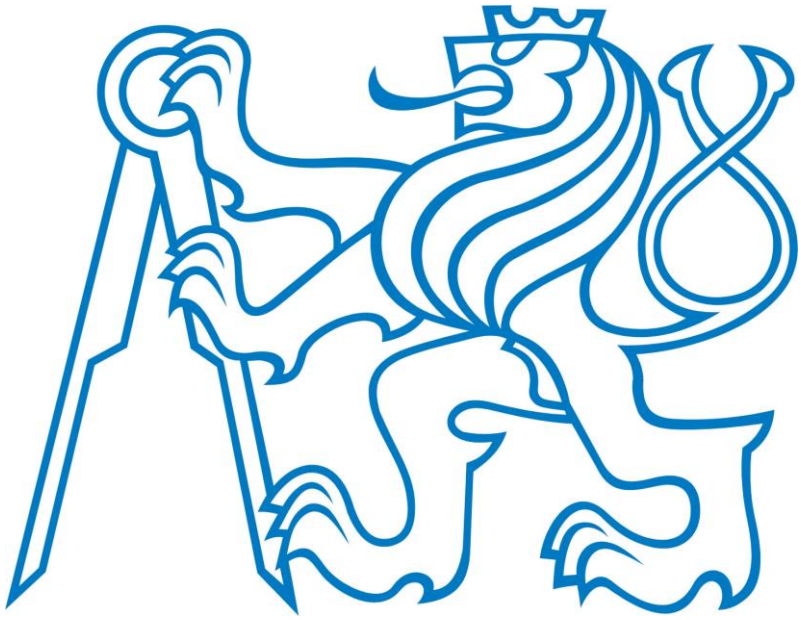
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická

DIPLOMOVÁ PRÁCE

2014 **Bc. Tomáš Remek**



Dialogový systém s hlasovou komunikací na platformě Android

Voice Driven Dialog System
at Android Platform

Diplomová práce

Studijní program: Komunikace, multimédia a elektronika
Studijní obor: Multimediální technika

Vedoucí práce: doc. Ing. Petr Pollák, CSc.

Bc. Tomáš Remek

Prohlášení

Prohlašuji, že svou práci na téma „Dialogový systém s hlasovou komunikací na platformě Android“ jsem vypracoval samostatně pod vedením vedoucího práce, s použitím odborné literatury, konzultací pana Dvořáka a dalších informačních zdrojů, které jsou uvedeny v přehledu literatury a zdrojů.

V Praze dne ....

(podpis autora)

Poděkování

Rád bych tímto poděkoval vedoucímu práce Doc. Ing. Petru Pollákovi, CSc. za ochotu a vstřícnost při řešení problémů a za čas, který mé práci věnoval. Dále bych rád poděkoval panu Patriku Dvořákovi za odborné konzultace ohledně vývoje aplikací a v neposlední řadě PaedDr. Ivaně Remkové za jazykovou korekturu práce.

Zadání

1. Seznamte se s problematikou rozpoznávání a syntézy řeči s užším zaměřením na úplnou hlasovou komunikaci při ovládání aplikací na platformě Android.

2. Implementujte rozpoznávání řeči pro hlasový vstup a syntézu řeči pro hlasový výstup realizujte pomocí volně dostupných nástrojů použitelných na platformě Android.

3. Nastudujte problematiku tvorby aplikací na platformě Android a vytvořte aplikaci realizující jednoduchý informační systém, jehož základem bude vhodně vybraný databázový systém umožňující snadnou aktualizaci, resp. změnu informačního obsahu.

Anotace

Náplní této diplomové práce je problematika vývoje aplikací pro operační systém Android s možností úplné hlasové komunikace mezi uživatelem a vytvořenou aplikací. Obecná část práce obsahuje shrnutí důležitých vlastností operačního systému Android, popis metodiky tvorby aplikací s užším zaměřením na nástroje podporující hlasový vstup a výstup, ale i základní popis principu rozpoznávání a syntézy řeči. Obecným cílem práce je vytvoření jednoduchého informačního dialogového systému. Práce obsahuje obecný návrh dialogového systému a jeho částí. V další části je popsána implementace vlastní aplikace pro Android, která využívá dostupné moduly této platformy včetně komunikace s relační databází umístěnou na webovém serveru a umožňující tak jednoduchou správu obsahu. Konkrétním výsledkem této práce je funkční, volně dostupná aplikace pro Android, která slouží jako informační systém pro laboratoř zpracování řečového signálu na ČVUT FEL, kde uživatel pomocí dialogu se zařízením získá informace o aktuálně realizovaných projektech této laboratoře, stejně jako například o nabízených tématech závěrečných prací.

Klíčová slova: dialogový systém, Android, rozpoznání řeči, TTS, hlasová syntéza, informační systém, Android a relační databáze

Anotation

This master thesis deals with the study of the application development for Android operating system with the focus on full voice communications at this platform. The theoretical part contains the summary of important features of Android operating system and also the description of the methodology of how to create an application with voice input and output. Supporting tools for voice control available in Android with the basic description of the speech recognition as well as synthesis principles are also discussed. The main task of this thesis is to create a simple dialogue information system, so the work contains both brief description of general dialogue system design and its particular parts. The next section describes the implementation of available modules for Android application including communication with relational database which is placed on web server and which allows an easy and user friendly content management. The free Android application realizing an information system for speech processing laboratory in the CTU is the practical result of this thesis. Users can obtain information about current projects by voiced-driven dialogue as well as about the offered thesis topics.

Key words: dialogue system, Android, speech recognition, text to speech, voice synthesis, information system, Android, relation database

Obsah

Seznam obrázků

Seznam zkratek

SQL	Structured Query Language
PHP	Hypertext Preprocessor
iOS	mobilní operační systém od společností Apple Inc.
API	Application Programming Interface
HMM	Hidden Markov Models
TTS	Text-to-speech
LAMP	Linux+ Apache + MySQL + PHP
WAMP	Windows + Apache + MySQL + PHP
CSV	Comma-separated values
JDK	JAVA Development Kit
SDK	Software Development Kit
IDE	Integrated Development Environment)
SP	Scale-independent pixels
DP	Density-independent pixels
IIS	Internet Information Services

1- Úvod

Mobilní telefony jsou již několik let nedílnou součástí našeho každodenního života a každým rokem rostou nároky na funkce a obsah těchto přístrojů. Uživateli je poskytováno několik různých způsobů interakce s jejich mobilním telefonem, ovšem nejpřirozenějším způsobem interakce zůstává řeč.

Android byl vydán jako open-source operační systém společností Google roku 2008 v reakci na první iPhone od společnosti Apple z roku 2007, který odstartoval novou éru tzv. chytrých telefonů. Tato zařízení udělala z obyčejných mobilních telefonů přístroje, které umí mnohem více než telefonovat a posílat SMS. Zároveň s příchodem technologie dotykových displejů jako vstupního rozhraní přišel prudký rozvoj právě těchto zařízení a spolu s ním rychlý vývoj tehdy mladého operačního systému- Androidu.

Proč pro realizaci dialogového systému zvolit právě android? Odpověď je velmi jednoduchá. Android je open-source operační systém, což znamená, že jeho zdrojový kód je volně přístupný. Ačkoliv je takto prezentován, není to úplně pravda. Android je z velké části otevřený, ale některé části jsou uzavřenou technologií firmy Google. Nicméně linuxové jádro, knihovny, aplikace, rozhraní apod. jsou otevřené, což dává možnost nejen výrobcům mobilních telefonů, ale i běžným uživatelům si Android přizpůsobit, jak je potřeba. Aplikace si může vytvořit každý bez velkých omezení. Díky vývojovému prostředí Eclipse, které je volně dostupné, a je i oficiálně doporučeno k programování aplikací pro Android, nemusí mít vývojář velké znalosti programování v jazyce JAVA, ve kterém se Android programuje, aby vytvořil základní aplikaci například s několika funkčními tlačítky. Je samozřejmostí, že pro složitější aplikace je nutno mít s programováním
v jazyce JAVA zkušenosti, ale i zkušenějším programátorům vývojové prostředí Eclipse velmi ulehčí práci.

Dalším, a pravděpodobně hlavním důvodem, proč vybrat právě Android, je možnost využití voice recognition, což je online hlasový rozpoznávač, který podporuje češtinu, a Google

k němu poskytuje přes jeho API bezplatný přístup. Složitější je to se syntézou řeči pro češtinu. Google zatím nepodporuje text-to-speech pro češtinu, a tak je nutné hledat jiné aplikace, které syntézu češtiny umí. Komunikace mezi aplikacemi v Androidu není problémem. Aplikace na této platformě mohou pro vzájemnou komunikaci využívat tzv. Intents, a tak je možno pro vytvářený dialogový systém využít hlasový modul v českém jazyce z jiné aplikace.

Vzhledem k požadavku měnit informace v dialogovém systému je výhodnější vytvořit externí databázi a umístit ji na server, kde bude možnost jí pohodlně měnit a aktualizovat bez nutnosti stahovat aktualizace aplikace do všech zařízení. Android samozřejmě podporuje databázový systém, konkrétně se jedná o relační databázový systém SQLite, ovšem vzhledem k tomu, že Android zatím nepodporuje rozpoznávání řeči off-line a že přenášená data z a do databáze nejsou nikterak velká, je praktičtější vytvořit databázi externí.

Android je prudce se rozvíjejícím operačním systémem, který se stal během několika posledních let absolutně nejrozšířenějším OS pro chytré telefony, a tak i kvůli jeho dominanci na trhu s chytrými telefony a důvody uvedenými výše, je vytvořený hlasový dialogový systém naprogramován právě pro platformu Android.

2- OS Android a jeho historie

Prvním krokem mé práce bylo seznámení se s platformou, na které má být dialogový systém postaven. Důležitá byla nejen historie verzí, ve které jsem se seznámil s vývojem a zjistil, kdy byly jednotlivé moduly, především ty potřebné pro realizaci dialogového systému, implementovány, ale důležitá byla především struktura a princip fungování tohoto systému a jeho jednotlivých částí. Neméně podstatné bylo i aktuální rozložení verzí v chytrých telefonech ve světě, podle kterých bylo nutno vybrat nejvhodnější verzi, jež by umožňovala správnou funkcionalitu a zároveň fungovala na co nejvíce přístrojích.

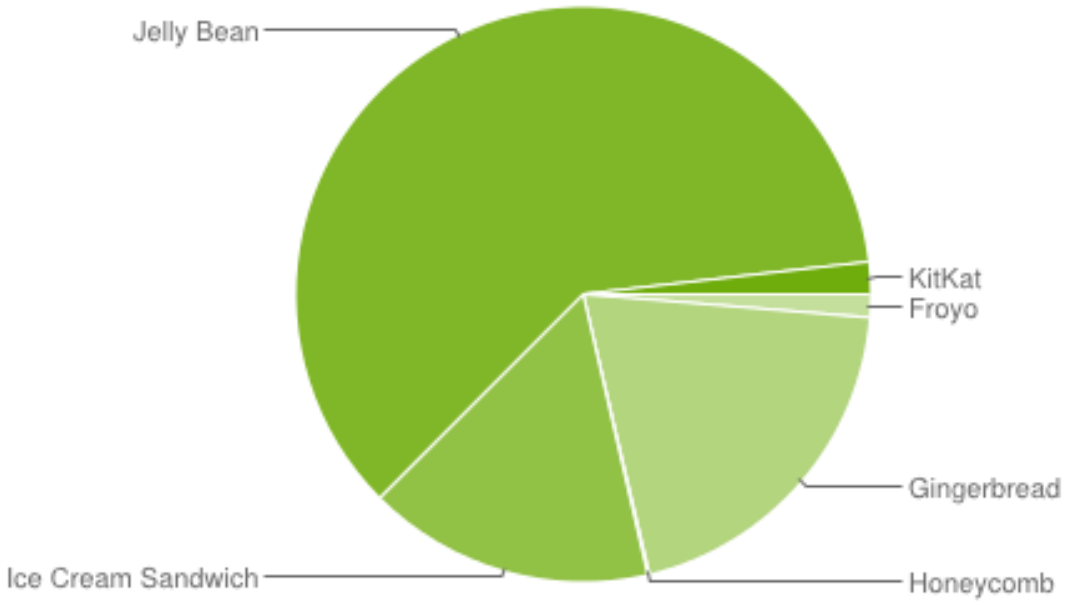
2.1 Historie operačního systému Android

Pokud se podíváme do úplných počátků této platformy, zjistíme, že Android jako takový nepochází od společnosti Google, ale od pánů Andyho Rubina, Riche Minera, Nicka Searse a Chrise Whitea, kteří spolu v Palo Alto v Kalifornii založili v říjnu roku 2003 společnost nazvanou Android Inc. Původním záměrem této společnosti bylo vyvinout pokročilý operační systém pro digitální fotoaparáty. Ovšem trh s těmito produkty nebyl dostatečně velký, a tak se firma zaměřila na vývoj operačního systému pro chytré telefony, který by byl schopen konkurovat tehdejším systémům, jako byl Symbian nebo Windows.

Google koupil tento vývojový projekt v srpnu roku 2005 a udělal z něj svojí dceřinou společnost. Do čela byl dosazen výše zmíněný Andy Rubin (mimo jiné spoluzakladatel firmy Danger). Milníkem ve vývoji byl 5. listopad roku 2007, kdy bylo společnostmi jako HTC, SONY, Samsung, T-mobile, Texas Instrument, a především Google vytvořeno uskupení „Open Handset Alliance“, jehož smyslem bylo vyvinout otevřený standard pro mobilní zařízení. Zároveň byl ten den představen Android OS jako nová platforma pro mobilní zařízení, která je založena na linuxovém jádru Kernel verze 2.6. Za necelý rok byl vydán první telefon s Androidem, a to HTC Dream. Na český trh se dostal počátkem roku 2009. Operačním systémem tohoto zařízení byl první plně funkční Android, a to verze Android 1.0- Astro. V této verzi bylo již několik základních aplikací, které jsou typické pro Android dodnes, a to např: Google Contacts, G-mail client, Google Callender apod.

2.2.1 Verze OS Android

Celá platforma prošla složitým vývojem a sérií inovací. Vývojové tempo je velmi vysoké, protože s novou verzí operačního systému přichází Android zhruba dvakrát do roka. Pro porovnání např. s operačním systémem Windows se jedná doslova o zběsilé tempo rozvoje, protože nové verze operačního systému Windows vycházely v průměru po dvou letech. Podobně je na tom Apple se svým iOS. Níže je uveden přehled několika verzí Androidu, které byly důležitými milníky tohoto operačního systému. Informace pro celou tuto kapitolu jsem čerpal z [5] , [6] a [7].



Obr. 1: Graf rozložení verzí OS Android, převzato z [7].

Android 1.0 - Astro (API level 1)

Jak bylo již zmíněno, první plně funkční verzí byl Android 1.0, později nazvaný Astro. Kromě základních aplikací podporoval WiFi, Bluetooth, GPS, různé typy polohových senzorů, paměťové karty, dotykové displeje nebo fotoaparát. A hlavně tzv. Android Market, což je aplikace, díky které si uživatel může stahovat do svého zařízení aplikace schválené Googlem.

Android 1.5 - Cupcake (API level 3)

Tato verze byla vydána 30. dubna roku 2009. Opět rozšiřovala a opravovala chyby některých aplikací a byla založena na jádře kernel v2.6.27. Podstatné v této verzi bylo přidání softwarové klávesnice, jelikož do této doby musely mít chytré telefony k ovládání Androidu klávesnici hardwarovou. Přibyla dále podpora tzv. widgetů (lze si je představit jako zástupce v systému Windows) pro jednodušší přístup k aplikacím, nebo pro informace aplikací běžících v pozadí, např. počasí, zprávy apod. Dále možnost nahrání videosouborů a následné přehrání, a to ve formátech MPEG-4 a 3GP. S tím se také změnila aplikace pro focení, přišlo vylepšení webového prohlížeče (funkce copy, paste apod. mezi prohlížečem a aplikacemi nebo funkce vyhledávání v textu), vylepšení přenosů a párování u technologie bluetooth (A2DP-Advanced Audio Distribution Profile). Obecně došlo ke zrychlení systému, který uměl lépe pracovat s aplikacemi jako Gmail, fotoaparát nebo internetový prohlížeč.

Android 1.6 - Donut (API level 4)

Verze 1.6 Donut běžela na vylepšeném linuxovém jádře kernel 2.6.29. Byla vydána 15. září 2009 a kromě vylepšení a oprav předchozích verzí přinesla pro tuto práci velmi podstatnou věc, a to modul pro syntézu řeči.

Tímto modulem je **Multi-lingual speech synthesis engine,** nazvaná Pico, vytvořená společností SVOX, která uměla syntetizovat v angličtině (britský i americký akcent), francouzštině, italštině, němčině a španělštině. Ostatní aplikace mohly tento modul využívat k syntéze řeči. Jazyková

mutace pro češtinu vytvořena nebyla a v době psaní této práce stále není. Ovšem díky otevřenosti Androidu je možno tento modul obsluhovat jinými aplikacemi, které český jazyk podporují, a lze tak v androidu využít syntetizátor češtiny. Pro vytvářený dialogový systém je tudíž tato verze stěžejní.

Dále byla v této verzi přidána funkce, která zobrazuje vytíženost baterie v závislosti na spuštěných aplikacích. Vylepšena byla i technologie WiFi, která byla doplněna protokoly 802.1x.Nově přidána byla i podpora větších displejů s lepší rozlišovací schopností (WVGA- Wide Video Graphics Array).

Android 2.3 - Gingerbread (API level 9)

Vydáno 6. prosince 2010 pod názvem Android 2.3 Gingerbread s novým jádrem Kernel 2.6.35. Implementována byla podpora technologie NFC (Near Field Communications), umožňující vysokorychlostní bezdrátové přenosy na kratší vzdálenosti. Podporovány jsou nyní i nové typy displejů, jako je například AMOLED. Významným počinem byla podpora různých typů senzorů (např. barometr, akcelerometr nebo gyroskop). Dále je tu podpora pro internetové volání VoIP(Voice over Internet Protocol) komukoliv, kdo má SIP (Session Initiation Protocol) account.

Pro možnosti dialogového systému byl velmi důležitou částí, vydanou s touto verzí, hlasový vstup **Voice Action**, který umožnil uživateli vyhledávat v textu, na webových stránkách nebo zadávat pokyny nebo diktovat SMS pomocí hlasu. **Čeština** je v rámci Voice Search od Googlu k dispozici právě od roku 2010, v aplikacích pro mobilní telefony od roku 2011 spolu s verzí 4.0. Google poskytuje k tomuto svému modulu bezplatný přístup, takže od této verze implementuje Android jak hlasový syntetizér, tak rozpoznávač řeči pro volné využití, což znamená, že dialogový systém na této platformě je možno realizovat již od této verze.

Android 4.0 - Ice Cream Sandwich (API level 14)

Důležitým milníkem byl pro platformu Android 1.říjen 2011, kdy byla vydána verze Android 4.0 s označením Ice cream sandwich (ICS). Ta z Androidu udělala univerzální OS pro chytré telefo-

ny i tablety. Bylo nutno přepracovat grafické prostředí, aby bylo možno jej používat pro oba typy přístrojů. Nově přidanou funkcí je kontrola pravopisu, vylepšený byl hlasový vstup a přidány nové standardizované slovníky. Webový prohlížeč byl zrychlen novými vykreslovací algoritmy. Dále byla přidána aplikace Android Beam, která umožňuje pohodlně přenášet data pomocí NFC (Near Field Communication).

Android 4.1 - Jelly Bean (API level 16)

27. června 2012 byla vydána další významná verze OS Android a to 4.1 nazvaná Jelly Bean, která je v současnosti ve svých verzích 4.1, 4.2 a 4.3 nejrozšířenější verzí vůbec. Důležitými vylepšeními jsou Google Now, což je inteligentní osobní asistent, který pomocí hlasového dialogu s uživatelem dokáže vyhledat informace, pomáhá s navigací, připomíná události v kalendáři nebo realizuje povely dané uživatelem. Je to v podstatě plnohodnotný osobní asistent a jedná se o velmi pokročilý dialogový systém. Vylepšení se dočkalo hlasové vyhledávání, aplikace fotoaparátu, multikanálové audio nebo USB audio. K funkci Android Beam byla přidána podpora bluetooth přenosů. Je nyní podporována i možnost více uživatelských účtů na jednom telefonu. Ve verzi. 4.2 byla přidána modifikace k linuxovému jádru, a to SELinux (Security-Enhanced Linux), což je modul, který poskytuje prostředky a mechanismy podporující přístupová bezpečnostní pravidla. Zajímavá je poté ve verzi 4.3 (API level 18) například podpora 4K rozlišení.

2.2.2 Rozšíření verzí

Jak je vidět z obr. 1, v současnosti je nejrozšířenější verzí Jelly Bean. Je to samozřejmě dáno především každoročním růstem prodeje chytrých telefonů, ve kterých se samozřejmě objevují novější verze OS. V roce 2013 dosáhl světový odbyt chytrých telefonů jedné miliardy kusů, což je růst zhruba o 40% oproti předchozímu roku (Údaje analytické společnosti IDC a Strategy Analytics) [8].

V posledním kvartálu roku 2013 byl podle společnosti ABIresearch [9] podíl operačních systémů v nových chytrých telefonech následující: Android v certifikované verzi je v 52% všech zařízení, v té necertifikované verzi (různé větve androidu pro státy, kde nejsou podporovány všechny

služby Google, např. Čína) činí 25% z celkového počtu. Celkem tedy na Androidu běží 77% nových mobilních zařízení. Společnost Apple a její iOS se podílí na celkové produkci 18%. MS Windows Phone je třetím nejrozšířenějším OS se 4% z celkové produkce. Zbytek jsou operační systémy, jako BlackBerry, Firefox, Symbian nebo MS Windows Mobile. [7]

Verze	Označení	API level	Distribuce
2.2	Froyo	8	1.3 %
2.3.3 – 2.3.7	Gingerbread	10	20 %
3.2	Honeycomb	13	0.1 %
4.0.3 – 4.0.4	Ice Cream Sandwich	15	16.1 %
4.1.x	Jelly Bean	16	35.5 %
4.2.x		17	16.3 %
4.3		18	8.9 %
4.4	KitKat	19	1.8 %

Tab. 1: Verze Androidu podle distribuce [8].

2.3 Vlastnosti Androidu:

Jak už bylo zmíněno, Android je operační systém pro chytré mobilní telefony a tablety postavený na linuxovém jádře. Hlavními rysy tohoto systému jsou podle [7]:

- bezplatné využití a přizpůsobení OS výrobci mobilních zařízení
- bezplatné stažení vývojového prostředí pro aplikace

- rychlý a snadný vývoj aplikací využívající rozsáhlých databází softwarových knihoven a vývojových nástrojů
- rovnost základních a přidaných aplikací v přístupu ke zdrojům
- optimalizované využití paměti aplikacemi
- vysoká kvalita audiovizuálního obsahu - vektorová grafika a většina audio a video formátů
- vývoj a testování aplikací je možné na většině počítačových platforem, jako např. Windows, Linux nebo iOS

Architektura OS Android je rozdělena na 5 vrstev, z nichž každá má svou funkci a společně spojují hardware zařízení s uživatelem. První vrstva je jádro systému založené na OS Linux, a to Kernel 2.6.x, jehož verze se liší pro různé verze OS Android. Pro použití v mobilních telefonech byl Kernel upraven, především kvůli spotřebě energie. Tato vrstva se stará přímo o hardware zařízení. Zajišťuje řízení paměti a procesů, síťových prvků nebo zabezpečení systému. Vlastnosti této vrstvy určují vlastnosti celého systému.

Druhou vrstvou jsou Libraries čili knihovny. Ty jsou napsány v programovacích jazycích C a C++ a mají za úkol rozšířit základní funkce jádra. Jedná se například o Media libraries, které umí pracovat s multimediálními soubory nebo např. SQLite libraries pro práci s databázemi.

Třetí vrstva Runtime obsahuje virtuální stroj Darvik, vyvíjený od roku 2005 právě pro Android, a také základní knihovny programovacího jazyka Java. Dalvik Virtual Machine (DVM) je registrově orientovaná architerktura, která pracuje s Linuxovým jádrem a využívá jeho vlastnosti, jako jsou správa paměti nebo práce s vlákny. Každá spuštěná aplikace je samostatný proces, který má vlastní instanci DVM. Aplikace napsaná ve zdrojovém kódu Java je přeložena do Java byte kódu, poté kompilována Dalvik kompilátorem na Dalvik byte kód a nakonec spuštěna na Dalvik Virtual Machine.

Application Framework je název čtvrté vrstvy, kde jsou programátorům zpřístupněny služby využívané přímo v aplikacích. Je to rozšiřitelná sada softwarových komponent používaná všemi

aplikacemi v operačním systému. Jako příklad lze uvést View, což je sada prvků, které jsou použity při sestavování uživatelského rozhraní nebo Activity manager, který řídí životní cyklus aplikací.

Poslední a nejvyšší vrstvu tvoří Aplikace (Applications). Tato vrstva je viditelná pro uživatele. Aplikace jsou zde psány v programovacím jazyce Java a dělí se pouze na aplikace dodáváné s operačním systémem a na ty, které si uživatel sám stáhne a nainstaluje.

Mezi základní stavební prvky aplikací řadíme activity , service, content provider a broadcast reciever. Tyto součásti (mimo content provider) spolu mohou komunikovat pomocí tzv. intents.

Activity neboli aktivita je hlavním prvkem každé aplikace a odpovídá jedné obrazovce. Aplikace může mít více aktivit a každá z nich je potomkem třídy Activity. Každá z aktivit obsahuje GUI pro komunikaci s uživatelem, který je schopen mezi aktivitami přepínat. Základní startovací aktivitou je MainActivity. Vzhledem k úspoře výpočetního výkonu funguje v Androidu tzv. Activity Manager, který řídí životní cyklus aktivity. Ten ukládá do zásobníku informace o všech spuštěných aktivitách a aktuálně zobrazovaná aktivita je na vrchu tohoto zásobníku. Životní cyklus aktivity ukazuje Obr.2, kde jsou vidět základní stavy, ve kterých se aktivita může nacházet. „Activity launched“ je spuštění, kdy je inicializována aktivita. Dále je zde „Activity running“, kdy je aktivita zobrazena na displeji a dochází k interakci s uživatelem. „App process killed“ nastává, když Activity Manager zrušil aktivity kvůli nedostatku paměti. „Activity shut down“ je ukončení dané aktivity Activity Managerem a je to stav, kdy již aktivita nevyužívá žádnou paměť.

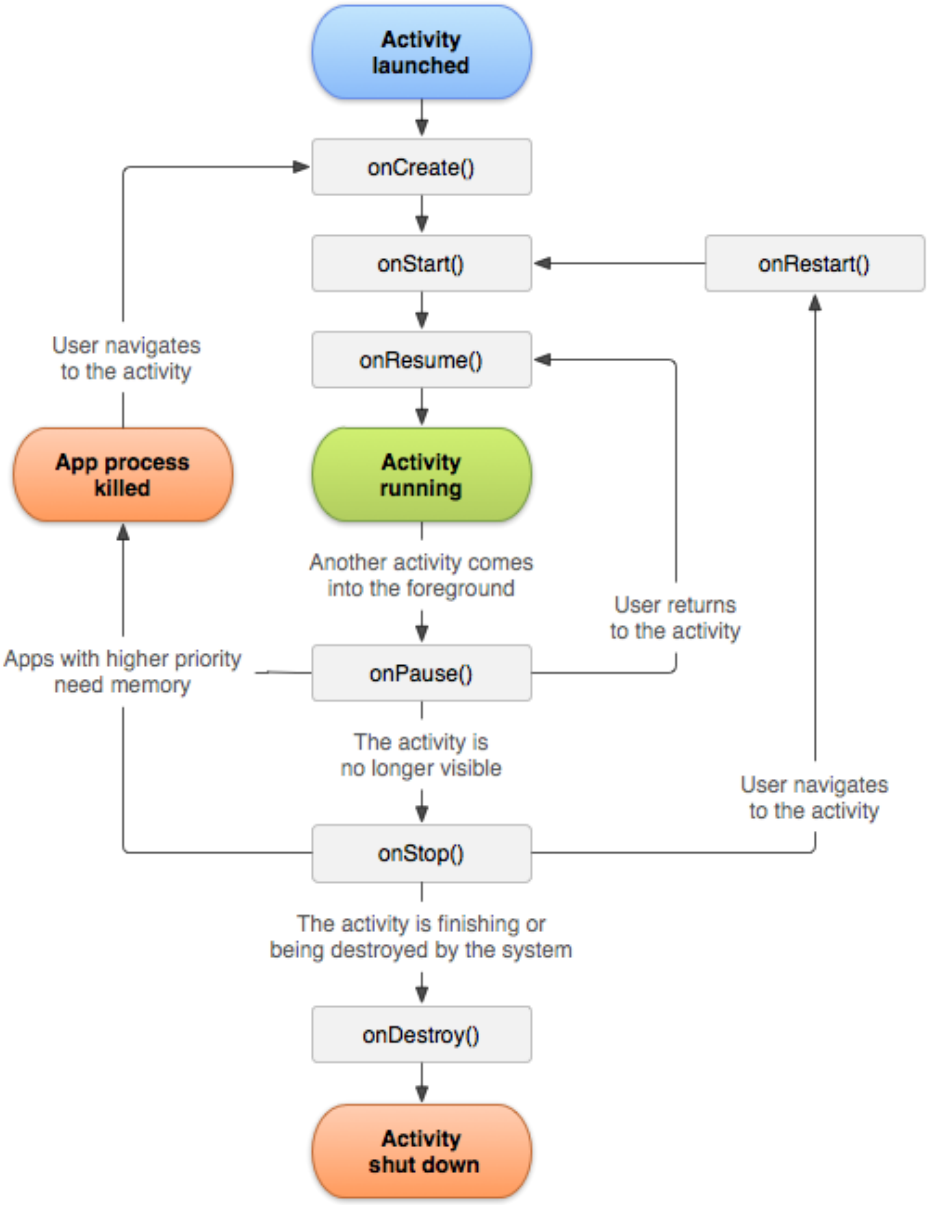
Service je proces běžící na pozadí bez uživatelského rozhraní, který se většinou používá k vykonávání dlouhotrvajících úkolů nebo k přístupu ke vzdáleným zdrojům, jako připojení k serveru apod.

Content provider poskytuje přístup k datům. Je to aplikační rozhraní pro sdílení dat mezi aplikacemi nebo mezi jednotlivými aktivitami. Data jsou uložena v databázích, souborech nebo na webu a každá aplikace, která bude mít povolení, je může použít.

Broadcast reciever umožňuje reagovat podle určení na určitá příchozí oznámení. Reaguje například na doručení SMS nebo stažení dat a nadefinovanou reakcí na ně odpovídá (zvuk, vibrace, výpis ve stavovém řádku apod.).

Pod pojmem intent si lze představit zprávu používanou ke spuštění aktivity nebo služby. Obsahuje název komponenty, kterou chceme spustit, akci, kterou je třeba ukončit, adresu ulože-

ných dat potřebných ke spuštění a typ komponenty. Intent lze použít i jako jednoduchý „kontejner“ na data, která jsou mezi aplikacemi nebo aktivitami posílána.



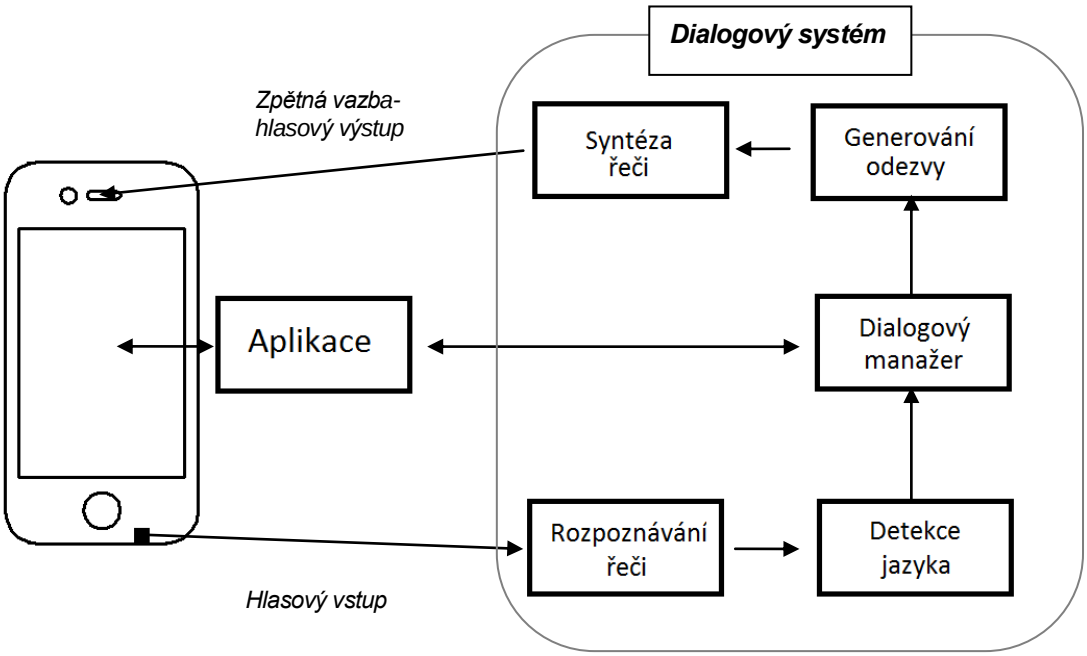
Obr. 2: Životní cyklus aktivity, převzato z [7].

3. Aplikace s hlasovým rozhraním

Pro vytvoření hlasového dialogového systému na platformě Android bylo nutno se nejdříve seznámit s jednotlivými základními stavebními prvky takovýchto aplikací. Od základů návrhu dialogových systémů přes teorii syntézy a rozpoznávání řeči, až po jednotlivé prvky potřebné k vlastní realizaci, jako jazyk PHP, SQL databáze a jejich správa a samozřejmě i s vývojovým prostředím aplikací pro Android. Jednotlivé části budou podrobněji diskutovány v následujících částech této kapitoly.

3.1 Hlasový dialogový systém

Hlasový dialogový systém slouží jako rozhraní při komunikaci člověka a počítače pomocí hlasu. Počítačem jsou zde myšleny aplikace ať už internetové nebo počítačové, různé systémy řízení a monitorování, nebo databázové systémy. Existuje široká škála různých druhů systému od těch nejjednodušších, založených na dialogu pomocí několika málo povelů, až po systémy schopné komunikovat souvislou řečí.



Obr. 3: Schéma hlasového dialogového systému.

Na obrázku vidíme schéma typického hlasového dialogového systému obsahující všechny potřebné moduly k realizaci takovéhoto systému. Nejpodstatnějším modulem je dialogový manažer, který řídí, spojuje a vyhodnocuje informace od a k ostatním modulům systému tak, aby byl správně splněn úkol zadaný dialogem. Je málo časté, že by požadovaný výsledek nebo informaci získal uživatel zadáním jednoho dotazu, ať už proto, že dotaz může být nepřesný nebo neúplný, rozpoznávání řeči bylo nepřesné, nebo je jednoduše jen vyžadováno potvrzení volby. Proto je nutno, aby si byl dialogový manažer schopen zajistit a potvrdit všechny potřebné informace ke správnému vyřešení dané úlohy [1].

Dialogové systémy rozdělujeme na tři typy podle přístupu k dialogu. Jsou to systémy s konečným počtem stavů, systémy založené na rámcích a systémy založené na agentech.

3.1.1 Dialogový systém s konečným počtem stavů

Dialogový systém s konečným počtem stavů si lze představit jako stavově přechodovou síť, kde jednotlivé uzly sítě představují stavy dialogu, ve kterých je od uživatele získávána nějaká informace, která rozhoduje o dalším postupu sítí. Spoje mezi uzly pak představují všechny možné vývoje dialogu. Typické je pro tyto systémy jednoduchost povelů, kdy jsou odpovědi uživatele co možná nejjednodušší, jako ANO/NE, nebo dialogový systém „podsouvá“ uživateli ta správná slova, která by měl v dialogu použít (Systém: „Chcete majonézu, nebo kečup?“; Uživatel: „Majonézu“). Tato strategie se nazývá řízení s iniciativou systému. Výhodou tohoto typu dialogového systému je právě jednoduchost dialogu. Systému stačí rozpoznat jen malý počet izolovaných slov, jejichž slovník je v každém uzlu sítě předdefinován, a tudíž jsou zde nízké nároky na výkon modulu rozpoznávání i porozumění řeči. Zároveň to lze ale také považovat za nevýhodu, protože není možné v takto definované síti uskutečnit dialogovou změnu mimo tuto síť. Typicky se jedná o opravu zadaných údajů, přidání dodatečných informací nebo vložení více položek najednou v rámci urychlení zadávání. [1]

3.1.2 Dialogový systém využívající strukturu rámců

Dialogový systém využívající strukturu rámců využívá trochu odlišný přístup k dialogu. Pro každou řešenou úlohu je definován rámec nebo struktura rámců, jejichž sloty reprezentují jednotlivé části řešené úlohy. Systém potřebuje dané rámce zaplnit informacemi získanými z dialogu s uživatelem. Když jsou vhodnou informací vyplněny všechny rámce, nebo jejich dostatečný počet, systém splní požadovanou akci. Aby získal potřebné informace, klade uživateli vhodné otázky formulované tak, aby byl uživatel nucen na ně odpovídat tak, jak systém potřebuje. Výhodou je, že na rozdíl od dialogového systému s konečným počtem stavů může uživatel zadávat informace v různém pořadí nebo i jedinou promluvou. Pokud nejsou vyplněny všechny potřebné sloty, dialogový manažer vyhodnotí, jaké informace je třeba získat a položí uživateli doplňkovou otázku nebo otázky připravené pro zaplnění daného rámce. Jedná se o tzv. smíšenou iniciativu dialogu. Často se ale vyskytuje případ, kdy není možno získat od uživatele relevantní informace touto smíšenou iniciativou, a tak systém zareaguje tím, že přepne na iniciativu systému a v následném dialogu s uživatelem pokládá otázky tak, aby obdržel jednoslovné odpovědi a dobral se tak požadovaných informací, které poté pošle k vyhodnocení. Pro více specifických oblastí se používají systémy několika rámců, které zvládnou tyto složitější dialogové úlohy.

Je zřejmé, že k úspěšnému dokončení dialogu je u dialogových systémů využívající strukturu rámců třeba méně dialogových výměn než u předchozího systému, a to především možností zadávat více informací najednou. Samozřejmě to zvýší nároky na jednotlivé bloky, ovšem dialog se zrychlí. I přes větší pružnost v komunikaci mezi systémem a uživatelem není stále možné tímto systémem realizovat složitější komunikaci, protože stavy systému jsou pořád závislé na těch předešlých a vždy je tu nutnost vyplnit sloty relevantní informací. [1],[2].

3.1.3 Dialogový systém založený na agentech

Dialogový systém založený na agentech je ve své původní myšlence systém, který by umožnil uživateli současnou a nezávislou komunikaci s více informačními zdroji najednou. V prostředí hlasového dialogu je zde ovšem nutnost zadávat a odebírat všechny potřebné informace sek-

venčně, což je náročné pro systém i uživatele při orientaci v dialogu s danou částí informačního zdroje. Řešením tohoto problému je využití architektury spolupracujících agentů. Princip spočívá v tom, že mezi sebou jednotlivé části spolupracují a pokud tedy uživatel zadá nějaký příkaz v jedné oblasti, informace se přenese i do ostatních oblastí, kde se hledá logická souvislost zadaného příkazu a ve vedlejších oblastech. Takže při komunikaci agentů se nepřenáší jen data, ale i znalosti logicky nabyté z dosavadního dialogu. Aby toto mohlo fungovat, agenti sdílí informace na třech úrovních: syntaktické, pragmatické a sémantické. Tento typ dialogového systému velmi úzce souvisí s vývojem umělé inteligence [1].

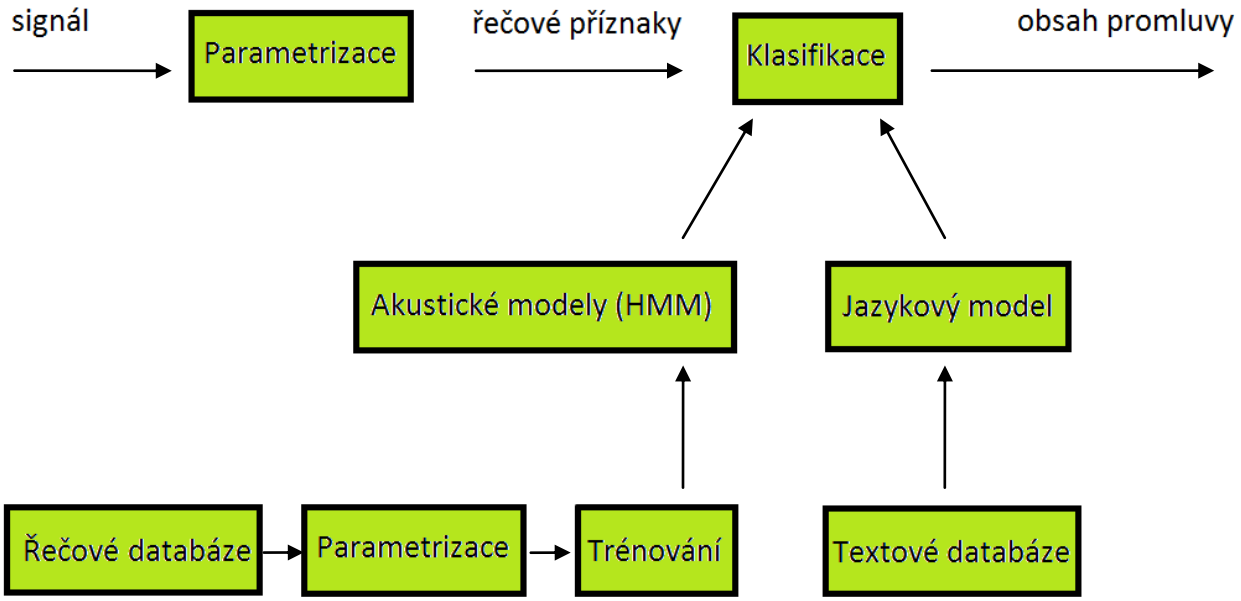
3.2 Rozpoznávání řeči

Rozpoznávání řeči je společně se syntézou řeči jedním ze základních prvků vytvořené aplikace. Díky rozpoznávání řeči jsme schopni ovládat aplikaci konkrétními hlasovými povely, a je tudíž možno vytvořit dialog mezi uživatelem a zařízením.

Rozpoznávání řeči se využívá pro ovládání jednoduchých zařízení, jako jsou telefony, počítače, nebo nachází využití v automobilech. Setkáváme se s nimi v informačních a dialogových systémech, čehož je vytvořená aplikace typickým příkladem. Využívá rozpoznávání jednoduchých příkazů v mobilním telefonu k získání informací z dialogového systému. Rozpoznávání řeči se dále používá například pro rozpoznávání mluvčího. Jiným využitím rozpoznávačů jsou diktovací systémy, transkripce audio nahrávek nebo on-line titulkování. Neméně důležitou oblastí, kde se s rozpoznávači řeči setkáváme, je pomoc handicapovaným osobám. Rozlišujeme různé typy rozpoznávačů, od jednoduchých, které dokáží identifikovat jednotlivá izolovaná slova, přes složitější, které dokáží rozpoznat krátké fráze, až ke složitým rozpoznávačům, které jsou schopny rozpoznávat spojitou řeč [2].

Schéma rozpoznávání řeči je v základu následující: příchozí řečový signál je nejdříve parametrizován, což znamená, že je rozdělen na krátké řečové segmenty, které jsou poté popsány danými parametry, označovanými jako řečové příznaky nebo deskriptory. Typickými příznaky jsou kepstrální koeficienty, které získáme inverzní Fourierovou transformací logaritmu komplexního

spektra. Součástí této fáze může být i například robustní parametrizace (potlačování vlivu šumového pozadí) nebo speciální parametrizační techniky. V druhém kroku jsou různými technikami klasifikovány příchozí příznaky a je vytvořen obsah promluvy [2], [10].



Obr. 4: Blokové schéma procesu rozpoznávání řeči.

Klasifikátory řeči lze dělit do dvou skupin: na ty, které pracují na principu porovnávání se vzory (template maching), a na klasifikátory, jež využívají statistických metod. V období sedmdesátých a osmdesátých let byla aktuální první metoda, která zpracovávala slovo jako celek a na základě využití metody dynamického programování porovnávala vzdálenost posloupnosti příznakových vektorů rozpoznávaného slova s obrazy slov uložených ve slovníku rozpoznávače. Od této metody se již téměř upustilo kvůli malému rozsahu vzorů a v současnosti se využívá metoda druhá.

V systémech pracujících na základě statistických metod jsou v dnešní době nejčastěji slova, části slov (fonémy, trifóny apod.) nebo celé promluvy modelovány pomocí skrytých Markovových modelů (HMM - Hidden Markov Models). Základem tohoto způsobu rozpoznávání je konečný

automat, kde jsou jednotlivé stavy automatu propojeny do posloupnosti. Zprůměrováním velkého množství promluv (promluvy jsou součástí tzv. trénovací databáze) vznikne statistický model parametrů, který nahrazuje konkrétní vzor [1], [2].

Příchozí signál je nejdříve navzorkován. Vzorky řečového signálu jsou analyzovány v pravidelných intervalech 20-30ms, protože je signál v tomto intervalu považován za stabilní. Následuje parametrizace.

Nejčastěji se jako příznaky používají mel-frekvenční kepstrální koeficienty, které aproximují nelinearitu vnímání frekvence lidského sluchu. Na signál je nejdříve aplikována diskrétní Fourierova transformace. Dále je aplikována melovská banka filtrů, která má nelineární frekvenční osu a obsahuje různý počet pásem, nejčastěji je to 22. Jednotlivé filtry mají modulovou přenosovou frekvenční charakteristiku trojúhelníkového tvaru, kde se sousední frekvenční pásma překrývají přesně o polovinu. Výstupem z banky filtrů je výkonové spektrum. Podstatné pro výpočet konečných příznaků jsou logaritmy energie jednotlivých pásem, ze kterých získáme koeficienty mel-kepstra aplikováním inverzní diskrétní kosinové transformace (IDCT).

Po parametrizaci máme signál zakódován do posloupnosti kepstrálních koeficientů, jejichž shluky (spektrálně podobné zvuky) definují jednotlivé stavy modelu. Výskyt v jednom z těchto stavů je popsán pravděpodobností stavu a délka setrvání v tomto stavu je popsána pravděpodobností přechodu mezi stavy. Rozpoznání poté probíhá vyhodnocením možnosti průchodu získané posloupnosti kepstrálních vektorů jedním z natrénovaných modelů. Ten model s největší pravděpodobností průchodu všemi stavy je poté určen jako výsledek [2]. Součástí procesu vyhodnocování je i jazykový model, čerpající z textové databáze.

V aplikaci není úkolem implementovat vlastní rozpoznávač řeči, jednak kvůli náročnosti realizace vlastního rozpoznávače a také kvůli kvalitě, neboť dostupný rozpoznávač od Googlu je na velmi vysoké úrovni. Rozpoznání řeči je tedy ve vytvořené aplikaci prováděno na Googlovském serveru, kde se využívají skryté Markovovy modely. V případě Google rozpoznávače jsou také jako parametr použity melovské kepstrální koeficienty (MFCC). Evropský ústav pro telekomunikační normy v roce 2000 standardizoval MFCC algoritmus pro použití v mobilních telefonech [11].

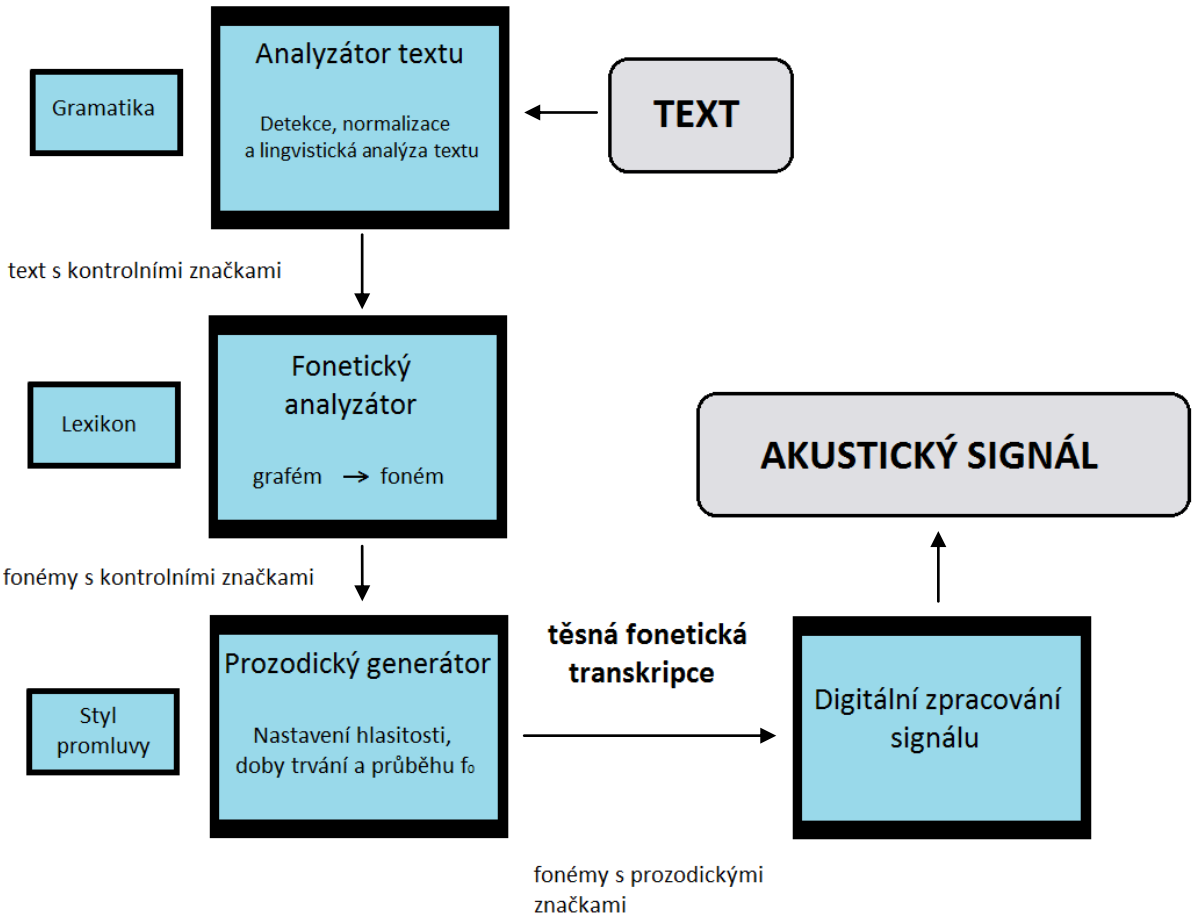
Google umožňuje bezplatný přístup ke svému rozpoznávači od Android API level 9. Tvůrci aplikací pro Android mohou tudíž tuto službu použít ve svých aplikacích. Implementace je formou knihoven a syntaxe je popsána na stránkách developer.android.com. Jedná se o online rozpoznávač, takže bez připojení k internetu nefunguje. Na server Google se neposílá celý zvukový záznam, ale v zařízení probíhá rovnou předzpracování a posílají je pouze koeficienty. Alternativní možnost pro český jazyk by znamenala naprogramovat svůj rozpoznávač a implementovat jej do Androidu, nebo přenášet celý zvukový záznam na server s rozpoznáváním řeči [7].

3.3 Syntéza řeči

Zpětnou vazbu aplikace umožňuje hlasový syntetizátor, který převádí text na řeč. Android má v základu již od verze 1.6 API level 4 zabudovaný modul, který lze využít k syntéze řeči. Ovšem dostupný je pouze pro velké světové jazyky. Nepodporuje ale češtinu, proto je nutno najít aplikaci, která češtinu umí. Syntéza z textu je nejkomplikovanější úlohou počítačové syntézy řeči, protože na vstupu systému je pouze text - grafémy, takže je nutná analýza textu a převod na fonémy, reprezentující odpovídající ozvučení. Proto lze obecně text-to-speech systémy rozdělit do dvou základních bloků.

Prvním z nich je zpracování přirozeného jazyka v textové podobě a druhým je vlastní syntetizátor akustického signálu. Zpracování textu se skládá z několika částí. První z nich je analýza textu, skládající se z transkripce, fonetizace a prozodie. **Transkripce** je v prvním kroku přepis znaků, číslovek nebo ASCI znaků do slov. V druhém kroku je to morfologická analýza, která dá slovům podle kontextu správný tvar (především se v tomto kroku řeší číslovky). A třetí krok je syntaktická analýza. Druhou částí zpracování textu je **fonetizace**, která má za úkol ze sekvence grafémů vytvořit sekvenci fonémů, což znamená přepsat text z podoby psané do podoby, jak se čte. Využívá se zde fonetické abecedy. Třetím krokem analýzy je **prozodie**, která určuje podle kontextu a ostatních parametrů vyznění věty. Rozumí se tím, jestli věta vyzní jako tázací či oznamovací, zda skonči-

la věta nebo je ve větě čárka. Nástroji k tomu jsou modifikace základní frekvence, trvání hlásek nebo hlasitosti nastavené při produkci hlásek [1], [2], [10].



Obr. 5: Schéma hlasového syntetizéru

Druhým blokem TTS je syntetizátor akustického signálu, jehož úkolem je příchozí data z analýzy textu změnit na akustický signál. Ten je vytvořen z časové posloupnosti zvuků, které odpovídají příslušným elementům zvolených při fonetizaci textu. Syntetizátor obsahuje inventář fonetických elementů, který osahuje akustické realizace fonémů a jejich varianty (ty se mění podle kontextu).

Mezi základní metody syntézy řečového signálu řadíme artikulační syntézu, formantovou syntézu a konkatenační syntézu. U artikulační syntézy se modeluje celý proces vytváření řeči. V praxi se zatím nevyužívá pro její značnou složitost. Formantová syntéza byla využívána pro TTS systémy zhruba do osmdesátých let minulého století. Funguje na principu modelování hlasového traktu pomocí formantů. Hlasivkový tón vytváří pro znělé hlásky generátor impulsů a neznělé jsou generovány z bílého šumu. Signál pak prochází soustavou pásmových propustí - rezonátorů. Výsledkem je syntetizovaná řeč.

V současných TTS systémech se často používá konkatenační syntéza, a to i přesto, že má obecně větší paměťové nároky. Ty ale nejsou pro výpočetní výkon dnešních přístrojů velkým problémem. Nepotřebujeme u tohoto typu detailní znalost procesu vytváření řeči, což znamená jednodušší a rychlejší návrh. Dále pracuje přímo s řečovým signálem, nebo jeho částmi, a proto je syntetizovaná řeč více přirozená. Funguje na principu skládání řečových jednotek, což je vžitý termín pro pojmenování stejného typu řečových zvuků (nejčastěji difon, popřípadě trifon, hláska apod.). Tyto jednotky jsou uloženy v inventáři jednotek. Ten může být buď omezený, využívající slova a věty z konkrétní, specifické oblasti, nebo může být neomezený, což znamená, že v tomto případě řetězí subslovní elementy, ze kterých lze tvořit libovolná slova. Inventář subslovních elementů je tvořen buď ručně, nebo automaticky z velkého korpusu. Řečové jednotky mohou, ale nemusejí být dále modifikovány. Pokud ne, jsou jednotlivé části pouze řetězeny za sebe, ale s modifikací lze zlepšit věrohodnost syntetizovaného hlasu. Modifikace se snaží o plynulé přechody mezi jednotlivými segmenty minimalizováním nespojitostí a je také schopna měnit prozodické vlastnosti. Řeč vzniklá konkatenační syntézou napodobuje řečníka z inventáře [2], [10], [16].

Pokud jde o způsoby řetězení, lze jmenovat několik základních typů. Prvním je prosté řetězení. Ve slovníku jsou uloženy řečové jednotky obvykle v podobě celých slov. Ta jsou za sebe jednoduše řetězena, takže zde není možnost modifikace prozodie. Kvalita hlasu z hlediska přirozenosti slov je zde vysoká, neboť jsou reprodukována celá slova. Dalším typem je řetězení s překryvem zvané OLA. Jednotkami jsou zde obvykle slabiky nebo různé subslovní elementy. Funguje na principu sčítání přesahů jednotek. Vedle metody OLA je tu metoda řetězení PSOLA, která pracuje se všemi typy řečových jednotek. Pracuje s pozicemi hlasivkových pulsů, které jsou kritické pro kvalitu výsledného řečového signálu. Pulsy se určují ze signálu jako samotné f0, nebo

méně často především pro srovnávací experimenty pomocí elektroglotografu [16]. Výhodou oproti metodě OLA je pitch synchronní překryv a řetězení, synchronizace f0 byla totiž jejím největším problémem [1], [10], [16].

V posledních letech se objevuje další přístup k syntéze řeči, a to statistická parametrická syntéza. Důvodem je řešení problémů s modifikací signálu, jež snižuje kvalitu, a také s těžkopádnými změnami hlasu, stylu apod. Tato metoda uplatňuje princip modelování vlastností řečových jednotek. K tomu jsou použity výhradně skryté Markovovy modely. Řeč je zde generována z modelu, nepracuje s řečovými jednotkami na signálové úrovni, ale jen s modely. I zde se využívají koeficienty melovského kepstra MFCC a i zde je třeba trénovací databáze. Modely jsou zde ovšem kontextově závislé. Syntéza probíhá následovně. Po obdržení posloupnosti fonémů je systém převede na posloupnosti kontextové závislých jednotek. Zřetězí se odpovídající kontextově závislé HMM a jsou určeny doby trvání stavů jednotlivých modelů. Poté jsou generovány posloupnosti vektorů spektrálních parametrů. Poslední částí je generování buzení, ze kterého je vytvořeným filtrem generován řečový signál [16].

V TTS systémech pro Android v anglickém jazyce se podle [15] užívá právě tato třetí metoda. Na stejném principu funguje i jediný lehce dostupný český syntetizér pro Android, který snese přísnější měřítka, a to hlas Iveta od společnosti SVOX, která vytvořila samotný modul Pico jako základní hlasový syntetizér pro Android. Jedná se o placený produkt, který lze stáhnout a bezplatně na třicet dní vyzkoušet z oficiálního Android Marketu. Cena této aplikace v době tvorby této práce nepřesahovala 60 Kč, což je v porovnání s ostatními syntetizéry cena velmi nízká. Ačkoli se jedná o velmi kvalitní hlasovou syntézu, není bezchybná. Problémy nastávají u cizích slov, zkratek apod., kdy je slovo přečteno špatně. Kvůli této vlastnosti bylo nutno v aplikaci vytvořit dvě varianty odpovědi programu: jednu, která bude vypsána na display, a druhou, která bude syntetizérem přečtena.

3.4 PHP

Vytvořená aplikace musí komunikovat s databází umístěnou na webovém serveru. K tomu je třeba vytvořit skript v jazyce PHP, který bude obsluhovat databázi, vyhodnocovat přijatá data a po vyhodnocení odesílat a také transformovat data reprezentující odpověď serverové části. Tato data jsou převáděna do datového formátu JSON, který je nezávislý na počítačové platformě, čímž se přesně hodí pro účely této aplikace. Je to textový, jazykově nezávislý formát, jehož vstupem je libovolná datová struktura a výstupem vždy řetězec [3].

PHP (Hypertext Preprocesor)je skriptovací programovací jazyk určený především pro tvorbu dynamického webu a webových aplikací v různých formátech jako například HTML nebo XHTML. Jeho vznik se datuje rokem 1994 a jeho autorem je Rasmus Lerdorf. Ten vytvořil jednoduchý systém počítání přístupu ke svým stránkám, jenž byl napsán v programovacím jazyce PERL a později přepsán do jazyka C. Tyto skripty byly poté vydány jako Personal Home Page Tools- PHP. Po spojení s programem Form Interpreter vznikla v roce 1995 verze PHP/FI. Ta mohla komunikovat s databázemi a obsahovala širokou implementaci pro jazyk C. Díky tomu bylo možno vytvořit první jednoduché dynamické webové aplikace. V roce 1999 bylo přepsáno jádro a vydáno nové pod názvem Zend Engine a o rok později byla vydána nová verze PHP 4 postavená právě na tomto jádře. O čtyři roky později v roce 2004 vyšla verze PHP 5 na jádře Zend Engine 2. Tato verze PHP obsahuje kromě výkonových a optimalizačních vylepšení prvky pro podporu objektově orientovaného programování. Také definuje jednoduché a konzistentní rozhraní pro napojení k databázím nebo databázovým systémům a současně jsou verze 5.x zatím posledními vydanými. Současně s vylepšováním PHP 5 se vyvíjí PHP 6, které by mělo oproti verzi pět například plně podporovat Unicode.

Podstatou PHP je, že se požadavek zpracovává na straně serveru a zpět se přenáší pouze výsledek procesu. Pokud jde o platformy, PHP skripty lze ve většině případů přenášet bez nutnosti úprav, rozdíl je pouze u několika systémově závislých funkcí. Syntaxe jazyka je inspirována řadou jiných programovacích jazyků, jako jsou Perl, C, Pascal a Java. PHP je jazyk, kterému nestačí pouze prohlížeč určité verze, jako je tomu u HTML nebo JavaScriptu, ale je nutná instalace. Základem jsou knihovny a webový server. Ten je třeba instalovat a konfiguro-

vat, obvykle jím bývá Apache nebo Microsoft IIS. PHP podporuje mnoho různých knihoven. Jsou to například knihovny pro zpracování textu a grafiky, pro práci se soubory. Důležité jsou také knihovny umožňující přístup k databázovým systémům typu MySQL, ODBC, Oracle, apod. Zaručena je také podpora mnoha internetových protokolů, jako HTTP, SMTP, SNMP, FTP, IMAP, POP3, LDAP a další. Kvůli jednoduchosti použití, univerzálnosti, podpoře množství protokolů i platforem se stal PHP nejrozšířenějším skriptovacím jazykem pro web. Je nejčastější součástí (oproti alternativám, jako je Perl nebo Python) charakteristického přístupu k tvorbě a správě webu, kterou je kombinace operačního systému Linux (nebo Windows), některého z databázových systémů (např. MySQL) a webovým serverem Apache souhrnně označovaného jako LAMP (s Windows WAMP).

3.5 SQL, MySQL a phpMyAdmin

Všechny informace, které dialogový systém potřebuje, musí být někde uloženy. V oblasti mobilních telefonů je v současnosti nejpoužívanějším relačním databázovým systémem SQLite. Z důvodu univerzálnosti aplikace je ale výhodnější potřebnou databázi příkazů a informací umístit na webový server než přímo do aplikace. Jednak se nezvyšuje velikost a náročnost aplikace, ale také se informace dají aktualizovat bez nutnosti stáhnutí si a instalace poslední verze dané aplikace. Internetové připojení je nutné kvůli rozpoznávání řeči a při komunikaci mezi aplikací a databází nejsou příliš vysoké datové toky, proto není důvod nevyužít externí databázi na webovém serveru. Pro toto použití je výhodnější využít místo SQLite, typicky používaným databázovým systémem přímo v aplikacích, databázový systém MySQL. Možností při výběru vhodného systému je samozřejmě více, ať už Firebird , Microsoft SQL Server apod., ovšem MySQL, jehož vývojářem je nyní Oracle Corporation, pro potřeby vytvářeného dialogového systému zcela vyhovoval.

S databázovými modely se setkáváme od šedesátých let minulého století. Zpočátku se používaly dva typy modelů. První byl hierarchický, jehož princip byl modelování hierarchie (podřízenosti a nadřízenosti) mezi entitami. Další byl síťový model, který byl založen na teorii grafů, kde uzly v grafu odpovídají entitám a orientované hrany definují vztahy mezi jednotlivými entitami.

Ovšem tyto modely se ukázaly jako nedostatečné, a tak byly nahrazeny tzv. relačním modelem, který je standardem dodnes. Jak už napovídá název, základním pojmem je relace, kterou si lze zjednodušeně představit jako tabulku, kde jednotlivé sloupce prezentují vlastnosti entity. Relace je tedy celá tabulka a jeden konkrétní řádek je prvek relace. Soubor těchto relací poté tvoří relační schéma, tedy databázi. Tento model byl vyvinut již v 70. letech panem E. F. Coddem, který navrhoval, aby sloupce jednotlivých tabulek databáze byly v určitém vztahu ke sloupcům jiných tabulek [13].

Relační databáze napodobuje způsob lidského myšlení, je intuitivní. Skládá do systematických celků podobné objekty a složitější rozkládá na jednodušší. Sloupce jedné tabulky jsou vázány na sloupce jiných tabulek a jejich vztahy jsou závislé na hodnotách a parametrech uložených v příslušných sloupcích. Tabulky jsou tedy v určitém vztahu - relaci.

3.5.1 Jazyk SQL

MySQL využívá strukturovaný dotazovací jazyk SQL (Structured Query Language). Jedná se o standardizovaný jazyk pro práci s daty v relačních databázích. Vývoj začal již v 70. letech 20. století, kdy firma IBM vytvořila jazyk SEQUEL, který byl později přejmenována právě na SQL. Na základě tohoto jazyka byl později přijat americkou standardizační organizací ANSI (člen organizací ISO a IEC) standard pro komunikaci s relačními databázemi pod názvem SQL-86. Aktuálním standardem je SQL3 (SQL-99), který již umí pracovat s objektovými prvky. I tak je ale omezena jeho implementace do jednotlivých typů databází, protože ne všechny standard striktně dodržují. Kód jazyka SQL není psán do samostatného programu jako u imperativních programovacích jazyků, ale patří mezi deklarativní programovací jazyky. Jako takový je jeho kód vkládán do jiného procedurálního jazyka. SQL se skládá z různých částí, z nichž jsou některé koncipovány pro tvůrce a administrátory a jiné pro programátory nebo běžné uživatele. Nejběžnější částí je jazyk DML (Data Manipulation Language), se kterým pracuje většina programátorů a uživatelů nejčastěji. Obsahuje totiž několik základních příkazů pro práci s databázovými systémy. Jsou jimi příkaz INSERT, který dokáže do tabulky vkládat jeden nebo více záznamů. Dále příkaz UPDATE (neboli Searched UPDATE), který dokáže hromadně přepisovat a měnit hodnoty sloupců v tabulce podle nadefinovaných

parametrů. DELETE (nebo také Searched DELETE) je příkaz, který maže záznamy z dané tabulky. Stejně jako u příkazu UPDATE podle stanovených podmínek. Pokud nejsou omezovací podmínky použity, vymažou se všechny záznamy v tabulce. Nejčastěji používaným příkazem je ovšem SELECT, který vrací záznamy z jedné nebo více tabulek podle zadaných kritérií [14].

Další částí jazyka SQL je jazyk DDL (Data Definition Language) obsahující příkazy pro definici dat. Pomocí tohoto jazyka se vytvářejí, upravují nebo doplňují struktury databáze, ať už se jedná o tabulky, indexy nebo schémata či katalogy. Patří sem příkazy jako CREATE, což je vytváření nových objektů, ALTER, což je jejich editace nebo DROP, který objekty maže. Pro řízení dat lze využít další vrstvu, a to DCL (Data Control Language) a její příkazy, jako např. GRANT (přiděluje oprávnění objektům) REVOKE (odnětí práv uživateli) a další. Další částí pro návrháře a správce je jazyk VDL (View Definition Language), který určuje vytváření pohledů (virtuální tabulka složena z jiných). SDL (Storage Definition Language) zase definuje způsob ukládání tabulek.

3.5.2 MySQL

MySQL je otevřený systém sloužící ke správě relačních databází podnikové úrovně spouštěný ve více vláknech. Systém byl vyvinut švédskou společností TcX v roce 1996, kdy tato firma hledala velmi rychlý a pružný databázový systém. Tehdejší trh nic podobného nenabízel, a proto se firma rozhodla pro vývoj takovéhoto systému. Nazvali jej MySQL a vycházel z jiného systému správy databází, konkrétně z mSQL. TcX se podařilo splnit požadavky na vývoj a nový systém byl rychlý, pružný a spolehlivý. K manipulaci s daty, jejich vytváření či zobrazení využívá, jak už bylo zmíněno, strukturovaný dotazovací jazyk SQL společnosti IBM s několika specifickými rozšířeními přímo pro MySQL.

MySQL není jen databáze ale systém, který databázi spravuje. Vymezuje přístupová a administrátorská práva různé úrovně, jednotlivé akce zapisuje do protokolů a je neustále spuštěn na pozadí. Dokáže regulovat připojování uživatelů nebo odesílání odpovědí na jejich požadavky. Lze jej používat na Linuxu, MS Windows, iOS i ostatních operačních systémech. Úspěch si získala právě díky univerzálnosti v oblasti operačních systémů, podpoře velkého množství programovacích jazyků (C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, Tcl), kvůli výkonu, spolehlivosti, rychlosti,

a především pak tím, že je tento systém volně šiřitelný. Dvojí licencování zajistilo MySQL velkou oblibu, neboť můžete mít jak rozšířenou placenou verzi, nebo „ořezanou“ free verzi. Oproti ostatním databázovým serverům je architektura MySQL poněkud odlišná. Je rozdělena do tří vrstev. V té první jsou obecné služby související se sítí, jako klient/server apod. Druhá vrstva už je jedinečná pro MySQL. Nachází se zde téměř všechna logika systému, stejně jako kódy pro analýzu, optimalizaci nebo rozbor. Tato vrstva využívá tzv. enginy, které jsou v pomyslné třetí vrstvě. K těm server přistupuje pomocí daného API, ve kterém jsou implementovány příkazy a postupy pro splnění požadavku na daný engine. Server přistupuje přímo k jednotlivým enginům.

Důležité vlastnosti MySQL jsou následující. MySQL je vícevláknový, což znamená, že s vytvořením nového spojení k serveru vytvoří serverový program podproces, který vykoná požadavky klienta, což vede k velké rychlosti systému. Všem klientům je při připojení vytvořen nový podproces. Vývojáři MySQL se snaží striktně dodržovat standard ANSI SQL92, takže systém tomuto standardu plně odpovídá. Další důležitou vlastností je systém nápovědy. Ke všem příkazům, které lze zadávat do příkazové řádky, je přidělena nápověda se všemi možnostmi a parametry daného příkazu. Dále je tu přenosnost mezi platformami, systém bude fungovat na jiných operačních systémech, a netřeba tedy kvůli MySQL přecházet na jiný operační systém. Další významnou vlastností je, že MySQL obsahuje množství různých rozhraní pro programování aplikací – API (Aplication Programming Interface). Lze jmenovat rozhraní pro Perl, Python, C/C++, TCL, Java apod., což znamená velkou variabilitu i v přístupových rozhraních [4].

3.5.3 PhpMyAdmin

Pro práci s MySQL databází bylo nutno vybrat nástroj, který by dovoloval rychle a jednoduše spravovat jak celou databázi, tak jednotlivé buňky a data v nich. Nejen velmi užitečným nástrojem, ale v podstatě i standardem při práci s MySQL je použití programového systému phpMyAmin, který poskytuje možnost správy databáze přes webové rozhraní. První verze byla vydána v roce 1998. Je napsaný v jazyce PHP a podporuje širokou škálu operací nejen pro MySQL, ale také pro MariaDB nebo Drizzle. Všechny často používané operace, které phpMyAdmin podporuje (správa databází, tabulek, sloupců, definice vztahů, indexace, uživatelé, oprávnění atd.), lze pro-

vést pomocí uživatelského rozhraní, nebo je tu možnost spustit libovolný SQL příkaz přímo. K jeho značné oblibě přispěl i překlad do 72 jazyků včetně češtiny. Jeho vlastnostmi jsou především intuitivní rozhraní, podpora pro většinu funkcí MySQL (prohlížet, mazat, kopírovat, zahodit, přejmenovat apod.), správa uložených procedur a triggerů, import dat z CSV a SQL, nebo naopak export do různých souborů typu CSV, SQL, XML, PDF, ISO/IEC 26300- OpenDocument Text, Word, Latex a jiných, správa více serverů a mnoho dalších významných vlastností usnadňující práci s databází MySQL. Stejně jako tento databázový systém je phpMyAdmin licencován jako GNU GPL (General Public License), takže je uživatelům volně dostupný. V tomto systému lze pracovat s databází bez větší znalosti jazyka SQL. Prostředí je velmi uživatelsky přívětivé a taktéž intuitivní [12].

3.6 Vývojové prostředí Eclipse

Otevřené licencování a zdrojový kód získaly Androidu velkou oblibu, protože zde byla možnost si Android upravit k obrazu svému. Totéž platí o aplikacích. Ty se pro Android píší v programovacím jazyce JAVA. Zjednodušením práce v tomto jazyce a zároveň základním nástrojem pro vývoj aplikací v Androidu se stala open source vývojová platforma (IDE- Integrated Development Environment) s názvem Eclipse. Jedná se o univerzální nástroj pracující na principu zásuvných modulů, které dovolují rozšířit podporu jazyků například o C++ či PHP, dále umožňují vývojové prostředí rozšířit například o návrh XML, HTML apod. Jedná se o klasické vývojové prostředí, které dokáže složit základní kostru aplikace, upozorňuje na chyby v kódu a nabízí jejich opravy. Nabízí i emulátor, jenž vytvoří virtuální zařízení na platformě Android, na kterém lze testovat funkčnost aplikace s konzolí, která vypisuje aktuální operace a případné chyby. Je to velmi užitečný nástroj při odlaďování aplikace.

4. Realizace

Prvním krokem k vytvoření aplikace byl její návrh. Ten se skládal z návrhu struktury dialogového systému, jeho částí a z výběru nejvhodnějších prostředků k realizaci těchto částí. Po dokončení návrhu bylo nutno jednotlivé prvky systému vytvořit. Jsou jimi například externí databáze, PHP skript, nebo části implementované přímo do vlastní aplikace jako hlasový syntetizér a rozpoznávač. To vše poté spojit do funkčního celku, obalit grafickým uživatelsky přívětivým prostředím, naplnit dialogový systém správnými informacemi a v posledním kroku vyzkoušet funkčnost.

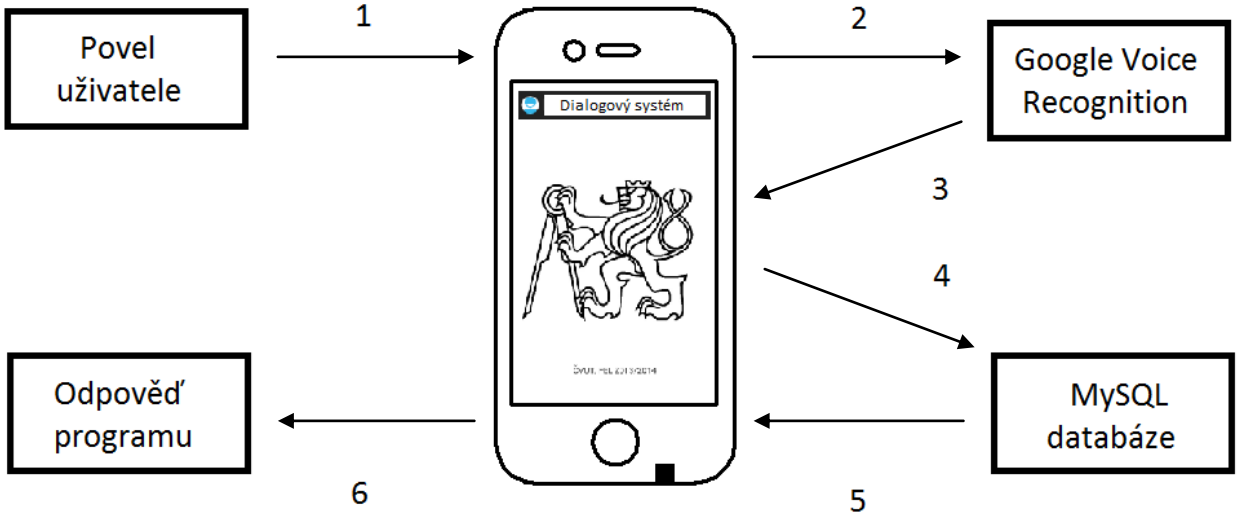
4.1 Návrh aplikace

Prvním krokem k vytvoření hlasového dialogového systému byl jeho teoretický návrh a volba operačního systému. Z důvodů, které jsem uvedl v teoretické části této práce, jsem vybral OS Android. Vlastní dialogový systém jsem se poté rozhodl založit na dialogu pomocí několika povelů uživatele, které budou jasně řídit směr dialogu. Zvolil jsem tak typ dialogového systému s konečným počtem stavů. V každém uzlu sítě si uživatel může hlasovým povelem zvolit další postup sítí. Výhodou tohoto typu dialogového systému je přesně daná struktura a poměrně jednoduchá realizace uzlů. Dále jsem hledal potřebné části systému, které lze použít na platformě Android. Nejdříve bylo nutno prostudovat možnosti, jak by aplikace mohla pracovat off-line a jaké moduly a podpůrné aplikace k tomu použít.

Hned od začátku bylo ale jasné, že pro český jazyk je myšlenka off-line aplikace na této úrovni nereálná. Neexistuje totiž žádný modul přímo pro android, který by uměl rozpoznávání řeči v češtině. Šlo by takovýto modul vytvořit, ale to je projekt, jehož náročnost značně přesahuje obsah diplomové práce. Podobné by to bylo s implementací externího rozpoznávače. V současnosti je v podstatě jedinou rozumně implementovatelnou možností voice recognition od společnosti Google (současný název je Google Voice Search). Jedná se o velmi vyspělý on-line hlasový rozpoznávač, který plně podporuje češtinu. Pro navrhovanou aplikaci má hned několik výhod. Tou hlavní je, že Google k němu poskytuje přes jeho API bezplatný přístup a v neposlední řadě je tento modul v Androidu již implementován.

Další částí systému, kterou bylo nutno vyřešit, byl způsob ukládání dat. V návrhu jsem nejprve počítal s databází umístěnou uvnitř vlastní aplikace. Ale vzhledem k nutnosti připojení k internetu (Google voice recognition) bylo praktičtější umístit databázi na web, kde by se dala velmi jednoduše spravovat, a pro aktualizace informací v ní obsažených by tedy nebylo nutno aktualizovat celou aplikaci, ale jen externí databázi.

Následujícím blokem byl návrh části, která by byla schopna převádět text na řeč- text to speech. Opět je v tomto případě Android přívětivý, protože obsahuje knihovny, které tuto funkci podporují. Ovšem zde se objevil problém, a to, že Google v tomto případě češtinu nepodporuje a žádný modul není k dispozici. Našel jsem pouze jednu využitelnou aplikaci pro android, která uměla českou syntézu řeči, a tou je placená aplikace od společnosti SVOX. K dispozici byly ještě jedna nebo dvě volně dostupné aplikace s podobnou funkcí, ovšem u nich byla kvalita syntézy v porovnání s produktem firmy SVOX velmi špatná.



Obr. 6: Schéma navrženého dialogového systému.

Finální návrh systému můžeme vidět na obr. 6. Po spuštění aplikace v Androidu zadá uživatel hlasový příkaz (1). Ten je rozpoznán pomocí Google voice recognition (2), který ze serveru vrátí text (3), ten je poté poslán do externí databáze (4), ve které je daný příkaz vyhodnocený a databáze následně pošle odpověď (5), kterou SVOX hlasový syntetizér převede na řeč a uživateli odpoví (6).

4.2 Programování pro Android

Před začátkem vlastního programování bylo nutno se rozhodnout, jaké vývojové prostředí bude pro vývoj aplikace nejlepší. Nejběžnější a zároveň oficiálně doporučovaný je program Eclipse IDE. Jedná se o vývojové prostředí pro práci s jazykem JAVA podobný NetBeans. Ovšem pro Eclipse existuje rozšíření ADT (Android Developer Tools), které je vytvořeno přímo pro vývoj aplikací určených pro Android. Tudíž jsem si zvolil právě Eclipse.

Po instalaci, při které je potřeba kromě Samotného Eclipse nainstalovat i podporu JAVA pro vývojáře JDK (JAVA DevelopmentKit) a SDK (Software Development Kit), můžeme založit nový projekt. Při vytváření nového projektu zvolíme jméno nové aplikace a hlavně minimální API level, což je minimální verze Androidu, se kterou bude aplikace plně kompatibilní. Znamená to, že na starších verzích nemusí aplikace pracovat správně. Vzhledem k zastoupení verzí systému Android jsem dal minimální API 14 (Android 4.0 Ice Cream Sandwich) a aplikace je přímo cílená na přístroje s Androidem, který má API level 18 (Android 4.3 Jelly Bean). Takže pokryje více než tři čtvrtiny přístrojů s Androidem. Po založení nového projektu Eclipse vytvoří celou adresářovou strukturu a soubory, jednoduše vše, co je nutné k fungování základní aplikace.

4.2.1 Activity

Jak už bylo řečeno v teoretické části této práce, aktivita je hlavní část programu, prezentující uživateli data získaná od nižších vrstev. Aplikace se obvykle skládá z více aktivit, které jsou k sobě volně vázány. Každá aktivita patří do třídy *„android.app.Activity“* a hlavní z nich je *„MainA-*

ctivity“, která se spustí jako úvodní aktivita při startu samotné aplikace. Každá aktivita může začít další aktivitu. Ta výchozí se zastaví, ovšem systém ji zanechá v zásobníku, který je typu LIFO (last in – first out), takže po kliknutí na tlačítko zpět je možno v této aktivitě pokračovat. V každé vytvořené aktivitě je povinná metoda *„public void onCreate()“*, která je zavolána ve chvíli, kdy se startuje nová aktivita. Součástí této metody je *„setContentView(R.layout.main)“*, což je volání námi navrženého grafického prostředí dané aktivity ze složky layout.

Úvodní aktivita je v této aplikaci poměrně jednoduchá. Aby byla aplikace přehlednější, na úvodní stránce je několik možností výběru před spuštěním samotného dialogového systému. Je zde tutoriál, kde je krátký návod, jak se v dialogovém systému pohybovat a jaké příkazy systému nejlépe dávat. Dále je zde test, což je aktivita, kde uživatel vyzkouší, zda jsou všechny potřebné prvky funkční- rozpoznávač řeči i syntetizér. V této první aktivitě jsou v našem případě naprogramována jen tlačítka, která po stisknutí začnou novou aktivitu. Typický zápis pro funkci tlačítka vypadá takto:

```
public void buttonTestClicked(View button) {
      Intent intent1 = new Intent(this, TestActivity.class);
      startActivity(intent1);}
```

Spuštění nové aktivity probíhá přes takzvané intent třídy. Intent je velmi důležitý prvek při programování Android aplikací. Je to jednoduchý objekt, který umí přenášet data mezi aktivitami a může ve spojení se *„startActivity“* spouštět jiné aktivity. Rozlišujeme implicitní intent, který určuje pouze záměr, jenž na rozdíl od druhého typu tzv. explicitního intentu neříká nic o přesné realizaci. V praxi to znamená, že data v implicitním intentu projdou intent filtrem ostatních aktivit a ta aktivita, která je schopná požadavek vyřídit, se spustí. Samozřejmě jsou ošetřeny možnosti žádné shody, kdy se objeví chybová hláška, tak i více možností, kdy se objeví nabídka pro uživatele, který si vybere některou z možností. Do aktivity je vložena knihovna intentů příkazem import stejně jako ostatní balíčky. V úvodní aktivitě máme jen ty základní:

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.widget.Toast;
```

- „*Android.app.Activity*“ je knihovna, která implementuje aplikační model pro Android. Nejvýznamnější třídou, starající se o start a ukončení aplikace, ovládání kontrolních prvků, oznámení apod. je třída „*Application*“.
- „*Android.os.Bundle*“ je třída, která mapuje páry hodnota-proměnná. Vždy je tento import společně s importem pro aktivity, neboť jak metoda *„onCreate()“*, tak metoda *„onFreeze()“* berou bundle jako parametr.
- „Android.content.Intent“ implementuje funkce intent. Content implementuje tzv. content providers, které umožňují přístup k obsahu.
- Android.view obsahují třídy pro práci a ovládání grafického prostředí (GUI), konkrétně v našem případě menu a View.
- Posledním balíčkem Main Activity je třída pro tvorbu „vyskakujícího“ informačního okna v Androidu, a tím je android.widget.Toast.

Stěžejní pro celou aplikaci je implementace hlasového syntetizéru. Ten je obsažen ve třech aktivitách aplikace, nejdříve jen jako testovací modul, pro vyzkoušení funkčnosti, poté v aktivitě poskytující tutoriál a nakonec v hlavní aktivitě jako součást dialogového systému. Pro správnou funkci modulu je nutno nejdříve importovat příslušné knihovny. Pro TTS jsou to tyto dvě:

```
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
```

V definování třídy aktuální aktivity musí být implementován *OnInitListerer*. Aby totiž objekt TTS mohl být použit, musí být dokončena jeho inicializace a ke kontrole této podmínky slouží právě *onInitListener*. Po standardizované implementaci (viz. [16]) je hlasový syntetizátor v programu

spuštěn příkazem *speakOut*. Objekt si připraví text do proměnné typu string a zavolá vlastní syntetizér. Syntaxe je následující.

```
private void speakOut() {
      String text =getText(R.string.tutor).toString();
      {tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);}
     }
```

Při dokončení činnosti TTS je vhodné zavolat metodu tts.shutdown(), která uvolní kapacitu, kterou zabírá chod syntetizéru.

Hlasový rozpoznávač je součástí dvou aktivit aplikace a poskytuje možnost rozpoznání řeči a převodu řeči do textové podoby. Balíček, který je nutno naimportovat je:

```
import android.speech.RecognizerIntent;
```

Jak je vidět, rozpoznávač řeči je implementován formou intentu, což znamená, že v případě jeho zavolání se spustí nová aktivita, která provede rozpoznání řeči a vloží jeho výsledek zpět do intentu, ze kterého lze tento výsledek uložit do proměnné a dále s ním pracovat. Volání tohoto modulu vypadá v kódu takto:

```
Intent intent= new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL, "cs-CS");
intent.putExtra(RecognizerIntent.EXTRA_SPEECH_INPUT_COMPLETE_
     SILENCE_LENGTH_MILLIS, new Long(200));

try { startActivityForResult(intent, RESULT_SPEECH);
     txtText.setText("");
     }
catch (ActivityNotFoundException a) {
     Toast t = Toast.makeText(getApplicationContext(),
     "Chyba! Vase zarizeni nepodporuje Speech to Text",
      Toast.LENGTH_SHORT);
     t.show();
    }
```

Do rozpoznávače se po jeho zavolání uloží předvolba (intent.putExtra()), že výchozím jazykovým modelem má být čeština, popřípadě lze manuálně nastavit i dobu, za kterou se po řečové aktivitě vypne naslouchání. V příkladu to je 200ms. Je zde i vidět ošetření pro případ, že by se aktivita nepodařila spustit a v tom případě by se na displeji zobrazila chybová hláška s uvedeným textem. Kompletní syntaxi implementace rozpoznávače lze najít v [16].

Nejdůležitější aktivitou ve vytvořeném dialogovém systému je DialogActivity. Obsahuje totiž jak rozpoznávač a syntetizér řeči, ale také nástroje pro komunikaci s webovým rozhraním, a v této aktivitě je veden dialog s uživatelem. Do této aktivity bylo nutno pro správnou funkčnost naimportovat velké množství balíčků. Konkrétně pro komunikaci s webovým serverem, balíčky nutné k chodu hlasového syntetizéru, balíčky pro práci s datovým formátem JSON apod. Důležitými částmi kódu je spojení s databází čili vyslání požadavku na vzdálený server a přijetí odpovědi. K přenosu dat ze strany serveru slouží formát JSON, o jehož vlastnostech je napsáno výše. Hlavní část programu potom vypadá takto:

```
httpclient=new DefaultHttpClient();
Intent intent = getIntent();
message = intent.getStringExtra(SttActivity.EXTRA_MESSAGE);
current_screen = intent.getStringExtra(SttActivity.EXTRA_MESSAGE2);
command=message;
displayCommand=message;

if (command==null){
      command="start";
      displayCommand="start";
      current_screen="start";
      }
      else{
            command=command.replace(" ","");
      }
```

Tato část přijímá data z rozpoznávače a ukládá je do daných proměnných. Tato data reprezentují povel uživatele a informaci o uzlu, ve kterém se uživatel v dialogovém systému nachází. Je zde ošetření pro první spuštění, kdy jsou povel a aktuální uzel nastaveny jako start. Pokud se

nejedná o první spuštění, je zde ošetřena možnost víceslovných příkazů tím, že jsou vymazány mezery. V důsledku toho musí být odpovídající víceslovné příkazy v databázi také uloženy bez mezer.

```
Address = ("http://noel.feld.cvut.cz/dialog-speechlab/
          ?command="+command+"&screen="+current_screen) ;

Httppost = new HttpPost(address);
ResponseHandler<String> responseHandler=new BasicResponseHandler();
final String response=httpclient.execute(httppost, responseHandler);
```

Tato část kódu posílá požadavek na server v podobě adresy, ve které je uveden příkaz – command a aktuální uzel – screen. A pomocí *ResponseHandler* je tento výsledek uložen do proměnné *response*.

```
JSONParser parser = new JSONParser();

try {
     Object obj = parser.parse(text);
     JSONObject jsonObject = (JSONObject) obj;
     result_write = (String) jsonObject.get("result_write");
     result_say = (String) jsonObject.get("result_say");
     target_screen_temp = (String) jsonObject.get("target_screen");

if (!target_screen_temp.equals(""))
     target_screen = target_screen_temp;
     else target_screen = current_screen;


if (!target_screen_temp.equals(""))
     current_screen = target_screen;
```

Zde už je vidět převod přijatého objektu ve formátu JSON a extrakce dat v něm uložených. Tato data jsou odpovědi databáze na dané příkazy. Je to text, který má systém vypsat na obrazovku, jiný text, který má říci pomocí syntetizéru a poté název uzlu, do kterého se uživatel zadaným povelem dostal. Také je zde ošetřen případ, kdy uživatel zadá příkaz, který není v databázi. Na

obrazovce aplikace se objeví (a je i syntetizován) univerzální chybový text uložený v databázi a uživatel je vyzván k opakování příkazu. Vzhledem k výše uvedeným podmínkám zůstává stále ve stejném uzlu, takže požadavek může opakovat vícekrát. Když systém vypíše a přečte všechny informace, aplikace čeká na stisknutí tlačítka pro další komunikaci nebo ukončení. V případě zmáčknutí tlačítka se spustí rozpoznávání řeči a začíná další posun v dialogovém systému.

Celý cyklus tedy vypadá takto: uživatel stiskne spouštěcí tlačítko a aplikace spustí rozpoznávač. Systém rozpozná příkaz, pošle ho na webový server, kde je pomocí PHP skriptu v databázi vyhledán odpovídající záznam, který je PHP skriptem zapouzdřen do formátu JSON, poslán jako odpověď do aplikace a zde je dekódován z JSONu a přijatá data jsou přečtena a vypsána.

4.2.2 Layout

Layout, jak už napovídá název, je grafické rozložení dané aktivity. Jedná se o samostatný xml soubor, který definuje formát, rozvržení textů a obrázků. Dále definuje typ, barvu a velikost písma, stejně jako pozadí apod. Eclipse nám nabízí rozhraní, kde můžeme pracovat buď přímo v xml souboru, nebo v grafickém rozhraní, kde jsou jednotlivé prvky předdefinovány, dané vlastnosti se zaškrtávají nebo doplňují v nabízeném menu a výsledek se ihned zobrazuje. Úpravy se dají dělat i v samotném náhledu, kde při výběru a pohybu myší lze měnit pozice, velikosti a přetažením myši vkládat různé objekty. Mně více vyhovuje práce přímo v kódu; toto grafické rozhraní jsem využíval při pozicování jednotlivých prvků a jako náhled naprogramovaného rozložení. Typickou syntaxi layoutu budu prezentovat na následujícím příkladu *activity_php_synteza.xml*, což je layout pro hlavní aktivitu aplikace *DialogActivity*, kde probíhá vlastní dialog mezi uživatelem a zařízením.

```
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="@drawable/lev"
>
</RelativeLayout>
```

V hlavičce zvolíme, jestli bude layout relativní nebo lineární. Dále orientaci stránky (vertical či horizontal) a také výšku - height a šířku- width. Vlastností “fill_parent“ říkáme, že se obrazovka roztáhne na maximální rozměr. V každém layoutu můžeme nastavit samozřejmě i pozadí stránky, a to jak statické, tak i dynamické. V tomto případě je pozadím statický obraz nazvaný lev uložený ve složce *drawable*.

```
    <EditText
          android:id="@+id/EditText01"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_marginLeft="40dp"
          android:layout_marginRight="40dp"
          android:gravity="center_vertical|center_horizontal"
          android:singleLine="true"
          android:text="" />
```

Objekt *EditText* prezentuje textové pole, do kterého může uživatel psát text. Stejně jako může sloužit pro zobrazení či úpravu textu. Pro jednoznačnou identifikaci je mu nutno přiřadit id. Syntaxi tohoto úkonu vidíme na prvním řádku. Dále zde máme několik vlastností pro určení polohy a velikosti textového pole. „Wrap_content“ u vlastnosti height nám určuje, že výška textového pole se bude měnit podle velikosti obsahu. Margin nám obecně udává odsazení od okraje stránky nebo jiného objektu, takže pokud máme definováno margin_Left=“40dp“, znamená to, že objekt bude odsazen 40dp od levého okraje stránky.

```
    <TextView
          android:id="@+id/tv"
          android:layout_centerHorizontal="true"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_alignParentLeft="true"
          android:layout_below="@+id/Button01"
          android:layout_marginTop="18dp"
          android:layout_marginLeft="40dp"
          android:layout_marginRight="40dp"
          android:text="" />
```

Objekt *TextView* musí mít stejně jako všechny ostatní objekty svůj identifikátor. Je to jen prosté zobrazení textu, kde se nastavují vlastnosti pro velikost okna, okraje, velikost a barva textu apod. To stejné platí pro tlačítko, jehož zápis je v podstatě totožný.

Layouty v této aplikaci jsou typické pro aplikace podobného typu, kde grafická stránka není tak důležitým prvkem. V této aplikaci především, protože dialogový systém používá jako hlavní způsob interakce hlas.

4.2.3 Jednotky DP a SP

V programování aplikací pro android se setkáváme s problémem, jakou jednotkou určovat velikosti. Každé zařízení má rozdílnou velikost displejů, kde se setkáváme s délkami uhlopříček od třípalcových základních smartphonů, přes časté rozměry oblíbených zařízení, tedy okolo čtyř až pěti palců, až k desetipalcovým tabletům. Každé z těchto zařízení má jiné rozlišení a jiné DPI (hustotu pixelů na palec). Android řeší tento problém jednotkou dp, což je density-independent pixel. Jedná se o virtuální jednotku, která je založena na fyzické velikosti displeje a je definována vztahem: 1dp = 160 px/dpi. Poměr dp a px se tedy mění v závislosti na dpi. Tato jednotka je doporučena pro zadávání velikostí v layoutech.

Jedinou výjimkou v tomto doporučení je velikost textů. Pro ty je zavedena jednotka sp-Scale-independent Pixels. Je podobná jednotce dp, ale zohledňuje také obecné nastavení velikosti fontů uživatele.

4.2.4 MySQL databáze

Jako databázi jsem zvolil jeden z nejpoužívanějších databázových systému MySQL. Pomocí standardu pro správu tohoto typu databáze, nástroje phpMyAdmin, bylo možno databázi přehledně naplnit daty, ty pak složit do logických celků a propojovat jednotlivé uzly dialogového stromu mezi sebou konkrétními příkazy.

Do této části projektu byla vložena veškerá logika. Aplikace uvnitř Androidu a PHP skript jsou neměnnými částmi aplikace, které se po odladění již nemusí dále upravovat, ale změnou v databázi lze měnit logiku a obsah. Nejdříve bylo nutno navrhnout, jak co nejpřehledněji a nej-

jednodušeji vyplnit databázi a zároveň vymyslet komunikační logiku, podle které bude aplikace pracovat a řídit dialog.

Pro pohyb v systému byly stanoveny dva parametry, které jasně určují, kde jsme a kam se v síti posouváme. Je to parametr aktuální polohy screen a příkaz command. Aplikace pošle například z hlavního MENU příkaz START na server v podobě webové adresy.
Například: http://noel.feld.cvut.cz/dialog-speechlab/?command=START&screen=MENU

PHP skript vezme proměnné *command* a *screen* a pošle tato data - aktuální uzel a příkaz - do databáze, kde hledá nejdříve záznamy s názvem MENU jakožto parametrem *screen*, které si uloží, a poté hledá, jestli je příkaz START v uzlu MENU definovaný. Pokud ano, uživatel se při tomto příkazu z aktuálního uzlu MENU posune do uzlu dalšího, a ten je s aktuálním spojen právě příkazem START. Aby byl systém uživatelsky přívětivější, je nutno, aby uživatel nemusel složitě hledat přesný příkaz. Proto jsem v návrhu počítal s možností, že jeden příkaz může být definován různými slovy nebo také větami. Například pokud by chtěl uživatel v dialogu postoupit z hlavního menu do uzlu Témata bakalářských prací, lze tak učinit povely: Bakalářská práce, bakalářka, témata bakalářských prací apod.

4.2.5 PHP skript

Základní funkcí PHP skriptu je obsloužit databázi a zapouzdřit její odpověď do formátu JSON. Ve skriptu jsou definovány proměnné potřebné pro přístup k databázi a následujícími příkazy se vytvoří spojení s MySQL serverem a po ověření uživatele s konkrétní databází.

```
mysql_connect($mysql_host,$mysql_user,$mysql_pass);
mysql_select_db($mysql_db);
```

Konkrétní příkaz databázi je uložen jako samostatná funkce, která je poté volána z hlavního programu. Příkaz je následně odeslán na MySQL server ke zpracování, syntaxe příkazů vypadá takto:

```
$sql_query = "SELECT commands, result_write, result_say, target_screen FROM tasks  WHERE screen = '".addslashes($input_screen)."'";
$query_result = mysql_query($sql_query);
```

Prohledání databáze vypadá následovně: na MySQL server je poslán příkaz, který prohledává databázi a ukládá všechny záznamy, které mají parametr s názvem aktuálního uzlu dialogového systému (screen). Do pole proměnných Array jsou všechny tyto záznamy postupně uloženy. V dalším kroku je cyklem prohledáváno pole a hledá se shoda příkazu uživatele a uložených hodnot.

Ačkoli je použitý rozpoznávač řeči kvalitní, nikdy není zajištěno dokonalé rozpoznání a zamýšlený příkaz se může lišit od výsledku rozpoznávače. Často se jedná o foneticky podobná slova, jako například správa a zpráva, kde bez kontextu rozpoznávač nedokáže přesně určit gramatický přepis. Z tohoto důvodu jsem při hledání shody příkazu a záznamů zavedl jistou toleranci, se kterou se příkaz a záznam porovnává. Skript vybere takový příkaz, který se shoduje alespoň z 80 procent. Toto číslo jsem určil experimentálně, vyšší už nemělo znatelný význam a pro nižší hodnoty naopak zkreslovalo výsledky. Například pro příkazy přihlásit a odhlásit vybralo vždy ten, který byl v poli proměnných dříve. Pokud ani v rámci tolerance neodpovídá žádný ze záznamů příkazu, skript pošle požadavek na MySQL server, aby mu odeslal záznam s názvem error, což je univerzální chybová hláška (např. „Zadaný příkaz nebyl rozpoznán.“), která je poté odeslána místo odpovědi.

Po vybrání správného příkazu si skript z pole Array vytáhne záznam, který se shoduje, a ten zakóduje do formátu JSON a vypíše ho na obrazovku. Pokud se vyskytne problém, je vytvořena chybová hláška a uložena do souboru záznam, který slouží nejen v tomto případě jako nástroj pro hledání chyb a odladění skriptu.

```
if (is_array($selected_task) && count($selected_task)>0) {
        echo json_encode($selected_task);
        error_log("JSON data: ".json_encode($selected_task)."\n", 3, "./zaznam.log");
} else {
        error_log("JSON data: ".json_encode($selected_task)."\n", 3, "./zaznam.log");
}
```

Odpověď skriptu vypsaná na obrazovce je poté přečtena aplikací pomocí funkcí pro odpověď webu a dále zpracovávána ve vlastním mobilním zařízení. Funkce skriptu tedy končí vypsáním odpovědi. Při testování odpovědí skriptu a databáze jsem pro prvotní odladění nepoužíval celou aplikaci, ale příkazy jsem vypisoval rovnou jako adresy do webového prohlížeče a kontroloval vypsané odpovědi. Jako příklad lze uvést adresu úvodní stránky systému:

http://noel.feld.cvut.cz/dialog-speechlab/?command=start&screen=start

Odpověď webu na tuto adresu je výpis výsledku zakódovaného do JSON formátu. Pro úvodní uzel dialogového systému vypadá v internetovém prohlížeči odpověď následovně:


```
{"result_write":"Menu.\nT\u00e9mata Bakal\u00e1\u0159sk\u00fdch
prac\u00ed.\nT\u00e9mata Diplomov\u00fdch prac\u00ed.\nT\u00e9mata Diser-
ta\u010dn\u00edch prac\u00ed.\nVedouc\u00ed prac\u00ed.\nAktu\u00e1ln\u00ed
projekty.","result_say":"Dobr\u00fd den.\r\nV\u00edtejte v infor-
ma\u010dn\u00edm syst\u00e9mu katedry obvod\u016f. Naleznete zde informace
o t\u00e9matech z\u00e1v\u011bre\u010dn\u00fdch prac\u00ed z oblasti zpra-
cov\u00e1n\u00ed \u0159e\u010di, jejich vedouc\u00edch nebo o ak-
tu\u00e1ln\u00edch projektech. Vyberte si z n\u00e1sleduj\u00edc\u00edch
mo\u017enost\u00ed. T\u00e9mata Bakal\u00e1\u0159sk\u00fdch prac\u00ed,
t\u00e9mata diplomov\u00fdch prac\u00ed, tem\u00e1tat diser-
ta\u010dn\u00edch prac\u00ed, vedouc\u00ed prac\u00ed nebo ak-
tu\u00e1ln\u00ed projekty. ","target_screen":"menu"}
```


Je zde vidět, jak jsou speciální znaky zakódovány, a zároveň je vidět, jak JSON zachovává pár proměnná – hodnota.

PHP skript je umístěn na fakultním webu laboratoře zpracování řeči – „SpeechLab“ na adrese http://noel.feld.cvut.cz/dialog-speechlab/ pod názvem index.php. Což je vždy první skript, který se spustí po otevření dané stránky.

4.3 Výsledná aplikace a její testy

Po naprogramování všech částí a jejich postupném odladění vznikla fungující aplikace pracující přesně podle návrhu. Všechny části se podařilo implementovat a spojit ve funkční celek. Testování databáze a PHP skriptu probíhalo v internetovém prohlížeči zadáváním adres a kontrolováním výsledků, popřípadě chybových hlášek ukládaných do vedlejšího souboru. Testování části

aplikace pracující ve vlastním zařízení nejdříve probíhala pouze ve vývojovém prostředí Eclipse, kde je možno si nakonfigurovat přes „Android Virtual Device Manager“ více virtuálních zařízení s různými parametry, jako jsou verze operačního systému Android, procesor, RAM, nebo velikost displeje, rozlišení apod.

Obr. 7: Virtuální zařízení programu Eclipse.

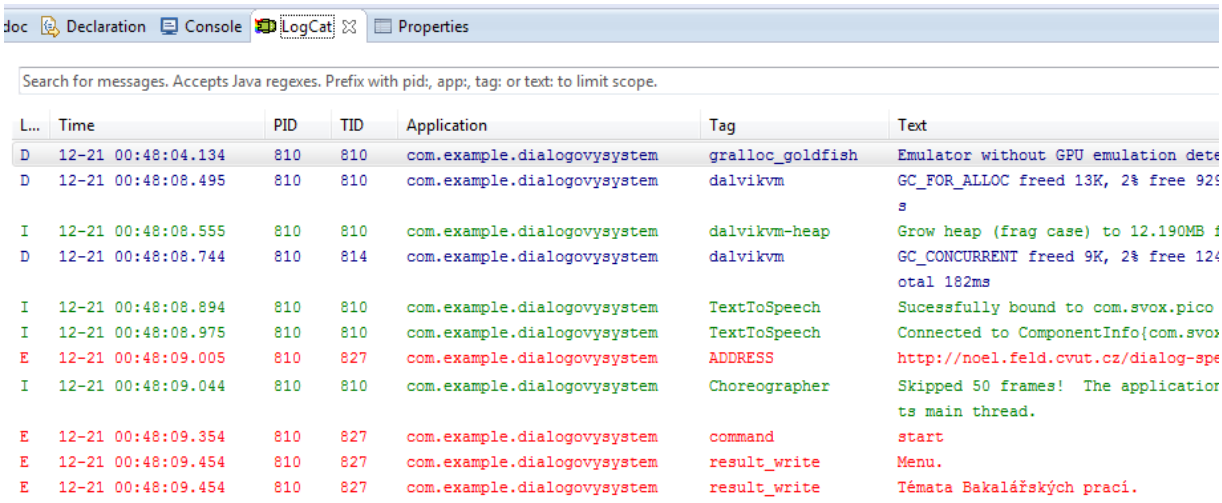
Kontrola funkčnosti zde neprobíhá jen vizuálně, ale Eclipse má dva nástroje, které pomáhají při zjišťování problémů nebo pro kontrolu funkčnosti jednotlivých prvků. Jsou jimi kontrolní konzole a LogCat. Data v konzoli jsou prezentována následovně. Mají za úkol poskytnout informace o instalaci aplikace do virtuálního zařízení a o aktuálním stavu.

```
[2014-12-21 01:19:55 - Dialogovy system] Android Launch!
[2014-12-21 01:19:55 - Dialogovy system] adb is running normally.
[2014-12-21 01:19:55 - Dialogovy system] Performing
com.example.dialogovysystem.MainActivity activity launch
```

```
[2014-12-21 01:20:00 - Dialogovy system] Uploading Dialogovy system.apk onto devi-
ce 'emulator-5554'
[2014-12-21 01:20:23 - Dialogovy system] Installing Dialogovy system.apk...
[2014-12-21 01:20:34 - Dialogovy system] Success!
[2014-12-21 01:20:34 - Dialogovy system] Starting activity
com.example.dialogovysystem.MainActivity on device emulator-5554
[2014-12-21 01:20:35 - Dialogovy system] ActivityManager: Starting: Intent
```

Obr. 8: Kontrolní konzole programu Eclipse.

LogCat zobrazuje chronologicky procesy probíhající v aplikaci. Pro testování a kontrolu je velmi užitečným příkazem Log.e("tag", proměnná), který umožňuje v LogCatu vypisovat potřebné proměnné. Takže programátor může kontrolovat hodnoty proměnných v průběhu celého procesu zpracování požadavku, jak vidíme na obr. 9.



Obr. 9: LogCat programu Eclipse.

Virtuální zařízení ovšem nepodporují syntézu řeči v češtině, a hlavně nepodporují hlasový rozpoznávač. Tudíž finální testování probíhalo na fyzických zařízeních. K dispozici jsem měl tři různá zařízení, malý, méně výkonný smartphone Samsung s uhlopříčkou displeje 3,2 palce, dále smartphone střední třídy Lenovo s uhlopříčkou 4,5 palce a posledním zařízením, na kterém byl systém testován, byl výkonný tablet Google Nexus s uhlopříčkou 7,1 palce. Všechna tato zařízení

splňovala minimální požadavek na verzi OS Android pro tuto aplikaci, a to 4.0 a vyšší. U všech přístrojů se po instalaci a nastavení telefonu podle návodu v příloze chovala podobně. Přístroj Samsung měl jak při testování, tak při opětovných instalacích problémů nepatrně více, ovšem vždy se podařilo aplikaci rozeběhnout. Po problémech s aktualizací verzí na tomto přístroji jsem na testování použil další zařízení. A to tablet nižší třídy od firmy ASUS, starší telefon HTC, a nakonec dva přístroje od Samsungu, jeden podobný výše zmiňovanému a druhý téměř nový kus. Zjistil jsem, že na ostatních přístrojích je instalace a fungování též bez problémů, ale i u druhého Samsungu nižší třídy a staršího HTC se projevily některé potíže s instalací a náhlým vypínáním aplikace. Nakonfiguroval jsem si proto ve virtuálním zařízení v Eclipse totožnou konfiguraci, jako mají problémové přístroje, a i zde probíhalo vše v pořádku. Vysvětlení těchto problémů je pravděpodobně v největší přednosti Androidu, a to v open source licencování. Velké společnosti, jako Samsung, HTC nebo ASUS, modifikují základní verze Androidu a upravují, přidávají nebo optimalizují systémové části pro své přístroje. Pravděpodobně jedna z těchto modifikací u těchto starších modelů omezuje bezproblémové fungování aplikace. U nového přístroje Samsung aplikace fungovala stejně jako na ostatních přístrojích, u starších modelů většinou stačilo starší verzi aplikace nejdříve odinstalovat a poté začít novou instalaci.

Při testování bylo vidět, že všechna zařízení potřebují kvalitní připojení k internetu, protože pokud ho nemají, při rozpoznávání řeči dojde k přerušení z důvodu nedostupnosti serveru Google pro hlasové rozpoznávání. S tímto problémem souvisí i občasné přerušení odpovědi serverové části databáze. Pro správné fungování je tedy nutný kvalitní wi-fi signál nebo kvalitní mobilní připojení. Dalším bodem pro správnou funkci aplikace je stažení českého modulu pro hlasovou syntézu a jeho správné nastavení jakožto výchozí hlasový syntetizér v obecných nastaveních Androidu. Syntéza bude na novějších přístrojích fungovat i bez toho, ale text bude předčítán podle anglické gramatiky, takže nesrozumitelně.

5. Závěr

V rámci této práce je řešena problematika programování pro nejrozšířenější mobilní platformu současnosti - operační systém Android, s užším zaměřením na možnosti hlasového vstupu a výstupu u vytvářených aplikací. Obecná část popisuje principy fungování systému Android a možnosti, které tato platforma poskytuje. Samotné vytvoření aplikace s hlasovým vstupem a výstupem zahrnuje obsáhlejší problematiku než jen prosté programování pro Android. Tato problematika je potom popsána v dalších částech obecné části práce. Stejně jako s operačním systémem je nutné pracovat i s databázovými systémy, kde pro účely této práce byla zvolena databáze typu SQL, jejíž obsluha je možná pomocí skriptu v programovacím jazyce PHP. Důležitou částí práce je i popis základních principů hlasové syntézy a rozpoznání řeči, jejichž principy je nutné znát pro následné správné používání dostupných modulů. Nakonec práce obsahuje stručný popis návrhu jednoduchého dialogového systému.

Na základě výše uvedených teoretických základů jsem navrhl a naprogramoval jednotlivé části dialogového informačního systému pro laboratoř SpeechLab. Konkrétním výsledkem mé práce je funkční aplikace pro platformu Android, která podává uživateli pomocí dialogu informace o tématech bakalářských a diplomových pracích, projektech apod. Uživatel může pomocí hlasových příkazů řídit dialog tak, aby se dostal k požadované informaci. Pokyny pro aplikaci jsou navrženy tak, aby byly pro uživatele intuitivní a i systém vlastním dialogem napovídá uživateli, který pak snadno zvolí správný pokyn. Systém dokáže pracovat jak s jednotlivými slovy, tak i pokyny ve větách, takže pravděpodobnost shody příkazu s hodnotou v databázi ve vysoká. Všechny jednoduché příkazy, stejně jako věty musí být uloženy v databázi a pro daný uzel dialogového systému definovány. Orientaci v systému zajišťují jak samotné hlasové výstupy, tak i textové výpisy a příkazy „zpět“ „hlavní menu“ nebo „opakuj“, popřípadě pomohou uživateli se zorientovat.

Po seznámení se s programováním pro platformou Android si myslím, že potenciál tohoto open-source operačního systému není zdaleka vyčerpán. Obří podpora od Google, rozsáhlá fóra a rozšířenost Androidu dávají vývojářům prostředky a možnosti k vytváření složitých a smysluplnl-

ných aplikací. Dostupnost a otevřené licencování zajišťují Androidu stálou přízeň výrobců mobilních zařízení a jeho jednoduchost, přehlednost a velký počet aplikací zase přízeň zákazníků.

Výsledná aplikace jako taková je univerzální dialogový systém pro jakékoliv využití. Tím, že obsah a příkazy jsou uloženy v databázi umístěné na serveru, je možno změnit jak obsah, tak příkazy, aby se z dialogového informačního systému pro potřeby katedry obvodů stal něčím zcela jiným. Mohl by to být navigátor, který vás provede budovou školy, informační systém, který podá informace o studijním oddělení, anebo jen aplikace, která vám oznámí dnešní menu v menze. Dialogový systém, po spojení s rozličnými funkcemi, které nabízí Google nebo řešeními na straně serveru, by mohl být použit v rámci fakulty jako plnohodnotný přihlašovací systém například na zkoušky či předmět, nebo by to mohl být asistent pro zrakově handicapované studenty.

Vytvořenou aplikaci s dialogovým systémem pro platformu Android je možno stáhnout ze stránek laboratoře zpracování řečového signálu na adrese *http://noel.feld.cvut.cz/speechlab/start.php?page=download&lang=cz*. Na stejně adrese je k dispozici návod na instalaci a správné nastavení zařízení.

Seznam literatury a zdrojů:

Literatura

[1] PSUTKA Josef, MÜLLER Luděk, MATOUŠEK Jindřich, RADOVÁ Vlasta. Mluvíme s počítačem česky. Nakladatelství Academica, Praha, 2006.

[2] UHLÍŘ Jan, SOVKA Pavel, POLLÁK Petr, HANŽL Václav, ČMEJLA Roman. Technologie hlasových komunikací. Nakladatelství ČVUT, Praha, 2007.

[3] ECMA International. The JSON Data Interchange Format. [online] [2013]. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[4] ORACLE corporation. MySQL. [online] [2014]. Dostupné z: http://www.mysql.com/

[5] David Mysliveček. Svět Androida. [online] [11.05.2013]. Dostupné z: http://www.svetandroida.cz/kratke-ohlednuti-za-historii-androidu-201305

[6] Jan Tipmann. Androidmarket.cz. [online] [03.08.2013]. Dostupné z: http://www.androidmarket.cz/android/infografika-historie-androidu/

[7] Android developers comunity. [online] [2014]. Dostupné z: Http://developer.android.com/

[8] Lagardére active. Zet.cz. [online] [28.01.2014]. Dostupné z: http://www.zet.cz/tema/svetove-prodeje-chytrych-telefonu-loni-poprve-prekrocily-hranici-miliardy-kusu-vede-samsung-1950

[9] ABI research. London. [online] [29. 01.2014]. Dostupné z: https://www.abiresearch.com/press/q4-2013-smartphone-os-results-is-google-losing-con

[10] Doc. Ing Petr Pollák. Přednášky předmětu ZRE. Praha. [2014]

[11] Sanja Primorac a Mladen Russo. IEEE- Speech recognition on Android. University of Split, Chorvatsko. [21.05.2014]. Dostupné v knihovně IEEE.

[12] PhpMyAdmin contributors. [online] [2014]. Dostupné z: http://www.phpmyadmin.net/

[13] Jaromír Skřivan. Databáze a jazyk SQL. [online] [04.08.2000]. Dostupné z: http://interval.cz/clanky/databaze-a-jazyk-sql/

[14] Neznámý autor. 602SQL Server – úplná dokumentace.[online]. Dostupné z: http://www.602.cz/602sql/wb81/napoveda/xml/html/index.html

[15] Skupina autorů. Kasetsart University, Thaisko. IEEE- TTS systém on Android. Dostupné v knihovně IEEE.

[16] Jindřich Matoušek. Přednáška: Syntéza řeči. Západočeská univerzita v Plzni. [24.04.2013]. Dostupné z: http://www.fit.vutbr.cz/study/courses/ZRE/public/pred/12_synteza_matousek/tts.pdf

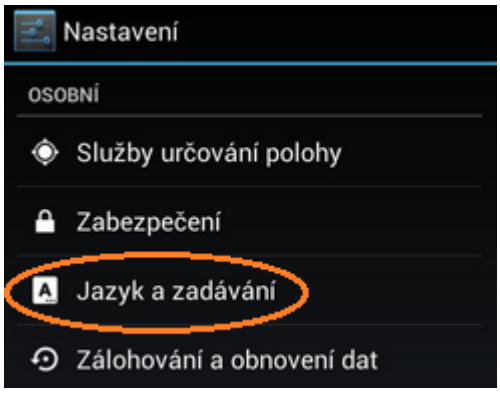
Přílohy:

Návod na instalaci aplikace.

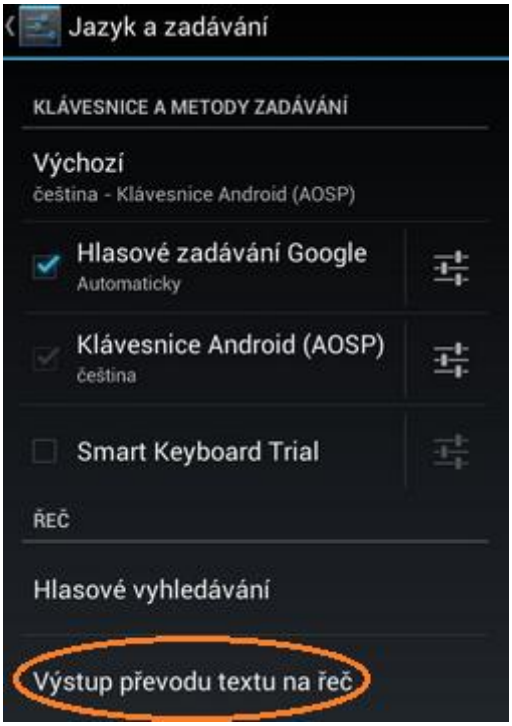
Pro správné fungování aplikace je nutno mít k dispozici instalační soubor *„Dialogovy system.apk“*, umístěný na webu laboratoře zpracování řečového signálu na ČVUT FEL s sekci download. Pro správnou funkci češtiny je nutno si stáhnout český hlas Iveta od společnosti SVOX, který je dostupný jak ve zkušební měsíční bezplatné verzi, tak i v plné verzi s cenou do 60 Kč na „Google Play Store“. Alternativou je méně kvalitní a méně spolehlivý syntetizátor E-speak.

Po stažení a instalaci hlasového syntetizéru je nutno jej v základním nastavení zařízení zvolit jako výchozí. Postup je následující:

1. Otevřít nabídku *„NASTAVENÍ“*
2. Vybrat ze seznamu *„JAZYK A ZADÁVÁNÍ“*

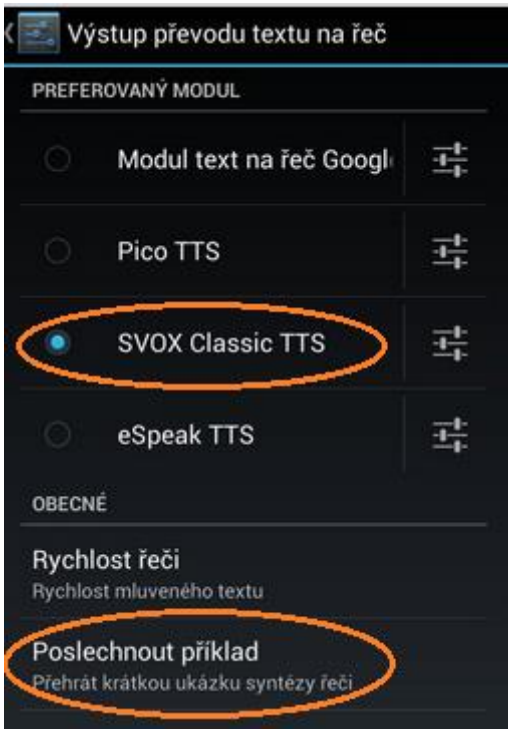


3. Z dalšího seznamu vybrat *„VÝSTUP PŘEVODU TEXTU NA ŘEČ“*



4. Pokud je SVOX správně nainstalovaný, měla by se v seznamu preferovaný modul zobrazit možnost *„SVOX Classic TTS“*. Pokud ne, postupujte podle návodu k instalaci, který je dostupný na stránkách výrobce i na Google Play.

   Pokud ano, vyberte *„SVOX Classic TTS“* jako výchozí.

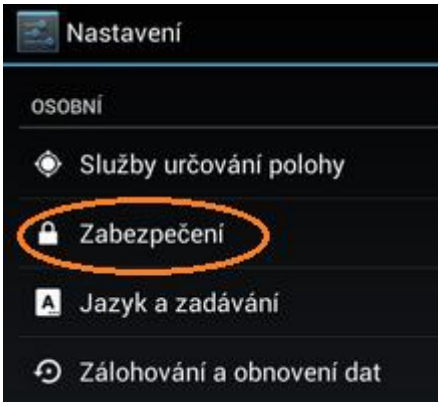


5. Zvolte možnost *„Poslechnout příklad“* a pokud uslyšíte příklad syntézy v češtině, je syntetizátor připraven k použití.

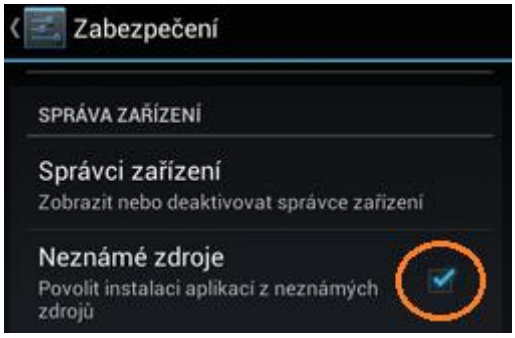
   Pokud ne, první zkontrolujte hlasitost, výstup syntetizátor bývá ztlumený na minimum. Dále zkontrolujte, jestli je stažená aplikace *„SVOX Classic TTS“* spuštěna, pokud ne, manuálně ji spusťte kliknutím na jeho ikonu v menu aplikací. Pokud i tak syntetizátor nefunguje, postupujte podle manuálu výrobce.

Když je syntetizátor nastaven, přejdeme k instalaci vlastní aplikace.

1. Prvním krokem je povolení instalace aplikací od neznámých zdrojů, které je u všech zařízeních standardně zakázáno. V nastavení telefonu tudíž zvolíme možnost *„ZABEZPEČENÍ“.*



2. Poté se zaškrtne možnost *„NEZNÁMÉ ZDROJE“,* která umožní na vaše zařízení instalovat i aplikace z jiných zdrojů, než je Google Store.



3. Poté už stačí jen spustit instalaci aplikace otevřením staženého instalačního souboru s názvem *„Dialogovy system.apk“*. Pokud je instalována novější verze, doporučuji před tímto krokem odinstalovat předchozí verzi aplikace.

   Objeví-li se v tomto kroku problém, pravděpodobně je poškozen instalační soubor. Proto jej zkuste stáhnout znovu.

   V poslední části instalace klikneme na tlačítko otevřít pro spuštění aplikace.

Po prvním spuštění aplikace se zobrazí úvodní obrazovka aplikace, kde uživatel vybere test zařízení, ve kterém se otestuje jak funkčnost rozpoznávače, tak syntetizéru. Zároveň je na úvodní obrazovce možnost přejít na stránku Informace, kde se nalézá tutoriál, který vás provede ovládáním aplikace.

Pokud by aplikace z nějakého důvodu nefungovala, zkontrolujte nejdříve dostupnost připojení k internetu. Dále ovládání hlasitosti, aby nebyl ztlumen mikrofon nebo reproduktor. Zkontrolujte si také svou verzi operačního systému Android. Tato aplikace je určena pro verze 4.0 a vyšší. Problém může být také nesprávné nastavení syntetizéru, proto zkontrolovat i tuto část. Pokud problém přetrvá, zkuste aplikaci přeinstalovat.

Naplnění databáze MySQL daty pomocí phpMyAdmin

Na následujícím obrázku je vidět struktura databáze, kde každý *task* odpovídá jednomu přechodu mezi uzly dialogového systému. Dále je zde vidět rozdíl mezi vypisovaným textem a jeho částečným fonetickým přepisem.

Vyhledávání | Vložit | Export | Import | Úpravy | Spouště

id	task	screen	commands	result_write	result_say	target_screen
1	start	start	start	Menu. Témata Bakalářských prací. Témata Diplomovýc...	Dobrý den. Vítejte v informačním systému laborato...	menu
10	bakalar	menu	TémataBakalářskýchP Bakalářsképrác bakalář ...	Témata Bakalařských prací : 1- Detektor řečo...	Témata Bakalařských prací : Téma jedna. Dete...	bakalar
11	opakovat	bakalar	TémataBakalářskýchP Bakalářsképrác bakalář ...	Témata Bakalařských prací : Detektor řečové ...	Témata Bakalařských prací : Detektor řečové ...	bakalar
12	vedouci	bakalar	vedoucíprací vedoucipráce vedoucí učitelé prof...	Vedoucí prací: Doc. Ing. Petr Pollák, CSc. I...	Vedoucí prací: Docent Inženýr Petr Pollák, kand...	vedouci
13	bakalarTema1	bakalar	jedna témajedna téma1 prvnítéma prvnitema Det...	Detektor řečové aktivity na bázi ANN. - realiza...	Detektor řečové aktivity na bázi áenen. Náplní ...	bakalar
14	bakalarTema2	bakalar	dva dvě témadva téma2 témačíslodvě druhétema ...	Grafické rozhraní pro diktovací systém na PC, vedo...	Grafické rozhraní pro diktovací systém na písí. ...	bakalar

Na obrázku níže poté vidíme editaci jednotlivých přechodů. Je zde uvedeno *id*, které společně s polem *task* usnadňují orientaci v systému. Dále je zde uveden výchozí uzel v podobě pole *screen* a také ten cílový *target_screen*. V poli nazvaném *commands* jsou vypsány všechny příkazy definované pro tento přechod. A nakonec jsou zde v polích *result_say* a *result_write* uvedeny odpovědi databáze na rozpoznaný povel.

Pole	Typ	Nulový	Hodnota
id	int(10) unsigned		13
task	varchar(255)		bakalarTema1
screen	varchar(100)		bakalar
commands	text		jedna témajedna téma1 prvnítéma prvnitema Detektorřečovéaktivity detektoraktivity detektorřeči detektorřečovéaktivitynabáziann
result_write	text		Detektor řečové aktivity na bázi ANN. - realizace detektoru řeči na bázi neuronové sítě typu MLP (nástroje TNet) - výběr vhodných příznaků na vstupu MLP (MFCC, BF, TRAP) - analýza přesnosti detekce na reálných datech se silným a variabilním šumovým pozadím Doc. Ing. Petr Pollák, CSc.
result_say	text		Detektor řečové aktivity na bázi áenen. Náplní práce je realizace detektoru řeči na bázi neuronové sítě typu eM eL Pé. Dále výběr vhodných příznaků na vstupu eM eL Pé a analýza přesnoti detekece na reálných datech. Vedoucím práce je docent polák. Nyní se můžete vrátit do hlavního menu, nebo získat informace o dalších tématech. Pro opakování možností zvolte příkaz Bakalářské práce, nebo rovnou název či číslo práce.
target_screen	varchar(100)		bakalar

Proveď