OULUN AMMATTIKORKEAKOULU

DISCORD-MUSIIKKIBOTIN KEHITTÄMINEN PYTHON-KIELELLÄ

Niko Kolehmainen & Miikka Tyvelä
Opinnäytetyö (AMK)
Kevät 2025
Tietotekniikan tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
Ohjelmistokehityksen suuntautumisvaihtoehto

Tekijät: Niko Kolehmainen & Miikka Tyvelä
Opinnäytetyön otsikko: Discord-musiikkibotin kehittäminen Python-kielellä
Työn ohjaaja: Juha-Matti Huusko
Työn valmistumislukukausi ja -vuosi: Kevät 2025
Sivumäärä: 39

Tämän opinnäytetyön tavoitteena oli kehittää Discord-palvelimelle musiikkibotti Python-ohjelmointikielellä. Botti kykenee toistamaan käyttäjien pyytämiä kappaleita Youtube- ja SoundCloud-suoratoistopalvelusta, mahdollistaen musiikin kuuntelun suoraan palvelimen keskusteluympäristössä. Botti on varustettu sulautetulla mediasoittimella, joka parantaa käyttäjäystävällisyyttä.

Työssä tutkittiin Discordin API:n hyödyntämistä botin toiminnallisuuksien luomisessa ja palvelimen reaktioiden automatisoinnissa. Lisäksi projekti keskittyi botin suorituskyvyn optimointiin ja äänenlaadun ylläpitämiseen reaaliaikaisessa käytössä, jolloin botin kyky vastata käyttäjien pyyntöihin sujuvasti korostui.

Tulosten perusteella botti pystyi toimimaan tehokkaasti ja ylläpitämään hyvän äänenlaadun, vaikka se oli kytketty reaaliaikaiseen ympäristöön. Johtopäätöksenä voidaan todeta, että huolellinen optimointi ja API:n ominaisuuksien hyödyntäminen mahdollistavat suorituskykyisen ja käyttäjäystävällisen musiikkibotin kehittämisen Discord-ympäristöön.

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Information Technology
Option of Software Development

Author: Niko Kolehmainen & Miikka Tyvelä
Title of thesis: Developing a Discord Music Bot with Python
Supervisor: Juha-Matti Huusko
Term and year when the thesis was submitted: Spring of 2025
Number of pages: 39

The aim of this thesis was to develop a music bot for a Discord server using Python. The bot enables users to play songs directly in the server's chat environment by retrieving music from streaming services, such as YouTube or SoundCloud.

Key aspects of the project included utilizing Discord's API to build bot functionality, optimizing the bot's performance, and maintaining audio quality in real-time usage. The bot, equipped with an embedded media player, proved effective in handling user requests smoothly and delivering high audio quality. In conclusion, careful optimization and API integration facilitate the development of an efficient and user-friendly music bot for Discord.

SISÄLLYS

SANASTO

käsite	selite
API	Sovellusrajapinta, joka mahdollistaa ohjelmien keskinäisen kommunikoinnin ja tietojen vaihdon. Esimerkiksi Discordin API mahdollistaa ulkoisten sovellusten, kuten bottien, integroinnin Discord-ympäristöön. API on lyhenne sanoista Application Programming Interface.
Botti	Ohjelmisto, joka suorittaa automaattisesti ennalta määrättyjä tehtäviä, kuten viestien moderointia tai musiikin toistoa. Discord-botit voivat reagoida käyttäjien komentoihin ja suorittaa monipuolisia toimintoja palvelimilla.
Discord	Viestintäalusta, joka tarjoaa reaaliaikaisia tekstipohjaisia ja ääneen perustuvia keskustelukanavia. Suosittu erityisesti peliyhteisöissä ja projektityöryhmissä.
Embedded App	Sovellus, joka toimii upotettuna toisessa alustassa, kuten Discordissa. Näitä käytetään esimerkiksi pelien ja muiden interaktiivisten toimintojen kehittämiseen suoraan Discordin käyttöliittymässä.
FFmpeg	Ohjelmistotyökalu äänen ja videon käsittelyyn. Musiikkiboteissa FFmpeg mahdollistaa äänen suoratoiston eri lähteistä, kuten YouTubesta tai SoundCloudista.
Gateway	Discordin WebSocket-pohjainen rajapinta, joka välittää tapahtumatietoja reaaliaikaisesti. Tämä mahdollistaa esimerkiksi viestien ja käyttäjätoimintojen seurannan sovelluksille.

GitHub	Versionhallinta- ja yhteistyöalusta, jossa ohjelmistokehittäjät voivat hallita projektin koodia. Projektissa GitHubia käytettiin versionhallintaan ja yhteistyön koordinointiin.
HTTP	Verkkoprotokolla, jota käytetään tiedonsiirtoon palvelimen ja asiakkaan välillä. Discord API hyödyntää HTTP-pyyntöjä resurssien, kuten viestien ja käyttäjätietojen, hallintaan. HTTP on lyhenne sanoista HyperText Transfer Protocol.
Koodekki	Ohjelmisto tai laitteisto, joka pakkaa tai purkaa ääni- tai videotiedostoja. Discord käyttää Opus-koodekkia äänen pakkaamiseen äänikanavilla.
OAuth2	Autentikointiprotokolla, jota käytetään turvalliseen kirjautumiseen ja pääsynhallintaan. Discordissa OAuth2 mahdollistaa botin liittämisen palvelimelle turvallisesti.
Python	Monipuolinen ja suosittu ohjelmointikieli, joka tunnetaan yksinkertaisesta syntaksistaan ja laajasta sovellusalueestaan. Python on yleinen valinta bottien ja automaatioiden kehittämisessä, ja sitä käytettiin tämän projektin Discord-musiikkibotin ohjelmointikielenä.
REST	Arkkitehtuurimalli, jota käytetään API-suunnittelussa. Discordin HTTP API on REST-pohjainen, mikä mahdollistaa resurssien luomisen, lukemisen, päivittämisen ja poistamisen. REST on lyhenne sanoista Representational State Transfer.
RFC 6585	Internet-standardin mukainen spesifikaatio, joka määrittää tapoja hallita HTTP-pyyntöjen määrää (Rate Limiting). Discord API käyttää tätä standardia liiallisen käytön estämiseen.

SDK	Ohjelmistokehityspaketti, joka sisältää työkaluja, kirjastoja ja dokumentaatiota ohjelmistojen kehittämiseen. Esimerkiksi Discordin Embedded App SDK tukee vuorovaikutteisten pelien ja toimintojen rakentamista. SDK on lyhenne sanoista Software Development Kit.
VS Code	Lyhenne sanoista Visual Studio Code, VS Code on suosittu ja monipuolinen koodieditori, jota käytettiin tämän projektin pääasiallisena kehitysympäristönä. Se tukee Pythonia ja sisältää laajennuksia, kuten Git-integraation ja terminaalin.
WebSocket	Protokolla, joka mahdollistaa kaksisuuntaisen, reaaliaikaisen viestinnän asiakkaan ja palvelimen välillä. Gateway API hyödyntää tätä teknologiaa.

1 JOHDANTO

Tämän opinnäytetyön tavoitteena on kehittää Discord-palvelimelle musiikkibotti Python-ohjelmointikielellä. Botti on ohjelmisto, joka suorittaa automaattisesti käyttäjien antamia komentoja ja tuo palvelimelle lisäominaisuuksia, kuten musiikin toiston. Tässä työssä botti on suunniteltu tarjoamaan palvelimen käyttäjille mahdollisuuden kuunnella musiikkia suoraan keskustelukanavilla ilman tarvetta siirtyä toisille alustoille.

Discord on laajalti käytetty viestintäalusta, erityisesti harrastus- ja projektiryhmissä, tarjoten reaaliaikaisen keskustelun ja ääniyhteyden. Palvelu sisältää API-rajapinnan (Application Programming Interface), joka mahdollistaa käyttäjien oman botin kehittämisen ja automatisoitujen toimintojen toteuttamisen palvelimella. Tämän rajapinnan kautta botti voi liittyä keskustelukanavalle, vastata käyttäjien komentoihin ja soittaa musiikkia.

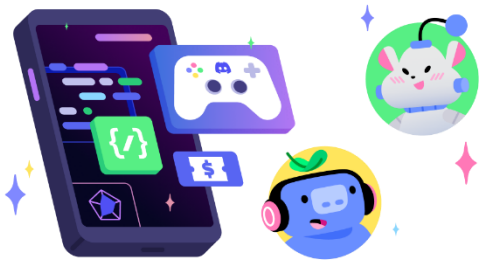
Työn lähtökohtana oli tutkia Discordin API:n hyödyntämistä botin toiminnallisuuksien ja palvelimen automaatioiden luomisessa siten, että käyttökokemus olisi mahdollisimman sujuva. Musiikkibotti toimii palvelimella eräänlaisena mediasoittimena, joka toistaa käyttäjien pyytämiä kappaleita YouTube- ja SoundCloud-suoratoistopalveluista [1][2]. Botti on varustettu sulautetulla mediasoittimella, jonka avulla pystytään tarjoamaan laadukasta ääntä reaaliajassa.

Musiikkibotin kehittämisessä on teknisiä haasteita, kuten suorituskyvyn optimointi ja äänenlaadun ylläpitäminen reaaliaikaisessa käytössä. Projektin merkitys korostuu siinä, että hyvin optimoitu ja käyttäjäystävällinen botti voi parantaa palvelimen jäsenten kokemusta huomattavasti, tehden yhteisestä viestintäympäristöstä monipuolisemman ja viihdyttävämmän.

Tämä työ tarjoaa Discord-yhteisöille konkreettisen ratkaisun, jolla musiikin voi tuoda osaksi palvelinalustaa.

2 DISCORD-SOVELLUSTEN KEHITYKSEN YLEISKATSAUS

Discord-sovellukset tarjoavat kehittäjille laajan valikoiman mahdollisuuksia muokata ja parantaa alustaa, jota miljoonat käyttäjät hyödyntävät päivittäin. Sovellukset voivat olla kaikkea yksinkertaisista työkaluista, kuten moderointiboteista, aina monimutkaisiin peleihin ja vuorovaikutteisiin aktiviteetteihin, jotka rikastavat käyttäjäkokemusta. Discordin kehittäjädokumentaatio tarjoaa kattavan lähtökohdan sovellusten mahdollisuuksien ymmärtämiseen ja niiden kehittämiseen tarvittaviin työkaluihin tutustumiseen [3].



KUVA 1. Havainnekuva eri Discord-sovelluksista, kuten peleistä, integraatioista ja palvelinten muokkaamisesta [3].

2.1 Discord-sovellusten ominaisuudet ja toiminnot

Discord-sovellukset tarjoavat kehittäjille mahdollisuuden rakentaa alustan sisälle toimintoja, jotka tekevät siitä käyttäjille mielekkäämmän ja interaktiivisemman. Keskeisiin ominaisuuksiin kuuluvat viestien hallinta ja lähettäminen, joiden avulla sovellukset voivat luoda viestejä Create Message -rajapinnan kautta, vastata vuorovaikutuksiin ja hallita viestien sisältöä. Näitä ominaisuuksia käytetään erityisesti moderoinnissa ja palvelinten automaatiossa. Esimerkiksi botti voi

poistaa asiattomia viestejä reaaliajassa tai lähettää ilmoituksia palvelimen säännöistä uusille käyttäjille.

Käyttäjäkokemusta voidaan edelleen parantaa tarjoamalla vuorovaikutteisia elementtejä, kuten painikkeita, valikoita ja lomakkeita, jotka tekevät toiminnallisuuksista selkeämpiä ja helpommin käytettäviä. Sulautettujen pelien ja kokemusten luominen on toinen merkittävä ominaisuus, jonka mahdollistaa Discordin Embedded App SDK. Tämä työkalu antaa kehittäjille mahdollisuuden rakentaa alustan sisällä toimivia pelejä ja sosiaalisia aktiviteetteja, jotka toimivat saumattomasti eri laitteilla. Esimerkiksi trivia-pelejä tai yhteistyöhön perustuvia tehtäviä voidaan suorittaa suoraan Discordin käyttöliittymässä ilman tarvetta siirtyä ulkoisiin sovelluksiin.

Palvelinten hallinnan kustomointi on tärkeä osa sovellusten tarjoamia toimintoja. Sovellukset voivat hallita kanavia, käyttäjiä ja muita palvelimen resursseja. Lisäksi moderointityökalut, kuten automaattiset sääntörikkomusten tunnistusjärjestelmät, tekevät palvelimien ylläpidosta tehokkaampaa. Rich Presence -ominaisuus puolestaan tarjoaa kehittyneitä mahdollisuuksia yhdistää pelien ja sovellusten tiedot käyttäjien profiileihin, mikä voi lisätä käyttäjien sitoutumista alustaan. Maksulliset premium-ominaisuudet tarjoavat kehittäjille myös keinon rahoittaa sovellusten ylläpitoa ja jatkokehitystä.

2.2 Sovellusten käyttöönoton kontekstit

Discord-sovellusten asennuskontekstit määrittävät, miten ja missä niitä voidaan käyttää. Palvelinkohtaiset sovellukset, kuten moderointityökalut, asennetaan suoraan palvelimelle ja niitä voivat käyttää kaikki palvelimen jäsenet, joilla on tarvittavat oikeudet. Tällaiset sovellukset voivat hallita viestikanavia, jakaa rooleja ja valvoa sääntöjen noudattamista reaaliaikaisesti. Tämä tekee niistä erityisen hyödyllisiä suurille ja aktiivisille yhteisöille, joissa automaatio helpottaa ylläpitoa.

Käyttäjäkohtaiset sovellukset puolestaan mahdollistavat yksityisemmän kokemuksen, jossa sovellus toimii yksittäisen käyttäjän tarpeiden mukaan. Esimerkiksi henkilökohtaiset pelitilastot tai yksityiset työkalut voidaan yhdistää käyttäjän profiiliin ja tarjota mukautettuja toimintoja eri palvelimilla. Tämä tekee niistä ihanteellisia yksilöllisempiä sovelluksia varten, kuten pelien sisäiset lisäosat tai musiikkibotit.

2.3 Discordin tarjoamat rajapinnat (API)

Discordin API-rajapinnat muodostavat sovellusten kehityksen teknisen perustan. Näistä tärkein on HTTP API, joka toimii REST-arkkitehtuurin pohjalta. Se mahdollistaa resurssien, kuten viestien, käyttäjien ja kanavien, hallinnan suoraviivaisilla pyynnöillä. API tukee muun muassa resurssien luomista, muokkaamista ja poistamista, mikä tekee siitä monipuolisen ja tehokkaan työkalun kehittäjille. Samalla kehittäjien tulee huomioida RFC 6585 -standardin mukainen pyyntöjen hallinta, sillä liiallinen käyttö voi johtaa API-avaimen peruuttamiseen [4][5].

Gateway API tarjoaa WebSocket-pohjaisen ratkaisun tapahtumatietojen käsittelyyn reaaliajassa. Tämä tekee siitä erityisen hyödyllisen sovelluksille, jotka seuraavat käyttäjien toimintaa, kuten viestien lähettämistä tai kanavaan liittymistä. Gateway mahdollistaa myös reaaliaikaiset reaktiot, kuten roolien jakamisen tai automaattisten vastauksien lähettämisen.

2.4 Yhteenveto

Discord-sovellukset tarjoavat laajan valikoiman mahdollisuuksia, joiden avulla alusta voidaan räätälöidä ja parantaa vastaamaan sekä käyttäjien että ylläpitäjien tarpeita. Sovellusten kehityksessä avainasemassa ovat Discordin API-rajapinnat, jotka tarjoavat työkalut monimutkaisten ja vuorovaikutteisten toimintojen

toteuttamiseen. Olipa kyseessä palvelimen automaatio, pelien integrointi tai yksityiset työkalut, Discordin kehittäjäympäristö mahdollistaa joustavat ja tehokkaat ratkaisut eri käyttötarkoituksiin. Sovellusten monipuolisuus ja jatkuvasti kehittyvä dokumentaatio tekevät niistä houkuttelevan valinnan sekä aloittelijoille että kokeneille kehittäjille.

3 PYTHON OHJELMOINTIKIELENÄ

Python on jo pitkään ollut yksi maailman suosituimmista ja monipuolisimmista ohjelmointikielistä. Sen suosio perustuu erityisesti selkeään syntaksiin, aktiiviseen kehittäjäyhteisöön ja laajaan kirjastotarjontaan, joka kattaa niin web-kehityksen, data-analytiikan, koneoppimisen kuin erilaisten sovellusohjelmointirajapintojen (API) kanssa työskentelyn. Suuren suosion myötä Pythonia päivitetään jatkuvasti, ja siitä on saatavilla runsaasti dokumentaatiota sekä valmiita ratkaisuja erilaisiin ongelmatilanteisiin. Pythonin joustavuus näkyy myös siinä, että koodi on helposti siirrettävissä eri käyttöjärjestelmien välillä, mikä on etu erityisesti projekteissa, joissa kehitysympäristö, testaus ja tuotantoympäristö saattavat toimia eri alustoilla.

KUVA 2. Python-ohjelmointikielen tunnus [6].

Discord-musiikkibotin tapauksessa Python tarjoaa useita merkittäviä etuja. Ensinnäkin sen asynkroninen ohjelmointimalli (kuten asyncio) soveltuu ihanteellisesti tilanteisiin, joissa botin täytyy reagoida nopealla tahdilla useisiin komentoihin reaaliaikaisesti, esimerkiksi toistettaessa musiikkia usean eri kanavan välillä. Asynkroniset funktiot ja tapahtumasilmukka varmistavat, ettei äänen suoratoisto hidastu tai viive kasva suurillakaan palvelimilla. Toiseksi Pythoniin on saatavilla useita valmiita kirjastoja, kuten discord.py, joka yksinkertaistaa huomattavasti monimutkaisia Discord-rajapintoja. Näiden kirjastoiden ansiosta kehittäjien ei tarvitse itse kirjoittaa kaiken kattavia API-pyyntöjä tai -kutsuja, vaan tyypilliset toiminnot – kuten viestien hallinta,

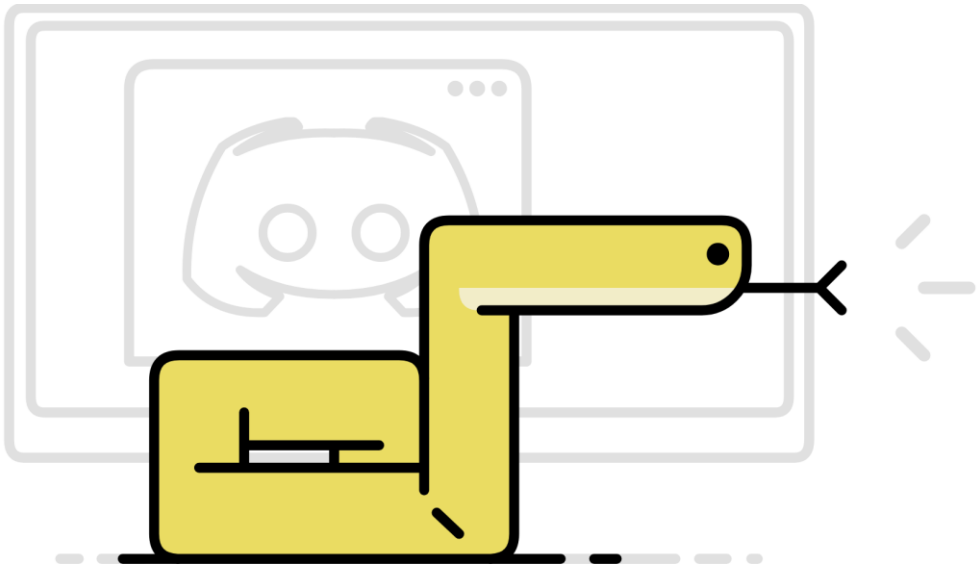
äänikanaviin liittyminen tai äänenkäsittely – on valmiiksi paketoitu helppokäyttöisiksi metodeiksi. Lisäksi Pythonin dynaaminen tyypitys helpottaa prototyyppien ja kokeellisten ratkaisujen kehittämistä, sillä koodin korjaaminen ja laajentaminen onnistuu ilman raskaita rakenteellisia muutoksia, jotka ovat ominaisia tietyille vahvasti tyypitetyille kielille.

Pythonin laaja kirjastoekosysteemi tukee myös muita botin kannalta tärkeitä osa-alueita. Koska musiikkibotti joutuu käsittelemään potentiaalisesti suuria tietomääriä (esimerkiksi hakujen tai soittolistojen hallinnan osalta), Pythonin valmiit työkalut tiedon varastointiin, käsittelyyn ja analytiikkaan (mm. pandas, sqlite3 tai monet NoSQL-kirjastot) tarjoavat hyvät puitteet. Jos botin ominaisuuksia tulevaisuudessa laajennetaan, voidaan muun muassa koneoppimismenetelmiä, tekstianalytiikkaa tai suositusalgoritmeja hyödyntää varsin helposti Pythonin kirjastoilla, kuten scikit-learn tai tensorflow. Näin esimerkiksi musiikkibotti voisi sovittaa soittolistoja automaattisesti käyttäjien makumieltymysten mukaan.

Lisäksi Pythonin kehittäjäystävällisyys heijastuu moniin käytännön toimiin projektin elinkaaren aikana. Kehitysympäristön pystyttäminen onnistuu nopeasti venv-virtuaaliympäristöillä, jolloin projektin riippuvuudet pysyvät selkeästi erossa muista järjestelmään asennetuista kirjastoista. Versionhallinnan (esimerkiksi GitHub) yhteydessä Pythonin koodia on myös helppo testata ja validoida erilaisin työkaluin, kuten pytest-testikirjastolla tai CI/CD-prosessia automatisoivilla ratkaisuilla (esimerkiksi GitHub Actions). Kaiken kaikkiaan Pythonin kypsyys, laaja tuki ja kehitysresurssit takaavat, että Discord-musiikkibotin ominaisuuksia voi kehittää jatkuvasti eteenpäin, skaalata laajemmalle käyttäjäkunnalle ja integroida monimutkaisiin sovellusympäristöihin – ilman, että itse kielen tai kirjastojen rajoitukset muodostuisivat pullonkaulaksi [6].

4 DISCORD.PY API JA FFMPEGOPUSAUDION KÄYTTÖ

Discord.py on Python-ohjelmointikielelle kehitetty kirjasto, joka tarjoaa laajan tuen Discordin API:n toiminnallisuuksille. Tämä kirjasto on yksi suosituimmista työkaluista bottien kehittämiseen Discord-alustalla, sillä se abstrahoi monimutkaiset API-kutsut yksinkertaisiksi ja helposti hallittaviksi metodeiksi. Discord.py mahdollistaa viestien lähettämisen ja muokkaamisen, kanavien ja käyttäjien hallinnan sekä monimutkaisempien toimintojen, kuten ääniominaisuuksien, integroinnin. Tämän työn tavoitteena oli hyödyntää Discord.py:n tarjoamia ominaisuuksia luomaan toimiva ja käyttäjäystävällinen musiikkibotti, joka pystyy suoratoistamaan ääntä reaaliaikaisesti Discord-palvelimen äänikanavilla [7].



KUVA 3. Discord.py-kirjaston logo, joka yhdistää Pythonin ja Discordin visuaaliset elementit [7].

Discord.py:n asynkroninen ohjelmointimalli, joka perustuu Pythonin asyncio-kirjastoon, tarjoaa merkittäviä etuja reaaliaikaisten sovellusten, kuten musiikkibottien, kehityksessä. Tämä malli mahdollistaa useiden rinnakkaisten tehtävien suorittamisen ilman, että botti kuormittaa palvelinta liiallisilla viiveillä.

Esimerkiksi komennot, kuten musiikin toiston aloittaminen tai pysäyttäminen, voidaan käsitellä samanaikaisesti viestien moderointitoimintojen kanssa. Tämä tekee Discord.py:stä hyvin soveltuvan keskikokoisiin projekteihin, joissa toiminnallisuuksien monipuolisuus ja helppokäyttöisyys ovat keskiössä.

Äänen käsittelyssä FFmpegOpusAudio valittiin tämän projektin keskeiseksi komponentiksi. Discord käyttää natiivisti Opus-koodekkia, joka on suunniteltu tehokasta äänen pakkausta ja vähäistä viivettä varten. FFmpegOpusAudio hyödyntää tätä koodekkia suoraan, mikä poistaa tarpeen ylimääräiselle äänen muuntamiselle. Tämä on erityisen tärkeää reaaliaikaisessa suoratoistossa, jossa viiveen minimointi ja äänenlaadun säilyttäminen ovat kriittisiä tekijöitä. FFmpeg, johon FFmpegOpusAudio pohjautuu, tukee myös laajaa valikoimaa mediatiedostomuotoja, kuten MP3, AAC ja WAV, mikä mahdollistaa joustavan ja monipuolisen äänisisällön toiston [8].

Discord.py:n natiivien ääniominaisuuksien, kuten PCMAudio- ja WaveAudio-luokkien, rajoitukset tekivät FFmpegOpusAudio-luokasta huomattavasti houkuttelevamman vaihtoehdon. Esimerkiksi PCMAudio on hyödyllinen tapauksissa, joissa ääni on jo valmiiksi oikeassa formaatissa, mutta se ei tue laajaa tiedostomuotojen kirjoa tai suoratoistolähteitä. WaveAudio puolestaan rajoittuu pelkästään WAV-tiedostoihin, mikä tekee siitä vähemmän käytännöllisen monimutkaisemmissa käyttötapauksissa. FFmpegOpusAudio yhdistää korkean äänenlaadun, laajan tiedostomuototuen ja yksinkertaisen integraation, mikä teki siitä tämän projektin tarpeisiin parhaiten sopivan ratkaisun [9].

Kun tarkastellaan muita äänenkäsittelyteknologioita, kuten JavaScript-kirjasto Discord.js:n usein käyttämää Lavalink-järjestelmää, FFmpegOpusAudio tarjoaa huomattavasti yksinkertaisemman toteutuksen. Lavalink käyttää erillistä palvelinta, joka vastaanottaa ja käsittelee äänipyynnöt, ja vaikka tämä voi olla hyödyllistä suurille palvelimille, se lisää monimutkaisuutta ja vaatii enemmän resursseja [10]. Tämän projektin tavoitteena oli yksinkertainen ja helposti ylläpidettävä järjestelmä, joka ei vaatinut ulkoisia palvelimia, mikä teki FFmpegOpusAudio-luokan sopivammaksi valinnaksi.

Pythonin yleinen suosio ohjelmointikielenä, sen helppolukuisuus ja laaja kirjastoekosysteemi tukivat myös Discord.py:n valintaa. Pythonin käyttö tekee projektin laajentamisesta ja ylläpidosta suoraviivaista, ja kirjasto tarjoaa hyvän yhteensopivuuden muiden työkalujen, kuten data-analytiikan ja koneoppimisen, kanssa. Discord.py:n ja FFmpegOpusAudio-yhdistelmä tarjoaa joustavan alustan, joka täyttää sekä tekniset että käyttäjäkokemuksen vaatimukset.

Yhteenvetona Discord.py ja FFmpegOpusAudio osoittautuivat tehokkaiksi ja yhteensopiviksi ratkaisuiksi tämän projektin musiikkibotin kehittämisessä. Ne tarjosivat optimaalisen yhdistelmän suorituskykyä, helppokäyttöisyyttä ja teknistä joustavuutta, mikä mahdollisti käyttäjäystävällisen ja toimivan botin rakentamisen. Näiden työkalujen avulla botti pystyy tarjoamaan korkealaatuista ääntä reaaliaikaisesti ja vastaamaan dynaamisesti käyttäjien tarpeisiin Discord-palvelimilla.

5 KEHITYSYMPÄRISTÖ

Discord-musiikkibotin kehityksessä käytettiin useita työkaluja ja alustoja projektin hallintaan, versionhallintaan ja kehitysympäristön konfigurointiin. Kehitysympäristön tarkoituksena oli luoda tehokas ja organisoitu rakenne, joka tukee projektin tavoitteiden saavuttamista. Tämä kappale käsittelee GitHubin roolia versionhallinnassa, Visual Studio Code -editorin käyttöä, Pythonin virtuaaliympäristön hallintaa ja Discord Developer Portalin hyödyntämistä botin rekisteröinnissä ja oikeuksien hallinnassa [11][12].

5.1 GitHub ja versionhallinta

Projektin versionhallinta oli keskeisessä roolissa, sillä se mahdollisti projektin koodin organisoinnin ja hallinnan systemaattisella tavalla. Kehitys alkoi GitHub-repositorion luomisella, jossa määritettiin projektin alkuasetukset, kuten README-tiedosto, jossa kuvattiin projektin tarkoitus ja tavoitteet. Repositorio antoi selkeän rakenteen, jossa projektin eri osat voitiin jakaa tiedostoiksi ja kansioiksi. Lisäksi GitHubin tarjoamat ominaisuudet, kuten haarat (branches), mahdollistivat erilaisten ominaisuuksien kehittämisen rinnakkain ilman, että pääprojekti vaarantui.

Kun repositorio oli luotu, projekti kloonattiin paikallisesti Visual Studio Codeen. Tämä antoi kehittäjille mahdollisuuden työskennellä koodin parissa paikallisesti ja tehdä muutoksia ennen niiden lähettämistä takaisin GitHubiin. Jokainen merkittävä muutos koodissa dokumentoitiin ja päivitettiin GitHubiin. Käytännössä tämä tarkoitti, että kehittäjät loivat selkeät commit-viestit, kuten "Lisätty musiikin suoratoiston perusominaisuus" tai "Korjattu virhe komennon käsittelyssä". Näin projektin etenemistä voitiin seurata läpinäkyvästi.

GitHubin pull request -ominaisuudet tarjosivat mahdollisuuden tarkastella ja arvioida ehdotettuja muutoksia ennen niiden yhdistämistä päähaaraan. Tämä oli

erityisen hyödyllistä projektin tiimityössä, sillä se edisti koodin laatua ja ehkäisi mahdollisia virheitä. GitHubin avulla myös dokumentointi oli yksinkertaista, sillä repositorion Wiki-osio ja README-tiedosto tarjosivat selkeän paikan tallentaa ohjeet ja tekniset tiedot botin kehittämisestä ja käyttöönotosta.

5.2 Visual Studio Code ja Pythonin virtuaaliympäristö

Visual Studio Code valittiin kehitysympäristöksi sen monipuolisuuden ja laajennettavuuden vuoksi. Editorin Python-integraatio tuki tehokasta ohjelmistokehitystä, tarjoten ominaisuuksia, kuten koodin automaattisen täydennyksen, virheiden korostamisen ja integroinnin terminaaliin. Lisäksi Visual Studio Code mahdollisti GitHub-integraation, jolloin commitit ja pushit voitiin tehdä suoraan editorin kautta.

Kehitysympäristön tärkeä osa oli Pythonin virtuaaliympäristön luominen. Virtuaaliympäristö luotiin komennolla:

```
python -m venv venv
```

Virtuaaliympäristön aktivoinnin jälkeen projektin kaikki riippuvuudet asennettiin erilliseen ympäristöön. Tämä tarkoitti esimerkiksi seuraavien kirjastojen asentamista:

```
pip install discord.py
```

```
pip install ffmpeg-python
```

Virtuaaliympäristö varmisti, että kaikki projektin riippuvuudet olivat erillään käyttöjärjestelmän globaalista Python-asennuksesta, mikä vähensi versioristiriitojen riskiä. Lisäksi se mahdollisti ympäristön tallentamisen GitHub-repositorioon requirements.txt-tiedoston avulla, jolloin muut kehittäjät pystyivät helposti asentamaan samat riippuvuudet omille koneilleen.

5.3 Discord Developer Portal ja botin perustaminen

Discord-botin kehityksen ytimessä oli sen rekisteröiminen Discord Developer Portaliin. Tämä prosessi alkoi uuden sovelluksen luomisella, jossa botille annettiin nimi ja kuvaus. Sovellukselle määritettiin myös kuvake, joka valittiin visuaalisesti erottuvaksi, jotta botti olisi helppo tunnistaa palvelimen jäsenille.

Botin toiminnallisuuden mahdollistamiseksi sille luotiin API-avain, joka toimii ikään kuin salasanana, jolla botti voi kommunikoida Discordin palvelinten kanssa. Tämä API-avain on kriittinen osa bottia, ja sen suojaaminen on tärkeää, sillä sen vuotaminen voisi antaa ulkopuolisille pääsyn botin toimintoihin.

Bottia varten määritettiin tarvittavat käyttöoikeudet, kuten pääsy äänikanaviin ja oikeudet lähettää viestejä. Näiden käyttöoikeuksien määrittäminen tapahtui tarkkaan harkiten, jotta botti voi suorittaa kaikki tarvittavat toiminnot ilman ylimääräisiä oikeuksia, jotka voisivat aiheuttaa turvallisuusriskejä. Lisäksi OAuth2-työkaluja käytettiin botin linkittämiseen palvelimille, mikä mahdollisti sen lisäämisen useille palvelimille yksinkertaisella URL-osoitteella.

Kehitysympäristön kokonaisuus

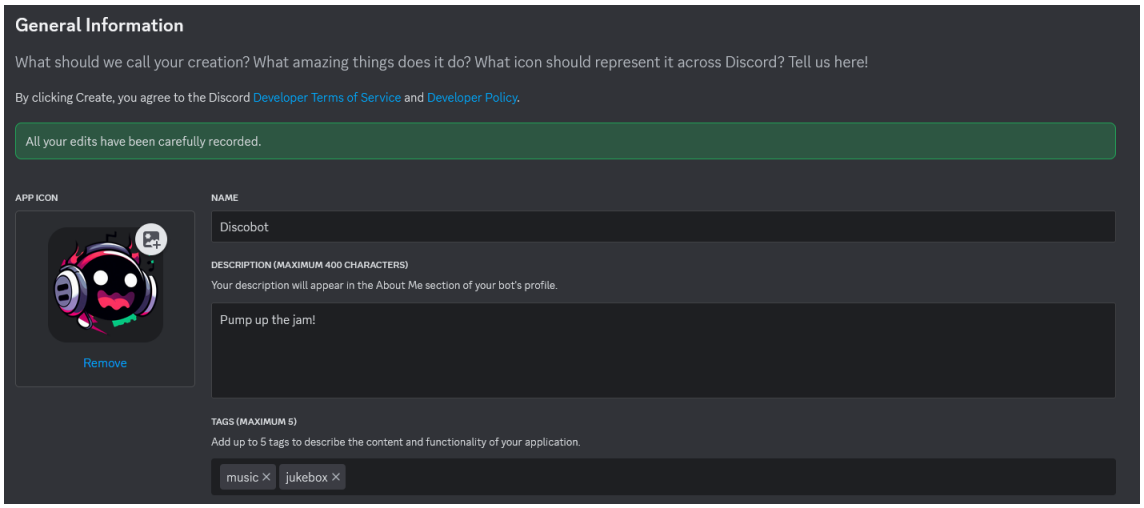
Kaikkien näiden työkalujen ja alustojen yhteiskäyttö mahdollisti tehokkaan ja hyvin organisoidun kehitysympäristön. GitHub varmisti koodin versionhallinnan ja yhteistyön sujuvuuden, Visual Studio Code tuki kehitystä intuitiivisena editorina, ja Discord Developer Portal antoi kaikki tarvittavat työkalut botin rekisteröintiin ja hallintaan. Näiden elementtien saumaton yhteensopivuus muodosti perustan onnistuneelle ja skaalautuvalle botin kehitysprosessille.

6 BOTIN ALUSTAMINEN

Discord-musiikkibotin kehittäminen alkoi ympäristön ja työkalujen asennuksella sekä botin käyttöoikeuksien määrittämisellä Discord-palvelimella. Tämä vaihe sisälsi botin rekisteröinnin Discord Developer Portaliin, API-avaimen luomisen ja tarvittavien käyttöoikeuksien asettamisen. Lisäksi botti liitettiin palvelimelle, missä se testattiin ensimmäisten komentojen avulla.

6.1 Discordin API ja botin määrittäminen

Botin alustaminen alkoi sen perustietojen määrittämisellä Discord Developer Portalissa. Tämä sisälsi botin nimen, kuvauksen ja ikonin lisäämisen. Botin ikoni generoitiin Microsoft Designer -sivustolla, jossa tekoäly luo yksinkertaisia ja visuaalisesti erottuvia kuvia. Nämä tiedot auttavat käyttäjiä tunnistamaan botin palvelimella ja antavat sille ammattimaisemman ilmeen.



KUVA 4. Discordin kehittäjäportaalissa annetut botin perustiedot sekä Designer-sivustolla generoitu ikoni.

API-avain, joka toimii botin tunnisteena ja autentikointityökaluna, luotiin osana rekisteröintiprosessia. Tämä avain on välttämätön botin ja Discordin API:n välisessä kommunikoinnissa. Botille määritettiin käyttöoikeudet, jotka mahdollistivat sen pääsyn palvelimen viesti- ja äänikanaviin sekä oikeuden hallita näitä resursseja. Näiden käyttöoikeuksien hallinta varmistaa, että botti voi suorittaa tarvitsemansa toiminnot mutta ei saa tarpeettomia oikeuksia, jotka voisivat aiheuttaa turvallisuusriskejä.

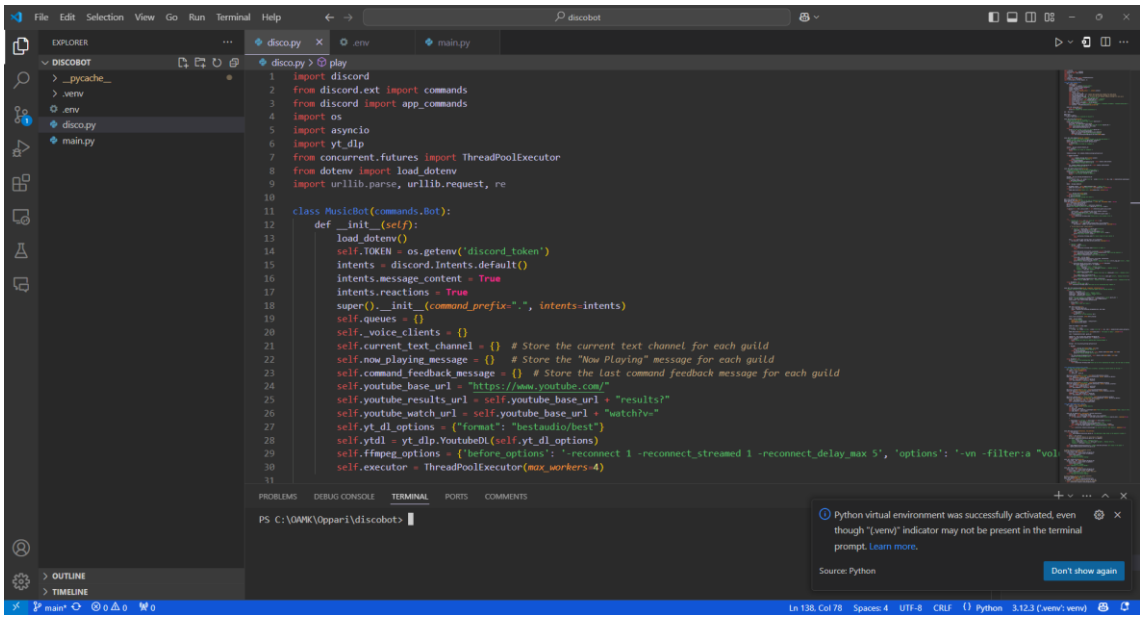
KUVA 5. Botille annetut tarvittavat käyttöoikeudet Discordin kehittäjäportaalissa.

6.2 Visual Studio Code ja Pythonin virtuaaliympäristö

Discord-botin kehittämiseen käytettiin Visual Studio Codea (VS Code), joka toimii monipuolisena koodieditorina Python-kehityksessä. Työympäristöä varten luotiin Pythonin virtuaaliympäristö, jotta projektin riippuvuudet voitiin pitää hallinnassa. Virtuaaliympäristö luotiin komennolla:

python -m venv venv

Tämän jälkeen virtuaaliympäristö aktivoitiin ja siihen asennettiin tarvittavat kirjastot, kuten discord.py. Näiden kirjastojen avulla botti voi käyttää Discordin API:a ja suorittaa sen määrittämiä toimintoja. Virtuaaliympäristön käyttö helpotti projektin hallintaa, sillä se eristi botin riippuvuudet käyttöjärjestelmän globaaleista kirjastoista. Tämä oli tärkeää erityisesti projektin skaalautuvuuden ja yhteistyön kannalta, sillä jokainen tiimin jäsen pystyi helposti ottamaan käyttöön saman ympäristön riippuvuudet.



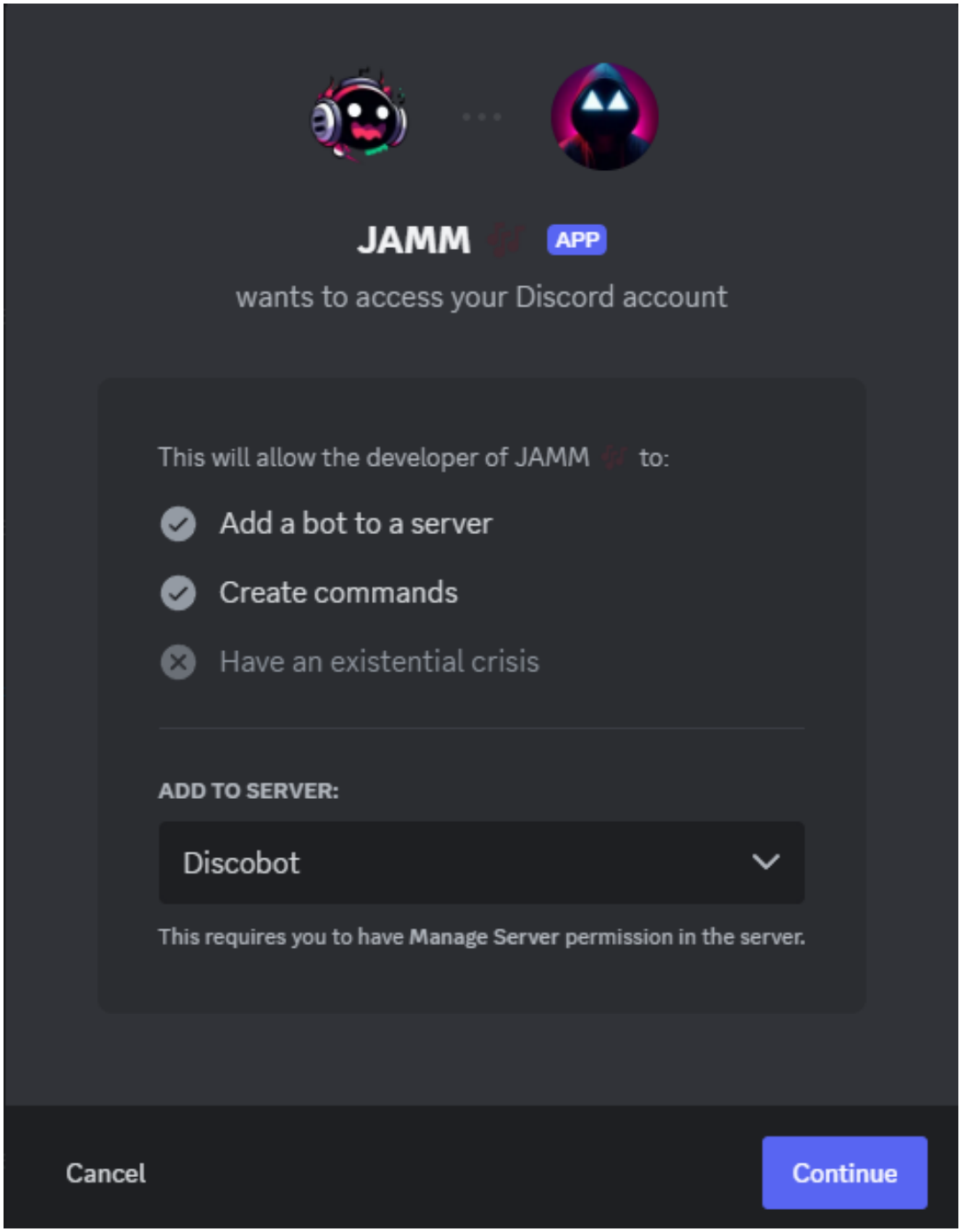
KUVA 6. Visual Studio Code -ympäristö näyttää koodieditorin ja terminaalin, jossa Pythonin virtuaaliympäristö on aktivoitu.

6.3 Botin liittäminen Discord-palvelimelle

Botin liittäminen palvelimelle suoritettiin OAuth2-autorisaatioprosessin avulla. Tämä tapahtui käyttämällä kehittäjäportaalista saatua URL-osoitetta, joka sisälsi botin tunnuksen ja sen tarvitsemat käyttöoikeudet. Kun URL-osoite avattiin, botti lisättiin valitulle palvelimelle. Prosessi oli suoraviivainen ja varmisti, että botti sai

juuri ne oikeudet, jotka sen toimintaan tarvittiin, kuten viestien lähettämisen, reaktioiden hallinnan ja ääniominaisuudet.

Tämä vaihe varmisti botin läsnäolon palvelimella ja mahdollisti sen toimintojen testaamisen käytännössä. Liittäminen palvelimelle oli keskeinen osa kehityksen etenemistä, sillä se toi botin abstrakteista API-kutsuista todelliseen ympäristöön, jossa sen toiminnot voitiin arvioida ja hioa käyttäjien tarpeita vastaaviksi [13].



KUVA 7. Botin lisääminen Discord-palvelimelle kehittäjäportaalista saadulla URL-osoitteella.

6.4 Botin ensimmäinen kokeilu

Kun botti oli yhdistetty palvelimelle, sen toiminnallisuuksia testattiin ensimmäisen komennon avulla. Bottiin ohjelmoitiin yksinkertainen ?play-komento, jolla käyttäjä voi toistaa kappaleen syöttämällä YouTube- tai SoundCloud-linkin komentoa seuraavaksi parametriarvoksi. Tämä testattiin palvelimen tekstikanavalla, josta botti automaattisesti liittyi palvelimen äänikanavalle ja aloitti kappaleen suoratoiston.

Testausta suoritettiin virtuaaliympäristössä suoraan VS Code -editorin terminaalinäkymässä. Tämä mahdollisti botin toimintojen reaaliaikaisen tarkkailun sekä virheiden tunnistamisen ja korjaamisen. Ensimmäiset testit osoittivat botin olevan vakaa, ja sen suorituskyky vastasi odotuksia. Tämä vaihe oli tärkeä, sillä se loi perustan botin myöhemmille parannuksille ja lisätoiminnoille.

```
(.venv) PS C:\OAMK\Oppari\discobot> .\.venv\Scripts\python.exe main.py
2024-10-21 15:22:58 INFO     discord.client logging in using static token
2024-10-21 15:22:59 INFO     discord.gateway Shard ID None has connected to Gateway (Session ID: b1db25d239282e1e35ad70053be3e787).
Discobot#8365 has connected to Discord!
2024-10-21 15:23:53 INFO     discord.voice_state Connecting to voice...
2024-10-21 15:23:53 INFO     discord.voice_state Starting voice handshake... (connection attempt 1)
2024-10-21 15:23:53 INFO     discord.voice_state Voice handshake complete. Endpoint found: stockholm9368.discord.media
2024-10-21 15:23:53 INFO     discord.voice_state Voice connection complete.
[youtube] Extracting URL: https://youtu.be/EfvwDwlS6aY?si=r35XS-IqVObZVTgR
[youtube] EfvwDwlS6aY: Downloading webpage
[youtube] EfvwDwlS6aY: Downloading ios player API JSON
[youtube] EfvwDwlS6aY: Downloading mweb player API JSON
[youtube] EfvwDwlS6aY: Downloading player 606a66b3
[youtube] EfvwDwlS6aY: Downloading m3u8 information
2024-10-21 15:26:38 INFO     discord.player ffmpeg process 21856 successfully terminated with return code of 0.
```

KUVA 8. VS Coden terminaalinäkymä botin yhdistämisestä Discord-palvelimeen ja ensimmäisen komennon suorittamisesta.

6.5 Alustamisen merkitys

Botin alustaminen Discord Developer Portalin, Visual Studio Coden ja virtuaaliympäristön avulla tarjosi vahvan perustan projektin kehitykselle. API-avaimen hallinta ja tarkasti määritellyt käyttöoikeudet varmistivat, että botti oli

turvallinen ja toiminnallinen Discord-palvelimilla. Kehitysympäristön rakenne oli keskeinen tekijä projektin onnistumisessa, sillä se mahdollisti iteratiivisen ja skaalautuvan lähestymistavan botin kehittämiseen. Ensimmäiset kokeilut ja onnistumiset loivat perustan botin jatkokehitykselle, jossa uusia ominaisuuksia ja optimointeja voitiin lisätä käyttäjäkokemuksen parantamiseksi.

7 BOTIN JATKOKEHITYS

Botin kehitys on edennyt merkittävästi alkuperäisen alustuksen jälkeen, ja siihen on lisätty huomattava määrä uusia toiminnallisuuksia, jotka tekevät siitä monipuolisemman ja käyttäjäystävällisemmän. Jatkokehitys on keskittynyt erityisesti soittolistan hallinnan parantamiseen, reaaliaikaisten vuorovaikutusten lisäämiseen, käyttöliittymän visuaalisiin parannuksiin sekä suorituskyvyn optimointiin. Lisäksi botin toimintojen skaalautuvuutta ja turvallisuutta on kehitetty, jotta se voi palvella suurempia käyttäjäyhteisöjä luotettavasti.

7.1 Soittolistan hallinta ja käyttöliittymän parannukset

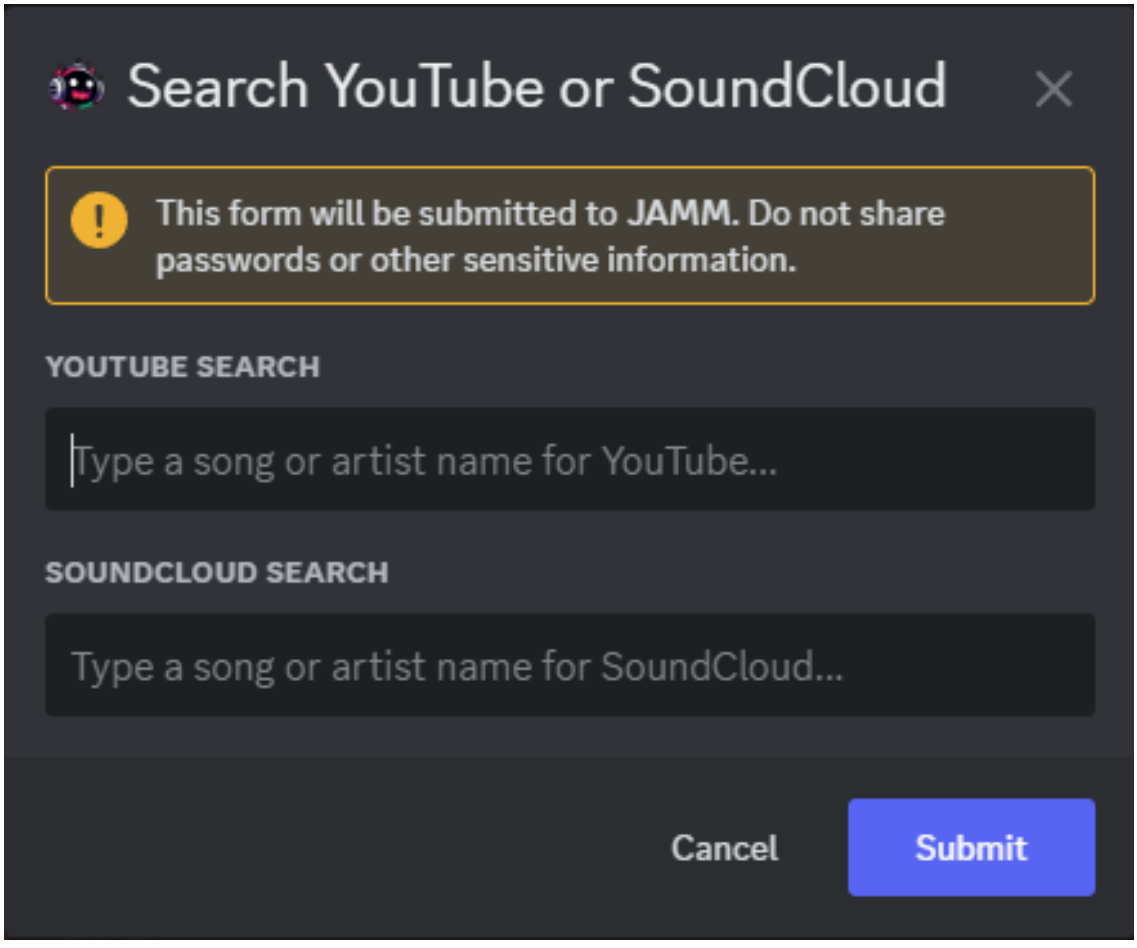
Botin soittolistan hallintaa on laajennettu merkittävästi, jotta käyttäjät voivat hallita musiikin toistoa entistä monipuolisemmin ja intuitiivisemmin. Soittolistaan voidaan nyt lisätä, poistaa ja siirtää kappaleita helposti suoraan Discordin käyttöliittymästä. Esimerkiksi remove_song-komento mahdollistaa yksittäisen kappaleen poistamisen, kun taas move_song-komennolla käyttäjä voi siirtää kappaleen haluamaansa kohtaan soittolistassa. Lisäksi käytössä on Shuffle-toiminto, joka tarjoaa helpon tavan sekoittaa soittolista satunnaiseen järjestykseen. Tämä ominaisuus on erityisen hyödyllinen tilanteissa, joissa halutaan luoda yllätyksellistä ja dynaamista musiikkikokemusta.

Botille on lisätty myös uusia komentoja, jotka tekevät soittolistan hallinnasta ja musiikin toistosta joustavampaa. Näitä ovat esimerkiksi queue_status, joka näyttää käyttäjille koko soittolistan kappaleiden toistojärjestyksen. Discordin merkkirajoitusten vuoksi soittolistassa näkyy seuraavan kymmenen kappaleen tiedot. Mikäli jonossa on enemmän kappaleita, jäljellä olevien kappaleiden määrä ilmoitetaan jonon lopussa. Tämä toteutus varmistaa selkeän ja helppokäyttöisen viestinnän myös pidempien soittolistojen tapauksessa.

Soittolistojen käsittelyssä botti ohittaa automaattisesti kappaleet, jotka ovat rajoitettuja, esimerkiksi yksityiset tai ikärajoitetut videot. Käyttäjälle näytetään viesti ohitetuista kappaleista ja syy, miksi niitä ei voitu lisätä soittolistaan. Tämä varmistaa sujuvan käyttökokemuksen ilman yllättäviä virheitä.

Pause, Resume ja Stop-komennot mahdollistavat musiikin hallinnan reaaliajassa. Käyttäjät voivat keskeyttää kappaleen toiston hetkellisesti, jatkaa sitä tai pysäyttää sen kokonaan, jolloin botti vapauttaa äänikanavan ja palvelimen resurssit tehokkaasti. Nämä toiminnot tekevät musiikin hallinnasta suoraviivaista ja käyttäjäystävällistä.

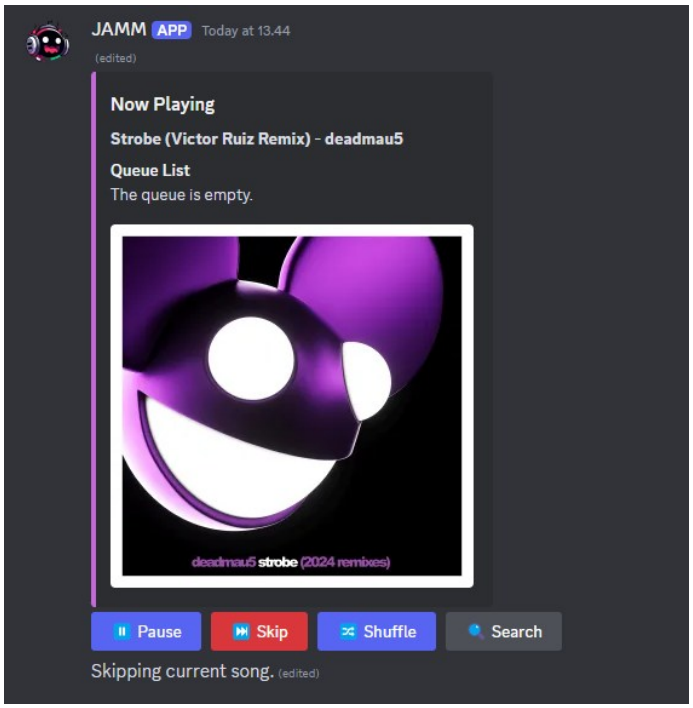
Lisäksi bottiin on integroitu uusi YouTube- ja SoundCloud-hakuominaisuus, jonka avulla käyttäjät voivat etsiä kappaleita suoraan Discordissa ilman tarvetta kopioida linkkejä ulkoisista selaimista. Hakutuloksista käyttäjä voi valita kappaleita, jotka lisätään automaattisesti soittolistaan tai toistetaan välittömästi. Tämä ominaisuus tekee botista entistä intuitiivisemman ja vähentää käyttäjän tarvetta käyttää ulkoisia työkaluja musiikin hallintaan.



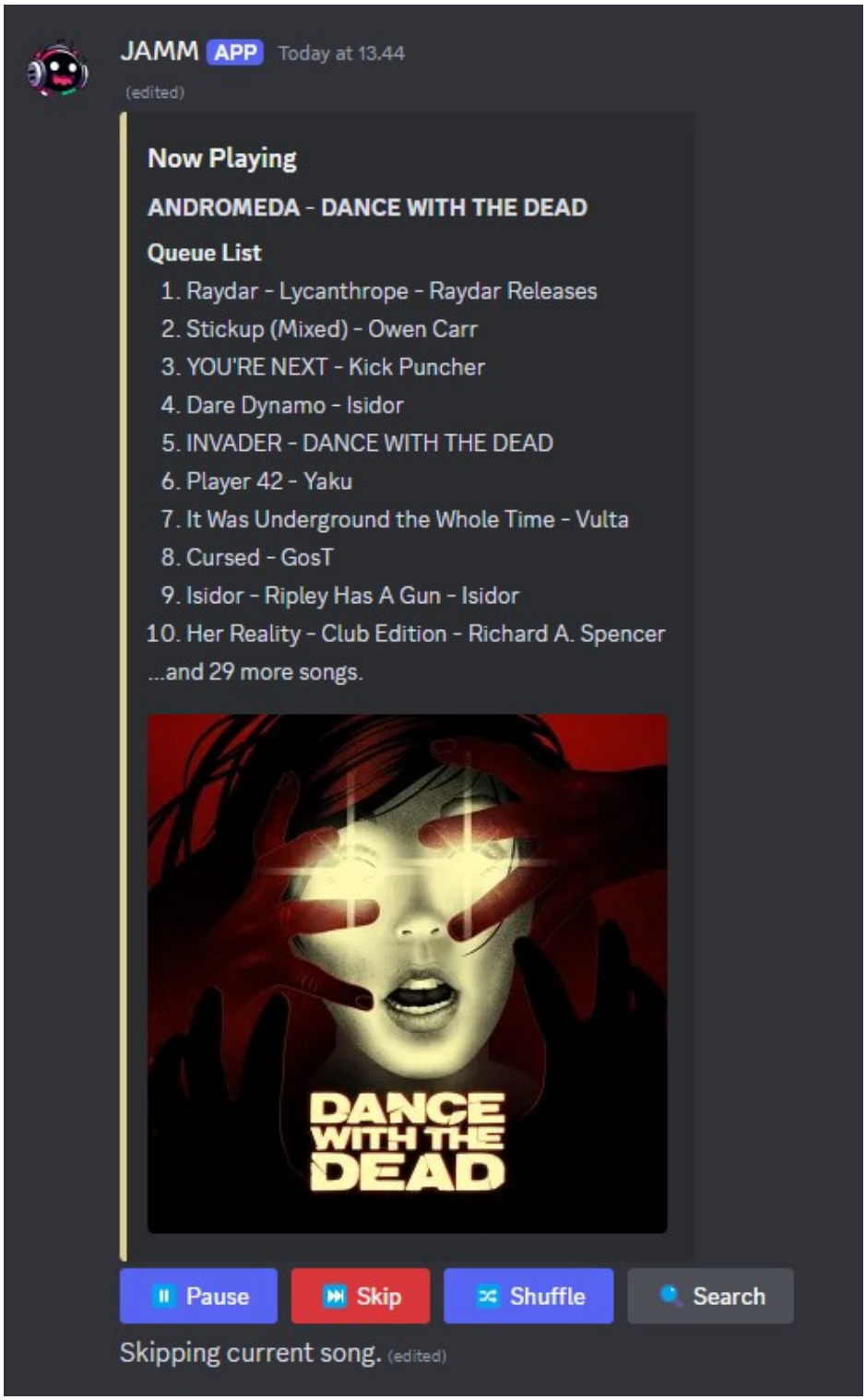
KUVA 9. Näkymä Youtube- ja SoundCloud-hakukentistä.

Käyttöliittymää on parannettu vuorovaikutteisilla elementeillä, kuten painikkeilla ja pudotusvalikoilla, joiden avulla käyttäjät voivat hallita toistoa entistä visuaalisemmin. Esimerkiksi PlaybackControls-painikkeet tarjoavat selkeän tavan hallita toistoa, ohittaa kappaleita ja sekoittaa soittojonon ilman, että käyttäjän tarvitsee käyttää monimutkaisia tekstipohjaisia komentoja. Näiden vuorovaikutteisten elementtien ansiosta botin käyttö on intuitiivista ja vaatii vähemmän teknistä osaamista.

"Now Playing" -toimintoa on myös päivitetty vastaamaan käyttäjien odotuksia visuaalisesti houkuttelevasta musiikkibotista. Tämä toiminto näyttää reaaliaikaisia tietoja toistettavasta kappaleesta, kuten kappaleen nimen, esittäjän ja albumin kansikuvan. Lisäksi botti hyödyntää ColorThief-kirjastoa analysoidakseen kansikuvan pääväriä, jolloin viestin värimaailma mukautuu visuaalisesti kappaleen estetiikkaan. Tämä pieni, mutta visuaalisesti merkittävä lisäys tekee botin käyttöliittymästä houkuttelevamman ja ammattimaisemman. Lisäksi käyttäjät voivat tarkastella soittolistan seuraavia kappaleita suoraan jononäkymästä, mikä lisää läpinäkyvyyttä ja parantaa yhteisön mahdollisuuksia suunnitella musiikkikokemustaan.



KUVA 10. Näkymä soitettavasta kappaleesta kansikuvineen sekä ColorThief-kirjaston avulla haettu pääväri Discordin sulautetun soittimen väripalkkiin.



KUVA 11. Näkymä soitettavasta kappaleesta ja jonosta soittimessa.

Näiden parannusten myötä botista on tullut monipuolisempi ja intuitiivisempi työkalu, joka sopii erilaisten Discord-yhteisöjen tarpeisiin. Soittolistan hallinnan ja käyttöliittymän uudistukset tekevät botin käytöstä miellyttävämpää ja antavat käyttäjille enemmän mahdollisuuksia hallita musiikkia omien mieltymystensä mukaan. Tämä laaja toiminnallisuuksien kirjo nostaa botin perinteisen musiikkibotin tasolta monipuoliseksi ja käyttäjäkeskeiseksi palveluksi..

7.2 Suorituskyvyn optimointi ja turvallisuus

Jatkokehityksen aikana botin suorituskykyä on optimoitu hyödyntämällä ThreadPoolExecutor-luokkaa, joka jakaa tehtäviä useille säikeille. Tämä vähentää pääsilmukan kuormitusta ja parantaa botin reaktiivisuutta erityisesti suurilla palvelimilla tai tilanteissa, joissa käyttäjät antavat useita komentoja samanaikaisesti. Botin kyky käsitellä useita palvelimia on optimoitu siten, että jokaisella palvelimella on omat riippumattomat jononsa, tekstikanavansa ja äänikanavansa. Tämä eristysmekanismi ehkäisee ristiinkytkentäongelmia ja varmistaa, että botin toiminnot pysyvät tarkasti hallittuna.

Turvallisuus on ollut keskeinen osa kehitystä. API-avaimet ja muut kriittiset tiedot tallennetaan turvallisesti ".env"-tiedostoon, josta ne ladataan ympäristömuuttujina load_dotenv-kirjaston avulla. Tämä käytäntö estää tietojen tahattoman vuotamisen julkiseen koodiin. Lisäksi bottiin on implementoitu yksityiskohtainen virhelokitustoiminto, joka tallentaa virheet selkeästi lokitiedostoon. Käyttäjille annetaan samalla ymmärrettäviä virheilmoituksia, mikä vähentää turhautumista ja parantaa käyttäjäkokemusta.

7.3 Kokonaisvaltainen musiikkikokemus

Kaiken kaikkiaan botin kyky tarjota monipuolisia ominaisuuksia, kuten hakutoimintoja, soittolistojen hallintaa ja reaaliaikaisia päivityksiä, tekee siitä

erinomaisen ratkaisun kaikenlaisille Discord-yhteisöille. Integraatiot YouTuben ja SoundCloudin kanssa sekä kyky käsitellä soittolistoja saumattomasti parantavat botin toiminnallisuutta huomattavasti.

Parannukset, kuten ikärajoitettujen videoiden automaattinen ohittaminen ja hakutulosten dynaaminen esittely, takaavat, että musiikkikokemus säilyy sujuvana ja korkealaatuisena. Botin responsiivisuus ja tekninen vakaus yhdistettynä käyttäjäystävällisyyteen tekevät siitä arvokkaan lisän mille tahansa palvelimelle, joka arvostaa musiikin ja yhteisön yhdistämistä.

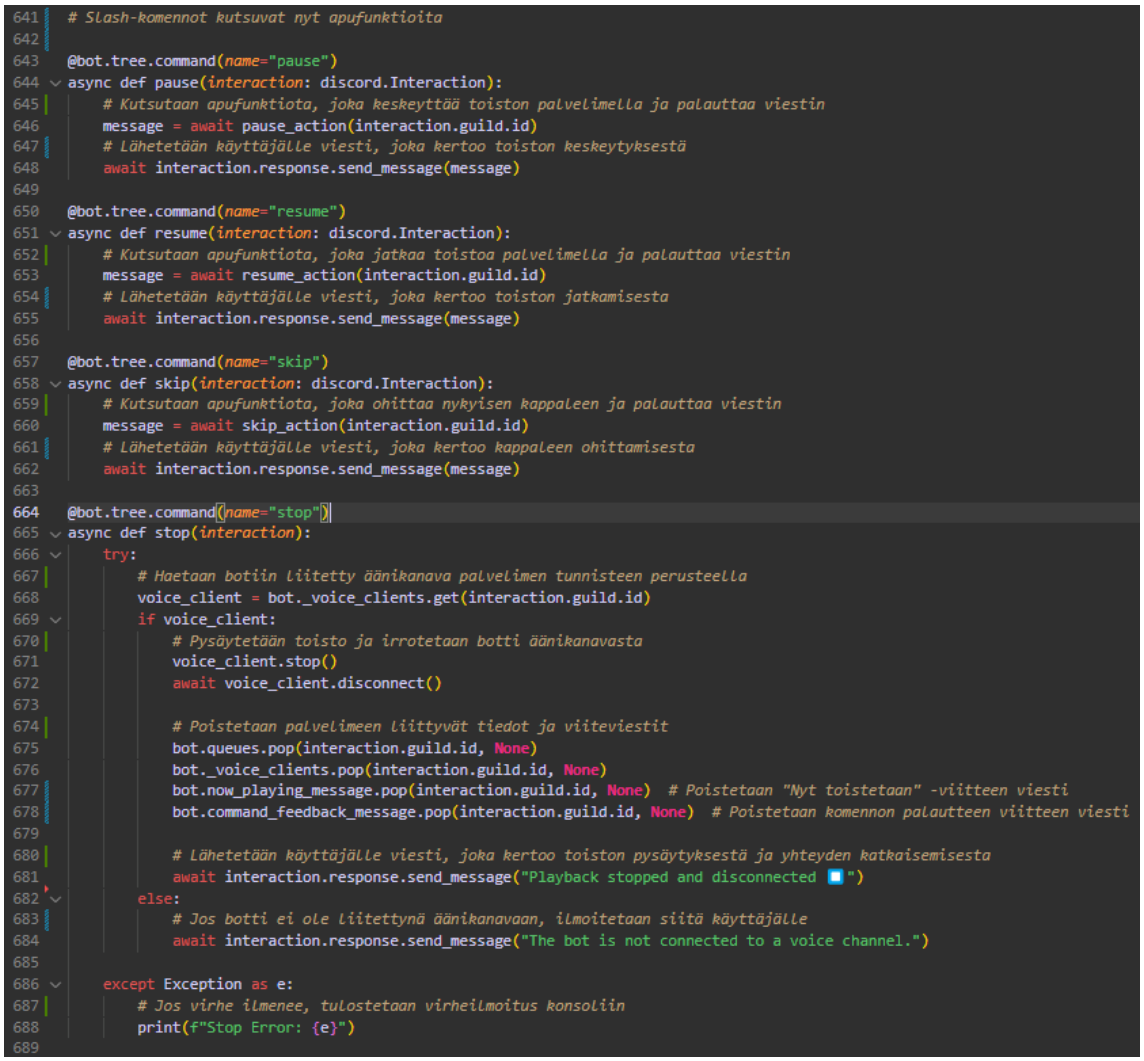
8 MUSIIKKIBOTIN KOMENNOT

Discord-musiikkibotin ydintoiminnallisuus perustuu komentojen avulla suoritettaviin toimintoihin, jotka on toteutettu Discordin tarjoamien API-rajapintojen ja Discord.py-kirjaston avulla. Näiden komentojen avulla käyttäjät voivat hallita musiikkia, kuten soittaa kappaleita, keskeyttää toiston, jatkaa keskeytettyä toistoa, ohittaa kappaleita, lopettaa toiston tai muokata soittolistan järjestystä. Botin komentojen suunnittelu ja toteutus on tehty mahdollisimman intuitiiviseksi, jotta käyttäjäkokemus olisi sujuva ja käyttäjät voisivat ohjata botin toimintaa helposti.

Yksi tärkeimmistä komennoista on “play”-komento, jonka avulla käyttäjä voi aloittaa kappaleen tai soittolistan toiston. Komento vastaanottaa käyttäjän antaman linkin tai hakutermin ja liittää botin tarvittaessa automaattisesti äänikanavaan. Mikäli annettu syöte viittaa soittolistaan, botti lisää kappaleet jonoon ja aloittaa toiston ensimmäisestä kappaleesta. Käyttäjä saa ilmoituksen kappaleen toistamisesta tai jonoon lisäämisestä, mikä parantaa käytettävyyttä ja antaa selkeän palautteen toiminnon onnistumisesta.

“Pause”- ja “resume”-komennot mahdollistavat musiikin toiston keskeyttämisen ja jatkamisen sujuvasti. “Pause”-komento tarkistaa, onko musiikki parhaillaan soimassa, ja jos on, toisto keskeytetään. Tämä tarjoaa käyttäjälle mahdollisuuden hallita musiikkia ilman, että toistoa tarvitsee pysäyttää kokonaan. “Resume”-komento puolestaan tarkistaa, onko toisto keskeytetty, ja jatkaa sen siitä kohdasta, mihin se keskeytettiin. Nämä kaksi komentoa toimivat yhdessä ja tarjoavat käyttäjille joustavuutta hallita musiikkia keskeytysten aikana.

Musiikin hallintaan kuuluu lisäksi “skip”-komento, joka mahdollistaa nykyisen kappaleen ohittamisen ja siirtymisen seuraavaan jonossa olevaan kappaleeseen. Jos jonossa ei ole kappaleita, käyttäjälle ilmoitetaan tilanteesta, mikä auttaa välttämään epäselvyyksiä käytössä. Samoin “stop”-komento lopettaa musiikin toiston kokonaan ja katkaisee botin yhteyden äänikanavaan. Tämä komento myös tyhjentää palvelimelle tallennetun jonon ja palauttaa botin valmiustilaan odottamaan seuraavia käyttäjäkomentoja.



```python
# Slash-komennot kutsuvat nyt apufunktioita

@bot.tree.command(name="pause")
async def pause(interaction: discord.Interaction):
    # Kutsutaan apufunktiota, joka keskeyttää toiston palvelimella ja palauttaa viestin
    message = await pause_action(interaction.guild.id)
    # Lähetetään käyttäjälle viesti, joka kertoo toiston keskeytyksestä
    await interaction.response.send_message(message)

@bot.tree.command(name="resume")
async def resume(interaction: discord.Interaction):
    # Kutsutaan apufunktiota, joka jatkaa toistoa palvelimella ja palauttaa viestin
    message = await resume_action(interaction.guild.id)
    # Lähetetään käyttäjälle viesti, joka kertoo toiston jatkamisesta
    await interaction.response.send_message(message)

@bot.tree.command(name="skip")
async def skip(interaction: discord.Interaction):
    # Kutsutaan apufunktiota, joka ohittaa nykyisen kappaleen ja palauttaa viestin
    message = await skip_action(interaction.guild.id)
    # Lähetetään käyttäjälle viesti, joka kertoo kappaleen ohittamisesta
    await interaction.response.send_message(message)

@bot.tree.command(name="stop")
async def stop(interaction):
    try:
        # Haetaan botiin liitetty äänikanava palvelimen tunnisteen perusteella
        voice_client = bot._voice_clients.get(interaction.guild.id)
        if voice_client:
            # Pysäytetään toisto ja irrotetaan botti äänikanavasta
            voice_client.stop()
            await voice_client.disconnect()

            # Poistetaan palvelimeen liittyvät tiedot ja viiteviestit
            bot.queues.pop(interaction.guild.id, None)
            bot._voice_clients.pop(interaction.guild.id, None)
            bot.now_playing_message.pop(interaction.guild.id, None)  # Poistetaan "Nyt toistetaan" -viitteen viesti
            bot.command_feedback_message.pop(interaction.guild.id, None)  # Poistetaan komennon palautteen viitteen viesti

            # Lähetetään käyttäjälle viesti, joka kertoo toiston pysäytyksestä ja yhteyden katkaisemisesta
            await interaction.response.send_message("Playback stopped and disconnected ⏹")
        else:
            # Jos botti ei ole liitettynä äänikanavaan, ilmoitetaan siitä käyttäjälle
            await interaction.response.send_message("The bot is not connected to a voice channel.")

    except Exception as e:
        # Jos virhe ilmenee, tulostetaan virheilmoitus konsoliin
        print(f"Stop Error: {e}")
```

KUVA 12. Osa botin komentojen koodiosiosta Visual Studio Codessa.

Käyttäjille, jotka haluavat elävöittää soittolistojaan, on tarjolla “shuffle”-komento, joka sekoittaa jonossa olevat kappaleet satunnaiseen järjestykseen. Tämä toiminto tuo vaihtelua erityisesti silloin, kun jonossa on paljon kappaleita. Toteutus käyttää Pythonin “random.shuffle”-funktiota, joka mahdollistaa nopean ja tehokkaan sekoituksen. Käyttäjille ilmoitetaan aina selkeästi, kun jonon järjestys on sekoitettu, mikä auttaa heitä seuraamaan botin toimintaa.

Botin komentojen suunnittelu ja toteutus keskittyvät käyttäjäystävällisyyteen ja tekniseen luotettavuuteen. Kaikissa komennoissa on otettu huomioon virheenkäsittely, jotta käyttäjille annetaan selkeä palaute, jos esimerkiksi syöte on virheellinen tai toistettava kappale ei ole saatavilla. Asynkronisten funktioiden

käyttö Discord.py-kirjastossa takaa, että botti voi käsitellä useita komentoja samanaikaisesti ilman merkittävää viivettä, mikä parantaa palvelimen yleistä käyttökokemusta. Komentojen palautteet, kuten ilmoitukset onnistuneista toiminnoista tai mahdollisista virheistä, on suunniteltu parantamaan käyttäjäkokemusta ja tekemään botin vuorovaikutuksesta sujuvaa ja johdonmukaista.

9 POHDINTA

Tämän opinnäytetyön tavoitteena oli kehittää toimiva ja käyttäjäystävällinen Discord-musiikkibotti hyödyntäen Python-ohjelmointikieltä ja Discordin API-rajapintaa. Projekti tarjosi kattavan näkökulman reaaliaikaisen ohjelmiston kehittämiseen sekä antoi mahdollisuuden syventyä API-rajapintojen tehokkaaseen käyttöön. Valmiin botin kyky toistaa musiikkia sujuvasti sekä YouTubesta että SoundCloudista sekä tukea käyttäjien vuorovaikutteisia toimintoja täytti asetetut tavoitteet ja jopa ylitti odotukset tietyillä osa-alueilla.

Projektin toteutuksessa ilmeni useita haasteita, erityisesti botin suorituskyvyn ja äänenlaadun optimoinnissa. Näihin ongelmiin vastattiin onnistuneesti hyödyntämällä asynkronista ohjelmointimallia ja FFmpeg-teknologiaa, mikä varmisti botin kyvyn käsitellä useita käyttäjäpyyntöjä samanaikaisesti ilman merkittäviä viiveitä tai suorituskykyongelmia. Käyttökokemusta parannettiin kehittämällä visuaalinen käyttöliittymä, joka tekee botin käytöstä intuitiivista ja helppoa myös vähemmän teknisesti suuntautuneille käyttäjille.

Tulosten perusteella botti toimii vakaasti ja tarjoaa laadukkaan musiikkikokemuksen Discord-palvelimilla. Tämän lisäksi sen arkkitehtuuri on skaalautuva, mikä mahdollistaa ominaisuuksien laajentamisen tulevaisuudessa. Kehitystyö paljasti myös mahdollisia parannuskohteita. Esimerkiksi botin kyky hakea automaattisesti genrekohtaista musiikkia tai luoda käyttäjien toiveisiin perustuvia soittolistoja voisi tuoda lisäarvoa. Lisäksi botin integrointi muihin Discord-sovelluksiin, kuten pelibotteihin, voisi tarjota kokonaisvaltaisen viihdekokemuksen käyttäjille.

Kaiken kaikkiaan projektin lopputulos oli onnistunut. Valmis botti parantaa Discord-palvelinten käyttökokemusta tarjoamalla käyttäjille mahdollisuuden nauttia musiikista suoraan palvelimen äänikanavilla. Projekti tarjosi myös runsaasti oppimiskokemuksia, kuten syvempää ymmärrystä Python-ohjelmoinnista, asynkronisista rakenteista ja API-integraatiosta. Työn aikana hankitut taidot ja tiedot ovat suoraan sovellettavissa muihin

ohjelmistokehitysprojekteihin, ja valmis botti tarjoaa hyvän perustan jatkokehitykselle ja uusien ominaisuuksien lisäämiselle. Tämä projekti toimi esimerkkinä siitä, kuinka teknologialla voidaan lisätä arvoa ja parantaa yhteisöllisiä kokemuksia modernissa viestintäympäristössä.

LÄHTEET

[1] Youtube. https://www.youtube.com/. Viitattu 6.2.2025.

[2] SoundCloud. https://soundcloud.com/. Viitattu 6.2.2025.

[3] Discord Developer Portal 2024. Overview of Apps. Discord Inc. Luettavissa: https://discord.com/developers/docs/quick-start/overview-of-apps. Luettu: 25.11.2024.

[4] Discord API -dokumentaatio. API Docs for Bots and Developers. Rate Limits. Discord Inc. Luettavissa: https://discord.com/developers/docs/topics/rate-limits. Luettu: 11.2.2025.

[5] Nottingham, M., Fielding, R. 2012. Additional HTTP Status Codes. IETF. Luettavissa: https://datatracker.ietf.org/doc/html/rfc6585. Luettu: 11.2.2025.

[6] Python 3.13.2 -dokumentaatio 2025. Luettavissa: https://docs.python.org/3/. Luettu 12.1.2025.

[7] Discord.py API -dokumentaatio 2024. API Reference. Luettavissa: https://discordpy.readthedocs.io/en/stable/api.html#. Luettu: 2.12.2024.

[8] FFmpeg-dokumentaatio 2024. Luettavissa: https://ffmpeg.org/documentation.html. Luettu: 12.12.2024.

[9] FFmpeg GitHub -dokumentaatio 2025. Luettavissa: https://github.com/FFmpeg/FFmpeg. Luettu: 26.1.2025.

[10] Discord.js Guide -dokumentaatio 2024. Luettavissa: https://discordjs.guide/. Luettu: 12.12.2024.

[11] Visual Studio Code -dokumentaatio 2025. Luettavissa: https://code.visualstudio.com/docs. Luettu: 14.1.2025.

[12] Github -dokumentaatio 2025. Luettavissa: https://docs.github.com/en. Luettu: 1.2.2025.

[13] Oauth2 -dokumentaatio 2025. Discord Inc. Luettavissa: https://discord.com/developers/docs/topics/oauth2. Luettu: 6.2.2025.