Haaga-Helia
ammattikorkeakoulu Oy

Faris – englantia lausumaan opettava sovellus

Joonas Pitkonen

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2018

Haaga-Helia ammattikorkeakoulu Oy

Tiivistelmä

Tekijä(t)
Joonas Pitkonen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Faris – englantia lausumaan opettava sovellus

Sivu- ja liitesivumäärä
32 + 0

Opinnäytetyössä ensimmäisessä osassa käsitellään puheentunnistusta ja sen historiaa sekä rakennetta. Toisessa osassa käsitellään Android-mobiilijärjestelmälle kehitetyn puheentunnistus-sovelluksen kehitysprosessia ja sen eri vaiheita. Faris on sisäisen tietokannan omaava puhetta tulkitseva sovellus, jonka tarkoituksena on toimia apuna harjoiteltaessa englanninkielisten sanojen lausumisessa. Sovellus kehitettiin tekijän halusta tuottaa oma näkemyksensä aiemmasta ryhmätyönä toteutetusta sovelluksesta.

Työn tarkoituksena on tuottaa tutkimus puheentunnistuksesta ja sen haasteista sekä toimiva, puhetta ymmärtävä sovellus puhelimelle. Tutkimus suoritettiin kvalitatiivisin menetelmin ja sovelluksen kehityksessä käytettiin Scrumban-menetelmää sekä Android Studiota.

Opinnäytetyö itsessään jakautuu teoriaosaan sekä toiminnalliseen sovellukseen, jotka yhdessä muodostavat työn kokonaisuuden. Itse opinnäytetyön toteuttaminen suoritettiin kahdessa osassa, sovelluksen kehitys sekä toteuttaminen, jonka jälkeen teoriaosio, jossa käydään läpi puheentunnistus, sen historia ja haasteet sekä valmiin sovelluksen osat ja vaiheet.

Tämän työn toisen osan alussa käydään läpi kehitykseen liittyvät termit ja työkalut sekä niiden lyhyet kuvaukset. Tämän jälkeen siirrytään käymään läpi sovelluksen kehitysprosessin tärkeimmät vaiheet alusta loppuun ja avataan lukijalle niiden toiminnallisuutta. Jokainen kuvattu vaihe sisältää kuvia kyseisen vaiheen lopputuloksesta auttamaan lukijaa hahmottamaan miltä sovellus näyttää sitä lataamatta.

Kehitysprosessin vaiheiden kuvauksen jälkeen opinnäytetyön toisen osan teoriaosassa vertaillaan syntynyttä sovellusta kahteen markkinoilla olevaan samankaltaiseen mutta kuitenkin selkeästi erilaiseen sovellukseen. Osio on jaettu molempien sovelluksien kohdalla kuvaukseen, vertailuun ja lopputulokseen.

Lopuksi pohdinta, jossa käydään läpi syntynyttä lopputulosta ja mahdollisia jatkokehityksen kohtia. Käydään läpi tekijän omaa oppimista sekä kehityksen aikana havaittuja huomion arvoisia kohtia.

Asiasanat
Android, mobiilikehitys, ohjelmointi, SQLite, puheentunnistus

Sisällys

1 Johdanto

Opinnäytetyöni aihe valikoitui syksyllä 2017 Ohjelmistokehitys II – kurssilla ryhmätyönä toteuttamani sovelluksen innoittamana. Ryhmämme toteutti hyvin alkeellisen ja karun rautalankasovelluksen, jota oli tarkoituksena käyttää silloisen asiakkaan toiminnassa. Sovelluksen karu ulkoasu sekä yleinen alkeellisuus jäivät vaivaamaan itseäni ja tästä syystä halusin toteuttaa itsenäisesti oman näkemykseni siitä, millainen lopullisen tuotoksen olisi kuulunut olla. Aiheen valinnassa painoi myös tekijän kiinnostus perehtyä syvemmin mobiili- ja erityisesti Android-kehityksen maailmaan.

Tavoitteena opinnäytetyöllä oli kehittää toimiva englanninkielen sanoja lausumaan opettava sovellus Android-puhelimille ja julkaista se Google Play–kaupassa ilmaiseksi ladattavaksi. Aiempaa kokemusta mobiilikehityksestä tekijällä ei ole yhtä online-kurssia sekä projektikurssilla tehtyä sovellusta enempää.

Sovellus toteutettiin puhtaasti Javalla, kehitysympäristönä toimi Android Studio eikä ulkoasun kanssa käytetty ulkoisia kirjastoja vaan toteutettiin täysin Androidin omin ehdoin. Puheentunnistukseen valikoitui Googlen TTS, Text-To-Speak, moottori sen helposti ymmärrettävän dokumentaation sekä aiemmin hyväksi koetun toimivuuden perusteella. Sanojen varastoinnissa käytettiin SQLiteä.

Opinnäytetyössä käydään aluksi läpi puheentunnistuksen historiaa ja kehitystä, jonka jälkeen pohditaan puheentunnistuksen haasteita. Teoria osuuden jälkeen avataan toiminnallisessa projektissa käytettyjen työkalujen ja kielien merkityksiä sekä projektissa käytetyt menetelmät. Seuraavassa osiossa käydään läpi työvaiheet ja miten ne toteutettiin, jonka jälkeen verrataan syntynyttä lopputulosta vastaavanlaisiin jo markkinoilla oleviin sovelluksiin. Lopuksi pohdinto, jossa käydään läpi teoriaosion tuloksia sekä sovelluksen kehityksen vaiheiden onnistumisia.

1.1 Käsitteet

Opinnäytetyössä käytettävät käsitteet ja niiden kuvaukset:

Activity - Activity, näkymä, on mihin sovellus piirtää käyttöliittymänsä. Yksi näkymä on yksi sovelluksen näyttö ja yleensä sovellukset koostuva useammasta näkymästä. Jokaisella sovelluksella on päänäkymä, jonka käyttäjä näkee avatessaan sovelluksen. Näkymien ei tarvitse liittyä tiiviisti toisiinsa vaan yleensä niitä voidaan käyttää vaikkapa toisen palvelun käynnistämiseen.

Android - Android on Googlen vuonna 2005 ostaman Android Inc. yhtiön kehittämä ja Googlen vuonna 2008 julkaisema Linuxin kernelin modifioituun versioon perustuva mobiilikäyttöjärjestelmä, joka on suunniteltu pääosin kosketusnäyttöjä käyttäville laitteille. Android on ollut eniten myyty käyttöjärjestelmä älypuhelimissa vuodesta 2011 ja tableteissa vuodesta 2013 lähtien. Tätä opinnäytetyötä kirjoitettaessa tuorein versio on elokuussa 2018 julkaistu Android ”Pie”. Eri versioita androidista käytetään myös pelikonsoleissa, digitaalisissa kameroissa sekä muissa laitteissa.

Android Studio - Android Studio on virallinen Android-sovellusten kehittämistä varten tarkoitettu ohjelmistoympäristö, joka pohjautuu IntelliJ IDEAan. Android Studio tarjoaa Gradle-pohjaisen järjestelmän sekä emulaattorin kehitystyötä helpottamaan.

Fragment - Fragmentti kuvastaa osiota käyttöliittymässä, jonka taustalla on FragmentActivity. Yhdellä activitylla voi olla useita fragmentteja, joita se pystyy kierrättämään halutessaan muissa activiteissa. Sillä on oma elinkaarensa, se vastaanottaa omat syötteensä ja voidaan lisätä tai poistaa activityn ollessa käynnissä. Fragment on siis kuin ali-activity jota voi uudelleenkäyttää muissa activityissa.

Google Play – kauppa - Google Play on nykymuodossaan vuonna 2012 julkaistu digitaalinen jakelualusta, joka toimii virallisena sovelluskauppana Android-käyttöjärjestelmälle. Play sisältää ladattavaa musiikkia, pelejä, e-kirjoja, elokuvia ja ohjelmia sekä ilmaiseksi että maksua vastaan.

Markov model - Markovin malli saa nimensä venäläisen matemaatikko Andrey Markovin mukaan. Todennäköisyysteorian mukaan Markovin malli on stokastinen malli, jota käytetään mallintamaan satunnaisesti muuttuvia järjestelmiä. Siinä oletetaan tulevan tilan riippuvan vain nykyisestä tilasta, eikä aiemmin tapahtuneista tekijöistä. Markovin malli on nykyään korvannut lähes täysin edeltäjänsä Dynamic time warping – pohjaisen lähestymistavan puheentunnistuksessa.

Java - Java on Sun Microsystemsin vuonna 1995 julkaisema objektiorientoitunut ja luokkapohjainen ohjelmointikieli. Java on ollut vuodesta 2016 suosituimpien ohjelmointikielien joukossa ja sitä käyttää arviolta 9 miljoonaa kehittäjää. Javassa on käytössä vahva tyypitys, joka tarkoittaa sitä, että jokaisella muuttujalla on tyyppi ja muuttuja voivat saada ainoastaan tyyppinsä mukaisia arvoja. Se sisältää myös runsaasti eri ominaisuuksia kuten graafisen käyttöliittymäkirjaston, rinnakkaisuuden hallinnan, verkko-ominaisuudet ja rikkaat rajapinnat.

Scrumban - Scrumban on Scrumin ja Kanbanin ketterä hybridimuoto, joka antaa tiimille liikkumavaraa projektin vaatimusten muuttuessa tai vaihtuessa. Scrumban siis yhdistelee Scrumista tutun rakenteen Kanbanin venyvyyteen sekä visuaalisointiin ja onkin loistava vaihtoehto tiimeille, jotka harkitsevat siirtymistä Scrumista Kanbaniin.

SQLite - SQLite on relaatiotietokantajärjestelmä, joka on toteutettu pienenä C-kirjastona. Erona muihin tietokantoihin SQLite linkitetään sitä käyttävään sovellukseen, jolloin erillistä ODBC-yhteyttä ei tarvita.

Trello - Trello on projektinhallintatyökalu, jonka kehitti Fog Creek Software vuonna 2011. Se mahdollistaa tehtävienjaon ja projektin edistymisen seurannan tiimin jäsenten välillä.

yEd Graph Editor - yEd on javalla toteutettu diagrammien ja graafien generoimiseen tarkoitettu ohjelma toimii Windowsilla, Linuxilla, Mac OS ja kaikilla alustoilla, jotka tukevat Java Virtual Machinea. yEd tukee myös .xls- ja XML-pohjaisten tiedostojen tuontia.

2 Teoriataustaa

Tässä osiossa käydään läpi puheentunnistukseen ja mobiilioppimiseen liittyvää teoriaa. Aluksi läpikäydään puheentunnistuksen lähihistoriaa, jonka jälkeen tutustutaan Markovin piilomalliin, joka liittyy vahvasti nykymuotoiseen puheentunnistukseen. Tarkoitus ei kuitenkaan ole tuottaa tutkimusta Markovin mallista vaan käydä läpi sen liittyvyys puheentunnistukseen. Toisessa aliluvussa pohditaan puheentunnistukseen liittyviä haasteita. Tavoitteena on vastata kysymyksiin:

- Mitkä ovat puheentunnistuksen haasteet?
- Mitä hyötyä opinnäytetyössä kehitetystä sovelluksesta on puheentunnistuksen ymmärtämisen kannalta?

2.1 Puheentunnistuksen historiaa

Jotta voitaisiin ymmärtää nykyaikaista puheentunnistuksen suosiota, on hyvä myös tuntea matka, joka kuljettiin tähän pisteeseen päästäksemme. Koska kyseessä ei ole laajempi historian tutkimus, ei vuosikymmeniä ja edistysaskelia jaeta omiin alilukuihinsa vaan ne käydään läpi tässä aliluvussa. Historian läpikäyminen aloitetaan vuodesta 1952 tämän vuosiluvun merkitessä ensimmäisen ”oikean” puhetta tunnistavan laitteen maailmalle esittelyä. Kunniamaininnan ansaitsee 20-luvulla kehitetty lelu ”Radio Rex” joka hyppäsi kopistaan kuullessaan oman nimensä. Tämä tapahtui leluun kiinnitetyllä sähkömagneetilla joka 500Hz:n taajuuden ylittyessä laukaisi jousen, jolloin koira vaikutti ilmestyvän kopistaan kutsuttaessa. (Markowitz, J. 6.3.2003)

Puheentunnistus nykymuodossaan otti vauvan askeliaan 50-luvulla. Vuonna 1952 Bell Laboratories kehitti ”Audrey” nimisen laitteen, joka pystyi tunnistamaan puheesta numeroita. 1962 IBM esitteli ”Shoebox” joka ymmärsi 16 englanninkielistä sanaa (Pinola, M. 2.11.2011)

70-luvulle päästäessä kiinnostus puheentunnistusta kohtaan oli kasvussa ja se ottikin suuria askelia eteenpäin kiitos Yhdysvaltojen Puolustusministeriön rahoituksen. Tästä tuloksena syntyi ”Harpy” joka pystyi tunnistamaan jopa 1011 sanaa, joka vastaa suurin pirtein 3 vuotiaan sanavarastoa (Pinola, M. 2.11.2011). Lisäksi Bell Laboratories esitteli maailmalle järjestelmän, joka kykeni tunnistamaan useiden eri ihmisten ääniä (Kikel, C. 6.7.2018).

80-luvulle päästäessä koettiin läpimurto *hidden Markow model*, HMM, muodossa. Aihetta käsitellään myöhemmin erillisessä aliluvussa sen puheentunnistuksen kannalta tärkeän

roolin takia. Samoihin aikoihin puheentunnistusta alettiin tuomaan kaupallisessa tarkoituksessa ihmisten koteihin muun muassa Worlds of Wonder's Julie nuken muodossa. Julie pystyi oppimaan vastaamaan käyttäjän puheeseen ja jopa käymään hyvin alkeellista keskustelua käyttäjän kanssa. Puheentunnistuksessa oli kuitenkin ongelmana jokaisen lausutun sanan välillä pakollinen tauko (Pinola, M. 2.11.2011).

90-luvulle tultaessa kuluttajat saivat ensimmäisen heille suunnatun puheentunnistus tuotteen Dragon Dictate. Dragon Dictate pystyi kuuntelemaan noin 100 sanaa minuutissa eikä sanojen välissä tarvinnut pitää taukoja. Korkean hintansa vuoksi monikaan ei päässyt kokeilemaan kyseistä tuotetta. Vuonna 1993 Apple julkaisi ensimmäisen sisäänrakennetun puheentunnistus ja äänikomennot mahdollistavan ohjelmiston. Kolme vuotta myöhemmin BellSouth julkaisi VALin jonka tarkoituksena oli tunnistaa soittajan lausumat sanat ja vastata kuulemansa perusteella. VALia onkin kutsuttu kaikkien painajaismaisten äänikomennoilla toimivien valikkojen tiennäyttäjäksi. (Pinola, M. 2.11.2011)

Vuoteen 2001 mennessä koneellinen puheentunnistus ylitti 80% rajapyykin tarkkuudessa. Tässä vaiheessa puheentunnistuksen kehittymisen edistys hidastui, kunnes Google astui mukaan Google Voice Searchillaan. Kyseessä oli sovellus, joka mahdollisti puheentunnistuksen tuomisen massojen käyttöön (Sonix – A short history of speech recognition. Luettu 19.11.2018) Edellä mainitun sovelluksen pohjaa käytetään myös nykyisessä puheentunnistus moottorissa pois lukien Googlen pilvipohjainen puheentunnistus palvelu.

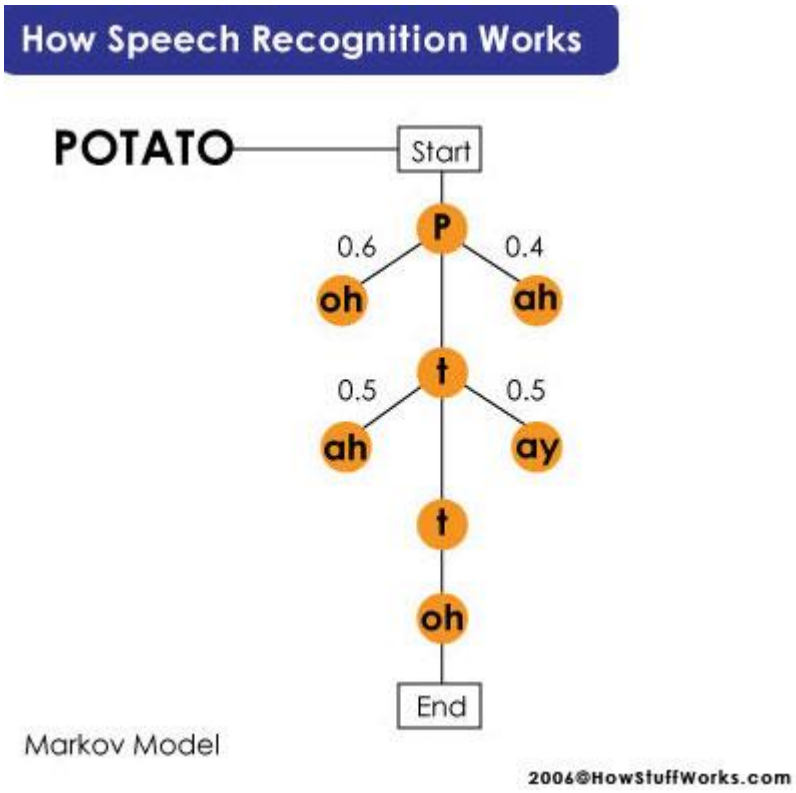
Nykyään puheentunnistusta käytetään lukuisissa sovelluksissa ja laitteissa helpottamaan ihmisten elämää. Kielten opiskelu itsenäisesti on helpottunut huomattavasti lukuisten sovellusten ansiosta, eikä käyttäjän tarvitse istua kielikurssilla, jos ei näin halua. Myös kuluttajien koteihin suunnitelluissa laitteissa puheentunnistusta on alettu hyödyntämään voimakkaasti. Applen luoma Siri on puhetta ymmärtävä virtuaaliavustaja, joka toimii Applen tuoteperheen laitteissa kuten watchOS tai HomePod ja se kykenee vastaamaan käyttäjän kysymyksiin, tekemään suosituksia sekä delegoimaan pyyntöjä muille internetin palveluille. Vastaavia virtuaaliavustajia tarjoavat myös Windowsin Cortana ja Amazonin Alexa jota voi pyytää vaikkapa tilaamaan tietyn tuotteen Amazonin sivuilta.

2.2 Markovin piilomalli

Tekstin ymmärrettävyyden vuoksi käydään aluksi läpi taustaa Markovin mallin teoriasta. Markovin ketju, Markov chain, on stokastinen prosessi, jossa tuleva tila riippuu vain edellisestä tilasta. Samalla se on Markovin mallin yksinkertaisin malli. Puheentunnistuksen kannalta kiinnostavin sekä vahvimmin aiheeseen liittyvä malli on Markovin piilomalli, hidden

Markov model, jonka ajatuksena on, että tila on vain osin havainnoitavissa. Tämä tarkoittaa, että havainnot ovat yhteydessä järjestelmän tilaan, mutta niiden riittämättömyyden takia ei tilaa voida määrittää tarkemmin.

Puheentunnistuksessa Markovin piilomallia käytetään havainnoitaessa puheen muodostamia ääniaaltoja piilotetun tilan ollessa puhuttu teksti. Tässä tapauksessa voidaan käyttää Viterbi algoritmia, joka laskee todennäköisyyden tehtyjen havainnointiketjujen perusteella. Markovin piilomalli tuottaa puhutun sanan tunnistuksessa nopeamman sekä tarkemman tuloksen Havainnollistavassa kuvassa (kuva 1) nähdään, kuinka puheentunnistus toimii Markovin mallin mukaan.



Kuva 1. Ed Grabianowski. How Speech Recognition Works. [Viitattu 19.11.2018]. Saatavissa: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/speech-recognition2.htm

Kuvassa 1 Markovin mallia havainnollistamassa on lausuttava sana potato, peruna. Puheentunnistus yrittää ennustaa Markovin mallin mukaisesti ensimmäisen kirjaimen jälkeen tulevan todennäköisen äänteen ja koostaa tällä menetelmällä jatkaen kokonaisen sanan. Kuvassa kirjaimen P jälkeen todennäköisempi haluttu äänne on ”oh” jolloin tunnistus valitsee tämän vaihtoehdon ja jatkaa kirjaimeen t toistaen tämän mallin kuten edellä on mainittu.

2.3 Puheentunnistus

Puheentunnistus terminä tarkoittaa koneen tai ohjelman kykyä tunnistaa sanoja ja lauseita käyttäjän puheesta ja muuttaa ne koneen ymmärrettävään muotoon. Puheentunnistus toimii käyttämällä akustiseen ja kielen mallinnukseen algoritmeja, esimerkiksi Viterbiä. Tässä tapauksessa akustinen malli tarkoittaa puheen ja äänisignaalin suhdetta toisiinsa; kielen mallinnus taas äänen sovittamista sana jaksoon, joka helpottaa samalta kuulostavien sanojen erottamista toisistaan (Rouse, M. 12.2018). Yksinkertaistettuna puheentunnistus on siis koneelle puhumista ja puheen tunnistetuksi tapahtumista.

Puheentunnistusta hyödyntävät järjestelmät voidaan jakaa kahteen leiriin: harjoitusta vaativat ja vaatimattomat. Järjestelmiä, jotka eivät käytä harjoittamista kutsutaan puhujasta riippumattomiksi järjestelmiksi ja harjoitusta käyttäviä puhujasta riippuvaisiksi. Harjoitusta vaativissa malleissa käyttäjä lausuu sanoja järjestelmän kuunnellessa ja analysoidessa puhujan ääntä, jolloin puhteentunnistusta hienosäädetään käyttäjän puhetyyliin sopivammaksi ja näin tunnistuksen tarkkuus paranee huomattavasti. Puhujasta riippuvaiset mallit sopivat paremmin tarkkaa tunnistusta vaativiin tapauksiin, joissa käyttäjän tulee pystyä operoimaan tietyin ennalta määritetyin sanoin järjestelmän kanssa. Riippumattomat mallit taas sopivat paremmin muun muassa pelillisiin sovelluksiin, johtuen vähemmän tarkasta tunnistuksestaan, jolloin tietyn puhemallin tunnistaminen ei ole välttämätöntä.

2.4 Puheentunnistuksen haasteet

Vaikka puheentunnistus on ottanut kehityksessä isoja askelia eteenpäin 50-luvun jälkeen, on suurin ongelma yhä ratkaisematta: kuinka karsia pois taustalla vaikuttava meteli ja kuinka tuottaa tarkka tunnistus tulos juuri halutusta äänestä. Puheentunnistusta käyttävä henkilö saattaa olla paikassa, jossa on vahva kaiku tai ympärillä paljon liikennettä. Tällöin puheentunnistuksesta tekee haastavaa ylimääräisen metelin pois suodattaminen ja ihmisääneksi luokiteltavan äänen tunnistaminen (Forsberg, M. 24.2.2003. Why is Speech Recognition Difficult?). Myös toisten lähellä olevien ihmisten puhe saattaa häiritä tunnistuksen tarkkuutta.

Toinen huomattava haaste puheentunnistukselle on elekielen puuttuminen tunnistettaessa puhetta. Joskus halutun sanoman välittämiseksi puhuja voi elehtiä käsillään ja kehollaan pitääkseen yllä kuuntelijoiden mielenkiintoa tai ilmaistakseen tunteitaan. Puheentunnistusta käyttävällä koneella tai ohjelmalla ei ole minkäänlaista mahdollisuutta tulkita tätä puhujan elehdintää tai asentoa sanoja lausuttaessa, jolloin joidenkin sanojen ja lauseiden merkitys saatetaan tulkita virheellisesti. Etenkin maissa, kuten Italiassa, joissa käsiä

käytetään tehostamaan sanojen merkitystä, menettävät puheentunnistuksessa tämän ominaisuuden tuoman arvon.

Ongelmia puheentunnistukselle tuottavat myös vahvat aksentit sekä puhe- ja kirjakielen erot lausuttaessa sanoja. Verrattaessa esimerkiksi amerikkalaisen henkilön lausumaa sanaa vahvan Walesilaisen aksentin omaavan henkilön lausumaan sanaan, huomataan kuinka sama sana voi kuulostaa täysin eri sanalta kahdesta eri aksentista johtuen. Myös tämän opinnäytetyön aliluvussa 4.2 on esimerkki sanojen lausumistavoista eri kielissä. Tämä hankaloittaa puheentunnistusta huomattavasti, ellei ohjelmaa tai konetta ole opetettu tunnistamaan juuri tiettyjä maakohtaisia lausumismalleja. Aksenttien ja maakohtaisten lausuntamallien lisäksi puhujan artikulointi ja puheen nopeus vaikuttavat tuloksen tarkkuuteen. Tämän opinnäytetyön yhteydessä toteutetun sovelluksen testauksessa huomattiin puheentunnistusmoottorilla olevan vaikeuksia erottaa tiettyjä lausunnaltaan samankaltaisia sanoja toisistaan varsinkin puheen ollessa nopeampaa tai vähemmän artikuloitua. Aksenttien ja muiden mallien lisäksi ongelmia tuottavat homonyymit, muodoltaan sama kuin jokin toinen, erimerkityksinen sana. Tällä hetkellä puheentunnistus ei ole tasolla, jolla se kykenisi erottelemaan muodoltaan samankaltaisia sanoja kovin tarkasti ilman suurempaa kontekstia ja silloinkin epävarmasti. (Forsberg, M. 24.2.2003. Why is Speech Recognition Difficult?).

Taulukko 1. Lausuttavan ja tunnistetun sanan erot

Lausuttava sana	**Tunnistettu sana**
Hire	Hair
Hire	Higher
Hire	Hare

Taulukossa 1 on havainnollistettu yhden eniten ongelmia tuottaneen sanan tunnistukseen liittyviä ongelmia. Sana hire, palkata, sai lausuttaessa lähes aina eri merkityksen sovelluksen yrittäessä tunnistaa puhujan lausumisesta oikeaa sanaa. Vaikka osan syystä voi vierittää sanojen ääntämisen samankaltaisuudelle, myös puhujan pehmeä R kirjaimen lausunta vaikutti vahvasti tunnistetun tuloksen tarkkuuteen. Rauhallinen ja jopa ylikorostettu artikulointi lisäsivät tunnistuksen tarkkuutta joidenkin sanojen kohdalla. Tämä johtuu todennäköisesti puheentunnistusmoottorin bugista, joten ohjelmoijalla tai käyttäjällä on hyvin rajallinen mahdollisuus vaikuttaa tarkkuuteen.

Tällä hetkellä puheentunnistuksen tarkkuus riippuu yhtä paljon puhujan kuin sovelluksen toiminnasta tunnistushetkellä. Jos puhuja liikkuu ulkona kovan tuulen tai liikenteen lähettyvillä, voi puheentunnistuksen tulos olla hyvinkin vääristynyt verrattuna lausuttuihin

sanoihin. Parhaassa tilanteessa taustalla ei olisi minkäänlaista ylimääräistä meteliä, mutta tämä edellyttäisi täysin hiljaista tilaa, jossa käyttää puheentunnistusta. Todellisuudessa tällaista tilannetta ei ole, ellei käyttäjä rajoita kaikkia puheentunnistusta vaativia toimiaan vaikkapa kodin rauhassa suoritettavaksi. Tällainen toiminta veisi pohjan puheentunnistuksen käyttämiseltä ihmisten arjessa esimerkiksi kielten harjoittelua varten kehitetyiltä sovelluksilta, joiden ajatuksena on, että käyttäjä voi harjoitella missä vain liikkuessaan, tai äänikomennoilla toimivilta laitteilta työpaikalla, jotka eivät välttämättä sijaitse äänettömässä tilassa.

2.5 Kehitetyn sovelluksen hyödyt tutkimuksen kannalta

Tutkimuksen lisäksi tämän opinnäytetyön rinnalla kehitettiin puheentunnistusta hyödyntävä pelillinen sovellus, jonka tarkoituksena on opettaa käyttäjää lausumaan englanninkielisiä sanoja. Jotta puheentunnistuksen toimintaa ja logiikkaa voitaisiin paremmin ymmärtää, oli sovelluksen kehittäminen oivallinen tapa tutustua puheentunnistuksen mahdollisuuksiin sekä haasteisiin. Aiemmissa aliluvuissa kuvatut mallit ja haasteet konkretisoituvat parhaiten, kun niitä todistetaan itse työskentelemällä puheentunnistuksen parissa.

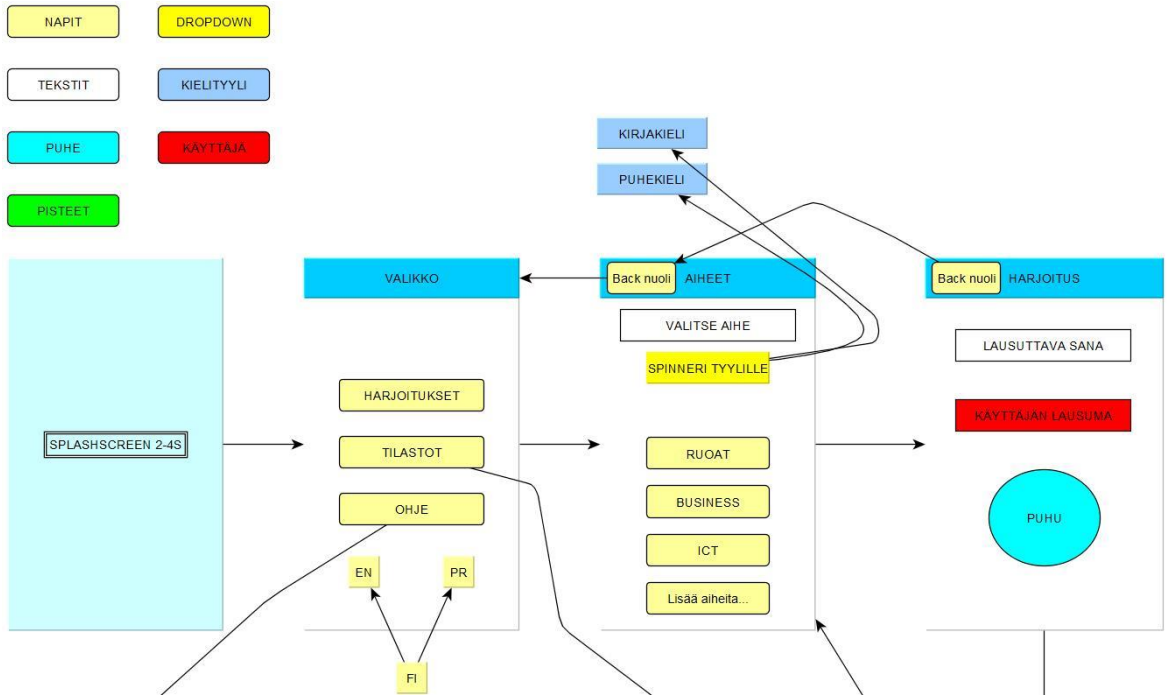
3 Puhuvan sovelluksen kehitysprosessi

Käydään aluksi sovelluksen kehitysvaiheet ja avataan niitä paremmin ymmärrettäviksi lukijalle. Koska tavoitteena oli luoda englantia lausumaan opettava sovellus, piti ensiksi valita käytettävä puheentunnistusmoottori. Alkuperäisenä ajatuksena oli käyttää Carnegie Mellon Universityn kehittämää CMU Sphinxiä mutta tämä ratkaisumalli hylättiin jo suunnittelun alkuvaiheilla dokumentaation puutteellisuuden ja hankalan ymmärrettävyyden vuoksi. Lopullisessa ratkaisussa päätettiin käyttää Googlen omaa TTS, text-to-speech, -moottoria runsaan ja selkeämmän dokumentaation sekä ennestään löytyvän kokemuksen ansiosta. Seuraava ongelma löytyi sanavarastojen muodossa. Android sekä TTS tukevat XML-tiedostosta suoraan lukemista ja rautalankamallissa tätä käytettiinkin ratkaisuna, kunnes todettiin ratkaisun olevan liian yksinkertainen eikä tekijän osaamista kuvaava. Ratkaisuksi valikoitui SQLite-tietokanta josta sovellus hakee sanat valitun aiheen mukaan ja sekoittaa ne satunnaiseen järjestykseen käyttäjän arpoessa uuden sanan. Viimeinen ja eniten päänvaivaa tuottanut vaihe oli lopullisen ulkoasun suunnittelu sekä toteuttaminen. Aiempaa kokemusta käyttöliittymän suunnittelusta on tekijällä hyvin vähän, joten heti alussa ulkoasun toteutus rajattiin toteutettavaksi täysin Androidin omilla työkaluilla helpottamaan tehtävää.

3.1 Sovelluksen rakenne

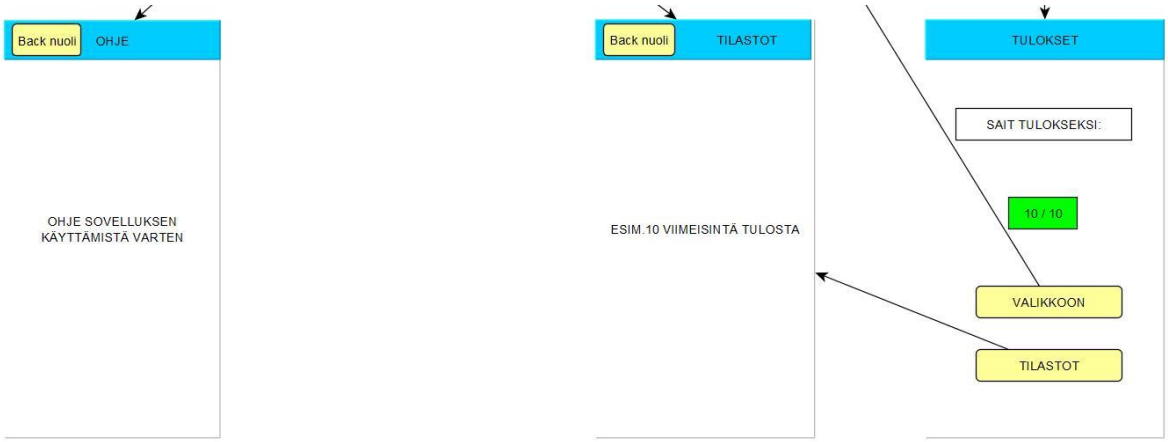
Sovelluksen rakentaminen lähti käyntiin rakentamalla rautalankamalli sovelluksen toiminnoista ja kulusta. Ilman selkeää kulkua sovellukselle olisi turha ryhtyä kehittämään minkäänlaisia toimintoja (Kuva 1). Alun perin ajatuksena oli tarjota mahdollisuus puhutun ja kirjoitetun kielen valitsemisen välillä, mutta tämä idea päätettiin jättää pois lopullisesta versiosta muiden toiminnallisuuksien vaatiessa enemmän huomiota kehityksen aikana.

Jotta sovelluksen käyttö olisi mielekästä ja pitäisi yllä käyttäjän mielenkiintoa, valikoitui toteutustavaksi pelillinen muoto jossa käyttäjä toistaa sovelluksen tarjoamia sanoja ja ansaitsee pisteitä suorituksena mukaisesti. Lopuksi sovellus näyttää käyttäjälle ansaitut pisteet ja montako sanaa meni oikein. Sovelluksella on myös mahdollista katsoa top-10 tulokset aiheittain. Sovellus ei kysy eikä listaa käyttäjän nimeä koska oletuksena on, ettei sovellusta käytä kuin puhelimen omistaja. Pisteitä ei myöskään lähetetä ulkoisiin kohteisiin, vaan ne tallentuvat SQLite-tietokantaan.



Kuva 1. Sovelluksen rautalankamalli

Kaavion koon vuoksi se on tässä jaettu kahteen osaan (Kuva 2). Tämä helpottaa muuten turhan pieneksi menneen kaavion tulkitsemista ja rajaa sovelluksen oleellisimmat toiminnallisuudet omaan kuvaansa.



Kuva 2. Sovelluksen rautalankamalli 2

Sovelluksen käyttökokemus tapahtuu loogisessa järjestyksessä:

1. Käyttäjän avatessa sovelluksen ruudulle ilmestyy SplashScreen noin kahden sekunnin ajaksi joka esittää sovelluksen logon ja toimii muutenkin antamalla sovellukselle ammattimaisemman kuvan.
2. SplashScreenin jälkeen käyttäjä on alkuvalikossa. Valikolla on 3 fragmenttina toteutettua näkymää:
    - Koti: Oletusnäkymä josta käyttäjä pääsee valitsemaan aiheen
    - Pisteet: Näkymä, josta käyttäjä näkee top 10–pisteet aiheittain

- Ohje: Näkymä josta käyttäjä näkee ohjeet yleisimpiin mahdollisiin ongelmatilanteisiin liittyen sovelluksen toimivuuteen

3. Käyttäjän siirtyessä aiheen valintaan sovellus tarjoaa RecyclerView-muotoisen listan, josta käyttäjä voi valita haluamansa aiheen.
4. Harjoituksessa sovellus tarjoaa 10 kappaletta valitun aiheen mukaisia sanoja jotka käyttäjä voi halutessaan kuunnella. Toistettaessa sanaa sovellus vertaa käyttäjän sanomaa sanaa sovelluksen valitsemaan sanaan ja joko hyväksyy tai hylkää vastauksen.
5. Käyttäjän vastattua kymmeneen sanaan sovellus ohjaa käyttäjän pistenäkymään jossa käyttäjä näkee pisteensä sekä kuinka monta sanaa sai oikein. Näkymässä on vaihtoehdot takaisin Koti-näkymään palaamiselle tai uudelleen yrittämiselle.

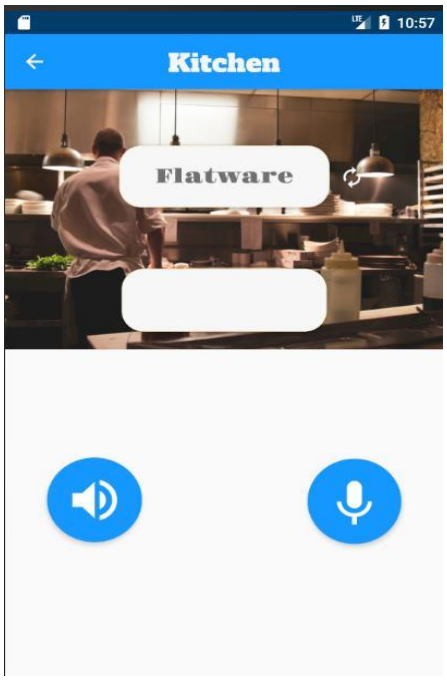
3.2 Puheentunnistuksen toteutus

Puheentunnistuksessa käytettiin Googlen omaa Text-To-Speechia, jatkossa TTS, joka tunnistaa annetun sanan ja lukee sen ääneen käyttäjälle. Sovelluksessa TTS lukee sanat tietokannasta valitun aiheen mukaisesti satunnaisessa järjestyksessä ja näyttää valitun sanan käyttäjälle.

”TTS moottorin tarvitsee tietää mitä kieltä puhua, sillä esimerkiksi sana ”Paris”, lausutaan erilailla ranskaksi ja englanniksi. Näin ollen ääni ja sanasto ovat kielikohtaisia resursseja, jotka täytyy ladata ennen kuin moottori voi aloittaa puhumisen.”

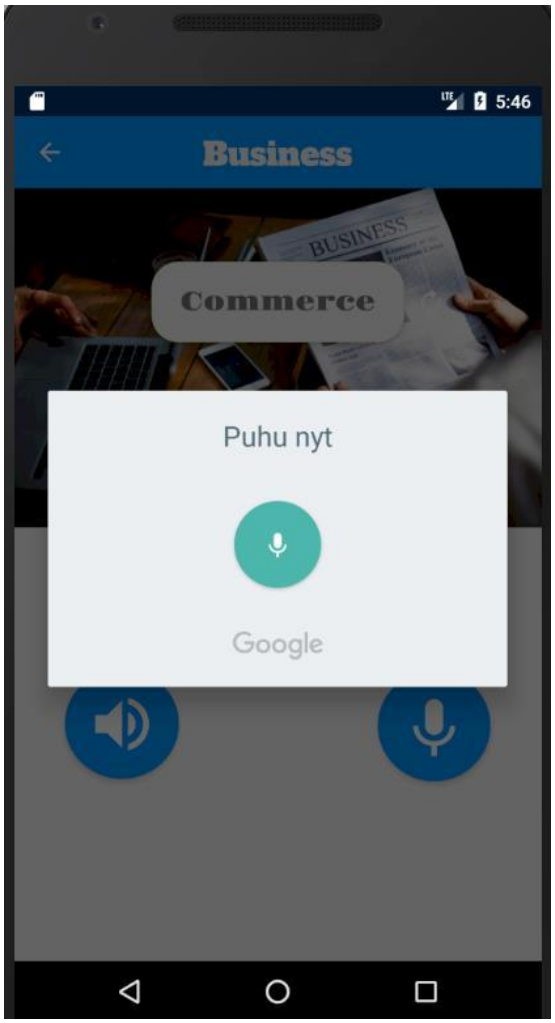
An introduction to Text-To-Speech in Android.
https://android-developers.googleblog.com/2009/09/introduction-to-text-to-speech-in.html
Luettu 19.10.2018

Aiheen valinnan jälkeen sovellus ohjaa käyttäjän harjoitus-näkymään (Kuva 2), jossa sovellus arpoo ja näyttää lausuttavan sanan käyttäjälle tämän painaessa arvonta nappia. Käyttäjä näkee ruudulla sovelluksen esittämän sanan ja voi halutessaan kuunnella sanan ennen lausumista painamalla vasemmalla olevaa ”kuuntelemista” symboloivaa painiketta. Käyttäjän ollessa valmis yrittämään sanan lausumista, painaa hän oikealla olevaa mikrofonin kuvaketta.



Kuva 2. Harjoitus-näkymä

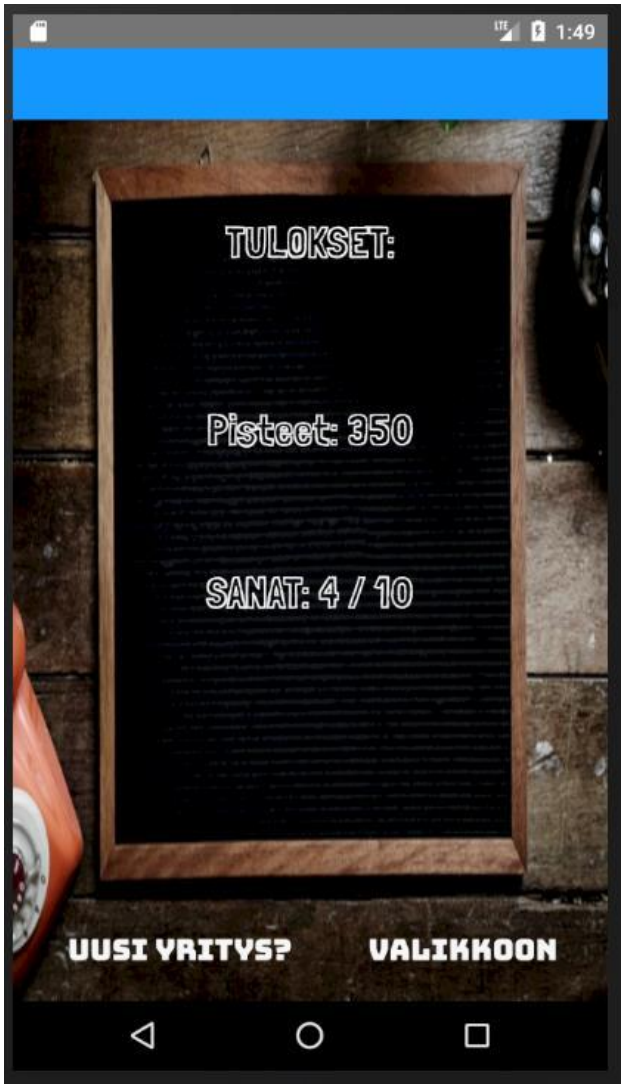
Mikrofonia painettaessa avautuu näytölle puheentunnistuksesta vastaava ikkuna (Kuva 3) joka pyytää käyttäjää puhumaan ja kuuntelee vastaanotetun puheen.



Kuva 3. Puheentunnistuksen ponnahdusikkuna

Sovelluksen saatua käyttäjältä vastaus, näytetään se arvotun sanan alapuolella vihreänä jos oikein ja punaisena jos väärin menneenä. Käyttäjä saa 100 pistettä jos vastaus on sama kuin arvottu sana ja -50 pistettä vastatessaan väärin tai puheentunnistuksen tunnistaessa vastauksen heikosti.

Sovellus laskee oikein menneiden ja arvottujen sanojen määrän kunnes lukumäärä ylittää kymmenen jonka jälkeen sovellus ohjaa käyttäjän automaattisesti pistenäkymään (Kuva 4). Pistenäkymässä käyttäjä näkee tuloksensa pisteinä ja kuinka monta sanaa sai oikein kymmenestä arvostusta sanasta.



Kuva 4. Pistenäkymä

Vaikka alkuperäisessä suunnitelmassa oli mahdollisuus antaa käyttäjälle vapaus valita puhutun ja kirjoitetun kielen lausumisen välillä, luovuttiin tästä ominaisuudesta

puheentunnistusta rakennettaessa. Moottorina käytetty Googlen TTS tunnistaa kielen sille annetun kieliasetuksen mukaisesti. Tämä tarkoittaa sitä, että eroavalla aksentilla kuin oletuksena asetettu kieli, sovellus ei joko tunnista puhetta tai arvaa väärin puhujan sanoman sanan. Myös tiettyjen sanojen, esimerkiksi ”hire” ja ”server”(palvelin), tunnistaminen on TTS:lle hankalaa niiden muistuttaessa läheisesti ”hair” ja ”server”(tarjoilija). Tunnistuksen epävarmuuden sekä ajanpuutteen takia mahdollisuus puhutun ja kirjoitetun kielen välillä valitsemiselle päätettiin karsia pois ja jättää mahdolliseksi lisäykseksi tulevaisuutta varten.

3.3 SQLite sanavarasto

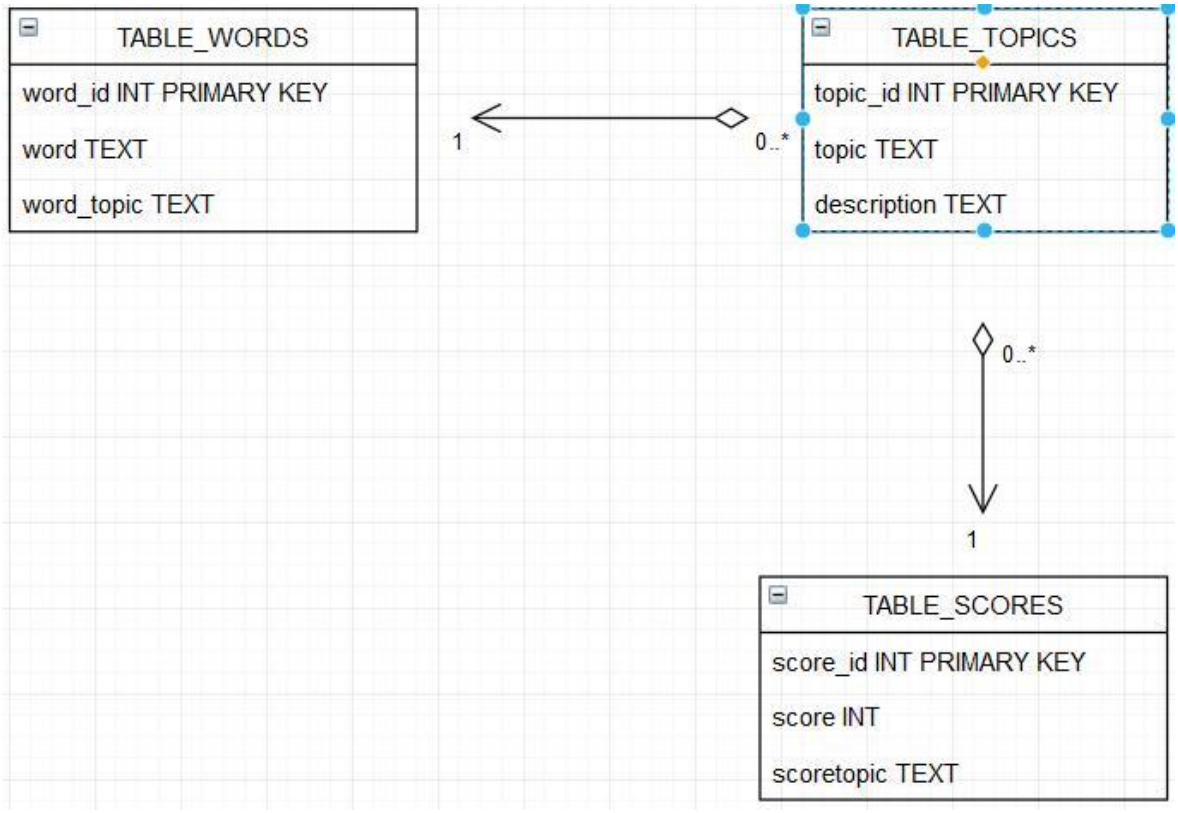
Alkuperäisessä rautalankamallissa sovellus käytti sanojen hakemiseen XML-tiedostossa olevaa sanalistaa jotta puheentunnistusta pystyi testaamaan. Lopullisessa versiossa sanat kuitenkin haetaan SQLite-tietokannasta kahdesta syystä: valmis tietokanta mahdollistaa jatkokehityksen esim. käyttäjän syötteitä varten ja koska XML-mallinen ratkaisu olisi ollut liian yksinkertainen kun tarkoituksena on osoittaa tekijän omaa osaamista.

Alla (kuva 6) on kuvattu tietokannan rakenne. Sovellus käyttää erillistä **Utils**-luokkaa määrittämään tietokannan tarvitsemat tiedot tietokannan ja taulujen luomista varten. Tämä ratkaisu helpottaa ehkäisemään kirjoitusvirheitä ja tekee tietokannan käsittelystä yksinkertaisempaa **Adapterin** puolella.

```
package Util;

public class Utils {

    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "wordsDB";
    //Tables
    public static final String TABLE_WORDS = "wordtable";
    public static final String TABLE_TOPICS = "topictable";
    public static final String TABLE_SCORES = "scoretable";
    //Word tables attributes
    public static final String KEY_WORD_ID = "word_id";
    public static final String KEY_WORDS = "word";
    public static final String KEY_WORD_TOPIC = "word_topic";
    //Topic tables attributes
    public static final String KEY_TOPIC_ID = "topic_id";
    public static final String KEY_TOPICS = "topic";
    public static final String KEY_DESCRIPTION = "description";
    //Score tables attributes
    public static final String KEY_SCORE_ID = "score_id";
    public static final String KEY_SCORE = "score";
    public static final String KEY_SCORE_TOPIC = "scoretopic";

}
```

Kuva 6. Tietokannan rakenne

Alla (kuva 7) sovelluksen tietokannan UML-kaaviosta.

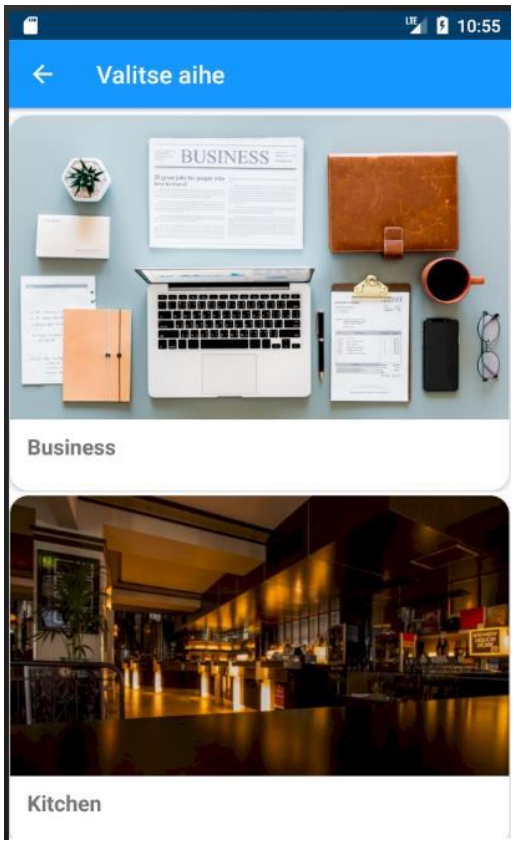


Kuva 7. Tietokannan UML-kaavio

Tietokantaa hyödynnetään kolmessa tarkoituksessa sovelluksen käytön aikana:

- Näyttämään käyttäjälle aihe vaihtoehdot. (Kuva 8)
- Hakemaan ja sekoittamaan lausuttavat sanat.
- Tallentamaan käyttäjän top-10 pisteet aiheittain. (Kuva 9)

Aihe-näkymä saa aiheen nimet tietokannasta, mutta kuvat tulevat erillisestä array-listasta. Androidin oma drawable-kansio on valmiiksi käännetty ja näin nopeuttaa kuvien piirtoa näytölle verrattuna kannasta hakemiseen. Myös aiheen kuvausteksti päätettiin korvata aiheita kuvastavilla kuvilla joista käyttäjä tunnistaa helposti mistä esitettävissä aiheissa on kyse.



Kuva 8. Aiheiden listaaminen tietokannasta

Sovellus tulostaa kannasta 10 viimeisintä tulosta ja laittaa ne järjestykseen suurimmasta pienimpään. Sovellus tallentaa vain aiheen ja pisteet sillä oletuksella että käyttäjä on aina sama eikä käyttäjälle edes tarjota mahdollisuutta tunnistautua.

Kuva 9. Pisteet tietokannasta

3.4 Adapter

Adapterin rooli sovelluksessa oli suuri ja ansaitsi tekijän mielestä läpikäynnin erillisessä osiossa. Adapter-luokkaa (Kuva 10) käytettiin tietokannan ja näkymien välillä sitomaan kannasta haetut tiedot yhteen näkymien kanssa. Sovellus käyttää kahta eri adapteria hoitaakseen kahdessa näkymässä vaadittavat tietokannan käsittelyt:

- CardAdapter aiheenvalinnan ”korttien” näyttämiseksi
- ScoreAdapter pisteiden listaamista varten

CardAdapter vastaa aiheenvalinnan tietojen hausta ja sijoittamisesta oikeille paikoilleen vastaamaan sovelluksen piirtämiä aiheeseen liittyviä kuvia. ScoreAdapter taas sijoittaa pisteet pistetaulukko näkymään käyttäjän saamat pisteet oikeassa järjestyksessä kannasta haettuna.

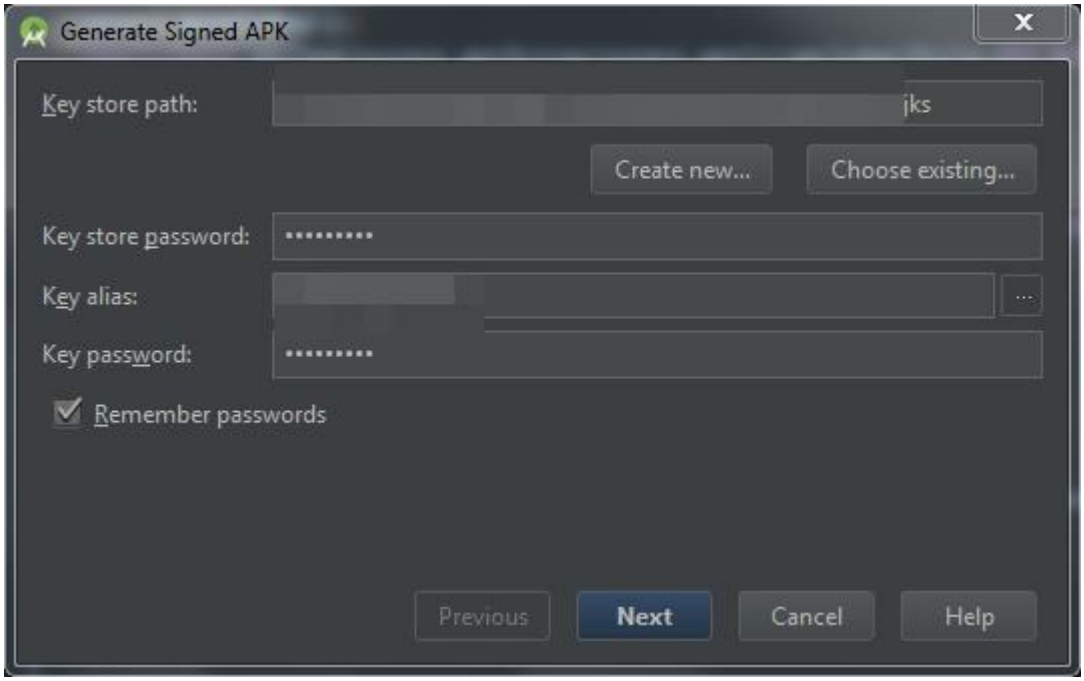
```
public class CardAdapter extends ArrayAdapter<Topic> {
    private Integer[] topicImages = {R.drawable.rawpixel_559744_unsplash, R.drawable.kitchenpic,

    private int layoutResource;
    private Activity activity;
    private ArrayList<Topic> tList = new ArrayList<>();

    public CardAdapter(Activity act, int resource, ArrayList<Topic> data) {
        super(act, resource, data);
        layoutResource = resource;
        activity = act;
        tList = data;
        notifyDataSetChanged();

    }

    @Override
    public int getCount() { return tList.size(); }

    @Override
    public Topic getItem(int position) { return tList.get(position); }

    @Override
    public int getPosition(Topic item) { return super.getPosition(item); }

    @Override
    public long getItemId(int position) { return super.getItemId(position); }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        View row = convertView;
        ViewHolder holder = null;

        if (row == null || (row.getTag() == null)) {
            LayoutInflater inflater = LayoutInflater.from(activity);
            row = inflater.inflate(layoutResource, parent,  attachToRoot: false);
            holder = new ViewHolder();

            holder.imageView = (ImageView) row.findViewById(R.id.pics);
            holder.topicText = row.findViewById(R.id.topic_Text);
            holder.descriptionText = row.findViewById(R.id.description_Text);

            row.setTag(holder);
        }else {
            holder = (ViewHolder) row.getTag();
        }

        holder.topic = getItem(position);
        holder.imageView.setImageResource(topicImages[position]);
        holder.topicText.setText(holder.topic.getTopic());
        holder.descriptionText.setText(tList.get(position).getDescription());

        final ViewHolder finalHolder = holder;
        row.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

Kuva 10. Adapter-luokka

3.5 Google Play-kauppaan julkaisu

Tässä aliluvussa käydään läpi sovelluksen Play-kauppaan julkaisun vaiheet. Valmiin sovelluksen julkaiseminen Google Play-kauppaan kuului alusta asti projektin suunniteltuihin vaiheisiin. Tätä työtä kirjoitettaessa Faris on julkaistu suljetussa beta-vaiheessa ja odottaa palautteen mukaisia korjauksia ja parannuksia. Koska Play-kauppaan julkaiseminen vaatii kehittäjätilin aktivoimisen, joka kustantaa 25$ ja on kertaluonteinen toisin kuin vaikkapa Applen lisenssi, joka on hinnaltaan 99$ ja vaatii uusimisen vuosittain.

Jotta sovellus voidaan siirtää Play-kauppaan, tarvitaan sovelluksesta APK-tiedosto. Tämä voidaan toteuttaa nopeasti ja helposti Android Studion omilla työkaluilla (Kuva 11). Kuvasta on tietoturva syistä johtuen sensuroitu herkimmät kohdat. APK-tiedoston tulee olla alle 150mt kokoinen tai Google Play Console ei hyväksy sitä. Poikkeustapauksissa isompien APK-tiedostojen lisääminen onnistuu, mutta vaatii sovelluksen pilkkomisen lisäosiksi. Tällöin sallittu koko tiedostolle nousee 2gt luokkaan (Android Developers. APK Expansion Files. Luettu: 26.11.2018). Edellä mainitussa lähteessä kerrotaan virheellisesti APK-tiedoston sallituksi kooksi 100mt, vaikka todellisuudessa koko on 150mt kuten aiemmin tässä kappaleessa on mainittu.



Kuva 11. APK-tiedoston luominen

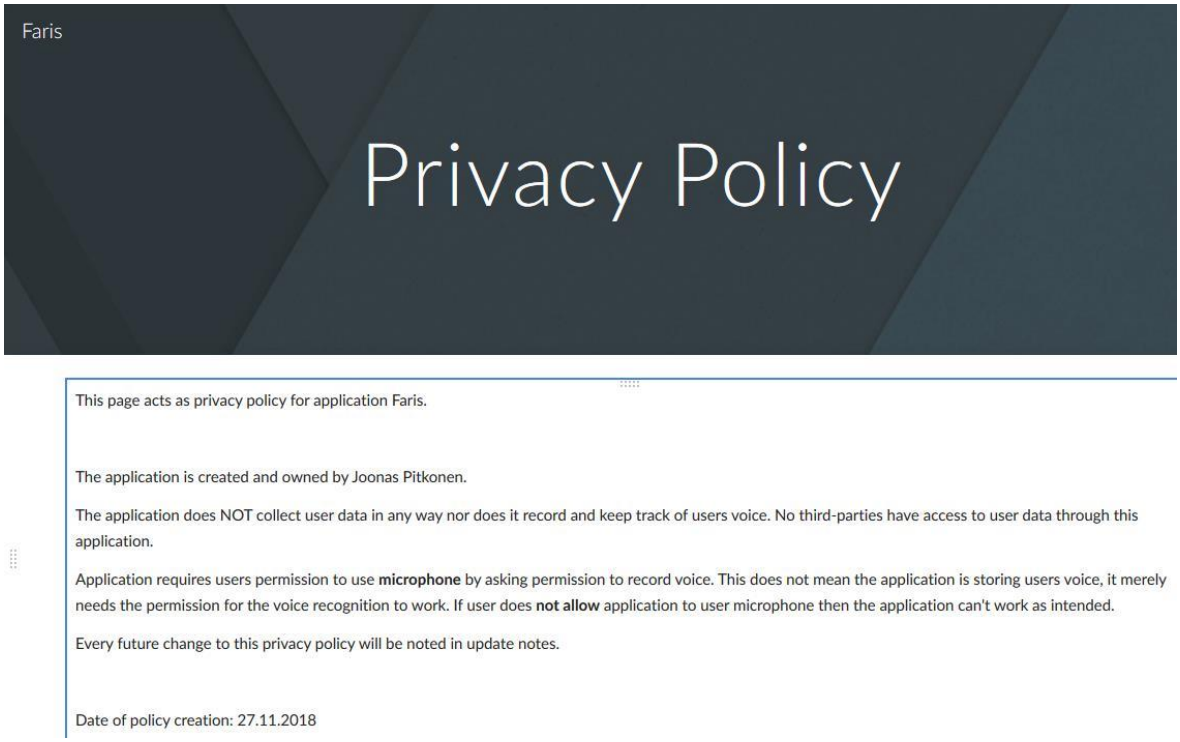
Tiedoston allekirjoitusavain kannattaa tehdä luomisen yhteydessä sillä muuten Play-kauppa ei hyväksy tiedostoa. Toinen vaihtoehto on käyttää Googlen omaa allekirjoitusta, jonka pitäisi nopeuttaa tiedoston luomista. Tämä ei kuitenkaan ole mitenkään pakollista Android Studion ollessa itsessään tarpeeksi nopea. Kun APK-tiedosto on allekirjoitettu ja

luotu, voidaan se siirtää Google Play Consoleen, jonka kautta julkaisua hallinnoidaan. Täältä voidaan valita, julkaistaanko sovellus alpha, beta vai valmis julkaisu muodossa.

Ennen tuotteen julkaisua tulee täyttää kysely, jossa määritetään sovellukselle sen luonne sekä ikärajat. Jos sovellus sisältää muun muassa väkivaltaa, kiroilua, alastomuutta tai huumeiden käyttöä, nousee sovelluksen ikärajasuositus automaattisesti. Faris luokiteltiin opetukselliseksi sovellukseksi, ei kuitenkaan peliksi, ja ikärajaksi muodostui +3v sallittu kaikille.

Sovelluksen luonteen ja rajoitusten hyväksynnän jälkeen tulee määrittää tuotteelle Play-kaupassa näkyvät kuvat ja esittelyteksti, jotta mahdolliset uudet käyttäjät ymmärtävät millainen sovellus on kyseessä. Vähintään kaksi kuvaa, taustalla näkyvä banneri ja sovelluksen Play-kaupassa näkyvä logo, vaaditaan jotta sovellus voidaan lisätä kauppaan.

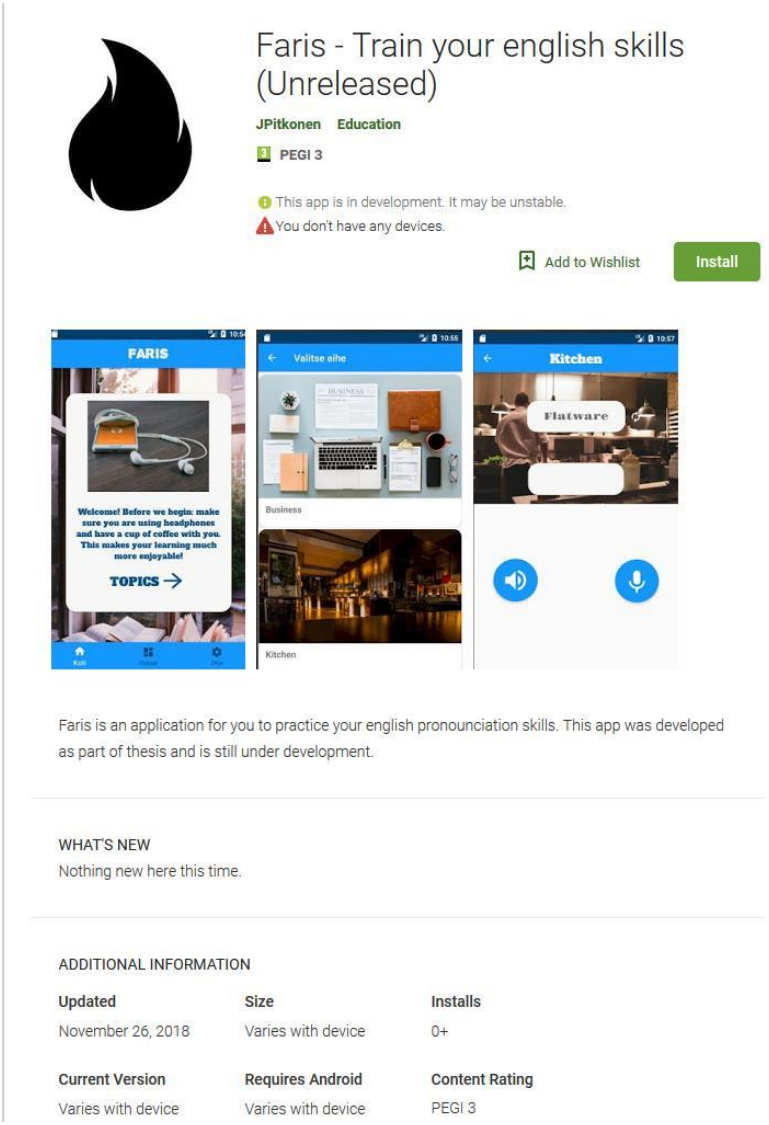
Jos lisättävä sovellus ei vaatisi erityisiä lupia käyttäjältä toimiakseen, voitaisiin sovellus julkaista ilman tietosuojakäytäntöä. Fariksen tapauksessa tietosuojakäytäntö oli lisättävä johtuen sovelluksen vaatimasta mikrofonin käyttöoikeudesta. Alla olevassa kuvassa (Kuva 12) on yksinkertainen Googlen ohjeiden mukainen tietosuojakäytäntö joka, on linkitetty sovelluksen yhteyteen.



Kuva 12. Tietosuojakäytäntö Farikselle.

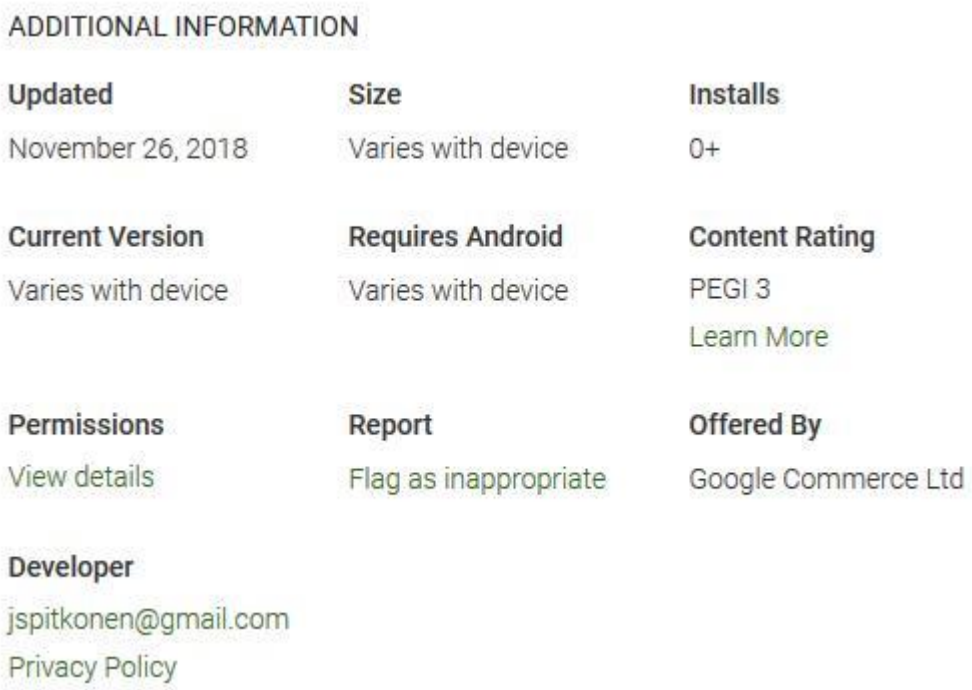
Työn ollessa vielä kehitysvaiheessa, tulee myös tietosuojakäytäntö muuttumaan tarvittaessa.

Kaikkien edellä mainittujen vaiheiden ollessa kunnossa, voidaan sovellus siirtää haluttuun vaiheeseen, oli se sitten alpha, beta tai valmis julkaisu. Jos kyseessä on suljettu alpha tai avoin beta vaihe, voidaan sovellukselle määrittää valitut testaajat tai rajoittaa heidän määräänsä. Fariksen ollessa avoimessa betassa, on testaajille lähetetty kutsu joka sisältää aktivoimislinkin ja lataus ohjeet. Gmail-tili vaaditaan jotta testaaminen onnistuu. Ennen testaamaan pääsyä joudutaan odottamaan kuitenkin hetki. Yleisesti sovelluksella kestää kolmesta viiteen tuntia ilmestyä Play-kauppaan johtuen Googlen suorittamasta hyväksymisprosessista. Alla olevassa kuvassa (Kuva 13) nähdään Faris ja sen kaikki saatavilla olevat tiedot, jotka käyttäjät Play-kaupassa näkevät sovelluksen kauppasivua tarkastellessaan.



Kuva 13. Fariksen kauppasivu Play-kaupassa

Näkymästä käyttäjät voivat tarkastella kuvakaappauksia sovelluksesta tai vaikkapa tarkastella tietosuojakäytäntöä. Tekijän yhteystiedot ovat myös näkyvillä yhteydenottoja varten (Kuva 14).



Kuva 14. Tarkempi katsaus Play-kaupan tarjoamiin tietoihin sovelluksesta.

Julkaisun ollessa vielä beta vaiheessa, ei tarkempia tietoja esimerkiksi lataajien määrästä tai sovelluksen koosta ole käyttäjien nähtävissä. Tätä työtä kirjoitettaessa testaajia on vasta kaksi ja sovelluksen koko on noin 38.47mt.

4 Faris ja samankaltaiset sovellukset

Faris ei ole eikä pyri olemaan maailmaa mullistava sovellus. Muiden puhetta tunnistavien ja lausumaan opettavien sovellusten joukossa se ei loista häikäisevällä ulkoasullaan eikä jännittävillä ominaisuuksillaan. Sovellus on kuitenkin kevyt ja nopea käyttää kun käyttäjä haluaa opetella lausumaan yksittäisiä sanoja ilman sen kummempia vaatimuksia tai säätämisiä. Vertailtaessa vastaavan kaltaisiin sovelluksiin löytyy kaikista ainakin yksi tai kaksi yhtäläisyyttä jotka sitovat sovellukset yhteen ja sitäkin enemmän ominaisuuksia jotka eivät ole samanlaisia keskenään.

Tässä osiossa vertaillaan Farista kahden samankaltaisen sovelluksen, Mondly Languagesin sekä Awaben kanssa. Lähtökohdat valintakriteereille olivat sovelluksien puhetta hyödyntävät ominaisuudet, vaihteleva rakenne tai ominaisuudet sekä ilmaisuus. Aluksi esitellään lyhyesti verrattava sovellus ja listataan samankaltaisuudet ja eroavaisuudet.

4.1 Mondly Languages

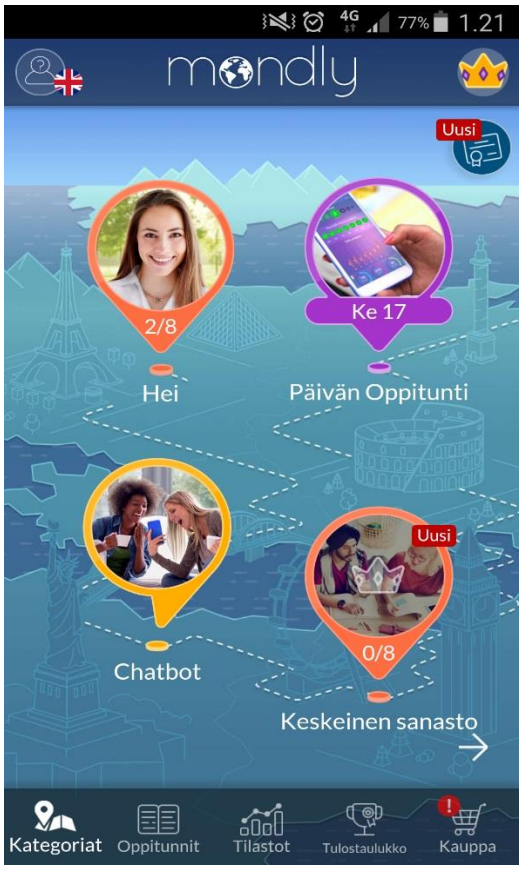
Mondly Languages (Kuva 15) on romanialaisen Mondlyn sovellus, joka tarjoaa mahdollisuuden oppia 33 eri kieltä integroimalla chatbot ja puheentunnistus teknologiaa oppimisen tueksi. Sovelluksen lataaminen on ilmaista mutta sen käyttö on tarkasti rajattu ilman vuositilausta ja sisältää sovelluksen sisäisiä ostoja. Mondlyssa on mahdollista harjoitella englanninkielen lausumista oppitunteihin jaetuissa harjoituksissa tai chatbotin kanssa keskustelemalla ja se pitää kirjaa käyttäjän edistymisestä erillisessä tilastot-näkymässä. Jokainen oppitunti koostuu vaihtelevista harjoitusmuodoista kuten sanojen sovittamista oikean käännöksen kanssa tai kuullun lauseen toistamista.

Samankaltaisuudet:

- Sanojen kuuntelu ja lausutun sanan vertailu
- Pelillisyys
- Pisteenlaskenta
- Aiheen valinta

Eroavaisuudet:

- Oppitunneiksi jaettu harjoittelu
- Lauseiden muodostus
- Tyylitelty ulkoasu
- Sovelluksen sisäiset ostot
- Voittoa tavoitteleva sovellus



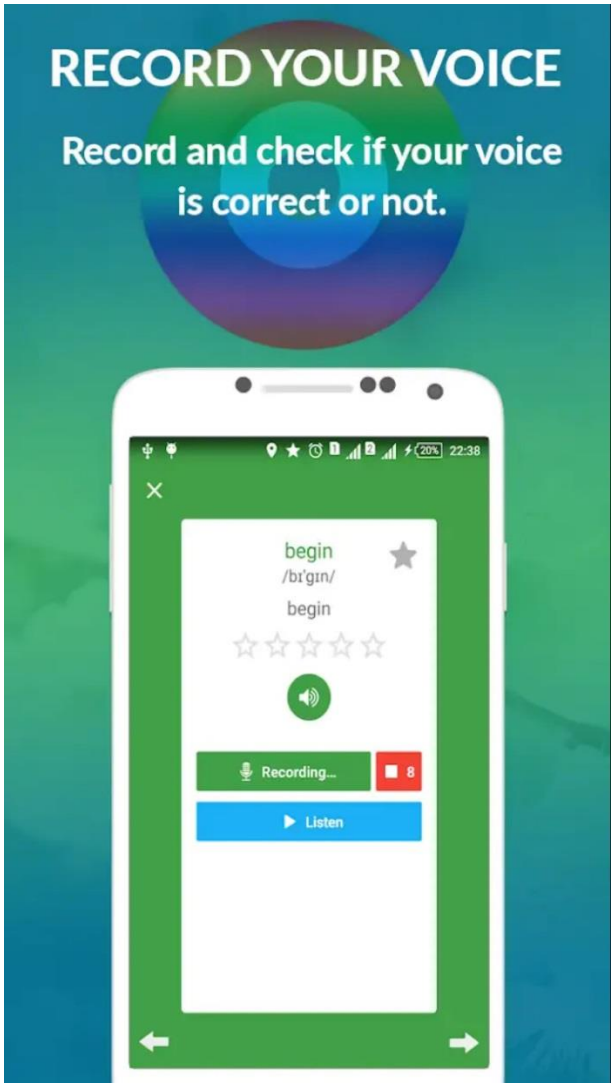
Kuva 15. Mondly Languages sovelluksen aloitusnäkymä.

Mondly on suunniteltu kaupalliseksi tuotteeksi, jonka tarkoituksena on tuoda rahaa opetusta vastaan. Ilmainen versio tarjoaa mahdollisuuden kielen opiskeluun mutta on rajoitettu eikä käytännössä anna käyttäjän harjoitella mielensä mukaan. Oppitunnit ovat pidempiä kuin Fariksen kymmenen sanan harjoitukset ja sisältävät sanojen lisäksi lauseita sekä oikean käännöksen yhdistämistä oikeaan sanaan. Verrattuna Farikseen on Mondly selvästi tarkemmin suunniteltu ja toteutettu kokonaisuus, jonka käyttöä voi varauksetta suositella uutta kieltä oppimaan haluavalle käyttäjälle sen helppokäyttöisyyden ja monipuolisuuden ansiosta. Nopeudessaan sovellukset ovat samalla viivalla ja molemmat tarjoavat käyttäjälle mahdollisuuden kuunnella sanan, jonka jälkeen käyttäjän lausumaa sanaa verrataan sovelluksen kuulemaan sanaan. Faris ja Mondly kuitenkin painivat eri sarjassa juurikin Mondlyn ammattimaisen kehityksen ja sovelluksen sisäisten osto mahdollisuuksien takia. Faris on yhden henkilön toteuttama eikä sisällä sovelluksen sisäisiä ostoja tai erillistä sisältöä vuositilauksen takana.

4.2 Awabe – English Pronunciation

Awabe - English Pronunciation (Kuva 16) on osa laajempaa Awaben englantia opettavien sovellusten sarjaa, joka keskittyy sanojen ja lauseiden opettelemisen sijaan enemmän ääntämiseen ja oikein lausumiseen. Lataaminen ja käyttäminen on ilmaista mutta sovellus

sisältää mainoksia. Kuten Fariksessa ja Mondlyssa, käyttäjä voi kuunnella ruudulla näkyvän sanan ja toistaa sen, jolloin sovellus palkitsee vastauksen asteikolla 1 – 5 tähteä. Käyttäjä voi myös kuunnella oman lausumansa sanan ja verrata sitä itse sovelluksen mallivastaukseen.



Kuva 16. Awaben harjoitusnäkymä

Samankaltaisuudet:

- Sanojen kuuntelu
- Pisteenlaskenta, tässä tapauksessa asteikko 1-5 tähteä

Eroavaisuudet:

- Oman puheen kuunteleminen
- Pisteenlaskun tähtien antaminen
- Painopiste äänteiden opettamisessa
- Mahdollisuus katsoa ääntämiseen liittyviä videoita Youtuben välityksellä

Sovellus itsessään on hyvin sekava kokonaisuus. Vaikka käyttäjällä on käytännössä mahdollisuus harjoitella eri ääntämismalleja sanojen mukaisesti, pilaa jatkuva mainosten ponnahtaminen ruudulle käyttökokemuksen ja vie pois keskittymistä itse opiskelusta. Mahdollisuus katsoa videoita Youtuben välityksellä tuntuu turhalta ominaisuudelta ja hiukan raskaalta toiminnolta videon avautuessa erilliseen selainikkunaan. Sovellus myös tarjoaa aktiivisesti mahdollisuutta ladata käännös-lisäosia tai vaihtoehtoisesti jokin toinen Awaben sovelluksista. Käyttäjän äänen kuuntelu tallentaa kaiken kuulemansa äänen, joten äänitettäessä on parasta olla täysin hiljaisessa ympäristössä jollei halua kuulla sekavaa meteliä oman äänensä lisäksi. Verrattuna Farikseen on Awabe – English Pronunciation kunnianhimoisempi yritys auttaa ihmisiä oppimaan lausumaan englantia. Fariksen keskittyessä sanojen toistamiseen ja jättäessä käyttäjän lausumisen oikeellisuuden käyttäjän itsensä tarkistettavaksi, Awabe opettaa nimenomaan ääntämistä sanojen sijaan ja tarjoaa mahdollisuuden oman ja näytöllä olevan sanan vertaamiselle. Kokonaisuudesta jää kuitenkin sekava tunne sillä sovellus kyllä markkinoi itseään ääntämistä opettavana työkaluna, mutta samaan aikaan tuputtaa käyttäjälle jatkuvalla syötöllä ladattavia lisäominaisuuksia, jotka muuttavat koko sovelluksen lähemmäksi kääntäjää kuin opetustyökalua.

5 Pohdinto

Tässä osiossa käydään läpi työstä syntyneet tulokset ja niiden mahdolliset jatkokehitysmahdollisuudet. Arvioidaan myös tekijän omaa oppimista ja opinnäytetyöprosessia kokonaisuudessaan.

5.1 Tulokset

Projektin konkreettisena tuloksena syntyi Android-käyttöjärjestelmällä toimiva mobiilisovellus, joka antaa käyttäjän harjoitella englanninkielen lausumista, sekä teoriaosio, jossa käytiin läpi puheentunnistuksen historiaa ja rakennetta, valmiin sovelluksen osat sekä vertailtiin vastaavanlaisiin markkinoilla oleviin sovelluksiin.

Puheentunnistus toimii ja on tarkkuudeltaan tarpeeksi hyvä Fariksen käyttötarkoitukseen. Tarkkuuteen, ja puheentunnistukseen yleensäkin, ei Googlen TTS kuitenkaan anna kehittäjän vaikuttaa tiettyjä valmiita osia lukuun ottamatta vaan puheen prosessointi tapahtuu Googlen kautta. Tämä myös rajoittaa offline-mahdollisuuksia puhelimen vaatiessa esiladatun kielipaketin, jotta sovellus voi käyttää puheentunnistusta offline-tilassa eikä tähän voi kehittäjä vaikuttaa.

Ratkaisematon ongelma sovelluksessa on aiheenvalinta-näkymä, jossa käyttäjä voi valita haluamansa sanat aihevalikosta. Valittaessa ”Sports” sovellus kaatuu ilman selkeää virhettä. Joissain puhelimissa sovellus palaa takaisin aiheen valintaan eikä näin ollen päästä käyttäjää suorittamaan harjoitusta loppuun. Tämä on suuri ongelma toteutuksessa ja vaatii mahdollisimman nopeaa korjausta.

Kokonaisuutena valmis sovellus on kuitenkin toimiva puhelinmalleissa, joissa ongelmia ei ole, ja osin jopa parempi kuin alkuperäisissä suunnitelmissa oli kaavailtu. Paljon myös jätettiin pois osia jotka alun perin kuuluivat oleellisina osina sovellukseen kuten puhutun ja kirjoitetun kielen valitsemisen mahdollisuus. SQLite-tietokannan sisällyttäminen sovellukseen ei ollut osana suunnitelmia sovellusta kehittämään lähdettäessä, vaan sisällytettiin mukaan vasta myöhemmässä vaiheessa tekijän havaittua sen tuoman lisäarvon sovellukselle. Myös sovelluksen pelillisyys nousi ratkaisevasti isompaan rooliin kuin alun perin oli tarkoitus.

Julkaistaessa sovellusta Play-kauppaan tuli vastaan kaksi huomattavaa tekijää, jotka vaativat ratkaisua. Ensiksi esteenä oli APK-tiedoston odotettua suurempi koko. Play-kaupan 150mt raja ylittyi Fariksen kohdalla niukasti koon ollessa 156mt. Koodia siivoamalla saatiin pudotettua pois 3mt tiedoston koosta mutta sovellus oli silti liian suuri. Suuri koko vei

myös pohjaa väitteeltä, jonka mukaan Faris olisi kevyt sovellus. Ratkaisu ongelmalle oli kuvatiedostojen pakkaaminen, jolla saatiinkin leikattua tiedoston kokoa nykyiseen 38,47mt tasolle. Toisena haasteena oli luoda tietosuojakäytäntö sovellukselle. Tällaista tehtävää en ole ennen joutunut tekemään, joten sen luominen tyhjästä tuotti haasteita. Lopulta sovellus saatiin kuitenkin Play-kauppaan ja odottaa siellä julkaisua tuotantoon.

5.2 Jatkokehitys

Jatkokehityksen kannalta Faris on hyvällä pohjalla. Sanavarastoa on helppo kasvattaa lisäämällä sanoja ja aiheita sovelluksen muistiin. Koodi sisältää myös valmiin metodin sanojen lisäämiselle mahdollista ominaisuutta varten, jossa käyttäjälle annetaan mahdollisuus lisätä omia sanojaan. Tämä ominaisuus kuitenkin muuttaisi sovellusta radikaalisti ja toisi sitä lähemmäksi käännöstyökalua mitä se ei ole.

Realistisesti asiaa katsottuna Faris kaipaisi hiotumpaa ulkoasua ja tarkempaa palautetta, jonka käyttäjä saa lausumansa sanan jälkeen. Tällä hetkellä ainoa palaute on vihreä tai punainen väri ja näytöllä näkyvä pistemäärä. Nykyinen malli nojaa ajatukseen, jossa sovellus on puhelimen käyttäjän henkilökohtaisessa käytössä, joten mahdollinen online-pistetaulukko vaatisi kirjautumisen ja koko taulukon uudelleen suunnittelun. Tälle yksi mahdollinen ratkaisu olisi Googlen Leaderboards joka tarjoaa API:t pisteenlaskua sekä Google Playta varten ja aktivoidaan Google Play Consolen kautta. Leaderboards hoitaa pisteenlaskun, vertailun, tulosten näyttämisen ja käyttäjän tiedot näkyviin, jolloin jokainen pelaaja voi helposti verrata tulostaan muihin pelaajiin. (Google Developers. Leaderboards. Luettavissa: https://developers.google.com/games/services/common/concepts/leaderboards Luettu: 21.11.2018)

Navigoinnin suhteen sovellusta voisi kutsua eräänlaiseksi hybridiksi, jos sitä jotenkin erikoisemmin halutaan kuvata. Aloitusnäkymässä on käytetty ”pohja-navigaatiota”,(bottom navigation) jonka kautta siirrytään pistetaulu ja ohjenäkymien välillä fragmentti-näkymien avulla. Fragmenttien käytön ratkaisi tarve sovelluksen navigaatiolle, jotta käyttäjä pystyy helposti navigoimaan näkymiin, jotka eivät liity itse harjoitteluun. Koska Googlen ohjeistuksissa neuvotaan käyttämään pohja navigaatiota ja fragmentteja navigoitaessa kolmen ja viiden näkymän välillä, oli ratkaisu enemmän kuin täydellinen Fariksen navigointia varten. Muut näkymät ovat kuitenkin toteutettu perinteisemmin activity-muodossa ja fragmentit lisättiin vasta myöhemmässä vaiheessa pohja navigaation mukana. Mikään ei estä tällaisen ratkaisun käyttöä sovelluksessa, mutta yleisiin ohjeistuksiin ja hyviin tapoihin kuuluu jommankumman valitseminen näkymien esittämistä varten. Jatkossa sovelluksen muuttaminen käyttämään pelkästään fragmentteja olisi mahdollinen kehityksen kohde.

Fragmentit latautuvat näytölle nopeammin kuin activityt ja sovelluksen kokoa saataisiin vähennettyä kahteen tai kolmeen activityyn.

Julkaisun aikana ilmenneet ongelmat APK-tiedoston koon ja joidenkin puhelinmallien kohdalla harjoituksen keskeytyminen ovat asioita, jotka täytyy käsitellä ja korjata mahdollisimman nopeasti, jos sovellusta halutaan jatkossakin tarjota käyttäjille Play-kaupassa. Kuvien kokoa on mahdollista pienentää vielä entisestään, jolloin sovelluksen kokoa saadaan pienemmäksi kuin se jo on. Mitä tulee puhelinmallien tukemiseen ja harjoittelun estävän virheen korjaamiseen, sovellus vaatii vielä työtä. Huolellisen testauksen järjestäminen jatkossa auttaisi paljon sovelluksen jatkokehityksessä.

5.3 Oma oppiminen ja opinnäytetyöprosessi

Lähdin tekemään opinnäytetyötä mielessäni haastaa itseni realistisissa mitoissa, jotta projektissa olisi uutta ja haastavaa, mutta kuitenkin myös paremmat mahdollisuudet onnistua kuin epäonnistua. Omasta mielestäni lopputulos on onnistunut ja kehitysprosessi, vaikka ajoittain hyvin stressaava olikin, oli opettavainen sekä nautittava kokonaisuus. Projektin aikana tuli Android-kehitys reilusti tutummaksi itselleni ja koen olevani varmempi oman osaamiseni suhteen kuin projektin alussa olin.

Jaoin tietoisesti työni kahteen osaan: sovelluksen kehitykseen ja teoriaosioon. Itse sovelluksen kehitykseen käytin suuren osan kesästä ja alkusyksystä, teoriaosio syntyi syksyn aikana. Tämä ratkaisu perustui puhtaasti haluuni keskittyä rauhassa sovelluksen kehittämiseen ja uuden oppimiseen. Tiesin teorian olevan itselleni haastavampi ja vähemmän mielenkiintoinen urakka, joten jätin sen toiseksi, jotta voisin sovelluksen valmistuttua keskittyä täysin sen kirjoittamiseen.

Aivan alussa en käyttänyt projektinhallintaan minkäänlaisia työkaluja vaan rakensin pohjan sovellukselle pelkkä ajatus mielessäni. Hyvin pian tämä tapa osoittautui riittämättömäksi, jopa haitalliseksi, kehityksen kannalta ja tästä oppineena turvauduin projektinhallinnassani Trelloon. Projektin osien jakaminen pieniin osiin helpotti huomattavasti keskittymistäni ja auttoi viemään sovellusta eteenpäin.

Valmiista projektista jäi hiukan ristiriitainen tunne. Toisaalta projekti onnistui tietyillä osa-alueilla paremmin kuin uskoinkaan, toisaalta siitä oli myös pakko jättää pois olennaisia osia kuten testaus käyttäjillä, puhutun ja kirjoitetun kielen välillä valitseminen sekä kääntäminen toisille kielille. Kielten osalta koko osuus hylättiin turhana johtuen Androidin

automaattisesta käännösominaisuudesta, jolloin omia käännöksiä olisi turha tuottaa näin pieneen sovellukseen.

Ajanhallinnollisesti projekti luisui alkuperäisestä aikataulustaan johtuen lähinnä tekijän omasta väärin arvioinnista. Aikataulut olivat olemassa mutta niistä ei pidetty niin tarkasti kiinni kuin alun perin oli tarkoitus. Tämä näkyi joidenkin ratkaisujen muuttamisena tai pois jättämisenä aikataulun lipsuessa enemmän tai vähemmän eri osien kohdalla. Tästä opittuna projektin aikataulua tulisikin kunnioittaa ja tekijä on ottanut tästä opikseen.

Varmuus mobiilikehittäjänä on kuitenkin kasvanut projektin aikana ja koen kykeneväni tuottamaan toimiva Android sovelluksia. Kehityksessä ja julkaisussa ilmenneiden ongelmien ansiosta koen tiedostavani paremmin testauksen tärkeyden ohjelmistokehityksessä ja ymmärrän sen tarpeellisuuden projektin koosta riippumatta. Paljon on vielä opittavaa, mutta projektin aikana tutuksi tulleet kehitystyökalut ja kertynyt tietotaito ovat nyt huomattavasti paremmat kuin alussa.

Lähteet

An introduction to Text-To-Speech in Android. Developers Blog.
Luettavissa: https://android-developers.googleblog.com/2009/09/introduction-to-text-to-speech-in.html Luettu: 8.6.2018

APK Expansion Files. Developers
Luettavissa: https://developer.android.com/google/play/expansion-files Luettu: 26.11.2018

Forsberg, M. Why is Speech Recognition Difficult? 24.2.2003
http://www.speech.kth.se/~rolf/gslt_papers/MarkusForsberg.pdf Luettu: 21.11.2018

Haider, K. 10 Free Mobile Apps to Help You Learn English Faster. hongkiat.com
Luettavissa: https://www.hongkiat.com/blog/mobile-apps-learn-english/ Luettu:28.9.2018

Hindy, J. 10 best language learning apps for Android. Android Authority
Luettavissa: https://www.androidauthority.com/best-language-learning-apps-android-750779/ Luettu: 27.9.2018

Kepuska, V, Bohouta, G. Comparing Speech Recognition Systems (Microsoft API, Google API And CMU Sphinx). Scribd.com
Luettavissa: https://www.scribd.com/document/342676269/Comparing-Speech-Recognition-Systems-Microsoft-API-Google-API-And-CMU-Sphinx Luettu: 10.6.2018

Krzeminska, M. 10 best language learning apps. lingualift.com
Luettavissa: https://www.lingualift.com/blog/best-language-learning-apps/ Luettu: 27.9.2018

What is Scrumban? Leankit.com
Luettavissa: https://leankit.com/learn/agile/what-is-scrumban/ Luettu: 21.9.2018

Markowitz, J. Toys That Have a Voice. 6.3.2003
Luettavissa: http://www.speechtechmag.com/Articles/PrintArticle.aspx?ArticleID=30031 Luettu: 17.11.2018

Pachiyappan, N. Why should we use SQLite in Android development?. 7.4.2016. Quora.com

Luettavissa: https://www.quora.com/Why-should-we-use-SQLite-in-Android-development Luettu: 14.7.2018

Pahuja, S. What is Scrumban? AgileAlliance.org
Luettavissa: https://www.agilealliance.org/what-is-scrumban/ Luettu: 21.9.2018

Pinola, M. Speech Recognition Through the Decades: How We Ended Up With Siri. 2.11.2011
Luettavissa: https://www.pcworld.com/article/243060/speech_recognition_through_the_decades_how_we_ended_up_with_siri.html Luettu: 17.11.2018

Oschler, R. Can CMU Sphinx be used for text to speech conversion? Quora.com
Luettavissa: https://www.quora.com/Can-CMU-Sphinx-be-used-for-text-to-speech-conversion Luettu: 10.6.2018

Rabiner, L, R. A Tutorial on Hidden Markow Models and Selected Applications in Speech Recognition.
Luettavissa: https://www.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/tutorial%20on%20hmm%20and%20applications.pdf Luettu: 20.11.2018

A short history of speech recognition. Sonix.ai
Luettavissa: https://sonix.ai/history-of-speech-recognition Luettu: 19.11.2018