VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

METRONOM PRO MOBILNÍ ZAŘÍZENÍ ANDROID

METRONOME FOR THE ANDROID MOBILE DEVICE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **FRANTIŠEK POMKLA**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Pomkla František**
Program: Informační technologie
Název: **Metronom pro mobilní zařízení Android**
Metronome for the Android Mobile Device
Kategorie: Uživatelská rozhraní

Zadání:

1. Seznamte se s principy tvorby mobilních aplikací se systémem Android se zaměřením na chytré hodinky. Prostudujte existující vývojová prostředí.
2. Seznamte se s existujícími řešeními metronomů pro mobilní zařízení, diskutujte jejich nedostatky.
3. Navrhněte metronom pro mobilní zařízení, které bude komunikovat s chytrými hodinkami a pomocí vibrací bude udávat rytmus. Součástí řešení bude také vyhodnocení přesnosti úderů na základě zvuku úderů.
4. Implementujte navrženou mobilní aplikaci a ověřte její funkčnost.
5. Zhodnoťte dosažené výsledky a navrhněte další možná vylepšení.

Literatura:

- CASTLEDINE, E.: Vytváříme mobilní web a aplikace pro chytré telefony a tablety. 1. vyd. Brno: Computer Press, 2013. 288 s. ISBN 978-80-251-3763-5.
- DANIEL, S.F.: Android Wearable Programming. Packt Publishing, 2015. 205 s. ISBN 978-17-852-8015-3.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 24. října 2019

Abstrakt

Cílem této práce je implementace metronomu pro smartphone s OS Android a smartwatch s OS Tizen, který umožňuje asynchronní fungování i navázání Bluetooth spojení a zasílání dat ze smartphone na smartwatch zařízení. Metronom má sloužit především hudebníkům hrajícím v kapele, kterým umožní ukládání specifického tempa pro konkrétní písničky. Tato tempa lze uložit do playlistů a zaslat pomocí Bluetooth komunikace. Aplikace ve smartwatch představuje jiný způsob zdůraznění tempa, a to pomocí vibrací. Další funkcí sloužící především pro bubeníky je vyhodnocování přesnosti úderů v rámci daného tempa pomocí smartphone aplikace. Pro implementaci byly použity programovací jazyky C# a Java společně s využitím SQLite databází.

Abstract

The purpose of this project is the implementation of metronome for smartphone with OS Android and smartwatch with OS Tizen. Both applications provide asynchronous functionality of metronome as well as data sharing through Bluetooth connection. These metronome applications should help musicians in music bands. It allows to save specific song's tempo into playlists and than share them via Bluetooth connection with smartwatch application. The smartwatch application produces vibrations instead of click sound. Other smartphone metronome functionality is to scan drum stroke by internal device microphone and compare it with metronome beat. The programming languages C# and Java with SQLite databases were used for implementation.

Klíčová slova

metronom, smartwatch, smartphone, chytrý telefon, chytré hodinky, GUI, Material Design, Android, Tizen, Java, C#, SQLite, Bluetooth.

Keywords

metronome, smartwatch, smartphone, GUI, Material Design, Android, Tizen, Java, C#, SQLite, Bluetooth.

Citace

POMKLA, František. *Metronom pro mobilní zařízení Android*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Metronom pro mobilní zařízení Android

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
František Pomkla
28. května 2020

Poděkování

Rád bych poděkoval panu Ing. Vladimíru Bartíkovi, Ph.D. za to, že mi umožnil realizovat můj nápad v rámci této práce. Dále velmi děkuji za jeho odbornou pomoc, konzultace i za komplikovaných okolností, příjemný přístup a rady při vypracovávání praktické i písemné části. V neposlední řadě bych také rád poděkoval členům hudební skupiny *Majvely* a všem ostatním za pomoc s testováním aplikace.

Obsah

Kapitola 1

Úvod

Pokud to někdo s hrou na hudební nástroj myslí vážně, musí zvládnout kromě patřičné techniky i udržet správnou rychlost hraní. V dnešní době je běžnou součástí výbavy hráčů i v hudebních uskupeních zařízení, které jim má pomoci s udržením správného tempa. Tímto zařízením je metronom. Na trhu s mobilními aplikacemi existuje velké množství metronomů, které nad rámec běžného přehrávání dob poskytují ještě některé další funkcionality. Neumožňují však vyhodnocovat, zda hudebník hraje správně rychle. Současným trhem ale hýbají i nová elektronická zařízení, kterými jsou chytré hodinky. Svou malou velikostí a umístěním na zápěstí umožňují praktické a rychlé ovládání, které by se dalo využít i při aplikaci, jako je metronom. Existující řešení na těchto zařízeních však nenabízejí potřebné funkce.

I výše zmíněné problémy řeší tato práce, jejímž cílem je vytvořit aplikaci pro chytrý telefon pro operační systém Android a aplikaci pro chytré hodinky pro operační systém Tizen. Tyto dvě aplikace zajišťují samostatný běh, ale především umožňují vzájemnou komunikaci. Účelem této komunikace je přenos seznamu uložených temp v aplikaci telefonu do aplikace v hodinkách, které lze na hodinkách následně přehrát pomocí vibrací. Při návrhu aplikace pro chytrý telefon a Bluetooth spojení jsem navázal na práci Bc. Aleše Ondráčka [37], kterou jsem dále rozšířil.

Obsah této práce je rozdělen dle kapitol do několika logických celků. V kapitole 2 budou stručně představeny operační systémy Android a Tizen, pro které byly aplikace vyvíjeny. V závěru této kapitoly jsou ještě zmíněny některé další významné OS chytrých hodinek. Popisu, slovnímu hodnocení a funkcionalitě existujících řešení pro obě platformy se věnuje kapitola 3. Následně je v kapitole 4 popsán první návrh obou aplikací včetně uživatelského rozhraní a požadovaných funkcí metronomu. Implementací obou aplikací se zabývá kapitola 5. Průběhu testování aplikací se věnuje kapitola 6. Zhodnocení dosažených výsledků a možné další pokračování ve vývoji je naznačeno v poslední kapitole 7.

Kapitola 2

Specifikace vytváření aplikací pro operační systémy Android a Tizen

V následující kapitole budou přiblíženy a stručně popsány operační systémy Android a Tizen. Vysvětlena bude architektura, druhy aplikací, vývojové prostředí a některé základní principy. Mnoho informací v kapitole 2.1 bylo čerpáno z knihy Programujeme pro Android [40].

2.1 Operační systém Android

Android je operační systém založený na jádře Linuxu, který byl vytvořen a je dále vyvíjen společností Google. Je založen na platformě open source, jde tedy o software s otevřeným zdrojovým kódem. Primárně slouží jako platforma pro PDA, tablety a chytré telefony. Jde o jeden z nejmladších operačních systémů a jeho první oficiální verze Android 1.0 spatřila světlo světa v roce 2008. Na konci roku 2019 se podíl systému Android na trhu s chytrými telefony pohyboval na téměř 75% [34].

2.1.1 Programovací jazyky

Prvním důležitým krokem pro vývoj Android aplikací je výběr programovacího jazyka. Aplikace se nejčastěji vyvíjí v následujících programovacích jazycích:

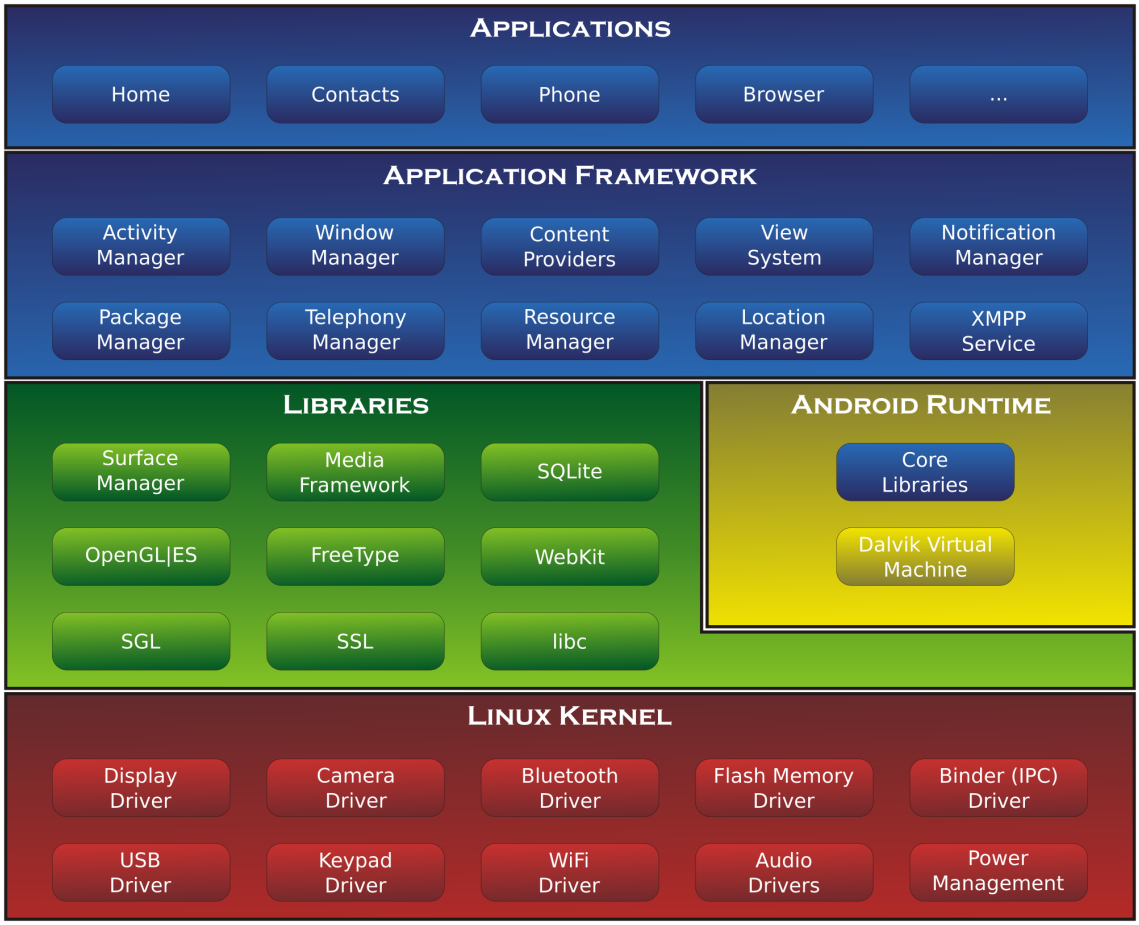
- **Java** – jde o oficiální a nejvíce populární programovací jazyk pro vývoj Android aplikací.
- **Kotlin** – moderní programovací jazyk představený v roce 2017 jako sekundární oficiální jazyk. V mnoha ohledech se podobá jazyku Java.
- **C/C++** – aplikace lze vyvíjet i v jazycích C/C++ za použití nástroje Android NDK [8]. Doporučuje se pro vývoj her pro Android platformu.

Pro úplnost zde zmiňuji ještě ne příliš preferované možnosti mezi vývojáři, kterými jsou jazyky: C#, BASIC, Corona a kombinace HTML, CSS a JavaScriptu [21].

2.1.2 Architektura systému

Architektura systému na obrázku 2.1 se skládá z pěti vrstev, přičemž každá z nich provádí různé operace. Nejnižší vrstvou je vrstva jádra Linuxu, jejíž hlavní úlohou je abstrakce

mezi použitým hardwarem a softwarem ve vyšších vrstvách. Další vrstvou jsou knihovny, které nabízí množství rozhraní API pro vývoj aplikací. Následuje vrstva obsahující virtuální stroj DVM (Dalvik Virtual Machine) a základní Java knihovny. Virtuální stroj DVM vznikl, protože programátoři vyvíjející aplikace pro Android využívali knihovny Java, které jsou open source, avšak virtuální stroj JVM (Java Virtual Machine) volně šiřitelný není. DVM využívá základních vlastností Linuxového jádra, jako jsou koordinace běžících procesů, správa paměti nebo práce s vlákny. Čtvrtou vrstvou, pro vývojáře nejdůležitější, je vrstva aplikační. Umožňuje jim přistoupit k nejrůznějším službám, které mohou vývojáři využít přímo ve svých aplikacích. Nejvyšší vrstvou jsou samotné aplikace, které jsou na zařízení již předinstalovány, či aplikace, které si může uživatel stáhnout z Google Play [40, str. 17–20].



Obrázek 2.1: Architektura systému Android. Převzato z [19].

2.1.3 Android API

Android je významným tahounem v implementaci nových technologií v mobilních telefonech a jeho vývoj je velmi rychlý. Se vznikem nových verzí, které využívají nové knihovny a umožňují nové funkce, vzniká problém s kompatibilitou se staršími verzemi. K řešení těchto potíží slouží úrovně Android API. Zajišťují kompatibilitu mezi naprogramovanými aplikacemi a přístroji, na kterých mohou být nainstalovány. Aktualizace rámce API je navržena tak, aby novější API byly kompatibilní i se staršími verzemi, jde tedy především

o aditivní změny. Vývojář se musí při vytváření nové aplikace rozhodnout, jestli má být určena co největšímu množství zařízení nebo pro nejnovější verzi Androidu [40, str. 47–48].

2.1.4 Životní cyklus aplikace

Android aplikace nemá kontrolu nad svým životním cyklem a běží ve svém vlastním procesu. Paměť a řízení procesu aplikace jsou řešeny za jejího běhu. V případě nedostatku systémové paměti je nucen operační systém ukončit některé méně důležité procesy. Jelikož není zaručen běh procesů aplikace, je důležité si některá data v rámci aplikace ukládat, aby nedošlo k jejich ztrátě. Automaticky se ukládají pouze stavy uživatelských rozhraní, ale informace z instančních proměnných se ztratí. K těmto účelům můžeme použít například datové úložiště typu klíč-hodnota nebo relační databáze SQLite [40, str. 39].

2.1.5 Životní cyklus aktivity

Pro pochopení různých stavů aplikace je důležité si vysvětlit životní cyklus aktivity na obrázku 2.2. Aplikace je běžně tvořena více aktivitami, které jsou mezi sebou provázány. Při spuštění aplikace se zobrazí jedna aktivita, která je obvykle určena jako hlavní. Každá aktivita umožňuje spouštět další aktivitu. Při spuštění nové aktivity je předchozí aktivita pozastavena a uchována v zásobníku. Zásobník představuje fronta typu LIFO (last in, first out – poslední dovnitř, první ven) [40, str. 40].

Rozlišujeme 7 metod životního cyklu aktivity:

- **onCreate** – metoda je volána při vytvoření aktivity
- **onStart** – metoda je volána, když se aktivita stane viditelnou pro uživatele
- **onResume** – metoda je volána, pokud uživatel začne komunikovat s aktivitou
- **onPause** – metoda je volána, když aktivita přestane být viditelnou pro uživatele
- **onStop** – metoda je volána, když aktivita přestane být viditelnou pro uživatele, a to pro účely pokračování komunikace jiné aktivity
- **onRestart** – metoda je volána poté, co je aktivita zastavena a má být opět obnovena
- **onDestroy** – metoda je volána před ukončením aktivity [30]

V rámci životního cyklu aktivity lze sledovat tři hlavní smyčky:

Úplný životní cyklus aktivity

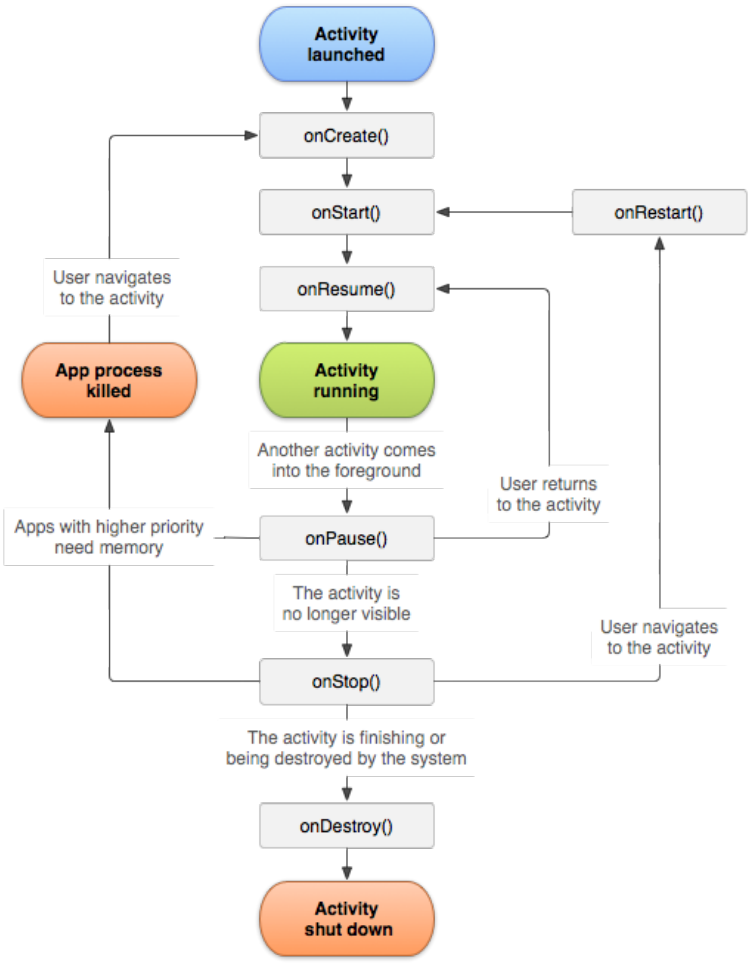
Průchod celým životním cyklem začíná voláním metody onCreate a končí voláním metody onDestroy. Při vytvoření aktivity se provedou činnosti, které se mají provést jen jednou (nastavení GUI, inicializace některých proměnných...), a nastaví se její globální stav pro případ zpětného obnovení. Před ukončením aktivity se uvolní všechny prostředky, které aktivita využila.

Viditelný životní cyklus aktivity

Viditelný životní cyklus se odehrává mezi metodami onStart a onStop. V této době vidí uživatel aktivitu na displeji svého zařízení. Zároveň lze během této doby udržovat prostředky potřebné k zobrazení aktivity pro uživatele.

Životní cyklus aktivity v popředí

Do popředí se aktivita dostává mezi metodami onResume a onPause. To umožňuje aktivitě zobrazit se nad všemi ostatními aktivitami a komunikovat s uživatelem. Podle následné interakce s uživatelem lze aktivitu pozastavit nebo pokračovat v jejím běhu, a to i několikrát za sebou [40, str. 41].



Obrázek 2.2: Životní cyklus aktivity. Převzato z [30].

2.1.6 Vývojové prostředí

Pro mobilní část aplikace lze zvolit vývojové prostředí Android Studio založené na IntelliJ IDEA, které vyvíjí firma Google. Jde o doporučované prostředí, které umožňuje vývoj aplikací pro Android nejjednodušším a nejkomfortnějším způsobem. Je k dispozici pro platformy Windows, Mac OS a Linux.

Jako alternativu a předchůdce Android Studia zmiňuji vývojové prostředí Eclipse v kombinaci s pluginem Android Development Tools [9].

2.1.7 Material Design

Velké množství aplikací pro Android se v dnešní době řídí tzv. *Material Designem* [1]. Jedná se o určitý standard a designový jazyk vyvinutý společností Google, který má sjednotit vzhled Android aplikací. Umožňuje vytvářet uživatelské prostředí, které reaguje na dotek uživatele velmi intuitivním způsobem, nastavuje citlivé animace a přechody a používá jasnější barvy, aby vynikly důležité informace [33].

2.2 Operační systém Tizen

Tizen je operační systém, který je stejně jako Android založený na linuxovém jádře a softwarové knihovně GNU C Library. Je vyvíjen především společnostmi Linux Foundation, Samsung, Intel a dalšími. Je určený pro chytré telefony, televize, tablety, chytré hodinky a IoT. První verze operačního systému Tizen 1.0 sloužila pouze pro webové aplikace. Od verze 2.0 je možné vyvíjet i nativní aplikace a od verze 3.0 je možné aplikace vyvíjet za pomoci .NET Core frameworku [35].

2.2.1 Druhy aplikací

Existují tři způsoby jak vyvíjet aplikace pro Tizen:

- **Webové aplikace**
- **Nativní aplikace**
- **Tizen .NET aplikace**

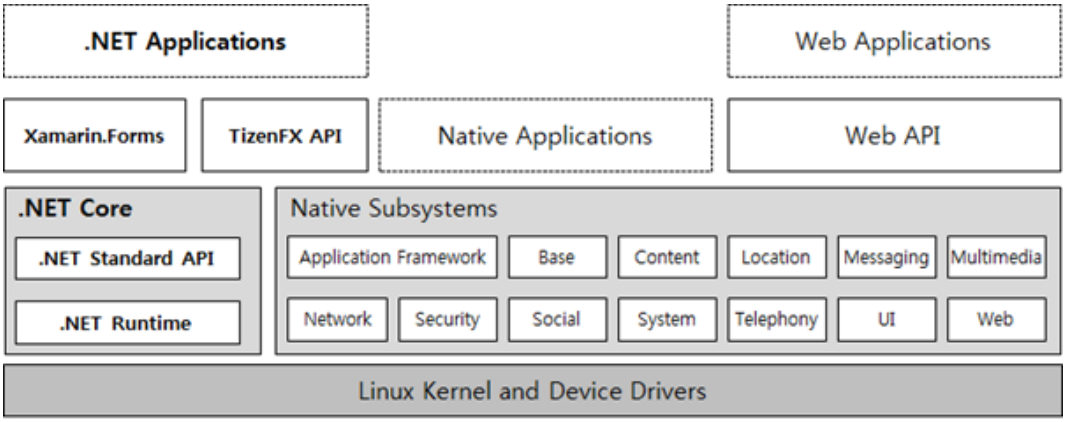
Pro vývoj webových aplikací se používají programovací jazyky HTML5, CSS a JavaScript. Tyto aplikace využívají Tizen Web Framework. Vývoj webových aplikací je jednoduchý s minimální učící křivkou.

Pro vývoj nativních aplikací se používají programovací jazyky C a C++. Umožňují vývoj pokročilejších aplikací s velkým množstvím funkcí za využití různých senzorů zařízení a existujících knihoven programovacích jazyků C a C++.

Pro vývoj Tizen .NET aplikací se používá programovací jazyk C#, Tizen .NET framework a rozhraní Xamarin.Forms, které umožňuje definovat uživatelské rozhraní aplikace.
Tizen .NET představuje nový a efektivní způsob, jak vyvíjet aplikace pro Tizen [17].

2.2.2 Architektura systému

Architektura systému znázorněná na obrázku 2.3 je složena ze tří vrstev. První vrstvou je vrstva jádra Linuxu a systémových řadičů, jejíž hlavní úlohou je abstrakce mezi použitým hardwarem a softwarem ve vyšších vrstvách. Další vrstvou je vrstva jádra. Ta umožňuje přistoupit k nejrůznějším službám systému (například k mediálním, webovým, síťovým) a také k rozhraním API. Nejvyšší vrstvou jsou již samotné aplikace – nativní, webové a nebo .NET aplikace.



Obrázek 2.3: Architektura operačního systému Tizen. Převzato z [17].

2.2.3 Vývojové prostředí

Vývojové prostředí Tizen Studio je primárně určeno pro vývoj nativních a webových aplikací. Pro Tizen .NET aplikace je primárním vývojovým prostředím Visual Studio s rozšířením pro vývoj Tizen aplikací. Pro účely debuggování .NET aplikací je nutné použít také Tizen Studio.

2.3 Některé další operační systémy pro chytré hodinky

Kromě výše zmíněné a popsané platformy Tizen zde pro přehled zmíním ještě některé další významné tahouny na trhu operačních systémů chytrých hodinek. Musím zde okrajově zmínit i OS Garmin, o kterém jsem ale našel pouze minimum informací.

2.3.1 Operační systém watchOS

Operační systém watchOS vyvinutý společností Apple je určený pro chytré hodinky Apple Watch. Vychází z operačního systému iOS. První verze watchOS spatřila světlo světa v roce 2015 [39, str. 279]. Aktuální verze watchOS 6 byla vyvinuta na konci roku 2019 [31].

Vývojové prostředí

Standardně se aplikace pro iOS vyvíjí ve vývojovém prostředí Xcode. Pro aplikace watchOS je nutné, aby bylo toto prostředí rozšířeno o Watch Kit framework. Programovacím jazykem vývojářů je Swift nebo Objective-C [20].

2.3.2 Operační systém Wear OS

Wear OS (původně Android Wear) je operační systém vyvíjený společností Google. Jde o verzi operačního systému Android, která je určena pro chytré hodinky. První verze tohoto systému vyšla v roce 2014 [36].

Vývojové prostředí

Obdobně jako aplikace pro Android i aplikace pro Wear OS se doporučuje vyvíjet v Android Studiu [16].

Kapitola 3

Existující řešení pro platformy Android a Tizen

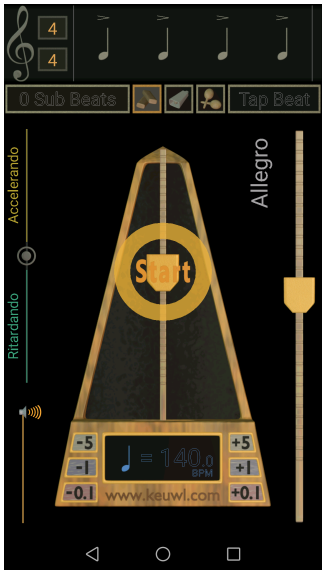
V této kapitole se seznámíme s některými existujícími řešeními aplikací metronomu pro systém Android, které jsou k dispozici ke stažení z obchodu Google Play. Vybral jsem takové aplikace, které jsou oblíbené mezi uživateli a zároveň rozdílné, co se týče vzhledu i funkcionality. Pro hodinky běžící na systému Tizen jsem nalezl tři existující řešení, která zde zmíním.

3.1 Existující řešení pro Android

Metronomů pro operační systém Android existuje mnoho a velká část z nich nabízí různé funkcionality nad rámec pouhého přehrávání dob jako je zrychlování tempa nebo ukládání temp do seznamů – playlistů. Existují však i řešení nabízející funkce nad rámec metronomu a disponují například ladičkou na hudební nástroje nebo možností během přehrávání metronomu nahrávat zvuk z mikrofonu zařízení.

3.1.1 Metronome (keuwlsoft)

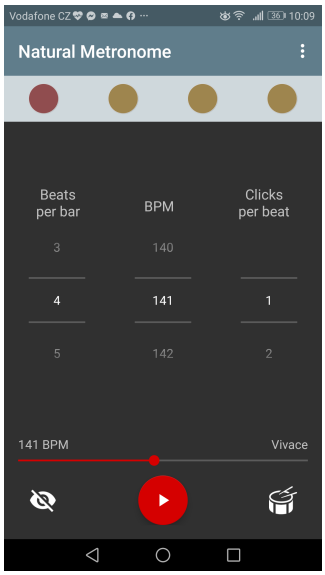
První existující řešení 3.1 je specifické svým vzhledem, který připomíná klasický dřevěný metronom včetně kyvadla. Ovládání je poměrně intuitivní, avšak nepříliš praktické. Tlačítka jsou velmi malá, a to především ta pro změnu rychlosti tempa. Problematické je i nastavování tempa na postranním seekbaru. Naopak pozitivní je fakt, že umožňuje zrychlovat nebo zpomalovat tempo po taktech, a to například i o 0.5 BPM - beats per minute (údery za minutu). Umožňuje nastavit tempo pomocí klikání na tlačítko, a tak zjištění tempa písničky, kterou uživatel právě poslouchá, není nikterak problematické. Možnost nastavení poddob pod každou dobou je velmi praktická - pokud uživatel potřebuje změnit vyhrávání ze čtvrtinových dob na osminové, může to udělat jednoduše pomocí tohoto tlačítka. Na druhou stranu zvuková banka s nabídkou 3 zvuků mě příliš nenadchla. Občas se také přihodí, že metronom lehce zrychlí nebo zpomalí. Metronom bohužel nefunguje na pozadí a ani při vypnuté obrazovce, což jsou pro mě nepostradatelné funkce. Další komplikací je nemožnost ukládat různá tempa do playlistů. Výsledný vzhled také není příliš zdařilý. Myslím, že tato aplikace může velmi dobře posloužit pro osobní příležitostní cvičení na hudební nástroj, pro potřeby hraní v kapele je ale silně nedostačující.



Obrázek 3.1: Metronome

3.1.2 Natural metronome

Aplikace znázorněna na obrázku 3.2 cílí na vzhledově jednoduché a funkční řešení. Nabízí nastavování tempa pomocí seekbaru, výběru z čísel a rychlosti klikání na tlačítko. Umožňuje nastavit počet dob v taktu a, stejně jako předchozí řešení, nastavit poddoby na každou dobu. Bohužel nelze změnit akcent první doby a ani nastavit akcent na žádnou jinou dobu. Metronom je funkční i na pozadí a při zhasnuté obrazovce. Opět zde není žádná možnost uložit si tempa do playlistů a pak jednoduše nastavovat požadovaná tempa písniček. V základní verzi má aplikace pouze jeden zvuk, což lze za příplatek změnit. Velká výhoda tohoto řešení je jeho jednoduché a intuitivní ovládání. Schází ale potřebné funkce jako změna zvuku a především ukládání do playlistů.

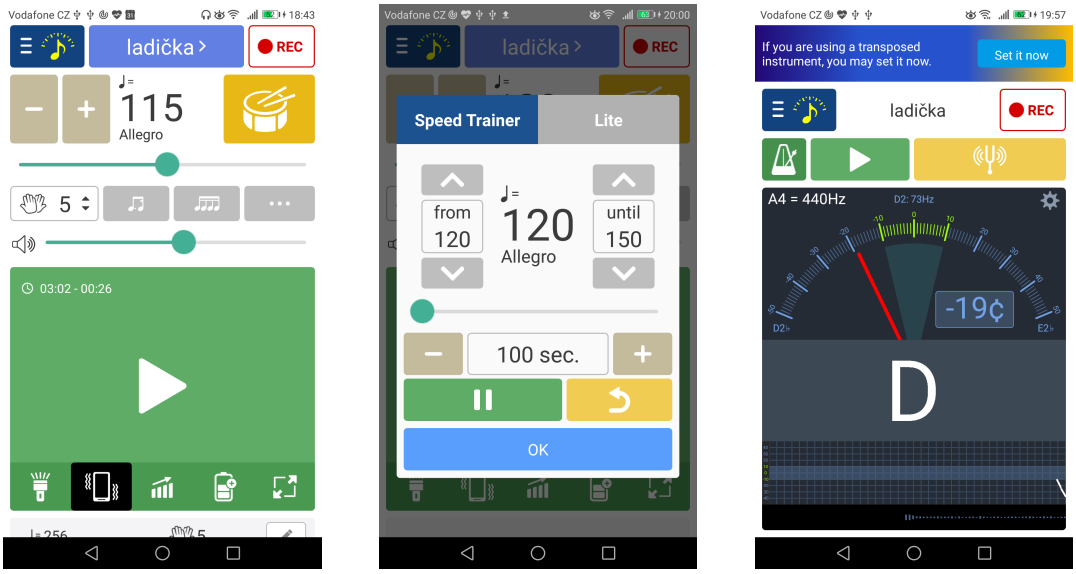


Obrázek 3.2: Natural metronome

3.1.3 Tuner a Metronom

Tuner a Metronom 3.3 od vývojářů Soundcorset je metronom a ladička v jedné aplikaci. Z pohledu metronomu nabízí mnoho funkcí. Nastavení tempa lze provést pomocí klikání na tlačítko nebo přejetím prstu po seekbaru. Pro přesné nastavení tempa slouží tlačítka + a -, která přidávají nebo odebírají 1 BPM. Stejně jako u dříve zmíněných metronomů lze nastavit i počet akcentů. Aplikace podporuje nastavení rytmu, který bude přehrávat na každý beat, ať už jde o čtvrtiny, trioly nebo osminy. Navíc má nahrané i základní rytmy bicích nástrojů jako jsou: Rock Groove, Swing, Blues nebo třeba Bossa Nova, které lze přehrát ve zvoleném tempu. Tuto funkcionalitu by mohli ocenit především uživatelé hrající a tvořící písničky na melodické nástroje. Dále lze nastavit zrychlování tempa. Nejdříve zvolíme výchozí a konečné tempo, do kterého má metronom dojít. Následně nastavíme čas, za který metronom zrychlí z výchozího do konečného tempa. Jinou možností jak zrychlovat nebo zpomalovat metronom je nastavení časového úseku, za který se má metronom zrychlit nebo zpomalit o 1 BPM. Další funkcionalitou je zobrazení historie přehrávání jednotlivých temp, které lze poté jedním kliknutím opět obnovit. Navíc se tyto položky v historii dají pojmenovat a tudíž může tato vlastnost poskytnout uživateli seznam s příslušnými tempy pro přehrávání. Nevýhodou je, že se tyto položky v historii nedají uspořádat do playlistů. Dále lze přímo v rámci aplikace nahrávat zvuk z mikrofonu zařízení a přehrávat písničky v zařízení společně s metronomem. Zajímavé je, že si aplikace uchovává informaci o tom, jak dlouho byl metronom spuštěn. Poté uživateli poskytuje graf hraní v rámci uplynulého týdne i měsíce.

Součástí aplikace je i poměrně citlivá ladička, která může posloužit uživatelům, kteří hrají na strunné nástroje.



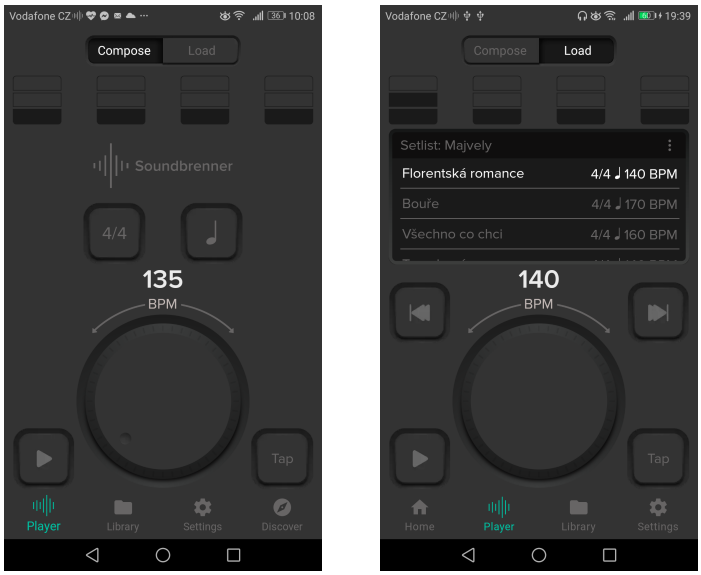
(a) Metronom (b) Zrychlování tempa (c) Ladička

Obrázek 3.3: Tuner a metronome

3.1.4 Soundbrenner

Dle mého názoru jde o nejzdařilejší řešení aplikace pro účely hudebních uskupení, které má nejvíce funkcí v bezplatné verzi. Metronom na obrázku 3.4 disponuje nastavováním

tempa pomocí kolečka nebo rychlostí klikání na tlačítko. Umožňuje nastavovat akcenty u každé doby individuálně. Dále je možné vytvářet playlisty s tempy písniček, které lze následně na samostatné obrazovce spouštět. Také můžeme nastavit zvuk metronomu z více než 10 různých možností. A pokud chceme, lze zapnout blikání LED diody nebo obrazovky na každou dobu. Metronom zajišťuje spojení několika mobilních zařízení v rámci aplikace pomocí služby Ableton Link [7], následně umožňuje synchronní běh metronomů. Společnost Soundbrenner také vytvořila Soundbrenner Pulse. Jde o hodinky, které umožňují propojení s mobilní aplikací a synchronní fungování, kdy hodinky vibrují na zápěstí podle zvoleného tempa. V roce 2019 vytvořila tato společnost novou verzi hodinek Soundbrenner Core 3.5, které umožňují samostatné fungování bez této mobilní aplikace [28].



(a) Okno s metronomem (b) Okno s playlisty

Obrázek 3.4: Soundbrenner aplikace

Obrázek 3.5: Soundbrenner Core. Převzato z [27].

3.2 Existující řešení pro Tizen

Operační systém Tizen se zdaleka nepyšní takovou popularitou jako Android. Jde to znát i na počtu aplikací, které lze pro hodinky stáhnout v internetovém obchodě *Galaxy Store*. To se však poslední dobou mění a počet aplikací pro Tizen stále přibývá.

3.2.1 Wearable Metronome

Jedná se o jednoduchou, avšak velmi povedenou aplikaci s jedním hlavním oknem a zdařilým vzhledem 3.6. Po zapnutí metronomu stisknutím tlačítka na obrazovce začnou hodinky podle zvoleného tempa vibrovat a blikat. Vibrace jsou dostatečně silné a dlouhé. Neumožňuje však ukládání různých temp do playlistů a jejich následné přehrávání. Pokud uživatel potřebuje rychle nastavit velmi rozdílné tempo písničky, musí poměrně dlouho otáčet lunetou hodinek.

Obrázek 3.6: Wearable metronome

3.2.2 Metrónome

Tato aplikace má opět jedno hlavní okno 3.7 a kromě vibrací na každou dobu umožňuje přehrávat i zvuk, buď pomocí vestavěného reproduktoru, nebo po připojení bezdrátových Bluetooth sluchátek. Problémem je však asynchronní zvuk ve sluchátkách s vibracemi v hodinkách. Aplikace dále umožňuje přehrávat akcenty vždy na první dobu v podobě silnějších vibrací, avšak rozdíl v intenzitě je jen velmi malý. Tento akcent lze nastavovat pomocí čísla, které představuje počet dob v rámci jednoho taktu. Tyto doby se následně zobrazují velkým číslem v horní části displeje. Velkým nedostatkem a v praxi velmi potřebnou funkcí je běh aplikace i v případě, že se zhasne obrazovka. Metrónome však tento běh na pozadí neumožňuje.

Obrázek 3.7: Metrónome. Převzato z [22].

3.2.3 MetronomeVibrations

MetronomeVibrations 3.8 představuje ze všech aplikací pro Tizen nejjednodušší řešení. Ihned po otevření začne metronome vibrovat a nelze pozastavit. Nabízí možnost nastavit tempo pouze z 9 přednastavených hodnot v rozmezí tempa od 40 BPM po 200 BPM, což pro potřeby hraní v kapele zdaleka nedostačuje. Zároveň neumožňuje ani běh aplikace na pozadí, takže se po zhasnutí displeje hodinek vypne.

Obrázek 3.8: MetronomeVibrations

3.3 Shrnutí existujících řešení

V následujících dvou tabulkách shrnuji funkcionalitu existujících řešení.

Název aplikace	Zjišťování tempa pomocí klikání	Funkce během vypnuté obrazovky	Možnost zrychlování tempa v průběhu hraní	Ukládání tempa do seznamu	Možnost propojení s hodinkami	Snímání a vyhodnocování přesnoti úderů
Metronome (keuwlsoft)	✓	x	✓	x	x	x
Natural metronome	✓	✓	x	x	x	x
Tuner a Metronom	✓	✓	✓	(✓)	x	x
Soundbrenner	✓	✓	x	✓	(✓)	x

Tabulka 3.1: Existující řešení – Android

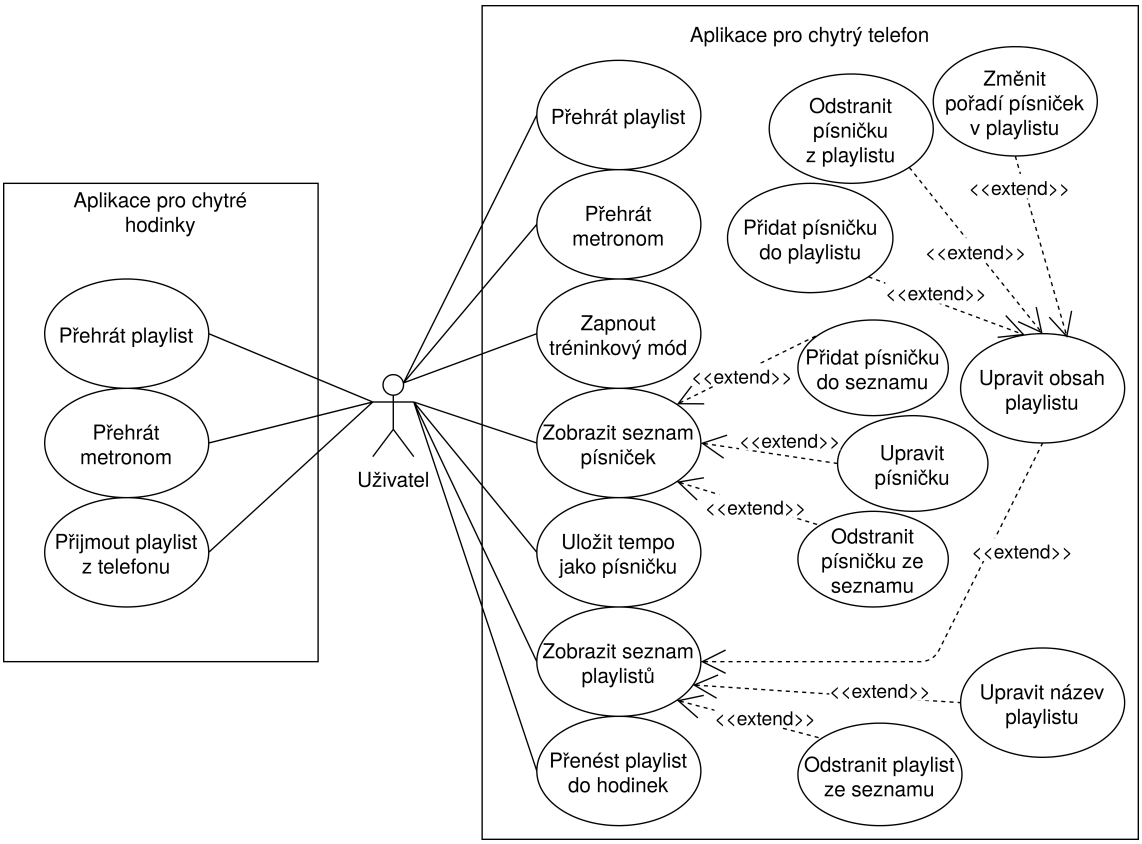
Název aplikace	Nastavení přesného tempa	Funkce během vypnuté obrazovky	Možnost přehrávat zvuk	Možnost blikání obrazovky podle tempa	Možnost propojení s telefonem
Wearable Metronome	✓	✓	x	✓	x
Metrónome	✓	x	✓	x	x
MetronomeVibrations	x	x	x	x	x

Tabulka 3.2: Existující řešení – Tizen

Kapitola 4

Návrh aplikace pro chytrý telefon a chytré hodinky

V následující kapitole si shrneme požadavky a představíme si první návrh metronomu včetně jeho GUI. Tyto návrhy představím zvlášť pro chytrý telefon a pro chytré hodinky. Na konci této kapitoly se ještě zmíním o plánované architektuře obou aplikací. Následující diagram 4.1 demonstruje případy užití aplikací.



Obrázek 4.1: Diagram případů užití aplikace pro chytré hodinky a pro chytrý telefon

4.1 Aplikace pro telefon

Mobilní aplikace by měla umožňovat více funkcí ve srovnání s aplikací pro hodinky. To je dáno třeba větším displejem telefonu, který poskytuje jednoduché psaní na klávesnici a zobrazování údajů. Dalšími výhodami je možnost přehrávání zvuku z integrovaného reproduktoru nebo ze sluchátek a v neposlední řadě i mnohonásobně větší kapacita baterie, než kterou disponují hodinky.

4.1.1 Požadavky pro metronom

Aplikace by měla poskytovat následující funkce:

- přehrávání metronomu včetně jednoduchého nastavování a následné úpravy tempa
- nastavování tempa pomocí klikání na tlačítko v tempu poslouchané melodie
- změna zvuku metronomu a možnost zrychlovat metronom
- nastavení počtu dob v rámci jednoho taktu a přehled odehraných taktů
- možnost ukládat tempo jako písničku s názvem a příslušným tempem
- možnost ukládání písniček do playlistů, úprava pořadí písniček v rámci playlistu
- přehrávání playlistů a přepínání mezi písničkami v rámci playlistu i přepínání mezi playlisty samotnými
- propojení aplikace v chytrém telefonu s aplikací v chytrých hodinkách a přenos playlistů z telefonu do hodinek
- tréninkový mód pro snímání přesnosti úderů na danou dobu včetně následného vyhodnocení

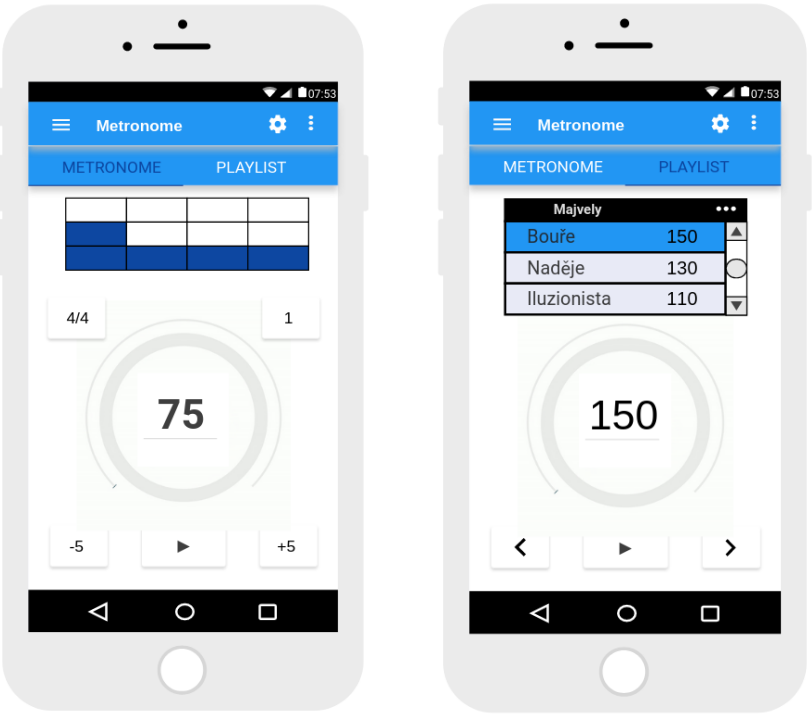
Grafické uživatelské rozhraní metronomu by mělo uživateli poskytnout všechny potřebné funkce, které budou snadno dohledatelné a spustitelné. Během hudebních zkoušek, a především vystupování na koncertech, je důležité, aby bylo ovládání rychlé a zabralo co možná nejméně času. Dlouhá časová prodleva by totiž mohla narušit hladký průběh vystupování kapely. Uživatel by měl mít k dispozici několik možných zvuků metronomu, aby si mohl vybrat na základě svých preferencí. Ukládání písniček do playlistů je vhodné dělat během hudebních zkoušek, kdy si kapela stanoví tempo písničky a pořadí písniček v playlistu na budoucí koncert. V případě potřeby uživatel zašle svou databázi playlistů do svých hodinek. Tréninkový mód by měl sloužit především uživatelům z řad bubeníků, kterým poslouží pro zlepšení jejich přesnosti.

4.1.2 Návrh GUI

GUI metronomu pro mobilní zařízení je složeno ze dvou fragmentů – dvou samostatných obrazovek, a to metronomu a playlistu. Výchozí obrazovka metronomu a současně první fragment aplikace 4.2a obsahuje v panelu nástrojů možnost nastavení. V rámci další nabídky je zde spárování s aplikací pro chytré hodinky, přechod do tréninkového módu, uložení tempa pod specifickým názvem a seznam uložených temp – songlist. Dále je tu pole zobrazující počet dob v rámci jednoho taktu a nastavení akcentů na jednotlivé doby. Pod polem se

vlevo nachází tlačítko nastavující druh taktu, které dynamicky vkládá do pole počet dob na základě tohoto tlačítka. Vpravo pod polem je situováno tlačítko pro nastavení doby pomocí klikání. Uprostřed obrazovky je navrhnut otočný seekbar s kolečkem umožňující nastavení tempa, které se zobrazuje uprostřed tohoto kolečka. Vpravo dole se nachází tlačítko pro rychlou změnu tempa o 5 BPM navíc a vlevo tlačítko pro snížení tempa o 5 BPM. Mezi nimi se nachází tlačítko na zapnutí a vypnutí metronomu.

V druhém fragmentu 4.2b v playlistu se nachází tabulka se seznamem písniček, obsahující název písničky a příslušné tempo. V rámci tabulky je mezi těmito písničkami možno listovat. Pro pohodlnější přepínání mezi písničkami v playlistu slouží tlačítka vpravo a vlevo dole. Po výběru písničky se uprostřed kolečka načte informace o tempu písničky a tlačítkem uprostřed dole lze metronom zapnout a vypnout. V rámci panelu nástrojů je zde nastavení, spárování s aplikací pro chytré hodinky, přejití do tréninkového módu a seznam playlistů.



(a) Okno s metronomem (b) Okno pro přehrávání playlistů

Obrázek 4.2: Návrh mobilní aplikace

4.2 Aplikace pro hodinky

Metronom na chytrých hodinkách by měl být velmi jednoduše ovladatelný. Jeho hlavní funkcí je přehrávat tempo metronomu pomocí vibrací a umožnit přehrávání písniček v rámci playlistu. Tyto playlisty jsou mu zaslány z aplikace telefonu.

4.2.1 Požadavky pro metronom

Aplikace by měla disponovat následujícími funkcemi:

- vibrace udávající tempo metronomu

- jednoduché nastavování a následné úpravy tempa
- přehrávání playlistů a přepínání mezi písničkami v rámci playlistu i přepínání mezi playlisty samotnými
- propojení aplikace v chytrých hodinkách s aplikací v chytrém telefonu a přenos playlistů z telefonu do hodinek

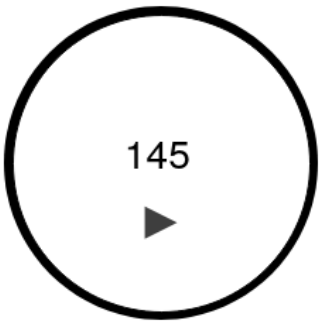
4.2.2 Návrh GUI

GUI metronomu pro chytré hodinky je přizpůsobeno vzhledu aplikací, které běží na operačním systému Tizen na hodinkách od firmy Samsung. Úvodní obrazovka 4.3a obsahuje položky k výběru. Jsou to nastavení, spuštění metronomu a otevření seznamu playlistů.

Při volbě metronomu se zobrazí jednoduché rozhraní 4.3b. Dole je tlačítko pro spuštění metronomu a uprostřed se nachází aktuální přehrávané tempo, které lze upravit otáčením lunety, jíž disponují chytré hodinky od firmy Samsung.

Playlist 4.3c obsahuje informace o dané písničce, jako jsou její tempo a název. Dále nabízí možnost přepínání mezi písničkami z playlistu a, stejně jako na obrazovce metronomu, i tlačítka pro spuštění.

(a) Úvodní okno



(b) Okno s metronomem

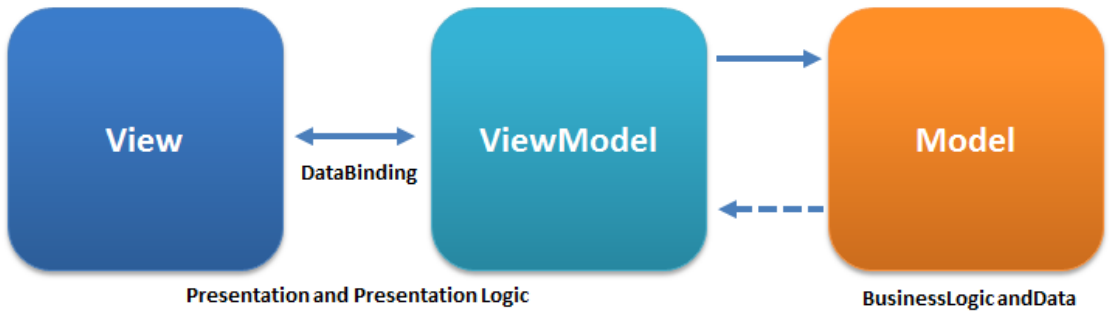
(c) Okno pro přehrávání playlistu

Obrázek 4.3: Návrh aplikace pro chytré hodinky

4.3 Architektura aplikací

Pro mobilní i hodinkovou aplikaci bych zvolil návrhový vzor Model–View–ViewModel (MVVM) 4.4. Nabízí praktické řešení, jak oddělit uživatelské rozhraní od logiky aplikace. Skládá se ze tří vrstev:

- **Model** – vrstva uchovávající a řídící data, se kterými aplikace pracuje. Nemá žádný přehled o uživatelském rozhraní.
- **View** – vrstva představující uživatelské rozhraní. V nejběžnějším případě se jedná o okno aplikace nebo stránku představující UI. V rámci vývoje aplikací pro Android v programovacím jazyce Java se jedná o XML layouty. Při vývoji .NET Tizen aplikací se jedná o XAML soubory.
- **ViewModel** – vrstva spojující View a Model. Zároveň si udržuje informace o stavu aplikace. Zpracovává data z View vrstvy a ukládá je do Model vrstvy. V rámci Android aplikací se k tomu používá tzv. obousměrný data binding. Pro .NET Tizen aplikace se používá obousměrný data binding v kombinaci s implementací rozhraní INotifyPropertyChanged [23].



Obrázek 4.4: Návrhový vzor MVVM. Převzato z [13].

Pro účely ukládání songlistů a playlistů bych využil databázi SQLite.

Kapitola 5

Implementace aplikace pro chytrý telefon a chytré hodinky

V této kapitole bude vysvětlena samotná implementace obou aplikací. Průběh práce jsem si zálohoval pomocí verzovacího systému Git a data si ukládal také na stránku GitHub.

5.1 Výsledné řešení – aplikace pro telefon

Pro vývoj aplikace pro telefon jsem zvolil vývojové prostředí Android studio v kombinaci s programovacím jazykem Java.

Barvy aplikace

Barvy uživatelského rozhraní byly vybrány za pomocí barevného systému *Material Designu* [15]. V rámci tohoto systému si vývojář zvolí primární a sekundární barvu. Primární barva je v rámci aplikace zobrazována nejčastěji. Sekundární barva slouží jako odlišný a doplňující prvek a doporučuje se ji využívat pro plovoucí tlačítka, zvýrazněný text nebo pro zobrazení nějakého procesu. Následně jsou vývojáři automaticky nabídnuty ještě světlé a tmavé odstíny těchto barev, které může použít pro dosažení ještě lepšího designu svého UI [14]. Pro svou aplikaci jsem jako primární barvu zvolil tmavší odstín modré a jako sekundární barvu oranžovou.

Grafické uživatelské rozhraní

Vzhled výsledného řešení aplikace do značné míry vychází z výše uvedeného návrhu. Došlo ale k několika změnám, které popíšu v následující části práce. Vzhled cílí na jednoduché, funkční a intuitivní řešení.

Hlavní okno

V hlavním okně s metronomem 5.1a zůstalo tlačítko pro nastavení počtu dob v rámci taktu a tlačítko pro zapnutí a vypnutí metronomu.

Ke změnám došlo v rámci panelu nástrojů, kde jsou na hlavní obrazovce nově zobrazeny i možnosti otevření songlistu, uložení daného tempa a možnost spojení s hodinkami. V dalších možnostech se skrývají nastavení a zapnutí tréninkového režimu. Akcenty byly z původních polí nahrazeny jednotlivými kroužky, které při zapnutém metronomu mění

svou velikost, a tím zobrazují aktuální přehrávanou dobu v rámci taktu. Tlačítko pro nastavení tempa podle klikání bylo nově umístěno do levé dolní části okna. Na jeho původním místě (vpravo nahoře) bylo vytvořeno pole pro zobrazení aktuální doby a taktu. Tlačítka pro rychlé zrychlení a zpomalení tempa o 5 BPM dostaly také nový kruhový design.

Když jsem přemýšlel nad tím, jakým způsobem nastavovat tempo metronomu, i z existujících řešení mi jako nejpraktičtější připadá použití otočného seekbaru. Zároveň jsem cílil na synchronizaci designu aplikací s hodinkami, které mají kruhový displej. Bohužel jsem zjistil, že Android Studio ve své paletě nástrojů disponuje pouze řádkovým seekbarem. Musel jsem tedy nalézt tuto komponentu mezi vývojáři pro Android a nakonec jsem využil *Croller* [18].

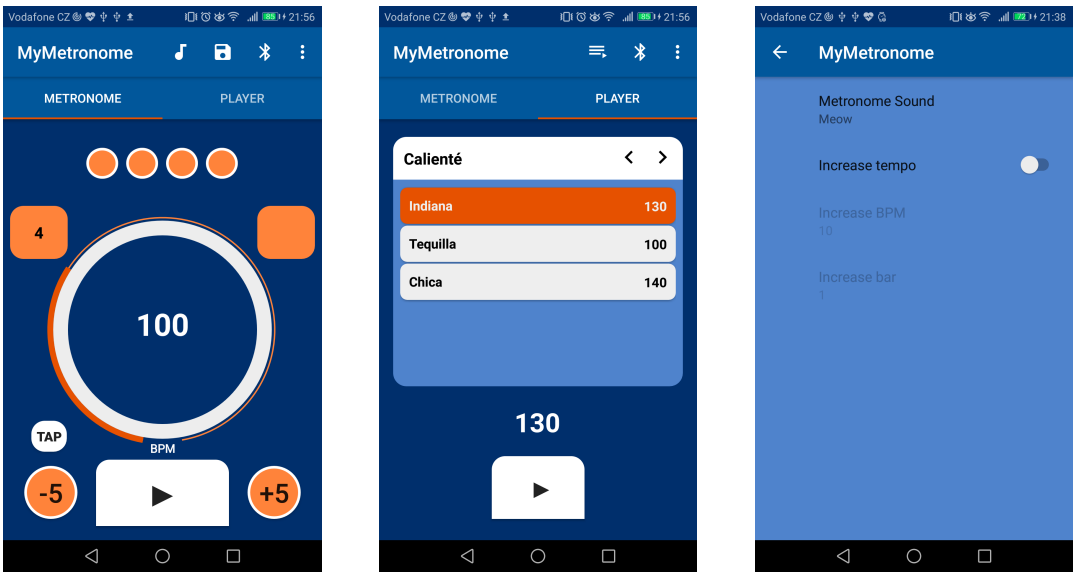
Playlist

Okno s playlisty 5.1b si zachovalo seznam písniček a tlačítko pro spuštění a vypnutí metronomu v dolní části obrazovky.

Nově byl upraven panel nástrojů, který nyní obsahuje možnost otevření playlistů a možnost spojení s hodinkami. V dalších možnostech jsou opět nastavení a tréninkový mód. V seznamu písniček v playlistu lze pomocí šipek přepínat mezi playlisty. Byl odstraněn otočný seekbar, aby nedocházelo ke změnám tempa již uložených písniček. Na jeho místě je nyní číslo označující tempo aktuálně přehrávané písničky.

Nastavení

V rámci nastavení 5.1c je možno změnit zvuk metronomu. Další možností je nastavení zrychlování nebo zpomalování metronomu o určený počet BPM za určený počet taktů.



(a) Okno s metronomem (b) Okno s playlisty (c) Okno s nastavením

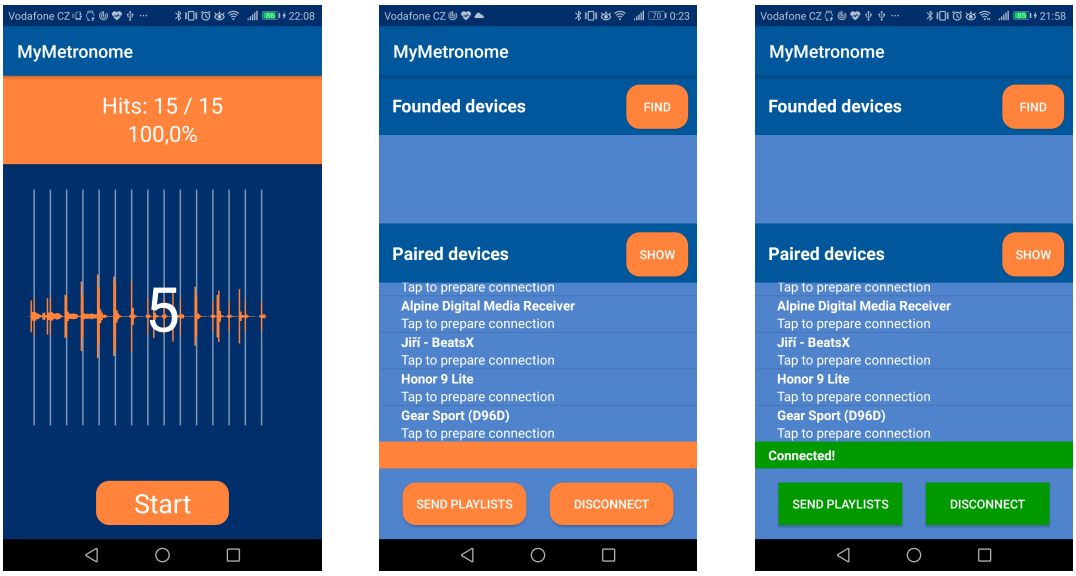
Obrázek 5.1: Aplikace MyMetronome

Tréninkový mód

V tréninkovém módu 5.2a se uživateli vykresluje graf na základě snímání zvuku z mikrofonu. Zvuk snímaný z mikrofonu, který je vykreslován oranžovou barvou je porovnán s přesnou dobou danou metronomem, která je vykreslována bílými čarami. V horní části okna se následně zobrazí počet přesných úderů a počet všech uplynulých dob metronomu. Po ukončení tréninkového módu se zobrazí i procentuální úspěšnost přesných úderů vůči všem dobám.

Bluetooth spojení

V okně pro Bluetooth spojení 5.2b si uživatel vybere ze seznamu Bluetooth spárovaných zařízení nebo provede párování s naleznutým zařízením. Následně ustanoví spojení a čeká, až je spojení navázáno druhým zařízením 5.2c.

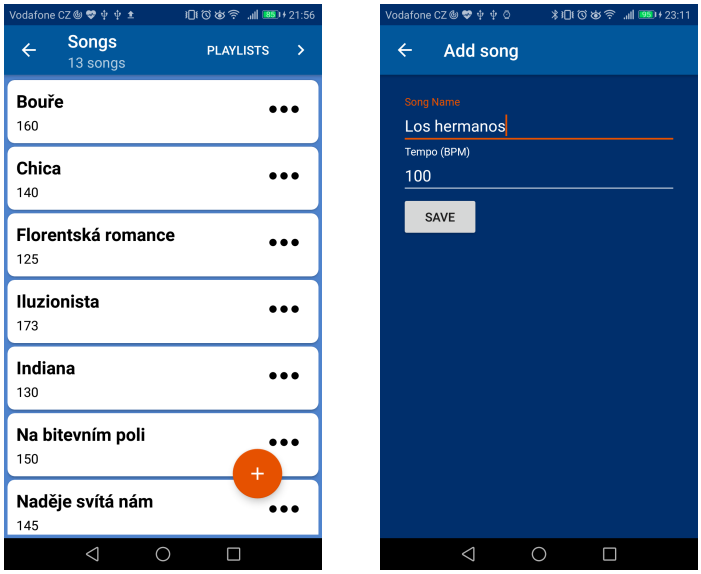


(a) Tréninkový mód (b) Bluetooth spojení (c) Spojení navázáno

Obrázek 5.2: Aplikace MyMetronome

Seznam s písničkami

Seznam písniček 5.3a obsahuje položky, které jsou vždy složeny z názvu a příslušného tempa v BPM. Tyto položky lze upravovat nebo mazat pomocí kliknutí na možnost *další* úplně vpravo. Pro přidávání písniček 5.3b slouží plovoucí tlačítko.

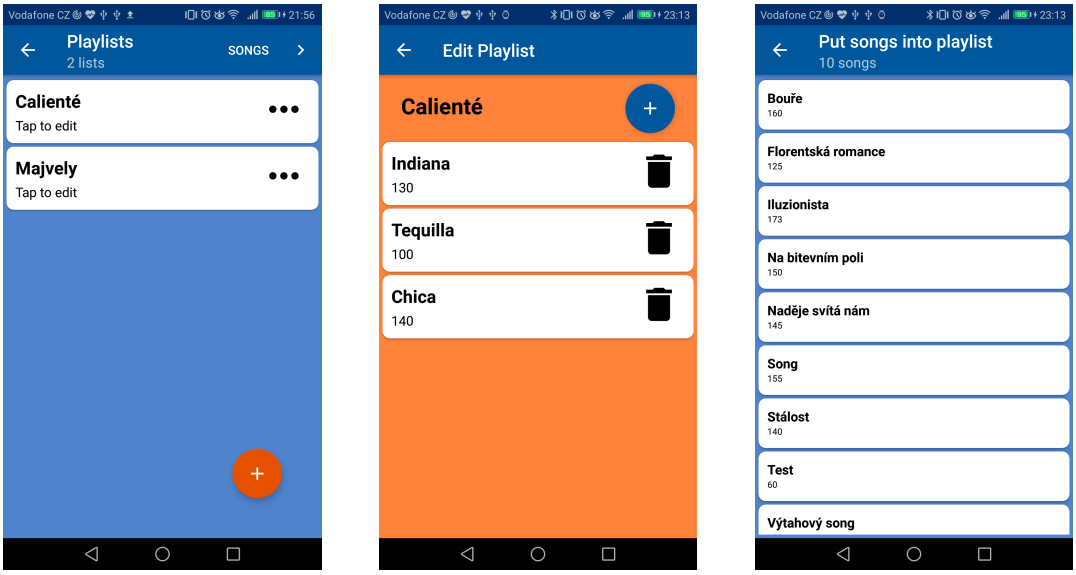


(a) Seznam s písničkami (b) Uložení písničky

Obrázek 5.3: Aplikace MyMetronome

Seznam s playlisty

Seznam playlistů 5.4a obsahuje položky s názvy jednotlivých playlistů. Tyto položky lze upravovat nebo mazat pomocí kliknutí na možnost *další* úplně vpravo. Pro úpravu playlistu stačí kliknout na příslušnou položku. Poté se otevře náhled na playlist 5.4b, v rámci kterého lze jednoduše měnit pořadí písniček pomocí gesta *drag and drop*. Tyto písničky lze také jednoduše odstranit tlačítkem s logem popelnice. Pokud chce uživatel přidat písničku do playlistu, klikne na plovoucí tlačítko a otevře se mu nové okno 5.4c se seznamem písniček, které tento playlist ještě neobsahuje.



(a) Seznam s playlisty (b) Úprava playlistu (c) Vkládání do playlistu

Obrázek 5.4: Aplikace MyMetronome

Implementace

Jelikož jsem byl před implementací ve vývoji Android aplikací úplný začátečník, jako úvod do problematiky jsem mimo jiné použil knihu The Big Nerd Ranch Guide [38]. Další užitečné zdroje mi poskytla stránka Android Developers [2] určená pro vývoj Android aplikací.

Hlavní aktivita

Hlavní část metronomu tvoří aktivita obsahující dva fragmenty. Každý fragment tvoří v podstatě sub-aktivitu a umožňuje mít vlastní design pomocí svého XML layoutu. Tato část byla modelována pomocí oboustranného databindingu a architektury MVVM, která byla blíže popsána v části 4.3. Zároveň jsem v tomto úseku částečně vycházel z práce mého kolegy Bc. Aleše Ondráčka [37]. Všechny takovéto části jsou v kódu řádně označeny.

První fragment 5.1a představuje hlavní okno aplikace metronomu. Pro tento fragment byla vytvořena třída `MetronomeModel` tvořící Model. Tato třída si uchovává informace o aktuálním tempu. Zároveň `MetronomeModel` rozšiřuje třídu *BaseObservable*, která v případě změn dat v rámci `MetronomeModel` upozorní View, který je tvořen XML layoutem `fragment_metronome.xml`. Tuto třídu a XML layout spojuje ViewModel třída `MetronomeViewModel` poskytující funkce pro základní ovládání metronomu.

Druhý fragment 5.1b je tvořen třídou `PlayerViewModel` představující ViewModel a XML layoutem `fragment_player.xml` tvořícím View. V rámci tohoto fragmentu nedochází k databindingu ani není potřeba uchovávat data na pozadí, takže třída tvořící Model nebyla implementována.

Odvození délky intervalu z BPM

Pro účely funkce metronomu bylo potřeba určit délku v milisekundách mezi jednotlivými dobami přehrávání metronomu, která je vždy určena na základě tohoto vztahu:

$$interval = 60000/BPM$$

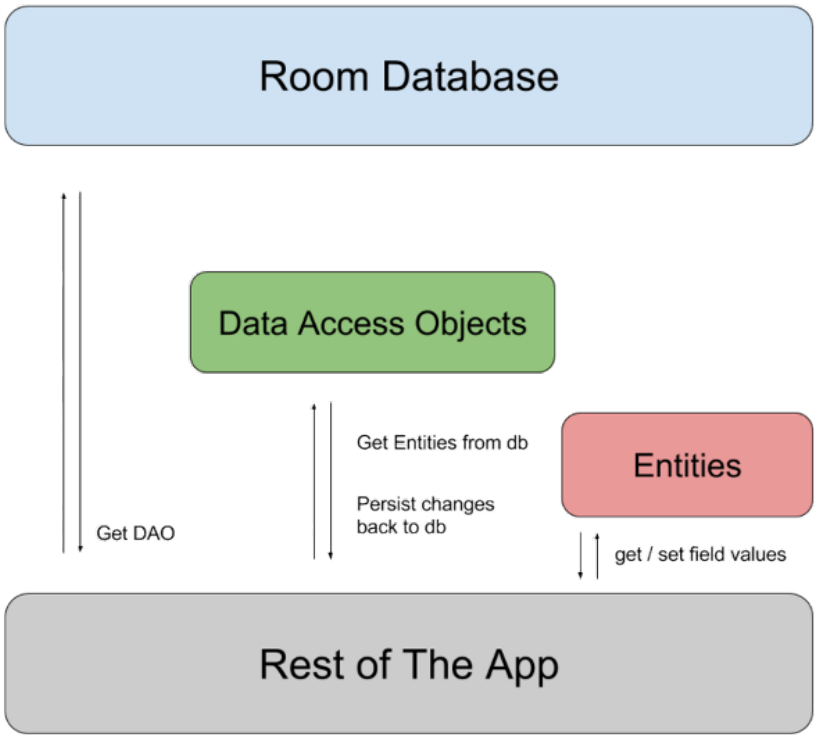
Seznam písniček a playlistů

Pro účely ukládání dat do seznamu písniček a playlistů bylo potřeba využít databáze. Android pro implementaci databáze silně doporučuje použít *Room* tvořící abstraktní vrstvu nad SQLite.

Room architektura 5.5 se skládá z následujících vrstev:

- **Database** – obsahuje databázové servery tvořící hlavní přístupový bod k perzistentním datům
- **DAO** – obsahuje metody používané k přístupu k databázi
- **Entities** – reprezentuje tabulku v databázi

Aplikace využívá *Room* databázi k přístupu k DAO. Následně DAO slouží k přístupu k entitám z databáze a následnému uložení změn těchto entit zpět do databáze. Nakonec aplikace používá entity k získání a zapsání hodnot, které odpovídají příslušným sloupcům v tabulkách databáze.



Obrázek 5.5: Room architektura. Převzato z [25].

Pro seznam písniček a seznam playlistů byla vytvořena samostatná databáze. Každá písnička v rámci databáze obsahuje unikátní identifikátor, název písničky a číselné tempo. Každý playlist obsahuje unikátní identifikátor, název playlistu a seznam s obsahem písniček. Při běhu aplikace lze přistupovat k datům v databázi pomocí přístupu k její instanci, která se vytvoří pomocí příkazu `Room.databaseBuilder()`.

Seznam písniček tvoří samostatnou aktivitu `Songlist`, která při své inicializaci během metody *onCreate* přistupuje k instanci databáze `SongDatabase` a data si ukládá do seznamu. Jako je znázorněno na obrázku 5.3a, seznam písniček je zobrazován uživateli pomocí widgetu *RecyclerView*. Pro správnou prezentaci dat widgetu je potřeba vytvořit adapatér zajišťující správné zobrazení a funkci jednotlivých položek, který představuje třída `SongAdapter`. Dále je důležité vytvořit *LayoutManager*, který se stará o správné umístění jednotlivých položek. To je implementováno následujícím způsobem:

```
1   // pristup k instanci databaze
2   SongDatabase db = SongDatabase.getInstance(getApplicationContext());
3
4   // ulozeni dat z databaze do seznamu
5   ArrayList<Song> list = (ArrayList<Song>) db.songDao().getAllSongs();
6
7   // nastaveni layout manager
8   mLayoutManager = new LinearLayoutManager(this);
9   mRecycleView.setLayoutManager(mLayoutManager);
10
11  // nastaveni adapteru
12  mAdapter = new SongAdapter(list,db,getApplicationContext());
13  mRecycleView.setAdapter(mAdapter);
```

Výpis 5.1: Úsek kódu z funkce `onCreate` aktivity `Songlist`

Obdobným způsobem byla pro účely seznamu playlistů vytvořena aktivita `Playlist`, jež je znázorněna na obrázku 5.4a. Ta přistupuje během inicializace k databázi `PlaylistDatabase` a prezentuje data pomocí widgetu *RecyclerView*. Kromě toho byla implementována aktivita `CurrentPlaylist`, která je na obrázku 5.4b, jejíž data tvoří písničky v playlistu a jsou opět reprezentovány položkami ve widgetu *RecyclerView*. V rámci této aktivity lze i měnit pořadí položek, a to pomocí operace *drag and drop*. Poslední přidruženou aktivitou s playlisty je `ChooseSongFromDatabase` na obrázku 5.4c. Ta obsahuje položky s písničkami, které je možné vložit do příslušného playlistu. Poté co je položka jednou zvolena, je odstraněna z výběru a není již možné ji do příslušného playlistu přidat. Výsledkem je, že jeden playlist může obsahovat danou písničku pouze jedenkrát.

Nastavení aplikace

Pro aktivitu nastavení aplikace `SettingsActivity` 5.1c jsem využil doporučenou knihovnu *Preference Library*. Ta spravuje uživatelské rozhraní aktivity a změny ukládá do interního úložiště. Zároveň ctí *Material Design* a umožňuje tak konzistentnost v UI napříč zařízeními a verzemi Androidu. Kromě `SettingsActivity` bylo potřeba vytvořit XML layout `settings_preferences.xml`, který definuje vzhled aktivity, a fragment `SettingsFragment` sloužící k propojení `settings_preferences.xml` a `SettingsActivity`. K uložení dat aktuálního nastavení se využívá rozhraní *SharedPreferences*, které se hodí zejména k ukládání

malých množství dat, a to ve formátu klíč-hodnota. Tato data se ukládají jako XML do interního úložiště aplikace.

Jedním z možných nastavení aplikace je volba zvuku metronomu. Během inicializace třídy `MetronomeViewModel` při spuštění aplikace nebo během vytvoření hlavní aktivity se načte hodnota z paměti na základě klíčového slova `"sound"`. Následně se zavolá funkce `setSound()` třídy `MetronomeViewModel` s hodnotou, která získá z klíče, a nastaví se příslušný zvuk metronomu.

Metronom disponuje nastavením z pěti různých zvuků. Použité zvuky metronomu byly staženy z databáze audio nahrávek Freesound [3] a následně upraveny pomocí programu Audacity [4] na délku do 50ms. Autory zvuků jsem zmínil v rámci zdrojového kódu.

Další možností v nastavení je zrychlování tempa, které je také implementováno pomocí rozhraní *SharedPreferences*.

Tréninkový mód

Součástí řešení je i tréninkový mód 5.2a, který slouží především ke cvičení přesnosti a je primárně určen uživatelům hrajícím na bicí nástroje. Naprogramován byl jako samostatná aktivita `Trainer`. Další součást aktivity jsou třídy `MetronomeView` a `VisualiserView`, které představují základní komponenty reprezentující UI nazývající se jako *View*. `MetronomeView` vykresluje na každý úder metronomu bílou čáru představující přesnou dobu. `VisualiserView` vykresluje oranžovou barvou graf znázorňující údery uživatele snímané z vestavěného mikrofonu. Pro kontrolu správného úderu uživatele na dobu metronomu je tu třída `Intersection`, která vyhodnocuje, zda došlo k průniku obou grafů. Dojde-li k průniku, je uživateli přičten správný úder. Po odstartování tréninkového módu tlačítkem *Start* začne odpočet pěti dob a následně se již snímá přesnost úderů. Po ukončení tréninkového módu kliknutím na tlačítko *Stop* se uživateli zobrazí počet přesných úderů vůči počtu všech úderů. Z těchto údajů se vypočte jeho procentuální úspěšnost.

5.2 Výsledné řešení – aplikace pro chytré hodinky

Následující část bude věnována grafickému uživatelskému rozhraní jednotlivých oken aplikace a následnému popisu implementace metronomu pro chytré hodinky. Okna aplikací budu nazývat stránkami.

Grafické uživatelské rozhraní

Vzhled aplikace pro hodinky do značné míry vychází z výše uvedeného návrhu. Cílí na jednoduché, funkční a intuitivní řešení.

Při implementaci uživatelského rozhraní jsem využil projekt *Wearable Circular UI*, který představuje rozšíření frameworku Xamarin.Forms. Jde o projekt, který má za účel zlepšit vývoj aplikací pro chytré hodinky s OS Tizen, aby byl co nejjednodušší a nejefektivnější. Poskytuje množství komponent, které lze využít v rámci aplikace [32].

Použité ikony byly staženy z internetové stránky Flaticon [5] pod otevřenou licencí *Free license (with attribution)* a její autoři byli řádně zmíněni v rámci zdrojového kódu. Některé ikony byly následně upraveny pomocí online editoru Pixlr [6] do výsledné podoby.

Ikona aplikace a úvodní obrazovka

Ikona aplikace a zároveň tzv. *splash screen* 5.6a (úvodní obrazovka) zobrazená na několik málo sekund při zapnutí aplikace je tvořena hlavními barvami, které metronom využívá a vzhledově připomíná stránku s metronomem. Navíc je doplněna o iniciály názvu aplikace MM – MyMetronome.

Hlavní stránka

Hlavní stránka aplikace 5.6b zůstala vzhledově stejná jako v návrhu. Nové jsou barvy a ikony možností výběru.

Stránka s metronomem

Stránka s metronomem 5.6c si zachovala své dva hlavní prvky, a to tlačítko pro spuštění a vypnutí metronomu a číslo zobrazující rychlost tempa. Pomocí otočné lunety lze toto tempo nastavit. Vzhledově stránka připomíná otočný seekbar hlavního okna metronomu pro chytrý telefon.

(a) Ikona aplikace a úvodní obrazovka

(b) Hlavní stránka

(c) Stránka s metronomem

Obrázek 5.6: Aplikace MyMetronome

Seznam s playlisty

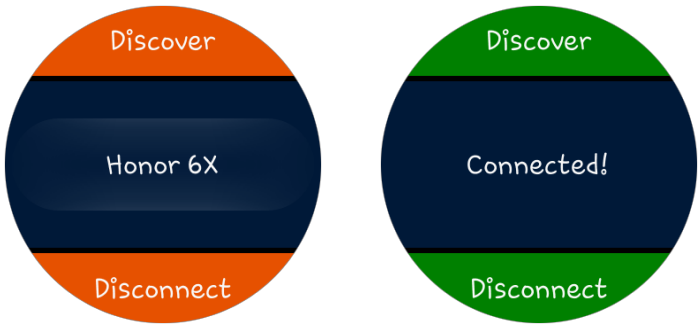
Seznam playlistů 5.7a umožňuje jednoduchý výběr na základě zvoleného názvu playlistu. Poté se uživateli otevře okno s názvem první písničky v playlistu 5.7b a příslušným tempem. Pomocí otočné lunety může listovat mezi písničkami daného playlistu.

(a) Seznam s playlisty (b) Přehrávání playlistu

Obrázek 5.7: Aplikace MyMetronome

Bluetooth spojení

Okno pro Bluetooth spojení 5.8a disponuje dvěma tlačítky. Tlačítko *Discover* umožňuje nalézt dostupná Bluetooth zařízení. Následně se tato zařízení zobrazí v seznamu a kliknutím dojde k pokusu o spojení. Pokud dojde ke spojení 5.8b, změní se barva tlačítek na zelenou a uprostřed se objeví text *Connected!*. Pomocí tlačítka *Disconnect* lze spojení přerušit.



(a) Nastavení spojení (b) Spojení navázáno

Obrázek 5.8: Aplikace MyMetronome

Implementace

Během implementace aplikace pro chytré hodinky jsem se s vývojem aplikací pro Tizen setkal poprvé. Nejdříve jsem chtěl vytvořit nativní aplikaci v programovacím jazyce C/C++, protože s těmito jazyky mám velkou zkušenost. Tyto aplikace však umožňují pouze funkcionální programování. Proto jsem zvolil značně jednodušší cestu prostřednictvím vývoje .NET Tizen aplikací s využitím frameworku Xamarin.Forms umožňující objektově orientované programování a oddělení vzhledu od funkcionality aplikace. Programovacím jazykem je však C#, se kterým jsem se do té doby nesetkal.

Pro počáteční orientaci mi velmi dobře posloužila možnost nahlédnout na ukázkově vytvořené aplikace od vývojářů Samsungu na jejich GitHub stránce [24]. Byly to především aplikace WavPlayer a SQLite.NET.Sample, které mi následně byly inspirací pro vytvoření aplikace vlastní. Využité části z příkladů aplikací byly řádně citovány v rámci zdrojového kódu.

Hlavní stránka

Po spuštění aplikace se zobrazí první stránka 5.6b, která představuje seznam možností metronomu. V případě kliknutí na jednu z možností se otevře nová stránka tak, že se vloží na vrchol modálního zásobníku příkazem `Navigation.PushModalAsync()`. V případě návratu se odstraní z modálního zásobníku a zobrazí se stránka předchozí. Tímto způsobem je zajištěn přechod mezi jednotlivými stránkami aplikace.

Stránka s metronomem

Stránka s metronomem 5.6c byla naprogramována pomocí oboustranného databindingu a architektury MVVM blíže popsané v kapitole 4.3. Třída `MetronomeModel.cs` představuje v této architektuře Model. Uchovává si informaci o nastaveném tempu po celou dobu běhu aplikace a je inicializována v rámci třídy `MetronomeViewModel.cs`. Tato třída tvoří ViewModel a zajišťuje funkce pro správný běh metronomu. K vytvoření instance této třídy dojde již při spuštění aplikace. Tato instance je následně předávána jako parametr jednotlivým stránkám, které k ní poté mohou přistoupit. View reprezentuje třída `Metronome.xaml.cs`, jejíž vzhled definuje soubor XAML `Metronome.xaml`.

Ke správnému vytvoření oboustranného databindingu je třeba nejdříve v rámci příslušného souboru XAML definujícího vzhled stránky nastavit možnost oboustranného databindingu u příslušného atributu pomocí textu `Mode=TwoWay`. Následně je třeba implementovat rozhraní INotifyPropertyChanged. V případě změny hodnoty atributu se provede volání `OnPropertyChanged()` oznamující změnu souboru XAML.

Playlisty

Playlisty lze v rámci aplikace pro hodinky přehrávat pouze po jejich přenosu z aplikace v telefonu do hodinek. Pro účely uložení přenesených playlistů byla potřeba implementace SQLite databáze. Ta je součástí třídy `PlayerViewModel.cs`. Jelikož verze databáze *SQLitePCLRaw.provider.sqlite3.netstandard11* z balíčku *NuGet* použitelná pro účely .NET Tizen aplikací neumožňuje ukládat složené datové typy, jako je seznam, bylo potřeba písničky zasílat pomocí řetězců s definovanými pravidly pro zpracování. Tato pravidla a detailnější způsob přenosu dat mezi zařízeními budou popsány v následující kapitole.

5.3 Bluetooth spojení

Pro účely přenášení playlistů z telefonu do hodinek bylo využito spojení pomocí technologie Bluetooth.

Bluetooth Low Energy (BLE)

Původně jsem chtěl spojení provést pomocí *Bluetooth Low Energy (BLE)*, který je podporovaný na hodinkách s OS Tizen od verze 2.3.1 a na Android zařízeních od Android API verze 18. Zajišťuje totiž přenos dat při využití malého množství energie a není tak náročný na baterii zařízení jako klasický Bluetooth přenos známý jako *Bluetooth Classic*. Protokol *Attribute Protocol (ATT)* zajišťující *BLE* přenos je optimalizován pro výměnu malého množství dat. Používá se například při párování chytrého náramku s chytrým telefonem. Při tomto spojení náramek představuje *General Attribute Profile (GATT) server* a telefon

představuje *GATT klienta*. Jelikož .NET Tizen aplikace umožňují pouze roli *GATT klienta*, musel jsem implementovat aplikaci pro mobil v roli *GATT serveru*. Tam jsem se ale setkal s problémem, protože ne všechna zařízení s OS Android umožňují funkci jako *GATT server* [11]. Jelikož jsem se snažil vytvořit aplikaci, která bude použitelná pro jakékoliv Android zařízení s verzí API 23 a vyšší, musel jsem zvolit jiné řešení.

Bluetooth Classic

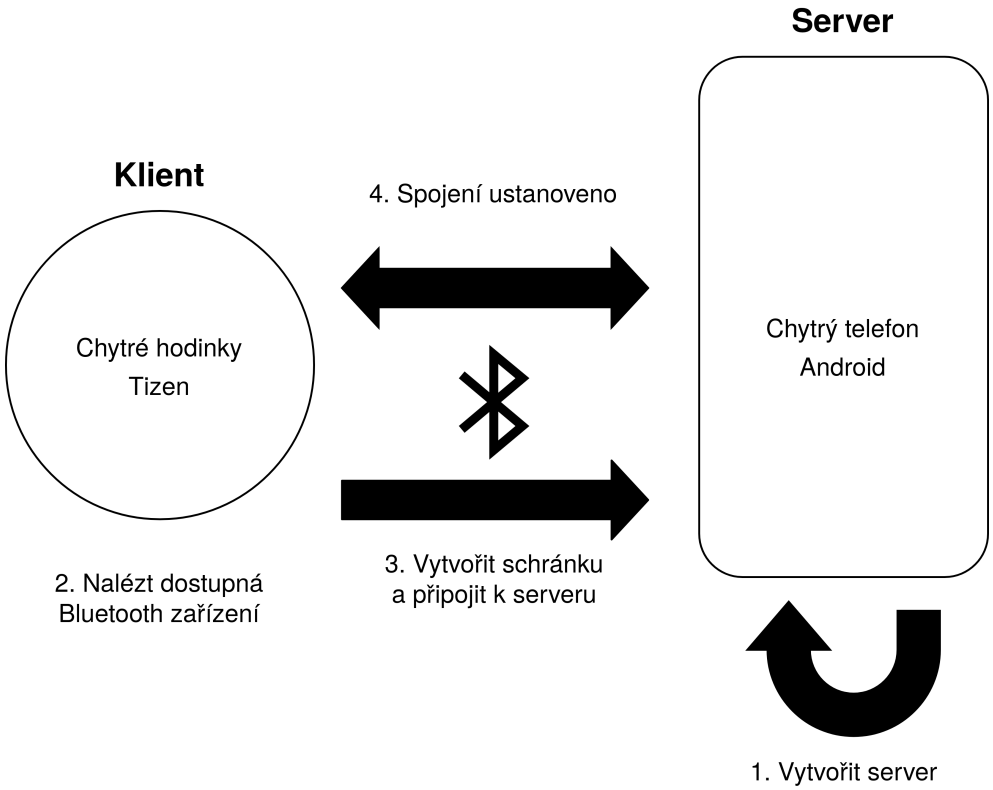
Zvolil jsem tedy řešení propojení a následného přenášení dat přes *Bluetooth Classic*, které je demonstrováno na obrázku 5.9. Toto spojení připomíná komunikaci klient-server pomocí schránek. Potřebné informace ohledně vytvoření *Bluetooth Classic* spojení jsem čerpal pro mobilní část ze stránek Android Developers [12] a pro část s hodinkami na stránkách Tizen Docs [10]. Pro vytvoření spojení je potřeba, aby jedno zařízení fungovalo jako server a druhé jako klient. Zvolil jsem řešení, kdy mobilní aplikace představuje server a aplikace pro hodinky představuje klientskou stranu.

Server

Server tedy nejdříve spustí nové vlákno `AcceptThread()`, v kterém vytvoří schránku s unikátním identifikátorem Universally Unique Identifier (UUID), který představuje 128 bitové číslo. Následně voláním funkce `accept()` v tomto vlákně vyčkává na požadavek na připojení klientem se stejným UUID. Poté co vlákno přijme požadavek, vytvoří schránku `BluetoothSocket` a ukončí svou činnost. Následně je vytvořeno nové vlákno `ConnectedThread()`, které lze využít pro zasílání a přijímání dat – `OutputStream` a `InputStream`.

Klient

Z klientského pohledu je potřeba vytvořit požadavek na připojení k serveru. Nejdříve musí hodinky najít server pomocí Bluetooth vyhledávání a následně zaslat požadavek pomocí funkce `CreateSocket()` se stejným unikátním identifikátorem UUID, jako má server. Pokud je schránka vytvořena, tak se pomocí metody `Connect()` připojíme k serveru. Následně lze zasílat a přijímat data jako bajty.



Obrázek 5.9: Jednoduché schéma demonstrující vytvoření Bluetooth komunikace mezi telefonem a hodinkami

Pro účely přenášení playlistů jsou data z playlistů kódována a následně zasílána jako bajty. K tomu slouží funkce `sendData()` v rámci třídy `BluetoothActivity`. Ta projde seznam playlistů a poté postupně kóduje data následujícím způsobem. Každý playlist je označen na začátku písmenem P, potom následuje symbol * a název playlistu. Tento řetězec je ukončen symbolem #. Výsledné kódování playlistu může vypadat například takto:

$$P * Chinaski\#$$

Písničky playlistu mají na začátku písmeno S, potom následuje symbol * a pokračuje název písničky a tempo oddělené také symbolem *. Řetězec je ukončen symbolem #. Kódování písničky může vypadat například takto:

$$S * Klara * 150\#$$

Aplikace v hodinkách tyto zprávy dekóduje a následně z nich vytvoří playlisty s písničkami, které uloží do databáze.

Kapitola 6

Testování

Cílem testování obou aplikací bylo ověřit jejich nezávislou funkci i vzájemnou výměnu dat. Obě aplikace byly během implementace průběžně testovány a vylepšovány.

V průběhu vytváření aplikace pro chytrý telefon jsem kontroloval správnou rychlost úderů metronomu. Zjistil jsem, že ve vyšším tempu než 140 BPM je přehrávaný zvuk občas velmi lehce opožděn. Při synchronním běhu s metronomem od společnosti Soundbrenner 3.1.4 jsem ale nezaznamenal, že by docházelo ke zrychlování nebo zpomalování tempa, pouze ke zpoždění zvuku. Nejdříve jsem se tento problém snažil odstranit za pomoci programu Audacity [4], a to změnou velikostí přehrávaných zvuků metronomu na délku maximálně 50 ms. To pomohlo, ale nakonec jsem analyzoval, že problém je způsoben přehráváním zvuku pomocí knihovny *Soundpool* [29]. Nepomohlo ani využití jiných tříd *Timer*, *Thread* nebo knihovny *MediaPlayer*. Výše zmíněný problém jsem se pokusil maximálně zredukovat tak, aby měl co nejmenší vliv na výslednou přesnou funkci metronomu.

Při implementaci aplikace pro chytré hodinky jsem zjistil, že jsou vibrace rychlejší, než by v daném tempu měly být. Musel jsem proto změnit funkci pro výpočet intervalu mezi jednotlivými vibracemi na následující funkci:

$$interval = 59000/BPM$$

6.1 Zkoušky s kapelou

V rámci testování výsledné aplikace pro chytrý telefon jsem si vytvořil 2 funkční playlisty, jeden pro každou z kapel, v které hraji. Do těchto playlistů jsem vložil písničky s odpovídajícím tempem. Během hudební zkoušky jsem následně testoval funkčnost metronomu se sluchátkami. Kromě testování mobilní aplikace jsem na zkoušce testoval i aplikaci pro chytré hodinky. Zmíněné playlisty jsem přeposlal za pomocí Bluetooth komunikace do aplikace v hodinkách. Sám jsem metronom během hraní zkoušel a zároveň jsem požádal ostatní členy kapely, aby si hru s hodinkami vyzkoušeli. Metronom v hodinkách byl testován při hře na následující hudební nástroje:

- Bicí nástroje
- Cajón
- Elektrická kytara
- Ukulele

- Klavír

Následně jsme s ostatními členy kapely vyhodnotili, že nejlepším pomocníkem pro přesnou hru byl při hře na klavír. To je dáno do velké míry tím, že při hře na klavír hráč příliš neotáčí zápěstím ani nedělá příliš dlouhé a prudké pohyby rukou. Během hraní na strunné nástroje se jeho funkce také dobře osvědčila. Nejlepší možnost je, pokud hráč umístí hodinky na zápěstí ruky, kterou mění akordy. V případě hraní s hodinkami na bicí nástroje nebo cajón je při vyšším tempu (160 BPM a více) už poměrně těžké rozpoznat doby udávané vibracemi. Zároveň by hodinkám z pohledu metronomu prospěla možnost ještě silnějších vibrací, ačkoliv jsem použil nejsilnější možné vibrace, které hodinky poskytují.

Je potřeba sdělit, že na udávání tempa pomocí vibrací je nutné si zvyknout, obvzlášť pokud hráč není zvyklý používat běžný metronom. I pro hráče zvyklého na hru s metronomem jde o odlišný způsob udávání tempa. Jestliže si však na vibrace místo zvuku hráč zvykne, umožní mu to citlivěji reagovat na hlasitost svého hraní vůči ostatním v kapele, protože má možnost hrát bez sluchátek.

6.2 Testování tréninkového módu

Tréninkový mód byl během implementace průběžně testován, a to nejdříve s použitím na elektronických bicích a následně na akustické bicí soupravě. Tempo metronomu v tréninkovém módu je 150 BPM. Náročnost kladená na přesnost úderu uživatele není příliš velká. Zároveň však správný úder musí splnit tu podmínku, že je proveden během necelých 40 ms, kdy dojde k přehrání a vykreslení čáry představující úder metronomu. Celkový časový interval mezi dvěma dobami metronomu přitom představuje necelých 400 ms. Běžný úder snímaný z elektronických bicích představoval v průměru zvuk o délce okolo 120 ms. Znamená to, že uživatel může udeřit o několik desítek ms před přehráním doby metronomu a úder mu bude uznán jako správný. Tento úder však musí být silnější než 40 dB (tato hodnota byla naměřena pomocí mobilní aplikace Sound Meter [26]), aby nedocházelo k vyhodnocování slabých zvuků nebo dozvuků z úderů.

Při testování na akustické bicí soupravě to bylo komplikovanější. Zvuk úderu na buben snímaného mikrofonem byl velmi silný a delší než 400 ms. V tomto případě bylo trefení správného úderu značně jednodušší a uživatel mohl udeřit o téměř 150 ms před dobou danou metronomem. Pro účely přesnějšího snímání úderů tak bylo potřeba buben lehce zatlumit.

Další test proběhl při úderech s bubenickými paličkami o stůl. Zjistil jsem, že uživatel může trénovat i tímto způsobem. Jelikož je délka zvuku úderu paličky velmi krátká, tento fakt umožňuje lépe vyhodnotit přesnost úderu vůči metronomu.

Pokusil jsem se také v rámci implementace vytvořit tréninkový režim i pro rychlejší tempo než již zmíněných 150 BPM, ale přesnost těchto úderů už by se jen velmi těžko vyhodnocovala, protože čím je tempo vyšší, tím je časový interval mezi dobami metronomu kratší. To znamená, že bylo potřeba vyhodnocovat údery za kratší čas než 40 ms. Tam ale docházelo k chybám způsobeným příliš krátkou dobou mezi snímáním zvuku mikrofonem, která nesmí být kratší než 35 ms.

6.3 Testování uživatelem

Po dokončení aplikací byly ještě provedeny testy pěti koncovými uživateli. Ty si braly za cíl zjistit, zda je uživatelské prostředí dostatečně intuitivní a nedochází k nejasnostem při

využívání jednotlivých funkcí obou aplikací. Uživatel měl mimo jiné za úkol vytvořit v rámci aplikace v telefonu několik písniček, následně je vložit do playlistu, změnit jejich pořadí a přehrát. Následně bylo úkolem ustanovit Bluetooth spojení, playlist přenést do aplikace v chytrých hodinkách a přehrát ho v rámci hodinek. V rámci Bluetooth aktivity 5.2b dělalo problém většině uživatelům zjistit, jak navázat spojení. Proto jsem přidal možnost tlačítka pro pomoc, kde je v jednotlivých krocích vysvětleno, jak obě zařízení propojit. Jinak se jim prostředí jevilo jako dostatečně intuitivní a ovládání jim bylo jasné.

Testy aplikací byly provedeny na následujících zařízeních:

Název zařízení	Verze Androidu
Google Pixel XL	10.0
Honor 6X	7.0 (Nougat)
Huawei Nova 5T	10.0
Sony XZ2 Compact	10.0
Honor 8X	10.0
Honor 9 Lite	8.0 (Oreo)

Tabulka 6.1: Testovaná mobilní zařízení – Android

Název zařízení	Verze Tizenu
Samsung Gear Sport	3.0

Tabulka 6.2: Testovaná zařízení chytrých hodinek – Tizen

Kapitola 7

Závěr

Cílem této bakalářské práce bylo vytvořit aplikace pro chytrý telefon s operačním systémem Android a pro chytré hodinky s operačním systémem Tizen. Obě aplikace měly umožnit asynchronní fungování metronomu a navázání Bluetooth spojení pro účely přenosu playlistů z telefonu do hodinek. Metronom v hodinkách by měl zdůrazňovat tempo pomocí vibrací. Jednou z plánovaných funkcí aplikace v telefonu bylo i vyhodnocování přesnosti úderů bubeníka v daném přehrávaném tempu. Jedním z hlavních záměrů implementace aplikace pro chytré hodinky bylo, aby byl uživatel schopen metronom využít pro osobní účely, při hudebních zkouškách, ale i při vystupování na koncertech bez nutnosti využití mobilního telefonu. Playlist v hodinkách by mu měl poskytnout komfort v podobě jednoduchého a rychlého přepínání mezi tempy písniček.

Všechny výše zmíněné požadavky se podařilo úspěšně naimplementovat a ověřit testováním. Po testování nejsem plně spokojen se silou vibrací hodinek, ačkoliv jsem použil nejsilnější možné vibrace, které hodinky nabízejí.

V rámci práce jsem se seznámil s principy tvorby mobilních aplikací pro Android a s tvorbou aplikací pro chytré hodinky s OS Tizen. Také jsem v práci rozebral existující řešení pro obě platformy a zmínil jejich nedostatky.

Jednou z praktických funkcí, kterou plánuji v budoucnu přidat do aplikace pro chytré hodinky, je přehrávání zvuku metronomu do bezdrátových Bluetooth sluchátek. V tomto případě by se však uživatel musel rozhodnout mezi udáváním dob vibracemi nebo zvukem. Vibrace a zvuk by se totiž velmi těžko synchronizovaly z důvodu zpoždění zvuku při používání bezdrátových Bluetooth sluchátek.

Další možné rozšíření by mohlo být vytvořeno v rámci tréninkového režimu. Mohl by být rozšířen o více přehrávaných temp pro cvičení. Také se aktuálně uživateli zobrazují statistiky ve formě přesných úderů vůči všem úderům a následně procentuální úspěšnost. Myslím si, že další rozšíření by mohlo cílit na zvýšení motivace uživatele – například zisk bodů za správné údery doplněný statistikou bodů z minulých cvičení.

Aplikace plánuji umístit na Google Play a Galaxy Store, aby byly k dispozici i ostatním uživatelům nejen z řad bubeníků. Metronom pro hodinky se nyní nachází v beta testování a pro jeho zpřístupnění na Galaxy Store se čeká na zveřejnění mobilní aplikace v Google Play.

Literatura

[1] Material Design. [online]. [cit. 2020-20-04]. Dostupné z: https://material.io/.

[2] Android Developers. [online]. [cit. 2020-24-04]. Dostupné z: https://developer.android.com/.

[3] Freesound. [online]. [cit. 2020-22-04]. Dostupné z: https://developer.android.com/reference/android/media/SoundPool.

[4] Audacity. [online]. [cit. 2020-22-04]. Dostupné z: https://www.audacityteam.org/.

[5] Flaticon. [online]. [cit. 2020-22-04]. Dostupné z: https://www.flaticon.com/.

[6] Pixlr. [online]. [cit. 2020-22-04]. Dostupné z: https://pixlr.com/x/.

[7] Ableton. *Ableton Link: Connect music making apps with Ableton Live* [online]. [cit. 2020-16-04]. Dostupné z: https://www.ableton.com/en/link/.

[8] Android Developers. *Android NDK* [online]. [cit. 2020-22-04]. Dostupné z: https://developer.android.com/ndk.

[9] TechTarget. *Android Studio* [online]. [cit. 2020-10-04]. Dostupné z: https://searchmobilecomputing.techtarget.com/definition/Android-Studio.

[10] Tizen Docs. *Bluetooth* [online]. [cit. 2020-22-04]. Dostupné z: https://docs.tizen.org/application/dotnet/guides/connectivity/bluetooth#connecting-to-other-devices-using-spp.

[11] Android Developers. *Bluetooth low energy overview* [online]. [cit. 2020-22-04]. Dostupné z: https://developer.android.com/guide/topics/connectivity/bluetooth-le.

[12] Android Developers. *Bluetooth overview* [online]. [cit. 2020-22-04]. Dostupné z: https://developer.android.com/guide/topics/connectivity/bluetooth#ConnectDevices.

[13] Medium. *Collection Binding with DiffUtil and MVVM in Android* [online]. [cit. 2020-17-04]. Dostupné z: https://android.jlelse.eu/collection-binding-with-diffutil-and-mvvm-in-android-233247e23513.

[14] Material Design. *The color system* [online]. [cit. 2020-17-04]. Dostupné z: https://material.io/design/color/the-color-system.html#color-usage-and-palettes.

[15] Material Design. *Color Tool* [online]. [cit. 2020-17-04]. Dostupné z: https://material.io/resources/color/.

[16] Android Developers. *Create and run a wearable app* [online]. [cit. 2020-14-04]. Dostupné z: https://developer.android.com/training/wearables/apps/creating.

[17] Samsung Developers. *Creating Your First App* [online]. [cit. 2020-11-04]. Dostupné z: https://developer.samsung.com/galaxy-watch-develop/creating-your-first-app/overview.html.

[18] GitHub. *Croller* [online]. [cit. 2020-24-04]. Dostupné z: https://github.com/harjot-oberai/Croller.

[19] Wikimedia Commons. *File:Android-System-Architecture.svg* [online]. [cit. 2020-10-04]. Dostupné z: https://commons.wikimedia.org/wiki/File:Android-System-Architecture.svg.

[20] Medium. *How to make your first Apple Watch App* [online]. [cit. 2020-14-04]. Dostupné z: https://medium.com/swiftist/how-to-make-your-first-apple-watch-app-37289bf953b4.

[21] Android Authority. *I want to develop Android apps — What languages should I learn?* [online]. [cit. 2020-22-04]. Dostupné z: https://www.androidauthority.com/develop-android-apps-languages-learn-391008/.

[22] Galaxy Store. *Metrónome* [online]. [cit. 2020-11-04]. Dostupné z: https://galaxystore.samsung.com/geardetail/A26K9lFRcp.

[23] Dotnetportal.cz. *MVVM: Model-View-ViewModel* [online]. [cit. 2020-17-04]. Dostupné z: https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel.

[24] GitHub. *Samsung / Tizen-CSharp-Samples* [online]. [cit. 2020-22-04]. Dostupné z: https://github.com/Samsung/Tizen-CSharp-Samples/tree/master/Wearable.

[25] Android Developers. *Save data in a local database using Room* [online]. [cit. 2020-20-04]. Dostupné z: https://developer.android.com/training/data-storage/room.

[26] Google Play. *Sound Meter* [online]. [cit. 2020-24-04]. Dostupné z: https://play.google.com/store/apps/details?id=com.gamebasic.decibel&hl=cs.

[27] InceptiveMind. *Soundbrenner Core: The 4-in-1 smart music tool* [online]. [cit. 2020-16-04]. Dostupné z: https://www.inceptivemind.com/soundbrenner-core-the-4-in-1-smart-music-tool/4456/.

[28] Guitar Girl Magazine. *Soundbrenner Launches The Core, The Multipurpose Watch Made For Musicians Allowing Them To Play Better, Anytime, Anywhere.* [online]. [cit. 2020-16-04]. Dostupné z: https://guitargirlmag.com/news/namm/namm-2019/soundbrenner-launches-the-core-the-multipurpose-watch-made-for-musicians-allowing-them-to-play-better-anytime-anywhere/.

[29] Android Developers. *SoundPool* [online]. [cit. 2020-22-04]. Dostupné z: https://developer.android.com/reference/android/media/SoundPool.

[30] Android Developers. *Understand the Activity Lifecycle* [online]. [cit. 2020-10-04]. Dostupné z: https://developer.android.com/guide/components/activities/activity-lifecycle.

[31] MacRumors. *WatchOS 6* [online]. [cit. 2020-14-04]. Dostupné z: https://www.macrumors.com/roundup/watchos-6/.

[32] Tizen Docs. *Wearable Circular UI* [online]. [cit. 2020-22-04]. Dostupné z: https://docs.tizen.org/application/dotnet/guides/wcircularui/overview.

[33] Interaction Design Foundation. *What is Material Design?* [online]. [cit. 2020-20-04]. Dostupné z: https://www.interaction-design.org/literature/topics/material-design.

[34] statcounter. *Mobile Operating System Market Share Europe* [online]. Duben 2020 [cit. 2020-10-04]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[35] Gadyatskaya, O., Massacci, F. a Zhauniarovich, Y. Security in the Firefox OS and Tizen Mobile Platforms. *Computer*. Červen 2014, roč. 47, s. 57–63.

[36] Hamblen, M. Makers of Wearable Systems Woo Coders. *Computerworld*. 2014, roč. 48, č. 6, s. 2. ISSN 00104841.

[37] Ondráček, A. *Bubenický metronom pro Smartphone a Smartwatch*. Vysoké učení technické v Brně. Fakulta informačních technologií, 2019.

[38] Phillips, B. *Android programming : the Big nerd ranch guide*. Third edition. Atlanta, GA: Big nerd ranch, 2017. ISBN 978-0-13470-605-4.

[39] Tiano, J. *Learning Xcode 8*. Packt Publishing, 2016. ISBN 9781785885723.

[40] Ujbányai, M. *Programujeme pro Android*. 1. vyd. Grada Publishing, a.s., 2012. ISBN 978-80-247-3995-3.