java velmi oblíbená lze prokázat tím, že se s ní dnes můžeme setkat téměř na každém zařízení, ať už se jedná o PC, mobilní telefon nebo domácí spotřebič.

3.5 XML

Značkovací jazyk XML byl vyvinut konsorciem a standartizován konsorciem W3C. Tento jazyk lze využít jako výměnný formát dat, a lze ho i považovat za nový databázový model (*Mlýnková, 2008*).

Syntaxe XML je založena na *elementech*. Každý element je v dokumentu označen značkou, neboli takzvaným *tagem*. Většina elementů se skládá z dvou tagů, počátečního a koncového. Každý *tag* se zapisuje do špičatých závorek < > (*Kosek, 2009*).

4 Vlastní práce

4.1 Návrh aplikace

Cílová aplikace je vyvíjena pro verzi KitKat. Tato verze byla vybrána z důvodu velkého podílu nasazených verzí na zařízeních na trhu v době vývoje a zároveň pro potřebnou podporou pro jednotlivé komponenty, které jsou v projektu využívány.

Celkově je aplikace použitelná na více jak 86% zařízeních na trhu, viz obrázek č. 7 v kapitole 3.3.3.

Návrh aplikace řeší nedostatky nalezené v aplikacích uvedených v předchozí kapitole. Z analýzy vyplývá následující řešení.

Navržená aplikace poskytuje uživately možnost najít cvik v databázi dle jednoho ze dvou kritérií a to dle svalových partií, nebo dle cvičebního náčiní. Uživatel tak získá seznam cviků, které bude moci provádět pro procvičení vybraných svalů, nebo za využití daného náčiní. Uživateli je pak nabídnut grafický i textový popis vybraného cviku. Grafický popis zastupuje animovaný obrázek, který je dostupný bez potřeby internetového připojení.

V rámci návrhu vlastního cvičebního plánu aplikace umožní přidávat, editovat a mazat záznamy. Každý záznam pak umožní přidávat nebo mazat cviky a pauzy. Výběr přidávaného cviku bude probíhat stejně, jako výběr cviku z databáze, až na to, že na konci si uživatel vybere typ měřiče a nastaví jeho hodnotu. U pauzy je k dispozici pouze časové měření. Spustit plán jde od libovolného cviku. Aplikace následně automaticky spouští další cviky nebo pauzy, které jsou další v pořadí. Pokud je na řadě cvik, tak se uživateli zobrazí obrazovka s údaji, jak cvik správně provést a tlačítkem, které spustí nastavený měřič.

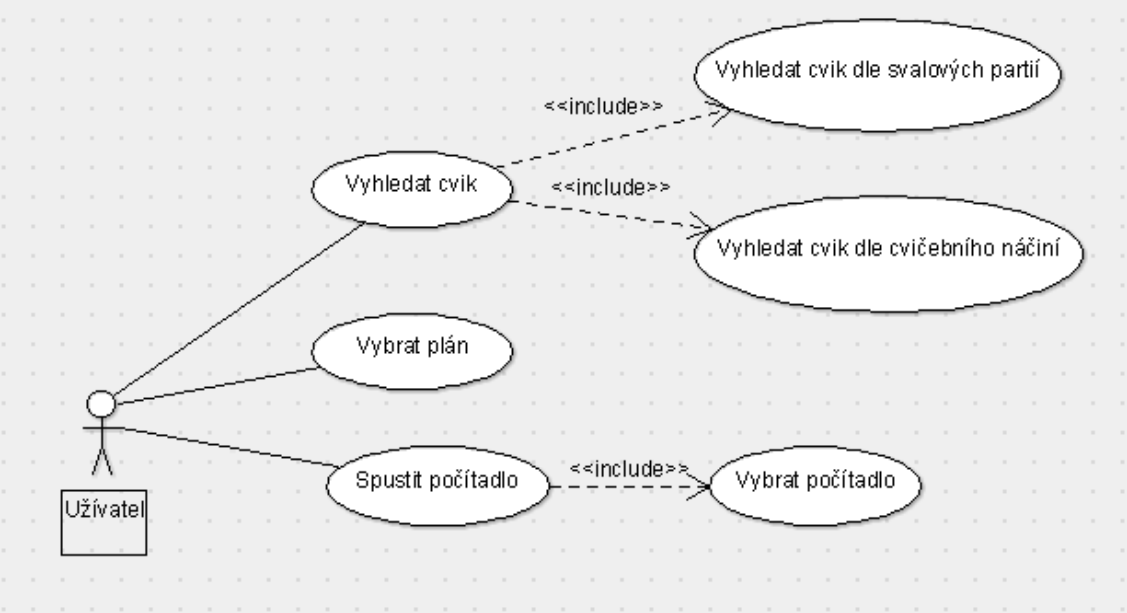
K měření cviků slouží tři způsoby měření. První způsob je časový. Uživatel si zvolí daný čas, a ten bude následně odměřen pomocí odpočtu. Druhý způsob je pomocí doteku. Uživatel zvolí kolikrát chce cvik provést, a následně se při každém provedení dotkne displeje, což způsobí odečtení jednoho cviku z dané hodnoty. Třetí možností bude

polohové počítadlo. Uživatel si opět zvolí kolikrát bude chtít daný cvik provést. Počítadlo pak pokaždé zaznamená, když zařízení změní polohu o 90° a připočte jeden cvik.

K informování uživatele, že téměř odcvičil nastavené množství cviků, nebo že se nastavená doba cvičení blíží ke konci, slouží kromě grafického znázornění také zvuková signalizace. Tu je možné zapnout nebo vypnout.

4.1.1 USE CASE

Níže přiložený obrázek obsahuje USE CASE diagram, znázorňující pouze část funkcionality.



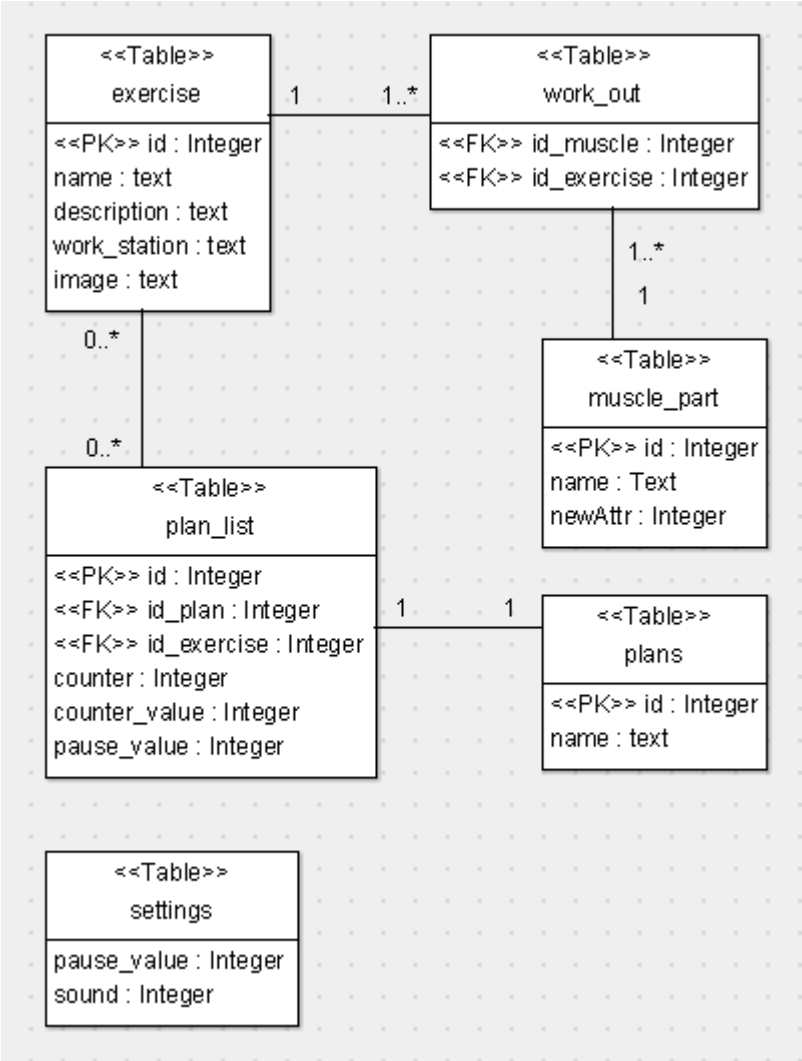
Obrázek 12: USE CASE diagram (vlastní zpracování)

4.2 Datová struktura

Tato kapitola se zabývá datovým modelem aplikace.

4.2.1 UML diagram

UML diagram níže zobrazuje, jak je strukturovaná databáze, kterou aplikace využívá.



Obrázek 13: Datový model (vlastní zpracování)

4.2.2 Popis dat

Databáze obsahuje celkem 6 tabulek. Pět tabulek obsahuje data o cvicích a plánech, a šestá je určená k uchovávání dat týkajících se nastavení aplikace.

První tabulka s názvem exercise obsahuje následující data. Id typu Integer obsahuje identifikační číslo cviku a je zároveň primárním klíčem této tabulky. Name, jak již samotný název napovídá, nese název cviku a je tudíž typu Text. Description typu Text obsahuje textový popis cviku. Pod work_station typu Text se nachází cvičební náčiní, které je k provedení cviku nutné. Jako poslední je image, které nese název obrázku, který popisuje, jak se má cvik správně provést.

Druhá tabulka nesoucí název muscle_part obsahuje data o tom, jaké svaly daný cvik procvičuje. Id typu Integer obsahuje identifikační číslo záznamu a je zároveň primárním klíčem této tabulky. Name typu Text nese název svalové partie.

Třetí tabulkou je tabulka work_out. Id_exercise typu Integer je identifikační číslo cviku a slouží k propojení s tabulkou exercise. Id_muscle typu Integer je identifikační číslo svalové partie. Tato tabulka je vytvořena proto, aby bylo možné ke každému cviku přiřadit více procvičovaných svalových partií. Tudíž, jeden záznam v tabulce exercise může být spojen s více záznamy z tabulky work_out, minimálně by však měl být spojen alespoň s jedním. Stejně tak platí, že jeden záznam z tabulky muscle_part může být spojen s jedním nebo více záznamy v tabulce work_out.

Tabulka číslo čtyři nese název plan_list. Jejím obsahem jsou data týkající se jednotlivých cviků v cvičebním plánu. Id je typu Integer a zastupuje identifikační číslo záznamu. Je také použito jako primární klíč. Id_plan reprezentuje identifikační číslo plánu a obsahuje hodnotu typu Integer. Obdobně je na tom id_exercise, až na to, že zastupuje konkrétní cvik. Pomocí těchto dvou hodnot je záznam z tabulky propojen s odpovídajícími záznamy z tabulek exercise a plans. Counter typu Integer reprezentuje počítadla. Může nabývat čtyř hodnot. Pokud je hodnota 0, tak je na dané pozici v plánu nastavena pauza. Když je hodnota 1, je nastaveno polohové počítadlo. Pokud je 2, tak je nastaveno dotekové počítadlo, a když je hodnota 3, tak je vybráno počítadlo časové. S tím souvisí následující dvě hodnoty a to counter_value a pause_value. Tyto hodnoty reprezentují počáteční hodnoty počítadel. Zároveň platí, že pokud je jedna hodnota nenulová, tak druhá nulová je.

Pátou tabulkou je již zmíněná tabulka s názvem plans. Ta obsahuje pouze id jako identifikační číslo záznamu a name, které reprezentuje název jednotlivých cvičebních plánů.

Poslední tabulka je pojmenovaná settings a nese informace o defaultní hodnotě pauzy a stavu vypnutí/zapnutí zvukové signalizace.

4.3 Layouty

Tato podkapitola se věnuje popisu návrhu a následné tvorby layoutu, který je využit v aplikaci.

4.3.1 Návrh layoutu

Zde je uveden návrh layoutu pro aktivity *Plans* a *Plans_List*. Layout obsahuje ListView pro zobrazení seznamu a dvě tlačítka. První tlačítko slouží k přidání položky do seznamu a druhé pro odebrání položky ze seznamu. ListView zabírát zhruba horních 80% displeje. Tlačítko pro přidání je umístěno dole v levém rohu displeje, jelikož je to tak pro uživatele, který je zvyklí na to, že vlevo jsou tlačítka pro potvrzování nebo přidávání, více intuitivní. Obdobně je to s tlačítkem pro odebrání, které je umístěno vedle prvního tlačítka, tedy dole v pravém rohu displeje.

Obrázek 14: Návrh layoutu pro seznamy (vlastní zpracování)

4.3.2 Vytvoření layoutu

Výše uvedené rozložení je pomocí XML nadefinováno ve složce *layout* v souboru s názvem *activity_plan_list*.

```
<ListView
    android:layout_width="wrap_content"
    android:layout_height="380dp"
    android:id="@+id/listViewPlans"
    android:layout_alignParentTop="true"
    android:layout_alignParentStart="true"
    android:background="#ffffff" />
<Button
    android:layout_width="160dp"
    android:layout_height="80dp"
    android:text="Přidat"
    android:id="@+id/addPlanButton"
    android:layout_alignParentBottom="true"
    android:layout_alignParentStart="true" />
<Button
    android:layout_width="160dp"
    android:layout_height="80dp"
    android:text="SMAŽ"
    android:id="@+id/deletePlanButton"
    android:layout_alignTop="@+id/addPlanButton"
    android:layout_alignParentEnd="true" />
```

Výsledkem výše uvedeného kódu je rozložení odpovídající návrhu layoutu.

Obrázek 15: Výsledný layout (vlastní zpracování)

4.4 Kód

V této podkapitole je uvedono a popsáno několik částí kódu, které jsou součástí vyvíjené aplikace. Nejprve je vždy uvedeno, co v dané části probíhá, nebo k čemu slouží, a pak je přiložen příslušný kus kódu.

4.4.1 Časový odpočet

Aby bylo možné zvolit libovolný čas, který se bude měřit, je nutné nejprve počítadlo nějak nastavit. K tomu slouží aktivita *TimeCounterSetup*. Ta k nastavení využívá 4 instancí objektu *NumberPicker*. Každé této instanci je na začátku přiřazen widget, a poté je nastavena na určité rozmezí hodnot pomocí metod *setMaxValue()* a *setMinValue()*. Nakonec je ještě každé instanci zakázáno, aby se výběr čísel opakoval dokola. Chce se tedy, aby výběr začínal minimální hodnotou a končil maximální. K tomu slouží metoda *setWrapSelectorWheel()*, kterou nastavíme na *false*.

```
m1 = (NumberPicker) findViewById(R.id.m1);
m2 = (NumberPicker) findViewById(R.id.m2);
s1 = (NumberPicker) findViewById(R.id.s1);
s2 = (NumberPicker) findViewById(R.id.s2);
m1.setMaxValue(5);
m1.setMinValue(0);
m1.setWrapSelectorWheel(false);
m2.setMaxValue(9);
m2.setMinValue(0);
m2.setWrapSelectorWheel(false);
s1.setMaxValue(5);
s1.setMinValue(0);
s1.setWrapSelectorWheel(false);
s2.setMaxValue(9);
s2.setMinValue(0);
s2.setWrapSelectorWheel(false);
```

Poté, co jsou instance *NumberPicker* nastaveny, stačí zjistit jejich hodnotu a odeslat ji počítadlu. K tomu slouží vytvořené tlačítko a jeho *OnClickListener*, ve kterém se vypočte celková hodnota převedená na sekundy a předá se následně spuštěné aktivitě *TimeCounter*.

```
int time = (m1.getValue()*10+m2.getValue())*60+
(s1.getValue()*10+s2.getValue());
Intent i = new Intent(TimeCounterSetup.this,
TimeCounter.class);
i.putExtra("time",time);
startActivity(i);
```

Časové počítadlo funguje na principu odpočtu a je pro něj vytvořená aktivita *TImeCounter*. K odpočtu je využit objekt *CountDownTimer*. Při spuštění aktivity je volána metoda *count*(), které je předána hodnota obsahující nastavený čas z aktivity *TimeCounterSetup*. V té je vytvořena nová instance třídy *CountDownTimer* a nastavena počáteční hodnota odpočtu na hodnotu (t+1)*1000, což odpovídá t+1 sekundám. Metoda *onTick()* se pak stará o jednotlivé nastavení aktuálního času po uplynutí jedné sekundy. Nastavení probíhá tak, že se do proměnné „m“ typu *Integer* nejprve vypočte počet minut. Výpočet je jednoduché vydělení aktuální hodnoty času 60 000 milisekundami. Celočíselný datový typ *Integer* zajistí, že je výsledek beze zbytku. Do proměnné „s“ typu long se pak vypočte hodnota zbytku, což odpovídá sekundám. Poté se pomocí metody *setText()* nastavý do *TextView* řetězec složený z aktuálních hodnot minut a sekund oddělené dvojtečkou. Zde je ještě podmínkou *if* ošetřeno, aby se před počet sekund, pokud bude menší jak 10, vložila 0. Nakonec se *CountDownTimer* spustí zavoláním metody *start()*.

```
private void count(int t) {
    new CountDownTimer((t+1)*1000, 1000) {
        public void onTick(long time) {
            int m = (int) time/60000;
            long s = (time/1000)-(m*60);
            if(s<10)timeText.setText(m+":0"+s);
            else timeText.setText(m+":"+s);
            if(time < 4000)alarmBeep.start();
        }
        public void onFinish() {
            alarmBeepFin.start();
            timeText.setText("Hotovo!");
        }
    }.start();
}
```

Když počítadlo dojde na konec nastaveného času, zavolá se metoda OnFinish(), která na displey vypíše „Hotovo“.

4.4.2 Dotekové počítadlo

Podobně jako u časového počítadla je potřeba i zde počítadlo nejprve nastavit na požadovanou hodnotu. K tomu je určena aktivita *CounterSetup*. Ta funguje podobně jako aktivita *TimeCounterSetup* popsaná v předchozí podkapitole, pouze využívá 3 instance objektu *NumberPicker*. Je tedy možné nastavit hodnotu maximálně v řádu stovek. Hlavní rozdíl je pak ve výpočtu nastavené hodnoty, který probíhá v *OnClikListener*.

```
Bundle bun = getIntent().getExtras();
int value =
p1.getValue()*100+p2.getValue()*10+p3.getValue();
if (bun.getInt("type") == 1) {
    Intent i = new Intent(CounterSetup.this,
TouchCounter.class);
    i.putExtra("count",value);
    startActivity(i);
}
```

Pro dotekové počítadlo je vytvořena aktivita *TouchCounter*. Pomocí metody *init()* jsou nejprve nastaveny počáteční hodnoty proměných a přiřazeny widgety. Je zde také předána hodnota, která byla nastavena v aktivitě *TouchCounterSetup* jako požadovaný počet cviků. Ta je předána pomocí *Bundle*, které umožňuje v Androidu přenášet data z jedné aktivity do druhé. Stačí pouze předtím než je druhá aktivita spuštěna přidat k *Intent* pomocí metody *putExtra()* požadovaná data společně s označením, podle kterého jsou v druhé aktivitě opět nalezena. V druhé aktivitě jsou data opět získána pomocí již zmiňovaného *Bundle,* do kterého je pomocí složení metod *getIntent().getExtras()* nastaven přidaný obsah. Nakonec již stačí z *Bundle* zavoláním metod *getInt()* nebo *getString()* získat data daného typu.

```
    protected void init() {
        Bundle b = getIntent().getExtras();
        count = b.getInt("count",0);
        startPoint = b.getInt("count",0);
        view = (Button) findViewById(R.id.counterView);
        alarmBeep = MediaPlayer.create(this,  R.raw.beep01);
        view.setOnClickListener(clickListener);
        if(b.getInt("count") == 0){
            view.setText("Ale no tak!");
        }
        else view.setText("Začínáme na: "+count);
    }
```

Celý princip počítadla pak spočívá v jednoduchém použití tlačítka, které je roztaženo přes celý displej. Původně bylo v plánu využít monitorování dotyku displeje, ale docházelo k tomu, že bylo detekováno několik doteků naráz. Metoda s tlačítkem, efektivně a jednoduše vyřešila to, že se vždy započítá pouze jeden dotek a další je zaznamenán až poté, co se uživatel od displeje oddálí. V *OnClickListener*, který se o detekcy stará, pak probíhá jednoduché odčítání vykonaných cviků od počáteční hodnoty a jejich vypsání na displej. Pokud se navíc jedná o počítadlo použité v plánu, tak je po dosažení hodnoty 0 aktivita ukončena a aplikace se vrátí na předchozí aktivitu, ze které je pak automaticky spuštěn další cvik nebo pauza v pořadí.

```
View.OnClickListener clickListener = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Bundle b = getIntent().getExtras();
        if(view.getText().equals("Great") ||
view.getText().equals("Ale no tak!"))finish();
        if(count >= 1){
            --count;
            view.setText("Zbývá: "+count);
            if(count < 4)alarmBeep.start();
        }
        if(startPoint != 0 && count == 0) {
            view.setText("Great");
            if (b.getInt("plan", 0) == 1) {
                Intent returnIntent = new Intent();
                setResult(Activity.RESULT_OK,returnIntent);
                finish();
            }
        }
    }
};
```

4.4.3 Polohové počítadlo

Polohové počítadlo funguje podobně jako předchozí počítadla. To znamená, že se nejprve nastaví počet cviků, které má počítadlo za úkol odpočítat. K tomu je využita stejná aktivita *CounterSetup* jako u dotekového počítadla. Jestli se má spustit dotekové počítadlo nebo polohové, ovlivňuje hodnota, která je nastavena v předcházející aktivitě *Counter*, sloužící k výběru počítadla. Následně je tato hodnota předána pomocí *Bundle* aktivitě *CounterSetup*, kde se podle její hodnoty rozhodne, které počítadlo má být spuštěno. Hodnota 1 znamená dotekové počítadlo a hodnota 2 počítadlo polohové.

```
if (bun.getInt("type") == 1) {
    Intent i = new Intent(CounterSetup.this, TouchCounter.class);
    i.putExtra("count",value);
    startActivity(i);
}
if (bun.getInt("type") == 2) {
    Intent i = new Intent(CounterSetup.this, PositionCounter.class);
    i.putExtra("value",value);
    startActivity(i);
}
```

Polohové počítadlo funguje na principu detekce změny náklonu mobilního zařízení. K zaznamenání změny je využita instance *OrientationEventListener*, která zaznamenává o kolik stupňů se zařízení nakloní (měří se v rozmezí 0-359 stupňů). Logika zaznamenávání spočívá v detekci průchodu náklonu zařízení určitým rozpětím v první čtvrtině . K tomu, aby se nezapočítával průchod daným rozpětím v obouch směrech, slouží hodnota typu *Boolean*, která udává, zda se zařízení dostává do nové polohy, nebo naopak vrací do původní. Aby bylo ošetřeno, že bude detekováno natočení i v opačném směru, je nastaveno rozpětí i pro čtvrtou čtvrtinu. Zaznamenaný počet je také zobrazován na displeji a je volaná metoda *end()*.

```
public void onOrientationChanged(int degree) {
    if((degree >= 40 && degree <= 50) && up == true ){
        up = false;
        if ((v == count+3 || v == count+2 || v == count+1)
&& beep == false) alarmBeep.start();
        beep = true;
        ++c;
    }
    if((degree >= 60 && degree <= 90)){
        up = true;
        beep = false;
    }
    if((degree >= 310 && degree <= 320) && down == true ){
        down = false;
        ++c;
    }
    if((degree >= 270 && degree <= 300)){
        down = true;
    }
    counterView.setText("Počet: "+c);
    count = c;
    end();
}};
```

Metoda *end()* porovnává, zda dosažená hodnota odcvičených cviků odpovídá nastavené, cílové hodnotě. Pokud ano, tak je počítadlo ukončeno a uživatel se vrátí na předchozí aktivitu.

```
private  void end() {
    Bundle b = getIntent().getExtras();
    if (b.getInt("value") == count){
        Intent returnIntent = new Intent();
        setResult(Activity.RESULT_OK,returnIntent);
        finish();
    }
}
```

4.4.4 Nahrání obrázků

Pro nahrání obrázků v aktivitě *Exercise* slouží metoda *gif()*. Pomocí *Bundle* je z předchozí aktivity předán název obrázku, který odpovídá vybranému cviku. Pak je využita externí knihovna Ion. Instance této knihovny zprostředkovává načítání animovaného obrázku typu GIF. Nejprve se vybere widget, do kterého bude obrázek nahráván. Poté je pomocí *error()*

nastaven neanimovaný obrázek, který je nahrán v případě, že se nepodaří nahrát animovaný gif a následuje nastavení do módu animace. Nakonec je pomocí *load()* vybrán animovaný gif. Všechny obrázky jsou uloženy ve složce s názvem asset.

```
public void gif() {
    Bundle b = getIntent().getExtras();
    String exerciseImage = b.getString("image");
    Ion.with(image)
            .error(R.drawable.cvik)
            .animateGif(AnimateGifMode.ANIMATE)
            .load("file:///android_asset/"+exerciseImage+".gif");
}
```

4.4.5 Zvuková signalizace

K zvukové signalizaci slouží instance objektu *MediaPlayer*, která umožňuje aplikaci přehrát zvukový soubor. Tento soubor je uložen ve složce *raw*. Tuto složku je ale nejprve potřeba vytvořit. To se v AndroidStudiu provede tak, že se klikne pravým tlačítkem myši na adresář *res*. Potom se vybere v záložce *New* možnost *Android Resource Directory*. To otevře okno, ve kterém se vybere v záložce *Resource type* možnost *raw* a potvrdí se kliknutím na tlačítko *Ok*. Když je takto složka raw vytvořena, tak se do ní nahrají vybrané zvukové soubory. V případě této aplikace se jedná o mp3 soubor s názvem *beep01*. V aktivitě je potom jednoduše tento zvukový soubor přiřazen pomocí metody *create()* vytvořené instanci.

```
alarmBeep = MediaPlayer.create(this,  R.raw.beep01);
```

Nakonec se někde v kódu zvukový soubor spustí odstartováním vytvořené instance pomocí metody *start()*.

```
if(count < 4)alarmBeep.start();
```

4.4.6 Plnění seznamů

Seznamy jsou v aplikaci využity na několika místech. Jedná se například o aktivitu pro výběr cviku, nebo aktivitu pro správu plánů. Logika je víceméně všude stejná, takže zde bude prezentována ukázka pouze pro jednu z těchto aktivit a to aktivitu *Plans*.

Nejprve je pomocí metody *nactiDatabase()* načten seznam existujících plánů z tabulky *plans* v databázi. K získání dat a práci s databází slouží pomocná třída *DBHelper*.

```
public void nactiDatabase(){
    myDbHelper = new DBHelper(this);
    list = myDbHelper.getListPlans("SELECT * FROM plans");
}
```

Tímto je tedy k dispozici seznam existujících plánů. Nyní je třeba tento seznam vypsat na displej a nastavit, aby bylo možné se přes každou položku seznamu překliknout dále. K tomu slouží metoda *nacti()*. Zde se nejprve získá seznam obsahující pouze názvy plánů pomocí cyklu *while*. V cyklu se prochází seznam plánů získaný pomocí metody *nactiDatabase()* a z každého plánu je vybrána a uložena hodnota reprezentující název plánu do nového seznamu *aa*. Poté co je seznam s názvy připraven, je pomocí adaptéru vytvořen seznam a ten je díky widgetu ListView zobrazen na dispej. Nakonec se nastaví *setOnItemClikListener*, který obstará co se má stát, pokud se na nějakou položku ze seznamu klikne.

```
public void nacti() {
    ArrayList<String> aa = new ArrayList<String>();
    int i = 0;
    while(i < list.size()){
        aa.add(list.get(i).getName());
        i++;
    }
    ArrayAdapter adapter = new ArrayAdapter(this,
android.R.layout.simple_list_item_1, aa);
    lv.setAdapter(adapter);
    lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
            Intent i = new Intent(Plans.this, PlanList.class);
            i.putExtra("choice",list.get(position).getId());
            startActivity(i);
        }
    });
}
```

5 Výsledky a diskuse

5.1 Zhodnocení výsledné aplikace

Tato kapitola se věnuje zhodnocení výsledné aplikace. Aplikace byla poskytnuta několika osobám k otestování a získání zpětné vazby.

Klady

Aplikace se vyznačuje především svými počítadly, které se starají o hlídání odcvičených cviků. Pro větší pohodlí uživatele a lepší praktické využití je aplikace vybavena zvukovým ohlášením, že se blíží konec počtu nastavených cviků. Tato zvuková signalizace se spouští tři poslední cviky před koncem, nebo v případě časového počítadla při třech posledních sekundách.

Nedostatky

Mezi nedostatky aplikace spadá občasné, špatné počítání polohového počítadla. To je způsobeno rychlým přechodem z jedné polohy do druhé, při kterém dojde k přeskočení daného rozsahu stupňů, kde probíhá započítání cviku. Tato chyba se ovšem vyskytuje při cvičení jen zřídka.

5.2 Další možný rozvoj

V budoucnosti by aplikace mohla být rozšířena především o určitou zpětnou vazbu, která by poskytovala uživateli přehled o dosavadních výsledcích jeho snažení. Tuto funkci obsahuje v současnosti velké množství jiných aplikací pro správu fitness tréninku, a tak by tím bylo dosaženo větší konkurence schopnosti výsledné aplikace.

Dále by se pak mohla přidat možnost přidávat do galerie své vlastní cviky. Jelikož by to obnášelo i nutnost přidání možnosti odebrání cviku z galerie, bylo by dobré zamezit těmto úpravám na původní sadě cviků. Zde by se také muselo vyřešit přidání animovaného obrázku.

Bylo by také možné ve spolupráci s grafikem upravit GUI tak, aby bylo více vzhledově a i funkčně příjemné. Zde by se dala také přidat možnost výběru designu aplikace.

6 Závěr

Byly probrány teoretické základní znalosti, nutné k pochopení vývoje aplikace pro OS Android. V kapitole „Teoretické části“ byly probrány jednotlivé základní komponenty programu, životní cyklus aktivity, základy OS Android a některé jeho verze. Uvedené znalosti byly využity při tvorbě aplikace, která byla hlavním cílem bakalářské práce.

Výsledná aplikace splňuje dané požadavky, které byly vyvozené z potřeby autora a z následné analýzy dostupných aplikací podobného typu na trhu. Aplikace umožňuje uživateli vyhledat cvik z galerie dle dvou kritérií, a to buď dle svalových partií nebo dle cvičebního náčiní. Dále je možné spravovat jednotlivé cvičební plány. Aplikace asi nejvíce vyčnívá z řady fitness trenérů dostupných v současné době na trhu svými počítadly, které jsou u ostatních aplikací povětšinou pouze ve formě odpočtu, a nebo nejsou k dispozici žádné.

7 Seznam použitých zdrojů

Murphy, M. L. (2011). *Android 2.* Brno: Computer Press.

Zackhour, S. (2007). *Java 6.* Brno: Computer Press.

Allen, G. (2013). *Android 4.* Brno: Computer Press.

Lacko, L. (2015). *Vývoj aplikací pro Android*. Brno: Computer Press.

*Verze***.** (28. Leden 2017). Načteno ze Svět Androida: https://www.svetandroida.cz/historie-androidu-201506

JetFit Foto. (19. Únor 2017). Načteno z 5 nejlepších fitness aplikací: http://holmesplace.cz/cs/5-nejlepsich-fitness-aplikaci-a2100.html

Fitness and Bodybuilding Foto. (19. Únor 2017). Načteno z Google Play: https://play.google.com/store/apps/details?id=softin.my.fast.fitness

7 minute Foto. (19. Únor 2017). Načteno z Android Tip: http://www.androidtip.cz/tri-uzitecne-android-aplikace-pro-vase-zdravi/

Homo Workouts Foto. (19. Únor 2017). Načteno z Apkmonk: http://www.apkmonk.com/app/com.fitness22.bodyweightworkout/

API. (19. Únor 2017). Načteno ze Android Market: http://androidmarket.cz/android/jak-se-vyvijel-operacni-system-android-napric-jeho-historii-az-do-soucasnosti-2-dil/

Vývoj pro Android komponenty. (19. Únor 2017). Načteno ze Zdroják: https://www.zdrojak.cz/clanky/vyvoj-pro-android-ii/

Android (Market Share). (1. Březen 2017). Načteno z Developer Android: https://developer.android.com/about/dashboards/index.html

Android (API). (1. Březen 2017). Načteno z Developer Android: https://developer.android.com/guide/topics/manifest/uses-sdk-element.html

SQLite – ultralehké SQL. (8. Únor 2017). Načteno z Root: https://www.root.cz/clanky/sqlite-ultra-lehke-sql/

Vyvijime pro Android Activity. (5. Únor 2017). Načteno ze Zdroják: https://www.zdrojak.cz/clanky/vyvijime-pro-android-prvni-krucky/

Vyvíjíme pro Android Fragmenty. (5. Únor 2017). Načteno ze Zdroják: https://www.zdrojak.cz/clanky/dej-androidu-tablety/

Mlýnková, I. (2008). *XML technologie*. Praha: Grada Publishing.

Kosek, J. (2009). PHP a XML. Praha: Grada Publishing.

IT network. (8. Únor 2017). Načteno z IT network: http://www.itnetwork.cz/java/android/tutorial-programovani-pro-android-v-jave-zivotni-cyklus-a-novy-projekt

8 Seznam obrázků

9 Seznam Tabulek

10 Přílohy

Příloha 1: CD se zdrojovými kódy