Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj mobilní aplikace pro pohybové aktivity pro OS Android

Vypracoval: František Veselý
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2021

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**František VESELÝ**
Osobní číslo:	**E18576**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Vývoj mobilní aplikace pro pohybové aktivity pro OS Android**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem práce je vytvořit mobilní aplikaci a provést její distribuci pomocí elektronického obchodu s aplikacemi. Aplikace bude využívána při pohybových aktivitách uživatele a bude využívat dostupné senzory mobilního zařízení (např. proximity senzor, akcelerometr, gyroskop). Aplikace bude shromažďovat data v průběhu pohybové aktivity uživatele a bude schopna získaná data dále vizuálně zpracovat a porovnat s historickými daty uživatele.

Metodický postup:

1. Studium odborné literatury.
2. Návrh a popis vývoje a implementace výsledné aplikace, umístění do veřejnosti přístupného e-shopu.
3. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
4. Vypracování doporučení a závěrů.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Ferrone, H. (2019). *Learning C# by Developing Games with Unity 2019.* (Fourth Edition). Birmingham, UK: Packt.
2. Hagos, T. (2018). *Learn Android Studio 3: Efficient Android App Development.* New York, NY: Apress.
3. Lake, I., & Meier, R. (2018). *Professional Android.* 4th Edition. Indianapolis, Indiana (USA): Wrox.
4. Marsicano, K., Stewart, C., & Phillips, B. (2019). *Android Programming: The Big Nerd Ranch Guide.* 4th Edition. Atlanta, GA (USA): Big Nerd Ranch.
5. Mishra, S. M. (2015). *Wearable Android: Android Wear and Google FIT App Development.* Hoboken, NJ: John Wiley & Sons.
6. Price, M. J. (2019). *C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development.* Birmingham, UK: Packt.
7. Troelsen, A., & Japikse, P. (2017). *Pro C# 7: With .NET and .NET Core.* New York, USA: Apress.

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **17. ledna 2020**
Termín odevzdání bakalářské práce: **16. dubna 2021**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13 (26)
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 25. března 2020

Prohlášení

Prohlašuji, že svou bakalářskou práci „Vývoj mobilní aplikace pro pohybové aktivity pro OS Android“ jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to - v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou - elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

…… ……

Datum Podpis

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Mgr. Radimu Remešovi, za cenné rady a připomínky při vypracovávání mé bakalářské práce. Dále bych chtěl poděkovat celé rodině, která mi poskytovala podporu během studia.

Obsah práce

1 Úvod bakalářské práce

Vývoj mobilních aplikací je možný pomocí několika programů a jazyků. Pro tuto bakalářskou práci byl zvolen nejběžnější postup při tvorbě aplikace pro operační systém Android, kterým je jazyk Java a pomocí aplikace Android Studio. Jedná se o program, který je přímo určený pro tvorbu aplikací na různá zařízení, která používají Android jako svůj operační systém.

Cílem této bakalářské práce je vytvoření mobilní aplikace pro pohybové aktivity, která bude využívat senzory vestavené ve většině mobilních telefonů. Aplikace pro určité pohybové aktivity využívá akcelerometr a pro určité aktivity senzor přiblížení. Zároveň aplikace umí vykreslit postupy v každé pohybové aktivitě. Nakonec by aplikace měla být uveřejněna a volně k dispozici v internetovém obchodě Google Play.

V teoretické části je historie a popis operačního systému Android. Následně se v první části věnuji firmě, která tento operační systém vytváří. Jelikož aplikace je vytvořena v Android Studiu, tak se věnuji i této aplikaci, včetně popisu jazyka Java.

Praktická část bakalářské práce obsahuje popis vývoje aplikace pro mobilní zařízení běžící na operačním systému Android pomocí aplikace Android Studio. Je zde popsáno vytvoření uživatelského rozhraní a následně propojení s logikou aplikace. Dále popis naprogramování samotné logiky, ukládání dat až po propojení s hardwarovými senzory mobilního zařízení.

2 Android

Jedná se o nejrozšířenější operační systém pro mobilní zařízení. Je na trhu od roku 2008 a jeho základem je upravená verze linuxového jádra. (Mobilmania.cz, 2020)

Pro vývoj aplikací na tento operační systém se využívá programovacího jazyka Java a systému knihoven Android. Základní struktura odpovídá běžnému vývoji v tomto jazyce. Programovat aplikaci na Android lze na zařízeních používající kterýkoliv rozšířený operační systém. Pro takový vývoj existuje několik vývojových prostředí, nicméně v této bakalářské práci jsem zvolil nástroj Android Studio, který je vyvinutý přímo vývojářem systému Android.

2.1 Historie Android

Psal se rok 2003, když vznikla společnost Android Inc., kterou založili Andy Rubin, Rich Miner, Nick Sears a Chris White. Vzápětí byla tato společnost v roce 2005 odkoupena společností Google Inc. a díky správnému odhadu udělala z Android Inc. svou dceřinou společnost.

Ihned po přesunu pod křídla Google Inc. byl do vedení dosazen jeden z jejích zakladatelů, Andy Rubin, díky kterému se posouval vývoj tohoto operačního systému a díky kterému později Google vstoupil na trh s chytrými zařízeními. V roce 2007 nejen, že Google získává patenty ohledně mobilních zařízení, ale také vzniká společnost Open Handset Alliance, která seskupuje výrobce mobilních telefonů. Členy jsou společnosti Google, Samsung, Intel, Qualcomm, Sony a mnoho dalších společností, které se pohybují v odvětví chytrých telefonů. (Mysliveček, 2013)

Po přibližně jednom roce od vzniku společnosti OHA byl uveden na trh první chytrý telefon s operačním systémem Android. Jednalo se o HTC Dream, který byl od roku 2009 prodáván i v České republice. Od té doby dochází k neustálému vylepšování systému Android.

2.2 Verze Android

Každá verze operačního systému Android se pojmenovala od verze 1.5 podle amerických sladkostí až do verze 10, která již nemá žádný přívlastek, ale pouze číselné označení. (-fs, 2020) Android je vyvíjen především pro dotyková zařízení, ale nejde jen o mobilní telefony, příkladem může být televize nebo chytré hodinky. V následující tabulce 1 je přehled dosud vydaných verzí operačního systému Android, včetně názvu a verze API[1].

Tabulka 1 Verze OS Android

Verze OS Android k 1.2.2021			
Verze	**Název**	**API**	**Rok vydání**
1.0 - 1.1	Base	1, 2	2008, 2009
1.5	Cupcake	3	2009
1.6	Donut	4	2009
2.0 - 2.1	Eclair	5 - 7	2009, 2010
2.2	Froyo	8	2010
2.3	Gingerbread	9, 10	2010, 2011
3.0 - 3.2	Honeycomb	11 - 13	2011
4.0	Ice Cream Sandwich	14, 15	2011
4.1 - 4.3	Jelly Bean	16 - 18	2012, 2013
4.4	KitKat	19	2013
5.0 - 5.1	Lollipop	21, 22	2014, 2015
6.0	Marshmallow	23	2015
7.0 - 7.1	Nougat	24, 25	2016
8.0 - 8.1	Oreo	26, 27	2017
9.0	Pie	28	2018
10.0	Q	29	2019
11.0	Android 11 (R)	30	2020

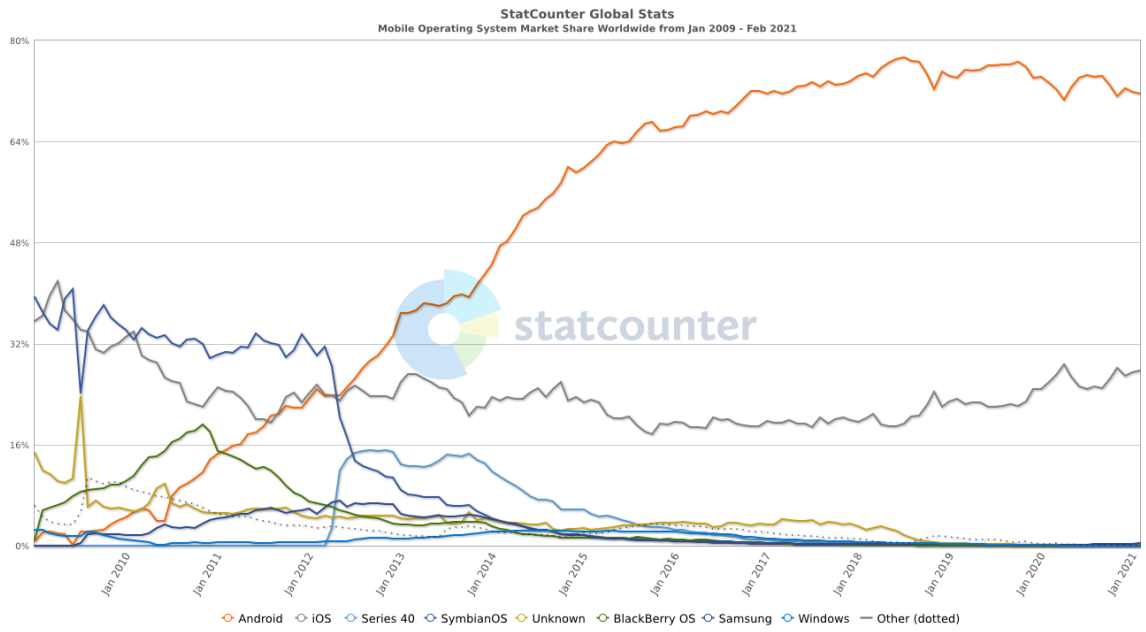
Zdroj: (David Ortinau, 2018) a (developers, Android Studio, 2020)

[1] API je úroveň rozhraní, která obsahuje soubory nových funkcí, procedur, protokolů a knihoven.

2.3 Podíl mobilních operačních systémů

Aktuální stav na trhu s mobilními operačními systémy je jednoznačný. Celosvětově fungují pouze dva, z toho jeden je velice uzavřený a používá ho pouze jeden výrobce mobilních telefonů. Android používá drtivá většina výrobců a díky tomu je mnohem více rozšířený, jak je možné zpozorovat na obrázku 1. Díky tomu má aplikace vyšší šanci, že zaujme více uživatelů. Aktuální situace ani nenaznačuje vstup nějakého nového operačního systému na celosvětový trh.

Obrázek 1 Zastoupení mobilních operačních systémů

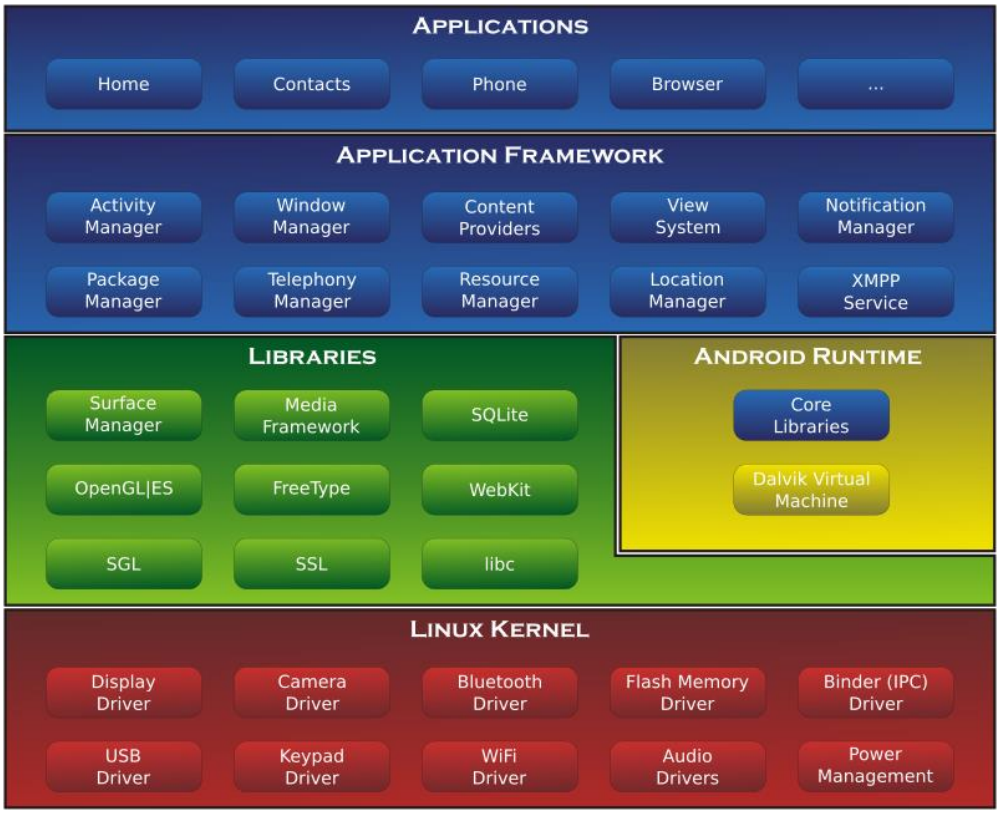


Zdroj: (Statcounter, 2021)

2.4 Architektura Android

Architektura systému Android je tvořena 5 samostatnými vrstvami, které navzájem spolupracují.

Obrázek 2 Architektura Android



Zdroj: (Burton, 2015)

2.4.1 Linux Kernel

Linux Kernel je jádro a tím i nejnižší vrstva architektury operačního systému Android. Jedná se o vrstvu mezi softwarem ve vyšších vrstvách a používaným hardwarem. Systém Android je postaven na základech Linuxu verze 2.6 a díky tomu využívá jeho vlastnosti, které jsou popsané na obrázku 2 výše. Lze vyzdvihnout, že díky tomuto jádru dokáže Android zajistit běh více aplikací najednou, používá zabudované ovladače a zajišťuje zabezpečení.

2.4.2 Libraries

V českém jazyce knihovny, jsou napsané v jazyce C a C++. Slouží k zajištění základních funkcí systému a lze je využít při vytváření aplikace. Z mnoha knihoven zde rád zmíním například android.database, která v sobě obsahuje i SQLite, což je odlehčená verze databáze například pro mobilní zařízení. Další zajímavou knihovnou může být android.view, díky které je možné vytvářet funkční prostředí aplikace a posledním

příkladem může být android.widget, která umožňuje rozvíjet uživatelské prostředí o různá tlačítka, seznamy a jiné vychytávky.

2.4.3 Android Runtime

Android Runtime je složen z knihoven programovacího jazyka Java a Android RunTime, Android 4.x a starší verze používají virtuální stroj Dalvik Virtual Machine. Jeho hlavní role v architektuře operačního systému Android je překlad aplikace, jelikož jazyk Java není nativní jazyk pro Android.

2.4.4 Application Framework

Application Framework je tedy nejdůležitější vrstva pro vývojáře aplikací pro zařízení s operačním systémem Android. Nachází se zde mnoho služeb a knihoven napsaných v jazyce Java a zároveň usnadňují vývojářům přístup i práci. Activity Manager kontroluje životní cyklus aplikace, Notification Manager umožní uživatele upozorňovat pomocí vyskakovacích oken a v systému Views lze vytvářet jednotlivé obrazovky aplikace s využitím tlačítek, seznamů a jejich rozložení. Tato aplikační vrstva mi také umožní přístup k senzorům mobilního zařízení potřebných pro tuto práci.

2.4.5 Applications

Případně Basic Applications je nejvyšší vrstva architektury a jedná se o aplikace, které používají koncoví uživatelé. Některé aplikace jsou nativní a jiné od vývojářů třetích stran, přičemž platí, že aplikaci je možné stáhnout přes oficiální obchody (Google Play, AppGallery) nebo pomocí APK souboru. (Swift, 2015) (Burton, 2015)

3 Vývoj aplikace pro Android

Každý vývoj aplikace a v podstatě jakéhokoliv softwaru se dá shrnout do následujících pěti částí.

Konceptualizace je první a nejdůležitější část. Jako první u každého vývoje musí být nápad, či podnět. Je potřeba si ujasnit hlavní myšlenku aplikace. Je na místě, udělat si průzkum, co by měla zahrnovat a případné problémy potencionálních uživatelů.

Druhá etapa je **definice** a v té je důležité si ujasnit funkcionality budoucí aplikace. Neméně podstatné je určit si cílovou skupinu uživatelů, které bude aplikace sloužit. Posledním bodem je vymezení návrhu.

Následuje část **návrhu**, ve které se stávají předem definované věci hmatatelnými. Pod tím si lze představit tvorbu prvních funkčních modelů, které lze testovat s uživateli. Pro tyto účely jsou vytvářeny názorné grafické návrhy aplikace.

Vývoj je část, při které dochází k vytvoření samotné aplikace jako celku. Vzniká první verze a postupně se odhalují chyby a nedostatky v kódu.

Publikace je poslední etapa ve vývoji aplikace. Díky této fázi se k vytvořené a hotové aplikaci dostanou koncoví uživatelé prostřednictvím elektronického obchodu. Po zveřejnění je nutné sledovat reakce uživatelů a případné nedostatky se pokusit vylepšit pomocí budoucích možných aktualizací. (Cuello & Vittone, 2013)

3.1 Základní části aplikace Android

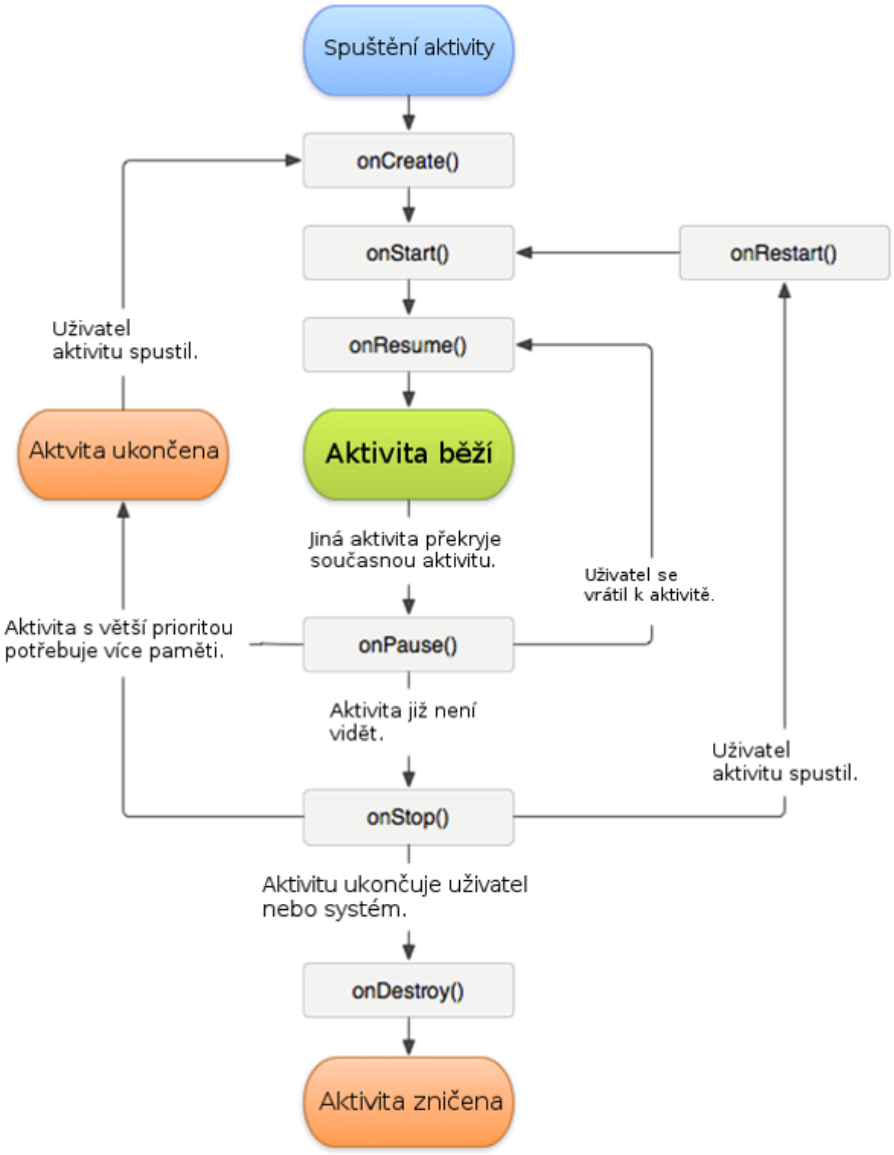
Každá Android aplikace je tvořena čtyřmi základními komponenty, které podrobně rozepíši zvlášť. Každý komponent je vstupní bod, přes který může do aplikace vstupovat buď uživatel, nebo systém.

3.1.1 Activity

Activity je vstupní bod pro komunikaci s uživatelem. Reprezentuje jednu obrazovku s uživatelským rozhraním, z toho vyplývá, že aplikace má obvykle více aktivit, mezi kterými se uživatel pohybuje. Tato Activity zajišťuje, aby měla aplikace dostatečnou paměť v zařízení, zároveň je možnost předávat si informace napříč jednotlivými obrazovkami. Na obrázku 3 níže je možné vidět životní cyklus jedné aktivity, ze které lze vypozorovat základní stavy. Na začátku je aktivita úspěšně spuštěna a běží (onCreate()). Následně se aktivita může dostat do stavu pauzy (onPause()), aktivita teda pořád běží, ale je překrytá například příchozím hovorem, nebo upozorněním jiné aplikace. Následně,

například při přijetí onoho příchozího hovoru, se aktivita zastaví (onStop()), tudíž k ní uživatel ztratí přístup, ale může se k ní ihned po ukončení hovoru vrátit. Poslední krok je ukončení aktivity (onDestroy()), kdy zanikne. (Frank, 2015)

Obrázek 3 Životní cyklus aktivity



Zdroj: (Frank, 2015)

3.1.2 Service

Nezobrazuje uživatelské rozhraní, jde o proces, který běží na pozadí. Tento proces může být spuštěn z aktivity a nechat ho běžet, nebo s ním komunikovat. Typicky může jít o výpočet nějakých dat, stahování nebo přehrávání hudby. Představuje to různé typy služeb, což ovlivňuje chování operačního systému k nim. Například při přehrávání hudby se informace o skladbě mohou přenést ve formě upozornění do popředí a v tom případě

systém tuto službu neukončí. Při běžné službě na pozadí o ní uživatel ani nemusí vědět, tudíž má systém více možností a při nedostatku operační paměti může tuto službu ukončit.

3.1.3 Broadcast reciever

Broadcast reciever je další komponent, který nemá uživatelské rozhraní. Slouží jako naslouchadlo vzkazů a podle určení s nimi pracuje, například jako oznámení. Je možné doručit informaci i do aplikace, která není spuštěna. Většina tohoto vysílání pochází ze systému a může jít o informaci, že je slabá baterie, vypnula se obrazovka, byla doručena zpráva SMS anebo proběhlo dokončení stahování.

3.1.4 Content Provider

Content Provider je v podstatě aplikační rozhraní pro uchování dat v souboru, databázi, na webu, nebo jiném úložišti, kam má aplikace přístup. Díky tomuto komponentu lze sdílet data s jinými aplikacemi, které je mohou upravovat. Pro příklad nám poslouží přidání fotky k profilu některého z kontaktů v seznamu kontaktů. Aplikace nás ze seznamu kontaktů přepne do aplikace fotoaparátu a následně má seznam kontaktů k dispozici danou fotografii. (developers, Application Fundamentals, 2021)

3.2 Vývojové prostředí

Před jakýmkoliv začátkem je potřeba se rozmyslet, který nástroj bude pro tvorbu aplikace nejvhodnější. S tím souvisí i instalace určitých programů, z některých je možné si zvolit dle vlastní libosti, nicméně některé jsou dané pro daný operační systém. Výhodou systému Android je, že všechny nástroje a kódy jsou k dispozici zdarma na všech počítačových operačních systémech.

3.2.1 Android Studio

Jedná se o vývojové prostředí přímo od Google, které je zdarma k dispozici a nabízí velké množství emulátorů, případně možnost připojit si vlastní zařízení. Android Studio je vytvořené na IntelliJ IDEA, což je vývojové prostředí společnosti JetBrains, která zároveň spolupracuje se společností Google na vývoji, toto prostředí kontroluje kód v reálném čase.

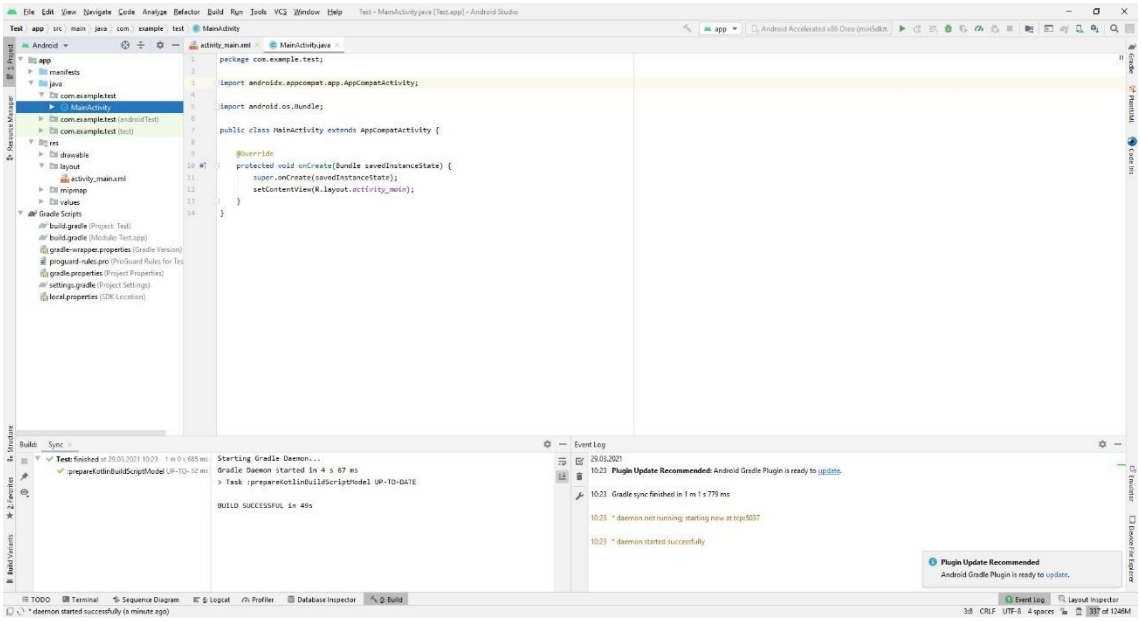
Tato aplikace existuje již od roku 2013 a dochází k neustálým aktualizacím a vylepšením. (developers, Android Studio, 2020) Od zmíněného roku je také dostupná na systémech Windows, Linux a MacOS.

Jedná se o doporučené vývojové prostředí, dříve bylo velice rozšířené prostředí Eclipse, které využívalo plugin Android Developer Tools, jehož vývoj však byl ukončen.

Vývoj v Android Studiu požaduje instalaci balíku Java JDK, který je k dispozici zdarma na oficiálním webu společnosti Oracle. Samotná instalace Android Studia nabídne instalaci sady nástrojů pro vývoj (SDK), kdy každá nová verze se váže k novější verzi systému Android a verzi API a nabídne vyčlenění operační paměti počítače pro emulátor. Pro prvotní kontrolu funkčnosti jsou emulátory praktické, nicméně z důvodu relativně zdlouhavého procesu jsem později instaloval každou verzi rovnou do mobilního zařízení, což je potřeba povolit v Android Studiu, ale i v telefonu.

Na obrázku 4 je vidět vývojové prostředí Android Studio, kromě pravého horního okna, kde je samotný kód, jsou ostatní okna konfigurovatelná a lze tam nastavit různé informace, které jsou pro vývojáře důležité. Vlevo je vidět struktura adresářů aplikace. Lze zde vidět soubory s příponou java, což značí soubor, kde se pro danou aktivitu programuje logika. K takovému souboru patří i soubor s příponou xml, kde se vytváří vzhled oné aktivity. Dále se ve složce values nachází soubor strings.xml, kam se zapisují případné řetězce textu, které má aplikace vypisovat, aby byla možná snadnější případná oprava.

Obrázek 4 Prostředí aplikace Android Studio



Zdroj: Autor

3.2.2 Java

Tento programovací jazyk byl představen na konferenci SunWorld v roce 1995 a byl vyvinut firmou Sun Microsystems. Při představení byl program v jazyce Java prezentován na virtuálním stroji a tím nebyl limitován hardwarem ani operačním systémem počítače. Dnes už to obecně neplatí a převažuje překlad za běhu do nativního jazyka. Při prezentaci byl jazyk prezentován jako jednoduchý, objektově orientovaný, bezpečný, nezávislý na architektuře, portabilní a dalších mnoho přívlastků. (Novotný, 2003) Od roku 2007 je tento jazyk vyvíjen jak open source. Jazyk patří k nejpoužívanějším a nejpopulárnějším. (TIOBE Index for February 2021, 2021)

4 Senzory mobilního zařízení

Každé mobilní zařízení má mnoho různých senzorů a v širším smyslu je možné za senzor považovat jakýkoliv zdroj informací, který předává data. Senzor je tedy převodník, který měří fyzikální veličiny a mění je na elektronický signál, který mobilní zařízení využije k některé činnosti. Příklad za ty nejzákladnější může být klávesnice, mikrofon, displej, nebo fotoaparát v mobilním telefonu. (Škopek, Techbox: váš telefon je prošpikovaný senzory, 2013) Dál se zaměřím pouze na senzory, které u své aplikace využívám.

4.1 Senzor přiblížení

Tento senzor se v anglickém jazyce nazývá proximity sensor, jeho nejideálnější poloha je vedle sluchátka pro hovory. Jakmile uživatel přijme hovor a přiloží mobilní telefon k uchu, tento senzor zaznamená přiblížení objektu, v tomto případě ucha, a displej zhasne, aby nedocházelo k nechtěným dotekům. Díky tomuto čidlu lze nastavit na telefonu ochranu proti odemknutí mobilu v kapse.

Senzor přiblížení funguje na krátkou vzdálenost, maximálně přibližně 10 centimetrů. Senzor je tvořen infradiodou a infra detektorem. Jakmile tedy detektor zaznamená paprsek vyslaný z diody, tak dojde k požadované činnosti, kterou je typicky zhasnutí displeje. Může jít i o jinou technologii, nicméně princip se tím nemění. (Škopek, Techbox: váš telefon je prošpikovaný senzory, 2013) (Chroust & Kužel, 2015)

4.2 Akcelerometr

Jedná se o základní senzor v mobilních telefonech. Tento senzor nám při otočení mobilu na šířku otočí i obraz, při zvednutí telefonu provede automatické probuzení, případně pomocí otřesů počítá kroky vlastníka. Jak už název napovídá, tak měří kromě orientace telefonu i zrychlení, a to ve třech osách vůči zemi.

V chytrých telefonech se nachází akcelerometr, který pracuje na principu piezoelektrického jevu. Zkráceně lze říci, že jde o krystal, který umí generovat elektrické napětí při jeho deformaci. Takové elektrické napětí lze měřit, jelikož krystal se nedeformuje a náboj je díky tomu úměrný síle zrychlení. (Chroust & Kužel, 2015)

4.3 Gyroskop

Poslední, ale neméně důležitý senzor bývá někdy označován spolu s akcelerometrem jako pohybový senzor, ale i přes spolupráci těchto senzorů se jedná o dva samostatné. Toto zařízení také měří naklonění a natočení mobilního telefonu ovšem oproti výchozímu

stavu. Gyroskop tedy měří otáčení, aniž by k tomu potřeboval zrychlení přístroje a gravitaci Země.

V praxi jde o malý čip, stejně jako akcelerometr, který měří úhlovou rychlost otáčení okolo dané osy. Nejhojněji jsou využívány v letecké dopravě, typický je umělý horizont pro orientaci pilotů. (Čížek, 2018)

5 Vývoj mobilní aplikace pro pohybové aktivity

Následující část práce se zabývá návrhem a postupem tvorby aplikace pro pohybové aktivity pro operační systém Android.

5.1 Základní informace

Jak jsem již na začátku zmínil, cílem této práce je vytvoření mobilní aplikace na Android, která bude využívána při pohybových aktivitách uživatele, speciálně jde o kliky, shyby, sed-lehy a dřepy. Následně aplikace uloží počet opakování a později je možné tyto data vizuálně porovnat s historickými daty uživatele.

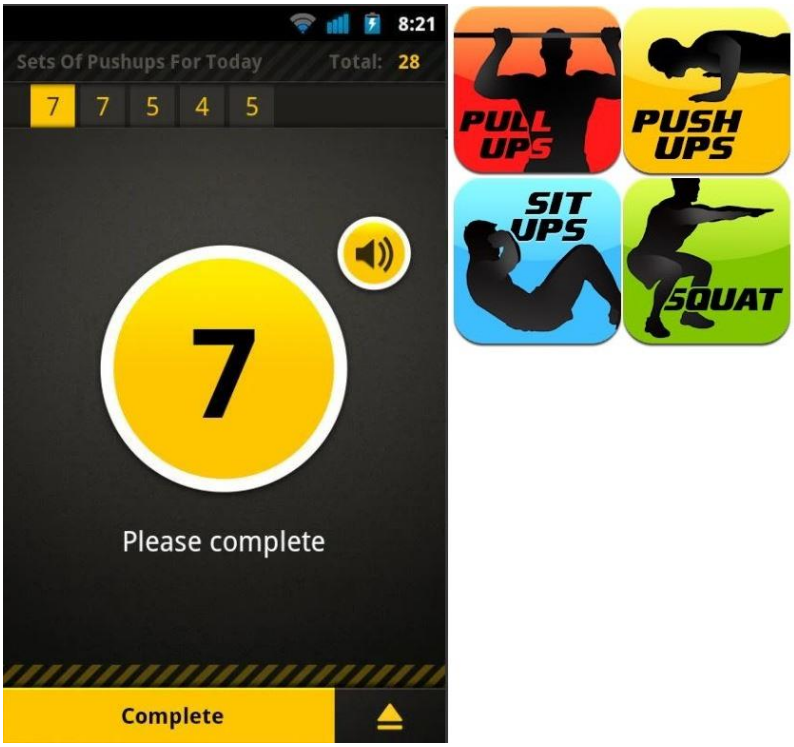
Aplikace je primárně určena pro mobilní zařízení operačního systému Android od verze 8.0, využívá tedy nejméně API 26. Díky takovému omezení má možnost nainstalovat si tuto aplikaci 60,8 % uživatelů operačního systému Android. Což vzhledem k trendu častého obměňování mobilních telefonů beru za dostatečnou hodnotu, která bude narůstat.

5.2 Související aplikace

Když jsem hledal funkčně podobnou aplikace v internetovém obchodě Google Play nenašel jsem takovou, která by měla přesně tyto funkce, které jsem od podobné aplikace čekal. Jsou zde aplikace na jednu specifickou činnost, které jsou možná lépe zpracované, ale nenabízejí tolik pohybů v jedné aplikaci.

Dle mého názoru jsou nejlépe zpracované aplikace od Simple Design Ltd., respektive NorthPark, které jsou navrženy zvlášť pro každou aktivitu, což je dle mého škoda. Na přiloženém obrázku 5 je možné vidět ikony jednotlivých aplikací a v levé části se nachází snímek obrazovky při cvičení. Funkčně jsou ale nejblíž aplikaci, která je vytvořena a popsána v této práci.

Obrázek 5 Aplikace od společnosti Simple Design Ltd.



Zdroj: Google Play

5.3 Architektura mobilní aplikace

Samotná architektura se nachází v příloze 1, kde jsou vyobrazeny použité třídy aplikace. Třída AppCompatActivity rozšiřuje veškeré třídy, které slouží pro zobrazování nějakého obsahu na obrazovku mobilního zařízení. Z důvodu přehlednosti celého diagramu tříd zde jsou naznačené vazby pouze v rámci rozšiřovaných tříd. Jak jsem zmínil, rozšiřuje pouze třídy, které mají zobrazovat samotný obsah aplikace, tudíž jde o třídy MainActivity, About_Activity a skupiny tříd Instructions, Record a Training. Jelikož hlavní stránka

aplikace, kterou představuje třída MainActivity, slouží jako rozcestník k ostatním obrazovkám, tak stojí mimo zbylou architekturu a má vazbu pouze na třídu Intent, která zajišťuje zobrazení ostatních obrazovek, tedy tříd. Použití této třídy je možné pozorovat u kapitol o přechodech na jiné obrazovky ve zdrojovém kódu 3. Třídy Instructions a About slouží pouze k zobrazení jednoduchého textu, proto nepoužívají jiné třídy. Naopak třídy Training musí uložit počet opakování dané pohybové aktivity. Pro ukládání používají třídu FitAppFileUtils, kde jsou příslušné metody. Všechny třídy Record nejprve použijí také třídu FitAppFileUtils, kde s použitím třídy MonthData_barchart načtou do listu všechny hodnoty v každý den určeného měsíce. Následně třídy Record využívají třídu FitAppGraphUtils pro vykreslení grafu. Třída FitAppGraphUtils používá list hodnot typu MonthData_barchart, odkud jsou použity metody pro zjištění dne a počtu opakování, následně použije třídu MyValueFormatter pro naformátovaný styl zobrazovaných hodnot, což jsou v tomto případě celá čísla.

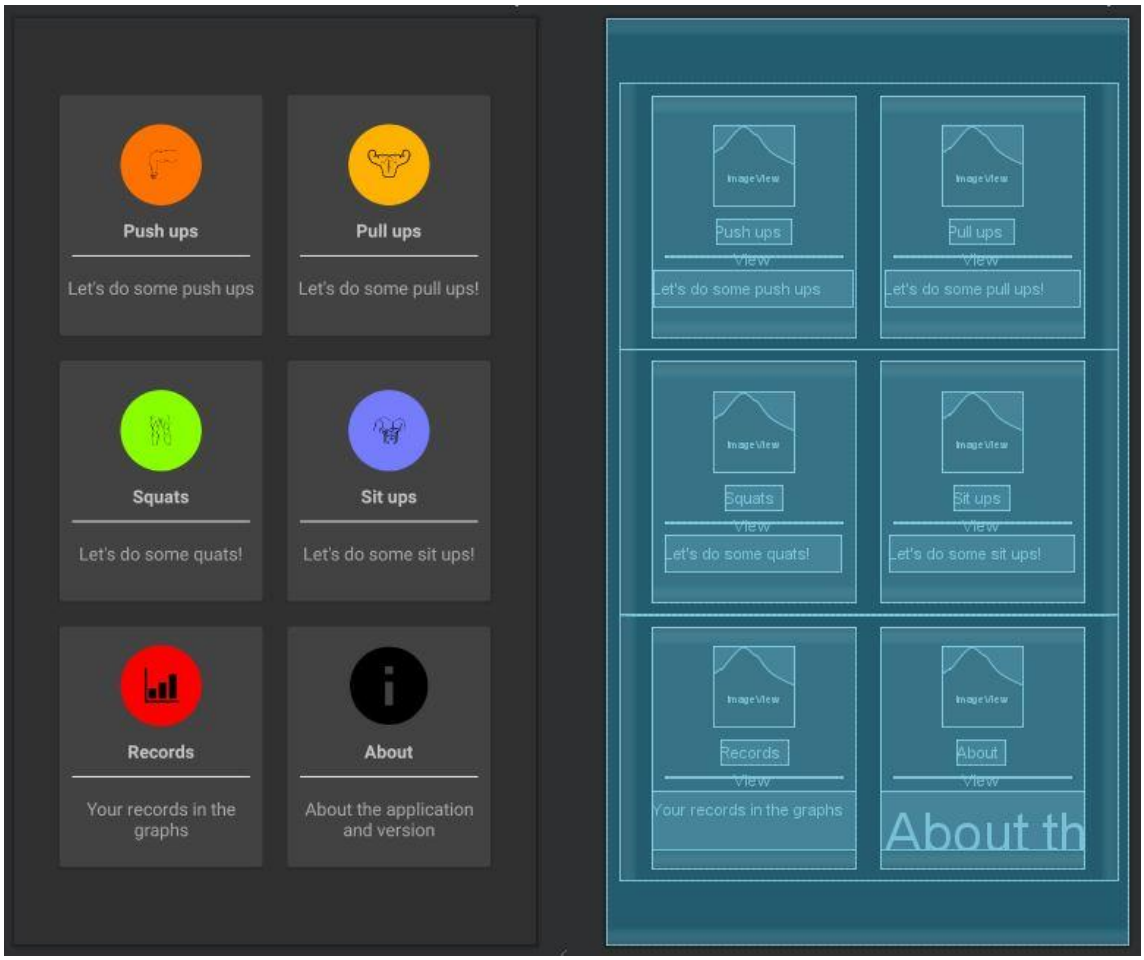
5.4 Uživatelské rozhraní

Pro uživatele jde o jednu z nejdůležitějších částí celé mobilní aplikace. V této části se uživatel pohybuje, proto je nutné dbát na snadný pohyb a orientaci v aplikaci. Neméně důležitou částí uživatelského rozhraní je pro potenciální uživatele i design, protože pokud se uživateli prostředí aplikace nelíbí, tak se ani nemusí dostat ke všem funkcím. V této aplikaci je několik různých sekcí, přičemž u každé pohybové aktivity jsou tyto sekce velice podobné, aby bylo docíleno uceleného vzhledu aplikace. V následujících kapitolách jsou jednotlivé sekce popsány i s obrázky návrhů.

5.4.1 Hlavní obrazovka

Do této sekce se uživatel dostane při každém spuštění mobilní aplikace. Jedná se o hlavní menu, a tedy i rozcestník, přes který se uživatel dostane do ostatních částí aplikace. Jednotlivé položky jsou umístěny do vlastních karet, takzvaných CardView, kde se nachází název pohybové aktivity a ikona odpovídající pohybové aktivitě v barevně odlišeném kruhu. Tyto panely jsou pro uživatele příznivější než obyčejný seznam se stejnou nabídkou. Na obrázku uživatelského rozhraní 1 je možné vidět rozvržení hlavní obrazovky aplikace v programu Android Studio, kde jsou jednotlivé panely hlavní dominantou.

Uživatelské rozhraní 1 Hlavní obrazovka



Zdroj: Autor

Po rozkliknutí panelu About se uživatel okamžitě přemístí na obrazovku s informacemi o aplikaci. Tento panel je od ostatních odlišný v tom, že uživateli nenabídne jinou možnost volby. Tato sekce má pro uživatele pouze informační funkci a objevují se zde pouze textová pole, takzvaná TextView.

Další jednodušší sekce je Records. V případě, že uživatel zvolí tento panel, aplikace mu zobrazí nabídku, ze které si uživatel může vybrat, aniž by se přesunul z hlavní obrazovky. Tento seznam nabídne zobrazení grafu zvolené pohybové aktivity.

Ostatní panely, i přes odlišnou pohybovou aktivitu, jsou velice podobné. Po zvolení dané aktivity se zobrazí, na pohled stejný, seznam. Nabídne možnost přesměrovat uživatele do různých částí dané pohybové aktivity.

Na dalším obrázku uživatelského rozhraní 2, lze nalézt zdrojový kód jednoho panelu v jazyce XML. Dvojice panelů, které jsou na obrazovce vedle sebe, jsou umístěny v horizontálním LinearLayout, tedy lineárním rozložení, ve kterém jsou jednotlivé panely

umístěny. Po pevném nastavení rozměru každého panelu obsahuje tento panel další LinearLayout, který je tentokrát vertikální, jelikož je tento panel orientován na výšku, viz horní obrázek. V každém panelu se nachází jako první ImageView, tedy prostor pro jednotlivé ikony spolu s barevným kruhovým pozadím. Další složku panelu tvoří samotný název pohybové aktivity, zvýrazněným písmem. Pod názvem lze nalézt část View, která tvoří oddělovací čáru od názvu a krátkého motivačního popisu. Poslední částí je další textové pole TextView s motivačním popisem aktivity.

Uživatelské rozhraní 2 Kód panelu kliků v jazyce XML

```
<!--Panel kliky-->
<android.support.v7.widget.CardView
    android:id="@+id/card_pushup"
    android:foreground="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:layout_width="160dp"
    android:layout_height="190dp"
    android:layout_margin="10dp"
    android:onClick="ShowPopup_PushUp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical">
        <ImageView
            android:layout_width="64dp"
            android:layout_height="64dp"
            android:src="@mipmap/iconpushup_js_foreground"
            android:background="@drawable/circlebackground_orange"
            android:padding="10dp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textStyle="bold"
            android:layout_marginTop="10dp"
            android:text="Push ups"/>
        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="@color/lightGray"
            android:layout_margin="10dp"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Let's do some push ups!"
            android:padding="5dp"
            android:textColor="@android:color/darker_gray"
            android:gravity="center"/>
    </LinearLayout>
</android.support.v7.widget.CardView>
```

Zdroj: Autor

5.4.2 Sekce About

Nachází se zde pouze několik textových polí, jak si lze na obrázku uživatelského rozhraní 3 všimnout, se základními informacemi o aplikaci. Pro běh a správné fungování aplikace je tato sekce bezvýznamná. Cílem této sekce byl design hlavního menu, aby vypadal celistvě. Navíc pro některé uživatele mohou být tyto informace zajímavé a přínosné. Tentokrát je stránka řešena o poznání snadněji. Obrazovku tvoří jedno LinearLayout, které je vertikální, tudíž jsou všechna textová pole pod sebou a zároveň se nepřekrývají. V každém TextView je nastavena maximální šíře obrazovky, ale výška se upravuje podle textu v daném poli. U každého pole je nastavena větší velikost písma a odsazení textu.

Uživatelské rozhraní 3 Kód sekce About v jazyce XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".About_Activity"
    android:orientation="vertical">

    <TextView
        android:id="@+id/aboutApp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/about_app"
        android:padding="10dp"
        android:textSize="20sp"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/version"
        android:text="@string/version"
        android:padding="10dp"
        android:textSize="20sp"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/author"
        android:text="@string/developer"
        android:padding="10dp"
        android:textSize="20sp"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/icons"
        android:text="@string/icons_jakub_schwarz"
        android:padding="10dp"
        android:textSize="20sp"/>
</LinearLayout>
```

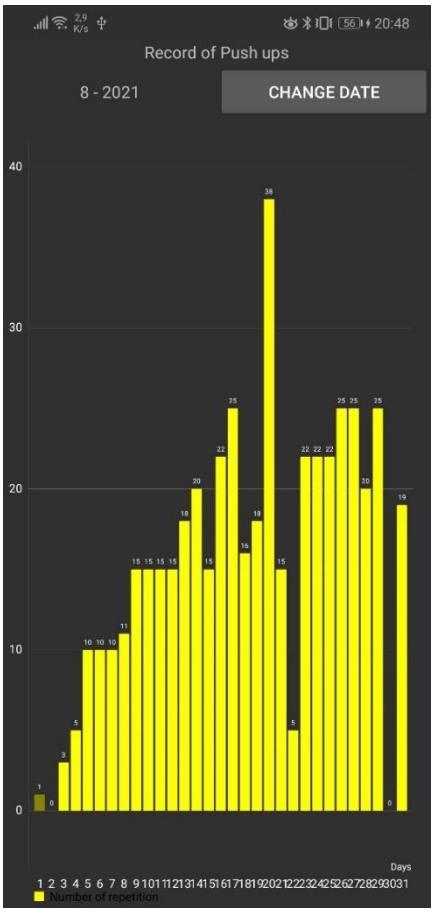
Zdroj: Autor

5.4.3 Sekce Records

Jak už bylo zmíněno výše, je možnost zvolit graf určité pohybové aktivity. Pro příklad jsem zvolil obrázek uživatelského rozhraní 4 s historickými daty kliků. Nahoře je vždy popisek, který uživatele ujistí, která data jsou zobrazována. Ihned pod tímto popiskem je možnost zvolit si jiný měsíc, ze kterého se mají hodnoty zobrazit. Zvolený datum se vypíše v levé polovině. Po každém otevření těchto statistik je uživateli zobrazován aktuální měsíc.

Pod možností vybrat si datum je hlavní část této sekce, tedy graf prostřednictvím GraphView. Na ose Y je možné vidět počet opakování daného cviku ve vybraném měsíci. Osa X zobrazuje dny zvoleného měsíce. Každý den má nad svým sloupečkem zobrazen počet opakování, čímž je zajištěna lepší přehlednost pro uživatele. Díky tomuto grafu si uživatel dovede porovnat své historické hodnoty.

Uživatelské rozhraní 4 Obrazovka Records



Zdroj: Autor

Na dalším obrázku, uživatelské rozhraní 5, je opět k vidění celý kód v jazyce XML k této části aplikace. Celá obrazovka je uspořádána pomocí vertikálního LinearLayout. Nahoře obrazovky je TextView, které obsahuje název stránky. Pod ním se nachází další LinearLayout, které nám dovolí umístit TextView s datem a Button pro změnu data vedle sebe. Aby tato pole zabírala každé polovinu šíře obrazovky bylo nutné nastavit layout_width na 0dp a zároveň layout_weight na .50, čímž je myšleno 50 % prostoru. Poslední tečkou je gravity="center", což zajistí vycentrování obou položek. Nejdůležitější část, tedy graf, je už zde definován jako BarChart, jelikož pro zobrazování počtu opakování bylo nejlepší zvolit graf sloupcový. Pro vykreslení grafu jsem použil knihovnu MPAndroidChart od autora Philipp Jahoda. Tato knihovna nabízí široké možnosti zobrazování a mnoho druhů grafů.

Uživatelské rozhraní 5 Kód aktivity Records v jazyce XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Record.Record_Pushup"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/record_of_push_ups"
        android:id="@+id/textview_nadpisPush"
        android:layout_gravity="center"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/horni_castPush">
        <TextView
            android:id="@+id/textView_datePush"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight=".50"
            android:gravity="center"/>
        <Button
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.50"
            android:id="@+id/btn_changeDatePush"
            android:text="@string/change_date"/>
    </LinearLayout>
    <!--graf-->
    <com.github.mikephil.charting.charts.BarChart
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/graphPushups"/>
</LinearLayout>
```

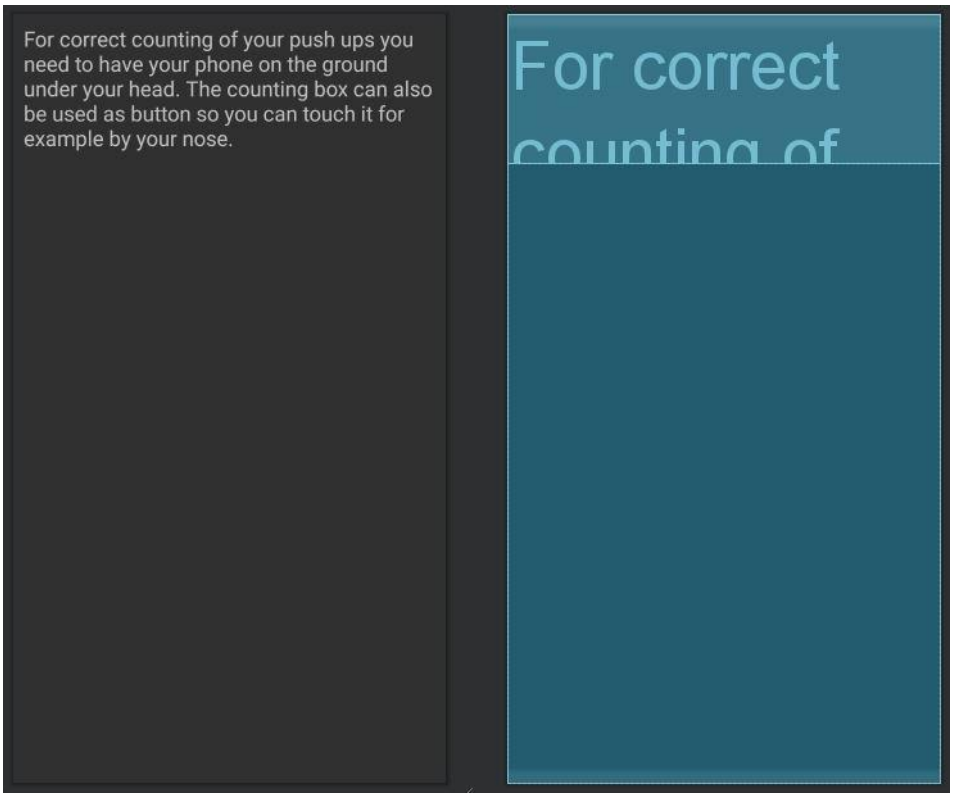
Zdroj: Autor

5.4.4 Sekce pohybových aktivit

Tyto sekce je potřeba popsat najednou. Nabídka po kliku na panel jednotlivých pohybových aktivit zobrazí tři možnosti: Training, Records, Instructions.

Možnost Instructions obsahuje pouze jediné textové pole, které uživateli říká, jak a kde by měl mít svůj mobilní telefon pro správné fungování aplikace, tedy správnému načítání počtu opakování vybrané pohybové aktivity. Na obrázku uživatelského rozhraní 6 je opět možné vidět návrh z Android Studia, kde je možné vidět popsané TextView. Aplikace byla vytvořena, aby počítání jednotlivých opakování fungovalo správně pouze ve vybrané sekci, tudíž nebude fungovat načítání kliků v sekci shybů.

Uživatelské rozhraní 6 Návrh obrazovky Instructions v aplikaci Android Studio



Zdroj: Autor

Po zvolení sekce Records aplikace přesměruje uživatele do výše popsané obrazovky se statistickými údaji dané pohybové aktivity.

Uživatelské rozhraní 7 Obrazovka Training

Zdroj: Autor

Nakonec je nejdůležitější část, tedy Training. Na této sekci závisí celá aplikace. Přesto, že každý cvik funguje na jiném principu, uživatelské rozhraní je na pohled totožné. Pro příklad jsem použil obrazovku pro načítání kliků, jak lze vidět na obrázku uživatelského rozhraní 7. V horní části je popisek, kterou aktivitu aplikace monitoruje a ihned pod ní se nachází velká plocha, která ukazuje počet opakování pohybové aktivity. Tento ukazatel zároveň funguje jako tlačítko, které je také schopné načítat opakování. Rozhodl jsem se pro tuto možnost z důvodu, kdyby některým uživatelům nebylo příjemné mít u sebe mobilní telefon při vykonávání této aktivity, ale chtěli si i přes tuto skutečnost zaznamenávat

své pokroky. Na spodní části obrazovky nalezneme tlačítko Save, tedy uložit zaznamenaný počet.

Na dalším obrázku uživatelského rozhraní 8 je zdrojový kód v XML jazyce této obrazovky. Opět je tvořen LinearLayout, ve kterém se nachází TextView s názvem pohybové aktivity. Pod tím je většinu zabírající tlačítko pro načítání opakování a zároveň ukazatel provedených opakování dané pohybové aktivity. Zde je opět použita hodnota layout_height odpovídající 0dp. Je to z důvodu vyplnění co největší plochy právě tímto ukazatelem, který uživatele zajímá nejvíce. I přes fakt, že layout_weight je nastaven na 1.0, tedy 100% plochou jsou vidět ostatní položky, protože ony mají jejich výšku nastavenou pouze na minimální požadovanou jejich obsahem, čímž je myšlena hodnota layout_height="wrap_content". Poslední položka je další Button, tentokrát slouží pro ukládání zaznamenaných opakování. V každé aktivitě odpovídá barevné pozadí tlačítka barevnému kolečku na hlavní obrazovce, které je možné vidět v uživatelském rozhraní 1.

Uživatelské rozhraní 8 Kód aktivity Training v jazyce XML

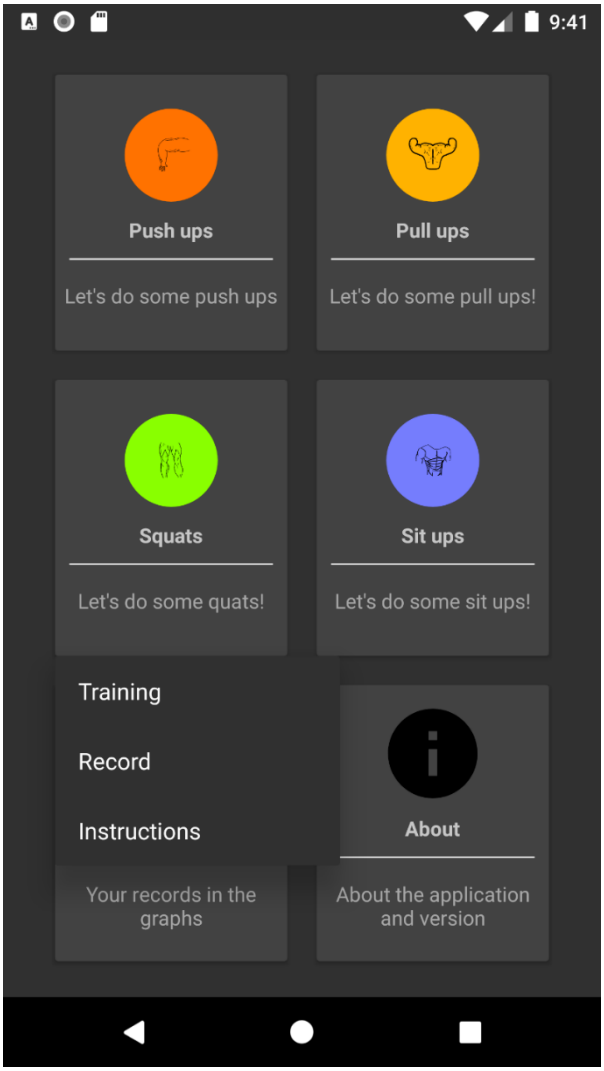
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Training.Training_Pushup"
    android:keepScreenOn="true"
    android:orientation="vertical">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/push_ups"
        android:id="@+id/pracTXT_pushup"
        android:layout_gravity="center"/>
    <Button
        android:id="@+id/pracBTN_pushup"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1.0"
        android:layout_marginTop="10dp"
        android:textSize="150sp"
        android:padding="5dp"/>
    <Button
        android:id="@+id/saveBTN_pushup"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:backgroundTint="@color/orange"
        android:text="@string/save"
        android:onClick="saveOnClickPush"
        android:textColor="@color/cardViewBackground" />
</LinearLayout>
```

Zdroj: Autor

5.4.5 Rozbalovací menu

Tato funkcionalita slouží pro přesnější a rychlejší přesměrování uživatele do požadované obrazovky aplikace. Na obrázku uživatelského rozhraní 9 lze vidět, že je vytvořena jednoduše jako seznam položek, které jsou nadefinované v kódu hlavní obrazovky. Nabídka vypadá stejně u všech pohybových aktivit a je pozměněna u panelu Records, aby byl uživatel přenesen do grafického zobrazení vybrané pohybové aktivity.

Uživatelské rozhraní 9 Hlavní obrazovka s rozbalovací nabídkou

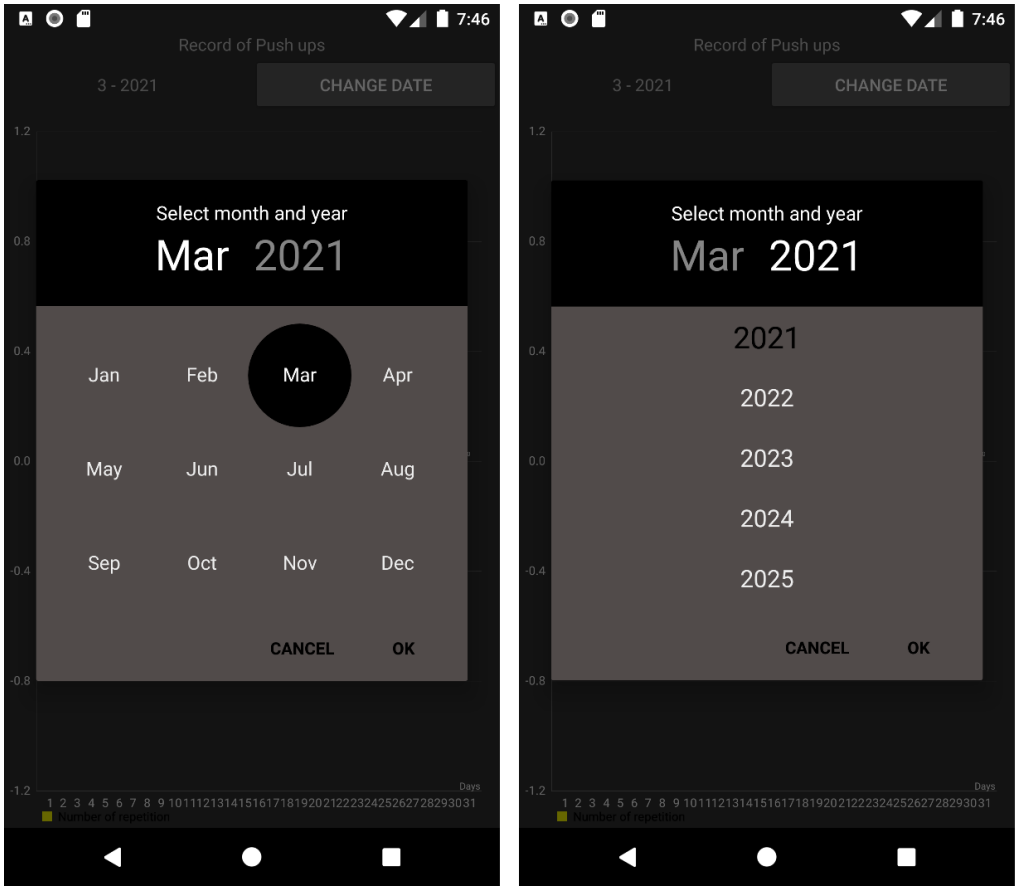


Zdroj: Autor

5.4.6 Změna data

Při potřebě uživatele změnit měsíc při zobrazování hodnot v grafu má uživatel možnost kliknout na tlačítko Change Date, které lze vidět na obrázku uživatelského rozhraní 4. Po kliknutí na zmíněné tlačítko se uživateli zobrazí vyskakovací okno pro změnu měsíce, což představuje levá polovina obrázku uživatelského rozhraní 10. Po zvolení požadovaného měsíce se okno posune na pravou polovinu uživatelského rozhraní 10, pro výběr roku. Pro tuto funkčnost jsem použil knihovnu MonthAndYearPicker od autora Kumara Prema, jelikož byla velice příkladně zpracována a zdokumentována.

Uživatelské rozhraní 10 Obrazovka pro výběr měsíce a roku



Zdroj: Autor

5.5 Analýza pohybových aktivit

Pro bezchybné načítání všech pohybových aktivit bylo nutné si zanalyzovat daný pohyb z pohledu mobilního zařízení. Pro tuto činnost jsem si vytvořil pomocnou aplikaci, která mi dovolila zaznamenávat hodnoty z různých senzorů a jejich os. V kapitolách níže popisuji, jak jsem tyto data vyhodnotil, a tedy jaké hodnoty aplikace hledá v datech ze senzorů. Pro všechny ukázky jsem zvolil spojnicový typ grafu, aby bylo možné sledovat průběh pohybu.

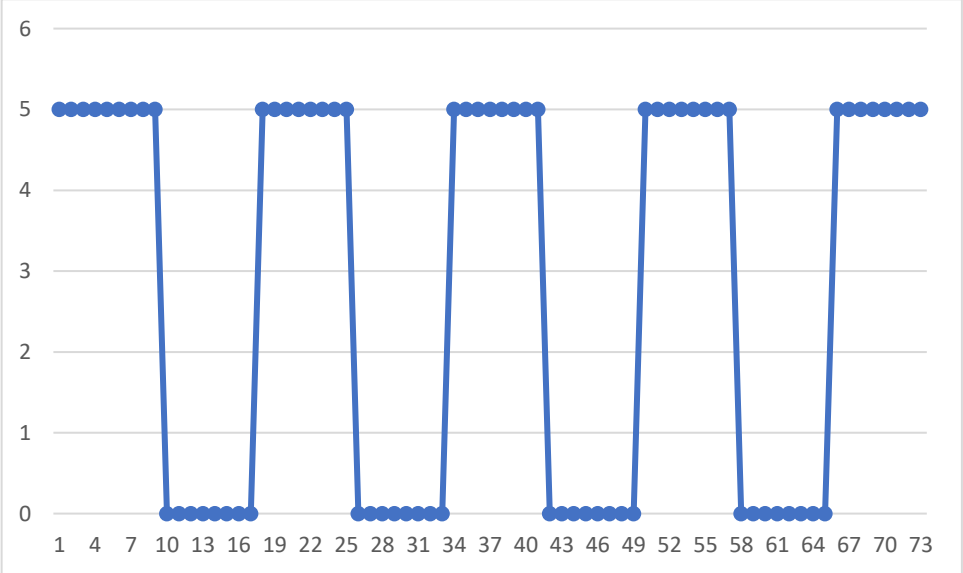
5.5.1 Kliky

Zmíněná pohybová aktivita je oproti ostatním velice odlišná. Z toho důvodu jsem se rozhodl pro použití senzoru přiblížení, tedy proximity senzoru. Při provádění tohoto cviku uživatel telefon nedrží v rukou, ani ho nemá na těle. Telefon leží pod uživatelem, ideálně pod prsy nebo hlavou.

Senzor přiblížení, ať už je jeho technologie jakákoliv, nabývá pouze dvou hodnot. První hodnota je jeho maximální rozsah, tedy neregistruje žádné přiblížení. Druhá je většinou nula, uživatel tedy přiblížil nějaký objekt k tomuto senzoru, respektive mobilnímu zařízení.

Z tohoto důvodu nebylo složité řešit nějaká složitá data, jelikož klik se skládá z pohybu v loktech nahoru a k zemi. Jelikož senzor funguje na relativně malou vzdálenost, takže se při pohybu nahoru uživatel dostatečně vzdálí od tohoto senzoru. Přesně tuto situaci lze vidět na grafu 1, který prozrazuje, že zařízení, na kterém jsem sbíral hodnoty senzoru, má maximální rozsah zmíněného senzoru 5 centimetrů. Cvik jako takový začíná v horní poloze, a proto senzor na začátku ukazoval maximální hodnotu. Následně jsem provedl pohyb dolů, což značí první linka dolů. Po dobu, kterou jsem strávil v dolní poloze cviku, senzor vykazuje nulovou hodnotu, když se mé tělo vzdálilo od senzoru, tedy zpět do horní polohy, začal senzor zaznamenávat opět hodnotu 5 centimetrů. Tímto lze vyvodit, že pro naměřené hodnoty na grafu jsem musel provést přesně 4 opakování.

Graf 1 Naměřené hodnoty ze senzoru přiblížení



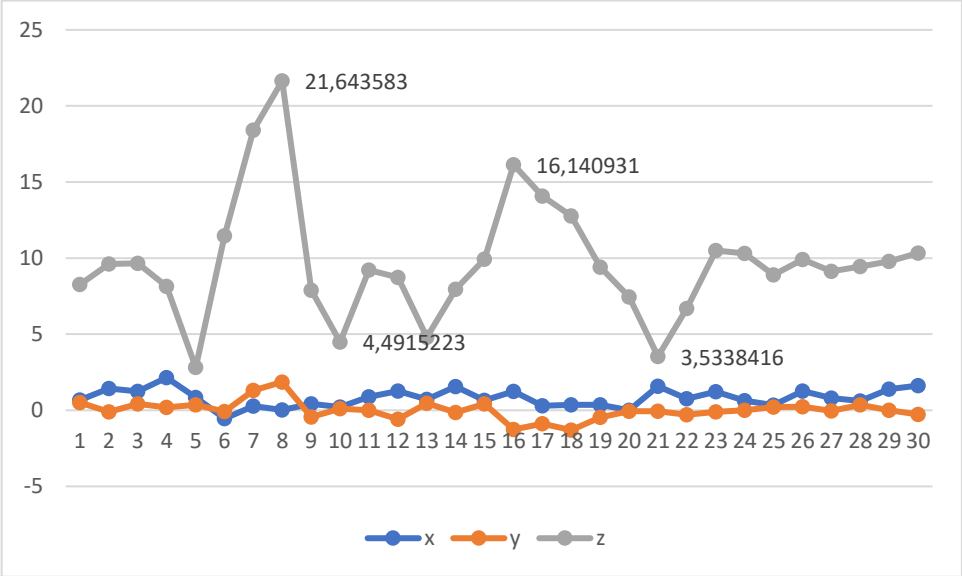
Zdroj: Autor

5.5.2 Dřepy

Pohybová aktivita, při která se uživatel pohybuje ve vzpřímeném postoji v kolenech nahoru a dolů. Vyplývá z toho, že vzniká zrychlení na jedné ose. Proto jsem si vytvořil aplikaci, která zaznamenávala data z akcelerometru. Bylo tedy nutné si ujasnit, jak telefon bude držen, aby měření bylo spolehlivé. Pro orientaci uživatele o počtu opakování jsem usoudil, že ideální bude držet telefon v předpažených pažích při orientaci na šířku.

Následně jsem si v této pozici zkusil udělat několik dřepů, abych měl představu, jaké hodnoty akcelerometr zaznamenává. Po několika pokusech a sesbíraných datech bylo zřejmé, které hodnoty budou podstatné pro správné fungování. Na grafu 2 lze jednoznačně najít pohyb na ose Z. Samozřejmě, že podle různých rychlostí vykonávaného pohybu se měnily i hodnoty, avšak nebylo to nijak značné.

Graf 2 Naměřené hodnoty akcelerometru při dřepech



Zdroj: Autor

Na grafu 2 lze pozorovat relativně stejné hodnoty na začátku, do bodu 4 na ose X, které se objevují i na konci, od bodu 23 a dále. Vypozoroval jsem, že pokud mobilní telefon leží na stole, osa Z vykazuje hodnoty oscilující okolo hodnoty 9,78. Tím jsem získal klidovou hodnotu, od které se další hodnoty pohybují. Samozřejmostí je, že akcelerometr měří jakékoliv zrychlení, takže se projeví i mírně se třesoucí ruce, případně jiné vlivy. Pro získání hodnot jsem provedl nejdříve jeden dynamický dřep, což je možné pozorovat mezi body 4 a 11. Následně jsem provedl opět jedno opakování dřepu, tentokrát méně dynamické. Tento pomalý dřep se odehrává mezi body 12 až 23. Učinil jsem tak, abych předešel chybám v načítání opakování, jelikož každý uživatel má trochu jiné pohyby.

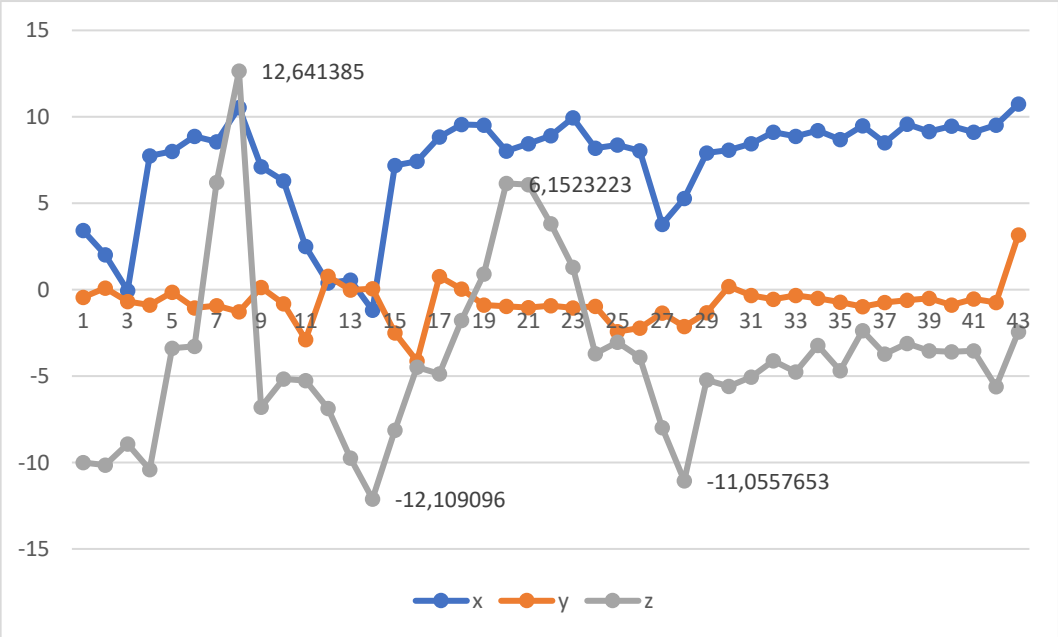
Hodnoty v bodě 4 až 7 označují dynamický pohyb těla do dolní polohy dřepu. Mezi body 8 až 11 se mé tělo dynamicky vrátilo do výchozí polohy pohybové aktivity. Body 9 až 11 znázorňují určitý zpětný pohyb při zastavení pohybu vzhůru, respektive může být způsobený pohybem paží. Nicméně body 11 a 12 ukazují prodlevu mezi prvním a dalším dřepem v horní poloze cviku. Druhý jsem provedl pomaleji, proto stihl akcelerometr nasbírat více hodnot. Pohyb do 16. bodu je pomalý pohyb dolů. Od bodu 17. do bodu 23 je pomalý pohyb nahoru. Tím jsem získal klíčové hodnoty na ose Z pro pohyb jednoho dřepu, které jsou 5 a 16.

5.5.3 Sed-lehy

Cvik, který začíná na zemi v poloze lehu s pokrčenými koleny a pohyb spočívá v pohybu horní poloviny těla ke zmíněným kolenům. Opět bylo nutné vyřešit pozici telefonu při cviku. Nejideálnější pozice mobilního telefonu je na prsou uživatele, jelikož bylo zapotřebí mapovat pohyb právě horní poloviny těla. Opět je to pozice, při které uživatel vidí na displej telefonu, což mu přináší přehled o počtu provedených opakování.

Zde jsem do mojí pomocné aplikace přidal gyroskop, který identifikuje pohyb telefonu, nejen zrychlení. Avšak po prvních několika opakováních pohybové aktivity jsem přišel na to, že není natolik užitečný, jak jsem myslel. To byl důvod zůstat u akcelerometru, který v klidové poloze displejem dolů na stole vykazuje hodnoty -9,8.

Graf 3 Naměřené hodnoty akcelerometru při sed-lezích



Zdroj: Autor

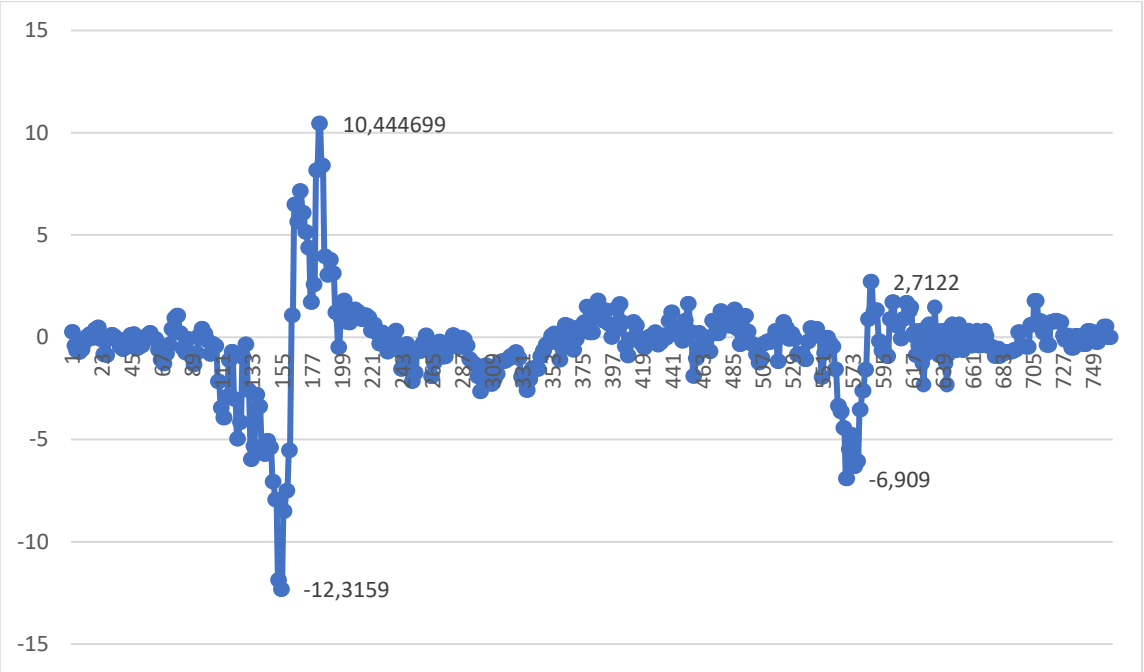
Opět jsem tedy získal hodnotu, okolo které se budou hodnoty pohybovat. Na grafu 3 je možné opět vidět všechny osy, aby bylo zřejmé, proč jsem zvolil hodnoty z osy Z jako podstatné. Jako u předešlé pohybové aktivity jsem provedl jedno dynamické opakování a druhé, které bylo děláno o něco pomaleji. Zde je možné nalézt prodlevu mezi cviky v bodech 16 a 17. V prvních bodech je vidět, že jsem začal cvik v poloze lehu s telefonem na prsou, proto se hodnoty pohybují blízko hodnotě -10. Horní poloha cviku se nachází v bodě 8, dynamický pohyb opět na zem je tedy možné vidět od tohoto bodu do bodu 16. V této klidové fázi na zemi senzor vypisuje hodnoty mezi -4 a -5, stejně jako je možné vidět ke konci, od bodu 29. Způsobeno to je natočením displeje telefonu na uživatelův obličej, tedy aby bylo možné kontrolovat načítání opakování. Pohyb pomalého sed-lehu se odehrává mezi body 17 až 29, kdy od 17. bodu až do bodu 20 šlo o pohyb nahoru ke kolenům. Od bodu 21 se tělo pohybovalo opět k zemi, a to až do 29. bodu. Takto jsem nalezl hodnoty pohybu, od kterých se bude odvíjet počítání sed-lehů. Aplikace tedy bude vyhledávat hodnoty na ose Z, které tentokrát jsou -11 pro dolní polohu a hodnotu 6 pro polohu sedu.

5.5.4 Shyby

Na začátek opět popíšu pohyb této pohybové aktivity. Cvik začíná v poloze visu a uživatel se přitahuje k hrazdě. Jde tedy o pohyb nahoru a zpět dolů. Poloha telefonu je tentokrát v kapse kalhot. Uživatel na něj bohužel nevidí, ale jelikož jsou ruce zaneprázdněny držením se hrazdy, tak nebylo možné jinam telefon umístit. Stejně dobře by však počítání fungovalo při zavěšení telefonu na krk například na klíčenku či jiné vhodné uchycení.

Vzhledem k pohybu padla volba opět na akcelerometr, bohužel díky započítání gravitace se hodnoty určovaly velice těžko, tudíž jsem byl donucen použít lineární akcelerometr, který ji nezapočítává a bere v úvahu pouze samotné zrychlení mobilního zařízení. Oproti akcelerometru ovšem započítá veškeré zrychlení, což vede k více hodnotám, které se pohybují blízko hodnoty 0. Z tohoto důvodu je v grafu 4 vidět pouze osa Y, kterou jsem vyhodnotil jako směrodatnou.

Graf 4 Naměřené hodnoty lineárního akcelerometru na osy Y při shybech



Zdroj: Autor

Jako u předešlých pohybových aktivit jsem udělal několik pokusů a výsledné hodnoty zpracoval do grafu 4, aby bylo možné vypracovat vzor pohybu, který byl centrem pozornosti. Při použití telefonu v kapse je možné vidět pohyb na ose y lineárního akcelerometru. Zde se bohužel nelze orientovat podle bodů na ose X, jelikož je hodnot zaznamenaných několikrát více. Nicméně první a dynamický shyb je velice dobře viditelný. První výrazný pohyb křivky až k hodnotě -12,3159 je pohyb těla nahoru, tedy směrem k hrazdě. Dynamický pohyb zpět do visu je možné pozorovat bezprostředně poté, vrcholí v hodnotě 10,4446. Následně jsem vydržel ve visu, ale přes veškerou snahu nehýbat se je vidět, že lineární akcelerometr je mimořádně důsledný v zaznamenávání sebemenších pohybů. Velice pomalý shyb je možné zpozorovat při druhém velkém výkyvu hodnot. Při pohybu nahoru jsem dosáhl hodnoty -6,909, následně při pohybu zpět do visu dokonce jen 2,7122. Z grafu nám tedy vyplývá, že aplikace hledá pohyb, kdy při stoupání nahoru bude tento pohyb nabývat hodnoty menších než -3. Při klesání do visu musí hodnota nabýt hodnoty větších než 2. Řekl bych, že tyto hodnoty jsou plně dostačující, jelikož zbylé hodnoty se pohybují v rozmezí maximálně 2.

Těmito úkony byl dokončen výzkum jednotlivých pohybových aktivit a mohl jsem je začít zapracovávat do kódu aplikace.

5.6 Rozbor zdrojového kódu

V této části uvedu určité části zdrojového kódu, které jsou důležité pro správné fungování aplikace. Začnu úplnými základy až po problematiku pohybových aktivit.

5.6.1 Inicializace uživatelského rozhraní

Názornou ukázku zprostředkuje třída Instructions_Pushup, která zajišťuje zobrazení obsahu při přechodu do instrukcí pro kliky a nachází se na obrázku zdrojového kódu 1. K vidění je metoda onCreate(), která se nachází v každé aktivitě, volá se při každém spuštění aktivity. Právě v této metodě dochází k inicializaci uživatelského rozhraní. Další důležitá metoda přiřazuje této třídě obsah podle nadefinované dispozice z XML souboru. Samotné jednotlivé proměnné se přiřadí pomocí findViewById() k vybranému komponentu z XML souboru. Díky této inicializaci všech komponent se vzájemně propojí a lze s nimi dále pracovat. Na tomto příkladu je vidět pouze jedna proměnná, se kterou však žádná akce neprobíhá, jelikož má za úkol pouze zobrazování textu, který je v tomto případě přiřazen v souboru XML.

Zdrojový kód 1 Ukázka inicializace proměnných ve třídě Instructions_Pushup

```
public class Instructions_Pushup extends AppCompatActivity {

    TextView textView;

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

        textView=findViewById(R.id.pushUp_instructions);
    }
}
```

Zdroj: Autor

5.6.2 Přímý přechod na jinou obrazovku

Nejjednodušší a nejdůležitější věc zároveň je přechod na jinou obrazovku, jelikož jednostránkové aplikace v podstatě neexistují, nebo se jedná o hodně jednoduché a jednoúčelové aplikace. Pro názorný příklad použiji obrázek uživatelského rozhraní 1 hlavní obrazovky, kde mám více druhů přechodů.

Teorie je snadná, uživatel klikne na nějaké tlačítko, které ho přenese na jinou obrazovku. Pro příklad poslouží přechod z hlavní obrazovky aplikace do informací o aplikaci. Část

třídy hlavní obrazovky MainActivity je vyobrazena na obrázku zdrojového kódu 2. Oproti obrázku zdrojového kódu 1 nahoře zde přibyla ještě řádka setRequestOrientation(), což je metoda která zajišťuje zobrazení, v tomto specifickém případě, pouze na výšku. Následně dochází k inicializaci všech panelů, které bylo možné vidět na obrázku uživatelského rozhraní 1 v kapitole Uživatelské rozhraní.

Zdrojový kód 2 Inicializace panelů a přiřazení funkce pro přechod na jinou obrazovku

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener, PopupMenu.OnMenuItemClickListener {
    CardView pushupCard,pullupCard,situpCard,squatCard, aboutCard,statisticsCard;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //na výšku
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);

        //inicializace CardViews
        pushupCard=findViewById(R.id.card_pushup);
        pullupCard=findViewById(R.id.card_pullup);
        situpCard=findViewById(R.id.card_situp);
        squatCard=findViewById(R.id.card_squat);
        aboutCard =findViewById(R.id.card_about);
        statisticsCard=findViewById(R.id.card_stats);
        //přesměrování přímo
        aboutCard.setOnClickListener(this);
    }
```

Zdroj: Autor

Po inicializaci všech panelů je v metodě onCreate() jen jediná řádka, kde je vidět práce s panelem About. Pomocí setOnClickListener() dojde k vygenerování metody onClick(), kterou lze nalézt ve zdrojovém kódu 3. Tato metoda nám dovolí zvolit akci, která nastane po kliku na výše inicializovaný panel. V tomto případě je jediný panel s přímou volbou jen About, tudíž je zde vidět, že uživatel bude přesměrován na obrazovku aktivity About_Actvity.

Zdrojový kód 3 Přímý přechod na jinou obrazovku aplikace

```
@Override
public void onClick(View v) {
    Intent intent=new Intent( packageContext: this, About_Activity.class);startActivity(intent);
}
```

Zdroj: Autor

5.6.3 Přechod na jinou obrazovku s možností volby

Opět se děj odehrává ve třídě MainActivity, platí tedy obrázek zdrojového kódu 2 výše, kde došlo k inicializaci panelů. Pro příklad použiji pouze pohybovou aktivitu kliků, tedy Push up. Níže, na obrázku zdrojového kódu 4, lze vidět celý proces pro panel kliků. Metoda ShowPopup_PushUp() není na první pohled propojena s inicializovanou proměnou panelu dané pohybové aktivity. Propojení s touto metodou se ovšem tentokrát

odehrává v souboru XML, toto spojení lze najít v uživatelském rozhraní 2, v poslední řádce části CardView, ve které se nachází spojení onClick, kde je uvedena přesně tato metoda. V první řádce zmíněné metody dochází k inicializaci vyskakovacího menu, další řádka má podobnou funkčnost jako poslední řádka v onCreate() na obrázku, tentokrát však pro zmíněné menu. Třetí řádka vyskakovací menu zobrazí a čerpá obsah z XML souboru, kde je nadefinované menu, které je možné vidět na obrázku uživatelského rozhraní 9. Poslední řádka zajišťuje pouze zobrazení.

Zdrojový kód 4 Přechod na jinou obrazovku pomocí vyskakovacího menu – zobrazení menu

```
public void ShowPopup_PushUp(View v){
    PopupMenu popupMenu=new PopupMenu( context: this,v);
    popupMenu.setOnMenuItemClickListener(this);
    popupMenu.inflate(R.menu.popup_menu_pushup);
    popupMenu.show();
}
```

Zdroj: Autor

Metoda onMenuItemClick() ve zdrojovém kódu 5 slouží k rozpoznání, kterou možnost si uživatel zvolil, zde uvedené id, která identifikují komponenty XML souboru, čerpají právě ze souboru, ve kterém je nadefinované samotné menu. Poté co tato metoda rozpozná, na kterou položku uživatel klikl, provede požadovanou akci.

Zdrojový kód 5 Přechod na jinou obrazovku pomocí vyskakovacího menu - definice funkce každé položky menu

```
@Override
public boolean onMenuItemClick(MenuItem item) {
    switch (item.getItemId()){
        case R.id.popup_instruct_push: openInstructionsPushup();return true;
        case R.id.popup_train_push: openTrainingPushup();return true;
        case R.id.popup_rec_push:openRecordPushup();return true;
```

Zdroj: Autor

V případě všech panelů a jejich menu se jedná o různé metody, které mají uživateli promítnout požadovanou obrazovku na displej jejich mobilního zařízení. V případě kliků jsou to tři metody na obrázku zdrojového kódu 6.

Zdrojový kód 6 Přechod na jinou obrazovku pomocí vyskakovacího menu - metody pro přechod na obrazovku

```
//kliky
private void openRecordPushup() {
    Intent intent=new Intent( packageContext: this, Record_Pushup.class);startActivity(intent);
}

private void openTrainingPushup() {
    Intent intent=new Intent( packageContext: this, Training_Pushup.class);startActivity(intent);
}

private void openInstructionsPushup() {
    Intent intent=new Intent( packageContext: this, Instructions_Pushup.class);startActivity(intent);
}
```

Zdroj: Autor

5.7 Pohybové aktivity

Nejdůležitější část aplikace tvoří kód pro samotné pohybové aktivity. Každá z nich má vlastní třídu Training, ve které se shoduje většina proměnných. Základní obrazovka pro trénink je vyobrazena na obrázku uživatelského rozhraní 7. Což znamená, že v každé třídě se shodují proměnné typu TextView a Button. Na obrázku zdrojového kódu 7 u kliků, lze vidět, že přibyla proměnná typu Integer, v tomto případě nazvaná pushupInt, která v sobě uchovává počet opakování po dobu, kdy je obrazovka pro trénink aktivní. Následně se zde nachází proměnné typu SensorManager, Sensor, boolean. Stejné typy proměnných se nachází ve všech třídách Training. SensorManager zprostředkovává přístup k senzorům obecně a proměnná typu Sensor definuje pomocí třídy SensorManager, který senzor je požadován. Na všech obrázcích (Zdrojový kód 8, Zdrojový kód 9, Zdrojový kód 10 a Zdrojový kód 11) lze vidět stejnou metodu countIn(), která zajišťuje načítání počtu opakování. Druhá řádka v této metodě ihned po načtení nastaví tuto hodnotu na obrazovku mobilního zařízení, což je možné vidět na obrázku uživatelského rozhraní 7. Další shodná metoda je onClick(), kterou využívám u všech pohybových aktivit, aby měl uživatel možnost si počet opakování naklikat i bez potřeby mít telefon u sebe při vykonávání aktivit. Tato metoda nás odkáže na metodu countIn() a slouží výhradně pro hlavní tlačítko. Následně je možné u každého cviku nalézt metodu pro ukládání dat, která je tlačítku Save přiřazena opět v souboru XML, jak je možné zpozorovat na obrázku uživatelského rozhraní 7. Podrobně je metoda popsána v kapitole Práce s daty, ale dojde zde k uložení naměřených hodnot do souboru a uživatele přenese opět na hlavní obrazovku. Metoda onSensorChanged() je automaticky vygenerována po inicializaci senzoru samotného, tudíž se nachází u všech tříd Training, ale jelikož je obsah rozdílný, tak se jí budu věnovat u aktivit samotných.

Zdrojový kód 7 Třída Training

```
public class Training_Pushup extends AppCompatActivity implements View.OnClickListener, SensorEventListener {
    TextView textView;
    Button btn, save;
    int pushupint;
    SensorManager sensorManage;
    Sensor proximitySensor;
    boolean locked = false;
    int max;

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

        textView = findViewById(R.id.pracTXT_pushup);
        btn = findViewById(R.id.pracBTN_pushup);
        save = findViewById(R.id.saveBTN_pushup);
        pushupint = 0;
        btn.setText(Integer.toString(pushupint));

        //přičítání opakování i na klik
        btn.setOnClickListener(this);

        sensorManage = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        proximitySensor = sensorManage.getDefaultSensor(Sensor.TYPE_PROXIMITY);

    }

    public void countIN() {
        pushupint++;
        btn.setText(Integer.toString(pushupint));
    }

    @Override
    public void onClick(View v) {
        countIN();
    }
```

Zdroj: Autor

5.7.1 Počítání kliků

Jelikož obecné části kódu jsem popsal výše, uvedu pouze, jak systém zjistí, že byl udělán jednotlivý klik. Ve zdrojovém kódu 8 je vyobrazena metoda onSensorChanged() ve třídě Training_Pushup a lze vidět, že první věc v této metodě je podmínka, která se ujišťuje, že data jsou ze senzoru přiblížení. Pokud je podmínka splněna, nastaví se zde maximální frekvence senzoru. Proměnná typu Integer nazvaná max je nastavena na 90 % maximálního rozsahu senzoru přiblížení, což je zde z důvodu, aby uživatel musel udělat poctivý klik, jelikož tyto senzory mají dosah až 10 centimetrů. Další důvod byl, že načítání nefungovalo správně, a proto bylo nutné přidat boolean locked a upravit tuto vzdálenost. Dále už čerpám z grafu 1, díky čemuž vím, že klik v dolní poloze nabývá

hodnoty 0. Hodnotu ze senzoru získám pomocí event.values[0], kdy event je jakákoliv změna senzoru a pole values uchovává hodnotu senzoru na nulté pozici. Pokud je tedy tato hodnota menší než 90 % rozsahu senzoru, tedy 0, a zároveň je boolean locked = false, proběhne přičtení opakování v metodě counInt(). Proměnná locked zajišťuje přičítání pouze jednoho opakování za dobu přiblížení u senzoru, jelikož jak bylo vidět v grafu 1, senzor po určité době, kterou jsem nastavil o několik řádků výše, přijímá nové, aktuální, hodnoty. Bez použití proměnné locked tedy docházelo k přičítání více opakování, pokud jsem v dolní poloze setrval déle. Z toho důvodu se ihned po přičtení jednoho opakování locked opět zneguje, aby se mohlo při oddálení těla od senzoru opět vrátit a aplikace mohla počítat další opakování, což značí druhá podmínka.

Zdrojový kód 8 Metoda onSensorChanged() ve třídě Training_Pushup

```
@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_PROXIMITY) {
        sensorManage.registerListener( listener: this, proximitySensor,  samplingPeriodUs: 2 * 1000 * 1000);
        max = (int) (proximitySensor.getMaximumRange() * 0.9);
        if (event.values[0] < max && !locked) {
            countIN();
            locked = !locked;
        }
        if (event.values[0] > max && locked) {
            locked = !locked;
        }
    }
}
```

Zdroj: Autor

5.7.2 Počítání dřepů

Jelikož je akcelerometr senzor, který je obsáhlejší, tak zde přibyla proměnná typu Integer min a proměnná typu float pro uchování hodnoty ze senzoru. Ve zdrojovém kódu 9 se objevuje metoda onSensorChanged(), ve které se ukládá hodnota event.values[2], kde druhá pozice v poli values odpovídá hodnotě osy Z. Následně se akce přesune do metody accelControl(), kterou jsem si vytvořil pouze pro podmínky načítání počtu provedených opakování. Zde jsou podmínky obdobné těm u kliků na obrázku zdrojového kódu 8 výše. Tentokrát hodnota max odpovídá 16 a v proměnné min je uložena hodnota 5, jak jsem vyhodnotil z grafu 2. Proměnná locked typu boolean funguje na stejném principu, tudíž se budu věnovat jen porovnávání hodnot v podmínkách. Pokud je hodnota akcelerometru na ose Z větší, než 16, dojde k přičtení opakování pomocí metody countIn(). Při hodnotě menší než 5 dojde opět k uvolnění.

Zdrojový kód 9 Metody ve třídě Training_Squat pro rozpoznání pohybu dřepu

```
@Override
public void onSensorChanged(SensorEvent event) {
    accelerationAZ = event.values[2];
    accelControl();
}
private void accelControl() {
    if (accelerationAZ>max && !locked){
            countIn();
            locked=!locked;
    }
    if (accelerationAZ<min && locked){
        locked = !locked;
    }
}
```

Zdroj: Autor

5.7.3 Počítání sed-lehů

Při počítá počtu opakování sed-lehů se neobjevuje nová proměnná, pouze jsem přejmenoval název pro hraniční hodnoty, podle polohy, kdy je tato hodnota naměřena. Jak je možné pozorovat v grafu 3, tak i zde na obrázku zdrojového kódu 10 v metodě onSensorChanged(), že ukládám hodnotu osy Z do proměnné typu float, se kterou opět pracuji v další metodě, po vzoru dřepů, a to v metodě accelerometerControl(). Jelikož zde pracuji se zápornými čísly, tak došlo k otočení znamének oproti podmínkách u dřepů. Zde hledám hodnotu menší než -11, kdy dojde k přičtení počtu opakování pohybové aktivity. Poté se počet přičte v metodě countIn(). Druhá podmínka, naměřená hodnota je větší než 6, opět odemkne pomyslný zámek v podobně proměnné locked.

Zdrojový kód 10 Metody ve třídě Training_Situp pro rozpoznání pohybu sed-lehu

```
@Override
public void onSensorChanged(SensorEvent event) {
    accelerationZ=event.values[2];
    accelerometerControl();
}

//tady budeme přičítat v horní poloze
private void accelerometerControl() {
    if (accelerationZ< onDown && !locked){
        countIn();
        locked=!locked;
    }
    if (accelerationZ> onTop &&locked){
        locked=!locked;
    }
}
```

Zdroj: Autor

5.7.4 Počítání shybů

Ve zdrojovém kódu 11 v metodě onSensorChanged() lze pozorovat, že tentokrát používám osu Y, jak jsem uvedl v grafu 4, a tedy se pozice v poli values změnila na pozici 1. U této pohybové aktivity je typ senzoru změněný na lineární akcelerometr, ale operace jsou totožné. Opět jsem změnil názvy proměnných hraničních hodnot, abych měl lepší přehled, kdy se tato hodnota vyskytuje v pohybu shybu. První podmínka říká, že naměřená hodnota je menší než -3, poté proběhne přičtení v metodě countIn(). Boolean locked funguje naprosto identicky a tvoří pojistku pro správnou funkčnost. Druhá podmínka, kdy je naměřená hodnota větší než 2, opět pojistku uvolní.

Zdrojový kód 11 Metody ve třídě Training_Pullup pro rozpoznání pohybu shybu

```
@Override
public void onSensorChanged(SensorEvent event) {
    accY=event.values[1];
    accelerometerControl();
}

//řešíme přičítání
private void accelerometerControl() {
    if (accY<linup&&!locked){
        countIn();
        locked=!locked;
    }
    if (accY>linDown){
        locked=!locked;
    }
}
```

Zdroj: Autor

5.8 Práce s daty

V aktuální fázi jsem získal všechna potřebná data a aplikace správně načítá počty opakování jednotlivých senzorů. Na řadu přichází zpracování dat. Pro ukládání zjištěného počtu opakování je nutné stisknout tlačítko Save, které je možné vidět na obrázku uživatelského rozhraní 7. Spojení tlačítka a metody je nastaveno v příslušném XML souboru. Na tomto obrázku zdrojového kódu 12 je možné vidět metodu ve třídě Training_Pushup, čemu odpovídá i samotná metoda saveOnClickPush(). Jak jsem psal výše, je nutné stisknout tlačítko pro uložení. Na první řádce lze vidět metodu savePushUpData() ve třídě FitAppFileUtils. Tato metoda má hlavní úkol přenést počet opakování do třídy FitAppFileUtils, kde je řešena veškerá práce s daty. Poté co tento proces proběhne je vyvolána metoda onBackPressed(), což jak název napovídá, zajistí návrat zpět, na hlavní obrazovku.

Zdrojový kód 12 Metoda pro ukládání počtu opakování ve třídě Training_Pushup

```
public void saveOnClickPush(View view) {
    FitAppFileUtils.savePushUpData(pushupint, view.getContext());
    onBackPressed();
}
```

Zdroj: Autor

5.8.1 Ukládání dat

Samotné ukládání je řešeno samostatným CSV souborem pro každou pohybovou aktivitu. Název tohoto souboru je uložen ve formátu String do samostatné proměnné, jak je uvedeno ve zdrojovém kódu 13 v prvních řádcích ve třídě FitAppFileUtils. Tyto proměnné jsou použity později i pro načítání dat ze souboru. Každá pohybová aktivita má svou metodu, která je pojmenována podle ní. To lze vidět na obrázku zdrojového kódu 13 hned pod jednotlivými proměnnými. Pro kliky slouží metoda savePushUpData(), která má v sobě pouze jednu řádku, která zajistí přesun jména souboru a počet opakování do metody saveDataToFile().

Zdrojový kód 13 Uložené názvy souborů a metoda pro ukládání dat kliků ve tříd FitAppFileUtils

```
public class FitAppFileUtils {

    private static final String pushUpFile = "pushups.csv";
    private static final String pullUpFile = "pullups.csv";
    private static final String sitUpFile = "situps.csv";
    private static final String squatFile = "squats.csv";

    public static void savePushUpData(Integer count, Context context) {
        saveDataToFile(pushUpFile, count, context);
    }
```

Zdroj: Autor

Tato metoda je uvedena ve zdrojovém kódu 14, ukládání do souboru vyžaduje podmínku try, kdyby se projevil jakýkoliv problém při ukládání. Jako první úkon je zjištění aktuálního data, v době ukládání do definovaného formátu. Následně do proměnné row typu String ukládám datum, který je oddělený od počtu opakování středníkem. Nakonec této proměnné přidám novou řádku. Dále je práce se samotným souborem, kde MODE_APPEND značí, že pokud je soubor vytvořený, tak se data budou zapisovat do stejného souboru. Na další řádce je metoda write(), která proměnnou row zapisuje do souboru. Po úspěšném uložení hodnot do souboru se uživateli zobrazí krátká zpráva o úspěšném uložení. V případě, že ukládání z nějakého důvodu neproběhne, tak se provede část catch, která uživateli vypíše, že ukládání selhalo.

Zdrojový kód 14 Obecná metoda pro ukládání dat ve třídě FitAppFileUtils

```
private static void saveDataToFile(String filename, Integer count, Context context) {
    try {
        String datum = new SimpleDateFormat( pattern: "dd-MM-yyyy", Locale.getDefault()).format(new Date());
        String row = datum + ";" + count + "\n";

        FileOutputStream fileOutputStream;
        fileOutputStream = context.openFileOutput(filename, MODE_APPEND);
        fileOutputStream.write(row.getBytes());
        Toast.makeText(context,  text: "Number of repetition was saved", Toast.LENGTH_LONG).show();
    } catch (IOException e) {
        Toast.makeText(context,  text: "Data save failed!", Toast.LENGTH_LONG).show();
        e.printStackTrace();
    }
}
```

Zdroj: Autor

5.8.2 Vykreslování uložených dat

Pokud má uživatel úspěšně uložené hodnoty a má potřebu si je zobrazit ve sloupcovém grafu, tak stačí přejít na hlavní stránce na panel Records a zvolit si v menu vybranou pohybovou aktivitu. Ihned po přechodu na obrazovku s historickými daty uživatele se tato data zobrazí v aktuálním měsíci. Na první obrázku zdrojového kódu 15, lze zpozorovat třídu Record_Pushup, do které se uživatel dostane, po výběru grafu s počty opakování kliků. Metoda setContentView() čerpá obsah ze souboru, který lze vidět na obrázku uživatelského rozhraní 5. Následně dojde k inicializace základních komponent. Tlačítko changeDate je popsané v kapitole Změna data, zde je vidět, že při kliknutí na něj se aplikace přesune do metody onClick(). Proměnná today typu Calendar získá hodnotu aktuální data.

Zdrojový kód 15 Metoda onCreate() ve třídě Record_Pushup

```
public class Record_Pushup extends AppCompatActivity implements View.OnClickListener {
    TextView nadpis, datum;
    Button changeDate;
    Calendar today;
    BarChart barChart;

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

        nadpis = findViewById(R.id.textview_nadpisPush);
        datum = findViewById(R.id.textView_datePush);
        changeDate = findViewById(R.id.btn_changeDatePush);
        barChart = findViewById(R.id.graphPushups);

        changeDate.setOnClickListener(this);
        today = Calendar.getInstance();

        Context context = getApplicationContext();
        //načtení dat ze souboru do listu
        List<MonthData_barchart> data = FitAppFileUtils.loadPushUpData( moth: today.get(Calendar.MONTH) + 1, today.get(Calendar.YEAR), context);
        //získání aktuálního data a jeho formátování do požadovaného tvaru
        String dateLabel = (today.get(Calendar.MONTH) + 1) + " - " + today.get(Calendar.YEAR);
        //nastavení aktuálního data do popisku
        datum.setText(dateLabel);
        //vykreslení grafu
        FitAppGraphUtils.createGraph(barChart, data);
    }
```

Zdroj: Autor

Ve zdrojovém kódu 15 lze vidět, že dojde k vytvoření listu typu MonthData_barchart, tato třída se nachází na obrázku zdrojového kódu 16. Slouží pouze k naformátování dat ve vytvořeném listu hodnot. Bere v úvahu zvolený, v tomto případě aktuální, měsíc, což je nadefinované v druhé polovině řádku. Třída MonthData_barchart uloží do každé jedné pozice den měsíce a k němu odpovídající počet opakování, který byl zaznamenán aplikací

Zdrojový kód 16 Třída MonthData_barchart

```
public class MonthData_barchart {
    int day;
    int count;

    public int getDay() { return day; }

    public void setDay(int day) { this.day = day; }

    public int getCount() { return count; }

    public void setCount(int count) { this.count = count; }

    public MonthData_barchart(int day, int count) {
        this.day = day;
        this.count = count;
    }
}
```

Zdroj: Autor

5.8.3 Získání dat ze souboru

Pro další postup se vrátím na obrázek zdrojového kódu 15. Po vytvoření listu je potřeba ho naplnit, což probíhá v pravé části řádku. Zde je možné vidět, že je volána metoda loadPushUpData(), která se opět nachází ve třídě FitAppFileUtils. To znamená, že vycházím z proměnných, ve kterých jsou uložené názvy souborů ve zdrojovém kódu 13. Ovšem tentokrát je na obrázku zdrojového kódu 17 metoda loadPushUpData(), uvnitř se nachází jediná řádka, která vede do obecné metody pro načítání dat ze souboru. Nicméně bylo potřeba přenést do ní název souboru, požadovaný měsíc a rok. Tím jsem přeskočil do druhé části zdrojového kódu 17, kde je obecná metoda loadDataFromFile(), která vrátí naplněný list typu MonthData_barchart. V první řádce je ukládána pozice souboru, dále je vytvořen list, který bude později naplněn. Pomocí cyklu for se vytvoří v listu 31 pozic, které budou připraveny přijímat hodnoty ze souboru. Tato hodnota tam není náhodně, jelikož uživatel nemusí cvičit s aplikací každý den, tudíž dochází k nahrání hodnoty opakování 0 do každého dne měsíce. Zbytek se stejně jako u ukládání odehrává pomocí try, pro případ, že by se přihodila nečekaná situace. Následně dochází pomocí cyklu while k procházení celého souboru po jednotlivých řádkách. Každou řádku rozdělím pomocí středníku do pole row. Následně si definuji formát data, ve kterém jsem si datum uložil do souboru. Měsíc a rok je v podmínce porovnáván s měsícem a rokem přeneseným ze třídy Records, může být aktuální, nebo zvolený. Pokud se data vzájemně shodují, dojde

k uložení počtu opakování do proměnné countFromFile z rozděleného pole row, které bylo použito i pro porovnání data ze souboru se zvoleným, ovšem počet opakování se nachází na 1. pozici. Následně je použita pomocná proměnná countFromData, která z listu data vyčte počet opakování v daný den. Poslední řádka části try přidává do listu na určeném indexu, kterým je den, součet hodnot z listu a souboru. Tím se uloží celkový počet opakování za každý den v měsíci. Když se projde celý soubor, metoda se ukončí a vrátí list s hodnotami ze souboru a akce se přesune opět do třídy Record na obrázek zdrojového kódu 15.

Nakonec se ještě v rámci načtení této obrazovky, která je k vidění na obrázku uživatelského rozhraní 4, vygeneruje aktuální měsíc, který se nastaví do TextView pomocí metody setText().

Zdrojový kód 17 Načítání dat ze souboru ve třídě FitAppFileUtils

```
public static List<MonthData_barchart> loadPushUpData(Integer moth, Integer year, Context context) {
    return loadDataFromFile(pushUpFile, moth, year, context);
}

private static List<MonthData_barchart> loadDataFromFile(String filename, Integer selectedMonth, Integer selectedYear, Context context) {
    File file = new File( pathname: context.getFilesDir() + "/" + filename);
    List<MonthData_barchart> data = new ArrayList();
    for (int i = 0; i < 31; i++) {
        data.add(i, new MonthData_barchart( day: i + 1,  count: 0));
    }

    try {
        BufferedReader br = new BufferedReader(new FileReader(file));
        String line;
        while ((line = br.readLine()) != null) {
            String[] row = line.split( regex: ";");
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
            LocalDate date = LocalDate.parse(row[0], formatter);
            if (date.getMonth().getValue() == selectedMonth && date.getYear() == selectedYear) {
                int countFromFile = Integer.parseInt(row[1]);
                int countFromData = data.get(date.getDayOfMonth() - 1).getCount();
                //setcount přičítá k hodnotě z listu hodnotu ze ssouboru
                //get na index v něm určený
                data.get(date.getDayOfMonth() - 1).setCount(countFromData + countFromFile);
            }
        }
    } catch (IOException e) {
        e.printStackTrace();
    }

    return data;
}
```

Zdroj: Autor

5.8.4 Vytvoření sloupcového grafu

Poslední viditelná řádka na obrázku zdrojového kódu 15 slouží k samotnému vykreslení získaných dat. Tentokrát se operace odehrávají ve třídě FitAppGraphUtils, která je vyobrazena na obrázku v příloze 2. Tvoří ji pouze jediná metoda createGraph() sloužící tvorbě grafu. Metoda přenesla data ze souboru, s jejichž pomocí vykreslí graf. V první řádce dojde k vytvoření listu typu BarEntry, který je uzpůsoben pro plnění grafu daty. Je

součástí knihovny MPAndroidChart, takže přesně odpovídá potřebám pro osy každého grafu. Pomocí cyklu for tedy přenesu hodnoty z listu data do listu barEntryArrayList. Na dalších řádcích probíhá pouze personalizace onoho grafu, včetně různých popisků. Ze všech těchto řádek vytknu pouze metodu setValueFormatter(), jelikož tato knihovna zobrazuje hodnoty jako číslo se dvěma desetinnými místy. Tato funkcionalita byla vysoce nevhodná, jelikož aplikace není uzpůsobena na počítání poloviny pohybových aktivit. Formátování tedy zmíním později. Předposlední řádka ukazuje proměnnou grafu volající metodu invalidate(). Tato metoda způsobí vykreslení celého grafu dle všech specifik. Ukončením této metody vznikne graf, který je zobrazen na obrázku uživatelského rozhraní 4.

5.8.5 Zobrazení celých čísel

Třída MyValueFormatter rozšiřuje třídu, která je součástí knihovny MPAndroidChart. Jak jsem poznamenal výše, bylo nutné uvádět hodnoty jako celá čísla. Toto je nastaveno pomocí vzoru uvedeného v konstruktoru třídy ve zdrojovém kódu 18.

Zdrojový kód 18 Třída MyValueFormatter pro formátování hodnot v grafu

```
public class MyValueFormatter extends com.github.mikephil.charting.formatter.ValueFormatter {
    private DecimalFormat mFormat;

    public MyValueFormatter() { mFormat = new DecimalFormat( pattern: "#"); }

    @Override
    public String getFormattedValue(float value) { return mFormat.format(value); }
}
```

Zdroj: Autor

5.8.6 Změna data

Na závěr se vrátím do třídy Record, jelikož je zde možnost změnit datum a tím i zobrazované hodnoty grafu. Po stisku tlačítka Change Date se aplikace přesune do metody onClick() v příslušné třídě. Na první řádce této metody, ve zdrojovém kódu 19, probíhá ověření, že bylo kliknuto na zmíněné tlačítko. Dojde k vytvoření instance kalendáře a následně kód čerpá z implementované knihovny MonthAndYearPicker. Uživateli se na obrazovce zobrazí možnost výběru měsíce a roku, přesně jako na obrázku uživatelského rozhraní 10.

Zdrojový kód 19 Metoda pro změnu data a předělání grafu

```
@Override
public void onClick(View v) {
    if (v.getId() == R.id.btn_changeDatePush) {
        final Calendar calendar = Calendar.getInstance();
        MonthPickerDialog.Builder builder = new MonthPickerDialog.Builder( context: Record_Pushup.this,
                (selectedMonth, selectedYear) -> {
                        String dateLabel = (selectedMonth + 1) + " - " + selectedYear;
                        datum.setText(dateLabel);
                        Context context = getApplicationContext();
                        List<MonthData_barchart> data = FitAppFileUtils.loadPushUpData( moth: selectedMonth + 1, selectedYear, context);

                        FitAppGraphUtils.createGraph(barChart, data);
                }, calendar.get(Calendar.YEAR), calendar.get(Calendar.MONTH));

        builder.setActivatedMonth(calendar.get(Calendar.MONTH))
                .setActivatedYear(calendar.get(Calendar.YEAR))
                .setMinYear(1989)
                .setMaxYear(2121)
                .setTitle("Select month and year")
                .build().show();
    }
}
```

Zdroj: Autor

Poté, co uživatel zvolí požadovaná data, uloží se do proměnných selectedMonth a selectedYear. Datum následně zformátuji do potřebného tvaru a pomocí metody setText() se zobrazí v levé polovině horní části obrazovky uživatelského rozhraní 4, aby si byl uživatel jistý, že zvolil správný datum. Vzápětí je opět vytvořen list typu MonthData_barchart, který pomocí metody loadPushUpData(), pro případ kliků, načte hodnoty ze souboru a dojde k vykreslení nového grafu. Pod celou touto částí je samotné nastavení okna pro výběr data. Nastavuji zde, že má uživatel na výběr pouze měsíc a rok, rozpětí let, které je možné si zvolit a samozřejmě informační popisek. Metoda show() slouží pro samotné zobrazení uživatelského rozhraní 10.

6 Uvedení na trh

Posledním cílem této bakalářské práce je uvedení vytvořené mobilní aplikace do obchodu Google Play. Jedná se o oficiální obchod, ze kterého lze instalovat aplikace do mobilních zařízení s operačním systémem Android.

6.1 Proces registrace

Pro možnost publikování vlastní aplikace do online obchodu je potřeba se registrovat jako vývojář do Google Play Console, která později nabízí přehledné prostředí pro spravování již vydané aplikace. Pro registraci je nutné dovršení věku 18 let. Následně je důležité zadat veřejné jméno vývojáře, kontaktní údaje a potvrdit smluvní podmínky. Pro potvrzení a dokončení registrace je potřeba uhradit jednorázový poplatek. Po obdržení platby přijde potvrzovací e-mail s informací o vytvoření účtu.

6.2 Povinnosti před nahráním aplikace

Prostředí Google Play Console nabízí jednoduché a přehledné prostředí. Možnost vytvořit aplikaci je barevně zvýrazněna oproti ostatním možnostem. První povinností je zadat název aplikace v rozsahu 0-50 znaků. Dále je nutné zvolit výchozí jazyk a definovat, zda se jedná o hru, či aplikaci, případně jestli bude zpoplatněna. Poté zbývá potvrdit vývojářské zásady. Všechny atributy lze později změnit, kromě zpoplatnění aplikace.

Následně je uživatel přesunut na další stránku, kde je o mnoho více možností. Před vydáním plnohodnotné verze lze aplikaci poskytnout pro testovací účely. Zde musí vývojář určit testery, kteří budou mít přístup k aplikaci. Při zvolení této možnosti ovšem není aplikace veřejně dostupná v obchodě Google Play.

Další panel upravuje nastavení aplikace, tedy jak bude aplikace na Google Play prezentována. Jsou zde dotazy, jestli je přístup k aplikaci omezen polohou, nebo přihlašovacími údaji uživatele. Pokud aplikace obsahuje reklamy, je nutné to zde zmínit. Následně je potřebné vyplnit dotazník pro hodnocení obsahu od oficiálních hodnotících institucí, kde prvním krokem je zařazení aplikace, dle jejího obsahu. Dotazník následně požaduje informace, zda se obsah aplikace týká násilí, sexuality, nevhodného vyjadřování, drog, případně propagace výrobků, či aktivity s věkovým omezením. Nacházejí se zde dotazy, jestli aplikace sdílí fyzickou polohu uživatele, nebo se jedná o aplikaci typu prohlížeče. Podle zadaných údajů dotazník vyhodnotí věková omezení pro různé země a vypíše, podle jaké hodnotící instituce byl tento status udělen, v Evropě se

jedná o PEGI[2]. Poté jste vyzváni k vybrání cílové skupiny, kdy pro uživatele mladší 13 let jste vyzváni k přidání zásad ochrany soukromí, proto jsem se rozhodl pro cílení na starší věkové skupiny. Přesto jsem musel potvrdit, že aplikace není neúmyslně přitažlivá pro děti, což samozřejmě ověřuje později sám Google. Poté zbývá potvrdit, že se nejedná o zpravodajskou aplikaci.

Od této chvíle je potřeba zařadit aplikaci v rámci obchodu Google Play. Určení kategorie aplikace probíhá již pomocí kategorií, které jsou uvedeny v online obchodu. V mém případě se jedná o aplikaci v kategorii zdraví a fitness, následně jsem měl možnost přidání štítků, které aplikaci začlení lépe do skupiny podobných aplikací, zde jsem zvolil zdraví a fitness, sport, bojová umění a hubnutí. Povinnou položku tvoří i kontaktní e-mailová adresa, která bude zobrazena u zveřejněné aplikace. Pro úplnost záznamu v Google Play je nutné zadat veřejný název aplikace, krátký a úplný popis. Ikona aplikace musí splňovat podmínku maximální velikosti až 1 MB a velikost 512 x 512 pixelů. Aby bylo možné aplikaci propagovat je zapotřebí nahrát grafiku, případně video větších rozměrů. Poslední povinnou částí je nahrání snímků obrazovek mobilního telefonu a tabletů s obrazovkami o velikosti 7 a 10 palců. Všechny tyto snímky musí splňovat poměr stran 16:9 nebo 9:16 a každá strana snímku se musí pohybovat mezi 320 a 3840 pixely s podmínkou velikosti do 8 MB na jeden snímek obrazovky. Tyto snímky obrazovky jsem pořídil pomocí aplikace Android Studio a vestavěného emulátoru, kde je možné si nadefinovat vlastní zařízení. Po nahrání všech snímků dojde k nahrání vlastní aplikace a čeká se na schválení.

Jakmile je aplikace úspěšně schválena, přijde oznamovací e-mail s odkazem do Google Play Console, kde je možné vygenerovat odkaz na veřejně publikovanou aplikaci a také statistiky o stahování, včetně recenzí.

6.3 Statistiky v Google Play

V období psaní této bakalářské práce si aplikaci nainstalovalo do svých zařízení celkem 26 uživatelů, z nichž napsalo recenzi 22. Tento počet není nijak velký z důvodu, že jsem pro uveřejnění v obchodě Google Play použil nově registrovaný účet, bez předchozích záznamů. Tudíž se aplikace ve vyhledaných výsledcích pohybovala na spodních pozicích. Většina uživatelů, kteří si aplikaci Home Workout – Counter nainstalovali, použila zařízení registrované v České republice. Druhou zemí v pořadí bylo Dánsko, ze kterého

[2] Pan-European Game Information

si aplikaci nainstalovali dva uživatelé a ze Spojených států amerických se objevil jeden uživatel.

Z hodnocení uživatelů jsem zjistil, že nejčastěji zastoupeným mobilním zařízením byl Redmi Note 8T, které použili 3 uživatelé. Zbytek mobilních zařízení byl zastoupen po jednotlivých kusech a různých výrobcích. Zajímavé bylo i složení verzí systému Android, kde jasnou převahu zaznamenal Android 10 (API 29) s celkovými 14 zařízeními a tedy 63,6% podílem. Jelikož bylo možné instalovat aplikaci pouze na zařízení s verzí systému Android 8 (API 26) a vyšších, tak není jasné, zda by se objevilo zastoupení i starších verzí. Druhá nejčastější verze systému byla použita na 4 zařízeních a jednalo se o Android 9 (API 28). Verze Android 11 (API 30) se vyskytla na 3 mobilních zařízeních a Android 8.1 (API 27) byl zastoupen pouze jedním zařízením.

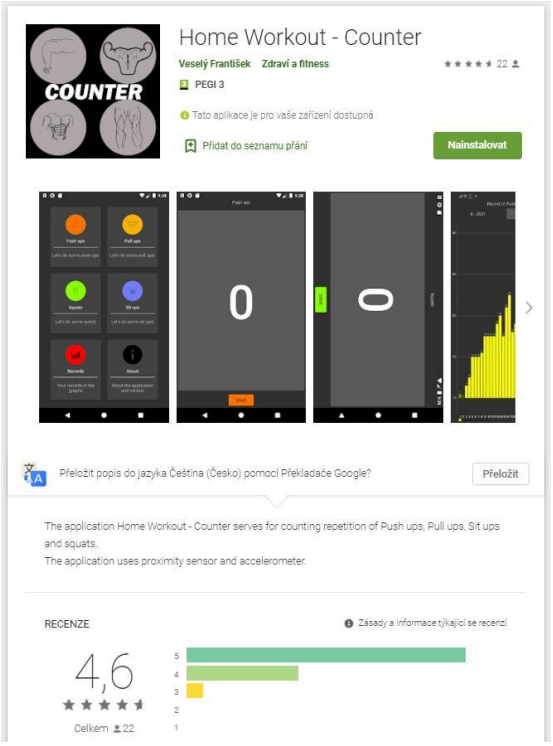
Všechny použité údaje byly přehledně zobrazeny v prostředí pro vývojáře v Google Play Console, kde lze zároveň nalézt recenze s přiřazením k použitému zařízení a verzí systému Android.

6.4 Hodnocení

Samotné recenze uživatelů mají pro mě velkou vypovídající hodnotu o provedené práci. Aplikace má podle mého úsudku vynikající hodnocení 4,636, které lze v zaokrouhlené podobě vidět na obrázku 6, kde je vyobrazena stránka vytvořené aplikace.

Recenze jsem shrnul a začnu nejdříve jejími částmi, kde bylo aplikaci něco vytýkáno. V jedné recenzi zaznělo, že by se uživateli líbilo, kdyby aplikace uměla počítat kalorie pohybů, což některé konkurenční aplikace nabízejí. Další výtka byla nemožnost zvolit si barevné téma aplikace, uživatel by preferoval světlý design. Se vzhledem

Obrázek 6 Aplikace v Google Play



Zdroj: Autor

souvisela i část další recenze, kde uživatel uvedl, že hlavní ikona aplikace, která se nachází v levém horním rohu obrázku 6, by mohla být výraznější, barevnější. Zároveň tento uživatel uvedl, že samotné ikony pohybových aktivit přímo v hlavní nabídce aplikace jsou málo výrazné. Ve dvou recenzích se objevila žádost o volbu jazyka aplikace. Prostředí aplikace je v anglickém jazyce, pro nalákání širšího spektra uživatelů, nicméně jde o další prostor k vylepšení aplikace. Celkem čtyři recenze uváděly, že se vyskytl občasný problém s počítáním opakování cviku. Bohužel u tří recenzí nebylo uvedeno, o jaký cvik šlo, pouze jedna uváděla, že se jednalo o dřepy. K přičítání opakování pohybových aktivit to byly jediné recenze, nicméně další dva uživatelé uvedli, že se jim zdáli nedostatečné instrukce a nevyhovovalo jim držení mobilního zařízení při cvičení. Přesto, že se jednalo pouze o dvě recenze, tak tento popis mohl způsobit nepřičítání některých opakování, tudíž by další verze této aplikace mohla obsahovat instruktážní video, nebo animaci. V dalších 2 recenzích byl tip na vylepšení, že by tato aplikace mohla obsahovat více cviků a funkcí. Poslední věc, kterou jsem zařadil, jako negativum bylo, že by jeden uživatel uvítal možnost zobrazení všech pohybových aktivit za jeden den v jednom grafu. Tato myšlenka pro mě byla nová, ale velice mě zaujala.

Celkem 10 uživatelů ve svých recenzích vysloveně uvedlo, že aplikace funguje dle jejich očekávání. Dále 6 uživatelů uvedlo, že se jedná o jednoduchou a přehlednou aplikaci. Samotné prostředí aplikace pochválili 3 uživatelé a jednoho uživatele ikony pobavily. Dva uživatelé uvedli, že oceňují možnost nemít při vykonávání cviků telefon u sebe a mají tedy možnost naklikat si počty provedených cviků později. Zároveň o provedených cvicích mají přehled i pomocí grafu. Jednomu uživateli se líbila možnost přiblížit si graf a pohybovat se v něm. Dva uživatelé uvedli, že aplikaci používají na denní záznam svých cviků a aplikace jim pomáhá udržet si formu v aktuální situaci. Zároveň jiní dva uživatelé napsali, že se jednalo o pochopitelnou a srozumitelnou aplikaci. Další dva uživatelé ocenili nápad.

7 Závěr

V teoretické části je popsán vývoj a návrh mobilní aplikace na operační systém Android. V této části jsou popsány vrstvy architektury OS Android a části samotné aplikace, jejichž znalost pomáhá při vývoji aplikace. Zmínil jsem základní informace o programovacím jazyku Java a vývojovém prostředí aplikace Android Studia. Na závěr teoretické části jsem uvedl základní pohybové senzory mobilního zařízení.

V praktické části bakalářské práce jsem popsal, jak jsem získal data o pohybových aktivitách. Následně jsem představil vzhled aplikace, včetně zdrojových souborů a jak aplikace funguje s ohledem na získané poznatky o pohybových aktivitách. Poslední částí byly praktické zkušenosti s publikací a zhodnocení zpětné vazby.

Práce na této aplikace mi zprostředkovala velké množství zkušeností v oboru, ve kterém jsem neměl žádnou zkušenost. Samotnému vývoji a testování velice pomáhalo zálohování verzí před úpravami, jelikož ne vždy se jednalo o úspěšné pokusy. V začátcích vývoje mi byl velice nápomocný emulátor, avšak v pozdějších fázích, kdy jsem testoval samotné pohybové aktivity, již nedostačoval. Nešel zde reálně nasimulovat pohyb telefonu při provádění cviků, tudíž jsem byl donucen použít fyzické zařízení, což naštěstí Android Studio podporuje.

Pro další rozšíření aplikace jsem získal tipy od reálných uživatelů aplikace, díky jejich recenzím. Zároveň je zde prostor pro zlepšení samotného počítání pohybových aktivit, jelikož i přes testování před samotným zveřejněním, není spolehlivě funkční pro všechny. Další rozšíření aplikace bych viděl v ukládání hodnot ze cvičení do databáze, pro rychlejší chod při větším počtu dat. Na aplikaci mám v plánu stále pracovat a vylepšit ji pomocí těchto tipů.

8 Summary and keywords

The objective of this bachelor thesis is to create a fitness application for Android devices, using the Android studio and publishing the application in the Google Play. The theoretical part of the bachelor thesis presents the development platform, history of Android and Java language.

The second part of this thesis is practical and describes the development of the mobile application from the creation of the user interface, the programming of the application logic to a connection with motion sensor and a proximity sensor of the devices.

The application can count number of repetition of squats, push ups, pull ups and sit ups by using motion sensors or touching the display. The application also can show a graph with number of repetitions.

Keywords: Android, application, fitness, development, java, squats, pull ups, push ups, sit ups, motion sensors, proximity sensor, accelerometer

9 Seznam použité literatury

Burton, M. (2015). *Android Application Development For Dummies.* Hoboken: John Wiley & Sons, Inc.

Cuello, J., & Vittone, J. (2013). *Designing Mobile Apps.*

Čížek, J. (2018). *Pojďme programovat elektroniku: Jak vlastně funguje akcelerometr a gyroskop nejen ve vašem telefonu*. Načteno z zive.cz: https://www.zive.cz/clanky/pojdme-programovat-elektroniku-jak-vlastne-funguje-akcelerometr-a-gyroskop-nejen-ve-vasem-telefonu/jak-funguje-gyroskop/sc-3-a-194858-ch-114926/default.aspx#articleStart

David Ortinau, o. (21. 8 2018). *Understanding Android API levels*. Načteno z Microsoft Docs: https://docs.microsoft.com/cs-cz/xamarin/android/app-fundamentals/android-api-levels?tabs=windows

developers, A. (2020). *Android Studio*. Načteno z Android developers: https://developer.android.com/studio/releases

developers, A. (2021). *Application Fundamentals*. Načteno z Android developers: https://developer.android.com/guide/components/fundamentals.html

Frank, J. (2015). *Lekce 8 - Android programování - Životní cyklus aktivity*. Načteno z IT netowk: https://www.itnetwork.cz/java/android/zaklady/tutorial-programovani-pro-android-v-jave-zivotni-cyklus-a-novy-projekt

-fs. (27. 2 2020). *Verze Androidu: Historie od 1.0 po 11*. Načteno z Computer world: https://computerworld.cz/internet-a-komunikace/verze-androidu-historie-od-1-0-po-11-55904

Chroust, M., & Kužel, F. (2015). *Smartphony mají 19 smyslů. Znáte je všechny?* Načteno z Mobilmania.cz: https://mobilmania.zive.cz/clanky/smartphony-maji-19-smyslu-znate-je-vsechny/sc-3-a-1329584/default.aspx

Kapoun, J. (18. 08 2010). *Historie Google*. Načteno z Business world: https://businessworld.cz/cio-bw-special/historie-google-6729

Mobilmania.cz. (2020). *Android*. Načteno z Mobil mania.cz: https://www.mobilmania.cz/android/sc-173/default.aspx

Mysliveček, D. (11. 5 2013). *Krátké ohlédnutí za historií Androidu*. Načteno z Svět Adnroida: https://www.svetandroida.cz/kratke-ohlednuti-za-historii-androidu/

Novotný, L. (2003). *Historie a vývoj jazyka Java*. Načteno z fi.muni.cz: https://www.fi.muni.cz/usr/jkucera/pv109/2003p/xnovotn8.htm

Statcounter. (2021). *Mobile Operating System Market Share Worldwide*. Načteno z Statscounter: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-200901-202102

Swift, O. (2015). *Android Programming Guide.* Scotts Valley: CreateSpace Independent Publishing Platform.

Škopek, P. (2013). *Techbox: váš telefon je prošpikovaný senzory*. Načteno z mobilnet.cz: https://mobilenet.cz/clanky/techbox-vas-telefon-je-prospikovany-senzory-12496

Škopek, P. (2013). *Techbox: váš telefon je prošpikovaný senzory*. Načteno z Mobilnet.cz: https://mobilenet.cz/clanky/techbox-vas-telefon-je-prospikovany-senzory-12496

TIOBE Index for February 2021. (2021). Načteno z TIOBE: https://www.tiobe.com/tiobe-index/

10 Seznam obrázků, tabulek, uživatelských rozhraní, grafů, zdrojových kódů a příloh

Seznam obrázků

Seznam tabulek

Seznam uživatelských rozhraní

Seznam grafů

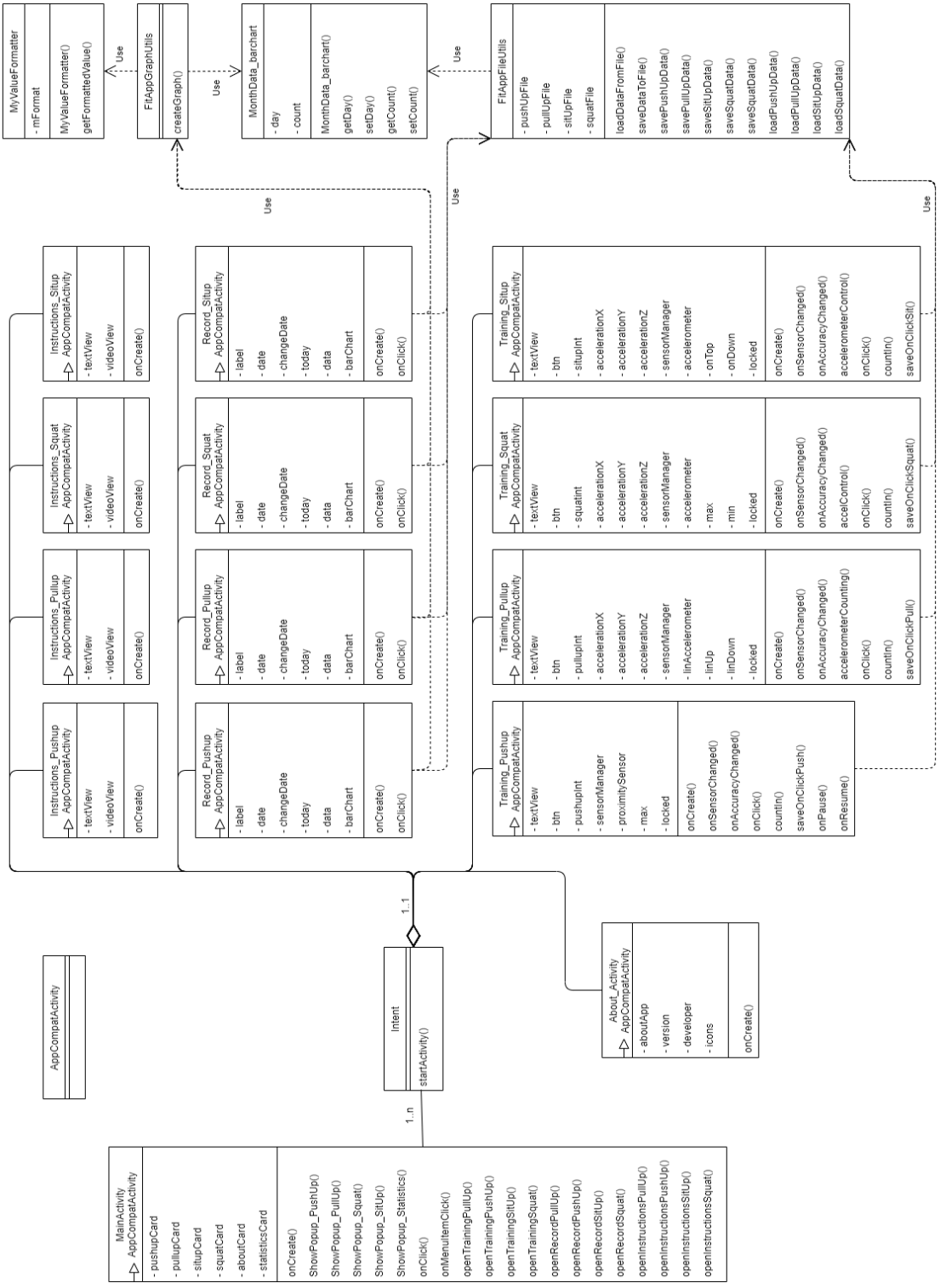
Seznam zdrojových kódů

Seznam příloh

Přílohy

Příloha 1

Diagram tříd



Zdroj: Autor

Příloha 2

Příloha 2 třída FitAppGraphUtils

```
public class FitAppGraphUtils {

    public static BarChart createGraph(BarChart barChart, List<MonthData_barchart> data) {
        //po zvolení data, potřebuji naplnit hodnoty ze souboru
        //vytvoření Array listu
        List<BarEntry> barEntryArrayList=new ArrayList<>();
        //přebírám data do array listu
        for (int i=0;i<data.size();i++){
            int day=data.get(i).getDay();
            int repet=data.get(i).getCount();
            barEntryArrayList.add(new BarEntry(day,repet));
        }
        BarDataSet barDataSet=new BarDataSet(barEntryArrayList, label: "Number of repetition");
        barDataSet.setValueTextColor(Color.WHITE);
        barDataSet.setAxisDependency(YAxis.AxisDependency.LEFT);
        barDataSet.setColor(Color.YELLOW);
        //celá čísla
        barDataSet.setValueFormatter(new MyValueFormatter());
        Description description=new Description();
        description.setText("Days");
        description.setTextColor(Color.WHITE);
        barChart.setDescription(description);
        BarData barData=new BarData(barDataSet);
        barChart.setData(barData);
        XAxis xAxis=barChart.getXAxis();
        xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
        xAxis.setDrawGridLines(false);
        xAxis.setDrawAxisLine(false);
        xAxis.setGranularity(1f);
        //popisky osy x - dny
        xAxis.setLabelCount(data.size());
        //vykreslení
        xAxis.setTextColor(Color.WHITE);
        YAxis yAxis=barChart.getAxisLeft();
        yAxis.setTextColor(Color.WHITE);
        //zmizí pravá Y osa
        barChart.getAxisRight().setEnabled(false);
        //každý sloupec má stejný prostor
        barChart.setFitBars(true);
        //vykreslení
        barChart.invalidate();
        return barChart;
    }
}
```

Zdroj: Autor

Příloha 3

CD se zdrojovým kódem aplikace, instalačním souborem FitApp.apk a plným textem bakalářské práce.