Jussi Soiluva

Keskustelurobotin kehittäminen pilvipalveluilla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

3.4.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jussi Soiluva Keskustelurobotin kehittäminen pilvipalveluilla 28 sivua + 1 liite 2.3.2018
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Lehtori Olli Alm

Insinöörityön tarkoituksena oli tutkia keskustelurobotin toteutusta web-sovelluksessa käyttäen IBM Cloud -pilvipalveluista saatavia Cloud Foundry-, Watson Conversation- ja Watson Tone Analyzer -tekniikoita. Tavoitteena oli kehittää palvelu, joka tunnistaa käyttäjän tunnetilan keskustelussa ja antaa sopivan vastauksen käyttäjälle tunnetilan mukaan automaattisesti.

Työ aloitettiin tutustumalla keskustelurobotin perustekniikoihin, rakennetun mallin keskustelurobottiin ja tekoälykeskustelurobottiin, minkä jälkeen syvennyttiin enemmän keskustelurobotin kehittämiseen tarvittaviin kehitysympäristöpalveluihin. Työssä selvitettiin, että rakennetun mallin keskustelurobotti on tämänkaltaisessa projektissa parempi vaihtoehto-, ja siihen sopiva kehitysympäristö löytyi IBM Cloud -pilvipalvelusta. Tutustuttiin myös kuluttajamarkkinoilla oleviin Amazon Echo- ja Apple Siri -tuotteisiin, jotka käyttävät keskustelurobottitekniikkaa.

Insinöörityössä kehitetyn palvelun tavoite oli ensisijaisesti näyttää keskustelurobotin automaatiomahdollisuudet asiakaspalvelutilanteessa. Toteutettuun web-sovellukseen käytetyt palvelut jakautuvat useisiin REST API -rajapintoihin, joista tärkeimmät palvelut toimivat IBM Cloud -pilvipalveluympäristössä.

Watson Conversation -palveluun kehitetyllä keskustelurobotilla pystytään tarjoamaan yksityiskohtainen keskustelurobotti, joka pystyy vastaamaan käyttäjän pyyntöihin nopeasti ja varmasti. Myös keskustelurobotin toteutus voidaan ulkoistaa kokonaan palveluun, jotta sitä voidaan tarjota monessa eri sovelluksessa. Keskustelurobotin kehittäminen Watson Conversationilla vaatii kuitenkin paljon suunnittelua ja yksityiskohtaista mallintamista, jotta keskustelurobotti vastaa käyttäjälle oikein. Yksityiskohtaista mallinnusta voidaan myös pitää hyödyllisenä ominaisuutena, koska nopeasti lisättävät yksityiskohdat lisäävät keskustelurobotin laatua.

Avainsanat	keskustelurobotti, web-sovellus, pilvipalvelut, Node.js, IBM Cloud

Author Title Number of Pages Date	Jussi Soiluva Developing chatbot with cloud services 28 pages + 1 appendices 2 March 2018
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Olli Alm, Senior Lecturer

The goal of this project was to research and study chatbot development and develop an web application using Cloud Foundry, Watson Conversation and Watson Tone Analyzer available on the IBM Cloud platform services. The goal was to learn how to deploy and combine services to use the skills learned in future projects.

The study was started by exploring basic chatbot technologies the build model chatbot and the artificial intelligence chatbot, after which more information about the development environment services needed to develop the chatbot is given. We also get acquainted with products on the consumer market that use the chatbot technology in their products.

The aim of the developed service in this study is primarily to showcase the chatbot in the customer service situation. The services used in the implemented web application are divided into several REST API interfaces. The most important REST API's operate in the cloud environment platform of IBM Cloud.

The Watson Conversation developed chatbot can provide a detailed chatbot platform that can respond quickly and surely to user requests. Also, the implementation of chatbot can be completely outsourced to the service so that it can be provided in many different applications. Developing a chatbot with Watson Conversation requires a lot of design and detailed modeling so that the chatbot answers for the user correctly. Detailed modeling can also be considered useful as a feature, because the easy adding of details to increase the quality of the chatbot.

Keywords	Chatbot, Web development, Cloud platform service , Node.js, IBM Cloud

Sisällys

Lyhenteet

Http	HyperText Transfer Protocol eli hypertekstiprotokolla, selaimille ja WWW-palvelimille tarkoitettu tiedonsiirtoprotokolla.
Npm	Paketinhallintajärjestelmä Node.js Javascript-projekteissa.
Javascript	Komentosarjakieli, jota käytetään dynaamisten verkkosivujen toteutuksessa.
Json	JavaScript Object Notation, datan kuvauskieli, jota käytetään tiedonvälitykseen palvelimen ja asiakkaan välillä.
Nlp	Natural Language Processing, tietojenkäsittelytieteen ja tekoäly ala, joka koskee tietokoneiden ja ihmisten vuorovaikutusta.
Cf	Cloud Foundry, avoimen lähdekoodin pilven palvelun sovellusalusta.
Rest	Representational State Transfer, ohjelmointirajapintojen HTTP-protokollaan pohjautuva arkkitehtuurimalli.
Sdk	Software development kit, ohjelmistokehityspaketti on kokoelma työkaluja ja luokkakirjastoja, joilla kehitetään sovelluksia tietylle alustalle.
Html	Hypertest Markup Language, hypertekstin merkintäkieli, joka on avoimesti standardoitu kuvauskieli.

1 Johdanto

Viime vuosien aikana palveluiden automatisoituminen on ottanut suuria teknologisia kehitysaskeleita, ja käyttäjiä pyritään palvelemaan nopeammin ja kellon ympäri. Tämä automatisoituminen vaikuttaa yhä enemmän jokaisen ihmisen arkielämässä. Yksi automaatioteknologian merkittävimpiä palveluita ovat keskustelurobotit, jotka pyrkivät palvelemaan käyttäjiä nopeasti ja helpottamaan palvelujen käyttöä.

Insinöörityön tavoitteena on kehittää keskustelurobotti-web-sovellus, jossa käyttäjä keskustelee lennon varauksesta kirjoittamalla keskustelurobotille varaustiedot. Keskustelurobotti myös reagoi käyttäjän tunnetilaan, joka voi muuttaa keskustelurobotin vastausta. Insinöörityössä on myös tavoitteena selvittää, miten keskustelurobotti kehitetään ja miten keskustelurobotissa käytetyt tekniikat vaikuttavat web-sovelluksen lopputulokseen.

Insinöörityössä perehdytään dialogikeskustelurobotti- ja tekoälykeskustelurobottitekniikoihin ja lyhyesti keskustelurobottikehityksen historiaan. Näiden tekniikoiden hyötyjä ja heikkouksia vertaillaan insinöörityön keskustelurobotin kehittämisessä. Myös selvitetään nykyisten keskustelurobottitekniikoiden käyttö kuluttajamarkkinoilla ja niiden käyttöönoton haasteet.

Työssä selvitetään keskustelurobottisovelluksissa ja -kehitysympäristöissä, IBM Cloudin Watson Conversationissa ja Amazon Lexissa, käytettyjä teknologioita ja palveluja. Tutustutaan myös keskustelurobotti Amazon Echo- ja Apple Siri -tuotteisiin, jotka ovat saatavilla kuluttajamarkkinoilla.

Projektissa käytettävien IBM Cloudin Watson Conversation-, Watson Tone Analyzer- ja Node.js-tekniikoiden tarkoitusta tutkitaan tarkemmin ja kuvataan tekniikoiden käyttöä. Selvitetään myös tekniikoiden yhdistäminen keskustelurobotin web-sovelluksessa ja niiden välinen kommunikointi sovelluksessa.

Insinöörityössä esitetään keskustelurobotin web-sovelluksen suunnitelma ja suunnitelmaa toteutetaan kehityksessä. Keskustelurobottisovelluksen käyttö ohjeistetaan ja kehityksen haasteita tutkitaan, jotta voidaan parantaa kehittämisen tapoja. Myös jatkokehityksessä tutkitaan sovelluksen parantamista ja arvioidaan kaikkien web-sovelluksessa käytettyjen tekniikoiden käyttöä.

2 Keskustelurobotti ja teknologiat

2.1 Keskustelurobotti

Keskustelurobotti perustuu ihmisen ja koneen väliseen kanssakäymiseen, joka tapahtuu joko koneen rakennetun dialogin tai tekoälyn kautta. Keskustelua käydään koneen kanssa joko tekstin tai äänen välityksellä. [1.] Rakennetun dialogin ja tekoälykeskustelurobotin ero on, että rakennetussa dialogissa on valmiit dialogit, joita keskustelurobotti seuraa käyttäjän pyyntöjen mukaan, kun taas tekoälykeskustelurobotissa käytetään algoritmeja, jotka käsittelevät käyttäjän pyynnön ja rakentavat parhaan vastauksensa algoritmien mukaan. Tekoälyn algoritmeihin kuuluu mm. Natural Language Processing (NLP) -tieteenala, joka käsittelee luonnollisen kielen tunnistamista tietokoneissa. [2.]

1960-luvulla ensimmäisten julkisesti kehitettyjen keskustelurobottien tavoitteena oli läpäistä Turing-testi. Koehenkilö keskusteli tietokoneen kautta joko ihmisen tai rakennetun keskustelurobotin kanssa. Jos koehenkilö ei pystynyt erottamaan, milloin hän keskusteli keskustelurobotin kanssa, oli keskustelurobotti läpäissyt Turing-testin. Ensimmäinen merkittävä keskustelurobotti oli vuonna 1966 julkaistu Eliza, jonka kehittäjä Joseph Weisenbaum oli kiinnostunut läpäisemään Turingin-testin. Eliza-keskustelurobotti oli terapeutti, jonka tarkoituksena oli kysyä avoimia kysymyksiä ja pystyä antamaan järkeviä jatkokysymyksiä ja vastauksia käyttäjälle. Suoriutumalla Turingin-testin läpi pystyttiin todistamaan, että tietokone pystyisi esittämään ihmisen keskustelukäyttäytymistä itsenäisesti. [3.]

Keskustelurobotin yleisin käyttötarkoitus on asiakaspalveluiden automaatio ja tässä suurin etu on, että keskustelurobotit pystyvät palvelemaan asiakkaita koko vuorokauden ajan ja vastaamaan asiakkaille välittömästi. Keskustelurobotti pystyy korvaamaan asiakaspalvelun työntekijän helpoissa asiakaspyynnöissä ja valituksissa, mutta ei vielä yksityiskohtaisissa palveluissa. [4.]

Vaikka vieläkin yritetään tehdä paljon lisää tekoälykeskustelurobotteja, jotka pyrkivät läpäisemään Turing-testin, ovat markkinoille yhä enemmän tulossa rakennettujen dialogien keskustelurobotit. Dialogikeskusteluroboteilla pystytään helposti palvelemaan tietyn alan palveluiden yksityiskohtaisia toimintoja. Tekoälykeskustelurobottien kouluttaminen tai rakentaminen vaatii paljon aikaa, koska tekoälyn kouluttamiseen tarvitaan valtavia

määriä dataa ja jos koulutusdatassa on virheitä, se voi vaikuttaa keskustelurobotin palveluun merkittävästi. Vielä ei ole tehty massatuotantoon sopivaa tekoälykeskustelurobottia, jolla pystytään toteuttamaan kaikkia palveluja. Tämän vuoksi rakennetun dialogin käyttö on helpompi ottaa käyttöön yksityisissä yrityksissä. [5.]

2.2 Keskustelurobottisovellusten kehitysympäristöt

Keskustelurobottisovellusten kehitysympäristöjä on paljon, ja kaikkein suurin ero on niissä käytettyjen tekniikoiden ja käyttöliittymän helppokäyttöisyys. Tässä perehdytään kahden suuren yrityksen tuottamiin keskustelurobottisovellusten kehitysympäristöpalveluihin. IBM tarjoaa omassa pilvipalvelussaan IBM Cloudissa Watson Conversation -kehitysympäristön. Amazon tarjoaa pilvipalveluissaan Amazon Lex -kehitysympäristöpalvelun. Näillä keskustelurobotin kehitysympäristöillä pystytään automatisoimaan asiakaspalveluita, joilla taas voi tehostaa yksityisten yrityksien toimintaa. Keskustelurobotilla pystytään palvelemaan asiakkaita ympäri vuorokauden ja korvaamaan yrityksissä toimivaa henkilökuntaa.

IBM Cloud Watson Conversation

IBM Cloud tarjoaa Watson Conversation -keskustelurobottipilvipalvelun, jossa voi rakentaa oman keskustelurobotin Watson Conversation -käyttöliittymässä. Watson Conversation -keskustelurobotti tukee 12 eri kielen kehittämistä: englanniksi, arabiaksi, kiinaksi, tsekiksi, hollanniksi, ranskaksi, saksaksi, italiaksi, japaniksi, koreaksi, portugaliksi ja espanjaksi. [6.] Watson Conversationissa voidaan siis kehittää halutulla kielellä Watson Conversation käyttöliittymässä ja Watson Conversation tunnistaa kielen sanaston ja erilaiset kielen lauserakenteet. Melkein kaikissa kielissä on myös epämääräisen sanan tunnistamisen operaatio (Entity fuzzy matching), joka pyrkii tunnistamaan väärinkirjotukset. Kielen voi valita Watson Conversationin käyttöliittymästä, kun aloittaa keskustelurobottiprojektin.

Watson Conversation keskustelurobotti koostuu kolmesta eri osiosta:

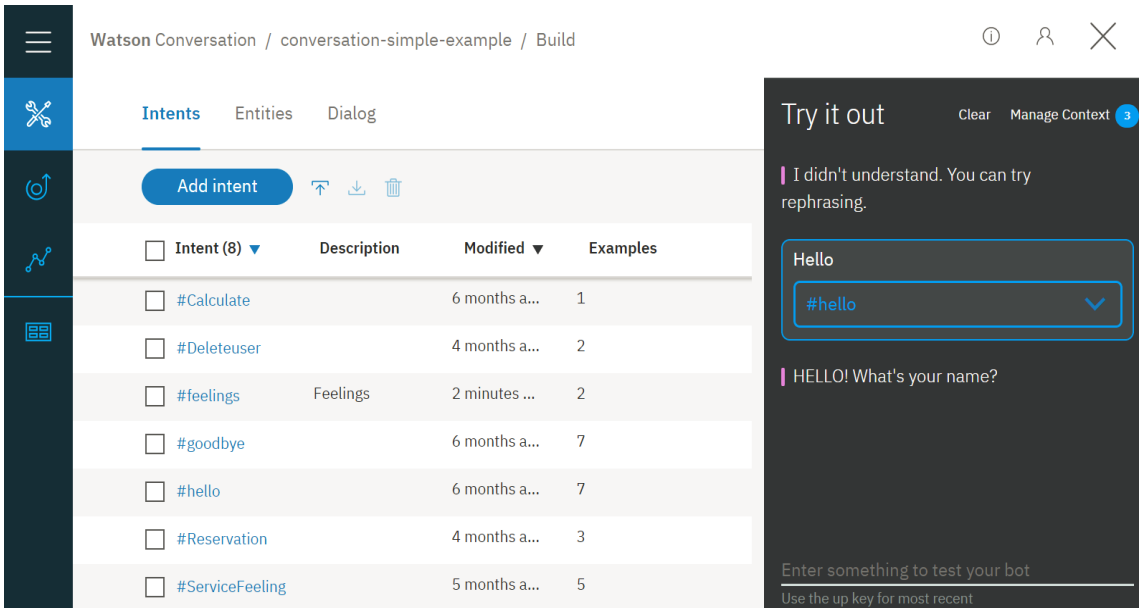
- aikomus (Intents)
- joukko (Entities)

- dialogi (Dialog).

Kehitys aloitetaan käynnistämällä Watson Conversation -käyttöliittymä IBM Cloudista. Käyttöliittymän käynnistyttyä aikomukseen lisätään keskustelurobotin käyttäjän tulevia pyyntöjä ja tiedusteluja, joita voidaan luokitella esimerkiksi kysymykset- ja tervehdykset-kategorioihin. Esimerkiksi aikomuksessa voisi olla kategoria ”Tervehdykset”, ja tähän kategoriaan kuuluisivat ”Hei!”-, ”Moi!”- ja ”Hyvää päivää!” -lausahdukset. Aikomukset ovat merkitykseltään hyvin samankaltaisia, ja tämän vuoksi ne kuuluvat samaan kategoriaan. Joukkoihin lisätään niin sanottuja avainsanoja, joita käyttäjän pyynnöstä tai tiedustelusta voisi olla esimerkiksi ”Missä on Helsinki?” Helsinki voisi kuulua joukkoon, jota etsitään pyynnöstä, ja samaan joukkoon voisi kuulua ”Tukholma” ja ”Oslo”. [6.]

Dialogiin määritellään keskustelun polku, joka usein alkaa aikomuksesta eli siitä mitä käyttäjä pyytää keskustelurobotilta. Tästä pystytään vaikuttamaan dialogin kulkuun etsimällä joukko sanoja tai Conversationiin luoduilla muuttujilla. Conversation-muuttujat on luotu Conversation-käyttöliittymään. Kehitettyä dialogia pystyy testaamaan testidialogi-paneelista. [6.]

Watson Conversationin kehittäminen perustuu enimmäkseen dialogin rakentamiseen. Esimerkkinä seuraava: Jos joukkoihin ei ole lisätty tiettyä avainsanaa, jonka käyttäjä on antanut vastaukseksi dialogissa, Conversation ei pysty oppimaan ja lisäämään avainsanaa itsestään joukkoihin välittömästi. Sanan pystyy kuitenkin opettamaan testidialogissa suoraan valitsemalla sille oikean joukon kategorian. [6.] Kuvassa 1 näkyy Watson Conversationin aikomus-näkymä ja testidialogi, missä voi opettaa tulevia käyttäjän pyyntöjä oikeaan kategoriaan.



Kuva 1. Watson Conversation -kehitysympäristö. Testidialogi oikealla puolella.

Watson Conversationiin rakennettuun dialogipalveluun otetaan yhteyttä käyttämällä palvelun REST API -rajapintaa. Kun tehdään pyyntö REST API -rajapintaan ja kun käyttäjä on lisännyt Conversation-työtilan tunnukset REST-rajapintaan, tulee pyynnön vastaus aina JSON-muodossa, jonka objektista voidaan purkaa Conversation-dialogin vastaus. Samalla vastauksessa tulee muita metatietoja, joita voi hyödyntää kehittäjän sovelluksessa. [7.] Kuvassa 2 näkyy esimerkki viestin lähetyksestä Conversation-palveluun.

```
var watson = require('watson-developer-cloud');

var conversation = watson.conversation({
  username: '{username}',
  password: '{password}',
  version: 'v1',
  version_date: '2018-02-16'
});

conversation.message({
  workspace_id: '9978a49e-ea89-4493-b33d-82298d3db20d',
  input: {'text': 'Hello'}
},  function(err, response) {
  if (err)
    console.log('error:', err);
  else
    console.log(JSON.stringify(response, null, 2));
});
```

Kuva 2. Node.js ”Hello” -viestin lähetys Watson Conversation -ympäristöön.

Tämä mahdollistaa sovelluksen kehittämisen eri ohjelmointikielellä vain tekemällä kutsuja Watson Conversation REST API -rajapintaan ja purkamalla JSON-tiedot oman sovelluksen käyttöön. [7.] Tämä mahdollistaa keskustelurobottipalvelun ulkoistamisen kokonaan Watson Conversation -puolelle, ja siihen voidaan tehdä pyyntöjä monesta eri sovelluksesta. Kuvassa 3 näkyy Conversationin lähettämä JSON-vastaus, josta puretaan vastausteksti sovelluksen käyttöön.

```
{
  "intents": [
    {
      "intent": "hello",
      "confidence": 0.8742280006408691
    }
  ],
  "entities": [],
  "input": {
    "text": "Hello"
  },
  "output": {
    "text": [
      "Hi! What can I do for you?"
    ],
    "nodes_visited": [
      "node_2_1501875253968"
    ],
    "log_messages": []
  },
  "context": {
    "conversation_id": "59aebcf9-3df2-4f85-8632-cf8e4c760260",
    "system": {
      "dialog_stack": [
        {
          "dialog_node": "root"
        }
      ],
      "dialog_turn_counter": 1,
      "dialog_request_counter": 1,
      "_node_output_map": {
        "node_2_1501875253968": [
          0
        ]
      },
      "branch_exited": true,
      "branch_exited_reason": "completed"
    }
  }
```


Kuva 3. JSON-vastaus Watson Conversationista.

Myös Conversation-palvelua pystyy kehittämään omasta kehitysympäristöstä käyttämällä REST API -funktioita. Esimerkiksi uuden aikomuksen luominen voidaan käynnistää kutsumalla ”createIntent”-functiota ja lisäämällä siihen kaikki aikomuksen parametrit. [8.] Conversation voidaan myös yhdistää suoraan Slack-keskustelupalveluun [9]. Kuvassa 4 näkyy uuden aikomuksen luominen koodilla suoraan Conversation-palveluun.

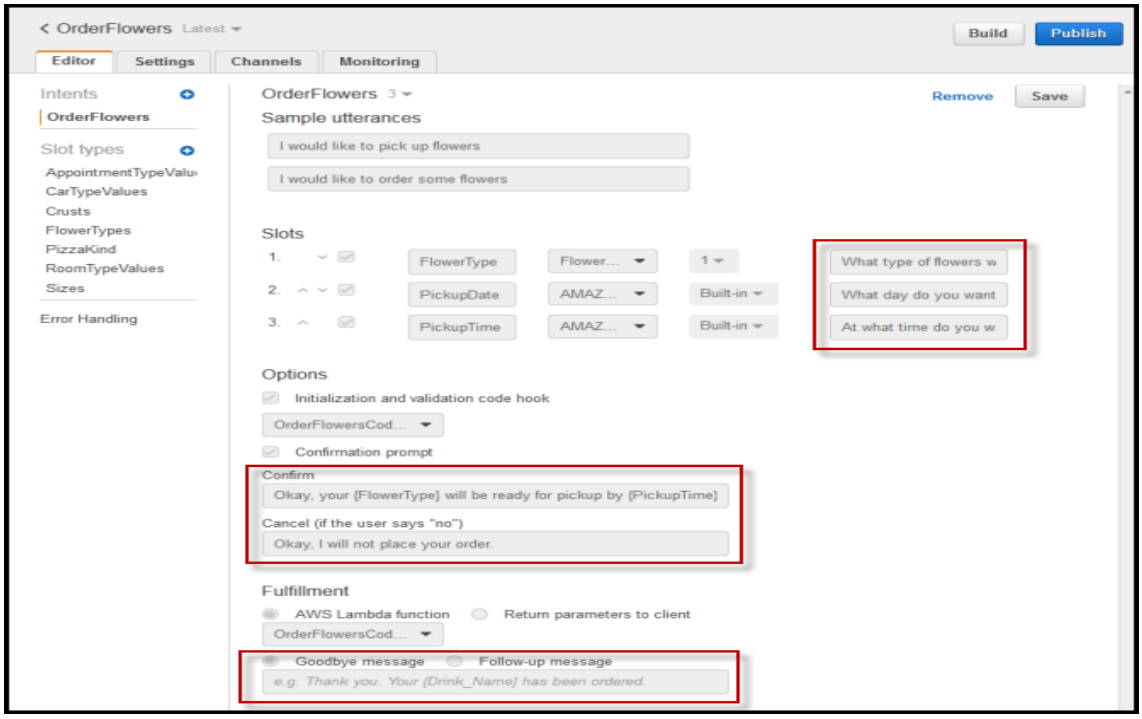
```
var watson = require('watson-developer-cloud');

var conversation = new watson.ConversationV1({
  username: '{username}',
  password: '{password}',
  version_date: '2018-02-16'
});

var params = {
  workspace_id: '9978a49e-ea89-4493-b33d-82298d3db20d',
  intent: 'hello',
  examples: [
    {
      text: 'Good morning'
    },
    {
      text: 'Hi there'
    }
  ]
};

conversation.createIntent(params, function(err, response) {
  if (err) {
    console.error(err);
  } else {
    console.log(JSON.stringify(response, null, 2));
  }

});
```

Kuva 4. Esimerkki tervehdys-aikomuksen luomisesta ”createIntent” -funktiolla.

Amazon Lex

Amazon Lex -pilvipalvelu tarjoaa keskustelurobottikäyttöliittymän, jossa voi tekstillä ja äänellä olla vuorovaikutuksessa palveluun. Palvelun kehitystä voi tehdä vain englanniksi. [10.] Palvelussa käytetään koneoppimista ja valmista sanantunnistusjärjestelmää, joka oppii käyttäjän vuorovaikutuksesta lisää sanoja automaattisesti sen omaan käyttöön. Luonnollisen kielen tunnistamiseen Amazon Lex käyttää siihen kehitettyä NLU (Natural Language Understanding) -palvelua, joka ymmärtää pyynnön kontekstin. [11.]

Amazon Lex koostuu kahdesta osasta: aikomus (Intents) ja vastaustilat (Slots). Aikomuksen tarkoitus on sama kuin Watson Conversationin, mutta tiedustelua tai pyyntöä nimitetään ilmaisuksi (Uttarance). Vastaustilat ovat tiedustelujen tai pyyntöjen vastauksia, jotka voivat olla muun muassa sarjassa olevia kysymyksiä tai yksittäisiä vastauksia käyttäjän pyyntöihin. [11.] Vastaustilat siis sisältää Watson Conversationin erilleen luokitellun joukko-osion. Kuvassa 5 näkyy Amazon Lex -kehitysympäristö aikomus-näkymä.



Kuva 5. Amazon Lex -kehitysympäristö.

Amazon Lex toimii myös samalla REST API -rajapintapohjalla, missä pyynnöt tai tiedustelut toteutetaan keskustelurobottiin lähettämällä HTTP POST -metodilla JSON-dataa ja saamalla JSON-vastauksen pyyntöön. Tämä mahdollistaa myös sovelluksen kehittämisen eri ohjelmointikielillä, mutta myös keskustelurobotti pystytään kehittämään omassa kehitysympäristössä. [12.] Amazon Lexissä on mahdollista kehittää omasta kehitysympäristöstä suoraan keskustelurobottia. Esimerkiksi HTTP POST -metodilla voi luoda uuden aikomuksen keskustelurobottiin lisäämällä URI-parametreihin ”intents/name/versions” ”/name/” -parametrin tilalle uuden aikomuksen nimen. [13.]

Amazon Lexin voi myös integroida suoraan eri viestintä- tai keskustelupalveluihin, kuten Facebook Messenger, Slack, Twilio ja Kik. Amazon Lexissä on käyttöönottoasetukset, joista voi valita integroitavan sovelluksen, esimerkiksi Facebook Messenger. [11.]

2.3 Keskusteluorobottituotteet

Amazon Echo

Amazon Echo on älykäs henkilökohtainen avustajakaiutin (Kuva 6), jolla käyttäjä pystyy tekemään pyyntöjä, ja Echo antaa vastauksia pyyntöihin tai suorittaa pyynnön. Amazon Echo -avustajan nimi on Alexa, joka toimii kuuntelemalla käyttäjän kysymystä tai pyyntöä mikrofonin kautta heti, kun käyttäjä sanoo ”Alexa” ja antaa käyttäjälle vastauksen kaiuttimesta. Amazon Echo -kaiutin asennetaan mobiililaitteen Alexa-sovelluksen kautta oman kodin langattomaan internetyhteyteen.

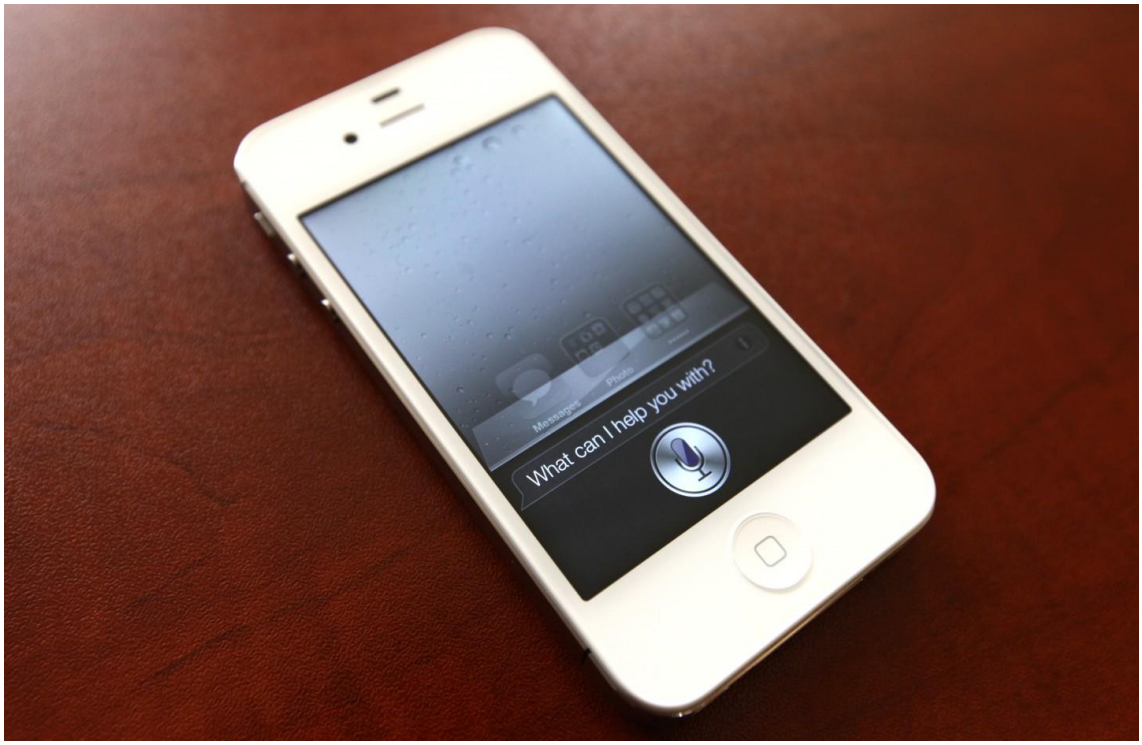
Kuva 6. Amazon Echo -kaiutin.

Amazon Echolla käyttäjä pystyy puheen avulla pyytämään Alexaa käyttämään erilaisia palveluja, esimerkiksi pyytämään kuuntelemaan musiikkia eri musiikkipalveluista, lukemaan käyttäjälle tietoa internetistä ja kertomaan päivän tärkeimmät uutiset. [14;15.] Vuoden 2017 toukokuussa Echo-laitteiden käyttäjiä oli 2,6 miljoonaa yksittäistä käyttäjää. [16.]

Amazon Echo -sovelluksia kehitetään Alexa Skill Kit (ASK) -ympäristössä, jossa voi kehittää valmiita palveluja. Siihen voidaan yhdistää Amazon Lexissä kehitetty keskusteluorobotti lataamalla Amazon Lex -bottiskeeman JSON-tiedostona ja lisäämällä JSON-tiedosto ASK-kehitysympäristöön. [17.]

Apple Siri

Vuonna 2011 Apple julkaisi iPhone 4S -matkapuhelimiinsa äänellä ohjattavan henkilökohtaisen avustajan Sirin (Kuva 7) [18]. Siri pystyy hallitsemaan mobiililaitteessa olevia sovelluksia käyttäjän pyynnöstä ja suorittamaan niissä haluttuja pyyntöjä. Kuten Echossa, Siri vastaa käyttäjän pyyntöihin puheella, mutta voi myös käydä keskustelua tekstin välityksellä. [19.] Vuonna 2017 Sirillä oli 41,1 miljoonaa yksityistä käyttäjää [16].



Kuva 7. IPhonen Siri-käyttöominaisuus..

Sirin ensimmäiset kehitysvaiheet alkoivat vuonna 1993, kun ensimmäisen tulevan Sirin prototyyppi oli kehitetty. Siri julkaistiin ensin Applen sovelluskauppaan App Storeen omana sovelluksena, mutta Apple osti myöhemmin Sirin kehittäneen yrityksen ja yhdisti sovelluksen omaan käyttöjärjestelmäänsä. [18.]

Sirin toiminta pohjautuu tekoälyyn ja NLP-kehitykseen, ja se koostuu kolmesta komponentista: keskustelusta rajapinnassa, yksityiskohtaisesta asiayhteyden tunnistamisesta ja palvelun delegoinnista. Siri pyrkii luonnollisen kielen tunnistamisella ymmärtämään käyttäjän pyynnön ja vastaamaan kehitetyllä tekoälyllä, joka arvioi todennäköisimmän vastauksen pyyntöön. [19.]

Apple Sirin käyttöominaisuuksia voidaan käyttää iOS-sovelluksien kehityksessä käyttämällä SiriKit-palvelua [20].

3 Projektin keskustelurobotin palveluiden ja tekniikoiden käyttöönotto

3.1 IBM Cloudin käyttöönotto ja pilvipalvelu–web-sovelluksen aloittaminen

Keskustelurobotin kehittäminen IBM Cloudissa aloitetaan luomalla tunnukset ja kirjautumalla IBM Cloud -palveluun. Palvelussa on 30 päivän maksuton koeaika, jonka jälkeen palvelu on maksullinen. ”Catalog”-sivulla on ”Cloud Foundry apps”, josta voi valita oman web-sovelluskehittämispohjan, esimerkiksi Node.js, PHP tai Python. Insinöörityön keskustelurobottiin käytettiin ”SDK for Node.js” -pohjaa, jolla tehdään back-end-puolen kutsuja JavaScriptillä. Web-sovelluksen aloittaminen on maksuton koeajan aikana. Kun sovelluksen luonti on käynnistetty, voidaan ladata sovelluksen aloituspaketti, jossa on tarvittavat koodit lokaaliin kehittämiseen.

Sovellukseen lokaalisti tehdyt muutokset saa siirrettyä pilvipalveluun käyttämällä Cloud Foundry CLI -komentorivilisäosaa. Cloud Foundryn komentorivikäskyillä pystytään ottamaan yhteyttä omaan IBM Cloud -käyttäjään ja pilvipalveluun luotuun web-sovellukseen. Tällä yhteydellä siirretään lokaalisti tehdyt koodimuutokset pilvipalvelulle. Kun lokaalisti tehdyt muutokset on siirretty pilvipalveluun, muutokset automaattisesti julkaistaan pilvipalvelun web-sovellukseen. Komentoriviin kirjoitetaan projektin hakemistosta ”cf push”, ja tämä aloittaa tiedostojen siirron Cloud Foundryn pilvipalveluun.

3.2 Watson Conversation -palvelun käyttöönotto

IBM Cloudissa on paljon eri pilvipalveluita, mutta keskustelurobotin kehittämiseen tarvitaan Watson Conversation -palvelua. Watson Conversation -palvelussa on omat hintaluokitukset riippuen sovelluksen käytön laajuudesta. Lite-versiolaajuus on ilmainen, ja kuukaudessa voi suorittaa 10 000 API-pyyntöä palveluun, 5 työtilaa, 100 aikomusta ja 25 joukkoa. Standard-versiossa saa tehdä loputtoman määrän API-pyyntöjä 0,0025 dollaria pyyntöä kohden, 20 työtilaa, 2 000 aikomusta ja 1 000 joukkoa. Mutta tavalliseen kehityskäyttöön riittää Lite-versiosuunnitelma. Conversation-palvelun käyttöönotto otetaan avaamalla Conversation-sovellus internetselaimessa ja luomalla oma työtila

(Workspace). Kun uusi työtila on luotu, päästään kehittämään keskusteluro-botin keskustelun kulun ja rakenteen mallintamista.

Kun aikomus, joukko ja dialogit on lisätty luvussa 2.1 mainittuun tapaan, pitää ottaa pääsytiedot (Service credentials) muistiin ja yhdistää ne omaan sovellukseen.

3.3 Watson Tone Analyzer -palvelu ja sen käyttöönotto

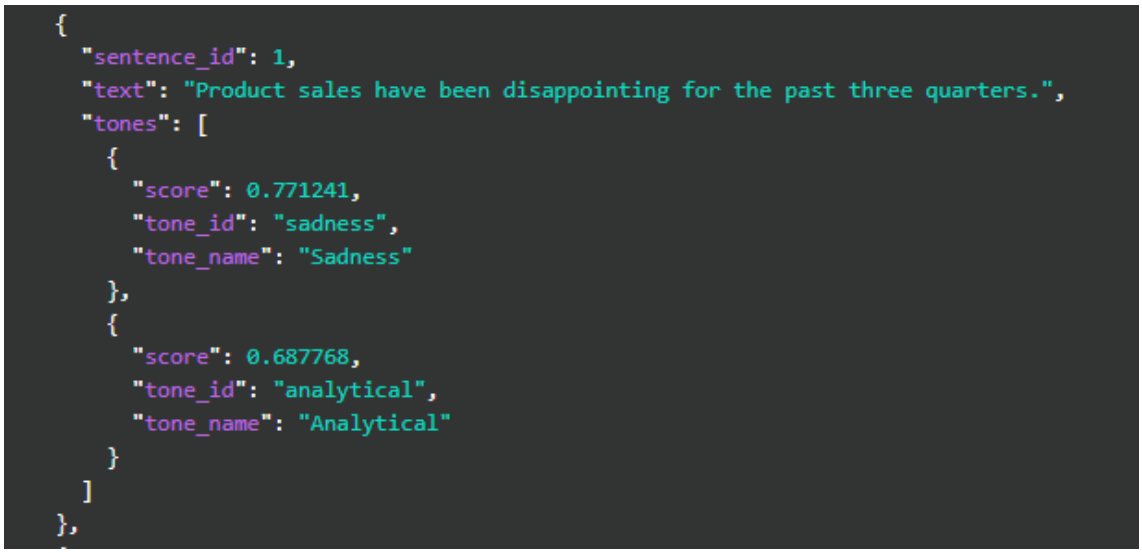
IBM Watson Tone Analyzer -palvelu käyttää kielellistä analyysiä havaitsemaan tunne- ja kielisävyjä kirjallisesta tekstistä. Palvelu voi analysoida äänipätkiä ja dokumentteja. Palvelua voidaan käyttää ymmärtämään käyttäjien dokumenttien kontekstia, miten kirjoitettua viestintää ymmärretään, ja parantamaan siten viestinnän sävyä. Yritykset voivat käyttää palvelua oppiakseen asiakkaidensa viestinnän sävyä ja voidakseen reagoida kullekin asiakkaalle sopivasti tai ymmärtää ja parantaa asiakaskeskusteluja kokonaisuudessaan. [21.]

IBM Watson Tone Analyzer -palvelu perustuu teoreettiseen psykolingvistiikan tutkimusalueeseen, joka tutkii kielellisen käyttäytymisen ja psykologisten teorioiden välistä suhdetta. Palvelu käyttää kielellistä analyysiä sekä kirjallisen tekstin ja emotionaalisten ja kielellisten sävyjen piirteiden välistä korrelaatiota, jotta voidaan kehittää pisteitä näistä sävynmittauksista. [22.]

Sävyillä voidaan keskusteluro-bottipalvelussa antaa erilaisia vastauksia lähettämällä käyttäjän viesti ensin Tone Analyzeriin ja lähettämällä sävyjen tulokset Conversationiin metatietona. Tone Analyzerin tunnesävytiedot ovat

- viha
- pelko
- ilo
- suru
- inho.

Kuvassa 8 näkyvät Tone Analyzerin metatiedot JSON-muodossa.



```
{
  "sentence_id": 1,
  "text": "Product sales have been disappointing for the past three quarters.",
  "tones": [
    {
      "score": 0.771241,
      "tone_id": "sadness",
      "tone_name": "Sadness"
    },
    {
      "score": 0.687768,
      "tone_id": "analytical",
      "tone_name": "Analytical"
    }
  ]
},
```


Kuva 8. Esimerkki Tone Analyzerin JSON-vastauksen antamista analysointituloksista.

Watson Tone Analyzer -palvelu otetaan käyttöön samasta ”catalog”-sivusta kuin Conversation, mutta Tone Analyzer tarvitsee vain käynnistää ja saadaan pääsytiedot, jotka lisätään oman sovelluksen tietoihin.

Watson Tone Analyzeria käytetään insinöörityössä kehitetyssä web-sovelluksessa tunnistamaan käyttäjän kirjoituksesta tunteita, niin että keskustelurobotti voisi reagoida käyttäjän tunnetilan mukaan eri tavoin.

3.4 Node.js-tekniikan käyttöönotto ja käytetyt pakkaukset

Node.js-tekniikka valittiin insinöörityön web-sovelluksen kehityspohjaksi, koska se oli vaihtoehtona luvussa 3.1 mainitussa Cloud Foundry –web-sovelluskategoriassa. Itselläni oli kokemusta tekniikan käytöstä ja tiesin sen helpottavan projektin kehittämistä.

Node.js on web-kehitykseen käytettävä alustariippumaton back-end-puolen sovellusalusta, jolla pystytään tekemään monia samanaikaisia yhteyksiä. Node.js on avointa lähdekoodia, minkä vuoksi sillä on valtava kirjasto erilaisia valmiita pakkauksia, joita käyttäjät voivat käyttää omiin projekteihinsa. [23.] Jotta saa Node.js-palvelut toimintaan projektissa, täytyy tietokoneelle asentaa Node.js, esimerkiksi Google Translate -pakkaus (google-translate-api), joka käyttää Googlen kääntäjän selainsovellusta. Kyseisellä pakkauksella on helppo valita kieli, jonka haluaa kääntää toiseksi kieleksi. Kaikki pakkaukset

asennetaan projektiin kirjoittamalla komentoriviin projektin hakemistossa ”npm install pakkauksen nimi”.

Node.js:llä pystytään yhdistämään kaikki Watson-pilvipalvelut ja lisäämään Node.js:n omasta kirjastosta sen käyttäjien tekemiä REST API -rajapintapakkauksia. Node.js on siis kaiken keskus, jolla tehdään back-end-kutsuja Watson-pilvipalveluihin ja Node.js-pakkauksiin, mutta samalla voidaan kehittää front-endin puolta HTML-tiedostolla.

Projektissa käytettiin eri Node.js-pakkauksia, joilla voitiin suorittaa toimintoja tai helpottaa kehittämistä omassa kehitysympäristössä. Projektiin asennettiin seuraavassa esitettävät pakkaukset:

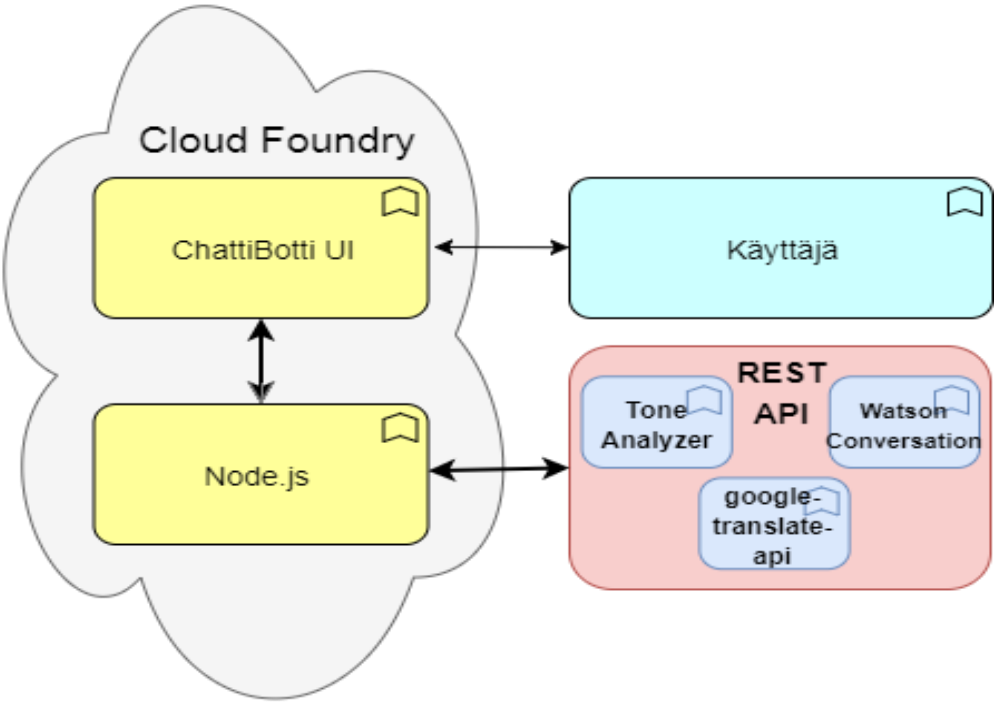
Express-pakkauksella hallitaan ja suoritetaan yksinkertaisemmin HTTP-pyyntöjä API-rajapintoihin [24]. Cfenv-pakkauksella seurataan, käynnistetäänkö Cloud Foundry -sovellusta lokaalisti vai Cloud Foundry -pilvipalvelussa [25]. Body-parser-pakkaus parsii automaattisesti HTTP-pyynnöstä API-rajapintaan body-objektin tiedot, mikä auttaa objektin parsintaprosessia [26]. Google-translate-api-pakkauksessa käytetään Googlen kielenkääntösovellusta [27]. Watson-developer-cloud-pakkauksessa on Watsonin kaikki pilvipalvelut ja sitä voidaan tehdä pyyntöjä Watson Conversation- ja Watson Tone Analyzer -palveluihin [28].

4 Keskustelurobotin suunnittelu ja toteutus

4.1 Keskustelurobotin kehittämisen tavoite

Insinöörityön keskustelurobotin tekemisessä tavoitteena oli kehittää web-sovellus, joka reagoisi automaattisesti käyttäjän kirjoittamiin pyyntöihin analysoimalla pyynnön ja tekstin tunnetilan sekä antaisi tunnetilasta riippuen käyttäjälle erilaisen vastauksen. Keskustelurobotti kehitetiin englanniksi, koska suomen kielen tuki ei ollut vaihtoehtona Watson-pilvipalveluissa, mutta tavoitteena oli tehdä oma suomen kielen tuki keskustelurobottiin. Tunnetila tunnistettiin Watson Tone Analyzerilla, ja keskustelurobotin keskustelurakenne rakennetaan Watson Conversationilla. Tunteiden tunnistamisella Watson Conversationissa säästetään tunnesanojen lisääminen avainsanoihin ja niiden tunnistaminen dialogeissa.

Tämän tarkoituksena oli saada mahdollisimman kognitiivista esittävä keskustelurobottisovellus ilman todellista kognitiivisuutta. Tämän keskustelurobotin käyttötarkoituksiin kuuluivat lennon varaus ja tunteiden tunnistaminen keskustelussa. Tunteiden tunnistaminen on osa keskustelurobotin automaatiota, jossa reagoidaan ja tulkitaan käyttäjän kirjoituskieltä heti sovelluksessa käyttäjälle. Tunteiden tunnistamisella pystyttäisiin tallentamaan käyttäjien käyttäytymistä asiakaspalvelusovelluksessa ja muuttamaan dialogia käyttäjän tunnetilan mukaan. Esimerkiksi jos käyttäjän käyttämä kieli olisi vihaista, keskustelurobotti reagoisi rauhalliseen tapaan. Sovellus kehitettiin Node.js-teknologian pohjalle, jolla pystytään tekemään ja hallitsemaan API-rajapinta pyyntöjä eri REST API -rajapintoihin (Kuva 9).



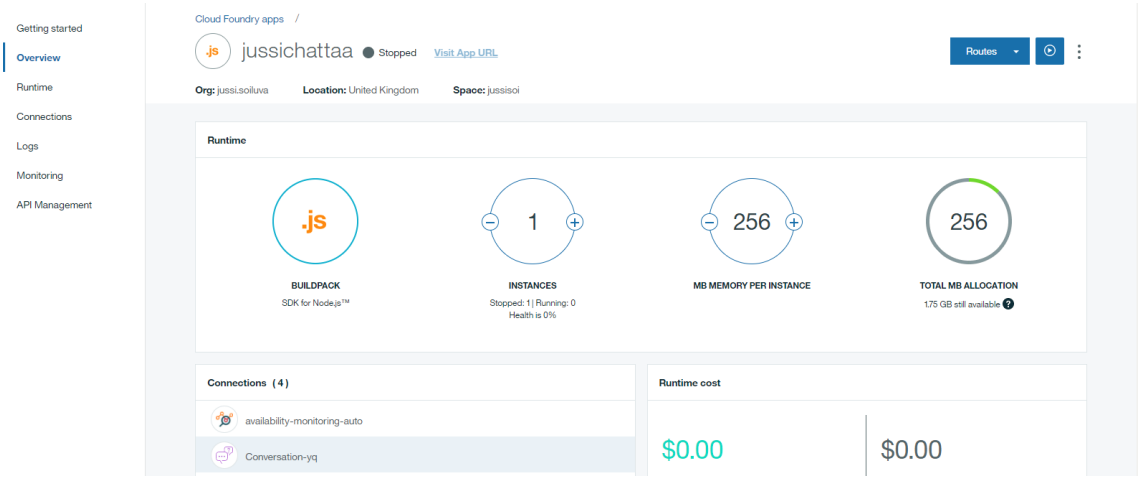
Kuva 9. Insinöörityön teknisen toteutuksen esimerkkirakenne.

Tämän sovelluksen tärkeimpänä toiminnallisuutena on saada lennonvaraus tehtyä keskustelurobotin kysymyksien mukaan ja saada keskustelurobotti reagoimaan käyttäjän kirjoituskielen mukaan eri tavoin. Esimerkiksi jos käyttäjä varaa lentoa vihaisesti, keskustelurobotti tunnistaa vihaisuuden tunteen tekstistä ja reagoi siihen halutulla toiminnalla.

Keskustelurobotin kehittämisen tarkoitus ei ollut luoda sovellusta julkiseen käyttöön. Tarkoituksena oli oppia keskustelurobottiprojektin tekninen toteutus ja ymmärtää toteutuksen haasteet. Tarkoitus oli myös käyttää tämän projektin opittuja tekniikoita tulevaisuuden projekteissa.

4.2 Keskustelurobotin kehitys

Keskustelurobotin kehitys aloitettiin luvussa 3.1 mainitulla tavalla. Ensin käynnistettiin Node.js-pohjainen web-sovellus, jossa oli oma osoite, ja kopiotiin esimerkki-Node.js-web-sovelluskansio paikalliseen kehitykseen. Node.js-web-sovelluskansion tärkeimmät tiedostot olivat Node.js-back-end-pyyntöjenhallitsijatiedosto server.js ja Node.js:ään kuuluva pakkausten hallintakirjastotiedosto package.json. Mukana tulivat myös front-end-puolen HTML-tiedosto ja CSS-tiedosto. Muokkasin HTML-tiedostoon erittäin yksinkertaisen keskustelurobotin ulkoasun, jossa käyttäjä pystyy kirjoittamaan ja lähettämään viestin.



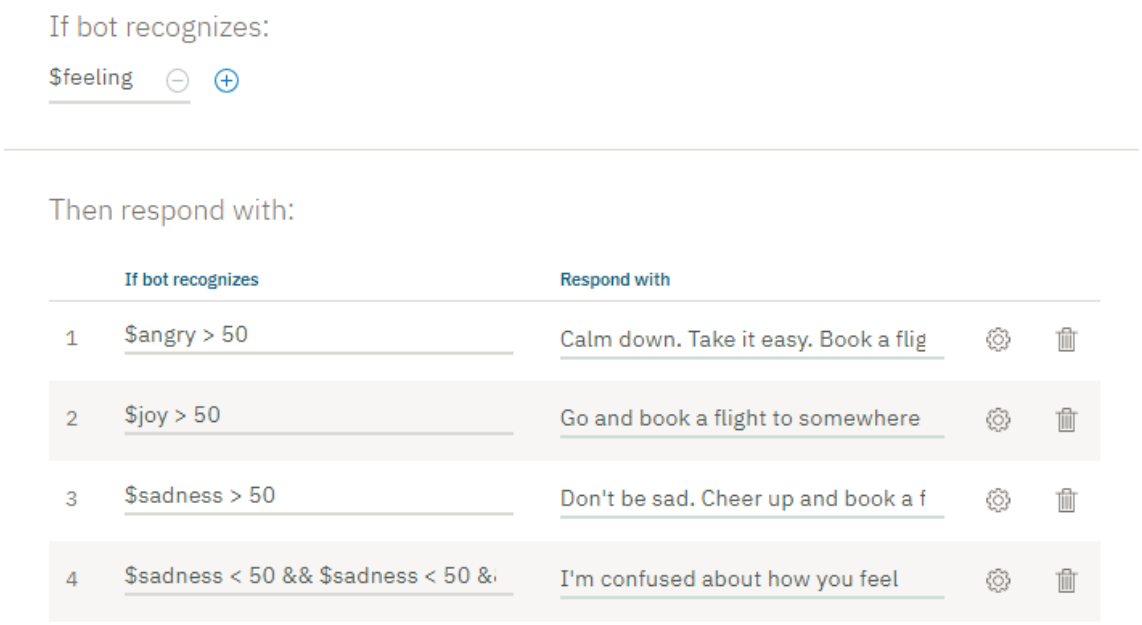
Kuva 10. Kehittäjän Cloud Foundry Node.js projektinhallintaympäristö, jossa voi esimerkiksi seurata valvontalokia.

Kun sovelluksen perusta oli rakennettu, siirryin Cloud Foundry -yhteyksien testailuun, jotta saisin siirrettyä lokaalisti kehitetyn sovelluksen IBM Cloudiin luotuun pilvipalveluun, jossa sovellus on siirron jälkeen käynnissä. Tämä tehtiin luvussa 3.1 mainitulla komentorivisanomalla.

Kun projektin tiedostot oli saatu siirrettyä pilveen, aloin yhdistää Watson Conversation -palvelua lisäämällä tämän Node.js-kirjaston omaan sovellukseen, jolla saadaan Watson-

palvelut käyttöön projektissa. Server.js-tiedostoon lisätään Conversation-palvelu muuttujaksi. Seuraavaksi piti käynnistää IBM Cloudin puolella Watson Conversation -palvelu luvun 3.2 mukaisesti, jotta saisin käyttöoikeustunnukset ja keskustelurobotin dialogin pohjan. Lopuksi tein Conversationin yhdistämisen front-endin puolen HTML-tiedostoon. Käytin HTML-tiedostossa jQuery-Javascript-kirjastoa, jolla tein HTTP POST -pyyntöjä server.js-tiedoston Watson Conversation -palveluun. Samalla yhdistin Conversationin antaman vastauksen käyttöliittymään.

Kun sain testaustoiminnot käyttöliittymässä toimimaan, ryhdyin yhdistämään Watson Tone Analyzeria. Sen yhdistämisen kehitin samalla pohjalla kuin Conversationin: lisäämällä Tone Analyzer -kirjaston server.js-tiedostoon ja lisäämällä Tone Analyzerin käyttöoikeustunnukset objektin sisään. Käyttöliittymä puolelle lisättiin pisteytys 0–100 tunnetilojen tunnistukseen ja yksinkertainen ulkoasu tunnepisteytykselle, jotta voidaan nähdä analysoidun tekstin tulos ja näin pystyä testaamaan eri tunnetilojen vastauksia (Kuva 11).

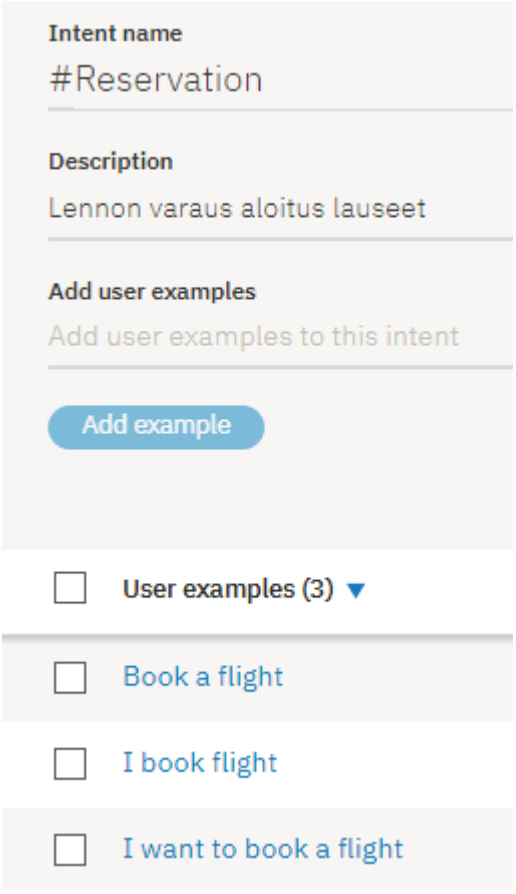


Kuva 11. Esimerkki siitä, miten tunnetilapisteytys muuttaa vastausta Watson Conversationissa.

Piti myös muistaa lähettää tunteiden pistearvot Conversation POST JSON -viestin mukana, jotta myöhemmin pystyisin muokkaamaan Conversation sovellukseen puolella

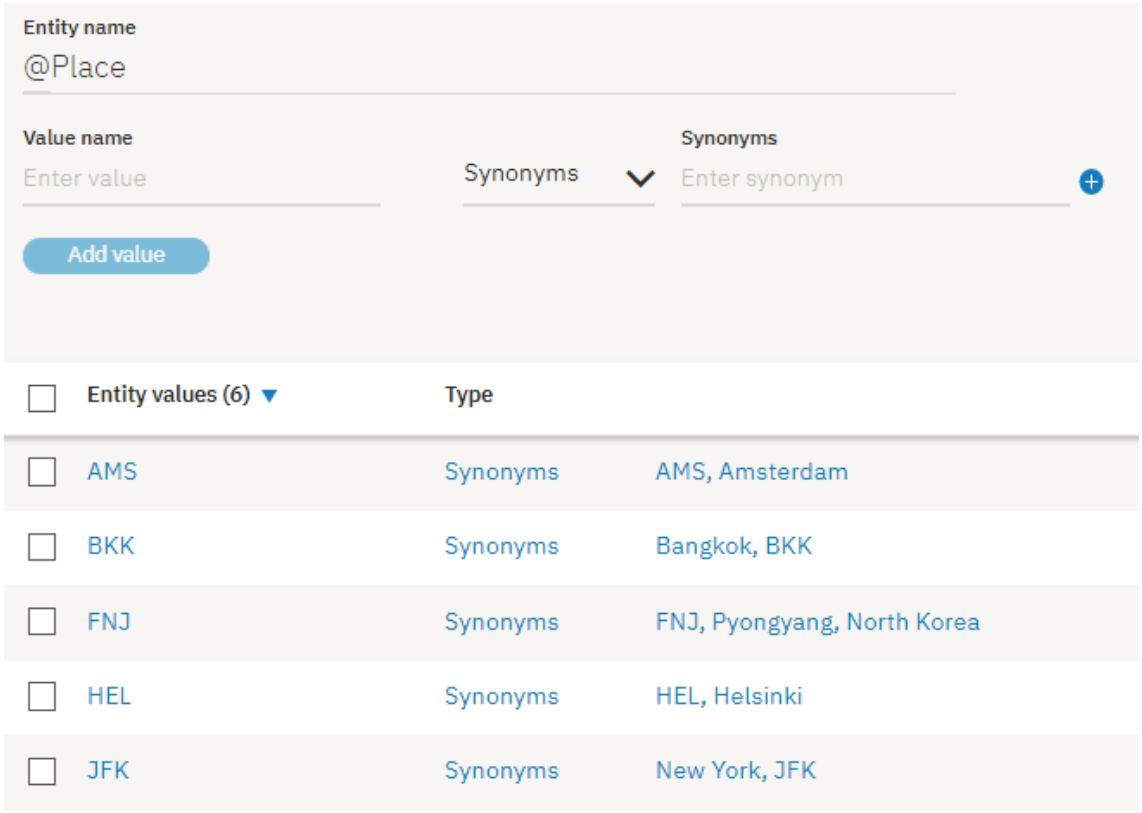
keskustelurobotin keskustelurakennetta tunnetilapistearvojen mukaan. Tämä tehtiin lisäämällä Watson Converstion olemassa olevaan context-olioon. Kuvassa 8 (s. 14) näytetään tunteiden lisäys context-olioon, joka lähetetään Watson Conversationiin. Watson Conversationin käyttöliittymässä context-oliossa olevista tunteista tehdään muuttujia.

Keskustelurobotin keskustelu rakennettiin Watson Conversation sovelluksessa. Ensin lisättiin Watson Conversation -sovelluksessa käyttäjän aikomukset (Intents), jotka koostuivat tervehdyksestä, lennon varauksen aloittamisesta ja kellonajan kysymisestä. Kuvassa 12 näkyvät varausaikomuksen esimerkkilauseet.



Kuva 12. Esimerkki lennonvarauksen aikomuslausahduksista.

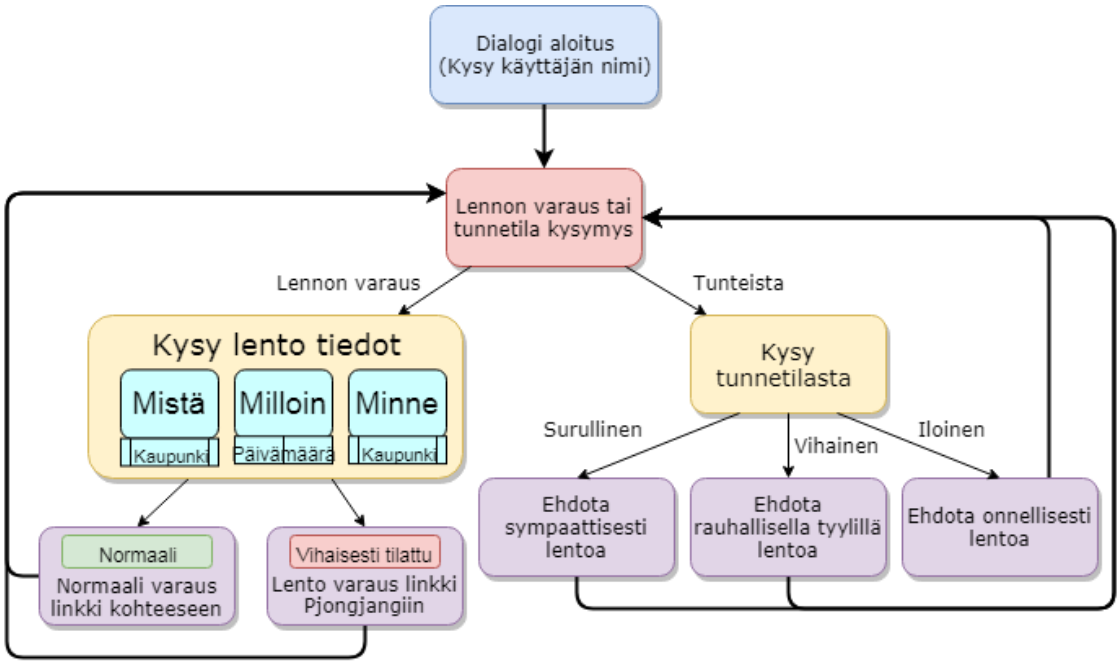
Joukkojen avainsanoja olivat lentokentät, kaupungit ja lennon varmistamiseen vaaditut kyllä/ei-vastaukset (Kuva 13).



Kuva 13. Esimerkki kaupunki joukosta, jossa avainsanoina ovat lentokentän lyhenteet tai kaupungin nimi. Arvona pysyy lentokentän lyhenne, joka lisätään Skyscanner-hakuun.

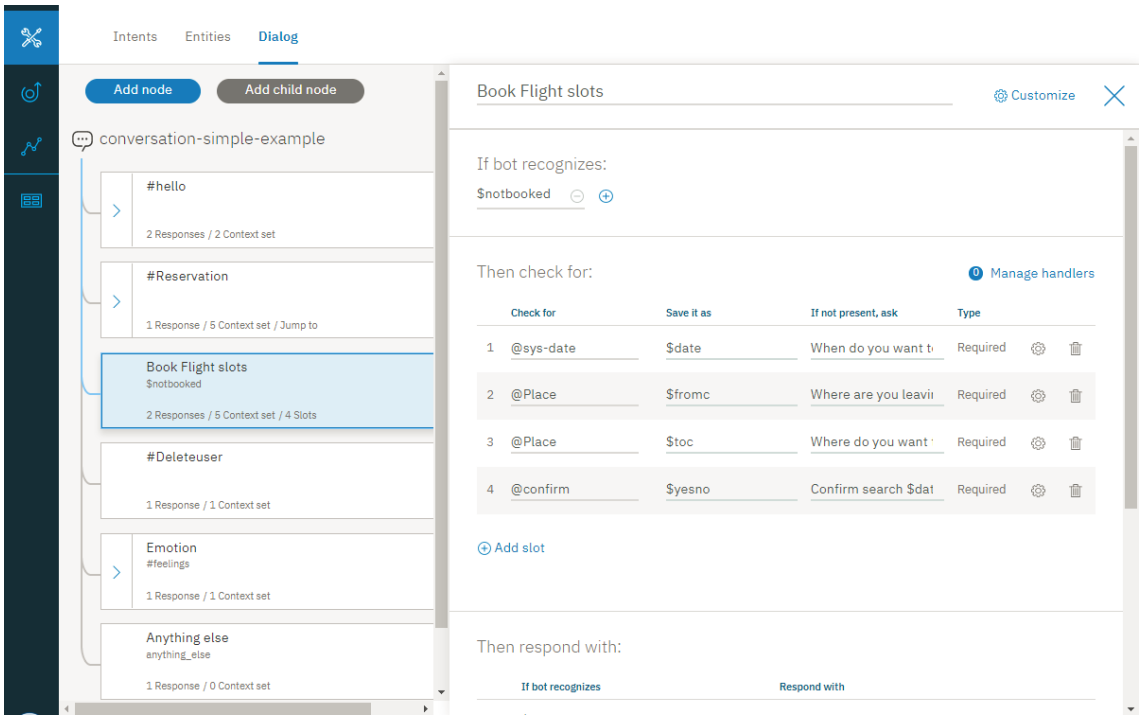
Dialogiosio koostui tervehdyksestä, lennonvarauksesta ja käyttäjän poistamisesta (Kuva 14). Tervehdyksessä keskustelurobotti tervehtii ensin, minkä jälkeen se kysyy käyttäjän nimeä ja ottaa nimen muistiin koko sovelluksen käytön ajaksi. Lennonvarausdialogi aktivoituu, kun käyttäjä kysyy lennonvaraustyyppisen aikomuksen. Kun lennonvaraus on aktivoitu, lennonvarauksessa käytetään Slot-toimintoa, jossa säilytetään kolme kysymystä:

- Milloin lento halutaan varata? (When do you want to book a flight?)
- Mikä on lähtöpaikka? (Where are you leaving from?)
- Mikä on lennon kohde? (Where are you flying to?)



Kuva 14. Vuokaavio lennonvarausdialogin rakenteesta.

Kysymyksiin oletetaan saatavan vastauksiksi avainsanoja, jotta ne rekisteröityvät kyseisen kysymyksen muuttuja-arvoiksi. Lennon varmistuksessa vastataan vain ”kyllä” tai ”ei”, jotka löytyvät entiteettien avainsanoista. Viimeisenä lisäsin Conversation-palveluun tunnepisteytykset lennonvarausdialogiin if-funktiolla, jotta voisin antaa eri vastauksen tunnetilasta riippuen.



Kuva 15. Dialogin ikkuna, jossa slots-kysymysvalikko auki.

Front-endin puolella yhdistin kaikki tarvittavat Node.js-kirjaston palvelut jQuery POST -metodilla omiin POST-pyyntöihin, jotka ottivat yhteyttä Node.js server.js back-end-palveluun. Tämän jälkeen käyttöliittymäpuolella kehitin skyscanner-haun kaikista annetuista vastauksista. Tämä tehtiin ottamalla kopio skyscannerin haku-web-osoitteesta ja asettamalla vastaukset oikeihin paikkoihin.

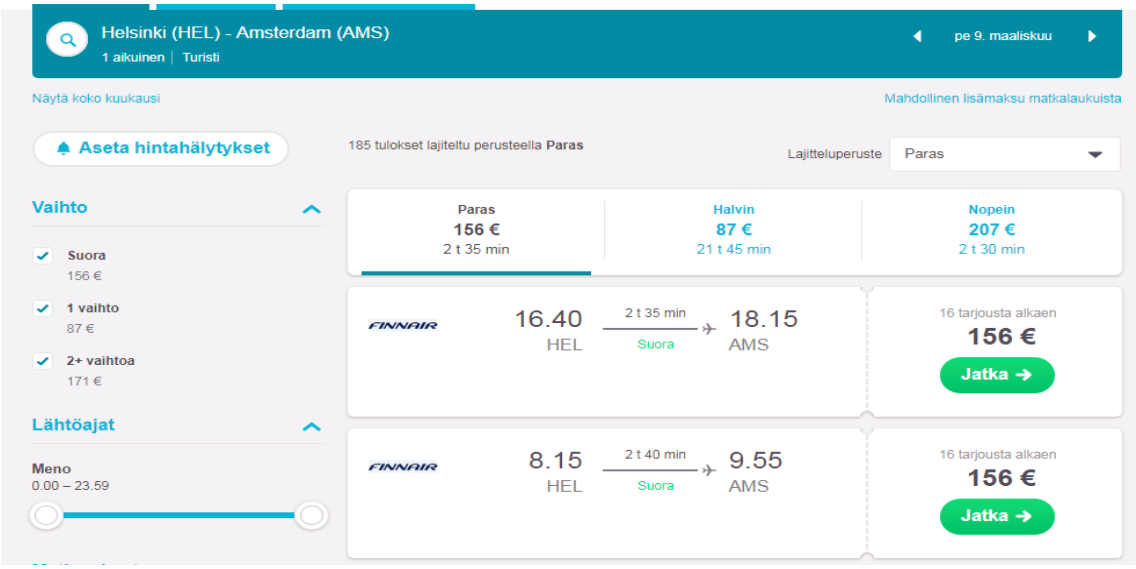
Viimeisenä lisäsin keskustelurobottisovellukseen suomen kielen tuen. Asensin google-translate-api Node.js-paketin, johon valittiin kielet suomesta englanniksi. Tein myös toisen POST-kutsun englannista suomeksi. Conversation-sovellus ymmärtää vain englantia, minkä vuoksi ennen kuin käyttäjän pyyntö lähetetään Conversationiin, se pitää kääntää englanniksi. Sama asia piti toteuttaa toisin päin. Kun vastaus tulee Conversationista käyttäjälle englanniksi, se käännetään englannista suomeksi. Front-endin puoleen lisättiin kytkin, jolla sai suomen kielen tuen käynnistettyä.

4.3 Keskustelurobotin käyttö

Kun keskustelurobotin avaa verkkoselaimessa, se tervehtii ja kysyy käyttäjän nimeä. Keskustelurobotti antaa tämän jälkeen kaksi vaihtoehtoa, mitä tehdä. Keskustelurobotille voi joko kertoa tunteistaan tai varata lennon.

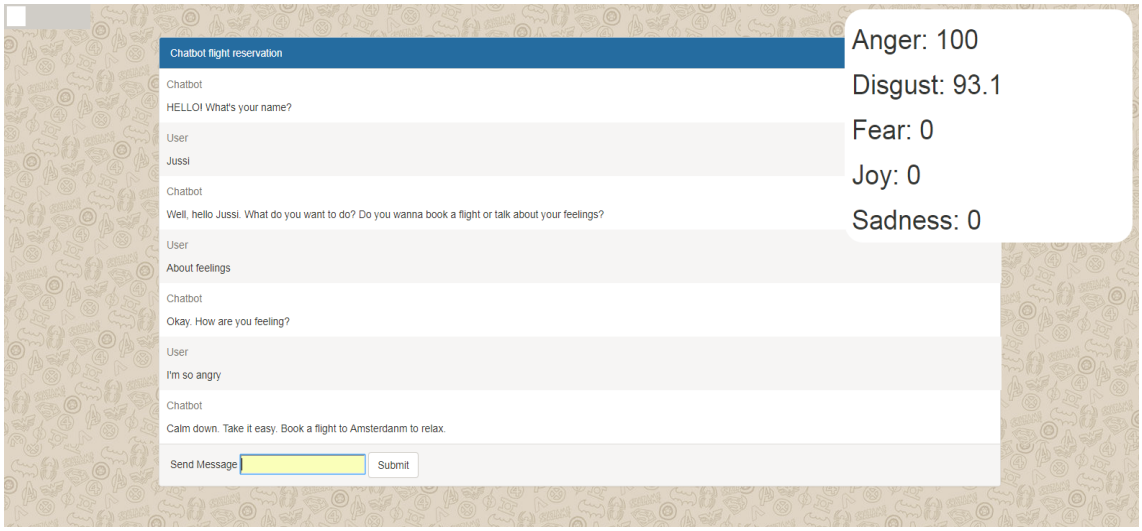
Käyttäjä käynnistää lennon varauksen pyytämällä lennonvarausta. Tämän jälkeen keskustelurobotti käynnistää kysymykset luvussa 4.2 mainitun slot-jonon mukaan. Keskustelurobotti kysyy käyttäjältä samaa kysymystä, jos käyttäjän vastaus ei löydy entiteetin avainsanoista. Keskustelurobotin lentovarauksessa voi vastata joko yksitellen jokaiseen keskustelurobotin kysymykseen tai kirjoittaa kaiken yhdelle riville: ”Milloin?”, ”Mistä?” ja ”Minne?” haluaa matkustaa. Keskustelurobotin lennonvarauskomponenttiin on lisätty tunteidenlukijatila, jossa keskustelurobotti voi reagoida käyttäjän käyttäytymiseen. Esimerkiksi jos käyttäjä tilaa vihaisesti käyttäen kirosanoja tilauksessaan, keskustelurobotti ehdottaa automaattisesti käyttäjälle lentoa Pjongjangiin, Pohjois-Koreaan.

Vastattujen matkatietokysymyksien jälkeen käyttäjä vielä varmistaa lennon varauksen tiedot. Lennon varmistuksen jälkeen käyttäjä saa keskustelurobotilta linkin, joka avautuu skyscanner.com-osoitteeseen, jossa on kaikki keskustelurobotille annetut matkahakutiedot, ja skyscanner-sivustolla se antaa parhaat lentotarjoukset (Kuva 16).



Kuva 16. Esimerkki Skyscannerin hakutuloksesta, kun klikkaa keskustelurobotin tarjoamaa linkkiä.

Jos käyttäjä valitsee tunteista keskustelun, keskustelurobotti kysyy käyttäjältä, miltä hänestä tuntuu kyseisellä hetkellä. Kun käyttäjä on kertonut tunteistaan, tekstin tunneanalysointi alkaa ja riippuen tunnepisteistä voi saada erilaisen vastauksen. Esimerkiksi jos käyttäjän teksti ylittää yli 50 pisteen rajan ilossa, surussa tai vihassa saa käyttäjä siihen sopivan vastauksen. Jos käyttäjän teksti jää alle 50 pisteen keskustelurobotti vastaa, että se ei ymmärrä käyttäjän tunnetilaa. Dialogi keskustelurobotin kanssa päättyy lennon varauksen jälkeen, mutta dialogin voi taas aloittaa alusta tervehtimällä dialogissa. Keskustelurobotissa myös tarjotaan mahdollisuutta käyttää suomen kieltä keskusteluissa. Kuvassa 17 näkyy esimerkki keskustelusta sovelluksen kanssa.



Kuva 17. Keskustelurobotin keskustelu-ulkoasu. Oikeassa ylänurkassa on tunnetilapisteytys. Vasemmassa yläkulmassa on suomen kielen tukikytkin.

4.4 Keskustelurobotin haasteet ja jatkokehitysajatukset

Keskustelurobotin tekemisessä oli paljon eri haasteita. Ensimmäisen keskustelurobotin tekemisen ensimmäinen haaste oli selvittää, mistä aloittaa projekti, koska keskustelurobotin olisi voinut kehittää monella eri tavalla ja palvelujen pyyntöjen järjestämiset olisi myös voinut tehdä monin eri tavoin. Esimerkiksi olisin voinut kehittää tekoälykeskustelurobotin toisella pilvipalvelutuottajalla, mutta en ollut varma niiden luotettavuudesta, joten kehitin keskustelurobotin rakennettavalla mallilla.

Kun tapa ja järjestys oli päätetty, usein haasteet liittyivät IBM Cloud Watson -pilvipalvelun käyttöoikeuksien hallintaan, Conversation-palvelun vääriin vastauksiin ja käyttöliittymäkehityspäätöksiin.

Projektissa on paljon käyttöoikeuksia ja yhteyksien tekoja, joissa pienetkin virheet usein pysäyttävät toiminnallisuuksia kokonaan. Jos Watson Tone Analyzer -käyttäjätunnus on vahingossa väärin, palvelu ei toimi eikä voi olla varma, johtuuko virhe käyttöliittymän puoleisesta koodista vai onko virhe Node.js-puolella.

Watson Conversationissa suurimmaksi haasteeksi tuli saada oikea vastaus käyttäjän kyselyyn web-sovelluksessa. Dialogin kanssa keskustelu toimi hyvin Watson Conversationin kehitysympäristössä, mutta kehitetyssä web-sovelluksessa ei saatu haluttuja vastauksia käyttäjälle. Lopulta löytyi ohje, jossa ohjeistettiin lähettämään Watson Conversationin REST API -pyyntö JSON-objektissa edellisen viestin tiedot seuraavaan viestin objektiin, jotta Watson Conversation REST API osaa seurata dialogin kulkua sen mukaan ja antaa oikeasta dialogista vastauksen. Watson Conversationin kehitysympäristössä testidialogiin lähetetyt viestit pysyivät siis automaattisesti oikeassa dialogissa, mutta web-sovelluksen kehityksessä pitää lähettää REST API:iin edellisen viestin metadatatiedot JSON-objektissa. Itse Watson Conversationissa oli paljon opittavaa: miten käyttää sen muuttujia dialogissa ja kuinka saada aktivoitua juuri oikean dialogi. Tämä näkyi varsinkin siinä, kun sain oikeita vastauksia Watson Conversation käyttöliittymässä, mutta projektiin luodussa web-sovelluksessa saattoi aktivoitua aivan väärä dialogi. Kuvassa 18 näkyvät esimerkki-JSON-tiedot, jotka lähetetään Watson Conversationin dialogiin.

```
{
   "context":{
      "conversation_id":"bcdf2123-10fc-44ec-a411-703b5cc53c1b",
      "system":{
         "dialog_stack":[
            {
               "dialog_node":"slot_1_1506944712586",
               "state":"in_progress"
            }
         ],
         "dialog_turn_counter":3,
         "dialog_request_counter":3,
         "_node_output_map":{
            "node_16_1506946491544":[
               0
            ],
            "node_5_1507114053784":[
               0
            ]
         }
      },
      "continue":false,
      "angry":"12.23",
      "joy":"10.65",
      "disgust":"5.46",
      "fear":"6.02",
      "sadness":"20.82",
      "user":"Jussi",
      "toc":null,
      "date":null,
      "fromc":null,
      "yesno":null,
      "notbooked":true
   }
}
```


Kuva 18. Esimerkki context-objektista, jossa on dialogin metatiedot, jotka lähetetään web-sovelluksesta Watson Conversationiin.

Haasteena oli myös, kuinka laajan dialogin haluaisi rakentaa Watson Conversationiin. Yksityiskohtien lisääminen dialogeihin, esimerkiksi enemmän avainsanoja joukkoihin, lisää avauslauseita aikomukseen tai dialogeista laajempia vie paljon aikaa, joten projektissa pyrittiin pysymään yksinkertaisten pyyntöjen ja dialogien äärellä.

Yksi jatkokehitysmahdollisuus ja nykyiseen sovellukseen yhdistäminen olisi suorittaa enemmän toimintoja itse Watson Conversation -palvelussa. Esimerkiksi Watson Conversationiin olisi voinut antaa vastaukseksi suoraan HTML-koodipohjaisia vastauksia, jotta ei tarvitsisi kehittää web-sovelluksen käyttöliittymän puoleisessa HTML-tiedostossa ollenkaan vastauksen tyylejä. Keskustelurobotin vastaukset eivät siis tulisi käyttöliittymään pelkästään tekstinä vaan valmiina HTML-koodipalasina.

Skyscanner-palvelun haasteena oli, että sai haettua vain yhden suunnan lentoja. Skyscanner API -palvelu, josta olisi saanut meno-paluuvarauksen hakuehtotiedot, olisi ollut maksullinen. Tämän vuoksi ainoaksi vaihtoehdoksi jäi muuttaa skyscannerin haku-web-osoitteen tietoja vain menolentohauksi.

Oikeaan Skyscanner API -palveluun yhdistäminen, jolla saisi meno-paluutiedot lisättyä keskustelurobottiin, olisi antanut enemmän oikean varausjärjestelmän tuntua. Skyscanner API -palvelulla olisi myös voinut lisätä muita hakuehtoja keskustelurobottipalveluun, mikä olisi parantanut palvelun yksityiskohtaisuutta.

Puheominaisuuden lisääminen palveluun olisi voinut tuoda enemmän nykyaikaisten keskustelupalvelujen tunnetta. Puheen lisääminen keskusteluun herättäisi käyttäjässä myös enemmän mielenkiintoa kuin pelkästään tekstin kanssa keskustelu.

4.5 Keskustelurobotin arviointi

Keskustelurobotti toimii perustoiminnoiltaan hyvin ja luotettavasti lennon varauksen löytämisessä ja osaa tunnistaa yksinkertaisesti kirjoitettujen lauseiden tunteita. Toiminnallisuuksia testattiin antamalla sovellus ystäville testattavaksi kertomatta mitään sovelluksen toiminnallisuuksista. Kun ystävät etenivät sovelluksen päädialogin mukaan, kehotin heitä kirjoittamaan vihaisesti keskustelurobottisovellukseen ja kertomaan palvelun erosta.

Palautteesta tuli esille, että sovelluksen selkeät vahvuudet ovat luotettavassa toiminnassa ja heikkoudet löytyvät suppeasta keskustelukirjaston kehittämisestä. Sovellukseen olisi voinut lisätä paljon enemmän avainsanoihin kaupunkeja ja lisätä lentovarauspyyntöihin eri tunnetilan antamia vastauksia, mutta päätoiminnallisuudet toimivat.

IBM Cloud Watson Conversation -keskustelurobotin rakentaminen oli ensin haastavaa, mutta saatuani yhdistettyä oman web-sovelluksen käyttöliittymän Watson Conversationiin rakennettuihin dialogeihin on kehittäminen miellyttävää. Aikomusten ja entiteettisanojen lisääminen on helppoa ja niiden käyttö dialogeissa toimii hyvin. Myös dialogien testaaminen on nopeaa, ja tämä nopeutti kehittämistä. Watson Conversation -sovelluksen kehittämiseen löytyi paljon esimerkkejä internetistä, ja tämä helpotti niissä tilanteissa, joissa ei tiennyt, miten edetä.

Watson Conversationin ainoa ja tietoinen heikkous, mutta samalla vahvuus, ovat sen staattiset avainsanojen lisäämiset. Watson Conversation ei osaa itsestään lisätä sanoja avainsanoihin, ja tämä teettää paljon työtä niiden lisäämiseen projektissa. Toisaalta sen tarkkuus taas pysyy hyvänä, koska se suorittaa vain niitä avainsanoja, jotka palvelulle on opetettu.

Watson Tone Analyzer toimii melko luotettavasti, mutta jos palveluun syötetyssä tekstissä on paljon ristiriidassa olevia tunnesanoja, ei palvelu pysty antamaan selkeää pisteytystä tekstipätkästä. Jos pisteytykset ovat epäselkeitä, käyttäjä saa keskusteluro-botilta saman vastauksen.

5 Yhteenveto

Insinöörityön tavoitteena ollut keskustelurobotti-web-sovellus onnistuttiin kehittämään, ja sen kehittämisestä saatiin paljon uutta arvokasta kokemusta. Sovelluksen jatkokehitykseen on paljon ideoita, mutta rakennettavan keskustelurobotin tärkeimmät ominaisuudet ymmärrettiin ja kehityksessä käytettyjen teknologioiden käyttö opittiin. Kehittämisen aikana myös oppi paljon pilvipalveluiden käytöstä ja siitä, miten hallita useita pilvipalveluita samassa kehitysympäristössä.

Mikäli haluaa kehittää nopeasti yksityiskohtaisen keskustelurobotin, olisi IBM Cloudin Watson Conversation varteenotettava ratkaisu. Tekoälykeskustelurobotin luominen on vielä melko haastavaa ja tuotteen lopullinen luotettavuus epävarmaa. Rakennetulla keskustelurobotilla kehittäjä saa käyttäjälle tarkalleen niitä vastauksia, jotka kehittäjä on itse lisännyt rakennettuun keskustelurobottiin. Rakennetun keskustelurobotin suurin haaste on sen skaalautuvuudessa. Jos keskustelurobottia kehittää yhä suuremmaksi, sen kehittämisestä tulee raskaampaa ja hallitsemisesta vaikeampaa.

Projektissa kehitettyä sovellusta on vaikea verrata tuotannollisiin keskustelurobotteihin, koska se on puutteellinen käyttöliittymältään ja dialogin kanssa käytävät keskustelumahdollisuudet ovat rajalliset verrattuna tuotannollisiin keskustelurobotteihin.

Projektin tärkeimmät onnistumiset tulivat Watson Conversationin ja Watson Tone Analyzerin yhdistämisessä. Node.js-tekniikan ansiosta on helppo tehdä IBM Cloudin Watson

-pilvipalvelujen REST API -rajapintaan pyyntöjä, joiden vastauksia yhdisti pyyntöjen jälkeen omaan käyttöliittymään.

Lähteet

1 Complete guide on chatbots – Development to Promotion. 2016. Verkkodokumentti. Maruti Techlabs. <https://www.marutitech.com/complete-guide-chatbots/>. Luettu 10.10.2017.

2 Miller, Glenn. 2016. Chatbot vs. AI Bot - which is here to stay, which to invest in and why? Verkkodokumentti. <https://chatbotslife.com/chatbot-vs-ai-bot-which-is-here-to-stay-which-to-invest-in-and-why-7ea79a454df3>.Luettu 10.10.2017.

3 Salecha, Manisha. 2016. Story of ELIZA, the first chatbot developed in 1966. Verkkodokumentti. <https://analyticsindiamag.com/story-eliza-first-chatbot-developed-1966/ >. Luettu 10.10.2017.

4 Hyken, Shep. 2017. AI And Chatbots Are Transforming The Customer Experience. Verkkodokumentti. <https://www.forbes.com/sites/shephyken/2017/07/15/ai-and-chatbots-are-transforming-the-customer-experience/#423f519041f7>. Luettu 5.1.2018.

5 Hooper, Matt. 2017. Why AI doesn't need to think for itself yet. Verkkodokumentti. < https://www.mycustomer.com/community/blogs/matt-hooper/why-ai-doesnt-need-to-think-for-itself-yet>. Luettu 10.10.2017.

6 Watson Conversation Tool Overview. 2017. Verkkodokumentti. IBM Watson, Youtube Inc.<https://www.youtube.com/watch?v=sSfTcxDrmSI>. Luettu 10.10.2017..

7 Building a client application. 2017. Verkkodokumentti. IBM. <https://console.bluemix.net/docs/services/conversation/develop-app.html#building-a-client-application >. Luettu 8.1.2018.

8 Create intent. 2017. Verkkodokumentti. IBM. <https://www.ibm.com/watson/developercloud/conversation/api/v1/?node#create_intent>. Luettu 8.1.2018.

9 Testing in Slack. 2017. Verkkodokumentti. IBM. <https://console.bluemix.net/docs/services/conversation/test-deploy.html#testing-in-slack>. Luettu 8.1.2018.

10 Amazon Lex FAQs. 2018. Verkkodokumentti. Amazon Web Services Inc. <https://aws.amazon.com/lex/faqs/>. Luettu 20.2.2018.

11 Introducing Amazon Lex: Service for Building Voice/Text Chatbots - March 2017 AWS Online Tech Talks. 2017. Verkkodokumentti. Amazon Web Services - Webinar Channel, Youtube Inc.<https://www.youtube.com/watch?v=tAKbXEsZ4Iw>. Luettu 5.1.2018.

12 Managing Conversation Context. 2018. Verkkodokumentti. Amazon Lex Developer Guide, Amazon Web Services Inc.<https://docs.aws.amazon.com/lex/latest/dg/context-mgmt.html>. Luettu 20.2.2018.

13 PostText. 2018. Verkkodokumentti. Amazon Lex Developer Guide, Amazon Web Services Inc. <https://docs.aws.amazon.com/lex/latest/dg/API_CreateIntentVersion.html>. Luettu 20.2.2018.

14 O'Boyle Britta & Grabham, Dan. 2018. What is Alexa? And what can Amazon Echo do?. Verkkodokumentti. <https://www.pocket-lint.com/smart-home/news/amazon/138846-amazon-echo-what-can-alexa-do-and-what-services-are-compatible>. Luettu 5.1.2018.

15 Titcomb, James. 2017. How to use the Amazon Echo and Alexa. Verkkodokumentti. <http://www.telegraph.co.uk/technology/0/use-amazon-echo-alexa/>. Luettu 5.1.2018.

16 Reisinger, Don. 2017. Apple's Siri Has Lost Millions of Users Over the Last Year. Verkkodokumentti. <http://fortune.com/2017/07/11/apple-siri-usage/>. Luettu 8.1.2018.

17 Amazon Lex FAQs. 2018. Verkkodokumentti. Amazon Web Services Inc. <https://aws.amazon.com/lex/faqs/>. Luettu 20.2.2018.

18 Clifford, Catherine. 2017. AI Here's how Siri made it onto your iPhone. Verkkodokumentti. <https://www.cnbc.com/2017/06/29/how-siri-got-on-the-iphone.html>. Luettu 8.1.2018.

19 O'Boyle, Britta. 2015. What is Siri? Apple's personal voice assistant explained. Verkkodokumentti. < https://www.pocket-lint.com/apps/news/apple/112346-what-is-siri-apple-s-personal-voice-assistant-explained/>. Luettu 8.1.2018.

20 SiriKit. 2018. Verkkodokumentti. Sirikit, Apple Inc. <https://developer.apple.com/sirikit/>. Luettu 8.1.2018.

21 About. 2017. Verkkodokumentti. IBM. <https://console.bluemix.net/docs/services/tone-analyzer/index.html#about> Luet-tu 8.1.2018.

22 The science behind the service. 2017. Verkkodokumentti. IBM. <https://console.bluemix.net/docs/services/tone-analyzer/science.html#the-science-behind-the-service>. Luettu 8.1.2018.

23 What is Node.js? 2017. Verkkodokumentti. Tutorialspoint. <https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm>. Luettu 8.1.2018.

24 Express. 2017. Verkkodokumentti. NPM Inc. <https://www.npmjs.com/package/express>. Luettu 8.1.2018.

25 Cfenv. 2017. Verkkodokumentti. NPM Inc. <https://www.npmjs.com/package/cfenv>. Luettu 8.1.2018.

26 Body-parser. 2017. Verkkodokumentti. NPM Inc. <https://www.npmjs.com/package/body-parser>. Luettu 8.1.2018.

27 Google-translate-api. 2017. Verkkodokumentti. NPM Inc. <https://www.npmjs.com/package/google-translate-api>. Luettu 8.1.2018.

28 Watson-developer-cloud. 2017. Verkkodokumentti. NPM Inc. <https://www.npmjs.com/package/watson-developer-cloud>. Luettu 8.1.2018.

Insinöörityön keskustelurobotti

https://jussichattaa.eu-gb.mybluemix.net/