Tampereen ammattikorkeakoulu

Peliryhmän chattibotin toteutus

Toni Techtolin

OPINNÄYTETYÖ
Maaliskuu 2021

Tietojenkäsittely
Pelituotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Pelituotanto

TECHTOLIN, TONI:
Peliryhmän chattibotin toteutus

Opinnäytetyö 26 sivua
Maaliskuu 2021

Tämän opinnäytetyön tavoitteena oli ratkaista peliryhmän tapahtumien järjestämiseen ja muistutuksiin liittyviä ongelmia. Opinnäytetyön tarkoitus oli toteuttaa Discord-palvelun kautta toimiva chattibotti tapahtumien järjestämistä ja niistä muistuttamista varten.

Opinnäytetyö esittää yhden monista tavoista lähestyä yksinkertaisen chattibotin toteutusta. Siinä selitetään käytetyt tekniikat ja käydään läpi ohjelmointiin liittyvät ratkaisut.

Opinnäytetyön vaatimat ohjelmat asennettiin sekä Windows-käyttöjärjestelmää käyttävälle tietokoneelle että Raspberry Pi -tietokoneelle. Opinnäytetyön chattibotti otettiin myös käyttöön Raspberry Pi -tietokoneella. Toimivuutta testattiin käytännössä tapahtuman järjestämisessä sekä muistutuksien hyödyllisyyden suhteen.

Asiasanat: chattibotti, Discord, Javascript, Raspberry pi

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Business Information Systems
Game Development

TECHTOLIN, TONI:
Realisation of a Chatbot for a Game Group

Bachelor's thesis 26 pages
March 2021

The Objective for this thesis was to solve problems of a game group regarding event arrangement and reminding about events. The purpose of this thesis was to implement a chatbot, which functions on the Discord platform and allows event arrangement and gives reminders about events.

The thesis demonstrates one of the many ways to approach implementing a simple chatbot. The technologies used are explained, as well as the decisions made in the programming process.

The programs required for the implementation were installed on a computer with Windows operating system and on a Raspberry Pi computer. The chatbot was put to use on the Raspberry Pi computer. The functionality of the chatbot was tested through organizing an actual event, as well as with studying how helpful the reminders are.

Key words: chatbot, Discord, Javascript, Raspberry pi

SISÄLLYS

LYHENTEET JA TERMIT

API	Application Programming Interface, Ohjelmointirajapinta. API mahdollistaa kahden eri ohjelman keskustelun ennalta määritettyjen menettelyiden mukaisesti.
Argumentti	Funktiota kutsuttaessa sille välitettävä tieto.
Black Desert Online	Massiivinen monen pelaajan verkkopeli.
Discord	Viestintä- ja IP-puhelu-sovellus.
Kaavantunnistaminen	Toimintatapa, jossa syötettä verrataan ennalta määritettyihin sääntöihin ja toimitaan niiden perusteella.
Moduuli	Ohjelmiston itsenäinen osa; Lisäosa.
npm	Node.js-ajoympäristön pakettimanageri, jolla voidaan asentaa moduulipaketteja.
Null	Null merkitsee arvon puuttumista tai sitä, että oliota ei ole olemassa.
Olio	Koodissa olevan luokan ilmentymä; ohjelmakoodia sisältävä tietorakenne.
Webhook	Verkossa käytetty tekniikka tiedonvälitykseen käyttäjän määrittämällä tavalla.

1 JOHDANTO

Tämän opinnäytetyön tarkoitus on kehittää peliryhmälle ohjelmoitava chattibotti, joka edistää ryhmän tapahtumien järjestämistä onnistuneesti. Tämä chattibotti tulee käyttöön Black Desert Online -peliä pelaavalle peliryhmälle, joka haluaa mahdollisimman yksinkertaisen ja tehokkaan tavan saada tietoa pelin sisällä tapahtuvista asioista Discord-palvelimellaan. Peliryhmä on käyttänyt tätä palvelinta jo pidemmän ajan.

Taustalla ovat ongelmat, jotka liittyvät aikataulutukseen ja pelin tapahtumiin ilmoittautumiseen. Tällä hetkellä tapahtumiin osallistuminen on täysin peliryhmäläisten oman muistin varassa. Se aiheuttaa ongelmia onnistuneiden tapahtumien järjestämiseen. Tällaisia tapahtumia ovat esimerkiksi pelissä tapahtuvat tehtävät, jossa peliin osallistujilla on eri roolit, joista on sovittava etukäteen tehtävän onnistumisen takaamiseksi. Chattibotin tulisi myös mahdollistaa tapahtumiin itseilmoittautuminen Discord-palvelimen kautta, jolla chattibotti toimii. Sen pitää myös olla toiminnassa kellon ympäri niin, että virheen tapahtuessa chattibotti jatkaa toimintaansa. Täysin tarpeita vastaavaa chattibottia ei löytynyt valmiina käytettäväksi. Näin ollen itse tehty chattibotti on paras vaihtoehto.

Opinnäytetyöstä hyötyy lähinnä peliryhmä, jolle chattibotti tuotetaan. Se on käyttökelpoinen myös muille peliryhmille, jotka tarvitsevat aikataulua ja tapahtumien järjestämistä helpottavan chattibotin.

2 CHATTIBOTIT

Botteja ja niiden käyttöä on tutkittu hyvin runsaasti. Niitä on ollut olemassa pitkään, ne toteuttavat pieniä tehtäviä ja toimivat ennalta määrättyjen sääntöjen mukaisesti. Botit ovat ohjelmia, ei-ihmistoimijoita. (Laaksonen, Laitinen, Koivula & Sihvonen 2020, 63–78.)

Botteja tulee vastaan jokapäiväisessä elämässä paljon enemmän kuin voisi nopeasti olettaa. Kun sähköpostiin tulee roskapostia, hyvin usein sen takana on botti. Monien asiakaspalveluiden nettisivujen chateissa keskustellaan chattibotin kanssa ennen ihmisen kanssa viestimistä. Applen Siri, Amazonin Alexa sekä Googlen Google Assistant ovat chattibotteja. Edellä nimettyjä chattibotteja voi ohjata puheella ja niiltä voi kysyä apua moniin eri ongelmiin. Ne voivat hakea käyttäjälle esimerkiksi reitin haluttuun kohteeseen tai ratkaista annettu laskutehtävä. Tässä opinnäytetyössä käsitellään chattibottia, jolla on hyvin rajatut toiminnot. Se ei esimerkiksi toimi ääniohjatusti eikä hae tietoa muuta kuin ennalta määritellyistä asioista.

Chattibotin toteutukseen on kaksi lähestymistapaa riippuen käytetyistä algoritmeista ja tekniikoista: kaavantunnistaminen ja koneoppiminen. Kaavantunnistamisessa chattibotti vertaa käyttäjän syöttämää tietoa sisältämäänsä kaavaan ja valitsee ennalta määritetyn vastauksen. (Adamopoulou & Moussiades 2020, 4.) Opinnäytetyön chattibotin toiminta perustuu kaavantunnistamiseen.

3 KÄYTETYT TEKNIIKAT

Seuraavissa luvuissa käydään järjestyksessä läpi eri tekniikat, joita on käytetty toteutuksessa. Läpikäynti aloitetaan fyysisestä tietokoneesta ja askel askeleelta lähestytään Discord-palveluiden välityksellä tapahtuvaa keskustelua chattibotin kanssa.

3.1 Raspberry Pi

Raspberry Pi on hyvin pieni, kokoonsa nähden tehokas ja täysin toimiva tietokone. Chattibotin voisi antaa toimia millä tahansa käytössä olevalla tietokoneella. Erillinen tietokone on kuitenkin parempi vaihtoehto, koska chattibotin pitää toimia tauotta. Tästä syystä Raspberry Pi on todella hyvä, sillä se on äänetön ja kuluttaa vain vähän virtaa.

Raspberry Pi -tietokoneelle voi asentaa useita eri Linux-pohjaisia käyttöjärjestelmiä. Raspberry Pin kehittäjä Raspberry Pi Foundation on tuottanut oman käyttöjärjestelmänsä Raspberry Pi OS (aiemmin Raspbian), jota chattibotin tietokone käyttää. Kyseisestä käyttöjärjestelmästä löytyvät kaikki tarvittavat toiminnot vaatimuksien täyttämiseksi.

3.2 JavaScript

JavaScript on tulkattava oliopohjainen ohjelmointikieli. Se on todella yleinen ohjelmointikieli verkkosivuilla sekä muissa ohjelmissa. Yksi merkittävä ero muihin kieliin on sen muuttujien luokkamäärityksen puuttuminen, mikä tarkoittaa vapaampaa muuttujien käsittelyä.

JavaScript-kieltä pystyy käyttämään millä tahansa alustalla, jolle voidaan asentaa tai on integroitu JavaScript-ajoympäristö (Peltomäki 2017). Chattibotin ohjelmoinnissa on käytössä JavaScript, koska Discord.js ja Node.js, joita käsitellään luvussa 3.4, pohjautuvat JavaScriptiin.

3.3 JSON

JSON on avoimen standardin formaatti tiedonvälitykseen. Sen rakenne on alkujaan suunniteltu JavaScriptin pohjalta, ja JSON-tiedostot ovat ihmisen ymmärrettävissä. JSON on suunniteltu kahden kokonaisuuden ympärille: oliot ja taulukot. Olio on järjestämätön lista nimen ja arvon muodostamia pareja, joissa nimi on merkkijono ja arvon tyyppi on yksi JSONin tukemista tyypeistä. Taulukko on järjestetty lista arvoja, joilla ei ole erillistä nimeä. JSONin tukemat arvotyypit ovat luku, merkkijono, totuusarvo ja null. (Lee, Anjos & Satti 2021, 3.)

JSON-tiedostoon voidaan tallentaa hyvin monenlaista tietoa. Se voi sisältää suuria määriä dataa, esimerkiksi siihen voi tallentaa kaikkien käyttäjien nimet, lempinimet, puhelinnumerot ja niin edelleen. Esimerkkinä JSON-tiedostosta on lista Discord-palvelimen valvojien nimistä (kuvio 1).

```
{
    "moderators": [
        {"discordName":"Rageila", "ingameName":"Rageila"},
        {"discordName":"Sepu", "ingameName":"Sepulch"},
        {"discordName":"Fwuff", "ingameName":"DragonScythe"}
    ]
}
```

KUVIO 1. JSON-olio, joka sisältää valvojien nimet.

Chattibotissa JSON-tiedostoja käytetään tiedon tallentamiseen kiintolevylle, jotta tieto ei katoa virheen tapahtuessa tai tietokoneen uudelleenkäynnistyksessä. Tämä sallii myös mahdollisissa virhetilanteissa manuaalisen korjaamisen tiedostosta käsin millä tahansa tekstinmuokkausohjelmalla.

3.4 Node.js, Discord.js ja PM2

Node.js on avoimen lähdekoodin JavaScript-ajoympäristö, jonka avulla chattibotti toimii. Se on suunniteltu skaalattaviin verkkosovelluksiin ja toimii asynkronisesti, eli useampia asioita voidaan suorittaa yhtäaikaisesti. Tämän ansiosta chattibotti toimii, vaikka sillä olisi useampi käyttäjä samanaikaisesti.

Discord.js on Node.js-moduuli, joka mahdollistaa Discord-palvelun APIn kanssa keskustelun (Discord.js Dokumentaatio. 2021). Se on suosituin JavaScript-pohjainen koodikirjasto Discordin APIlle, ja sen kanssa työskentely on helposti ymmärrettävää, koska se pohjautuu olioihin. Tästä syystä se on valittu chattibotin toteutukseen.

PM2 on Node.js-prosessimanageri, jota käytetään tämän chattibotin käynnissä pysymisen varmistamiseen. Sen voi asettaa varmistamaan, että chattibotti käynnistyy uudelleen, jos jokin kriittinen virhe tapahtuu. PM2 pystyy käynnistämään chattibotin heti, kun tietokone käynnistyy. Tämän ansiosta esimerkiksi sähkökatkoksen jälkeen chattibotti käynnistyy automaattisesti uudelleen.

3.5 Discord

Discord on ohjelma, jota käytetään peliryhmien sisäiseen kommunikointiin. Sen välityksellä voi soittaa puheluita ja lähettää viestejä, kuvia sekä videoita. Kaikki Discordissa olevat palvelimet ovat oletuksena yksityisiä, eikä niihin voi liittyä vapaasti. Niihin tarvitsee aina kutsun linkkinä, ja vain valitut henkilöt voivat luoda kyseisiä linkkejä. Kuka tahansa voi luoda tilin Discordiin ja sen jälkeen oman palvelimen. Palvelimella voi olla useita kanavia, jotka ovat aina joko teksti- tai puhekanavia. Tämän opinnäytetyön chattibotti tulee toimimaan vain tekstikanavien kautta.

Kaikilla Discord-palvelimilla kaikki käyttäjät kuuluvat yhteen tai useampaan ryhmään (Discordissa nimellä Role). Näille ryhmille voidaan antaa erilaisia käyttöoikeuksia palvelimella ja sen kanavilla. Yksittäisiä käyttäjiä sekä ryhmiä voidaan merkata viestiin, jolloin he saavat erillisen ilmoitusmerkinnän vastaanotetusta viestistä.

3.6 Google Sheets

Google Sheets on Googlen ilmainen taulukkolaskentaohjelma. Sen lisäksi että sen käyttö on ilmaista, useat käyttäjät voivat käyttää ja muokata samaa taulukkoa. Taulukoiden muokkaaminen onnistuu myös samanaikaisesti, ja muokkaushistoria tallentuu. Muokkaushistoriaa voidaan käyttää vanhan version palauttamisessa.

Taulukkoon on mahdollista myös sisällyttää koodia, jonka avulla tietoa voi käsitellä hyvin nopeasti. Tässä opinnäytetyössä sitä on käytetty alkuperäisen aikataulun käsittelyssä sellaiseen muotoon, että se on chattibotin ymmärrettävänä tietona.

4 CHATTIBOTIN OHJELMOINTI

Tässä opinnäytetyössä ensimmäinen askel chattibotin toteutuksessa on Node.js-ajoympäristön asentaminen. Chattibotin ohjelmointia tehdään aluksi Windows-käyttöjärjestelmää käyttävällä tietokoneella. Tästä syystä Node.js asennetaan ohjelmalla, joka ladataan Node.js-verkkosivuilta. Node.js-ajoympäristöä asennettaessa samalla asentuu npm, joka on JavaScript-pakettimanageri. Kun nämä kaksi ovat valmiita, luodaan kansio chattibotille. Sen jälkeen komentoriviä käyttäen mennään kyseiseen kansioon ja ajetaan komento ”npm install discord.js”. Tämä lataa Discord.js-moduulin chattibotin kansioon, minkä jälkeen voidaan aloittaa chattibotin ohjelmointi.

Chattibotti käyttää useita eri kooditiedostoja. Näiden tiedostojen sisältämästä koodista osa suoritetaan Discord-palvelimelle lähetettävillä komentoviesteillä ja osa on taustalla toimivia skriptejä, joita käytetään useissa eri kooditiedostoissa. Lähes kaikki näistä kooditiedostoista ovat moduuleja, joissa voidaan määrittää, mitä asioita kyseisestä moduulista näkyy ulospäin. Lisäksi chattibotilla on yksi tiedosto, joka sisältää asetukset. Tähän tiedostoon tallennetaan chattibotin suojausavain, komentojen etuliite, omistajan tunniste sekä muita tunnisteita ja avaimia.

4.1 Tiedoston tallennus

Chattibotin muistissa oleva tieto pitää turvata, vaikka ajon aikana tapahtuisi virhe tai tietokone sammuisi sähkökatkon takia. Siksi chattibotti tallentaa tietoa tiedostoihin, mikä onnistuu hyvin lyhyen skriptin avulla.

Ensimmäisenä skriptissä otetaan File System -moduuli käyttöön. Se on Node.js-moduuli, joka auttaa tiedostojen käsittelyssä. Tämä moduuli on koodissa nimellä ”fs”, joka haetaan komennolla ”require('fs')”. Kyseinen komento lukee sulkujen sisällä annetun tiedoston tai yleisen moduulin, jonka nimi on annettu, tuottaa siitä

olion ja palauttaa sen, jotta sitä voidaan käyttää koodissa. Palautettu olio on tallennettu ”const fs” -muuttujaan, jossa ”fs” on koodin sisäinen muuttujanimi ja ”const” kertoo koodille, että kyseistä muuttujaa ei voi muokata tämän rivin jälkeen.

Seuraavaksi määritetään, mitä osia tästä skriptistä voidaan käyttää ulkoa päin. Tässä skriptissä ainoa asia, jota halutaan käyttää ulkoa, on funktio nimeltä ”saveFile”, joka yksinkertaisesti tallentaa sille annetun tiedoston annettuun polkuun.

Lopuksi skriptissä määritetään sen ainoa funktio. Tämä funktio käyttää fs-moduulia tiedostoon kirjoittamiseen. Funktiota kutsuttaessa sille pitää antaa polku (path) ja jotain mitä tallennetaan tiedostoon (file). Kun saveFile-funktiota kutsutaan, se kutsuu fs-moduulin writeFile-funktiota, joka ottaa vastaan edellä mainitun polun sekä tallennettavan tiedon. Näiden lisäksi writeFile-funktiolle voi antaa funktion, jota kutsutaan, kun se on valmis. Tässä tapauksessa writeFile-funktiolle annettu funktio tarkastaa, palauttiko se virheen. Virheen tapahtuessa konsoliin kirjautuu tiedot tapahtuneesta virheestä (kuvio 2).

```
1   const fs = require('fs');
2
3   module.exports = {saveFile}
4
5   function saveFile(path, file){
6       fs.writeFile(path, file, function(err) {
7           if (err) {
8               console.log(err);
9           }
10      });
11  }
```

KUVIO 2. Tiedontallennusskriptin koodi.

4.2 Ajan laskeminen

Osa peliryhmän seuraamista tapahtumista toistuu viikoittain. Tästä syystä chattibotille ohjelmoidaan skripti, joka käsittelee ajan laskemisen. Ennen skriptin ohjelmointia täytyy päättää, kuinka aikaa tullaan käsittelemään.

Yksi tapa on eritellä kaikki päivät erikseen, niiden alle tunnit ja niiden alle minuutit, mikäli minuuttitarkkuutta tarvitaan. Tällaisen aikajärjestelmän käsittely koodin puolella on hieman työlästä, jos täytyy laskea esimerkiksi aika seuraavaan kolmeen tapahtumaan riippumatta siitä, missä päivässä, tunnissa tai minuutissa ollaan. Tästä johtuen chattibotille ohjelmoitiin myös ajan laskeminen millisekunneissa. Molemmissa tavoissa käytössä on JavaScriptin oma Date-olio, joka on tarkoitettu yleisesti ajan laskemiseen.

JavaScriptin Date-olio esittää yksittäistä pistettä ajassa alustasta riippumattomassa muodossa. Date-oliot sisältävät numeerisen tiedon, joka esittää kulunutta aikaa millisekunneissa vuoden 1970 tammikuun ensimmäisen päivän alusta UTC-aikajärjestelmässä. (MDN Web docs n.d.)

Kaikilla JavaScriptin Date-olioilla on funktioita, joita voidaan käyttää suoraan tämänhetkisen päivän, tuntien ja minuuttien näyttämiseen, mutta näitä funktioita ei tulla käyttämään itse tapahtumien ajan laskemisessa. Sen sijaan skriptiin tehdään oma funktio, joka laskee millisekunnit viikon alusta. Näin saamme laskettua millisekunneissa tietyn pisteen viikossa, joka on sama numeerisesti joka viikko. Tähän funktioon on myös sisällytetty aikavyöhykkeen tuoma aikaero UTC-ajasta (kuvio 3).

```
function getMilli() {
    var hourAdjustment = 2;
    var curDay = new Date();
    var weekMilli = ((curDay.getTime() - (1540155600000 + (3600000 * hourAdjustment))) % 604800000);
    return weekMilli;
}
```

KUVIO 3. Millisekuntien laskenta viikon alusta.

Lisäksi skriptissä on funktio, joka muuntaa aikaisemman funktion antamat millisekunnit päiviksi, tunneiksi ja minuuteiksi. Näin saadaan muokattua selkeästi ymmärrettävä aika. Skripti sisältää myös funktion, joka palauttaa tiedon siitä, kuinka monta sekuntia kuluu, kunnes siirrytään seuraavaan minuuttiin. Tätä funktiota käytetään chattibotin käynnistyshetken normalisointiin.

4.3 Kooditapahtumat

Tämä luku käsittelee koodissa olevia tapahtumia, joita selvyyden vuoksi kutsun kooditapahtumiksi. Jotta chattibotti saadaan reagoimaan Discord-palvelussa lähetettyihin viesteihin, sille täytyy määrittää kooditapahtumat, joita sen täytyy kuunnella.

Ensimmäinen on kooditapahtuma, jota chattibotin täytyy kuunnella on ”ready”, joka tapahtuu, kun chattibotti on valmis ottamaan yhteyden Discordin APIin. Toinen kooditapahtuma, jota se kuuntelee, on ”err”. Tämä tapahtuu aina, kun jokin virhe ilmaantuu Discordin APIn kanssa keskusteltaessa. Näissä kahdessa tapauksessa chattibotti kirjoittaa konsoliin, mitä on tapahtunut, ja jatkaa koodin suoritusta.

Seuraava kooditapahtuma tulee Discord-palvelun puolelta, kun uusi jäsen liittyy Discord-palvelimelle. Tässä tapauksessa chattibotti lähettää automaattisesti tervetuloviestin ja ohjeistaa uutta jäsentä lukemaan ryhmän säännöt.

Viimeinen kooditapahtuma, jota chattibotti kuuntelee, on ”message”. Tämä tapahtuu aina, kun uusi viesti saapuu Discord-palvelimelle tai chattibotti vastaanottaa yksityisviestin. Tässä kooditapahtumassa sen täytyy tarkastaa viestin sisältöä sekä sitä, kuka on lähettäjä ja mihin viesti saapui. Ensimmäisenä täytyy katsoa, onko viestin lähettäjä chattibotti, koska sen ei tule jutella muille chattiboteille tai itsellensä. Sitten tarkastetaan, onko viesti saapunut yksityisviesteihin. Viimeisenä selvitetään, alkaako viesti komentojen etuliitteellä.

4.4 Komentojen määrittely ja suoritus

Chattibotin komentojen määrittelyskripti lukee kaikki tiedostot ”commands”-nimisestä kansiosta ja tarkastaa, sopivatko ne komennoiksi etsimällä ”.js”-tiedostopäätettä. Tiedostopääte ”.js” määrittää tiedostot JavaScript-tiedostoiksi, ja tässä tapauksessa ne ovat moduuleita, jotka voidaan suorittaa. Kaikista tiedostoista, jotka läpäisevät tarkastuksen, tehdään olio ”require”-komennon avulla. Nämä

oliot tallennetaan muistiin kokoelmaan, joka sisältää nimi-arvo-pareja. Näissä pareissa nimenä on tiedoston nimi ja arvona ”require”-komennon tuottama olio, jota voidaan kutsua Discord-palvelun kautta komennon sisältävällä viestillä.

Kun Discord-palvelussa lähetetään viesti, jossa ensimmäinen merkki vastaa komentojen etuliitettä, chattibotti lukee viestin. Viestistä poistetaan etuliite ja se eritellään osiin välilyöntien kohdalta ja osat laitetaan listaan. Listan osista muutetaan kaikki kirjaimet pieniksi kirjaimiksi. Tämän jälkeen komentokokoelmasta etsitään nimi-arvo-paria, jossa nimi vastaa listan ensimmäistä osaa. Jos kokoelmasta löytyy osuma, kyseisen olion skripti suoritetaan ja sille annetaan myös viittaus alkuperäiseen viestiin sekä listan loput osat argumentteina. Jos kokoelmasta ei löydy osumaa, chattibotti reagoi viestiin kysymysmerkillä, jotta käyttäjä tietää, että jokin meni pieleen.

Yksi näistä komennoista on käytössä tapahtumien muistutusten tilaamiseen. Käyttäjä voi vapaasti tilata yksittäisten tapahtumien muistutukset, jotka chattibotti lähettää tietyn aikaa ennen tapahtuman alkamista. Käyttäjä voi myös peruuttaa tilauksiaan tai tarkistaa, mitkä tapahtumat on tilannut. Tilanteissa, joissa komentoa ei voida suorittaa, chattibotti vastaa viestiin virhetekstillä. Esimerkiksi käyttäjä, joka ei ole tilannut tapahtuman muistutusta, mutta yrittää perua tilauksen, saa virheilmoituksen. Tilaajan tunnistenumero sekä lista tapahtumista, jotka he ovat tilanneet, tallennetaan erilliseen JSON-tiedostoon. Lisäksi käyttäjä voi käyttää komentoa, joka lisää heidät ryhmään, jota muistutetaan kaikista tulevista tapahtumista erillisellä viestillä.

Toinen tärkeä komento käsittelee tapahtumien järjestämistä. Käyttäjät voivat ilmoittautua tapahtumiin itsenäisesti Discord-palvelun kautta, perua ilmoittautumisensa ja tarkastaa, ketkä ovat ilmoittautuneet. Jos tapahtumassa on erillisiä ryhmiä roolijakoa varten, käyttäjät pystyvät valitsemaan haluamansa roolin. Vain valvojilla on oikeus tyhjentää osallistujalistoja.

4.5 Chattibotin käynnistyspiste

Käynnistyspisteenä chattibotille on tiedosto nimeltä index.js. Se tarkoittaa sitä, että Node.js-ajoympäristöä käsketään suorittamaan tämä tiedosto, joka käynnistää chattibotin toiminnan. Index.js-tiedosto vastaa Discord.js:n pääteohjelman käyttöönotosta, joka vastaa Discord-APIn kanssa keskustelusta. Tämä pääteohjelma myös käyttää edellä mainittuja skriptejä toimintaansa.

Discord-APIin yhdistämiseen pääteohjelma vaatii suojausavaimen, joten asetustiedosto pitää ottaa myös käyttöön index.js-tiedostolle. Kun kirjautuminen on onnistunut, Discord-API kertoo pääteohjelmalle, että se on valmis, ja pääteohjelma laukaisee ”ready”-kooditapahtuman. Tätä kooditapahtumaa kuunteleva skripti vuorostaan aloittaa chattibotin komentojen suoritusketjun.

4.6 Chattibotin suoritusketju

Suoritusketju koostuu useista koodipätkistä, joilla kaikilla on oma tehtävänsä. Se suoritetaan aina minuutin välein. Suurin osa suoritusketjusta vastaa ennalta määritetyistä tapahtumista tiedottamisesta peliryhmän jäsenille. Tiedottamisesta vastaavat koodipätkät ovat erilaisia tapoja muistuttaa jäseniä tulevista tapahtumista ennen niiden alkamista, esimerkiksi yksityisviestillä 40 ja 20 minuuttia ennen tapahtumaa.

Ennalta määritettyjen tapahtumien muistutusta varten chattibotti tarvitsee aikataulun. Black Desert Online -pelin tarjoajan antama aikataulu tapahtumista on kuvana, eikä sitä ole valmiiksi chattibotin ymmärtämässä muodossa tarjolla. Tästä syystä käytössä on Google Sheets, johon aikataulu on kopioitu manuaalisesti. Lisäksi sinne on tehty vapaasti muokattava toinen aikataulu, johon voidaan lisätä normaalin aikataulun ulkopuolella olevat tapahtumat. Näiden kahden taulun lisäksi sieltä löytyy painike, johon on yhdistetty skripti, joka käsittelee aikataulut chattibotin ymmärtämään muotoon.

Skripti lukee ensimmäisen taulukon ja laskee joka solulle millisekunneissa ajan, jolloin sen sisältämä tapahtuma alkaa. Jokaisessa solussa on kirjoitettuna tapahtuma, joka alkaa kyseisenä aikana, joten skripti luo olion jokaisen solun ja sen millisekuntien perusteella. Nämä oliot lisätään listaan, joten olioita tulee yhtä monta kuin eri aikoina alkavia tapahtumia on. Seuraavaksi skripti tekee saman toiselle aikataululle, lisää sen perusteella luodut oliot samaan listaan aikaisempien olioiden kanssa ja järjestää listan ajan perusteella. Tämä lista voidaan kirjoittaa tekstinä erilliseen soluun, ja teksti kopioidaan suoraan JSON-tiedostoon. Chattibotti käyttää tätä tiedostoa, kun se etsii seuraavaa tapahtumaa. Tapahtumien järjestyksen selvittämisessä on käytössä skripti, joka on ohjelmoitu siten, että se voi muistuttaa mistä tahansa tapahtumista, jotka on listattu aikatauluihin.

Tätä skriptiä käytetään useissa paikoissa eri tavoilla ja hieman eri tarkoituksiin. Tästä syystä ainoa funktio, jota voidaan kutsua skriptin ulkopuolelta, käsittelee kutsun ennen seuraavan tapahtuman etsimistä. Funktio ottaa vastaan kaksi argumenttia: tapahtuman nimi, jota etsitään sekä osumien määrä. Näistä molemmat ovat valinnaisia argumentteja, joten funktiota voidaan käyttää neljällä eri tavalla. Kutsumalla ilman argumentteja saadaan seuraava tapahtuma. Antamalla vain osumien määrä, saadaan valitun määrän mukaisesti seuraavat tapahtumat. Pelkällä tapahtuman nimellä saadaan aika, jolloin kyseinen tapahtuma on seuraavan kerran. Kun kutsussa annetaan sekä tapahtuman nimi että osumamäärä, saadaan esimerkiksi ajat seuraaviin kolmeen tapahtumaan, jotka vastaavat funktiolle argumenttina annettua nimeä.

Kun kutsu on käsitelty, seuraavan tapahtuman etsiminen aloitetaan kutsumalla toista funktiota. Sille lähetetään argumentteina sillä hetkellä viikosta kulunut millisekuntimäärä sekä mahdollisesti tapahtuma, jota etsitään. JSON-aikataulutiedostoa käydään läpi olio kerrallaan, ja jokaisesta tarkastetaan sen tapahtuma-aika. Jos aika on suurempi kuin argumenttina annettu millisekuntimäärä, on tarkastettava vastaako se mahdollisesti annettua tapahtuma-argumenttia. Mikäli taulukosta ei löydy osumaa kuluneelle viikolle, haetaan taulukon alusta ensimmäinen olio tai etsitään taulukosta ensimmäinen olio, joka vastaa annettua tapahtuma-argumenttia. Jos tarkastettu olio vastaa tapahtuma-argumenttia tai tapahtuma-argumenttia ei ole annettu, oliosta tehdään kopio. Jos kopiota ei tehdä,

myöhemmin tehtävät muutokset vaikuttavat alkuperäiseen aikatauluun. Kopioidun olion ajasta vähennetään argumentin millisekuntien verran aikaa, jolloin jäljelle jää millisekuntien määrä tapahtuman alkuun. Tämä olio palautetaan funktion kutsujalle, ja kutsuja käsittelee tiedon loppuun.

Viime kappaleissa käsiteltyä skriptiä sekä sen funktioita kutsutaan suoritusketjussa ensimmäisenä ilman argumentteja. Sen palauttamassa oliossa oleva millisekuntimäärä muutetaan ymmärrettävään muotoon ajanlaskentaskriptin avulla. Sitten chattibotin näyttämä aktiviteetti Discord-palvelimella muutetaan näyttämään tapahtuman nimi ja aika tapahtuman alkamiseen (kuvio 4).

KUVIO 4. Esimerkki chattibotin näyttämästä aktiviteetista.

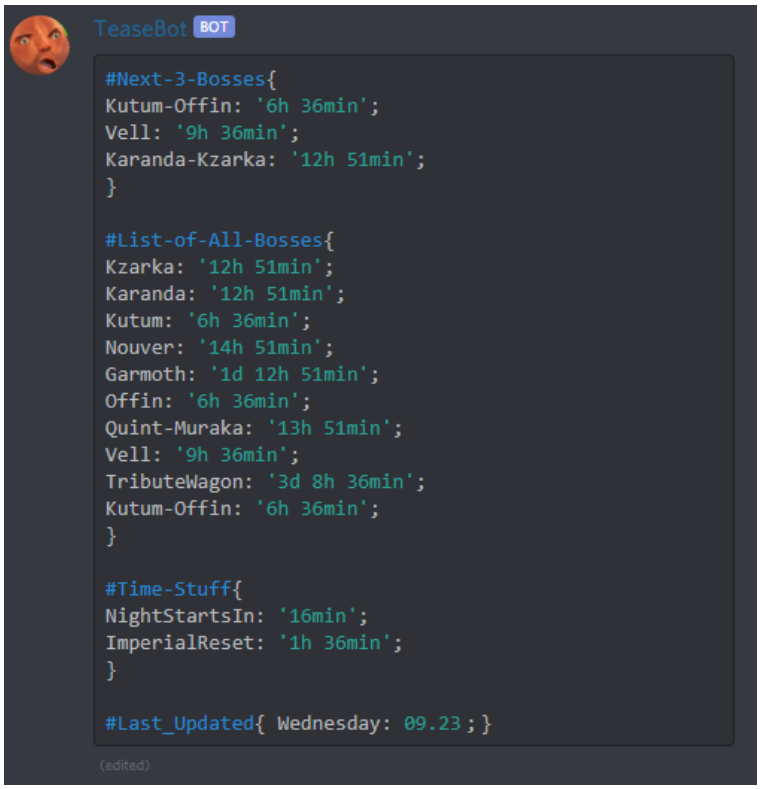
Seuraavaksi chattibotti tarkastaa, onko tapahtuman alkamiseen alle 40 minuuttia. Jos aikaa on alle 40 minuuttia, chattibotti lähettää muistutuksen ennalta määritetylle tekstikanavalle Discord-palvelimella. Tähän muistutukseen merkataan ryhmä, joka on määritetty asetuksissa. Lisäksi tapahtuman muistutuksen tilanneille jäsenille lähetetään muistutus yksityisviestinä. Muistutuksien lähettämisessä täytyy ottaa huomioon, että jäsen on saattanut lähteä Discord-palvelimelta. Tällaisessa tilanteessa aiheutuu virhe, joka pysäyttää chattibotin toiminnan. Tämä estetään käsittelemällä virhe erikseen sen tapahtuessa.

Viimeinen osio suoritusketjussa hallinnoi niin sanottua ilmoitustaulua. Tämä on erillinen tekstikanava Discord-palvelimella, jolla vain chattibotti voi muokata ja lähettää viestejä. Tällä tekstikanavalla chattibotti päivittää yhtä viestiä, johon on upotettuna ajat tunneissa ja minuuteissa, kunnes tapahtumat alkavat. Koska Discord sallii sisällön upottamisen viesteihin, niihin voidaan upottaa koodia, joka värjäytyy merkintäkielen mukaisesti.

Merkintäkielillä yleensä pyritään kuvaamaan tekstin rakennetta ja erottamaan looginen rakenne sisällöstä. Merkintäkielen käyttämisen tulos on usein nähtä-

vissä ohjelmointityökaluissa, joissa eri tekstin osat muuttuvat erivärisiksi kontekstin mukana. Ilmoitustaulussa tätä käytetään hyödyksi tiedon erittelemisessä väreillä. Discord-palvelussa merkintäkielenä on Markdown, jolla viestiin upotettu koodi voidaan merkata sisältämään monia eri ohjelmointikieliä sekä formaatteja kuten C++, CSS, HTML, JavaScript ja JSON. Kaikissa vaihtoehdoissa tekstin värit vaihtuvat sen syntaksin mukaisesti, eikä niitä voi vapaasti vaihtaa itse. Chattibotille valittiin CSS sen merkitsemistavan ja värityksen perusteella.

Ilmoitustaululla näytetään ensimmäisenä ajat kolmen seuraavan tapahtuman alkamiseen. Tämän jälkeen näytetään listana ajat kaikkien tapahtumien alkamiseen. Näin saadaan tehtyä yksi tekstikanava, josta käyttäjät voivat aina katsoa ajan seuraavaan haluamaansa tapahtumaan (kuvio 5).



KUVIO 5. Chattibotin ilmoitustaulu Discord-palvelimella.

4.7 Discord-palvelimen valmistaminen

Ensimmäinen askel chattibotin lisäämiseen Discord-palvelimelle on uuden sovelluksen luominen Discordin kehittäjien verkkopalvelussa. Sovelluksen voi luoda kuka tahansa omalla Discord-käyttäjätunnuksellaan. Sovellusta luodessa sille

täytyy antaa nimi, mutta sitä voidaan muokata myöhemmin. Luomisen jälkeen on mahdollista antaa sovellukselle kuvausteksti ja profiilikuva, jotka näkyvät chattibottia lisättäessä palvelimelle. Tältä sivulta on tärkeää ottaa sovelluksen tunnistenumero talteen, koska sitä tarvitaan chattibotin kutsumiseen Discord-palvelimelle.

Seuraavaksi sovellukselle tehdään bottikäyttäjä. Ilman bottikäyttäjää sitä ei voida kutsua Discord-palvelimille. Se onnistuu helposti kehittäjien verkkopalvelun kautta. Bottikäyttäjää luodessa palvelu varoittaa, että sitä ei voida poistaa luomisen jälkeen. Luomisen jälkeen bottikäyttäjän nimi ja profiilikuva voidaan vaihtaa samalla sivulla, missä se luotiin. Nämä ovat näkyvillä Discord-palvelimilla, joille chattibotti kutsutaan. Näiden lisäksi bottikäyttäjältä löytyy suojausavain, jota käyttämällä kuka tahansa voi kirjautua bottikäyttäjälle. Tästä syystä sitä ei kannata jakaa, ja se kannattaa pitää salassa mahdollisimman hyvin.

Bottikäyttäjän sivulla on myös työkalu, jonka avulla voidaan varmistaa, että bottikäyttäjää lisättäessä palvelimelle se saa tarvittavat käyttöoikeudet. Jokaisella käyttöoikeudella on oma bittilukunsa, jonka perusteella lasketaan kokonaisluku. Tätä lukua käytetään nettiosoitteessa, jolla chattibotti kutsutaan Discord-palvelimelle.

Kun bottikäyttäjä on tehty ja sovelluksen tunnistenumero sekä käyttöoikeuksien laskennallinen kokonaisluku tiedetään, voidaan kutsua chattibotti Discord-palvelimelle osoitteessa https://discord.com/oauth2/authorize?client_id=Sovelluksen-Tunnistenumero&scope=bot&permissions=KäyttöoikeuksienKokonaisluku.
Osoitteeseen täytyy korvata sovellustunnistenumero sekä käyttöoikeuksien kokonaisluku tiedossa olevilla arvoilla. Tällä sivulla käyttäjä voi valita, mille Discord-palvelimelle chattibotti kutsutaan. Chattibotin voi kutsua ainoastaan palvelimille, joissa käyttäjällä on hallintaoikeudet.

Tämän jälkeen chattibotti on lisättynä Discord-palvelimelle. Palvelimelta puuttuu vielä ryhmät, joita chattibotti voi muistuttaa. Tämän lisäksi ryhmän tunnistenumero täytyy lisätä chattibotin asetustiedostoon, samoin kuin valvojien ryhmän tunnistenumero. Ryhmän tunnistenumeron saa selville kahdella eri tavalla. Nopea tapa on kirjoittaa viesti, jossa ryhmä on merkattuna, mutta ennen merkkausta

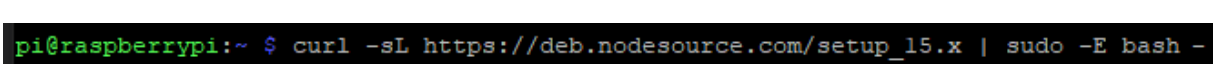
kirjoitetaan kenoviiva. Kun viesti on lähetetty, merkkaus näyttää ryhmän nimen sijaan tunnistenumeron. Toinen tapa vaatii kehittäjätilan aktivoimista, joka onnistuu Discord-palvelun käyttäjäasetuksista. Tämän jälkeen Discord-palvelimen asetuksissa ryhmän nimeä voidaan napsauttaa hiiren kakkospainikkeella, mikä antaa vaihtoehdon kopioida ryhmän tunnistenumero.

Viimeisenä pitää luoda kanavat, joille muistutuksia lähetetään. Se onnistuu helposti napsauttamalla hiiren kakkospainikkeella Discord-palvelimen kanavalistaa, ja valitsemalla ”luo kanava”. Jotta kanavan nimi ei vaikuta chattibotin toimintaan, täytyy luoda webhook, joka sallii viestin suoraan lähettämisen linkitetylle kanavalle. Tekstikanavaa napsauttamalla hiiren kakkospainikkeella voidaan mennä kanavan asetuksiin, joista löytyy webhookin luominen. Webhookin nimellä ei ole muuta merkitystä kuin sen tunnistaminen itse, koska chattibotti tarvitsee vain webhookin URL:stä löytyvää tunnistetta. Tunniste on URL:n viimeisen kauttaviivan jälkeen oleva merkkijono. Tämä merkkijono tallennetaan myös chattibotin asetustiedostoon.

4.8 Raspberry Pi -tietokoneelle siirto

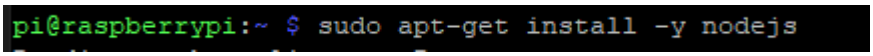
Kun chattibotti on valmis käytettäväksi, se siirretään Raspberry Pi -tietokoneelle. Tiedot siirretään USB-tikun avulla, koska tämä on yksinkertainen ja nopea ratkaisu. Jotta chattibotti voi toimia Raspberry Pi -tietokoneella, sille täytyy asentaa Node.js-ajoympäristö. Moduuleita ei tarvitse asentaa erikseen, koska ne siirtyivät chattibotin tiedostojen mukana.

Node.js:n asennus Raspberry Pi -tietokoneelle toimii eri tavalla kuin Windows-tietokoneelle. Ensin tietokoneelle täytyy ottaa käyttöön NodeSource nimisen yhtiön tarjoama repositorio, jota varten käytetään komentoriviä. Komentoriville kirjoitettavalle komennolle täytyy antaa tiedostosijainti, josta haetaan tietoa. Tässä tapauksessa tietoa haetaan ulkoiselta verkkosivulta (kuvio 6).



KUVIO 6. NodeSourcen tarjoaman repositorion käyttöönottokomento.

Seuraavaksi Node.js täytyy asentaa tietokoneelle repositoriosta. Tämä tehdään myös komentorivillä käyttäen komentoa ”sudo apt-get install -y nodejs” (kuvio 7). Onnistuneen asennuksen varmistamiseksi käytetään komentoa ”node -v”, joka kirjoittaa komentoriville Node.js-versionumeron.



KUVIO 7. Node.js:n asennuskomento.

Tämän jälkeen asennetaan PM2-prosessimanageri sekä tarvittavat asetukset, jotta chattibotti pysyy aina käynnissä. Asennus käy jälleen komentorivin kautta komennolla ”npm install pm2@latest -g”. Koska chattibotin käynnistys tapahtuu tiedostosta index.js, käytetään komentoa ”pm2 start index.js”. Tämä kertoo PM2:lle mikä tiedosto halutaan suorittaa. Jotta PM2 saadaan suorittamaan tiedosto heti tietokoneen käynnistyessä, täytyy käyttää komentoa ”pm2 startup”. Tämä luo käynnistykseen sidotun skriptin. Komennolla ”pm2 save” varmistetaan, että tiedoston suorittaminen käynnistyy uudelleen, mikäli sen suoritus loppuu mistä tahansa syystä.

4.9 Chattibotin testaus ja käyttöönotto

Ennen chattibotin lopullista käyttöönottoa testataan, ovatko kaikki skriptit toimivia, sekä toimivatko ne halutulla tavalla. Testaamisessa käytetään Discord-palvelinta, joka on luotu chattibotin testaamista varten, ja sinne kutsutaan peliryhmän valvojia testaajiksi. Jos testaamisen aikana tapahtuu virheitä, niiden syyt selvitetään ja korjataan. Näin varmistetaan, että chattibotti toimii oikein.

Ensimmäisenä testataan virheiden kirjaaminen sekä komentojen suoritus. Tätä varten tehdään tarkoituksellisesti viallisia komentotiedostoja sekä käytetään komentoja väärin. Kun virheiden kirjauksen on todettu toimivan, sekä komennot latautuvat ja suoritetaan oikein, testataan tiedontallennusta. Tämä tehdään tallentamalla tekstiä tiedostoon, jota ei ole vielä olemassa, jolloin skriptin pitäisi luoda tiedosto. Sen jälkeen tallennetaan eri teksti tähän samaan tiedostoon ja tarkastetaan, muuttuuko sen sisältö.

Kun chattibotin perustoiminnat on varmistettu toimiviksi, testataan suoritusketjun osia. Tämä aloitetaan testaamalla viestin lähettämistä tekstikanavalle sekä webhookin kautta että vastauksena käyttäjän lähettämään viestiin. Tämän jälkeen voidaan testata itseilmoittautuminen ryhmään, joka on tehty muistutuksia varten. Kun käyttäjät pystyvät ilmoittautumaan ryhmään, testataan ryhmän merkitsemistä viestiin tunnistenumerolla, joka on tallennettuna asetuksiin. Lisäksi tässä vaiheessa testataan yksityisviestien lähetystä käyttäjille heidän tunnistenumeroidensa perusteella.

Seuraavaksi varmistetaan, että Discordissa näytettävä aktiviteetti on oikein ja ilmoitustaulu päivittyy. Aktiviteettia testattaessa täytyy ottaa huomioon, että tapahtuman nimi mahtuu näytettäväksi. Ilmoitustaulua varten on tärkeätä varmistaa, että skripti lähettää ilmoitustaulukanavalle uuden viestin, jos kanavalla ei ole viestiä. Samalla varmistetaan, että ilmoitustaulu päivittyy oikein ja näytetyt ajat viittaavat aina seuraavaan tapahtumakertaan. Kun nämä kaikki asiat ovat toimintakunnossa, suoritusketjun automaattiset osiot on tarkistettu.

Tämän jälkeen testattavana on tapahtumiin itseilmoittautuminen ja niihin liittyvien ryhmien hallinta. Tapahtumiin itseilmoittautumisessa ensimmäisenä testataan, mitä nimiä komento voi käsitellä ilman virheitä. Koska Discord käsittelee käyttäjien merkinnän sekä hymiöt numerosarjoina, niitäkin voidaan käyttää ilmoittautumisessa nimenä ja ryhmien nimissä ilman virheitä. Tämän lisäksi osallistujia tarkastettaessa hymiöiden ja merkittyjen käyttäjien numerosarjat näkyvät normaaleina samalla tavalla kuin ilmoittautuessa. Tässä vaiheessa testataan myös se, että vain valvojat voivat tyhjentää ryhmiä.

Kun kaikki chattibotin toiminnat on testattu, se voidaan ottaa käyttöön peliryhmän Discord-palvelimella. Tässä vaiheessa täytyy muistaa päivittää chattibotin asetustiedostoon ryhmien ja webhookien tunnistenumerot. Tämän jälkeen chattibotti on täysin toimintakunnossa ja se voidaan ottaa käyttöön virallisesti. Kaikille peliryhmän jäsenille kerrotaan chattibotin toiminnoista yleisellä tasolla käyttöönoton yhteydessä. Lisäksi valvojille annetaan opastus sen toiminnoista tarkemmin ja selitys siitä, kuinka chattibottia voidaan käyttää tapahtumien järjestämisen helpottamiseksi.

5 POHDINTA

Opinnäytetyön tarkoituksena oli toteuttaa Discord-palvelimelle chattibotti, joka auttaa peliryhmää järjestämään tapahtumia onnistuneesti. Chattibotin toteutus onnistui mielestäni hyvin, koska peliryhmän tarpeet täytettiin. Sen toteuttaminen on helpottanut huomattavasti tapahtumien järjestämistä ja ihmisresurssien jakamista eri tehtäviin. Osallistujamäärät ovat myös kasvaneet peliryhmän tapahtumamuistutusten ansiosta. Tämän lisäksi kaikki tulevat tapahtumat ovat listattuina peliryhmän Discord-palvelimella, mikä helpottaa tiedon löytämistä.

Joitakin osia chattibotista tullaan jatkokehittämään varmasti. Jo opinnäytetyötä tehdessä käyttäjät antoivat kehitysideoita; esimerkiksi infotauluun lisättiin kolmen seuraavan tapahtuman aloitusajat. Valvojat antoivat palautetta chattibotin testauksen aikana, mikä otettiin huomioon tekemällä muun muassa käsky, joka listaa kaikki muut chattibotin komennot. Lisäksi lähes kaikille komennoille luotiin apufunktio, joka kirjoittaa sen toiminnat Discord-palvelimelle, kun käskyviestin ensimmäinen argumentti on ”help”. Chattibottia on mahdollista laajentaa myös niin, että se toimii useilla Discord-palvelimilla samanaikaisesti.

Chattibotin toteutusta varten haetusta tiedosta on ollut paljon hyötyä. Chattibottien päätöksenteon algoritmit ovat selventyneet opinnäytetyötä suunniteltaessa ja toteutettaessa. Toisaalta chattiboteista lukeminen on herättänyt kysymyksiä siitä, mikä luokitellaan chattibotiksi. Tällä hetkellä chattibotti-termin määritelmä on hyvin laaja ja se voi herättää todella useita erilaisia mielikuvia lukijoille.

Minkä tahansa botin toteutukseen voidaan käyttää useita eri ohjelmointikieliä. JavaScript on yleisin Discord-bottien toteutuksessa, mutta esimerkiksi Pythonille ja C#:lle löytyy kirjastoja botin luomiseen. Tämän chattibotin kehityksen aikana olisi ollut hyvä tietää tarkemmin Node.js-ajoympäristön toiminnasta, esimerkkinä sen tapa käsitellä tiedostoja ”require”-komennon yhteydessä. Tämän komennon tuottama olio tallennetaan myös välimuistiin Node.js:n kokoelmaan. Seuraavan kerran kun samaa tiedostoa haetaan ”require”-komennolla, Node.js:n kokoelmaan tallennettu olio palautetaan. Tämän takia mahdolliset muutokset tiedostoon eivät lataudu, ja sen kiertämiseksi olio täytyy poistaa ensin kokoelmasta.

Tulevaisuudessa voisi olla kiinnostavaa tutkia ja ohjelmoida chattibottia eri ohjelmointikielillä, esimerkiksi C#:lla. Se toisi osaamista laajemmalta alueelta bottien ohjelmointia varten, ja samalla tutustuisi muiden ohjelmointikielien ominaisuuksiin.

LÄHTEET

Adamopoulou, E. & Moussiades, L. 2020. Chatbots: History, technology, and applications. Machine Learning with Applications, Volume 2, 100006

Discord.js Dokumentaatio. 2021. Luettu 10.4.2020.
https://discord.js.org/#/docs/main/stable/general/welcome

Laaksonen, S.-M., Laitinen, K., Koivula, M. & Sihvonen, T. 2020. Puhekaverina botti. Lähikuva – Audiovisuaalisen Kulttuurin Tieteellinen Julkaisu, 33 (1), 63–78. https://doi.org/10.23994/lk.91435

Lee, J., Anjos, E. & Satti, S. R. 2021. SJSON: A succinct representation for JSON documents. Information Systems, Volume 97, 101686

MDN Web Docs, Date. n.d. Luettu 23.4.2021
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

Peltomäki, J. 2017. JavaScript-kieli. Uudet ominaisuudet. 1. painos. Helsinki: BoD - Books on Demand.