# KESKUSTELUROBOTTI SISÄLLÖNHALLINTAJÄRJESTELMÄÄN

Case: Cubescom Digital Oy

LAHDEN AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan insinööri (AMK)
Ohjelmistotekniikka
Syksy 2018
Sanna Lahtinen

**Tiivistelmä**


| Tekijä(t) | Julkaisun laji | Valmistumisaika |
|---|---|---|
| Lahtinen, Sanna | Opinnäytetyö, AMK | Syksy 2018 |
| | Sivumäärä | |
| | 30 | |


Työn nimi

**Keskustelurobotti sisällönhallintajärjestelmään**
Case: Cubescom Digital Oy

Tutkinto

Tieto- ja viestintätekniikan insinööri (AMK)


Tiivistelmä

Opinnäytetyössä käydään läpi, miten keskustelurobotti eli chatbot rakennetaan sisällönhallintajärjestelmään.Toimeksiantajana toimi Cubescom Digital Oy. Yrityksen liiketoimintaa pyrittiin parantamaan kyseisellä ratkaisulla sekä helpottamaan asiakaspalvelun työn määrää.

Toteutuksessa chatkanavan pohjana toimi IBM:n kehittämä Watson tekoäly. Watson tarjoaa useita tekoälyyn pohjautuvia lisäosia, kuten luonnollisen kielen ymmärtämisen, visuaalisen tunnistuksen ja nopeutetun tiedon optimoinnin. Ratkaisuna kyseiseen toteutukseen käytettiin Watsonin Assistant-palvelua, joka mahdollistaa chatkeskustelun luomisen.

Toteutus rakennettiin asiakaspalvelutarkoitukseen yrityksen uusille verkkosivuille. Koska verkkosivut olivat WordPress-pohjaiset, työssä kerrotaan kuinka WordPressin asennus tapahtuu, sekä millainen on sen rakenne ja kuinka chatbotti integroitiin tähän sisällönhallintajärjestelmään.

Työ pohjautuu suureksi osaksi omakohtaiseen kokemukseen ja tekemiseen. Chatkanavan toiminnallisuus toteutettiin käyttäen PHP- ja JavaScript-ohjelmointikieliä. Haasteena työssä oli juuri oikeanlaisen ratkaisun toteuttaminen yrityksen käyttötarkoitusta varten.



Asiasanat

IBM Watson, tekoäly, chat, chatbot, WordPress, plugin, PHP, JavaScript, jQuery, JSON

**Abstract**


| Author(s) | Type of publication | Published |
|---|---|---|
| Lahtinen, Sanna | Bachelor's thesis | Autumn 2018 |
| | Number of pages | |
| | 30 | |


Title of publication

**Chatbot in the content management system**
Case: Cubescom Digital Oy

Name of Degree

Information and communications technology


Abstract

The objective of this Bachelor's thesis was to build a chatbot for a contact management system. The thesis was commissioned by Cubescom Digital Oy. The purpose of the automated chat was to improve the business of the company and to make customer service work easier.

The chat was based on IBMs artificial intelligence called Watson. Watson offers many different AI-based services like natural language understanding, visual recognition and high-speed data optimization. Watson's Assistant service was used to implement the chat.

The chat was built for customer service on the company's new website. The website was created using WordPress. It is explained in the thesis how WordPress is installed, what kind of architecture it has and how the chatbot was integrated in the contact management system.

The thesis mostly based on the author's own experience and work. It is explained in the work how the chat was made by using the PHP and JavaScript scripting languages. The challenge of the work was finding the best solution for the company's purposes.



Keywords

IBM Watson, artificial intelligence, chat, chatbot, WordPress, plugin, PHP, JavaScript, jQuery, JSON

SISÄLLYS

# 1 JOHDANTO

Yrityksen tavoitteelliseen markkinointiin liittyy vahvasti verkkosivut, jotka tuovat yritykselle näkyvyyttä sekä asiakkaita. Asiakkaan on helppo lähestyä yritystä ja ottaa yhteyttä, kun sivut ovat visuaalisesti toimivat ja yhteydenotto on tehty helpoksi. Yhteydenottolomakkeiden lisäksi chat-kanava on moderni ratkaisu myynnin edistämiseen. Tänä päivänä asiakaspalvelun chat-ikkuna ponnahtaa monella sivulla ensimmäisenä esille. Chatin taustalla on joko asiakaspalvelija, joka vastaa käyttäjän kysymyksiin reaaliajassa, tai chatbot palvelu, jolla tarkoitetaan automatisoitua chat-kanavaa.

Opinnäytetyön tavoite on rakentaa Cubescom Digital Oy:n uusille verkkosivuille automatisoitu chat. Cubescom Digital Oy on suomalainen ICT-talo, joka on toiminut Lahdessa jo yli 30 vuotta. Yrityksellä on noin 200 asiakasta Suomessa sekä Pohjoismaissa. (Cubescom 2018.) Osaaminen on laajaa sekä yritys kasvaa jatkuvasti, joten myös markkinointiin kiinnitetään enemmän huomiota. Projekti aloitettiin suunnittelemalla ja toteuttamalla yritykselle kokonaan uudet verkkosivut, jotka tukevat myyntiä jatkossa paremmin. Näin ollen myös chat-kanava haluttiin uudistaa ja modernisoida. Edellinen chatpalvelu vanhoilla verkkosivuilla oli toteutettu perinteisemmin, jossa käyttäjä oli suoraan kontaktissa asiakaspalvelijaan, mutta vain rajattuina päivinä ja kellonaikoina. Kun asiakaspalvelija ei ollut tavoitettavissa, chat-ikkunan tilalle aukesi yhteydenottolomake. Chatin toiminta perustuu asiakaspavelijan tavoitettavuuteen.

Asiakaspalvelun parantamisen vuoksi, uusien sivujen chat-kanava haluttiin toteuttaa hyödyntämällä tekoälyä. Opinnäytetyössä käydään läpi miten kyseinen toteutus voidaan ratkaista ja kuinka valittua palvelua voidaan hyödyntää keskustelurobottia rakentaessa.

# 2 KESKUSTELUROBOTTI

## 2.1 Chatbot

Chatbot on ohjelma, joka on suunniteltu simuloimaan keskustelua ihmisten kanssa. Se on avustaja, joka viestii käyttäjien kanssa tekstiviesteillä, chat-ikkunoiden välityksellä tai muissa sovelluksissa. Tarkoituksena chatbotilla on auttaa yrityksiä pääsemään lähemmäs asiakkaita. Myös kuluttajat ovat kiinnostuneita chatbottien tekniikasta. Marraskuussa 2017 Kreikan Thessalonikissa järjestetyssä Internet Science-konferenssissa esitettiin tutkimus, jonka mukaan ihmiset valitsevat ennemmin vuorovaikutuksen chatbotin kanssa kuin toisen ihmisen. (Brandtzaeg & Folstad 2017.) Tutkimuksen mukaan keskeiset tekijät, jotka motivoivat käyttäjiä, ovat

- tuottavuus. Chatbot esittää vastaukset paljon nopeammin kuin asiakaspalvelija.
- viihde. Käyttötarkoituksestaan riippuen, chatbotit myös viihdyttävät ihmisiä. Ne saattavat olla esimerkiksi silloin ajanvietteenä, kun käyttäjällä ei ole muuta tekemistä.
- sosiaalisuus. Chatbot voi myös auttaa käyttäjää keskustelutaitojen kehittämisessä. Ne antavat muun muassa mahdollisuuden puhua ilman tuomitsevuuden tunnetta.
- uteliaisuus. Chatbotin uutuus viehettää ihmisiä, ihan kuin mikä tahansa uusi tuote tai villitys. (Veretskaya 2017.)

Suurin haaste automatisoidussa chatissa kuitenkin on se, kuinka chatbot opetetaan ymmärtämään käyttäjän luonnollista kieltä. Jotkut tekoälyratkaisut sisältävät luonnollisen kielen tuen, mutta sitä ei ole vielä tänä päivänä kehitetty niin pitkälle, että se olisi käytössä monilla eri kielillä. Kaikissa liiketoimissa asiakkaat ja jokainen kohderyhmä ilmaisevat itseään omalla tavallaan. Kieleen vaikuttavat muun muassa markkinoiden sen hetkiset mainoskampanjat, maan poliittinen tilanne ja tuotteiden julkaiseminen. Ihmisten puhuminen muuttuu sään, kaupungin, mielialan ja kuukauden ajankohdan mukaan. Siksi chatbotin kouluttaminen ymmärtämään kaikki käyttäjän syötteet, vaatii valtavasti työtä. (Veretskaya 2017.)

Riippuen siitä, miten botit on ohjelmoitu, ne voidaan jakaa kahteen eri luokkaan: yksinkertaiset ja älykkäät chatbotit. Yksinkertainen chatbot tarkoittaa sellaista palvelua, joka toimii

tiettyjen komentojen mukaan, jotka sille on etukäteen määritelty. Älykkäät chatbotit taas kehittyvät ja niitä pystytään kouluttamaan. (Veretskaya 2017.)

Yksinkertaiset botit toimivat ennalta kirjoitettujen avainsanojen perusteella. Kehittäjän on kirjoitettava jokainen näistä komennoista erikseen käyttämällä säännöllisiä lausekkeita (regular expression) ja merkkijonoja. Jos käyttäjä esittää botille kysymyksen ilman ennalta määrättyä avainsanaa tai jos avainsana tai merkkijono on virheellinen, botti ei sitä ymmärrä ja vastaa sen mukaisesti käyttäjälle. Kyseisissä tilanteissa yksinkertainen botti määritellään vastaamaan käyttäjän syötteeseen usein tarkentavalla kysymyksellä. Botti voi palauttaa käyttäjälle virheviestin tai pyytää tätä esittämään asiansa toisella tavalla.

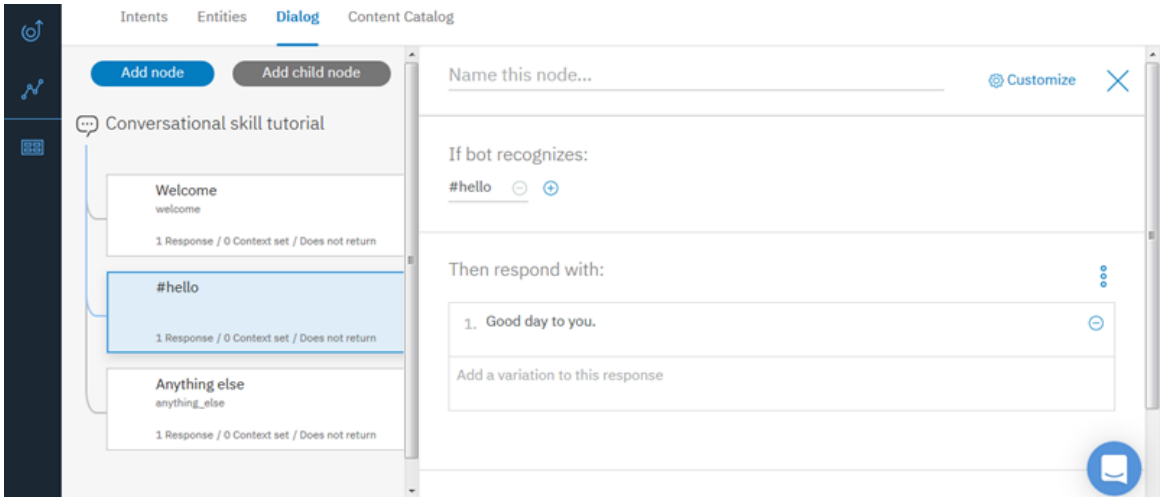
Älykäs chatbot käyttää tekoälyä. Ennalta määritettyjen vastausten sijaan, botti reagoi käyttäjän syötteisiin asiaankuuluvilla ehdotuksilla. Kaikki käyttäjän syötteet kirjataan ylös botin hallintaan myöhempää käyttöä varten. Kehittäjä pystyy jälkikäteen opettamaan bottia ymmärtämään käyttäjän kysymyksiä, johon botti ei vastausta hetkellisesti löytänyt. Forresterin raportissa mainitaankin tähän viitaten, että tekoäly ei ole taikuutta eikä se ole valmis tuottamaan ihmisille ainutlaatuisia kokemuksia, vaan se vaatii edelleen työtä. (Ask, Facemire, Hogan, Gill, Jacobs, Naparstek, Willsea, Galan & Harrison 2016.) Kuitenkin muun muassa kirjotusvirheiden suodattaminen onnistuu älykkäältä botilta paljon paremmin kuin yksinkertaiselta. Jos käyttäjän syötteessä on virhe, pystyy älykäs botti silti löytämään oikeaan kontekstiin kuuluvan vastauksen.

## 2.2 IBM Watson

Watson on IBM:n kehittämä tekoäly, jota on kehitetty vuodesta 2007 alkaen. IBM eli International Business Machines Corporation on teknologiayritys, joka on erikoistunut pilvialustoihin ja kognitiiviseen tietojenkäsittelyyn. (IBM 2018c.)

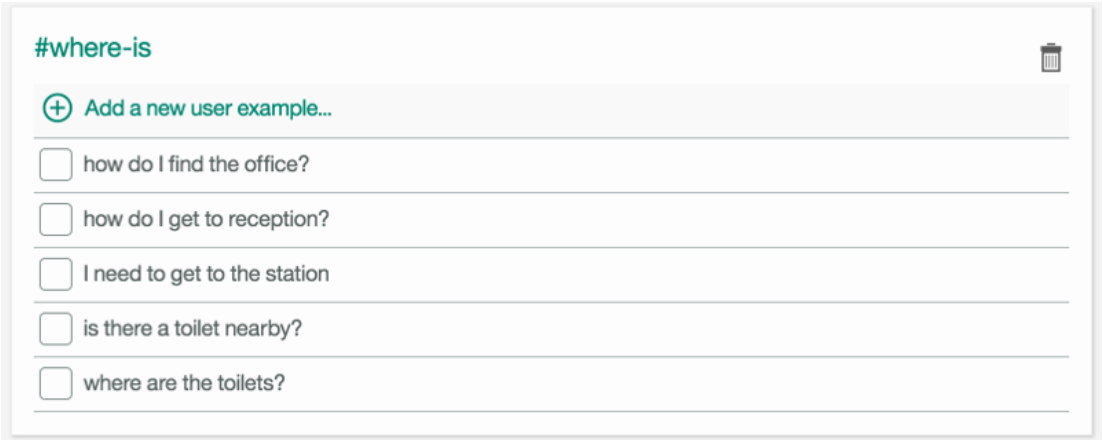
IBM:n kehittämä tekoäly on opetettu ymmärtämään käyttäjän luonnollista kieltä ja päättelemään kysymyksiin vastauksia itsenäisesti. Watson on rakennettu niin, että se ei tarvitse kehittäjältä suuria määriä dataa. (IBM 2018a.) Käyttäjä toimii vuorovaikutteisesti keskustelurobotin kanssa käyttöliittymän kautta. Kun sovellus vastaanottaa käyttäjän viestin, se lähetetään eteenpäin Watson Assistant -palveluun. Samalla sovellus muodostaa yhteyden työtilaan, johon on luotu keskustelulle rakenne. Palvelu tulkitsee sitten käyttäjän syötteen ja näin ohjaa keskustelua etenemään loogisesti. Kuva 1 on kuvankaappaus Watson työtilan keskustelun rakenteesta. Automaattinen tervehdysviesti aloittaa keskustelun, jonka jälkeen keskustelu botin kanssa etenee käyttäjän syötteisiin vastaten. Watsonin dialogi sisältää solmuja (node), jotka on määritelty sen mukaan, mitä keskustelun tekijä on etukäteen voinut chatbotille määrittää. Nodet sisältävät aina tietyn erillisen tunnisteen (intent) tai

tunnistekokonaisuuden (entity), jonka perusteella botti käyttää dialogia edetäkseen keskustelussa. Tunnisteella tarkoitetaan avainsanaa, joka liittyy keskustelun sisältämiin aiheisiin. Esimerkiksi kuvassa 1 näkyy tunniste, joka on määritetty yleiseksi tervehdysviestiksi avainsanalla ”hello”.




KUVA 1. Watson työtilan keskustelurakenne (IBM Cloud Documentation 2018a)

Kehittäjä luo keskustelun rakenteen Watson työtilaan. Työtilaan luodaan vaihtoehtoisia vastauksia sen varalle, mitä käyttäjä kirjoittaa keskustellessaan botin kanssa. Vastaukset luodaan sen mukaan, mihin käyttötarkoitukseen chatbot on luotu. Esimerkiksi kuva 2:ssa on määritelty tunniste avainsanalla ”where-is”, jonka botti tunnistaa käyttäjän kysyessä reittiohjeita. Kysymyksiin on valmiiksi luotu vastaukset, joita botti pystyy oman kykynsä mukaan soveltamaan.



KUVA 2. Kuvankaappaus tunnisteesta (intent) (Burns 2016)

Yksittäiset tunnisteet erottaa Watson Assistantissa #-etumerkistä. Tunnistekokonaisuudet erottaa toisistaan etumerkillä @. Tällaista tunnistekokonaisuutta kannattaa hyödyntää rakentaessa dialogia, kun käyttäjän vastausviesti botille kysymykseen on esimerkiksi kyllä tai ei. Myöskin sähköpostiosoite, puhelinnumero sekä käyttäjän nimi on helppo sijoittaa tunnistekokonaisuuden sisälle säännöllisenä lausekkeena.

Watson perustuu tekoälyyn. Tämä viittaa siihen, että se osaa myös oppia itse. Kehittäjä on silti Watsonin oppimisesta päävastuussa. Kun käyttäjä käy botin kanssa keskusteluja, ne tallentuvat työtilaan, jota kehittäjä voi käydä läpi. Tapauksissa, joissa botilla ei ole ollut käyttäjälle vastausta tai se ei ole ymmärtänyt käyttäjän kysymystä, pystyy kehittäjä jälkikäteen opettamaan bottia. Tunnistamatta jäänyt käyttäjän syöte voidaan määritellä jo valmiina olevan tunnisteen alle, tai vaihtoehtoisesti luoda täysin uusi tunniste. Kun käyttäjä kysyy seuraavan kerran samaisesta aiheesta, osaa botti vastata siihen. Näin Watson oppii jatkuvasti uutta.

### 2.2.1 Watsonin lisäosat

Watson Assistantin ohelle on kehitetty useampi rajapinta ja työkalu. Rajapintojen avulla voidaan kehittää chatbottia lisäämällä sen ominaisuuksia sekä tietävyyttä. Upotettu tekoäly osaa rajapintojen avulla muun muassa hyödyntää kone- sekä syväoppimista ja pystyy jakamaan käyttämäänsä ja luomaansa dataa muuallakin. Se pystyy käyttämään koneoppimista muun muassa visuaaliseen tunnistukseen. Watson pystyy oivaltamaan itse asioita nopeutetun tietojen optimointitoimintojen avulla. Tällaisia lisätyökaluja ovat muun muassa luonnollisen kielen ymmärtäminen sekä piilotettujen arvojen löytäminen syötteiden vastauksiksi. (IBM 2018b.)

Tekstin kääntäminen puheeksi tai puheen kääntäminen tekstiksi on myös Watsoniin rakennettu työkalu. Käyttäjän jatkuva analysointi on tärkeää tekoälyn oman oppimisen kannalta. Watsonilla on myös mahdollisuus tulkita käyttäjän persoonallisuutta ja oppia näin käyttäjän tunnetilasta esimerkiksi analysoimalla syötetyn tekstin sävyä. Kun käytetty kieli on tuettu, Watson oppii myös nopeammin. Lisäosat toimivat tuetuilla kieleillä myös käytännössä sujuvammin. Esimerkiksi luonnollisen kielen ymmärtäminen on yksi tällaisista Watsonin lisäosista, joka toimii paremmin, kun käytössä oleva kieli on tuettu. Jos kielellä ei ole täyttä tukea, luonnollisen kielen ymmärtäminen ja kirjotusvirheiden tulkitseminen voi olla haastavaa botille. Watsonin laaja tarjonta erinäisiin lisäosiin vaikuttaa sen markkina-

arvoon. Kehittäjä pystyy luomaan Watsonin työkaluilla mitä erilaisempia ratkaisuja pohjautuen tekoälyyn. (IBM 2018b.)

### 2.2.2 Kielisyys

Watson Assistant tukee rajatun määrän kieliä. Kielien tuki tarkoittaa sitä, että kuinka Watson on ohjelmoitu ymmärtämään tiettyä kieltä; toimiiko esimerkiksi luonnollisen kielen ymmärtäminen tai puheentunnistus tuetulla kielellä. Englannin kieleen Watsonilla ja sen jokaisella työkalulla on täysi tuki. Kuvassa 3 on kuvankaappaus taulukosta, jossa on lueteltuna Watsonin tukemat kielet. Taulukossa on lueteltuna muutama Watsonin ominaisuus, joilla on kielituki kyseisillä kielillä.

| Language | Defining intents and dialog | Absolute scoring and 'Mark as irrelevant' | Search | Content Catalog |
|---|---|---|---|---|
| English (en) | GA | GA | GA | GA |
| Arabic (ar) | GA | Beta | NA | NA |
| Chinese (Simplified) (zh-cn) | GA | GA | Beta | NA |
| Chinese (Traditional) (zh-tw) | Beta | Beta | Beta | NA |
| Czech (cs) | Beta | Beta | Beta | NA |
| Dutch (nl) | GA | GA | Beta | NA |
| French (fr) | GA | GA | Beta | GA |
| German (de) | GA | GA | Beta | GA |
| Italian (it) | GA | GA | Beta | GA |
| Japanese (ja) | GA | GA | Beta | GA |
| Korean (ko) | GA | GA | Beta | NA |
| Portuguese (Brazilian) (pt-br) | GA | GA | Beta | GA |
| Spanish (es) | GA | GA | Beta | GA |

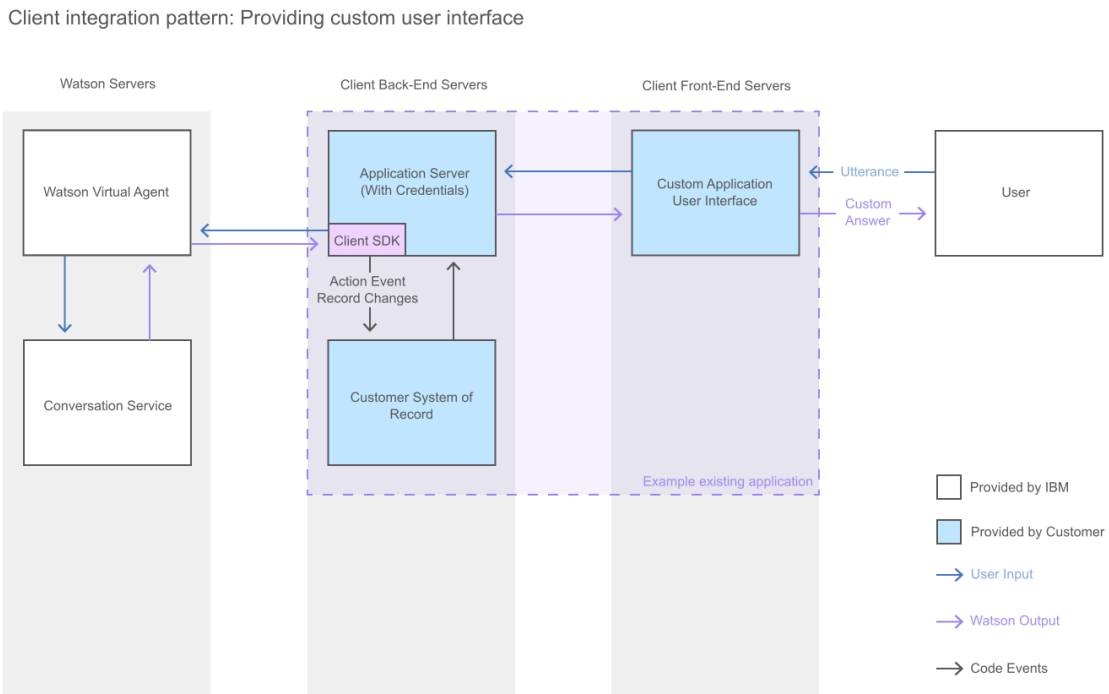
Feature support details

KUVA 3. Watsonin kielitukemia ominaisuuksia (IBM Cloud Documentation 2017)

Lisäosien kielten tuki on rajattu kolmeen luokkaan: GA, Beta sekä NA. GA:lla tarkoitetaan lisäosan ominaisuutta, joka on käytettävissä kyseisellä kielellä täysin. Ominaisuutta voidaan kuitenkin kehittää ja päivittää senkin jälkeen, vaikka se olisi yleisesti saatavilla. Beta tarkoittaa taas, että ominaisuus on tuettu vain beta-julkaisuna. Sitä vielä kokeillaan ja kehitetään sen mukaan, ennen kuin se on yleisesti saatavilla tuetulla kielellä. NA ilmaisee sitä, että ominaisuus ei ole saatavilla kyseisellä kielellä. Suomen kieli ei ollut Watsonin tukema kieli botin toteutushetkellä. Se kuitenkin tarjoaa projektitasoisen tuen suomen- ja ruotsin kielille, joita projektissa englannin lisäksi käytettiin.

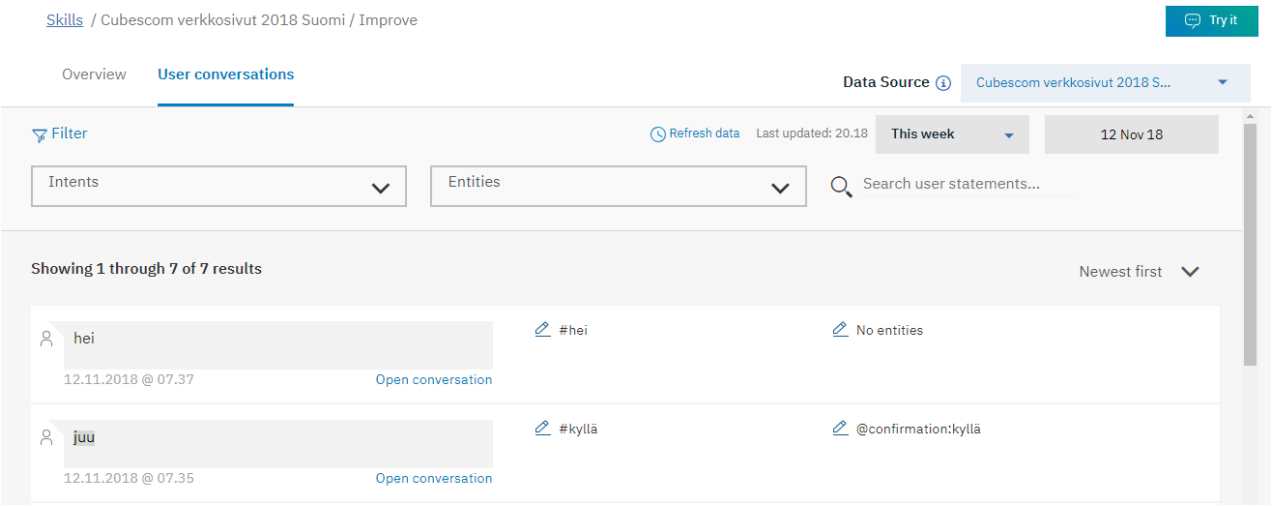
### 2.2.3 Alustaratkaisu

Watson-pohjainen chatbot voidaan kehittäjästä riippuen asentaa täysin kustomoituun käyttöliittymään. Erinäiset CMS-alustat, kuten WordPress, tarjoavat valmiita lisäosia, joihin oma chatbot voidaan asentaa ilman ymmärrystä koodaamisesta. IBM:n dokumentaatio suosittelee kuitenkin, että kehittäjän on mahdollisuus luoda täysin oma käyttöliittymä JavaScriptillä. (IBM Cloud Documentation 2018b.) Näin kaikki sovelluksen ulkoasusta käyttäytymiseen on täysin kehittäjän vastuulla. Kun käyttöliittymä on valmis, voidaan rakentaa yhteys Watson Assistantiin. Kuva 4 havainnollistaa, miten syöte siirtyy käyttäjältä Watsoniin ja vastausviesti käyttäjälle. Huomioitavaa on se, että chatsovelluksen ja Watson Assistantin välillä on oltava toimiva HTTPS-yhteys.



KUVA 4. Integraatio käyttöliittymästä Watsoniin (IBM Cloud Documentation 2018b)

Alustasta riippumatta pohjana Watsonilla on Assistantin työpöytänäkymä, jossa keskustelu on luotu ja mistä sitä hallinnoidaan. Sovelluksen on kuitenkin saatava yhteys työtilaan, jotta käyttäjä voi saada vastausviestinsä. Kuvassa 4 näkyy, että käyttäjän viesti siirtyy eteenpäin sovelluksen käyttöliittymästä. Tämä toteutetaan HTTPS-kutsulla, joka muodostaa yhteyden Watsonin rajapintaan. Kutsu vaatii toimiakseen vähintään kyseisen työtilan tunnisteen sekä käyttäjätunnuksen ja salasanan työtilan hallintaan. Näin käyttäjän syötteet pääsevät perille Watson Assistantiin ja vastausviestit takaisin käyttäjälle. Keskustelurakenteen muokkaaminen ja Watsonin oppiminen kuitenkin tapahtuu yksinkertaistetussa versiossa Watsonin työtilan kautta. (IBM Cloud Documentation 2018b.) Kuva 5 on kuvankaappaus työtilasta. Kuvassa näkyy käyttäjän kanssa käydyt keskustelut. Kehittäjä voi jälkikäteen muokata kyseisestä näkymästä tunnisteiden merkitystä.




KUVA 5. Käydyt keskustelut Watson työtilassa

# 3 WORDPRESS

## 3.1 WordPress sisällönhallintajärjestelmänä

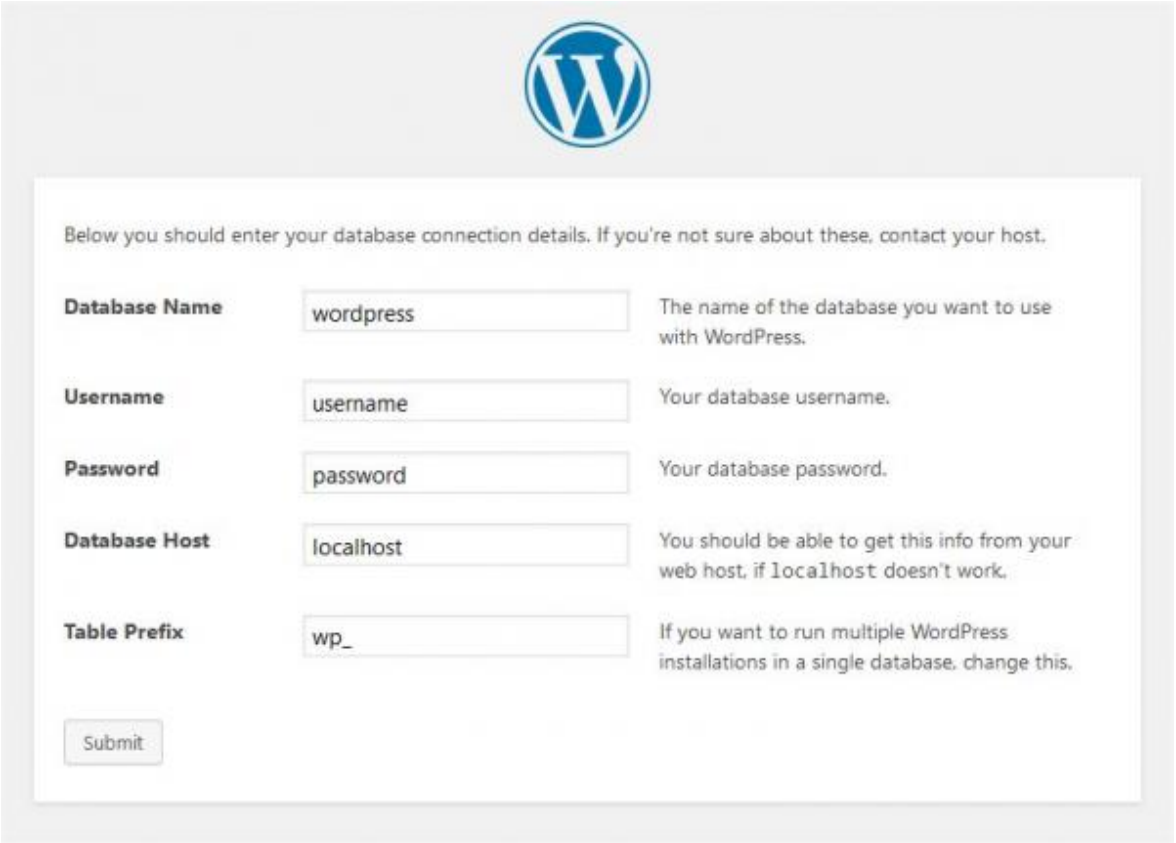
WordPress on vuonna 2003 perustettu ilmainen sekä yleisin avoimen lähdekoodin ohjelmisto, jota käytetään verkkosivujen, blogien ja sovellusten pohjana. WordPress on PHP-kieleen sekä MySQL-tietokantaan perustuva CMS, eli sisällönhallintajärjestelmä. Se on suunniteltu käyttäjäystävälliseksi ja helppokäyttöiseksi. Sen suorituskykyyn ja turvallisuuteen on kiinnitetty erityistä huomiota. 31,6 % kaikista verkkosivualustoista ympäri maailmaa on WordPress pohjaisia. (WordPress 2018.)

Sadat vapaaehtoiset kehittävät jatkuvasti WordPressiä. Kehittäjät luovat muun muassa jatkuvasti laajennuksia, lisäosia sekä teemoja, joita käyttäjät pystyvät hyödyntämään rakentaessaan mitä yksilöllisempiä verkkosivuja tai toteutuksia, eli kuka tahansa voi itse esimerkiksi luoda täysin kustomoidun teeman tai lisäosan omaan käyttöönsä sopivaksi. WordPressiin kuitenkin löytyy myös maksullisia laajennuksia, jotka ovat yksityisten kehittäjien tai yritysten luomia. (WP101, 2018.)

Sisältöä sivuille pystyy luomaan esimerkiksi kirjoittamalla sitä web-pohjaiseen editoriin WordPressin hallinnan puolelta. Sisällöntuottaja voi asentaa tarvitsemansa lisäosat sekä muokata ulkoasua ja sivupohjia käyttämällä teemoja, jotka sallivat esimerkiksi elementtien raahaamisen sivupohjaan. Tällaiset ratkaisut tekevät WordPressistä helpon sisällöntuoton kannalta ilman osaamista ohjelmoinnista.

## 3.2 WordPressin asennus

Sisällönhallintajärjestelmänä WordPress on kuuluisa helposta asentamisesta. Useat palvelutarjoajat tarjoavatkin työkaluja, joilla WordPress asennetaan käyttöön automaattisesti (WordPress Codex 2018). Kuva 6 on WordPressin automaattisesta asennuksesta selaimessa. Näin käyttäjän ei itse tarvitse luoda muun muassa tietokantaa käsin. WordPress Codex -sivusto antaa ohjeet sen asentamiseen myös manuaalisesti.




KUVA 6. WordPress asennus selaimessa (WordPress Codex 2018)

Manuaalisessa asennuksessa käyttäjä tarvitsee web-palvelimen sekä tietokannan. Ensiksi käyttäjän on ladattava ja purettava WordPress-paketti palvelimelle. WordPress käyttää MySQL-tietokantaa, joten tietokannan ja käyttäjän luominen uudelle WordPress-asennukselle on välttämätöntä. MySQL-käyttäjälle on annettava kaikki oikeudet tietokannan muokkaamiseen ja käyttämiseen. MySQL-tietoannan voi esimerkiksi luoda käyttäen komentoriviä.

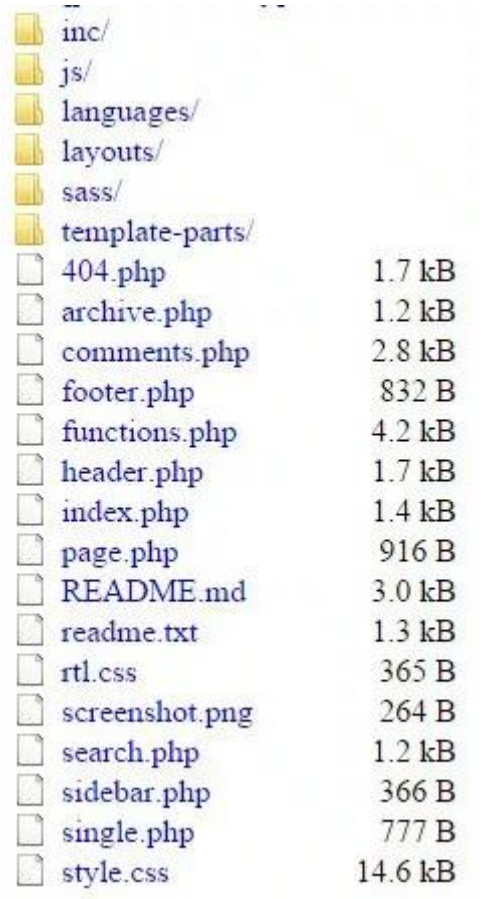
WordPress asennusta varten asennuksesta löytyvä wp-config-sample.php -tiedosto on nimettävä uudelleen wp-config.php -tiedostoksi. Kyseiseen konfigurointitiedostoon täytetään oman tietokannan tiedot, kuten käyttäjänimi sekä salasana. (WordPress Codex 2018.) Uusi WordPress asennus on käyttövalmis, kun sen tietokanta on luotu ja konfigurointi tehty.

## 3.3 WordPressin rakenne ja arkkitehtuuri

WordPress-juurihakemisto sisältää asennustiedostoja. WordPress asennuksessa .htaccess on palvelinasetustiedosto. Se hallinnoi muun muassa permalinkkeja ja uudelleenohjauksia. Permalinkillä tarkoitetaan verkkosivujen linkkejä, jotka ovat muodoltaan lyhyitä ja pysyviä. Uudelleenohjauksella taas viitataan siihen, jos sivusto on siirretty ja halutaan ohjata käyttäjä uudelle sivulle. Juurihakemisto sisältää myös wp-config.php -tiedoston, johon WordPressin asennuksen yhteydessä määriteltiin tietokannan tiedot. Wp-config.php sisältää myös muita globaaleja asetuksia. .htaccess- ja wp-config.php -tiedostoja muokatessa on oltava erityisen varovainen, sillä pieni virhe tiedostossa voi aiheuttaa palvelun kaatumisen. (WpBeginner 2016.)

WordPress asennuksen juurihakemisto sisältää myös muun muassa kansiot wp-admin, wp-content ja wp-includes. Sivuston visuaalisuuteen ja toiminnallisuuksiin sisältyvä tieto on wp-content kansion alla. Wp-content sisältää esimerkiksi themes- ja plugins-kansiot.

Uusien teemojen ja lisäosien lataaminen WordPress-asennukseen onnistuu myös purkamalla ladatut kansiot themes- ja plugins-kansioiden alle. Kaikkia uusia lisäosia ei siis tarvitse ladata WordPress-käyttöliittymän kautta. Näin esimerkiksi toimitaan, jos luotu teema tai lisäosa on itse tehty. Asennusvaiheessa juuri tehtyä ja kustomoitua teemaa tai lisäosaa ei edes ole saatavilla muilla tahoilla kuin kehittäjällä itsellään.




KUVA 7. Esimerkki WordPress teeman kansiorakenteesta (Kamal 2015)

Kehittäjän näkökulmasta WordPress kuitenkin tarjoaa vielä enemmän mahdollisuuksia. Teemojen sekä lisäosien luomisessa käytetään HTML, CSS, JavaScript ja PHP -kieliä. Kaikki toiminnallisuus yleisesti WordPressin teemojen rakenteessa tapahtuu sen ”functions.php”-tiedostossa. WordPressin teeman rakenne on täysin verrattavissa tavalliseen HTML-sivuun, mutta esimerkiksi ylä- ja alatunnisteille, sisältöalueelle ja mahdolliselle sivupalkille ovat omat PHP-tiedostonsa teeman kansion alla. Kuvassa 7 on hahmotelma siitä, mitä tiedostoja teeman alla voi olla. Teemoissa yleisesti on otettu huomioon myös haku-, arkisto- sekä sivut yleisille sisältösivuille sekä blogipostauksille.

## 3.4 WordPress lisäosat

Lisäosat eli pluginit, ovat WordPressissä laaja käsite. Lisäosa on pienempi ohjelmisto, joka toimii yhdessä verkkosivun kanssa. Lisäosa sisältää erilaisia toimintoja, jotka voidaan asentaa WordPress teemaan. Niillä muun muassa laajennetaan jo olemassa olevien

ominaisuuksien toimintoja tai luodaan kokonaan uusia. Lisäosa voi tuoda visuaalisen lisän, pelkän toiminnallisuuden tai molempien yhdistelmän WordPress pohjaiseen sivuun. Niillä helpotetaan muun muassa sisällöntuottajan työtä ilman ohjelmointia. (WpBeginner 2018.)

Lisäosat mahdollistavat esimerkiksi verkkokaupan tai kuvagallerian tekemisen, hakukoneoptimoinnin, valikon tai vain blogiotsikon värin vaihtamisen. Niillä voidaan lisätä ominaisuuksia, jotka toimivat sivun taustalla tai vain visuaaliseen puoleen liittyviä korjauksia. Huusko kirjoittaa blogitekstissään siitä, että lisäosat eivät kuitenkaan ole pakollisia, jotta verkkosivusto toimisi. Ne tuovat kuitenkin lisäarvoa sekä tuottavat käyttäjille paremman kokemuksen. (Huusko 2017.)

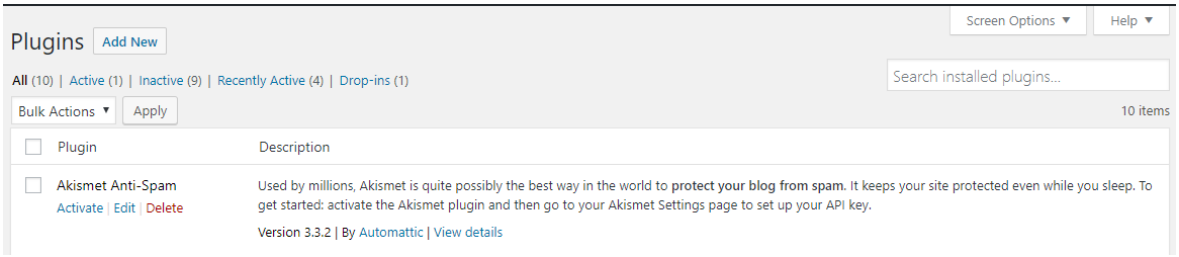
WordPressin avoin lähdekoodi luo haasteita lisäosien käyttöön. Lisäosat, joita tekemässä on ollut useampi kehittäjä, ovat yleensä myös koodiltaan laadukkaampia. Tällaisten lisäosien käyttö sujuu usein ongelmitta. Joskus lisäosat eivät kuitenkaan toimi keskenään toistensa kanssa tai teeman, joka sivustoon on asennettu. Laajennuksien päivittäminenkin voi tuoda ongelmia, jos lisäosat eivät ole yhteensopivia keskenään.

### 3.4.1 Lisäosan rakenne ja arkkitehtuuri

Lisäosa on yksinkertaisimmillaan vain yksi PHP-tiedosto, joka sisältää kommenttikentän. Lisäosan pitää sisältää vähintään kommenttirivi sen nimestä otsikko (header) tiedostossa ja esimerkiksi kuvaus siitä, mitä se tekee, sivun osoite, jossa on mahdollisesti dokumentaatiota kyseisestä lisäosasta, versio, sekä tekijän nimi. Kuvassa 8 on esimerkki siitä, mitä tietoja lisäosasta voidaan loppukäyttäjälle näyttää. Nämä tiedot näkyvät hallintanäkymässä lisäosaluettelossa kyseisen lisäosan kohdalla. Kuvassa 9 on lisäosa, joka on jokaisessa uudessa WordPress asennuksessa mukana.

```
<?php
/*
Plugin Name:  WordPress.org Plugin
Plugin URI:   https://developer.wordpress.org/plugins/the-basics/
Description:  Basic WordPress Plugin Header Comment
Version:      20160911
Author:       WordPress.org
Author URI:   https://developer.wordpress.org/
License:      GPL2
License URI:  https://www.gnu.org/licenses/gpl-2.0.html
Text Domain:  wporg
Domain Path:  /languages
*/
```

KUVA 8. Kuvankaappaus lisäosan kommenttikentästä (WordPress Documentation 2018)




KUVA 9. Askimet lisäosan hallintanäkymä (Moeez 2017)

WordPress lisäosa voi sisältää PHP-tiedostojen lisäksi JavaScript-, CSS-, kuva- sekä kielitiedostoja. Lisäosan hakemisto asennetaan kansiopolkuun wp-content/plugins/, joka sijaitsee WordPressin juurihakemiston alla. Lisäosat voivat sisältää myös tekstitiedostoja, kuten README.txt, joka sisältää tietoa, kuten kuvauksen, tunnisteet ja version.

Lisäosa voi sisältää hallintanäkymän admin- ja käyttöliittymän public-kansiot. Näihin voidaan jaotella esimerkiksi molempien omat CSS- ja JavaScript-tiedostot. Hallintanäkymän koodi sisällytetään admin-kansion alle. Tänne kirjoitettu koodi muun muassa määrittelee sen, mitä asetuksia loppukäyttäjällä on mahdollisuus tehdä asentaessaan lisäosaa sivuilleen.

### 3.4.2 Lisäosan toiminnallisuus

Lisäosan toiminnallisuus tapahtuu taas samaan tapaan kuin WordPress teemassa, eli functions.php-tiedostossa. Lisäosan functions.php nimetään kuitenkin eri tavalla; lisäosan nimen on oltava oikeanlaisessa muodossa tiedostossa. Esimerkiksi, jos lisäosan nimi on ”example”, tiedosto nimettäisiin example-functions.php. Tähän tiedostoon voidaan muun muassa määritellä, mitä tyyli- ja JavaScript-tiedostoja lisäosa käyttää, miten käyttöliittymän toteutus ilmenee WordPress sivulla sekä hallintapuolen asetuksista. Riippuen siitä, millainen lisäosa on kyseessä, hallintanäkymässä voidaan näyttää esimerkiksi verkkokauppaa luodessa yrityksen kauppatunnukset ja muut myyntiin ja maksamiseen liittyvät asetukset, tai vain pienempää lisäosaa koskevat ulkoasuun liittyvät asetukset. Hallintanäkymä ei kuitenkaan ole pakollinen.

# 4 CHATBOT CUBESCOM DIGITAL OY:LLE

## 4.1 Suunnittelu

Aihe opinnäytetyöhön tuli siitä, että yritykselle haluttiin luoda kokonaan uudet verkkosivut. Tavoite oli, että sivut olisivat tyylikkäät, näyttävät sekä päätavoitteena uusilla sivuilla olisi lisätä myyntiä. Uusille sivuille haluttiin mahdollisimman paljon erilaisia toimintoja, jolla asiakas saadaan ottamaan yhteyttä. Sivuille tehtiin paljon yhteydenottolomakkeita sekä toiminnallisuuksia, joissa asiakas pystyi tekemään muun muassa kartoituksen omista nykyisistä verkkosivuistaan. Myyntiä haluttiin lisätä myös mahdollisuudella luoda helposti erilaisia markkinointisivuja. Markkinointisivuja piti pystyä luomaan nopeasti ja yksinkertaisesti esimerkiksi tietyn kampanjan aikaan. Lomakkeiden lisäksi sivuille haluttiin käyttöön asiantunteva chat-kanava, jonka kautta asiakas voi helposti saada hakemansa tiedon.

Vanhoilla sivuilla oli käytössä perinteisempi chat, joka taas kuormitti asiakaspalvelijaa ja juuri tästä syystä ei ollut aina käytössä. Vanha chat-kanava toimi siten, että asiakaspalvelija vastasi asiakkaan kysymyksiin, kun oli paikalla ja chat-kanava aktiivisena. Edellinen chat oli tawk.to lisäosan pohjalle rakennettu toteutus. Tawk.to on chat-sovellus, joka upotetaan verkkosivuihin tai mobiilisovellukseen. Hyvät puolet tässä ratkaisussa olivat ne, että asiakas oli välittömästi yhteydessä asiantuntevaan tahoon, mutta huonoa taas se, että työajan ulkopuolella kukaan ei ollut vastaamassa asiakkaan kysymyksiin. Chat-kanavaan tuli tällöin tilalle vain yhteydenottolomake. Chatista haluttiin automatisoitu myynnin edistämisen takia.

Uudet sivut toteutettiin WordPress-sisällönhallintajärjestelmällä. Chatin pohjaksi valittiin IBM:n Watson tekoäly, sillä se sopi kyseiseen käyttötarkoitukseen parhaiten. Watson Assistantin työpöytänäkymästä keskustelua on yksinkertaista ja helppoa luoda, joten kuka tahansa pystyy kehittämään ja opettamaan tekoälyä ilman ohjelmointia. Yrityksen omalle chatbotille pystyttiin siis luomaan yksilöllinen keskusteludialogi sekä valmiit vastausviestit, jotka sopivat kontekstiin. Watson oli myös siitä syystä hyvä ratkaisu yrityksen käyttöön, sillä chat ympäristöjä luotiin useita eri kielillä. Watson Assistantissa saman käyttäjätunnuksen alle pystyy luomaan useamman työtilan, jotka eivät ole riippuvaisia toisistaan. Uusille sivuille tehtiin siis käännösmahdollisuus suomenkielestä myös ruotsiin ja englantiin. Chatbotin piti siis myös kääntyä näille kielille, jotta sen käyttötarkoitus asiakaspalvelijana olisi mahdollisimman monipuolista.

Keskusteluikkunalle saatiin graafikon luoma suunniteltu ulkonäkö. Chat sai samanlaisen teeman visuaalisuuteen kuin uudet sivut.


KUVA 10. Kuvankaappaus graafikon suunnitelmasta

## 4.2 Toteutus

Cubescomin asiakaspalveluun tarkoitettu chatbot pystytettiin WordPress -pohjaisiin verkkosivuihin. Toteutusta tehdessä kollegoilta opittua oli se, että WordPressin kanssa valmiita teemoja ja lisäosia kannattaa hyödyntää ajan säästämiseksi. Niinpä Watson chatbottiakin varten löytyi WordPressissä jo valmis lisäosa. Valmis lisäosa loi chat ikkunan sivuille, ja admin näkymästä kehittäjä pystyi itse asettamaan muun muassa Watson chatbotin työtila ID:n sekä muita käyttöliittymän asetuksia. Lisäosa toimi hyvin ja kehittäjän ei tarvinnut luoda kuin Watsonin keskustelunrakenne.

Valmis lisäosa ei kuitenkaan soveltunut kyseisen yrityksen käyttöön. Cubescomille luodun chatin oli tarkoitus olla monikielinen, ja monikielisyys toteutettiin useammalla työtilalla, eli suomen-, ruotsin ja englanninkieliset chatit olivat täysin omia kokonaisuuksiaan, joille kehittäjän oli kaikille erikseen luotava keskustelun rakenne. Näin ollen valmis lisäosa ei soveltunut monikieliseen ratkaisuun, sillä työtiloja siihen pystyi asettamaan kerrallaan vain yhden. Lisäosan koodin muokkaaminen koettiin tässä tilanteessa liian työlääksi, joten päätettiin toteuttaa itse WordPress lisäosa tätä käyttötarkoitusta varten. Kyseistä lisäosaa oli helppo markkinoida asiakkaille, sillä vastaavanlaista monikielistä toteutusta ei ole vielä ohjelmoitu. Watson chatbotille tehtiin kustomoitu käyttöliittymä, joka on koodattu PHP:lla.

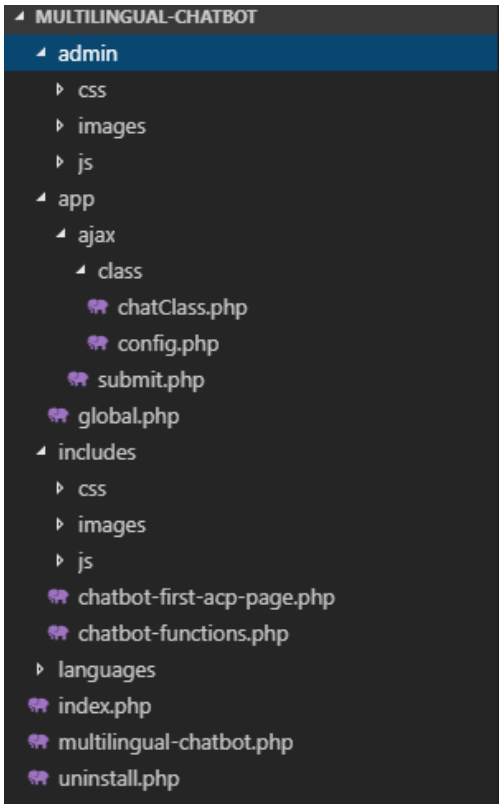
Kyseinen toteutus vaati toimiakseen omat konfigurointitiedostonsa muiden scriptien sekä käyttöliittymän koodin lisäksi. Konfiguroinnissa määriteltiin Watson Assistantista toteutettujen työtilojen tunnisteet. Jokaisella chatbotilla on oma tunnisteensa, jotta mahdollistetaan oikein toimiva HTTPS-yhteys Watson työtilaan sekä toiminnallisuus käyttöliittymätoteutukseen.

### 4.2.1 Toteutuksen rakenne

Chat lisäosa oli erillinen kansio, joka asennettiin valmiille verkkosivulle. Se voitaisiin tarvittaessa poistaa käytöstä helposti ilman, että verkkosivujen koodia tarvitsee muokata.

Chat-ikkunalle tehtiin HTML-kielellä käyttöliittymä index.php, joka sisälsi keskustelualueen, sekä yläpalkin, johon sijoitettiin painikkeet kielivalinnoille ja alapalkin syötekentän käyttäjän tekstille. Kaikki tyylittely tehtiin valmiin graafisen suunnitelman pohjalta CSS:n avulla. Chatin visuaaliseen puoleen vaikuttavat tekijät kuten CSS -tiedostot sekä Bootstrap, olivat sijoitettuna omaan kansioonsa sekä kuvat omaansa. Toteutuksessa käytettiin myös useita JavaScript -tiedostoja, jotka sijoitettiin samaan tapaan kansiorakenteeseen tyylin ja kuvien kanssa includes-kansion alle. Kuvassa 11 näkyy lisäosan tiedostot sekä kansiorakenne.

JavaScriptilla ohjelmoitiin eniten yksittäisiä toiminnallisuuksia kyseisessä toteutuksessa. Muun muassa kielen vaihtaminen ja käydyn keskustelun tallentaminen selaimen muistiin toteutettiin JavaScriptillä. PHP:llä taas toteutettiin esimerkiksi tarvittava konfiguraatio sekä Watsonin API-kutsu.




KUVA 11. Lisäosan kansiorakenne

### 4.2.2 JavaScript toiminnot

Kyseisessä toteutuksessa chatbot haluttiin kääntää kolmelle kielelle: suomelle, ruotsille ja englannille. Suomi on botin oletuskieli, jolla se automattisesti aloittaa keskustelun. Kielen valinta toteutettiin niin, että chat-ikkunassa on painikkeita, joita klikkaamalla kielen voi valita. Chatin kieltä vaihdettiin painikkeella, joka aktivoi jQueryn on.click -funktion. Scripti vaihtoi työtilan tunnisteen, jotta käyttöliittymään saatiin oikea Watson Assistant käyttöön. Kielen vaihto toteutettiin siten, että lisäosa sisälsi konfigurointitiedoston, johon PHP:llä määriteltiin käytössä olevat työtilojen tunnisteet kullekin kielelle sekä Watson Assistantin käyttäjätunnuksen ja salasanan. On.click-funktio haki aktivoituessaan oikean työtilan tunnisteen käyttöön.

Kun käyttäjä valitsee kieleksi esimerkiksi suomen, scripti vaihtaa kyseessä olevan työtilan tunnisteen suomenkieliseen chattiin. Kun sivu ladataan uudestaan, chat on automaattisesti suomenkielinen, eli käytössä on kieli 1, ja kieli vaihtuu käyttäjän toimesta muun muassa ruotsiin, eli kieli 2:een kuvan 12 mukaisesti. Jotta kielivalinta olisi käyttäjäkohtaisesti oikein, se piti myös tallentaa. Kielivalinta haluttiin tallentaa selaimen muistiin. Tässä tilanteessa vaihtoehtoina olisi esimerkiksi ollut localStorage- ja sessionStorage taulukot sekä

PHP Cookies eli evästeet. JavaScriptin sessionStorage koettiin kyseiseen tilanteeseen järkevimmäksi ratkaisuksi, sillä se ei tallenna käyttäjästä mitään ylimääräistä tietoa tai hidasta käyttökokemusta millään tavalla. SessionStorage on selaimessa sijaitseva tallennusominaisuus, joka on hiukan kevyempi kuin localStorage. SessionStorage nimittäin tyhjenee automaattisesti heti, kun käyttäjä sulkee välilehden, kun taas localStorageen voidaan tallentaa tietoja jopa sessioiden välillä. LocalStorage tyhjentyy monissa tapauksissa vasta kun käyttäjä tyhjentää sen selaimen asetuksista manuaalisesti. Tallennetut tiedot poistetaan, kun kyseinen istunto päättyy.

OnClick -funktio aktivoi siis samalla tallennusominaisuuden, ja selaimen muisti tallensi valitun kielen muistiin. Kun uusi kielivalinta tehtiin, myös selaimen muisti tyhjeni. Käyttäjä pystyi siis aloittamaan chatin uudestaan eri kielellä. Keskustelu pysyi tallessa niin kauan, kunnes uusi kielivalinta tehtiin tai selain suljettiin kokonaan.


KUVA 12. Kielivalinnantoteutus scripti

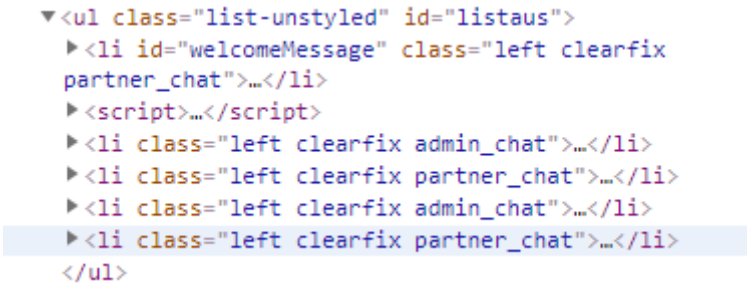
Kaikille käyttäjän klikkaustoiminnoille tehtiin oma funktionsa. Siinä määriteltiin muun muassa jQueryn onClick -funktioilla, miten chat ikkunan avaaminen tapahtuu. OpenChat.click -funktio sisälsi määritelmät siitä, miten chat ikkuna näkyy, kun painike aktivoidaan.Tällöin myös sivuston alakulmassa oleva painike chatin avaamista varten katoaa. Kun chat taas suljetaan, onClick-funktio mahdollistaa tarvittavat CSS-ominaisuudet, kuten alapainikkeen palauttamisen ja ikkunan piilottamisen. Samainen funktio kutsui myös scrollTop-funktiota. Kun chat ikkuna avattiin, keskustelu rullautui automaattisesti siten, että viimeisin viesti oli aina näkyvissä. Kuvassa 13 on esiteltynä käyttöliittymän painikkeiden JavaScript-toiminnot.

customjs.js

```
var openChat = ('#openChat, .openChat2');
var closeChat = ('#closeChat');
var messageSection = ('.message_section');
var chatButton = ('#chat-button');

$(openChat).click(function (){
    $(messageSection).css('display', 'block');
    $(chatButton).css('display', 'none');
    var div = $("#chat");
    div.scrollTop(div.prop('scrollHeight'));
});

$(closeChat).click(function(){
    $(messageSection).css('display', 'none');
    $(chatButton).css('display', 'block');
});
```

KUVA 13. Custom.js-tiedoston onClick -funktio

Tärkeä osa käyttökokemusta oli se, että käyttäjä pystyy selaamaan yrityksen sivuja ja jatkamaan aloitettua keskustelua botin kanssa myöhemmin. Eli viestit piti saada tallennettua samaan tapaan kuin chatin kielivalinta. SessionStorage koettiin parhaaksi vaihtoehdoksi tähänkin tallentamiseen, sillä se on paljon käyttäjäystävällisempi vaihtoehto kuin localStorage. Koettiin että chat-viestejä ei tarvitse tallentaa käyttäjälle pidemmäksi aikaa, kuin yhden istunnon ajaksi. Tarvittavat muuttujat keskustelurakenteesta alustettiin, kuten li- ja ul-elementit sekä taulukko, jonka sisään nämä tallentuivat.




KUVA 14. Chatkeskustelu listausnäkymänä sivun lähdekoodissa

Li- ja ul-elementit määrittelevät HTML kielessä listausnäkymän. Ul-elementin sisällä ovat kaikki yksittäiset listan rivit, eli li-elementit. Kuvassa 14 näkyy, kuinka oikeat luokat määrittyvät chatviesteille li-elementteihin. Scripti aktivoitui joka kerta, kun käyttäjä painoi chat ikkunan Lähetä-painiketta. Käyttäjän ja botin viestit tallentuivat selaimen muistiin listausmuodossa, josta ne oli helppo tulostaa käyttöliittymään JSON.parse toiminnon avulla. JSON.parse poistaa mahdolliset ylimääräiset HTML elementit viestistä, jolloin se tulostetaan oikeassa muodossa käyttäjälle.

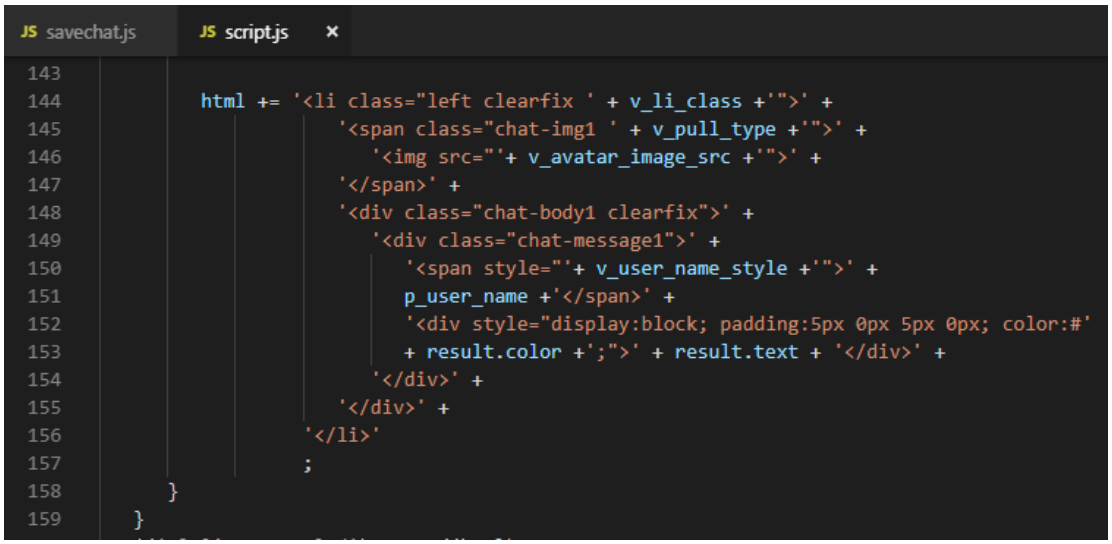
Jotta viestit saisivat oikeat ulkonäköön vaikuttavat määritelmänsä, toteutettiin tämä niin, että parittomat viestit, eli botin viestit, tulostuivat omalla li -luokallaan ja parilliset, eli käyttäjän viestit omallaan. Kuvassa 15 näkyy osa tallennusscriptiä, jossa document.addEventListener -funktio, määrittelee sen, mitä jokaisen käyttäjän painikkeen aktivoinnin aikana tapahtuu. Näin käyttäjän ja chatbotin syötteet tallentuivat saman aikaisesti sessionStorageen. SessionStorage päivittyi jokaisella Lähetä-painikkeen aktivoinnilla.

savechat.js × script.js

```
const ul = document.querySelector('ul');
const li = document.querySelector('li');
const listaus = document.getElementById('listaus');
var item = listaus.getElementsByTagName('li');
const form = document.getElementById('form');
const input = document.getElementById('chatInput');
var itemsArray = [];
const storedItems = [];

$(document).ready(function(event) {

  var arrayItems = [];

  for (var key in sessionStorage) {
    if (key.substring(0, 5) == 'items') {
      arrayItems.push(sessionStorage[key]);
    }
  }
  for (var i = 0; i < arrayItems.length; i++) {
    var items1 = JSON.parse(arrayItems[i]);

    if (i%2 == 0) {
        $('#listaus').append('<li class="admin_chat">'
        + items1 + '</li>');
    } else {
        $('#listaus').append('<li class="partner_chat">'
        + items1 + '</li>');
    }
  }
});
```

KUVA 15. Tallennusscripti

Kun koko chatbot päätettiin tehdä itse, myös käyttöliittymän rakenne oli suunniteltava käyttäjäystävälliseksi. Listausnäkymä toimi tähän toteutukseen hyvin, sillä syötteiden kuuluu tulla ennalta määritellyssä järjestyksessä näkyville käyttöliittymään. Toiminnallisuus haluttiin yksinkertaiseksi keskusteluarakenteeksi. Jokaiselle syötteelle, botin sekä käyttäjän, piti luoda omat luokkansa, sekä scripti, joka niitä loi sitä mukaan, kun keskustelua käytiin. Kuvassa 16 JavaScript koodi määrittelee yksittäisen viestin rakenteen. Jos viesti tulee botilta, CSS -luokka on eri kuin käyttäjän syötteessä. Viestin lähettäjän nimi määriteltiin samalla tapaa HTML:n span-elementillä. Chatbotin viesti sisälsi myös kuvan.




KUVA 16. Scripti joka luo oikean luokan li-elementille

### 4.2.3 PHP toiminnot

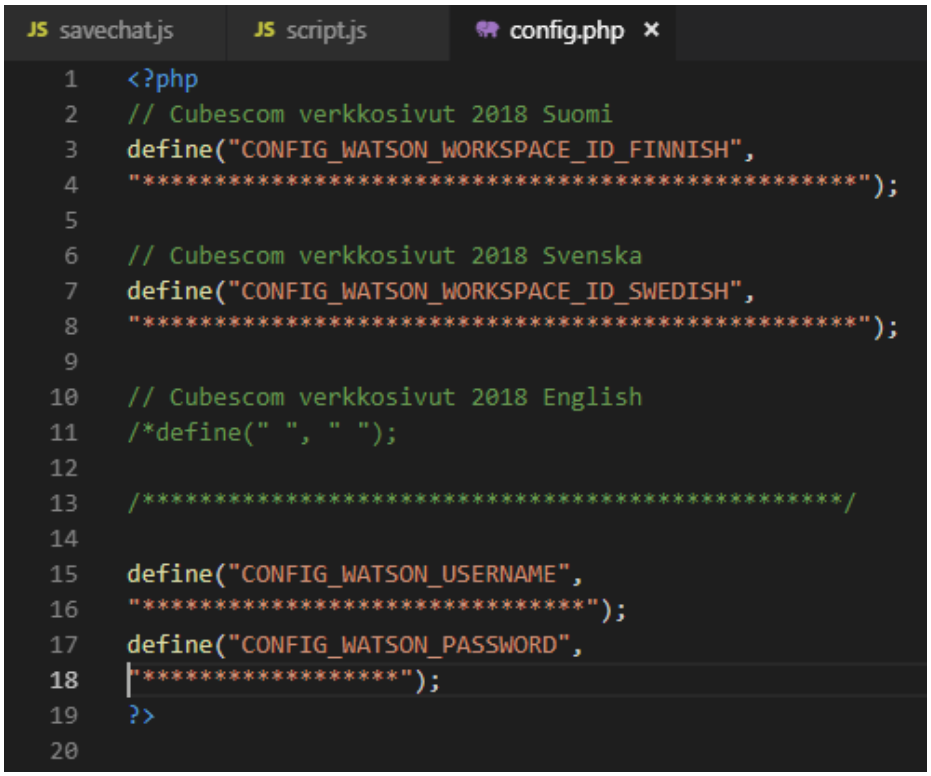
WordPress lisäosa toteutettiin pääasiassa PHP-kielellä. Lisäosan functions.php -tiedostoon lisättiin admin käyttöliittymän lisäksi toiminto siitä, miten chatin käyttöliittymä tulisi verkkosivuilla näkymään. Käyttöliittymän tulostumiselle luotiin funktio display_chatbot(), johon määriteltiin, mihin WordPress toteutuksessa lisäosa tulee näkymään. Chat-ikkuna halutiin lisätä kiinni alatunnisteeseen jo ennalta määritellyn CSS:n takia. Kuvassa 17 näkyy, kuinka chat-ikkuna nostetaan näkyviin display_chatbot()-funktiolla.

chatClass.php chatbot-functions.php ×

```php
<?php
/*
 * Add my new menu to the Admin Control Panel
 */

// Add a new top level menu link to the ACP
function chatbot_Add_My_Admin_Link()
{
    add_menu_page(
        'Multilingual Chatbot Page', // Title of the page
        'Multilingual Chatbot', // Text to show on the menu link
        'manage_options', // Capability requirement to see the link
        'multilingual-chatbot/includes/chatbot-first-acp-page.php'
        // The 'slug' - file to display when clicking the link
    );
}
add_action( 'admin_menu', 'chatbot_Add_My_Admin_Link' );

function dislpay_chatbot() {
    include_once("wp-content/plugins/multilingual-chatbot/index.php");
}
add_action('wp_footer', 'dislpay_chatbot');

?>
```

KUVA 17. Functions.php -tiedosto

PHP:llä kirjoitettiin myös konfiguraatio tiedosto, joka sisälsi tässä toteutuksessa Watson työtilojen tunnisteet sekä käyttäjänimen ja salasanan Watson Assistantiin. Konfigurointitiedostosta tehtiin yksinkertainen, sillä sen ei tarvinnut sisältää muita toimintoja tai funktioita. Tiedostosta haettiin tarvittavat tunnisteet käytössä olevaan Watsonin työtilaan. Muissa PHP-tiedostoissa määriteltiin milloinkin käytössä oleva työtila.




KUVA 18. Suomen- ja ruotsinkieliset työtilatunnisteet config.php:ssa

Valmiiseen käyttöliittymään oli saatava myös toiminnallisuus. Vastaukset käyttäjän syötteeseen tulivat Watson Assistantista, josta oli saatava myös yhteys käyttäjään. Sitä mukaan kun keskustelua käytiin, botti haki käyttäjälle sopivia vastauksia työtilastaan. Yhteys palveluun tehtiin HTTPS-kutsulla.

WordPressissä käytettiin HTTPS-kutsuna cURL nimistä kirjastoa. CURL on osa WordPressin omien rajapintakutsujen sovellusliittymää ja sen verkkoympäristö on luotu monille sovelluksille ja palveluille. PHP sisältää libcurl nimisen moduulin, joka tarjoaa cURL-toimintoa. Libcurl on avoimen lähdekoodin URL-kirjasto. (Tasker 2017.)

ChatClass.php tiedostossa on tehty lisäosan HTTPS-kutsu Watson Assistantiin. Kuvassa 19 $api_url muuttuja kuvaa käytössä olevaa työtilaa, johon käyttäjän syötteiden on tarkoitus siirtyä. Se sisältää osoitteen Watsoniin sekä työtilatunnisteen. Koodissa on määritelty, että kun käyttäjä on syöttänyt käyttöliittymään viestin, kutsu Watsonissa siirtyy eteenpäin. Watsonin HTTPS-kutsua luodessa on tärkeä huomioida koko keskustelun konteksti. ChatClass.php:n koodissa $api_request_array[’context’] määrittelee sen, että meneekö keskustelu yhtenä kokonaisuutena Watsonin työtilan hallintaan. Jos kokonaisuutta ei olisi valmiiksi määritelty, botin olisi mahdotonta jatkaa keskustelua loogisesti eteenpäin esittämällä lisäkysymyksiä käyttäjälle. Näin ei kuitenkaan aina tarvitse tehdä. Silloin kun luodun

keskustelun rakenteessa ei ole botin vastauksilla alakohtia eikä keskustelun tarvitse edetä lisäkysymyksillä, keskustelu etenee tällöin määrittelemättömässä järjestyksessä.

chatClass.php

```
74
75      $api_url = "https://gateway.watsonplatform.net/assistant/api/v1/workspaces/"
76      . $watson_workspace_id . "/message?version=2018-02-16";
77
78      if(!empty($p_input_text))
79          $api_request_array['input']['text'] = $p_input_text;
80      else
81          $api_request_array['input'] = null;
82
83      if(!empty($p_watson_context))
84          $api_request_array['context'] = $p_watson_context;
85
86      $json_api_request = json_encode($api_request_array);
87
88      $ch = curl_init();
89      curl_setopt($ch, CURLOPT_URL, $api_url);
90      curl_setopt($ch, CURLOPT_POST, 1);
91      curl_setopt($ch, CURLOPT_RETURNTRANSFER,1);
92      curl_setopt($ch, CURLOPT_USERPWD, "$watson_username:$watson_password");
93      curl_setopt($ch, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);
94      curl_setopt($ch, CURLOPT_POSTFIELDS, $json_api_request);
95      curl_setopt($ch, CURLOPT_HTTPHEADER,
96      array("Content-Type: application/json"));
97      curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, 0);
98      curl_setopt($ch, CURLOPT_SSL_VERIFYHOST, 0);
99
100     $response = curl_exec($ch);
101
102     if($errno = curl_errno($ch)) {
103       $return_msg = 'Error connecting to the Watson API. Error:(' . $errno . ') - '
104       . curl_strerror($errno);
105     }
106     else {
107       $api_response = json_decode($response);
```

KUVA 19. ChatClass.php joka sisältää HTTPS-kutsun

CURL:lla on määritelty, mitä asioita rajapintakutsua tehdessä tulee huomioida. Curl_init() alustaa kirjaston käyttövalmiiksi. Curl_setopt -metodilla asetetaan toiminnan parameterja. Esimerkiksi CURLOPT_URL määrittelee yllämainitun $api_urlin, johon kutsu lähetetään, kun taas CURLOPT_POST on yleisin http-kutsun tyyppi. Onnistunutta kutsua varten tulee käyttää POST- tai GET-menetelmää.

Jos rajapinta hyväksyy kutsun, botin vastausviesti saadaan JSON muotoisena. Jotta viesti näkyisi käyttäjälle oikein, json_decode()-funktio purkaa JSON muotoisen vastausviestin merkkijonoksi. Jos viesti ei tässä tapauksessa tulisi perille, curl_errno palauttaisi viimeisimmän virheilmoitusnumeron näkyviin koodissa määritetyn virheviestin kanssa.

## 4.3 Käyttöönotto ja testaus

Kun Cubescomin uudet verkkosivut otettiin käyttöön, täytyi botinkin olla silloin käyttövalmis. Uusille sivuille voitiin ladata valmis lisäosa, jonka jälkeen ei tarvinnut määritellä ylimääräisiä asetuksia, vaan chatbot oli jo valmis kokonaisuus. Keskustelun rakennetta on helppoa muuttaa tarvittaessa. Chatbot oppii sitä mukaan lisää, kun käyttäjät keskustelevat sen kanssa. Sisällöntuottaja tai kehittäjä voi opettaa bottia sitä mukaan, jos se ei tunnista jotain käyttäjän syötteistä. Ennen virallista käyttöönottoa, botille luotiin valmiiksi keskustelurakenteeseen tietoa yrityksestä, kuten toimiston sijainti, aukioloajat sekä muita tietoja. Lisäksi sen haluttiin palvelevan asiakkaita myös kertomalla yrityksen osaamisalueista sekä tuotteista. Botin avulla pystyy myös jättämään yhteydenottopyyntöjä ja pyytämään yhteystietoja.

Testausta toteutettiin siten, että valmista lisäosaa ladattiin usealle WordPress sivulle. Haluttiin nähdä, miten se toimii erilaisten teemojen ja muiden lisäosien kanssa. Yhdessä asennuksessa huomattiin käyttöliittymän hajoaminen siten, että teeman tyylitiedostoissa joissain elementeissä oli käytetty samoja luokkanimiä kuin chat-ikkunassa. Chatin rakenteen luokkien nimistä oli saatava yksilölliset, jotta mikään ei ylikirjoita niitä jatkossa. Myös visuaalisen puolen toimivuutta testattiin usealla eri mobiililaitteella. CSS:ssä oli määritelty Bootstrapin mukaan useita eri näytön kokoja. Botin visuaalinen ilme siis muuttui päätelaitteen näyttökoon mukaan.

Testausta piti suorittaa myös useasti toiminnallisuuden kannalta. Esimerkiksi testattiin säilyikö botin toiminnallisuus, jos käyttäjä syöttäisi erikoismerkkejä tai sisältöä, joka saattaisi vaikuttaa sen toimintaan. Tämän suhteen ei kuitenkaan Watsonpohjaisella chatbotilla ollut ongelmia, vaikka useassa vastaavassa versiossa koko keskustelukanava saattoi jäätyä vain yhdestä tietystä sanasta tai merkkijonosta. Chatbot pystyi myös suhteellisen hyvin reagoimaan käyttäjän nopeisiin ja useisiin syötteisiin. Jos se ei ehtinyt vastata käyttäjälle useaan viestiin, jotka syötettiin liian nopeasti, vastausviestit tulivat näkyviin useampi viesti peräkkäin kuitenkin oikeassa järjestyksessä. Jos käyttäjä taas esimerkiksi latasi sivun liian nopeasti heti viestin syötettyään, siihen ei chatbot ehtinyt reagoida. Tällöin myöskään käyttäjän syöttämä viesti ei tallentunut sessionStorageen. Liian nopea sivun uudelleenlataus esti siis myös sessionStoragen toiminnan.

# 5 YHTEENVETO

Opinnäyteytyön tavoite oli suunnitella ja toteuttaa tekoälypohjainen chatbot Cubescomin uusille verkkosivuille. Kyseiseen toteutukseen aikaa meni noin 3-4 kuukautta mutta kehityskohtien kanssa voi jatkaa sekä lisätä ominaisuuksia myöhemmin lisää. Muun muassa kaikki visuaalisuuteen sekä responsiivisuuteen liittyvät parannukset on hyvä ottaa huomioon ennen virallista käyttöönottoa. Myös keskustelun rakenteen ja sen etenemisen loogisuus parantaa käyttökokemusta. Opinnäytetyön valmistumishetkellä toteutus oli käyttövalmis, mutta sitä ei ollut vielä otettu testikäyttöön.

Tuotteen erityisenä ominaisuutena oli sen monikielinen mahdollisuus, jonka perusteella sitä olisi helppo markkinoida muille yrityksille. Siksi WordPress-pohjaisesta lisäosasta kehitettiin myös vastaava Drupal module-versio samalla kun WordPress lisäosaa tehtiin. Haasteena kuitenkin saattaa nousta pinnalle se, että yrityksestä riippuen, botin on osattava keskustella ja käydä läpi asioita, jotka liittyvät kyseessä olevaan yritykseen ja sen tuotteisiin. Resurssien puutteesta keskustelun rakentaminen ja botin opettaminen jätetään asiakkaan vastuulle. Tällöin täytyy asiakkaan tutustua Watson Assistantin dialogin tekemiseen sen verran, että tekoäly sekä lisäosan ominaisuudet pääsevät oikeuksiinsa.

Kyseiseen tuotteeseen, eli yrityksen markkinointia parantavaan lisäosaan voidaan kehittää toimintoja sen käyttötarkoitusta parantaen. Esimerkiksi jatkokehitystä ajatellen muun muassa chatkeskustelussa jätetyt yhteydenottopyynnöt on rajapintojen ja HTTPS-kutsujen avulla lähetettävä eteenpäin oikealle taholle. Mitä nopeammin asiakkaan yhteydenottopyyntöön tai kysymyksiin reagoidaan, sitä parempaa kuvaa ja palvelua yritys itsestään antaa.

Jos kyseisestä toteutuksesta lähdettäisiin tekemään seuraavaa versiota, keskityttäisiin todennäköisesti enemmän tekoälyn omaan oppimiseen ja toimintaan. Kehittäjän työ keskustelun luojana haluttaisiin minimoida.

Tulevaisuudessa chatbottien käyttö yleistyy yritysten keskuudessa. Tekoälypohjaisia sovelluksia rakennetaan jatkuvasti yhä enemmän, sillä tekoälyä hyödyntämällä pysytään säästämään aikaa sekä rahaa. Mitä pidemmälle teknologia kehittyy, sitä inhimmillisemmin ja paremmin chatbotin kanssa kommunikointi tulee onnistumaan.

# LÄHTEET

Ask, J., Facemire, M,. Hogan, A., Gill, M., Jacobs, I., Naparstek, L., Willsea, W., Galan, J. & Harrison, P. 2016. The State of Chatbots. Forrester [viitattu 4.8.2018]. Saatavissa: https://www.forrester.com/report/The+State+Of+Chatbots/-/E-RES136207

Brandtzaeg Bae, P. & Folstad, A. 2017. Why people use chatbots. ResearchGate [viitattu 4.8.2018]. Saatavissa: https://www.researchgate.net/publication/318776998_Why_people_use_chatbots

Burns, S. 2016. Getting Chatty with IBM Watson. Medium [viitattu 8.8.2018]. Saatavissa: https://medium.com/@snrubnomis/getting-chatty-with-ibm-watson-1075c549ee9e

Cubescom 2018. Keitä me olemme [viitattu 26.10.2018]. Saatavissa: https://www.cubescom.fi/keita-me-olemme

Huusko, L. 2017. WordPress lisäosat [viitattu 2.10.2018]. Saatavissa: https://www.laurihuusko.fi/wordpress/wordpress-lisaosat/

IBM Cloud Documentation 2017. Supported Languages [viitattu 10.8.2018]. Saatavissa: https://console.bluemix.net/docs/services/conversation/lang-support.html#supported-languages

IBM Cloud Documentation 2018a. Getting Started tutorial [viitattu 8.8.2018]. Saatavissa: https://console.bluemix.net/docs/services/conversation/getting-started.html#gettingstarted

IBM Cloud Documentation 2018b. Building a custom chat interface [viitattu 2.9.2018]. Saatavissa: https://console.bluemix.net/docs/services/virtual-agent/integrate_custom-chat.html#integrate_custom-chat

IBM 2018a. Enterprise-ready AI [viitattu 8.8.2018]. Saatavissa: https://www.ibm.com/watson/about/index.html

IBM 2018b. Producst and services [viitattu 10.8.2018]. Saatavissa: https://www.ibm.com/watson/products-services/

IBM 2018c. Tietoa IBM:stä [viitattu 8.8.2018]. Saatavissa: https://www.ibm.com/ibm/fi/fi/?lnk=fab_fifi

Kamal, M. 2015. WordPress Theme Development using Underscores (_s). CakeWP [viitattu 2.10.2018]. Saatavissa: http://cakewp.com/wordpress-tutorials/wordpress-theme-development-using-underscores-_s/

Moeez 2017. Akismet WordPress Plugin: Importance and Installation Guide. WPblog [viitattu 10.10.2018]. Saatavissa: https://www.wpblog.com/akismet-wordpress-plugin-importance-and-installation-guide/

Tasker, P. 2017. PHP and cURL: How WordPress makes HTTP requests. Delicious Brains [viitattu 20.9.2018]. Saatavissa: https://deliciousbrains.com/php-curl-how-wordpress-makes-http-requests/

Veretskaya, O. 2017. What is a Chatbot and How to Use It for Your Business. Anadea [viitattu 4.8.2018]. Saatavissa: https://anadea.info/blog/what-is-a-chatbot-and-how-to-use-it-for-business

WordPress 2018. About [viitattu 2.10.2018]. Saatavissa: https://wordpress.org/about/

WordPress Codex 2018. Installing WordPress [viitattu 25.10.2018]. Saatavissa: https://codex.wordpress.org/Installing_WordPress

WordPress Documentation 2018. Header Requirements. Plugin Handbook [viitattu 2.10.2018]. Saatavissa: https://developer.wordpress.org/plugins/the-basics/header-requirements/

WpBeginner 2016. Beginner's Guide to WordPress File and Directory Structure [viitattu 25.10.2018]. Saatavissa: https://www.wpbeginner.com/beginners-guide/beginners-guide-to-wordpress-file-and-directory-structure/

WpBeginner 2018. What is: Plugin [viitattu 2.10.2018]. Saatavissa: https://www.wpbeginner.com/glossary/plugin/

WP101, 2018. What is WordPress? [viitattu 2.10.2018]. Saatavissa: https://www.wp101.com/tutorial/what-is-wordpress/