Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Otto Järvinen

Sovellus datan luomiseen ja hallintaan neuroverkkopohjaista puhesynteesiä varten

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

3.4.2019

Tekijä Otsikko Sivumäärä Aika	Otto Järvinen Sovellus datan luomiseen ja hallintaan neuroverkkopohjaista puhesynteesiä varten 51 sivua 3.4.2019
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	tieto- ja viestintätekniikka
Ammatillinen pääaine	mobiilisovellukset
Ohjaajat	tutkimusjohtaja Antti Keurulainen lehtori Ilkka Kylmäniemi

Insinöörityön tarkoituksena oli toteuttaa tietotekniikka-alan yritykselle web-pohjainen sovellus, jolla luodaan puheesta ja tekstistä koostuvaa aineistoa yrityksen oman neuroverkkopohjaisen puhesynteesimallin harjoittamista varten. Lisäksi tavoitteena oli kehittää sovellukseen työkalu aineiston ja puhesynteesimallin harjoittamisen hallintaan, jotta harjoittamista voidaan ohjata suoraan selaimessa.

Sovellus toteutettiin käyttäen Vue-ohjelmistokehystä käyttöliittymässä ja Node.js:ää palvelimena. Suuressa osassa sovelluksen kehitystä olivat myös Vuex-tilanhallintajärjestelmä, Vuetify-käyttöliittymäkirjasto sekä tietokantana käytetty MongoDB. Aineiston luomista varten sovellukseen kehitettiin äänitystyökalu, jolla voidaan äänittää puhetta lausekohtaisesti ja tallentaa äänite sekä äänitteessä puhuttu teksti käytettäväksi myöhemmin harjoittamiseen. Sovellukseen kehitettiin myös aineistonhallintajärjestelmä, jotta harjoittamisessa käytettävää aineistoa on mahdollista vaihdella halutulla tavalla. Lisäksi sovellukseen toteutettiin paneeli, josta harjoittamisen voi käynnistää ja pysäyttää.

Sovelluksen testausvaiheessa havaittiin, että äänityksen tuottama äänenlaatu ei ole riittävän hyvä laadukkaan puhesynteesimallin tuottamiseksi. Äänenlaadun heikon tason syyksi selvisi useimmissa selaimissa oletuksena päällä oleva automaattinen äänenmuokkaus, joka muun muassa tasaa äänenvoimakkuuksia ja poistaa taustahälinää. Kun äänenmuokkaus poistettiin ohjelmakoodissa käytöstä, äänenlaatu parani selvästi.

Lopputuloksena saatiin toivotut ominaisuudet sisältävä sovellus, jolla voidaan äänittää, hallita ja käyttää aineistoa puhesynteesimallin harjoittamisessa. Sovellus kuitenkin vaatii lisää testausta äänenlaadun osalta, ennen kuin sitä voidaan käyttää täysipainoisesti puheen äänittämisessä. Jos äänenlaadun katsotaan olevan tarpeeksi hyvä, aineiston kerääminen nopeutuu huomattavasti. Tällöin sovellus tarjoaa yritykselle tehokkaan tavan kerätä ja hallita puhesynteesin kehittämisen kannalta arvokasta aineistoa.

Avainsanat	Vue.js, puhesynteesi, JavaScript, selainpohjainen äänitys

Author Title Number of Pages Date	Otto Järvinen Application for Creating and Managing Data for an Artificial Neural Network-Based Speech Synthesis 51 pages 3 April 2019
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Mobile Solutions
Instructors	Antti Keurulainen, Research Manager Ilkka Kylmäniemi, Senior Lecturer

The goal of the thesis was to develop a web-based application for creating data, which is used to train a speech synthesis model. In this context data means <audio, text> pairs, which are required by neural network to train speech synthesis. The application was conducted for an IT company. Another goal was to develop a tool for managing data and the training so that training can be controlled in the application.

The application was made by using Vue framework in the front-end and Node.js in the back-end. Also Vuex (state management pattern), Vuetify (material component framework) and MongoDB (database) were widely used. For creating data, a recording tool was developed to the application. The tool is used to record speech sentence by sentence and save the audio and spoken text so that the data can be used in the training. The application also includes a data managing system, which makes it possible to choose what data is used in the training. For controlling training, a panel was developed which allows the user to start or stop the training.

While testing the application it was easy to notice that the quality of recorded audio is not good enough for producing a decent speech synthesis model. The reason for poor quality was audio constraints, for example auto gain control and noise suppression, which are toggled on by default in browser. When audio constraints were toggled off in the application code, the quality of recorded speech became substantially better.

As a result, the company got a working application with required features. It is possible to record, manage and use data in the training of a speech synthesis model. However, the application requires more testing about the audio quality before it can be used to record speech. If the audio quality will be good enough, creating data becomes much faster. Then the application offers the company an efficient way to create and manage precious data that is needed to train speech synthesis.

Keywords	Vue.js, Speech Synthesis, JavaScript, Browser-based recording

Sisällys

1 Johdanto

Laadukkaan puhesynteesin kehittämisessä hyödynnetään nykyään syviä neuroverkkoja, jotka ovat koneoppimisen muoto. Neuroverkkojen oppimista kutsutaan harjoittamiseksi (*training*). Oppiakseen uusia asioita kone tarvitsee oppimateriaalia, joka tässä tapauksessa tarkoittaa tietyntyyppistä dataa. Datan (puhutaan myös *aineistosta*) tulee koostua <ääni, teksti> -pareista, joissa ääni tarkoittaa puhuttua lausetta äänimuodossa ja teksti samaa lausetta tekstimuodossa. Lauseita tarvitaan aineistoksi valtava määrä, jotta lopputuloksena on selkeää puhetta tuottava puhesynteesimalli. Kun mallia on kehitetty riittävästi, sille voidaan parhaimmillaan syöttää sellainen tekstimuotoinen lause, jota se ei ole koskaan nähnytkään, ja palautuksena saadaan selkeästi puhuttu lause, jota on vaikea erottaa oikean ihmisen puheesta.

Insinöörityön tarkoituksena oli kehittää Bitville Oy:lle web-sovellus, jolla aineistoa voidaan luoda ja käyttää osana puhesynteesin kehittämistä. Sovelluksen tarkoituksena on mahdollistaa lauseiden ketterä äänitys suoraan selaimessa, mikä nopeuttaa uuden aineiston luomista ja siten kehittää puhesynteesimallia paremmaksi. Aineiston määrällä on puhesynteesin kehittämisen kannalta ratkaiseva merkitys, sillä mitä enemmän on dataa, sitä paremmin neuroverkot oppivat.

Tarkoituksena oli myös saada selville, kuinka laadukasta puhetta selainpohjaisella äänityksellä voidaan saada aikaiseksi. Laadun ei haluta olevan huonompi kuin äänityskopissa äänitetyn puheen. Jos selaimessa äänitetty puhe on tarpeeksi hyvää, sovellusta voidaan alkaa käyttää osana aineiston keräämistä.

Sovelluksen arvo olisi yritykselle suuri, sillä se mahdollistaisi äänitysten tekemisen missä vain. Näin ollen ääninäyttelijän ei tarvitsisi saapua aina toimistolle äänityksiä varten, vaan hän voisi äänittää puhetta omatoimisesti esimerkiksi kotonaan. Kun molemmille osapuolille sopivia aikatauluja ei tarvitsisi enää sopia, uutta aineistoa kerääntyisi huomattavasti nykyistä nopeammin.

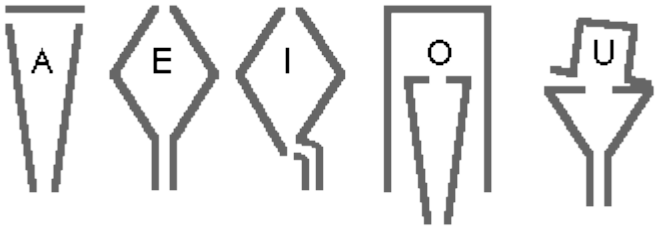
Insinöörityöraportissa tehdään ensin tiivis katsaus puhesynteesin historiaan, minkä jälkeen käydään läpi sovelluksen suunnittelu ja toteutus. Lopuksi kerrotaan sovelluksen testauksesta ja selvinneistä tuloksista.

2 Puhesynteesin kehittäminen

2.1 Puhesynteesin määritelmä

Puhesynteesillä tarkoitetaan puhetta, jota tuotetaan keinotekoisesti tietokoneen tai muun laitteen avulla. Puhesynteesi on kehittynyt vuosien saatossa siihen pisteeseen, että sitä on vaikea erottaa oikeasta ihmisen puheesta. [1.]

Aikaisimmat yritykset tuottaa synteettistä puhetta ulottuvat 1700-luvulle asti. Vuonna 1779 venäläinen professori Christian Kratzenstein esitti viiden pitkän vokaalin (/a/, /e/, /i/, /o/ ja /u/) fysiologiset erot ja teki laitteen, jolla tuottaa ne synteettisinä ääninä. Hän rakensi ihmisen ääntöväylää muistuttavia resonaattoreita (ks. kuva 1) ja sai ne tuottamaan ääntä värähtelevien kieliliuskojen avulla. [2.] Vaihtelemalla akustisia resonaattoreita ja valitsemalla käsin formantteja taajuuksia, rajoitettua puhetta onnistuttiin generoimaan tällä mekaanisella laitteella [3].

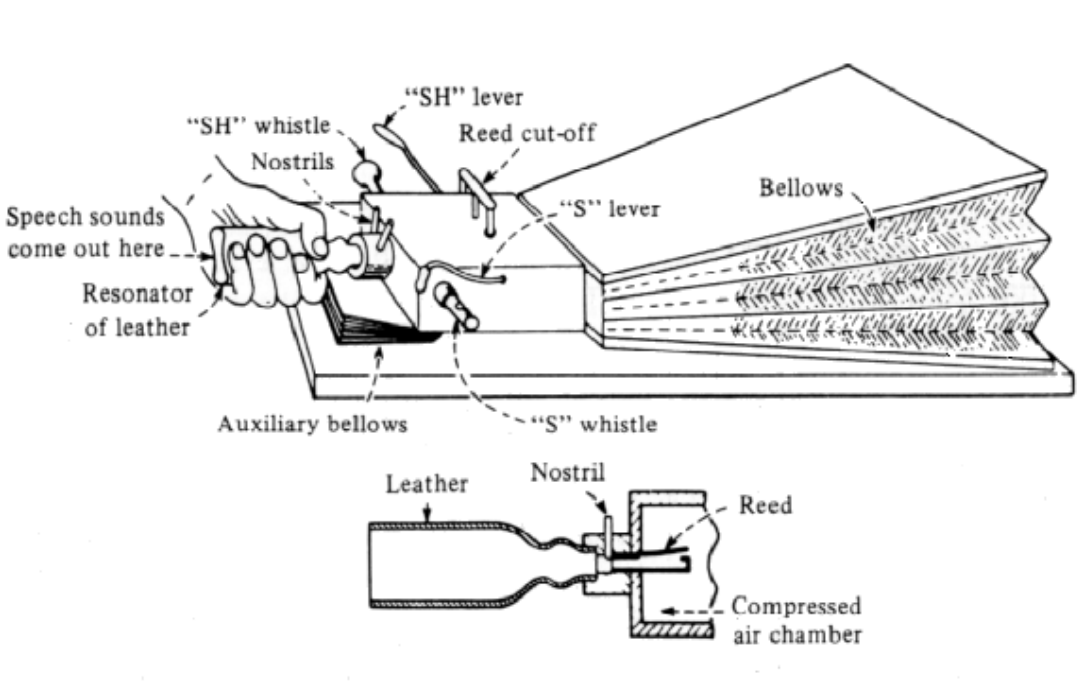


Kuva 1. Kratzensteinin resonaattorit vokaaliäänten synteesille. Resonaattorit aktivoidaan puhaltamalla ilmaa kieliliuskan läpi. [3.]

Myöhemmin Wienissä vuonna 1791 Wolfgang von Kempelen esitteli akustis-mekaanisen puhekoneen, joka pystyi tuottamaan yksittäisiä ääniä ja joitain ääniyhdistelmiä. Keksinnön tärkeimmät osat olivat keuhkoja mallintava painekammio, äänihuulia simuloiva, värähtelevä kieliliuska sekä nahkainen putkilo, joka muistuttaa ihmisen ääntöväylää. Muuntelemalla nahkaputkilon muotoa hän pystyi tuottamaan laitteellaan erilaisia vokaaliääniä. Laitteella pystyi muodostamaan myös konsonantteja tukkimalla neljää eri ilmakäytävää sormia käyttäen. Von Kempelen kehitti myös kielen ja huulet sisältävän ääntöväylämallin voidakseen tuottaa klusiileja eli äänteitä, jotka syntyvät ilmavirran pysäytyksestä ja sen jälkeen nopeasta purkautumisesta ääntöväylässä. (Suomalaisessa foneemijärjestelmässä klusiileja ovat muun muassa /p/ ja /t/). Hänen

tutkimustensa pohjalta muodostettiin myöhemmin teoria, jonka mukaan ääntöväylä, äänihuulten välinen ontelo ja huulet ovat ihmisen artikulaation perusta.

1800-luvun puolivälissä Charles Wheatstone rakensi kuuluisan versionsa von Kempelenin puhekoneesta (kuva 2). Se oli hieman alkuperäistä versiota monimutkaisempi ja pystyy tuottamaan vokaalit ja suurimman osan konsonanteista. Kone pystyi tuottamaan jopa täysiä sanoja. Vokaalit tuotettiin kielilliuskalla ja konsonantit, nasaalit mukaan lukien, kovalla ilmavirralla sopivan äänikäytävän läpi. [2.]

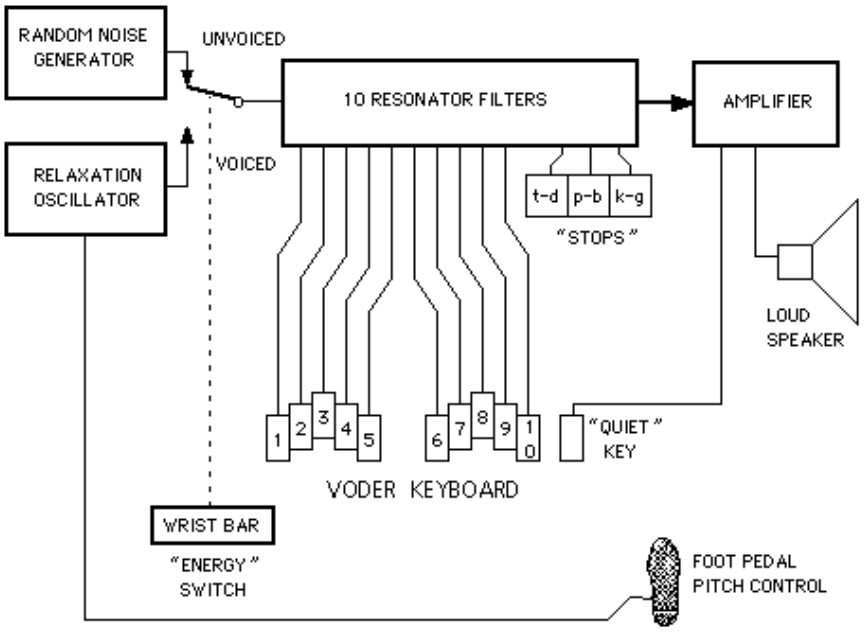


Kuva 2. Wheatstonen rekonstruktio von Kempelenin puhekoneesta [2].

1930-luvulla puhesynteesin kehitys otti harppauksen eteenpäin, kun Bell Telephone Laboratories (nyk. Nokia Bell Labs) kehitti insinööri Homer Dudleyn johdolla Vocoderin (Voice Coder). Sen avulla puheääni voitiin analysoimalla ”purkaa” osiin ja lähettää puhelinlankoja pitkin kohteeseen, minkä jälkeen se kyettiin kääntämään takaisin puhesignaaliksi. Vocoderia käytettiin yhtenä huippusalaisen viestinnän välineenä toisessa maailmansodassa. [4.]

Vocoderista inspiroituneena Bellin laboratoriossa kehitettiin kuvassa 3 näkyvä Voder (Voice Operating Demonstrator), jonka Dudley esitteli suurelle yleisölle vuonna 1939. Voderia voidaan pitää kaikkien aikojen ensimmäisenä jatkuvaa puhetta tuottavana pu-

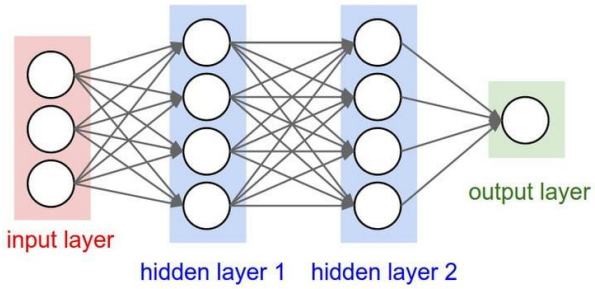
hesyntetisaattorina. Lähtösignaali saatiin joko äänilähteestä tai suhinaa tuottavasta oskillaattorista, ja se suoritettiin kymmenen BPF:n (Bandpass filter) läpi. Jalkapedaalilla kontrolloitiin suhinaoskillaattorin sävelkorkeutta. Kolme lisänäppäintä tarjosivat hetkellisen herätteen simuloimaan klusiileja. [5.]



Kuva 3. Voder-syntetisaattorin toimintaperiaate [5].

Ensimmäinen kunnollinen, suomenkielinen puhesyntetisaattori, SYNTE2, esiteltiin vuonna 1977. Se oli ensimmäisiä mikroprosessoripohjaisia syntetisaattoreita ja maailman ensimmäinen liikuteltava TTS (Text To Speech) -systeemi. Muutamia vuosia myöhemmin esiteltiin paranneltu SYNTE3-syntetisaattori, ja se oli monen vuoden ajan markkinoiden johtava tuote Suomessa. [2.]

Nykyään puhesynteesin kehityksessä hyödynnetään neuroverkkoja. Ne ovat laskentamalleja, joissa on haettu inspiraatiota ihmisten aivoista. Ne koostuvat kuvan 4 mukaisista sisään- ja ulostulotasoista sekä yleensä piilokerroksista (*hidden layers*). [6.] Piilokerrokset yhdessä sisääntulo- ja ulostulokerroksen kanssa muodostavat, joskus hyvin monimutkaisenkin, kuvauksen sisääntulon ja ulostulon välillä. Neuroverkko toimii funktioapproksimaattorina, mutta syvä neuroverkko kykenee mallintamaan periaatteessa mitä tahansa funktiota eli kuvausta. [7.]



Kuva 4. Neuroverkot [6].

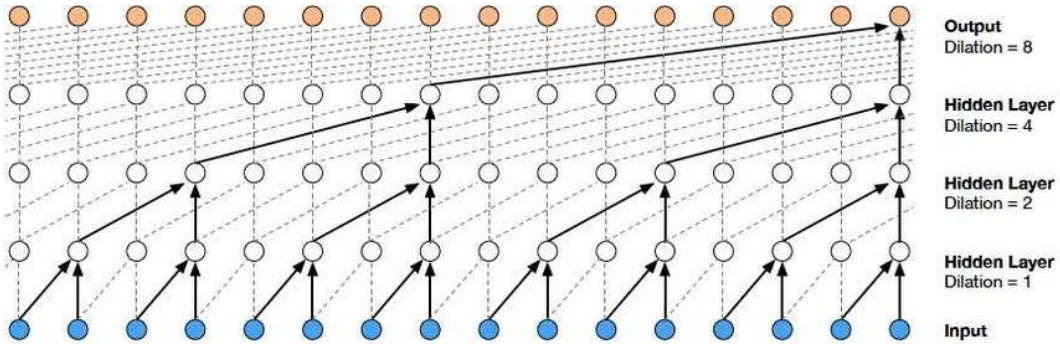
Syvät neuroverkot (*Deep learning*) on koneoppimisen muoto, jossa neuroverkko koostuu useista kerroksista sisääntulo- ja ulostulokerrosten välillä. Syvien neuroverkkojen luonteeseen kuuluu, että ne tarvitsevat paljon dataa. Datanäytteitä syötetään verkkoon toistuvasti ryhmittäin peräkkäin, ja jokaisen ryhmän jälkeen neuroverkko säätää omia parametrejään tullakseen jatkuvasti paremmaksi suorittamaan tehtäväänsä. Syvissä neuroverkkomalleissa neuronit on kytketty toisiinsa tietyllä tavalla ja jokainen neuroni suorittaa yksinkertaisen laskutoimituksen itsenäisesti. Neuroverkkojen avulla kone oppii tekemään hahmontunnistusta ja siten tunnistamaan toistuvia kaavoja ja yhtäläisyyksiä. [7.]

Puhesynteesiteknologia on perustana jokaiselle TTS-systeemille. On olemassa kaksi klassista ja eniten käytettyä TTS-menetelmää: *parametrinen* (*parametric*) ja *konkatenaatio* (*concatenative*).

Konkatenaatiosynteesissä lyhyistä ääniklipeistä muodostetaan suurempi äänite laittamalla niitä peräjälkeen yhteen. Puheesta tulee selkeää, koska sanat on korkealaatuisesti äänitetty, mutta myös epäluonnollisen kuuloista, sillä kaikkien sanojen äänitys kaikilla eri intonaatiolla ja tunnetilolla lähentelee mahdottomuutta. Jotta voisi generoida minkä tahansa lauseen, täytyisi äänittää kaikki mahdolliset sanat ja niiden eri muodot, mikä on lähestymistapana tehoton.

Parametrinen menetelmä ei käytä äänitettyjä ääniä, vaan generoi puhetta useiden eri parametrien avulla. Se käyttää tekstin kielellisiä ominaisuuksia, kuten foneemeja, sekä vokoderia. Lopputuloksena on äänen aaltomuoto. Tuotettu ääni ei kuitenkaan ole niin hyvä kuin voisi olettaa, sillä luonnollisen äänen luonti parametrien avulla ei toimi aina niin kuin toivoisi. Ongelman onneksi ratkaisevat syvät neuroverkot, joiden mallit ovat erittäin hyviä oppimaan datan ominaisuuksia.

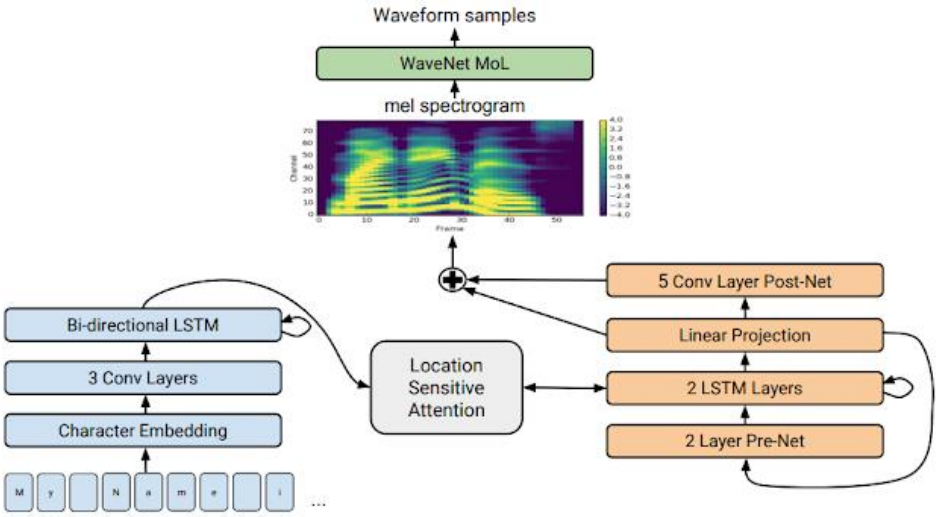
WaveNet on DeepMindin luoma syvä neuroverkko, jonka tekniikkaa kuvaava tieteellinen julkaisu julkaistiin syyskuussa 2016. Sitä testattiin kahdella eri kielellä (englannilla ja mandariinikiinalla), ja tulokset ylittivät kaikki parhaat olemassa olevat TTS-menetelmät. Se ottaa huomioon jokaisen aiemmin luodun näytteen generoidessaan uuden (ks. kuva 5). Kun WaveNetille syöttää puhetta, tekstiä ja kielen foneettiset ominaisuudet, se kykenee tuottamaan selkokielistä puhetta. Puhe ei kuitenkaan synny ilmaiseksi, sillä sen harjoittaminen WaveNetillä vaatii tietokoneelta huomattavasti laskentatehoa. [8.]



Kuva 5. WaveNet-neuroverkon kerrokset [9].

Alle vuosi WaveNet-julkaisun jälkeen Google julkaisi Tacotronin keväällä 2017. Tacotron on TTS-malli, joka tuottaa spektogrammin suoraan kirjaimista. Kun sille annetaan <teksti, ääni> -pareja, malli voidaan harjoittaa täysin tyhjästä. Spektogrammi syötetään Griffin-Lim-vokooderiin, joka tuottaa siitä aaltomuotoista, synteettistä puhetta. [10.]

Tacotron 2 julkaistiin joulukuussa 2017. Se käyttää *sequence-to-sequence*-mallia muodostaakseen spektogrammin (ks. kuva 6). Muokattu WaveNet-malli toimii vokooderina, kun spektogrammista syntetisoidaan aaltomuotoisia näytteitä. [11.]



Kuva 6. Tacotron 2 -mallin arkkitehtuuri [8].

Puheen laatua voidaan arvioida MOS-asteikolla (*Mean opinion score*), jossa arvosanaksi saadaan numero yhden ja viiden väliltä. Aluksi joukko testihenkilöitä laitetaan kuuntelemaan ääntä, jonka laatua jokainen arvioi asteikolla yhdestä viiteen. Annettujen arvosanojen keskiarvo määrittää MOS:n. [12.]

Tacotron 2 -mallia esittelevässä tieteellisessä julkaisussa esitellään mallin lisäksi eri puhesynteesitekniikoiden saavuttamia tuloksia MOS-asteikolla mitattuna. Testi toteutettiin valitsemalla ensin 100 satunnaista puhenäytettä. Generoitu ääni lähetettiin arvioitavaksi Amazon's Mechanical Turkin kaltaiseen ihmisarviointipalveluun (*human rating service*). Jokainen näyte arvioitiin vähintään kahdeksan testihenkilön toimesta 0,5 tarkkuudella asteikolla yhdestä viiteen. Tulosten perusteella jokaiselle tekniikalle laskettiin subjektiivinen MOS. Vertailukohtana puhesynteeseille on aito ihmisen puhe, jonka arvosanaksi on saatu 4,582. Huonoimman tuloksen (3,492) sai parametrinen menetelmä. Konkatenaatiomenetelmän tulos on 4,166 ja WaveNetin 4,341. Tacotron, joka käyttää Griffin-Lim-vokooderia, sai MOS-arvosanaksi 4,001. Kaikkein parhaan tuloksen sai julkaisussa esitelty Tacotron 2 tuloksella 4,526. Tacotron 2:n saama tulos on siis jo hyvin lähellä aidon puheen laatua. [11.]

Yksi tietokoneen suurimmista haasteista puhesynteesiä tuotettaessa on tekstin monitulkintaisuus: yksi sana voi tarkoittaa montaa eri asiaa, jolloin tekstiä lukiessa kontekstin ymmärtäminen nousee suureen osaan. Siksi on tärkeää jo esiprosessointivaiheessa

rajata tapoja, joilla lauseen voi puhua läpi siten, että jäljelle jää vain täsmällisin vaihtoehto.

Esiprosessoinnissa käydään teksti läpi ja siivotaan sitä niin, että tietokone tekee vähemmän virheitä puhuessaan lauseet. Numerot, päivämäärät, ajat, lyhenteet, akronyymit ja erikoismerkit täytyy muuntaa sanoiksi, sillä ne voidaan lausua monella eri tavalla. Esimerkiksi numero 1972 voidaan lausua paitsi lukumääränä ja vuotena (”tuhat yhdeksänsataaseitsemänkymmentäkaksi”) myös pin-koodina (”yksi yhdeksän seitsemän kaksi”). Siinä missä ihmiset ymmärtävät sanan asiayhteyden ja osaavat lausua sen oikein, tietokoneilla ei ole asiasta käsitystä. Siksi tietokoneet joutuvat turvautumaan tilastollisiin todennäköisyystekniikoihin tai neuroverkkoihin osatakseen tuottaa todennäköisesti oikeanlaisen lausumistavan sanalle. Esimerkiksi jos sana ”vuosi” esiintyisi samassa lauseessa kuin ”1972”, tietokoneella olisi apusana, johon perustaa arvaus. Tietokonetta hämäävät myös homografiset sanat, jotka lausutaan eri tavalla sen perusteella, mitä ne tarkoittavat. Suomen kielessä tällaista ongelmaa ei juurikaan ole, koska sanat lausutaan useimmiten samalla tavalla kuin ne kirjoitetaan. Englannin kielessä tällainen lause voisi olla vaikka ”I read the newspaper”, jossa sana ”read” lausutaan eri tavalla riippuen aikamuodosta. [1.]

2.3 Puhesynteesin käyttökohteita

Jos tavoitteena on kääntää lyhyt tai kohtuullisen pitkä teksti puheeksi yhtä tarkoitusta varten, silloin puhesynteesi ei pysty toistaiseksi kilpailemaan laadullisesti äänitystudiossa äänitetyn ääninäyttelijän puheen kanssa. Toisaalta, jos personoitua puhetta halutaan luoda suuria määriä ja toistuvasti, puhesynteesin käyttö on mitä luultavimmin paras vaihtoehto.

Puhesynteesiä voidaan käyttää esimerkiksi asiakastuessa. Ellei asiakkailla ole todella ainutlaatuisia kysymyksiä, automaattisen tuen tarjoaminen on varteenotettava vaihtoehto. Puhesynteesin tuomat hyödyt, kuten nopeat, asiakkaan nimen sisältävät kohdistetut vastaukset, voivat vähentää samaa kaavaa noudattavaa työtä.

Kun luodaan materiaalia kielen oppimiseen, tietyntyyppistä ääninäyttelijää ei aina ole saatavilla demonstroimaan oikeanlaista ääntämystä. Puhesynteesi voidaan liittää

osaksi oppimista yhdistämällä se opiskelijan edistymiseen ja kiinnostuksenkohteita tutkiviin metodeihin personoidun äänisisällön luomiseksi.

Jotkin organisaatiot käyttävät puhesynteesiä luodakseen monikielistä oppimateriaalia työntekijöilleen ympäri maailman.

Puhesynteesi tuo lisää mahdollisuuksia sisällöntuottajille. Luonnollisen kuuloisella äänellä puhuttuna artikkelit ja blogijulkaisut tavoittavat paremmin niitä, jotka haluavat kuunnella äänikirjoja tai podcasteja työskennellessään tai kuntoillessaan. Yhdistettynä visuaaliseen sisältöön tämä voi avata ovia myös kustannustehokkaille videomarkkinoille.

Kun tuotetaan selostavaa puhetta sisältäviä videoita, uusien ideoiden ja asiakkaiden pyyntöjen perusteella käsikirjoitusta joudutaan muokkaamaan jatkuvasti. Puhesynteesin avulla prosessia voidaan huomattavasti nopeuttaa, sillä puhe voidaan generoida heti uusiksi aina uusimmasta käsikirjoitusversiosta. Projektin lopussa, kun käsikirjoitus on saavuttanut lopullisen muotonsa, puhesynteesi voidaan korvata ääninäyttelijällä.

Yksi suosituimmista ominaisuuksista virtuaalisissa avustajissa, kuten Applen Sirissä, on niiden kyky asettaa muistutuksia. Tietokoneella generoitu puhe voi esimerkiksi herättää unesta tai pitää huolen siitä, että tärkeät asiat eivät unohdu. Käyttökokemus paranee, mitä enemmän puhe on personoitavissa erilaisilla tyyleillä. Esimerkiksi herätysääni voisi olla käyttäjän halutessa pehmeä ja rauhallinen tai vaihtoehtoisesti energinen ja motivoiva.

Radiolähetyksinä tuotetut tiedotteet voivat hyötyä puhesynteesistä. Esimerkiksi akuutit liikennetiedot tai vaihtelevat sääennusteet voidaan välittää kuulijalle jatkuvasti päivitettyinä.

Puhesynteesi antaa äänen kodin älykkäille laitteille. Se voi ilmoittaa, jos robottipölynimuri on jumissa. Se voi pyytää vahvistamaan jääkaapin automaattisesti generoiman ruokaostoslistan. Se voi myös ilmoittaa turvallisuusjärjestelmän havaitsemasta häiriöstä. [13.]

Tieto puheen ajoituksesta on puhesynteesiprosessin sivutuote. Puhemerkit kertovat, milloin sanan ääntäminen alkaa ja loppuu aktiivisen äänistriimauksen aikana. Peleissä tämä informaatio auttaa saamaan hahmot elämään synkronoimalla niiden ilmeitä puheen sisällön perusteella. [13.]

2.4 Datan vaatimukset

Datalla tarkoitetaan tässä yhteydessä aineistoa, jolla koneoppimisalgoritmia harjoitetaan. Data koostuu suuresta määrästä teksti-ääniparejа, joissa teksti on lause kirjoitetussa muodossa ja ääni on nauhoitettu äänite samasta lauseesta. Datan tulee olla laadukasta, jotta lopputuloksena kehitetty synteettinen puhe on mahdollisimman ihmismäistä. Ei ole olemassa oikeaa vastausta siihen, millaista datan tulisi täsmälleen olla, mutta loogisia virheitä välttämällä aineistosta saadaan tarpeeksi hyvää laadukkaan puhesynteesimallin aikaansaamiseksi.

Puhesynteesimallia harjoitettaessa koneoppimisalgoritmi vertailee aineistoon kuuluvia äänitteitä keskenään. Perusoletus algoritmin puolelta on, että äänitteessä sanotaan siihen liitetty teksti. Siksi on tärkeää, että lausetta äänitettäessä sanat artikuloidaan oikein. Muuten virhe saattaa toistua myöhemmin puhesynteesimallin ääntäessä sanaa. Kone saattaa tulkita myös äänityksessä tapahtuvan yskäisyn tai kolahduksen osana sanaa ja yrittää mallintaa sen itse sanan yhteyteen. Jotta lopputuloksessa ei kuulla ei-toivottuja asioita, äänitteen onnistuminen on syytä varmistaa, ennen kuin sitä käytetään osana aineistoa.

Onnistumisen kannalta tärkeää on tehdä äänitteet samanlaisiksi aloituksen ja lopetuksen osalta. Jokaisen äänitteessä olevan puheen pitäisi siis alkaa samasta kohdasta esimerkiksi siten, ettei alussa ole yhtään tyhjää. Sama pätee myös äänitteen loppuun. Jos puheen aloitusaika äänitteissä vaihtelee, algoritmi ei välttämättä saa hyvin kiinni puheesta, jota se yrittää oppia. Lopputuloksena voi olla epämääräistä ja irrationaalista synteettistä puhetta, josta ei erota yhtään oikeaa sanaa.

Data on arvokasta. Puhesynteesin kehittäminen on esimerkki sellaisesta hankkeesta, johon dataa tarvitaan hyvin paljon. Toivotunlaista dataa ei ole helposti saatavilla riittävästi, jolloin sitä täytyy luoda itse. Monipuolisesta ja suuresta datavarastosta on syn-

teettistä puhetta kehittävälle yritykselle jatkuvasti hyötyä, sillä samaa dataa voi käyttää aina uudelleen.

2.5 Tämänhetkinen datankeruumenetelmä

Bitville Oy:n koko datankeruuprosessi tapahtuu yrityksen tiloissa. Äänitykset tehdään toimistossa sijaitsevassa äänityskopissa. Koppi on vaimennettu, jotta äänitteisiin ei tartu ylimääräisiä taustaääniä tai tarpeetonta jälkikaiuntaa. Toisaalta äänityskoppi aiheuttaa sen, että äänitetty puhe resonoi matalilla taajuuksilla, eli matalat äänet voimistuvat. Mikrofonina käytetään ammattitason puhemikrofonia, jonka edessä on popsuodatin. Mikrofonin ja tietokoneen välissä on Focusrite Scarlett 2i2 -äänikortti. Äänityksessä käytetään Sound Forge -äänieditointiohjelmaa.

Ääninäyttelijän lisäksi tarvitaan erillinen äänittäjä valikoimaan tekstit ja hallitsemaan äänitysprosessia. Äänittäjä työskentelee kopin ulkopuolella, mutta hänellä on puheyhteys ääninäyttelijään kuulokemikrofonin avulla. Ääninäyttelijällä on kopissa näyttö, johon äänittäjä valitsee tekstit näytettäväksi. Flow-tilan säilyttämiseksi tekstiä äänitetään kerrallaan noin kahden minuutin verran, minkä jälkeen äänite tallennetaan. Äänen halutaan olevan äänitysvaiheessa mahdollisimman laadukasta ja luonnollista, joten automaattista suodatusta tai muuta prosessointia ei äänelle vielä tehdä.

Kun äänite on tehty, ensin leikataan pois epäonnistuneet äänitykset. Samalla äänitettä siistitään poistamalla mahdollisia häiriööääniä, kuten rasahduksia tai yskäisyjä. Sitten äänite pätkitään lauseiksi siten, että jokaisesta lauseesta tulee oma äänitiedostonsa.

Kaikille äänitiedostoille suoritetaan kolme laatua parantavaa skriptiä. Taajuuskorjaimella eli EQ:lla (*Ekvalisaattori*) leikataan matalat taajuudet pois. Tällöin äänitteistä saadaan vähennettyä äänityskopin aiheuttamaa matalien taajuuksien ”tehostusta”. Lopputulos kuulostaa heikompilaatuisissakin kaiuttimissa hyvältä, kun matalimmat taajuudet on leikattu pois. Toisena skriptinä vaimennetaan lähellä hiljaisuutta olevat äänet täysin. Näin voidaan karsia yksi mahdollisuus puhesynteesin laadun huononemiseen. Lopuksi äänitteille suoritetaan normalisointi. Se analysoi ääniaallon, laskee siitä neliöllisen keskiarvon (*rms-level*) ja säätää kaikkien äänitteiden äänenvoimakkuuden samalle tasolle.

Kun skriptit on ajettu, äänitiedostot tallennetaan wav-formaatissa 16-bittisenä ja näytteenottotaajuus on 22 050 hertsiä.

Äänitteiden tallennuksen jälkeen äänittäjä luo Excel-tiedoston, johon kirjoitetaan äänitetyt lauseet tekstimuodossa siten, että jokainen lause on allekkain omassa solussaan. Viereiseen soluun kirjataan id, joka on kyseisestä lauseesta tehdyn äänitiedoston nimi. Tällä tavoin saadaan linkitys lauseen teksti- ja äänimuodon välille. Lopuksi Excel-tiedoston pohjalta luodaan skriptin avulla tekstitiedosto, jossa on lauseet allekkain ja samalla rivillä lauseen kanssa polku lauseesta tehtyyn äänitiedostoon. Tätä tiedostoa kutsutaan datasetiksi, ja sitä käytetään sellaisenaan puhesynteesin harjoittamiseen. [14.]

2.6 Sovelluksen vastaus haasteisiin

Insinöörityönä toteutettava datankeruusovellus tarjoaa mahdollisuuden tehdä äänityksiä missä ja milloin vain. Koska sovellus on selainpohjainen, äänittäjä tarvitsee tietokoneen lisäksi ainoastaan Internet-yhteyden sekä mikrofonin ja mahdollisesti äänikortin. Sen sijaan, että äänitys työllistäisi kahta henkilöä (äänittäjää ja ääninäyttelijää), työ on mahdollista hoitaa yhden henkilön (ääninäyttelijän) toimesta. Ääninäyttelijä valitsee äänitettävän tekstin ja hoitaa itse äänityksen aloittamisen ja lopettamisen sekä laaduntarkistuksen ja tallentamisen. Laaduntarkistuksella tarkoitetaan varmistamista, että lause on lausuttu oikein ja että puheen alku tai loppu ei ole leikkaantunut äänitteestä pois.

Äänitteistä ei koidu ääninäyttelijälle ylimääräistä vaivaa, sillä ne eivät tallennu äänittäjän omalle tietokoneelle, vaan suoraan yrityksen palvelimelle. Siksi heti tallennuksen jälkeen yksittäinen äänite on helposti yrityksen saatavilla jatkokäsittelyä varten.

Sovelluksessa luoduille äänitteille on mahdollista tehdä automaattista editointia, kuten kohinanvaimennusta ja trimmausta, jotta äänitteistä tulee yhdenmukaiset. Siihen kuitenkin liittyy riskejä, jotka voivat saada äänitteen jopa huonommaksi kuin ennen editointia. Esimerkiksi kohinanvaimennuksessa täytyy määrittää desibeliraja, jonka perusteella äänitteestä vaimennetaan tiettyyn äänenvoimakkuuteen asti yltävät äänet. Kun äänitetään erilaisissa ympäristöissä, kohinan määrä voi vaihdella suuresti. Se tekee desibelirajan määrittämisen hankalaksi. Jos valitaan liian matala raja, kohinaa jää ää-

nitteeseen. Jos valitaan liian korkea, se haukkaa dataa itse puheesta, mitä ei missään nimessä haluta tapahtuvan. Vaikka yhdessä äänitykseen valitussa paikassa editointi onnistuisi, toisessa sama desibeliraja voi antaa aivan erilaisen lopputuloksen. Automaattiseen trimmaukseen vaikuttavat samat asiat. Jos annetaan koneen päättää, milloin äänitteen kuuluu alkaa, voi tapahtua sellaisia virheitä, että lauseen alku tai loppu jäävät pois editoidusta äänitteestä.

Automaattisen editoinnin toteuttaminen vaatii tarkempaa tutkimista, ennen kuin sitä voidaan soveltaa sovelluksessa tuotettuihin äänitteisiin. Siksi äänitteet tallennetaan järjestelmään sellaisinaan. Niitä on kuitenkin tarvittaessa mahdollista editoida manuaalisesti vanhalla menetelmällä, ennen kuin harjoittaminen käynnistetään.

Sovellus tarjoaa käyttöliittymän datasetin täydelliselle hallinnalle. Jokaisella ääninäyttelijällä on omat datasetit, joita voidaan muokata käyttöliittymästä käsin. Harjoittaminen voidaan myös käynnistää ja pysäyttää samassa ikkunassa. Kun puhesynteesimallia on harjoitettu riittävästi, sitä voidaan testata käännettäessä tekstiä puheeksi.

Edellä mainitut ominaisuudet ovat asioita, jotka tavallisesti on muutettu suoraan palvelinkoneen koodiin. Insinöörityönä toteutettava sovellus suoraviivaistaa monta vaihetta siten, että ennen palvelinkoneeseen suoraan tehdyt muutokset voidaan nyt tehdä käyttöliittymässä. Myös paikkariippuvaisuus vähenee merkittävästi, sillä sovelluksen tarjoamat ominaisuudet mahdollistavat puhesynteesin kehittämisen missä vain.

3 Sovelluksen suunnittelu

3.1 Vaaditut ominaisuudet

Insinöörityönä kehitettävän sovellukseen ominaisuudet voidaan jakaa neljään eri kokonaisuuteen, jotka ovat äänitteenhallinta, datasetin hallinta, harjoittamisen kontrollointi sekä harjoitetun puhesynteesimallin käyttö. Jokaisen kokonaisuuden tulee toimia moitteettomasti, sillä muuten vaikutukset saattavat näkyä negatiivisesti datanlisäysprosessin seuraavissa vaiheissa.

Datasetti muodostuu teksti-äänipareista, joissa teksti on yksittäinen lause ja ääni on sama lause puhuttuna. Oikeanlaisen datan keräämisen vuoksi on ehdottoman tärkeää, että äänitys tehdään lausekohtaisesti. Äänitettävän tekstin tulee siis olla pilkottuna lauseiksi siten, että jokaisen lauseen pystyy äänittämään ja tallentamaan erikseen. Tämä lähestymistapa mahdollistaa yksittäisten lauseiden täyden hallinnan: jos myöhemmin huomataan, että jokin lause on lausuttu huonosti tai siinä on jotain muuta ongelmaa, sen voi poistaa ilman, että muu äänitysdata muuttuu. Tällä toimenpiteellä voitaisiin peruuttaa esimerkiksi harjoitettavan mallin negatiivinen oppiminen poistamalla huono lause ja jatkamalla harjoittamista aiemmasta tallennuspisteestä lähtien.

Koska lauseita saattaa olla sivulla paljon allekkain, on hyvä olla näkyvä tieto siitä, mitä lausetta ollaan parhaillaan äänittämässä. Siksi äänitettävää lausetta pitää korostaa esimerkiksi jollain värillä. Sovelluksen tarvitsee tukea vain yhtä äänitystä kerrallaan, joten useamman raidan samanaikainen äänitys voidaan estää. Estetyt lauseenäänityspainikkeet osaltaan myös viestivät käyttäjälle, mikä lause on aktiivinen.

Äänityksen laatu heikkenee huomattavasti, jos ääni klippaa eli menee särölle. Äänittäjä ei välttämättä aina huomaa klippausta kesken äänityksen, ja siksi äänityssivulle tulee implementoida mikrofonin sisääntulosignaalin voimakkuutta indikoiva mittari, jonka väri muuttuu äänen klipatessa vihreästä punaiseksi. Kun väri on vaihtunut punaiseksi, äänittäjä tietää äänitteen klipanneen ja voi nauhoittaa sen uudestaan.

Puhesynteesin harjoittaminen vaatii tietokoneelta paljon muistia. Tästä syystä harjoittamisen onnistumiseen vaikuttaa myös yksittäisen äänitteen pituus. Bitvillessä on havaittu, että liian pitkä äänite voi saada näytönohjaimen muistin loppumaan, jolloin har-

joittaminen keskeytyy. Tällä hetkellä muisti saattaa loppua noin viidentoista sekunnin mittaisen äänitteen takia. Testeissä on selvinnyt, että muisti riittää vielä neljäntoista sekunnin mittaisiin äänitteisiin, joten sen täytyy olla enimmäiskesto jokaiselle äänitettävälle lauseelle.

Kun lause on äänitetty, sitä täytyy pystyä esikuuntelemaan ennen tallennusta. Joskus äänitettäessä saattaa käydä esimerkiksi niin, että ääninäyttelijä aloittaa puhumisen vähän liian aikaisin, jolloin lauseen ensimmäiset kirjaimet eivät tule nauhoitetuiksi. Se saattaa vaikuttaa lopputulokseen siten, ettei puhesynteesi toimi oikein kyseisen sanan tai kirjainyhdistelmän kohdalla muissakaan lauseissa. Siksi täytyy olla mahdollista hylätä nauhoitus tai äänittää lause uudestaan, kunnes lause on äännetty riittävän selkeästi.

Yrityksen käyttämä harjoittamismalli vaatii äänitteeltä tietyntyyppistä tiedostomuotoa. Siksi äänite on tallennettava yksikanavaisena, 16-bittisenä WAV-tiedostona, jonka näytteenottotaajuus on 22 050 hertsiä. Äänitteet täytyy voida tallentaa joko yksittäin tai kaikki kerrallaan. Kerrallaan tallentamisen hyödyt tulevat esiin esimerkiksi silloin, kun ollaan äänittämässä runsaasti puhetta yhdeltä istumalta. On hyvä, jos käyttäjän ei tarvitse kuluttaa aikaa asioihin, joita voidaan nopeuttaa automaattisilla keinoilla.

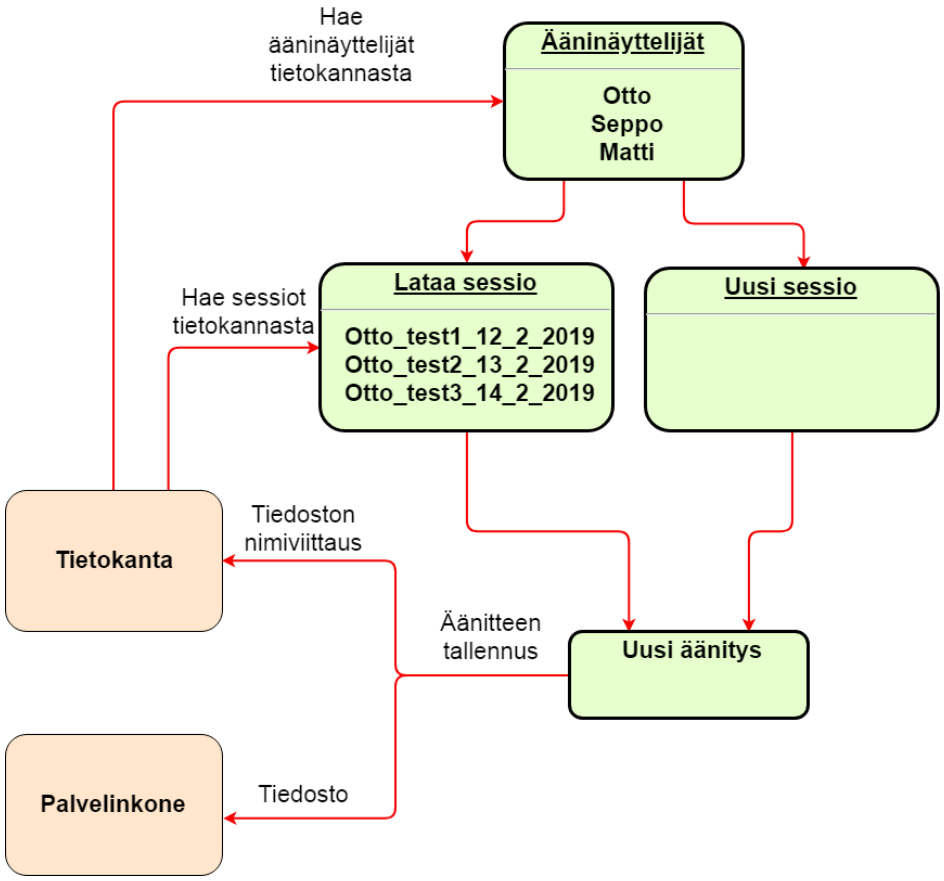
Äänitteet tallennetaan palvelinkoneelle kuvan 7 (s. 17) mukaisesti. Tallennuksen yhteydessä tiedostonimestä tehdään tekstimuotoinen nimiviittaus, joka tallennetaan tietokantaan. Näin tekemällä tietokantaan ei kerry kookkaita tiedostoja. Esimerkiksi jos äänitiedoston nimi olisi *audio123.wav*, tietokantaan voitaisiin tallentaa pelkkä merkkijono *audio123* ja itse tiedosto suoraan palvelinkoneelle. Myöhemmin, kun äänitettä halutaan kuunnella, tietokantaan tallennettua merkkijonoa käyttämällä voidaan hakea oikea tiedosto palvelinkoneelta. Jotta ratkaisu toimii, jokaisen äänitiedoston nimen tulee olla uniikki. Muuten voi käydä niin, että äänitiedostoa haettaessa palvelinkoneelta saadaan väärä tiedosto.

Sujuvan äänittämisen edistämiseksi tulee olla mahdollista puhua kaikki lauseet keskeytyksettä, jotta intonaatio säilyy lauseiden välillä. Intonaatio on tärkeä osa ihmismäistä puhetta, joten sen säilyvyys äänitettäessä saattaa vähentää äänen robottimaisuutta lopputuloksessa. Toiminnallisuus voidaan toteuttaa esimerkiksi pikanäppäimillä, joiden avulla saadaan automatisoitua yhdellä painalluksella äänityksen pysäytys, tallennus sekä seuraavaan lauseeseen siirtyminen ja sen äänityksen aloittaminen.

Sovelluksessa tulee ottaa huomioon, että ääninäyttelijöitä voi olla useita. Harjoittamiseen käytettävän datasetin tulee sisältää vain yhden ihmisen ääntä, joten äänitysten tulee olla tallennettu henkilökohtaisiksi. Tullessaan äänityssivulle käyttäjän pitää siis pystyä tunnistautumaan esimerkiksi valitsemalla itsensä ääninäyttelijälistasta, jolloin jokainen äänitetty lause tallentuu käyttäjäkohtaisesti.

Kun harjoittaminen käynnistetään, siinä ei välttämättä haluta käyttää jokaista äänitettä. Lauseita voidaan muun muassa äänittää erilaisissa ympäristöissä, jotta saadaan käsitystä harjoittamisen onnistumisesta kaikenlaisissa olosuhteissa. Siksi on hyvä, jos äänityksiä voi ryhmitellä erilaisten äänitystilanteiden mukaan. Ryhmittelyn puolesta puhuvat myös testit, joissa halutaan kokeilla esimerkiksi vieraskielisten sanojen harjoittamista. Jos esimerkiksi vieraskielisiä sanoja sisältäviä lauseita on kymmeniä, ne on mukavampi lisätä tai poistaa datasetistä kerrallaan ryhmänä kuin yksitellen.

Äänityksistä koostuvaa ryhmää kutsutaan tästä eteenpäin *sessioksi*. Jotta jokainen tehty äänitys saadaan ryhmiteltyä session alle, täytyy ääninäyttelijän valita tunnistautumisen yhteydessä, luoko hän uuden session vai jatkaako vanhasta (ks. kuva 7). Uusi sessio on oletuksena tyhjä, kun taas vanhoista löytyvät kaikki niihin tallennetut äänitteet. Sekaannusten välttämiseksi session nimestä on hyvä käydä selväksi sekä ääninäyttelijän nimi että session luomisaika. Aktiivista sessiota tulee voida tarkastella äänityksen ohella. Siihen kuuluvia äänityksiä tulee voida paitsi kuunnella myös tarvittaessa poistaa. Lisäksi session kokonaiskesto on tärkeä nähdä, jotta ollaan selvillä äänitysdatan todellisesta määrästä. Lausemäärästä sitä on yksinään mahdotonta arvioida, koska yhden lauseen kesto voi vaihdella yhden ja neljäntoista sekunnin välillä.



Kuva 7. Äänityssivun toimintaperiaate.

On hyvä, jos ääninäyttelijällä on mahdollisuus kirjata huomioita muistiin äänitystilanteessa. Siksi äänityssivulle kehitetään muistiinpanoja kirjaava ja näyttävä työkalu. Ääninäyttelijä voi kirjoittaa muistiinpanoja esimerkiksi äänitysolosuhteista tai äänitysprosessin aikana ilmenneistä ongelmista. Muistiinpanot tallennetaan sessiokohtaisesti, ja niitä voidaan myöhemmin tarkastella ja muokata.

Datasetin sisällön täytyy olla muokattavissa käyttöliittymästä. Käyttäjän pitää siis voida lisätä ja poistaa dataa datasetistä. Käyttöliittymässä ei kajota yksittäisiin äänityksiin, vaan siellä hallitaan äänityskokonaisuuksia eli sessioita. Sekaannuksen välttämiseksi sivulla näytetään vain yhden ääninäyttelijän sessiot kerrallaan, koska jokaisella ääninäyttelijällä on oma datasetti.

Harjoittaminen on tavallisesti käynnistetty manuaalisesti palvelimella, mikä pakottaa käyttäjän olemaan toimistolla. Siksi harjoittamisen halutaan olevan mahdollista aloittaa käyttöliittymästä. Harjoittamisen käynnistys vaatii parametreiksi datasetin ja vapaaeh-

toisesti tallennuspisteen. Tallennuspisteen avulla voidaan jatkaa harjoittamista siitä, mihin viimeksi on jääty. Tallennuspisteen käyttämisen sijasta harjoittaminen voidaan kuitenkin aloittaa myös alusta. Yksi sovelluksen tärkeimmistä ominaisuuksista äänityksen ohella on harjoittamisen hallinta käyttöliittymästä. Vaatimuksena on paitsi mahdollisuus käynnistää harjoittaminen halutuilla parametreillä, myös pysäyttää se ja tarvittaessa uudelleenkäynnistää. Lisäksi käyttäjän tulee pystyä näkemään käyttöliittymässä tietoja harjoittamiseen liittyen, jotta voidaan nähdä senhetkinen tila ja päätellä, onko harjoittaminen parhaillaan käynnissä.

Kun riittävä määrä harjoittamista on tehty ja halutaan kokeilla, miltä puhesynteesimalli kuulostaa, se pitää voida aktivoida käyttöliittymässä ennen tekstin kääntöä puheeksi. Käyttöliittymässä on otettava huomioon käyttäjää informoiva ulkoasu siten, ettei tule epäselvyyksiä aktiivisesta puhesynteesimallista.

3.2 Käyttöliittymä

Äänityssivulta tulee käydä selkeästi ilmi, kuka on äänittämässä ja mikä sessio on aktiivinen. Näiden tietojen tulee olla äänitettäessä kunnossa, jotta voidaan varmistua siitä, että äänitykset tallentuvat oikein eikä äänite tallennu esimerkiksi väärän ääninäyttelijän nimen alle.

Klippimittari implementoidaan äänityssivulle näkyvälle paikalle, jotta äänittäjä voi varmistua siitä, ettei ääni missään vaiheessa klippaa. Mittarin esittämä mikrofonin sisääntulosignaali on väriltään vihreä, ja se liikkuu horisontaalisessa suunnassa signaalin voimakkuuden perusteella. Jos signaali on liian voimakas ja ääni klippaa, vihreä väri välähtää punaisena. Mittarin viereen implementoidaan myös pieni, vihreä ruutu, joka klippauksen myötä muuttuu punaiseksi. Siinä missä mittarin signaali vaihtuu klipatessa nopeasti punaiseksi ja heti takaisin vihreäksi, ruudussa oleva punainen väri jää näkyviin, kunnes se nollataan vihreäksi. Ruudun punainen väri kertoo, että äänite on klipannut jossain vaiheessa. Äänittäjä huomaa sen, vaikka signaalin väri olisi jo ehtinyt jo palautua punaisesta vihreäksi.

Äänityssivulla tulee äänittämisen lisäksi pystyä hallinnoimaan äänitettäviä lauseita. Sivulla tulee olla tekstikenttä, johon voidaan kirjoittaa tai liittää suuria määriä tekstiä

kerralla. Äänitystyökalu sitten pilkkoo tekstin automaattisesti lauseiksi, jotta lausekohtainen äänitys on mahdollista. Tekstikentän muokkaaminen yksittäisten lauseiden sijasta mahdollistaa nopeat tekstinkorjaukset ja tarvittaessa usean lauseen samanaikaisen päivityksen, kun pilkotut lauseet haetaan suoraan tekstikentästä. Sivun täytyy myös tukea useita tekstikenttiä, jotta yhdelle sivulle ei kerry liikaa tekstiä ja sen myötä lauseita. Äänitystyökalu näyttää äänitettävät lauseet aina aktiivisesta tekstikentästä, joten tekstikentän vaihdon tulee vaihtaa myös lauseet.

Jokaisella lauseella täytyy olla oma alue, jossa äänitykseen liittyvät painikkeet sijaitsevat. Aluksi jokaisella lauseella on näkyvissä vain äänityksen aloittava painike. Äänityspainiketta painettaessa sen tilalle ilmestyy äänityksen pysäytyspainike ja 14 sekunnin aikarajaa esittävä mittari. Kun äänitys pysäytetään, pysätyspainikkeen tilalle tulevat jälleen äänityspainike ja napit äänitteen toistoa ja tallennusta varten. Jos äänitystä ei pysäytetä itse ja 14 sekunnin aikaraja ehtii kulua umpeen, äänitys pysähtyy itsestään ja toimii, kuin pysäytyspainiketta olisi painettu. Äänitettä toistettaessa toistonappi vaihtuu taukonapiksi, kunnes siitä painetaan uudelleen tai kun äänitys on kuunneltu loppuun. Sitten painike vaihtuu takaisin toistonapiksi. Tallennetun äänitteen täytyy erottua tallentamattomista, jotta samaa äänitettä ei tallenneta tahattomasti useaan kertaan. Kuitenkin sama lause täytyy pystyä äänittämään uudelleen tallentamisen jälkeenkin, jolloin tallennusmahdollisuuden täytyy palata kyseiseen lauseeseen.

Käyttöliittymässä täytyy olla erillinen nappi, jota painamalla pikanäppäimet tulevat aktiivisiksi. Samasta napista pitää käydä ilmi, ovatko pikanäppäimet aktiivisia vai eivät. Kun pikanäppäimet aktivoidaan, niiden käyttöä kuvaava ohje ilmestyy näkyviin. Samalla aktiivinen lause (oletuksena ensimmäinen) ympäröidään esimerkiksi mustalla viivalla. Aktiivista lausetta voidaan vaihdella navigoimalla ylös- tai alaspäin nuolinäppäimillä. Oikea nuolinäppäin aloittaa aktiivisen lauseen äänityksen. Kun oikeaa nuolinäppäintä painetaan uudestaan äänityksen ollessa päällä, äänitys pysäytetään, äänite tallentuu, seuraava lause vaihdetaan aktiiviseksi ja sen äänitys aloitetaan automaattisesti. Jos lauseita on samalla sivulla useita eivätkä ne mahdu yhteen näkymään, tarvitaan automaattisesti toimivaa vieritystä. Automaattisen näytön vierityksen tulee huolehtia siitä, että aktiivinen lause on aina näkyvissä äänittäjälle. Tämä kaava toistuu niin pitkään, kuin lauseita riittää. Kun viimeinen lause on äänitetty ja tallennettu, automaattinen äänitys lopetetaan. Jos yksittäisen lauseen äänitys epäonnistuu esimerkiksi väärin lausutun sanan vuoksi, vasemman nuolinäppäimen painallus pysäyttää äänityksen ja jättää sen

tallentamatta. Oikealla nuolinäppäimellä lauseen äänitys voidaan uusia ja sitten jatkaa siitä, mihin jäätiin. Pikanäppäinten käyttö ei saa estää hiiren käyttöä äänitystilanteessa. Hiirellä täytyy siis voida aloittaa ja pysäyttää äänitys, vaikka pikanäppäimet olisivat käytössä. Kun pikanäppäimet deaktivoidaan, aktiivista lausetta indikoiva kehikko poistuu näkyvistä. Luonnollisesti pikanäppäimet eivät saa toimia silloin, kun ne eivät ole aktiivisia.

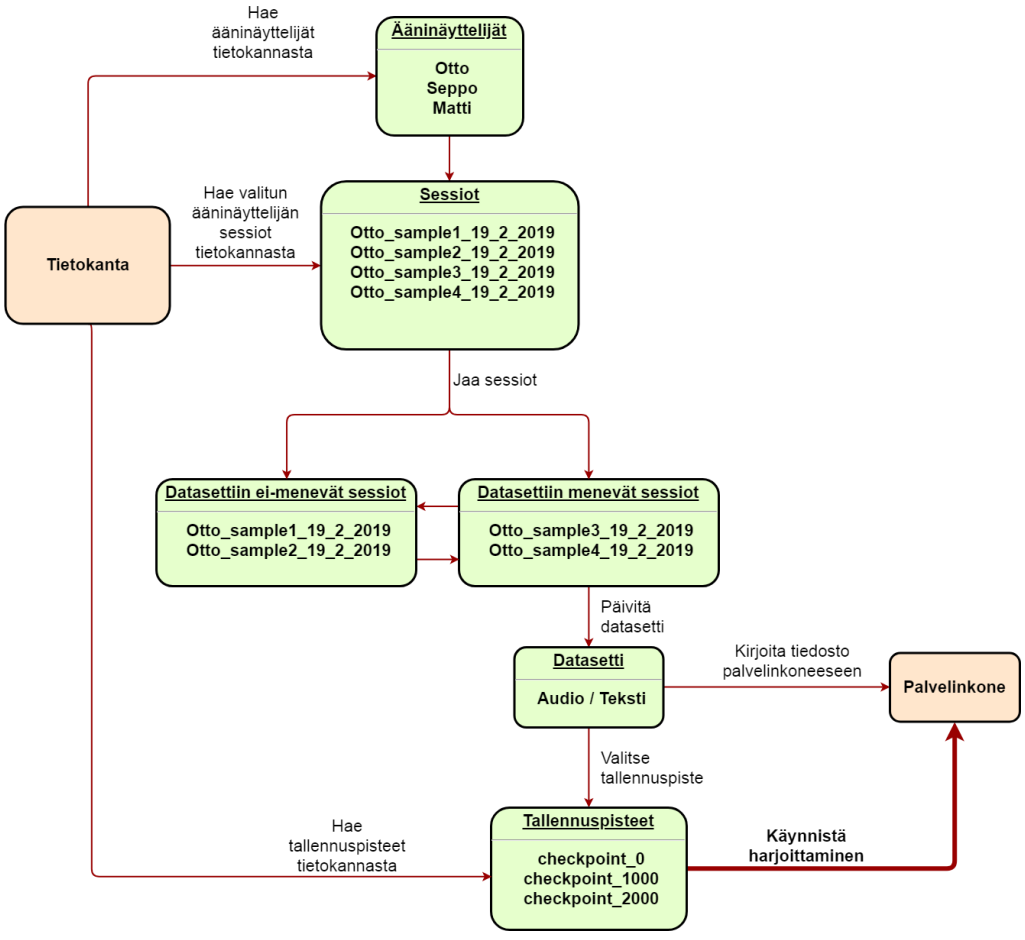
Aktiivista sessiota täytyy olla mahdollista tarkastella äänityssivulla, jotta voidaan paitsi varmistua sen sisällöstä myös muokata sisältöä. Jokainen sessioon kuuluva äänite listataan siten, että äänitteestä käy ilmi puhuttu lause tekstimuodossa, luomisaika ja kesto. Lisäksi äänitteen tulee olla kuunneltavissa ja poistettavissa. Myös session sisältämien äänitteiden yhteiskeston olisi hyvä olla näkyvillä. Muistiinpanot näytetään ponnahdusikkunassa, joka aukeaa napista. Muistiinpanoja kirjoitetaan niille varattuun tekstikenttään, minkä jälkeen ne voidaan tallentaa tai muutokset voidaan peruuttaa.

Sovellukseen luodaan oma sivu harjoittamisen kontrollointia varten. Sivulla tulee voida paitsi hallinnoida harjoittamisessa käytettäviä datasettejä myös käynnistää ja pysäyttää varsinainen harjoittaminen (ks. kuva 8). Lisäksi sivulta täytyy nähdä käynnissä olevaan harjoittamiseen liittyvää tietoa.

Hallinnointisivulla käyttäjän tulee ensin valita se ääninäyttelijä, jonka datasettiä ollaan muokkaamassa. Kun ääninäyttelijä on valittu, kaikki hänelle kuuluvat sessiot listataan kuvan 8 mukaisesti kahteen vierekkäin olevaan listaan siten, että vasemmalle puolelle menevät ne sessiot, jotka eivät ole datasetissä, ja oikealle ne, jotka joko ovat datasetissä tai jotka ovat menossa sinne. Jokainen sessio saa oman valintaruudun, jonka avulla sessioiden siirtely datasettiin menevien ja ei-menevien välillä sujuu helposti. Lisäksi jokaisen session äänitteiden yhteenlaskettu pituus ja määrä näytetään sivulla.

Datasetin päivittämistä varten kehitetään nappi, jota painamalla kaikista oikealla puolella olevien sessioiden äänitteistä luodaan datasetti. Tämä vaihe on aina pakollinen ennen harjoittamisen aloittamista, jotta voidaan varmistua datasetin sisällöstä. Ennen harjoittamista täytyy valita myös tallennuspiste, josta harjoittaminen aloitetaan. Sitä varten sivulle luodaan pudotusvalikko, johon tuodaan kaikki ääninäyttelijälle kuuluvat tallennuspisteet. Jokaisella ääninäyttelijällä tulee olla oletuksena mahdollista valita tallennuspisteeksi 0, jotta harjoittaminen voidaan aloittaa tarvittaessa alusta.

Käynnistysnapin tulee olla lukittuna, kunnes datasetti on päivitetty ja tallennuspiste valittu. Kun käynnistysnappia painetaan, sen tulee heti muuttua lukituksi ”Harjoittamista käynnistetään” -napiksi, jotta useampaa käynnistystä ei voida tehdä heti useita kertoja peräkkäin. Sitten kun harjoittaminen on alkanut, nappi vaihtuu ”Pysäytys”-nimeä kantavaksi napiksi. Kun pysäytysnappi on näkyvillä, voidaan olla varmoja siitä, että harjoittaminen on käynnissä. On mahdollista harjoittaa vain yhtä mallia kerrallaan, joten tiedon täytyy näkyä kaikille sivulla vieraileville käyttäjille samalla tavalla. Kun harjoitus pysäytetään, pysäytysnappi muuttuu heti lukituksi ”Harjoittamista pysäytetään” -napiksi, ettei tehdä useita pysäytyspyyntöjä toistuvasti. Kun harjoittaminen on turvallisesti pysäytetty, painike muutetaan takaisin käynnistysnapiksi. Värit pidetään käyttäjälle informatiivisina siten, että käynnistysnappi on vihreä, pysäytysnappi punainen ja välitiloissa olevat painikkeet oransseja.



Kuva 8. Harjoittamisen käynnistämistä edeltävät toimenpiteet.

3.3 Käytetyt teknologiat

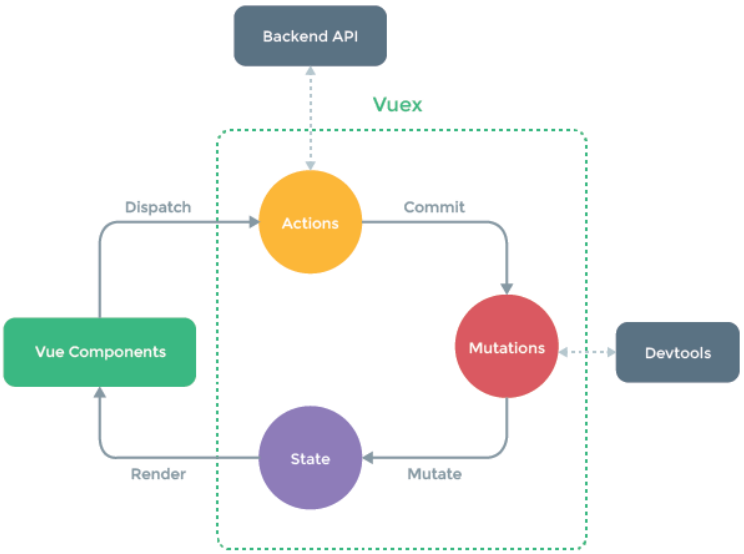
Insinöörityönä kehitetty sovellus toteutettiin MEVN-arkkitehtuurilla, joka koostuu tietokantana toimivasta MongoDB:stä, Noden web-palvelinpaketti Express.js:stä, Javascriptin ohjelmistokehys Vue.js:stä ja Node.js-palvelimesta. Puhesynteesin harjoittaminen suoritetaan Python-palvelimella. Pythonin web-palvelinpakettina käytetään Flaskia. Projektin edistymistä seurattiin Pivotal Tracker -ohjelmalla, joka on hallintatyökalu ketterän kehityksen projekteihin. Koodinhallintaan käytettiin Git-versionhallintajärjestelmää.

Vue on progressivinen ohjelmistokehys käyttöliittymien rakentamiseen. Sen ydinkirjasto keskittyy ainostaan käyttäjälle näkyvään osaan koodista, joten muiden kirjastojen ja jopa olemassa olevien projektien liittäminen yhteen Vue-projektin kanssa on helppoa. [15.]

Vuen kanssa alkuunpääseminen on melko suoraviivaista. Sen käyttö ei vaadi Webpackia, mikä saattaa madaltaa kynnystä päästä alkuun. Vuen takana ei ole Reactin ja Angularin tavoin suurta yhtiötä, vaan hyvin aktiivinen avoimen lähdekoodin yhteisö, mikä pitää ohjelmistokehyksen tuoreena ja kehityksen nousujohteisena. [16.]

Vuex on tilanhallintakaava ja -kirjasto Vue.js-sovelluksille. Se tarjoaa kaikille sovelluksen komponenteille jaetun tilan (*store*), jonne voi väliaikaisesti tallentaa dataa. Store koostuu pääasiallisesti kolmesta asiasta: tila (*state*), toiminnot (*actions*) ja mutaatiot (*mutations*). Tila säilyttää dataa, johon kaikilla sovelluksen komponenteilla on pääsy. Se on hyödyllinen esimerkiksi silloin, kun usea komponentti käyttää samaa dataa. Jos sovellus tukee esimerkiksi useita kieliä, aktiivinen kieli voidaan tallentaa tilaan. Kun jokainen komponentti hakee aktiivisen kielen tilasta, voidaan varmistua siitä, että data on sama kaikilla. Jos komponentissa vaihdetaan kieltä ja uusi kieli tallennetaan tilaan, se muuttuu automaattisesti kaikissa komponenteissa, jotka hakevat datan tilasta. Jotta voidaan varmistua siitä, että Vuex toimii reaktiivisesti, tilaan tehtävät muutokset täytyy tehdä mutaatioissa (ks. kuva 9). Hyvä käytäntö on lähettää data ensin komponentista Storen toimintoihin. [17.] Toiminnot ovat samantyyppisiä kuin mutaatiot, mutta sen sijaan, että ne muuttaisivat tilaa, ne suorittavat mutaatioita. Toisin kuin mutaatiot, toiminnot voivat sisältää asynkronisia toimintoja, kuten esimerkiksi tietokantakutsuja. Ainoa oikea tapa muuttaa tilaa on suorittaa mutaatio. Tila on rakennettu reaktiiviseksi, joten

kun tilaa muutetaan, sitä tarkkailevat Vue-komponentit päivittyvät automaattisesti. Jos mallista poiketaan esimerkiksi siten, että tilaa muutetaan suoraan Vue-komponentista, komponentin automaattinen päivitys ei välttämättä toimi. Samoin jos asynkronista toimintoa suoritetaan synkronisessa mutaatiossa, ei voida olla varmoja siitä, että ohjelma ehtii suorittamaan esimerkiksi kaikki tietokantakutsut. [18.]



Kuva 9. Vuexin rakenne ja sijainti Vue-sovelluksessa. Sivun näkymä päivittyy automaattisesti, kun tilaa (*state*) muutetaan [17].

Vuetify on Vuen käyttöliittymäkirjasto, joka tarjoaa Material Design -standardit täyttäviä käyttöliittymäkomponentteja käytettäväksi. Komponentit on rakennettu helpoiksi oppia ja muistaa. Samaan aikaan niissä on suuri määrä toiminnallisuutta, joka on kirjoitettu kehittäjälle valmiiksi. Vuetifyn kattavassa dokumentaatiossa on listattuna kaikki komponentit ja niiden property-muuttujat. Vuetifyn takana on aktiivinen kehittäjäyhteisö, mikä takaa jatkuvat päivitykset ja tarvittaessa nopean avunsaannin. [19.]

MongoDB on avoimen lähdekoodin ei-relationaalinen tietokanta, jonne data tallennetaan BSON-dokumentteina (*Binary JSON*) ja jossa se esitetään JSON-dokumentteina. MongoDB:ssä on käytössä hieman erilainen termistö MySQL:ään verrattuna. MySQL:stä tuttu *taulu* (*table*) on MongoDB:ssä *kokoelma* (*collection*). Lisäksi *riviä* (*row*) kutsutaan *dokumentiksi* (*document*) ja *kolumnia* (*column*) *kentäksi* (*field*). Mon-

goDB:ssä dokumentit voivat vaihdella vapaasti, joten tietokantarakennetta ei tarvitse erikseen määritellä. Dokumenttiin lisättävä uusi kenttä voidaan siis luoda ilman, että se vaikuttaa muihin kokoelman dokumentteihin. Vaihtoehtoisesti voidaan käyttää validoitua tietokantamallia, jos halutaan varmistua jokaisen kokoelman sisällöstä. MongoDB:n dokumenttidatamalli (*document data model*) viittaa suoraan sovelluskoodin olioihin, mikä tekee tietokannasta helpon käyttää. Dokumentit tarjoavat mahdollisuuden esittää datan relaatiot hierarkkisessa järjestyksessä, joten taulukoiden ja muiden monimutkaisten rakenteiden tallennus käy helposti sellaisenaan. [20.]

MongoDB sopii hyvin sovelluksen tietokannaksi, sillä sovelluksessa on noudatettu olio-orientoitunutta työskentelytapaa. Datan tallennus ja haku on nopeaa, koska se on samassa muodossa niin sovelluksessa itsessään kuin tietokannassakin.

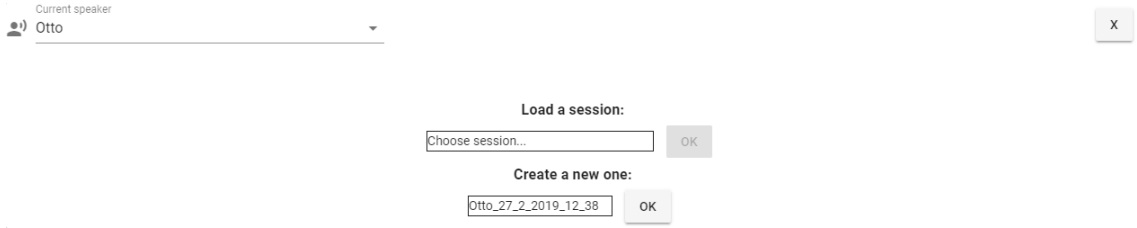
4 Sovelluksen toteutus

Insinöörityönä kehitettyä sovellusta lähdettiin toteuttamaan osittain Bitville Oy:n yksityisen Course Builder -sovelluksen yhteyteen, jotta joitain sen valmiita ominaisuuksia voitaisiin hyödyntää kehityksessä. Course Builderissa on Microsoft PowerPoint -tyyppinen sivunvaihto ja tekstikenttä, johon voi kirjoittaa ja tallentaa tekstiä sivukohtaisesti. Ajatuksena oli korvata Course Builderin sivupohja uudella näkymällä aina, kun halutaan tehdä äänityksiä. Ajatus mahdollisti sivunvaihdon ja vaihtuvan tekstikentän käyttämisen sovelluksessa.

Valmiina oli myös Python-palvelimella suoritettava Tacotron 2 -malli, jolla harjoitettiin suomenkielistä puhesynteesiä. Tarkoituksena oli saada palvelin kommunikoimaan käyttöliittymän kanssa siten, että harjoittamiseen liittyvää dataa voidaan lähettää molempiin suuntiin.

4.1 Kirjautuminen

Ennen kuin voidaan tehdä äänityksiä, ääninäyttelijän tulee olla valittuna ja session luotuna tai ladattuna. Molemmat tiedot tallennetaan Vuexin tilaan, mistä niitä voidaan käyttää ympäri sovellusta. Kun äänitysikkuna avataan, tarkistetaan aivan ensimmäisenä ääninäyttelijän ja session tila. Jos ääninäyttelijä ja sessio eivät löydy tilasta, avataan kirjautumispaneeli (ks. kuva 10). Koska sovellus menee yksityiseen käyttöön ja Course Builderissa on kirjautuminen jo pakollista, ääninäyttelijän nimi valitaan valmiista listasta erillisen kirjautumisen sijaan. Ääninäyttelijälista toteutettiin Vuetifyn v-select-komponentilla. Kun ääninäyttelijän nimi on valittu, tehdään tietokantahaku henkilön sessioille. (Tietokannan toiminnasta kerrotaan luvussa 4.3) Ladattujen sessioiden nimet listataan pudotusvalikossa ”Load a session:” -otsikon alla. Samalla generoidaan automaattisesti nimi uudelle sessiolle. Nimi sisältää ääninäyttelijän nimen lisäksi päivämäärän ja kellonajan. Käyttäjä voi myös halutessaan vaihtaa uuden session nimen mieleisekseen. Kirjautuminen on estetty, kunnes ääninäyttelijän nimi on valittu. Myös session nimi tulee olla valittuna, jos ollaan lataamassa vanhaa sessiota.



Kuva 10. Äänitystyökalun kirjautumispaneeli.

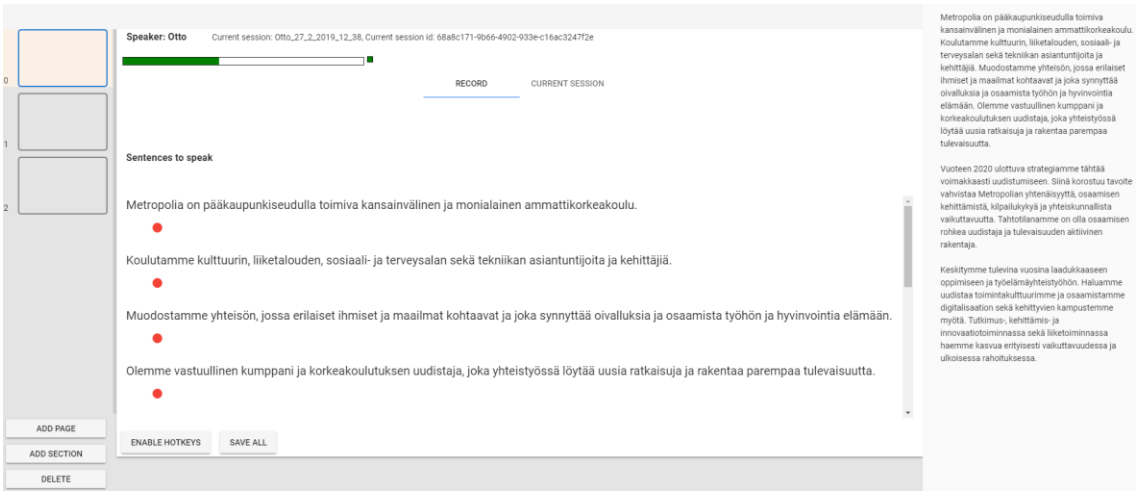
4.2 Äänitysprosessi ja äänitteenhallinta

Ensimmäinen asia kirjautumisen jälkeen on tekstikentän tekstin pilkkominen äänitettäviksi lauseiksi. Aluksi pilkotaan teksti pisteen perusteella lauseiksi, jotka sitten tallennetaan taulukkoon. Seuraavaksi tehdään silmukka, jossa käydään lauseet läpi. Jokaisen lauseen edestä poistetaan välilyönti, jos sellainen on. (Oikeinkirjoitussääntöjen mukaan useita lauseita sisältävissä teksteissä pisteen ja uuden lauseen erottaa välilyönti.) Lopulta luodaan uusi ”Narration”-niminen luokka, jonne tallennetaan kielikohtaisesti lauseet, luokan sisältävän sivun id ja luokan oma, uniikki id. Narration-luokan konstruktorissa jokaista lausetta varten luodaan Sentence-luokka, johon asetetaan lause ja referenssinä Narration-luokan id. (Jokaisen uniikisti generoidun id:n luontiin käytetään ”uuid”-nimistä npm-pakettia, joka luo satunnaisia 32-merkkisiä merkkijonoja [21].) Jokainen lause-olio saa Sentence-luokasta äänitykseen tarvittavia property-muuttujia, joihin tullaan muun muassa tallentamaan äänitiedosto ja sen kesto.

Narration-luokka tallennetaan olioksi jo aiemmin Course Builderissa luotuun sivu-olioon. Yhtä sivua kohden voi olla vain yksi Narration-luokka. Jos luokka on jollain sivulla jo olemassa, luodaan sivulle mentäessä vain uudet Sentence-luokat ja päivitetään Narration-luokan sisältö.

Jokaisella sivulla siis on tai tulee olemaan oma Narration-olio, joka pitää sisällään datan lauseiden äänitystä varten. Aina kun vaihdetaan sivua, haetaan aktiivisen Narration-olion sisältö sivulle. Tämä mahdollistaa esimerkiksi nopean tekstinmuokkauksen: jos tekstissä (kuvan 11 oikea laita) on kirjoitusvirhe, se voidaan korjata, ja sivua edestakaisin vaihdettaessa virhe on korjaantunut myös lause-olioon.

Kuvassa 11 on äänityssivun näkymä. Ylälaidassa on äänitykseen liittyvää perustietoa: ääninäyttelijän nimi, aktiivinen sessio ja session id. Tiedot haetaan Vuexin tilasta. Niiden alapuolella on liian kovasta sisääntulosignaalista varoittava klippimittari ja kaksi välilehteä, joista ensimmäinen on avoinna oleva ”Record”-ikkuna ja toinen aktiivisen session tarkastelun mahdollistava ”Current session”. Oikealla on vapaasti muokattava tekstikenttä. Jokaisella vasemmalla vaihdettavasta sivusta on oma tekstikenttä, ja sivua vaihdettaessa tekstikentän lisäksi vaihtuvat myös näkymän keskelle pilkotut lauseet. Lauseiden alapuolella on pikanäppäimet aktivoiva ”Enable hotkeys” -nappi ja äänitteiden tallennusta nopeuttava ”Save all” -nappi.



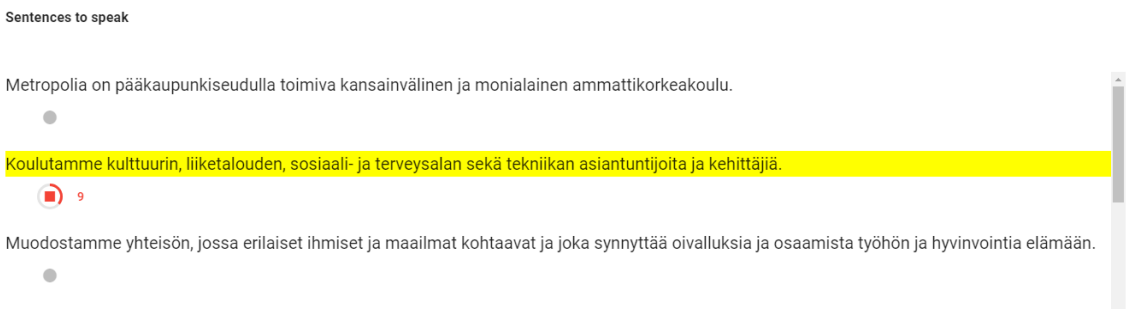
Kuva 11. Äänityssivu.

Klippimittarin pohjana toimii *volume-meter*-niminen Github-projekti [22]. Koodia muokattiin siten, että se soveltui Vue-komponentissa käytettäväksi. Siitä tehtiin luokkapohjainen ja toimintaperiaatetta muutettiin paremmin signaalia esittäväksi. Mittarin viereen tehtiin pieni vihreä ruutu, joka muuttuu punaiseksi äänisignaalin klipatessa. Ruudusta painamalla mittari ”nollataan” vaihtamalla väri takaisin vihreäksi.

Koska äänitykseen tehty koodi on toteutettu olio-ohjelmointina, jokainen lause on helposti muokattavissa yksilöllisesti niille annettujen property-muuttujien eli ominaisuuksien avulla. Lauseet renderöidään iteroimalla lause-oliot, joten ominaisuuksiin on helppo pääsy. If-lauseen avulla voidaan esimerkkikoodin 1 mukaisesti tuoda halutut asiat näkyviin ominaisuukisien tilasta riippuen.

Kun äänitysnappia painetaan, äänittämisestä vastaavalle *recordAudio()*-funktiolle annetaan parametrina äänitettävän lauseen id. Id:n avulla etsitään kyseinen lause-olio ja tallennetaan se muuttujaan. Seuraavaksi muutetaan Vuexin tilaan lauseen *isRecording*-muuttuja todeksi. Suodattamalla lause-oliot kyseisen ominaisuuden osalta saadaan äänitettävälle lauseelle käyttöliittymään näkyviin äänitystä informoiva korostus ja pysäytysnappi (ks. kuva 12). Samalla lukitaan muiden lauseiden äänityspainike.

Koska yhden äänityksen enimmäispituus on 14 sekuntia, tarvitaan ajastinta näyttämään jäljellä oleva aika ja lopettamaan äänitys, jos aikaraja umpeutuu. Ajastimen toteuttamiseen käytettiin Vuetifyn *v-progress-circular*-komponenttia (ks. kuva 12), joka visualisoi ajan kulkua käyttäjälle. Lause-oliolle luotiin *recordingTimecode*-muuttuja, jota jatkuvasti päivitetään uudella millisekuntiarvolla niin kauan, kuin äänitys on käynnissä. Arvo annetaan komponentille, joka päivittää itse itsensä aina arvon muuttuessa. Sekunnit saadaan laskettua jakamalla millisekunnit tuhannella ja pyöristämällä vastaus sekuntiin, joka on pienempi tai yhtäsuuri kuin alkuarvo.



Kuva 12. Äänitettävä lause.

Selainpohjaisen äänityksen voidaan katsoa koostuvan kuudesta eri vaiheesta, jotka ovat äänityksen aloittaminen, äänidatan keräys, äänityksen lopetus, äänidatan muuntaminen binäärimuotoon, URL-osoitteen luonti binäärimuotoisesta äänidatasta ja äänitteen toisto.

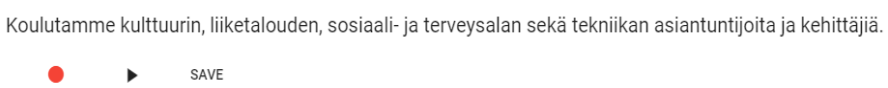
Äänityksen aloittamiseksi aletaan kuunnella mikrofonin lähettämää dataa kutsumalla funktiota *navigator.mediaDevices.getUserMedia* ja antamalla sille parametriksi ”audio: true”. (MediaDevices on vapaasti käytettävä rajapinta, joka mahdollistaa mikrofonin lisäksi pääsyn myös muihin tietokoneeseen liitettyihin medialaitteisiin, kuten esimerkiksi web-kameroihin.) Funktio *getUserMedia* palauttaa lupauksen (*promise*) ja tuo mikrofo-

nin äänidatan käytettäväksi. Seuraavaksi luodaan MediaRecorder-olio, jonka konstruktoriin äänidata annetaan. Äänitys käynnistyy, kun kutsutaan MediaRecorderin *start*-metodia.

Tässä vaiheessa äänitys on käynnissä, mutta sen tuottamaa dataa ei tallenneta. Datan tallentamista varten kuunnellaan MediaRecorderin ”dataavailable”-tapahtumaa, ja aina tapahtuman lauettua työnnetään uutta dataa taulukkoon säilytettäväksi. Tämä tapahtuma laukaistaan oletuksena silloin, kun MediaRecorderin *stop*-metodia kutsutaan. MediaRecorderin *start*-metodille voidaan kuitenkin antaa parametrinä millisekunteina aika, kuinka usein ”dataavailable” halutaan laukaistavan. Parametriksi annettiin 1000, eli yksi sekunti. Yhden sekunnin päivitysväli antaa mahdollisuuden seurata kulunutta aikaa, ja sen avulla voidaan helposti pysäyttää äänitys, jos se ylittää 14 sekunnin rajan. Samalla myös tallennetaan dataa audiostreamista joka sekunti.

Äänitys pysäytetään painamalla pysäytysnappia, joka kutsuu MediaRecorderin *stop*-metodia. Se laukaisee ”stop”-eventin ja lopettaa äänittämisen. ”Stop”-eventissä otetaan ensin talteen äänityksen kesto millisekunteina, minkä jälkeen muutetaan äänitetyn lause-olion isRecording-muuttuja epätosiksi Vuexin tilassa. Vuen reaktiivisuuden ansiosta näkymä päivittyy suoraan käyttöliittymään. Pysäytysnappi muuttuu takaisin äänitysnapiksi, ja lukitut äänitysnapit ovat jälleen käytössä.

Seuraavaksi käsitellään talteenotettu äänidata. Datasta luodaan uusi Blob-olio, joka on eräänlainen raakadataa sisältävä tiedostomuotoinen olio. Parametriksi oliolle annetaan ”type: ’audio/webm’”, jotta tiedosto saadaan webm-muotoon. Jotta audiota voidaan myöhemmin toistaa, Blob annetaan parametrinä URL-osoitteen luovalle *URL.createObjectURL*-metodille. Kaikki äänityksen pysäytyksen jälkeen luodut muuttujat tallennetaan väliaikaisesti tilassa olevaan lause-olioon. Olioon tallennetaan myös tieto siitä, onko lauseesta olemassa äänite. Jos äänite löytyy, lauseen äänitysnapin viereen tuodaan näkyviin toisto- ja tallennusnappi (ks. kuva 13).



Kuva 13. Lause äänityksen jälkeen.

Toistonappi luotiin käyttäen HTML:n klassista audio-elementtiä. Sillä on attribuutti ”src” (lähde), johon kuuluu asettaa toistettava äänitiedosto. Tässä tapauksessa lähteeksi annetaan Blob-tiedostosta luotu URL-osoite.

Ajatus on, että äänitteet eivät tallennu tietokantaan automaattisesti nauhoituksen jälkeen, vaan tallennus täytyy tehdä manuaalisesti mahdollisen esikuuntelun jälkeen. Tämä auttaa osaltaan varmistumaan äänitiedostojen sisällöstä; halutaan olla varmoja siitä, että tallennetaan vain hyvälaatuiset äänitteet. Kun äänitteen ”Save”-nappia painetaan, luodaan siitä uusi File-objekti. Tiedostonimeksi generoidaan uniikki id. Tämän jälkeen luodaan uusi FormData-olio, jolle annetaan äänitiedoston lisäksi äänitteeseen liittyviä ominaisuuksia, kuten juuri luotu tiedostonimi, sivun id, session nimi ja id, käyttäjän id, äänitteen pituus, äänitteessä puhuttu lause tekstinä sekä ääninäyttelijän nimi. Tässä vaiheessa vaihdetaan lause-olion isSaving-muuttujan arvo trueksi, mikä muuttaa äänitteen tallennusnapin pyöriväksi latauskuvakkeeksi. Muutos paitsi informoi käyttäjää meneillään olevasta tallennuksesta myös estää tallennusnapin painamisen monta kertaa peräkkäin. Seuraavaksi FormData-olio lähetetään XMLHttpRequest-pyynnöllä node-palvelimelle tallennettavaksi tietokantaan. (Tietokannan toiminnasta kerrotaan tarkemmin luvussa 4.3) Onnistuneen tietokantakutsun jälkeen latauskuvake vaihdetaan vihreäksi ”Saved”-tekstiksi vaihtamalla isSaving-muuttuja takaisin epätodeksi ja isSaved-muuttuja puolestaan todeksi. Nappien näkyvyys perustuu siis lause-olion muuttujien arvoihin esimerkkikoodin 1 mukaisesti. Jos tietokantakutsu jostain syystä epäonnistuu, vaihdetaan latauskuvake takaisin tavalliseksi tallennusnapiksi ja näytetään käyttäjälle toast-ilmoitus epäonnistumisen syystä.

```
<td v-if="sentence.hasAudio">
     <h3
          v-if="sentence.isSaved && !sentence.isRecording"
          style="color: green; margin-left: 0.5em;"
     >Saved</h3>
     <v-btn
          v-else-if="!isSaving"
          :disabled="someSentenceRecording"
          flat
          @click="saveAudioAsWav(sentence)"
          style
     >Save</v-btn>

     <v-btn v-else flat loading>Save</v-btn>
 </td>
```

Esimerkkikoodi 1. Äänitteen tallennusnappi renderöidään sen ominaisuuksien perusteella if-lauseen avulla. Koodi sijaitsee Vue-komponentin template-osiossa.

”Save all” -nappi kehitettiin äänitystyön nopeuttamiseksi, koska se mahdollistaa yhdellä napinpainalluksella useiden äänitteiden tallentamisen kerralla. Kun nappia painetaan, filtteröidään lause-olioista ne lauseet, jotka on äänitetty, mutta ei vielä tallennettu. Sitten lause-oliot iteroidaan ja tallennetaan lause kerrallaan tietokantaan.

”Enable hotkeys” -nappi aktivoi näppäimistön nuolinäppäimet vaivattomamman ja nopeamman äänityksen mahdollistamiseksi. Nappia painamalla aktivoidaan kuuntelija, joka seuraa käyttäjän näppäinpainalluksia. Kuuntelija reagoi silloin, kun painetaan jotain neljästä nuolinäppäimestä. ”Nuoli ylös”- ja ”nuoli alas” -näppäimiä käytetään lauseiden selaamisessa. Jokaisella lause-oliolla on ”isActive”-muuttuja, jonka ollessa tosi lause on aktiivinen. Vain yksi lause voi olla kerrallaan aktiivinen. Aktiivinen lause ympäröidään yksinkertaisella CSS:llä toteutetulla mustalla ulkoreunalla, ja se tulee näkyviin silloin, kun pikanäppäimet on aktivoitu. Aktiivinen lause on oletuksena aina ensimmäinen, ja indeksi on nolla. Kun painetaan ”nuoli alas” -näppäintä, lisätään aktiivisen lauseen indeksiä säilövään muuttujaan yksi. Sitten vaihdetaan kaikkien lauseiden ”isActive”-muuttuja epätodeksi, minkä jälkeen haetaan uusi aktiivinen lause indeksin perusteella ja vaihdetaan sen ”isActive”-muuttuja todeksi. Samat toimenpiteet tehdään myös ”nuoli ylös” -näppäintä painettaessa, mutta silloin indeksistä vähennetään yksi. Aina ennen jompaankumpaan suuntaan siirtymistä tarkistetaan, että indeksille on olemassa lause. Näin säästytään virheilmoituksilta silloin, kun painetaan ”nuoli ylös” -näppäintä ensimmäisen lauseen ollessa aktiivinen tai ”nuoli alas” -näppäintä viimeisen lauseen ollessa aktiivinen.

”Nuoli oikealle” -näppäin tarkistaa ensin, onko jonkin lauseen ”isRecording”-muuttuja tosi eli onko äänitys käynnissä. Sitten asetetaan Boolean-tyyppisen ”autoRecord”-muuttujan arvoksi tosi. Tämän muuttujan arvosta voidaan äänitysvaiheessa päätellä, halutaanko siirtyä äänityksestä toiseen automaattisesti. Jos ”nuoli oikealle” -näppäintä painettaessa äänitys ei ole käynnissä, aloitetaan aktiivisen lauseen äänitys. Jos äänitys puolestaan on käynnissä, lopetetaan lauseen äänitys, tallennetaan se esikuuntelua varten ja aloitetaan suoraan seuraavan lauseen äänitys. Aina ennen seuraavaan lauseeseen siirtymistä tarkistetaan, onko seuraavaa lausetta olemassa. Jos sitä ei löydy, automaattinen siirtyminen estetään. Sama kaava toistuu niin pitkään, kunnes viimeisen lauseen äänitys on suoritettu.

”Nuoli vasemmalle” -näppäintä painetaan silloin, kun halutaan peruuttaa parhaillaan äänityksessä oleva lause. Se muuttaa ”cancelCurrentRecord”-nimisen Boolean-muuttujan todeksi ja lopettaa äänityksen. Tavallisesti äänitys tallennetaan automaattisesti silloin, kun äänitys pysäytetään, mutta ”cancelCurrentRecord”-muuttujan ollessa tosi tallennusta ei suoriteta. Äänityksen keskeyttämisen jälkeen käyttäjä on vapaa aloittamaan saman lauseen äänittämisen uudestaan tai vaihtamaan halutessaan aktiivista lausetta.

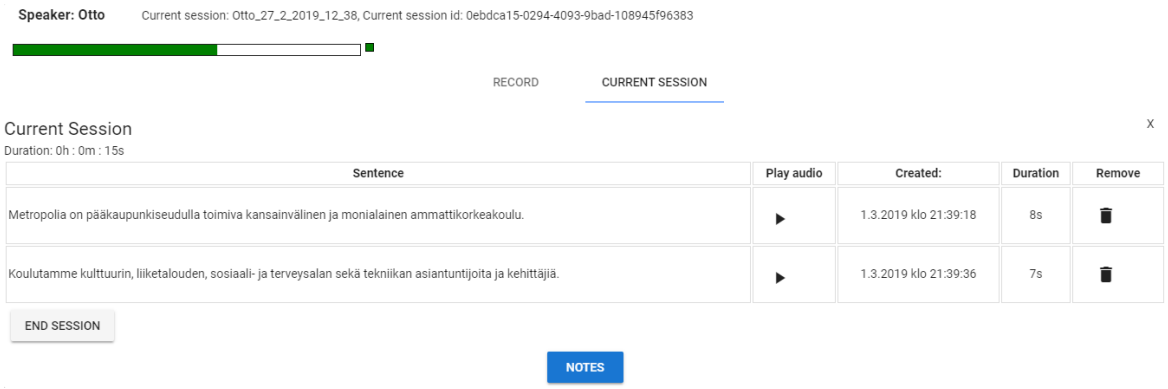
Kaikki lauseet eivät mahdu kerrallaan näkyviin, jos niitä on paljon yhdelle sivulle. Siksi pikanäppäimillä äänitettäessä näkymän halutaan vierittyvän automaattisesti siten, että aktiivinen lause on aina näkyvissä. Ominaisuuden toteuttamiseen käytettiin *vue-scrollto*-nimistä npm-pakettia [23]. Kun aktiivista lausetta vaihdetaan, kirjasto keskittää näkymän annetun DOM-elementin perusteella.

Current Session -välilehdessä käyttäjä voi tarkastella aktiivisen session sisältöä (ks. kuva 14). Aina muutoksen, esimerkiksi sivunvaihdon tai session latauksen, yhteydessä tietokannasta haetaan aktiivisen session data. Äänitteen toistamista varten äänilähteeksi annetaan polku palvelinkoneen äänitteitä säilövään kansioon ja tiedostonimeksi äänite-olioon tallennettu nimiviittaus. Äänitteen luomisaika ja kesto on tallennettu äänite-olioon, josta ne haetaan näytettäväksi. Äänite poistetaan lähettämällä ääninäyttelijän nimi sekä session ja äänitteen id:t node-palvelimelle, jossa tietojen avulla tietokannasta haetaan oikea kenttä poistettavaksi. Vastauksena lähetetään päivitetty taulukko sessio-olioista. Oliot tallennetaan Vuexin tilaan, mikä päivittää näkymän ja pyyhkii poistetun äänitteen näkyvistä.

”End Session” -nappi lopettaa käynnissä olevan session. Käytännössä nappia painamalla tilassa säilytetyt muuttujat *currentSpeaker*, *currentSessionName*, *currentSessionId* ja *currentSession* nollataan muuttamalla kolmen ensimmäisen arvo tyhjäksi merkkijonoksi ja viimeinen tyhjäksi olioksi. Tämä vie käyttäjän takaisin kirjautumissivulle. Session lopettaminen hävittää äänityssivun tallentamattomat lauseet, minkä toisaalta tekee myös pelkkä sivunvaihto.

Mustiinpanojen (*Notes*) toteuttamista varten luotiin napista aukeava ponnahdusikkuna. Toteutuksessa käytettiin Vuetifyn v-dialog-komponenttia. Komponentin sisällä on tekstikenttä sekä peruutus- ja tallennusnappi. Tekstikentän sisältöä hallitaan reaktiivisella v-

model-attribuutilla, jolle annetaan muistiinpanot sisältävä muuttuja. Käyttäjän kirjoittamat muistiinpanot tallennetaan tietokantaan aktiivisen session alle. Tallennuksen jälkeen session tiedot haetaan tietokannasta uudestaan, jotta päivitetyt muistiinpanot näkyvät käyttäjälle ilman, että sivua tarvitsee erikseen päivittää.



Kuva 14. Näkymä session tarkastelua varten.

4.3 Tietokanta

Sovelluksen käyttämäksi tietokannaksi valittiin MongoDB, jota suoritetaan node-palvelimella. Mongoose.js:n avulla luodaan olio-mallisia tietokantarakenteita palvelimella. Käytössä on myös Axios http-yhteyttä varten sekä Express.js, joka on Noden web-palvelinpaketti.

Kun tietokantayhteys on luotu, määritetään rakenne tietokantaan menevälle datalle. Sitä varten luodaan uusi *mongoose.Schema*-olio, johon asetetaan kaikki tarvittavat muuttujat. Esimerkkikoodissa 2 kuvataan olion rakennemalli. Jokaista ääninäyttelijää kohden luodaan vain yksi dokumentti, eli tässä tapauksessa kyseinen malli. Dokumentti sisältää ääninäyttelijän nimen ja sessiot, jotka tallennetaan olioina taulukkomuotoon. Sessio-oliolla on id, session oma nimi, ääninäyttelijän nimi, muistiinpanot, äänitteet ja tieto siitä, onko sessio datasetissä. Äänitteet ovat myös taulukkomuodossa. Äänite-olio sisältää tiedostonimen (*fileId*), sivun id:n, äänitetyn lauseen tekstinä sekä äänitteen keston ja luomisajan. MongoDB ei varsinaisesti vaadi ennalta määritettyä tietokantarakennetta, mutta sen olemassaolo varmistaa, ettei tietokantaan tallennu väärää dataa tai oikeaa dataa väärässä rakenteellisessa muodossa. Jokaiselle ääninäyttelijälle tulisi generoida oma id, jonka avulla oikea ääninäyttelijä haetaan tietokannasta. Tässä ta-

pauksessa, kun sovellus menee yrityksen sisäiseen käyttöön ja ääninäyttelijöitä on rajallinen määrä, tietokantakutsut tehdään ääninäyttelijän nimen perusteella. Tietoturvauhkien välttämiseksi jokaisessa tietokantakutsussa varmistetaan, että käyttäjä on todennettu.

```
const SpeakerSchema = new Schema({
      name: String,
      sessions: [
            {
                  id: String,
                  name: String,
                  speaker: String,
                  isInDataset: Boolean,
                  notes: String,
                  recordings: [
                        {
                           fileId: String,
                           pageId: String,
                           sentence: String,
                           duration: Number,
                           createdAt: Date,
                        }
                  ]
            }
      ]
});
```

Esimerkkikoodi 2. Ääninäyttelijän luoman datan tietokantarakenne.

Ensimmäisen kerran tietokantaan tallennetaan dataa siinä vaiheessa, kun nauhoitettu äänite tallennetaan. Äänitteestä ja siihen liittyvistä ominaisuuksista kootaan uusi FormData-olio, joka lähetetään XMLHttpRequest-pyynnöllä node-palvelimelle. Node-päätepisteessä (*endpoint)* äänitiedosto tallennetaan webm-muotoon *file-system*-JavaScript-kirjaston writeFile-metodilla. Tätä webm-tiedostoa käytetään esikuuntelussa, eikä sitä tallenneta tietokantaan. Sen sijaan heti webm-tiedoston luomisen jälkeen äänitteestä luodaan wav-tiedosto käyttäen hyväksi FFmpeg-kirjastoa. Tiedostosta tehdään mono-tallenne, ja taajuudeksi asetetaan 22 050 hertsiä. Lopuksi wav-tiedosto tallennetaan palvelinkoneelle.

Seuraavaksi tietokannasta haetaan sen ääninäyttelijän tietoja, jonka luomaa äänitettä ollaan tallentamassa. Jos ääninäyttelijältä ei löydy aikaisemmin tallennettuja äänitteitä, hänelle luodaan uusi dokumentti, johon tallennetaan esimerkkikoodin 2 mukaiset tiedot. Jos ääninäyttelijällä on jo ennaltaan tallennettuja äänityksiä, tarkistetaan aluksi, onko kyseistä sessiota olemassa. Jos sessio on jo olemassa, luodaan olio äänitteelle ja sen ominaisuuksille ja lisätään se olemassa olevan session ”recordings”-taulukkoon. Jos

sessio on uusi, luodaan uusi sessio-olio ja lisätään se dokumentin ”sessions”-taulukkoon.

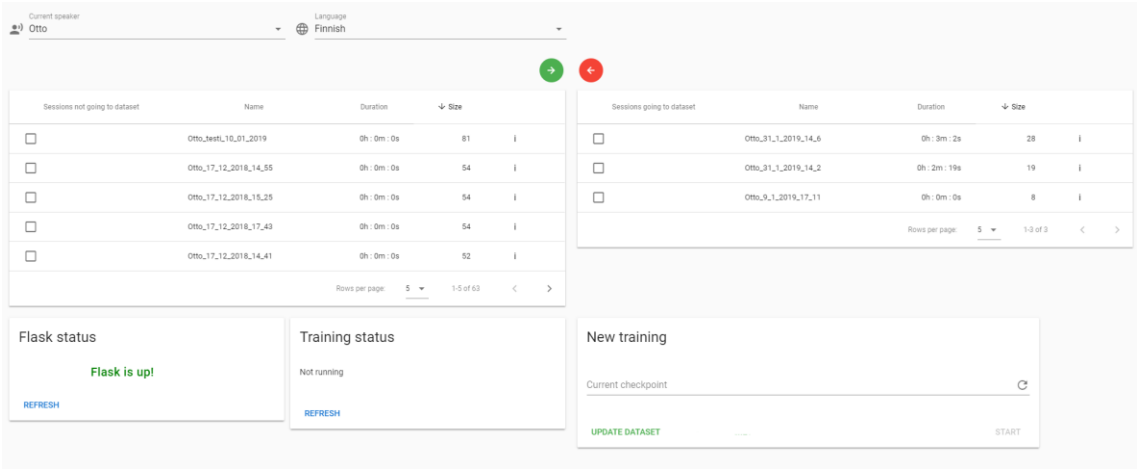
Tietokantaan ei tallenneta itse äänitiedostoa, vaan ainoastaan tiedoston nimi tekstimuodossa. Äänitteen luomisajankohtaa esittävä ”createdAt”-ominaisuus saadaan luomalla uusi Date-olio. Datasettiin sisältymistä kuvaava ”isInDataset”-ominaisuus on oletuksena epätosi. Lopuksi luotu tai päivitetty dokumentti tallennetaan ja vastauksena lähetetään tiedostonimi ja onnistunutta tiedoston tallennusta esittävä viesti takaisin käyttöliittymään. Epäonnistuneessa tietokantakutsussa takaisin lähetetään HTTP-tilakoodi 500 (*Internal Server Error*) ja viesti ”Server error”.

Aina kun kirjaudutaan äänityssivulle lataamalla olemassa oleva sessio, tietokannasta haetaan kaikki kyseisen session tiedot ääninäyttelijän nimen ja valitun session id:n perusteella. Haetut tiedot asetetaan Current Session -välilehdessä olevaan tauluun (ks. kuva 14). Jokaisella tallennetulla lauseella on painike, josta äänitteen voi poistaa. Kun äänite poistetaan, ääninäyttelijän nimi sekä session ja äänitteen id:t lähetetään node-palvelimelle. Ensin haetaan oikea dokumentti nimen perusteella, minkä jälkeen selvitetään poistettavan äänitteen indeksi käyttämällä hyväksi session ja äänitteen id:tä. Äänite poistetaan Array-olion splice-metodilla ja dokumentti tallennetaan. Onnistuneen poiston merkiksi vastauksena lähetetään päivitetty sessio ja onnistumista kuvaava viesti. Poiston epäonnistuessa lähetetään HTTP-tilakoodi 500 ja epäonnistumista kuvaava viesti. Samalla sivulle kehitettiin myös painike session poistoa varten, mutta sen katsottiin olevan liian vaarallinen suuren datamäärän tahattoman menettämisen kannalta, joten se ei ole käytössä.

Muistiinpanojen tallennusta varten tarvitaan ääninäyttelijän nimen lisäksi session id ja muistiinpanot String-muodossa. Nimen ja id:n perusteella tietokannasta haetaan oikea sessio, jonka *notes*-muuttuja korvataan uusilla muistiinpanoilla. Lopuksi päivitetty dokumentti tallennetaan. Vastauksena lähetetään tietokantakutsun onnistumisen perusteella oikea HTTP-tilakoodi ja viesti.

4.4 Datasetin hallinta

Datasetin muokkausta varten luotiin uusi sivu. Suunnitelmana oli, että sessiot listataan vierekkäin siten, että sivun vasemmalla puolella ovat datasettiin kuulumattomat ja oikealla datasettiin menevät sessiot. Kuva 15 esittää toteutetun sivun. Ensin käyttäjän tulee valita ääninäyttelijän nimi, minkä jälkeen tietokannasta haetaan halutun ääninäyttelijän sessiot. Sessiot tallennetaan Vuexin tilaan, josta niitä voidaan renderöidä sivulle reaktiivisesti. Jokaisella sessiolla on ”isInDataset” -niminen Boolean-muuttuja, jonka perusteella sessiot suodatetaan oikeaan listaan. Muuttujan oletusarvo on epätosi, mikä asettaa kaikki sessiot datasettiin kuulumattomien listalle.



Kuva 15. Hallintasivu datasetin muokkausta ja harjoittamisen käynnistämistä varten.

Listat ovat Vuetifyn ”Data table” -komponentteja. Komponentissa on valmiina ominaisuuksina sessioiden järjestely sekä mahdollisuus päättää, montako sessiota näkyy yhtä sivua kohden. Jokaisesta sessiosta näytetään nimi, äänitteiden kokonaiskesto ja äänitteiden määrä. Tiedot on tallennettu sessio-olion muuttujiin. Lisäksi rivien oikeassa laidassa olevasta ”i” (*info*) -napista aukeavat sessiosta kirjoitetut muistiinpanot.

Sessioilla on vasemmassa laidassa valintaruudut, joita painamalla halutut sessiot saadaan valittua. Valitut sessiot lisätään array-listaan. Kun painetaan vihreää ”nuoli oikealle” -nappia, suodatetaan valittujen sessioiden listasta jokainen sessio-olio, jonka ”isInDataset”-muuttuja on epätosi. Sitten muutetut sessio-oliot tallennetaan tietokantaan, minkä jälkeen ne asetetaan tilaan. Valitut sessiot siirtyvät uuteen listaan välittömästi. Toimenpide sessioiden siirtämisessä datasettiin kuulumattomien listaan on sama sillä

erotuksella, että valittujen sessioiden taulukosta suodatetaan ne sessio-oliot, joiden ”isInDataset”-muuttuja on tosi, minkä jälkeen ne muutetaan epätosiksi.

Muistiinpanojen näyttämistä varten luotiin Vuetifyn ”Dialog”-komponentti, joka aukeaa näkyväksi ”i”-nappia klikkaamalla. Klikkaus muuttaa sivu-komponentissa määritellyn ”dialog”-Boolean-muuttujan todeksi, mikä avaa ikkunan päällimmäiseksi. Komponentin sisälle luotiin tekstikenttä, johon liitetään session muistiinpanot. Muistiinpanot tuodaan Vuexin tilasta ja asetetaan v-model-attribuuttiin, jonka käyttö mahdollistaa sen, että muokattu teksti päivittyy jatkuvasti tilaan. Tässä tapauksessa tekstiä ei kuitenkaan haluta päivitettävän tilaan automaattisesti, jotta muistiinpanojen muokkaus voidaan peruuttaa. Vasta kun käyttäjä tallentaa muistiinpanot, ne tallentuvat tietokantaan ja päivitetetyt sessio-oliot asetetaan tilaan.

Kun datasettiin menevät äänitteet on valittu, painetaan ”Update Dataset”-nappia. Ääninäyttelijän nimi lähetetään node-palvelimelle, missä kaikki käyttäjän sessiot haetaan tietokannasta. Ensin sessio-olioista suodatetaan ne, joiden ”isInDataset”-muuttuja on tosi. Suodatettuja sessio-olioita iteroivan silmukan sisään tehdään toinen silmukka, joka iteroi sessiolle kuuluvat äänite-oliot. Jokaisesta äänite-oliosta otetaan talteen äänitiedoston id (sama kun tiedostonimi) ja tekstimuotoinen lause. Lopputuloksena yhdessä taulukossa ovat kaikki äänite-oliot, jotka ovat menossa datasettiin.

Seuraavaksi äänite-oliot lähetetään axios-pyynnöllä Python-palvelimelle, missä varsinaista harjoittamista operoidaan. Siellä äänite-olioiden sisältämästä datasta kirjoitetaan tekstitiedosto, joka toimii datasettinä puhesynteesiä harjoitettaessa. Äänite-oliot iteroidaan esimerkkikoodin 3 mukaisesti. Datasetti tallennetaan palvelinkoneelle ääninäyttelijäkohtaisesti.

```
f = open(path, "w")
for recording in recordings:
     f.write("records/" + speaker + "/" + recording['fileId'] + ".waw|" +
     recording['sentence'] + "\n")
```

Esimerkkikoodi 3. Datasetiksi kutsuttavan tekstitiedoston kirjoitus Pythonilla. Muuttuja ”path” sisältää polun, jonne tiedosto tallennetaan. Jokainen lause on omalla rivillä. Tiedoston sijaintia ja lausetekstiä erottaa pystyviiva.

4.5 Oppimisen käynnistäminen käyttöliittymästä

Kuvan 15 (s. 36) kolme alimmaista paneelia liittyvät harjoittamisen hallintaan. Flask status -paneelin tarkoitus on viestiä käyttäjälle, onko Python-palvelin päällä. Palvelimen täytyy luonnollisesti olla päällä silloin, kun halutaan aloittaa harjoittaminen. Tarkistus tehdään aina ladattaessa sivu tai painettaessa ”Refresh”-nappia. Tarkistuksessa ensin paneeliin laitetaan pyörivä latauskuvake yksinkertaisen Boolean-muuttujan avulla. Latauskuvakkeena käytetään Vuetifyn v-progress-circular-komponenttia. Sitten tehdään axios-pyyntö node-palvelimelle, mistä lähetetään toinen pyyntö edelleen Python-palvelimelle. Jos Python-palvelimelta saadaan vastaus, tiedetään sen olevan päällä. Silloin paneeliin tuodaan näkyviin vihreä palvelimen päälläoloa kuvaava teksti. Jos palvelimelta ei saada vastausta, paneeliin tuodaan punainen teksti viestimään käyttäjälle, että palvelin on alhaalla.

Training status -paneelista käyttäjä näkee harjoittamiseen liittyvää informaatiota, kuten iteraatioiden määrän ja kokonaiskeston. Harjoittamisen myötä päivittyvä data tallennetaan toistuvasti tietokantaan, josta se haetaan käyttöliittymään sivun latautuessa tai ”Refresh”-napista.

Harjoittaminen käynnistetään New Training -paneelista. Ensin käyttäjän täytyy valita tallennuspiste. Tallennuspisteitä luodaan harjoittamisen aikana tasaisin väliajoin, ja ne tallennetaan ääninäyttelijäkohtaisesti Python-palvelimelle. Ne nimetään iteraatioiden määrän perusteella; nimiä voisivat olla esimerkiksi checkpoint_1000 ja checkpoint_1500. Harjoittaminen voidaan aloittaa alusta tai sitä voidaan jatkaa valitusta tallennuspisteestä. Tallennuspisteiden nimet halutaan listata pudotusvalikkoon. Haun suorittava funktio suoritetaan päivityskuvaketta painettaessa tai ääninäyttelijää vaihdettaessa sivun vasemmasta ylälaidasta. Haku alkaa axios-pyynnöllä node-palvelimelle parametrinaan ääninäyttelijän nimi. Sieltä tehdään axios-pyyntö Python-palvelimelle samalla parametrillä. Python-palvelimella luodaan uusi muuttuja (ks. esimerkkikoodi 4), johon listataan kaikki halutun ääninäyttelijän tallennuspistekansiosta löytyneet tiedostonimet. Muuttuja tuodaan node-palvelimen kautta takaisin axios-pyynnön lähettäneeseen funktioon, missä siihen lisätään uusi nimi, checkpoint_0. Lisätyn nimen tarkoituksena on mahdollistaa harjoittamisen käynnistäminen nollasta. (Jokin tallennuspisteistä täytyy valita, jotta harjoittaminen voidaan aloittaa.) Lopuksi tiedostonimet sisältävä taulukko laitetaan aakkos- ja numerojärjestykseen. Taulukko annetaan items-attribuuttina

Vuetifyn v-select-komponentille, joka on pudotusvalikko. Vuetifyn komponentit tarjoavat reaktiivisuutta, joten tiedostonimet päivittyvät pudotusvalikkoon automaattisesti.

```
checkpoints = [c for c in
listdir('./speechengine/saved_models/taco_checkpoints' +
str(request.args['speaker']) + '/')]

return jsonify(checkpoints)
```

Esimerkkikoodi 4. Tiedostonimien haku ääninäyttelijäkohtaisesta tallennuspistekansiosta.

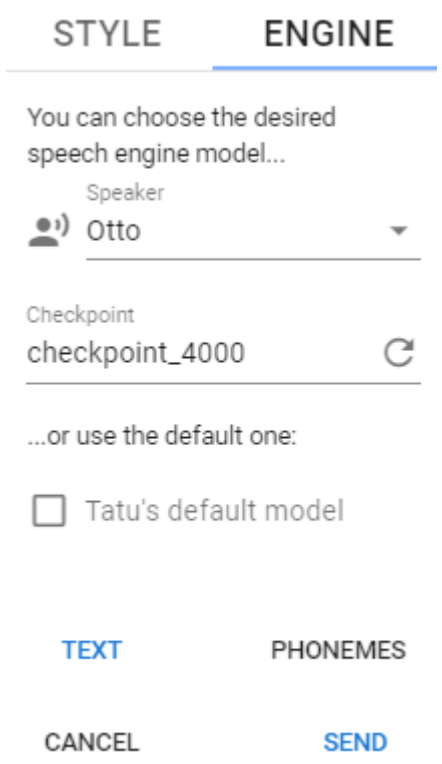
Harjoittamisen käynnistävä painike on näkyvissä, jos harjoittaminen ei ole käynnissä jo valmiiksi. Silloin näkyvissä on pysäytyspainike. Asia tarkistetaan etsimällä tietokannasta aktiivista dokumenttia, johon käynnissä olevan harjoittamisen tiedot tallennetaan. Dokumentissa on status ”In progress”, jos harjoittaminen on käynnissä. Kun harjoittaminen pysäytetään, status muutetaan eriksi, jotta tieto vapaasta palvelimesta on saatavilla. Pysäytyspainike on esillä ollessaan aina painettavissa, mutta käynnistypainike on lukittuna, jos tietyt ehdot eivät täyty. Jotta napista tulee aktiivinen, käyttäjän tulee valita ääninäyttelijä ja tallennuspiste. Harjoittamista ei voida aloittaa ilman näitä tietoja, sillä niiden perusteella määräytyvät käytettävä datasetti, tallennuspiste ja uusien tallennuspisteiden ääninäyttelijäkohtainen kohdekansio.

Kun käynnistysnappia painetaan, ensimmäisenä muutetaan isTrainingStarting-muuttuja todeksi. Muuttujan ollessa tosi painikkeen tilalle tulee käynnistymistä kuvaava teksti. Koska nappi menee heti piiloon, käyttäjä ei voi tehdä useaa käynnistystä peräkkäin, mikä sekoittaisi järjestelmän ja saattaisi aiheuttaa virheen. Seuraavaksi harjoittamisen vaatimat tiedot (eli ääninäyttelijä ja tallennuspiste) lähetetään node-palvelimelle. Sieltä tiedot lähetetään edelleen Python-palvelimelle, missä harjoittaminen aloitetaan. Kun harjoittaminen on alkanut, lähetetään vastaus node-palvelimen kautta takaisin käynnistysfunktioon. Käyttäjää informoidaan vihreällä ”Training started” -toast-viestillä. Samalla isTrainingStarting-muuttuja muutetaan takaisin epätodeksi, mikä tuo pysäytysnapin näkyviin. Lopuksi ajetaan funktio, joka hakee harjoittamiseen liittyvät tiedot Training status -paneeliin. Jos harjoittamisen käynnistäminen epäonnistuu esimerkiksi alhaalla olevan Python-palvelimen vuoksi, käyttäjälle näytetään silloin toast-viesti ”Failed to start training” ja isTrainingStarting-muuttujan arvo vaihdetaan epätodeksi, mikä tuo käynnistysnapin takaisin näkyviin.

Pysäytysnappia painettaessa isTrainingStopping-muuttujan arvo vaihdetaan ensin todeksi, mikä tuo näkyviin pysäyttämistä kuvaavan tekstin ja estää samalla napin toistuvan painamisen. Sitten harjoittaminen pysäytetään kutsumalla node-palvelimesta python-palvelimen tiettyä päätepistettä. Pysäytyksen onnistuttua näytetään toast-viesti ”Training finished” ja tyhjennetään harjoittamistietoa säilövä muuttuja. Samalla muutetaan Boolean-muuttujat isTrainingOn ja isTrainingStopping epätosiksi, mikä tuo käynnistysnapin takaisin näkyviin.

Harjoitettua puhesynteesimallia halutaan voida testata. Course Builder -sovelluksessa on mahdollista tuottaa kirjoitetusta tekstistä puhuttu versio antamalla harjoitettu tallennuspiste generaattoriin. Sitä varten generoinnin yhteyteen luotiin kuvan 16 esittämä näkymä, jossa käytettävä puhesynteesimalli valitaan. Näkymä mahdollistaa uuden mallin testaamisen nopeasti ilman, että mallia tarvitsee muuttaa käsin itse koodissa. Ääninäyttelijöiden nimet on tallennettu Vuexin tilaan, josta ne tuodaan Vuetifyn v-select-komponentin items-attribuuttiin. Komponentin v-modeliksi asetetaan chosenSpeaker-muuttuja, joka pitää sisällään valitun ääninäyttelijän. Aina, kun ääninäyttelijää vaihdetaan listasta, haetaan valitun ääninäyttelijän tallennuspisteet python-palvelimelta ja tuodaan niin ikään tallennuspisteitä listaavan v-select-komponentin items-attribuuttiin. Sen komponentin v-model-arvoksi annetaan chosenCheckpoint-muuttuja, joka sisältää valitun tallennuspisteen. Kun käyttäjä painaa ”Send”-nappia, valitut ääninäyttelijä ja tallennuspiste lähetetään python-palvelimelle, missä niiden avulla löydetään oikea puhesynteesimalli käytettäväksi.

Kuvan 16 ikkunassa on myös ”Tatu’s default model” -valintaruutu. Jos ruutua painaa, ääninäyttelijä- ja tallennuspistelista lukittuvat. Silloin ”Send”-nappi lähettää oletukseksi valitun, parhaan puhesynteesimallin käytettäväksi generointiin. Tämänhetkinen paras malli on määritelty jo aikaisemmin ja sitä käytetään pääasiallisesti erilaisten tekstien testaamisessa. Tallennuspiste löytyy ääninäyttelijän omasta listasta, mutta käyttöönotto on nopeampaa, kun sitä ei tarvitse aina etsiä erikseen kerta toisensa jälkeen.



Kuva 16. Ikkuna, jossa valitaan puheen luomiseen käytettävä tallennuspiste eli puhesynteesimalli. ”Send”-napista aloitetaan tekstin muuntaminen puheeksi.

5 Toteutetun sovelluksen tulokset

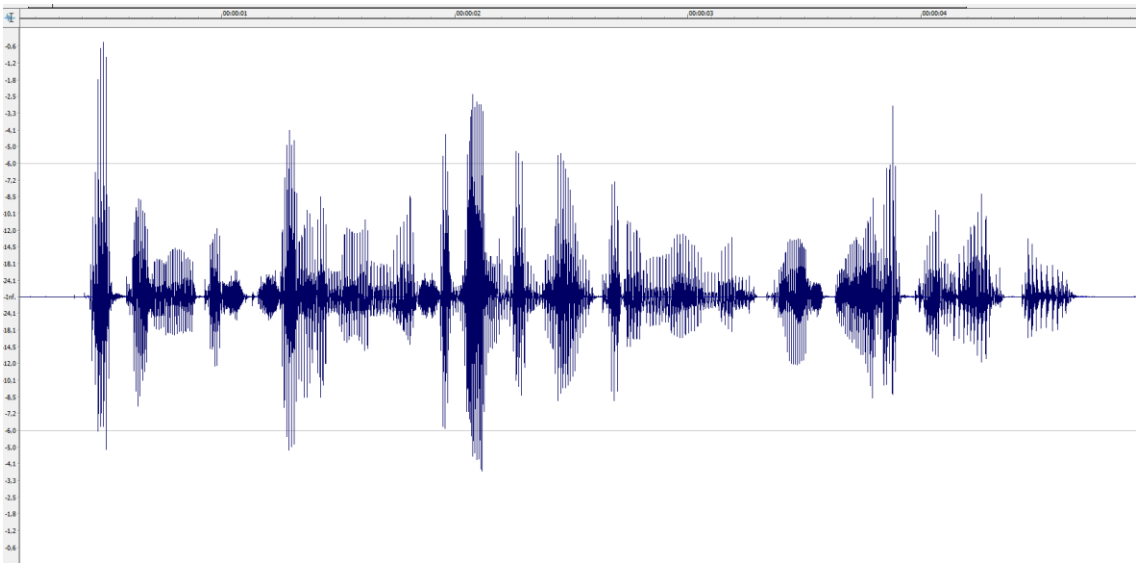
5.1 Testaus

Kun insinöörityönä toteutettu sovellus oli saatu kaikin puolin toimivaan kuntoon, ääninäyttelijä testasi sen äänitystyökalua. Testissä haluttiin selvittää, millaisia eroja äänenlaadussa on silloin, kun äänitetään erilaisissa ympäristöissä. Erilaisilla ympäristöillä tarkoitetaan tässä tapauksessa sekä äänitystilojen vaihtelevaa akustiikkaa että taustahälinän eri määriä. Tavoitteena oli siis saada vastaus kysymykseen, onko äänittäminen mahdollista erilaisissa tiloissa ilman, että äänitteen laatu heikkenee ratkaisevasti.

Puhesynteesin harjoittamiseen käytettävän datan täytyy olla laadultaan hyvää, ja tiedossa on jo, että hyvää laatua voidaan äänittää erillisessä äänityskopissa. Äänityskopin ehdoton valtti on hyvä eristys, joka takaa taustahälinättömän puheenlaadun. Valitettavana heikkoutena on kuitenkin sen fyysinen sijainti, mikä pakottaa ääninäyttelijän saapumaan yrityksen tiloihin aina, kun äänityksiä halutaan tehdä. Se paitsi lisää yritykselle koituvia kustannuksia myös hidastaa huomattavasti datan keräystä, sillä joka kerta yrityksen täytyy erikseen sopia äänittämistä varten aikataulu ääninäyttelijän kanssa.

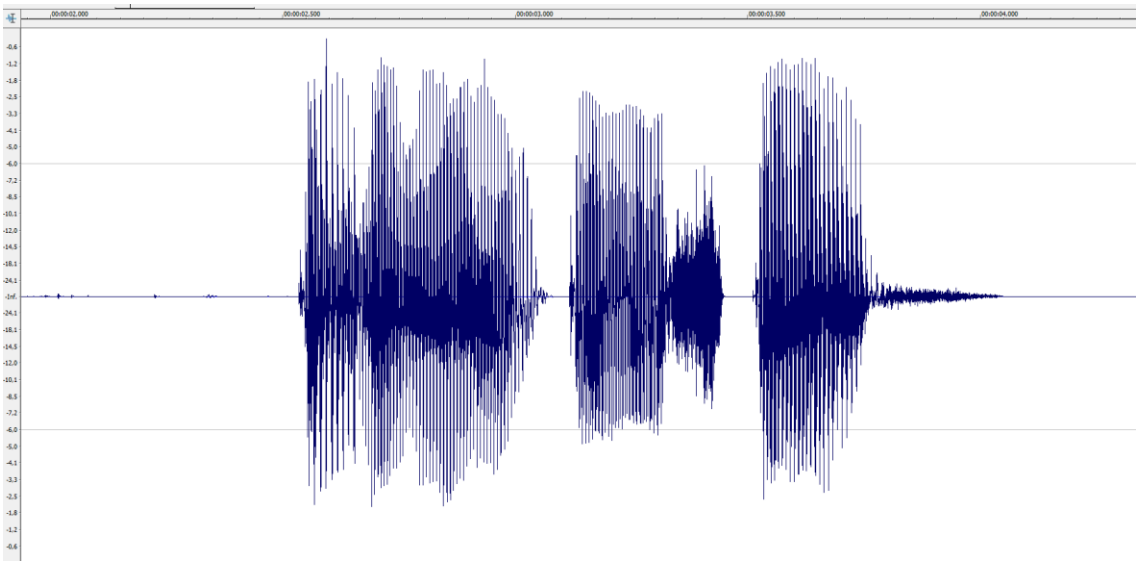
Tämänhetkisen äänitystavan tiedetään tuottavan ääntä ”raakamuodossa”, jota ei kompressoida tai muutenkaan prosessoida automaattisesti. Siksi uutta, selainpohjaista äänitystapaa voidaan verrata nykyiseen ja varmistaa, että laatu on yhtä tasokasta. Jos eri tavalla tuotettujen äänitteiden ääniaalloissa on huomattavia eroja, voidaan päätellä, että toinen näyte on laadullisesti erilainen kuin toinen.

Kuvassa 17 on esimerkki nykyisellä tavalla tuotetusta äänitteestä, joka sisältää puhutun lauseen. Äänitettä ei ole kompressoitu, mikä näkyy muun muassa selkeinä tasoeroina ääniaalloissa.



Kuva 17. Esimerkki aaltomuotoisesta äänitteestä, jota ei ole prosessoitu.

Jos verrataan kuvan 17 äänitettä aaltomuotoon, joka kuuluu kuvan 18 kompressoidulle äänitteelle, havaitaan selkeä eroavaisuus näiden kahden välillä. (Huom. Kuvissa ei ole sama äänitiedosto.) Kuvassa 18 äänenvoimakkuudet on tasattu ja tasot on nostettu alkuperäistä ylemmäksi.



Kuva 18. Esimerkki aaltomuotoisesta äänitteestä, jota on kompressoitu.

Äänitystä päätettiin testata neljässä erilaisessa tilassa: toimiston kokoushuoneessa, taukotilassa, äänityskopissa ja toimistorakennuksen meluisassa ala-aulassa. Äänityksissä käytettiin sangallista kondensaattorimikrofonia, joka kytkettiin Focusrite Scarlett -

äänikorttiin. Jokaisessa tilassa ääninäyttelijä äänitti parikymmentä lausetta. Äänitettäessä ääninäyttelijä piti huolen siitä, että äänisignaali ei mene särölle. Äänitysasetukset ja tekniset olosuhteet, kuten mikrofonin sijainti suhteessa suuhun, olivat samat kaikissa tiloissa.

Tuloksia analysoitiin sekä kuuntelemalla äänitteitä että vertailemalla niiden aaltomuotoja Sound Forgella tehtyihin äänitteisiin. Testiäänitteistä pystyi jo pelkästään kuuntelemalla havaitsemaan äänenvoimakkuuden voimakasta aaltoilua, mikä kertoo siitä, että äänisignaalia on muokattu jossain vaiheessa prosessia. Lisäksi kummastusta herätti taustahälinän ja suhinan puute, vaikka äänityksiä tehtiin myös meluisassa ala-aulassa. Jotta epäilyksille saatiin vahvistus, tehtiin uusia testiäänityksiä äänityskopissa siten, että äänitettiin samaan aikaan sekä uudella että vanhalla äänitystavalla. Sitten molempien tapojen tuottamia äänitteitä verrattiin aaltomuotoisina Sound Forgessa. Tiedettiin, että jokin on mennyt vikaan, jos uudella tavalla tuotettu aaltomuoto eroaa vanhasta. Lopputulos oli, että vanhalla tekniikalla äänitetty tiedosto oli kuvan 17 mallinen, kun taas uusi näytti kuvan 18 tapaisesti prosessoidulta.

Epäonnistumisen syyksi alettiin epäillä äänitystyökaluna käytettyä MediaRecorderia. Sen tilalle etsittiin muita mahdollisia äänitystapoja. Lyhyen selvityksen jälkeen MediaRecorderin sijasta päätettiin kokeilla WebAudioRecorder-JavaScript-kirjastoa. WebAudioRecorder tukee kolmea äänen tallennusmuotoa, jotka ovat Waveform Audio (.wav), Ogg Vorbis (.ogg) ja MPEG-1 Audio Layer III (.mp3). [24.] Se nauhoittaa ääntä ja muuntaa sen Blob-olioksi. Blob-olio esittää muuttumattoman raakadatan tiedostotyyppisenä oliona. Esimerkiksi JavaScriptin File-rajapinta perustuu Blobiin perien siltä sen toiminnallisuuden sekä laajentaen sitä tukemaan käyttäjän järjestelmän tiedostoja. [25.]

Kun sovellus saatiin jälleen toimimaan siihen vaihdetun äänitystyökalun jälkeen, tehtiin samanlainen äänitteen laatua mittaava testi. Testiäänityksistä pystyi edelleen havaitsemaan äänenvoimakkuuden aaltoilua, ikään kuin jokin säätelisi sitä automaattisesti. Myös ääniaallot olivat samanlaisia kuin edellisen testin päätteeksi, joten ongelma ei ollut vielä ratkennut.

Asiaa tutkittaessa selvisi, että selaimet tekevät automaattista äänen editointia. Säätöjä kutsutaan nimellä *audio constraints*. Niistä kolme oleellista ääntä muuttavaa asiaa ovat

echoCancellation, noiseSuppression ja autoGainControl. Sovelluksen kehityksessä testaamiseen käytettiin lähinnä Google Chromea, jossa kaikki kolme muokkainta ovat päällä oletuksena. [26.] Ominaisuuden hyöty lienee siinä, että suuria piikkejä ja äänen säröytymistä ei pääse tapahtumaan selainympäristössä. Tähän projektiin ei kuitenkaan haluttu odottamatonta äänenmuokkausta, joten muokkaimet piti saada pois päältä. Tutkimisen jälkeen selvisi, että muokkainten tilan voi määritellä parametriksi getUserMedia-metodille esimerkkikoodin 5 mukaisesti. Varmuuden vuoksi jokainen property-muuttuja asetettiin epätodeksi.

```
let constraints = {
      audio: {
            sampleSize: false,
            channelCount: false,
            echoCancellation: false,
            autoGainControl: false,
            noiseSuppression: false,
            sampleRate: false
      }
};

navigator.mediaDevices
      .getUserMedia(constraints)
      .then(stream => {
            this.$store.dispatch("setIsSentenceRecording", {
            sentence: this.currentlyRecordedSentence,
            isRecording: true
      });
```

Esimerkkikoodi 5. Audio constraints -muuttujien määritys ja asettaminen äänitystyökaluun.

Korjauksen jälkeen suoritettiin jälleen testi, jossa verrattiin uuden ja vanhan äänitystavan tuottamia ääniaaltoja. Äänestä hävisi aaltoilu ja ääni kuulosti luonnollisemmalta. Ainoa ero vanhaan äänitystapaan verrattuna oli aaltomuotojen perusteella enää se, että uusi äänitystapa tuotti kuusi desibeliä hiljaisempaa ääntä. Muuten aaltomuodot olivat käytännössä identtisiä keskenään. Jokin asia äänityksessä aiheutti odotettua hiljaisemman lopputuloksen, eikä asiaan saatu selvyyttä. Äänitykseen pystyttiin kuitenkin lisäämään vahvistusta (*gain*) ohjelmallisesti. Esimerkkikoodi 6 esittää tavan, jolla vahvistusta lisättiin äänitykseen. Ensin luodaan uusi AudioContext-olio, jolle luodaan eräänlainen äänenvahvistusnappi. Napille annetaan arvoksi 2, jonka tarkistettiin lisäävän äänisignaalin äänenvoimakkuutta 6 desibeliä. Sitten nappi liitetään AudioContextiin, jota käytetään äänityksessä.

```
this.audioCtx = new AudioContext();

let audioLevel = this.audioCtx.createGain();
```

```
audioLevel.gain.value = 2;

let mixer = this.audioCtx.createGain();
audioLevel.connect(mixer);

let source = this.audioCtx.createMediaStreamSource(stream);
source.connect(audioLevel)
```

Esimerkkikoodi 6. Äänisignaalin vahvistus kuudella desibelillä. Vahvistukselle annetun arvon 2 (200 %) oikeellisuus tarkistettiin Sound Forgesta, jossa se tarkoittaa kuuden desibelin vahvistusta.

Äänitteitä vertailtiin jälleen. Äänisignaalin vahvistuksen jälkeen uusi äänitystapa antoi täysin samanlaisen ääniaallon vanhaan äänitystapaan verrattuna, minkä katsottiin tarkoittavan ei-prosessoitua ääntä. Sovellus oli jo testattu muuten toimivaksi, ja nyt myös äänitystyökalun tuottama äänenlaatu oli saatu riittävälle tasolle.

5.2 Saavutetut tavoitteet ja jatkokehitysmahdollisuudet

Projektin alussa sovellukseen suunnitellut ominaisuudet saatiin toteutettua kokonaisuudessaan. Sovellus tarjoaa yritykselle nopean ja järjestelmällisen tavan hallita puhesynteesimallin harjoittamisen vaativaa dataa. Äänitystyökalulla voidaan nauhoittaa äänitteitä, jotka tallentuvat järjestelmällisesti tietokantaan. Äänitteiden ryhmittelyn ansiosta datasettiin voidaan sisällyttää paitsi kaikenlaisia, myös halutessa vain tietyntyyppisiä äänitteitä monipuolista testausta ajatellen. Sovelluksen käyttöliittymä tarjoaa täyden hallinnan harjoittamiselle, mikä mahdollistaa äänityksen lisäksi sekä harjoittamisen että tulosten kuuntelun suoraan selaimessa. Kaiken kaikkiaan datan keräämisen pitäisi sovelluksen ansioista nopeutua huomattavasti.

Ennen kuin sovellusta voidaan käyttää kokonaisvaltaisesti, täytyy tehdä lisää testejä etenkin äänen laadun suhteen. Äänen laatu on kriittinen asia puhesynteesin kehityksessä, sillä se vaikuttaa ratkaisevasti lopputulokseen. Vaikka sovelluksessa äänitettyjen äänitteiden aaltomuodot saatiinkin täsmäämään puhekopissa tehtyihin äänitteisiin, laatueroja on silti tutkittava lisää. Tarvitaan lisää vastauksia esimerkiksi eri selaimiin liittyvissä kysymyksissä. Selaimiin tulee jatkuvasti päivityksiä, jotka saattavat arvaamattomasti muuttaa sovelluksen toimintaa. Tällä hetkellä yksi suurimmista ratkaisemattomista asioista on äänitteiden editointi, joka pitää edelleen hoitaa manuaalisesti. Sovellus tarvitsee mahdollisesti jonkinlaisen ääniedointityökalun, sillä täydellisesti toimivien,

automaattisten editointialgoritmien kehittäminen on erittäin monimutkaista ja haastavaa. Toinen vaihtoehto on hoitaa editointi jatkossakin manuaalisesti ilman sovellusta.

6 Yhteenveto

Insinöörityössä kehitettiin sovellus, jonka avulla voidaan kerätä dataa käytettäväksi neuroverkkopohjaisen puhesynteesin harjoittamiseen ja hallita harjoittamista käyttöliittymästä käsin. Sovelluksella voidaan kerätä ja ryhmitellä aineistoa sekä hallita kokonaisvaltaisesti puhesynteesimallin harjoittamista selaimessa.

Sovellus kehitettiin ratkaisemaan nykyiseen datanluontitapaan liittyviä ongelmia. Se nopeuttaa datankeräysprosessia ja tarjoaa käyttöliittymän, jossa harjoittamista voidaan hallita. Lisäksi se ratkaisee paikkasidonnaisuuteen liittyvän ongelman mahdollistamalla datan luonnin missä vain. Sovellus kuitenkin tuo mukanaan haasteita, jotka liittyvät lähinnä selainpohjaisen äänityksen tuottamaan äänenlaatuun. Epävarmuutta aiheuttaa erityisesti selainten jatkuvasti muuttuva toiminta, joka saattaa muuttaa myös selainpohjaista äänitystä hyödyntäviä sovelluksia. Lisäksi haasteita aiheuttaa äänitteiden editointi, jota sovelluksessa ei tällä hetkellä voi tehdä. Äänitteet on editoitava ennen niiden käyttöä, joten toistaiseksi se on tehtävä sovelluksen ulkopuolella.

Insinöörityön avulla saatiin käsitystä selainpohjaisen äänityksen hyvistä ja huonoista puolista osana aineiston luomista puhesynteesiä varten. Työssä toteutettu sovellus täyttää ominaisuuksiltaan sille ennalta määritellyt tavoitteet. Sovellus on valmis testattavaksi, ja tulevaisuudessa se on mahdollisesti yrityksen käytössä osana paremman puhesynteesin kehitystä.

Lähteet

1 Woodford, Chris. 2018. Speech Synthesizers. Verkkoaineisto. Explainthatstuff. <https://www.explainthatstuff.com/how-speech-synthesis-works.html>. Päivitetty 23.12.2018. Luettu 3.2.2019.

2 Lemmetty, Sami. 1999. History and Development of Speech Synthesis. Master's Thesis. Aalto-yliopisto, Laboratory of Acoustics and Audio Signal Processing. <http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap2.html>. Päivitetty 1.6.1999. Luettu 13.2.2019.

3 Kratzenstein's resonators. Verkkoaineisto. Haskins Laboratories. <http://www.haskins.yale.edu/featured/heads/SIMULACRA/kratzenstein.html>. Luettu 13.2.2019.

4 Vocoders and Voders. Verkkoaineisto. Engineering and Technology History Wiki. <https://ethw.org/Vocoders_and_Voders>. Luettu 14.2.2019.

5 The Voder (1939). Verkkoaineisto. Haskins Laboratories. <http://www.haskins.yale.edu/featured/heads/SIMULACRA/voder.html >. Luettu 15.2.2019.

6 Dormehl, Luke. 2019. What is an artificial neural network? Here's everything you need to know. Verkkoaineisto. Digital Trends. <https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network>. 5.1.2019. Luettu 21.2.2019.

7 Keurulainen, Antti. 2019. Tutkimusjohtaja, Bitville Oy, Espoo. Keskustelu 7.3.2019.

8 Seijas, Jesús. 2018. Into a better Speech Synthesis Technology. Verkkoaineisto. Becoming Human. <https://becominghuman.ai/into-a-better-speech-synthesis-technology-29411b64f2a2>. 17.7.2018. Luettu 21.2.2019.

9 Pachet, Francois. WaveNet architecture. Verkkoaineisto. ResearchGate. <https://www.researchgate.net/figure/WaveNet-architecture_fig49_319524552>. Luettu 21.2.2019.

10 Wang, Yuxuan; Skerry-Ryan, RJ; Stanton, Daisy; Wu, Yonghui; Weiss, Ron J.; Jaitly, Navdeep; Yang, Zongheng; Xiao, Ying; Chen, Zhifeng; Bengio, Samy; Le, Quoc; Agiomyrgiannakis, Yannis; Clark, Rob & Saurous, Rif A. 2017. Tacotron: Towards End-to-End Speech Synthesis. Verkkoaineisto. Interspeech 2017. <https://arxiv.org/abs/1703.10135>. 2017. Luettu 20.3.2019.

11 Shen, Jonathan; Pang, Ruoming; Weiss, Ron J.; Schuster, Mike; Jaitly, Navdeep; Yang, Zongheng; Chen, Zhifeng; Zhang, Yu; Wang, Yuxuan; Skerry-Ryan, RJ; Saurous, Rif A.; Agiomyrgiannakis, Yannis & Wu, Younghui. 2017. Natural TTS Synthesis by Conditioning WaveNet on Mel Spectogram Predictions. Verkkoaineisto. International Conference on Acoustics, Speech and Signal Processing (ICASSP). <https://arxiv.org/abs/1712.05884>. 2018. Luettu 20.3.2019.

12 Unuth, Nadeem. 2019. Mean Opinion Score (MOS): A Measure of Voice Quality. Verkkoaineisto. Lifewire. <https://www.lifewire.com/measure-voice-quality-3426718>. Päivitetty 27.2.2019. Luettu 20.3.2019.

13 Kwiatkowski, Sebastian. 2018. Speech Synthesis as a Service. Verkkoaineisto. Towards Data Science. <https://towardsdatascience.com/speech-synthesis-as-a-service-5c65d17e62f4>. 16.6.2018. Luettu 12.2.2019.

14 Hakuri, Jyri. 2019. Asiantuntija, Bitville Oy, Espoo. Keskustelu 5.3.2019.

15 Introduction. Verkkoaineisto. Vue.js-dokumentaatio. <https://vuejs.org/v2/guide/index.html>. Luettu 10.2.2019.

16 Lasn, Indrek. 2018. From Zero to Hero with Vue – But first, why Vue?. Verkkoaineisto. FreeCodeCamp. <https://medium.freecodecamp.org/from-zero-to-hero-with-vue-why-vue-8c7e981b494>. 29.8.2018. Luettu 10.2.2019.

17 What is Vuex? Verkkoaineisto. Vuex-dokumentaatio. <https://vuex.vuejs.org>. Luettu 10.2.2019.

18 Mutations. Verkkoaineisto. Vuex-dokumentaatio. <https://vuex.vuejs.org/guide/mutations.html>. Luettu 10.2.2019.

19 Why Vuetify? Verkkoaineisto. Vuetify-dokumentaatio. <https://vuetifyjs.com/en/getting-started/why-vuetify>. Luettu 21.3.2019.

20 MongoDB and MySQL Compared. Verkkoaineisto. MongoDB. <https://www.mongodb.com/compare/mongodb-mysql>. Luettu 5.2.2019.

21 Node-uuid. Verkkoaineisto. Github. <https://github.com/kelektiv/node-uuid>. Luettu 11.3.2019.

22 Wilson, Chris. Volume-meter. Verkkoaineisto. Github. <https://github.com/cwilso/volume-meter>. Luettu 5.3.2019.

23 Vue-scroll-to. Verkkoaineisto. Github. <https://github.com/rigor789/vue-scrollto#readme>. Luettu 11.3.2019.

24 WebAudioRecorder.js. Verkkoaineisto. Github. <https://github.com/higuma/web-audio-recorder-js>. Luettu 4.2.2019.

25 Blob. Verkkoaineisto. MDN web docs. <https://developer.mozilla.org/en-US/docs/Web/API/Blob>. Luettu 4.2.2019.

26 Naicu, Octavian. 2017. Supported Audio Constraints in getUserMedia(). Verkko-aineisto. Pipe Video Recorder. <https://blog.addpipe.com/audio-constraints-getusermedia/>. 8.11.2017. Luettu 21.3.2019.