jamk.fi

Virtuaalinen asiakaspalveluneuvoja

AWS:n palveluilla toteutettuna

Janne Eela
Samuli Rukkila

Opinnäytetyö
Marraskuu 2020
Liiketalouden ala
Tietojenkäsittelyn tutkinto-ohjelma (AMK)

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Eela, Janne Rukkila, Samuli	Opinnäytetyö, AMK	marraskuu 2020
	Sivumäärä 40	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Virtuaalinen asiakaspalveluneuvoja
AWS:n palveluilla toteutettuna

Tutkinto-ohjelma
Tradenomi (AMK), tietojenkäsittely tutkinto-ohjelma

Työn ohjaaja(t)
Kiviaho, Niko

Toimeksiantaja(t)
Kela, Jyväskylä

Tiivistelmä

Chatbottien käyttö osana internetissä tapahtuvaa asiakaspalvelua on alati kasvava ilmiö. Chatbottien käytöllä tavoitellaan nopeaa ja ympärivuorokautista asiakaspalvelua, joka on samalla yritykselle edullista. Suomenkieliset chatbotit eivät ole kovin yleisiä, koska sen toteuttaminen tällä kielellä on varsin haastavaa. Suomen kielessä on monia taivutusmuotoja ja sanoilla voi olla useita eri merkityksiä. Lisäksi kielen käyttäjäkunta on palvelun tuottamisen kustannuksiin suhtautettuna liian pieni. Tästä syystä monet palvelut eivät toimi suomen kielellä.

Opinnäytetyön tavoitteena oli luoda toimeksiantajalle virtuaalisen asiakaspalveluneuvojan prototyyppi hyödyntäen AWS:n (Amazon Web Services) palveluita. Samalla tutkittiin kyseisten palveluiden soveltuvuutta tähän tarkoitukseen, sekä sitä, että voitiinko muiden alustojen pilvipalveluita käyttää asiakaspalveluneuvojan tukena.

Aiheesta ei ole paljon aikaisempaa tutkimusta, joten teoreettisessa viitekehyksessä paneuduttiin AWS:n eri palveluiden sisältöön ja toimintaan. Tutkimus toteutettiin kehittämistutkimuksena. Virtuaalinen asiakaspalveluneuvoja toteutettiin käyttäen AWS:n palveluja yhtenä kokonaisena infrastruktuurina, jossa palvelut keskustelevat keskenään. Neuvojan logiikka kirjoitettiin käyttäen TypeScriptiä, joka spesifioitiin Kelan tarpeisiin.

Lopullisena tuloksena saatiin toimiva asiakaspalveluneuvoja, jota ajetaan henkilökohtaisessa AWS:n testiympäristössä. Samalla saatiin tuloksia siitä, onko mahdollista saada muita pilvipalvelualustoja keskustelemaan AWS:n kanssa. Samalla arvioitiin, kuinka hyvin virtuaalinen asiakaspalveluneuvoja toimii tapauksissa, joissa tekstiä prosessoiva palvelu parsii vain englannin kieltä, mutta käyttäjä voi joko puhua tai kirjoittaa sitä.

Avainsanat (asiasanat)

AWS, chatbot, tekoäly, kehittämistutkimus, asiakaspalveluneuvoja

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Eela, Janne Rukkila, Samuli	Bachelor's thesis	November 2020
		Language of publication: Finnish
	Number of pages 40	Permission for web publication: x

Title of publication

Virtual customer service advisor

Implemented with Amazon Web Services (AWS)

Degree programme

Business Information systems

Supervisor(s)

Kiviaho, Niko

Assigned by

Kela, Jyväskylä

Abstract

The usage of chatbots with ongoing customers service is a growing phenomenon. One of the benefits of using chatbots are lower costs compared to maintaining actual humans. Chatbots which work in Finnish are quite rare because developing one to work with it is a challenge. This is mainly because in Finland there are many inflections within the language and one phrase may mean many things. It also isn't beneficial to develop a chatbot for this language because from the financing perspective, the amount of people who speak this language is very small.

The aim of this thesis was to create a virtual customer service advisor prototype for our client while utilizing AWS's (Amazon Web Services) services in the creation. Simultaneously research was conducted on the possibility to integrate other platform's services within our customer service advisor.

There wasn't much research made from the subject before this thesis, so the theoretical data basis was mainly based on the documentation of AWS's services. The research was created using design-based research. The virtual customer service advisor was developed using AWS's services as a one big infrastructure where these services where in contact with each other. The logic of the advisor was written with TypeScript which were specified for the needs of our client, Kela.

The final product was a working customer service advisor which was running on AWS's test platform. At the same time, we were able to develop results as is it possible to integrate AWS's services with other platform's services. Tests were concluded on how well the advisor works while user is speaking or typing Finnish for chatbot that understands only English.

Keywords/tags (subjects)

AWS, chatbot, artificial intelligence, design-based research, customer service advisor

Miscellaneous (Confidential information)

Sisältö

Kuviot

Käsiteluettelo

Virtuaalinen asiakaspalveluneuvoja	Chatbot, jonka kanssa käyttäjä pystyy kommunikoimaan puheen tai tekstin avulla
Pilvipalvelu	Tietoteknisten palveluiden tarjoaminen internetin välityksellä
Chatbot	Tietokoneohjelma, jonka on tarkoitettu käymään keskusteluja ihmisten kanssa
NoSQL	Perinteisestä relaatiomallista poikkeava tietokanta, joka ei seuraa kiinteästi määriteltyä taulukkoskeemaa
HTML	Avoimesti standardoitu kuvauskieli, jolla voidaan kuvata *WWW*-sivun elementtien rakennetta (Hypertext Markup Language)
Binääripaketti	Binaarimuodoksi käännetty ohjelma, joka mahdollistaa esimerkiksi helpomman tavan asentaa ohjelmia
Node.js	Alustariippumaton *JavaScript*-ajoympäristö, joka mahdollistaa koodin suorittamisen suoraan palvelimella
REST-API	Ohjelmointirajapintojen (*API*) toteuttamiseen tarkoitettu arkkitehtuurimalli, joka perustuu *HTTP*-protokollaan
Base64 (string)	Koodausjärjestelmä, joka edustaa binääristä dataa. Suunniteltu kuljettamaan binaarimuodossa tallennettua tietoa kanavien yli (esimerkiksi kuvia) tekstimuodossa
SSML	*XML*-pohjainen merkintäkieli sovelluksien puhesynteesiä varten
YAML	Merkintäkieli, jota käytetään usein konfiguraatiotiedostojen määrittelyssä (YAML Ain’t Markup Language)

1 Johdanto

Chatbotit, niin tekstillä kuin puheellakin toimivat, ovat alati kasvava osa internetin maailmaa. Asiakaspalvelua varten tehtyjen chatbottien päätarkoituksena on tarjota asiakkaalle välittömästi apua ajasta ja paikasta riippumatta, joka on myös kustannustehokkaampaa ihmiseen verrattuna. Monet yritykset tarjoavat jo tekstillä toimivia chatbotteja nettisivuillaan, mutta puheella toimivat chatbotit ovat vielä kohtuullisen harvinaisia.

Chatbotit eivät kuitenkaan ole vielä voineet syrjäyttää ihmisasiakaspalvelijoita. Usein chatbotit ovatkin alkeellisia, ja ymmärtävät vain yksinkertaisia lauseita. Chatbotin luominen ei helppoa varsinkaan suomen kielelle, koska se on tekoälyn kannalta vaikea ymmärtää. Tämä johtuu siitä, että suomen kieli on kieliopillisesti hyvin vapaamuotoista ja lauseita voidaan tulkita monesta eri kontekstista. Tämä tuottaa vaikeuksia etenkin puheen ymmärtämisessä sekä parsimisessa. Tästä syystä monet palvelut eivät tarjoa suomen kieltä vaihtoehtona puheen ymmärtämiselle, juuri sen harvinaisuuden sekä monimutkaisuuden takia.

Yksi ratkaisu suomenkielisen chatbotin luomiseen on käyttää AWS:n (Amazon Web Services) pilvipalveluita. Tämä kokonaisuus sisältää erilaisia palveluja, joilla voi luoda toimivan chatbotin pienellä kustannuksella. Amazon *Lex* on palvelu, jolla voi rakentaa toimivia chatbotteja. *Lex* käyttää samaa tekoälyä kuin Amazon *Alexa*, joka on suosittu Amazonin kehittämä ja julkaisema virtuaaliavustaja. Tämän opinnäytetyön tarkoituksena on luoda Kelalle virtuaalisen asiakaspalveluneuvojan prototyyppi hyödyntäen AWS:n palveluita.

Toimiva virtuaalinen asiakaspalveluneuvoja mahdollistaisi Kelan asiakkaille ympärivuorokautisen asiakaspalvelun. Asiakaspalvelun osittainen siirtäminen virtuaaliseen muotoon voisi myös vähentää Kelan asiakaspalvelun kuormitusta. Aikaisempaa tutkimusta AWS:llä toteutetuista suomenkielisistä, virtuaalisista asiakaspalveluneuvojista ei ole. Tämän tutkimuksen tulokset valottavat mahdollisuuksia käyttää AWS:n palveluita suomenkielisissä olosuhteissa.

2 Tutkimusasetelma

Tässä luvussa käydään läpi opinnäytetyön tavoitteet ja rajaukset sekä esitellään tutkimuksen toimeksiantaja. Lisäksi esittelemme tutkimuskysymykset sekä -menetelmät.

2.1 Opinnäytetyön tavoitteet ja rajaukset

Opinnäytetyön tavoitteena on tuottaa pilvipalveluiden avulla luotu virtuaalinen asiakaspalveluneuvoja sekä tutkia pystyykö sitä hyödyntämän Kelan järjestelmässä. Asiakaspalveluneuvojan tärkein toiminnallinen tavoite on ymmärtää, vastata sekä esittää jatkokysymyksiä englannin kielellä. Mahdollisuutta ymmärtää suomen kieltä tutkitaan sekä pyritään myös integroimaan prototyyppiin. Lisäksi tavoitteena on tuottaa asiakaspalveluneuvoja, joka pystyy vastaamaan yksinkertaisiin Kelaan ja Kelan palveluihin liittyviin kysymyksiin, ja joka osaa suorittaa yksinkertaisia toimintoja, kuten ajanvarauksen luonnin ja omien tietojen näyttämisen.

Virtuaalista asiakaspalveluneuvojaa ei tulla integroimaan Kelan järjestelmiin tässä opinnäytetyössä, vaan se jää ainoastaan tutkimustasolle. Sen on kuitenkin tarkoitus sisältää toimintoja, joita Kelassa voitaisiin hyödyntää. Käyttötarkoitus, tarpeellisuus sekä kehittämisen ja ylläpidon kustannukset rajataan pois tästä tutkimuksesta.

2.2 Toimeksiantaja

Toimeksiantajanamme toimii Suomen kansaneläkelaitos, eli Kela. Kelan tehtävänä on huolehtia Suomessa sekä muualla maailmalla asuvien suomalaisten sosiaaliturvasta. Vuosittain Kela palvelee jopa noin 3,6 miljoonaa asiakasta (Kela lyhyesti 2018). Kela määritteli tutkimustamme koskettavat teknologiset rajaukset, tutkimuksen aiheen sekä suuntaa antavat tutkimuskysymykset. Näiden kysymysten pohjalta loimme kolme tutkimuskysymystä, jotka esittelemme seuraavassa kappaleessa.

Kela on jo aikaisemmin julkaissut yleiseen käyttöön chatbotteja. Yksi näistä on chattirobotti, joka vastaa opiskelijoiden kysymyksiin tapauksissa, jossa he siirtyvät yleisen

asumistuen saajiksi. Tähän chattirobotin kehitykseen oli osallistunut myös opiskelijoita. (Kela tarjoaa uudenlaisen palvelun opiskelijoille 2017.)

2.3 Tutkimuskysymykset

Tutkimuksen aihetta on rajattu tarkasti, koska chatbot käsitteenä on laaja. Tässä tutkimuksessa halutaan keskittyä chatbotin luomiseen AWS:n palveluiden avulla, sekä siihen liittyviin komplikaatioihin, kuten myös sen soveltuvuuteen suomen kielelle. Tutkimuskysymykset on määritelty tarkasti toimeksiantajan toiveiden mukaan:

- Kuinka hyvin AWS:n palvelut soveltuvat virtuaalisen asiakaspalveluneuvojan toteuttamiseen?
- Voidaanko AWS:n palveluihin yhdistää kolmannen osapuolen palvelu, jonka avulla virtuaalinen asiakaspalveluneuvoja voitaisiin toteuttaa suomen kielellä?
- Voidaanko *Amazon Sumerian*-hostin chatbot-logiikka vaihtaa AWS:n omista (Amazon *Lex* & *Polly*) palveluista johonkin toiseen alustaan, joka tukisi suomen kieltä?

2.4 Tutkimus- ja kehittämismenetelmät

Tutkimus toteutetaan kehittämistutkimuksena, koska tutkimuksen ja prototyypin tarkoituksena on tuottaa toimiva virtuaalinen asiakaspalveluneuvoja-prototyyppi Kelalle. Tarkoituksena ei ole kerätä informaatiota chatbotin käytöstä tai haastatella muita ihmisiä sen käytöstä.

Kehittämistutkimuksessa kehitetään tuotetta, menetelmää tai organisaatiota muutoksen aikaansaamiseksi. Tämä tutkimusmuoto eroaa enemmän perinteisestä laadullisesta sekä määrällisestä tutkimuksesta, sillä siihen kuuluu myös ongelman poistaminen. Perinteisessä kvalitatiivisessa ja kvantitatiivisessa tutkimuksessa valittua tutkimusongelmaa analysoidaan syvällisesti ja sille esitetään jokin ratkaisuehdotus, jota ei

kuitenkaan yleensä konkreettisesti toteuteta tutkimuksen aikana. Kehitystutkimuksessa puolestaan ongelmaan esitetään ratkaisu, joka myös toteutetaan. Kehittämistutkimus siis ikään kuin jatkaa siitä, mihin perinteinen tutkimus päättyy. (Kananen 2015, 39-40.)

3 Pilvipalvelut ja aikaisemmat tutkimukset

Tässä luvussa käymme läpi tietoperustaa Amazonin ja Googlen pilvipalveluille, joita hyödynnämme tutkimuksessamme. Käymme läpi AWS:n infrastruktuuria, sekä sen käyttötarkoituksia. Kerromme samalla, kuinka AWS pohjautuu erilaisten toimintojen käyttöön sekä kuinka hyvin se tukee käyttötarkoituksiamme. Lopuksi kerromme aikaisemmista tutkimuksesta sekä AWS:n suomen kielen tuesta. Näiden läpikäyminen on tärkeää lopullisen prosessin teoriaperustan ymmärtämisen kannalta.

3.1 Amazonin pilvipalvelut

AWS on vuonna 2006 perustettu pilvipalvelualusta, joka tarjoaa IT-infrastruktuuripalveluita niin yrityksille kuin yksityisille asiakkaille. AWS:n yksi suurimmista eduista on sen mahdollisuus korvata etupääoman kalliit infrastruktuurikustannukset alhaisilla muuttavilla kustannuksilla. Tämän ansiosta yrityksien ei tarvitse suunnitella sekä hankia palvelimia etukäteen. AWS tarjoaa luotettavan ja skaalautuvan infrastruktuurialustan, joka on vallannut jo satoja tuhansia yrityksiä ympäri maailmaa. (About AWS n.d.)

Amazon Lex

Amazon *Lex* on AWS:n palvelu, jolla voidaan rakentaa keskustelu-käyttöliittymiä ohjelmistoille käyttäen kommunikointivälineenä tekstiä ja puhetta. *Lex* käyttää samaa keskustelutekniikkaa kuin Amazonin oma palvelu, *Alexa*. Palvelun avulla voidaan rakentaa vaativia, luonnollisen kielen chatbotteja jo olemassa oleviin sovelluksiin. Palvelun voi myös integroida saumattomasti muiden AWS:n palvelujen kanssa, joita on esimerkiksi *DynamoDB* ja *Lambda*. Tällä hetkellä *Lex* ymmärtää vain englannin

kieltä, joten tapauksissa, jossa se halutaan lokalisoida toiselle kielelle, tarvitaan toinen sovellus. (Amazon Lex: Developer Guide 2020, 1.)

Lex-palvelun käyttö aloitetaan yleensä luomalla botti, jolle konfiguroidaan aikomuksia (engl. *intents*), eli eri teemoihin liittyviä kysymyksiä, joita käyttäjä voi kysyä. Aikomukset täytetään esimerkkikysymyksillä, joilla voidaan opettaa *Lex* ymmärtämään aikomuksia. Erilaisia aikomuksia ovat esimerkiksi pitsan tilaaminen, hotellin varaaminen tai kukkalähetyksen tilaaminen. Aikomukset sisältävät myös usein aukkoja, joihin voidaan sijoittaa tieto jostain valinnasta. Erilaisia aukkoja ovat esimerkiksi hotellin varauksessa: hotellin nimi, majoittumisaika tai huoneen tyyppi. Kun tarvittavat aukot (engl. *slots*) on täytetty, voidaan aikomus määritellä onnistuneeksi, ja tästä voidaan lähettää tieto aikomuksen onnistumisesta käyttäjälle, sekä, tässä tapauksessa, sähköinen varaus hotellin omaan järjestelmään. (Amazon Lex: Developer Guide 2020, 3-4.)

Amazon Sumerian

Amazon Sumerian on AWS:n palvelu, jolla voidaan luoda korkealaatuisia virtuaalitodellisuuden (engl. *virtual reality*, *VR*), lisätyn todellisuuden (engl. *augmented reality*, *AR*) tai 3D-sovelluksia selaimella. Luominen on mahdollista ilman aikaisempaa ohjelmointitaustaa, koska *Sumerian* sisältää suuren kirjaston valmiita resursseja; tosin myös omien resurssien tuonti on myös mahdollista. *Sumerianin* editorin sisältämät työkalut mahdollistavat vuorovaikutuksen käyttäjän ja sovelluksen välillä sekä erilaisten tapahtumien ja animaatioiden tekemisen valmiilla palikoilla. Myös oman koodin kirjoittaminen on mahdollista. Editorissa voidaan muokata ja lisätä toiminnallisuuksia täysin omien tarpeiden mukaisesti käyttäen selaimen skriptauskieltä, *JavaScriptia*. (Amazon Sumerian: User Guide 2020, 1-2.)

Sumerian sovelluksen julkaiseminen luo staattisen nettisivun Amazon *CloudFront* -palveluun. Sovellukseen pääsee käsiksi sen jälkeen joko suoraan sovelluksen osoitteesta tai upottamalla sovellus toiselle nettisivulle. Koska sovellus toimii selaimessa, sitä voi käyttää tietokoneella, mobiililaitteella sekä suurimmalla osasta VR-laitteista. Sovelluksen kehitystä sekä käyttämistä varten selaimen täytyy olla *WebVR* tai *WebXR* yhteensopiva. (Amazon Sumerian: User Guide 2020, 1-2, 57.)

AWS Lambda

AWS *Lambda* on palvelu, jolla voi ajaa ohjelmalohkoja pilvessä ilman, että tarvitsee varata tai pitää huolta palvelimista, joissa koodilohko suoritetaan. Tarvittava koodi ajetaan vain, kun *Lambda*-funktiolle niin kerrotaan. *Lambda* voi virtuaalisesti ajaa koodia mille tahansa sovellukselle ilman, että käyttäjä joutuu huolehtimaan ylläpidosta. Esimerkkitapauksia sen käyttökohteista on muiden palvelujen laukaisijat, kuten uuden tiedoston tallennus AWS:n *S3*-palvelun ämpäriin (engl. *bucket*) tai uuden rivin lisääminen AWS:n *DynamoDB*-palvelun tauluun. Toisina käyttötapauksina on esimerkiksi vastaukset eri palveluiden *http*-kyselyihin. (AWS Lambda: Developer Guide 2020, 1.)

Lambda onkin yksi AWS:n suosituimmista palveluista. Gladys Rama (2020) kertoi artikkelissaan, että *”Lambda on kehittynein ja eniten käytetty palvelimeton alusta Datadog-palvelussa, jota melkein 50 % AWS-käyttäjistä käyttää”.* Uusia funktiota voidaan luoda suoraan *Lambda*-palvelun käyttöliittymässä, jos ohjelmointikieli ei tarvitse erillistä kääntämistä. Esimerkkejä ohjelmointikielistä, joita ei tarvitse erikseen kääntää ovat yleisimmät kielet, kuten *Node.js* tai *Python*. *Lambda*-funktiolle voidaan kirjoittaa erilaisia yksikkötestejä tai testitapauksia, jossa sitä voidaan manuaalisesti ajaa halutulla datalla. Toinen tapa koodilohkon viemiseen on esimerkiksi AWS:n komentokehotteen (engl. *AWS Command Line Interface*) käyttö. (AWS Lambda: Developer Guide 2020, 8, 14.)

Amazon DynamoDB

AWS:n *DynamoDB* on *NoSQL*-tietokantapalvelu, joka mahdollistaa datan tallentamisen sekä hakemisen nopeasti, kuten myös ennustettavan suoristuskyvyn hyvällä skaalautumisella. *DynamoDB*:n avulla kehittäjän ei tarvitse huolehtia laitteiston asennuksista, kokoonpanosta, ohjelmistojen korjaamisesti tai skaalautumisesta. Sen käyttö mahdollistaa tietokantataulukoiden luomisen, jotka voivat tallentaa sekä hakea rajattoman määrän dataa ja palvella kaiken tasoista pyyntöliikennettä. Hyvän skaalautuvuuden vuoksi se kykenee taulukoiden kapasiteetin suurennukseen sekä pienennykseen ilman suorituskyvyn heikkenemistä tai katkoja. (Amazon DynamoDB: Developer Guide 2020, 1.)

DynamoDB-palvelu koostuu pääasiassa tauluista (engl. *table*), esineistä (engl. *item*), sekä attribuuteista. Tieto tallennetaan tauluihin, jotka sisältävät kokoelman dataa. Data varastoidaan tauluissa sijaitseviin nimikkeisiin (engl. *item*), jotka sisältävät rajattoman määrän kokoelmia eri attribuuteista. Esimerkkinä taulu voisi olla nimeltään ”talo”, joka sisältään nimikkeitä eri taloista. Esimerkillisiä attribuutteja talolle voisi olla ”rakennusvuosi”, ”koko” ja ”hinta”. Nimikkeet sisältävät aina yksilöivän attribuutin, jota kutsutaan *DynamoDB*:ssä nimellä pääavain (engl. *primary key*). Pääavainta käytetään eritoten datan hakulausekkeissa, jossa voidaan tarkasti määritellä, mikä nimike halutaan palauttaa. (Amazon DynamoDB: Developer Guide 2020, 2-3.)

Amazon Translate

Amazon *Translate* on tekstin kääntämispalvelu, joka käyttää edistyneitä koneoppimistekniikoita korkealaatuisen käännöksen tekemiseen. Näissä koneoppimistekniikoissa on mallinnettu kielen kääntämisen käytettäviä neuroverkkoja. Palvelun avulla käyttäjille voidaan tarjota monikielinen käyttökokemus integroimalla se muihin sovelluksiin, jolloin se voi kääntää esimerkiksi raportteja, julkaisuja tai asiakaspalvelukeskusteluita. Käyttämällä lisäksi Amazon toista palvelua nimeltään *Comprehend*, voidaan käännettävästä tekstistä jättää kääntämättä eri avainsanastoa, kuten ”Kela”. Palvelu tukee myös englannin kielen kääntämistä suomen kieleksi. (Amazon Translate: Developer Guide 2020, 1-3.)

Translate-palvelun neuroverkkojen avulla käyttäjä voi valita laajasta listasta niin tulevan, kuin käännettävän kielen. Toiminnan keskipisteenä toimii kaksi komponenttia*, enkooderi* sekä *dekooderi. Enkooderi* lukee tekstin syötön jälkeen lausekkeen sana kerrallaan ja luo siitä semanttisesti edustavan kokonaisuuden, josta se koittaa päätellä lausekkeen tarkoituksen. Tämän jälkeen *dekooderi k*äyttää kyseistä kokonaisuutta luodessaan käännetyn lauseen. Lausekkeen kääntämistä on paranneltu käyttäen Amazonin tarkkaavaisuusmekaniikkaa (engl. *attention mechanisms*), jonka avulla dekooderi osaa keskittyä sanoihin, joilla on eniten merkitystä lausekkeen päätarkoitukselle. (Amazon Translate: Developer Guide 2020, 4.)

AWS tarjoaa *Translate*-palvelun kanssa valmiin integraation tiettyihin palveluihin automaattisesti, joita on muun muassa Amazon *Polly*, Amazon *S3* ja AWS *Lambda*. Tällä

hetkellä sitä ei voi suoraan integroida *Sumerian*-palveluun, mutta tähän voidaan välikätenä viedä esimerkiksi *Lambda*-funktio, joka hoitaa keskustelun näiden kahden palvelun välillä. (What Is Amazon Translate? n.d.)

AWS EC2

AWS *EC2* (tunnetaan myös nimellä *Amazon Elastic Compute Cloud*) tarjoaa skaalautuvan ja laskentakapasiteetillisen pilvipalvelun AWS:ssä. *EC2*:lla on mahdollista käynnistää virtuaalipalvelimia, joille voidaan vapaasti konfiguroida suojaus, verkkoyhteydet sekä tallennustila. Kyseinen palvelu poistaa tarpeen investoida oikeaan laitteistoon sekä tekee palvelimien hallinnasta helpompaa. (Amazon Elastic Compute Cloud: User guide for Linux Instances 2020, 1.)

EC2-palvelu tarjoaa valmiita levykuvia (engl. *Amazon Machine Image*, *AMI*), jotka ovat eräänlaisia pohjia sisältäen valmiiksi konfiguroitua informaatiota, kuten käyttöjärjestelmän, sovelluksia ja sovelluspalvelimen. Tästä levykuvasta käyttäjä voi käynnistää uuden instanssin, jonka aikana kopio levykuvasta viedään virtuaaliselle pilvipalvelimelle, jossa se suoriutuu. Yhdestä levykuvasta voidaan luoda monia eri instansseja, jotka eroavat eritoten muistin määrässä ja laskentatehossa. (Amazon Elastic Compute Cloud: User guide for Linux Instances 2020, 4-5.)

3.2 Googlen pilvipalvelut

Vaikka tutkimuksen päätavoite oli tutkia, kuinka hyvin AWS:n palvelut soveltuvat virtuaalisen asiakaspalveluneuvojan tekoon, olimme ottaneet tietoperustaan mukaan myös Google *Cloudin*. Tämä oli tärkeä vaihe kehitykselle, koska halusimme osoittaa, kuinka AWS:n virtuaalinen asiakaspalveluneuvoja voidaan lokalisoida suomen kielelle, käyttäen ulkopuolisia palveluita.

Google *Cloud* on pilvipalvelualusta, joka tarjoaa erilaisia pilvipalveluita käyttäjilleen. *Cloud* koostuu joukoista fyysisiä resursseja, kuten palvelimista ja kiintolevyistä. Samaan kokonaisuuteen kuuluvat myös virtuaaliset resurssit, kuten virtuaalikoneet, jotka sijaitsevat Googlen palvelinkeskuksissa ympäri maailmaa. Jokaisen käytetyn palvelun täytyy kuulua yhteen projektiin. (Google Cloud overview 2020).

Google: Speech-to-Text

Googlen *Speech-to-Text*-palvelu mahdollistaa puheen muuntamisen tekstiksi käyttäen hyväksi Googlen tekoälyteknologiaa. Se tukee kirjoitushetkellä jopa 125 eri kieltä. Puheentunnistusta voidaan mukauttaa tunnistamaan esimerkiksi harvinaiset sanat tarjoamalla vinkkejä tai parantamalla tiettyjen sanojen tai lauseiden tarkkuutta (Speech-to-Text N.d.)

Kyseiseen palveluun voi lähettää puhetta kolmella eri tavalla. Yksinkertaisin tapa on lähettää kysely palvelun *API*-rajapintaan käyttäen synkronista tunnistusta (engl. *synchronous recognition*). Saman kyselyn aikana *API* palauttaa lähetetyn äänitiedoston tekstiksi käännettynä. Kyseisen tavan haittapuolena ovat rajoitukset pituuteen; äänitiedosto saa kestää maksimissaan vain yhden minuutin, jonka takia se soveltuu parhaiten lyhyisiin ja yksinkertaisiin äänitiedostoihin. Jos lähetettävää ääntä on paljon, on parempi käyttää asynkronista tunnistusta (engl. *asynchronous recognition*), jolloin käännettävän äänen tuloksia voidaan tiedustella ajoittain. Tämä vaihtoehto sopii pitempiin äänitiedostoihin, koska sen kokonaiskesto saa olla maksimissaan 480 minuuttia. Viimeisenä vaihtoehtona palvelu tarjoaa reaaliaikaista tunnistusta (engl. *streaming recognition*), joka mahdollistaa tulosten näkymisen jo ennen kuin käyttäjä lopettaa puheensa. (Speech-to-Text basics n.d.)

3.3 AWS:n globaali infrastruktuuri

Amazonilla on useita datakeskuksia ympäri maailman, jotka muodostavat globaalin infrastruktuurin Amazonin pilvipalveluille. AWS tarjoaa asiakkailleen palveluita ajasta ja paikasta riippumatta. Amazonilla on maailman suurin ja dynaamisin pilvipalveluiden ekosysteemi, jolla on miljoonia aktiivisia käyttäjiä sekä kymmeniä tuhansia kumppaneita. (Global Infrastructure n.d.)

Infrastruktuuri perustuu alueisiin (engl. *region*) sekä saatavuusalueisiin (engl. *availability zone*). Nämä sisältävät useita eri datakeskuksia, joissa palvelut sisältävät palvelimet sijaitsevat. Yksi AWS:n datakeskuksien rykelmä, eli alue, on rykelmän fyysinen alueellinen sijainti maailmassa. Näitä alueita on tällä hetkellä 24 kappaletta ympäri maailmaa ja määrä kasvaa jatkuvasti. Niitä löytyy muun muassa Pohjois- ja Etelä-

Amerikasta, Euroopasta, Kiinasta, Aasian Tyynenmeren alueelta, Etelä-Afrikasta sekä Lähi-Idästä. (Regions and Availability Zones n.d.)

Alueet sisältävät useampia saatavuusalueita (engl. *availability zones*, *AZ*), joita on tällä hetkellä 77 kappaletta. Saatavuusalue sisältää myös yhden tai useamman erillisen datakeskuksen, jolla on muista keskuksista riippumaton sähkö, verkosto sekä liitettävyys. Näitä hyötyjä verrattuna yhden datakeskuksen konseptiin ovat muun muassa parempi saatavuus, vikasietokyky sekä skaalautuvuus. Esimerkiksi sovellus voidaan jakaa useamman saatavuusalueen välillä, jolloin sen toiminta ei lakkaa, vaikka yhteen datakeskukseen kohdistuisi vakava häiriö, kuten luonnonilmiö tai sähkökatkos. (Regions and Availability Zones n.d.)

3.4 Suomen kielen tuki Amazonin palveluissa

AWS tarjoaa tällä hetkellä viittä erilaista koneoppimispalvelua liittyen eri kieliin. Nämä ovat Amazon *Comprehend*, Amazon *Transcribe*, Amazon *Translate*, Amazon *Polly* sekä Amazon *Lex*. Palveluiden avulla voidaan kääntää, tunnistaa tai muokata tekstiä sekä puhetta. (Language services for AI n.d.)

Edellä mainittujen palveluiden ainut yhteinen tuettu kieli on englanti. Suppein tuki on Amazon *Lex*-palvelulla, joka tukee virallisesti ainoastaan englantia (Amazon Lex: Developer Guide, 4). Laajin kielten tuki on tekstin kääntämiseen tarkoitetulla Amazon *Translate*-palvelulla. *Translate* tukee tällä hetkellä 55 eri kieltä tai niiden varianttia (Amazon Transalate Features n.d.). Translate on myös yksi harvoista palveluista, joka tukee suomen kieltä (Amazon Translate: Developer Guide 2020, 1-3). Myös Amazon *Comprehend* tukee suomen kieltä, mutta tuki on vain osittainen. Suomen kielen tuki on ainoastaan vallitsevan kielen tunnistamisessa, joka on vain yksi *Comprehend*-palvelun ominaisuuksista (Amazon Comprehend: Developer Guide 2020, 5-6).

Suomen kielen heikko tuki ei ole kuitenkaan estä suomenkielisen sovelluksen kehittämistä. Useita eri palveluja käyttämällä on mahdollista saada näyttämään siltä, että suomen kielen tuki olisi olemassa. Tämän kehittämistutkimuksen tuloksena on yksi esimerkki siitä.

3.5 Aikaisemmat tutkimukset

AWS:n chatbotin mahdollisuuksista, etenkin Amazon *Lexin* kanssa, on jo tutkittu jonkun verran. Eritoten on tutkittu sitä, kuinka hyvin se soveltuu geneerisen chatbotin tekoon. Aikaisemmissa tutkimussa on myös todettu, ettei chatbotin teko suomen kielellä ole sillä hetkellä mahdollista, sillä *Lex* tukee tälläkin hetkellä vain englannin kieltä - niin tekstissä, kuin puheessa.

Eritoten *Lexiä* on tutkittu siltä kannalta, kuinka hyvin se soveltuu chatbotin luomiseen, samalla vertaillen muita suuria pilvipalvelualustoja, jotka tarjoavat samaa toiminnallisuutta, kuten Microsoftin *Azure* ja Googlen *Cloud*. Näissäkin tapauksissa on vain huomautettu, ettei *Lex* tue kuin englannin kieltä.

Sitä, kuinka *Lex* voitaisiin saada tukemaan suomen kieltä esimerkiksi käyttämällä Amazon *Translate* -palvelua, ei ole aikaisemmin tutkittu. Loppujen lopuksi teksti, joka *Lexille* voidaan syöttää, pitää olla englannin kieltä, eikä sitä voi vaihtaa. Kielen kääntämisessä ja sen tulkitsimisessa jälkeenpäin voi helposti syntyä virheitä varsinkin monimutkaisemmissa lauseissa. Asiakaspalveluneuvojan toimivuutta voidaan kuitenkin tehostaa ohjaamalla käyttäjä tekemään mahdollisimman yksinkertaisia kysymyksiä.

4 Virtuaalisen asiakaspalveluneuvojan toteutus Kelalle

Tässä kappaleessa käymme läpi virtuaalisen asiakaspalveluneuvojan toteutusta sekä kehitystä. Aloitimme kehitysprosessin suunnittelemalla mitä palveluita tarvittaisiin, jotta virtuaalinen asiakaspalveluneuvoja saadaan toteutettua. Opinnäytetyön toimeksiantajan toive oli, että hyödyntäisimme projektissa AWS:n palveluita, joten ne muodostivat projektin pohjan. Tämän jälkeen suoritimme työnjaon, jossa jaoimme projektin vastuualueet niin, että toinen vastasi pääasiassa *Sumerian*-käyttöliittymän kehityksestä, ja toinen *Lexin* sekä *Lambdan* logiikasta sekä tiedonkulusta. Seuraavaksi esittelemme tarkemmin kunkin työvaiheen sisältöä.

Aloitimme asiakaspalveluneuvojan rakentamisen Amazon *Lex*in avulla, jossa mietimme mahdollisia kysymyksiä sekä asioita, joita Kelan asiakas haluaisi kysyä tai tehdä. Nämä kysymykset jaoteltiin *Lexin* omiin aikomuksiin, joita olivat muun muassa ajanvaraus, reittiohjeet lähimpään Kelan toimistoon, sekä asiakkaan tietojen näyttäminen. Näistä kerromme tarkemmin tuonnempana. Jokainen aikomus, johon tarvittiin ylimääräistä logiikkaa, yhdistettiin *Lambda*-funktioon. Nämä funktiot suorittivat annetulle datalle tarpeellisen validoinnin, jonka jälkeen voitiin lähettää takaisin *Lex*-palvelulle sopiva vastaus. Jakamalla mahdolliset kysymykset omiin osiin saimme pilkottua kokonaisuutta helposti.

4.1 Käyttöliittymä

Virtuaalisen asiakaspalveluneuvojan käyttöliittymä sisältää useita erilaisia sekä itsenäisiä kokonaisuuksia (engl. *entity*), joista tärkeimmät ovat syötepainikkeet keskustelua varten, aikomus- ja vastausalueet sekä *Sumerian*-host eli itse asiakaspalveluneuvojan kasvot. Koska kyseessä on puhekäyttöliittymä, on chatbotin kanssa mahdollista keskustella puheen avulla. Käytettävyyden ja testaamisen helpottamiseksi lisättiin mahdollisuus keskustella myös tekstin avulla.

Syötteiden eroavaisuuden lisäksi myös niiden painikkeiden toimintatavat eroavat toisistaan. Puhuminen tapahtuu painamalla mikrofonipainiketta koko puhumisen ajan, kun taas tekstisyötteen painike toimii ainoastaan tekstisyötelaatikon avaamiseen ja sulkemiseen. Tekstillä keskusteleminen tapahtuu avaamalla tekstisyötelaatikko ja kirjoittamalla siihen normaalin viestin tavoin. Molemmissa tapauksissa viesti prosessoidaan, jonka jälkeen käyttäjä saa vastauksen kolmella eri tavalla. Nämä ovat asiakaspalveluneuvojan puhe, tekstitys puheelle sekä mahdollisesti toimintoruudun tilalle tuleva vastausruutu.

Oletuksena käyttöliittymässä näytetään mahdolliset toiminnot, eli aikomukset (engl. *intents*), joiden tarkoitus on kertoa käyttäjälle mahdolliset sovelluksen toiminnot. Nämä toiminnot sekä näytettävät vastaukset ovat *HTML*-elementtejä. Kun käyttäjä puhuu *Sumerian*-hostille, eli aloittaa jonkun mahdollisista aikomuksista, aikomus-

ruutu poistuu näkyvistä. Tämän tilalle ruudulle tulee tilanteen mukaan joko kysymyksiä, vaihtoehtoja tai vastaus käyttäjälle. Kun käyttäjä on saanut toiminnon tehtyä, käyttäjälle näytetään taas kaikki mahdolliset toiminnot.

Kuvio 1. Kuvakaappaus *Sumerian-host* -sovelluksen käyttöliittymästä

4.2 Sumerian-host

Sovelluksen tärkeimmässä osassa on chatbot-logiikkaan integroitu virtuaalinen asiakaspalvelija *Sumerian*-host. Host on *Sumerianin* tarjoama resurssi, joka sisältää sisäänrakennettuja ominaisuuksia, kuten kommunikoinnin puheen avulla, animaatioita sekä erilaisia toimintoja ja käytöksiä, joiden avulla se voi olla vuorovaikutuksessa käyttäjän kanssa (Amazon Sumerian: User Guide 2020, 229). *Sumerianin* kirjastosta on mahdollista ladata eri näköisiä sekä sukupuolisia hosteja. Ominaisuuksiltaan kaikki hostit ovat samanlaisia, ainoastaan ulkonäössä ja vaatteissa on eroja. Hostin vaatteisiin on mahdollista tehdä myös pieniä muutoksia, kuten vaihtaa vaatteen väriä tai lisätä kuva. Tästä esimerkki on lisäämämme Kelan logo asiakaspalveluneuvojan paidassa, jonka voi huomata kuviosta 1.

Yksi hostin tärkeimmistä ominaisuuksista on puhuminen. Host käyttää puhumiseen *SSML* (Speech Synthesis Markup Language) -kieltä, joka mahdollistaa puhumisen lisäksi hostin animaatiot eleitä varten. Host pystyy siis eleiden avulla esimerkiksi näyttämään erilaisia tunteita. Puhumista varten on tehty lukuisia erilaisia ääniä, joista voi

valita sopivamman puhekomponentin omaa tarkoitusta varten. Ääni on valittavissa sekä mies- että naisäänille. Lista sisältää myös erilaisia aksentteja.

Hostille voi tehdä omia puhetiedostoja tai se voi käyttää ainoastaan Amazon *Lex*-palvelun palauttamia vastauksia. Chatbot-ominaisuutta varten hostille pitää lisätä dialogikomponentti, jossa määritellään mihin *Lex*-palvelun chatbottiin host halutaan yhdistää. Amazon *Polly*-palvelu hoitaa hostin puheäänen tuottamisen, joka on integroitu siihen valmiiksi.

Vuorovaikutusta varten Sumerian hostin tärkeimmät komponentit ovat skriptit sekä toimintojen tila (engl. *state machine*) -komponentit. Ne mahdollistavat muun muassa sovelluksen vuorovaikutuksen käyttäjän kanssa. Esimerkiksi toimintojen tilakomponentin avulla on mahdollista tehdä koko keskustelun prosessi valmiiksi tehdyillä palikoilla. Tutkimuksen tapauksessamme ongelmaksi tuli kieli, jota emme pystyneet muuttamaan käyttämällä valmiita komponentteja. Oman koodin kirjoittaminen skriptauskomponentissa mahdollistaa kuitenkin tekemään kaiken saman omilla muutoksilla, jolloin mahdollisuudet ovat lähes rajattomat.

4.3 Keskustelun prosessi

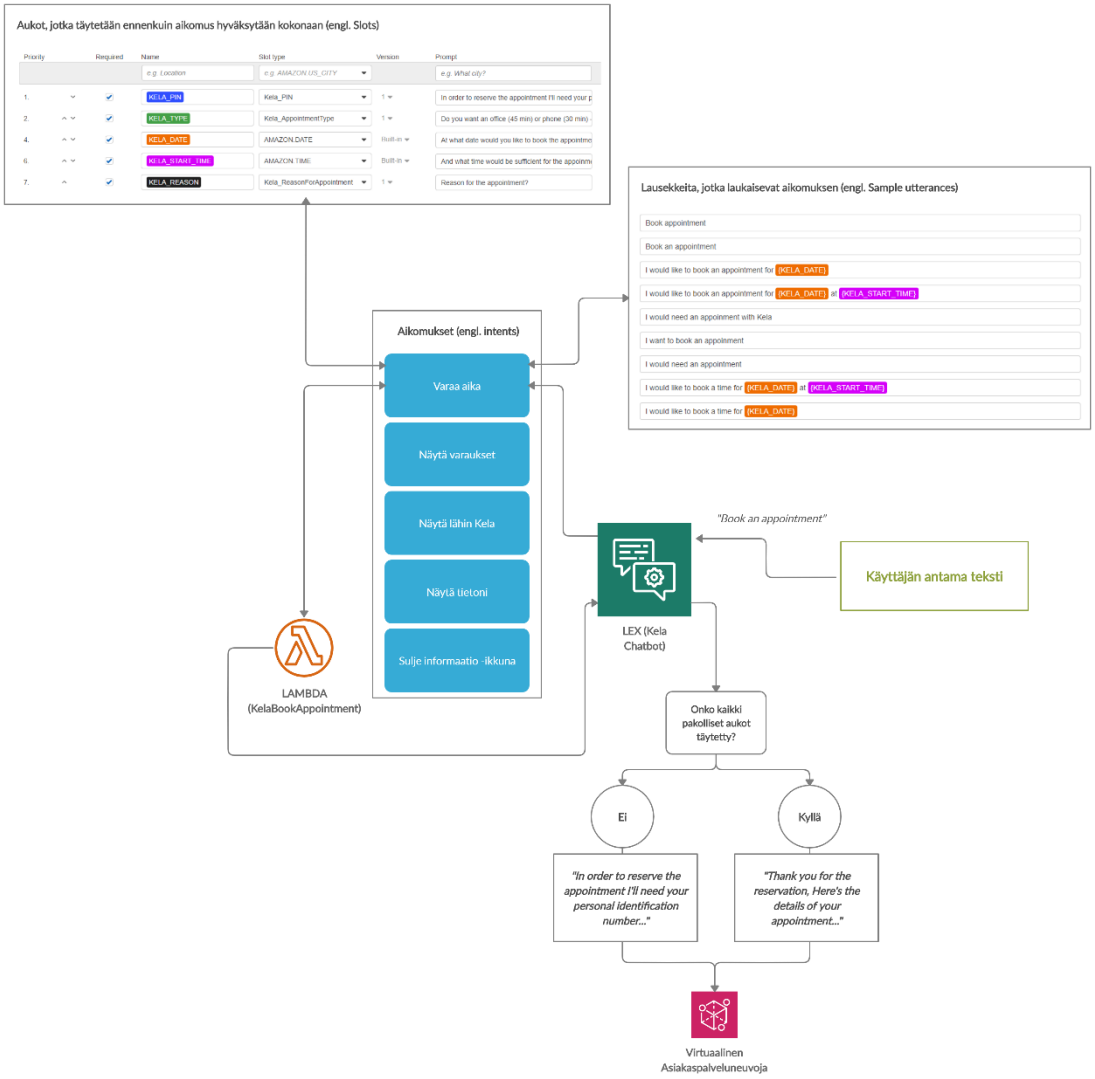
Keskustelun prosessilla tarkoitetaan asiakkaan sekä virtuaalisen asiakaspalveluneuvojan välillä tapahtuvaa interaktiota. Prosessi alkaa, kun asiakas joko kirjoittaa tai puhuu kysymyksen *Sumerian*-palvelun käyttöliittymässä. Tässä kohtaa on hyvä huomauttaa, että projektin alkuvaiheessa *Lex* käsitteli vain englanninkielistä puhetta tai tekstiä. Lopullisessa versiossa *Lex*-palvelulle lähetetään molempien vaihtoehtojen sijaan vain englanninkielistä tekstiä ja puheen prosessointi sekä parsiminen tapahtuu *Lex*-palvelun ulkopuolella. Selkeyden vuoksi käsittelemme prosessia vain sen lopullisessa muodossa.

Sumerian-palvelusta tuotu käyttäjän teksti tuodaan automaattisesti *Lex*-palvelulle, josta se siirretään oikeaan aikomukseen (engl. *intent*). *Lex*-palvelun eri aikomuksia sekä tekoälyä voidaan harjoittaa antamalla sille esimerkkisanontoja. Jos esimerkkinä aikomuksella on valmiiksi lausekkeita, kuten *”Book appointment”, ”I want to reserve*

time” tai *”I need to see Kela’s employee”*, voi *Lex* ymmärtää käyttäjän oikean aikomuksen jo sanoista *”time”* ja *”appointment”*. Pienemmissä määrissä tämä voidaan vielä todeta toimivaksi, mutta kun aikomuksien määrä kasvaa, voi *Lex* helposti sekoittaa samantapaiset aikomukset keskenään.

Jokaiselle aikomukselle voidaan lisätä rajaton määrä aukkoja (engl. *slots*), jotka voivat joko olla pakollisia tai vapaaehtoisia. Pakolliset aukot tulee täyttää informaatiolla ennen kuin aikomus voidaan hyväksyä kokonaan. Useimmiten aukkojen tarkistamisessa sekä niiden hyväksynnässä käytetään L*ambda*-funktioita, koska ilman niitä informaation validointi on hankalaa.

Kuten kuviosta 2. voidaan todeta, aikomus käynnistetään, kun käyttäjältä saapuu haluttu aikomus tekstimuodossa. Aikomuksen valinnan jälkeen, aletaan sen aukkoja katsomaan järjestelmällisesti ylhäältä alas. *Lex* kutsuu tässä vaiheessa ensimmäisen kerran *Lambda*-funktiota, joka on alustava kutsu, jossa eri tietoja voidaan alustaa tuleville interaktioille automaattisesti. Jokaisessa *Lexin* kutsussa *Lambda*-funktiolle tuodaan mukana myös tarvittavia metatietoja prosessin kulusta, kuten hyväksytyt aukot, tallennetut muuttujat sekä missä kohdassa aikomusta ollaan menossa. Jokaisesta aukosta lähetään erikseen määritelty viesti takaisin *Sumerian* hostille, joka, käyttäen hyväksi AWS *Polly* -palvelua, muuntaa tekstin puheeksi käyttäjälle.



Kuvio 2. Esimerkki *Lexin* prosessista ajanvarauksessa

Kun jokainen pakollinen aukko on täytetty ja käyttäjä hyväksyy lopullisen aikomuksen, hyväksytään aikomus ja tehdään määritellyt toimenpiteet *Lex*-palvelun ja *Lambda*–funktion puolella. Esimerkkitapauksessamme kyseinen tapaaminen tallennetaan *DynamoDB*-tietokantapalveluun.

Kun yllä mainittua prosessia katsotaan käyttäjän näkökulmasta, hän näkee käyttöliittymässä muutama esimerkkikysymyksen, joista hän voi valita haluamansa kysymyksen. Sumerian-host pitää huolen siitä, että eri aikomuksissa jokainen tarvittava aukko täytetään. Jokaisen aikomuksen lopussa käyttäjältä kysytään, ovatko tiedot varmasti oikein ja haluaako käyttäjä jatkaa aikomuksen varmistukseen. Esimerkiksi ajanvarauksessa käyttäjälle näytettäisiin ajanvarauksen kaikki tiedot ennen hyväksymistä.

Jos käyttäjä haluaa peruuttaa aikomuksen ennen sen viimeistelyä, viedään käyttäjä takaisin alkupisteeseen, jossa hän voi uudelleen valita uuden kysymyksen. On tapauksia, jolloin *Lex* ei ymmärrä käyttäjän aikomusta. Näissä tapauksissa käyttäjää pyydetään esittämään kysymyksensä uudelleen.

4.4 Lambda-funktiot ja Serverless-rajapinta

Kehityksen alussa mietimme vaihtoehtoja, joilla voitaisiin helpoiten siirtää koodimuutoksia AWS:n palveluun. Eniten ohjelmointia tarvitsivat *Lambda*-funktiot, jotka sisälsivät päälogiikan asiakaspalveluneuvojan käyttäytymisessä. Tässä päädyimme käyttämään *Serverless*-rajapintaa, jonka avulla muutoksien puskeminen AWS:n oli yksinkertaista sekä nopeaa. AWS:n infrastruktuuri kyseiseen tarkoitukseen on helppo toteuttaa *YAML*-tiedostolla, johon määriteltiin *Lambda*-funktiot omiin lohkoihin (kts. kuvio 3). Samaan tiedostoon voidaan myös lisätä muita tarvittavia AWS:n palveluita, kuten *DynamoDB*-taulu, joka sisälsi tässä projektissa ajanvaraukset sekä asiakkaat. *Lambda*-funktiot voidaan viedä yhdellä komennolla suoraan AWS:n palveluun. Kun käyttäjä antaa oikean komennon, *Serverless*-rajapinta lukee *YAML*-tiedoston viennin yhteydessä ja muuntaa sen AWS *CloudFormation*-palvelulle sopivaan muotoon. *CloudFormation* auttaa mallintamaan ja määrittelemään AWS-resursseja ja se toimii niin sanottuna suunnitelmana, luoden kaikki sille määritellyt resurssit (AWS CloudFormation: User guide 2020). Tämän määrittelyn avulla Lambda-funktioiden uudelleenluominen tai päivitys hoituu yhdellä komennolla.

```
functions:
  # Get user's Kela information
  KelaUserInformation:
    handler: lambdas/userInformation/userInformation.handler
    name: kela-userInformation
    description: Gets user information from DynamoDB via provided PIN

  # Book a new time for Kela meeting
  KelaBookAppointment:
    handler: lambdas/bookAppointment/bookAppointment.handler
    name: kela-bookAppointment
    description: Books an appointment for the user
```

Kuvio 3. Esimerkki *Lambda*-funktioiden määrittelystä *YAML*-tiedostoon, jonka *Serverless*-rajapinta lukee AWS:n viennin yhteydessä

Lambda-funktioiden kehityksessä päätimme käyttää *TypeScriptiä*, joka on laajennus *JavaScriptistä*. *TypeScript* mahdollisti rajapintojen tekemisen sekä tyypityksen *Lexiltä* tulevista sekä sinne lähetetyistä vastauksista. *TypeScriptin* kehitti *Microsoft*, jonka päätarkoituksena oli lisätä vahva tyypitys *JavaScriptiin*. Se lisää myös muita mahdollisuuksia, kuten rajapintaluokat, jotka ovat esimerkiksi *Javasta* tuttuja. *TypeScript* ei itsessään pakota tyypitystä, joten se voidaan ottaa käyttöön vähitellen. (Tarvainen 2016.)

4.5 Aikomukset

Seuraavaksi käsittelemme Kelan virtuaalisen asiakaspalveluneuvojan eri aikomuksia, eli osioita, joista sen kanssa voi keskustella. Aikomuksia on yhteensä 10, mutta olemme valinneet tähän niistä kaksi, koska suurin osa aikomuksista on yhden vaiheen toimintoja, joita suoritetaan yhdessä muiden aikomuksien kanssa.

Alustavassa *Lambda*-kyselyssä tarkistetaan, onko käyttäjän henkilötunnus vielä tallessa aikaisemmista kyselyistä. Jos se on tallessa, voidaan siirtyä suoraan ajanvaraus–aikomuksen aukkojen täyttöön, kun taas muussa tapauksessa aikomus aloitetaan henkilötunnuksen pyynnöllä. Lähes jokaisessa aikomuksessa käyttäjän henkilötunnus tarvitaan, jotta aikomus voidaan suorittaa loppuun. Kun käyttäjä antaa henkilötun-

nuksensa, se parsitaan sekä validoidaan ennen hyväksyntää. Parsinnan aikana annetusta henkilötunnuksesta poistetaan välilyönnit sekä muutetaan kaikki kirjaimet isoiksi kirjaimiksi. Ennen 2000-lukua syntyneillä käyttäjillä henkilötunnuksessa esiintyy viiva keskellä, mitä ei voida puheesta suoraan erotella, joten tässä muunsimme löytyvän sanan ”*viiva*” suoraan kirjaimeksi ”-”. Parsinta osaa myös valita henkilötunnuksen lauseen lopusta; esimerkkinä jos käyttäjä puhuu lauseen ”*Henkilötunnukseni on 010295 viiva 102E*”, osataan tästä päätellä, että henkilötunnus on 010295-102E (kts. kuvio 4.). Validoinnissa tarkastetaan, että henkilötunnuksen keskeltä löytyy välimerkki, joka ilmaisee henkilön syntymävuosisadan (Käsitteet: Henkilötunnus n.d.). Lopuksi henkilötunnusta koitetaan etsiä *DynamoDB*-tietokannasta. Jos kyseinen henkilötunnus löydetään, tallennetaan se sessiomuuttujiin eikä sitä kysytä enää tulevissa aikomuksissa tai kysymyksissä.

```
private isCenturySymbolInvalid(): void {
  const symbols = ["-", "A"];
  const centurySymbol = this.pin.slice(6, 7);
  for (let i = 0; i < symbols.length; i++) {
    if (centurySymbol === symbols[i]) {
      return;
    }
  }
  this.invalidPin = true;
  console.error(
    `Provided PIN: [${this.pin}] is invalid. Reason:
    PIN's 7th letter should contain either
    slash (-) or A character. Actual
    character: [${centurySymbol}]`
  );
}

private parsePinFromString(): void {
  this.pin = this.pin.replace(/ /g, '').toUpperCase();
  this.convertHyphon();
  if (this.pin.length > 10) this.pin = this.pin.substr(this.pin.length - 11);
}

private convertHyphon(): void {
  const words = ['HYPHON', 'STREAK', 'SLASH', 'LINE', 'MINUS', 'DASH'];

  for (let i = 0; i < words.length; i++) {
    if (this.pin.includes(words[i])) {
      this.pin = this.pin.replace(words[i], '-');
    }
  }
}
```

Kuvio 4. Henkilötunnuksen validointi sekä parsiminen *Lambda*-funktiossa

Ajan varaaminen

Ajanvarauksessa käytettiin pohjapiirustuksena Kelan omilta sivuilta löytyvää ajanvarausta. Kelan omassa ajanvarausjärjestelmässä asiakkaan ajanvaraukseen liitetään seuraavat tiedot: henkilötunnus, tapaamistyyppi (puhelimella tai paikan päällä), päivämäärä, aika sekä syy tapaamiselle. Käyttäjä aloittaa ajanvarauksen lausahduksilla, kuten: ”*Varaa aika*”, ”*Tarvitsen ajan*”, ”*Minun pitää varata aika*” tai ”*Minun pitää nähdä Kelan työntekijää*”.

Ajanvaraus-aikomus aloitetaan tarkistamalla, onko käyttäjän henkilötunnus jo tallessa aikaisemmista aikomuksista. Henkilötunnuksen prosessi etenee aikaisemmassa kappaleessa kuvatulla tavalla. Tämän jälkeen edellä mainitut aukot käydään yksi kerrallaan läpi, missä niiden arvot validoidaan sekä tarvittaessa parsitaan oikeanlaisiksi. Jokaisen aukon prosessi on struktuurisesti lähes identtinen toisiinsa nähden ja jokainen hyväksytty aukko tallennetaan käyttäjän sessiomuuttujiin, jotta ne säilyvät kysymyksien välillä tallessa. Käymme tässä muutaman aukon validoinnin prosessin läpi:

- Ajanvaraus alkaa tyypin määrittelyllä, joita ovat puhelinvaraus (30 minuuttia) tai varaus toimistolle (45 minuuttia). Käyttäjän lauseesta etsitään sanoja ”*toimisto*” tai ”*puhelin*”, joista päätellään ajanvarauksen tyyppi.

- Ajanvarauksen päivämäärää valittaessa varmistetaan, että seuraavat ehdot täyttyvät: valittu päivämäärä ei ole menneisyydessä, valittu päivämäärä ei ole lauantaina tai sunnuntaina, valitun päivämäärän ollessa kuluva päivä, kello ei saa olla yli klo 15:00 toimistoajanvarauksissa tai klo 15:30 puhelinajanvarauksissa.

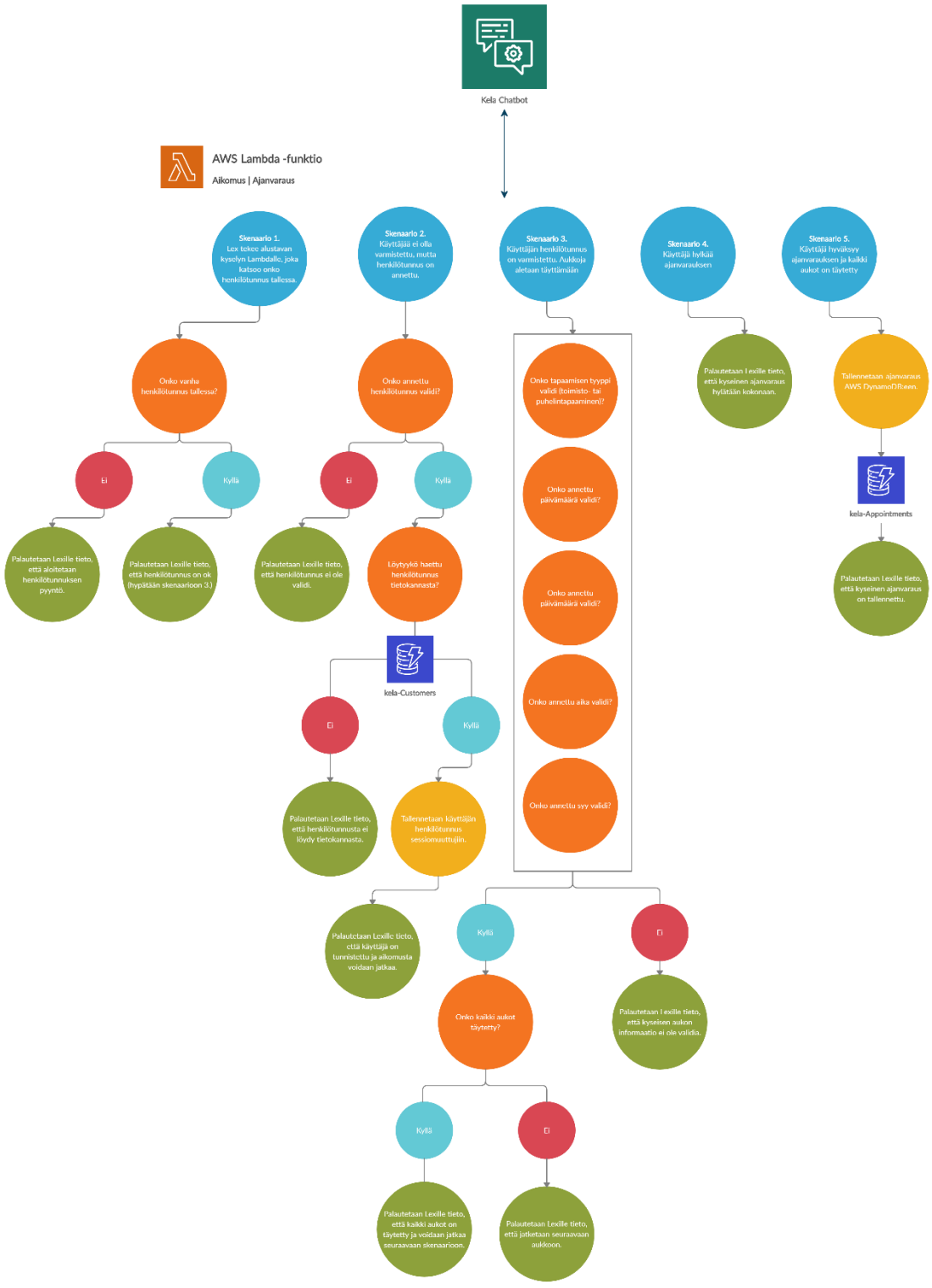
- Kun ajanvarauksen aikaa tarkistetaan, halutaan seuraavien ehtojen täyttyvän: aika ei saa alkaa ennen klo 08:00, aika ei saa alkaa klo 15:00 jälkeen toimistoajanvarauksissa tai klo 15:30 puhelinajanvarauksissa, toimistoajanvarauksissa aika saa alkaa vain tasatunteina ja puhelinajanvarauksissa 30 minuutin välein, kyseistä aikaa ei olla vielä otettu itsensä tai toisen käyttäjän toimesta.

Kun kaikki aukot on tarkistettu ja ne sisältävät validin tiedon, viedään kooste tiedoista hyväksyttäväksi käyttäjälle. Käyttäjän hyväksynnän jälkeen kyseinen kooste tiedoista viedään Lambda-funktion avulla *DynamoDB*-tietokantaan (kts. esimerkki

tietokoosteesta kuviosta 5.). Pelkistetty versio *Lambdan* ajanvarauksen toiminnasta sekä skenaarioista löytyy kuviosta 6.

```
{
  'Type': { S: "office" },
  'Pin': { S: "121295-121F" },
  'AppointmentReason': { S: "Illness" },
  'StartDateTime': { S: "2019-11-26T09:00:00" },
  'FirstName': { S: "Jaska" },
  'LastName': { S: "Jokunen" },
}
```

Kuvio 5. Esimerkki ajanvarauksen tietokoosteesta, joka viedään *DynamoDB*-tietokantaan



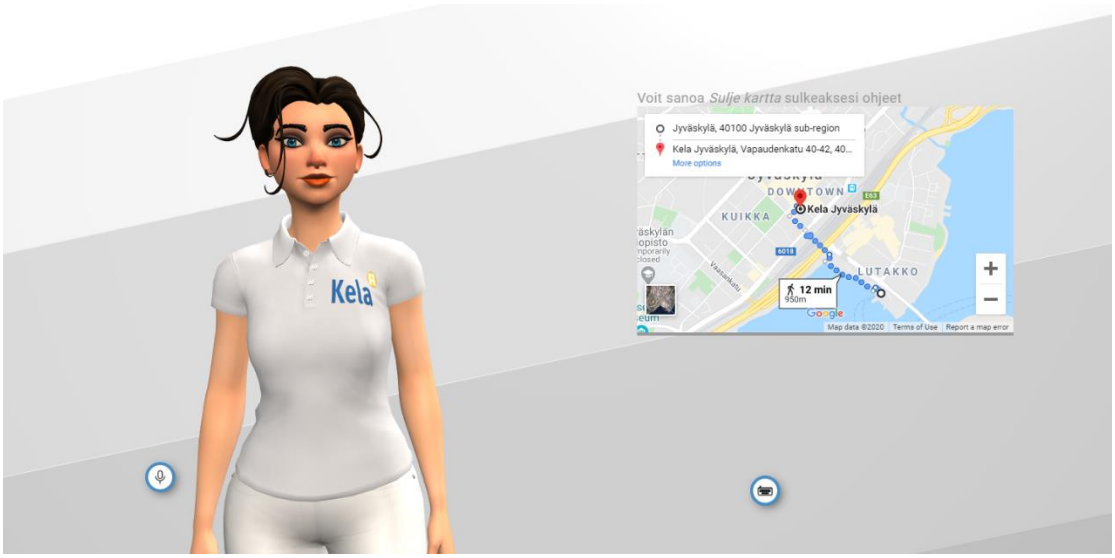
Kuvio 6. *Lambda*-funktion prosessin kulku ajanvaraus-aikomuksessa

Lähimmän Kelan toimipisteen sijaintitieto

Lähimmän Kelan toimintapisteen näyttö graafisesti toimi ylimääräisenä aikomuksena, jossa haluttiin samalla testata karttaohjeiden lähetystä sähköpostin tai tekstiviestin

avulla. Aikomuksen ainoa aukko on karttaohjeiden lähetystapa, joka on joko sähköposti tai tekstiviesti. Kyseinen aikomus voidaan laukaista esimerkiksi näillä lausahduksilla: ”*Näytä lähin Kela*”, ”*Mistä löydän lähimmän Kelan*” tai ”*Voitko näyttää lähimmän Kelan*”.

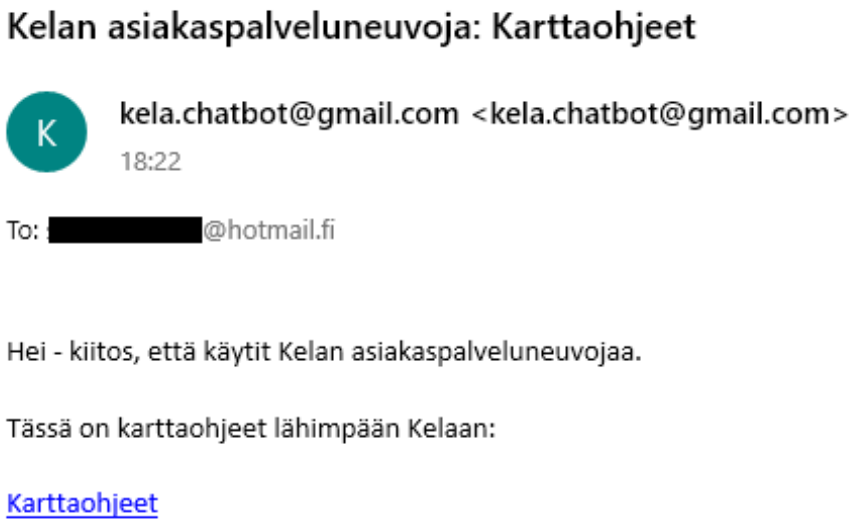
Pyytäessä tietoja lähimpään Kelan toimipaikkaan, kysytään käyttäjältä oikeudet nähdä hänen sijaintinsa. Jos käyttäjä antaa tähän oikeudet, näytetään käyttöliittymässä visuaalisesti kartta sekä nopein reitti lähimpään Kelaan käyttäen hyväksi *Google Maps* -karttapalvelua (kts. kuvio 7.). Reitti Kelalle suunnitellaan hakemalla käyttäjän *x* ja *y* -koordinaatit sekä samalla tieto lähimmästä Kelan toimipaikasta, joiden välille määritellään reitti.



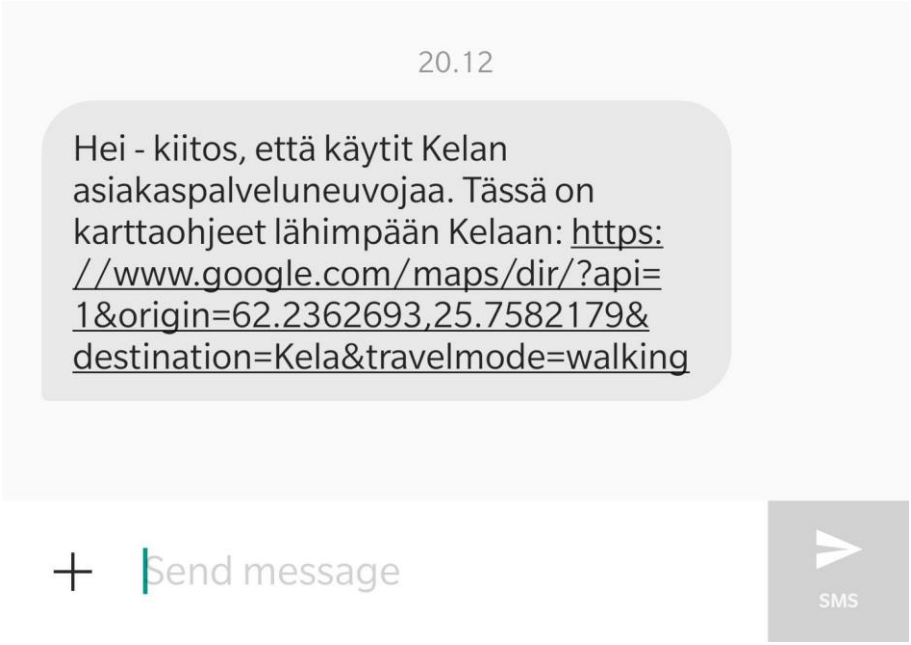
Kuvio 7. Virtuaalinen asiakaspalveluneuvoja tulostaa kartan, joka näyttää reitin lähimpään Kelan toimipaikkaan

Kartan näytön jälkeen käyttäjältä kysytään, haluaako hän, että kyseinen reitti lähetään hänelle joko tekstiviestillä tai sähköpostilla. Jos käyttäjä näin haluaa, haetaan hänen henkilötunnuksensa, tai jos sitä ei ole tallessa, pyydetään sitä häneltä. Käyttäjän puhelinnumero tai sähköposti haetaan suoraan *DynamoDB*-tietokannasta, johon käyttäjä on tallennettu. Sähköpostin lähetyksessä käytetään hyväksi AWS:n palvelua *Simple Email Service* (lyh. *SES*), jossa sähköpostiviesti voidaan muokata käyttötarkoitukseen sopivaksi käyttäen hyväksi *HTML*-merkintäkieltä (kts. kuvio 8.). Täältä sähköposti lähetään käyttäen hyväksi virtuaaliselle asiakaspalveluneuvojalle tehtyä sähköpostia. Tekstiviestin lähetyksessä käytetään toista AWS:n palvelua nimeltään *Simple*

Notification Service (lyh. *SNS*), joka on eritoten tekstiviestien lähetykseen tarkoitettu palvelu (kts. kuvio 9.).



Kuvio 8. Karttaohjeiden lähetys lähimpään Kelan toimipaikkaan sähköpostin kautta



Kuvio 9. Karttaohjeiden lähetys lähimpään Kelan toimipaikkaan tekstiviestillä

4.6 Keskustelu suomen kielellä

Yksi tutkimuksemme tavoitteista oli selvittää, onko AWS:llä toteutettu virtuaalinen asiakaspalveluneuvoja mahdollistaa lokalisoida suomen kielelle. Tutkimuksen aikana selvisi, että tämä on tekstin osalta toteutettavissa AWS:n omien palveluiden avulla. Puheen suomentamisessa jouduimme hyödyntämään kolmannen osapuolen palvelua. Tästä kerromme lisää tutkimustuloksissa. Tahdoimme kuitenkin avata myös kyseisen prosessin teknistä puolta, jota tämä kappale käsittelee.

Tutkimuksen alussa tavoitteenamme oli suorittaa suomentaminen käyttäen vain AWS:n omia palveluita. Opinnäytetyön kirjoitushetkellä *Lex* tukee vain englannin kielen puhetta ja tekstiä, joten päätimme tehdä käännöksen. Tekstin kääntämisen saimme suoritettua käyttäen AWS:n *Translate*-palvelua ennen sen lähetystä eteenpäin Lexille, mutta puheen kohdalla prosessi ei ollut niin yksinkertainen. Aiemmin mainittu *Translate*-palvelu tukee vain tekstin konvertointia. Lisäksi huomasimme, että AWS:n palvelu *Transcribe*, jota voidaan käyttää puheen muuntamisessa tekstiksi, ei tukenut vielä suomen kieltä (What Is Amazon Transcribe? n.d.).

Jatkoimme selvittämällä, mitä kolmannen osapuolen palvelua voisimme hyödyntää suomenkielisen puheen muuntamisessa tekstiksi. Meillä kummallakin oli aikaisempaa kokemusta Googlen *API*-palveluista, joten aloitimme etsinnän niiden parista. Googlelta löytyi tähän tarkoitukseen sopiva palvelu nimeltään *Speech-to-Text*, jonka päätimme valita.

Sumerianissa on saatavilla paljon erilaisia valmiita toimintoja (engl. *action*), joiden avulla voidaan suorittaa prosesseja ilman omaa koodia. Esimerkiksi puheen tai tekstin lähettäminen ja vastaanottaminen *Lex*-palvelun kanssa onnistuu ilman ylimääräisen koodin kirjoittamista. Valmiiden toimintojen käyttö on yleensä erittäin nopeaa ja tehokasta, mutta tällöin on hyvin vaikea päästä käsiksi prosessin eri vaiheisiin. Koska sovellukselle syötetty puhe ei ollut *Lexin* tukemaa kieltä eli englantia, meidän piti tehdä *Sumerianin* ja *Lexin* välille ylimääräinen prosessi, joka käy muuntamassa puheen tekstiksi ja kääntää sen sitten englannin kielelle. Jos käyttäjä syöttää tekstiä puheen sijasta, suoritetaan sama prosessi ilman puheen muuntamista tekstiksi.

Aloitimme prosessin tekemällä *JavaScriptillä* oman skriptin, joka laukaistaan, ennen kuin Sumerian normaalisti lähettäisi puheen tai tekstin *Lexille*. Käyttäjän painaessa mikrofonin kuvaa, tallennetaan puhe *Blob*-muotoon (*Binary large object*), jota käytetään usein kuvien, audion tai muiden multimediatiedostojen muotona tietokannoissa. Kyseisen *Blob*-äänitiedoston muutimme *base64String*-muotoon, eli binaaritiedostosta tekstitiedostoon. Ennen kääntämistä varten tehdyn *Lambda*-funktion kutsumista, muutimme *base64string*-muotoisen tiedoston *JSON*-muotoon (kts. kuvio 10). Tässä prosessissa oli tärkeää pitää äänitiedosto mahdollisemman natiivimuodossa, koska monet kääntämisprosessit saattavat heikentää sen toimivuutta sekä äänenlaatua.

```
function invokeLambda(base64, ctx) {
    console.log("Started translating base64string to audio file via Lambda");

    ctx.worldData.lambda = new AWS.Lambda();

    const invokeParams = {
        FunctionName: "arn:aws:lambda:eu-west-1:26xxxx468:function:kela-ConvertController",
        Payload: JSON.stringify({ base64string: base64 })
    };

    ctx.worldData.lambda.invoke(invokeParams, (error, data) => {
        if (error) alert(error);
        else {
            if (data && data.Payload) {
                const payload = JSON.parse(data.Payload);
                console.log('Payload contains:');
                console.log(payload);
                updateTextEntity(payload.msg, ctx);
            }
        }
    });
}
```

Kuvio 10. *JavaScriptillä* luotu skripti, joka lähettää *base64string*-muodossa olevan äänitiedoston *Lambdalle* ja vastaanottaa sieltä palautuvan vastauksen tekstinä

Alun perin konvertointi oli tarkoitus suorittaa käyttäen hyväksi *Lambdan* funktiota, jonka avulla olisi ollut mahdollisuus kutsua *HTTP*-kyselyllä Googlen *Speech-to-Text* -API:a ja tämän jälkeen palauttaa sieltä konvertoitu tekstipätkä takaisin *Sumerianille*. Ongelmaksi koitui *Sumerianilta* tulleen *base64string*-tekstin muuntaminen validiksi äänitiedoksi. Ainoa toimivaksi osoittautunut ohjelma tähän konvertointiin oli *FFmpeg*. *FFmpeg*-kirjasto voidaan lisätä koodiin, mutta jotta sitä voidaan käyttää siinä, tarvitaan

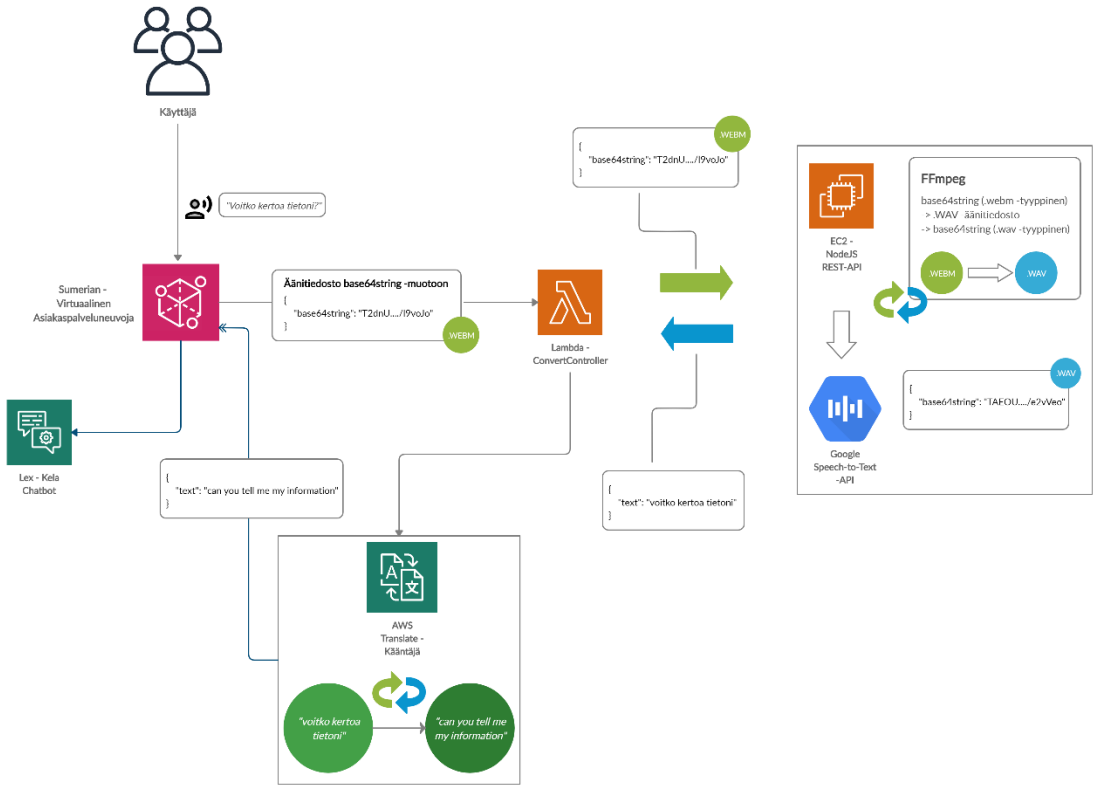
erillinen binääripaketti, joka pitää löytyä koodia suorittavalta koneelta. Lambdan funktioon kyseistä pakettia ei voitu asentaa helposti. Internet-lähteitä hyödyntämällä löysimme valmiita paketteja, jotka olisivat tämän voineet mahdollistaa kiertoreittien kautta. Nämä paketit eivät kuitenkaan enää toimineet *Lambdan* nykyisessä versiossa. Konvertointia ei siis voitu hoitaa pelkästään *Lambdan* avulla, joten muutimme sen välikappaleeksi, joka vastaanottaa pyynnön *Sumerianilta*, lähettää sen eteenpäin, vastaanottaa tekstinpätkän, muuntaa sen englanninkieliseksi tekstiksi käyttäen hyväksi AWS:n palvelua *Translate*, ja palauttaa lopulta *Sumerianille* englanninkielisen tekstinpätkän.

NodeJS REST-API

Kuten yllä todettiin, Sumerianista tulleen *base64string*-muotoisen tekstitiedoston muuntaminen tuetuksi äänitiedostoksi ei ollut *Lambdassa* mahdollista. Päätimme ratkaista ongelman luomalla virtuaalisen *Linux*-palvelimen käyttäen AWS:n palvelua *EC2*. Tämän ansioista pystyimme asentamaan tarvittavat *FFmpeg*-binääripaketit kyseiselle virtuaalipalvelimelle. Loimme palvelimelle *REST-API*:n käyttäen hyväksi *Node.js*-ympäristöä, jonka määritelimme käsittelemään vain yhtä ennalta määriteltyä *POST*-kyselyä, jonka lähdekoodi löytyy kuviosta 11. Tämä kysely vastaanottaa *base64string*-muotoisen tekstin, jonka jälkeen *FFmpeg*-kirjasto muuntaa sen validiksi *WAV*-tyyppiseksi äänitiedostoksi, tallentaen sen hetkellisesti palvelimelle. Jos muuntaminen onnistuu, muunnetaan kyseinen äänitiedosto uudestaan *base64string*-muotoon, joka on tässä vaiheessa tyypiltään natiivi *WAV*-tyyppinen äänitiedosto. Kun muunneltava äänipätkä halutaan lähettää *Speech-to-Text API*-palveluun kyselyn mukana, täytyy sen olla *Base64*-enkoodattu (Speech-To-Text: Embedding audio content n.d.). Kyselyn tuloksena *API* palauttaa suomenkielisen tekstinpätkän, joka lähetetään takaisin *Lambda*-funktiolle. Lopuksi palvelimella luotu äänitiedosto poistetaan. Prosessi on esitetty visuaalisesti tiivistetyssä muodossa kuviossa 12.

```
const FORMAT = 'wav';
const FOLDER = './audio/';
const FILE_NAME = random();
const FILE_NAME_WAV = FILE_NAME + '.wav';
const LANGUAGE_CODE = 'fi-FI';

const speech = require('@google-cloud/speech');
const fs = require('fs');
const ffmpegPath = require('@ffmpeg-installer/ffmpeg').path;
const ffmpeg = require('fluent-ffmpeg');
ffmpeg.setFfmpegPath(ffmpegPath);

app.post('/convert', (req, res) => {

  perf.start();

  if (!req.body || !req.body.base64string) {
    console.log('Base64string is null; returning 500-error');
    console.log('Execution took: ' + perf.stop().time + ' milliseconds');
    return res.status(500).send({
      error: true, msg: 'Body (base64string) was empty!' });
  }

  const base64string = req.body.base64string
    .split(';base64,')
    .pop();
  const client = new speech.SpeechClient();

  console.log('Starting the convertion of file: ' + FILE_NAME);

  fs.writeFile(
    FOLDER + FILE_NAME,
    base64string,
    { encoding: 'base64' },
    err => {
      if (err) throw err;
      console.log('New file created for conversion: ' + FILE_NAME);

      ffmpeg(FOLDER + FILE_NAME)
        .toFormat(FORMAT)
        .on('error', err => {
          console.error('Error occured while converting: ' + err.message);
        })
        .on('progress', progress => {
          console.log(JSON.stringify(progress));
          console.log(
            'Processing conversion: ' + progress.targetSize + ' KB converted'
          );
        })
        .save(FOLDER + FILE_NAME_WAV)
        .on('end', () => {
          console.log(
            `Finished converting file: ${FILE_NAME} to ${FILE_NAME_WAV}`
          );
          const audioBytes = fs
            .readFileSync(FOLDER + FILE_NAME_WAV)
            .toString('base64');

          console.log('Sending .WAV -file to Google API: ' + FILE_NAME_WAV);

          const request = {
            audio: { content: audioBytes },
            config: { languageCode: LANGUAGE_CODE }
          };

          return client
            .recognize(request)
            .then(response => {
              const convertedText = response[0].results
                .map(res => res.alternatives[0].transcript)
                .join('\n');
              console.log(
                `Successfully converted .WAV -audio file
                   (${FILE_NAME_WAV}) to text: [${convertedText}]`
              );
              removeFiles();
          console.log('Execution took: ' + perf.stop().time + ' milliseconds');
          return res.send({ error: false, msg: convertedText });
            })
            .catch(err => {
              console.error('Error while calling Google API: ' + err.message);
              removeFiles();
          console.log('Execution took: ' + perf.stop().time + ' milliseconds');
              return res.status(500).send({ error: true, msg: err.message });
            });
        });
    }
  );
});
```

Kuvio 11. Lähdekoodi, joka hoitaa konvertoinnin *base64string*-tekstistä *WAV*-äänitiedostoksi ja tämän jälkeen suomenkieliseksi tekstiksi käyttäen apuna Googlen *Speech-to-Text -API*:a



Kuvio 12. Virtuaalisen asiakaspalveluneuvojan teknisempi äänen suomentamisen prosessi

Yllä käsittelimme *Lambda*-funktion prosessia, jossa funktio suomentamisen sijaan vain palautti ja vastaanotti informaatiota. Funktiolle lisättiin ylimääräinen toiminto, jonka avulla se kutsuu *Translate*-palvelua sen jälkeen, kun se on saanut *EC2*-palvelimelta äänestä muunnetun tekstinpätkän takaisin. Tämä käännös käsittelee suomenkielisen tekstinpätkän muunnoksen englanninkieliseksi tekstiksi. *Translatelle* luotiin myös oma terminologia, jossa se ei huomio sanaa *Kela* tai sen variantteja, kuten: *kela*, *kelaan*, *kelan*, *Kelan*. Näin poistettiin mahdollisuus, jossa *Translate* koittaisi kääntää lauseissa esiintyvän *Kela*-sanan englanninkieliseksi sanaksi. Lopuksi englanninkielinen teksti lähetetään takaisin *Sumerianille*, josta se jatkaa alkuperäisen prosessin mukaan takaisin *Lexille*.

5 Tutkimustulokset

Tässä kappaleessa käymme läpi tutkimuksen tulokset sekä siihen pohjautuvat johtopäätökset. Käsittelemme myös virtuaalisen asiakaspalveluneuvojan soveltuvuutta oikeaan käyttöön, sekä vastaamme tutkimuskysymyksiin.

5.1 Soveltuvuus

Lopullinen, suomen kielellä toimiva, virtuaalinen asiakaspalveluneuvoja toimii hyvin, kun kyseessä ovat yksinkertaiset kysymykset. Todelliset asiakaspalvelutilanteet eivät kuitenkaan yleensä ole näin yksinkertaisia, vaan asiakkaan puhe on monimutkaisempaa ja sisältää esimerkiksi murre- ja slangisanoja. Asiakaspalveluneuvojan rajoitetut ilmaukset voivat aiheuttaa hankaluuksia esimerkiksi iäkkäille tai jostakin vammasta kärsivälle henkilöille. Toisaalta puhuva asiakaspalveluneuvoja helpottaa esimerkiksi sokean henkilön asiointia Kelan palveluissa. Lisäksi on otettava huomioon, että suomen kielen kääntäminen toiseen kieleen konekääntämisellä voi tuottaa aina ei-haluttuja tuloksia, varsinkin kun kyseessä ovat monimutkaisemmat sekä pidemmät lauseet. Tästä voi aiheutua väärinkäsityksiä ja turhauttavia tilanteita käyttäjälle. Tästä syystä päätimme pitää virtuaalisen asiakaspalveluneuvojan kysymykset mahdollisimman yksinkertaisina. Inhimillisen puheen moninaisuuden ja vaihtelevuuden tavoittaminen onkin jatkuva haaste tekoälyä hyödyntävissä sovelluksissa.

Riippuen kysymyksestä sekä aikomuksesta, vastausaika *Sumerianin* hostille on noin kaksi sekuntia. Ottaen huomioon, että kyseessä on tutkimuksen yhteydessä luotu tuote ja tietoa joudutaan prosessoimaan monessa eri paikassa, on tulos varsin tyydyttävä. Nykypäivän kuluttuja on kuitenkin kenties tottunut hyvin nopeaan palveluun varsinkin virtuaalisessa asiakaspalvelussa. Ei olekaan täysin varmaan, miten palvelu suoriutuisi, jos käyttäjiä olisi yhtä aikaa hyvin paljon.

Olettaen, ettei AWS:n chatbotin logiikkaan liittyviä palveluita haluttaisi vaihtaa johonkin toiseen järjestelmään, soveltuisi nykyinen infrastruktuuri välttävästi tämän tuotteen laajentamiseen. *Lambda*-funktioiden kehitykseen löytyy *Serverless*-raja-

pinta, joka mahdollistaa *YAML*-tiedoston avulla helpon laajentamisen. Myös *Sumerian*-hostin sekä käyttöliittymän jakaminen muille kehittäjille on mahdollista pakkaamalla kokonaisuus *ZIP*-paketiksi käyttäen hyväksi *Sumerianin* omaa toimintoa.

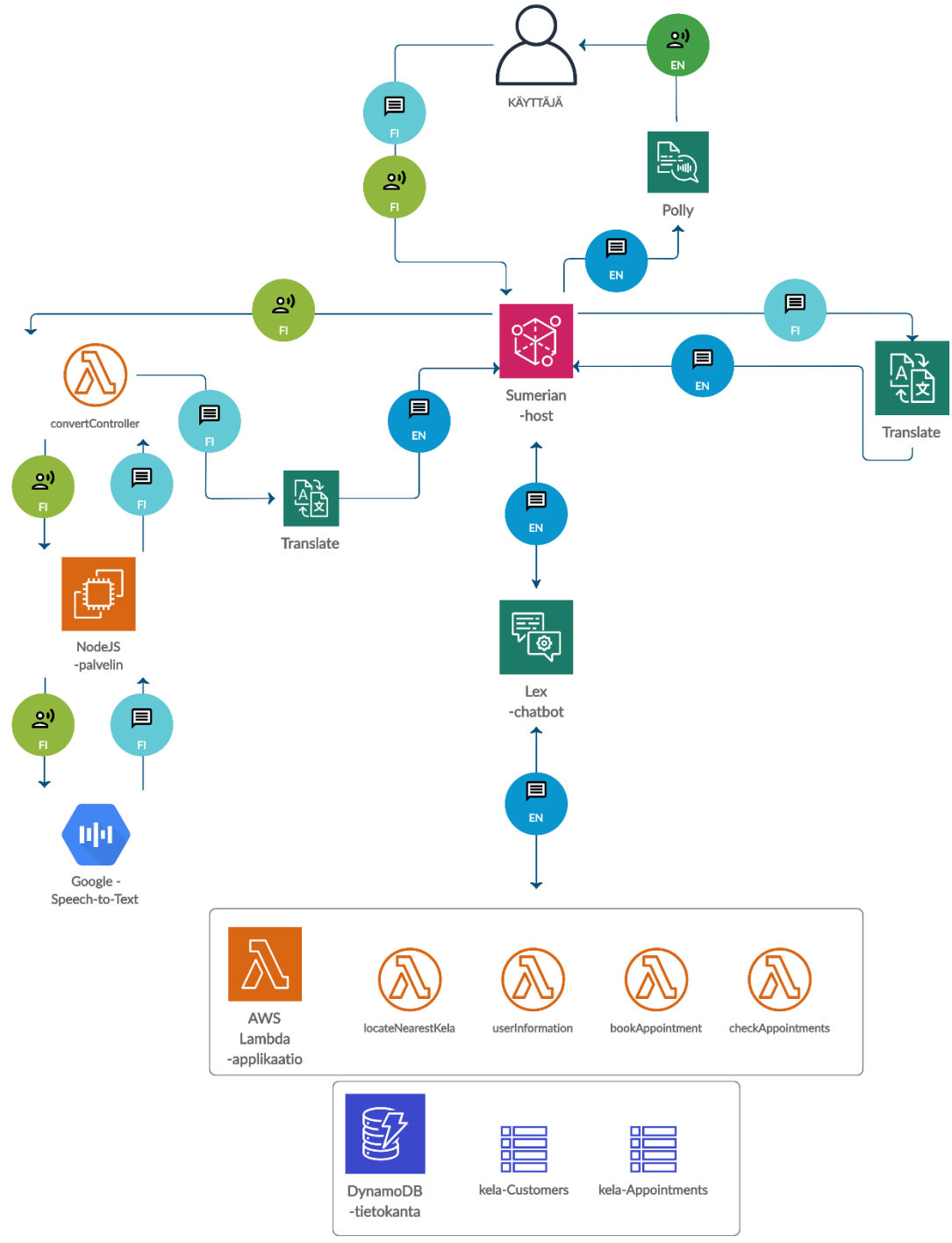
Loppujen lopuksi luodussa tuotteessa on hyvä pohja todelliseen käyttöön tulevan asiakaspalveluneuvojan luomiselle. Tällä hetkellä asiakkaan on mahdollista varata aika, löytää lähin Kelan toimipaikka visuaalisesti ohjeiden kera sekä nähdä omat henkilökohtaiset tietonsa. Todellisessa käytössä tuotteeseen tarvittaisiin vielä lisää toimintoja, kuten ajanvarausten muokkaus ja poisto sekä vastaaminen taloudellisiin ja kuntoutuksiin liittyviin kysymyksiin.

5.2 Suomenkielinen toteutus

Ensinäkemältä AWS sekä sen palvelut eivät mahdollistaneet asiakaspalveluneuvojan suomenkielistä toteutusta, koska *Lex*, joka toimii äänen sekä tekstin ymmärtämisessä sekä prosessoinnissa, ei tukenut suomen kieltä.

Tekstin suomentaminen onnistui ottamalla aluksi *Sumerianissa* vastaan suomenkielistä tekstiä, jonka jälkeen se käytettiin AWS:n oman palvelun, *Translaten* läpi, joka käänsi sen takaisin englanninkieliseksi tekstiksi. Lopulta käännetty teksti lähetettiin eteenpäin *Lexille* prosessoitavaksi. Näin käyttäjä pystyi keskustelemaan chatbotin kanssa suomen kielellä, koska konepellin alla keskustelu pysyikin englanninkielisinä.

Suomenkielisen puheen kanssa tuli haastavuuksia vastaan, koska AWS ei tarjonnut suomenkielisen puheen ymmärtämistä vielä. Tutkiessa asiaa pystyttiin huomaamaan, että Googlen oma palvelu *Speech-To-Text -API* tarjoaa kattavan sekä tähän kehittämistyöhön soveltuvan palvelun, jota virtuaalisessa asiakaspalveluneuvojassa voidaan käyttää hyväksi. Kehittämistyössä käytettiin hyväksi lukuisia AWS:n palveluita, joiden avulla voitiin ottaa yhteyttä Googlen *API*:in, lähettäen sinne suomenkielisen puheen. Google palauttaa puheen suomenkielisenä tekstinä, jonka käytämme jälleen AWS *Translaten* kanssa (kts. kuvio 13.)



Kuvio 13. Pelkistetty virtuaalisen asiakaspalveluneuvojan AWS-infrastruktuuri, josta voidaan huomata, kuinka tekstin ja äänen lokalisointi sekä kääntäminen onnistuu

5.3 Sumerian-hostin vaihto toiseen järjestelmään

Tämänhetkisessä tuotteessa käyttäjän tuottama teksti käännettään suomesta englanniksi ja puhe joudutaan viemään ulkopuoliseen, kolmannen osapuolen palveluun,

jotta se saadaan muunneltua tekstiksi, ja tämän jälkeen prosessoitua sekä parsittua sitä. Järjestelmä toki toimii tällaisenaan, mutta on hiukan kömpelö sekä altis virheille. Opinnäytetyön toimeksiantajan yksi tutkimukseen liittyvä kysymys olikin selvittää, voitaisiinko *Sumerian*-hostin keskustelunprosessin logiikassa käyttää jotain toista palvelua, joka samalla myös tukisi suomen kieltä.

Lähtiessä selvittämään vastausta kysymykseen huomattiin jo kehitysvaiheessa asioita, jotka viittaisivat siihen, että kyseinen prosessi olisi mahdollista toteuttaa. *Sumerianin* on alun perin tarkoitettu toimivan eritoten *Lexin* kanssa, johon viittaavat eritoten *Sumerianin* valmiit prosessit sen ja *Lexin* välillä, joissa kehittäjän ei itse tarvitse ohjelmoida tätä interaktiota. Tämän yhteyden lisäksi *Lexiltä* palautunut teksti käytetään suoraan Amazon *Polly*-palvelun läpi, joka muuntaa tekstin puheeksi. *Sumerian* kuitenkin mahdollistaa myös täysin vapaat kädet omien skriptien kirjoittamiseen *JavaScriptin* avulla *Sumerianin* editorissa. Tutkimalla *Sumerianin* lähdekoodia selaimen *Web Developer Tools* -työkalun avulla, voidaan huomata, että keskustelu *Sumerianin* ja *Lexin* välillä koostuu yksinkertaisista *HTTP*-kyselyistä. Kyseisten toimintojen yliajo osoittautui mahdolliseksi, koska tapauksissa, jossa käyttäjä puhuu, pystyimme lähettämään *HTTP*-kyselyn tulkinnan välissä, jolla pystyimme muuntamaan suomenkielisen puheen englanninkieliseksi tekstiksi. Lisäksi prosessia, jossa palautunut teksti muutetaan tekstistä puheeksi käyttäen *Pollya*, voidaan muokata. *Polly* käyttää puheentunnistuksessa yleistä *SSML* (Speech Synthesis Markup Language) -kieltä, jota myös muut yleiset alustat käyttävät. Tästä voimmekin siis päätellä, että *Sumerian* hostin chatbot–logiikan ylläpito on mahdollista myös muissa alustoissa.

6 Pohdinta

Tässä kappaleessa pohdimme tutkimuksen onnistumista ja luotettavuutta. Käymme myös läpi mahdolliset esille tulleet jatkokehitysaiheet.

Tämän opinnäytetyön tarkoituksena oli tuottaa virtuaalinen asiakaspalveluneuvoja Kelalle hyödyntäen AWS:n palveluita. Tutkimuksessa selvitettiin myös AWS:n chatbotin luontiin tarkoitettujen palveluiden soveltuvuutta suomenkielisessä käytössä. Tut-

kimuksen tuloksena saatiin toimiva, AWS:ssä sijaitseva, chatbot, joka osaa vastata yksinkertaisiin Kela-aiheisiin kysymyksiin. Suomen kieltä pystyttiin käyttämään hyväksi chatbotissa – vaikkakin tämä jouduttiin toteuttamaan kääntämisellä.

Tutkimuksen kehitysvaiheen aikana voimme pitää tutkimusta luotettavana, koska suurin osa sen tietoperustasta sekä lähteistä koostuvat suoraan AWS:n omasta dokumentaatiosta. AWS:ssä palvelut tosin saavat toistuvasti sekä usein päivityksiä, ja voi olla mahdollista, että tulevaisuudessa tutkimukselle tärkeät palvelut, kuten *Lex*, *Polly* sekä *Transcribe* lisäävät suomenkielisen tuen; tämä on kuitenkin ainakin vielä epätodennäköistä, koska suomen kielen osuus muista kielistä on pieni.

Luodun AWS:n infrastruktuurin kokonaisuudet voivat saada pakollisia päivityksiä, jolloin yhteys näiden palveluiden välillä voi rikkoutua ajan kanssa, eivätkä tulokset enää vastaa nykytilannetta. Tässä tapauksessa tutkimuksen johtopäätökset eivät saata enää pitää paikkaansa. Hyvänä esimerkkinä *Sumerian*-palvelu saa usein suuria päivityksiä ja se tukee tällä hetkellä vanhentunutta ja uudistunutta skriptausformaattia. Tutkimuksessa luodun virtuaalisen asiakaspalveluneuvojan skriptaus on luotu vanhemmalla formaatilla, joka tulee poistumaan tulevaisuudessa. Tästä syystä *Sumerian*-hostin lähdekoodi pitäisi päivittää, jotta se toimisi myös vanhemman skriptausformaatin poistuttua.

Kuten mainitsimme tietoperustassa, AWS:n palveluiden hyödyntämisestä suomenkielisissä chatboteissa ei ole juuri aikaisempaa tutkimusta. Tutkimuksen tärkein anti jo olemassa olevalle tietoperustalle on, että nykytilanteessa suomenkielisen chatbotin luonti käyttäen lähes vain AWS:n palveluita ei ole suositeltavaa – vaikka se onkin mahdollista. Tutkimuksen ansiosta tiedetään myös, että tarvittaessa AWS:n *Sumerianin* käyttöliittymää voidaan käyttää yhdessä kolmannen osapuolen chatbot-palveluiden kanssa.

Kyseiselle projektille suositellaan jatkokehitystä, jos se halutaan vapauttaa julkiseen käyttöön. Käymme tässä läpi lyhyesti suurimmat jatkokehityksen tarpeet, joita huomasimme tutkimuksen tekovaiheessa:

- *Sumerianin* käyttöliittymänäkymä toimii parhaiten 1980x1080-resoluution sekä 16:9-kuvasuhteen näytöillä, joten asiakaspalveluneuvojan käyttö esimerkiksi älypuhelimella on haastavaa, koska puhe- ja tekstinapit eivät siellä näy kunnolla. Käyttöliittymä kaipaisi siis responsiivisuutta.

- Selvitimme tutkimuksen aikana, että *Sumerian*-hostin vaihto toiseen järjestelmään on mahdollista. Kyseistä prosessia ei kuitenkaan konkreettisesti alettu toteutamaan tässä tutkimuksessa.

- Vaikka asiakaspalveluneuvoja toimii suomen kielellä, *Sumerianin* hosti puhuu vielä englantia. *Sumerianin* puhe tuotettiin käyttäen hyväksi AWS:n palvelua *Polly*, joka ei osannut luonnollisesti puhua suomen kieltä. Jatkokehityksenä voitaisiin testata eri kolmannen osapuolen palveluita tähän tarkoitukseen.

- Asiakaspalveluneuvojan tärkein aikomus oli ajanvaraus sekä varattujen aikojen katselu. Kyseisiä aikoja ei kuitenkaan voida poistaa tai muokata. Tämä olisi tärkein jatkokehityksen aihe aikomuksille.

Lähteet

About AWS. N.d. Viitattu 06.06.2020. https://aws.amazon.com/about-aws/.

Amazon Comprehend: Developer Guide. 2020. PDF. Viitattu 12.07.2020. https://docs.aws.amazon.com/comprehend/latest/dg/comprehend-dg.pdf.

Amazon DynamoDB: Developer Guide. 2020. PDF. Viitattu 27.10.2020. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/dynamodb-dg.pdf.

Amazon Elastic Compute Cloud: User Guide for Linux Instances. 2020. PDF. Viitattu 28.10.2020. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdf.

Amazon Lex: Developer Guide. 2020. PDF. Viitattu 26.10.2020. https://docs.aws.amazon.com/lex/latest/dg/lex-dg.pdf.

Amazon Sumerian: User Guide. 2020. PDF. Viitattu 07.06.2020. https://docs.aws.amazon.com/sumerian/latest/userguide/sumerian-guide.pdf.

Amazon Transalate Features. N.d. Viitattu 27.10.2020. https://aws.amazon.com/translate/details/.

Amazon Translate: Developer Guide. 2020. PDF. Viitattu 28.06.2020. https://docs.aws.amazon.com/translate/latest/dg/translate-dg.pdf.

AWS CloudFormation: User Guide. 2020. Viitattu 08.11.2020. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-ug.pdf.

AWS Lambda: Developer Guide. 2020. PDF. Viitattu 26.10.2020. https://docs.aws.amazon.com/lambda/latest/dg/lambda-dg.pdf.

Global Infrastructure. N.d. Viitattu 09.11.2020. https://aws.amazon.com/about-aws/global-infrastructure/.

Google Cloud overview. 2020. Viitattu 08.11.2020. https://cloud.google.com/docs/overview.

Kananen, J. 2015. Kehittämistutkimuksen kirjoittamisen käytännön opas. Jyväskylä: Suomen Yliopistopaino Oy. Viitattu 08.12.2019.

Kela lyhyesti. 2018. Elämässä mukana – muutoksissa tukena. Viitattu 08.12.2019. https://www.kela.fi/kela-lyhyesti.

Kela tarjoaa uudenlaisen palvelun opiskelijoille. 2017. Tietoa Kelasta – Ajankohtaista. Viitattu 29.01.2020. https://www.kela.fi/-/kela-tarjoaa-uudenlaisen-palvelun-opiskelijoille.

Käsitteet: Henkilötunnus. N.d. Tilastokeskus. Viitattu 13.07.2020. https://www.stat.fi/meta/kas/hetu.html.

Language services for AI. N.d. Viitattu 26.06.2020. https://aws.amazon.com/machine-learning/language/.

Rama, G. 2020. Report: AWS Lambad Popular Among Enterprises, Container Users. Viitattu 27.10.2020. https://awsinsider.net/articles/2020/02/04/aws-lambda-usage-profile.aspx.

Regions and Availability Zones. N.d. Viitattu 9.11.2020. https://aws.amazon.com/about-aws/global-infrastructure/regions_az/?p=ngi&loc=2.

Rouse, M., Brush, K. & Scardina J. 2019. Guide to AI in customer service using chatbots and NLP. Viitattu 29.01.2020. https://searchcustomerexperience.techtarget.com/definition/chatbot.

Speech-to-Text basics. N.d. Google Cloud. Viitattu 09.11.2020. https://cloud.google.com/speech-to-text/docs/basics.

Speech-to-Text. N.d. Google Cloud. Viitattu 12.07.2020. https://cloud.google.com/speech-to-text.

Tarvainen, J. 2016. Mikä on TypeScript? Viitattu 28.06.2020. https://symfony.fi/artikkeli/mika-on-typescript.

What Is Amazon Transcribe? N.d. Viitattu 10.07.2020. https://docs.aws.amazon.com/transcribe/latest/dg/what-is-transcribe.html.

What Is Amazon Translate? N.d. Viitattu 28.10.2020. https://docs.aws.amazon.com/translate/latest/dg/what-is.html.