Obchodní akademie, Vyšší odborná škola a Jazyková škola
s právem státní jazykové zkoušky Uherské Hradiště

MATURITNÍ PROJEKT

E-METRONOM

2022 Ctibor Mlýnek, 4. B

Obchodní akademie, Vyšší odborná škola
a Jazyková škola s právem státní jazykové
zkoušky Uherské Hradiště

Vnitřní předpis OA, VOŠ a JŠ, čj. 025/ORG/2020

ZADÁNÍ MATURITNÍ PRÁCE INFORMAČNÍ TECHNOLOGIE

Jméno žáka:	Ctibor Mlýnek
Téma maturitní práce:	E-Metronom
Vedoucí práce:	Mgr. Jaroslav Hodl
Oponent práce:	Ing. Bc. Martin Šimůnek
Způsob zpracování:	Žák vytvoří mobilní aplikace pro telefony s OS Android simulující rovnoměrné odklepávání rytmu hudební skladby. Hlavní funkce aplikace: • Odklepávání rytmu dle zvoleného nastavení • Možnost nastavit rytmus, jeho délku (v průběhu času se může měnit dle nastavení), přidání pomlk • Možnost zdůraznění počátku každého taktu • Uložení a načtení nastavení rytmu ze souboru či databáze • Výběr z více zvuků pro odklepávání Důraz bude kladen na uživatelskou přívětivost celé aplikace, snadné a rychlé ovládání. Součástí práce bude rovněž zpráva z testování aplikace, žák nechá otestovat minimálně 2 uživateli. Průběh vývoje aplikace bude dostupný ve veřejném repozitáři na GitHubu.

Obchodní akademie, Vyšší odborná škola
a Jazyková škola s právem státní jazykové
zkoušky Uherské Hradiště

Pokyny k odevzdání:	**Žák odevzdá práci v tištěné a elektronické podobě** • Tištěná podoba práce obsahuje uživatelskou a technickou dokumentaci. Tištěnou podobu (v kroužkové vazbě) žák odevzdá v jedné verzi na studijní oddělení školy, místnost 311. **Elektronická podoba práce obsahuje** • Dokumentaci ve formátu PDF/A • Resumé ve formátu PDF/A • Výsledný projekt, zdrojové soubory a potřebné knihovny pro spuštění projektu • Prezentaci projektu **Elektronická podoba** práce se nahrává do IS školy dle pokynů vedoucího práce nebo vedení školy. V případě, že se jedná o projekt, na kterém pracovalo více žáků, je povinnou součástí dokumentace podrobné rozdělení činností při práci na projektu.
Kritéria hodnocení:	Hodnocení se skládá z celkové kvality zpracování práce, dokumentace, z kvality prezentace při obhajobě práce, diskuse a z průběžného hodnocení žáka v rámci kontrolních dnů.
Obhajoba projektu	Obhajoba projektu se skládá ze dvou části - prezentace projektu (včetně podpůrné elektronické prezentace) a diskuse nad řešením. Celková délka obhajoby je 20 minut, délka prezentace projektu by neměla překročit 10 minut.

14. 10. 2021

.....

Datum

.....

Podpis ředitele školy

Obchodní akademie, Vyšší odborná škola
a Jazyková škola s právem státní
jazykové zkoušky Uherské Hradiště
Nádražní 22, 686 01 Uherské Hradiště
IČO: 60371731, tel.: 572 433 011

Prohlášení:

Souhlasím s tím, že s výsledky mé práce může být naloženo podle uvážení vedoucího maturitní práce a ředitele školy. V případě publikace budu uveden jako spoluautor. Prohlašuji, že jsem na celé maturitní práci pracoval samostatně a veškeré použité zdroje jsem citoval.

V Uherském Hradišti, dne 30. 3. 2022

.....

podpis absolventa

RESUMÉ

Aplikace nahrazuje klasický mechanický metronom, který rovnoměrně odklepává rytmus. Uživatel nemusí investovat větší množství peněz do mechanického metronomu, ale může mít metronom v mobilu vždy u sebe. Aplikace pomáhá všem, co hrají na hudební nástroj a chtějí se zlepšovat. E-Metronom hlídá, aby hráč nezrychloval ani nezpomaloval.

OBSAH

ÚVOD

Náplní mé maturitní práce je vytvořit mobilní aplikaci E-Metronom. Co to vlastně metronom je? Metronom je pomocník při cvičení na hudební nástroj, který pouze rovnoměrně odklepává rytmus. Většina používá mechanický metronom, který nemá žádné funkce, a tedy naše cvičení není tak dokonalé. Na trhu jsou aplikace, které pouze kopírují mechanický metronom. Většina k tomu nabízí škálu zvuků klepání. Nejedná se ale o funkci, která by nás mohla více zdokonalit. Mým cílem je vytvořit aplikaci s funkcemi, která je efektivnější než mechanický metronom. Zároveň chci vytvořit jednoduchou a přehlednou aplikaci. Aplikace bude určena pro uživatele, kteří chtějí efektivněji zdokonalit svoji hru na hudební nástroj.

Základní funkcí metronomu bude pouhé rovnoměrné odklepávání rytmu. Pokud se nám nebude nelíbit zvuk klepání, můžeme si vybrat zvuk z nabídky, který nám vyhovuje.

První vylepšenou funkcí bude vynechání dob. Opět pomocí nastavení nastavíme, kolik dob vynechá a kolik zahraje. Tímto způsobem můžeme trénovat, abychom nezrychlovali. Po dobu, kdy metronom přestane hrát, musíme udržet stejné tempo jako když hrát začne.

Jako další funkcí bude zvýraznění dob. Pokud budeme trénovat skladbu, která je na 4 doby, nastavíme, že chceme zvýraznit každou první dobu z těchto čtyř. Tímto zajistíme, že nevynecháme žádnou notu v daném taktu. V nastavení budeme vybírat mezi 3 a 4, podle toho na kolik je skladba dob nebo tuto funkci úplně vypnout.

Celé toto nastavení si bude možné si uložit. Pokud tedy budeme chtít pokračovat příště s nastavením z minula, jednoduše ho načteme a můžeme pokračovat v cvičení. Vše bude uloženo v aplikaci a my se nemusíme strachovat, že zapomeneme např. u jakého tempa jsme skončili.

Každé cvičení s metronomem může vyžadovat jiné nastavení. Vždy si můžeme vybrat, kterou funkci chceme zapnout a kterou použít nechceme.

1 UŽIVATELSKÁ DOKUMENTACE

1.1 Popis mobilní aplikace – E-Metronom

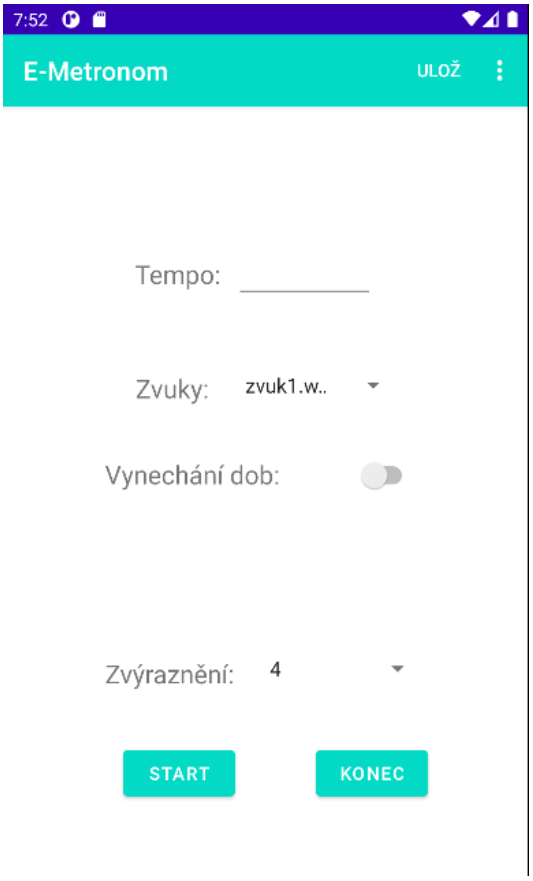
E-Metronom je mobilní aplikace, fungující na platformě Android, určená ať už pro začínající nebo pokročilé hráče na hudební nástroj.
Umožňuje rovnoměrně udržovat rytmus tak, aby uživatel nezrychloval ani nezpomaloval pomocí rovnoměrného odklepávání a vylepšených funkcí. Zvuk klepání si uživatel může vybrat z nabídky zvuků. Navíc umožňuje ukládání a načítání nastavení metronomu, aby uživatel se mohl kdykoli vrátit tam, kde skončil se cvičením.
Díky tomu, že je aplikace určena pro mobilní telefony, hráč nemusí mít s sebou mechanický metronom, ale jednoduše může kdekoli trénovat pomocí aplikace.
Vzhled aplikace byl navržen pro rychlé a jednoduché ovládání a snadnou orientaci.
Aplikace umí ukládat informace a následně načíst:

- Tempo
- Kolik zahrát
- Kolik vynechat

Prostý postup instalace je popsán dále v samostatné kapitole. Aplikace funguje na systému Android 4. 4 a vyšším.

1.2 Úvodní obrazovka

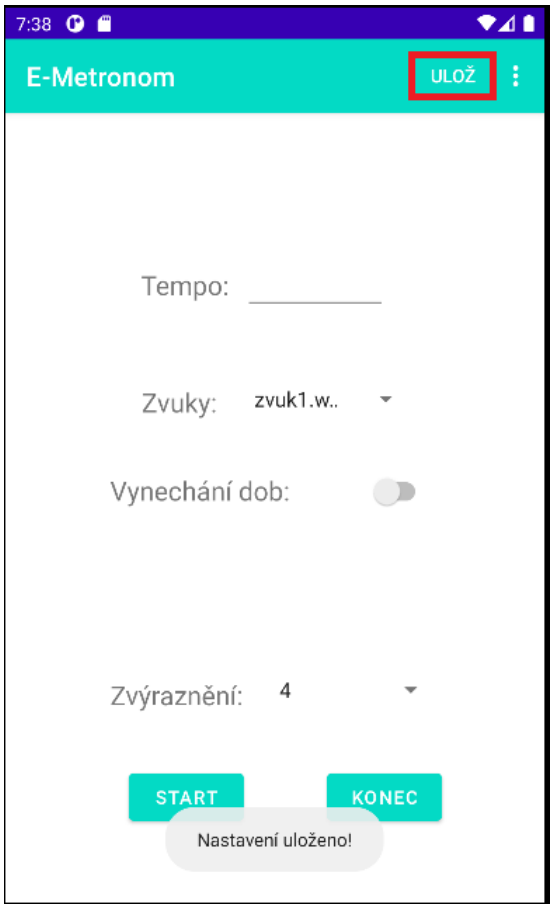
Po načtení aplikace naběhne úvodní obrazovka. Hlavní obrazovka aplikace slouží uživateli k nastavení metronomu. Tlačítko *Ulož* umožňuje uživateli si uložit nastavení metronomu. V horním rohu najdeme menu. Do pole *Tempo* zadáváme hodnotu, jak chceme, aby metronom rychle klepal. Rychlost klepání se používá od 50 do 150. Pomocí pole *Zvuk* si můžeme vybrat zvuk, kterým chceme, aby metronom odklepával rytmus. Pokud aktivujeme tlačítko *Vynechání dob*, rozbalí se nastavení, pomocí kterého můžeme aktivovat funkci „vynechání dob“. Funkce je omezená danou rychlostí, daným zvukem a funkce *Zvýraznění* musí být nastavena bez žádného zvýraznění. Jako poslední funkcí, kterou si uživatel může aktivovat zaškrtnutím pole je *Zvýraznění*. Ve spodní části pomocí tlačítek *Start* a *Konec*, spustíme klepání nebo vypneme.



Obrázek 1 Úvodní obrazovka Zdroj: [vlastní zpracování]

1.3 Ulož

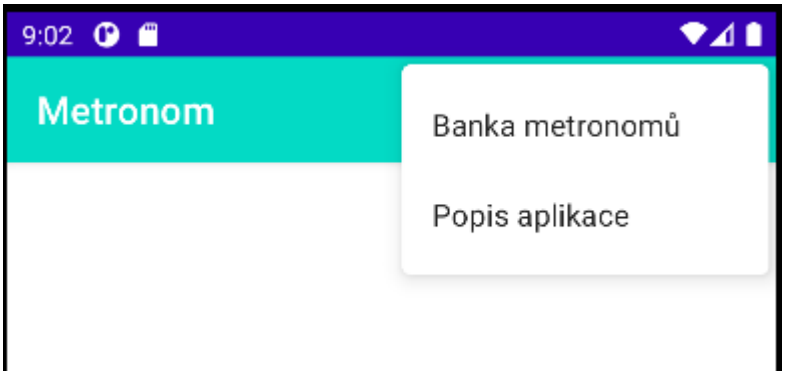
Důležitou funkcí, kterou najdeme v horní části, je funkce *Ulož.* Pomocí funkce si můžeme uložit nastavení metronomu do Banky metronomů. Pokud uložení proběhne v pořádku, budeme informováni, že nastavení bylo úspěšně uloženo.



Obrázek 2 Funkce ukládání Zdroj: [vlastní zpracování]

1.4 Menu

Po kliknutí do horního pravého rohu na menu, se nám zobrazí na výběr *Banka metronomů* a *Popis aplikace*.



Obrázek 3 Menu aplikace Zdroj: [vlastní zpracování]

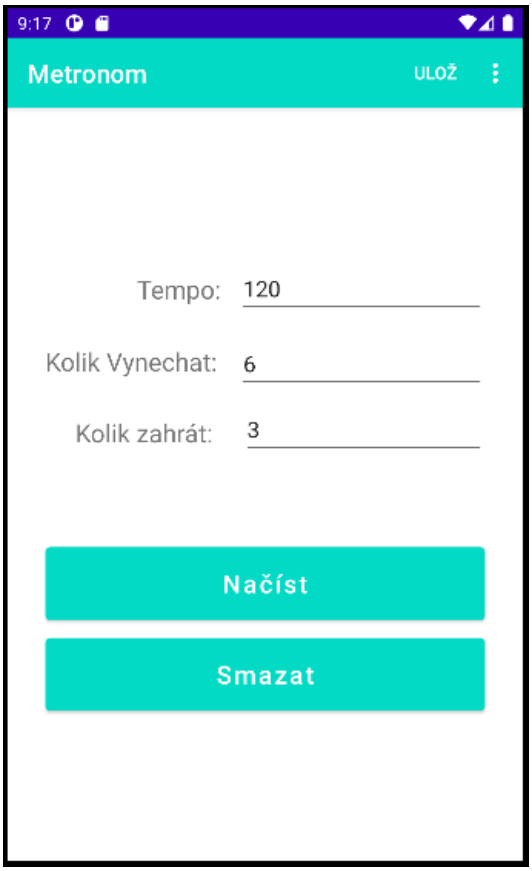
1.5 Banka metronomů

V bance metronomů se zobrazují všechna uložená nastavení metronomu, kde pomocí rychlého náhledu vidíme uložené nastavení. Jednotlivá nastavení lze rozkliknout a můžeme provádět další operace.

Obrázek 4 Banka metronomů Zdroj: [vlastní zpracování]

1.6 Konkrétní nastavení metronomu

Zde vidíme hodnoty, které jsme si uložily do banky metronomů. Pomocí tlačítka *Načíst* si můžeme data načíst na hlavní stránku a spustit metronom s tímto nastavením. Pomocí tlačítka *Smazat* toto nastavení smažeme z banky metronomů.



Obrázek 5 Určité nastavení metronomu Zdroj: [vlastní zpracování]

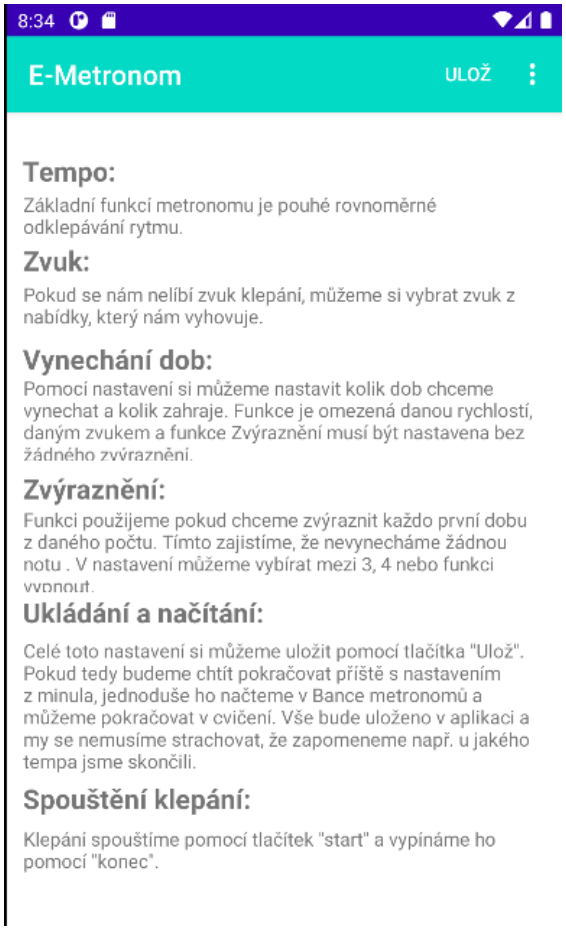
1.7 Smazání nastavení z banky metronomů

Pokud chceme nastavení z banky metronomů smazat, budeme vyzváni, zda jsme si jisti, že dané nastavení chceme odstranit. Pokud výzvu potvrdíme, nastavení se smaže. Pokud ne, nastavení se nám uchová v bance.

Obrázek 6 Smazání nastavení Zdroj: [vlastní zpracování]

1.8 Popis aplikace

Další položkou v menu je popis aplikace. V popisu aplikace se uživatel dočte, co jednotlivé funkce umí a jak aplikaci ovládat.



Obrázek 7 Popis aplikace Zdroj: [vlastní zpracování]

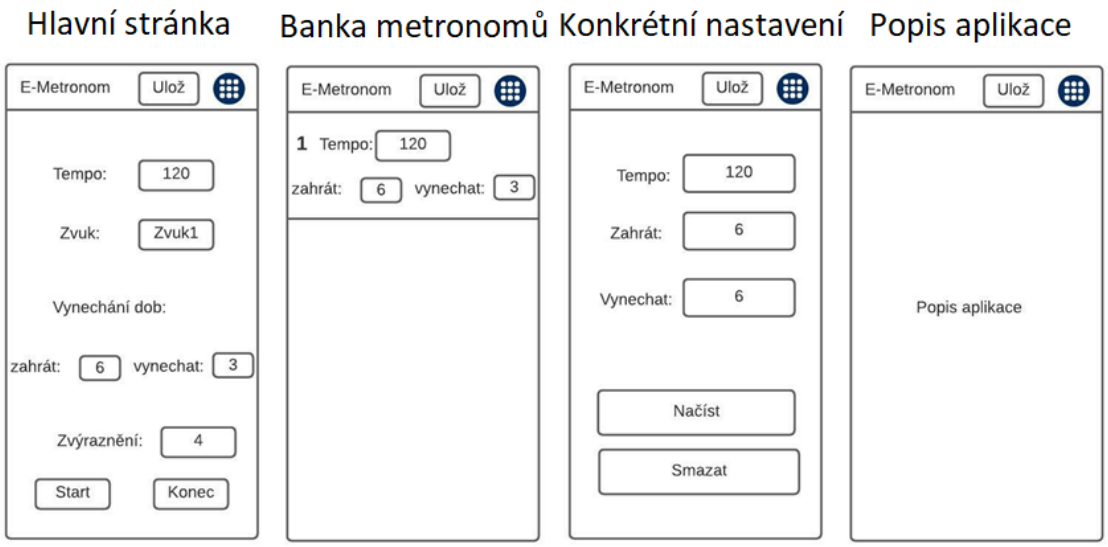
1.9 Instalace aplikace

1. Instalační soubor *metronom.apk* zkopírujte do telefonu.
2. Spusťte instalační soubor a postupujte podle pokynů.
3. Pokud vás telefon upozorní, že aplikace není ověřena, postupujte volbou pokračovat.
4. Pokud váš telefon zakazuje instalaci aplikací z neznámých zdrojů, přejděte do Nastavení >> Zabezpečení >> Povolit instalaci aplikací z neznámých zdrojů.
5. Po úspěšné instalaci naleznete zástupce v přehledu aplikací.

2 PROGRAMÁTORSKÁ DOKUMENTACE

Úkolem bylo vytvořit aplikaci, která plně nahrazuje mechanický metronom a poběží na telefonech s operačním systémem Android.
Mobilní aplikace je napsána v jazyce Java a vývojovém prostředí Android studio. Prostředí usnadňuje práci s návrhem jednotlivých obrazovek a práci s komponenty přímo určenými pro Android. Nezbytná byla také tvorba loga pro mobilní aplikaci, které bylo vytvořeno v aplikaci InkScape.
Obrazovky jsou pomocí layoutů navrženy tak, aby uživatel mohl aplikaci používat co nejsnadněji a přehledně se v ní orientovat.

Než jsem začal samostatnou aplikaci vytvářet, udělal jsem si nejdříve návrh, jak by aplikace mohla vypadat. Pomocí programu Lucidchart jsem vytvořil wireframe
aplikace.



Obrázek 8 Wireframe aplikace Zdroj: [vlastní zpracování]

2.1 Databáze

Jako databázový systém jsem zvolil SQLite. Pomocí databáze můžeme ukládat nastavení metronomu nebo uložené nastavení načíst. Databáze se skládá z jedné tabulky. V tabulce se nachází ID nastavení, tempo, kolik zahrát a kolik vynechat. Pomocí kódu jsem tabulku vytvořil a definuji co má být kam uloženo.

```
class NastaveniDatabaze extends SQLiteOpenHelper {

    private Context context;
    private static final String JMENO_DATABASE = "BankaMetronomu.db";
    private static final int VERSE_DATABASE = 1;

    private static final String TABLE_NAME = "moje_banka";
    private static final String SLOUPEC_ID = "_id";
    private static final String SLOUPEC_TEMPO = "tempo";
    private static final String SLOUPEC_VYNECHAT = "kolik_vynechat";
    private static final String SLOUPEC_ZAHRAT = "kolik_zahrat";

    NastaveniDatabaze(@Nullable Context context) {
        super(context, JMENO_DATABASE,  factory: null, VERSE_DATABASE);
        this.context = context;
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String query = "CREATE TABLE " + TABLE_NAME +
                " (" + SLOUPEC_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                SLOUPEC_TEMPO + " TEXT, " +
                SLOUPEC_VYNECHAT + " TEXT, " +
                SLOUPEC_ZAHRAT + " INTEGER);";
        db.execSQL(query);
```

Obrázek 9 Vytvoření databáze Zdroj: [vlastní zpracování]

Pomocí kódu se mohou přidat nové nastavení do databáze. Pokud uložení proběhlo v pořádku, aplikace vypíše text „Nastavení uloženo“. Pokud nastane chyba, aplikace vypíše „Nastavení se neuložilo“.

```
void pridaniNastaveni(String tempo, String vynechat, int zahrat){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues cv = new ContentValues();

    cv.put(SLOUPEC_TEMPO, tempo);
    cv.put(SLOUPEC_VYNECHAT, vynechat);
    cv.put(SLOUPEC_ZAHRAT, zahrat);
    long result = db.insert(TABLE_NAME, nullColumnHack: null, cv);
    if(result == -1){
        Toast.makeText(context, text: "Nastavení se neuložilo!", Toast.LENGTH_SHORT).show();
    }else {
        Toast.makeText(context, text: "Nastavení uloženo!", Toast.LENGTH_SHORT).show();
    }
}
```

Obrázek 10 Přidání Nastavení do databáze (vlastní zdroj)

Díky kódu se v Bance metronomů mohou vypsat z databáze všechna data.

```
Cursor zobrazVsechnaData(){
    String query = "SELECT * FROM " + TABLE_NAME;
    SQLiteDatabase db = this.getReadableDatabase();

    Cursor cursor = null;
    if(db != null){
        cursor = db.rawQuery(query, selectionArgs: null);
    }
    return cursor;
}
```

Obrázek 11 Zobrazení všech dat Zdroj: [vlastní zpracování]

Pokud se nám nelíbí nastavení, které máme uloženo v bance metronomů, můžeme ho smazat pomocí tohoto kódu. Pokud smazání proběhlo v pořádku, aplikace vypíše text „Data jsou smazána“. Pokud nastane chyba, aplikace vypíše „Smazání selhalo“.

```
    void smazaniJednohoNastaveni(String row_id){
        SQLiteDatabase db = this.getWritableDatabase();
        long result = db.delete(TABLE_NAME,  whereClause: "_id=?", new String[]{row_id});
        if(result == -1){
            Toast.makeText(context,  text: "Smazání selhalo!", Toast.LENGTH_SHORT).show();
        }else{
            Toast.makeText(context,  text: "Data jsou smazány!", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Obrázek 12 Smazání Nastavení Zdroj: [vlastní zpracování]

2.2 Třídy

Projekt je složen z 5 tříd:

- MainActivity
- NacitaniNastaveni
- Nacti
- NastaveniDatabaze
- UrciteNastaveni

2.3 Třída MainActivity

Třída MainActivity je hlavní třídou celého projektu. Ve třídě jsou nadefinovány podstatné funkce aplikace, které běží na hlavní obrazovce.

Pomocí Switche můžeme funkci „Vynechání dob“ aktivovat nebo deaktivovat. Pokud je Switch aktivovaný, zobrazí se pole pro nastavení funkce. Pokud Switch je deaktivovaný, žádný text se neukáže.

```
aSwitch2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if(aSwitch2.isChecked())
        {
            textView4.setText("Kolik zahrát:");
            textView5.setText("Kolik vynechat:");
            editTextNumber2.setVisibility(View.VISIBLE);
            editTextNumber3.setVisibility(View.VISIBLE);
            textView5.setVisibility(View.VISIBLE);
            textView4.setVisibility(View.VISIBLE);

            Intent intent = getIntent();
            String str2 = intent.getStringExtra( name: "message_key2");
            String str3 = intent.getStringExtra( name: "message_key3");
            editTextNumber3.setText(str2);
            editTextNumber2.setText(str3);
            swagLayout.setVisibility(View.VISIBLE);



        }
        else {
            textView4.setText("");
            textView5.setText("");
            editTextNumber2.setVisibility(View.INVISIBLE);
            editTextNumber3.setVisibility(View.INVISIBLE);
            swagLayout.setVisibility(View.GONE);
        }
    }
});



}
```

Obrázek 13 Switch - Vynechání dob Zdroj: [vlastní zpracování]

Multithreading

Pokud v aplikaci nevyužíváme multithreading, aplikace probíhá lineárně. Pokud probíhal jeden příkaz a je třeba vykonat další, musel se tento příkaz první dokončit. Pomocí multithreadingu, který se skládá ze dvou a více vláken můžeme vytvořit samostatně vykonávající posloupnosti kódu, které nejsou závislé na sobě. Vlákno můžeme vytvořit pomocí rozhraní Runnable nebo třídou Thread. Díky vláknům může tato aplikace fungovat. Ve vláknech probíhá samostatné klepání a funkce *Vynechání dob*. E-Metronom je tedy ovladatelný i tehdy, kdy probíhá klepání.

Hlavní funkcí je samostatné klepání, které funguje pomocí metody *Run*. Celá tato metoda funguje pomocí vlákna.

```
private class RunnableImpl implements Runnable {

    public void run()
    {
        try
        {
            stop = false;
            nastaveniRychlosti(Integer.parseInt(editText.getText().toString()));

            while(!stop)
            {
                if (metronomKdyzHraje())
                {
                    prehravaniZvuku(zvuk);
                }
            }
        }
        catch (Exception ex)
        {

        }
    }
}
```

Obrázek 14 Metoda run ve vlákně Zdroj: [vlastní zpracování]

Pomocí metody *startOpakovat(View v)* spustíme vlákno a rovnoměrné klepání se začne přehrávat. Podmínkou ale je, že funkce *Vynechávání dob* musí být deaktivovaná, jinak se tato funkce spustí místo rovnoměrného klepání. Celou metodu zaktivujeme kliknutím na tlačítko Start.

```
}else {
    t1 = new Thread(new RunnableImpl());
    t1.start();
    zvuk = spinner.getSelectedItem().toString();
    zvyrazeni = spinner2.getSelectedItem().toString();
}
```

Obrázek 15 Spuštění klepání Zdroj: [vlastní zpracování]

Pomocí metody *stopOpakovat(View v)* nastavíme hodnotu stop na „true“ a vlákno ukončíme. Pokud je aktivní funkce *Vynechání dob*, vlákno je též ukončeno a funkci ukončíme. Celou metodu aktivujeme kliknutím na tlačítko Stop.

```
public void stopOpakovat(View v) {

    stop=true;
    if (hraje) {
        hraje = false;
        mp.setLooping(false);
        mp.stop();
    }

}
```

Obrázek 16 Ukončení klepání Zdroj: [vlastní zpracování]

V metodě *nastaveniRychlosti* nastavujeme výpočet tak, aby metronom klepal počet daných klepů za minutu.

```
public void nastaveniRychlosti(int bpm)
{
    if (bpm == 0)
    {
        mod = 1000;
    }
    else
    {
        mod = 60000 / bpm;
    }
}
```

Obrázek 17 Nastavení rychlosti klepání Zdroj: [vlastní zpracování]

Pomocí metody *prehravaniZvuku* nastavujeme a načítáme zvuk. Metoda najde vybraný zvuk, který jsme si vybrali pomocí funkce *Zvuk*. Pokud zvuk nenajde, aplikace vypíše chybnou hlášku do příkazového řádku. Pomocí metody definujeme také funkci *Zvýraznění*, kde nastavujeme, jak se jednotlivé zvuky přehrají.

```
public void prehravaniZvuku(final String filePath)
{
    try
    {
        if (!soundPoolMap.containsKey(filePath))
        {
            AssetFileDescriptor afd = getAssets().openFd(filePath);
            if (afd == null)
            {
                System.out.println("Nelze najit:" + filePath);
                return;
            }

            int id = soundPool.load(afd,  priority: 1);
            soundPoolMap.put(filePath, id);
        }
        int id = soundPoolMap.get(filePath);
        int number = Integer.parseInt(zvyrazeni);
        if(kdyZvednout!=-1) {
            if (((kolikratJsemKlepnul) % number) == 0) {
                soundPool.play(id,  leftVolume: 1.0f,  rightVolume: 1.0f,  priority: 1,  loop: 0,  rate: 2.0f);
            } else {
                soundPool.play(id,  leftVolume: 1.0f,  rightVolume: 1.0f,  priority: 1,  loop: 0,  rate: 1.0f);

            }
        }else    soundPool.play(id,  leftVolume: 1.0f,  rightVolume: 1.0f,  priority: 1,  loop: 0,  rate: 1.0f);
        kolikratJsemKlepnul++;
        System.out.println("Zvuk " + id);

    }
    catch (Exception ex)
    {
        System.out.println(ex.getMessage() + ex.toString());
    }
}
```

Obrázek 18 Přehrání zvuku a funkce zvýraznění Zdroj: [vlastní zpracování]

Funkce *vynechaniDob* běží v samostatném vlákně. Pokud je zaktivovaný switch, funkce se spustí. Pokud ne, budou fungovat všechny ostatní funkce. Funkce je navržena, že uživatel je omezen rychlostí, která je přímo daná v kódu z důvodu počítání pauzy.

```
public void vynechaniDob()
{
    if (hraje)
    {

        new Handler().postDelayed(new Runnable()
        {
            @Override
            public void run()
            {
                if (hraje)
                {
                    mp.setLooping(true);
                    mp.start();


                    new Handler().postDelayed(new Runnable()
                    {
                        @Override
                        public void run()
                        {

                            mp.setLooping(false);
                            mp.stop();

                            vynechaniDob();


                        }
                    }, hrat);
                }


            }
        }, pauzaKlepani);
    }
```

Obrázek 19 Nastavení funkce vynechání dob Zdroj: [vlastní zpracování]

```
pauzaVynechaniPole = editTextNumber3.getText().toString();
hratVynechaniPole = editTextNumber2.getText().toString();


if (swagLayout.getVisibility()==View.VISIBLE) {

    if (!pauzaVynechaniPole.isEmpty() && !hratVynechaniPole.isEmpty()){
        pauzaKlepani = Long.parseLong( s: pauzaVynechaniPole +"000");
        hrat = Long.parseLong( s: hratVynechaniPole +"000");


        if (!hraje){
            hraje = true;
            mp = MediaPlayer.create( context: MainActivity.this, R.raw.zvuk1);

            vynechaniDob();
        }
    }
```

Obrázek 20 Funkce vynechání dob Zdroj: [vlastní zpracování]

2.4 Třída NacitaniNastaveni

Tato třída definuje zobrazení konkrétního nastavení pokud si rozklikneme naše nastavení v *Bance metronomů.*

Pomocí *nacteniAZobrazeniDat* se nám načtou data. Pokud v databázi zrovna žádné data nejsou, aplikace nás upozorní, že žádné data aktuálně uložené nejsou.

```
void nacteniAZobrazeniDat(){
    if(getIntent().hasExtra( name: "id") && getIntent().hasExtra( name: "tempo") &&
            getIntent().hasExtra( name: "kolik_vynechat") && getIntent().hasExtra( name: "kolik_zahrat")){

        id = getIntent().getStringExtra( name: "id");
        tempo_txt = getIntent().getStringExtra( name: "tempo");
        kolik_zahrat = getIntent().getStringExtra( name: "kolik_vynechat");
        kolik_vynechat = getIntent().getStringExtra( name: "kolik_zahrat");


        tempo.setText(tempo_txt);
        kolik_vynechat2.setText(kolik_zahrat);
        kolik_zahrat2.setText(kolik_vynechat);

    }else{
        Toast.makeText( context: this,  text: "Žádné data.", Toast.LENGTH_SHORT).show();
    }
}
```

Obrázek 21 Načtení nastavení v bance metronomů Zdroj: [vlastní zpracování]

Pomocí *potvrzovaciDialog* budeme vyzváni zda, data z databáze chceme vymazat nebo akci odmítnout a nastavení v databázi zůstane. Pokud akci potvrdíme, data se vymažou.

```
void potvrzovaciDialog(){
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("Smazat nastavení " + id + " ?");
    builder.setMessage("Jste si jisti, že chcete smazat nastavení " + id + " ?");
    builder.setPositiveButton( text: "Ano", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            NastaveniDatabaze myDB = new NastaveniDatabaze( context: NacitaniNastaveni.this);
            myDB.smazaniJednohoNastaveni(id);
            finish();
        }
    });
    builder.setNegativeButton( text: "Ne", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {

        }
    });
    builder.create().show();
}
```

Obrázek 22 Potvrzení o smazání dat z databáze Zdroj: [vlastní zpracování]

Tlačítkem „Načíst“, načteme data na úvodní obrazovku a data se nám vepíšou do patřičných polí. Tlačítkem „Smazat“ zavoláme metodu *potrvzovaciDialog* a data smažeme z databáze.

```
smazat_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { potvrzovaciDialog(); }
});
```

Obrázek 23 Tlačítko smazat Zdroj: [vlastní zpracování]

2.5 Třída Nacti

Pomocí třídy definujeme, jak se budou zobrazovat jednotlivá nastavení v bance metronomů. Nastavujeme výpis hodnot z databáze pomocí *ArrrayListu*.

```
public class Nacti extends AppCompatActivity {
   RecyclerView recyclerView;
    NastaveniDatabaze myDB;
    ArrayList<String> banka_id, tempo, kolik_vynechat, kolik_zahrat;
    UrciteNastaveni urciteNastaveni;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.nacti);
        recyclerView = findViewById(R.id.recyclerView);

        myDB = new NastaveniDatabaze( context: Nacti.this);
        banka_id = new ArrayList<>();
        tempo = new ArrayList<>();
        kolik_vynechat = new ArrayList<>();
        kolik_zahrat = new ArrayList<>();

        storeDataInArrays();

        urciteNastaveni = new UrciteNastaveni( activity: Nacti.this, context: this, banka_id, tempo, kolik_vynechat,
                kolik_zahrat);
        recyclerView.setAdapter(urciteNastaveni);
        recyclerView.setLayoutManager(new LinearLayoutManager( context: Nacti.this));
    }
```

Obrázek 24 Nastavení ArrayListu Zdroj: [vlastní zpracování]

Pomocí *vsechnyData* vypíšeme všechny data do banky metronomů.

```
void vsechnyData(){
    Cursor cursor = myDB.zobrazVsechnaData();
    if(cursor.getCount() == 0){
        Toast.makeText( context: this,  text: "Žádné nastavení", Toast.LENGTH_SHORT). show() ;
    }else{
        while (cursor.moveToNext()){
            banka_id.add(cursor.getString( columnIndex: 0));
            tempo.add(cursor.getString( columnIndex: 1));
            kolik_vynechat.add(cursor.getString( columnIndex: 2));
            kolik_zahrat.add(cursor.getString( columnIndex: 3));
        }

    }
}
```

Obrázek 25 Výpis dat z banky metronomů Zdroj: [vlastní zpracování]

2.6 Třída UrciteNastaveni

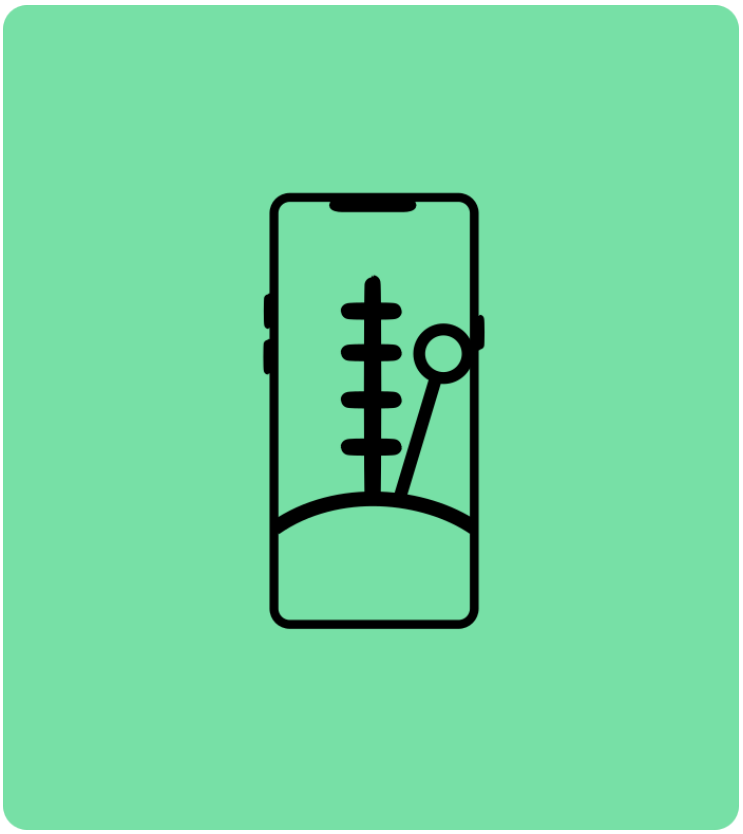
Pomocí této třídy definujeme jednotlivé nastavení v Bance metronomů.

```
public class mojeZobrazeni extends RecyclerView.ViewHolder {



    TextView banka_id_text, tempo_text, kolik_vynechat_text, kolik_zahrat_text;
    LinearLayout mainLayout;

    mojeZobrazeni(@NonNull View itemView) {
        super(itemView);
        banka_id_text = itemView.findViewById(R.id.banka_id_text);
        tempo_text = itemView.findViewById(R.id.tempo_text);
        kolik_vynechat_text = itemView.findViewById(R.id.kolik_vynechat_text);
        kolik_zahrat_text = itemView.findViewById(R.id.kolik_zahrat_text);
        mainLayout = itemView.findViewById(R.id.mainLayout);
    }

}
```

Obrázek 26 Zobrazení nastavení Zdroj: [vlastní zpracování]

2.7 Tvorba loga

Návrh loga aplikace jsem provedl první na papír. První návrh, který jsem pak nakreslil se mi ve výsledku nelíbil. Proto jsem návrh trochu upravil a vytvořil konečné logo aplikace E-Metronom. Logo má jasně znázorňovat o jakou aplikaci jde. Tvorbu loga jsem prováděl v aplikaci Inkscape. Inkscape jsem zvolil z důvodu, že je jednoduchý a umím v něm pracovat.



Obrázek 27 Logo aplikace Zdroj: [vlastní zpracování]

3 TESTOVÁNÍ

V rámci testování, jsem aplikaci nechal otestovat dvěma uživateli. Zvolil jsem svou učitelku na kytaru a kamaráda, který hraje na hudební nástroj. Oba používají mechanický metronom. Poskytl jsem jim svoji aplikaci, aby ji zkusili využít v praxi. Líbilo se jim, že aplikace je jednoduchá a přehledná. Pro učitelku byla velká výhoda uložení nastavení metronomu. Po každé hodině s žákem si uložila nastavení a v následující hodině pokračovali s tím nastavením, se kterým skončili. Druhý uživatel nejvíce ocenil funkci *zvýraznění* a *vynechání dob*. Pomocí těchto funkcí se mohl lépe zdokonalit ve hře na hudební nástroj. Také ocenil výběr zvuku klepání, jelikož trénuje dlouho. Oba uživatelé by ocenili, aby funkce *Vynechání dob* nebyla omezená rychlostí klepání, daným zvukem závislá na funkci *Zvýraznění*. Při testování našli pár nedostatků aplikace. Velikost písma v celé aplikaci bylo moc malé. Další nedostatek byl, že aplikace nevypíše žádné oznámení o uložení nastavení do banku. Tyto nedostatky jsem následně v aplikaci opravil.

ZÁVĚR

Podařilo se vytvořit přehlednou mobilní aplikaci, která slouží jako náhrada mechanického metronomu. Pomocí funkcí se hráč může rozvíjet v hraní více než s klasickým metronomem.
E-Metronom může mít uživatel vždy u sebe v mobilním zařízení.
Uživatel si také může snadno uložit nastavení a později se k němu kdykoliv vrátit. Pokud uživatel bude chtít jiný zvuk klepání, může si vybrat z nabídky zvuků.
Aplikaci je možno používat na všech zařízeních s operačním systémem Android 4. 4 a vyšším.
Do budoucna bude v aplikaci doděláno:

- Vylepšení funkce *Vynechání Dob* – aby nebyla závislá na daném tempu, zvuku a na funkci *Zvýraznění*.
- Doplnění více zvuků klepání
- Tmavý režim aplikace

Pokud by byl o aplikaci větší zájem, chystá se také verze pro zařízení iOS.

SEZNAM POUŽITÉ LITERATURY

1) Android | How to send data from one activity to second activity - GeeksforGeeks. *GeeksforGeeks | A computer science portal for geeks* [online]. Dostupné z: https://www.geeksforgeeks.org/android-how-to-send-data-from-one-activity-to-second-activity/
2) Java String to Int – How to Convert a String to an Integer. *Learn to Code — For Free — Coding Courses for Busy People* [online]. Dostupné z: https://www.freecodecamp.org/news/java-string-to-int-how-to-convert-a-string-to-an-integer/
3) Lekce 1 - Multithreading v Javě. *itnetwork.cz - Ajťácká sociální síť a materiálová základna pro C#, Java, PHP, HTML, CSS, JavaScript a další.* [online]. Copyright © 2022 itnetwork.cz. Veškerý obsah webu [cit. 13.03.2022]. Dostupné z: https://www.itnetwork.cz/java/vlakna/multithreading-v-jave
4) java - Play a sound every N milliseconds - Stack Overflow. *Stack Overflow - Where Developers Learn, Share, & Build Careers* [online]. Dostupné z: https://stackoverflow.com/questions/6018824/play-a-sound-every-n-milliseconds
5) Android SQLite Database Tutorial 1 # Introduction + Creating Database and Tables (Part 1) - YouTube. *YouTube* [online]. Copyright © 2022 Google LLC [cit. 13.03.2022]. Dostupné z: https://www.youtube.com/watch?v=cp2rL3sAFmI
6) Documentation | Android Developers. *Android Developers* [online]. Dostupné z: https://developer.android.com/docs
7) SoundPool in Android with Examples - GeeksforGeeks. *GeeksforGeeks | A computer science portal for geeks* [online]. Dostupné z: https://www.geeksforgeeks.org/soundpool-in-android-with-examples/
8) Initialize an ArrayList in Java - GeeksforGeeks. *GeeksforGeeks | A computer science portal for geeks* [online]. Dostupné z: https://www.geeksforgeeks.org/initialize-an-arraylist-in-java/
9) Android Tutorial => Define strings. *Learn programming languages with books and examples* [online]. Dostupné z: https://riptutorial.com/android/example/440/define-strings
10) Java Examples for android.content.res.AssetFileDescriptor. *Java Tips, Articles, Tutorials* [online]. Copyright © 2011 [cit. 27.03.2022]. Dostupné z: https://www.javatips.net/api/android.content.res.assetfiledescriptor

11) java - How To Test If Cursor Is Empty in a SQLiteDatabase Query - Stack Overflow. *Stack Overflow - Where Developers Learn, Share, & Build Careers* [online]. Dostupné z: https://stackoverflow.com/questions/7222873/how-to-test-if-cursor-is-empty-in-a-sqlitedatabase-query

12) How to change the default icon of Android App - GeeksforGeeks. *GeeksforGeeks | A computer science portal for geeks* [online]. Dostupné z: https://www.geeksforgeeks.org/how-to-change-the-default-icon-of-android-app/

13) Java Code Examples for MediaPlayer | Tabnine. *Code Faster with AI Code Completions | Tabnine* [online]. Dostupné z: https://www.tabnine.com/code/java/classes/uk.co.caprica.vlcj.player.MediaPlayer

14) Java Long parseLong() Method - Javatpoint. *Tutorials List - Javatpoint* [online]. Copyright © Copyright 2011 [cit. 27.03.2022]. Dostupné z: https://www.javatpoint.com/java-long-parselong-method

SEZNAM OBRÁZKŮ