VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

APLIKACE PRO ZÁZNAM FOTEK A VIDEOKLIPŮ OVLÁDANÁ HLASEM

VOICECONTROLLED APP FOR CAPTURING PHOTOS AND CLIPS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ADAM DALIBOR JURČÍK**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

162131

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Jurčík Adam Dalibor**
Program: Informační technologie
Název: **Aplikace pro záznam fotek a videoklipů ovládaná hlasem**
Kategorie: Uživatelská rozhraní
Akademický rok: 2023/24

Zadání:

1. Seznamte se s problematikou vývoje mobilních aplikací (zaměřte se na platformu Android) a jejich ovládání hlasem.
2. Vyhledejte, prostudujte a popište vhodné knihovny pro ovládání mobilních aplikací hlasem.
3. Vyhledejte a popište techniky pořizování fotografií a krátkých videoklipů na zvolené mobilní platformě.
4. Experimentujte s vhodnými knihovnami. Zhodnoťte jejich použitelnost pro ovládání aplikace pro pořizování obrazu a videa hlasem.
5. Vytvořte demonstrační aplikaci (aplikace) ukazující použitelnost ovládání aplikací hlasem ve vhodných scénářích pořizování fotografií a krátkých videoklipů.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování výsledků projektu.

Literatura:

- Tidwell et al.: Designing Interfaces: Patterns for Effective Interaction Design, O'Reilly, 2020
- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016
- Android Developers: https://developer.android.com/index.html

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

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 31.7.2024
Datum schválení: 8.7.2024

Abstrakt

Cílem této práce je vytvořit aplikaci, která řeší problémy spojené s nahráváním videa a fotografií téže osoby při sportovních aktivitách. Těmito problémy jsou například neustálá potřeba přítomnosti dálkového ovbládání nebo přerušování činnosti z důvodu ručního spuštění. Situaci lze vyřešit pomocí hlasového ovládání a klíčových slov, po kterých se spustí pořízení fotografie. V práci je podrobně rozebráno, jakým způsobem lze implementovat hlasové ovládání na operačním systému Android v programovacím jazyce Java dohromady s pořízením snímku či videa. Dále vysvětluje ukládání těchto souborů a ovládání hardware zařízení, například fotoaparátu. Práce podrobněji popisuje dva přístupy k rozpoznání hlasu a jejich funkčnost v různých způsobech využití.

Abstract

The goal of this work is to create an application that solves the problems of video and photo recording while the same person is playing sports. These problems are, for example, the need to carry a remote control or the constant interruption of the activity due to manual activation. This is solved by using voice control and keywords to trigger the taking of a photo. This paper discusses in detail how voice control can be implemented on the Android operating system in Java programming language. It also includes an explanation of how to store these files and how to control hardware devices such as the camera. The thesis includes a more detailed description of two approaches to voice recognition and describes their functionality in different applications.

Klíčová slova

Android, rozpoznaní hlasu, keyword spotting, Java, sport, fotografie, video, aplikace, Google Play, ovládání hlasem

Keywords

Android, voice recognition, keyword spotting, Java, sport, photo, video, application, Google Play, voice control

Citace

JURČÍK, Adam Dalibor. *Aplikace pro záznam fotek a videoklipů ovládaná hlasem.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Aplikace pro záznam fotek a videoklipů ovládaná hlasem

Prohlášení

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

.....
Adam Dalibor Jurčík
28. července 2024

Poděkování

Rád bych poděkoval panu prof. Ing. Adamu Heroutovi, Ph.D. za vedení práce, jeho připomínky a rady, které mi pomohly při tvorbě této bakalářské práce. Dále všem těm, co se zapojili do testování aplikace a jejich zpětnou vazbu.

Obsah

Kapitola 1

Úvod

Cílem této práce je vytvořit alternativní fotografickou aplikaci ovládanou hlasem, jež bude mít primární využití při sportovních aktivitách a pořizování skupinových fotografií. Hlavním úkolem bude umožnit uživateli ovládat telefon na dálku, kdy bude zařízení například ve stativu. V dnešní době se používají *bluetooth* spouště, gesta nebo zpoždění časovačem. Během sportovních aktivit tlačítko překáží, proto má tato aplikace sportovce zbavit potřeby mít k dispozici sportovního partnera, který by jej natočil. Pomocí hlasu bude sám ovládal záznam videa a fotografií, blesk a další funkce v aplikaci ovládající integrovanou kameru. Řešení bude mít v sobě i časovač, který odpočítá spuštění záznamu a poté vydá zvuk při ukončení. Následně se vytvořený soubor uloží do galerie, kde si fotografie a videa může uživatel prohlédnout. Implementace projektu bude v jazyce Java a využije automatické sestavení pomocí nástroje Gradle spolu s pluginem Groovy. Pro rozpoznání hlasu použije již existující knihovny, ze kterých následně vyberu tu nejvhodnější vzhledem k tomuto specifickému řešení.

Otestuji detailně tři knihovny, primárně v angličtině. Aplikace bude fungovat bez internetu a na zařízení s operačním systémem Android. Publikuji ji na Obchod Play, bude tedy k dispozici komukoli *online* ke stažení. Cílem je vybrat nejlepší možnou knihovnu pro další vývoj a použitelnost aplikace. Vhodná knihovna by měla být rychlá, přesná a co nejvíce přizpůsobitelná. Uživatel si bude moci nastavit svá klíčová slova pro dané funkce.

V úvodu práce vysvětluji fungování rozpoznání hlasu a nabídku funkcí, poté přibližuji tvorbu lidského hlasu. V další kapitole se věnuji vývoji aplikací na operačním systému Android. Uvedu, jakým způsobem vydat aplikaci na Obchod Play, a zmíním možnosti, které není vhodné použít. Návrhu aplikace se věnuji podrobněji, jelikož aplikaci lze naprogramovat i na jiné platformy. Následně představím knihovny, jež otestuji, a popíši implementaci zachycení fotografie a videa. Nakonec shrnu průběh testovaní s uživateli, a představím výsledky hlasového rozpoznání.

Kapitola 2

Rozpoznání hlasu

V dnešní době je tato technologie velmi pokročilá. Nabízí mnoho funkcí, jež lze využít v telefonech, automobilech i všude tam, kde se využívá výpočetní technika. Důležitým prvkem úspěšného rozpoznání hlasu je kvalitní mikrofon, který dokáže zachytit a přenést zvuky do zařízení s co nejmenším zkreslením. Moderní algoritmy využívající strojové učení umožňují systému přizpůsobit se individuálním hlasovým charakteristikám uživatele, čímž se zvyšuje přesnost a spolehlivost rozpoznání hlasu. Další inovací v této oblasti je integrace technologie rozpoznání hlasu do chytrých zařízení, jako jsou domácí asistenti (například Amazon Echo či Google Home). Tito asistenti umožňují ovládání domácích spotřebičů, nastavení připomínek nebo objednání zboží pouhým hlasovým příkazem. Moje práce se věnuje rozpoznání hlasu v mobilních zařízeních.

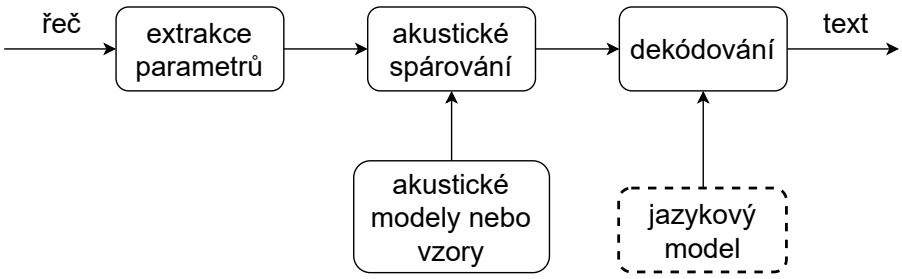
Neuronové sítě mají schopnost adaptovat se na různorodé řečové vzory a lépe zvládají složité situace, což přispívá k vyšší přesnosti rozpoznání. Klíčovým faktorem je také rychlost, zejména ve chvílích, kdy uživatelé očekávají okamžitou reakci. Proto je nezbytné, aby rozpoznávací modely dosahovaly co nejkratší doby odezvy, aniž by to ovlivnilo jejich přesnost. Díky stále se zvyšující výpočetní síle a objemu dostupných dat jsou moderní rozpoznávací systémy schopny lépe porozumět a adaptovat se na různorodost lidských hlasů, dialektů a akcentů. Slovníky jsou rozsáhlejší díky větší paměti. Některé vývojářské sady dokonce zasílají data na server kvůli důkladnějšímu zpracování. Napříč odvětvími a aplikacemi existují odlišné požadavky na rozpoznávání hlasu. Důležité je, aby rozpoznávací systémy byly flexibilní a modulární, aby mohly být přizpůsobeny specifickým potřebám a použitím.

Kromě samotného hlasového rozpoznávání mohou moderní systémy nabízet také další funkce. Například překlad do různých jazyků, generování textu na základě mluveného slova nebo identifikaci emocí z hlasu. Některá řešení dokonce nabízí i rozpoznání hlasu uživatele. Když vlastník zařízení řekne „Ok Google“, aplikace zareaguje. Pokud se cizí člověk pokusí říct stejný příkaz, zařízení nejeví žádnou známku reakce.

2.1 Princip rozpoznání hlasu

Při rozpoznání hlasu se využívají různé znalosti z mnoha vědních oborů. Jedná se o multidisciplinární přehled přírodních, technických i humanitních věd. Vzhledem k vytváření různých modelů se jedná o fyziologii, kde musí vývojáři pochopit funkci a propojenost sluchového a hlasového ústrojí. Další problematikou je akustika, ve které jsou fyzikální mechanismy prioritou kvůli tvorbě a slyšení řeči. Dále se dostáváme ke zpracování signálu, při kterém se využívá mnoho oblastí, jako je modelování, spektrální analýza anebo klasifi-

kace vzorů. Klíčová je především fonetika zaměřující se na zvukové slyšení hlasu. Fonologie studuje zvukovou stránku jazyka, jehož součástí jsou tzv. fonémy[1]. V českém jazyce je 36 fonémů, v anglickém až 42. Prozódie rozebírá zvukovou stránku jazyka, jedná se o přízvuk a melodii a jiné zbarvení hlasu. Cíleněji na to poukáži v kapitole 2.2. Spolupráce s lingvisty a odborníky, kteří se zabývají vývojem a zpracováním lidské řeči, může pomoci lépe porozumět jazykovým nuancím a specifikům. To vede k vylepšení přesnosti zpracovávání a porozumění rozpoznávacích systémů.



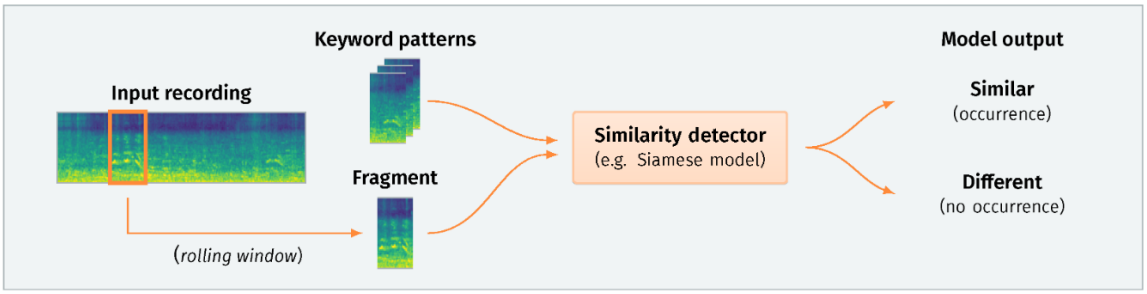
Obrázek 2.1: Schéma nejzákladnějšího rozpoznávače hlasu. Při vstupu řeči proběhne extrakce parametrů a vlastností zvuku, následně se vyhledává možná shoda, kdy akustický model produkuje odměny či body, jež se dále vyhodnotí. Posledním procesem je dekódování, kde se výsledky hypotézy přepíší do textu, a pokud rozpoznávač obsahuje jazykový model, proběhne například kontrola syntaxe nebo další prvky modelu.

Po vstupu zvukových dat do rozpoznávače (viz obrázek 2.1) dochází k procesu extrakce parametrů, který spočívá v odstranění nežádoucích okolních zvuků. Zachová pouze ty, které odpovídají lidské řeči. Nežádoucími zvuky mohou být např. auta, sbíječky a jiné zvukové šumy. Díky extrakci nežádoucích parametrů pracuje akustický model pouze s korektní částí zvuku a dokáže správně fungovat. Porovnává z fonetického slovníku hlásky, ze kterých poskládá slovo. Využívá se k tomu Skrytý Markovův model (dále jen HMM[2]), který má uvnitř stavy, jež označují hlásky a někdy kvůli kompaktnosti i celá slova. Jejich prvním úkolem je zkontrolovat, zda je slyšet fonémy správně. Tento proces probíhá díky porovnávání statistické pravděpodobnosti hlásek, přičemž nejpravděpodobnější z nich vybere. Pravděpodobnost je vypočítaná podle předcházejícího a následujícího fonému. Druhá část je shlukování zvolených písmen do slova. Používá se zde pravděpodobnost daného slova, která vyplývá z četnosti v hovorové řeči daného jazyka. Například po slově *„jede“* bude pravděpodobnější *„auto“* než *„myofunkce*[3] *“*. Ve třetí části rozpoznávače se kontroluje kontext věty a ten se mění v okamžiku, kdy prochází algoritmem nová slova. To může znamenat, že některá slova se i několikrát změní. Například ve větě by se neměla objevit dvě slovesa, to kontroluje jazykový model. Každý člověk vysloví slovo nebo hlásku při sdělení pokynu odlišně. To znamená, že hlasový projev pokynu má jinou intonaci a výslovnost. Uživatelé mohou také ve spisovné češtině tvořit věty, které nedávají smysl. Příkladem je *brněnské hantec*[4] nebo jiné světové nářečí. Slovník rozpoznávače nemůže obsahovat velké množství fonetických zápisů jednoho slova kvůli jeho velikosti. Vývojáři z tohoto důvodu nemohou zahrnout všechna nářečí a různé fráze daného jazyka.

[1] Foném je hláska, která je v jazyce významotvorná, její změna může tedy měnit význam slova.
[2] HMM – Hidden Markov model, v češtině Skrytý Markovův model jinak také řetězec.
[3] Myofunkce – funkce konkrétního svalu, zejména při léčbě/terapii ortodontických problémů.
[4] Pojem *„hantec“* je zkomoleninou slova *hantýrka*. Brněnská hantýrka je vnímána jako zajímavá, vtipná, ale těžko pochopitelná brněnská mluva.

Častěji se používají neuronové sítě, hlavně z důvodu rychlosti a flexibility. Další výhodou je porozumění podbarvení lidského hlasového projevu, nezvykle citově zabarvené výslovnosti nebo atypické až patologické výslovnosti. Tím, že tyto sítě pracují jako lidský mozek, se dokáží adaptovat na tyto možné, ale zásadní faktory, které vždy ovlivní rozpoznávání lidské řeči. Nevýhodou je trénink, jež může být velice zdlouhavý. Na začátku totiž síť nedokáže určit správné slovo z hlasového projevu lidí. Neumí se ihned přizpůsobit nepřesnostem a kvůli tomu nelze správně zpracovat zvuková data. Síť se potřebuje vše naučit rozpoznat a adaptovat se. Z těchto důvodů je potřeba neuronové síti nabídnout různá zvuková data ke zhodnocení, například reálná zvuková data z různého prostředí, dále záznamy mnoha lidí s rozmanitými druhy podbarvení hlasu či výslovnostmi na základě emocí. Díky tomu dokáže neuronová síť zpracovat řeč lépe než HMM. Také mnohem rychleji a přesněji umí vyhledat správné slovo.

Tématem konzultací s mým vedoucím práce bylo použití tzv. *keyword spottingu* nebo přepisu řeči. *Keyword spotting* se na rozdíl od přepisu řeči soustředí jen na klíčová slova, nikoliv na celé souvětí. Abychom mohli zpracovat celé souvětí, je potřeba jazykový model. Hlavní devízou *keyword spottingu* je rozpoznávání izolovaných slov [12]. Sice poslouchá a projde celý zvukový záznam, ale jakmile zjistí, že se nejedná o hledané klíčové slovo, zavrhne jej a nepokračuje v jeho rozpoznávání. Neprochází větný kontext a soustředí se pouze na slova. Díky tomu dokáže být přesnější a někdy i rychlejší vzhledem k hledanému slovu než přepis řeči do textu.



Obrázek 2.2: Detekce klíčových slov na základě podobnosti zvuku. Toto schéma znázorňuje typický postup. Vstupní záznam je zpracováván sekvenčně s krátkým rolovacím oknem (např. 800 ms). Systém porovnává každý extrahovaný fragment se sadou vzorů předem definovaných pro každé klíčové slovo. Model pak vydá rozhodnutí založené na vzdálenosti mezi fragmentem a každým vzorem. Pokud je vzdálenost dostatečně malá, je vzhledem k danému časovému razítku vygenerován výskyt klíčového slova [7].

Jedním z benefitů *keyword spottingu* je také objemnost knihovny. Ta je znatelně menší hlavně díky menšímu slovníku. Často v něm jsou jen natrénovaná slova v řádu desítek slov. Dalším benefitem je, že aplikace může běžet na pozadí a díky své jednoduchosti nemá vliv na rychlost samotného zařízení. Proto se využívá jako *wake-up word*[5], v češtině spouštějící slovo. Slouží na spuštění robustnějšího přepisu řeči do textu nebo aplikace, příkladem je funkce Siri či Ok Google. Soukromí uživatele je velmi zásadní. Pokud aplikace používá celkový přepis řeči, nelze určit, zda může tyto texty dále zneužít. Tento typ rozpoznání je trénován pouze na určitá klíčová slova, proto nerozumí konverzacím [13]. Je tedy pro uživatele bezpečnější, nedá se ale říct, že nejbezpečnější.

[5]Někdy se uvádí *hotword* anebo také *trigger word*.

Přepis řeči se specializuje na přepsaní celé věty, rychle zachytí, co uživatel řekl, a navíc po sobě kontroluje syntax a smysluplnost dané věty. V dnešní době tento proces probíhá v řádech milisekund. Je důležité, aby reakce na klíčové slovo přišla hned po vyslovení příkazu, nikoliv poté, co zařízení vyhodnotí, že skončila věta. Slovníky jsou často objemné, je tedy mnohdy zdlouhavější je prohledávat.

2.2 Tvorba lidské řeči

Řeč je artikulovaný zvukový projev člověka sloužící k přenosu myšlenek a dorozumívání. Tyto zvuky jsou shluky hlásek, jež spolu dohromady skládají smysluplné slovo, které nese přidanou hodnotu. Hláska je základní nejmenší a nejjednodušší jednotkou zvukové stránky řeči. Při tvoření hlásek vychází z plic vzduchový proud do hlasových orgánů. V těchto orgánech vzniká člověkem vyprodukovaný zvuk, lidský hlas. Lidský hlas je zvuk modulace proudu vzduchu kmitajícími hlasivkami. Tento proces procházející hlasivkami se nazývá fonace a hlasivky jsou hlavním fonačním orgánem. Z hlasivek postupuje výdechový proud vzduchu do artikulačních orgánů. Prioritou hlasivek je poslat vzduchový proud z výdechu do ústní a nosní dutiny, kde se za pomoci artikulačních orgánů na podobě hlásky podílí artikulační orgány. Ze vzniklých hlásek se skládají slova. Druhy hlásek (samohlásky a souhlásky) slouží k výstavbě a rozlišení slov podle síly tónů nebo šumu. Z fonetického hlediska není ostrá hranice mezi hláskami, přechod je přirozeně plynulý, pokud se nejedná o poruchu. Artikulace je vytváření hlásek pomocí koordinovaných pohybů mluvidel. Artikulaci ovlivňuje několik faktorů a mění se tím kvalita i kvantita hlásek ve slovech. Může jít o nekoordinované pohyby mluvních orgánů, které způsobují hláskovou atypickou výslovnost člověka. Poté se řeč stává nesrozumitelnou. Jedná se i o atypický výdechový proud, absence zubů, nemoc horních dýchacích cest, vady řeči nebo atypické hlasové podbarvování mluvního projevu pomocí emocí. A také často jako faktor zasahuje i emoční stránka člověka [15].

Na různé výslovnosti je možné reagovat slovníkem. Většinou ve slovníku existují dva či více fonetických zápisů. Druhou možností je trénovaná neuronová síť, která se dokáže adaptovat a někdy i učit za provozu při používání. Lidé svou intonací mohou změnit fonémy ve slově, ať už křikem, šepotem anebo vadami řeči. Vývojáři těchto modelů musí vzít tyto faktory v úvahu. I nejdokonalejší modely nedokáží určit některé lidské anomálie při řeči. Dalším problémem je nářečí, jež často mění kontext věty, slov a styl mluvy. Jazykový model s těmito možnostmi většinou nedokáže pracovat, jelikož mu nejsou známy, neumí je předvídat. Není úplně jednoduché zakomponovat spisovnou řeč společně s nářečím do jednoho modelu. Vzájemně se budou ovlivňovat, a tudíž zhoršovat přepis řeči.

2.3 Knihovny pro rozpoznání hlasu

Při hledání knihoven pro rozpoznání hlasu jsem jich objevil více, avšak po konzultaci s vedoucím práce jsem se rozhodl, že vyloučím knihovny, jež odesílají data na server pomocí internetu. A to z toho důvodu, aby byla aplikace atraktivnější pro uživatele, kteří buď sportují v prostředí, kde není signál, nebo si při sportu vypínají mobilní data a wifi, aby nebyli rušeni. Některé knihovny zpracovávají řeč mimo zařízení kvůli důkladnějšímu zpracování. Díky tomu dokáží využít rychlejší čip než ty, které nabízí dnešní telefony. Mají také přístup k větším slovníkům. Dalším bodem je samotná kompatibilita a udržitelnost knihovny. Důležité je, aby aplikace cílila na nejnovější operační systémy Android. V této době se jedná

o zařízení s Android 14. Proto nebudu testovat knihovny, jejichž podpora skončila nebo brzy skončí.

Mým úkolem bylo vybrat vhodné knihovny pro Android s cílem využít alespoň dvě z nich. Jedna knihovna využívá Speech to Text (dále jen STT[6]), druhá *keyword spotting*. Z hlediska rychlosti je mnohem lepší druhé řešení, avšak přepis řeči lze použít podobným způsobem, jen nebude tolik efektivní. Abych věděl, čemu se chci přiblížit, otestoval jsem jako první knihovnu od společnosti Google integrovanou v Androidu. Problémem zde bylo, že odesílala data na server. Zvládla vše dobře a v jakémkoliv podporovaném jazyce, avšak knihovnu jsem vyřadil z důvodu užití internetu. Byla ovšem dobrou ukázkou toho, čeho chci dosáhnout. Nalezl jsem mnoho industriálních řešení na míru zákazníkovi, s vytrénovanými klíčovými slovy podle jeho výběru. Je to jedna z dražších variant a moje řešení je málo komplexní na to, aby se tato investice vyplatila. Některé společnosti nabízely využití svých řešení zdarma do určité míry používání za předpokladu, že nebudou použity komerčně. Jedna z komerčních možností je od společnosti Picovoice. Jejich nabídka knihoven je rozmanitá, podmínky použití však dosti neúprosné. Přesto jsem ji otestoval. Pro soukromé účely, kdy je aplikace používána pouze na jednom či dvou zařízeních, ji lze využít. Nevýhodou je, že se musí instalovat do zařízení jako soubor `APK`[7]. Tyto vývojářské sady, zkráceně SDK[8], se specializují jen na daný okruh funkcí, jež zvládnou bez problémů. Použiji i knihovnu PocketSphinx. Jedná se již o pátou verzi softwaru, jenž se začal vyvíjet kolem roku 1980. Dalším z vybraných kandidátů je Vosk model, který vychází z Kaldi softwaru[9] Poslední dvě zmíněné sady patří do skupiny *open source* projektů, kde může každý přispět svou částí kódu nebo dokonce použít vlastní jazykový model.

Picovoice Porcupine

Vývojáři této společnosti nabízejí mnoho různých sad pro rozpoznání hlasu. Jejich jména jsou odvozena od zvířat, například *Leopard* a *Cheetah* slouží k přepisu řeči, *Porcupine* se soustředí na detekci klíčových slov. K dispozici jsou i sady pro odstranění zvukového šumu *Koala*. *Eagle* rozpoznává uživatele podle hlasu [10]. Jelikož má aplikace potřebuje pouze spustit záznam videa nebo fotografie, úplně mi postačí *Porcupine Wake Word SDK*. Sada je vysoce přesným a lehkým nástrojem na odposlech klíčových slov. Bohužel tyto sady jsou předplacené a každá z nich má jiné podmínky. Například u sady, kterou používám, je podmínka inicializace pouze na třech zařízeních za měsíc. Aplikace je totiž propojená s emailovým účtem přihlášeným k jejich webové aplikaci a podle toho se určuje, kolik uživatelských zařízení se připojilo.

I přes tuto nevýhodu otestuji sadu pro detekci. Zajímavý je i fakt, že jejich webová aplikace nabízí vytrénování vlastních klíčových slov, které pak integrujete do programu. Vytrénovat lze pouze slova delší než pět fonémů a sousloví nesmí být delší než tři slova. Soubory nejsou velké, a to hlavně díky formátu `.ppn`[10], který je pro člověka nečitelný. *Porcupine* využívá hluboké neuronové sítě, trénované na reálných situacích v reálném prostředí. Dokáže se proto rychle adaptovat na různé situace a klíčové slovo rozpoznat rychle, kdykoli a kdekoli. Vzhledem k nekomerčnímu využití je tato aplikace vynikajícím kandi-

[6] STT – Speech to Text, přepis řeči do textu

[7] https://docs.fileformat.com/cs/compression/apk/

[8] SDK – software developer kit, v češtině sada pro vývojáře softwaru.

[9] Kaldi je sada nástrojů pro rozpoznávání hlasu napsaná v C++. Na jeho vývoji se podílela i naše univerzita [11].

[10] Soubory PPN jsou soubory bez běžného formátu, často např. packPNM komprimovaná bitmapa (Bitmap Image File, BMP).

dátem, avšak pokud bude vložena do Obchodu Play, nebude fungovat. Uživatel ji stáhne, avšak rozpoznání bude nefunkční, jelikož účet přihlášený k jejich webové aplikaci se zablokuje. Testeři od Googlu totiž před vydáním aplikace testují její funkčnost. Dělají to na více zařízeních, a tím vypotřebují povolený počet inicializací.

CMU PocketSphinx

Sada vychází ze zkušeností dvacetiletého vývoje, je nejstarší z této trojice. Neznamená to však, že je zastaralá. Využívá velký slovník, jenž lze upravovat. Díky úpravám je možno dopsat další fonetický zápis slova. Jedná se již o pátou verzi, podporuje dokonce rozpoznání klíčových slov a dokáže je určit. Není však tak přesná jako sada od firmy Picovoice. Přece jen se jedná o univerzitní nekomerční projekt, jenž se udržuje díky studentům, skupině lidí ze sociální sítě LinkedIn a některých společností, jako je Alpha Cephei. Jedna ze zásadních věcí, kterou musím zmínit, je však flexibilita. Jako vývojář dokáži velmi rychle změnit slovník, doplnit další fonetický zápis slova nebo přidat slovo, které nebylo ve slovníku. Samozřejmě toto nedoporučuji bez hlubšího seznámení se s modelem.

Aplikace využívající tuto sadu potřebuje *Sphinx4 API*, které je napsáno v jazyce Java. Z tohoto API je možno využít mnoho funkcí, jako je rozpoznání uživatele, přepis textu anebo rozkouskování řeči [1]. Pro tyto funkce, obsáhlou podporu a pro použití i v nejnovějších Android zařízeních je pro mě vhodným kandidátem. Žel stále používá *pre-alpha* sadu pro vývojáře, jež není dotažená do konce a občas nenabídne nejpřesnější přepis. Pro integraci modelu a knihovny do projektu lze využít tutoriál z jejich stránek[11].

Alpha Cephei Vosk

Tato vývojářská sada dokáže přepisovat řeč a rozpoznávat uživatele. Není nejvhodnější na rozpoznávání klíčových slov, avšak správné nastavení tuto funkci umožňuje. Zkouším ji hlavně z toho důvodu, že nabízí český akustický model. Na svých stránkách[12] jich ale nabízí mnohem více. Využívá software Kaldi, tudíž funguje na podobném principu jako předchozí knihovna. Alpha Cephei se totiž podílela na vývoji obou. Avšak od sebe se v mnohém liší. Knihovna Vosk je integrovaná jako celek, a tudíž nelze libovolně upravovat její části. Nabízí přesný přepis řeči do textu. Po několika úpravách lze docílit toho, aby se neřídila větami, ale pouze slovy.

Po předělání logiky rozpoznání, kdy původně po dílčím výsledku se čeká na konec věty anebo na konec řeči, lze využít ještě finální výsledek, avšak ten bude ve většině případů stejný jako na konci řeči. Mé řešení hledá klíčový řetězec v hypotéze dílčího výsledku. Pokud hypotéza obsahuje hledané slovo, bude zvolena daná funkce. V tom okamžiku je model ukončen a znovu spuštěn. Vymaže se tím hypotéza, protože do dílčí hypotézy přicházejí další slova, která byla vybrána podle kontextu, co model slyšel. Funguje na principu přepisu řeči – pokud nezáleží na rychlém rozpoznání, dokáže být mnohem přesnější. Oba jazykové modely jsou do projektu vloženy jako Java moduly a oba pak potřebují ještě své závislosti, jež se stahují z repozitářů.

[11] https://cmusphinx.github.io/wiki/tutorialandroid/
[12] https://alphacephei.com/vosk/models

Kapitola 3

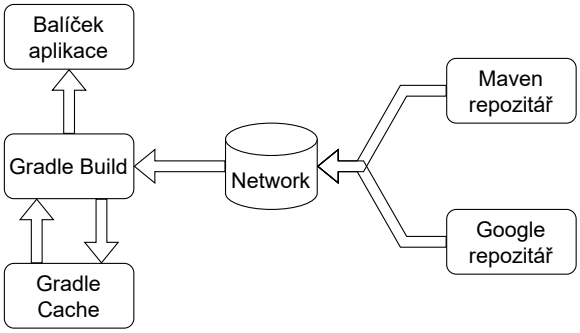
Vývoj aplikace na OS Android

Vybral jsem operační systém Android hlavně díky jeho dostupnosti a jednoduchosti. Dokonalým důkazem je webová stránka pro vývojáře, kde se nachází celá řada rad a návrhů a také dobře popsané změny v nových verzích systému Android [2]. Budu programovat v jazyce Java. Novější alternativou, dnes již velmi populární pro vývoj mobilních aplikací v systému Android, je také programovací jazyk Kotlin. První jeho verze se začala používat v roce 2017. Jeho popularita souvisí také s tím, že volně navazuje na jazyk Java. Díky tomu může pracovat s jeho komponentami, což vývojářům výrazně usnadní práci, protože nemusejí vyvíjet ve dvou jazycích dvě odlišné verze. Oba jazyky využívají stejné definice uživatelského prostředí jako je soubor `Manifest` [9]. Nezáleží tedy na programovacím jazyce, protože používají stejné soubory. Návrh aplikace budu vysvětlovat v pozdější kapitole 4. Lze ho využít i pro implementaci v Kotlinu nebo úplně na jinou platformu. Pro kompilaci a linkování všech potřebných souborů jsem si vybral Gradle, protože se o něm psalo na Android developeru [2], ačkoliv existují i jiné možnosti. Je založen na Apache Groovy, samotný Gradle nic nedokáže, využívá k tomu pluginy.

3.1 Základní soubory a nastavení

Doporučuji začít nastavením virtuálního zařízení, jež se sice nechová jako reálný telefon, ale kvůli rychlé kontrole, jak obrazovka vypadá, nebo otestování podpory verze Android je vynikající volbou. Vhodnější je samozřejmě testovat na reálném telefonu. Důležité je definovat si určité informace o projektu a to tak, že v souborech `AndroidManifest.xml` se nakonfigurují potřebné vlastnosti aplikace, uživatelské povolení, co aplikace využívá a podobně. Podrobnosti potřebné pro určité části zmíním v dalších kapitolách. Manifest určuje hlavní náhled aplikace (neboli obrazovku, která se zobrazí při spuštění). Dále definuje, jak se jmenuje, jaké téma používá a zmiňuje další důležité informace. O kompilaci a vytvoření balíčku se stará nástroj Gradle. Soubory s koncovkou `.gradle` nesou definice kódu, jako například jeho verzi, s čím a jak se bude projekt kompilovat a v neposlední řadě slouží ke složení projektu. V nejhlavnějším souboru se definují hlavní sestavovací pluginy, souvislosti a repozitáře. Ty nejpoužívanější a zároveň hlavní jsou Google a mavenCentral. Odtud se stahují knihovny, pokud je tam autoři nahráli. V jiném případě si je musím nahrát sám, pak může přibýt další soubor Gradle, který kompiluje tento modul.

Při každém spuštění či kompilaci se souvislosti spojené tímto nástrojem stáhnou anebo vloží do aplikačního balíku. V těchto souborech lze nastavit opravdu cokoliv, od verze podporovaných čipů až po kompilaci správnou verzí Javy. Důležitým nastavením je verze



Obrázek 3.1: Demonstrace fungování souvislosti a implementace jednotlivých knihoven. Lze vidět, že při kompilaci Gradle vše linkuje a skládá dohromady, přičemž již stažené knihovny načítá z *cache*.

kódu, minimální a cílené verze Android SDK. Tyto verze udávají podporu API pro systém Android. V dnešní době musí aplikace podporovat minimálně API 21, což je Android 5 neboli Lollipop. Cílená verze znamená, na jaké verzi se program kompiluje, což je v tuto chvíli API 34, Android 14 neboli Upside Down Cake. Podrobněji se o tomto píše v knize [9]. Tyto definice pak kontroluje Google při nahrání aplikace na jejich obchod a pokud nesplňuje tyto podmínky, nelze ji nahrát. Ovšem technologie se vyvíjí a Google vydává novou verzi každý rok. Od letních měsíců musíte cílit na API 35, neboli Android 15, se jménem Vanilla Ice Cream, abyste měli možnost ji vydat.

Soubory jsou rozděleny do skupin podle toho, co obsahují. První skupina je samotná aplikace, která obsahuje třídy, zdroje neboli obrázky, ikony, řetězce. Druhou skupinou jsou různé moduly, často se jedná o soubory, jež nepatří do základní implementace Android aplikace. Může se jednat např. o offline knihovny nebo různé balíčky. Třetí skupina se nazývá *Gradle Scripts* , protože v sobě má všechny soubory nástroje Gradle. Jsou to hlavně kompilující skripty, `build.gradle`, anebo jsou zde zobrazeny jejich nastavení, `settings.gradle`, společně s vlastnostmi `gradle.properties`. Hlavní kód se bude nacházet v třídě `MainActivity`, ale samozřejmě není jediná, v níž lze programovat. Implementací projektu se budu zabývat v kapitole 5.

Ikona aplikace je její nezbytnou součástí, slouží k jednoznačnému odlišení od ostatních aplikací v telefonu. Nachází se v balíčku `mipmap`, kde jsou rozděleny na pozadí ikony, popředí ikony, klasický obrázek ve tvaru čtverce a kulatou ikonu. Je to z toho důvodu, že uživatelské prostředí systému Android nabízí uživatelům vybrat si vzhled ikon, tudíž programátoři, spíše tedy grafici, musí takovou ikonu vytvořit pro obě dvě verze. V Android studiu stačí vložit obrázek, který splňuje velikost.

Obrázek 3.2: Na obrázku lze vidět většinu tvarů ikon v operačním systému Android. Hlavními tvary jsou tyto čtyři: (zleva) kruh, kruho-čtverec, čtverec se zaoblenými rohy a čtverec. I když se jedná o čtverec, Android rohy vždy zaoblí. Na tomto příkladu je možno vidět i stíny samotné ikony, což vytváří efekt 3D ikony.

Zvolil jsem takový obrázek, jež splyne s pozadím ikony. Tvoří ji fotoaparát spojený s mikrofonem, bílý obrys na šedém pozadí. Ikonou se myslí hlavní kruh, jež je vyznačen na obrázku 3.2. Vše ostatní je podklad. Od verze Android 8 mohou ikony obsahovat stín pro zlepšení uživatelského zážitku.

3.2 Rozložení obrazovky

Existují různé typy rozložení, které jsou děleny podle umístění a lokalizace objektů na obrazovce. Vytvořené náhledy s objekty pak ovládá třída, která je uvedená v definici souboru a napojená skrze svůj kód. Nezáleží na tom, zda programujete v Javě či Kotlinu, oba jazyky dokáží zpracovat definice v těchto souborech. Používají se tzv. vizuální kontejnery, v původní terminologii `Layouts`. Jsou odvozeny od třídy `ViewGroup` [6]. Prvky na obrazovce musí být flexibilní a reagovat na změnu orientace zařízení.

Prvním rozložením je `LinearLayout`. Uspořádává prvky horizontálně za sebe do jednoho řádku či vertikálně pod sebe. Klíčovým parametrem je `orientation`, který udává směr uspořádání prvků uvnitř.

Dalším rozložením je `RelativeLayout`, který umožňuje definovat vztahy mezi určitými prvky uvnitř rozložení. Aby to bylo možné, musí mít každý prvek jednoznačný identifikátor. Implicitně se vkládají od levého horního rohu. Tento typ je velice flexibilní v případě, že prvky mají jen minimální počet vazeb. Pokud se toto nedodrží, mohou nastat tzv. kruhové vazby a rozpory.

`FrameLayout` se používá při dynamicky se měnícím rozložení obrazovky. Pokud vložíte více prvků, zobrazí se na stejném místě. Jejich umístění je možné určit pomocí atributu `android:layout_gravity`.

Zmíním zde i `TableLayout`, jenž se od verze API 14 (Android 4.0) moc nevyužívá. Nahradil ho více intuitivní `GridLayout`. Jedná se o matici prvků, již má plně pod kontrolou vývojář. Definuje, kde přesně se mají zobrazit. Funguje jako tabulka, proto lze roztáhnout buňky přes více řádků i sloupců.

`ConstraintLayout` přidává prvkům omezení a funguje jako gumička. Přitahuje je například k okraji. Pokud tedy připnete daný prvek ke všem čtyřem okrajům nadřazeného rozložení, bude uprostřed obrazovky na obou osách. Nezáleží na velikosti obrázku či textového elementu. Vždy bude uprostřed.

`AbsoluteLayout` zobrazuje prvky na přesně specifikovaných souřadnicích na obrazovce. Není doporučeno jej používat kvůli jeho nulové flexibilitě. Zařízení mají odlišné velikosti obrazovek, nemluvě o změnách orientace.

3.3 Vydání v obchodě

Aby mohli programátoři vydávat aplikace, potřebují vývojářský účet u společnosti Google. Dále je nutný funkční e-mail a poplatek 25 dolarů jako důkaz toho, že nejsou roboti. Vše slouží jako ochrana před spamem a zároveň se předchází tomu, aby si uživatelé publikovali, co chtějí a kdy chtějí. Před vydáním aplikace musí vyplnit spoustu dotazníků ohledně svého věku, informací o aplikaci, k čemu bude sloužit anebo zda je nějakým způsobem výdělečná. Podrobnější popis dotazníků je ve zdrojích [6, 9]. Doporučuji spíše internetový odkaz na vývojářskou stránku Androidu, kde jsou potřebné informace aktuálnější [2, 3]. Dalším úkolem je vytvoření záznamu v obchodě. Je potřeba vložit ikonu a několik obrázků z aplikace. Na obrázcích nezáleží, jen musí splňovat předepsanou velikost. Dále je třeba vložit název aplikace a krátký i dlouhý popis toho, k čemu aplikace slouží.

Po vyplnění těchto informací už stačí vytvořit vydání aplikace. Android studio nabízí funkci generování aplikačního balíčku ve formátu `.aab`[1]. Jakmile získáte balíček aplikace, stačí ho vložit na web Google Console, jenž udělá všechno, co je potřeba. Načte číslo kódu, které nesmí být stejné jako některé již vložené. Ze souboru vloží verzi aplikace a programátor pouze napíše zásadní změny do popisu. Testeři v Google ho projdou a otestují. Pokud je vše v pořádku, je aplikace k dispozici v obchodě.

Aktualizace aplikace se provádí pomocí nového vydání. Postup je prakticky stejný jako u prvního vydání. Jedinou změnou je inkrementace čísla kódu. Buď můžete označit verzi jako v předchozím případě, nebo ponechat načtené informace z balíčku jež se nastavují v souboru `build.gradle`, který se nachází v modulu `app`. Je doporučeno informovat uživatele o tom, co je nového. Děje se to v popisu vydání a může to být ve více jazycích. Záleží na tom, jakým způsobem je nastaveno vydání v různých zemích světa. V mém případě to byla angličtina, čeština a slovenština. Google Console pak nabízí spoustu informací o tom, jak si vaše aplikace vede, kolik je instalací, jestli se při testování objevily nějaké chyby a podobně.

Existují tři možnosti, jak vydat aplikaci do obchodu. První z nich je *Interní testování*. To se používá pro testování v rámci společnosti nebo k ní mají přístup jen ti uživatelé, kteří se přihlásili do testování e-mailem. Vývojář je pak přidá do testovací skupiny. Druhou možností je zveřejnit *Otevřené testování*. Funguje na podobném principu, jen se do testování uživatelé nepřihlašují zasláním e-mailu, nýbrž přihlášením v Obchodu Play. Narazit na ni může kdokoli, to u první možnosti nelze. Poslední možností je již finální publikace, jež nabízí různé informace z obchodu a je viditelná všem uživatelům v dané povolené zemi.

[1] Formát `.abb` – Android app bundle je formát, ve kterém se aplikace distribuuje do Obchodu Play.

Kapitola 4

Návrh aplikace

Uživatel umístí telefon do stativu, zapne aplikaci a nastaví ji tak, aby na něj mířila. Při sportovní aktivitě dává heslovitě pokyny pro spuštění různých funkcí, jako například zapnutí svítilny nebo pořízení fotografie. Toto je cíl, jak by měla výsledná aplikace vypadat. V pozdějším vývoji je nezbytné implementovat funkčnost, aby si mohl člověk nastavit aplikaci z místa, kde aktivitu provozuje. To se hodí při nastavení odpočtu a délky videa. Obrazovky jsem se rozhodl rozdělit podle účelu, k jakému budou sloužit. Hlavní obrazovka zobrazí náhled kamery a bude obsahovat tlačítka po pravé straně. Některá z nich vytvořím jako jednoduché přepínače, ostatní poslouží k přepnutí do jiné části aplikace. Obrazovka nastavení bude obsahovat pouze textové vstupy na zadání slov, kterými chce uživatel ovládat focení a natáčení videa. Dále tu bude změna doby odpočtu a délky videa; vytvořím i tlačítko pro uložení a návrat na hlavní obrazovku. Poslední náhled v aplikaci zobrazí pouze dané uživatelské nastavení a nápovědy k tlačítkům a funkcím.

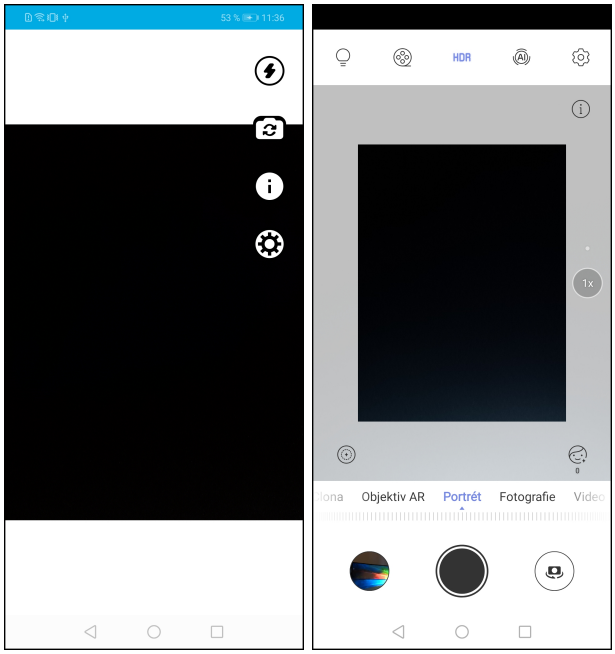
Uživatelské prostředí vytvářím jednoduché, jelikož aplikace slouží primárně k otestovaní knihoven, která je z nich nejvhodnější. Dalším cílem je, aby se v ní uživatelé dobře a rychle zorientovali. Inspiroval jsem se v knihách o tvorbě uživatelského prostředí [14, 8], kde lze najít spoustu rad, jak udělat prostředí přehledné a nenáročné. Sice se jedná o testovací verzi, avšak budu aplikaci vydávat veřejně z důvodu veřejného testování, proto se snažím vytvořit příjemný uživatelský zážitek, aby se uživatel nenechal odradit od jejího používání.

Náhledy těchto částí jsou v souborech `XML`.V těchto souborech se definují UI objekty s jejich vlastnostmi. Použiji rozložení obrazovky `RelativeLayout`, popsané v kapitole 3.2. Určitě se dají využít i jiné typy rozložení. Na barevné téma byla převzata modrá barva fakulty, jež je jako primární barva a nachází se v horní liště zařízení.

4.1 Hlavní obrazovka

Jak jsem již uvedl, mým cílem je zpříjemnit uživateli používání aplikace. Toho chci docílit vizuálním přiblížením se k vestavěné aplikaci výrobce. Proto jsou funkční tlačítka vpravo a na obrazovce se nenachází žádný text. Jednoduchost podtrhuje také to, že se zde objevuje celý náhled kamery, přičemž zbývající prázdné okraje jsou černé. Je běžné, že mobilní telefony nemají přední svítilnu a v jiných aplikacích s funkcí focení předním fotoaparátem je to řešeno podsvícením náhledu, tedy zbarvením okrajů do bílé barvy.

Na skupině obrázků 4.1 lze v mé aplikaci (obrázek vlevo) vidět, že se změní barva tlačítek. Děje se tak proto, aby byla tlačítka dobře vidět. Určitě to lze vyřešit i jejich zbarvením podle podkladu, ale barva by se v tomto případě měnila vždy, když uživatel



Obrázek 4.1: Nalevo se nachází obrazovka mé aplikace, napravo vestavěná aplikace výrobce Huawei. Obě mají zapnutou přední svítilnu. Náhled kamery z velké většiny případů nezabere celou plochu obrazovky, a proto vývojáři zbytek prostoru obrazovky využívají na tlačítka nebo na zesvětlení kompozice, aby obrazovka působila jako svítilna.

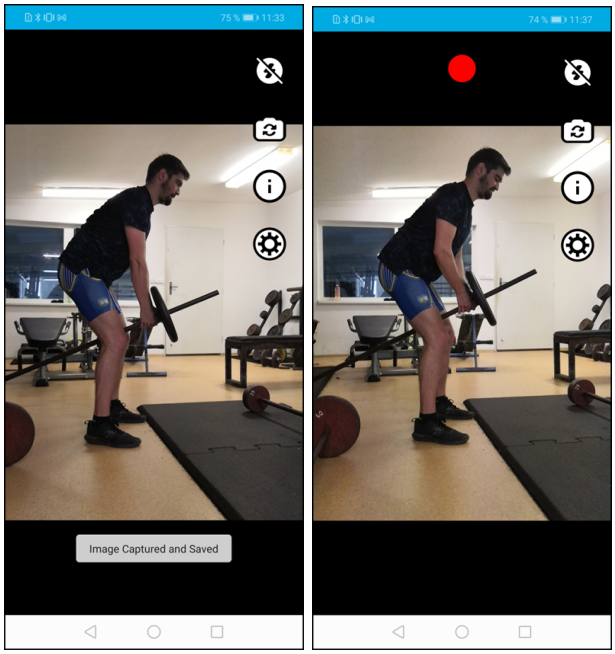
pohne s obrazem, což pro oči to není zrovna příjemné. Navíc změnou barvy se nezabývají v některých případech ani výrobci telefonů a ponechávají tlačítka špatně viditelná. Ikony tlačítek jsem zvolil takové, které jsou uživatelům již známé. Není důležité vybrat přesně tytéž, avšak během používání by měly být pochopitelné na první pohled. Většinu z nich nabízí Android Studio ve svém seznamu vektorových aktiv. Využil jsem i některé externí grafiky, například u ikony nastavení. Jejich podklad je černý právě proto, aby byly vidět i na bílém pozadí a zároveň to zlepšilo viditelnost na vícebarevném pozadí. Tlačítka lze umístit prakticky kamkoli, jednak i z toho důvodu, že by se v budoucnu dala ovládat hlasem. Pořízení fotografie přes spoušť zde není implementováno, jelikož není potřeba.

Hlavní obrazovka je definována v souboru `activity_main.xml`. Náhled fotoaparátu je umístěn uprostřed obrazovky pomocí objektu `PreviewView`. Pozadí je nastaveno na černou barvu, na barvu bílou se změní při použití přední svítilny. Na každém zařízení se zobrazuje jinak, jelikož telefony disponují jiným rozlišením hardwarové kamery. Android proto nabízí nastavení náhledu, aby se vlezl na obrazovku. Já používám `FIT_CENTER`, ale existují i jiná nastavení velikosti a umístění. Podrobnější popis je na stránkách Android developer[1]. Díky této proměnné se náhledy přední i zadní kamery umístí přímo doprostřed. Při pořízení fotografie se na obrazovce zobrazí vyskakovací oznámení, případně některé chybové výpisy. Upozornění se objeví ve spodní části obrazovky.

V horní části se dále nachází červený kruh, jenž upozorňuje uživatele na spuštěné natáčení videa. Pak už se zde vyskytují jen zmíněná tlačítka. Nastavení mají všechny stejné – výšku i šířku na 40sp[2]. Všechny tyto prvky jsou propojeny ve třídě `MainActivity`, ve funkci

[1] https://developer.android.com/media/camera/camerax/preview

[2] Hodnota rozměru definovaná v `XML`. Rozměr je specifikován číslem následovaným jednotkou měření, jako je 10px, 2in, 5dp. *Density-independent Pixels* (`dp`) je abstraktní jednotka založená na fyzické hustotě



Obrázek 4.2: Na fotografiích lze vidět hlavní obrazovku při používání aplikace a zároveň i ukázku použití, kdy díky fotografii či videu mohu zlepšit rovnost zad při cviku. Levá fotografie obrazovky zobrazuje obrazovku po vyfocení fotografie a pravá záznam videa.

`OnCreate`. Tento jediný náhled má příznak `FLAG_KEEP_SCREEN_ON`, jenž značí nezamknutí obrazovky zařízení po delší době nečinnosti. Důvodem je *always on* režim aplikace. Zatímco uživatel provozuje sportovní aktivitu, telefon je stále odemknutý a poslouchá příkazy. Pokud by se zařízení vypnulo, nebylo by schopno zachytit fotografii ani video.

4.2 Informační obrazovka

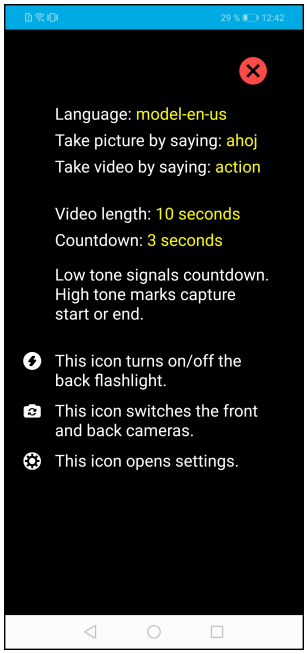
Tato část aplikace má pouze informační hodnotu, uživatel na obrazovce nemá možnost interagovat se zobrazením (pouze s tlačítkem návratu). Proto se zde nachází zvolená klíčová slova, odpočet, délka videa a pro uživatele důležité informace o tlačítkách. Tyto informace nejsou napsány v celých větách, nýbrž zkráceně v heslech, protože lidé dnes nečtou celé texty, ale spíše skenují různá hesla, výrazné nadpisy a obrázky. Tento fakt vyplývá z knihy o uživatelském prostředí pro webové a mobilní aplikace [5]. Proto jsou důležité informace zvýrazněny žlutě, aby uživatelé nemuseli hledat ve změti slov. Vždy při načtení informací se z úložiště načtou slova a přidají se za již vytvořené řetězce.

Řádky jsou roztaženy po celé obrazovce. Pokud má tedy zařízení širokou obrazovku, jsou po celé její šířce. Ikony na levé straně textu jsou miniatury z hlavní obrazovky. Žlutý text je vložen při načtení tohoto náhledu, je uložen v *SharedPreferences* (viz kapitola 4.5) a mění se podle uživatelského vstupu z nastavení.

Definice této obrazovky se nachází v souboru `activity_help.xml`. Uvnitř rozložení náhledu je jedno tlačítko o velikosti 60sp v horním pravém rohu pro návrat zpět. Pod

obrazovky. *Scale-independent Pixels* (`sp`) je jako jednotka `dp`, ale je škálována podle uživatelovy preferované velikosti písma. Jednotka *Pixels* (`px`) odpovídají skutečným pixelům na obrazovce. Dále existují přesné míry jako jsou milimetry anebo palce. Avšak jednotka *Points* (`pt`) je 1/72 palce. Detailnější popis je zde https://developer.android.com/guide/topics/resources/more-resources.

ním jsou textové prvky nesoucí informace o uživatelském nastavení. Miniatury ikon jsou vytvořeny pomocí obrázků. Propojení s třídou `HelpActivity` mají pouze prvky, jež obsahují měnitelné texty. Vložení textu popisuji v kapitole 4.5.



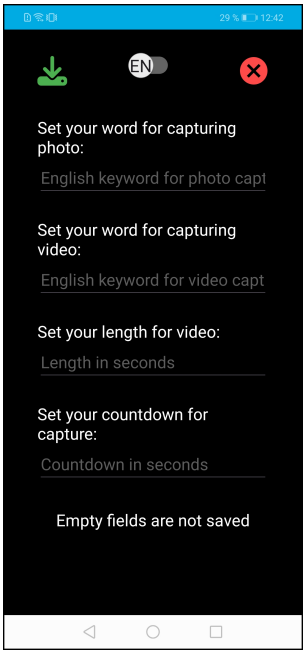
Obrázek 4.3: V pravém horním rohu je umístěno tlačítko, jež uživatele vrátí na hlavní obrazovku. Je červené, aby nesplývalo s textem. Text je rozdělen do skupin, které jsou odděleny malými mezerami. Horní část nese informace o nastavení slov a času, přičemž spodní část je nápověda k tlačítkům a zvukové signalizaci na hlavní obrazovce. Zvýraznění zlepšuje viditelnost důležitých informací.

4.3 Obrazovka nastavení

Nastavení čtyř proměnných informací uživatelem je řešeno pomocí textového vstupu, z nichž dva, délka videa a odpočtu, jsou číselné vstupy v sekundách. Nachází se zde také dvě tlačítka; jedno z nich je zpětné tlačítko a druhé slouží k uložení nových hodnot.

Přepínač nahoře uprostřed ukládá název modelu při každém přepnutí. Vize je, že změní jazyk aplikace (tzn. že všechny nápisy budou v češtině místo angličtiny). Klávesnice se uživateli zobrazí sama po kliknutí na textový vstup. Své nastavení uloží po kliknutí na tlačítko uložení. Poté se jeho hodnoty vloží do *SharedPreferences* (viz kapitola 4.5). Vyskakovací oznámení se kvůli klávesnici zobrazuje v horní části. Oznamuje, zda bylo nastavení uloženo.

Tlačítka pro uložení a návrat zpět mají stejnou velikost jako na jiné obrazovce. Definuji zde také nadpisy vstupů, což jsou jen textové prvky. Uživatelské vstupy i ostatní prvky jsou roztaženy po celé obrazovce s okrajem 40sp. Mezi nimi je mezera o velikosti 25sp. Poslední text je vycentrovaný. Tento soubor se jmenuje `activity_settings.xml` a prvky jsou propojeny s třídou `SettingsActivity`.



Obrázek 4.4: V horní části obrazovky se nacházejí dvě tlačítka. Zelené tlačítko vlevo označuje uložení nastavení a červené vpravo odkazuje na návrat na hlavní obrazovku. Uprostřed je přepínač mezi českým a anglickým modelem. Pod nimi se nacházejí čtyři textové vstupy s nadpisy, v nichž najdeme šedě popsánu očekávanou hodnotu. Dole je informace o prázdných polích.

4.4 Povolení práv aplikace

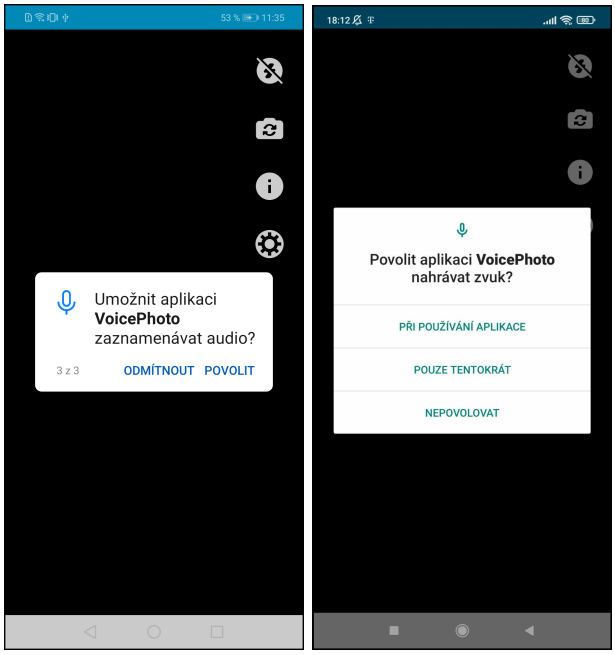
Pro rozpoznání hlasu z mikrofonu zařízení a správné fungování záznamu kamery je potřeba povolení. Povolení je rovněž nutné k uložení souboru do úložiště telefonu. Je zásadní, aby uživatel tato povolení udělil, jinak nemůže aplikace pracovat správně – nemá přístup k mikrofonu, nezobrazí náhled z kamery nebo nezpřístupní úložiště. Bez povolení se aplikace ukončí. Tato povolení se dají potvrdit při prvním spuštění aplikace nebo později v interním nastavení aplikací. Proto se v kódu nachází část, která zobrazuje modulární okno, ve kterém uživatel vybírá, zda povoluje použití potřebných hardwarových a softwarových práv touto aplikací. Po každém spuštění se následně kontroluje nastavení těchto definovaných proměnných.

Systém Android se vyvíjí; proto se různé funkce mohou využít v jedné verzi, ale v jiné jsou buď přepracovány, nebo z důvodu neaktuálnosti by se neměly používat vůbec. Určité části kódu jsou tedy psány dvakrát, kvůli změně ve verzi API 29, kdy uživatel nemusí povolovat přístup z důvodu zápisu do interního úložiště. Druhým řešením by bylo v kódu zapsat, že minimální API je 29, tudíž by nebylo třeba implementovat druhou verzi na nižší verze operačního systému. Pak by aplikace nebyla přístupná starším zařízením, jež lidé ještě stále používají. Vývojářům se někdy vyplatí to udělat tímto způsobem, může to však znamenat velký rozdíl v počtu stáhnutí a oblíbenosti.

Podle této distribuční tabulky, která se nachází přímo v aplikaci Android Studio, jsem se rozhodl vytvořit aplikaci na co největší počet Android zařízení. Minimální podporovanou verzí pro nahrání aplikace do Obchod Play je Android API 21. Počet zařízení s touto a vyšší

verzí je odhadnut na 99,6 % ze všech zařízení s operačním systémem Android, přičemž spousta nejstahovanějších aplikací stále podporuje[3] právě Android 5 s tímto API.

U některých aplikací se žádá o povolení až ve chvíli, kdy uživatel použije danou funkci. V mé aplikaci jsem k tomuto přistoupil poněkud jinak. Žádosti se zobrazují na začátku a všechny postupně [4]. To lze vidět u obrázku 4.5 vlevo. V levém dolním rohu modulárního okna se objeví počet povolení, který je třeba potvrdit. Vyvíjím aplikaci na celkem čtrnáct verzí Android, proto musím rozdělit zobrazení žádostí pro verze, jež používají omezené úložiště, a verze, které potřebují povolení na ukládání do interního úložiště. Vytvořil jsem si dvě skupiny žádostí a podle toho, jakou verzi zařízení má, se odkáži na danou skupinu.



Obrázek 4.5: Rozdíl mezi povolením uživatele je jednou z důležitých věcí, na kterou je třeba si dát pozor. Verze systému mohou používat i jiné funkce v rámci kódu. To může ovlivnit funkčnost aplikace, proto je dobré vždy odlišit přístupy do funkcí pro starší a novější implementace. Zde se dokonce liší v tom, že si uživatel může vybrat povolení jen v tento moment a po dalším spuštění se ho aplikace zeptá znovu.

Uživateli se ukáží postupně všechna potřebná povolení, jež musí odsouhlasit. Jelikož je toto demo aplikace, není zde do detailu promyšleno, co se stane při nepovolení daných přístupů. Pokud tato situace nastane, program se jednoduše ukončí, protože by nemohl pokračovat. Správně se to má vyřešit zobrazením modulárního okna s upozorněním, že je důležité povolit tyto přístupy kvůli správné funkčnosti aplikace. Pokud by tak uživatel stejně neučinil, měla by se zobrazit informace o tom, že nemá povolené použití mikrofonu, proto mu aplikace nefunguje tak, jak by měla.

[3] https://www.megumethod.com/blog/recommended-minimum-sdk-version-for-android-projects-copy

4.5 Předávání a ukládání uživatelských nastavení

Nastavení aplikace si může uživatel změnit a po opětovném zapnutí se načte. K tomu slouží třída `SharedPreferences`. Jedná se o nejjednodušší uchování vlastností aplikace ve formě strukturovaných údajů (klíč-hodnota). Podporuje datové formáty typu `string`, `boolean`, `int`, `long` či `float`. Uložené hodnoty jsou perzistentní, a proto jsou dostupné i po jejím ukončení. Využívá se k dlouhodobému ukládání dat jako jsou například konfigurace nebo nejvyšší dosažené skóre.

Při spuštění se načtou data pomocí metody `Activity.getPreferences(int mode)`, kde parametr značí mód přístupu. Má aplikace používá mód `MODE_PRIVATE` pro privátní přístup k uloženým datům. Zápis potřebuje instanci objektu `SharedPreferences.Editor`, jež lze získat metodou `SharedPreferences.edit`. Úpravy se provádí prostřednictvím metod:

- `putInt(String key, int value)`
- `putString(String key, String value)`
- `remove(String key)`

Potvrzení úprav se provede skrz metodu `SharedPreferences.Editor.commit`. Pokud je hodnota prázdná, vloží se základní přednastavená.

Kapitola 5

Implementace funkcí

Modely testovacích knihoven byly často integrovány jako modul v projektu. Bylo potřeba přidat závislosti pro práci s nimi. Pro použití vestavěné kamery v zařízení je zapotřebí závislostí, které umožňují s ní pracovat. Používám třídu `CameraX`, v pozdější kapitole 5.2 vysvětlím proč. V aplikaci implementuji dvě hlavní funkce. Jedná se o zachycení fotografie a videa při hlasovém příkazu uživatele. V této kapitole popíši, jak implementovat dané funkce a čeho se vyvarovat. Začnu s integrací hlasových knihoven a jejich modelů. V dalších se budu věnovat focení a natáčení, s tím souvisí zobrazení náhledu kamery. Nakonec vysvětlím ukládaní a testování jak knihoven, tak aplikace samotné. Abych použil vhodnou knihovnu, rozhodl jsem se je otestovat ještě předtím, než začnu implementovat hlavní funkčnost aplikace.

5.1 Výběr knihoven a jejich testování

Po vytvoření univerzálního projektu jsem vybral z celého seznamu pouze tři knihovny zmíněné v kapitole 2.3. Než začnu programovat samotný kód aplikace, musím v souboru Manifest definovat použití mikrofonu. V závislostech jsou funkce, které pracují s modelem. Hlavním krokem je správně vložit model rozpoznávače vedle projektu. V inicializační funkci je pak potřeba ho nastavit podle dokumentace knihoven. U knihovny Vosk to funguje pomocí metody `contains`. Jelikož se jedná o přepis řeči, hledám pouze slovo ve větě. Podobně pracuje i knihovna PocketSphinx. U vývojářské sady *Porcupine* byla slova označena během inicializace číslem podle pořadí. Tudíž kontrola probíhala pomocí čísel, nikoliv podle slov. Aplikace pak pokračuje spuštěním funkcí přiřazených ke klíčovým slovům.

Po výběru proběhlo testování všech vývojářských sad. Obnášelo to test úspěšnosti rozpoznání klíčového slova. Testování se účastnilo celkem dvacet testerů, každý z nich zopakoval slovo desetkrát. Testeři vždy vyslovili slovo v tichém prostředí (prázdná místnost, kde mluvili pouze oni) a poté slovo zopakovali v rušném prostředí, které mělo simulovat ruch během sportovní aktivity. V každém prostředí bylo tedy slovo nebo sousloví řečeno dvěstěkrát z důvodu zachycení různé výslovnosti a emocí řečníka, které mohou ovlivnit rozpoznání. Testovalo se celkem deset slovních spojení. V průběhu testování se potvrdilo, že znělé hlásky jsou snadněji rozpoznatelné než hlásky neznělé. Například u slova *now* je zvučná hláska n, ale zbytek zvučný není. Neznělá hláska S je například ve slově *shot*[1].

[1] Obě hlásky jsou zapsány v 7 bitovém převodu Mezinárodní fonetické abecedy. Velké S značí foneticky souhlásku ve slově *shin* (viz příloha A)

Picovoice

Podle tabulky 5.1 lze vidět, že tato knihovna má opravdu dobrou úspěšnost. Jedním z hlavních důvodu je využití neuronové sítě a specificky vytrénovaných klíčových slov. Bohužel má spoustu omezení, jako například počet uživatelů s aplikací. Díky tomu ji nelze vydat. Avšak pracovat s touto sadou bylo velice jednoduché, stačilo nahrát souvislosti a pak volat funkce této knihovny. Využít lze i jejich vytrénovaná přednastavená slova anebo si vytrénovat vlastní, jež se vloží do aplikace a při inicializaci se spustí model s tímto souborem. Omezení na počet klíčových slov není, ale počet vytrénovaných slov za měsíc je omezen na tři. Knihovnu lze stáhnout rovnou z veřejných repozitářů *mavenCentral* nebo *Google*. Stačí ji vepsat do závislostí, kde se definuje její číslo verze. Poté stačí už jen stáhnout vytrénovaná klíčová slova a vložit je do modelu.

Rozpoznání anglických slov pomocí Porcupine

slova	tiché prostředí	rušné prostředí
picture	94 %	78 %
snap	*nelze testovat*	
now	*nelze testovat*	
start	96 %	76 %
shot	*nelze testovat*	
take picture	98 %	93 %
snap it	94 %	85 %
start recording	97 %	94 %
take shot	95 %	90 %
ready action	96 %	93 %

Tabulka 5.1: Tabulka úspěšnosti rozpoznání anglického slova a sousloví při použití knihovny Picovoice. Test probíhal ve dvou prostředích – v tichém, kdy bylo slyšet pouze uživatele (v různých vzdálenostech od zařízení), a v rušném (při záznamu přednášky v angličtině anebo v ruchu města). *Porcupine* si vedl takřka bezchybně v tichém prostředí; funkčnost v rušném prostředí byla podobná rozpoznávači hlasu od Googlu. U slov, které mají méně než 5 fonémů, nejde vytrénovat neuronová síť, proto nebyly testovány.

CMU PocketSphinx

Při testování se vyskytovaly chyby, kdy tester řekl: „Start“. Aplikace měla v tom okamžiku vypsat stav na obrazovku, ale nestalo se tak. Při testu se stejným slovem, ale odlišnou výslovností: [stárt] se na obrazovce objevil text „Keyword spotted“, klíčové slovo bylo tedy rozpoznáno. Při procházení anglické fonetické abecedy (viz příloha A) se ukázalo, že pokud se toto konkrétní slovo vysloví rychle a krátce, může se stát, že nebude dobře rozpoznáno. Tento problém lze vyřešit jazykovým modelem. V případě, že uživatel mluví ve větách, dokáže jazykový model opravit špatně odposlechnutá slova. Avšak pokud se použijí jen klíčová slova, nelze větný kontext využít. U tabulky 5.2 je úspěšnost knihovny velmi nízká. Používá sice obsáhlý slovník, ale hodně záleží na slově, které se má rozpoznat. V rušném prostředí měla aplikace problém vůbec slovo zachytit. Přestože má funkci pro rozpoznání klíčových slov, je podobně rychlá jako přepis řeči.

Rozpoznání anglických slov pomocí knihovny CMU Sphinx

slova	tiché prostředí	rušné prostředí
picture	54 %	11 %
snap	44 %	5 %
now	56 %	7 %
start	43 %	9 %
shot	41 %	7 %
take picture	24 %	8 %
snap it	26 %	13 %
start recording	22 %	9 %
take shot	10 %	1 %
ready action	15 %	4 %

Tabulka 5.2: Tabulka úspěšnosti rozpoznání anglického slova a sousloví při použití knihovny CMU Sphinx. Test probíhal ve dvou prostředích – v tichém, kdy bylo slyšet pouze uživatele (v různých vzdálenostech od zařízení), a v rušném (při záznamu přednášky v angličtině anebo v ruchu města). Vývojářská sada má problém správně rozlišit klíčová slova. Zachytává dokonce i ticho a výslovnost hraje velkou roli. U některých testerů se nepodařilo ani rozpoznat jejich řeč.

Alphacei Vosk

Na rozdíl od Sphinx není tato knihovna natolik flexibilní a nemá ani funkci pro rozpoznávání klíčových slov. Přesto dokáže být úspěšnější a rychlejší.

Jak jsem již zmínil, tato knihovna nabízí integraci s každým modelem, který mají vývojáři knihovny na stránkách nebo vlastním modelem, jež podléhá pravidlům knihovny. Proto jsem využil český model, který by měl mít horší úspěšnost než anglický, podle jejich rovnice (chybovost slov/rychlost[2]). Anglický model má 9,85 WER/s na datasetu *librispeech test-clean*[3] a 10,38 WER/s na *tedlium*[4]. Český model byl testován tzv. *křížovou validací*[5] a získal jen 21,29 WER/s. Ovšem poté, co byl otestován, nebyl horší, nýbrž na stejné úrovni jako ten anglický (viz tabulka úspěšnosti 5.4). Nevýhodou je fakt, že slovník sice obsahuje pražské i moravské nářečí, ale neobsahuje např. slovo „pořiď“. Dle výsledků mého testování má vliv i výslovnost anglických slov. Čeština je náš mateřský jazyk, tím pádem pro nás přirozenější. Ve finální aplikaci, již budu publikovat, bude obsahovat oba modely, tedy anglický i český. Někteří uživatelé nemusí umět anglicky, proto mohou využít svého mateřského jazyka.

Důležité u této knihovny je vybrat správná slova. Knihovna má výborné výsledky v tichém prostředí, jakmile se však člověk ocitne v rušném prostředí, činí jí problém vytěsnit okolní zvuky. Proto je výhodné použít ve slově znělé hlásky, jsou totiž lépe slyšet ve změti zvuků. Dalším doporučením je délka slova; čím delší slovo je, tím snadnější je jeho registrace

[2]Chybovost slov jinak také Word error rate (WER) je základní metrika hodnocení výkonu systému rozpoznávání hlasu nebo strojového překladu. Výpočet je založen na Levenštajnově vzdálenosti, jež udává počet zaměněných písmen ve dvou řetězcích. Hodnota se následně dělí rychlostí zpracování.

[3]https://www.openslr.org/12

[4]https://www.openslr.org/51/

[5]Křížová validace, *cross validation* – metoda, při které je jedna datová sada rozdělena do podmnožin a používá se k ověření přesnosti modelu. Obecně to zahrnuje rozdělení dat do trénovací sady a testovací sady.

modelem. Optimální délka slova je 5–10 písmen. Někdy však více znamená méně; pokud uživatel použije více slov, nemusí model pracovat správně.

Rozpoznání anglických slov pomocí Vosk

slova	tiché prostředí	rušné prostředí
picture	86 %	69 %
snap	82 %	61 %
now	93 %	76 %
start	84 %	59 %
shot	79 %	58 %
take picture	49 %	38 %
snap it	89 %	5 %
start recording	40 %	37 %
take shot	20 %	2 %
ready action	31 %	14 %

Tabulka 5.3: Tabulka úspěšnosti rozpoznání anglického slova a sousloví při použití sady Vosk. Test probíhal ve dvou prostředích – v tichém, kdy bylo slyšet pouze uživatele (v různých vzdálenostech od zařízení), a v rušném (při záznamu přednášky v angličtině anebo v ruchu města). Překvapilo mě, že více slov nepomůže rozpoznání. Avšak aplikace byla úspěšnější než Sphinx, a to výrazně, i přesto, že vychází ze stejného softwaru a že se jedná o pouhý přepis řeči.

Rozpoznání českých slov pomocí Vosk

slova	tiché prostředí	rušné prostředí
foto	97 %	38 %
video	96 %	36 %
teď	98 %	19 %
start	94 %	47 %
sýr	85 %	21 %
akce	99 %	82 %
zachytit fotografii	96 %	4 %
start videa	84 %	33 %
připravit akce	99 %	79 %

Tabulka 5.4: Tabulka úspěšnosti českého slova a sousloví při použití sady Vosk. Test probíhal ve dvou prostředích – v tichém, kdy bylo slyšet pouze uživatele (v různých vzdálenostech od zařízení), a v rušném (při záznamu přednášky v angličtině anebo v ruchu města). V tichém prostředí si model vedl velmi dobře, v rušném prostředí si vedl hůře než anglický model. Při správném vyslovení znělých hlásek ve slově funguje bezproblémově i v rušném prostředí (viz slovo „akce“).

5.2 Implementace kamery

Ke knihovně Vosk, kterou jsem vybral podle testování jako nejvhodnější, potřebuji naprogramovat kvůli důkladnějšímu testu zbývající část aplikace. Za prvé jsem si zjišťoval, jak z aplikace přistupovat k vestavěnému fotoaparátu. První test proběhl s použitím ručního focení. Systém Android dokáže otevřít jednoduchou verzi fotoaparátu, kde lze pouze pořídit fotografii a uložit ji do galerie. Nelze ji však ovládat hlasem, pouze dotykem. První knihovnou, jež umožňuje ovládat kameru zařízení pomocí jiné aplikace, je `Camera1`, ta je však zastaralá a nové verze Androidu ji nepoužívají. Byly vytvořeny novější verze, které ji nahrazují, `CameraX`a `Camera2`. Po přečtení dokumentace těchto knihoven jsem si vybral první z nich. Druhá pracuje s funkcemi hluboko v kódu, a proto obsahuje mnoho funkcí. Univerzální knihovna `CameraX` obsahuje funkce na volání těchto důmyslnějších funkcí, a je tak mnohem jednodušší a rychlejší na implementaci. Dokáže pracovat s jakýmkoli zařízením od jakéhokoliv výrobce. Proto použiji `CameraX` na testování aplikace a rychlou implementaci. Díky těmto knihovnám je aplikace schopna spouštět a vypínat svítilnu, pokud je k dispozici. Dále může změnit směr focení, neboli použití přední nebo zadní kamery.

V další fázi aktualizace a upravení aplikace lze využít `Camera2`, jelikož nabízí mnohem větší variabilitu a hlavně výběr kamery. Dokáže pracovat s více kamerami zároveň a vybírat mezi nimi. Nyní přiblížím, jak propojit kameru v kódu.

Zobrazení náhledu kamery

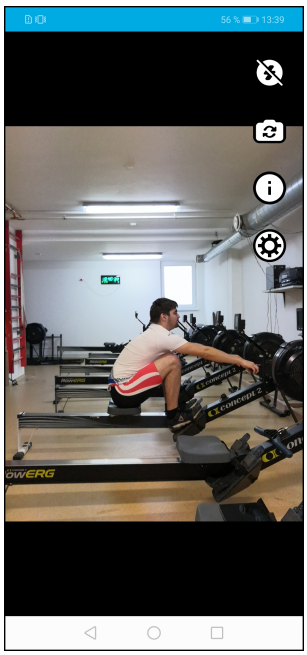
Aplikace se spustí s hlavní obrazovkou, která zobrazuje náhled skrze fotoaparát. Aby se zobrazil pohled skrze kameru telefonu, je potřeba do uživatelského prostředí vložit objekt zvaný `androidx.camera.view.PreviewView`[6]. V mém případě je nastavený na velikost nadřazeného elementu, avšak nikdy není přes celou obrazovku. Jeho velikost se určuje podle rozlišení kamery v zařízení a inicializuje se během spuštění hlavní obrazovky. To znamená, že když se uživatel přepne na jiný pohled, náhled se vypne a musí se znovu propojit. Pokud není povolen přístup ke kameře, kamera se nezapne a zůstane pouze černé pozadí. Po zapnutí se definuje, zda se zobrazí přední či zadní fotoaparát, pokud to zařízení umožňuje. Používám k tomu třídu `CameraSelector` v balíčku `androidx.camera.core`. Základní hodnota je nastavena na zadní kameru. Uživatel ji dokáže změnit po kliknutí na ikonu *otočení kamery*.

O propojení fotoaparátu s aplikací se stará funkce `startCamera`, ve které žádám o instanci `ProcessCameraProvider`. Tato instance nese obraz kamery, jež je promítaný na obrazovku. Poté k ní připojím dané instance pro vytvoření fotografie a videa. Tvoří se pomocí `ImageCapture.Builder` a `Recorder.Builder`. Tyto instance jsou spuštěny funkcemi, jež ovládá uživatel hlasem. Normálně se spouští pomocí spouště/tlačítka. Moje aplikace pořizuje fotografie na základě rozpoznání klíčového slova, které si uživatel volí podle sebe. Když je slovo rozpoznáno, aplikace spouští funkci pro zachycení fotografie či videa. Lze také nastavit odpočet a zvukově se signalizuje jak rozpoznání slova, tak i sekundový odpočet.

[6] https://developer.android.com/media/camera/camerax/preview

Nastavení zachycení fotografie

Vyfocení fotografie obstarává funkce `takePicture`. Na jejím začátku je vytvořen název pro soubor. Využívá se systémový čas zařízení v milisekundách, což je typické pro ukládání fotografií i videí. Fotografie je uložena ve formátu `jpeg`. Ve funkci se nastavují parametry k záznamu fotografie (název, formát, orientace). Tím se vytvoří instance fotografie, jež nese informace o uložení. Vyfocený soubor se ukládá mezi ostatní fotografie zařízení. Jedná se o základní adresář, ke kterému lze přistoupit pomocí definované proměnné `DIRECTORY_PICTURES` ve třídě `Enviroment`. Pokud je vše výše zmíněné nastaveno, zařízení provede zaznamenání a pokusí se o uložení na dané místo funkcí `onImageSaved`. Po provedení tohoto kroku zobrazí vyskakovací textu na obrazovce s informací o pořízení fotografie. V opačném případě vypíše aplikace chybovou hlášku s kódem chyby. V průběhu celého procesu bylo rozpoznání pozastaveno. Pokračuje až po dokončení záznamu. Tímto způsobem je vyřešeno překrývání příkazů uživatele. Například pokud se natáčí video, očekává se, že uživatel nechce zároveň pořídit fotografii nebo spustit druhé paralelní natáčení.



Obrázek 5.1: Ukázka zobrazení náhledu fotoaparátu.

Nastavení natočení videa

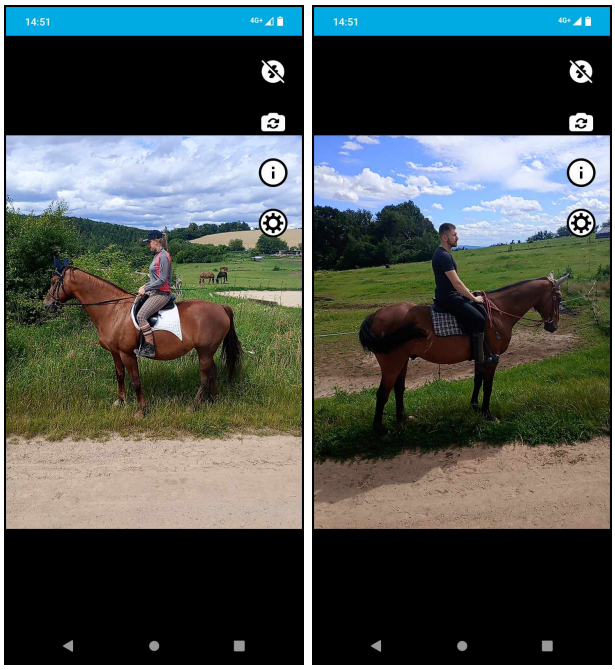
Natočení videa zprostředkovává funkce `captureVideo`. Hned na jejím začátku se testuje, zda se video právě nenatáčí; v případě, že ano, natáčení se zastaví. Tato funkce chrání před paralelním natáčením videa. Stejně jako u focení definuji na začátku informace o souboru (název složený z lokálního času, formát a adresář pro uložení záznamu). Video má nastavenou délku definovanou uživatelem, po které se vypne. To zprostředkovává třída `Timer`, jež nabízí funkci odpočet. Během přípravy natáčení třídou `Recorder.prepareRecording`[7] se načtou definované informace o souboru. Na začátku se zobrazí červená tečka nahoře uprostřed, která signalizuje probíhající natáčení. Po dokončení záznamu je video uloženo.

[7] https://developer.android.com/media/camera/camerax/video-capture

Pokud uživatel nepovolil přístup do adresáře, záznam se nezačne ani natáčet. Podobně jako u pořízení fotografie aplikace vypíše na obrazovku zprávu, zda bylo nebo nebylo video uloženo. Stejně tak provede zvukovou signalizaci při zachycení slova, sekundový odpočet a při ukončení záznamu.

5.3 Testování aplikace

Testování probíhalo nejdříve formou debugování. Hlavním cílem bylo odhalit chyby v implementaci nebo funkčnosti aplikace. Motivací bylo zúžit výběr z knihoven, aby testeři nemuseli testovat nedostačující aplikace. Byly proto vydány pouze dvě aplikace s využitím dvou různých knihoven. Jednalo se o uzavřené testování; přístup k nim měli pouze mí známí, testeři a vedoucí práce. Díky tomu jsem byl schopen odladit chyby na co nejvíce typech zařízení. Otestovalo se celkem osm různých verzí systému Android na čtrnácti typech mobilních zařízeních od pěti různých výrobců, z nichž jeden byl tablet. Aplikaci testovalo celkem dvacet testerů buď v reálném prostředí, nebo v uměle vytvořeném, které mělo simulovat to reálné.



Obrázek 5.2: Testeři aplikaci testovali při běžných činnostech nebo během provozování svých zálib. Na obrázcích lze vidět test, který probíhal s jezdci na koni. V tuto chvíli se testeři snažili zaznamenat, zda mají jezdci správný sed na koni (rameno, kyčel, pata musí být v jedné rovině). Zařízení umístili na stativ a spouštěli si zachycení fotografie.

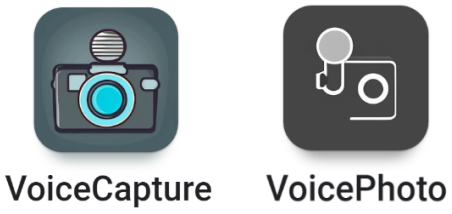
U obou aplikací jsem zvolil stejné uživatelské rozhraní i proto, aby to nemělo vliv na výsledky v testování rozpoznání hlasu. V průběhu testování změnily ikony v aplikaci často svůj vzhled, aby byly lépe rozpoznatelné. Například díky testerům jsou výraznější důležitá slova v informačním náhledu (žlutá barva). Od prvních uživatelů přišlo pár připomínek, například úprava funkčnosti ikon nebo přehlednost informační obrazovky. První problém se vyřešil opravou logiky tlačítek a druhý přepsáním informací na hesla s dvojtečkou. Přehlednosti bylo dosaženo zabarvením důležitých nastavení nažluto a seskupením podobných

informací k sobě, aby obrazovka nevypadala jako stránka v knize. Obarvení se dočkala i tlačítka „zpátky“ a „uložit“, kde návrat zpět na hlavní obrazovku dostal červenou barvu a uložení nastavení zelenou. Po tomto odladění jsem začal s testery průběžně testovat, která z aplikací je lepší. Po několika testováních se ukázalo, že s aplikací `VoicePhoto` se pracuje mnohem snadněji. Většina testerů se shodla na tom, že `VoiceCamera` je znatelně horší kvůli tomu, že nezachytila slova nebo nerozpoznala ta klíčová. `VoicePhoto` si vedla o poznání lépe. Usoudil jsem proto, že se budu věnovat pouze této aplikaci s knihovnou Vosk.

Jak je zmíněno v kapitole 2.3, testování knihoven na rozpoznání hlasu proběhlo na začátku. V tuto chvíli testuji finální verzi s vybranou knihovnou Vosk. Uživatelé ji testovali podle stejných pokynů a stejných slov v podobných prostředích. Jedinou změnou bylo, že testovali i focení a natáčení videa. To probíhalo buď sledováním displeje, kdy jsme se nesnažili nic vyfotit, nebo byl telefon ve stativu a tester byl od něj vzdálený tak, aby byl vidět na displeji. Aplikace měla velmi podobnou úspěšnost rozpoznání klíčového slova jako v kapitole 5.1. Uživatelé si pochvalovali nejen češtinu, ale i to, že aplikace dokáže fungovat *offline*. Někteří ji dokonce využívali v běžném životě mimo testování. Jejich poznatky mi pomohly odladit některé chyby či nedokonalosti.

5.4 Publikace na Obchod Play

Po prvotním testování, které mělo odhalit funkční chyby před ostrým vydáním, jsem vydal aplikaci veřejně, aby byla lépe přístupná lidem (nebo testerům). Vydání aplikace mi značně usnadnilo testování. Nemusel jsem testery přidávat na seznam ani nebyl důvod zasílat nové verze pomocí souboru `APK`. Testeři mohli aplikaci otestovat ihned, stačilo si pouze stáhnout aktuální verzi z Obchodu Play. Je to značné zrychlení, jelikož se aplikace automaticky aktualizuje z Obchodu Play po připojení na Wifi. Urychlilo to proces testování a zároveň se mi později i rychleji schvalovala aplikace společností Google. První schválení uživatele a aplikace trvá přibližně 10–14 dní. Většinou stane, že aplikace podmínkám vydání nevyhovuje, čímž se zdrží její vydání. Proto jsem se rozhodl aplikace vydat co nejdříve po vyplnění potřebných informací, které jsou nutné pro vytvoření záznamu v obchodu. Pro každou aplikaci jsem vytvořil logo a pořídil snímky obrazovky při používání na ukázku. Další podstatnou věcí jsou názvy aplikací; knihovna PocketSphinx má jméno VoiceCamera a Vosk se jmenuje VoicePhoto. Obě byly vydány do obchodu s aplikacemi, zprvu pouze interně jako zkušební verze. Poté jsem vydal veřejně jen VoicePhoto.



Obrázek 5.3: Ikony dvou aplikací, které byly testovány uživateli. Vlevo se nachází aplikace využívající knihovnu PocketSphinx, vpravo knihovnu Vosk. Mají odlišné ikony, aby byly snadno od sebe rozpoznatelné. Obě zobrazují fotoaparát s mikrofonem. Aplikaci napravo lze stáhnout a je stále ještě k vidění na Obchodu Play[8]. Druhá aplikace se zobrazuje pouze testerům.

[8] https://play.google.com/store/apps/details?id=vut.example.voskapp&hl=en_AU

Kapitola 6

Závěr

Cílem mé práce bylo vybrat vhodnou knihovnu pro rozpoznání hlasu na operační systém Android a následně ji využít pro demo aplikaci, která bude pořizovat fotografie a natáčet videa. Abych mohl takovou aplikaci vytvořit, nastudoval jsem potřebné materiály týkající se tvorby hlasu a hlasového rozpoznávání. Otestoval jsem zmíněné knihovny s mnoha uživateli. Tím jsem měl možnost vyzkoušet různé typy hlasu i výslovností. Výsledkem testování byla úspěšnost rozpoznání a připomínky testerů. Na základě těchto informací a vědomostí jsem vybral nejvhodnější vývojářskou sadu. Ta nabízí množství různých jazykových modelů, např. češtinu. Po vybrání jsem nastudoval také implementaci zachycení fotografie, natočení videa a následné uložení.

Z důvodu urychlení testování byla aplikace vydána v Obchodu Play, kde si ji má možnost stáhnout kdokoli. Za pomoci mnoha testerů byla možnost ji otestovat na více zařízeních s různou verzí systému Android. Podařilo se odladit aplikaci na vícero telefonech. Aplikaci lze využít i mimo sportovní prostředí například jako zrcadlo nebo focení selfie. Uživatelé ji používali i mimo testování a oblíbili si ji.

Aplikaci lze vyvíjet dále pro jiné platformy a případně vylepšit nastavení funkcí hlasem. Dalším vylepšením může být použití třídy `Camera2`, která umožní vybírat kameru zařízení (pokud má více než jednu), případně vytvořit vlastní jazykový model pro tuto knihovnu či upravit uživatelské prostředí. Tyto skutečnosti již nebyly cílem mé práce.

Vzhledem k fotografování a videu je dobré se rozhodnout, zda aplikaci udržovat pro starší verze Androidu. Používají totiž zastaralé grafické prostředí, což může zapříčinit nedokonalou funkčnost starších zařízení. Po dokončení nynější verze aplikace se vydá nová verze Andoridu. Důležité je zvážit, zda by aplikace měla fungovat i na telefonech starších osm let. Aplikace se ukázala jako užitečná, plánuji ji ponechat na Obchodu Play pro uživatele.

Literatura

[1] CMUSphinx. *Building an application with sphinx4 – CMUSphinx Open Source Speech Recognition* [online]. [cit. 2023-11-1]. Dostupné z: https://cmusphinx.github.io/wiki/tutorialsphinx4/.

[2] Google. *Android Mobile App Developer Tools – Android Developers* [online]. [cit. 2023-10-16]. Dostupné z: https://developer.android.com/.

[3] Google LLC. *Google Play Console* [online]. [cit. 2023-12-1]. Dostupné z: https://play.google.com/console/about/.

[4] Griffiths, D. Head First Android Development. In:. 2nd edition. Beijing: O'Reilly, 2017, kap. 19. ISBN 978-1-4919-7405-6.

[5] Krug, S. *Don't make me think, revisited: a common sense approach to web usability.* 3rd edition. San Francisco: New Riders, 2014. ISBN 978-0321965516.

[6] Lacko, Ľ. *Mistrovství – Android.* 1. vydání. Brno: Computer Press, 2017. Mistrovství. ISBN 978-80-251-4875-4.

[7] Lepak, Ł., Radzikowski, K., Nowak, R. a Piczak, K. J. Generalisation Gap of Keyword Spotters in a Cross-Speaker Low-Resource Scenario. *Sensors.* 2021, sv. 21, č. 24. DOI: 10.3390/s21248313. ISSN 1424-8220.

[8] Marsh, J. *UX for Beginners: A Crash Course in 100 Short Lessons.* Sebastopol, CA: O'Reilly Media, 2016. ISBN 978-1491912683.

[9] Marsicano, K. *Android programming: the Big Nerd Ranch guide.* 4th edition. Atlanta, GA: Big Nerd Ranch, 2019. ISBN 978-0135245125.

[10] Picovoice. *On-device Voice Recognition Intro – Picovoice Docs* [online]. [cit. 2023-10-19]. Dostupné z: https://picovoice.ai/docs/.

[11] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O. et al. The Kaldi Speech Recognition Toolkit. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding.* IEEE Signal Processing Society, Prosinec 2011. IEEE Catalog No.: CFP11SRW-USB.

[12] Psutka, J. *Mluvíme s počítačem česky.* Vyd. 1. Praha: Academia, 2006. ISBN 80-200-1309-1.

[13] Sol, N. Keyword spotting, What is it? *Https://medium.com/.* Jan 2023.

[14] Tidwell, J. *Designing Interfaces: Patterns for Effective Interaction Design.* 3rd. Sebastopol, CA: O'Reilly Media, 2020. ISBN 978-1492051964.

[15] Škodová, E., Jedlička, I. a kol.. *Klinická logopedie.* Praha: Portál, 2003. 616 s. ISBN 80-7178-546-6.

Příloha A

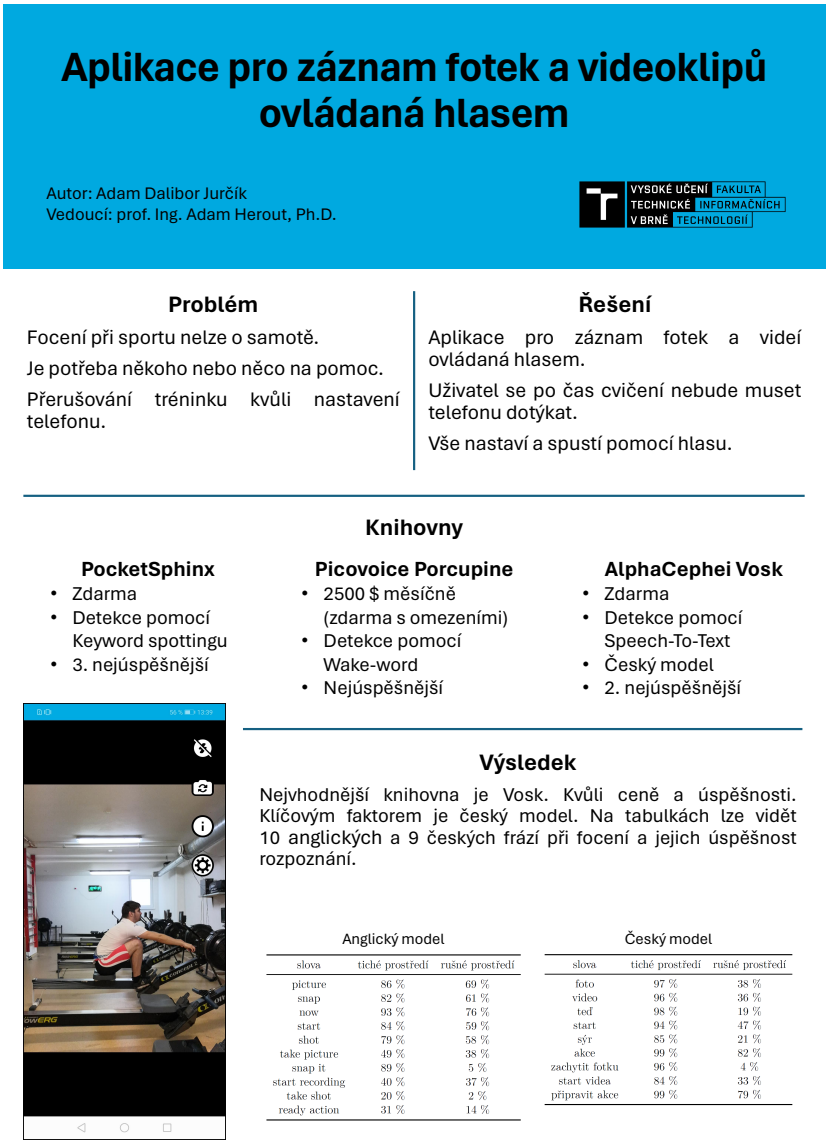
Fonetická abeceda britské angličtiny

	SAMPA	Slovo	Traskripce		SAMPA	Slovo	Traskripce
vokály	I	*pit*	pIt	plozivy	p	pin	pIn
	e	*pet*	pet		b	bin	bIn
	{	*pat*	p{t		t	tin	tIn
	Q	*pot*	pQt		d	din	dIn
	V	*cut*	kVt		k	kin	kIn
	U	*put*	pUt		g	give	gIv
	@	*another*	@"nVD@	frikativy	f	fin	fIn
	i:	*ease*	i:z		v	vim	vIm
	eI	*raise*	reIz		T	thin	TIn
	aI	*rise*	raIz		D	this	DIs
	OI	*noise*	nOIz		s	sin	sIn
	u:	*lose*	lu:z		z	zing	zIN
	@U	*nose*	n@Uz		S	shin	SIn
	aU	*rouse*	raUz		Z	measure	"meZ@
	3:	*furs*	f3:z		h	hit	hIt
	A:	*stars*	stA:z	sonory	m	mock	mQk
	O:	*cause*	kO:z		n	knock	nQk
	I@	*fears*	fI@z		N	thing	TIN
	e@	*stairs*	ste@z		r	wrong	rQN
	U@	*cures*	kjU@z		l	long	lQN
afrikány	tS	chin	tSin		w	wasp	wQsp
	dZ	gin	dZin		j	yacht	jQt
Pozn.	/i:/, /u:/, /3:/, /A:/, /O:/ se mohou značit /i/, /u/, /3/, /A/, /O/ /e/ se značí také jako /E/, např. pet /pEt/			alofony	?	network	ne?w3:k
					x	loch	lQx

Tabulka A.1: V tabulce jsou příklady hlásek ve slovech. Pro jejich zápis je využita SAMPA – Speech Assessment Methods Phonetic Alphabet (Řečové vyhodnocení metod fonetické abecedy). Jedná se o 7 bitový převod Mezinárodní fonetické abecedy. Tabulka byla převzata z knihy Mluvíme s počítačem česky.

Příloha B

Plakát



Obrázek B.1: Plakát o aplikaci

Příloha C

Obsah přiloženého paměťového média

```
/
├── latex/ ..... Zdrojové soubory textu ve formátu LaTeX
├── poster.pdf ..... Plakát o bakalářské práci
├── README.txt ..... Informace pro spuštění aplikací
├── UseCaseVideo.mp4 ..... Videozáznam použití aplikace
├── VoiceCamera/ ..... Adresář aplikace VoiceCamera
│   ├── app-release.apk ..... Instalační soubor aplikace
│   └── src/
│       ├── models/src/main/assets/sync/
│       │   ├── en-us-ptm/ ..... Anglický model
│       │   ├── cmudict-en-us.dict ..... Anglický slovník modelu
│       │   └── keywords ..... Seznam klíčových slov
│       ├── aars/ ..... Android archiv knihovny
│       └── app/src/main/
│           ├── java/edu/cmu/pocketsphinx/app/ ..... Zdrojové soubory Java
│           ├── res/ ..... Zdroje aplikace
│           └── AndroidManifest.xml ..... Manifest aplikace
└── VoicePhoto/ ..... Adresář aplikace VoicePhoto
    ├── app-release.apk ..... Instalační soubor aplikace
    └── src/
        ├── app/src/main/
        │   ├── java/vut/example/voskapp/ ..... Zdrojové soubory Java
        │   ├── res/ ..... Zdroje aplikace
        │   └── AndroidManifest.xml ..... Manifest aplikace
        └── models/src/main/assets/
            ├── model-cz/ ..... Český model
            └── model-en-us/ ..... Anglický model
```