VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

BUBENICKÝ METRONOM PRO SMARTPHONE A SMARTWATCH

DRUMMER'S METRONOME FOR SMARTPHONE AND SMARTWATCH

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ALEŠ ONDRÁČEK**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, PhD.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Ondráček Aleš**
Program: Informační technologie
Název: **Bubenický metronom pro Smartphone a Smartwatch**
Drummer's Metronome for Smartphone and Smartwatch
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s problematikou vývoje aplikací pro smartphone a smartwatch, zaměřte se na jednu zvolenou platformu.
2. Popište, jak bubeník na koncertě využívá metronomu a jaké existují dostupné pomůcky; analyzujte současný stav a identifikujte možnosti použití chytrých hodinek pro tento účel.
3. Prototypujte způsoby interakce bubeníka s metronomem využívajícím chytrých hodinek. Testujte své prototypy na uživatelích a iterativně je vylepšujte.
4. Navrhněte bubenický metronom pro smartphone a smartwatch.
5. Implementujte navrženou aplikaci, testuje ji na uživatelích a iterativně ji vylepšujte.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN-13: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN-13: 978-0321657299
- Android Developers: https://developer.android.com/index.html

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, značné rozpracování bodů 3 až 5.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Cílem této práce je vytvořit bubenický metronom pro smartwatch a smartphone, který bude umožňovat ukládání rytmických vzorů do playlistů. Výsledný produkt sestává ze dvou částí: z aplikace pro smartwatch a aplikace pro smartphone. Tyto dvě aplikace spolu pak komunikují prostřednictvím bluetooth. Metronom se tedy dá ovládat vzdáleně prostřednictvím hodinek, což je požadovaná cílová funkcionalita. Výsledkem je metronom implementovaný na míru bubeníkovi, hrajícímu s kapelou jak na zkouškách, tak i na menších či středně velkých koncertech.

Abstract

The purpose of this thesis is to create a drum metronome smartphone application, which would allow rhythm patterns to be stored in playlists. These would then be used to assist the drummer while playing at rehearsals and at smaller or medium-sized concerts Key properties of the smartphone application design include the ability to be controlled wirelessly via bluetooth, by a smartwatch application. The final product consists of two interconnected parts: a smartphone metronome application, and a smartwatch control application.

Klíčová slova

bubenický metronom, GUI, smartwatch, smartphone, komunikace prostřednictvím bluetooth, Java, Android Studio, Material Designe, SQLite

Keywords

drum metronome, GUI, smartwatch, smartphone, communication via bluetooth, Java, Android Studio, Material Designe, SQLite

Citace

ONDRÁČEK, Aleš. *Bubenický metronom pro smartphone a smartwatch.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, PhD.

Bubenický metronom pro smartphone a smartwatch

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana profesora Adama Herouta. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Aleš Ondráček
15. května 2019

Poděkování

V první řadě bych chtěl poděkovat panu profesorovi Adamu Heroutovi za úžasný přístup při vedení práce a za všechny jeho rady, které mi byly k užitku. Dále pak všem, kteří mi pomohli s testováním aplikace, především pak Janu Poláškovi, bubeníkovi kapely *Of Lost Things*.

Obsah

Kapitola 1

Úvod

Tato práce popisuje implementaci a testování bubenického metronomu. Tento metronom je implementován pro smartwatch a smartphone. Obě aplikace se dají využít samostatně, ale největší silou výsledného produktu je možnost ukládání rytmických vzorů do playlistů v rámci mobilní části aplikace. Tyto playlisty pak mohou být pomocí hodinek vzdáleně ovládány prostřednictvím technologie bluetooth.

Právě vzdálené ovládání aktuálně nabízené metronomy na *Google Play* postrádají. Tato práce tedy řeší tuto absenci. Hlavní výhodou je jednoduché a rychlé zavedení požadovaného rytmického vzoru prostřednictvím hodinek a možnost mít mobilní telefon v blízkosti mixážního pultu, což řeší mnohdy problematické vyvedení zvuku metronomu do odposlechu bubeníka. V kapitole 2.4 se zaměřím na porovnání s ostatními metronomy na trhu (jak softwarovými tak i hardwarovými).

Hlavní částí práce je pak popis základních principů při postupu vývoje aplikace pro Android za využití programovacího jazyka Java a vývojového prostředí *Android studio*. Další významnou částí práce je popis implementačních detailů, jako například práce s ukládanými daty za pomoci *SQLite*, a nebo dle mého názoru nejpodstatnější část, která popisuje komunikaci mezi hodinkami a mobilním telefonem prostřednictvím bluetooth. Dále se čtenář dočte něco málo o testování v provozu v průběhu vývoje aplikace.

Kapitola 2

Průzkum kontextu použití

V několika následujících podkapitolách je řečeno něco málo o metronomu samotném. Dále je popsáno, co vše by aplikace měla umět a pro koho je určena. V poslední podkapitole si pak představíme několik aktuálních produktů, které můžeme najít mezi nejpopulárnějšími ve službě *Google Play*.

2.1 Metronom a jeho stručná historie

V první řadě by bylo dobré říct si něco málo o metronomu (taktometru) [17] a jeho historii. Jak už z názvu vyplývá, jedná se o nástroj ke stanovení tempa interpretovaného hudebního díla. Nástroj byl původně mechanický. Dnes se setkáme častěji s digitálními verzemi. Podstatou tohoto zařízení je reprodukce zvuku o krátké době trvání, který reprezentuje jednu dobu v taktu. Tento zvuk dále nazývejme úder. Nejčastěji se setkáme s variantou metronomu, kde jeden úder představuje počátek čtvrťové noty.

Tempo úderů metronomu se nastavuje v jednotce M.M., tedy „Mälzelův metronom“. Tato jednotka je odvozena podle Jana Nepomuka Mälzela, který zhotovil první mechanický metronom. Dnes se však častěji setkáme s pojmem počet rázů či úderů za minutu; anglicky tedy beats per minute. Z toho je odvozena zkratka BPM, která je používána ve zbytku práce. [25]

Jak už jsem výše zmínil, autorem původního metronomu je J. N. Mälzel, který tento mechanismus vytvořil na zakázku pro L. van Beethovena. Původní princip metronomu byl založen na hodinovém stroji s pružinou a obzvlášť charakteristické je reverzní kyvadlo. Tato forma kyvadla se u klasických mechanických metronomů zachovala dodnes, avšak pružina jako pohon se užívá již zřídka. [17]

Dnes mnohem běžnější formou metronomu jsou metronomy elektronické. Některé z nich budou představeny v podkapitole 2.4.

2.2 Cílová skupina a cílová persóna výsledného produktu

Cílová skupina pro výslednou aplikaci zahrnuje veškeré hudebníky, kteří hrají podle metronomu. Obě dvě části aplikace (část pro smartwatch i smartphone) obsahují základní funkcionalitu metronomu, tzn. jsou vhodné pro cvičení.

Část pro smartwatch nevyužívá klasického konceptu metronomu a pro udávání tempa využívá vibrací. Část pro smartphone se drží klasického pojetí metronomu a tempo udává prostřednictvím zvuku.

Cílovou skupinou je tedy skupina muzikantů, kteří využívají metronom. Produkt byl ovšem vyvíjen cíleně pro bubeníky. Jakožto aktivní muzikant jsem na základě pozorování zjistil, že mnohé dosavadní řešení neobsahují možnost ukládání rytmických vzorů, což bubeníkům, kteří se řídí podle metronomu způsobuje značný problém. Bubnování je fyzicky náročné a během hraní setu mezi jednotlivými skladbami není vždy snadné zadat požadované tempo do metronomu (důvody jsou například zpocené ruce, nebo časová tíseň při živých vystoupeních mezi písněmi). Tento problém má řešit podpora ukládání rytmických vzorů do playlistů, ve kterých se dá snadno mezi tempy listovat.

Dalším problém vystává pokud je bubeník zvyklý hrát během koncertů s metronomem v odposlechu. Často není technicky možné tento požadavek na akcích realizovat. Více o tomto problému v podkapitole 2.3.3. Ná základě výše uvedených faktů byla sestavena tato cílová persóna:

- muž, 25 let
- zvyklý zacházet se smart technologiemi jako je smartwatch či smartphone
- pokročilý až poloprofesionální bubeník
- pravidelně vystupuje na malých a středněvelkých živých vystoupeních se svojí kapelou
- při tréninku i během živých vystoupení používá metronom
- hledá jednoduché řešení zapojení metronomu do odposlechu během vystoupení

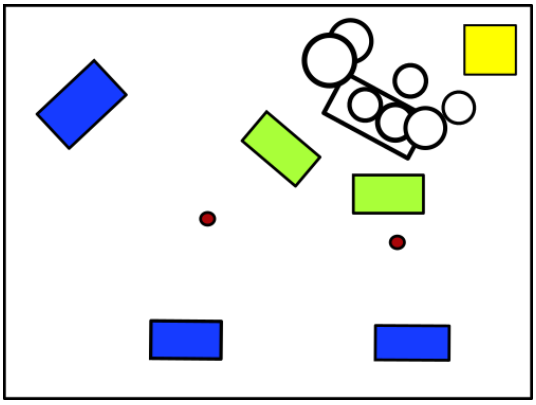
2.3 Typické příklady užití

Jak už bylo několikrát zmíněno, aplikace se skládá ze dvou částí, které jsou samostatně funkční. Dá se tedy využít samostatně část pro smartwatch, stejně jako pro smartphone. Přidanou hodnotu má však tento metronom pouze v momentě, kdy bubeník bude využívat obě aplikace zároveň spárované pomocí bluetooth. V této kapitole bych chtěl objasnit všechny tři příklady užití.

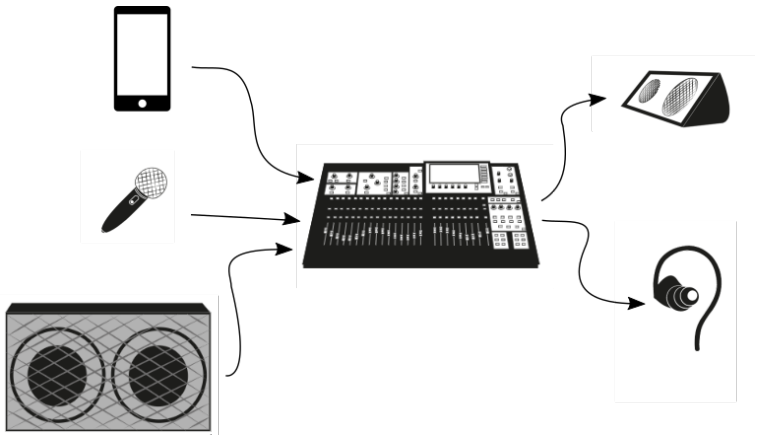
2.3.1 Typický příklad užití části aplikace pro smartphone

Tato část je vhodná pokud uživatel preferuje klasické udávání tempa pomocí zvukových signálů. Uživatel může využít playlistů a uložit si hraný set skladeb. V módu pro přehrávání playlistu je pak poměrně snadné přepnout rytmický vzor i v časové tísni. Toho se dá využít na zkouškách kapely, kde je mixážní pult blízko bubeníka a nebo je zázemí na toto zapojení zařízené (je vyvedená kabeláž k bubeníkovi, kde má menší dvojvstupý mixážní pult).

Na obrázku 2.1 je znázorněno rozložení zkušebny. Jako předlohu jsem využil naši zkušebnu o rozměrech 6×6 m^2. Z obrázku je zřejmé, že mixážní pult je přímo u bicích. Není tedy problém mobilní telefon s metronomem připojit rovnou do mixážního pultu, z kterého jsou zvuky všech nástrojů, zpěvu i samotného metronomu vedeny rovnou do monitoringových sluchátek bubeníka. Kytarové aparáty, mikrofon a metronom jsou na vstupu do mixážního pultu. Na výstupu je zpěvový monitoring, do kterého je vedena pouze zpěvová linka. Schéma zapojení je zobrazeno na obrázku 2.2.



Obrázek 2.1: Schéma rozložení zkušebny. Modrá – kytarové aparáty, zelená – zpěvové odposlechy, červená – mikrofony, žlutá – mixážní pult.



Obrázek 2.2: Schéma zapojení jednotlivých komponent do mixážního pultu.

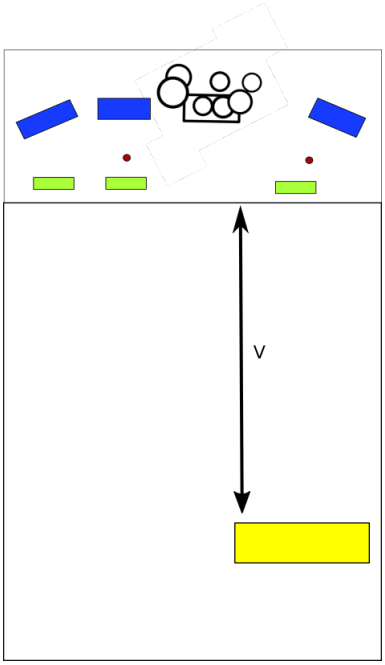
2.3.2 Typický příklad užití části aplikace pro smartwatch

Prvotním záměrem bylo, aby část pro smartwatch nefungovala pouze jako dálkové ovládání metronomu v mobilní části, ale aby také dávalo synchronizovaně vibrační impulsy do ruky. Po testech popsaných v kapitole 5.1 jsem od této části musel odstoupit. Hodinky tedy fungují jako vzdálený ovládač části pro smartwatch. Nicméně základní část tohoto konceptu byla zachována a pokud aplikace není spárována, může být využita jako metronom, který tempo určuje za pomocí vibrací. Vibrace jsou však slabší, což je důvod proč je tento metronom spíše vhodnější pro cvičení na jiný nástroj, než je bicí souprava.

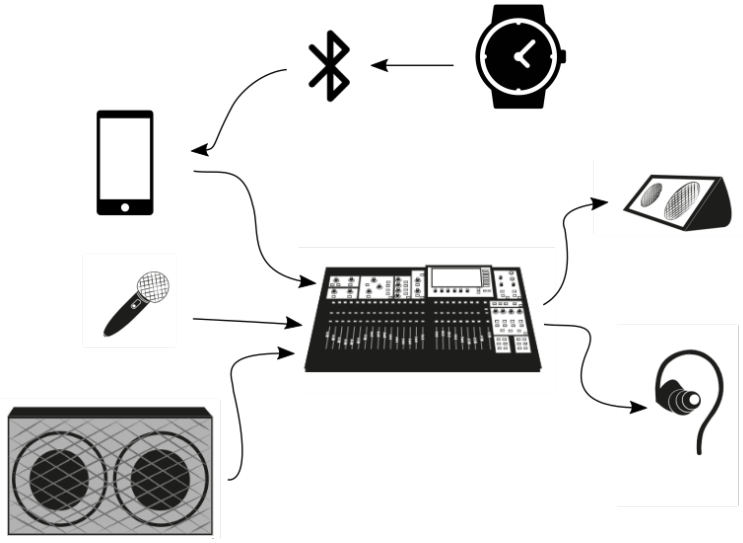
2.3.3 Typický příklad užití spárovaných aplikací

V případě spárování prostřednictvím bluetooth se řeší mnoho problémů se zapojením metronomu do odposlechu. Bubeník může jednoduše a přehledně ovládat spuštěný playlist vzdáleně prostřednictvím hodinek.

Na obrázku 2.3 je zobrazeno schéma rozmístění na středně velké hudební produkci, například ve středně velké koncertní místnosti. Tento příklad užití je vhodný pro bubeníky zvyklé mít v monitoringovém odposlechu zapojený metronom i na koncertech. Z osobní zkušenosti vím, že většina zvukařů neumožní zapojení mobilu přes AUX, jelikož je mixážní pult vzdálen od pódia, tudíž i od bicí soupravy, několik metrů, někdy i několik desítek (znázorněno jako vzdálenost V). Na obrázku 2.4 jenž zobrazuje schéma zapojení, je vidět, že část aplikace pro mobilní telefon je ovládán za pomoci hodinek prostřednictvím bluetooth. Mobil může být tedy přímo u zvukaře a mixážního pultu. Bubeníkovi postačí zvolit před koncertem požadovaný playlist a pak už pouze metronom vzdáleně ovládat. Vzdálenost V je pak ale limitována dosahem bluetooth.



Obrázek 2.3: Schéma rozložení při živé reprodukci hudby, například ve středně velkém klubu. Modrá – kytarové aparáty, zelená – zpěvové odposlechy, červená – mikrofony, žlutá – mixážní pult.



Obrázek 2.4: Schéma zapojení při živé reprodukci hudby, například ve středně velkém klubu.

2.4 Existující řešení, požadavky na produkt a jeho přidaná hodnota oproti existujícím řešením

Na *Google Play* je k dispozici opravdu hodně metronomů. V této kapitole bude představeno a srovnáno pět metronomů, které obsadily první příčky v popularitě. Některé z nich podporují vytváření playlistů. Na první příčky se dostaly i některé ne příliš dobře vizuálně vyvedené kousky, což je důkaz toho, že u metronomu není vzhled vše. Co se týče metronomů na hodinky, těch už tolik není. To je ovlivněno především různými operačními systémy. Nicméně se jedná o vibrační metronomy a zvukové, které nepřinášejí nic navíc. Na konci kapitoly naleznete screenshoty jednotlivých aplikací.

2.4.1 Pulse

Aplikace [30] je nejzákladnější verzí metronomu. Na hlavní obrazovce je zobrazen údaj o BPM. Za pomoci kruhového seekbaru se BPM dají měnit (tato možnost je jediný způsob jak změnit rychlost odklepávání). Celý design na mne působí poněkud futuristicky. Při množství metronomů na *Google Play*, je zajímavé, že podobný metronom se v poměrně velké konkurenci dokáže dostat tak vysoko v žebříčku popularity. Uživateli nabízí pouze možnost změny zvuku, volbu tempa. V každém taktu je pak kladen důraz na první dobu, což některým hudebníkům nemusí vyhovovat. Aplikace se může pochlubit poměrně pěkně zpracovaným grafickým znázorňováním tempa.

Po krátkém testu aplikace jsem objevil zásadní chybu v odbíjení rytmu. Po zastavení a opětovném spuštění odbíjení či změně BPM vzniká v odbíjení dob nepravidelnost, která dělá z tohoto metronomu nepoužitelný produkt. Při vývoji mého metronomu jsem taktéž narazil na podobný problém, který byl odstraněn použitím jiné knihovny pro přehrávání zvuků a změnou zvuků samotných (při rychlejších tempech musí být reprodukovaný zvuk dostatečně dlouhý na to, aby byl dobře rozpoznatelný, ale také nesmí být dlouhý tak, aby s ostatními údery splýval).

2.4.2 Metronome

Metronom od vývojářů Keuwlsoft [26] se vůbec neřídí standartem Material Designu. Celá aplikace má pouze úvodní obrazovku, jejíž značnou část zabírá grafické znázonění klasického mechanického metronomu s reverzním kyvadlem. Kolem tohoto metronomu jsou rozmístěny ovládací prvky (táhla a tlačítka), která jsou příliš malá na to, aby tento metronom mohl být použitelný bubeníkem v praxi. Uživatel má na volbu ze tří základních zvuků. Může si zvolit druh taktu. BPM se dají upravovat pomocí seekbaru, nebo prostřednictvím malých tlačítek *plus* a *mínus*.

Tento metronom je vhodný například pro kytaristy na domácí cvičení. Pro bubeníka je nevyužitelný. Jako zajímavý prvek, který jistě náročnější hudebníci ocení, je zobrazování termínů pro jednotlivé rozsahy BPM. Další poměrně užitečnou funkcionalitou je možnost nastavení *acceleranda*[1] či *ritardanda*[2]. [10]

2.4.3 Metronome Beats

Aplikace [29] se drží standartu Material Design. Oproti předchozí aplikaci je poměrně dobře rozvržená a snadno nastavitelná. Navíc obsahuje i výše zmíněné názvosloví temp a dokonce i možnost acceleranda a ritardanda. Uživatel si může nastavit důrazy na jednotlivé doby a dokonce je možné i zvolenou dobu vynechat. Jedná se tedy o jeden z povedenějších kousků, který v základní funkcionalitě nabízí vše, co by mohl i náročnější hudebník potřebovat.

Tato aplikace obsahuje i placenou část, která by měla funkcionalitu rozšířit o možnost ukládání playlistů. Tuto část bohužel nemohu ohodnotit, protože jsem produkt nezakoupil.

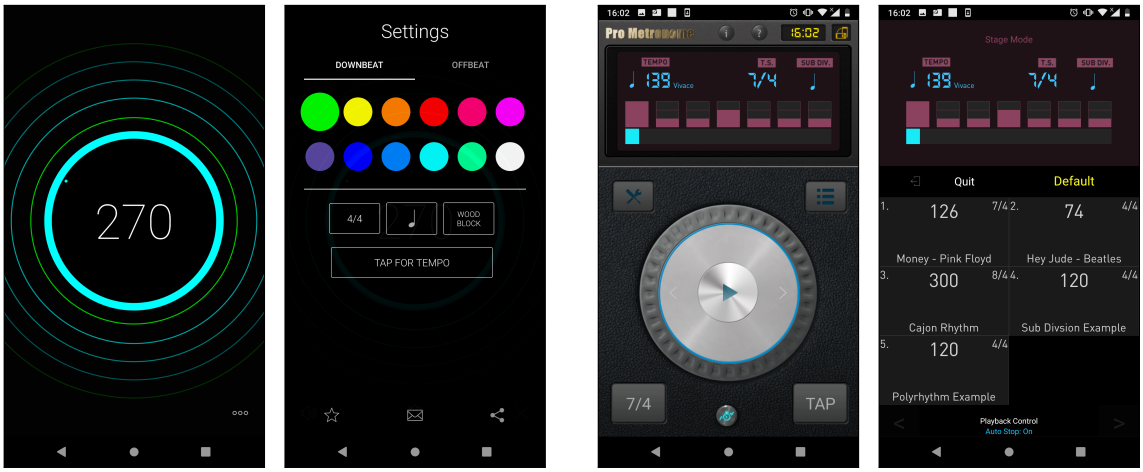
2.4.4 Pro Metronome

Tato aplikace [22] dle mého názoru patří mezi ty povedenější metronomy na *Google Play*. V základní verzi nabízí denní trial, ve kterém je uživateli umožněno ukládání rytmických vzorů, které jsou ukládány do jedné velké knihovny, ze které si je uživatel může opětovně přehrávat. Uživatel si může metronom spustit v tzv. stage módu, který nabízí přepínání temp pomocí dlaždic. Tato aplikace na mne působí poněkud překombinovaně co se designu týče. V plné verzi se však jedná o docela solidní metronom, který splňuje základní požadavky bubeníka.

2.4.5 Soundbrenner

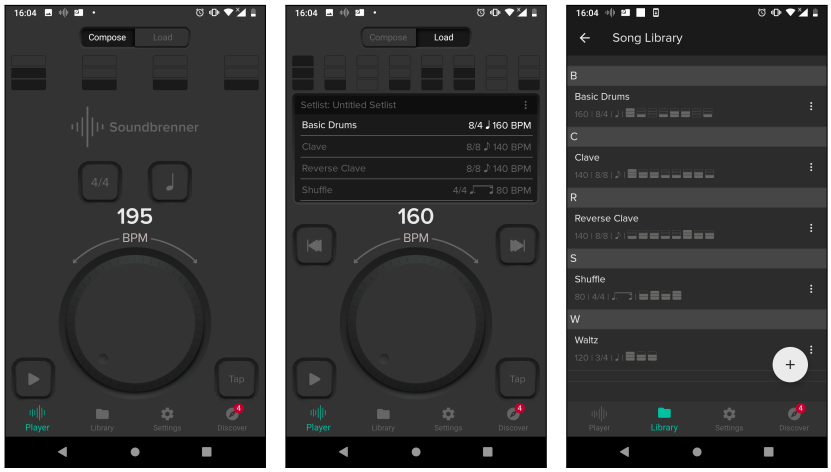
Asi nejlepší metronom na *Google Play* [28] cílený přímo pro bubeníky. Podporuje ukládání playlistů a rytmických vzorů v základní verzi. Celá aplikace má svůj osobitý design, působí velice přehledně. Hlavním tahákem Sondbrenneru je možnost zakoupení speciálního hardweru, který podporuje udávání tempa pomocí impulsu. Tento hardware připomíná hodinky. Dají se zakoupit různé pásky a tak mohou být tyto „hodinky“ připnuty na zápěstí, rameno, nohu či kolem pasu. Jako velké mínus považuji to, že prostřednictvím tohoto zařízení nelze ovládat samotnou aplikaci. Dokonce nezobrazuje hodnotu BPM.

[1] pozvolné zrychlování tempa
[2] pozvolné zpomalování tempa

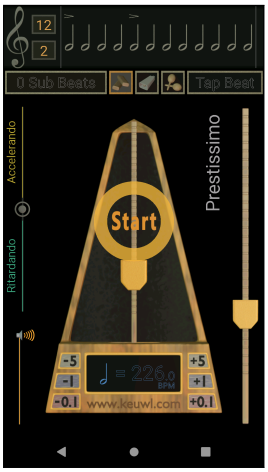


(a) Pulse (b) Pro Metronome

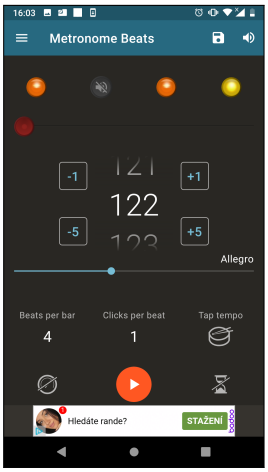
Obrázek 2.5: **(a) Pulse** nabízí pouze jednu možnost k zadání BPM a to pomocí seekbaru, což není příliš šťastná volba (zvolit tempo v časové tísni je takřka nemožné). **(b) Pro Metronome** nabízí i možnost playlistů. V pravo vidíte stagemode, během kterého se playlisty přehrávají. Přepínání mezi rytmy je zprostředkováno pomocí dlaždic. Každá dlaždice reprezentuje jeden rytmický vzor. Tento způsob přehrávání playlistu se mi taktéž nejeví jako příliš komfortní.



Obrázek 2.6: **Soundbrenner** má dle mého názoru nejlépe zpracované UI a to jak přehledností tak i designem. Volba BPM je opět umožněna pouze kruhového táhla (viz. obrázek vlevo). Uprostřed vidíte stagemode, který je oproti Pro Metronome přehlednější a použitelnější. Playlist je uspořádaný a uživatel jednotlivé rytmické vzory v něm obsažené přepíná pomocí tlačítek *next* a *back*, jako je tomu například u klasických přehrávačů hudby. Tlačítka jsou ale dle mého názoru příliš malá. Vpravo je vidět knihovna rytmů, každý rytmický vzor je přehledně vyobrazen jako element listu.



(a) Metronome (Keuwlsoft)



(b) Metronome Beats

Obrázek 2.7: **(a) Metronome** nabízí metronom pro náročnější hudebníky. Jeho UI však není vhodné pro bubeníky na přehrávky kapely a už vůbec ne na živé vystupování. **(b) Metronome Beats** se částečně drží *Material designu*, jak je vidět ze záhlaví aplikace. Volba tempa je umožněna několika způsoby. Uživatel si musí připlatit, aby se zbavil všudepřítomných reklam a zpřístupnilo se ukládání playlistů.

2.4.6 Můj výsledný metronom

Můj produkt se zaměřuje především na bubeníky. Snažím se tedy o co nejlépe ovládatelné UI, se kterým se dá lehce zacházet i v časové tísni a při fyzické zátěži. Z předchozích hodnocení aplikací je vidět, že ani možnost využívání playlistů není úplnou samozřejmostí. Moje aplikace tuto možnost nabízí. Největší přidanou hodnotou oproti ostatním aplikacím však patří možnost vzdáleného ovládání, které umožní zapojení popsané v podkapitole 2.3.3. Více dopodrobna bude mnou navržené UI popsáno v kapitole 4, kde naleznete i screenshoty aplikace.

Kapitola 3

Tvorba mobilních aplikací pro platformu Android

V této kapitole bude stručně popsáno, jak se dnes vyvíjí aplikace pro zařízení s *OS Android*. Předtím bych ale rád pár řádků věnoval samotnému operačnímu systému Android.

3.1 Operační systém Android

Android [1] je open-source operační systém, který běží na Linuxovém jádru a je vyvíjený firmou Google. Snad i díky faktu, že jde o open-source projekt, se jedná o OS, kterým jsou aktuálně nejčastěji osazovány chytré mobily. Podíl Androidu na trhu mobilních telefonů se aktuálně odhaduje na zhruba 80% [15]. S Androidem se však nepotkáme pouze u smartphonů ale i tabletů, chytrých televizí či hodinek.

I vysoká popularita *OS Android* má za následek opravdu velký výběr aplikací na *Google Play*, což je zcela jistě výhodou pro každého uživatele smartphonu. Množství aplikací se však někdy odráží na jejich kvalitě. To ovšem nemusí být pravidlem a na *Google Play* najdeme mnoho aplikací, které jsou srovnatelné s těmi nabízenými například uživatelům *iOS*. Mezi další výhody patří fakt, že spotřebitel má mnohem větší výběr zařízení na trhu. To sebou nese nevýhodu pro vývojáře, který musí počítat s tím, že jím vyvíjený software musí být kompatibilní s různými zařízeními (různá velikost displaye atd.).

Android je jeden z nejrychleji se vyvíjejících operačních systémů a za jeho krátkou dobu existence urazil dalekou cestu. Jeho rychlý vývoj sebou však nenese pouze pozitiva. S příchodem nových verzí se objevují i nové funkce, knihovny a technologie, které na starších verzích OS nemusí být podporovány. Z toho důvodu musí každý vývojář brát zřetel na to, jestli jeho aplikace má být cílena pro co největší okruh uživatelů (tedy i pro nižší API Level) či pro nejaktuálnější Android.

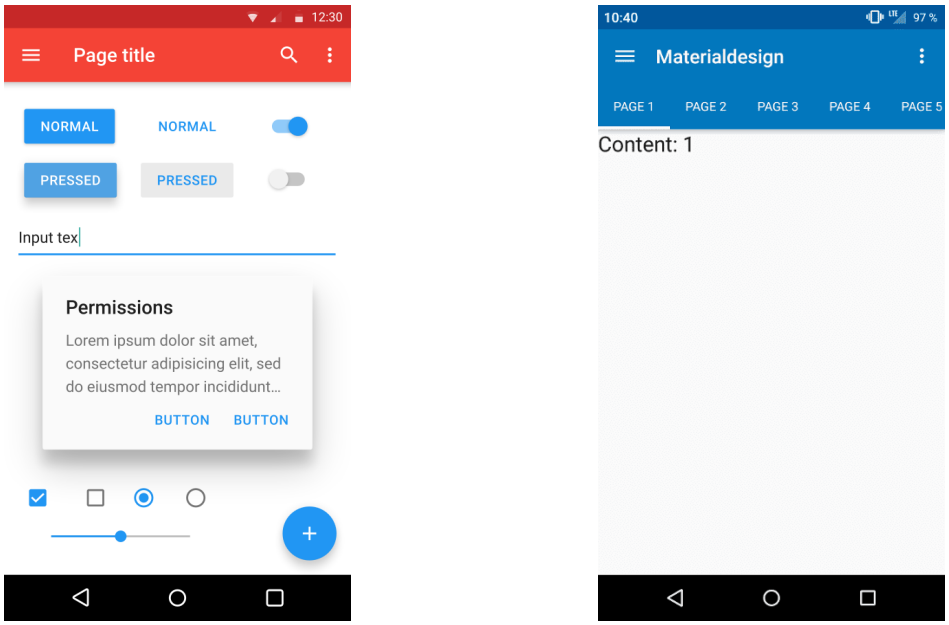
Pro mou práci není důležitý Android pouze na smartphone, ale i na smartwatch. Za tímto účelem vyvinula firma Google OS s názvem *Wear OS*, který byl donedávna známý pod názvem *Android Wear*[1]. Celkově se na našem trhu moc hodinek s *Wear OS* neprodává a pokud ano, jsou příliš drahé a jako student jsem si je nemohl dovolit. Naštěstí se na trhu dají sehnat hodinky, které neběží na oficiálním *Wear OS*, ale na jeho napodobeninách, kterou díky open-source licenci Androidu používá mnoho firem. V mém případě se jedná o hodinky Kingswear 88, které využívají modifikovaný Android verze 5.1.

[1] změna názvu proběhla v březnu roku 2018

3.2 Současný trend ve vývoji UI na platformu Android

V poslední době se uživatel Androidu stále častěji setkává s aplikacemi, které se řídí *Material Designem* [8]. Material design je standart a designový jazyk, který byl vyvinut společností Google v roce 2014. To znamená, že se dá využít u zařízení s API Level 21, tedy Android 5.0 a výše. Pokud vyvíjíte pro zařízení s nižším API, je potřeba využít knihovnu *v7appcompat*. Hlavní myšlenkou je sjednotit používané ovládací prvky, jako například tlačítka, záložky, spinnery, seekbary či checkboxy. V ideálním případě by pak každá aplikace byla velice podobná co se ovládání týče a pro uživatele o to intuitivnější. Díky častému používání, mi Material design přišel jako nejvhodnější pro vývoj mé aplikace. Elementy, které tento design využívá, se také snadno aplikují při implementaci.

Celý koncept Material designu je založen na přiblížení k realitě. To znamená, že aplikace by se měla chovat jako z reálného světa. To můžeme vidět například na stínování elementů, drobných animacích například při kliknutí atd. Dobrým příkladem jsou třeba záložky [9], které se dají přepínat i pomocí gesta swipe, které evokuje odhození záložky na jednu či druhou stranu. Zároveň při přepnutí stránky je vidět animace, která nám ukazuje kam „odhozená“ záložka zmizela. Lze tedy očekávat, že pokud tuto záložku budeme hledat, najdeme ji právě na této straně. Přehled pro mne zajímavých elementů Material designu najdete na obrázku 3.1.



(a) (b)

Obrázek 3.1: Na screenshotu **(a)** [24] můžeme vidět základní komponenty jako jsou switch, button, edit text, allertdialog, checkbox, floating button či seekbar. Všechny tyto prvky byly v mé práci využity. V horní části obrazovky je pak typický action bar, který obsahuje menu pro akce. Obrázek **(b)** [23] obsahuje typické záložky, které jsem použil taktéž, v knihovně playlistů a rytmů.

3.3 Základy aplikace pro Android a základy jejího vývoje

V dnešní době se aplikace pro Android a jiné OS pro chytré telefony dají vyvíjet za pomoci mnoha technologií. Vývojáři se snaží zpřístupnit vývoj i pro méně zdatné uživatele a uživatele, kteří nemají velké zkušenosti s programováním. Za tímto účelem vznikají nástroje jako Appypie či Appsbar, ve kterých je schopný obstojnou aplikaci vyvinout i naprostý laik. [21]

Pokud ale chceme vyvíjet aplikaci opravdu na úrovni, programování se nevyhneme. Dva hlavní proudy vývoje pro Android jsou firmy Microsoft a Google. Microsoft spolu se svým produktem Visual Studio 2015 uvedl i rozšíření pro programovací jazyk C# a to Xamarin [16]. Za pomoci Xamarinu můžeme vyvíjet aplikace jak pro Android tak i *iOS*. Jak se od Microsoftu dá očekávat, tak Xamarin podporuje i vývoj aplikací pro dnes již téměř nepoužívaný Windows Phone. Já jsem si pro vývoj aplikace zvolil vývojářské prostředí firmy Google a to *Android studio* [3]. *Android studio* bylo představeno roku 2013. Jedná se o náhradu klasického ADT[2]. Celé vývojové prostředí je založeno na JetBrains' IntelliJ IDEA. Vývoj aplikací je možný v programovacích jazycích Java a Kotlin. Třetí variantou je pak jazyk C/C++ za využití Java NDK.

3.3.1 Základní skladba aplikace z pohledu vývoje v Android Studiu

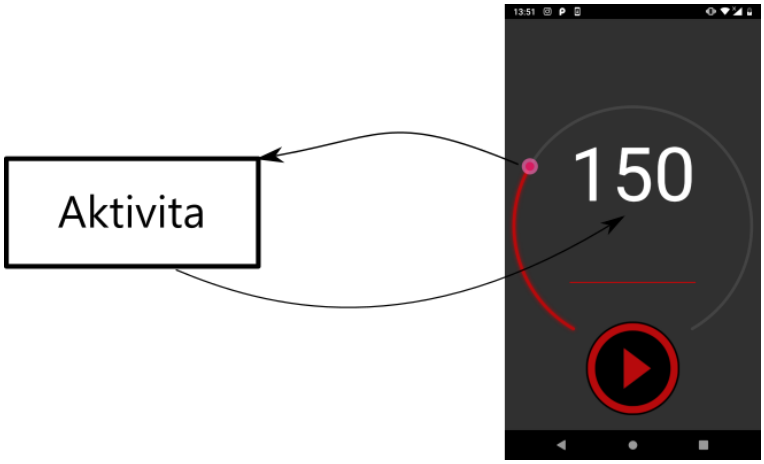
Každá aplikace se skládá ze dvou základních elementů a to aktivity a layoutu. Aktivitou nazýváme instanci třídy `Activity`, která je obsažena v Android SDK. Tato třída má za úkol zprostředkovávat interakci mezi uživatelem a samotnou aplikací prostřednictvím obrazovky. Aplikace může pak obsahovat několik instancí podtříd `Activity` v závislosti na složitosti aplikace. Zpravidla má však každá aplikace alespoň jednu aktivitu, kterou nazýváme hlavní. Každá aktivita má pak svůj tzv. životní cyklus. Ten se skládá ze čtyř stavů:

- Nonexistent
- Stopped
- Paused
- Resumed

Mezi těmito stavy pak aktivita přechází za pomoci metod jako je například `onCreate()`, `onDestroy()` atd.

Layout je množinou elementů uživatelského rozhraní. Je tvořen souborem formátu XML, například `activity_main.xml`. XML soubor obsahuje definice jednotlivých objektů. Pomocí těchto definic aktivita vykreslí layout na obrazovku zařízení. Nejprve je layout zpracován aktivitou během jejího přechodu ze stavu Nonexistent do stavu Stopped prostřednictvím metody `onCreate()`. V tomto stádiu je aktivita uložena do paměti. Metoda `onStart()` pak zajistí přechod do stavu Paused. Tento stav už patří mezi stavy viditelné. Poslední stav Resumed patří do části tzv. Foreground lifetime. V tomto stavu je aktivita připravena na interakci s uživatelem. Více o životním cyklu aktivity a layoutech naleznete v knize Android Programming [27], ze které jsem čerpal i během implementace.

[2] Eclipse Android Development Tools



Obrázek 3.2: Obrázek demonstruje interakci aktivity s uživatelem prostřednictvím obrazovky. Vzhled obrazovky je dán layoutem. Manipulace s objektem UI jako je v tomto případě seekbar, je zaznamenána aktivitou. Aktivita po zpracování události může změnit obsah jiného objektu, jako je v tomto případě textové pole nebo vykonat jakoukoliv jinou činnost.

Kapitola 4

Návrh GUI výsledné aplikace

Během používání a analýzy ostatních metronomů dostupných na *Google Play* a konzultací s Janem Poláškem (aktivní poloprofesionální bubeník) jsem našel mnoho inspirací a pokusil jsem se navrhnout takové GUI, které bude konkurence schopné ostatním metronomům a bude se především zaměřovat na cílovou persónu, tedy bubeníka.

Za tímto účelem by GUI mělo obsahovat pouze takové prvky, které bubeník opravdu potřebuje pro fungování na přehrávkách s kapelou a v ostrém provozu při hudební produkci na koncertech. Prostředí by mělo být přehledné, intuitivní a lehce ovladatelné v jakékoliv situaci, do které se bubeník může dostat. V dalších podkapitolách bude čtenář seznámen s GUI aplikace určené pro smartwatch i smartphone.

4.1 Hlavní prvky GUI bubenického metronomu

Mým hlavním cílem bylo eliminovat co nejvíce úkonů, které jsou pro bubeníky zbytečné. Výsledné rozhraní se snaží především o snadné vytvoření či editaci rytmického vzoru a jeho následné zařazení do playlistu. Dalším důležitým cílem je snadné přehrávání takto vytvořených playlistů.

4.1.1 Hlavní prvky GUI – část pro smartphone

Na základě průzkumu v rámci týmového projektu do předmětu ITU[1], jehož zadáním byl vývoj jednoduchého metronomu pro smartphone, jsem se rozhodl použít podobný koncept hlavního GUI.

Při spuštění metronomu by mělo být umožněno uživateli nastavit nový rytmický vzor. Hodnota BPM by měla být umožněna zadat několika způsoby; konkrétně pomocí tlačítek, progress baru a editovatelného textového pole. Dále by měla být umožněna volba zvuku úderu, možnost zvolení počtu úderů v taktu. Pro jednotlivé údery by měla být umožněna volba důrazu, či vynechání úderu.

Dalším důležitým prvkem části pro smartphone je aktvivita v rámci které jsou spravovány uložené playlisty a jednotlivé rytmické vzory. Tato část by se měla řídit standardem Material Designu. Playlisty a knihovna uložených rytmických vzorů pak bude reprezentována pomocí standardních seznamů.

Poslední část GUI je aktivita, kterou budu dále nazývat *stage mode*, jelikož tento název nejlépe vystihuje její podstatu. Aktivita by měla obsahovat jednoduché rozhraní, které

[1] Tvorba uživatelského rozhraní

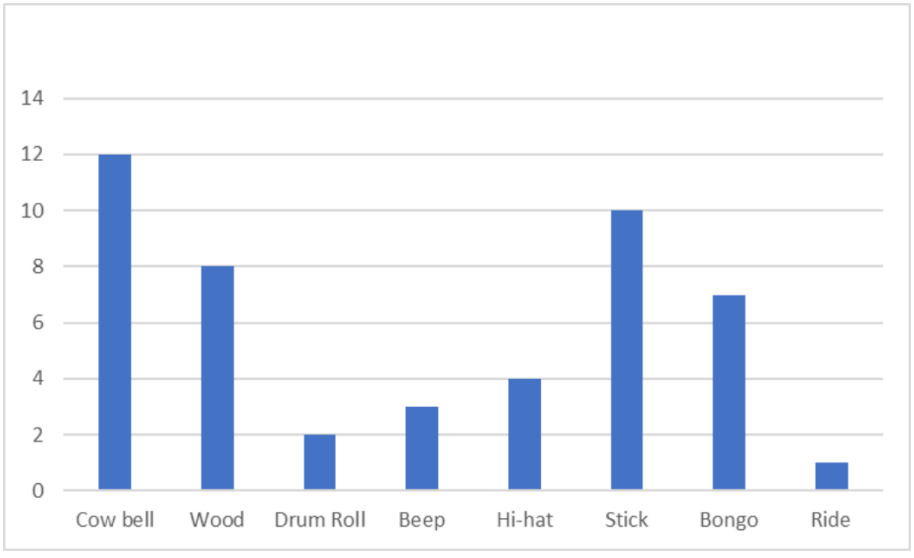
bubeníkovi umožní rychlé přepínání temp v rámci playlistu. Důležitá jsou velká tlačítka a vyobrazování pouze důležitých informací. Se stage modem je úzce spjata komunikace prostřenictvím bluetooth. GUI by tedy mělo obsahovat i možnost vybrání párovaného zařízení, jímž bude metronom vzdáleně ovládán.

4.1.2 Hlavní prvky UI – část pro smartwatch

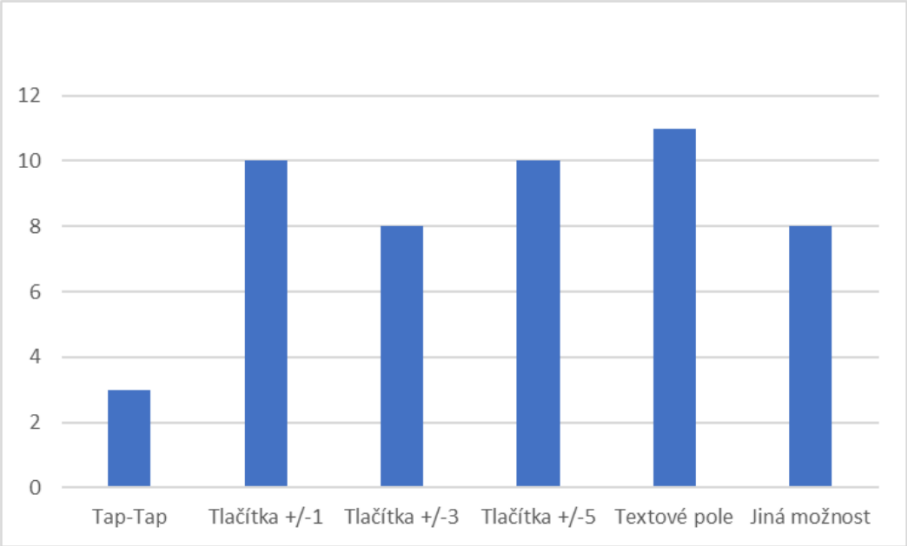
Část metronomu, která je určená pro smartwatch by měla sloužit jako vzdálené ovládání metronomu. Je třeba udělat hlavní ovládací prvky neúměrně velké vůči rozměru hodinek. To z důvodu snadného ovládání. Mezi jednotlivými obrazovkami by se mělo pohybovat pomocí záložek, které však budou skryté (bude se využívat pouze gesto *„swipe“*). To by mělo ušetřit prostor, kterého je na obrazovce hodinek poskrovnu. Jedna z těchto obrazovek by uživateli měla nabízet základní vibrační metronom. BPM se budou opět volit pomocí editovatelného textového pole a pomocí progress baru kruhového tvaru, který se na hodinkách vyloženě nabízí. Další obrazovka bude sloužit jako vzdálené ovládání metronomu. Aby toto ovládání mohlo být aktivováno, je zapotřebí zprovoznit propojení prostřednictvím bluetooth, které nám umožní třetí obrazovka. Zde si uživatel vybere se kterým zařízením chce hodinky párovat.

4.2 Návrh a finální verze UI bubenického metronomu

Jak už jsem se zmínil výše, tak v rámci předmětu ITU, jsem se spolužákem Pavlem Nováčkem, implementoval a testoval základní metronom. Díky tomu jsem měl k dispozici data z dotazníku, který jsme v rámci tohoto předmětu vytvořili a který byl zodpovězen dvanácti respondenty. Tato data jsem využil při implementaci základní funkcionality metronomu. Konkrétně se jednalo o dvě otázky ohledně zvuku úderů a možnosti nastavování tempa. Otázky, které jsem z tohoto dotazníku použil a náležité grafy výsledků naleznete níže.



Obrázek 4.1: Respondentům byly přehrány jednotlivé zvuky, které byly staženy z volně dostupné knihovny zvukových stop [12]. Akcenty (důrazy) jednotlivých samplů byly upraveny v softwaru Audacity [19]. Dotazovaní měli odpovídat, jestli by přehrávaný zvuk uvítali v metronomu.



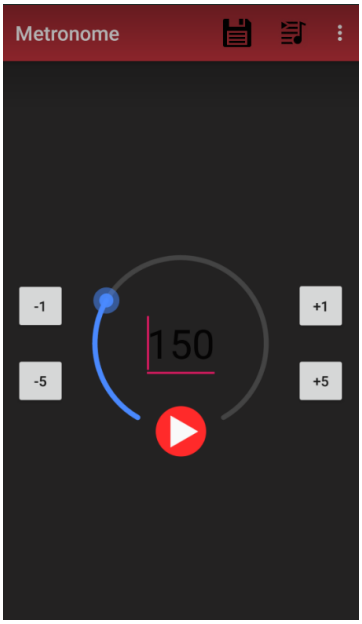
Obrázek 4.2: Respondentům byla položena otázka, jak by mělo být možno nastavit BPM. Tap-Tap tempo bylo z původního návrhu na základě výsledků vyřazeno. Jedná se o možnost kdy, uživatel kliká na tlačítko v požadovaném rytmu. Z několika kliknutí se poté spočítá frekvence odbíjení rytmu. Respondenti uváděli, že by uvítali i jinou možnost a to konkrétně seekbar.

4.2.1 Iterativní vývoj UI

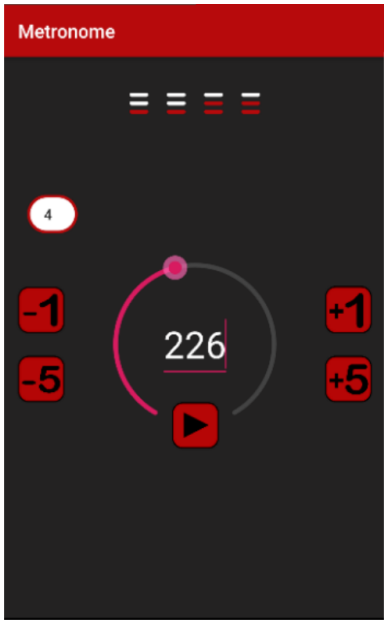
Finálního UI je výsledkem několika iterací testování metronomu v provozu a mnohých konzultací s Janem Poláškem. Během vývoje jsem se několikrát dostal do slepé uličky a nebo bylo mnou navržené řešení nepřijatelné pro cílového uživatele. Jako základ mojí aplikace jsem použil hlavní aktivitu a s ní několik souvisejících tříd z projektu do předmětu ITU. Tyto třídy byly implementovány mnou a jelikož se mi výsledný základní design a funkcionalita zamlouvala, rozhodl jsem se část zdrojového kódu recyklovat a využít i v rámci bakalářské práce. Zdrojové soubory, které jsou částečně převzaty z tohoto projektu jsou pak označeny ve zdrojovém kódu pomocí hlaviček.

Z výše uvedených dat jsem měl představu, jak by se mělo zacházet s metronomem v jeho nejzákladnější podobě, tedy bez ukládání playlistů atd. Za pomoci aplikace Inkscape jsem vytvořil základní mock-up, který byl prezentován Janu Poláškovi. Po konzultaci s ním byl mock-up přepracován a vznikl první návrh základního UI mého metronomu (obrázek 4.3). Můžete si povšimnout absence možnosti úpravy akcentu jednotlivých dob či jejich vynechání. Jan Polášek totiž označil tuto funkcionalitu za pro něj nepotřebnou. V pozdější fázi jsem se ale rozhodl tento „Accent bar“ přidat, protože po konzultaci s jinými hudebníky mi bylo sděleno, že tuto možnost někdy využívají. Navíc je tato funkcionalita součástí téměř každého z konkurenčních metronomu.

Na základě tohoto mock-upu začala implementace aplikace. Na pravidelných přehrávkách kapely *Of Lost Things*, tedy dvakrát týdně, byly představeny změny v aplikaci Janu Poláškovi. Za pomoci jeho zpětné vazby začaly vznikat různé verze a prototypy aplikace.



(a) Mock-up



(b) První implementace základní funkcionality

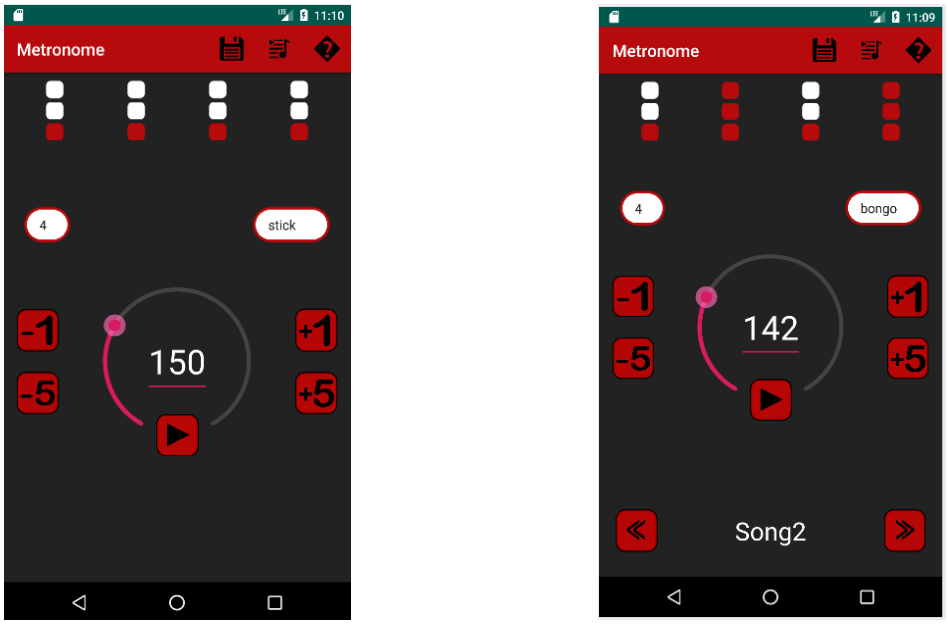
Obrázek 4.3: **(a) Mock-up** metronomu, který vznikl při konzultacích s Janem Poláškem. Oproti původní vezi, kterou jsem mu představil, byl přidán kruhový seekbar, jenž je k dispozici u většiny metronomů na *Google Play* a odebrán Accent bar. Na obrázku **(b) První implementace základní funkcionality** si můžete všimnout, že byly implementovány i akcenty a volba počtu dob v taktu (volba je umožněna pomocí spinneru v levé části obrazovky). Implementace se jinak drží původního mock-upu.

První funkční prototyp podporující práci s playlisty

První funkční prototyp, který umožňoval práci s playlisty byl výstupem mého semestrálního projektu. Do této fáze jsem se seznamoval s pro mne novým programovacím jazykem Java a Adnroid Studiem. Ukládání playlistů a rytmů, především pak jejich zobrazení v listu, se neřídí *Material Designem*. Aplikace splňuje cílovým uživatelem požadovanou funkcionalitu, ale její ovládání je poněkud nepřehledné a komplikované.

Struktury knihovny rytmů a playlistů je v této verzi aplikace následující: uživatel vytvoří požadované množství playlistů. Do těchto playlistů se pak ukládají jednotlive rytmické vzory. Rytmický vzor se však váže přímo na danný playlist, do kterého byl při jeho vytvoření přidělen. Tento koncept byl však Janem Poláškem zavržen. Dle jeho slov je důležité, mít možnost jeden rytmický vzor využít ve více playlistech a pro uživatele je nekomfortní vytvářet stejný vzor vícekrát z důvodu, že nelze recyklovat.

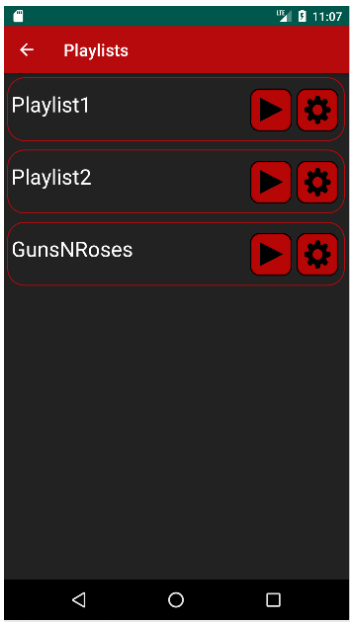
Tento prototyp byl testován na zkouškách a jedenkrát i při živém vystoupení. Při těchto testech jsme zjistili, že přehrávání playlistů v rámci hlavního okna (viz. obrázek 4.4b) není příliš komfortní. Tlačítka jsou malá a během koncertu není pohodlné mezi rytmy přepínat. Z toho důvodu byl v dalších verzích implementován stagemode jako je tomu například v aplikaci Soundbrenner.



(a) Hlavní okno

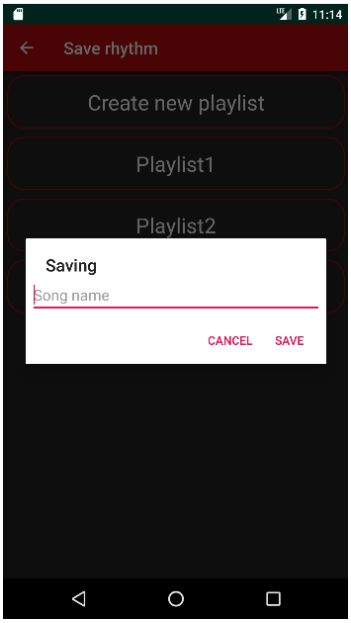
(b) Přehrávání playlistu

Obrázek 4.4: **(a) Hlavní okno** je rozšířeno o volbu zvuku. Každému uživateli nemusí vyhovovat zvolený zvuk. Zvuky jsou vybrány na základě zodpovězené otázky viz. 4.1. Action bar pak nabízí přístup do knihovny playlistů a uložení vybraného zvuku. **(b) Přehrávání playlistů** se ukázalo jako nepraktické.



(a) Knihovna playlistů

(b) Editace playlistu



(c) Ukládání rytmického vzoru

Obrázek 4.5: **(a) Knihovna playlistů** i **(b) Editace playlistu** neodpovídá standardu Material designu. Působí poněkud nepřehledně. Manipulace s playlisty či rytmy probíhá prostřednictvím tlačítek. Například předřazení písně Estranged písni Paradise ciy docílíme třemi kliknutími na tlačítko, které prezentuje šipka nahoru. V dalších implementacích jsem se rozhodl využívat Material design a jeho formu klasických listů, které jsou uživatelsky mnohem přívětivější. **(c) Ukládání rytmického vzoru** je pak zprostředkováno allert dialogem.

Výsledný metronom – aplikace pro smartphone

Výsledná aplikace pro smartphone si ponechala osvědčené hlavní okno s drobnými úpravami tlačítek a spinnerů (nastavení správného fontu písma dle Material Designu). Metronom již podporuje vzdálené ovládání pomocí bluetooth. Spárování aplikací a zařízení pak spustíme výběrem ikonky bluetooth v action baru.

Ponechán byl způsob ukládání nových rytmů. Uživatel si na základní obrazovce navolí požadované BPM, doby, akcenty a zvuk. Poté vzor uloží jednoduše pomocí ikonky diskety, která je obsažena v action baru. Tento způsob mi přijde praktičtější než u ostatních metronomů, jako třeba Soundbrenner, kde uživatel vyplňuje hodnoty do dialogového okna. Během zadávání těchto hodnot si vytvářený rytmický vzor nemůže spustit a odzkoušet.

Dalším velkým rozdílem je, že rytmický vzor již není svázán pouze s jedním playlistem. Aplikace nyní disponuje dvěma knihovnami: knihovnou playlistů a knihovnou rytmických vzorů. Rytmický vzor pak může být součástí libovolného počtu playlistů, čímž byla umožněna jeho recyklovatelnost, která v předchozích verzích cílovému uživateli chyběla. Prostředí, ve kterém je manipulováno s těmito knihovnami se pak řídí *Material Design*em a od výše uvedené verze je dle mého názoru a i dle názorů testerů přehlednější a intuitivnější.

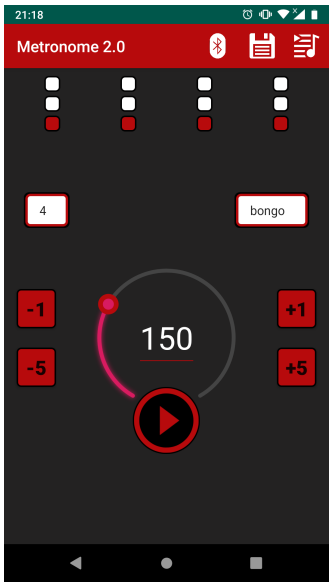
Ve finální verzi se také přehrávání playlistů oddělilo od hlavní obrazovky metronomu a vznikla samostatný stagemode. Oproti jiným metronomům jsem se snažil klást důraz na snadné použití při živém hraní, tzn. tlačítka jsou co největší, zobrazovány jsou jen podstatné údaje (název vzoru a BPM, následující a předchozí vzor).

Celý design aplikace je sice minimalistický, ale to má za důsledek jednoduché a intuitivní ovládání, což bylo na začátku vývoje metronomu jednou z hlavních priorit. Aplikace obsahuje veškerou funkcionalitu, která byla požadována. Výsledkem je tedy metronom, vytvořený na míru bubeníkovi.

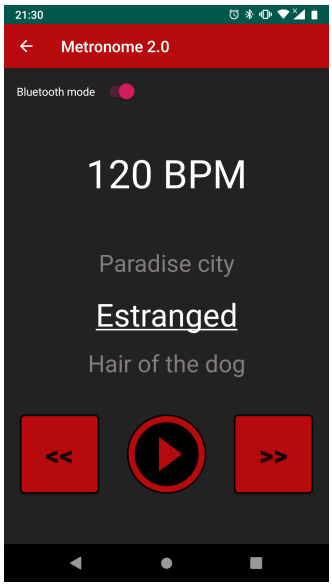
Výsledný metronom – aplikace pro smartwatch

Aplikace pro smartwatch se snaží držet designu hlavní aplikace pro smartphone. Celou aplikaci tvoří tři záložky. Z důvodu šetření místa tyto záložky nemají svoje typické záhlaví. Můžeme mezi nimi tedy přepínat pouze pomocí gesta swipe. Jedna ze záložek slouží k spárování zařízení a aplikací pomocí bluetooth. Další, která se zobrazuje při startu aplikace pak funguje jako klasický metronom s tím rozdílem, že negeneruje zvukové impulsy, nýbrž impulsy vibrační. Třetí záložka pak slouží jako vzdálené ovládání stagemódu spuštěném v aplikaci pro smartphone na druhém spárovaném zařízení. Původně bylo zamýšleno, že by hodinky generovaly vibrační impulsy i během tohoto vzdáleného ovládání, ale bohužel se mi nepodařilo synchronizovat zvuk generovaný mobilním telefonem s vibrací generovanou hodinkami.

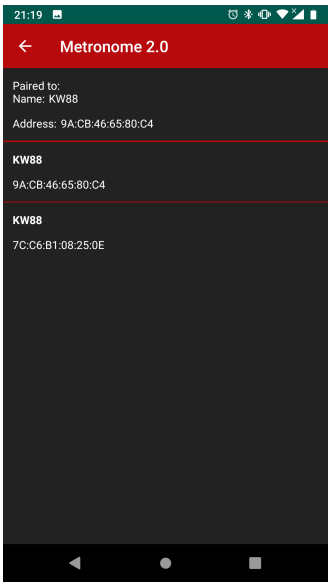
Při testování vibračního metronomu Janem Poláškem, bylo zjištěno, že vibrační impulsy nejsou dostatečně silné a při bubnování se jimi nedá řídit. Vibrační metronom je tedy spíš pouze jakýmsi doplňkem vzdáleného ovládání, který mohou využít hudebníci, kteří při hraní na svůj nástroj nejsou vystaveni takové fyzické námaze a vibrace jsou schopni vnímat (jako například pianisté i kytaristé). V potaz se ale musí také brát to, že aplikace byla testována pouze na hodinkách Kingswear 88. Jiný hardware může poskytovat lepší možnosti a silnější vibrace.



(a) Hlavní okno

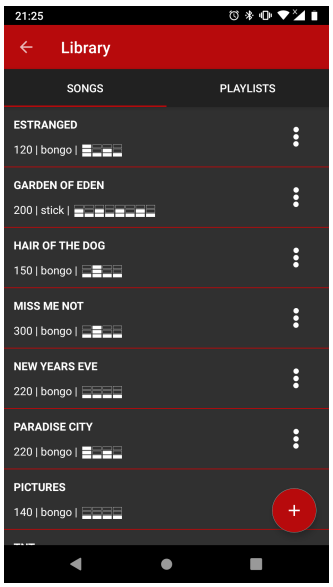


(b) Stagemode

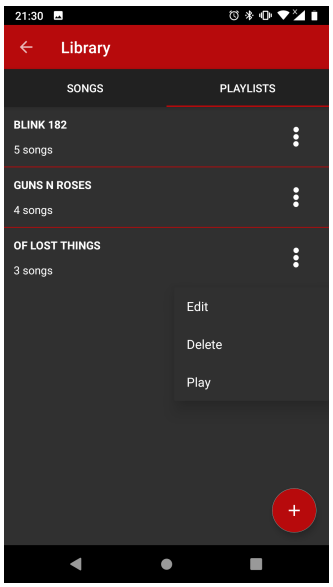


(c) Nastavení spojení

Obrázek 4.6: **(a) Hlavní okno** prodělalo drobné designové změny. **(b) Stagemode** disponuje velkými tlačítky a jednoduchým ovládáním. Uživateli zprostředkovává pouze důležitá data. V levé části můžeme za pomoci switche spustit režim vzdáleného ovládání. **(c) Nastavení spojení** uživateli zobrazí v listu zařízení se kterými je možno navázat spojení prostřednictvím bluetooth.

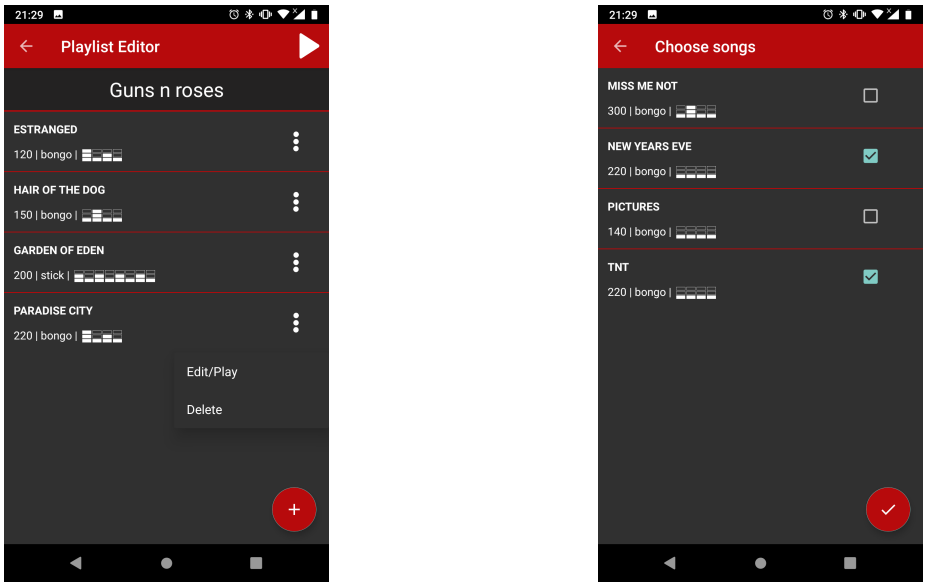


(a) Knihovna rytmických vzorů



(b) Knihovna playlistů

Obrázek 4.7: Mezi knihovnami se přepíná pomocí záložek. Záložky pak obsahují listy s obsahem jednotlivých knihoven. Každá položka je přehledně graficky prezentována (důrazy, BPM atd.), podobně jako je tomu u metronomu Soundbrenner. Pro přidání nových položek je využito plovoucího tlačítka *plus*.



(a) Editace playlistu (b) Knihovna playlistů

Obrázek 4.8: Pořadí rytmických vzorů se dá nyní v playlistu lehce měnit pomocí gesta *drag and drop*, tedy potažení elementu listu na požadovanou polohu. Pomocí plovoucího tlačítka *plus* se spustí výběr rytmických vzorů které chceme přidat do playlistu. K výběru poslouží checkboxy a potvrzovací plovoucí tlačítko.

(a) Vzdálené ovládání (b) Vibrační metronom (c) Nastavení spojení

Obrázek 4.9: **(a) Vzdálené ovládání** obsahuje pouze potřebné prvky. Tlačítka jsou co největší, aby bylo ovládání jednoduché a rychlé. **(b) Vibrační metronom** má podobný design jako hlavní obrazovka v aplikaci pro smartphone. Uživatel může BPM zadat pomocí seekbaru a textového editovatelného pole. **(c) Nastavení spojení** je pak prakticky stejné jako u hlavní aplikace.

Kapitola 5

Testování bubenického metronomu

Veškeré níže uvedené testy byly prováděny na následujících zařízeních:

- Nokia 6.1
- Nokia 6.0
- Xiaomi Mi 8 Lite
- Huawei P9 lite Mini
- Kingswear 88

Jedná se o telefony a hodinky pohybující se v nižší a střední cenové relaci. Díky několika typům smartphonů s různými verzemi *OS Android* se otestovala i kompatibilita s různými stroji. Co se týče hodinek měl jsem k dispozici pouze Kingswear 88 a tak jsem mohl kompatibilitu testovat pouze na virtuálních strojích. V tomto případě se však těžce testuje jak metronom dávající vibrační impulsy, tak i vzdálené ovládání prostřednictvím bluetooth.

5.1 Mnou zvolený testovací protokol

Jak už jsem uvedl, část testů a to konkrétně krátký dotazník byl převzat z projektu do předmětu ITU. Tato část mi však poskytla jen základní zpětnou vazbu. Nejpřínosnější částí testů bylo iterativní real-world testování, které bylo prováděno na přehrávkách kapely *Of Lost Things* a konzultace s Janem Poláškem, jenž je bubeníkem v této kapele. Jednotlivé iterace testů se konaly pravidelně od počátku vývoje aplikace. Tím, že hlavním testerem aplikace je aktivní bubeník, který má zkušenosti s téměř všemi zmíněnými konkurenčními aplikacemi, si troufám tvrdit, že výsledný produkt je implementován na míru každému bubeníkovi. Neméně důležitou zpětnou vazbou byly pravidelné konzultace s panem profesorem Adamem Heroutem.

Mimo toto testování přímo v provozu, jsem sestavil krátké scénáře. Uživatel procházející tyto scénáře měl ověřit správnou funkčnost. Scénáře byly zaměřeny především na výše zmíněné typické příklady užití (kapitola 2.3). Největším testem pak bylo vystoupení mé kapely na Brněnském Majálesu.

5.2 Realizace testů

Všechny následující testy byly prováděny mnou a členy kapely *Of Lost Things*. Tím jsem docílil toho, že docházelo i k nechtěným akcím, kterých by se během testování pouze vyvojářem nepovedlo docílit. Zároveň se jedná o testery, kteří jsou aktivními hudebníky – rozumí tedy dané problematice a mají zkušenosti i s jinými metronomy.

5.2.1 Test číslo 1 – Funkční testování tvorby playlistů

Tento test byl složen ze dvou částí. Uživatel, tedy tester, v první části dostal za úkol seznámit se s částí metronomu určeného pro smartphone. Po krátkém seznámení se s prostředím aplikace byl uživateli předložen seznam rytmických vzorů, které měl uživatel přidat do knihovny existujících temp. Po naplnění knihovny deseti vzory měl uživatel za úkol vytvořit playlist, který obsahoval několik z uložených vzorů. Playlist je implicitně řazen abecedně podle názvu vzorů. Uživatel měl za úkol seřadit playlist podle BPM vzestupně od nejpomalejšího tempa. Z knihovny rymických vzorů se v následujícím kroku testu měly odstranit dva vzory. Na konci testu byl zkontrolován výsledný playlist a porovnán s očekávanými daty, které by měl takto sestavený playlist obsahovat.

Tento test měl za úkol otestování funkcionality vytváření playlistů a práce s knihovnou rytmických vzorů. Zároveň jednotlivé kroky nebyly dopodrobna popsány a měly tedy ověřit i to, že je aplikace přehledná a intuitivní. Test potvrdil, že aplikace požadovanou funkcionalitou disponuje. Test dopadl u všech čtyř testerů zdárně (výstupní data se shodovala s očekávanými), proto byl prohlášen za úspěšný.

5.2.2 Test číslo 2 – Funkční testování propojení aplikací

Zadáním druhého testu bylo zprovoznit propojení smartphonu a hodinek za pomoci technologie bluetooth. Každý tester měl k dispozici svůj vlastní mobil a zapůjčené hodinky Kingswear 88. Tento test se prováděl ve dvou iteracích. Při prvním cyklu testování se sice všem testerům podařilo navázat komunikaci prostřednictvím bluetooth, ale až po podrobnějším vysvětlení jak při párování aplikací postupovat. Z tohoto důvodu jsem se rozhodl aplikaci upravit a při prvním párování zobrazovat malou nápovědu. Během aktivity, která zprostředkovává párování, má nyní uživatel možnost zobrazit i nápovědu, která by měla výše zmíněný problém eliminovat.

Při druhé iteraci tohoto testu měli testeři již tuto nápovědu k dispozici. Všem se povedlo zdárně aplikace spárovat. Test tedy proběhl úspěšně. Bohužel při druhém pokusu už testeři spárovat zařízení uměli z předchozího testování. Proto je možné, že je výsledek zkreslený a nápověda řeší problém jen částečně. Z funkčního hlediska jde však aplikace spárovat bez problému.

5.2.3 Test číslo 3 – Testování použitelnosti základního módu části aplikace pro smartphone

Na testování použitelnosti se soustředily i předchozí dva testy, ale byly zaměřeny především na požadovanou funkcionalitu. Následující testy budou zaměřeny na použitelnost metronomu. V rámci prvního testu uživatel používá pouze hlavní aktivitu aplikace, tedy základní metronom. Tento test byl opakován v nespočtu iterací, během kterých byly odhalovány různé chyby a metronom byl přizpůsobován k tomu, aby byl pro uživatele co nekomfortnější.

Během tohoto testování byla objevena například chyba nepřesnosti v přehrávání zvuků, o které jsem mluvil i v souvislosti s metronomem Pulse (viz. kapitola 2.4).

5.2.4 Test číslo 4 – Testování použitelnosti stage módu

Tento test byl zaměřený na testování módu, kdy bubeník přepíná rytmické vzory v rámci jednoho playlistu během hraní. Podobně jako třetí test probíhal test v několika iteracích. V momentě, kdy bylo implementované i vzdálené ovládání prostřednictvím aplikace na smartwatch se tento test rozšířil a testoval i právě tuto funkcionalitu v ostrém provozu. Kromě na přehrávkách kapely jsme tento mód dvakrát vyzkoušeli i při živém vystoupení v klubu, což považuji za nejdůležitější test, který potvrdil, že bubenický metronom je využitelný právě v těch situacích, pro které byl cíleně implementován.

5.2.5 Test číslo 5 – Testování možné vzdálenosti hodinek od metronomu

Standart bluetooth [4] nám zprostředkovává bezdrátovou komunikaci mezi oběma aplikacemi běžících na dvou strojích. Klasicky se zařízení pro tuto komunikaci dělí do tří tříd:

- První třída – přibližný dosah 100 metrů
- Druhá třída – přibližný dosah 10 metrů
- Třetí třída – přibližný dosah 1 metr

Všechna zařízení, která byla výše zmíněna spadají do první třídy. Na začátku testu jsem tedy předpokládal, že by mělo být komunikaci možno navázat v okruhu 100 metrů od mobilního telefonu. První test neprobíhal na zkoušce ani během živého hraní. Pomocí sportovního měřícího pásma jsem pouze testoval vzdálenostní hranici, po kterou je možné spojení udržet. Bohužel bylo zjištěno, že komunikace je možná zhruba na vzdálenost 70 metrů. Tato vzdálenost je však dostatečná pro téměř všechny hudební akce v klubech a i menší open-air akce. S rostoucí vzdáleností je logické, že roste i prodleva mezi zasláním požadavku a obdržením požadovaných dat zpět. Tato prodleva se však při maximální vzdálenost 70 metrů pohybuje v rámci jedné až dvou sekund, což je prodleva akceptovatelná a neomezující bubeníkovu činnost. Je nutné také brát v úvahu, že aplikace na jiném hardwaru se mohou chovat jinak a vzdálenost může být i delší. V tomto případě beru jako výhodu, že hodinky Kingswear 88 nejsou osazeny *Wear OS* ale klasickým Androidem verze 5.1. Aplikace určená pro smartwatch je proto kompatibilní s jakýmkoliv mobilním telefonem s API levelem 22 a vyšším. Aplikace na obdélníkovém a větším displayi sice nevypadá tak dobře, ale je plně funkční a použitelná. Test mohl být tedy proveden i na dvou mobilních telefonech. V tomto případě se vzdálenost opravdu blížila k výrobci uváděným 100 metrům.

Jak už jsem výše zmínil, metronom byl použit i na brněnském Majálesu. Zde byla aplikace ovládána prostřednictvím hodinek na vzdálenost 30 metrů.

5.2.6 Experimenty s vibračními impulsy

O vibracích a nezdařené synchronizaci jste se již dozvěděli. S tímto nezdarem souvisí několik experimentů a testů. Nejprve bylo důležité implementovat základní vibrační metronom pro hodinky. S tím souviselo pomocí experimentů zjistit nejvhodnější délku vibrace tak, aby byla i při rychlejší tempech rozeznatelná a nesplývala s ostatními impulsy. Z hudební teorie [17] se můžete dozvědět, že nejrychlejším tempem je tzv. *prestissimo*[1], které spadá do skupiny

[1] překlad z italského jazyka zní nejrychleji

temp *presto*[2]. *Prestissimo* by pak mělo odpovídat rychlosti 208 BPM. Z vlastních zkušeností pak vím, že bubeníci někdy používají i tempo 300 BPM, ale to pouze v zřídkavých případech (většinou se tempo půlí, tzn. pokud bubeník chce hrát 300 úderů za minutu, postačí mu zadat tempo 150 BPM). Musel jsem tedy zjistit správnou délku impulsu, která by nesplývala a zároveň je rozeznatelná i v těchto rychlejších rytmech. Na základě testování s Janem Poláškem jsme jako ideální délku shledali 100 milisekund.

Jak jsem se již zmínil v kapitole zabývající se konkurenčními metronomy (kapitola 2.4) firma vyvíjející metronom Soundbrenner nabízí ke koupi speciální hardware, který se pomocí bluetooth spáruje s mobilem a vibračními impulsy udává rytmus. Tento hardware nefunguje jako vzdálené ovládání metronomu. Za úkol má pouze udávat rytmus. Veškeré playlisty se pak řídí přímo z mobilu. Uživatel má možnost být řízen zvukovými i vibračními impulsy zároveň. Původně jsem tuto možnost zamýšlel zakomponovat i do mého produktu. Bohužel mi po několika implementacích a následných testech bylo Janem Poláškem sděleno, že vibrace a zvuk nejsou zcela synchronní. Proto bylo od tohoto řešení upuštěno.

[2]překlad z italského jazyka zní rychle

Kapitola 6

Implementace

Pro implementaci jsem se rozhodl využít vývojové prostředí *Android studio* a programovací jazyk Java. Návrh grafické části byla provedena pomocí layoutů ve formátu XML. Pro ukládání perzistentních dat jsem pak využil *SQLite*. Jakožto naprostý začátečník ve vývoji pro Android i jakožto začátečník s programovacím jazykem Java jsem nejprve musel načerpat nějaké zkušenosti. K tomu mi posloužila kniha Android Programming The Big Nerd Ranch Guide. Po lehkém seznámení se s problematikou ohledně vývoje pro tuto platformu jsem přistoupil k samotné implementaci. S jazykem Java jsem se tedy seznamoval až při implementaci mého metronomu.

6.1 Implementace – aplikace pro smartphone

Ve své podstatě se dá říci, že tato aplikace funguje jako server. Aplikace pro hodinky se pak dá přirovnat ke klientovi, který zasílá serveru požadavky. Ten na ně odpovídá. Veškerá perzistentní data jsou pak tedy uložena právě na mobilním telefonu. Aplikace s nimi tedy musí umět operovat, zpracovávat je a odpovídat na požadavky zaslané klientskou aplikací z hodinek. Z tohoto důvodu je tato část podstatně robustnější a složitější než část pro smartwatch.

6.1.1 Základní funkcionalita metronomu

Jak jsem se již zmínil, tato část je částečně převzata z implementace projektu do ITU (převzata byla část implementované pouze mnou). Z počátku jsem zamýšlel celý metronom implementovat pomocí některého z exitujících návrhových vzorů, které se při implementaci mobilních aplikací běžně využívají. Nejvíce mne zaujal návrhový vzor MVVM[1] [20]. Tento model, jak již název napovídá, se skládá ze tří vrstev:

- **Model** – reprezentuje data.
- **View** – tato vrstva má za úkol reprezentovat uživatelské rozhraní. V mém případě, kdy využívám programovací jazyk Java a *Android studio*, je UI reprezentováno prostřednictvím XML layoutů.
- **ViewModel** – nejdůležitější vrstva návrhového vzoru MVVM. Má za úkol spojovat dvě předešlé vrstvy a držet stav aplikace. Přijímá data z vrstvy View, zpracovává je

[1] model-view-viewmodel

a ukládá do vrstvy Model. K této výměně dat a stavů dochází za pomoci tzv. obousměrného data bindingu. Jak z názvu vyplývá, výměna dat probíhá v obou směrech.

Tento návrhový vzor byl použit pro implementaci základní funkcionality metronomu. Problémem je při implementaci výše zmíněný obousměrný data binding [18]. Implementace mi přišla poměrně náročná, a proto jsem pro implementaci dalších částí již tento návrhový vzor nepoužíval.

V mém případě View představuje layout `activity_main.xml`, který se váže k aktivitě `MainActivity`, Model je pak třída `RhythmData` a ViewModel třída RhythmViewModel. Vždy, když je aplikace spuštěna vzniká instance třídy `RhythmViewModel`. Tato instance pak uchovává stav aplikace při používání základní funkcionality. V této třídě jsou pak implementovány metody, které jsou volány při interakci uživatele s UI. V následující části kódu vidíte syntaxi a způsob jakým lze vytvořit v rámci XML layoutu instanci třídy `RhythmViewModel`. Třída `RhythmViewModel` musí dědit od třídy ViewModel. V celém XML pak můžeme využívat tuto instanci pro obousměrný data binding. Na řádku číslo 14 je příklad volání metody v rámci event listeneru `onTextChanged`, která je implementována v této třídě. Řádek číslo 15 ukazuje zápis oboustranného data bindingu pomocí symbolů `@` a `=`.

```
1    <data>
2        <import type="com.example.myapplication.viewModel.RhythmViewModel"
             />
3        <variable
4            name="rhythmViewModel"
5            type="com.example.myapplication.viewModel.RhythmViewModel"/>
6    </data>
7
8    <EditText
9        android:id="@+id/editTextBpm"
10       android:layout_width="wrap_content"
11       android:layout_height="wrap_content"
12       android:inputType="number"
13       android:maxLength="3"
14       android:onTextChanged="@{(cs, x, y, z)->
             rhythmViewModel.onTextChanged(cs)}"
15       android:text="@={rhythmViewModel.data.bpm}"
16       android:backgroundTint="@color/colorPrimary"
17       android:textColor="@color/text_white"
18       android:textSize="40sp" />
```

Při implementaci datového modelu (třída `RhythmData`), tedy toho jak bude vypadat třída, jejíž jedna instance bude představovat jeden rytmický vzor, jsem musel myslet na to, že jednotlivé aktivity si budou muset dokázat takovou instanci předat. Předávání proměnných do jiné aktivity probíhá pomocí třídy `Intent`. To může vypadat následovně:

```
1    Intent intent = new Intent(this , MainActivity.class);
2    intent.putExtra("songInfo", 5);
3    context.startActivity(intent);
```

V uvedeném případě je předán do druhé aktivity integer o hodnotě 5 (viz. řádek 2). Tímto způsobem však mohou být předány pouze proměnné základních datových typů. Z toho

důvodu musí třída `RhythmData` dědit od třídy `Parcelable` [11]. Tímto dosáhneme možnosti předávat pomocí třídy `Intent` i složitější objekty.

Material design vývojáři nabízí mnoho elementů, které ve svém GUI může vývojář využít. Jedním z nich je i seekbar. Seekbar, který je implicitně vývojářům k dispozici je však ve tvaru úsečky. Tento seekbar se mi ovšem osobně příliš nelíbil. K mému překvapení vývojářům není k dispozici kruhový seekbar. Z tohoto důvodu jsem se rozhodl využít seekbar, který byl volně k dispozici na GitHubu. Licence tohoto seekbaru je Apache verze 2.0. Ve zdrojovém kódu jsou pak náležité soubory označeny hlavičkou.

6.1.2 Databáze a ukládání perzistentních dat

Jednotlivé instance třídy `RhythmModel` je pak třeba uchovávat jako perzistentní data, pokud si uživatel přeje rytmický vzor uložit. Android podporuje několik možností jak data ukládat. [6]

- **Interní datové úložiště** – Pomocí této možnosti se ukládají privátní data aplikace do adresáře vyhrazeného k tomuto účelu. Každá aplikace má pak svůj vlastní adresář.
- **Externí datové úložiště** – Příkladem může být například ukládání fotografií do galerie, ze které jsou pak aplikací například čteny. K tomuto přístupu musí mít aplikace náležitá oprávnění.
- **Shared preferences** – Hodí se zejména pro ukládání menšího množství dat, které je ve formátu klíč-hodnota. Tato data jsou pak ukládána v interním uložišti aplikace ve formátu XML. Tento způsob jsem využil pro uložení informací o párovaném zařízení prostřednictvím bluetooth.
- **Databáze** – Způsob ukládání perzistentních dat do databází jsem zvolil pro ukládání playlistů a rytmických vzorů.

Jako databázový systém bylo použito *SQLite*, které je doporučováno i na oficiálních stránkách Android Developer. Jedná se o klasický relační databázový systém. Velkou výhodou tohoto databázového systému je, že databáze je v jednom souboru, který je nezávislý na platformě. Toho se dá využít zejména při migraci dat. *SQLite* je plně podporováno při vývoji aplikací pro Android v jazyku Java. Nalezneme mnoho tříd, které nám s tímto databázovým systémem značně ulehčí práci. Já jsem využil třídu `SQLiteOpenHelper`. Od této třídy dědí mnou implementované třídy `SongDatabase`, která se stará o práci s databází rytmických vzorů a `PlaylistDatabase`, která má za úkol starat se o jednotlivé databáze představující playlisty. Díky těmto třídám se pohodlně zpracovávají a ukládají persistentní data.

Jak jsem se již zmínil v kapitole 4.2.1, původně jsem zamýšlel, že by se každý rytmický vzor pevně vázal právě k jednomu playlistu. Tento koncept se ale při testování ukázal jako nepraktický. Finální podoba databází se z tohoto důvodu musela přepracovat. Před tím než si ukážeme strukturu databází, je třeba si ukázat jak vypadají objekty, které do těchto databází budou vkládány. Níže vidíte třídu `RhythmData` a její instanční proměnné.

```
1 public class RhythmData extends BaseObservable implements Parcelable {
2
3     private int songDatabaseId;  //primarni klic v~db. rytmickych vzoru
4     private int bpm;             //hodnota BPM
5     private long period;         //perioda uderu
```

```
6       private String songName = ""; //nazev rytmickeho vzoru
7       private String accentArr = ""; //string prezentujici durazy na doby
8       private int countOfBeats = 0; //pocet dob v~taktu
9       private String soundName = ""; //nazev zvuku odbijeni
10      private int ordNumb = -1;    //poradove cislo v~ramci playlistu
11      ...
12  }
```

Proměnná `ordNumb` má smysl v rámci přehrávání playlistu. Ukládá se tedy do jednotlivých databází, které představují daný playlist. Proměnná `period` pak není ukládána vůbec, protože se perioda opakování dá lehce dopočítat z hodnoty BPM. Vztah je následující a výsledek je v milisekundách:

$$T = \frac{60000}{BPM}$$

Struktura databází je znázorněna v následujících tabulkách:

ID (p.k.)	name	bpm	accentString	bCount	soundName	usage
1	Pictures	170	1111	4	Stick	3
2	T.N.T	150	1212	4	Wood	1
3	Song 2	145	1023	4	Bongo	0
...	...	...	...	...	...	...

Obrázek 6.1: S databází následující struktury pracuje třída `SongDatabase`. Do této databáze jsou ukládány jednotlivé rytmické vzory. Primárním klíčem jsou automaticky generovaná ID (viz. první sloupec). V sloupci `accentString` jsou uloženy jednotlivé důrazy na danou dobu pro konkrétní rytmický vzor. Tyto důrazy reprezentuje hodnota datového typu `String`. Každý jeden znak v tomto řetězci pak představuje jednu konkrétní dobu (0 – doba je vynechána, 1 – důraz prvního stupně, 2 – důraz druhého stupně, 3 – důraz třetího stupně). Poslední sloupec s názvem `usage` nám pak říká, v kolika playlistech je daný rytmický vzor využit. Tato informace je důležitá především při mazání jednotlivých záznamů (rytmický vzor nestačí odstranit pouze z databáze rytmických vzorů, ale i ze všech databází, které reprezentují jednotlivé playlisty a rytmický vzor v nich byl obsažen).

ID (p.k.)	ordNumb
1	3
2	1
3	2
...	...

Obrázek 6.2: Pro každý playlist je vytvořena samostatná databáze uvedeného formátu. Primární klíč je ID náležitého rytmického vzoru, který má být v playlistu obsažen. Pomocí tohoto ID se pak v databázi rytmických vzorů vyhledají odpovídající a potřebné údaje o vzoru. Druhá hodnota `ordNumb` pak určuje pořadí jednotlivých vzorů. Tímto způsobem je zajištěna možnost jednotlivé rytmické vzory použít v několika playlistech. Pokud je pak rytmický vzor odstraněn z playlistu, stále jej lze využít, jelikož jsou informace o něm uloženy v databázi z obrázku 6.1

.

Práce se třídou SQLiteOpenHelper

Tato třída [14] má za úkol co nejvíce zjednodušit práci s databázemi *SQLite* [13]. Pomocí Override metody `onCreate()` můžeme lehce vytvořit novou databázi. Metoda `onUpgrade()` nám pak zpřístupní již existující databáze. Implementace může vypadat například následovně:

```
1  public class RhythmData extends BaseObservable implements Parcelable {
2
3      @Override
4      public void onCreate({\it SQLite}Database db) {
5          String createTable = "CREATE TABLE " + TABLENAME + " (ID integer
               PRIMARY KEY AUTOINCREMENT, "+ COL0 +" text, " +
6                  COL1 + " integer, " + COL2 +" text,"+ COL3 + " integer, " +
7                  COL4 + " text, " + COL5 + " integer)";
8          db.execSQL(createTable);
9      }
10
11     @Override
12     public void onUpgrade({\it SQLite}Database db, int oldVersion, int
           newVersion) {
13         db.execSQL("DROP IF TABLE EXISTS " + super.getDatabaseName());
14         this.onCreate(db);
15     }
16 }
```

V metodě `onCreate()` vidíme řetězec s názvem `createTable`. V tomto řetězci je obsažen příkaz pro vytvoření tabulky. Na řádku číslo 8 je pak tento příkaz spuštěn pomocí metody `execSQL`. Podobným způsobem pak můžeme zacházet s databází. Jako příklad je uvedena metoda `updateDataById()`. Ta má za úkol aktualizovat data náležitého rytmického vzoru podle ID, tedy primárního klíče.

```
1      public void updateDataById( int ID ){
2          {\it SQLite}Database db = this.getWritableDatabase();
3          String sqlQuery = "UPDATE "+ this.getDatabaseName() + " SET " +
4                  COL0 + " = \'" + this.toSave.getSongName() + "\' , " +
5                  COL1 + " = " + this.toSave.getBpm() + " , " +
6                  COL2 + " = \'" + this.toSave.getAccentArr() + "\' , " +
7                  COL3 + " = " + this.toSave.getCountOfBeats() + " , " +
8                  COL4 + " = \'" + this.toSave.getSoundName() +
9                  "\' WHERE ID = " + ID;
10
11         Cursor cursor = db.rawQuery(sqlQuerry, null);
12         cursor.moveToFirst();
13         cursor.close();
14     }
```

Nejprve je vytvořen dotaz a uložen do řetězce `sqlQuery`. Tento dotaz se pak provede pomocí metody `rawQuery()`. Tato metoda pak vrací kurzor, který se dá využít například při dotazu `SELECT`, pokud je vybráno více položek databáze. Podobným způsobem jsou

implementovány obě třídy pracující s databázemi playlistů a databází rytmických vzorů (`PlaylistDatabase` a `SongDatabase`).

6.1.3 Knihovna rytmických vzorů a playlistů

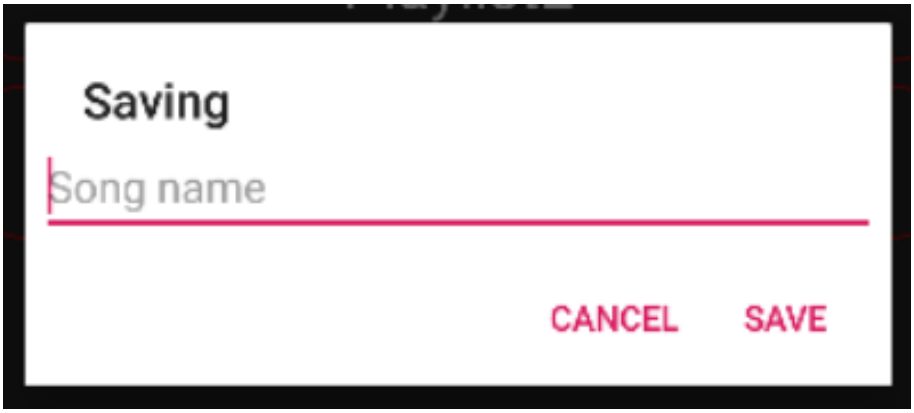
Jedná se GUI, ve kterém uživatel spravuje uložené playlisty a rytmické vzory. Toto GUI je rozděleno na dvě části pomocí záložek. Každá záložka je pak tvořena tzv. fragmentem, který určuje její grafickou podobu. Mezi těmito záložkami se dá přepínat pomocí gesta *swipe*. Adaptér těchto fragmentů, kterým jsou záložky implementovány, je popsán v následující podkapitole. Fragmenty pak obsahují `ListView`, který uživateli zobrazuje jednotlivé rytmické vzory či playlisty.

U implicitně obsaženého `ListView` je poměrně zajímavé, že chybí tak základní funkcionalita jako je například řazení listu pomocí gesta *drag and drop*. Toto přeskupování pro mne ovšem bylo poměrně podstatné. Implementace by byla velkou ztrátou času a tak jsem se rozhodl v časové tísni využít podtřídu třídy `ListView`, která byla dostupná volně k použití pod licencí Apache verze 2.0. Převzaté třídy jsou ve zdrojovém kódu označeny hlavičkou licence a je lehce dohledatelný zdroj.

Při správě knihoven je často využit tzv. alert dialog [2]. Tento dialog se dá požít například při mazání záznamů, kdy uživatel svoji akci musí potvrdit. Takový dialog se dá vykreslit následujícím způsobem:

```
1    new AlertDialog.Builder(MainActivity.this)
2        .setTitle("Saving song")
3        .setNegativeButton("Cancel", null)
4        .setPositiveButton("Save", new DialogInterface.OnClickListener() {
5            @Override
6            public void onClick(DialogInterface dialog, int which) {
7                        //akce ktere jsou provedeny pri stisku tlacitka
                             save
8                    }
9                })
10               .setView(linearLayout)
11               .create().show();
```

Výsledný dialog vidíme na obrázku 6.3.



Obrázek 6.3: Výsledný dialog

6.2 Implementace – aplikace pro smartwatch

Při implementaci aplikace pro smartwatch byly využity stejné postupy jako při implementaci základní funkcionality metronomu pro smartphone. Jediným rozdílem je, že není třeba ukládat žádná data. Pro fungování vibračního metronomu potřebujeme znát pouze BPM. Důrazy pomocí vibrací nejsou implementovány. Vzdálené ovládání je popsáno v následující podkapitole.

Zajímavou částí implementace je využití třídy `FragmentPagerAdapter`. Pomocí této třídy je implementovaný přechod mezi jednotlivými fragmenty UI pomocí gesta *swipe*. Podobný adaptér byl využit i při implementaci záložek v rámci knihovny pro správu rytmických vzorů a playlistů aplikace pro smartphone. Takový adaptér může vypadat následovně:

```
1  public class ViewPagerAdapter extends FragmentPagerAdapter {
2
3      public ViewPagerAdapter(FragmentManager fm){
4          super (fm);
5      }
6
7      @Override
8      public Fragment getItem(int i) {
9          Bundle bundle;
10         if (i == 0) return new RemoteModeFragment();
11         else if (i == 1) return new MetronomeFragment();
12         else return new SettingsFragment();
13     }
14
15     @Override
16     public int getCount() {
17         return 3;
18     }
19
20     @Nullable
21     @Override
22     public CharSequence getPageTitle(int position) {
23         if (position == 0) return "Metronome";
24         else return "Settings";
25     }
26 }
```

Díky této třídě je tedy umožněno snadné a rychlé přepínání mezi fragmentem, který zprostředkovává vibrační metronom, fragmentem nastavení spojení a fragmentem, který funguje jako dálkové ovládání aplikace pro smartphone.

6.3 Implementace komunikace prostřednictvím bluetooth

Dle mého názoru se jedná o nejzajímavější část implementace mých dvou aplikací. Díky této části spolu mohou aplikace navzájem komunikovat. Informace ohledně bluetooth komunikace byly čerpány z článku na stránkách Android Developers [5]. Pro navázání komunikace jsou důležité dva identifiklátory. Prvním je MAC adresa zařízení. Každá aplikace pak může fungovat na svém UUID. Velice zjednodušeně by se UUID dalo přirovnat k portům u sítí. UUID je pevně dáno implementací.

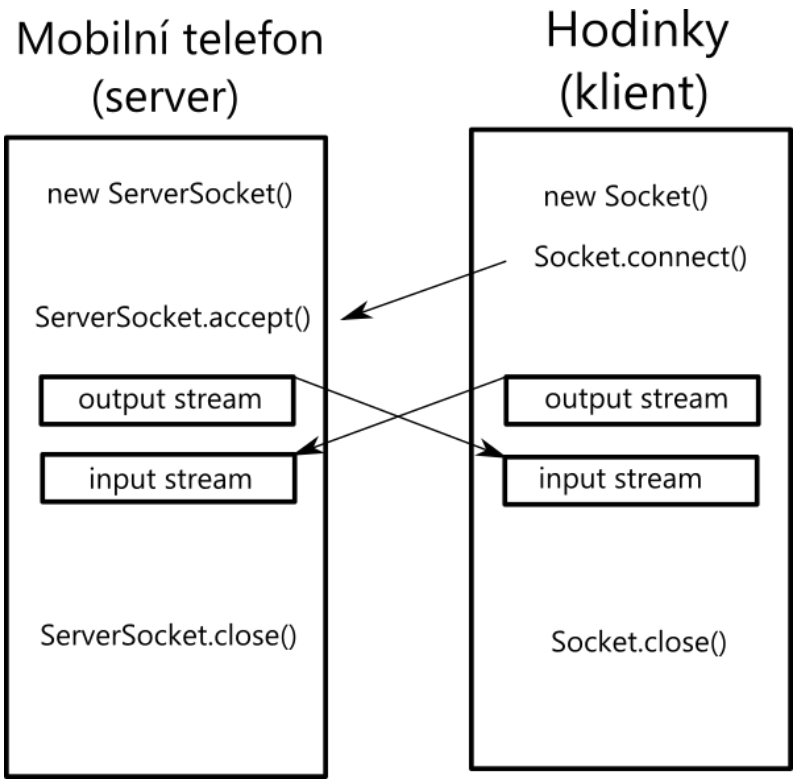
Jako UUID jsem zvlolil `00001101-0000-1000-8000-00805F9B34FB`, které je doporučováno i na oficiálních stránkách Android Developer. Pomocí MAC adresy pak můžeme navázat spojení. Mac adresa párovaného zařízení se však mění, a proto je nutné implementovat algoritmus, kterým se zaručí, že při vytváření spojení použijeme správnou MAC adresu. Pomocí následující funkce vytvořím `Set` všech zařízení, která kdy byla se zařízením, na kterém aplikace běží, spárována:

```
1   public Set<BluetoothDevice> getAllDevices(){
2
3
4       bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
5       if(bluetoothAdapter == null)
6       {
7           Toast toast = Toast.makeText(context, "No bluetooth adapter
                available", Toast.LENGTH_SHORT);
8           toast.show();
9       }
10
11      if(!bluetoothAdapter.isEnabled())
12      {
13          Intent enableBluetooth = new
                Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
14          activity.startActivityForResult(enableBluetooth, 0);
15      }
16
17      if (bluetoothAdapter.getBondedDevices().isEmpty()){
18          Toast toast = Toast.makeText(context, "You have no paired
                bluetooth devices", Toast.LENGTH_SHORT);
19          toast.show();
20      }
21
22      return bluetoothAdapter.getBondedDevices();
23  }
```

Tento `Set` je pak zobrazen prostřednictvím `ListView` uživateli. Uživatel zvolí správné zařízení, se kterým má aplikace komunikovat. Údaje o tomto zařízení jsou pak uloženy do interního datového úložiště ve formátu XML.

V této ukázce kódu si můžeme povšimnout i implementace tzv. toast zpráv, které jsou pro Material Design typické. Mají za účel informovat uživatele o chybách, problémech, atd.

Mnou využité knihovny nabízí poměrně jednoduchou implementaci, která je velice podobná socketovému programování síťové aplikace klient-server v jazyce C/C++. Schéma komunikace je znázorněno na obrázku 6.4.



Obrázek 6.4: Obě dvě aplikace vytvoří sockety. Serverový socket začne naslouchat a čeká na připojení socketu ze strany klienta. V momentě kdy se oba sockety připojí, vytvoří se output a input stream pro oba sockety. Pomocí těchto streamu pak aplikace komunikují. Tuto komunikaci pak většinou obsluhuje jiné vlákno [31]. Vlákna se využívají i při navázání komunikace, neboť metody `connect()` a `accept()` jsou metodami blokujícími.

Následující metoda je ukázkou vytvoření serverového socketu a navázání spojení. V metodě je také vidět jakým způsobem lze zpřístupnit output a input stream socketu.

```
1     public void openConnection() throws IOException
2     {
3         UUID uuid =
              UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
              //doporucene UUID
4         BluetoothServerSocket sock =
              bluetoothAdapter.listenUsingRfcommWithServiceRecord("metronome",
              uuid);
5         bluetoothAdapter.cancelDiscovery();
6         socket = sock.accept();
7
8         outputStream = socket.getOutputStream();
9         inputStream = socket.getInputStream();
10    }
```

Tato metoda slouží k vytvoření socketu a komunikace na straně klienta.

```
1     public void openConnection() throws IOException
2     {
3         UUID uuid = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
4         bluetoothAdapter.cancelDiscovery();
5         socket = device.createInsecureRfcommSocketToServiceRecord(uuid);
6         socket.connect();
7         outputStream = socket.getOutputStream();
8         inputStream = socket.getInputStream();
9     }
```

V momentě, kdy je navázáno spojení, zavolá se metoda, která spustí nové vlákno. Toto vlákno pak obsluhuje komunikaci. V rámci tohoto vlákna je v nekonečném cyklu parsován input stream.

```
1     public void beginListening()
2     {
3         runFlag = false;
4         readBufferPosition = 0;
5         messageBuffer = new byte[1024];
6         workerThread.start();
7     }
```

V mé aplikaci je potřeba zasílat pouze pokyny k přepnutí na další rytmický vzor, popřípadě údaje o rytmickém vzoru, na který bylo přepnuto. Proto lze veškerá komunikace zařídit jen pomocí přeposílání jednoduchých textových řetězců. Část pro smartphone může zasílat hodinkám zprávy následujícího formátu:

- `startInfo$[true|false]` – tato zpráva je zaslána při zahájení spojení. Pokud na mobilním telefonu v tento moment bylo spuštěno přehrávání zvuku (`true` nebo `false`), synchronizuje se hodinková aplikace tak, aby toggle buton *play/pause* bylo ve správném stavu.
- `songInfo$[jméno]$[BPM]` – tuto zprávu aplikace zasílá v momentě, kdy je zaznamenána změna rytmického vzoru. Zašle aplikaci na hodinkách potřebné informace (jméno rytmického vzoru a BPM).
- `play` – změní stav toggle buttonu na klientské aplikaci tak, aby bylo zobrazeno, že je rytmus aktuálně přehráván.
- `stop` – obdoba předchozí zprávy s opačným efektem. Tyto dvě zprávy zasílá i klientská aplikace.
- `connectionClosed` – tato zpráva říká, že bude spojení ukončeno.

Podobné zprávy pak zasílá i aplikace pro smartwatch:

- `next` – zašle mobilní aplikaci zprávu, kterou vyžádá přepnutí na následující rytmický vzor. Zpět obdrží zprávu `songInfo`.
- `back` – zašle mobilní aplikaci zprávu, kterou vyžádá přepnutí na předchozí rytmický vzor. Zpět obdrží zprávu `songInfo`.

Jak lze vidět, některé zprávy jsou rozděleny pomocí separátoru `$`. První část každé zprávy je její název. Další části jsou pak informace, které tato zpráva nese. Na konci každé zprávy je uveden přesný systémový čas, kdy byla zpráva odeslána. Toto opatření s časovou známkou jsem zavedl z důvodu, aby nemohla nastat situace, kdy by byla některá zpráva zpracována vícekrát. To by mohlo například způsobit špatné přesouvání mezi jednotlivými rytmickými vzory v playlistu.

Při implementace práce s vlákny jsem narazil na menší problém, který jsem po kratším studiu této problematiky naštěstí vyřešil. Při vývoji aplikací pro Android totiž k objektům GUI můžou přistupovat pouze vlákna, která jsou k tomu určena. Tento problém řeší následující konstrukce:

```
1    new Thread(new Runnable() {
2            //v novem vlaknu nelze pracovat s objekty GUI
3            view.post(new Runnable() {
4                @Override
5                public void run() {
6                    //diky teto konstrukci je prace s objekty gui umoznena
7                    connect.setVisibility(View.GONE);
8                    next.setVisibility(View.VISIBLE);
9                    back.setVisibility(View.VISIBLE);
10                   play.setVisibility(View.VISIBLE);
11                   bpmTextView.setVisibility(View.VISIBLE);
```

```
12                      nameTextView.setVisibility(View.VISIBLE);
13                  }
14              });
15      }).start();
```

Při hledání informací jak správně pracovat s vlákny v programovacím jazyce Java jsem využil následující literaturu: Java Threads [31].

Kapitola 7

Další možný postup při vývoji bubenického metronomu

Aplikace určená pro smertphone je nyní plně funkční metronom, který je dle mého názoru schopný obstát mezi konkurencí na *Google Play* a to i bez možnosti vzdáleného ovládání. Jako další krok se tedy nabízí zpřístupnit aplikace na *Google Play*. Za tímto účelem si vývojář musí zřídit účet na *Google Play Console* [7]. Zde je registrace zpoplatněna. *Google Play Console* však vývojáři, který aplikaci zpřístupní nabízí mnoho výhod do budoucího vývoje, ladění chyb a přidávání nových funkcionalit či úpravy UI. Každý uživatel totiž může aplikaci na *Google Play* hodnotit a psát recenze. Takové recenze by byly určitě zajímavou zpětnou vazbou pro další vývoj aplikace.

V první řadě bych ale chtěl zpřístupnit pouze Alfa verzi metronomu. Alfa verze pak může být veřejná či uzavřená. Jakožto aktivní hudebník se pohybuji v okruhu mnoha bubeníků a není pro mne tedy problém sehnat dostatečné množství testerů i pro uzavřenou verzi testování. Po odladění této Alfa verze by pak byla zveřejněna veřejná Beta verze. Beta verze je stále verzí aplikací, která není finálním produktem. Může obsahovat chyby, na *Google Play* je dostupná pro všechny uživatele, ale nemůže být zpoplatněna. V momentě kdy bych si byl opravdu jistý, že je metronom připravený a prodejný, přistoupil bych k zpoplatnění metronomu. Při porovnávání ostatních produktů se mi osobně příliš nelíbí možnost, kdy vývojář nabízí plnou verzi za poplatek, ale uživatel si nemůže plnou verzi vyzkoušet, jako je tomu například u metronomu Metronome Beats [29]. Proto bych volil možnost zkušební verze, která by byla časově omezená. Další variantou je pak zkušební verzi omezit například maximálním počtem uložených playlistů, či rytmických vzorů.

Jak jsem již zmínil, hodinky osazené klasickým Androidem nejsou zcela běžné. Z tohoto důvodu by bylo vhodné do budoucna implementovat aplikaci pro smartwatch i pro *Wear OS* či jiné platformy jako třeba *Tizen*. Aplikace určená pro hodinky může být na druhou stranu bez problému nainstalována i na smartphone. V tomto případě ale aplikace působí poněkud nevzhledně. Proto přichází v úvahu implementovat i aplikaci pro vzdálené ovládání určenou přímo pro mobilní telefon.

Kapitola 8

Závěr

Výsledkem bakalářské práce je bubenický metronom, který se snaží řídit standartem Material designu. UI je jednoduché a přehledné. Metronom je vhodný především pro hraní na zkouškách a koncertech. Nejdůležitějším prvkem i úspěchem je pak možnost metronom vzdáleně ovládat, a zbavit se tak zbytečné kabeláže a složitého zapojování metronomu na hudebních akcích v klubech a menších open-air akcích. Aplikace splňuje všechny požadavky, které na ni původně byly kladeny. Jedinou výjimkou je nevydařená synchronizace vibračních impulsů s impulsy zvukovými.

Bakalářská práce se však neskládala pouze z implementační části. Mezi další důležité etapy, kterými jsem musel projít, je průzkum konkurenčních aplikací na *Google Play* postupný iterační vývoj GUI, testování v běžném provozu. Nedílnou součástí byly i konzultace s panem profesorem Adamem Heroutem, které byly taktéž důležitou zpětnou vazbou.

Během práce jsem se seznámil s jazykem Java, se kterým jsem se v rámci bakalářské práce setkal poprvé a naučil jsem se základy vývoje aplikací pro *OS Android*. Také jsem nastudoval způsoby pro uchovávání perzistentních dat s využitím *SQLite*.

Do budoucna bych chtěl vytvořit třetí aplikaci, která by byla určena pro mobilní telefon. Ta by nahradila aplikaci pro smartwatch. Hlavním důvodem je, že většina cílových uživatelů nevlastní hodinky s Androidem. Vzdálené ovládání by pak mohlo být prováděno pomocí druhého smartphonu. Po tomto kroku bych dále chtěl mnou vytvořený metronom umístit na *Google Play*.

Literatura

[1] About the platform. [Online; navštíveno 09.4.2019].
URL https://developer.android.com/about

[2] AlertDialog. [Online; navštíveno 10.4.2019].
URL https://developer.android.com/reference/android/app/AlertDialog

[3] Android Studio. [Online; navštíveno 10.4.2019].
URL https://developer.android.com/studio/

[4] Bluetooth. [Online; navštíveno 10.4.2019].
URL https://cs.wikipedia.org/wiki/Bluetooth

[5] Bluetooth overview. [Online; navštíveno 10.4.2019].
URL https://developer.android.com/guide/topics/connectivity/bluetooth

[6] Data and file storage overview. [Online; navštíveno 10.4.2019].
URL https://developer.android.com/guide/topics/data/data-storage

[7] Google Play Console. [Online; navštíveno 10.4.2019].
URL https://developer.android.com/distribute/console

[8] Material Design. [Online; navštíveno 10.4.2019].
URL https://material.io/

[9] Material Design – Tabs. [Online; navštíveno 10.4.2019].
URL https://material.io/design/components/tabs.html#

[10] Musical Glossary. [Online; navštíveno 15.4.2019].
URL https://www.8notes.com/glossary/

[11] Parcelable. [Online; navštíveno 10.4.2019].
URL https://developer.android.com/reference/android/os/Parcelable

[12] Sample Swap. [Online; navštíveno 09.11.2018].
URL https://sampleswap.org

[13] SQLite. [Online; navštíveno 10.4.2019].
URL https://www.sqlite.org/index.html

[14] SQLiteOpenHelper. [Online; navštíveno 10.4.2019].
URL https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper

[15] statcounter. [Online; navštíveno 10.4.2019].
URL http://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide

[16] Xamarin. [Online; navštíveno 10.4.2019].
URL https://en.wikipedia.org/wiki/Xamarinl

[17] *Všeobecná encyklopedie ve čtyřech svazcích*. Nakladatelský dům OP, 1998, ISBN 80-85841-37-1.

[18] Adefioye, T.: Android Jetpack: Empower your UI with Android Data Binding. [Online; navštíveno 10.4.2019].
URL https://medium.com/@temidjoy/android-jetpack-empower-your-ui-with-android-data-binding-94a657cb6be1

[19] Audacity: Audacity. [Online; navštíveno 09.11.2018].
URL https://www.audacityteam.org/

[20] Dajbych, V.: MVVM. [Online; navštíveno 10.4.2019].
URL https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModelE

[21] Fojtík, J.: Nástroje pro tvorbu vlastních mobilních aplikací. [Online; navštíveno 10.4.2019].
URL https://spomocnik.rvp.cz/clanek/20667/NASTROJE-PRO-TVORBU-VLASTNICH-MOBILNICH-APLIKACI.html

[22] GmbH, E. X. T.: Pro Metronome. [Online; navštíveno 27.04.2019].
URL https://play.google.com/store/apps/details?id=com.eumlab.android.prometronome

[23] hutilicious: Komponenty material designu. [Online; navštíveno 20.04.2019].
URL https://forum.xda-developers.com/android/apps-games/template-material-design-template-t3026275

[24] Inc., G.: Komponenty material designu. [Online; navštíveno 19.04.2019].
URL https://google.com/design/spec/

[25] Joutsenvirta, A.; Perkiömäki, J.: *Music theory – time*. [Online; navštíveno 19.04.2019].
URL http://www2.siba.fi/muste1/index.php?id=102&la=en

[26] Keuwlsoft: Metronome. [Online; navštíveno 22.04.2019].
URL https://play.google.com/store/apps/details?id=com.keuwl.metronome

[27] Maesicano, B. P. . C. S. . K.: *Android Programming The Big Nerd Ranch Guide*. Big Nerd Ranch, 2017, ISBN 978-0134706054.

[28] Soundbrenner: The Metronome by Soundbrenner. [Online; navštíveno 03.04.2019].
URL https://play.google.com/store/apps/details?id=com.soundbrenner.pulse

[29] Stonekick: Metronom Beats. [Online; navštíveno 18.03.2019].
URL
https://play.google.com/store/apps/details?id=com.andymstone.metronome

[30] Technologies, C.: Pulse – Metronome. [Online; navštíveno 19.03.2019].
URL https://play.google.com/store/apps/details?id=com.trycrescendo.pulse

[31] Wong, S. O. . H.: *Java Threads*. O'Reilly, 2004, ISBN 0-596-00782-5.

Příloha A

Obsah přiloženého paměťového média

Na přiloženém CD naleznete následující:

- `poster/poster.pdf` – plakát prezentující moji aplikaci
- `video/video.mp4` – krátké video prezentující moji aplikaci
- `src/` – zdrojové soubory aplikace
- `tex_src/` – zdrojový kód TEXu pro vytvoření bakalářské práce ve formátu PDF
- `BP_xondra51.pdf` – textová část bakalářské práce v PDF
- `README.txt` – informace k práci s aplikací a popis jak aplikaci zprovoznit za využití vývojového prostředí Android Sudio