Ossian Johansson

Botti keskusteluohjelmaan Javalla

Discord-ohjelmointirajapinta ja teknologiat

Botti keskusteluohjelmaan Javalla

Discord-ohjelmointirajapinta ja teknologiat

Ossian Johansson
Opinnäytetyö
Kevät 2020
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma, Internet-palvelut ja digitaalinen media

---

Tekijä: Ossian Johansson
Opinnäytetyön nimi: Botti keskusteluohjelmaan Javalla, Discord-ohjelmointirajapinta ja teknologiat
Työn ohjaaja: Teppo Räisänen
Työn valmistumislukukausi- ja vuosi: Kevät 2020 Sivumäärä: 25 + 3 liitettä

Tämän toiminnallisen opinnäytetyön tarkoituksena oli tuottaa bottisovellus Discord-keskusteluohjelmaan Java-ohjelmointikielellä. Botin ominaisuuksiin kuuluu yleisten web-palveluiden yhdistäminen Discord-keskusteluohjelmaan hakukomentojen avulla, JSON-hakija komento ja muutama yksinkertaisempi peruskomento. Tunnetuista web-palveluista valittiin kaksi, Wikipedia ja YouTube. Java-teknologian lisäksi muita käytettyjä teknologioita ovat Maven, REST, JSON ja WebSocket. Sovellus yhdistettiin Discordiin kolmannen osapuolen JDA-rajapinnan avulla. Sovellusta testattiin Raspberry Pi 3 B pienoistietokoneella, jossa käyttöjärjestelmä toimi Linux Raspbian.

Tuloksena on toimiva bottisovellus, jolla on myös pitkä käyttöaika Raspberry Pi -laitteella. Sovellus on kevyt, ja se sisältää tiivistä ohjelmakoodia. Sovelluksen kaikki komennot toimivat ja antavat halutun tuloksen oikein käytettynä. Sovellukseen voidaan kehittää useita lisäominaisuuksia, tärkeimpänä JSON-hakijaan pohjautuvat toiminnot.

Työn aikana tekijän ohjelmointitaidot karttuivat ja tietämys useista web-palveluihin liittyvistä teknologioista lisääntyi huomattavasti. Nämä asiat antavat hyvän pohjan aloittaa työelämä ohjelmoinnin parissa.

Asiasanat: botti, java, json, rest, discord

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems, Internet Services and Digital Media

---

Author: Ossian Johansson
Title of thesis: Java based bot for a chat program, Discord API and technologies
Supervisor: Teppo Räisänen
Term and year when the thesis was submitted: Spring 2020 Number of pages: 25 + 3 appendices

---

The purpose of this functional thesis was to produce a Java based bot application for the Discord VoIP application. Features of the bot include connecting universal web services to the Discord chat program using search commands, a JSON checker command, and a few simpler basic commands. Two of the well-known universal web services were selected, Wikipedia and YouTube. In addition to Java technology, other technologies used in this work include Maven, REST, JSON, and WebSocket. The application was connected to Discord via a third-party API called JDA. The application was tested on a Raspberry Pi 3 B minicomputer running Linux Raspbian.

The result is a working bot application that also has a long runtime with the Raspberry Pi. The application is lightweight and contains concise program code. All commands in the application work and give the desired result when used correctly. Several additional features can be developed for the application, most importantly functions based on the JSON checker.

During the work, the author's programming skills accumulated, and knowledge of several technologies related to web services increased significantly. These things provide a good foundation to start working life in programming.

Keywords: bot, java, json, rest, discord

SISÄLLYS

1 JOHDANTO

Teksti- ja puhekeskustelut ovat tänä päivänä olennaisesti osa videopelaamista verkossa. Aikoinaan moninpelejä pelattiin lähinnä tietokoneella lähiverkossa tai pelikonsolilla, samassa huoneessa tai ainakin asunnossa. Pelaajien parissa Voice over Internet Protocolia eli VoIP-tekniikkaa ei vielä tarvittu kovin usein. Internetin suosion kasvettua, myös internetissä toimivien verkkopelien julkaisut ja suosio kasvoivat. VoIP-tekniikkaa alettiin tarvita, kun ihmiset alkoivat pelaamaan yhdessä omista kodeistaan käsin. Pelaajien välillä puheen laadun ja viivettömyyden merkitys on tärkeää, ja se korostuu etenkin e-urheilussa. E-urheilun eli elektronisen urheilun suosio onkin kasvanut lähivuosina ja jatkaa kasvuaan. (SEUL 2020, viitattu 26.5.2020.)

Ensimmäisiä pelaajille tarkoitettuja tai pelaajien suosiossa olleita VoIP-ohjelmia olivat esimerkiksi TeamSpeak ja Ventrilo. TeamSpeakissa ja Ventrilossa käyttäjät tarvitsivat kuitenkin oman palvelimen. Palvelin piti joko pystyttää itse tai vuokrata maksua vastaan. Ensimmäisissä VoIP-ohjelmissa oli myös paikottain suurta viivettä. VoIP-puheluissa olevaa viivettä on vuosien saatossa pyritty pienentämään mahdollisimman paljon. Tähän tavoitteeseen sitoutui vuonna 2015 julkaistun Discord VoIP-ohjelman kehittäjäryhmä. Discord ohjelma tarjoaa käyttäjilleen teksti ja puhekeskusteluun kanavia, joissa useat käyttäjät voivat samanaikaisesti kommunikoida. (Liite 1.) Discord kanavalle voi luoda useita puhe- ja tekstikanavia erilaisia tarkoituksia varten. Puhekeskusteluun tarkoitettuja ohjelmia oli jo olemassa vuosia ennen Discord ohjelmaa, mutta useissa ei ollut mahdollisuutta tekstikeskusteluun ja ohjelmissa oli suurempia ongelmia äänenlaadun ja viiveen kanssa.

Videopelaajille suunnatusta markkinoinnista huolimatta Discord ohjelmaa voi hyvin käyttää myös muutkin kuin pelaajat. Palvelu on pääosin kuin mikä tahansa ohjelma, jossa on erilaisten ryhmäkeskustelun mahdollisuus. Discordista on saatavilla myös mobiiliversio, joka on hyvin samanlainen, kuin työpöytäversio. Discord on ilmainen, mutta Discordia on mahdollista kuitenkin tukea liittymällä kuukausimaksulliseen Nitro jäsenyyteen, jolla saa ohjelman käyttöön liittyviä lisäetuja. Nitro jäsenyyden lisäedut eivät kuitenkaan ole olennaisia peruskäytettävyyden kannalta, eivätkä vaikuta tässä opinnäytetyössä käytössä oleviin ominaisuuksiin.

Tässä työssä pyrittiin tuottamaan bottisovellus Discord-ohjelmointirajapintaan Java-ohjelmointikielen avulla. Botin tarkoitus on tuoda lisäominaisuuksia Discordin tekstikanaville, yhdistämällä tunnettuja web-palveluita tekstikentän kautta saataville. Sen lisäksi bottiin luotiin muutamia lisäkomentoja ja myös jatkossa botin lisäkehitystä tukeva JSON-lukija. Seuraavassa kappaleessa esitellään työssä käytetyt teknologiat.

2 KÄYTETYT TEKNOLOGIAT

Tässä kappaleessa esitellään opinnäytetyössä käytettyjä teknologioita yleisesti. Tämä kappale antaa paremman kuvan lukijalle, miksi kyseiset teknologiat ja standardit ovat käytössä tässä opinnäytetyössä. Alaotsikoissa esitellään tiivistetysti Java, Maven, REST, JSON, WebSocket, Raspberry Pi 3 ja siinä käytettävä Linux Raspbian.

2.1 Java

Java on ohjelmointikieli ja tietojenkäsittelyalusta, jonka Sun Microsystems julkaisi ensimmäisen kerran vuonna 1995 (Oracle 2020a, viitattu 24.5.2020). Oracle Corporation osti myöhemmin Sun Microsystemsin vuonna 2010, ja Oracle hallitsee nyt Java teknologiaperheen kehitystä. Javan etuina ovat sen turvaominaisuudet, ja että se on arkkitehtuurillisesti neutraali. Toisin kuin useat muut ohjelmointikielet, Javan avulla voi kirjoittaa ohjelman joka toimii millä tahansa käyttöjärjestelmällä. Java ei suorita ohjelmaa tietokoneella suoraan, vaan ajaa ohjelman JVM:n eli Java-virtuaalikoneen kautta. (Farrel 2014, 10.) Java-virtuaalikone on toteutettu käyttöjärjestelmän päällä ajettavaksi. Javaohjelmat käännetään tavukoodiksi, ja virtuaalikone kääntää koodin konekielelle tiettyä käyttöjärjestelmää ja laittetta varten. Java-virtuaalikone hoitaa myös Javalle ominaisen automaattisen roskienkeräyksen, joka pyrkii vapauttamaan muistia. Roskienkeräys tunnistaa automaattisesti, että onko olio enää käytössä tai viitattuna ohjelman muihin osiin. Käyttämätön tai viittaamaton olio poistetaan käytöstä. (Oracle 2020b, viitattu 25.5.2020.)

Java on olio-ohjelmointikieli. Olio-ohjelmointi keskittyy olioihin. Oliot ovat asioita tai kokonaisuuksia, joita käytetään ohjelmassa. Olioilla voi olla omat ominaisuutensa ja toiminnot. (Farrel 2013, 6.) Olioiden perintä eli inheritance on oleellinen osa olio-ohjelmointia. Toinen olio voi periä toiselta sen määritykset. Tämä käy järkeen, kun usealla oliolla tarvitsee olla samoja toimintoja ja ominaisuuksia. (Farrel 2013, 258.) Olioita voisi olla esimerkiksi koira ja kissa. Kummatkin ovat eläimiä ja niillä on joitakin samanlaisia toimintoja ja ominaisuuksia. Tällöin esimerkiksi erillisen eläin luokan tekeminen kannattaa duplikaatin ohjelmakoodin välttämiseksi.

Olio-ohjelmoinnissa pyritään luomaan alhainen abstraktiotaso, joka tapahtuu jakamalla ohjelma mahdollisimman moneen olioon. Näin ohjelman semanttinen sisältö lisääntyy ohjelmoijan näkökulmasta. Toisena yleisenä ohjelmointiparadigmana olio-ohjelmointiin verraten voidaan pitää proseduraalista ohjelmointia, jossa keskitytään toimintojen eli niin sanottujen aliohjelmien luomiseen. Proseduraalisessa ohjelmoinnissa olioita tai instansseja ei luoda.

PYPL (2020, viitattu 25.5.2020) sivuston perusteella, joka mittaa ohjelmointikielten suosiota Google hakuihin pohjautuen, Java on yhä toiseksi suosituin ohjelmointikieli, Python kielen jälkeen. PYPL sivuston arvio ei tietenkään kerro koko totuutta. Oraclen omien sivujen mukaan Java on maailman käytetyin ohjelmointikieli (Oracle 2020c, viitattu 26.5.2020), ja tämä luultavasti pitää paikkansa, kun ottaa huomion sen laajuuden yrityskäytössä.

2.2 Maven

Apache Maven on käännösautomaatiosovellus ja työkalu, pääasiallisesti Java-projekteja varten. Maven pitää huolta ohjelmointiprojektin apukirjastojen lataamisesta ja hallinnasta. Maven hakee apukirjastot *pom.xml* -tiedoston perusteella, jonne käyttäjä asettaa tarvitsemansa kirjaston nimen ja version. (The Apache Software Foundation 2020, viitattu 2.6.2020.)

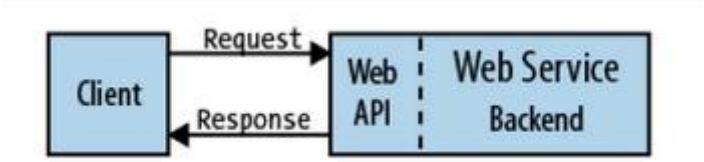
2.3 REST API

World Wide Web alkuaikoina ei ollut vielä tehty selkeitä arkkitehtuurimalleja web-palveluiden luomiseksi (Massé 2012, 1). Tänä päivänä REST eli Representational State Transfer ja sen ohjelmointirajapinta ilmaisevat yksinkertaisesti yhtä tärkeintä internetin arkkitehtuuristandardeista. REST-standardi määrittelee tiettyjä arkkitehtonisia rajoituksia web-palveluiden luomiseen. (katso kuvio 1.) Rajoituksiin ei perehdytä tässä työssä.

1. Client-server
2. Uniform interface
3. Layered system
4. Cache
5. Stateless
6. Code-on-demand

KUVIO 1. REST arkkitehtoniset rajoitukset (Massé 2012, 2)

REST-standardiin perustuvat ohjelmointirajapinnat antavat joukon tietoja ja toimintoja helpottaakseen tietokoneohjelmien välistä vuorovaikutusta ja mahdollistaa niiden vaihtaa tietoja keskenään. Kuvio 2 kuvastaa web-palveluiden ohjelmointirajapinnan rakennetta. REST-standardiin kuuluu erilaisia metodeja, kuten GET, POST, DELETE ja niin edelleen. REST-palveluissa usein käytetty tiedostomuoto on JSON.



KUVIO 2. Web API (Massé 2012, 6)

2.4 JSON

JavaScript Object Notation eli JSON on kevyt standardisoitu tiedostomuoto jäsenneltyyn tiedonvälitykseen, joka juontaa alkunsa JavaScript-ohjelmointikielestä. JSON:illa on natiivi JavaScript tuki, mutta JSON on siitä huolimatta JavaScriptistä riippumaton. JSON on lähes kaikkialla käytössä ja sillä on luonteva, muihin ohjelmointikieliin verrattava syntaksi. (Massé 2012, 7-8). JSON:ista on yksinkertaisuutensa ja helpon käytettävyytensä takia tullut standardi nykyaikaisten ohjelmointirajapintojen parissa (Jin, Sahni & Shevat 2018, 11).

JSON muodostuu kahdesta rakenteesta: nimien ja arvo parien kokoelmasta ja niiden jäsennellystä listasta. JSON-objekti on joukko nimi ja arvo pareja ilman järjestystä. (JSON 2020, viitattu 26.5.2020.) JSON voi myös sisältää taulukoita ja sisäkkäisiä JSON-objekteja. (katso kuvio 3.)

```
{
  "data": [{
    "type": "articles",
    "id": "1",
    "attributes": {
      "title": "JSON:API paints my bikeshed!",
      "body": "The shortest article. Ever.",
      "created": "2015-05-22T14:56:29.000Z",
      "updated": "2015-05-22T14:56:28.000Z"
    },
    "relationships": {
      "author": {
        "data": {"id": "42", "type": "people"}
      }
    }
  }],
  "included": [
    {
      "type": "people",
      "id": "42",
      "attributes": {
        "name": "John",
        "age": 80,
        "gender": "male"
      }
    }
  ]
}
```


KUVIO 3. Esimerkki JSON-tiedostosta internetselaimesta katsottuna

2.5 WebSocket

WebSocket-protokolla mahdollistaa kaksisuuntaisen tiedonsiirron asiakkaan välillä, joka käyttää epäluotettua koodia hallitussa ympäristössä, etäisäntään, joka on hyväksynyt tiedonsiirron kyseisestä koodista. Tämän tekniikan tavoitteena on tarjota mekanismi alunperin selainpohjaisille sovelluksille, jotka tarvitsevat kaksisuuntaista viestintää palvelimien kanssa, ja joka ei pohjautuisi useiden HTTP-yhteyksien avaamiseen. WebSocket tarjoaa yksinkertaisen TCP-yhteyden molempiin suuntiin. Protokollassa on kaksi osaa: kättely ja tiedonsiirto. Avauskättely lähtee asiakkaan sovelluksesta ja sisältää WebSocket-avaimen. Jos etäisäntä hyväksyy avaimen, kättely tapahtuu, ja tiedonsiirto voi alkaa. (Fette & Melnikov 2011, viitattu 2.6.2020.)

2.6 Raspberry Pi 3 B ja Linux Raspbian

Raspberry Pi 3 B on pienoistietokone, jossa ARM-tyypin mikroprosessori (Liite 2). Sillä on noin vuoden 2013 keskiverto älypuhelimen laskentateho. Raspberry Pi tietokoneet ovat suosittuja mikrokontrolleriharrastajien parissa ja IoT-laite käytössä. On suosittua luoda erilaisia IoT-projekteja

Raspberry Pi tietokoneita käyttäen, kuten vaikka sääasemia, internetradioasemia tai robotteja. Tässä työssä testattiin Raspberry Pi 3 B:tä bottisovelluksen palvelimena. Pienoistietokoneen etuna on pieni virrankulutus.

Linux Raspbian, nykyiseltä nimeltää Raspberry Pi OS, on Linux Debian jakeluun pohjautuva käyttöjärjestelmä, joka on tarkoittettu Raspberry Pi laitteita varten. Java-virtuaalikone suorittaa bottisovelluksen ARM-prosessorilla ja Linuxilla ajettavaksi.

3 DISCORD API

Discord API eli Discord-ohjelmointirajapinta mahdollistaa bottikäyttäjien luomisen Discord Developer Portalin kautta. Discord Developer Portaliin pääsemiseksi tarvitsee ensin tavallisen Discord tilin. Developer Portalissa on mahdollisuus luoda myös pelkkä sovellus, joka on riisuttu versio botista. Bottikäyttäjiä ja sovelluksia voi myös luoda useita eri tarkotuksiin. Bottikäyttäjän toiminta on monelta osin samanlainen, kuin tavallisen käyttäjän. Botille voi asettaa helposti oman kuvakkeen, kuten käyttäjälle. Oikeuksien antaminen botille kanavalla toimii myös samalla tavoin, kuin tavalliselle käyttäjälle. Tässä mielessä botin graafisen käyttöliittymän kehittämiseen ei siis tarvitse keskittyä. Developer Portalissa luodaan botille kutsulinkki, jonka avatessa botti voidaan selaimen kautta kutsua halutuille kanaville. Kutsulinkkiä luodessa voidaan jo halutessa asettaa botille valtuuksia valmiiksi.

Kun bottikäyttäjä on luotu, sillä ei ole minkäänlaista toiminnallisuutta. Bottikäyttäjä näkyy offline-tilassa kanavilla, joihin se on kutsuttu. Bottikäyttäjään tarvitsee yhdistää varsinainen bottisovellus. On huomioitava, että bottisovelluksen ylläpitämiseen tarvitaan kuitenkin siihen soveltuva palvelin. Discord Developer Portal tarjoaa botille tokenin, jolla oman sovelluksen voi yhdistää Discordiin. Discordissa on tällä hetkellä käytössä OAuth 2.0, joka on avoin pääsyvaltuutusten standardi (OAuth Community Site 2019, viitattu 13.3.2020).

Discord API perustuu kahteen ydinkerrokseen, HTTPS/REST-rajapintaan, joka on tarkoitettu yleisiin toimintoihin, ja turvalliseen ja pysyvään WebSocket yhteyteen. Tavanomaisin Discord API:n käyttötapa on palvelun tarjoaminen edellä mainitun OAuth2-rajapinnan kautta. (Discord Developer Portal 2020, viitattu 2.6.2020)

Discord API ei kuitenkaan tarjoa valmista rajapintaa eri ohjelmointikielille suoraan käytettäväksi. Tähän tarvitsee oman rajapinnan, joka on suunniteltu valitulle ohjelmointikielelle. Tähän työhön valittiin alustaksi seuraavassa kappaleessa esiteltävä JDA.

4 JDA

JDA eli Java Discord API on niin sanottu wrapper Discordin ohjelmointirajapinnan Java ohjelmointia varten. JDA on avoimen lähdekoodin projekti, jolla on Apache 2.0 lisenssi. (JDA 2020, viitattu 3.6.2020.) JDA tarjoaa yksinkertaistetut Java-funktiokutsut Discordiin WebSocketin kautta. JDA:a voi siis käyttää oman sovelluksen alustana, sillä se helpottaa sovelluksen tekemistä poistamalla tarpeen ohjelmoida yhteys Discordiin. JDA lisättiin ohjelmaprojektiin apukirjastona Mavenin avulla.

JDA hoitaa sovelluksen yhdistämisen Discordiin (katso kuvio 4.) ja tarjoaa kaikki Discord API:n tarjoamat metodit laajennattavaksi Java-sovellukselle. JDA myös ylläpitää itseään ja yhdistyy Discordiin automaattisesti jos yhteys katkeaa. Myöskään bottisovelluksen mahdolliset virheet ajossa, eivät kaada koko ohjelmaa.

```
[main] INFO JDA - Login Successful!
[JDA MainWS-ReadThread] INFO WebSocketClient - Connected to WebSocket
[JDA MainWS-ReadThread] INFO JDA - Finished Loading!
[TestServer][bottikanava] User: !ver
[TestServer][bottikanava] TestBot: This is Testbot version 0.1.3!
```

KUVIO 4. JDA:n toiminta konsolinäkymässä käynnistettäessä

Tämän työn sovelluksessa käytetään *onMessageReceived* -metodia, joka lukee käyttäjän viestin, ja suorittaa komentoja, jos viestin ehdot täyttyvät ohjelmoidun bottisovelluksen *Commands*-luokassa. (katso kuvio 5.)

```
23 public class Commands extends ListenerAdapter {
24
25     @Override
26     public void onMessageReceived(MessageReceivedEvent event) {
27         if (event.isFromType(ChannelType.PRIVATE)) {
28             System.out.printf("[PM] %s: %s\n", event.getAuthor().getName(), event.getMessage().getContentDisplay());
29         } else {
30             System.out.printf("[%s][%s] %s: %s\n", event.getGuild().getName(), event.getTextChannel().getName(),
31                     event.getMember().getEffectiveName(), event.getMessage().getContentDisplay());
32         }
33
34         String[] command = event.getMessage().getContentDisplay().split(" ");
35         String msg = "";
```

KUVIO 5. Commands-luokan alku

5 BOTTISOVELLUS

Bottisovelluksen ohjelmointikielenä on Java. Java on valittu ohjelmointikieleksi puhtaasti halusta kehittyä kyseissä ohjelmointikielessä ja olio-ohjelmoinnissa. Botin tarkoituksena on tarjota yksinkertaisia komentoja käyttäjille, hakumahdollisuudet yleisiin internetpalveluihin suoraan Discord tekstikentässä ja luoda kehitysmahdollisuuksia JSON-tiedostotyypin avulla. Discord yhdistää jo itsessään puhe ja tekstikeskustelut samaan ohjelmaan, ja botin tarkoituksena on lisätä muita palveluita suoraan Discordista käytettäväksi.

Bottisovelluksen main-metodissa luodaan uusi JDA-olio, joka toimii sovelluksen pohjan ytimenä. Sen lisäksi luodaan uusi olio Commands-luokasta, joka sisältää sovelluksen komennot. Commands-luokka jatkaa JDA:n ListenerAdapter-luokkaa, joka suorittaa Discordin tapahtumien tarkkailun. (katso kuvio 6.)

```
1  package io.aweseean.personal.discord.bots.oldbot;
2
3  import net.dv8tion.jda.core.JDA;
4  import net.dv8tion.jda.core.JDABuilder;
5  import net.dv8tion.jda.core.exceptions.*;
6  import net.dv8tion.jda.core.AccountType;
7  import net.dv8tion.jda.core.hooks.ListenerAdapter;
8
9  import javax.security.auth.login.LoginException;
10
11 public class OldBot extends ListenerAdapter {
12
13     public static void main(String[] args) throws LoginException, RateLimitedException, InterruptedException {
14
15         JDA jda = new JDABuilder(AccountType.BOT).setToken(Ref.TEST_TOKEN).buildBlocking(); // Remember token!
16         jda.addEventListener(new Commands());
17         // jda.addEventListener(new AudioCommands()); // ---- not in use currently
18     }
19 }
20
```

KUVIO 6. Sovelluksen main-funktio ja JDA-olion käyttöönotto

5.1 Komennoista yleisesti

Botissa on 10 komentoa. Kaikki komennot on ohjelmoitu itse. Koodinpätkien lisäksi komentojen tuloksia esitetään kuvankaappauksina Discordin työpöytäversiosta. Kuvankaappauksista näkee myös komentojen kirjoitusmuodon.

Komennon perusosan jälkeen vaaditaan useassa komennossa lisäparametrejä. Ne asetetaan välilyönnin jälkeen. Discordissa on omia komentoja, jotka käyttävät etuliitteenä vinoviivaa (/).

Vinoviiva on yleinen etuliite komennoille. Sekaannuksia välttäen, botin komennot käyttävät etuliitteenä huutomerkkiä (!). Ääkköset toimivat tarvittavissa hakukomennoissa. Myös useat virhesyötöt, kuten parametrin puuttuminen, on otettu huomioon komentojen luomisessa.

5.2 Peruskomennot

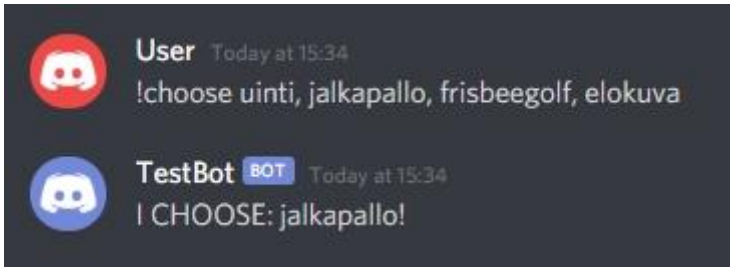
Sovellukseen kehitettiin muutamia peruskomentoja käyttäjien avuksi ja iloksi. Käytännöllisyyksistään huolimatta, nämä komennot ovat ohjelmoinniltaan lyhyitä ja yksinkertaisia, joten ne esitellään lyhyesti alaotsikoituina tämän otsikon alla.

5.2.1 Noppa

Noppakomennolla (!roll) voi pyytää botilta erilaisia satunnaislukuja väliltä 1-1000. Pelkän komennon kirjoittaminen ilman toista parametriä antaa tuloksen väliltä 1-6, kuten arpakuutio. Muun kuin numeron asettaminen toiseksi parametriksi, lähettää erityisen viestin kanavalle, kuten myös yritys arpoa luku 0. Noppakomennon kehittäminen lähti ajatuksesta tukea mahdollista pöytäroolipelien pelaamista Discord kanavalla. Noppakomennolla voi myös suorittaa arvontoja.

5.2.2 Satunnainen valinta

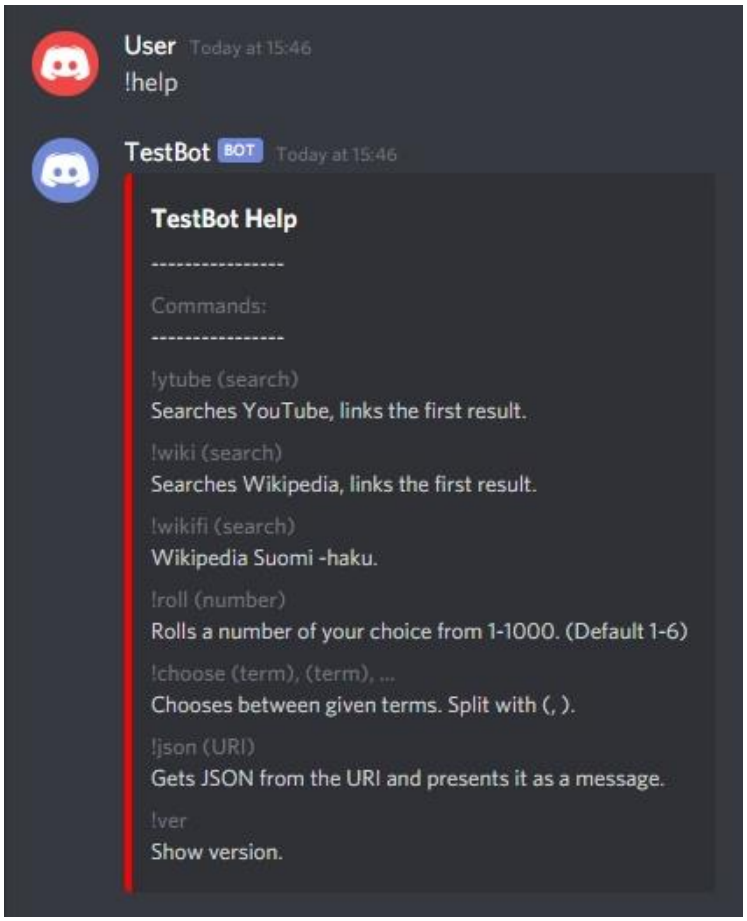
Satunnainen valinta- komennolla (!choose) voi antaa botin valita satunnaisesti käyttäjien antamien arvojen väliltä. (katso kuvio 7.) Parametrit täytyy erottaa pilkun ja välin avulla (,). Muuten ohjelma ei tunnista annettuja arvoja toisistaan. Käyttäjän on mahdollista syöttää myös useita sanoja yhteen vaihtoehtoon. Jos käyttäjä ei osaa esimerkiksi päättää miten lähteä viettämään vapaa-aikaansa, voi käyttäjä asettaa vaihtoehdot komennon parametreiksi.



KUVIO 7. Satunnainen valinta

5.2.3 Help-komento

Help-komento (!help) näyttää käyttäjälle botin käytettävät komennot viestinä. (katso kuvio 8.) Käyttäjä ei välttämättä aina tiedä tai muista kaikkia botin tarjoamia komentoja. Tällöin komento tulee tarpeeseen. Komento käyttää Discordin embedded kehystä, jolla voi korostaa viestejä.



KUVIO 8. Help-komennon vastaus

5.2.4 Versiokomento

Versiokomento (!ver) näyttää botin version. (katso kuvio 9.) Komento on kätevä kun halutaan nopeasti tarkistaa on onko botti yhä kanavalla ja käytettävissä, käyttämättä paljon tilaa tekstikanavalla.

KUVIO 9. Versiokomennon vastaus

5.3 Wikipedia haku -komennot

Wikipedia hakuja on kaksi: !wiki ja !wikifi. Komento !wiki etsii Wikipedia sivua englannin kieliseltä en.wikipedia.org sivustolta, ja !wikifi suomenkieliseltä fi.wikipedia.org sivustolta. Komentojen ohjelmointi muistuttaa lähes täysin toisiaan. Hakukomennot käyttää hyväkseen Wikipedia sivustolla olevan *special search* -hakukentän web-osoitetta, (katso kuvio 10) joka ohjaa käyttäjän suoraan ensimmäiseen hakutulokseen. Special search -hakukentän takia erityistä REST-ohjelmointia ei näiden komentojen osalta tarvittu tehdä. Käyttäjän tulee kirjoittaa hakusana mahdollisimman oikein, jotta oikea hakutulos saavutettaisiin. Botti asettaa vain käyttäjän toisen parametrin web-osoitteen hakukohtaan. Vastauksena botti lähettää ensimmäisen hakutuloksen linkkinä sille kanavalle tai käyttäjälle, josta komento on lähetetty. (katso kuvio 11.) Linkin muotoilun hoitaa Discord.

```
114         if (command[0].equalsIgnoreCase("!wiki")) {
115             if (command.length < 2) {
116                 msg = "Please enter a search parameter...";
117                 event.getChannel().sendMessage(msg).queue();
118             } else {
119                 StringBuilder sb = new StringBuilder();
120                 for (int i = 1; i < command.length; i++) { // Check the amount of parameters and split them with
121                                                            // underscore
122                     if (i > 1) {
123                         sb.append("_");
124                     }
125                     String param = command[i];
126                     sb.append(param);
127                 }
128                 msg = "https://en.wikipedia.org/w/index.php?search=" + sb.toString() + "&title=Special%3ASearch";
129                 event.getChannel().sendMessage(msg).queue();
130             }
131         }
```

KUVIO 10. Wikipedia haku -komennon koodinpätkä

KUVIO 11. Wikipedia.fi hakutulos.

5.4 YouTube-hakukomento

YouTube-hakukomennon periaate on sama kuin Wikipedia hakujen. Käyttäjä hakee komennolla YouTubesta haluamaansa videota, ja botti lähettää ensimmäisen hakutuloksen linkkinä kanavalle tai käyttäjälle. Myös tässä komennossa hakusanojen olisi oltava mahdollisimman spesifit, jotta oikea video löytyy ensimmäisenä. YouTubella ei kuitenkaan ole olemassa samantyyppistä *special search* -hakukenttää, kuten Wikipedian web-sivulla, joten Google Developer palvelut oli otettava käyttöön kehityksessä. Käyttäjä joka haluaa tehdä REST-pyyntöjä YouTubeen, on ensin hankittava Google Developer tili. Kun tili on käytössä, käyttäjä voi pyytää niin sanottua avainta Googlen ohjelmointirajapintaan pääsemiseksi. Google ja käyttäjä määrittelevät mihin avainta käytetään ja minne sillä on pääsyoikeudet.

Googlella on oma JSON-kirjasto Javaa varten, mutta tässä opinnäytetyössä käytetään kuitenkin alkuperäistä *org.json* -kirjastoa. Kutsuttakoon org.json -kirjastoa tässä tekstissä toisella nimellään

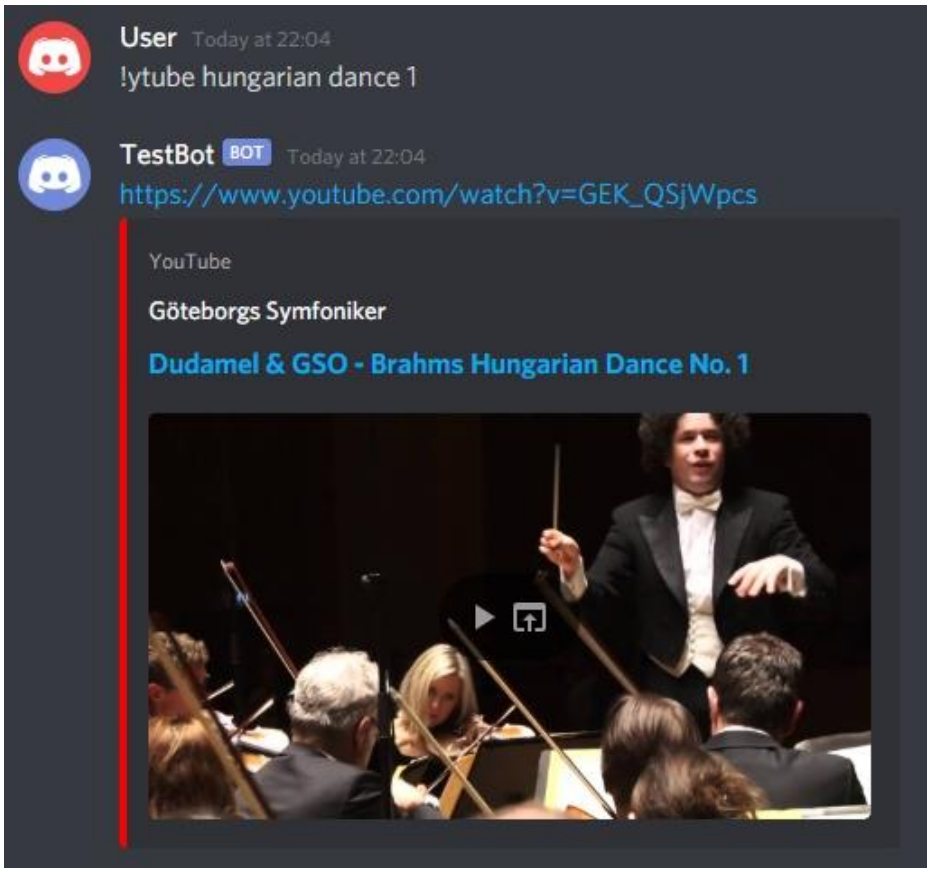
JSON-Java. JSON-Java mahdollistaa JSON-tiedostojen käsittelyn Java-ohjelmointikielellä. Se sisältää tarvittavat luokat ja metodit JSON käsittelyyn.

Komennossa käytetään YouTube-rajapinnan Search-metodia. Search on itseasiassa REST GET -metodi, johon lisätään parametrejä URI-osoitteen perään. (katso kuvio 12.) Parametri *maxResults*, määrittelee kuinka monen videon JSON-arvot haku palauttaa. Tässä tapauksessa tarvitaan vain yksi, eli ensimmäinen hakutulos. Käyttäjän kirjoittama hakuparametri asetetaan oikeaan kohtaan osoitteessa ja samoin Googlen API-avain. Vastauksena saadaan JSON-tiedosto.

```
"https://www.googleapis.com/youtube/v3/search?part=id&maxResults=1"
+ "&regionCode=FI&order=relevance&q="
        + ue + "&type=video&key=" + Ref.GOOGLE_APIKEY);
```

KUVIO 12. REST-haun URI osoite.

Ohjelma parsii JSON-tiedostosta tarvittavan *videoId* JSON-parin. Parin arvo asetetaan sitten tavallisen YouTube-linkin perään ja lähetetään kanavalle. Videota voi katsoa suoraan Discordista, avaamatta selainta. (katso kuvio 13.)

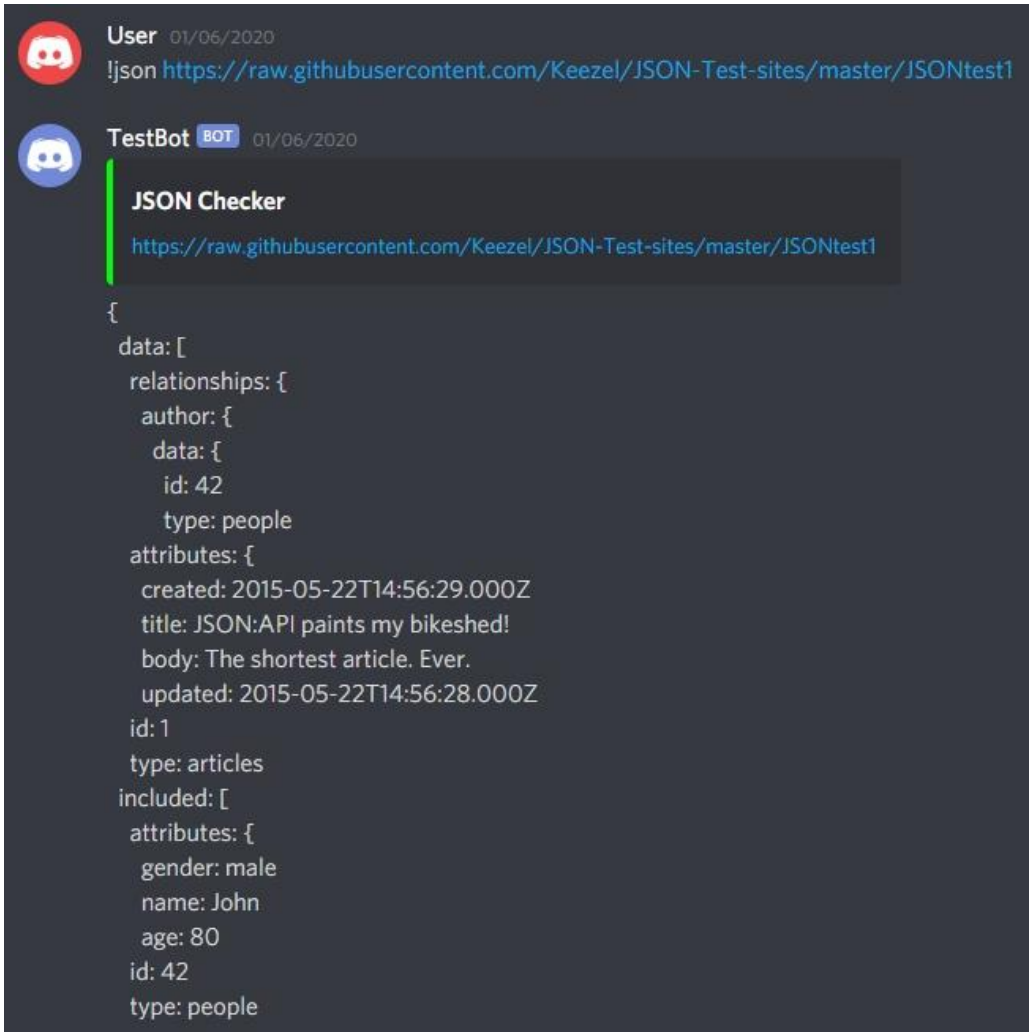


KUVIO 13. YouTube-haun tulos

5.5 JSON-hakija

JSON-hakijan tarkoituksena on hakea JSON-tiedosto URI-osoitteen perusteella ja eritellä se Discordin viestikentässä. Edellisessä kappaleessa esitelty JSON-Java on käytössä tässäkin toiminnossa. Hakija on hyvä lyhyiden JSON-tiedostojen näyttämiseksi. (katso kuvio 14.) Alle 50:n nimi ja arvo -parin pituiset tiedostot näkyvät vielä yhdellä sivulla Discordin työpöytäversiossa.

Toiminto erittelee myös sisäkkäiset JSON-objektit ja sisäkkäiset taulukot. Mielenkiintoiseksi toiminnon tekee myös se, että se erittelee nimi ja arvo -parit lisäämällä välilyönnit parien eteen, samaan tapaan miten ne näkyvät internetselaimella katsottuna. JSON-parien erittely tapahtuu rekursiivisilla toistorakenteilla. Ohjelmakoodi tarkastaa onko JSON-pari uusi JSON-objekti, JSON-taulukko taikka tavallinen JSON-pari. JSON-hakija sijaitsee omassa JsonChecker-luokassaan. Osa JsonChecker-luokan ohjelmakoodista on esillä liitteissä (liite 3).



KUVIO 14. JSON-hakijan tulos. Kyseessä on sama esimerkkitiedosto, kuin kuviossa 3.

6 JOHTOPÄÄTÖKSET JA BOTIN KEHITTÄMINEN

Tavoitteena oli luoda bottisovellus pelaajien ja muiden Discord-käyttäjien käyttöön. Jos käyttäjä ymmärtää jokseenkin botin kommennot ja kuinka niitä käytetään, on sovellus lähes tulkoon käyttäjän näkökulmasta ohjelmointivirheistä vapaa. Bottisovelluksen toiminta on myös nopeaa. Pyrkimyksenä oli myös tehdä sovelluksesta mahdollisimman kevyt. Tähän pyrittiin luomalla komennoista mahdollisimman lyhyitä koodin pätkiä, kuitenkaan toiminnallisuutta vaarantamatta. Varsinainen itseohjelmoitu osuus on vain hieman yli 400 riviä pitkä. Sovellus vie tilaa alle 24 megatavua, mutta näin mitattu versio sisältää käytöstä poissa olevia ja kehitteillä olevia ohjelman osia. Sovellukseen tuli silti paikottain koodin toistoa, jota olioiden ja interface -luokkien lisääminen vähentäisi.

Useiden muiden tunnettujen web-palveluiden lisääminen sovellukseen olisi mahdollista. Samantyyppisiä hakukomentoja voisi tehdä vaikka musiikkipalveluista tai kuvahauista. Lisäksi monia peruskomentoja pystyy helposti lisäämään tarvittaessa.

JSON-hakija erittelee nimi ja arvo -parit lisäämällä oikean määrän välilyöntejä niiden eteen. Tätä ominaisuutta voisi käyttää myös toisella tavalla. Välilyönnit voisi helposti korvata vaikka tallentamalla nimet ja arvot, niiden sijainnin mukaan tiedostossa. Näin voidaan tallentaa tai tulostaa informaatio eri muotoon, unohtamatta niiden rakennetta ja suhdetta. Tällä tavoin JSON-hakija ei ole pelkästään JSON-tiedoston tulostaja, vaan moninpuolisesti kehitettävä osa ohjelmaa.

JSON-hakijan pohjalta voisi kehittää monenlaisia toimintoja. Tietokonepelaajille voisi tehdä esimerkiksi komennon, jossa hakemalla oman käyttänimen tietyssä pelissä, sovellus lähettää kanavalle pelaajan statistiikkaa kyseisestä pelistä. Useissa verkkopeleissä tällaista informaatiota jaetaan REST-palveluna ainakin jossakin muodossa. Toisenlainen komento voisi olla vaikka urheiluottelun tilannekatsaus. Botti lähettäisi maalitilanteen ja peliajan kanavalle tyylitellysti.

Botti on ollut Raspberry Pi 3 B tietokoneen toimiessa palvelimena käynnissä useita kuukausia putkeen, ja se on yhä vastannut komentoihin. Bottisovellus voisi toimia myös pienemmällä Raspberry Pi Zerolla. Jos pienpalvelin ideaa vie vielä pidemmälle, Raspberry Pi Zeroa varten voisi tuottaa sähköä pienellä aurinkopaaneelilla, jolloin se voisi olla lähes itsensä ylläpitävä.

Sovelluksen tuottaminen antoi minulle paljon lisää ohjelmointitaitoa ja tietoja useista teknologioista ja niiden käytöstä. Ongelmien ratkaisu ja uusien toimintojen tekemistavat ovat mielestäni kehittyneet huomattavasti, työn aloitukseen verrattaen. Mielestäni työ on onnistunut, sillä tavoite saavutettiin, sovellus toimii hyvin ja sen kehittäminen voi jatkua tulevaisuudessa.

LÄHTEET

Suomen elektronisen urheilun liitto (SEUL ry). 2020. Mitä on e-urheilu? Viitattu 26.5.2020, https://seul.fi/mita-on-e-urheilu/

Oracle Corporation. 2020a. What is Java and why do I need it? Viitattu 24.5.2020, https://java.com/en/download/faq/whatis_java.xml

Farrell, J. 2014. Java Programming. 7th edition. Cengage Learning.

Oracle Corporation. 2020b. Java Garbage Collection Basics. Viitattu 25.5.2020, https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html

Farrell, J. 2013. An Object-Oriented Approach to Programming Logic and Design. 4th edition. Cengage Learning.

Oracle Corporation. 2020c. Java Software | Oracle Suomi. Viitattu 26.5.2020, https://www.oracle.com/fi/java/

The Apache Software Foundation. 2020. Maven - Welcome to Apache Maven. Viitattu 2.6.2020, https://maven.apache.org/

Carbonnelle, P. 2019. PYPL PopularitY of Programming Language index. Viitattu 25.5.2020, http://pypl.github.io/PYPL.html

Massé, M. 2012. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. O'Reilly.

Jin, B., Sahni, S. & Shevat, A. 2018. Designing Web APIs. O'Reilly.

JSON. 2020. Viitattu 26.5.2020, https://www.json.org/json-en.html

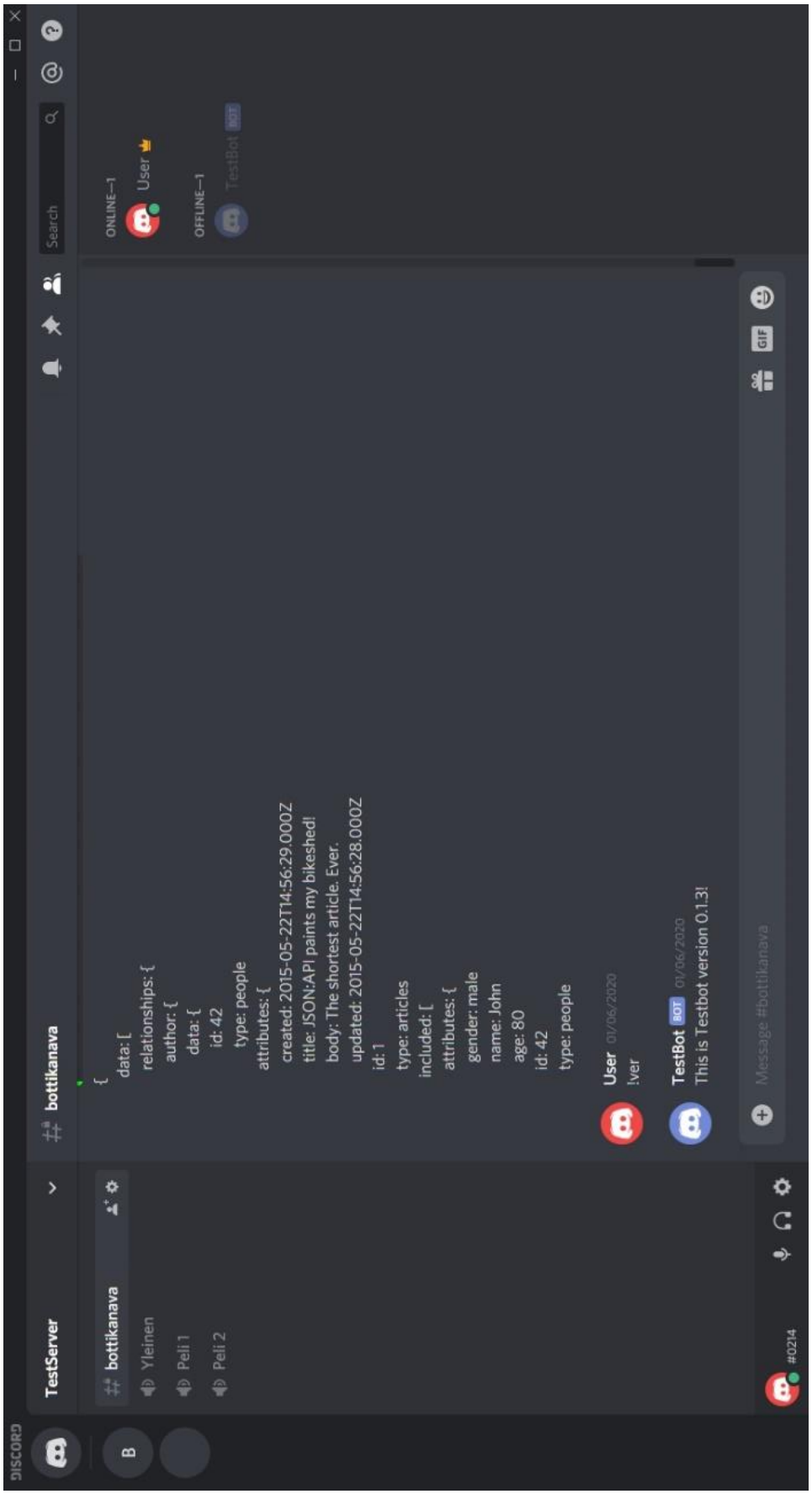
Fette, I. Melnikov, A. 2011. RFC 6455 - The WebSocket Protocol. Viitattu 2.6.2020, https://tools.ietf.org/html/rfc6455

OAuth Community Site. 2020. Viitattu 13.3.2020, https://oauth.net/2/

Discord Developer Portal. 2020. Discord Developer Portal — Documentation — Reference. Viitattu 2.6.2020, https://discord.com/developers/docs/reference

JDA. 2020. Viitattu 3.6.2020, https://github.com/DV8FromTheWorld/JDA

DISCORD KANAVANÄKYMÄ (SIVUTTAIN) LIITE 1



OSA JSONCHECKER-LUOKASTA LIITE 3

```
18    public void checker(JSONObject jsonObject, JSONArray keys) {
19        // key + value tävtyy selvittää, nested tai array myös
20        for (int i = 0; i < keys.length(); ++i) {
21
22            String key = keys.getString(i); // Here's your key
23
24            String value = jsonObject.optString(key); // Here's your value
25
26            if (value.startsWith("{")) {
27                System.out.println(value);
28                System.out.println(sb + key);
29                System.out.println("alkuspace " + spaceCount);
30                msgBuilder.append(sb + key + ": {\n" + initSpc);
31                System.out.println(sb + key);
32                spaceCount++;
33                spaceCount++;
34                System.out.println("oldcountti: " + oldCount);
35                spacer();
36                System.out.println("loppuspace " + spaceCount);
37                System.out.println(key);
38                String nested = jsonObject.optString(key); // Try creating new JSON object
39                JSONObject nestedJson = new JSONObject(nested);
40                JSONArray nestedKeys = nestedJson.names();
41                JSONObject nestedNames = new JSONObject();
42                nestedNames.toJSONArray(nestedKeys);
43                checker(nestedJson, nestedKeys);
44            } else if (value.startsWith("[")) {
45                inArray = true; // Boolean here, true when array
46                msgBuilder.append(sb + key + ": [\n" + initSpc);
47                System.out.println(sb + key + "[");
48                spaceCount++;
49                spaceCount++;
50                spacer();
51                String arrayed = jsonObject.optString(key);
52                JSONArray arrayJson = new JSONArray(arrayed);
53                JSONArray arrayedKeys = new JSONArray();
54                arrayLength = arrayJson.length();
55                JSONObject arrayedNames = new JSONObject();
56                for (int k = 0; k < arrayJson.length(); ++k) {
57                    arrayCount = k;
58                    arrayLength--;
59                    if (arrayJson.isNull(k)) {
60                        msgBuilder.append(sb + "null\n" + initSpc);
61                        continue;
62                    } else {
63                        arrayedJson = arrayJson.getJSONObject(k);
64                    }
65                    arrayedKeys = arrayedJson.names();
66                    arrayedNames.toJSONArray(arrayedKeys);
67                    checker(arrayedJson, arrayedKeys);
68                }
```