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

155071

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

Při obhajobě semestrální části projektu je požadováno:
Body 1, 2 a 3, značné rozpracování bodů 4 a 5.

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:	9.5.2024
Datum schválení:	9.11.2023

Abstrakt

Cílem této práce je vytvořit aplikaci, která řeší problémy spojené s nahráváním videa a fotografií téže osoby při sportovních aktivitách. Tyto problémy jsou například, potřeba mít u sebe dálkové ovládání nebo neustálé přerušování činnosti z důvodu ručního spuštění. Lze situaci 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 obsahuje vysvětlení ukládání těchto souborů a to, jak ovládat hardware zařízení jako například fotoaparát. Práce obsahuje podrobnější popis dvou přístupů k rozpoznání hlasu a popisuje 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 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
3. května 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

1 Úvod **2**

2 Rozpoznání hlasu **3**
2.1 Princip 4
2.2 Lidská tvorba řeči 5
2.3 Vývojářské sady pro hlasové rozpoznání 6
2.4 Výběr a testování 7

3 Vývoj aplikace na OS Android **13**
3.1 Základní soubory a nastavení 13
3.2 Rozložení obrazovky 15
3.3 Vydání v obchodě 16

4 Návrh aplikace **18**
4.1 Hlavní obrazovka 18
4.2 Informační obrazovka 20
4.3 Obrazovka nastavení 21
4.4 Povolení práv aplikace 22
4.5 Předávání a ukládání uživatelských nastavení 24

5 Implementace funkcí **25**
5.1 Integrace knihoven 25
5.2 Implementace záznamu 26
5.3 Publikace na Obchod Play 27
5.4 Testování 28

6 Závěr **29**

Literatura **30**

A Fonetická abeceda britské angličtiny **32**

B Plakát **33**

C Obsah přiloženého paměťového média **34**

Kapitola 1

Úvod

Cílem této práce je vytvořit alternativní fotografickou aplikaci ovládanou hlasem. Její primární využití bude při sportu a skupinových fotografií. Hlavním úkolem je umožnit uživateli ovládat jeho telefon na dálku, kdy je zařízení například ve stativu. V dnešní době se používají *bluetooth* spouště, gesta anebo zpoždění časovačem. Při sportování tlačítko překáží, proto má tato aplikace sportovce zbavit potřeby mít k dispozici sportovního partnera, aby ho natočil. Pomocí hlasu by sám ovládal záznam videa, fotky, blesk a různé základní funkce jako u vestavěné kamerové aplikace. Řešení bude mít v sobě i časovač, kterým odpočítá spuštění záznamu a poté udělá zvuk při ukončení. Následně se vytvořený soubor uloží do galerie, kde si fotky 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 už existující knihovny, ze kterých 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. Půjde ji stáhnout z Obchodu Play, neboli bude k dispozici komukoli *online*. Cíl je, aby byla zdarma, tudíž využití drahých řešení na rozpoznání hlasu není na místě.

V úvodu práce vysvětluji fungování rozpoznání hlasu a jaké nabízí funkce, přiblížím i lidskou tvorbu hlasu. V další kapitole se budu věnovat vývoji aplikací na operační systém Android. Naznačím, co je dobré si nachystat, jakým způsobem vydat aplikaci na Obchod Play. Zároveň uvedu možnosti, které není vhodné použít. Věnuji se návrhu aplikace více podrobněji, jelikož aplikaci lze naprogramovat i na jiné platformy. Následně představím knihovny, jež otestuji a popíši implementaci zachycení fotky a videa.

Kapitola 2

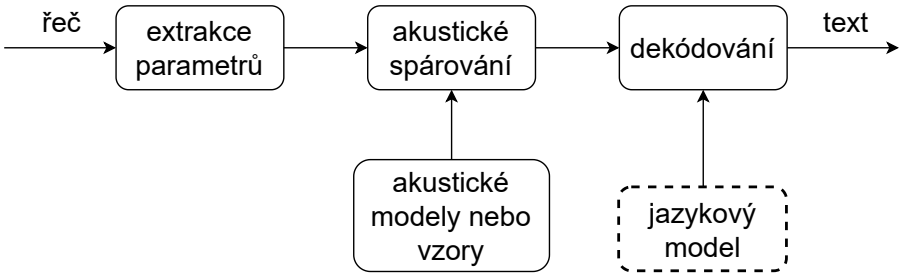
Rozpoznání hlasu

V dnešní době je tato technologie dosti pokročilá a nabízí mnoho funkcí, jež lze využít ať už v telefonech, autech, vlastně kdekoliv, kde se využívá nějaká výpočetní technika. Důležitým prvkem úspěšného rozpoznání hlasu je kvalitní mikrofon, který dokáže zachytit a přenést zvuky do zařízení s co nejmenším zkreslením. Skrytý Markovův model byl dlouho standardem v rozpoznávání hlasu, ale s nástupem neuronových sítí se tato metoda stala preferovanější. Neuronové sítě mají schopnost adaptovat se na různorodé řečové vzory a lépe zvládat složité situace, což přispívá k vyšší přesnosti rozpoznání. Rychlost je také klíčovým faktorem, 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 různé požadavky na rozpoznávání hlasu, a proto je důležité, aby rozpoznávací systémy byly flexibilní a modulární, aby mohly být přizpůsobeny specifickým potřebám a použitím. Jako například v mé aplikaci, kdy si uživatel bude moci nastavit svá klíčová slova pro dané funkce. Mým cílem je najít knihovnu, která je dostupná, rychlá a přesná.

Kromě samotného hlasového rozpoznávání mohou moderní systémy nabízet také další funkce, jako je překlad do různých jazyků, generování textu na základě mluveného slova nebo identifikace emocí z hlasu. Některá řešení dokonce nabízí i rozpoznaní uživatele neboli jeho hlasu, to znamená, že když vlastník zařízení řekne „Ok Google“, tak aplikace zareaguje. Pokud nějaký cizí člověk pokusí říct stejný příkaz, zařízení nejeví žádnou známku reakce. Způsob, jak toho dosáhnout, je jednoduchý, uživatel po přečtení pár vět do zařízení nahraje svůj hlas. Následně se pak kontroluje, zda je nějaká podobnost mezi uloženým a poslechnutým zvukem. Lze takto udělat i reprodukci hlasu. Uživatel namluví předepsané souvětí. Aplikace rozkouskuje věty na slova a ty na fonémy. Telefon podle fonetického zápisu tvoří z uložených fonémů slova. Tato funkce není novinkou, už mnohem dávno v minulém století byla řeč reprodukována. Nikdy k tomu ale nemělo přístup tolik uživatelů a tak jednoduše.

2.1 Princip

Při rozpoznání hlasu se využívá mnoho znalostí z různých vědních oborů. Jedná se o multidisciplinární přehled, jak přírodních, technických i humanitních. Vzhledem k vytváření různých modelů se jedná o fysiologii, kde vývojáři musí pochopit funkci a propojenost sluchového a hlasového ústrojí. Další problematikou je akustika, kdy fyzikální mechanismy jsou prioritou, kvůli tvorbě a slyšení řeči. Pak se dostáváme ke zpracování signálu, při kterém se využívá mnoho oblastí jako je modelování, spektrální analýza anebo klasifikace vzorů. Klíčová je fonetika, zaměřuje se na zvukové slyšení hlasu, fonologie studuje zase zvukovou stránku jazyku, jehož součástí jsou tzv. fonémy[1]. V českém jazyce je jich 36, v anglickém až 42. Prozódie rozebírá zvukovou stránku jazyka, jedná se o přízvuk, melodii a tak dále. Cíleněji poukážu v kapitole 2.2. Spolupráce s lingvisty a odborníky zabývající se vývojem, rozvojem nebo zpracováváním i produkcí lidské řeči může pomoci lépe porozumět jazykovým nuancím a specifikům, což může vést k vylepšení přesnosti zpracovávání a porozumění rozpoznávacích systémů.



Obrázek 2.1: Schéma nejzákladnějšího rozpoznávače. 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 přepíšou do textu a pokud rozpoznávač obsahuje jazykový model, tak 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 a zachování pouze těch, které odpovídají lidské řeči. Jedná se hlavně o zvuky auta, sbíječky a jiného zvukového šumu. Díky této fázi následně pracuje akustický model pouze s rozpoznávanou částí zvuku a dokáže správně fungovat. Porovnává si z fonetického slovníku hlásky, ze kterých poskládá slovo. Využívá se k tomu skrytý Markovův model (dále jen HMM[2] a má uvnitř stavy, jež označují hlásky a někdy kvůli kompaktnosti i celá slova nebo vytrénované neuronové sítě. Jejich prvním úkolem je zkontrolovat, jestli jde slyšet fonémy správně. Tento proces probíhá porovnáním statistické pravděpodobnosti hlásek a nejpravděpodobnější vybere. Pravděpodobnosti jsou vypočítané 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 slovo *„auto“* bude pravděpodobnější než *„myofunkce“*[3]. Ve třetí části se kontroluje kontext věty a ten se mění v okamžiku, kdy prochází rozpoznávačem nového slova. Může to znamenat, že některá slova se i několikrát změní. Například by se neměly objevit dvě slovesa ve větě, to má na starost jazykový mo-

[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 řetezec.
[3] Myofunkce – funkce konkrétního svalu, zejména při léčbě/terapii ortodontických problémů.

del. Problém nastává, když lidé opakovaně nezvládnou říct stejná slova vícekrát po sobě. Každé slovo nebo hlásku při sdělení pokynu vysloví odlišně. To znamená, že jejich hlasový projev pokynu má jinou intonaci a výslovnost, tudíž fonémy ve slovech se liší. Samozřejmě mohou taky tvořit věty, které nedávají smysl ve spisovné češtině. Příkladem je *brněnské hantec*[4] anebo jiné světové nářečí. Slovník rozhodně neobsahuje tolik fonetických zápisů jednoho slova, byl by obrovská podpora, kdyby tomu bylo tak. Vývojáři, z tohoto důvodu tak nemohou zahrnout všechny nářečí a různé fráze daného jazyka.

Jak zmiňuji, častěji se používají neuronové sítě, hlavně z důvodu rychlosti a flexibility. Další výhodou je porozumění podbarvení lidského hlasového projevu, nezvykle citově zabarvená výslovnost nebo atypická až patologická výslovnost. Tím, že tyto sítě pracují jako lidský mozek, se dokážou adaptovat na tyto možné, ale zásadní věci faktory, které vždy ovlivní rozpoznávání lidské řeči. Nevýhodou je trénink, jež může být velice zdlouhavý. Na začátku totiž síť nedokáže určit správné slovo z řečového projevu lidí. Neumí se ihned přizpůsobit nepřesnostem a tím není přesné zpracování zvukových dat. Síť se potřebuje vše naučit rozeznat a adaptovat se. Z těchto důvodů je potřeba neuronové síti umožnit časový prostor a nabídnout různá zvuková data ke zhodnocení. Mnoho reálných zvukových dat z různých prostředí, dále záznamy spousty lidí s rozmanitými druhy podbarvení hlasu či výslovnosti na základě emocí. Díky tomu dokáže zpracovat řeč lépe než HMM. Mnohem rychleji a přesněji umí vyhledat správné slovo.

Pro výběr fonému se používá Gaussova křivka[5]. Podle tohoto řešení dokáže akustický model rozlišit, o které skupiny nebo jednotlivé fonémy se jedná. HMM je převodník fonémů, což je jiný název pro stavový model. Určení správného stavu lze docílit, použitím odměn či cen, při změně stavu. Například nástroj Kaldi funguje na principu ceny stavů, což znamená, že pokud postupuje korektní cestou, cena se nízká a naopak. Na konci převodníku pak dokáže model posoudit, zda postupoval správně neboli vybere cestu s nejnižší cenou. Změna nastává pokud poslouchá *izolované slova*, jež nejsou přímo ve větě a jazykový model nelze využít na plný výkon, jelikož slovo je osamocené a tudíž nemá větný kontext. Často se jedná o první hypotézu modelu a v některých vzácných případech není totožná se slovem uživatele. Dokonce u některých knihoven i ticho bylo rozpoznáno jako očekávaná slova.

2.2 Lidská tvorba řeči

Řeč je artikulovaný zvukový projev člověka sloužící k přenosu myšlenek, 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, zvuk je modelace proudu vzduchu kmitajícími hlasivkami. Tento proces procházející hlasivkami se nazývá fonací a hlasivky jsou hlavním fonačním orgánem. Z hlasivek postupuje výdechový proud 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) sloužící 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

[4] Pojem *„hantec“* je zkomoleninou slova. Hantýrka a je vnímána jako zajímavá, vtipná, ale těžko pochopitelná brněnská mluva.

[5] Gaussova křivka je zobrazením tzv. Gaussovy funkce neboli tzv. normálního rozdělení pravděpodobnosti.

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 taky 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. Ale 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, neví je předvídat. Není úplně jednoduché zakomponovat spisovnou řeč společně s nářečím do jednoho modelu. Vzájemně se totiž budou ovlivňovat a tím pádem zhoršovat přepis řeči.

2.3 Vývojářské sady pro hlasové rozpoznání

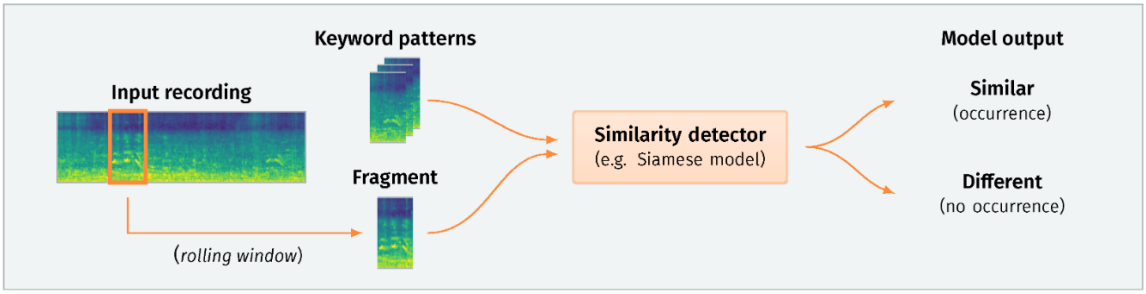
Při hledání narážím na spoustu knihoven, avšak po konzultaci s vedoucím jsme dospěli k tomu, že vyloučím knihovny, jež odesílají data na server pomocí internetu. Je to z důvodu, aby byla aplikace atraktivnější pro uživatele, kteří buď sportují v prostředí, kde není signál, nebo si při sportu vypínají mobilní data a wifi, aby nebyli rušeni. Některé knihovny totiž zpracovávají řeč mimo zařízení, kvůli důkladnějšímu zpracování. Díky tomu dokáží využít rychlejší čip než ten, který nabízí dnešní telefony. Mají taky přístup k větším slovníkům. Dalším bodem je tedy samotná kompatibilita a udržitelnost knihovny. Chci 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 anebo brzy končí.

Vhodná knihovna by měla splňovat tyto kritéria.

- Nepotřebuje přístup k internetu.
- Rychle rozpozná a přesně určí klíčové slovo.
- Je kompatibilní s nejnovějšími telefony s Android 14.
- Spadá pod *open source* projekty.

Dalším tématem konzultace bylo použití tzv. *keyword spottingu* či přepisu řeči. *Keyword spotting* se na rozdíl od přepisu řeči soustředí jen na klíčové slova nikoliv na celé souvětí, kvůli kterému je potřeba jazykový model. Jeho hlavní devízou je rozpoznávání izolovaných slov [12]. Sice poslouchá a projde celý zvukový záznam, ale hned když ví, že se nejedná o hledané klíčové slovo, tak ho zahodí a nepokračuje v jeho rozpoznání. Neprochází větný kontext a soustředí se pouze na slova. Díky tomu dokáže být přesnější a někdy i dokonce rychlejší vzhledem k hledanému slovu než přepis řeči do textu.

Jedním z benefitů je taky objemnost knihovny. Jsou o dost menší hlavně kvůli menšímu slovníku, často v něm mají jen natrénované slova, což je maximálně v řádu desítek slov. Dalším je fakt, že aplikace může běžet na pozadí a díky své jednoduchosti to nejde vůbec poznat



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].

na rychlosti samotného zařízení. Proto se využívá jako *wake-up word*[6], v češtině spouštějící slovo, jež slouží na spuštění robustnějšího přepisu do textu anebo aplikace, příkladem je funkce Siri či Ok Google. Soukromí uživatele je dost zásadní a pokud aplikace používá celkový přepis řeči, je vždy otázkou, k čemu dalšímu může tyto texty zneužít. Jelikož tento typ rozpoznání je trénován pouze na určité klíčové slova, tak nerozumí konverzacím [13]. Tudíž je bezpečnější vzhledem k uživateli, nedá se ale říct nejbezpečnější.

Přepis řeči se specializuje na přepsaní celé věty a tak sice rychle zachytí, co uživatel řekl, avšak ještě po sobě kontroluje syntaxi a smysluplnost vět. V dnešní době je to v řádu milisekund, však vzhledem k našemu řešení je hlavní rychlost rozpoznání. Je důležité, aby reakce na klíčové slovo přišla hned po vyřknutí a ne až potom, co zařízení vyhodnotí, že skončila věta. Slovníky jsou často objemnější, tudíž je zdlouhavější je prohledávat.

2.4 Výběr a testování

Úkolem je vybrat vhodné knihovny pro Android. V předešlé kapitole 2.3 jsem chtěl zjistit, které řešení je lepší vzhledem k mé aplikaci. Tudíž minimálním cílem je využít alespoň dvě. Jedna využívající Speech to Text (dále jen STT[7]) a druhá, co využívá *keyword spotting*. Papírově je mnohem lepší druhé řešení, avšak přepis řeči lze použít podobným způsobem, jen nebude efektivní. Abych věděl k čemu se chci přiblížit, první otestovanou knihovnou je základní rozpoznání hlasu od společnosti Google, tam byl ovšem problém, že odesílá data na server. Umí sice vše skvěle a v jakémkoliv podporovaném jazyku, avšak knihovnu musím vyřadit z důvodu využití internetu. Byla ovšem dobrou ukázkou čeho chci dosáhnout. Nacházím spoustu industriálních řešení na míru zákazníkovi, s vytrénovanými klíčovými slovy podle jeho výběru. Je to jedna z těch dražších cest a moje řešení je málo komplexní na to aby se to vyplatilo. Některé společnosti nabízely využití jejich řešení zdarma do určité míry používání avšak nesmí být použity komerčně. Jedna z nich je od společnosti Picovoice. Jejich nabídka různých knihoven je obrovská, ačkoliv jejich podmínky byly dost neúprosné. Tyto vývojářské sady, zkráceně SDK[8], se specializují jen na daný okruh funkcí, jež zvládnou

[6] Někdy se uvádí *hotword* anebo také *trigger word*.
[7] STT – Speech to Text, přepis řeči do textu
[8] SDK – software developer kit, v češtině sada pro vývojáře softwaru.

s přehledem. Narážím při hledání na novější verzi rozpoznávače SPHINX. Jedná se už o pátou verzi softwaru, jenž se začal vyvíjet kolem roku 1980. Posledním 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.

Po výběru proběhlo testování všech vývojářských sad. Obnášelo to test úspěšnosti rozpoznání klíčového slova. Testeři vždy řekly slovo v tichém prostředí (prázdná místnost, kde mluvili pouze oni) a poté test zopakovali v ruchu světa anebo při záznamu z přednášky. Každý z nich řekl slovo 10 krát a celkem bylo 10 testerů. Neboli v každém prostředí bylo slovo nebo sousloví řečeno 100 krát. Důvodem je, abych měl zaznamenané různé výslovnosti a emoce řečníka, které mohou ovlivnit rozpoznání. Testovalo se celkem 10 slovních spojení. V průběhu se mi potvrdilo, že znělejší hlásky pomáhají rozpoznání a například u slova *now* je zvučná hláska n, ale zbytek není. Na rozdíl od těch neznělých, což je třeba u slova *shot* hláska S[10].

Picovoice Porcupine

Vývojáři této společnosti nabízí mnoho různých sad pro rozpoznání hlasu. Jejich jména jsou odvozeny 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 fotky, úplně mi postačí *Porcupine Wake Word SDK*[11]. Bohužel tyto sady jsou předplacené a každá z nich má jiné podmínky. Například u sady, co 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. Úžasný je i fakt, že jejich webová aplikace, nabízí vytrénování vlastních klíčových slov, které pak integrujete do programu. Vytrénovat lze pouze slova delší jak 5 fonémů a sousloví nesmí být delší jak 3 slova. Soubory nejsou velké a to hlavně díky formátu `.ppn`[12], 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 rozpoznal rychle, kdykoli a kdekoli. Vzhledem k nekomerčnímu využití vynikající kandidát, avšak pokud aplikace bude vložena do Obchodu Play, nebude fungovat. Uživatel ji stáhne avšak, rozpoznání nebude fungovat, jelikož účet přihlášený k jejich webové aplikaci bude zablokován. Testeři od Googlu totiž před vydáním aplikace testují její funkčnost. Dělají to na více zařízeních a tím vypotřebují povolený počet inicializací.

Podle tabulky 2.1 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 je například počet uživatelů s aplikací. Díky tomu ji nelze vydat. Avšak pracovat s touto sadou bylo velice jednoduché, stačilo si nahrát souvislosti a pak volat funkce této knihovny. Využít lze i jejich vytrénované slova anebo jak zmiňuji, vytrénovat si vlastní, jež vložíte do aplikace a při inicializaci se spustí model s tímto souborem.

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

[10] Velké S značí souhlásku ve slově *shin* (viz příloha A)

[11] Porcupine je vysoce přesný a lehký nástroj na odposlech klíčových slov. Umožňuje vytvářet vždy poslouchající aplikace s podporou hlasu.

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

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 už jen stáhnout vytrénované klíčové slova a vložit je do modelu.

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 2.1: Tabulka úspěšnosti rozpoznání anglického slova a sousloví při použití knihovny Picovoice. Test probíhal ve dvou prostředích, přičemž jedno bylo za ticha, kdy bylo slyšet pouze uživatele (v různých vzdálenostech od zařízení). Druhé bylo při záznamu přednášky v angličtině anebo v ruchu města. *Porcupine* si vedl takřka stoprocentně při tichu a funkčnost při ruchu byla podobná rozpoznávači hlasu od Googlu. U slov, které mají méně než 5 fonémů, nejde vytrénovat neuronová síť. Proto nebyly testovány.

CMU PocketSphinx

Sada vychází ze zkušeností dvacetiletého vývoje, je nejstarší z této trojice. Neznamená to, že je zastaralá. Využívá velký slovník, jenž lze upravovat. Díky úpravám je možnost dopsat další fonetický zápis slova. Při testování se vyskytovaly chyby, kdy tester řekl: „Start“. Aplikace v ten moment měla vypsat něco na obrazovku, ale nestalo se. Tudíž jsem to zkusil sám se stejným slovem ale jinou výslovností: [stárt]. Na obrazovce se objevil text „Keyword spotted“, neboli kličové slovo bylo rozpoznáno. Po projití anglické fonetické abecedy (viz příloha A) mi došlo, že pokud toto specifické slovo řeknete rychle a krátce, muže se stát, že nebude dobře rozpoznáno. Tento Problém lze vyřešit jazykovým modelem, pokud uživatel mluví ve větách, jelikož se model opraví. Avšak když používáme jen klíčové slova potenciál, jenž nám nabízí kontrola textu, nelze využít. Jelikož jednoslovné věty nemají žádný smysl. Jedná se už o pátou verzi a není špatná, podporuje dokonce rozpoznání klíčových slov a dokáže je určit, ale není tak dokonalá jako sada od firmy Picovoice. Přece jen se jedná o univerzitní nekomerční projekt, jenž se udržuje za pomoci studentů, skupiny lidí ze sociální sítě LinkedIn a některých společností jako je Alpha Cephei.

U tabulky 2.2 je úspěšnost knihovny velmi špatná. Používá sice velký slovník a hodně záleží na slovu, které se má rozpoznat. Při rušném prostředí měla problém vůbec zachytit slovo. I přesto, že má funkci pro rozpoznání klíčových slov, tak je podobně rychlá jako přepis řeči. Jedna ze zásadních věcí, co musím zmínit je avšak flexibilita. Jakožto vývojář dokážu změnit slovník lusknutím prstu. Mohu přidat další fonetický zápis slova nebo přidat slovo, které nebylo ve slovníku. Samozřejmě to nedoporučuji bez hlubšího seznámení s modelem.

Aplikace využívající tuto sadu je potřeba *Sphinx4 API*, které je programované v jazyce Java. Má spoustu funkcí jako je rozpoznání uživatele, přepis textu anebo rozkouskování řeči [1]. Je dobrým kandidátem, jehož využiji jelikož má dobrou podporu a stále je relevantní i vzhledem k nejnovějším Android zařízení. Bohužel používá ještě *pre-alpha* sadu pro vývojáře. Není dotáhnutá do konce a občas nenabídne nejpřesnější přepis. Pro integraci modelu a knihovny do projektu lze využít tutoriál z jejich stránek[13].

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 2.2: Tabulka úspěšnosti rozpoznání anglického slova a sousloví při použití knihovny CMU Sphinx. Test probíhal ve dvou prostředích, přičemž jedno bylo za ticha, kdy bylo slyšet pouze uživatele (v různých vzdálenostech od zařízení). Druhé bylo při záznamu přednášky v angličtině či v ruchu velkoměsta. Vývojářská sada má problém rozlišit správně klíčové slova. Zachytává dokonce i ticho a výslovnost hraje velkou roli. U některých testerů se nepodařilo ani rozpoznat jejich řeč.

Alpha Cephei Vosk

Tato vývojářská sada dokáže přepisovat řeč a rozpoznávat uživatele. Není nejvhodnější na rozpoznání klíčových slov, avšak správné nastavení umožňuje tuto funkci. Zkouším ji hlavně, jelikož nabízí český akustický model. Nabízí jich ale mnohem více na jejich stránkách[14]. Využívá software Kaldi, tudíž funguje na podobném principu jako předešlá knihovna. Alpha Cephei se totiž podílela na vývoji obou. Avšak se od sebe v mnohém liší. Knihovna Vosk je integrovaná jako celek a tudíž nelze ji jen tak upravit její části. Knihovna nabízí přesný přepis řeči do textu. Po pár úpravách lze docílit toho, aby se neřídila větami, ale pouze slovy. Je to dobrá konkurence a proto otestuji tyto dvě knihovny proti sobě.

Po předělání 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 při konci řeči. Mé řešení hledá klíčový řetězec v hypotéze dílčího výsledku. Pokud hypotéza obsahovala hledané slovo, bude zavolána daná funkce. V ten moment byl model ukončen a znovu spuštěn. Vymaže se tím hypotéza, protože do dílčí hypotézy přicházely další slova, které byly vybrány podle kontextu, co model slyšel.

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

[13] https://cmusphinx.github.io/wiki/tutorialandroid/
[14] https://alphacephei.com/vosk/models

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 2.3: Tabulka úspěšnosti rozpoznání anglického slova a sousloví při použití sady Vosk. Test probíhal ve dvou prostředích, přičemž jedno bylo za ticha, kdy bylo slyšet pouze uživatele (v různých vzdálenostech od zařízení). Druhé bylo při záznamu přednášky v angličtině nebo v ruchu města. Kupodivu mě překvapilo, že více slov nepomůže rozpoznání. Avšak byla úspěšnější než Sphinx a to radikálně. I přesto, že vychází ze stejného softwaru a že se jedná o pouhý přepis řeči.

Funguje na principu přepisu řeči, a pokud se nesnažím o rychlé rozpoznání, tak dokáže být mnohem přesnější. Obě knihovny jsou do projektu vloženy jako Java modely, a pak obě potřebují ještě své závislosti, jež se stahují z repozitářů.

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

Tabulka 2.4: Tabulka úspěšnosti českého slova a sousloví při použití sady Vosk. Test probíhal ve dvou prostředích, přičemž jedno bylo za ticha, kdy bylo slyšet pouze uživatele (v různých vzdálenostech od zařízení). Druhé bylo při záznamu přednášky v angličtině anebo v ruchu města. V tichém prostředí si model vedl velmi skvěle. Ačkoliv v ruchu si vedl hůře než anglický model. Při správných znělých hláskách ve slově správně funguje i v rušném prostředí (viz slovo „akce“).

Jak jsem zmínil tato knihovna nabízí integraci s každým jejich modelem, co mají na stránkách. Proto jsem využil i český model, který by měl mít bohužel horší úspěšnost než anglický podle jejich rovnice (chybovost slov/rychlost[15]). Anglický model má 9,85 WER/s

[15]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.

na datasetu *librispeech test-clean*[16] a 10,38 WER/s na *tedlium*[17]. Český model byl testován tzv. *křížovou validací*[18] a získal jen 21,29 WER/s. Ovšem poté, co byl otestován nebyl horší, nýbrž na stejné úrovni jako ten anglický (viz tabulka úspěšnosti 2.4). Nevýhodou avšak je fakt, že slovník sice obsahuje pražské i moravské nářečí, ale neobsahuje třeba slovo „pořiď". Dle výsledků mého testování má vliv i výslovnost anglických slov. Čeština je náš mateřský jazyk a vzhledem k nám přirozenější. Ve finální aplikaci, již budu publikovat, bude obsahovat oba modely, jak anglický tak český. Někteří z uživatelů nemusí umět anglicky a tak mohou využít svého mateřského jazyka.

Důležité u této knihovny je vybrat správné slova. Má výborné výsledky v tichém prostředí, jakmile se člověk vyskytne v ruchu, má problém vytěsnit okolní zvuky nejspíš. Proto je prospěšné použít ve slově znělé hlásky. Jsou lépe slyšet ve změti zvuků. Dalším doporučením je délka slova, čím delší slovo je tím snadnější je jeho registrace modelem. Optimální délka slova je 5-10 písmen. Někdy víc je méně, pokud uživatel použije více slov nemusí model pracovat správně.

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

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

[18] 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.

Kapitola 3

Vývoj aplikace na OS Android

Vybral jsem si operační systém Android hlavně kvůli dostupnosti a jednoduchosti. Dokonalým důkazem je webová stránka pro vývojáře, kde se dá dočíst spoustu rad a návrhů, taky však spoustu změn v nových verzích [2]. Budu programovat v jazyce Java, protože je mi blízký. Není jediný, který se využívá na tvorbu Android aplikací, existuje i Kotlin, jenž je novější a začíná se využívat více a více. Přišel v roku 2017 a dokáže pracovat s komponentami napsanými v Javě, což je velké plus, že vývojáři nemusí 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 programování v Kotlinu nebo úplně na jinou platformu.

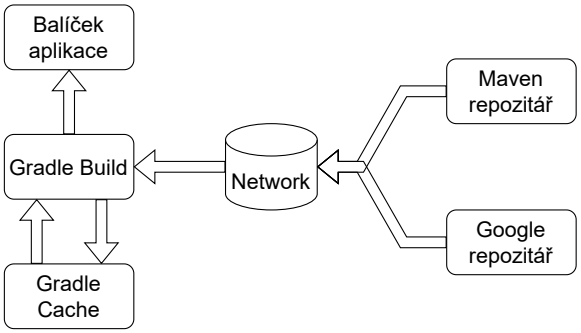
Za prvé je důležité stáhnout nejnovější verzi Android Studia[1]. V mém případě to je verze *Hedgehog*. Dřívější verze byly označovány podle čísel, ale později začal Google pojmenovávat verze po zvířatech a jdou postupně od A až po Z. Stejné řazení jako u verzí Androidu, akorát je podle sladkostí. Po stažení vývojového prostředí Android Studio, které je upravenou formou prostředí IntelliJ IDEA, je dobré založit projekt, kde si vyberu prázdnou aktivitu. Ukáže se modulární okno, kde zadám v jakém jazyce chci programovat a následně zvolím jaký nástroj použiji na kompilaci a linkování všech potřebných souborů. Já osobně jsem si vybral Gradle, protože se o něm psalo v radách na Android developeru [2], ačkoliv existují i jiné možnosti. Je založen na Apache Groovy, samotný Gradle nic nedokáže, využívá k tomu pluginy. Dále budete potřebovat nejnovější Android SDK, kvůli podpoře Android API. Po otevření projektu můžete začít programovat.

3.1 Základní soubory a nastavení

Doporučuji začít nastavením si virtuálního zařízení, jež se sice nechová jako reálný telefon, ale kvůli rychlé kontrole jak obrazovka vypadá nebo otestování podpory verze Android je vynikající volbou. Lepší 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 využívá a tak dále. Podrobnosti, jaké určité části potřebují, zmíním v dalších kapitolách, které budou o těch daných částech. Manifest určuje hlavní náhled aplikace, dále definuje jak se jmenuje a jaké téma používá a další důležité informace. O kompilaci a vytvoření balíčku se stará nástroj Gradle. Jeho soubory s koncovkou `.gradle` má celý projekt, a pak i části, ve kterých se nachází něco, co se musí

[1] https://developer.android.com/studio

kompilovat. Nesou definice kódu, jako například jeho verzi, s čím a jak se bude projekt kompilovat a v neposlední řadě slouží ke složení projektu. V tom nejhlavnějším souboru se definuje hlavní sestavovací pluginy, souvislosti a repozitáře. Ty nejpoužívanější a taky hlavní jsou Google a mavenCentral. Odtud se stahují knihovny, pokud je tam autoři nahráli. V jiném případu si je musím nahrát sám, pak může přibýt další soubor Gradle, který kompiluje tento modul.



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

Při každém spuštění či kompilaci se tyto 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 kódu, o té až později, a minimální i cílené verze Android SDK. Tyto verze udávají podporu API pro systém Android a v dnešní době musí aplikace podporovat minimálně API 21, což je Android 5 neboli Lollipop. Cílená verze znamená na jaké verzi se program kompiluje, to 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 nahrávání aplikace na jejich obchod a pokud nesplňuje tyto podmínky není aplikace vydaná. Ovšem technologie se vyvíjí a Google vydává novou verzi každý rok a od letních měsíců musíte cílit na API 35, neboli Android 15, se jménem Vanilla Ice Cream, aby jste 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 třeba o offline knihovny anebo různé balíčky. Třetí skupina se nazývá *Gradle Scripts* a to 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. Ohledně ikon, které jsou nezbytnou součástí každé aplikace, jinak bychom je od sebe nedokázali rozeznat. Základní ikony se nachází v balíčku `mipmap`, kde jsou rozděleny na pozadí ikony, popředí ikony, klasický obrázek ve tvaru čtverce a kulatou ikonu. Je tomu tak, protože uživatelské prostředí systému Android nabízí uživatelům vybrat si vzhled ikon, tudíž programátoři, spíše tedy grafici, musí vytvořit takovou ikonu 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. Ikdyž se jedná o čtverec Android rohy vždy zaoblí. Na tomto příkladu lze vidět i stíny samotné ikony a tak tvoří iluzi 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 se je podklad. Od verze Android 8 ikony mohou obsahovat stín, pro zlepšení uživatelského zážitku.

3.2 Rozložení obrazovky

Existují různé typy rozložení a jsou děleny podle umísťování a lokalizaci objektů na obrazovce. Vytvořené náhledy s objekty pak ovládá třída, která je uvedená v definici souboru a napojená skrz svůj kód. Nezáleží jestli 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 rozložení je `LinearLayout`. Uspořádává prvky horizontálně za sebe do jednoho řádku či vertikálně pod sebe. Klíčový parametr jsou`orientation`, udává směr uspořádání prvků uvnitř.

Další je `RelativeLayout` a 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 po dobu pokud mají prvky 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 kam přesně se mají zobrazit. Funguje jako tabulka, takže lze roztáhnout buňky přes více řádků i sloupců.

`ConstraintLayout` přidává prvkům omezení a funguje jako gumička. Přitahuje je například k okraji. Neboli pokud připnete daný prvek k 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 skrz jeho nulovou flexibilitu. Zařízení mají odlišné velikosti obrazovek a to nemluvě o změnách orientace.

3.3 Vydání v obchodě

Aby mohli programátoři vydávat aplikace, potřebují vývojářský účet u společnosti Google. Potřebují funkční e-mail a zaplatit 25 dolarů, jako potvrzení, že nejsou roboti. Dělá se to kvůli spamu a zároveň aby si uživatelé nepublikovali, co chtěli a kdy chtěli. Před vydáním aplikace, avšak musí vyplnit spoustu dotazníků, kvůli věkovému omezení, informacích o tom k čemu bude sloužit anebo zda je nějakým způsobem výdělečná. Je toho hodně a nechci všechno vypisovat, na podrobnější popis 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 pár obrázků z aplikace. Na obrázcích nezáleží jen musí splňovat předepsanou velikost. Dále název aplikace, který se bude ukazovat a nějaký krátký a dlouhý popis.

Po vyplnění těchto informací už stačí vytvořit vydání aplikace. Android studio nabízí funkci generování aplikačního balíčku ve formátu `.aab`[2]. 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 z už vložených. Ze souboru vloží verzi aplikace a programátor pouze napíše zásadní změny do popisu. Pak už jen uložit a nové vydání je hotové. 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 praktický stejný jako u prvního vydaní. Jediná změna je inkrementace čísla kódu. Tak jako předtím můžete označit verzi nebo ponechat načtené informace z balíčku. Nastavují se v souboru `build.gradle`, který se nachází v modulu `app`. Může vypadat například na tomto snímku číslo 3.3. Dále je doporučeno informovat uživatele o tom, co je nového. Dělá se to v popisu vydání a může to být ve více jazycích. Záleží jakým způsobem je nastaveno vydání v 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 další zprávy.

Existují tři možnosti jak vydat aplikaci do obchodu. První z nich je *Interní testování*, používá se pro testování v rámci společnosti anebo 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, akorát uživatelé se do testování nepřihlašují zasláním e-mailu, nýbrž přihlášením v Obchodu Play. Narazit na ni může kdokoli, to u první možnosti nešlo. Poslední možnost je už finální publikace, jež nabízí různé informace z obchodu a viditelná je všem v dané povolené zemi.

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

```
android {
    namespace "vut.example.voskapp"
    compileSdk 34
    defaultConfig {
        applicationId "vut.example.voskapp"
        minSdkVersion 21
        targetSdkVersion 34
        versionCode 20
        versionName "3.2"
        ndkVersion = "26.1.10909125"
        ndk {
            abiFilters 'armeabi-v7a', 'arm64-v8a', 'x86_64', 'x86'
        }
    }
    buildTypes {
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
            signingConfig signingConfigs.debug
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

Obrázek 3.3: Na obrázku lze vidět ID aplikace, stejně tak kompilovací verze SDK a také minimální a cílená. Dále je tu číslo kódu a jeho název. Verze NDK (Native Development Kit), což je soubor nástrojů a souborů hlaviček potřebných k vytváření nativního kódu pro Android pomocí jazyků jako je C nebo C++. Podporované architektury ABI (Application Binary Interface) a představuje rozhraní mezi aplikacemi a operačním systémem na úrovni binárního kódu. V možnostech kompilace pak je uvedena verze Javy. V tomto případě 1.8.

Kapitola 4

Návrh aplikace

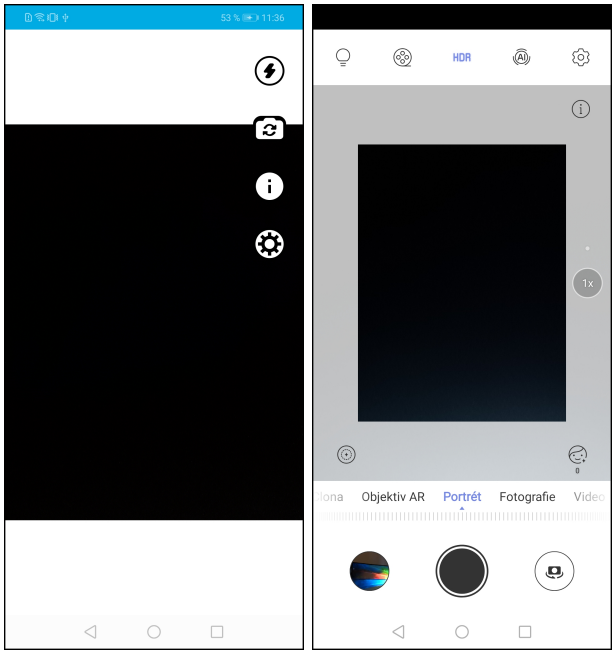
Aplikaci si nastavíte a namíříte na sebe a začnete používat a až po cvičení se zpátky k telefonu vrátíte. Tohle je cíl, jak by měl výsledek vypadat, uživatel dá telefon do stativu, kde zapne aplikaci a nastaví, aby na něj mířila. Při cvičení pak heslovitě bude říkat pokyny na spuštění různých funkcí jako například zapnutí svítilny anebo vyfocení fotky. V pozdějším vývoji aplikace je nezbytné, aby si mohl člověk nastavit aplikaci z místa, kde cvičí. To se hodí při nastavení odpočtu a délky videa. Rozložení obrazovek jsem se rozhodl rozdělit podle účelu, k jakému budou sloužit. Hlavní obrazovka, která bude zobrazovat náhled kamery a bude obsahovat tlačítka po pravé straně. Některé z nich budou jednoduché přepínače, ostatní budou fungovat jako přepnutí do jiné části aplikace. Nastavení bude obsahovat pouze textové vstupy na zadaní slov, kterými chce uživatel ovládat focení a natáčení videa. Dále tu bude změna doby odpočtu a délky videa a nebude chybět tlačítko na uložení a na návrat na hlavní obrazovku. Poslední náhled v aplikaci bude jenom zobrazovat dané uživatelské nastavení a nápovědy k tlačítkům a funkcím.

Uživatelské prostředí se chci snažit udělat jednoduché, jelikož aplikace je 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 bylo spoustu rad, jak jej udělat přehledné a nenáročné. Sice se jedná hlavně o testovací verzi, avšak chci aplikaci vydat veřejně právě kvůli testování, proto se snažím vytvořit aspoň příjemný uživatelský zážitek, aby to uživatele neodrazovalo od používání.

Náhledy těchto částí jsou souboru typu XML, kdy každá obrazovka má svůj. V těchto souborech se definují UI objekty s jejich vlastnostmi, jež se následně zobrazují. Použiji rozložení obrazovky `RelativeLayout`, popsaný v kapitole 3.2. Určitě se dají využít i jiné kombinace vnořených. Na barevné téma byla převzata modrá barva fakulty, jež je jako primární barva. Nachází se v horní liště zařízení či v liště v aplikací, již nepoužívám.

4.1 Hlavní obrazovka

Jak jsem uvedl, chci uživateli zpříjemnit používání, toho chci docílit přiblížením se vizuálně k vestavěné aplikaci od výrobce. Proto jsou funkční tlačítka napravo a na obrazovce se nenachází žádný text. Spousta testerů mi potvrdilo, že toto rozložení je super a opravdu je podobné jako, to co už znají a používají. Jednoduchost podtrhuje také to, že se zde nachází celý náhled z kamery, přičemž jsou zbylé nezaplněné okraje černé. Je normální, že mobilní telefony nemají přední svítilnu a v jiných aplikacích s funkcí focení předním fotoaparátem je to řešeno podsvícením náhledu, neboli zbarvené okraje do bílé barvy.



Obrázek 4.1: Nalevo obrazovka mé aplikace, napravo vestavěná aplikace výrobce Huawei. Obě mají zapnutou přední svítilnu. Náhled kamery z velké většiny případů nezabere celou plochu obrazovky a tak vývojáři zbytek obrazovky využívají na tlačítka jinak na zesvětlení obrazovky, aby vyzařovala více světla a působila tak jako svítilna.

Na skupině obrázků 4.1 lze vidět na mé aplikaci (obrázek vlevo), že se i změní barva tlačítek. Děje se to, protože by jinak nebyly vidět. Určitě to jde vyřešit i jejich zbarvení podle podkladu, ale to by se měnilo vždy, jak uživatel pohybuje s obrazem a není to zrovna příjemné na oči. Navíc se změnou nezabývají v některých případech ani výrobci telefonů a ponechají je špatně viditelné. Zvolil jsem uživatelům už známe ikony. Není důležité vybrat přesně ty stejné. Avšak během používaní by měli být pochopitelné na první pohled. Většinu z nich nabízí Android Studio ve svém seznamu vektorových aktiv. Avšak využil jsem i některé externí grafiky, například u ikony nastavení. Jejich podklad je černý právě proto, byly vidět i na bílém pozadí a zároveň to zlepšilo viditelnost na vícebarevném pozadí. Tlačítka lze umístit prakticky kdekoli, jednak i z důvodu, že v budoucnu by se i tyto tlačítka daly ovládat hlasem. Pořízení fotky přes spoušť zde není implementována, jelikož není potřeba.

Náhled fotoaparátu je umístěn veprostřed obrazovky pomocí objektu `PreviewView`. Jeho pozadí je nastaveno na černou barvu, mění se na bílou v určitých situacích. Na každém zařízení se zobrazuje jinak, jelikož používají jiné rozlišení hardwarové kamery. Android proto nabízí nastavení náhledu, se vlezl na obrazovku. Já používám `FIT_CENTER`, ale existují i jiné. Podrobnější popis je na stránkách Android developer[1]. Díky tomuto nastavení se náhledy přední i zadní kamery umístí přímo doprostřed. Na obrazovce se zobrazuje vyskakovací oznámení při záznamu fotky. Případně některé chybové výpisy. Vyskakuje ve spodní části obrazovky.

Hlavní obrazovka je definována v souboru `activity_main.xml`. Prvním prvkem a taky nejdůležitějším je náhled kamery. Má nastavené černé pozadí a šířku a výšku má stejnou jako nadřazený prvek. V horní části se nachází ještě červený kruh, jež upozorňuje uživa-

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

tele na spuštěné natáčení videa. Pak už se zde vyskytují jen zmíněné tlačítka. Nastavení mají všechny stejné, jak výšku tak šířku na 40sp[2]. Všechny tyto prvky jsou propojeny v třídě `MainActivity`, ve funkci `OnCreate`. Tento jediný náhled má příznak `FLAG_KEEP_SCREEN_ON`, jenž značí nezamknutí obrazovky zařízení. Důvodem je *always on* režim aplikace. Uživatel cvičí a telefon je stále odemknutý a poslouchá příkazy. Pokud by se vypnul nebyl by schopen zachytit fotografii ani video.

4.2 Informační obrazovka

Tato část aplikace má pouze informační hodnotu, uživatel zde nemůže nic upravovat. Při konzultaci s pár lidmi mi řekli, že by uvítali, aby se daly zobrazit ta slova, co si nastaví. Je to skvělý nápad jelikož druhý den už nevíme, co jsme dělali natož abychom věděli, co jsme si nastavili v jedné z deseti aplikací. Proto se zde nachází klíčové slova, odpočet, délka videa a důležité informace pro uživatele o tlačítkách. Tyto informace nejsou psané v celých větách, nýbrž co nejvíc heslovitě a zkráceně, protože lidé dnes nečtou celé články, spíše skenují různé hesla, výrazné nadpisy a obrázky, aspoň podle této knihy o uživatelském prostředí pro webové a mobilní aplikace [5]. Proto jsou důležité informace navíc zvýrazněny žlutě.

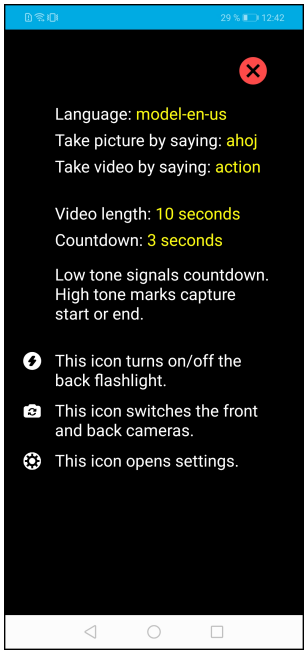
```
public SpannableString colorChange(String staticText, String dynamicText) {
    String combinedText = staticText + dynamicText;
    SpannableString spannableString = new SpannableString(combinedText);

    int startIndex = staticText.length();
    int endIndex = startIndex + dynamicText.length();

    spannableString.setSpan(new ForegroundColorSpan(Color.YELLOW), startIndex, endIndex, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
    return spannableString;
}
```

Obrázek 4.2: Část kódu, jež slučuje statický text před dvojtečkou a dynamický text nastavení za ní. Dynamický text je nastaven na žlutou výraznou barvu, aby byl více rozeznatelný. Využívám k tomu třídu `Spannable` a `SpannableString`.

Řádky jsou roztáhlé po celé obrazovce, tudíž pokud má zařízení širokou obrazovku jsou po celé šířce. Ikony na levé straně od textu jsou miniatury z hlavní obrazovky. Žlutý text je vložen při načtení tohoto náhledu. Je uložen v *SharedPreferences* (viz kapitola 4.5). Mění se podle uživatelského nastavení.

Definice této obrazovky se nachází v souboru `activity_help.xml`. Uvnitř rozložení náhledu je jedno tlačítko o velikosti 60sp, v horním rohu. Pod ním jsou textové prvky nesoucí informace o uživatelském nastavení. Miniatury ikon jsou udělané pomocí obrázků. Propojení s třídou `HelpActivity` mají pouze prvky, jež obsahují měnitelné texty. Vložení textu popisuji v kapitole 4.5. Na obrázku 4.2 lze vidět rozdělení a obarvení důležitých informací.

[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 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.



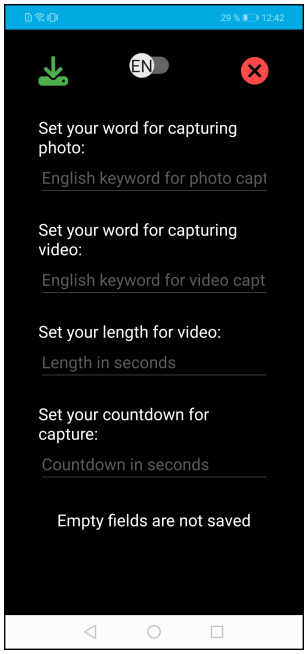
Obrázek 4.3: V pravém horním rohu je umístěno tlačítko, jež uživatele vrátí na hlavní obrazovku. Je červené aby nesplývalo s textem. Text je shluknut do skupinek, které jsou odděleny malými mezerami. Horní část nese informace o nastavení slov a času, přičemž spodní část je nápověda k tlačítkům a zvukové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

4.3 Obrazovka nastavení

Nastavení čtyř proměnných informací uživatelem je uděláno pomocí textového vstupu, z nichž dva, délka videa a odpočtu, jsou číselné vstupy v sekundách. Nachází se tam také dvě tlačítka, kde ovšem jedno z nich je zpětné tlačítko a to druhé je na uložení nových hodnot. O principu jak to funguje budu psát později v kapitole implementace.

Přepínač ukládá název modelu při každém přepnutí. Vize je, že změní i jazyk aplikace, ale v plánu bylo ho udělat například v další verzi aplikace. Proto i přepínač je jednoduchý. Klávesnice se uživateli zobrazí sama po kliknutí na vstup. Své nastavení uloží po kliknutí na tlačítko uložení. Poté se vloží jeho hodnoty do *SharedPreferences* (viz kapitola 4.5). Vyskakovací oznámení se zobrazuje v horní části, kvůli klávesnici. Oznamuje zda bylo nastavení uloženo.

Jako u předešlé obrazovky je zde taky použito tlačítko, jak u uložení, tak u zpětného tlačítka. Mají stejnou velikost jako na jiné obrazovce. Definuji ji zde taky nadpisy vstupů, což jsou jen textové prvky. Uživatelské vstupy i ostatní prvky jsou roztáhlé 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 propojené se třídou `SettingsActivity`.



Obrázek 4.4: V horní části obrazovky se nachází dvě tlačítka. Nalevo zelené značící uložení nastavení a napravo červené tlačítko na návrat na hlavní obrazovku. Uprostřed přepínač mezi českým a anglickým modelem. Pod nimi jsou čtyři textové vstupy s jejich nadpisy. Uvnitř nich je šedě popsána očekávaná hodnota. Vespod je informace o prázdných polích.

4.4 Povolení práv aplikace

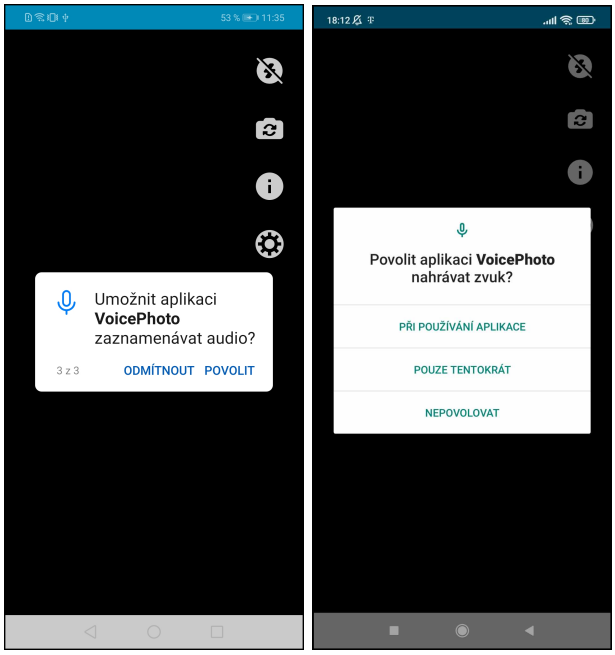
Je potřeba povolení, kvůli rozpoznání hlasu z mikrofonu zařízení a správné fungování záznamu kamery, stejně tak uložení souboru do úložiště telefonu. Je zásadní, aby uživatel tyto povolení udělil, jinak nemůže aplikace pracovat správně. To zapříčiní, že nemá přístup k mikrofonu natož zobrazit náhled z kamery nebo do úložiště. Tímto krokem se aplikace ukončí, protože nedokáže pokračovat. Tyto povolení se dají potvrdit při prvním spuštění aplikace nebo později v nastavení aplikace. Proto se v kódu nachází část, která zobrazuje uživateli modulární okno, ve kterém uživatel vybírá zda povoluje použití potřebných hardwarových a softwarových touto aplikací. Po každém spuštění se následně kontroluje nastavení těchto definovaných proměnných.

Avšak systém Android se vyvíjí, a tím pádem různé funkce se v jedné verzi mohou využít a v jiné verzi jsou buď předělané někdy dokonce zastaralé a neměli by se vůbec použít. Proto určité části kódu jsou psané dvakrát, kvůli změně ve verzi API 29, kdy uživatel nemusí povolovat přístup kvůli zápisu do interního úložiště. Druhým řešením by bylo v kódu zapsat, že minimální API je 29 a tím bych nemusel implementovat druhou verzi na nižší verze operačního systému. Pak by aplikace nebyla přístupná starším zařízení, jež lidé ještě stále používají. Vývojářům se to někdy vyplatí takto udělat, avšak někdy to znamená 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 nejvíc Android zařízení. Minimální podporovanou verzí pro nahraní aplikace do Obchod Play je Android API 21. Počet zařízení s touto verzí a vyšší

je odhadnutý 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 uživatel použije danou funkci. V mé aplikaci jsem k tomuto přistoupil trochu jinak. Žádosti se zobrazují na začátku a všechny postupně [4]. To lze vidět u obrázku 4.5 vlevo. V dolním levém rohu modulárního okna kolik z kolika. Vyvíjím aplikaci na celkem čtrnáct verzí Android. Proto musím rozdělit zobrazení žádostí pro verze, jež používají omezené úložiště a na ty, co potřebují povolení na ukládání do interního úložiště. Vytvořil jsem si dvě skupiny žádostí a podle toho s jakou verzi má zařízení, tak se odkáži danou skupinu verzí.



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

Uživateli se pak ukážou postupně všechny potřebné povolení, jež musí povolit. Jelikož je toto demo aplikace, není tu tak důmyslně promyšleno, co se stane při nepovolení. Když nastane tato situace, tak se jednoduše program ukončí, protože by nemohl pokračovat. Správně by se to mělo vyřešit tím, že se zobrazí modulární okno s upozorněním, že je důležité povolit tyto povolení, kvůli správné funkčnosti aplikace. Pokud by tak uživatel stejně neučinil, tak by se mělo zobrazit nějaká 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 tak 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. Zápis potřebuje instanci objektu `SharedPreferences.Editor`. Lze ji 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 (viz obrázek 4.6.

```
SharedPreferences ShPr = getApplicationContext().getSharedPreferences( name: "VoiceSet", Context.MODE_PRIVATE);

model = ShPr.getString( key: "model",  defValue: "model-en-us");
if(model.equals("model-en-us")) {
    KEYPHOTO = ShPr.getString( key: "kPhoto",  defValue: "picture");
    KEYVIDEO = ShPr.getString( key: "kVideo",  defValue: "action");
} else if (model.equals("model-cz")) {
    KEYPHOTO = ShPr.getString( key: "kPhoto",  defValue: "foto");
    KEYVIDEO = ShPr.getString( key: "kVideo",  defValue: "akce");
}
```

Obrázek 4.6: První řádek je propojení aplikace s úložištěm aplikace. V ukázce kódu se načítá konfigurace jazyka modelu. To rozděluje zda přednastavené slova jsou v češtině či angličtině. Uloží se do globální proměnné, která slouží pro kontrolu rozpoznání hlasu.

Kapitola 5

Implementace funkcí

Modely testovacích knihoven byly často integrovány jako modul v projektu. Bylo potřeba přidat závislosti pro práci s ním. Pro práci s vestavěnou kamerou v zařízení je zapotřebí závislostí. Na obrázku 5.1 je vidět, které z nich byly použity, vzhledem ke správné funkčnosti. Používám třídu `CameraX`, v pozdější kapitole 5.2 vysvětlím proč.

```
implementation "androidx.camera:camera-core:1.3.3"
implementation "androidx.camera:camera-camera2:1.3.3"
implementation "androidx.camera:camera-lifecycle:1.3.3"
implementation "androidx.camera:camera-video:1.3.3"
implementation "androidx.camera:camera-view:1.3.3"
implementation "androidx.camera:camera-mlkit-vision:1.4.0-alpha05"
implementation "androidx.camera:camera-extensions:1.3.3"
```

Obrázek 5.1: Lze vidět část závislostí k ovládání kamery v zařízení. Kvůli vydání na nejnovější telefony s operačním systémem Android aplikace používá nejmladší knihovny[1]. Závislosti rozděluji vždy do skupinek, aby bylo jasné, k čemu slouží.

V aplikaci implementuji dvě hlavní funkce. Jedná se o zachycení fotky 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.

5.1 Integrace knihoven

Po vytvoření univerzálního projektu jsem vybral z celého seznamu pouze 3 knihovny, jež splňovaly požadavky. Jak je uvedeno v kapitole 2.3, existují dva typy přístupu. Chci otestovat oba typy, *keyword spotting* používá vývojářská sada od Picovoice. Druhé dvě rozpoznávají pomocí *přepisu řeči*.

Než začnu programovat samotný kód aplikace, musím definovat v souboru Manifest použití mikrofonu a kamery. Ve starších verzích Androidu musím také upozornit zařízení, že budu ukládat soubory do interního úložiště. Tohoto docílím pomocí těchto řádků v kódu na obrázku 5.2. Ty první znamenají, že aplikace potřebuje fyzickou kameru k fungovaní a pokud

[1] https://developer.android.com/jetpack/androidx/releases/camera

ji zařízení nemá tak aplikace nebude fungovat. Proto ji nemohou uživatelé ani nainstalovat. Další řádky už se týkají definice povolení, které se ukáží uživateli při prvním spuštění (viz kapitola 4.4). Lze vidět že povolení na zápis do interního úložiště má atribut navíc. Je tam protože, toto povolení je potřeba pouze na verzi Android API 28 a nižší. Novější verze totiž používají tzv. omezené úložiště, kvůli soukromí. Aplikace mají přístup jen k tomu co potřebují.

```
<uses-feature
    android:name="android.hardware.camera"
    android:required="true" />

<uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    android:maxSdkVersion="28" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.CAMERA" />
```

Obrázek 5.2: V manifestu programátor musí vždy uvést, jaké aplikace bude používat a co potřebuje ke své funkcionalitě. Musel jsem tudíž uvést, že je důležité, aby zařízení mělo fotoaparát (první řádky), poté povolení, které uživatel musí povolit. Například použití mikrofonu či náhledu z kamery.

Pro integraci knihoven je potřeba závislostí a během spuštění aplikace inicializovat model. V závislostech jsou funkce, které pracují s modelem. Hlavním krokem je správně vložit model vedle projektu. Ve funkci je pak potřeba ho nastavit podle dokumentace. Následně během rozpoznávání je důležité vyfiltrovat jen ty klíčové. U knihovny Vosk to funguje pomocí metody `contains`. Jelikož se jedná o přepis řeči, tak hledám pouze slovo ve větě. Podobně pracuje i knihovna PocketSphinx. U vývojářské sady *Porcupine* byly slova označeny během inicializace číslem podle pořadí. Tudíž kontrola probíhala pomocí čísel nikoliv podle slov. Aplikace pak pokračuje spuštění funkcemi přiřazenými ke klíčovým slovům.

5.2 Implementace záznamu

K této nejvhodnější knihovně teď potřebuji dodělat zbytek aplikace kvůli důkladnějšímu testu celého projektu. První knihovnou, jež umožňuje ovládat kameru zařízení je `Camera1`, avšak je 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 tu první z nich. `Camera2` je užitečná knihovna, pokud programátor ví, co implementuje. Protože pracuje s funkcemi hluboko v kódu a tak obsahuje mnoho funkcí, zatímco univerzální knihovna obsahuje funkce na volání těchto důmyslnějších funkcí a tak je mnohem jednodušší a rychlejší na implementaci. Proto použiji `CameraX` na testování aplikace. Díky těmto knihovnám je aplikace schopna spouštět a vypínat svítilnu, pokud je k dispozici. Dále změna směru focení, neboli jestli zařízení zachytí fotku přední anebo zadní kamerou.

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

Zobrazení náhledu kamery

Aby se zobrazil pohled skrz kameru telefonu, je potřeba do uživatelského prostředí vložit objekt zvaný `androidx.camera.view.PreviewView`[2]. V mém případě nastavený na velikost nadřazeného elementu. Inicializuje se během spuštění hlavní obrazovky. To znamená, že když se uživatel přepne na jiný pohled, tak se náhled vypne a musí se proto znovu propojit. Pokud není povolen přístup ke kameře, tak se ani nezapne. Po zapnutí se definuje, jestli se zobrazí přední či zadní fotoaparát. 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`. Poté k ní připojím dané instance pro vytvoření fotky a videa. Tvoří se pomocí `ImageCapture.Builder` a `Recorder.Builder`.

Nastavení zachycení fotky

Vyfocení fotky obstarává funkce `takePicture`. Na jejím začátku je vytvořen název pro soubor. Je využit systémový čas zařízení v milisekundách. Fotka je focena ve formátu `jpeg`. Pak už nastavuji parametry k záznamu fotografie (název, formát, orientaci). Tím vytvořím instanci, jež nese informace o uložení. Soubor se ukládá mezi ostatní fotky zařízení. Je to základní složka, ke které lze přistoupit pomocí definované proměnné `DIRECTORY_PICTURES` v třídě `Enviroment`. Pokud vše výše zmíněné je nastaveno zařízení provede zaznamenání a pokusí se o uložení na dané místo funkcí `onImageSaved`. Po provedení tohoto kroku spustí zobrazení vyskakovacího textu na obrazovce, že se povedlo vyfotit fotografii jinak vypíše chybovou hlášku s kódem chyby. V průběhu celého procesu bylo rozpoznání pozastaveno. Pokračuje až po dokončení záznamu.

Nastavení natočení videa

Natočení videa zprostředkovává funkce `captureVideo` a hned na jejím začátku se testuje, zda-li se video právě nenatáčí a pokud ano tak tak ho zastaví. Stejně jako u focení na začátku definuji informace o souboru (název složený z lokálního času, formát a složku). Video má nastavenou délku, za kterou se má vypnout definovanou uživatelem. Je k tomu použita třída `Timer`. Během přípravy natáčení, `Recorder.prepareRecording`[3], se načtou definované informace o souboru. Po dokončení záznamu je video uloženo a podobně jako u vyfocení fotky vypíše na obrazovku zprávu zda bylo nebo nebylo uloženo.

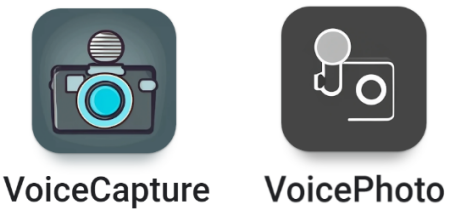
5.3 Publikace na Obchod Play

Po vyplnění potřebných informací, které jsou nutné kvůli vytvoření záznamu v obchodě, jsem vytvořil pro každou aplikaci logo a pořídil snímky obrazovky při používání. Další podstatná věc jsou názvy aplikací, knihovna PocketSphinx bude mít jméno VoiceCamera a Vosk se bude jmenovat VoicePhoto. Obě byly vydány do obchodu s aplikacemi. Ačkoliv první zmiňovaná pouze do interního testování, abych mohl potvrdit, že vývojářská sada Vosk je lepší. Což se prokázalo i mezi uživateli, což přiblížím v kapitole 5.4.

Vydání těchto aplikací usnadnilo hodně testování. Testeři si pouze stáhli aktuální verzi na Obchodu Play a mohli testovat. Nebyl tedy důvod jim stále zasílat nové verze pomocí

[2] https://developer.android.com/media/camera/camerax/preview
[3] https://developer.android.com/media/camera/camerax/video-capture

souboru `APK`[4]. Je to značné zrychlení, jelikož se dokonce automaticky aktualizují po připojení na Wifi. Navíc to urychlí i proces schválení aplikace společností Google, protože první schválení uživatele a aplikace trvá circa 10-14 dní. Tím pádem je cílem vývojáře si to odbít, co nejdřív. Vždycky se může vyskytne něco, co chybí nebo se musí doplnit a zdrží to tak vydání.



Obrázek 5.3: Ikony dvou aplikací, co byly testovány uživateli. Vlevo aplikace využívající knihovnu PocketSphinx, vpravo knihovnu Vosk. Mají odlišné ikony, kvůli rozeznání od sebe. I přes to obě zobrazují fotoaparát s mikrofonem. Aplikaci napravo lze stáhnout a je k vidění ještě stále na Obchodu Play[5]. Druhá aplikace se zobrazuje pouze testerům.

5.4 Testování

Testování probíhalo nejdříve formou debugování z mé strany. Hlavním cílem bylo odhalit chyby v implementaci nebo funkčnosti aplikace. Motivace byla zúžit výběr z knihoven, aby testeři nemuseli testovat nedostačující aplikace. Proto byly vydány pouze dvě aplikace s využitím dvou různých knihoven. Jednalo se o uzavřené testování, kde k nim měli přístup mí známí a vedoucí práce. Díky tomu jsem byl schopen odladit chyby na co nejvíce typech zařízení. Otestovalo se celkem 8 různých verzí systému Android na 14 mobilních zařízeních od 5 různých výrobců. Z nichž jedno bylo tablet.

U obou aplikací jsem zvolil stejné uživatelské rozhraní i proto aby to nemělo vliv na výsledky v testování rozpoznání hlasu. V průběhu testování ikony v aplikaci změnili často svou podobnu. Například díky testerům jsou výraznější (žluté) důležité slova v informačním náhledu. Další příklad je zelené a červené tlačítko *uložit* a *zpátky*. Od prvních uživatelů přišlo pár připomínek, jako například úprava funkčnosti ikon nebo přehlednost informační obrazovky. Řešením prvního problému bylo oprava logiky tlačítek a druhý jmenovaný byl řešen přepsáním informací na hesla s dvojtečkou. Přehlednosti bylo dosaženo zbarvením důležitých nastavení na žlutou barvu a seskupil podobné informace k sobě aby obrazovka nevypadala jako stránka v knize. Obarvení se dostalo i na tlačítka „zpátky“ a „uložit“, kde vrácení se na hlavní obrazovku dostalo červenou barvu a uložení nastavené zelenou. Po těchto úpravách byly aplikace dostupné více testerům.

Jak je zmíněno v kapitole 2.4, testování knihoven na rozpoznání hlasu proběhlo na začátku. V tuto chvíli testuji finální verze dvou vybraných. Uživatelé je testovali podle stejných pokynů a stejných slov v podobných prostředích. Někteří z nich ji dokonce využívali v běžném životě. Jejich poznatky mi pomohly vybrat vhodnější knihovnu. Většina z testerů řekli, že `VoiceCamera` je znatelně horší, kvůli nezachycení slov nebo nerozpoznání těch klíčových. `VoicePhoto` si vedla o poznání lépe.

[4] https://docs.fileformat.com/cs/compression/apk/
[5] 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. Následně ji využít pro demo aplikaci, která bude fotit fotky a natáčet videa. Abych mohl takovou aplikaci vytvořit, nastudoval jsem potřebné materiály týkající se hlasového rozpoznávání a lidskou tvorbu hlasu. Otestoval jsem zmíněné knihovny s mnoha uživateli. Tím jsem měl možnost testovat různé typy hlasu a 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. jako například český. Po vybrání jsem nastudoval také implementaci zachycení fotografie a natočení videa, dohromady s uložením daného souboru.

Z důvodu urychlení testovaní aplikace byla 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. Lze ji 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 na 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). Případné vytvoření vlastního jazykového modelu pro tuto knihovnu nebo upravit uživatelské prostředí, tyto skutečnosti nebyly cílem práce.

Vzhledem k focení a videu je dobré se rozhodnout, zda aplikaci udržovat pro starší verze Androidu. Používají zastaralé grafické prostředí. to může zapříčinit nedokonalou funkčnost starších zařízení. Po dodělání nynější verze aplikace se vydá nová verze Andoridu. Důležité je zvážit, zda by aplikace měla fungovat na telefonech starších 8 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

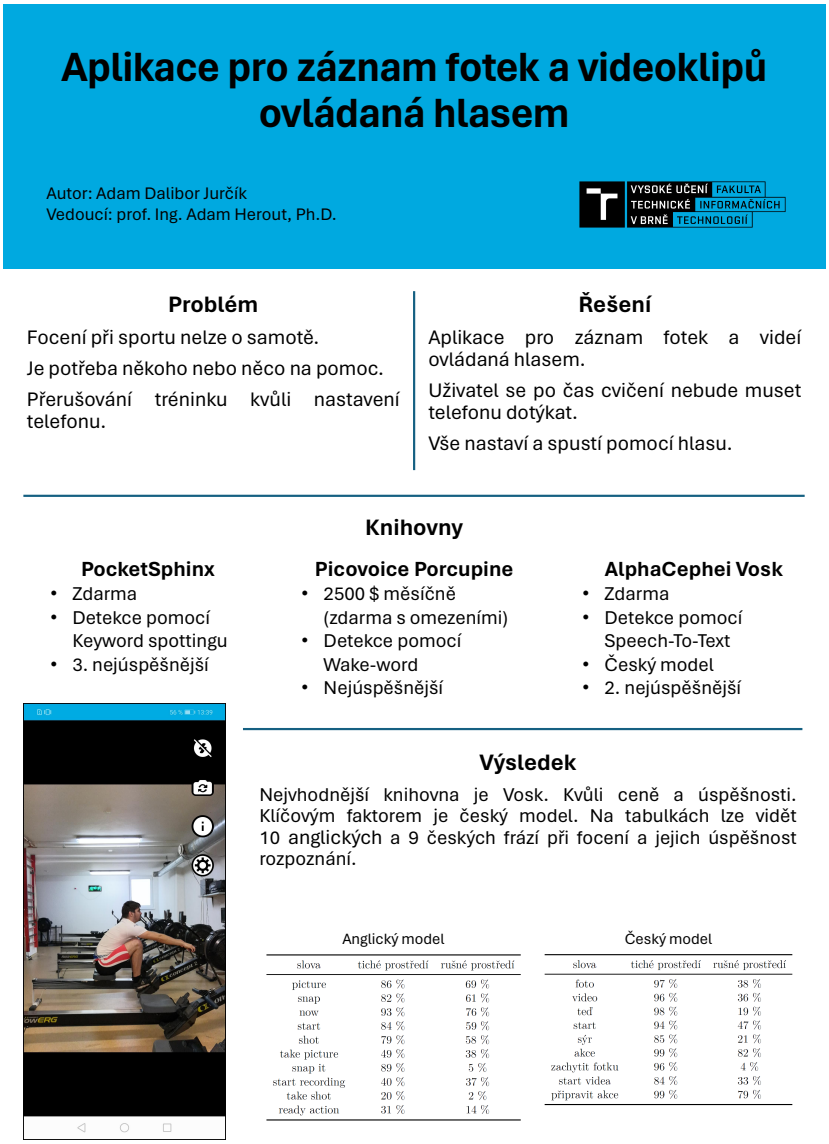
Fonetická abeceda britské angličtiny

	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
```