Chatbotin suunnittelu IT-tuen käyttöön

Ossi Kallunki

2021 Laurea

Laurea-ammattikorkeakoulu

Chatbotin suunnittelu IT-tuen käyttöön

Ossi Kallunki

Tietojenkäsittely

Opinnäytetyö

Toukokuu, 2021

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutusohjelma
Tradenomi (AMK)

Ossi Kallunki

Chatbotin suunnittelu IT-tuen käyttöön

Vuosi 2021 Sivumäärä 22

Tämän kehittämistyön tavoitteena oli kehittää chatbotin prototyyppi Microsoft Bot Framework-kehitysympäristössä. Prototyypin tarkoituksena oli todistaa chatbotin ja sen ominaisuuksien toimivuus käytetyillä menetelmillä. Opinnäytetyön tulosten pohjalta tullaan kehittämään Suomen Lähetysseura ry:n IT-tuen käyttöön chatbot, jolla voidaan suorittaa nopeasti rutiinitehtäviä, kuten käyttäjätietojen tarkistusta.

Työn tietoperustassa käsitellään chatbottien ominaisuuksia ja niiden käyttöä yrityksissä, sekä Microsoft Bot Frameworkilla toteutettavia chatbotteja. Opinnäytetyön kehitysosuudessa hyödynnetään ohjelmistokehitysmallia, jonka avulla suunnitellaan ja kehitetään iteratiivisesti chatbotin prototyyppi, joka kykenee välittämään viestejä, sekä suorittamaan PowerShell-skriptejä. Kehittämistyön tuloksena on prototyyppi, joka vastaa asetettuihin vaatimuksiin ja toimii kehitysympäristössä.

Asiasanat: Chatbot, Microsoft Bot Framework, PowerShell, Ohjelmistokehitys, Prototyyppi

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology
Bachelor's degree

Ossi Kallunki

Design of a chatbot for technical support

Year 2021 Pages 22

The objective of this thesis project was to develop a prototype of a chatbot using the Microsoft Bot Framework. The purpose of the prototype was to prove the feasibility of developing a chatbot with the used methods. Based on the results of the thesis, a chatbot will be developed for the use of the IT department at Felm (Finnish Evangelical Lutheran Mission). The chatbot will allow the rapid execution of routine IT tasks, such as searching for user information.

The theoretical part of the thesis report covers chatbots, their use in companies and chatbot development with Microsoft Bot Framework. In the developmental part of the thesis report, a software prototyping model was used to develop a prototype of the chatbot, which can transmit messages and execute PowerShell scripts. The results of the thesis show that the prototype meets the set requirements and functions well in a development environment.

Keywords: Chatbot, Microsoft Bot Framework, PowerShell, Software Development, Prototype

Sisällys

1 Johdanto

Tämän opinnäytetyön tavoitteena on kuvata chatbotin prototyypin kehittäminen Microsoft Bot Framework-ympäristössä. Kehittämistyössä selvitetään chatbottien toimintaperiaatteita sekä kehitetään chatbotin prototyyppi. Kehitettävän prototyypin pohjalta luotava chatbot tulee organisaation IT-tuen käyttöön rutiinitehtävien suorittamiseen. Se suorittaa käskystä ennalta määriteltyjä tehtäväsarjoja ja palauttaa vastauksen keskusteluikkunaan. Chatbotin tyypillisiä tehtäviä voivat olla esimerkiksi käyttäjätietojen muokkaaminen tai uuden käyttäjän luominen Active Directory-käyttäjätietokantaan.

Kehittämistyön kohteena oleva botti tullaan kehittämään tämän työn havaintojen pohjalta Suomen Lähetysseura ry:n käyttöön. Lähetysseuran IT-tuki palvelee noin 200:aa työntekijää, joista osa on ulkomailla. IT-tuen henkilökunnassa on pääsääntöisesti aina harjoittelijoita, joiden työhön perehdyttämistä chatbot nopeuttaa, sillä se ei vaadi erityistuntemusta erilaisista järjestelmistä, joihin botti on yhdistetty.

Opinnäytetyötä ennen on tehty esiselvitys chatbot-ohjelman teknisistä vaatimuksista, sekä sen soveltuvuudesta Microsoft Teams-alustalle, jolla valmista bottia tullaan käyttämään. Kehitettävä chatbot ei ole täysin uudenlainen sovellus, mutta Teams-alustalla ja Powershell-liitynnällä se vaatii myös uutta kehitystyötä.

Työssä käsiteltävä chatbot voi nopeuttaa huomattavasti rutiinitehtävien suorittamista, kuten käyttäjätietojen tarkistamista tai niiden muuttamista. Botti voidaan yhdistää useimpiin IT-tuen tietojärjestelmiin, kuten tikettijärjestelmään ja AD-käyttäjänhallintaan. Ilman bottia työntekijä joutuu käyttämään useita eri tietojärjestelmiä tiedon hakemiseen ja muokkaamiseen tai vaihtoehtoisesti käyttämään osaamista vaativia komentorivityökaluja. Yhdessä ikkunassa toimivalle chatbotille voi antaa selkokielisiä käskyjä, eikä käyttäjä joudu siirtymään järjestelmien välillä.

2 Työn lähtökohdat

Opinnäytetyö sai alkunsa harjoittelujaksollani Suomen Lähetysseurassa. Työssäni organisaation IT-tuessa käytin olemassa olevaa versiota chatbotista, jonka uutta versiota tässä kehittämistyössä käsitellään. Harjoittelujaksoni aikana kävi ilmi, että ohjelman vanha alusta sulkeutuu kesällä 2021 ja ohjelma tulisi siirtää uudelle alustalle. Käytettyäni ohjelmaa noin 5 kuukautta päivittäin, tunnen sen toiminnan hyvin, joten pidin sen kehittämistä sopivana opinnäytetyöaiheena.

Chatbotin olemassa oleva versio on toiminut useamman vuoden Skype For Business-viestintäalustalla. Se on ohjelmoitu pääasiassa PowerShell-komentosarjakielellä. Uusi ohjelma tulee toimimaan Microsoft Teams-viestintäalustalla, eikä sen kehitysympäristö tue suoraan PowerShell-pohjaisia botteja. Näin ollen tuli tarpeen selvittää, miten uudessa ympäristössä voidaan rakentaa chatbot, joka voi suorittaa PowerShell-sovelluksia. Koska aiemman botin toiminallisuudet on jo aiemmin määritelty, lisäämällä uuteen bottiin vain PowerShell-liitännän, säästytään ylimääräiseltä ohjelmoinnilta.

PowerShell-liitäntää tukee myös se, että Microsoft on luonut erittäin hyvät Active Directory- ja Azure-moduulit PowerShellille. Suuri osa botin liikenteestä tapahtuu näihin kahteen kohteeseen, joten yhteyden ottaminen suoraan Python-ohjelmasta ei ole mielekästä.

Microsoft Teams valikoitui botin toiminta-alustaksi, sillä se on IT-tuen käytössä paljon myös muissa työtehtävissä. Se on käytössä laajalti organisaation sisäisessä viestinnässä, joten rakentamalla chatbot tälle alustalle voidaan keventää IT-tuen työtaakkaa kommunikoidessa asiakkaiden kanssa.

2.1 Kehittämistavoite

Opinnäytetyön tavoitteena oli tuottaa kehittämistyö, joka kuvaa chatbotin prototyypin rakentamisen vaiheet ja kehitysympäristön. Kehittämistyö pyrki vastaamaan seuraaviin kysymyksiin:

- Miten chatbot ja sen kehitysympäristö kehitetään?
- Miten chatbot voi suorittaa PowerShell-komentoja?

Työssä kehitettävää prototyyppiä arvioin luvussa 2.3 esiteltävän kehitysmallin mukaisesti kehityksen aikana. Prototyypin kehityksen keskeisiä arviointikysymyksiä olivat:

- Tekeekö prototyyppi tehokkaasti sille tarkoitetut asiat?
- Onko ohjelmoinnissa käytetty tehokasta ja ymmärrettävää ohjelmointitapaa?
- Skaalautuuko prototyyppi, eli voidaanko siitä kehittää valmis ohjelma?

2.2 Rajaus

Opinnäytetyön aihealue on rajattu chatbotteihin yrityksen sisäisessä käytössä, Microsoft Bot Frameworkiin ja kehitysosaan. Työ on kehittämistyö, jossa ei tuoteta valmista bottia, vaan tarkastellaan botin ja kehittämisympäristön toimintaa yksinkertaisen prototyypin avulla. Aihealue rajautuu kehitettävän botin kehityksen mahdollistaviin tekijöihin.

Kehitystyössä keskitytään toimivan kehitysympäristön toteuttamiseen botille, sekä yksinkertaisen prototyypin ohjelmointiin. Prototyyppi sisältää ainoastaan suunnitelman mukaiset ominaisuudet.

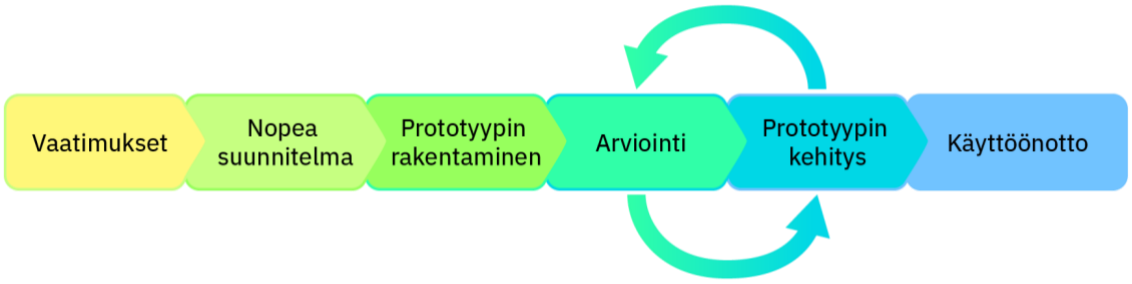
2.3 Kehittämismenetelmät

Kehittämistyö toteutettiin soveltuvuusselvitys-menetelmällä (proof of concept), joka tarkoittaa jonkin idean tai menetelmän toimivuuden todentamista. Kehittämistyössä soveltuvuusselvitys suoritetaan kehittämällä yksinkertainen prototyyppi botista, jonka toimivuus todentaa menetelmän soveltuvuuden.

Ohjelmistokehityksessä hyödynsin prototyyppimallia, jossa kehitetään prototyyppi, jota testataan, arvioidaan ja uudelleen kehitetään, kunnes saavutetaan haluttu lopputulos. Onnistuneen prototyypin pohjalta voidaan kehittää lopullinen ohjelmisto. Prototyyppimallissa on yleensä läsnä sen käyttäjät, mutta tässä tapauksessa toimin itse sekä kehittäjän, että käyttäjän roolissa. Hyödyin erityisesti mallin antamasta joustavuudesta, sillä se mahdollistaa erilaisten ominaisuuksien kokeilun helposti. (Guru99 2021.)

Guru99:n luomaan kehitysmalliin perustuva käyttämäni malli koostuu seuraavista osista (kuvio 1):

1. Vaatimukset: prototyypin kehitykselle asetetaan vaatimukset; mitä ominaisuuksia ohjelmassa tulee olla
2. Nopea suunnitelma: luodaan yksinkertainen kuva tai malli siitä, miltä ohjelma tulee näyttämään
3. Prototyypin rakentaminen: kehitetään varsinainen prototyyppi edellisen vaiheen pohjalta
4. Arviointi: käyttäjä arvioi kehitettyä prototyyppiä
5. Prototyypin kehitys: jos käyttäjä ei ollut tyytyväinen prototyyppiin, kehitetään sitä eteenpäin ja palataan edelliseen vaiheeseen. Osio jatkuu, kunnes saavutetaan hyväksyttävä lopputulos.
6. Käyttöönotto: Prototyypistä valmistunutta ohjelmaa testataan ja otetaan se käyttöön



Kuvio 1: Prototyyppimallin vaiheet

2.4 Keskeiset käsitteet

Botti	Tietokoneohjelma, joka suorittaa automatisoituja tehtäviä itsenäisesti verkossa
Chatbot	Ohjelma, joka pystyy käymään interaktiivista keskustelua käyttäjänsä kanssa
Sovelluskehys	Ohjelmistokehitystä nopeuttava runko, jonka päälle ohjelma rakennetaan

3 Chatbotit

Chatbot eli keskustelurobotti on ohjelma, joka kykenee käymään interaktiivista keskustelua sen käyttäjän kanssa. Chatbotteja käytetään nykyään paljon asiakaspalvelussa verkkosivuilla ja mobiilisovelluksissa. Ne ovat saatavilla myös silloin kun ihmisasiakaspalvelijat eivät ole käytettävissä. (Accenture 2016, 3.)

Nykyään tekoälyä hyödyntävät chatbotit voivat olla hyvin kehittyneitä, kuten Amazon Alexa, joka ymmärtää puhetta ja osaa vastata myös kokonaisin lausein. Chatbotit ovat erinomaisia tehtäviin, joissa haetaan tarkoin rajattua tietoja eri järjestelmistä toistuvasti. Hyvän chatbotin laajuus on määritelty tarkoin. Sen käyttäjän tulee tietää, mitä botti kykenee tekemään ja sillä tulee olla tarkoin määritelty ongelma, jonka se ratkaisee. (Fichter & Wisniewski 2017, 56–58.)

Opinnäytetyössäni tarkasteltava ohjelma on melko tyypillinen chatbot. Käyttäjä antaa sille ennalta määritettyjä käskyjä keskusteluikkunassa, joihin botti hakee vastauksen. Tällaista ohjelmaa kutsutaan sääntöpohjaiseksi chatbotiksi, joka tarkoittaa sitä, ettei ohjelma osaa tulkita muita, kuin sille määriteltyjä käskyjä. Esimerkiksi asiakaspalvelussa käytettävä chatbot voi olla itseoppiva, eli se oppii ymmärtämään uusia käskyjä koneoppimisen avulla. (Thorat & Jadhav 2020, 1.)

3.1 Chatbot organisaation sisäisessä käytössä

Chatbotteja on alettu viime vuosina käyttämään asiakaspalvelun lisäksi myös organisaatioiden sisäisesti. Niitä on käytetty antamaan ajantasaisia vastauksia työntekijöille esimerkiksi omaan ajankäyttöön tai organisaation käytäntöihin liittyen (Hash 2019).

Chatbot voi olla tehokas tiedonhakuväline organisaation sisällä, sillä tyypillisesti organisaatiossa käytetään tiedonhakuun intranettiä. Vaikka chatbot ei voi täysin korvata intranettiä, se voi nopeuttaa tiedon saantia huomattavasti. Jos chatbotin käyttöliittymä

sijaitsee esimerkiksi Microsoft Teamsissa, työntekijät pääsevät käsiksi tietoon nopeammin, sillä muukin kommunikaatio tapahtuu samassa ohjelmassa. Toinen botin etu on se, että chatbot vastaa kysymykseen suoraan, eikä työntekijä joudu etsimään tietoa mahdollisesti monimutkaisesta intranetistä. Esimerkiksi tietyn dokumentin hakeminen voi olla huomattavasti nopeampaa chatbotilla kuin intranetistä. (Adler 2019)

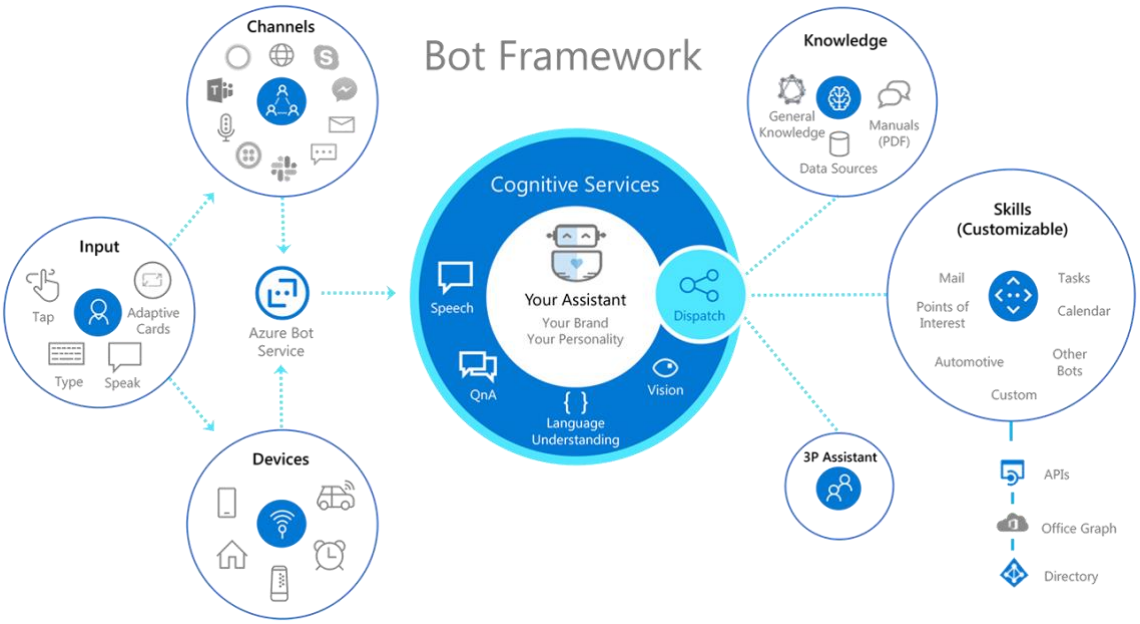
Yhdysvaltalaisessa Lemonade-vakuutusyrityksessä on kehitelty idea autonomisesta organisaatiosta. Sen tarkoituksena on välttää henkilökunnan rekrytoimista organisaation kasvaessa ja sen sijaan ratkaista skaalautumisongelmat luomalla älykkäitä ja autonomisia työkaluja. Lemonade on kehittänyt chatbotin organisaation keskiöön, joka muun muassa jakaa työtehtäviä, suorittaa ohjelmien testausta ja vastaa työntekijöiden kysymyksiin. Lemonaden botti kykenee myös luomaan ja löytämään dokumentteja sekä auttamaan IT-tuen tehtävissä. (Wininger 2018.)

Nichol (2018) odottaa, että lähivuosina botit kehittyvät henkilökohtaisiksi avustajiksi, jotka osaavat tulkita kontekseja ja oppivat tuntemaan käyttäjänsä. Hänen mukaansa organisaatiot voivat myöhemmin muuttua pitkälti autonomisiksi, jossa merkittävä osa toiminnoista on bottien suorittamaa.

3.2 Microsoft Bot Framework

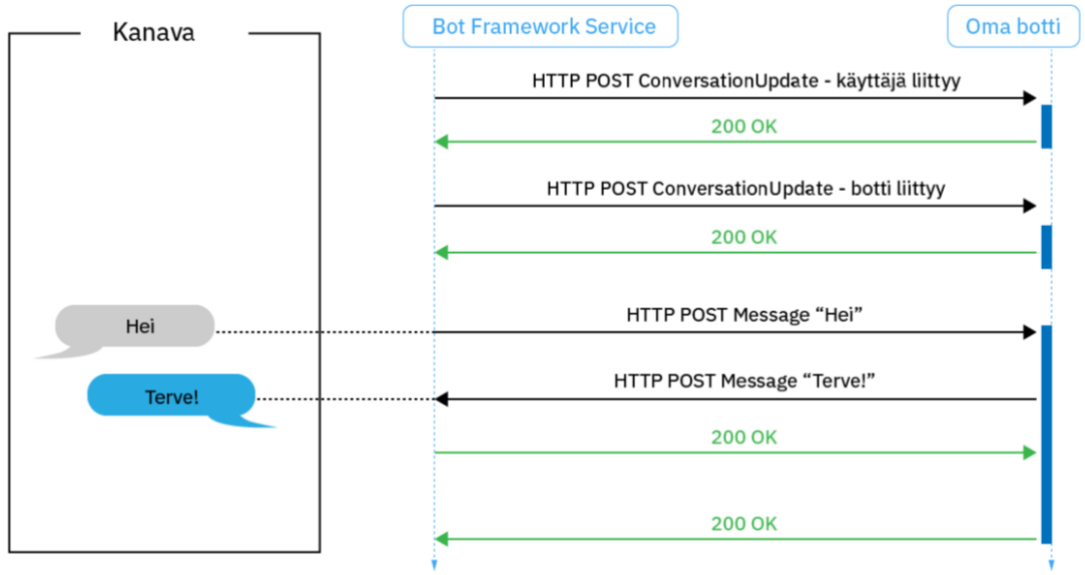
Microsoft Bot Framework on sovelluskehys chatbottien ja ohjelmistorobottien kehittämiseen. Siihen sisältyy sarja työkaluja, joilla voi kehittää chatbotteja, jotka ymmärtävät luonnollista kieltä. Sovelluskehys mahdollistaa kerran ohjelmoidun robotin käyttämisen erilaisilla alustoilla, kuten Microsoft Teamsissa tai Facebookin Messengerissä. Bot Frameworkiin sisältyy SDK (Software Development Kit), eli ohjelmistokehityspaketti.

Bot Framework on hyvin laaja ja monikäyttöinen sovelluskehys. Sillä kehitetyt botit voivat vastaanottaa tietoa monin eri tavoin, esimerkiksi puhumalla, kirjoittamalla tai klikkaamalla näytöltä botin tarjoamia vaihtoehtoja. Botit pystyvät myös suorittamaan luonnollisen kielen käsittelyä (natural language processing), jolla botti päättelee lauseen tarkoituksen monimuotoisista kysymyksistä (Microsoft 2021a).



Kuvio 2: Microsoft Bot Frameworkin osat ja ominaisuudet (Microsoft 2019)

Microsoft Bot Framework Service välittää informaatiota chatbotin ja siihen yhdistetyn chat-sovelluksen välillä, joka voi olla esimerkiksi Microsoft Teams. Botti käyttää aktiivisuusobjekteja kommunikoidessaan Bot Frameworkin kanssa. Tällaisia voivat olla esimerkiksi kuviossa 3 esiintyvät keskustelun päivitysaktiviteetit, joita käytetään, kun osapuoli liittyy keskusteluun. Vastaavasti viestiaktiviteetti sisältää keskusteluinformaation ja metadatan osapuolten välillä. (Microsoft 2020a.)

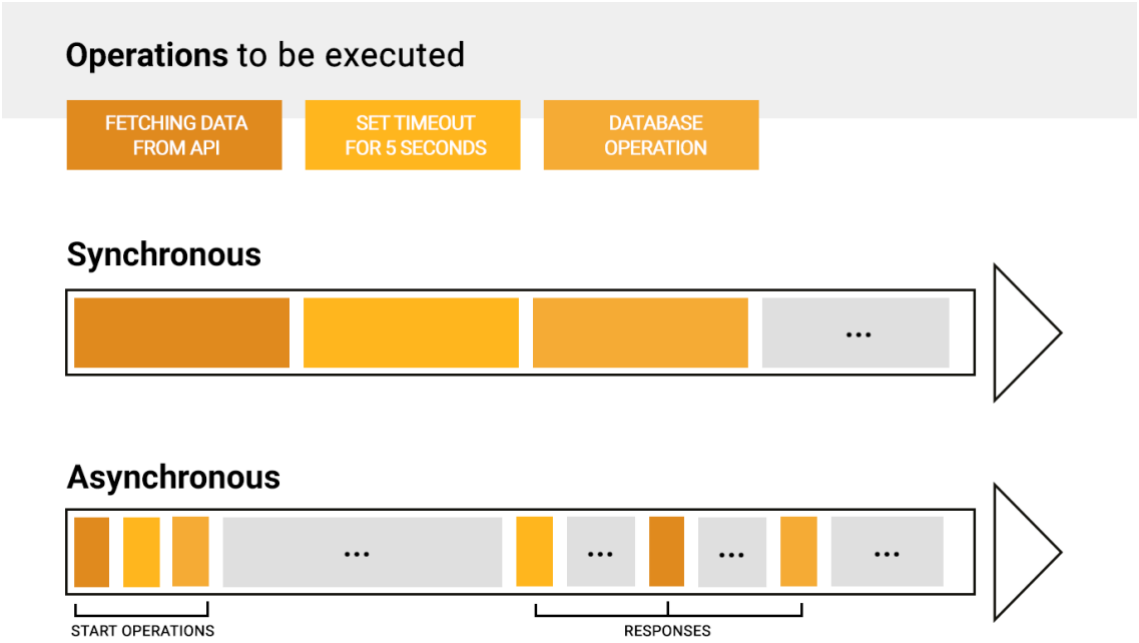


Kuvio 3: Bot Frameworkin viestinvälitys (Microsoft 2020a)

Chatbot kuuntelee keskustelukanavalla tapahtuvia aktiviteetteja. Näitä voivat olla esimerkiksi henkilön liittyminen keskusteluun, vastaanotettu viesti, reaktio viestiin tai muu keskustelussa tapahtuva muutos. (Microsoft 2017.)

3.3 Asynkroninen ohjelmointi

Microsoft Bot Framework käyttää boteissaan asynkronista ohjelmointitapaa. Se tarkoittaa ohjelman kykyä suorittaa sen osia erillään ohjelman pääsäikeestä. Asynkroninen ohjelmointi nopeuttaa ohjelmaa, sillä useamman osan suorittaminen tapahtuu samanaikaisesti (kuvio 4). Voidaan esimerkiksi suorittaa bottien käyttämiä HTTP-pyyntöjä ja samalla ajaa paikallisia tehtäviä. Näin ohjelma ei pysähdy hitaamman ulkoisen resurssin hakemisen ajaksi. (Velatio Technologies 2018.)



Kuvio 4: Synkronisen ja asynkronisen ohjelman erovaisuudet (Münster 2019)

Python-ohjelmointikielessä samanaikaisuus voidaan toteuttaa asyncio-moduulilla, jota myös Bot Framework käyttää. Bot Frameworkille tyypillisiä ovat niin kutsutut coroutinet, jotka ovat funktioita, jotka voivat pysäyttää ja jatkaa itsensä suoritusta useita kertoja. Normaalin funktion suoritus aloitetaan tietystä kohdasta ja lopetetaan tiettyyn kohtaan. (Ogbuji 2018.)

4 Chatbotin prototyypin kehittäminen

Tässä luvussa käsitellään chatbotin toteutusta Python-ohjelmointikielellä Microsoft Bot Frameworkissa. Kehittämisprosessia tarkastellaan luvussa 2.3 esitellyn prototyyppimallin avulla.

4.1 Vaatimukset

Kehitettävän chatbotin olennaisin ominaisuus on sen kyky tunnistaa käskyjä, suorittaa niille määritelty PowerShell-skripti ja palauttaa sen tulos käyttäjälle. Botin tulee myös kyetä toimimaan Microsoft Teams-alustalla. Näiden vaatimusten perusteella opinnäytetyön esiselvityksessä päädyttiin valitsemaan botin toteutusympäristöksi Microsoft Bot Framework.

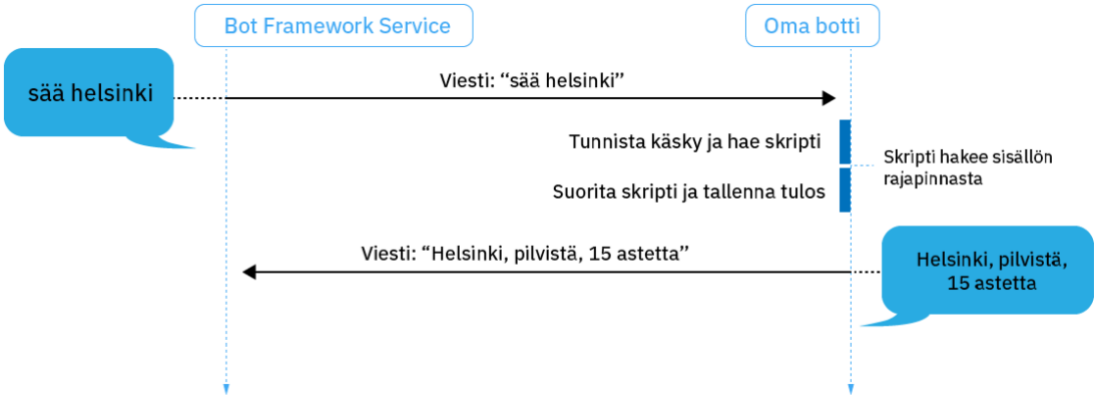
Chatbotin aiempi versio rakentuu PowerShell-skriptien ympärille, jotka hakevat tietoa eri järjestelmistä, kuten Microsoft Azuresta. Kehittämällä uuteen bottiin PowerShell-ominaisuuden, voidaan suuri osa vanhan botin toiminnalisuuksista siirtää suoraan uuteen bottiin.

Chatbotin kehittämistä tarkastellaan siitä näkökulmasta, että se toteutettaisiin Python-ohjelmointikielellä. Tätä valintaa tukee Pythonin helppo käyttöönotto, ohjelmoinnin nopeus ja sen sisäänrakennettu tuki PowerShell-skriptien suorittamiseen. Botin voi toteuttaa lähes samanlaisin vaihein myös .NET-ympäristöön tai Javascriptilla.

4.2 Nopea suunnitelma

Tässä vaiheessa tein yksinkertaisen kuvion, joka havainnollistaa prototyypin toimintaperiaatetta. Toimintaperiaatteen visualisointi varmistaa, että prototyypin kehitysvaiheessa on hyvä ymmärrys botin toiminnasta. Prototyypin ohjelmointi on helpompaa, kun botin suorittamat vaiheet on jaettu sopiviin osiin suunnitteluvaiheessa. Botti tulee vastaamaan keskusteluikkunassa esitettyihin käskyihin kuvion 5 esittämällä tavalla:

1. Käyttäjä lähettää käskyn keskusteluikkunassa
2. Botti tunnistaa käskyn ja hakee sitä vastaavan funktion ja skriptin
3. Botti suorittaa PowerShell-skriptin, palauttaa tuloksen ohjelmaan ja tallentaa sen muuttujaan tai taulukkoon
4. Botti lähettää muuttujan sisältämän vastauksen takaisin käyttäjälle keskusteluikkunaan



Kuvio 5: Botin tiedonhaun periaate

Prototyypille tuli kehittää myös PowerShell-skripti, jolla voidaan testata funktion toiminnallisuutta. Päädyin testaamaan API-pyynnön toteuttamista skriptissä ja tuloksen palauttamista pääohjelmaan. Skriptin valinnalla ei ollut erityistä merkitystä, sillä useimmat niistä toimivat samalla tavalla, eli palauttavat yhden tai useamman tiedon pääohjelmaan.

4.3 Prototyypin rakentaminen

4.3.1 Kehitysympäristö

Bot Frameworkin kehitysympäristö vaatii, että Python-ohjelmointikielen versio 3.6, 3.7 tai 3.8 on asennettu työasemalle. Toinen kehitysympäristön vaatima osa on Bot Framework Emulator, joka on ladattavissa Microsoftin Github-sivuilta. Bot Framework Emulator mahdollistaa bottien interaktiivisen testaamisen omalla laitteella ilman palvelinympäristön erillistä kehittämistä. (Microsoft 2020b.)

Kehittäjän tulee asentaa pip-paketinhallintajärjestelmällä Bot Frameworkin vaatimat moduulit, joita ovat botbuilder-core, asyncio, aiohttp ja cookiecutter. Botin kehitysympäristön tiedostot luodaan mallinteesta komentorivillä cookiecutter-paketin avulla. Tämä skripti kysyy kehittäjältä botin nimen ja luo samannimisen kansion, johon botin tiedostot sijoitetaan. Microsoft tarjoaa useita bottimallinteita, joiden pohjalta kehittämisen aloittaminen on nopeampaa. Prototyypissä hyödynsin echo-mallinetta, joka sisältää esimerkkikoodia viestien lähettämiseen ja vastaanottamiseen. (Microsoft 2020b.)

Botin sisältävässä kansiossa olennaisin tiedosto on bot.py, johon ohjelmoidaan botin toiminnallisuus. Toinen olennainen tiedosto on app.py, josta botti käynnistetään. Botin käynnistyttyä sen toimintaa voidaan testata Bot Framework Emulatorilla.

4.3.2 Botin rakentaminen

Botin pääohjelman tiedosto koostuu MyBot-luokasta, jossa tapahtuu viestinvälitys chat-ikkunaan, sekä kehittämästäni psRun-funktiosta, joka käsittää PowerShell-tiedoston suorittamisen ja sen tulosteen käsittelyn. PowerShell-ominaisuuden havainnollistamiseksi prototyypissä suoritetaan säätietojen haku OpenWeather Web API:sta eli verkko-ohjelmointirajapinnasta.

Botin käyttäjä syöttää prototyypin keskustelukanavassa käskyn, joka on muodossa ”sää [kaupungin nimi]”, jonka botti tallentaa ja syöttää psRun-funktion lähtötiedoksi. Funktio avaa tiedoston lähtötiedon kanssa. PowerShell-tiedostossa otetaan yhteys säätietorajapintaan. Se palauttaa JSON-muotoisen tiedon, joka käsitellään PowerShell-skriptissä ja palautetaan botin Python-ohjelmaan.

```
18  class MyBot(ActivityHandler):
19
20      #Seuraa saapuvia viestejä
21      async def on_message_activity(self, turn_context: TurnContext):
22
23          #Jos viesti on esim. "sää helsinki"
24          if(turn_context.activity.text[:3] == "sää"):
25              #Jaa pyyntö osiin ja tallenna kaupunkitieto
26              data = turn_context.activity.text.split(" ")
27              kaupunki = data[1]
28              #Kutsu powershell-funktio
29              psOutput = psRun(kaupunki)
30              #Palautetaan chat-ikkunaan powershellin tulos
31              await turn_context.send_activity(psOutput)
```

Kuvio 6: Botin pääluokka

Botin pääluokassa (kuvio 6) on asynkroninen funktio, joka kuuntelee keskustelukanavan uusia viestejä ”on_message_activity” -menetelmällä. Se on yksi monista aktiviteeteista, joiden muutoksia botti kykenee kuuntelemaan. Tämän alle on määritelty if-lauseke, joka tarkistaa, vastaako viestin sisältö haluttua ”sää”-käskyä. Jos lauseke toteutuu, ohjelma leikkaa käskyn välilyöntiä edeltävän osan pois split-metodilla ja jättää jäljelle käskyn kaupunkitiedon. Tätä kaupunkitietoa käytetään psRun-funktion lähtötietona, joka hakee säätiedot. Funktion tulos tallennetaan psOutput-muuttujaan, joka palautetaan keskustelukanavaan ”send_activity”-menetelmällä.

```python
from botbuilder.core import ActivityHandler, TurnContext
from botbuilder.schema import ChannelAccount
import subprocess, sys, os

# Hakee Powershell-skriptin, ajaa sen ja palauttaa sen tulosteen
def psRun(location):
    p = subprocess.Popen(["powershell.exe",
        os.path.join(sys.path[0], 'powershell-skripti.ps1') + " " + str(location)],

        shell=True,
        stdout=subprocess.PIPE,
        text=True)

    output = p.communicate()[0].strip()

    return output
```

Kuvio 7: Botin funktio, joka suorittaa PowerShell-skriptin

Kehitin bottiin funktion ”psRun”, jota kutsumalla voidaan suorittaa sille määritelty PowerShell-tiedosto (kuvio 7). Tiedoston suorittamiseen käytetään Pythonin subprocess-moduulia, jolla voidaan suorittaa ulkopuolisia ohjelmia ja palauttaa niiden tuloste ohjelmaan. Moduulille määritellään ohjelma, jota käytetään sekä tiedoston sijainti, joka halutaan suorittaa. Määrittelyn loppuun lisätään funktion lähtötiedoksi annettu kaupunkitieto, joka on myös PowerShell-tiedoston argumentti. Kun tiedosto on suoritettu, skriptin tuloste palautetaan ”output”-muuttujaan ja takaisin ohjelman pääluokkaan.

```powershell
$requestlocation = $args[0]

$params = @{
    Uri         = "http://api.openweathermap.org/data/2.5/weather?q=" + $requestlocation + "&appid=[redacted]"
    Method      = "POST"
    ContentType = "application/json"
}

$response = Invoke-RestMethod @params

$location = $response.name
$temp = [math]::Round($response.main.temp - 273.15).ToString()
$temp = $temp + " degrees"
$type = $response.weather[0].main

Write-Output $location"," $type"," $temp
```

Kuvio 8: PowerShell-skripti

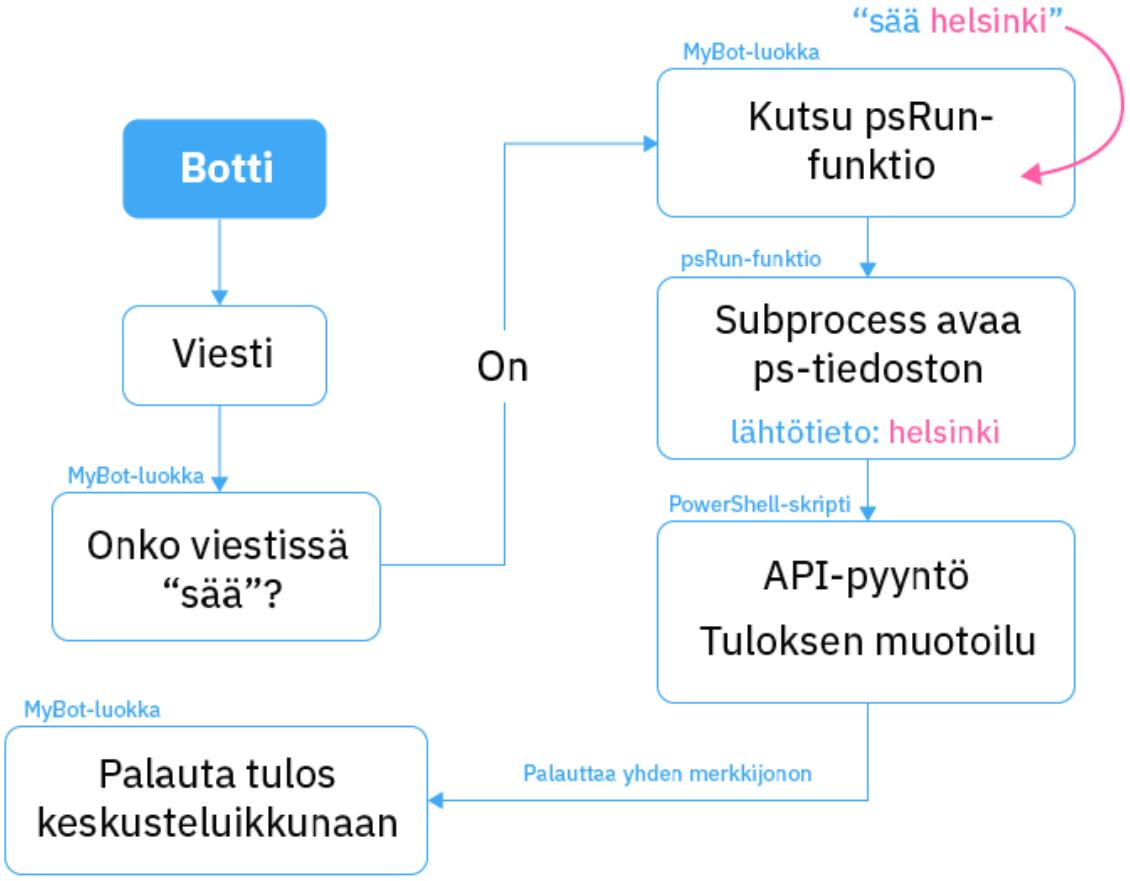
PowerShell-ominaisuuden testaamiseksi kehitin skriptin (kuvio 8), joka hakee OpenWeather-rajapinnasta ajankohtaisia säätietoja. Rajapinta vastaa palauttamalla JSON-objektin, josta voidaan rajata tarvittavat osat. Skripti sisältää API-pyynnön, sekä sen tuottaman vastauksen jakamisen sopiviin muuttujiin ja niiden muotoilun. Valmiissa botissa tullaan käyttämään samankaltaisia skriptejä, jotka hakevat tietoa ja palauttavat sen pääohjelmalle. Näin voidaan tuoda pääohjelmasta käskyn muuttuva osa, jolloin skripti hakee oikean tiedon rajapinnasta.

PowerShell-skripteissä tietoja voidaan tuoda kutsuvasta ohjelmasta args-taulukon avulla. Taulukossa voi olla useampi alkio, jotka on skriptin kutsussa erotettu välilyönnillä. Tämä tieto tallennetaan skriptin alussa ”requestlocation”-muuttujaan, joka syötetään rajapintapyynnön tietoihin.

4.4 Arviointi

Luvussa 2.3 esitellyn kehitysmallin mukaisesti prototyyppiä tulee arvioida kehittämisen jälkeen sekä toistaa kehitys- ja arviointivaiheita, kunnes saavutetaan haluttu tulos. Suoritin arviointeja ohjelman jokaisen osan kehittämisen jälkeen. Havainnoin ohjelman puutteita ja korjasin seuraavalla kierroksella ne. Arvioinnissa käytin seuraavia kysymyksiä:

- Tekeekö prototyyppi tehokkaasti sille tarkoitetut asiat?
- Onko ohjelmoinnissa käytetty tehokasta ja ymmärrettävää ohjelmointitapaa?
- Skaalautuuko prototyyppi, eli voidaanko siitä kehittää valmis ohjelma?



Kuvio 9: Prototyypin lopullinen toimintaprosessi

Viimeisellä arviointikierroksella piirsin kuvion 9 mukaisen vuokaavion, joka havainnollistaa prototyypin toteutuneen prosessin. Prosessi toteutui lähes samanlaisena, kuin se oli

suunniteltu. Lopullisessa prosessikaaviossa näkyy yksityiskohtaisemmin ohjelman toimintaprosessi.

Arvioinnissa olisi hyvän tavan mukaista käyttää ulkopuolista arvioijaa, joka olisi mielellään ohjelmaa käyttävä henkilö. Olisin voinut toimia tässä vaiheessa työn toimeksiantajan kanssa, mutta rajatun aikataulun vuoksi päätin suorittaa arvioinnin itse.

4.5 Käyttöönotto

Kehitysmallin mukaisesti tässä vaiheessa prototyypistä kehitetty ohjelma otettaisiin käyttöön. Käyttöönotto ei kuulu varsinaisesti opinnäytetyön rajaukseen, joten käsittelen sitä teoriassa.

Bot Frameworkilla kehitetty chatbot voidaan sijoittaa mille tahansa palvelimelle, joka voi suorittaa sekä Python-ohjelmointikieltä, että PowerShell-skriptejä. Käytännössä tällöin on kyseessä Windows-palvelin. Esimerkiksi Microsoftin IIS (Internet Information Services) -palvelimella on mahdollista suorittaa Python-ohjelmia. Yksinkertaisin tapa hostata bottia on ladata se Azure-pilvipalveluun, mutta silloin PowerShell-skriptien ajaminen ei onnistu käyttämälläni tavalla.

Palvelimesta riippumatta botti tulee yhdistää Azureen, jossa se rekisteröidään ja yhdistetään oikeaan keskustelukanavaan eli Microsoft Teamsiin. Azuressa luodaan botille oma resurssiallokaatio ja suoritetaan kanavarekisteröinti. Tässä vaiheessa lisätään myös botin serverin päätepiste eli osoite, josta Azure hakee botin viestit. Azure on hyödyllinen palvelu botin ylläpitäjälle, sillä se kerää lokeja ja analytiikkaa botin käytöstä. (Microsoft 2021b.)

Lopuksi botti tulee yhdistää kanavalle, jossa sitä käytetään eli esimerkiksi Microsoft Teamsiin. Ohjelman lisäksi Teamsia varten tulee tehdä paketti, jossa on json-muotoinen manifest-tiedosto, joka kuvaa botin tiedot ja ominaisuudet, sekä sovelluskuvakkeet (Microsoft 2021c). Tämän jälkeen botti ladataan Microsoft Teams kehittäjäportaalin kautta Teamsiin käytettäväksi (Microsoft 2021d).

5 Yhteenveto ja jatkokehitys

Opinnäytetyön tavoitteena oli kehittää chatbotin prototyyppi Microsoft Bot Framework-kehitysympäristössä. Prototyypin tarkoituksena oli toimia soveltuvuusselvitys-menetelmän (proof of concept) todistajana. Opinnäytetyössä kehitetty prototyyppi todisti konseptin toimivuuden kehitysympäristössä. Prototyyppi kykeni hakemaan tietoa ulkoisesta PowerShell-tiedostosta ja palauttamaan sen pääohjelmaan sekä keskusteluikkunaan. Kehityksessä käyttämäni prototyyppimalli mahdollisti joustavan ja iteratiivisen ohjelmistokehityksen.

Työ vastasi mielestäni hyvin asetettuihin kehittämiskysymyksiin ja oli teknisesti onnistunut. Saatujen tuloksien perusteella voidaan kehittää lopullinen ohjelma, joka vastaa organisaation tarpeisiin. Työn aikana olisi voinut tehdä enemmän yhteistyötä toimeksiantajan kanssa, mutta myös itsenäisellä työskentelyllä päästiin haluttuun päämäärään.

Prototyypistä puuttuu vielä valmiiseen ohjelmaan olennaisesti kuuluvia asioita, kuten virheiden käsittely. Kehittämistyön ulkopuolelle jää myös tarkempi selvitys botin palvelinympäristön mahdollisuuksista ja sen toimivuudesta loppukäytössä Microsoft Teamsissa. Työ vastasi näihin kehitystarpeisiin osittain, mutta niiden sopivuus tämänkaltaiselle botille ei ole varmistettua.

Lähteet

Sähköiset

Guru99. 2021. Prototyping Model in Software Engineering: Methodology, Process, Approach. Viitattu 19.5.2021. https://www.guru99.com/software-engineering-prototyping-model.html

Accenture 2016. Chatbots in Customer Service. Viitattu 11.2.2021. https://www.accenture.com/t00010101T000000__w__/br-pt/_acnmedia/PDF-45/Accenture-Chatbots-Customer-Service.pdf

Fichter, D. & Wisniewski, J. 2017. Chatbots Introduce Conversational User Interfaces. Online Searcher, 41 (1), 58.

Thorat, S. A. & Jadhav. V. 2020. A Review on Implementation Issues of Rule-based Chatbot Systems. International Conference on Innovative Computing & Communication (ICICC) 2020. Viitattu 24.3.2021. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3567047

Hash, S. 2019. Chatbots in the Contact Center, Part 2: Internal Use and Adoption. Viitattu 11.5.2021. https://blog.contactcenterpipeline.com/2019/03/chatbots-in-the-contact-center-part-2-internal-use-and-adoption/

Adler, S. How chatbots are enhancing employee experience with inspiring examples. Viitattu 11.5.2021. https://blog.intlock.com/chatbots-enhancing-employee-experience-inspiring-examples/

Wininger, S. The Rise of the Autonomous Organization.Viitattu 12.5.2021. https://www.lemonade.com/blog/rise-autonomous-organization/

Nichol, A. The next generation of AI assistants in enterprise. Viitattu 13.5.2021. https://www.oreilly.com/radar/the-next-generation-of-ai-assistants-in-enterprise/

Microsoft 2021a. Natural language processing in Composer. Viitattu 11.5.2021. https://docs.microsoft.com/en-us/composer/concept-natural-language-processing

Microsoft. 2019. Microsoft Bot Framework. Viitattu 24.3.2021. https://dev.botframework.com/

Microsoft. 2020a. How bots work. Viitattu 20.4.2021. https://docs.microsoft.com/fi-fi/azure/bot-service/bot-builder-basics?view=azure-bot-service-4.0

Microsoft. 2017. Activities overview. Viitattu 26.4.2021. https://docs.microsoft.com/en-us/previous-versions/azure/bot-service/dotnet/bot-builder-dotnet-activities?view=azure-bot-service-3.0

Velotio Technologies. 2018. An Introduction to Asynchronous Programming in Python. Viitattu 23.4.2021. https://medium.com/velotio-perspectives/an-introduction-to-asynchronous-programming-in-python-af0189a88bbb

Münster, K. 2019. Async/Await for Beginners— Understanding Asynchronous Code in Javascript. Viitattu 25.5.2021. https://javascript.plainenglish.io/async-await-for-beginners-understanding-asynchronous-code-in-javascript-748b57ae94e2

Ogbuji, U. 2018. Coroutines and asyncio. Viitattu 11.5.2021. https://developer.ibm.com/tutorials/ba-on-demand-data-python-3/

Microsoft. 2020b. Create a bot with the Bot Framework SDK for Python. Viitattu 27.4.2021. https://docs.microsoft.com/en-us/azure/bot-service/python/bot-builder-python-quickstart?view=azure-bot-service-4.0

Microsoft 2021b. Bot channels registration. Viitattu 7.5.2021. https://docs.microsoft.com/en-us/azure/bot-service/bot-service-quickstart-registration?view=azure-bot-service-4.0

Microsoft 2021c. Create a Microsoft Teams app package. Viitattu 7.5.2021. https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/build-and-test/apps-package

Microsoft 2021d. Upload your app in Microsoft Teams. Viitattu 8.5.2021. https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/deploy-and-publish/apps-upload

Kuviot