České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Aplikace pro zkoušení slovní zásoby na platformě Android

Eva Mayerová

Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

5. května 2014

Poděkování

Ráda bych tímto poděkovala vedoucímu mé práce, panu Ing. Pavlu Kubalíkovi, Ph.D., který vždy vznesl konstruktivní připomínky na obsah práce. Dále bych chtěla poděkovat kamarádům, kteří byli ochotni mou aplikaci otestovat. V neposlední řadě velice děkuji mému příteli za věcné připomínky a podporu, která byla velice potřebná při psaní celé práce. Konečně, velké díky patří mým rodičům za podporu, bez které bych vůbec nemohla vysokou školu studovat.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 5. května 2014 .....

Odkaz na tuto práci

Mayerová, Eva. *Aplikace pro zkoušení slovní zásoby na platformě Android.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

Abstrakt

Hlavním cílem této práce je implementace výukové aplikace pro operační systém Android se zaměřením na výuku cizích jazyků. Byla provedena analýza současných aplikací na trhu, vytvoření funkčních požadavků a návrh uživatelského rozhraní. Dále se práce zabývá samotnou implementací a konečným testováním. Součástí práce je uživatelská příručka.

Klíčová slova Android, cizí jazyky, výuka, mobilní aplikace

Abstract

The main objective of this work is the implementation of educational application for the Android operating system with a focus on foreign language learning. Analysis of existing applications on the market, functional requirements and user interface design were made. The thesis also describes the implementation and final testing. The part of the work is a user guide.

Keywords Android, foreign languages, teaching, mobile application

Obsah

Seznam obrázků

Úvod

Nacházíme se v době, kdy hranice mezi národy jsou méně zřetelné, než kdy dříve. V době, kde vládnou velké korporace s tisíci zaměstnanci z různých koutů světa. V době, kde můžeme téměř libovolně cestovat po celém světě. A právě v této době je znalost cizích jazyků nezbytná daleko více, než kdy předtím. Dává lidem výhody téměř ve všech oblastech života. Pomáhá získat lepší zaměstnání, dorozumět se na dovolené nebo porozumět zahraničním filmům.

Ať tak či tak, studium cizích jazyků je nezbytnou součástí života téměř každého člověka. V dnešní době se věková hranice pro výuku cizího jazyka na základních školách stále snižuje. Děti proto po ukončení střední školy mají slušný základ, na kterém mohou dále stavět, a jazyky tak zdokonalovat. Bohužel ale, většina lidí nepřichází do styku s cizí mluvou každý den, a tak nějakou dobu po škole mluvit postupně zapomene. Proto je dobré tuto znalost stále procvičovat.

Každý nemá čas a peníze na to, aby chodil na soukromé hodiny, o delším pobytu v kýžené zemi nemluvě. Je proto vhodné využít jinou, snadno dostupnou a jednoduchou, formu výuky. Ještě před několika lety by byl zcela jistou volbou počítačový program. Avšak dnes, kdy se mobilní zařízení stala o tolik chytřejšími, by bylo velice pohodlné mít výukovou aplikaci stále u sebe. Uživatel tak ušetří drahé hodiny, a může se učit například v metru, v čekárně u lékaře, nebo o přestávce v práci.

Cílem této práce je srovnat existující řešení těchto typů aplikací a následně vytvořit jednoduchou výukovou aplikaci se zaměřením na výuku cizích jazyků. Bude umožňovat tři typy zkoušení, a tak může zacílit co možná nejširší spektrum uživatelů. Dále bude možné vkládat vlastní lekce, exportovat a sdílet současný stav zkoušení a nastavit obtížnost. Aplikace je určena pro zařízení s operačním systémem Android. Bude podrobena testování a následnému vložení na Google Play.

Kapitola 1

Cíl práce

Jak již bylo zmíněno v úvodu, cílem této práce je následující:

- Provést rešerši existujících aplikací na trhu. Porovnat jejich klady a zápory, a na jejich základě navrhnout vhodnou funkčnost vytvářené aplikace.
- Navrhnout vhodné a co možná nejintuitivnější uživatelské rozhraní.
- Implementovat výslednou aplikaci, která bude umožňovat alespoň tři různé způsoby zkoušení tak, aby byla funkční pro mobilní telefony a tablety s operačním systémem Android.
- Vytvořit několik testovacích sad slovíček, která budou uživatelům po instalaci k dispozici.
- Aplikaci řádně otestovat.
- Vložit výsledný produkt na Google Play.

Kapitola 2

Rešerše existujících aplikací

Na Google Play se vyskytuje několik aplikací pro výuku cizího jazyka. Z průzkumu vyplynuly následující údaje:

Tabulka 2.1: Výběr některých výukových aplikací z Google Play

Název aplikace	Počet stažení	Cena	Hodnocení
Angličtina slovíčka[1]	50 000+	demo verze zdarma	4.4
Angličtina - Mobilní učitel[2]	50 000+	pět lekcí zdarma	4.1
Dril - angličtina efektivně[3]	50 000+	zdarma	4.3
Angličtina pokročilé zásoby[4]	100 000+	zdarma	4.1

- **Angličtina slovíčka**
  Při prvním spuštění aplikace doinstaluje obsah přesně podle potřeb uživatele, což může být velká výhoda. Umožňuje nastavit úroveň jazykové dovednosti a frekvenci výuky. Vyžaduje ale přístup k Internetu a dostatek místa v telefonu, aby posléze mohla stáhnout výslovnost. Tento krok může být výhodou i nevýhodou, ale je rozhodně dobré, že lze přeskočit. Umožňuje více typů zkoušení, a to pomocí tzv. flashcards (podrobněji popsáno níže), poslechu a výběru z možností. Aplikace uživatele hodně vede a nepovolí mu přílišné změny. Každý den je k dispozici pouze padesát slovíček s tím, že uživatel musí počkat do druhého dne na další. Slova si navíc nevybere sám, ale aplikace je přiřadí podle zvolené obtížnosti. Některým uživatelům toto nemusí vyhovovat.

- **Angličtina - Mobilní učitel**
  Tato aplikace rozhodně zaujme uživatele barevností uživatelského rozhraní. Obsahuje velké množství okruhů k učení, například výuku slovíček, gramatiky a frází. Zkoušení probíhá pouze pomocí poslechu. Po prvním stažení aplikace je k dispozici pouze jedna lekce, přičemž další lekce slibuje zdarma po dalším stažení. Nevýhodou této aplikace je možnost

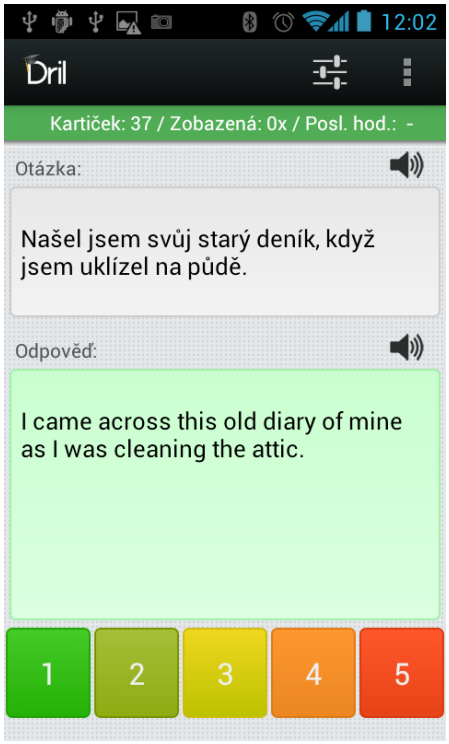
Obrázek 2.1: Angličtina - slovíčka

Obrázek 2.2: Angličtina - Mobilní učitel

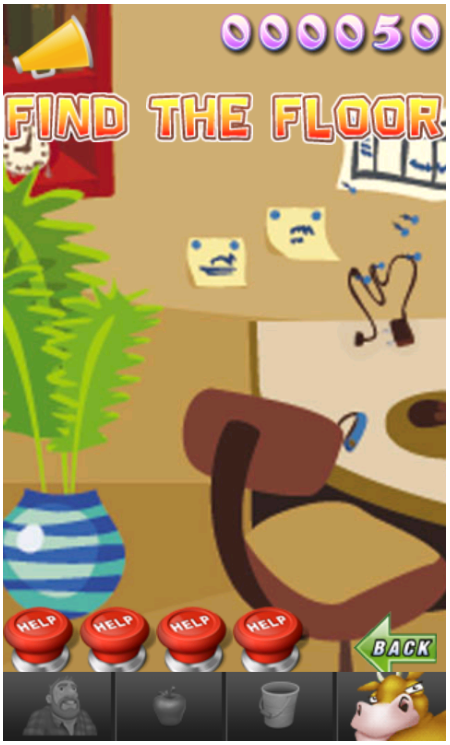
zkoušení pouze pomocí poslechu. Výhodou je velké množství okruhů ke zkoušení.

- **Dril - angličtina efektivně**
  Před prvním zkoušením je v aplikaci potřeba aktivovat kartičky. Ty jsou součástí několika učebnic, přičemž uživatel si může vybrat konkrétní slova, nebo celé knihy. Jakmile jsou aktivovány, učení probíhá pomocí drilu (popsáno níže). Uživatel svou odpověď na danou kartičku ohodnotí stupnicí jedna až pět jako ve škole. Dále je k dispozici statistika vyzkoušených lekcí s průměrnou známkou. Výhodou aplikace je její jednoduchost. Je k dispozici pouze jeden typ zkoušení, proto je aplikace přehledná.

- **Angličtina pokročilé zásoby**
  Aplikace je pojata formou hry. K dispozici je čtrnáct obrázků, kde jsou postupně kladeny dotazy typu „najděte stůl“. Aplikace je s největší pravděpodobností zaměřená na děti. Nevýhodou je malé množství lekcí, výhodou pak naprosto odlišný přístup od ostatních aplikací.



Obrázek 2.3: Dril - angličtina efektivně



Obrázek 2.4: Angličtina pokročilé zásoby

Většina zmíněných aplikací je poskytována zdarma a všechny mají relativně vysoké hodnocení. Co se týče způsobu výuky, většinou byly pozorovány tyto způsoby:

- Dril – Režim podobný obracení papírových karet (tzv. flashcards), kde na jedné straně je české slovo a na druhé straně slovo cizí. Student pak odkládá jednotlivé karty na dvě hromádky podle toho, zda překlad věděl nebo nevěděl. Poté opakuje stejný postup s hromádkou slovíček, která nevěděl, do té doby, než se naučí všechna slovíčka.
- Hra – Jiným způsobem učení je výuka pomocí hry. Aplikace Angličtina pokročilé zásoby[4] zobrazí obrázek místnosti, ve které je velké množství různých předmětů, a dává uživateli jednoduché úkoly v angličtině (např. „vyberte květinu“).
- Poslech – Méně častým, ale přece jen používaným, typem zkoušení je klasický poslech. Uživateli je slovo přehráno z reproduktoru nebo do sluchátek a jeho úkolem je napsat nebo vybrat správný překlad.

Z tohoto srovnání vyplynulo, že je velice málo aplikací, ne-li žádné, které umožňují učení způsobem testu (k jednomu slovíčku se zobrazí více možností

překladu, ze kterých uživatel vybere tu správnou). Dále většina aplikací požaduje připojení k Internetu, odkud stahuje vždy aktuálně zkoušené sady. Toto není moc šťastné řešení, pokud uživatel není připojen, nebo má omezený datový limit.

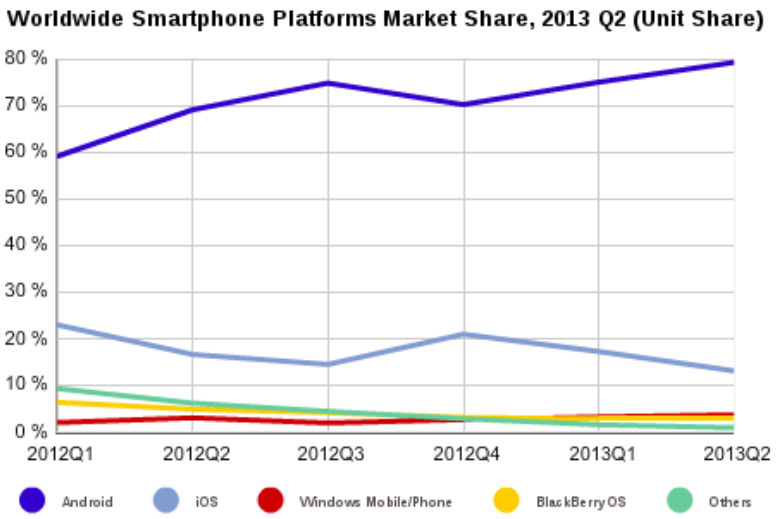
Na základě těchto poznatků je cílem vytvořit aplikaci, která se bude nějakým způsobem odlišovat, a zaplní tak prázdné místo mezi konkurencí.

Kapitola 3

Analýza

3.1 Popis platformy

Jako operační systém, pod kterým výsledná aplikace poběží, byl zvolen Android. Je to mobilní platforma, která v sobě zahrnuje kromě operačního systému také ovladače, knihovny a uživatelské rozhraní [5]. Hlavním důvodem výběru je jeho rozšíření, a to takové, že v polovině roku 2013 byl zastoupen více než 80 % mobilními telefony a tablety na trhu [6].



Obrázek 3.1: Zastoupení OS v mobilních zařízeních, 01/2012 – 06/2013[6]

3.1.1 Architektura

Android používá ARM architekturu procesoru. Je založená na architektuře typu RISC, která je založená na jednoduché a velmi optimalizované sadě instrukcí, tudíž způsobuje nižší spotřebu enegrie [20].

3.1.2 Historie Androidu

V roce 2007 byla společností Google představena Open Handset Alliance (OHA), což je uskupení 84 firem z oblasti vývoje hardware, software, mobilních operátorů a výroby mobilních zařízení [9]. Pod její záštitou vznikl Android Open Source Project (AOSP). Během následujících let představila společnost několik aktualizací Androidu.

- Cupcake (2009) - První oficiální aktualizace.
- Donut (2009)
- Eclair (2010)
- Froyo (2010)
- Gingerbread (2010) - Větší změny v uživatelském rozhraní.
- Honeycomb (2011) - První aktualizace pro tablety. Vytvoření fragmentů (více v kapitole Implementace).
- Ice Cream Sandwich (2011)
- Jelly Bean (2012) - Plynulejší uživatelské rozhraní. Kladen důraz na optimalizaci SW a HW.
- KitKat (2013)

3.2 Vývoj aplikací

K vývoji aplikací slouží Android SDK, které obsahuje potřebné API ke všem dostupným verzím Androidu. Aplikace se programují pomocí jazyka Java a další potřebné informace jsou uloženy v souborech typu XML. Více informací o struktuře Android projektu v kapitole 5.1.

3.2.1 Potřebný software

Jako vývojové prostředí pro vytváření aplikací Google oficiálně podporuje `Eclipse` [16]. Musí ale obsahovat plugin ADT, který obsahuje potřebné nástroje.

Obrázek 3.2: Ilustrační obrázek doposud vydaných verzí Androidu[7]

- Instalaci ADT pluginu lze provést přímo v prostředí Eclipse, a to následovně:
  Zvolení možnosti *Help* → *Install new software...*, kde se doinstalují balíky s klíčovým slovem ADT.

- Jednodušší možnost je stáhnout Eclipse přímo s nainstalovaným ADT pluginem z oficiálních stránek Androidu [22].

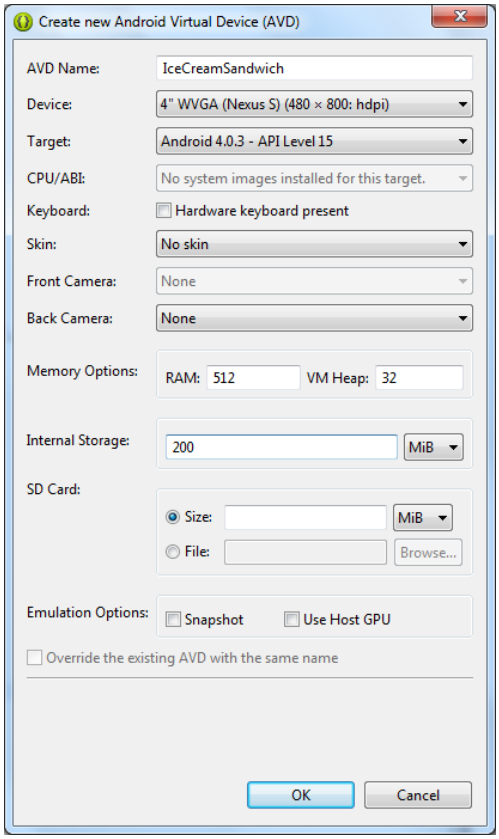
V neposlední řadě je nutné nainstalovat potřebné API balíky. Tato možnost se provede přímo v Eclipse stisknutím ikony *SDK Manager*. Z dostupných možností je třeba nainstalovat vše ve složce *Tools*, vše ve složce s poslední verzí API, a dále vždy *SDK Platform*, *Samples for SDK* a *Google APIs* pro všechny verze API, které mají být aplikací podporovány.

3.2.2 Příprava zařízení

Před začátkem samotného programování je potřeba připravit zařízení, na kterém se bude aplikace spouštět. Jsou dvě možnosti:

- **Emulátor**

  Jde o virtuální zařízení, které slouží k simulaci reálného prostředí. Je možné ho vytvořit a používat ve vývojovém prostředí Eclipse, které bylo zmíněno v kapitole 3.2.1. Umožňuje nastavit různé hardwarové a softwarové parametry. Náhled vytvoření nového zařízení na obrázku 3.3.



Obrázek 3.3: Dialog pro vytvoření virtuálního zařízení

- **Reálné zařízení**

  Druhou možností je testování na reálném zařízení. Aplikace se na telefon nebo tablet nahraje pomocí USB kabelu.

  Zařízení je ale potřeba pro tuto možnost nastavit. Nejprve je nutné zpřístupnit nabídku *Možnosti pro vývojáře*. To lze provést v sekci *Nastavení* → *Info o telefonu*, kde je třeba několikrát poklepat na položku *Číslo sestavení*. Poté, co se zpřístupní Možnosti pro vývojáře, je nutné povolit možnost *Ladění USB*.

  Dalším požadavkem je instalace Android USB ovladače pro konkrétní zařízení, který je dostupný ke stažení na Internetu. V této chvíli by už při kompilování aplikace v Eclipse měla být k dispozici možnost spuštění na připojeném telefonu.

Kapitola 4

Návrh

4.1 Funkční požadavky

Na základě předchozího zhodnocení a po dohodě s vedoucím práce byly na výsledný program stanoveny následující požadavky:

- **Zkoušení slov alespoň třemi způsoby** kvůli pokrytí co možná největšího spektra potenciálních uživatelů. Byly určeny následující typy zkoušení:
  - **Režim drilu** podobný jako u srovnávaných aplikací. Uživatel bude mít k dispozici testované slovo, a dále tlačítko, které slouží ke zobrazení překladu. Na základě toho, zda uživatel slovo znal či neznal, stiskne příslušné potvrzovací nebo zamítací tlačítko.
  - **Testovací režim** uživateli umožní vybrat správné slovo z více ekvivalentů. K dispozici budou tři úrovně obtížnosti. Podle toho bude mít uživatel na výběr ze tří, čtyř, nebo pěti slov. Zkoušení bude, stejně jako v režimu drilu, probíhat do té doby, než uživatel správně zodpoví všechna slova.
  - **Slovník** zobrazující všechna slova vybrané sady. Bude umožňovat jednotlivá slova v seznamu vyhledávat.

  Před začátkem každého zkoušení bude možné zvolit, zda se budou slova zobrazovat anglicko-česky nebo naopak.
- **Import textového souboru.** Aplikace bude umožňovat přidávat vlastní slovní zásobu ve formě textového souboru. Tím si uživatel bude moci vytvořit obsah aplikace přesně podle svých potřeb.
- **Export textového souboru**. Možnost exportu aktuálně zkoušené slovní zásoby s oddělením slov, která uživatel věděl správně a těch, která nevěděl.

- **Uložení rozpracované lekce**. V průběhu výuky může uživatel lekci opustit a uložit si aktuální stav zkoušení. Na začátku nového zkoušení se aplikace dotáže, zda má pokračovat v rozpracované lekci, nebo začít znovu.

- **Grafické zpracování výsledků.** Po každém úspěšném zkoušení se zobrazí graf s výsledky, které budou dále archivovány a srovnávány s budoucím zkoušením stejného souboru slov.

4.2 Návrh uživatelského rozhraní

Při návrhu uživatelského rozhraní byl kladen důraz na jednoduchost a přehlednost.

4.2.1 Hlavní nabídka

V hlavní nabídce proto budou jen čtyři položky, a to položky *výuka*, *správa lekcí*, *import* a *nastavení*.

Po zvolení možnosti *výuka* bude uživateli zobrazena nabídka s konkrétním typem zkoušení. Jak bylo již uvedeno v úvodu této kapitoly, tyto možnosti budou tři.

Z hlavní obrazovky se uživatel také pohodlně dostane do sekce *správa lekcí*, kde si bude moci připravovat jednotlivé slovníkové sady ke zkoušení. Bude zde možnost lekci odstranit či přejmenovat, vybrat ke zkoušení, nebo resetovat.

Import bude mít jedinou možnost, a to výběr požadovaného souboru. Dále se uživateli zobrazí dialog pro výběr souboru a následná indikace výsledku podle toho, zda se soubor podařilo do aplikace nahrát či nikoli.

V sekci *nastavení* bude mít uživatel možnost nastavit si obtížnost testovacího režimu. Na výběr bude zobrazení tří, čtyř, nebo pěti možných odpovědí.

4.2.2 Typy zkoušení

V sekci pro výběr typu výuky nebude chybět mimo jiné tlačítko, kterým si uživatel zvolí, zda chce být zkoušen česko-anglicky, nebo anglicko-česky. Jednotlivé sekce zkoušení budou popsány podrobněji:

- **Dril** - V režimu drilu se zobrazí testované slovo a tlačítko, které odhalí jeho překlad. Uživatel poté sám zvolí, zda překlad věděl, nebo nevěděl. Bude zde také informace o tom, kolik slov chybí do konce zkoušení. Režim bude obsahovat i možnosti sdílení zkoušených slov ostatním aplikacím a export do textového souboru. Rozpracovanou sadu slov bude možné při opouštění uložit.

- **Test** - Režim vzhledově i funkčně podobný drilu s tím rozdílem, že zobrazeno bude testované slovo a pod ním tři, čtyři nebo pět ekvivalentů překladu podle zvolené obtížnosti.
- **Slovník** - Zobrazení všech slov pod sebe, přičemž budou zobrazena přímo se svým ekvivalentem v anglickém jazyce. Bude zde možnost slova vyhledávat zadáním vstupu od uživatele.

Po dokončení zkoušení v režimu *dril* nebo *test* bude zobrazena výsledná statistika aktuálního zkoušení s možností zopakování testu, výběru nové sady slov nebo návratu do hlavní nabídky.

4.2.3 Vzhled aplikace

Většina existujících aplikací zabývající se touto tématikou volí velké množství barev, obrázků a tlačítek. Pro tuto aplikaci byl proto zvolen jednodušší design. Pozadí je laděno do tmavších, méně rušivých barev. V kontrastu s tím jsou ikonky pro upoutání pozornosti bílé a kresleny ručně, viz obrázek 4.1.

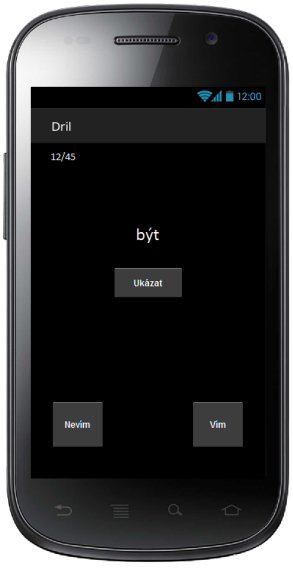
Ikony jsou vytvářeny ve standardní velikosti pro operační systém Android. Ten podporuje celou škálu velikostí displejů, a tak je potřeba vytvořit všechny ikony ve více velikostech. Konkrétní zařízení si už samo vezme příslušný obrázek podle velikosti. Více o vytváření vzhledu je popsáno v kapitole 5.3.

Tabulka 4.1: Přehled doporučených velikostí ikon [10]

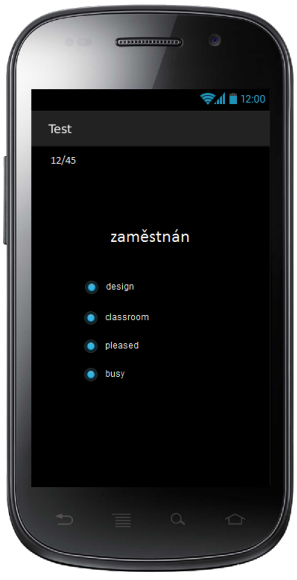
Rozlišení displeje (Android označení)	Velikost ikony (v pixelech)
ldpi	36x36
mdpi	48x48
hdpi	72x72
xhdpi	96x96
Google Play ikona	512x512

Obrázek 4.1: Návrh úvodní obrazovky

Obrázek 4.2: Návrh správy lekcí



Obrázek 4.3: Návrh režimu dril



Obrázek 4.4: Návrh režimu test

Kapitola 5

Programování pro Android

5.1 Struktura projektu

Každý Android projekt má přesně danou adresářovou strukturu, která se musí dodržovat.

- `src/` – Adresář, ve kterém jsou uloženy všechny zdrojové soubory v jazyku Java.
- `gen/` – Adresář obsahující automaticky generované soubory v jazyce Java, které obsahují informace z XML souborů.
- `assets/` – Libovolné soubory, se kterými aplikace dále pracuje.
- `libs/` – Veškeré použité knihovny ve formátu *JAR*.
- `res/` – Adresář, který obsahuje veškeré zdroje. Patří mezi ně například vše, co se týká vzhledu, použité proměnné a styly.
- `res/drawable/` – Všechny použité obrázky jako například ikony a pozadí.
- `res/layouts/` – Soubory popisující layouty jednotlivých *Aktivit*.
- `res/values/` – Obsahuje tři soubory:
  - `dimen.xml` – Soubor obsahující informace o konkrétních hodnotách, např. o velikosti textu, mezer, atd.
  - `strings.xml` – Soubor, který obsahuje všechny řetězce použité v aplikaci. Můžeme definovat soubory pro jiné jazykové sady, např. `strings-en.xml` pro angličtinu. Tyto hodnoty se použijí, pokud má mobilní zařízení nastavenou angličtinu jako výchozí jazyk.
  - `styles.xml` – Definice použitých stylů.

- `AndroidManifest.xml` – Hlavní konfigurační soubor, který nese veškeré informace o aplikaci. Mimo jiné nese také informace o oprávněních, která aplikace potřebuje ke svému běhu. S těmito oprávněními pak uživatel může nebo nemusí souhlasit před instalací aplikace.

Při vytvoření nového projektu je potřeba definovat *Minimum SDK* a *target SDK*. Minimum SDK definuje nejnižší možnou verzi Andoridu, pro kterou má být aplikace určena. Target SDK určije cílovou (nejvyšší) verzi, pro kterou je aplikace vytvářena. V případě této aplikace je Minimum SDK nastaveno na API verzi 14 (Android verze 4.0 – Ice Cream Sandwich) a Target SDK na současně nejvyšší API verzi 19 (Android verze 4.4 – KitKat).

Jako jediné oprávnění vyžaduje tato aplikace přístup ke čtení a zápisu na SD kartu. Je to z důvodu importu a exportu slovníkových sad.

5.2 Activity

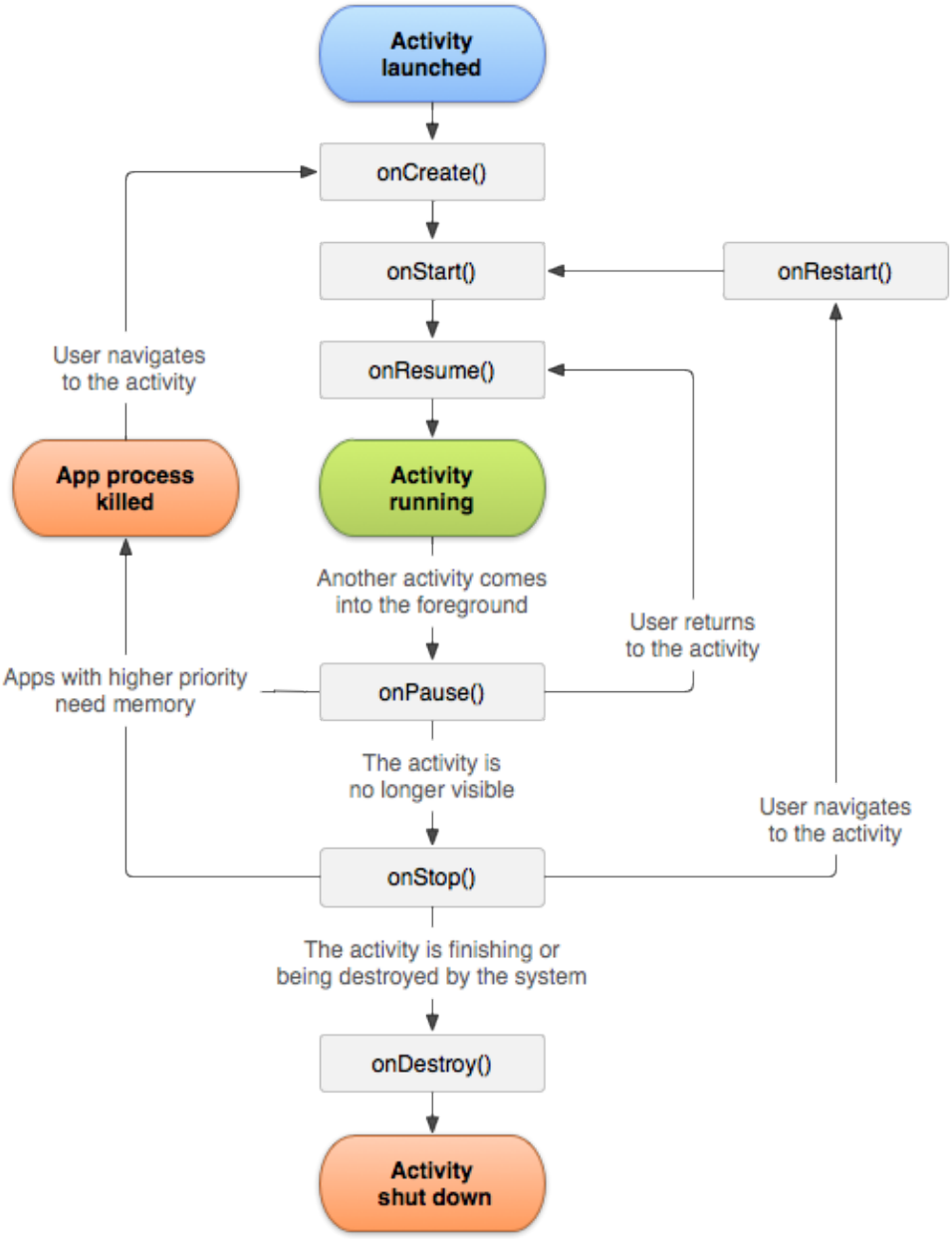
Každá obrazovka, která se v aplikaci nachází, je definována třídou v jazyce Java. Tato třída dědí od třídy zvané *Activity*. Ta je základním stavebním kamenem každé Android aplikace. Každá aktivita je propojena s nějakým *layoutem*.

Layout je část aplikace, která zajišťuje její vzhled. Je definovaný soubory typu XML, které podle předepsaných příkazů popisují *widgety* (v tomto smyslu prvky uživatelského rozhraní – tlačítka, textová pole, ...) a jejich parametry. Ve vývojovém prostředí Eclipse je možnost prvky na obrazovku umístit graficky pomocí myši, což stačí na základní operace. Složitější úpravy se pak provedou přímo v kódu.

Každá aktivita má svůj životní cyklus. Proto obsahuje metody, které se volají při vytvoření, přesunutí do pozadí, probuzení, nebo zániku této aktivity. Pro lepší názornost slouží obrázek 5.1. Dědící třída poté musí tyto metody implementovat. Nemusí nutně definovat všechny, ale metoda `OnCreate()`, která se volá při každém spuštění aktivity, musí být implementována vždy.

V této práci je implementováno několik aktivit.

- `MainActivity` – Základní třída, kterou musí obsahovat každá aplikace. Zastupuje klasickou funkci `main()`. Reprezentuje první obrazovku, která se při spuštění aplikace zobrazí.
- `LearningActivity` – Třída, která reprezentuje obrazovku pro výběr typu zkoušení.
- `DrilActivity, TestActivity, DictionaryActivity` – Třídy sloužící k samotnému zkoušení
- `LessonsListActivity` – Správa lekcí. Tato třída dědí ze speciálního typu aktivity, a to *ListActivity*.



Obrázek 5.1: Životní cylkus aktivity [21]

- `SettingsActivity` – Nastavení.
- `WellDoneActivity` – Aktivita zavolaná při úspěšnému ukončení zkoušení.
- `CharResultActivity` – Aktivita, která zobrazuje výsledky prostřednictvím grafu.

Listing 5.1: Ukázka implementace metody onCreate() ve třídě MainActivity

```
public class MainActivity extends Activity implements
    View.OnClickListener {

    Button learning;
    Button management;
    Button settings;
    Button Import;

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

        ...

        learning = (Button) findViewById(R.id.
            learningButton);
        management = (Button) findViewById(R.id.
            managementButton);
        settings = (Button) findViewById(R.id.
            settingsButton);
        Import = (Button) findViewById(R.id.importButton
            );

        learning.setOnClickListener(this);
        management.setOnClickListener(this);
        settings.setOnClickListener(this);
        Import.setOnClickListener(this);
    }
}
```

Tento kód pouze propojuje prvky v layoutu tak, aby bylo možné odchytávat kliknutí. O to se stará třída `View.OnClickListener`. Dále je třeba implementovat metodu `onClick()`, ve které se dále určí, co se má po stisku tlačítka provést.

Při otočení displeje, přesunutí aplikace do pozadí, nebo jakékoliv jiné změně, se aktivita dostane do stavu *paused*. Je však potřeba si uchovat nějaké

hodnoty i po této situaci. K tomu slouží třída *Bundle*, která uchovává informace různých typů. V metodě `onCreate` se pak Bundle předá jako parametr, se kterým je možné dále pracovat.

5.3 Vytvoření GUI

Jak již bylo zmíněno dříve, GUI se vytváří pomocí Layoutu. Displej může být orientován na výšku (`portrait`), nebo na šířku (`landscape`). Je proto vhodné definovat layout pro oba tyto případy. Každý layout se obyčejně umístí do složky `res/layout`. Pro definování layoutu v pozici landscape je nutné vytvořit další adresář, a to `res/layout-land`. Do této složky pak stačí jen umístit soubor se stejným názvem jako v `res/layout`, a aplikace ho sama použije.

V aplikaci *Zkoušení slovní zásoby* je layout pro landscape orientaci použit pouze u některých aktivit, a to u těch, které potřebují mít po otočení displeje jiné rozložení prvků.

Listing 5.2: Část layoutu hlavní obrazovky

```
<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/linearLayout1"
    android:layout_marginLeft="18dp"
    android:layout_toRightOf="@+id/linearLayout1"
    android:orientation="vertical" >

    <Button
        android:id="@+id/importButton"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:background="@drawable/importicon" />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="@string/importButton"
        android:textAppearance="?android:attr/
            textAppearanceMedium" />
</LinearLayout>
```

Jako jednotky velikosti jsou používány `dip` (zkráceně `dp`). Je to abstraktní jednotka, pro kterou platí, že jeden dp odpovídá jednomu pixelu na displeji

Obrázek 5.2: Aktivita s dvěma fragmenty

s rozlišením 160 dpi. Nejmenší pohodlná velikost tlačítka je definována na šířku 48 dpi [11].

Android také definuje složitější prvky uživatelského rozhraní. V této aplikaci je použito hlavně *ListView*, což je prvek zobrazující jednotlivé položky ve vertikálně posuvném okně [12]. Je použit ve třídě *LessonListActivity*, která, jak již bylo zmíněno výše, dědí od ListActivity.

5.4 Fragment

Fragment je třída, která reprezentuje chování celého nebo části uživatelského rozhraní [13]. Je součástí aktivity, která může obsahovat jeden nebo více fragmentů. Tato funkce přibyla ve verzi *Honeycomb* a využívá se hlavně pro rozdílné zobrazení obsahu na obrazovkách různých velikostí. V této práci jsou fragmenty použity při zobrazování grafů. Aktivita je v tomto případě rozdělena na dvě části, mezi kterými se přepíná pomocí tabů. Pro názornost slouží obrázek 5.2.

5.5 Intent

Při implementaci bylo mnohdy potřeba předávat data mezi jednotlivými Aktivitami. Při zkoušení je například nutné posílat si informace o už vyzkoušených a dosud nezodpovězených slovech. K tomuto účelu slouží třída *Intent*, která může držet hodnoty různých typů, a poslat je tak nově vytvořené aktivitě.

Listing 5.3: Poslání informací pomocí Intentu

```
Intent i = new Intent(this, DrilActivity.class);
i.putExtra("czToEn", czToEn);
startActivity(i);
```

Listing 5.4: Příjem informací v nové Aktivitě

```
Intent i = getIntent();
czToEn = i.getBooleanExtra("czToEn", czToEn);
```

Kapitola 6

Implementace

6.1 Uchovávání dat

Aplikace si v paměti potřebuje držet informace o jednotlivých slovníkových sadách, včetně toho, jak jsou v současnou chvíli rozpracované. Dále je potřeba vědět, jaké lekce jsou zrovna vybrané ke zkoušení a jaká je nastavená obtížnost, a to i po restartování aplikace. Je tudíž třeba mít vytvořen konfigurační soubor.

Java nabízí možnost vytvářet konfigurační soubory pomocí třídy *Properties*. Je to třída, která dědí z třídy *Hashtable*. Umožňuje držet si informace v podobě „klíč“ = „hodnota“. Všechny hodnoty jsou uchovávány pomocí datového typu `String` [14]. Informace o aktuálně označených lekcích a obtížnosti jsou tedy uloženy v souboru `data.properties`, přičemž k vytvoření tohoto souboru dojde při prvním spuštění aplikace a jeho modifikaci ve *Správě lekcí* a *Nastavení*.

Informace o jednotlivých slovníkových sadách jsou uloženy v souborech typu `XML`. Nejedná se o zrovna nejúspornější formát, ale k použití v této aplikaci se hodí. Dokáže totiž držet informace o stavu jednotlivých slov v souboru, jako je třeba počet špatných odpovědí nebo informace o tom, zda už bylo zkoušeno. Před každým novým zkoušením se do kontejneru načtou všechna slova ze souborů, které jsou uvedeny v konfiguračním souboru, a ta jsou pak předána konkrétní Aktivitě.

Při importu souboru obdrží aplikace data v textové podobě. Je nutné si tyto údaje převést do XML a uložit do interního úložiště. K tomu slouží vytvořená třída, jejíž jedna z funkcí dostane jako vstup textový soubor a vrátí `InputStream`, který se dále uloží do XML souboru. Stejný proces se děje při prvním spuštění aplikace, protože data jsou přibalena také v textovém souboru a aplikace si je musí převést do XML. Z toho důvodu může na začátku trvat pár sekund, než se zobrazí obsah hlavní obrazovky.

Všechna aplikační data se ukládají do interního úložiště aplikace, ke kterému ostatní aplikace nemají přístup. Pro získání cesty k internímu úložišti

slouží v Javě funkce

```
getApplicationContext.getFilesDir();
```

6.1.1 Parsování XML

K získání informací z XML souboru je potřeba vytvořit XML Parser. Existuje několik již hotových, jako například `DOM` nebo `SAX` parser. Vývojáři Androidu však doporučují `XmlPullParser`, který slouží k parsování XML přímo pro Android [15]. Implementace vlastního parseru využívající XmlPullParser je k nahlédnutí v příloze D.

6.2 Načítání kolekcí slovíček

Načítání slovíček se uskutečňuje v *LearningActivity*. Při přístupu na obrazovku s rozcestníkem typů zkoušení si aktivita načte seznam souborů, ze kterých má čerpat slovíčka. Každé z těchto slov, definováno v XML, má následující atributy:

- Český název
- Cizí (v tomto případě anglický) název
- Informaci o stavu slova (existují tři typy stavů, více v sekci 6.3)
- Informaci o tom, zda už bylo testováno
- Počet chybných odpovědí
- Časová prodleva mezi zobrazením slova a odpovědí u posledního zkoušení slova

Pro tato slova je definována samostatná třída *Word*, do jejíž instance se tyto údaje uloží. Všechna slova v souboru jsou dále pohromadě uložena v instanci třídy *Vocabulary*.

Při načítání slov se vytvoří instance třídy XmlParser (implementace viz příloha D), jejíž funkce `parse()` vrátí kolekci všech slovíček souboru. Návratová hodnota této funkce je `ArrayList<Word>`.

Třída dále projde všechna slova a rozdělí je do tří kolekcí podle stavu. Ty poté pomocí Intentu předá buď *DrilActivity*, *TestActivity* nebo *DictionaryActivity* společně s informací, zda bude zkoušení probíhat česko-anglicky, nebo anglicko-česky.

6.3 Systém výběru slov ke zkoušení

Při zkoušení byla slova nejprve vybírána náhodně, tedy pomocí třídy *Random*. Instance této třídy vybírala vždy ze všech slov. Ukázalo se ale, že je to problematické z toho důvodu, že slova se mohla několikrát za sebou opakovat, nebo se některé slovo zobrazilo už po několikáté, zatímco jiné se ještě nevybralo ani jednou. Proto bylo nutné vytvořit vlastní systém, který bude třídu Random také využívat, ale s jistým omezením.

Následující systém se ukázal být nejjednodušším a zároveň velice spolehlivým. Slova se rozdělila do tří kategorií.

- Aktuální
- Pozastavená
- Správně zodpovězená

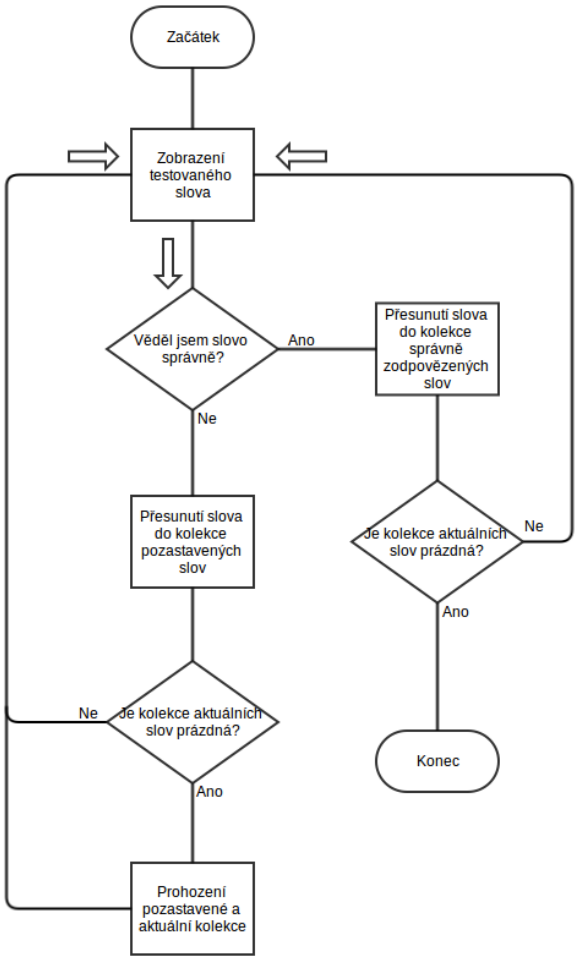
Na začátku zkoušení se všechna slova nachází ve stavu *aktuální*. Je to skupina, ze které může vlastní třída *RandomPicker* libovolně vybírat slova. Dále, pokud vybrané slovo uživatel zná, změní testované slovo stav na *správně zodpovězené*. Z této skupiny se již dále slova nevybírají. Pokud ale uživatel slovo neznal, stav slova se změní na *pozastavené*. V této chvíli jsou tato slova odložena stranou a generátor z nich také nevybírá. V momentě, kdy se ale vyprázdní kolekce *aktuálních* slov, změní slova z kolekce *pozastavených* stav na *aktuální* a celý proces se opakuje, dokud nejsou všechna slova označena stavem *správně zodpovězené*. Pro lepší názornost slouží diagram na obrázku 6.1.

6.4 Export textového souboru

Jedním z požadavků na funkčnost byl export textového souboru. Je potřeba, aby se rozlišila zvlášť slova, která byla zodpovězená správně a chybně. Díky uloženému stavu u každého slova tedy stačí uložit do jedné skupiny ta, která mají nastavený stav *správně zobpovězené* a do druhé skupiny zbytek. Typ souboru je kvůli lepší čitelnosti zvolen jako `TXT`. Ve chvíli, kdy tedy uživatel stiskne *Export*, proběhne operace, díky níž se nově vytvořený soubor umístí do adresáře *Downloads*. Z tohoto důvodu je nutné, aby měla aplikace přístup ke sdílenému úložišti. Název souboru odpovídá časové značce ve chvíli exportu. Obsah tohoto souboru dodržuje následující formát:

```
Known words:
czech word = english word

Mistaken words:
czech word = english word
```



Obrázek 6.1: Vývojový diagram algoritmu náhodného výběru slovíček

Obrázek 6.2: Action Bar v TestActivity

6.5 Sdílení souboru slov v Action Baru

Aplikace dále umí sdílet rozpracovanost lekce dalším aplikacím, jako je například editor SMS zpráv nebo GMail. Sdílený obsah je uložen ve formě textového řetězce a obsah přesně odpovídá obsahu exportovaného souboru. Možnost sdílení je zobrazena v horní nabídce, v tzv. *Action Baru*.

6.5.1 ActionBar

Action Bar je část obrazovky, která slouží k navigaci uživatele skrz aplikaci. Zachovává si stejnou podobu napříč různými aplikacemi, a umožňuje tak lepší orientaci. Navíc se přizpůsobuje různým velikostem obrazovky [17].

Action bar v této aplikaci s tlačítkem pro sdílení obsahu je zobrazen na obrázku 6.2

Export souboru je ukryt v tzv. *Overflow* menu. V něm jsou obvykle umístěné méně používané položky. Na tomto zařízení je pro něj zvláštní tlačítko na spodní straně displeje, stejně jako domovské tlačítko. Na jiných zařízeních však může být umístěno vpravo v Action Baru, viz obrázek 6.2.

Informace o sdíleném obsahu jsou odesílány ostatním aplikacím pomocí Intentu. O sdílení se stará *ShareActionProvider*, který zajišťuje data cílovým aplikacím [18].

6.6 Ukládání stavu slov

Při každém zkoušení je nutné mít možnost uložit započatý stav. Tato možnost se uživateli zobrazí při pokusu opustit obrazovku nebo je možné ji najít v Overflow menu.

Samotné ukládání je popsáno v následujících krocích:

- Vytvořit instanci třídy *XmlCreator* a odeslat jí všechny tři kontejnery se slovy
- Získat hodnotu typu *InputStream* po zavolání patřičné funkce
- Vytvořit nový XML soubor z příslušného *InputStream* s dočasným jménem
- Odstranit původní soubor s lekcí

- Přejmenovat dočasný soubor na název původní lekce

V sekci *Správa lekcí* se poté zobrazí rozpracovanost každé lekce v procentech. Přímo tam je také možno ji resetovat. Resetovat lekci je možné také při spuštění rozpracovaného zkoušení, kde se aplikace uživatele zeptá, zda chce pokračovat v uložené pozici nebo lekci resetovat.

6.7 Import souboru

Import souboru byl jedním z hlavních požadavků na výslednou aplikaci. Uživatel by měl mít možnost vybrat textový soubor ze svého souborového systému. Ten by se měl dále zpracovat na XML, se kterým bude aplikace dále pracovat.

Největším problémem bylo vytvoření dialogu pro listování souborovým systémem. Inspirace k samotnému dialogu k nalezení zde [19]. Poté bylo provedeno několik modifikací, jako například abecední řazení nebo způsob zobrazení. V seznamu se zobrazují pouze adresáře a textové soubory.

Zdrojový kód je realizován třídou *AndroidExplorer*, která dědí z ListActivity.

Aplikace také hlídá správný formát vstupního souboru. V případě, že soubor neobsahuje ani jeden korektní řádek, aplikace vrátí výjimku. V opačném případě zpracuje pouze řádky, které splňují předepsaný formát. Po vybrání příslušného souboru uživatelem se na displeji objeví krátká zpráva s informací, zda k importu došlo, či nikoli.

6.8 CustomAdapter ve správě lekcí

Při vytváření jakéhokoliv ListView je potřeba vytvořit instanci třídy *Adapter*, která se stará o předání dat a jejich zobrazení v ListView. Existují specifičtější typy adaptérů, podle sdíleného obsahu. Jedním z nich je např. *ArrayAdapter<String>*, který do ListView posílá pouze pole hodnot typu *String*. Dalším typem adaptéru je *SimpleAdapter*, který je velice zjednodušený a umožňuje předat jakékoliv hodnoty, např. mapu řetězců. Ve správě lekcí nestačí jako položka ListView pouhé TextView, a tak je nezbytné použít *CustomAdapter*.

6.8.1 CustomAdapter

Jedná se o typ adaptéru, který si programátor sám definuje. Vytvoří si vlastní třídu, která dědí od CustomAdapteru, a tu poté používá jako Adapter.

CustomAdapter musí definovat konstruktor, ve kterém zavolá konstruktor rodiče, a funkci `getView()`, díky které definuje obsah položkám v layoutu a obstarává kliknutí na vnitřní tlačítka.

V tomto případě je v položce ListView místo jednoduchého TextView použit komplexnější layout se dvěma TextView a třemi Buttons.

Obrázek 6.3: Náhled jedné položky ListView ve správě lekcí

Jeden z problémů se ukázal být v odchytávání kliknutí. Podařilo se zajistit kliknutí na celou položku, nebo jen na tlačítka, ale nikdy ne na obě zároveň. Nakonec se ukázalo, že tlačítko musí mít v XML layoutu nastaven parametr:

```
android:focusable="false"
```

Tímto je zajištěno, že layout neodchytává kliknutí pouze na konkrétní tlačítko, ale na celek.

6.9 Vykreslování grafů

Zobrazení výsledků každého zkoušení je realizováno pomocí grafu. Jeden graf slouží k přehledu o aktuálním zkoušení. Umisťuje slova do skupin podle toho, kolikrát bylo každé z nich zodpovězeno špatně. Tuto statistiku je možné zobrazit pomocí sloupcového nebo výsečového grafu. Po klepnutí na konkrétní sloupec nebo výseč grafu se zobrazí příslušná slova. Tato slova je možno exportovat do zvláštního souboru. Druhý graf slouží k dlouhodobému přehledu zkoušení jedné lekce. Zobrazuje procentuální úspěšnost v pěti posledních testech. Tyto údaje jsou vykresleny pomocí spojnicového grafu.

K zajištění této funkčnosti byla použita knihovna *HoloGraph*. Umožňuje vytvářet všechny tři typy grafů a její použití je velice jednoduché. Knihovna je open-source, tudíž je k dispozici zdrojový kód, který je třeba importovat jako samostatný projekt. Ten se v nastavení cílového projektu pouze přiřadí jako knihovna a poté je možné používat jeho funkce a třídy.

Kapitola 7

Testování

V této kapitole bude popsáno testování aplikace během vývoje a závěrečné testování uživateli. Budou zmíněny větší chyby, které se během totoho procesu objevily, a shrnutí uživatelských dotazníků.

7.1 Testovací zařízení

Testování probíhalo zejména na následujících zařízeních:

- Huawei Ascend G300 (dále jen Huawei)
- Prestigio MultiPad 9.7 Ultimate (dále jen Prestigio)

Tabulka 7.1: Technické parametry testovacích zařízení

	Huawei	Prestigio
Velikost displeje	4"	10.1"
Verze OS	4.0.3	4.1.1
Frekvence procesoru	1 GHz	1.6 GHz
Počet jader procesoru	1	2
Operační paměť	512 MB	1024 MB

7.2 Testování funkčnosti

Asi nejdůležitějším typem testování je testování funkčnosti. Během vytváření aplikace jich bylo prováděno několik, a to vždy po vytvoření funkčně celistvého bloku kódu. Byly prováděny zátěžovými testy na aplikaci a následovalo odstraňování chyb. Během testování se objevily následující nedostatky:

- **V režimu *test* se zobrazilo stejné slovo u více možností**
  Tato chyba se samozřejmě více objevovala při testování lekce s malým

počtem slov. Problém byl vyřešen přidáním následujícího bloku kódu, který se stará o výběr špatných možností. Proměnná `position` představuje pole indexů slov, které budou použity jako špatné odpovědi.

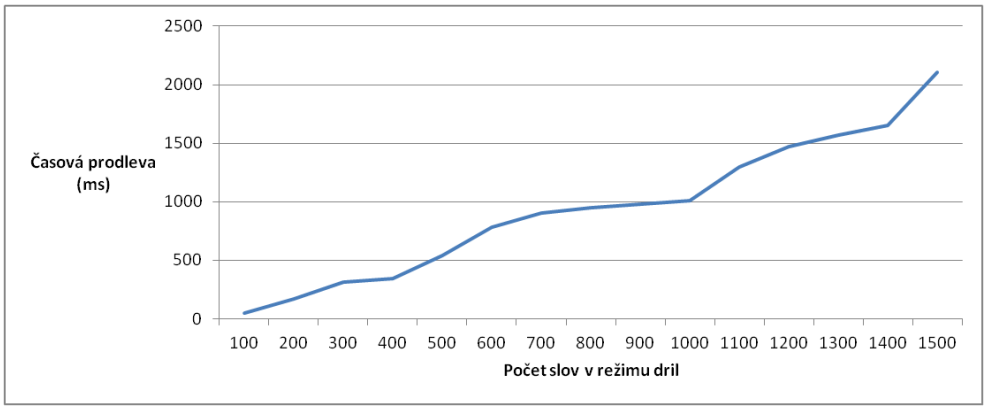
```
for (int i = 0; i < OPTIONS_NR - 1; i++)
    do {
        position[i] = generator.nextInt(allWords.
            size());
        for (int j = 0; j < i; j++) {
            while (position[i] == position[j])
                position[i] = generator.nextInt(
                    allWords.size());
        }
    } while ((allWords.get(position[i]).getCzech())
        .equals(pickedWord.getCzech()));
    ...
}
```

- **Špatné počítání výslených procent po uložení a znovunačtení lekce**
  Po uložení a znovunačtení lekce se výsledná procentuální úspěšnost začala počítat od začátku. Chybělo tedy uchování této hodnoty i po přerušení zkoušení. Problém byl vyřešen zjištěním počtu celkově testovaných slov z uloženého souboru před finálním předáním hodnoty aktivitě zobrazující výsledky, namísto předávání informace o počtu testovaných slov pouze mezi aktivitami.

- **Ošetření nesprávného formátu importovaného souboru**
  Při prvním testování importu nebylo provedeno ošetření špatného formátu souboru. Nyní aplikace nepřijme soubor, který je buď prázdný, nebo nemá žádný řádek ve správném formátu. V opačném případě načte všechny správné řádky a vytvoří nový XML soubor reprezentující nově vloženou lekci.

7.3 Testování rychlosti

Aplikace byla dále testována zátěžovými testy, které spočívaly ve zkoušení velkého množství slov najednou. Graf závislosti prodlevy mezi zobrazením následujícího slova v režimu drilu na počtu testovaných slov k nahlédnutí na obrázku 7.1. Podle křivky s exopnenciálním růstem lze vyvodit, že pro pohodlný běh aplikace je vhodné použít maximálně jeden tisíc slov v jednom zkoušení. Příčinu tohoto zpomalení je možné hledat ve faktu, že při každém odstranění aktivity jsou mazány všechny aktuální kolekce slov, a tím vzniká časová prodleva.



Obrázek 7.1: Graf zobrazující závislost počtu testovaných slov na časové prodlevě mezi aktivitami

7.4 Testování uživateli

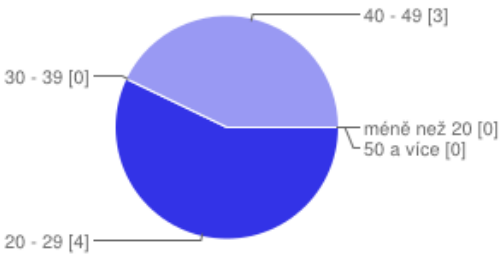
Finální testování bylo provedeno několika respondenty. Každý dostal minimální doporučený scénář průchodu aplikací a následně vyplnil dotazník. Většina z nich aplikaci nikdy před tím neviděla a ani nedostala žádné pokyny, jak se v ní orientovat. Doporučený scénář obsahoval následující položky:

- Vybrat si libovolnou lekci a otestovat ji režimem Dril.
- Vybrat si množinu slov, podle četnosti špatné odpovědi, a tu následně exportovat (např. exportovat slova, která byla zodpovězená chybně dvakrát).
- Importovat tuto množinu slov zpět do aplikace.
- Změnit název této nové lekce.
- Lekci následně otestovat, ale nejdříve změnit obtížnost testu.

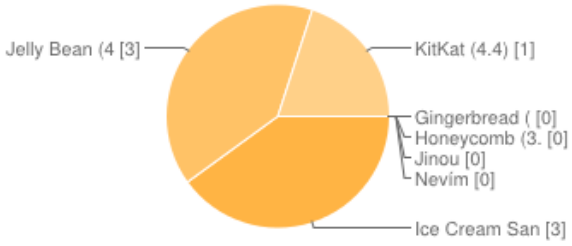
7.4.1 Dotazník

Všem respondentům byly položeny následující otázky:

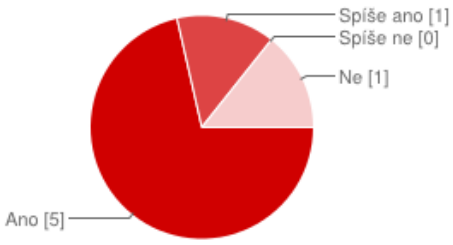
Kolik je Vám let?



Jakou máte verzi Androidu?



Bylo podle Vás uživatelské rozhraní intuitivní?

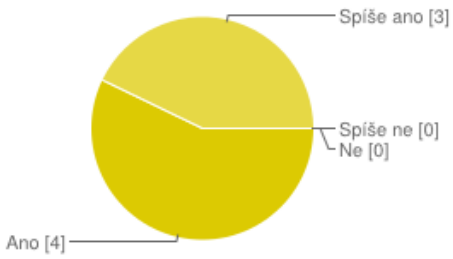


Pokud jste na předchozí otázku odpověděli negativně, uveďte prosím, kde jste měli problém.

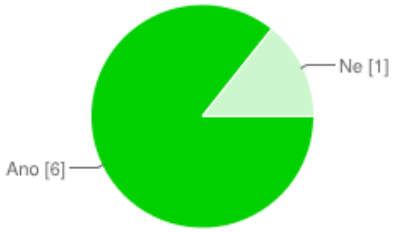
- *„Při prvním spuštění (režim test lekce 1) mělo být 68 slovíček, ale zobrazilo se jich víc (např. u slovíčka číslo 66/68 se muselo zodpovědět třeba 6 slovíček, aby se to posunulo na číslo 67/68). Pak se mi také*

někam uložil exportovaný soubor chybně zodpovězených slovíček a už se mi ho nepodařilo najít (název souboru to ukázalo jen na chvíli a přejmenování mi to nenabídlo) a aplikace mi potom nešla ukončit (pořád byla na displeji a nedala se zavřít).“

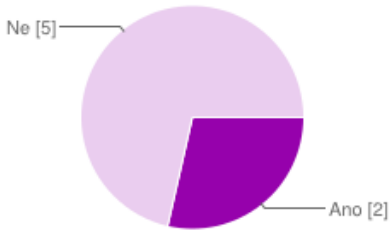
Líbí se Vám aplikace po grafické stránce?



Umíte si představit každodenní používání této aplikace?



Používáte, nebo jste v minulosti používali, jinou výukovou aplikaci pro učení cizího jazyka?



Pokud jste našli v aplikaci chybu, máte nějaké připomínky nebo vzkazy, můžete je vyjádřit zde.

- „*Viz výše, ale kdyby byla aplikace bez problémů, určitě by se našlo dost lidí, kteří by ji chtěli využívat*“

- *„Možná by bylo dobré zvětšit ikonky ve správě lekcí - „upravit“ a „odstranit““*

7.4.2 Shrnutí

Z dotazníků je možné vyčíst, že většina testujících osob je ve věkové kategorii 20-29 let nebo 40-49 let. Všichni mají verzi Androidu alespoň 4.0, což je dobrá zpráva, protože minimum SDK je nastaveno právě na 4.0.

Většina respondentů odpověděla na otázku „Bylo podle Vás uživatelské rozhraní intuitivní?“ kladně, a tak je možné usoudit, že uživatelské rozhraní je navrženo správně a neobsahuje žádně větší nejasnosti.

Testování uživateli však poukázalo i na několik nedostatků, které aplikace stále má. Poskytli tak velice ceněnou zpětnou vazbu. V aplikaci byly na základě těchto poznatků následně opraveny některé z těchto nedostatků, jako například zvětšení ikon, nebo zaměření na větší komunikaci aplikace s uživatelem tak, aby vždy věděl, co se děje.

Kapitola 8

Výhled do budoucna

Aplikace má stále mezery, které se nabízejí k dalšímu pokračování. Jedním z takových míst je podpora výslovnosti. Bylo by zajímavé, kdyby si uživatel mohl z reproduktorů přehrát výslovnost. Ještě ambiciózněji myšlenka by byla, aby uživatel mohl být rovnou zkoušen tak, že by správné odpovědi říkal do mikrofonu. Další možné rozšíření, které ale trochu odporuje jednomu z požadavků, je stahování slovníkových sad z Internetu, sdílení svých výsledků na sociálních sítích, a porovnávání tak výsledků s přáteli.

Závěr

Cílem této práce bylo seznámení se s existujícími řešeními výukových aplikací zaměřených na výuku cizího jazyka. Byla provedena rešerše těchto aplikací a vyvození závěru.

- Jeden z nedostatků se ukázal být v tom, že téměř žádné aplikace nepodporují výuku pomocí testu (výběr správného slova z více možností).
- Sporadicky se vyskytuje možnost importu/exportu textových souborů, a vytváření tak vlastního obsahu.
- Některé aplikace vyžadují připojení k Internetu, což je pro uživatele mnohdy nepraktické.

Na základě tohoto pozorování byly finálně stanoveny minimální funkční požadavky.

- Alespoň tři typy zkoušení (dril, test, slovník)
- Import a export souboru
- Grafické zpracování výsledků
- Možnost uložení rozpracované lekce

Po analýze požadavků byl proveden návrh uživatelského rozhraní. Byly zachovány standardní ovládací prvky Androidu (Action Bar) a vytvořeny vlastní ikony a pozadí obrazovek. Při tom byl kladen důraz na minimální požadované rozlišení vytvářených obrázků. Screenshoty současné podoby aplikace jsou k nahlédnutí v příloze B.

Při implementaci aplikace byla dodržována standardní struktura projektu včetně uzavírání dílčích skupin zdrojových souborů do samostatných balíků. Byly vytvořeny zdrojové soubory v jazyce Java a soubory definující uživatelské rozhraní v jazyce XML. Důraz byl kladen hlavně na implementaci jednotlivých typů zkoušení. Režim *dril* a *test* umožňují uložení rozpracované lekce, export souboru a sdílení obsahu ostatním aplikacím. Režim *slovník* umožňuje vyhledávání slov napříč všemi vybranými lekcemi.

Aplikace byla následně otestována několika uživateli. Většina z nich viděla aplikaci poprvé a díky tomu bylo testování věrohodnější. Testujícím osobám bylo uloženo pár úkolů k provedení v aplikaci, přičemž ani jedna neměla sebemenší problém úkoly splnit.

V současné době aplikace ještě není umístěna na Google Play, ale v nejbližší době k tomu dojde.

Literatura

[1] Krimsky, J.: *Angličtina slovíčka* [mobilní aplikace] [cit. 2014-4-17]. Dostupné z: `https://play.google.com/store/apps/details?id=info.krimsky.bvs`

[2] Eddica multimedia: *Angličtina - mobilní učitel* [mobilní aplikace] [cit. 2014-4-17] Dostupné z: `https://play.google.com/store/apps/details?id=eddica.demo.domaciucitel.ang.cz`

[3] Jurkovič, P.: *Dril - angličtina efektivně* [mobilní aplikace] [cit. 2014-4-17] Dostupné z: `https://play.google.com/store/apps/details?id=sk.peterjurkovic.dril`

[4] Manuel, O.: *Angličtina pokročilé zásoby* [mobilní aplikace] [cit. 2014-4-17] Dostupné z: `https://play.google.com/store/apps/details?id=com.oman.ln4kids`

[5] Google Inc., Android [online] [cit. 2014-4-19] Dostupné z: `http://developer.android.com/about`

[6] IDC, *Worldwide smartphon platforms market share, 2013* [online] [cit. 2014-4-19] Dostupné z: `http://www.icharts.net/chartchannel/worldwide-smartphone-platforms-market-share-2013-q2-unit-share_m33qyshgc`

[7] Campuswhizz: *Android versions* [obrázek]. In: *Android and All Its Versions* [online] [cit. 2014-4-19] Dostupné z: `http://www.campuswhizz.in/cwzz/android-versions/`

[8] Android Developers Blog: *Android 1.5 is here!* [online] [cit. 2014-4-19] Dostupné z: `http://android-developers.blogspot.cz/2009/04/android-15-is-here.html`

[9] Open Handset Alliance: *úvodní strana* [online] [cit. 2014-4-19] Dostupné z: `http://www.openhandsetalliance.com/`

[10] VisualPharm: *Choosing the Right Size and Format for Icons* [online] [cit. 2014-4-20] Dostupné z: `http://visualpharm.com/articles/icon_sizes.html`

[11] Google Inc., Android Design: *Metrics and Grids* [online] [cit. 2014-4-22] Dostupné z: http://developer.android.com/design/style/metrics-grids.html

[12] Google Inc., Android Reference: *ListView* [online] [cit. 2014-4-22] Dostupné z: http://developer.android.com/reference/android/widget/ListView.html

[13] Google Inc., Android API Guides: *Fragments* [online] [cit. 2014-4-22] Dostupné z: http://developer.android.com/guide/components/fragments.html

[14] Google Inc., Android Reference: *Properties* [online] [cit. 2014-4-22] Dostupné z: http://developer.android.com/reference/java/util/Properties.html

[15] Google Inc., Android Training: *Parsing XML data* [online] [cit. 2014-4-22] Dostupné z: http://developer.android.com/training/basics/network-ops/xml.html

[16] The Eclipse Foundation. *Eclipse Standard 4.3.2* [software]. [přístup 22. dubna 2014]. Dostupné z: https://www.eclipse.org/downloads/packages/eclipse-standard-432/keplersr2

[17] Google Inc., Android API Guides: *Action Bar* [online] [cit. 2014-4-23] Dostupné z: http://developer.android.com/guide/topics/ui/actionbar.html

[18] Google Inc., Android Reference: *ShareActionProvider* [online] [cit. 2014-4-23] Dostupné z: http://developer.android.com/reference/android/widget/ShareActionProvider.html

[19] Andr.oid Eric. Android-er: *Implement a simple File Explorer in Android* [online] [cit. 2014-4-24] Dostupné z: http://android-er.blogspot.cz/2010/01/implement-simple-file-explorer-in.html

[20] Havryluk, M.: *Představení platformy Android*. ČVUT, 2014 [cit. 2014-4-26] Dostupné z: https://edux.fit.cvut.cz/courses/BI-AND/_media/lectures/01.pdf

[21] Google Inc., *The activity lifecycle* [obrázek] In: *Android API Guides: Activities* [online] [cit. 2014-4-26] Dostupné z: http://developer.android.com/guide/components/activities.html

[22] Google Inc., *Get the SDK* [online [cit. 2014-05-02] Dostupné z: http://developer.android.com/sdk

Příloha **A**

Seznam použitých zkratek

ADT Android development tools

API Appliaction programming interface

ARM Advanced RISC machine

dip density-independent pixel

GUI Graphical user interface

HW Hardware

OS Operační systém

RISC Reduced instruction set computing

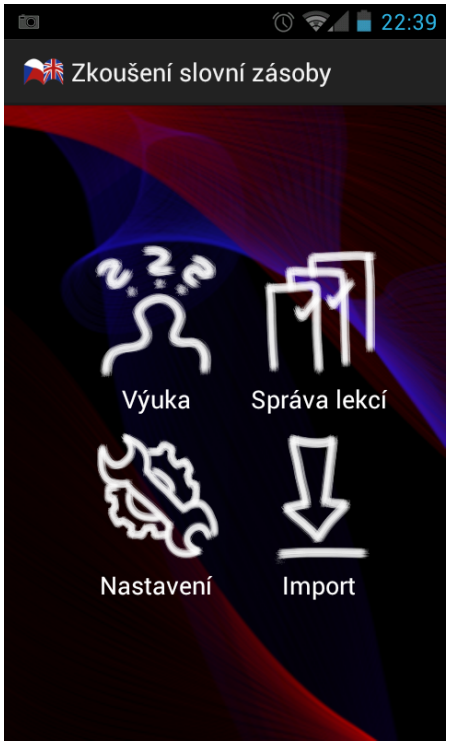
SDK Software development kit

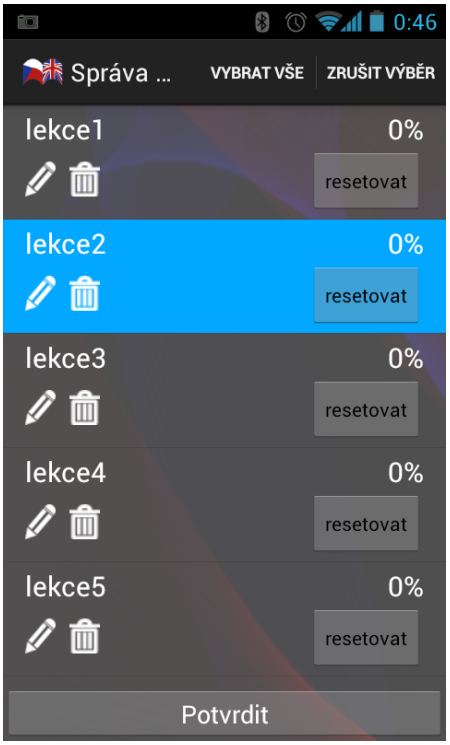
SW Software

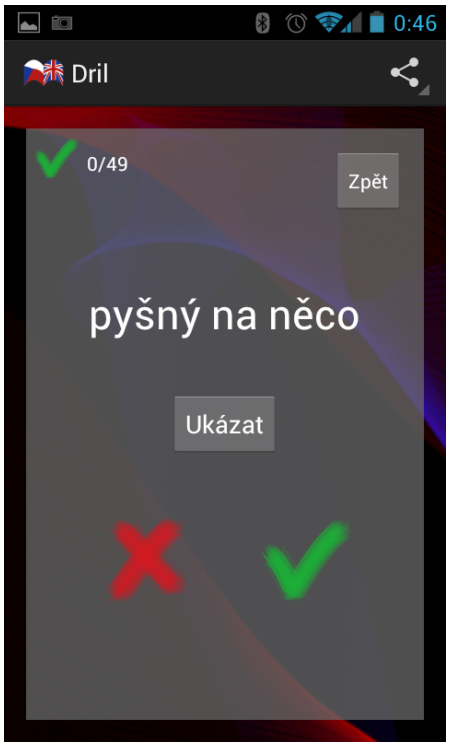
XML Extensible markup language

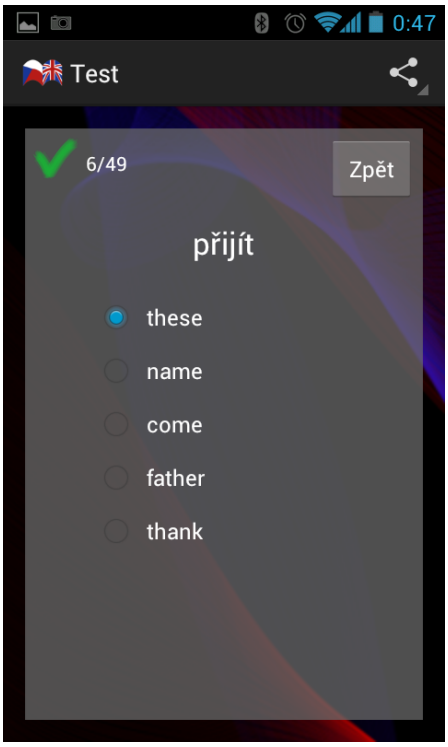
Příloha B

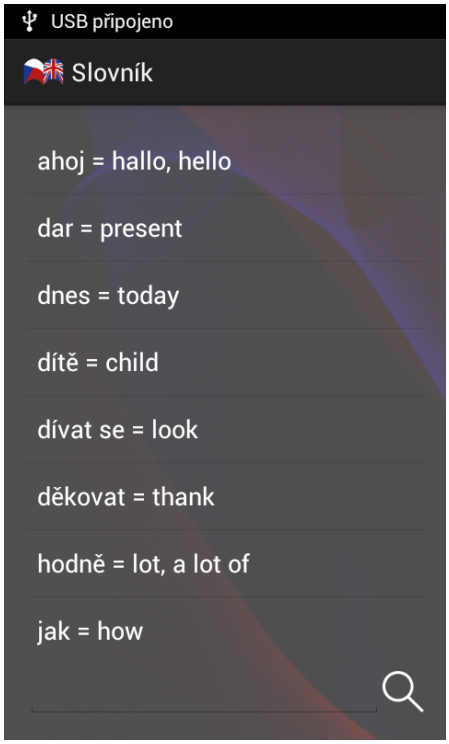
Screenshoty






0:46
Dril
0/49
Zpět
pyšný na něco
Ukázat


0:47
Test
6/49
Zpět
přijít
these
name
come
father
thank


USB připojeno
Slovník
ahoj = hallo, hello
dar = present
dnes = today
dítě = child
dívat se = look
děkovat = thank
hodně = lot, a lot of
jak = how


10:16
ChartResultActivity
AKTUÁLNÍ CHYBOVOST
DLOUHODOBĚ
34
12
4
0
0
0
1
2
3
4+
Tento graf popisuje statistiku chybovosti. Ve spodním řádku jsou uvedeny počty chyb u jednoho slova. Pro zobrazení chybovaných slov stiskněte požadovaný sloupec.
Jiný typ grafu

Příloha C

Instalační a uživatelská příručka

C.1 Požadavky na systém

Minimální požadovaný systém pro správný běh aplikace je Android OS, verze 4.0 (Ice Cream Sandwich).
Aplikace není výpočetně ani graficky náročná, a tak by mělo jakékoliv zařízení s uvedeným operačním systémem vyhovovat.

C.2 Instalace aplikace

- Před instalací aplikace je v zařízení nutné povolit možnost *Neznámé zdroje* v sekci *Nastavení* → *Zabezpečení*.
- Soubor *zkousenislovnizasoby.apk* umístěte kamkoliv do paměti zařízení.
- Přímo v zařízení poté klepněte na instalační soubor. Následně se zobrazí nabídka instalace.

C.3 První spuštění aplikace

Pokud chcete po prvním spuštění přistoupit rovnou k výuce, musíte nejprve aktivovat lekci ke zkoušení. To lze provést v sekci *Hlavní nabídka* → *Správa lekcí* klepnutím na požadovanou lekci. Pokud položka zmodrá, lekce je vybrána. K výuce můžete vybrat libovolný počet lekcí naráz. Následně stiskněte tlačítko *Potvrdit* a dále již můžete přistoupit k výuce.

C.4 Výuka

Aplikace podporuje tři různé typy výuky. Vyberte si jeden z nich kliknutím na sekci *Výuka* v hlavní nabídce.
Výuka režimem *Dril* nebo *Test* probíhá do té doby, dokud nezodpovíte všechna

slova správně. Na konci zkoušení se zobrazí grafy úspěšnosti a možnost lekci opakovat, nebo vybrat jinou ve *správě lekcí*.
Pro opuštění výuky během zkoušení stiskněte tlačítko *Zpět*. Zobrazí se možnost uložit aktuální stav zkoušení a vrátíte se do hlavní nabídky.

C.5 Spravování lekcí

V sekci *Správa lekcí* je možné provádět několik operací s vloženými lekcemi. Jsou k dispozici následující funkce:

- *Upravit lekci*

  Tato možnost se zobrazí po kliknutí na ikonu tužky. Umožňuje danou lekci přejmenovat.

- *Odstranit lekci*

  Tato možnost se zobrazí po kliknutí na ikonu popelnice. Umožňuje danou lekci odstranit.

- *Resetovat lekci*

  Tuto možnost lze provést stisknutím tlačítka *Resetovat*. Pokud byla daná lekce rozpracována, její rozpracovanost se nastaví na 0 %.

C.6 Import souboru

Pokud si do aplikace přejete vložit vlastní soubor s lekcí, postupujte následovně:

- V libovolném textovém editoru vytvořte soubor, který bude dodržovat následující formát:
  na každém řádku musí být *cizí slovo=české slovo* (nahraďte) a následné odřádkování. Dejte si pozor, aby soubor měl koncovku `.txt` a byl uložen v kódování `UTF-8`.

- Umístěte tento soubor do zařízení.

- Stiskněte sekci *Import* v hlavní nabídce a vyberte soubor z předchozího umístění.

- Aplikace následně indikuje, zda k importu souboru došlo, či nikoli. Soubor je ke zkoušení možné vybrat v sekci *Správa lekcí*.

C.7 Export souboru

Export souboru je v aplikaci možné provést dvěma způsoby:

- *Export rozpracované lekce*

  Tato možnost je k dispozici kdykoliv v průběhu zkoušení. Provede se stisknutím možnosti *Export*, která je ukrytá v *Overflow* menu (tři tečky nebo čárky pod sebou).
  V tomto případě bude exportovaný soubor dodržovat následující formát:

  Known words:
  cizí slovo=české slovo
  cizí slovo=české slovo
  ...

  Mistaken words:
  cizí slovo=české slovo
  ...

- *Export části lekce po dokončení zkoušení*

  Tato možnost je k dispozici po kliknutí na položku *Zobrazit graf*, která se objeví po úspěšném dokončení drilu nebo testu. Zobrazí se sloupcový graf, kde je možné kliknout na jeden ze sloupců, a příslušná slova exportovat.

  Tento soubor splňuje požadavky na importovaný soubor, a tak je možné ho následně do aplikace vložit jako novou lekci.

Příloha D

Implementace XML parseru

```
package com.example.slovnik1_0;

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import android.util.Xml;
import com.example.slovnik1_0.data.Word;

/**
 * Class, that implements the vocabulary file parser
 *
 * @author Eva Mayerova
 */
public class XmlParser {

    private static String namespace = null;

    public ArrayList<Word> parse(InputStream in) throws
       XmlPullParserException,
            IOException {
        try {
            XmlPullParser parser = Xml.newPullParser();
            parser.setFeature(XmlPullParser.
               FEATURE_PROCESS_NAMESPACES, false);
            parser.setInput(in, null);
            try {
                parser.nextTag();
```

```
            } catch (XmlPullParserException e) {
                Log.d("MESSAGE", e.getMessage());
            }
            return readVocabulary(parser);
        } finally {
            in.close();
        }
    }

    private ArrayList<Word> readVocabulary(XmlPullParser
        parser)
            throws XmlPullParserException, IOException {
        ArrayList<Word> results = new ArrayList<Word>();
        try {
            parser.require(XmlPullParser.START_TAG,
                namespace, "vocabulary");
        } catch (XmlPullParserException e) {
            String message = e.getMessage();
            Log.d("MESSAGE", message);
        }
        while (parser.next() != XmlPullParser.END_TAG) {
            if (parser.getEventType() != XmlPullParser.
                START_TAG) {
                continue;
            }
            String name = parser.getName();
            if (name.equals("word")) {
                results.add(readWord(parser));
            } else {
                skip(parser);
            }
        }
        return results;
    }

    private Word readWord(XmlPullParser parser) throws
        XmlPullParserException,
            IOException {
        parser.require(XmlPullParser.START_TAG,
            namespace, "word");
        String czech = null;
        String english = null;
        String state = null;
        boolean tested = false;
```

```
        float timeDelay = 0;
        int failures = 0;

        while (parser.next() != XmlPullParser.END_TAG) {
            if (parser.getEventType() != XmlPullParser.
               START_TAG)
                continue;
            String name = parser.getName();
            if (name.equals("czech")) {
                czech = readCzech(parser);
            } else if (name.equals("english")) {
                english = readEnglish(parser);
            } else if (name.equals("tested")) {
                tested = readTested(parser);
            } else if (name.equals("timeDelay")) {
                timeDelay = readDelay(parser);
            } else if (name.equals("failures")) {
                failures = readFailures(parser);
            } else if (name.equals("state")) {
                state = readState(parser);
            } else
                skip(parser);
        }
        return new Word(czech, english, state, tested,
           failures, timeDelay);
    }

    private String readCzech(XmlPullParser parser)
            throws XmlPullParserException, IOException {
        parser.require(XmlPullParser.START_TAG,
           namespace, "czech");
        String czech = readText(parser);
        parser.require(XmlPullParser.END_TAG, namespace,
            "czech");
        return czech;
    }

    private String readEnglish(XmlPullParser parser)
            throws XmlPullParserException, IOException {
        parser.require(XmlPullParser.START_TAG,
           namespace, "english");
        String english = readText(parser);
        parser.require(XmlPullParser.END_TAG, namespace,
            "english");
```

```
        return english;
    }

    private String readState(XmlPullParser parser)
            throws XmlPullParserException, IOException {
        parser.require(XmlPullParser.START_TAG,
            namespace, "state");
        String st = readText(parser);
        parser.require(XmlPullParser.END_TAG, namespace,
            "state");
        return st;
    }

    private boolean readTested(XmlPullParser parser)
            throws XmlPullParserException, IOException {
        parser.require(XmlPullParser.START_TAG,
            namespace, "tested");
        String tested = readText(parser);
        parser.require(XmlPullParser.END_TAG, namespace,
            "tested");
        if (tested.equals("true"))
            return true;
        else
            return false;
    }

    private float readDelay(XmlPullParser parser)
            throws XmlPullParserException, IOException {
        parser.require(XmlPullParser.START_TAG,
            namespace, "timeDelay");
        float delay = readFloat(parser);
        parser.require(XmlPullParser.END_TAG, namespace,
            "timeDelay");
        return delay;
    }

    private int readFailures(XmlPullParser parser)
            throws XmlPullParserException, IOException {
        parser.require(XmlPullParser.START_TAG,
            namespace, "failures");
        int fails = readInt(parser);
        parser.require(XmlPullParser.END_TAG, namespace,
            "failures");
        return fails;
```

```
}

private void skip(XmlPullParser parser) throws
   XmlPullParserException,
        IOException {
    if (parser.getEventType() != XmlPullParser.
       START_TAG)
        throw new IllegalStateException();

    int depth = 1;
    while (depth != 0) {
        switch (parser.next()) {
        case XmlPullParser.END_TAG:
            depth--;
            break;
        case XmlPullParser.START_TAG:
            depth++;
            break;
        }
    }
}

private String readText(XmlPullParser parser)
        throws XmlPullParserException, IOException {
    String result = "";
    if (parser.next() == XmlPullParser.TEXT)
        result = parser.getText();
    parser.nextTag();
    return result;
}

private int readInt(XmlPullParser parser)
        throws XmlPullParserException, IOException {
    String result = "";
    if (parser.next() == XmlPullParser.TEXT)
        result = parser.getText();
    parser.nextTag();
    return Integer.parseInt(result);
}

private float readFloat(XmlPullParser parser)
        throws XmlPullParserException, IOException {
    String result = "";
    if (parser.next() == XmlPullParser.TEXT)
```

```
            result = parser.getText();
        parser.nextTag();
        return Float.parseFloat(result);
    }
}
```

Příloha **E**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `ZkouseniSlovniZasoby.zip` ..... projekt kompatibilní s Eclipse
    - `HoloGraph.zip` ..... knihovna Holograph kompatibilní s Eclipse
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Mayerová_Eva_2014.pdf` ..... text práce ve formátu PDF