Tampereen ammattikorkeakoulu

Discord applikaation rakentaminen ja toiminnallisuus Pythonilla

Otto Kyösti

OPINNÄYTETYÖ
Toukokuu 2025

Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Ohjelmistotuotanto

KYÖSTI, OTTO:
Discord applikaation rakentaminen ja toiminnallisuus Pythonilla

Opinnäytetyö 37 sivua, joista liitteitä 0 sivua
Toukokuu 2025

Opinnäytetyössä toteutettiin toiminnallinen projekti, jonka tavoitteena oli kehittää oma Discord-applikaatio Python-ohjelmointikielellä ja saada se toimimaan itsenäisesti Linux-pohjaisella pientietokoneella. Työn taustalla oli tarve rakentaa hyödyllinen ja yhteisölle sopiva botti, joka palvelisi ystävien keskeistä omaa Discord-serveriä. Samalla opinnäytetyössä tutkittiin, kuinka Discordin API ja erityisesti Pythonin ohjelmistokirjastot, kuten discord.py, tukevat botin kehittämistä, sekä millaisia teknisiä ratkaisuja ja valintoja kehitysprosessiin sisältyy.

Työssä käsiteltiin laajasti botin kehittämiseen liittyviä vaiheita, kirjastojen valintaa, rakenteen suunnittelua ja komentojen toteutusta. Lisäksi tarkasteltiin botin hostaamista, erityisesti self-hosting-ratkaisuna sekä siihen liittyviä käytännön haasteita ja hyötyjä. Lopputuloksena syntyi käyttövalmis ja ylläpidettävä botti, joka toimii omassa palvelinympäristössään.

Työ palvelee erityisesti henkilöitä, jotka haluavat kehittää oman Discord-botin pienimuotoiseen käyttöön ja oppia käytännön ohjelmistokehitystä. Jatkokehityksessä olisi kiinnostavaa tutkia botin skaalautuvuutta useammalle serverille, suorituskyvyn optimointia sekä mahdollisuuksia hyödyntää koneoppimisen menetelmiä botin toiminnallisuuksissa.

Asiasanat: Discord-applikaatio, Python, ohjelmistokehitys, self-hosting

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Software Production

KYÖSTI, OTTO:
Building and Functionality of a Discord Application Using Python

Bachelor's thesis 37 pages, appendices 0 pages
May 2025

The purpose of this thesis was to develop a functional Discord application using Python programming language and to deploy it on a self-hosted Linux-based single-board computer. The objective was to gain practical experience in bot development and to meet the specific need of a private Discord server community. The project aimed to explore aimed to explore the technical aspects of building a bot, such as command structures, API communication, and hosting strategies. Data was gathered from web articles, guides, and official documentation as well as just advanced knowledge of Python programming language.

The result was a lightweight, modular, and customizable bot that functions reliably in a self-hosted platform, which serves a small community of users. The project demonstrates the strengths of Python as a programming language, the flexibility of software libraries designed for bot development as well as feasibility and benefits of self-hosting, especially for small-scale communities.

In conclusion, the thesis highlights Python’s suitability for Discord bot development and emphasizes the value of self-developing and self-hosting a Discord bot. Future improvements could focus on scalability and the integration of machine learning features to enhance the bot’s functionality.

Key words: Discord-application, Python, software development, self-hosting

SISÄLLYS

LYHENTEET JA TERMIT

VoIP	Suomeksi IP-puhe (englanniksi lyhenne sanoista *Voice over Internet Protocol)* on termi tekniikalle, jolla ääntä voidaan siirtää reaaliaikaisesti Internetin välityksellä
Serveri	Discord palvelin, joka koostuu useista puhe- ja tekstikanavista
Bot	Applikaatio, joka automatisoi normaalisti ihmisten hoitamia tehtäviä
API wrapper	Ohjelmistokirjasto tai -työkalu, joka helpottaa ja yksinkertaistaa Discordin ohjelmistorajapinnan käyttöä
Intent	Kirjaimellisesti suomeksi käännettynä ”aikomus”. Käytetään Discord-botti kontekstissa kuvaamaan botin oikeuksia Discord API:n eri osiin.
Cog	Cog on Discord-botin modulaarinen osa, joka jakaa botin toimintoja eri tiedostoihin selkeyden vuoksi.
Hosting	Hosting tarkoittaa palvelimen tarjoamista sovelluksen, kuten Discord-botin, ajamiseen verkossa.
SSH	Salausprotokolla (Secure Shell Protocol), jolla yhdistetään etänä turvallisesti palvelimeen.
Headless	Järjestelmä ilman graafista käyttöliittymää, käytetään komentoriviltä.

1 JOHDANTO

Kommunikaatio pelaajien välillä videopeleissä on yleinen ja tärkeä osa videopelikokemusta. Suurin osa moninpeleistä tarjoavat pelin sisällä mahdollisuuden pikaviestintään ja muiden pelaajien kanssa puhumiseen, mutta monet pelaajat ovat siirtyneet tai ovat siirtymässä käyttämään kolmannen osapuolen pikaviestintä- ja VoIP-sovelluksia, varsinkin PC-alustalla. Äänen laatu sekä viive, monipuoliset mukauttamismahdollisuudet ja yksityisyys ovat vain muutamia ominaisuuksia, jotka houkuttelevat pelaajia siirtymään kolmannen osapuolen sovelluksiin. Vuosien saatossa on julkaistu monia tällaisia sovelluksia, mutta näistä varmasti monipuolisin ja suosituin on Discord.

Vuonna 2015 julkaistu pikaviestintä- ja VoIP-sovellus Discord nousi nopeasti suosiossa pelaajien joukossa ja jo vuosi julkaisun jälkeen Discordia kutsuttiin parhaimmaksi sovellukseksi kategoriassaan (Marks 2016). Discord alun perin suunniteltiin äänen laatu, viive, turvallisuus ja yksinkertaisuus mielessä ja nykyään näiden lisäksi sovelluksesta löytyy monia käyttöä rikastavia ominaisuuksia. Yksi näistä ominaisuuksista on Discord API ja sen tuomat Discord-botit. Ominaisuuden luoda omia botteja julkaisusta lähtien ihmiset ovat luoneet yli 3 miljoonaa erilaista bottia erilaisiin tarkoituksiin, aina musiikin soittamisesta kokonaisen Dungeons and Dragons roolipelin vetämiseen (Nelly, 2020).

Discord-botit ovat tärkeä osa Discord-sovelluksen käytössä ja tässä opinnäytetyössä tavoitteena on tutustua paremmin Discord-bottien rakenteeseen ja toiminnallisuuteen. Käymme läpi mitä hyötyjä on oman botin rakentamisessa, mistä kaikista osista botti koostuu, sekä kuinka oman botin saa pyörimään omalle paikalliselle palvelimelle. Opinnäytetyössä käytetään ohjelmointikielenä Pythonia, sekä tutustutaan paremmin Pythonin Discord-kirjastoihin, joten perustietämystä Pythonista ohjelmointikielenä on tarpeellinen opinnäytetyötä lukiessa. Opinnäytetyössä on myös mainintaa JavaScript-kirjastoista Discord API:lle, mutta nämä jäävät vain vertailutasolle, sillä keskitymme pelkästään Python-kirjastoihin.

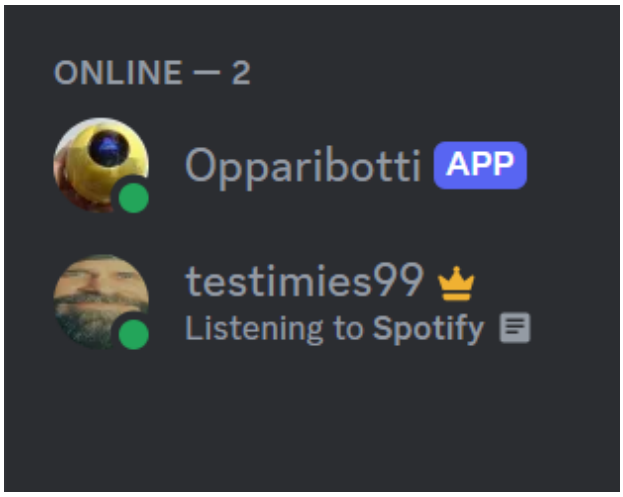
Opinnäytetyön tarkoituksena on luoda toimiva Discord-botti käyttämällä Pythonia ja Discord-botin kehittämiseen tarkoitettuja Python ohjelmakirjastoja. Toimivan

botin lisäksi tarkoituksena on saada botti pyörimään Linux-pohjaisella pientietokoneella, jolloin botti pyörisi omatoimisesti Linux-ympäristössä riippumatta kehittäjän tietokoneen tilasta. Opinnäytetyön tuloksista on hyötyä henkilölle, jolla on kiinnostusta kehittää omaa Discord-bottia vaikkapa kavereiden keskeiselle serverille, ja jolla on jonkinlaista kokemusta ohjelmistokehityksestä Pythonilla.

2 DISCORD JA PYTHON

2.1. Mitä ovat Discord-botit?

Discord-botit ovat käyttäjien kehittämiä applikaatioita, jotka ottamalla yhteyden Discordin ohjelmointirajapintaan eli API:iin, pystyvät suorittamaan automatisoituja tehtäviä serverillä, jolle botti on kutsuttu. Serverin käyttäjälistassa botti näyttää aivan tavalliselta käyttäjältä, mutta käyttäjänimen vierestä löytyy isoin kirjaimin ”APP”, joka indikoi, että kyseessä on bottikäyttäjä (kuva 1). Tämä tarkoittaa, että botteja pystyy ajamaan pienellä muokkaamisella myös tavallisilta käyttäjätileiltä, mutta tämä on vastoin Discordin käyttöehtoja. Käyttöehdot edellyttävät, että kaikilla boteilla on oltava oma bottikäyttäjätili (Guru Staff, 2022)



KUVA 1. Serverin käyttäjälista

Koska bottikäyttäjätili on vain tavallinen käyttäjätili naamioituneena bottikäyttäjätiliksi, on botilla sille annettujen käyttöoikeuksien puitteissa samat mahdollisuudet olla vuorovaikutuksessa muihin käyttäjiin kuin tavallisella käyttäjällä. Boteilta löytyy perustoimintojen lisäksi ominaisuuksia, joita tavalliset käyttäjät eivät pysty tekemään, esimerkiksi viestin massapoistaminen, mutta turvallisuuden ja yksityisyyden takia botti ei esimerkiksi pysty lukemaan tavallisten käyttäjien profiileja ja sieltä löytyviä linkitettyjen käyttäjätilien nimiä. Bottia kutsuessa serverille, sille annetaan käyttöoikeudet, jotka määrittävät kuinka paljon oikeuksia botilla on kyseisellä serverillä. Botille voi myös antaa järjestelmänvalvojan oikeudet serverillä, jolloin sillä on kaikki mahdolliset oikeudet muokkaa ja poistaa käyttäjiä.

Suuret serverit, yleensä jonkin tietyn aiheen kommuunit, käyttävät Discord-botteja paljolti järjestyksenvalvontaan sekä organisoimiseen. Botit pystyvät esimerkiksi vahtimaan viestejä tekstikanavilla ja poistaa sopimattomia viestejä, jos sellaisia tulee. Yritykset voivat valjastaa botteja esimerkiksi parantamaan asiakaspalvelua, automatisoimaan markkinointia tai keräämään dataa (Ullah, 2023). Discord-botteja pystyy käyttämään myös viihdetarkoitukseen. Musiikin soittaminen ja tekstipohjaisten pelien pelaaminen on mahdollista hoitaa kokonaan botin sisäisesti.

Bottien todellinen voima ja hyöty tulee esiin, kun niitä käytetään jonkin toisen rajapinnan yhteydessä. Otetaan esimerkiksi projektinhallintasovellus Trello. Integroimalla Trellon Discord-bottiin, botti pystyy lataamaan kaikki projektin päivitykset suoraan Discord serverille, josta käyttäjät pääsevät niihin suoraan käsiksi, ilman että jokaisen tarvitsee käydä itse nettisivulla katsomassa päivityksiä. (Guru Indights, 2022) Samalla tyylillä toimii myös musiikin soittaminen YouTubesta. Botin avulla, käyttäjät pystyvät kuuntelemaan YouTubesta musiikkia yhdessä ilman, että jokaisen tarvitsee käydä itse laittamassa musiikki päälle samaan aikaan. Botin hyöty ja monipuolisuus riippuu kuitenkin paljon tarpeesta, sekä botin kehittäjän taidoista.

2.2. Python vai JavaScript?

Ennen kuin Discord-bottia voi ryhtyä suunnittelemaan ja rakentamaan, tarvitsee valita ohjelmointikieli, millä koko botti kehitetään. Kaksi suosituinta ohjelmointikieltä Discord-botin kehittämiseen ovat Python ja JavaScript. Nämä kaksi kieltä ovat suosituimpia, koska kieliä tukevat Discordin API wrapperit ovat yksinkertaisimpia ymmärtää ja vaativat vähiten kokemusta peruskoodaamisen ulkopuolelta. Nämä wrapperit, eli ohjelmistokirjastot tai -työkalut, tarjoavat yksinkertaistettuja komentoja ja toimintoja, jotka helpottavat kehittäjiä ja heidän bottejansa Discordin kanssa kommunikoinnissa. Ilman niitä, kehittäjien täytyisi suoraan käsitellä raakoja HTTP-pyyntöjä ja vastauksia, mikä voi olla monimutkaista ja virhealtista. Tämän takia Discord-botteja suositellaan kehitettävän joko Pythonilla tai JavaScriptillä. Botin toimintojen näkökulmasta kielellä ei ole mitään väliä, molemmat kielet

tarjoavat samat ominaisuudet botille, joten valinta näiden kahden välillä riippuu paljon omasta kokemuksesta.

2.2.1 Python

Python itsessään on erittäin suosittu sekä helposti opeteltavissa oleva ohjelmointikieli. Pythonilta löytyy monta ohjelmistokirjastoa Discord-bottien kehittämiseen kuten discord.py, Pycord ja Nextcord. Pythonilla Discord-bottien kehittäminen ei kuitenkaan ole ollut pelkkää ruusuilla tanssimista. Vuonna 2021 suosituimman ohjelmistokirjaston discord.py:n ainoa ylläpitäjä päätti lopettaa kirjaston tukemisen (Ganesh, 2023). Tämä sai jotkut kehittäjistä luomaan omia ohjelmistokirjastoja, kuten Pycord ja Nextcord, ja sai jotkut kehittäjistä vaihtamaan suoraan Pythonista JavaScriptin ohjelmistokirjastoihin. Discord.py-ohjelmistokirjaston ylläpitäjä on sen jälkeen jatkanut kirjaston ylläpitoa ja vielä vuoden 2023 keskivaiheilla discrod.py oli toiseksi suosituin Discord-ohjelmistokirjasto GitHub-arvostelujen perusteella (Rothwell, 2023).

Python on helposti luettava ohjelmointikieli, josta on apua, jos esimerkiksi oman Discord-projektin monimutkaisuus ja laajuus nousee huomattavasti. Discord.py-ohjelmistokirjasto tuo kehitykseen myös laajamittaisen kehyksen komentojen luomiseen, joka helpottaa huomattavasti erityisesti monimutkaisien projektien kehittämistä. Pythonilla kehitetyt botit voivat myös hyödyntää Pythonin datankäsittely- ja koneoppimisominaisuuksia, mistä Python on hyvin tunnettu. Botista voi kuitenkin tulla erittäinkin monimutkainen Pythonilla tehdessä, jos vähimmäismäärällä kokemusta lähtee kehittämään. Varsinkin Pythonilla kehitettyjen bottien käyttämä async/await-syntaksi saattaa olla erittäin haastava konsepti ymmärtää.

2.2.2 JavaScript

JavaScript on myös yksi käytetyimmistä ohjelmointikielistä maailmalla, eikä ole yllätys, että JavaScriptiltä löytyy myös kattavia Discord-ohjelmistokirjastoja. JavaScriptin johtava Discord-ohjelmistokirjasto discord.js oli vielä vuoden 2023 keskivaiheilla kaikkein suosituin ohjelmistokirjasto GitHub-arvioiden perusteella (Rothwell, 2023). JavaScript ohjelmointikielenä on sanottu olevan helppo kieli

opetella, mutta vaikea kieli hallita täysin. Juuri tämän takia JavaScript onkin suosituin ohjelmointikieli ensi kertaa Discord-bottia rakentaville kehittäjille. Discord-bottia pystyy kehittämään myös TypeScriptillä, JavaScriptin sääntöjä valvovalle sisarkielelle, mutta vaikeuksia saattaa tulla JavaScriptille tarkoitettujen ohjelmistokirjastojen yhteensopivuudesta.

JavaScriptin Discord-ohjelmistokirjastot ja varsinkin discord.js ovat kehitetty tehokkuus mielessä. Kielen asynkronisuus ja kielen mahdollisuus integroida muitakin web-pohjaisia API-palveluita ovat hyödyllisiä ominaisuuksia boteille suuren käyttäjämäärän servereillä, jossa botin tarvitsee reagoida moneen tapahtumaan yhtäaikaisesti. Toisin kuin Pythonissa, JavaScriptin ohjelmistokirjastoissa ei ole minkäänlaista helppoa ja laajaa kehystä Discord-komentojen luomiseen, vaan kaikki niihin liittyvä tarvitsee tehdä itse, joka voi nostaa kynnystä JavaScriptin valitsemiseen.

2.3. Sopivan Python-kirjaston valinta

Yhtä tärkeää kuin kehittäjälle sopivan ohjelmointikielen valitseminen on sopivan Pythonille rakennetun Discord-ohjelmistokirjaston valitseminen. Kirjaston valitsemisessa tärkein asia mihin kannattaa kiinnittää huomiota on, että ylläpidetäänkö kirjastoa ja kuinka usein sitä päivitetään. Ohjelmistokirjastoa, jota ei enää päivitetä, ei ikinä kannata valita, sillä vaikka ne saattavat tällä hetkellä toimia, ne voivat lopettaa toimintansa heti kun Discord päivittää heidän API:ansa. Tämän takia on tärkeää, että joku on ylläpitämässä kirjastoja, jotta botit, jotka käyttävät niitä, eivät hajoa yhtäkkiä. Kaikki Pythonin ohjelmistokirjastot Discord API:lle tarjoavat samat ominaisuudet vaikkakin koodi ja syntaksi eroavat jokseenkin toisistaan.

2.3.1 discord.py

Discord.py on tällä hetkellä johtava ja suosituin Python-ohjelmistokirjasto Discord-bottien kehittämiseen (Rothwell, 2023). Itse asiassa suurin osa kilpailevista ohjelmistokirjastoista, kuten Pycord ja Nextcord, ovat forkattuja, eli alkuperäisen ohjelmiston kopion pohjalta kehitettyjä, projekteja. Näitä ohjelmistokirjastoja alkoi syntymään heti aikaisemmin mainitun discord.py:n ylläpitäjän päätöksen lopettaa

ohjelmistokirjaston tukeminen jälkeen. Tämän jälkeen ylläpitäjä on ilmoittanut jatkavansa discord.py-ohjelmistokirjaston ylläpitämistä ja se onkin nykyään suurin ja ylläpidetyin Python-ohjelmistokirjasto Discordille.

Ohjelmistokirjastona discord.py on erittäin laaja ja monipuolinen. Kirjastosta löytyy laajamittainen kehys komentojen luomiseen ja tuki myös slash-komentojen luomiseen. Async/await-syntaksi, vaikka ehkä hieman haasteellinen opetella ja ymmärtää, auttaa botin asynkronisessa toiminnassa ja mahdollistaa samanaikaisen vastaamiseen moneen tapahtumaan, joka on hyödyllistä varsinkin suurien käyttäjämäärien servereillä. Kirjaston dokumentaatio on vapaasti saatavilla ja hyvin rakennettu, joten informaation löytäminen on tehty myös erittäin helpoksi. Tässä opinnäytetyössä käytetään pääosin discord.py-ohjelmistokirjastoa kommunikoinnissa Discordin API:n kanssa.

3 BOTIN RAKENNE JA TOIMINNALLISUUS

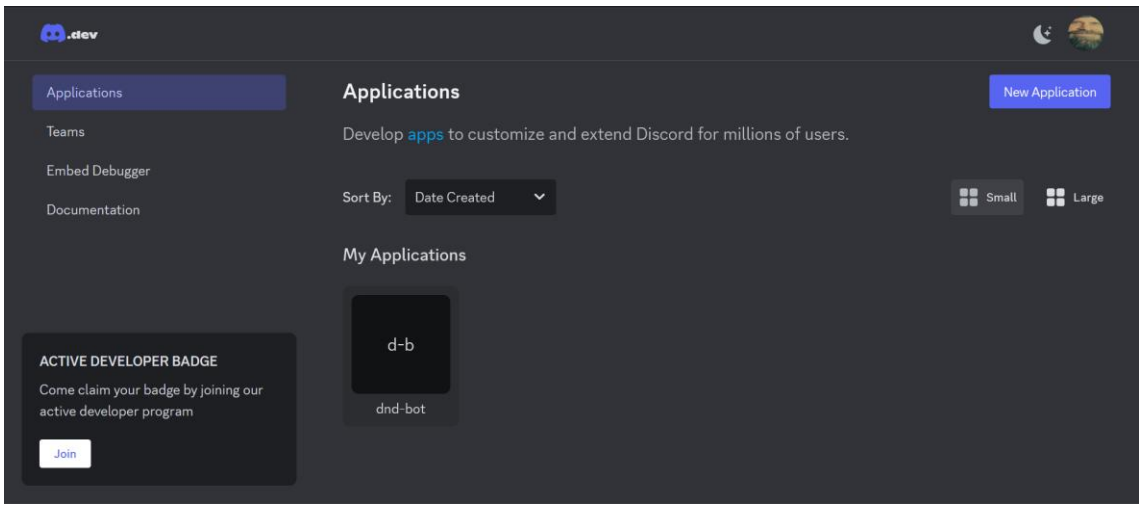
3.1. Esivaatimukset

Jos haluaa lähteä rakentamaan omaa Discord-bottia, ensiksi tarvitsee täyttää muutamat esivaatimukset. Tärkein näistä on jonkinlainen kokemus ohjelmoinnista Pythonilla. Vaikka on mahdollista kehittää Discord-botti erittäin vähällä tai jopa olemattomalla kokemuksella Pythonista, se voi johtaa siihen, että aikaa kuluu perusvirheiden selvittelyyn ja korjaamiseen, mikä voi aiheuttaa turhautumista. Jonkinlainen Python-osaaminen on tärkeää, jotta kehittäminen olisi sujuvaa. Muita esivaatimuksia Discord-botin kehittämiseen ovat bottikäyttäjätilin luominen, Discord-serverin luominen sekä oikeiden ohjelmistokirjastojen asentaminen.

3.1.1 Discord Developer Portal

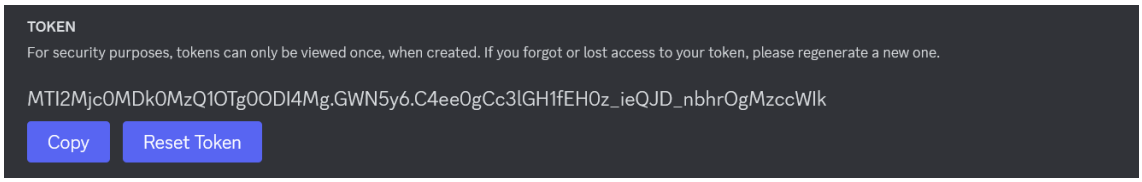
Discord Developer Portal on Discordin käyttämä, nettisivulla toimiva sovellusalusta, joka on kehittäjille tarkoitettu alusta applikaatioiden, jotka laajentavat tai mukauttavat Discordia, kehittämiselle. Nettisivulta löytyvien resurssien ja API-työkalujen avulla kuka tahansa voi luoda Discord-botteja, integroida kolmannen osapuolen palveluita itse Discordiin tai kehittää Discord-tuki mihinkä tahansa videopeliin. Jos Discord-bottia haluaa lähteä kehittämään, täytyy se aloittaa Discord Developer Portal:ista.

Päästäksesi käyttämään Discord Developer Portal:ia, tarvitsee sinulla olla oma Discord-käyttäjä. Kirjautumisen jälkeen aukeaa nettisivun pääsivu, jossa painamalla ”New Application”-nappia nettisivun oikeassa yläreunassa (kuva 2) pääsee luomaan applikaation. Mikä tahansa ohjelma, joka on vuorovaikutuksessa Discord API:n kanssa tarvitsee Discord-applikaation alustaksi, eivät pelkästään Discord-botit. Botteihin liittyvät rajapinnat ovat vain osa Discordin kokonaista rajapintaa. (Ronquillo, n.d.)



KUVA 2. Discord Developer Portal:in pääsivu

Applikaatiolle tarvitsee antaa nimi ja Discordin kehittäjäehdot ja -käytännöt täytyy hyväksyä. Tämän jälkeen aukeaa applikaatio-näkymä. Tällä sivulla näkyy ja pystyy muokkaamaan yleisiä tietoja liittyen luotuun applikaatioon. Discord-botin tietoja ja kaikista tärkeimmän, bot-tokenin, pääsee näkemään klikkaamalla ”Bot”-kategoriaa vasemmalla olevasta menusta. Discord-botti luodaan automaattisesti applikaation luomisen yhteydessä ja botti perii applikaation nimen. Tällä sivulla tärkeimmät kohdat ovat botin token (kuva 3), sekä ”Privileged Gateway Intents” eli asetukset, joissa muutetaan Discord-botin kykyä vastaanottaa Discordin lähettämiä mahdollisesti arkaluonteista informaatiota sisältäviä tapahtumia (Discord.py, n.d.).



KUVA 3. Esimerkki Discord botin tokenista

Painamalla ”Reset Token”-nappia saa esille botille tarkoitetun numerojen, merkkien ja kirjaimien sarjan. Se pitää ottaa talteen, sillä sitä käytetään koodissa kertomaan mihin bottiin koodinpätkä yhdistää. Tokenin pystyy näkemään vain kerran ja jos sen unohtaa, pitää token käydä resetoimassa, jolloin saa kokonaan uuden tokenin. Tämä token täytyy sen jälkeen käydä päivittämässä kaikkialla koodissa missä sitä käytetään. ”Privileged Gateway Intents”-asetukset kannattaa kaikki

varmuuden vuoksi laittaa päälle, jotta Discord-botti pystyy varmasti vastaanottamaan kaikkia mahdollisia tapahtumia.

3.1.2 Kirjastot

Kuten aikaisemmin opinnäytetyössä on mainittu, pääohjelmistokirjasto, jota tässä opinnäytetyössä tullaan käyttämään, on discord.py. Tämä ohjelmistokirjasto sisältää kaiken tarvittavan Discord API:iin yhdistämiseen, botin yleisen olemuksen muokkaamiseen, puhekanaville liittymiseen sekä komentojen tekemiseen. Riippuen ominaisuuksista, mitkä botille haluaa kehittää, on hyvinkin mahdollista, että vain discord.py ei riitä. Tässä opinnäytetyössä tullaan discord.py-ohjelmistokirjaston lisäksi käyttämään dotenv-ohjelmointikirjastoa, joka auttaa pitämään arkaluonteiset tiedot, kuten botin tokenin, pois peruskoodista, sekä helpottaa koodin luettavuutta pitämällä pitkät ja toistuvat muuttujat pois peruskoodista.

Hyvin yleisiä ominaisuuksia, ja varmaankin myös suosituimpia, ovat multimediaan liittyvät ulkoiset integraatiot, kuten esimerkiksi YouTube-videoiden toistaminen tai musiikin soittaminen Spotifysta. Näiden ominaisuuksien toteuttamiseen yleensä tarvitaan juuri kyseessä olevan ulkoisen resurssin kanssa vuorovaikuttamiseen tarkoitettuja kirjastoja. Tällaiset kirjastot tarjoavat valmiita rajapintoja, joiden avulla voidaan hakea, toistaa ja hallita sisältöä suoraan Discordin käyttöliittymästä ilman, että käyttäjän itse tarvitsee siirtyä toiseen palveluun. Kaikille palveluille ei kuitenkaan omia ohjelmistokirjastoja löydy, jolloin vaihtoehdoksi jää kirjaston kehittäminen itse, mutta tämä yleensä vaatii ohjelmoijalta laajempaa ja syvempää kokemusta muistakin ohjelmointikielistä, kuin Pythonista (Sidak, 2024).

3.2. Rakenne

Kun Discord-botin token on saatu talteen ja kaikki tarvittavat kirjastot on asennettu, voi itse botin toiminnallisuutta lähteä koodaamaan. Pythonille on monia ohjelmointiympäristöjä mistä valita, jotka tarjoavat hieman erilaisia ominaisuuksia. Tässä opinnäytetyössä käytetään Microsoftin luomaa Visual Studio Code:a. Se

on ilmainen, kevyt prosessorille, sekä tarjoaa ominaisuuksia, joista muissa ohjelmointiympäristöissä joutuisi maksamaan kuten esimerkiksi Git-integraation ja yhden älykkäimmistä koodintäydennyksistä (Gupta, 2024).

Karkeimmillaan Discord-botin pyörittämiseen tarvitsee vain yhden Python-tiedoston. Yksi tiedosto riittää yleensä kaikkein yksinkertaisimmille Discord-boteille ja riippuen siitä, kuinka monimutkaisen botista haluaa tehdä, Discord-botin toiminnallisuus saattaa kattaa useamman tiedoston. Toisaalta yksinkertaisimmankin botin toiminnallisuuden jakaminen eri tiedostoihin on ihan viisasta, sillä tämä auttaa projektikansion organisoinnissa.

3.2.1 Bot.py

Discord-botin päätiedosto on hyvä nimetä lyhyesti ja ytimekkäästi, kuten esimerkiksi bot.py. Tämä tiedosto on kaikkein olennaisin tiedosto botin pyörittämisessä, sillä se sisältää yleensä botin luomiskoodin (kuva 4), jossa määritetään mitä prefiksiä viestikomennot käyttävät sekä ”intents”, eli tapahtuma-aikomukset, jotka määrittävät mitä Discord API:n tapahtumia, kuten viestin lähettämistä tai käyttäjän liittymistä serveriin, botti kuuntelee, jotta se pystyy reagoimaan niihin. Päätiedosto sisältää usein myös kaikki botin käyttämät viesti- sekä slash-komennot, ellei niitä ole suuria määriä.

```python
# bot.py
import discord
from discord.ext import commands

import os
from dotenv import load_dotenv
load_dotenv()

token = os.environ.get("DISCORD_TOKEN")
intents = discord.Intents.default()
bot = commands.Bot(command_prefix = "!", intents = intents)

@bot.event
async def on_ready():
    print(f"Logged on as {bot.user.name}")

bot.run(token)
```

KUVA 4. Yksinkertainen bot.py-tiedosto

Kuva 4 illustroi erittäin yksinkertaisen Discord-botin, jonka voi käynnistää komentoriviä käyttäen. Botti käyttää dotenv-ohjelmistokirjastoa tallentamaan botin tokenin, jonka se lataa .env-nimisestä tiedostosta koodissa käytettäväksi. Discord-botin luominen vaatii aina ”intents”:ien esittämisen ennen bot-muuttujan luomista ja ajamista. Jos ei tiedä minkälaisiin tapahtumiin oma Discord-botti tulee reagoimaan, on hyvä antaa botille oletusarvoiset ”intents”:it, kuten kuvassa 4 on tehty rivillä 10. Riviltä 13 riville 15 on esitetty ”on ready”-niminen tapahtuma, joka tulostaa komentoriville tekstiä, kun Discord-botti on valmis ottamaan tapahtumia vastaan Discord-serverillä.

3.2.2 Komentojen rakenne

Yksi erittäin tärkeä osa Discord-botin rakennetta ovat komennot. Komennot ovat botin sisäisiä funktioita, joita käyttäjä voi kutsua viestipohjaisesti tai käyttämällä Discordin omaa käyttöliittymää. Komennot jaetaan kahteen eri kategoriaan: viestikomentoihin ja slash-komentoihin. Nämä kaksi komentotyyppiä erottaa siitä, kuinka käyttäjä kutsuu niitä Discordissa. Käyttäjä kutsuu viestikomentoja käyttäen ennalta määritettyä etumerkkiä ja kirjoittamalla komennon nimen Discordin viestikenttään, sekä mahdolliset argumentit. Slash-komennot toimivat hieman eri tavalla. Käyttäjä kutsuu slash-komentoja käyttämällä ”/”-etumerkkiä, joka avaa Discordin oman käyttäjäliittymän, josta löytyy kaikki botin omaavat slash-komennot. Näitä komentoja voi käyttää samalla lailla kuin viestikomentojakin, mutta Discordin oma käyttöliittymä auttaa paljon siinä, kuinka komentoa käytetään, kertomalla kuvauksessa komennon tarkoituksesta, sekä kertomalla argumenttien nimet niitä kirjoittaessa. Näiden komentotyyppien lisäksi on vielä kolmas komentotyyppi, hybridikomento, jota käyttäjä voi kutsua joko viestikomentona tai slash-komentona.

Koodipuolella komentotyypit kirjoitetaan myös eri tavalla, mutta niissä käytetään samanlaista rakennetta (kuva 5). Molemmat kirjoitetaan asynkronisina funktioina ja käyttävät discord.py-ohjelmistokirjaston komentokehyksestä löytyviä dekoraattoreita, jotka mahdollistavat kehittäjän muokata jo olemassa olevan funktion käyttäytymistä muuttamatta alkuperäisen funktion rakennetta ja näin ollen parantaa

ja laajentaa funktion toiminnallisuutta (Chng, 2023). Tämä tarkoittaa, että molemmat komentofunktiot lisäävät toiminnallisuutta jo valmiiksi discord.py-kirjastosta löytyvään komentofunktioon.

```
18  @bot.tree.command(name = "Ping!", description = "Pings the bot!")
19  async def ping_slash_komento(interaction: discord.Interaction):
20      await interaction.response.send_message("Pong! " + bot.latency)
21
22  @bot.command()
23  async def ping_viesti_komento(ctx: commands.Context):
24      await ctx.send("Pong!" + bot.latency)
```

KUVA 5. Komentojen rakenne koodissa, ylempänä slash-komento, alempana viestikomento

Slash- ja viestikomennot näyttävät hyvin samanlaisilta kirjoitettuna koodiksi, mutta eroja löytyy esimerkiksi viestiin vastaamisessa ja komentodekoraattorin käytössä. Kun slash-komentoa käytetään, botti lähettää komennon tuloksen vastauksena alkuperäiselle viestille, jolloin on helppo huomata mihin komentoon botti vastasi. Viestikomento lähettää vain uuden viestin viestiketjuun, josta voi olla vaikea huomata mihin viestiin botti vastasi, varsinkin jos viestiketjuun tulee monta viestiä samaan aikaan. Jotta botti pystyy vastaamaan viesteihin, botilla pitää olla ”message_content”-intent laitettuna päälle.

Viestikomennot käyttävät tavallista komentodekoraattoria ilman argumentteja, mutta slash-komennot käyttävät hieman erilaista versioita argumenteilla. Näitä argumentteja, yleensä nimi ja kuvaus, käytetään komennossa kertomaan tietoa komennosta, jotta se olisi helppokäyttöisempi käyttäjälle. Viestikomentoja voi olla vaikea käyttää, ellei niitä varten tee jotain apukomentoa, joka selittää jokaisen komennon käytön. Koska slash-komennot ovat integroitu Discordin käyttöjärjestelmään, niille annetut nimet ja kuvaukset antavat selkeän kuvan siitä mihin ja miten komentoa käytetään. Komentojen integroiminen Discordin käyttöjärjestelmään vaatii myös komentojen synkronoimista aina kun komentoja lisätään, poistetaan tai muokataan, joka on pakollista, jotta muutokset näkyvät Discordissa, mutta tämä hoituu yleensä helposti yhdellä rivillä koodia (kuva 6).

```
await bot.tree.sync()
```

KUVA 6. Koodinpätkä slash-komentojen synkronointiin

3.2.3 Äänen toistaminen

Riippuen minkälaisia toimintoja Discord-bottiin haluaa kehittää, on hyvin mahdollista, että jokin niistä vaatii äänen toistamista. Discord-botit ovat kykeneviä toistamaan ääntä liittymällä Discord-serverin puhekanavalle ja soittamaan joko äänitiedoston tai striimaamaan äänen jostain palvelusta url:n kautta, kuten esimerkiksi YouTubesta. Tämä vaatii lisäohjelmien asentamista ja YouTuben tapauksessa lisäohjelmistokirjastojen asentamista.

Äänen toistamiseen Discord-botti tarvitsee kolme asiaa: äänikanavan, jonne liittyä ja toistaa ääni, itse äänitiedoston tai url:n, josta toistaa ääntä ja multimediakehyksen, joka osaa sekä koodata, että tulkita erilaisia äänitiedostoja. Äänikanava ja äänitiedosto määritellään koodipuolella (kuva 7). Data tietystä äänikanavasta, jolle botin haluaa liittyvän, voi olla vaikea löytää, jonka takia hyvänä sääntönä on pitää, että ääntä toistavaa komentoa voi kutsua vain käyttäjä, joka on sillä hetkellä itse puhekanavalla. Näin data siitä, mille puhekanavalle botin pitää liittyä, tulee komentoa kutsuvan käyttäjän mukana.

```
27  @bot.hybrid_command(name = "Soundbite", description = "Plays a soundbite in a voice channel")
28  async def soundBite(ctx: commands.Context):
29      if ctx.author.voice and ctx.author.voice.channel:
30          await ctx.defer()
31          voice_channel = ctx.author.voice.channel
32          voice_client = await voice_channel.connect()
33
34          if not voice_client.is_playing():
35              voice_client.play(discord.FFmpegPCMAudio("chipi.mp3"))
36              while voice_client.is_playing():
37                  await asyncio.sleep(1)
38          else:
39              await ctx.send("I'm already playing!")
40
41          await voice_client.disconnect()
42      else:
43          await ctx.send("You must be in a voice channel to use this command!")
```

KUVA 7. Esimerkki ääntä toistavasta komennosta

Kuvassa 7 näkyvä koodinpätkä on tiivis esimerkki ääntä toistavasta komennosta. Koodissa ensiksi katsotaan, onko käyttäjä sillä hetkellä puhekanavalla. Jos käyttäjä on, koodi antaa rivillä 30 tiedon Discordin API:lle, että tällä koodilla voi kestää komennon prosessoinnissa kauemmin kuin yleensä. Jos tätä ei tee, botin ko-

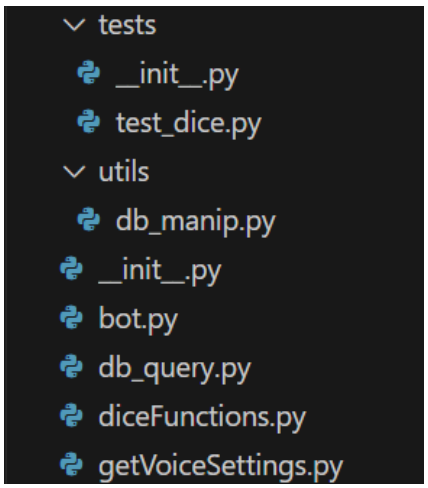
mento aikakatkaistaan, jos siinä menee kauemmin kuin normaalisti. Tämän jälkeen käyttäjältä saadaan puhekanava muuttujaan ja botti yhdistää puhekanavaan. Jos botti ei jo toista ääntä, botti lähtee toistamaan koodissa indikoitua äänitiedostoa. Samalla kun botti toistaa ääntä, koodi tarkastaa joka sekunti toistaako botti vieläkin ääntä. Kun botti ei enää toista ääntä, botti katkaisee yhteyden puhekanavaan.

Tärkeä osa äänikomennon toimintaa on FFmpeg-multimediakehys. FFmpeg on johtava multimediakehys, joka koodaa, tulkitsee, striimaa ja toistaa erilaisia ääniformaatteja. Ilman tätä multimediakehystä kuvassa 7 oleva äänikomento ei pysty toistamaan ääntä. FFmpeg tarvitsee erikseen ladata omalta nettisivultaan ja asentaa oman tietokoneen Path-ympäristömuuttujaan, jotta Discord-botin koodi osaa muuntaa äänitiedostot oikeaan formaattiin ja toistaa niitä Discord-serverillä.

3.2.4 Tiedostorakenne

Yksinkertaisimmillaan Discord-botin pyörittämiseen tarvitaan ainakin yksi päätiedosto, joka sisältää vähintään botin ajamiskomennon sekä siihen liittyvän autentikoinnin käyttäen generoitua tokenia. Tämä päätiedosto voi alkuvaiheessa sisältää myös kaikki botin ominaisuudet ja komennot, mutta botin ominaisuuksien ja komentojen lisääntyessä ja logiikan muuttuessa monimutkaisemmaksi on hyvä alkaa pilkkomaan toimintoja osiin ja luomaan omia tiedostoja eri toiminnoille. Tämä on yksi peruskivi selkeän ja ylläpidettävän koodin kirjoittamisessa.

Päätiedoston osiin pilkkomisessa on tärkeää pilkkoa koodia yksittäisiin toimintoihin ja toimintoalueisiin. Tämä noudattaa niin sanottua ”concerns” periaatetta, jossa jokainen tiedosto tai moduuli vastaa yhdestä selkeästä tehtävästä (Diazbeltran, 2022). Tällainen lähestymistapa, ja tiedostojen ja kansioiden nimeäminen niiden sisältämien toimintojen ja ominaisuuksien mukaan, ei ainoastaan helpota koodin ymmärtämistä ja ylläpitoa, vaan myös mahdollistaa sen, että eri toimintojen logiikkaa voidaan uudelleen käyttää eri tilanteissa. Esimerkiksi tiedostosta, joka on nimetty ”diceFunctions.py”-nimiseksi, voidaan jo nimen perusteella olettaa, että tämä tiedosto sisältää funktioita liittyen nopan heittämiseen (kuva 8).



KUVA 8. Esimerkki koodin pilkkomisesta osiin

Discord.py ottaa koodin osiin pilkkomisen vielä seuraavalle tasolle tarjoamalla kätevän rakenteen: Cogit. Cogien avulla komentoja ja kuuntelijoita voidaan ryhmitellä loogisesti yhteen. Tämä ei ole pelkästään tiedostollinen jako, vaan ne mahdollistavat toiminnallisuuksien kapseloinnin objektiorientoituneella tavalla. Cogit jaetaan yleensä komentoryhmiin eikä yksittäisten komentojen perusteella. Esimerkiksi ääneen liittyvät komennot voidaan laittaa Sound-cogiin ja moderointiin liittyvät komennot Moderation-cogiin.

3.3. Toiminnallisuus

Discord-botin toiminnallisuus koostuu erilaisista komennoista, kuuntelijoista, sekä mahdollisista ulkoisista integraatioista, kuten esimerkiksi tietokannoista tai verkkopalveluista. Ominaisuudet, komennot ja ulkoiset integraatiot voivat vaihdella monimutkaisuudessaan suuresti. Discord-botin voi kehittää yksinkertaiseksi viestibotiksi, joka lähettää ennalta kirjoitettuja viestejä käyttäjille tietyin väliajoin, tai botin voi kehittää monitoimiseksi multimediamonoliitiksi, joka toistaa YouTube-videoita, soittaa musiikkia ja pyörittää roolipeliä käyttäjille samaan aikaan. Loppujen lopuksi se, mitä Discord-botti pystyy tekemään, riippuu siitä, mihin tarkoitukseen bottia haluaa käyttää, ja kuinka paljon aikaa sen kehittämiseen haluaa kuluttaa.

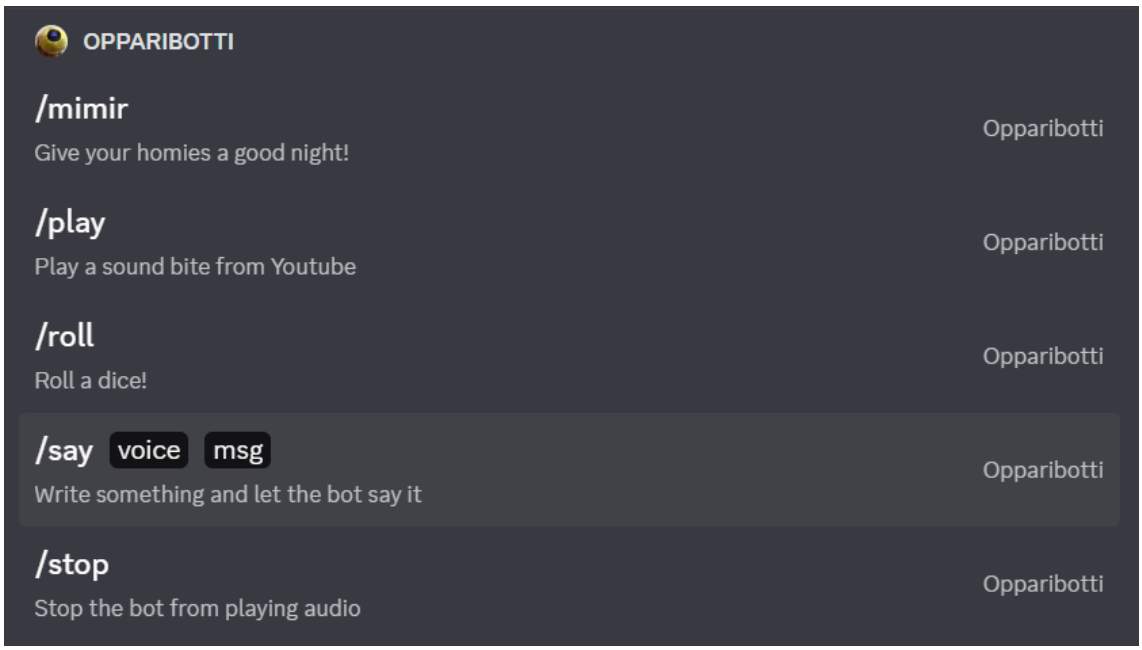
Discord-botin kehittäminen on pitkälti backend-painotteista, sillä kaikki varsinainen logiikka, tietojenkäsittely ja API-kutsut tapahtuvat palvelimen puolella. Botin

käyttäjälle näkyvänä käyttöliittymänä toimii itse Discord-sovellus, joka välittää käyttäjän komennot ja viestit botille tekstipohjaisen käyttöliittymän kautta. Tämä tarkoittaa, että botti pystyy erittäin rajallisesti vaikuttamaan siihen, kuinka esimerkiksi viestit ja muut elementit lopulta Discordissa näyttävät, sillä niiden ulkoasu määräytyy pitkälti Discordin omien rajoituksien mukaan. Toisaalta kehittäjän ei itse tarvitse huolehtia erillisen frontendin rakentamisesta ja ylläpidosta, mutta voi kuitenkin rakentaa varsin kattavan ja responsiivisen ”käyttöliittymän”, hyödyntäen Discordin omia komponentteja, kuten esimerkiksi slash-komentoja.

3.3.1 Komentojen toiminnallisuus

Olemme tutustuneet komentoihin jo kooditasolla, kuinka komentoja on kolmea eri tyyppiä, viesti-, slash- ja hybridikomentoja, sekä miten komennot käytännössä toimivat konepellin alla, mutta komennot ovat tärkeä osa myös käyttökokemusta Discord-sovelluksessa. Käyttäjän näkökulmasta komennot ovat keskeisin tapa olla vuorovaikutuksessa botin kanssa, joten komennoilla on hyvä olla selkeä nimi, sekä kuvaus (kuva 9), jotta käyttäjä ymmärtää kyseisen komennon tarkoituksen mahdollisimman nopeasti ja vaivatta. Toiminnallisuudessa on myös hyvä ottaa huomioon, kuinka komento reagoi käyttäjän antamaan syötteeseen ja mitä ja missä muodossa Discord-botti palauttaa takaisin Discordin käyttöliittymään. Hyvin suunniteltu komento ei ainoastaan tee sitä mitä pyydetään, vaan myös kommunikoi selkeästi ja johdonmukaisesti käyttäjän kanssa.

Viestikomentoja käyttäessä käyttäjä kirjoittaa komennon vapaamuotoisesti tekstikenttään, mutta tämä vaatii käyttäjältä muistamista, sillä Discord ei tarjoa minkäänlaista listaa botin omaavista viestikomennoista, toisin kuin slash-komennot. Viestikomennot eivät käytä hyödyksi Discordin omia natiiveja ominaisuuksia, jolloin niiden käyttö voi tuntua ulkopuoliselta verrattuna muuhun Discordin käyttöön. Vuonna 2021 Discord kuitenkin julkaisi slash-komentotuen Discord-boteille, parantaen käyttäjäkokemusta ja bottien helppokäyttöisyyttä, integroiden botin kanssa vuorovaikuttamisen Discordin omaan käyttöliittymään (Nelly, 2021).



KUVA 9. Lista Opparibotin omaavista slash-komennoista

Slash-komennot integroivat Discord-botin kanssakäymisen Discordin omaan käyttöliittymään ja näin ollen tarjoavat rakenteellisemman ka käyttäjäystävällisemmän vaihtoehdon. Jokainen slash-komento Discord API:n kanssa, jolloin ne näkyvät kuvan 9 tavoin listana komentoja suoraan käyttöliittymässä, kun viestikenttään kirjoittaa ”/”. Tämä mahdollistaa automaattisen täydennyksen valitulle komennolle ja sen omaaville argumenteille, mikä helpottaa käyttäjää oikean komennon löytämisessä ja vähentää virheiden määrää. Käyttäjän ei tarvitse muistaa tarkkoja komentoja ulkoa, vaan voi selata käytettävissä olevia komentoja ja täyttää tarvittavat kentät Discordin tarjoaman ohjauksen avulla.

Slash-komentojen rinnalla hybridikomennot tarjoavat kehittäjille mahdollisuuden säilyttää myös viestikomennot niille käyttäjille, jotka ovat tottuneet perinteiseen käyttöön. Ne hyödyntävät samoja taustalla olevia logiikoita kuin viesti- ja slash-komennot, mutta tarjoavat joustavamman käyttötavan eri käyttäjäryhmille. Hybridikomennot näkyvät samassa listassa missä slash-komennotkin, mutta niitä pystyy myös käyttämään viestikomentojen tavoin, kirjoittamalla komennon vapaamuotoisesti tekstikenttään.

3.3.2 Kuuntelijat

Komentojen lisäksi käyttäjät voivat olla vuorovaikutuksessa Discord-bottien kanssa epäsuorasti niin sanottujen kuuntelijoiden avulla. Kuuntelijat ovat funktioita, jotka reagoivat tiettyihin tapahtumiin, eli eventteihin, joita Discord API lähettää botille näiden tapahtuessa. Näitä tapahtumia voivat olla esimerkiksi viestin lähettäminen, käyttäjän liittyminen palvelimelle tai puhekanavalle tai käyttäjän nimen vaihtuminen. Tapahtumat, jotka Discord API lähettää botille ja johon botti voi reagoida, määritellään koodissa intents-kohdassa ja arkaluontoisemmat tapahtuma-aikomukset tarvitsee manuaalisesti laittaa päälle Discord Developer Portal:ista. Jos Discord-bottiasi käytetään yli 100 eri serverillä, täytyy Discordin verifioida bottisi, jotta voit laittaa arkaluontoisemmat tapahtuma-aikomukset asetuksen päälle (Discord.py, n.d.).

Kuuntelijat toimivat taustalla automaattisesti, tosin kuin komennot, jotka vaativat käyttäjältä jonkinlaisen syötteen. Kuuntelijat mahdollistavat botin aktiivisen reagoinnin Discord-serverin tapahtumiin käyttäjän syötteestä riippumatta. Esimerkiksi jos käyttäjä liittyy puhekanavalle, voi Discord-botti reagoida siihen ja liittyä samalle puhekanavalle, soittaa jonkinlaisen ääniefektin ja lähteä pois puhekanavalta. Tässä tilanteessa Discord-botti kuunteli jonkun tietyn käyttäjän puhekanavatilan muutosta. Tällaisten kuuntelijoiden hyödyntäminen ja käyttö vaatii kuitenkin huolellista suunnittelua, jotta botti ei kuormita turhaan palvelinta tai reagoi vääriin tilanteisiin, joissa tapahtuman konteksti ei ole selkeä. Varsinkin viesteihin reagoiminen voi olla hankalaa. Jos ei selkeästi määrittele mihin kaikkiin viesteihin botti voi reagoida, se voi esimerkiksi reagoida omaan viestiinsä, jonka botti on lähettänyt reaktiona johonkin toiseen viestiin, näin ollen pahimmassa tapauksessa aiheuttaen ikuisen loopin, jossa botti vastaa jatkuvasti omiin viesteihinsä.

3.3.3 Ulkoisien resurssien käyttäminen

Käyttäjillä on mahdollisuus lisätä Discord-bottien lisäksi myös integraatioita omistamilleen servereille. Discord-bottien ja integraatioiden ero piilee datan lähettämisessä ja vastaanottamisessa. Discord-botit ovat suunniteltu suorittamaan tiettyjä toimintoja serverin sisällä. Botit eivät pysty natiivisti yhdistämään muihin sovelluksiin tai palveluihin ja vastaanottamaan tai lähettämään dataa. Integraatiot toisaalta ovat luotu tällaiseen käyttötarkoitukseen ja joillekin sovelluksille Discord

on itse kehittänyt natiiveja integraatiota, kuten Twitchille tai YouTubelle (Tan, 2023).

Vaikka Discord-boteilla ei ole natiivia mahdollisuutta lähettää ja vastaanottaa dataa kolmannen osapuolen sovellusten kanssa, sovellusten integroiminen botin kanssa on kuitenkin mahdollista. Tällaisessa integraatiossa Discord itse ei ole se osapuoli, joka keskustelee sovellusten kanssa, vaan Python ja eri sovelluksille tehdyt ohjelmistokirjastot. Tällaisen integraation toteuttaminen riippuu paljon mitä ohjelmistokirjastoja kyseiselle palvelulle on saatavilla, sekä itse integroitavan palvelun tarjoamista rajapinnoista. Jotkin palvelut eivät tarjoa minkäänlaisia rajapintoja tai säilyttävät niitä maksumuurin takana, jolloin näiden palveluiden integroimisesta botin kanssa tulee olemaan vaikeaa. Joidenkin palveluiden integroiminen, esimerkiksi YouTube-videoiden soittaminen, voi olla hyvinkin hankalaa ilman YouTube-videoiden lataamiselle tarkoitettua ohjelmistokirjastoa. Se ei kuitenkaan ole mahdotonta, mutta ohjelmistokirjaston käyttäminen suoraviivaistaa kehitystä huomattavan paljon (kuva 10).

```python
@bot.hybrid_command(name = "say", description = "Write something and let the bot say it")
async def say(ctx: commands.Context, voice: str, msg: str):
    voice_settings = get_voice_settings(voice)
    await ctx.defer()
    audio = elevenlabs_client.generate(
        text = msg,
        voice = voice,
        model = "eleven_multilingual_v2",
        voice_settings = voice_settings
    )
    save(audio, "output.mp3")
    voice_channel = ctx.author.voice.channel
    voice_client = await voice_channel.connect()
    voice_client.play(FFmpegPCMAudio("output.mp3"))
    await ctx.send(f"{voice}: {msg}")
    await voice_client.disconnect()
    os.remove("output.mp3")
```

KUVA 10. Esimerkki ElevenLabs-ohjelmistokirjastoa hyödyntävästä komennosta.

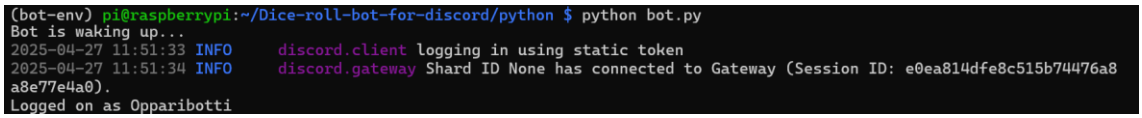
Discord-botti siis toimii välikätenä Discordin ja ulkoisten sovellusten välillä. Näitä ulkoisia resursseja yleensä käytetään komentojen ja kuuntelijoiden kautta, jolloin käyttäjälle ei tarvitse muistaa ja opetella ylimääräisiä systeemejä vain ulkoisten resurssien käyttöä varten. Integraatiota tehdessä on kuitenkin hyvä huomioida,

että jatkuva kommunikointi ulkoisten palveluiden kanssa voi kasvattaa botin resurssitarpeita, kuten muistinkäyttöä ja prosessointitehoa. Komentojen, jotka kommunikoivat ulkoisten palveluiden kanssa, käyttöä on hyvä rajoittaa, jotta Discord-botti ei ylikuormittuisi. Joillakin suurilla palveluilla on myös API-käyttöön liittyviä rajoituksia, kuten kutsumäärien rajoja tai pakollisia autentikointiprosesseja, jotka täytyy ottaa huomioon bottia suunniteltaessa.

4 BOTIN AJAMINEN

4.1. Ongelmat

Kun Discord-botin on suunnitellut järkevästi ja logiikat ovat kirjoitettuna, jää jäljelle ainoastaan botin ajaminen. Bottia on hyvä aina välillä ajaa jo kehitysvaiheessa, jotta yksittäisiä ominaisuuksia ja kokonaisuutta on helpompi testata. Tätä varten kannattaa perustaa testausserveri, jonne botti kutsutaan. Näin botin ominaisuuksia pystyy testaamaan yksittäisessä ympäristössä, eikä komentojen testaaminen häiritse esimerkiksi muita käyttäjiä, jos botti on jo kutsuttu jollekin julkiselle serverille. Botin ajaminen yksinkertaisimmillaan koostuu kahdesta argumentista: ohjelmointikielipohjaisesta komennosta, Pythonin tapauksessa ”python” ja JavaScriptin tapauksessa ”node”, sekä Discord-botin päätiedostosta, jonka komento suorittaa (kuva 11). Tämä komento suorittaa yleensä omalta kotikoneelta botin koodin ja käynnistää yhteyden palvelimiin käyttäen botille määriteltyä tokenia. Oman koneen käyttäminen botin ajamiseen nostaa kuitenkin ongelmia ilmaan, jos botin pitkäaikainen käyttö on mielessä.



KUVA 11. Esimerkki Discord-botin käynnistämisestä

Yksi suurimmista ongelmista oman koneen käytössä botin ajamiseen on jatkuva päällä pysyminen. Oman tietokoneen tarvitsee olla jatkuvasti käynnissä ja yhteydessä internettiin, jotta botti pysyy saatavilla Discord-serverillä. Tämä ei ole käytännöllistä, sillä esimerkiksi tietokoneen sammuttaminen tai internet-yhteyden katkeaminen katkaisee myös botin toiminnan. Riippuen myös botin monimutkaisuudesta, botin ajaminen saattaa viedä paljonkin resursseja, kuten muistia tai prosessointitehoja, sitä ajavalta laitteelta. Jos konetta haluaa käyttää samaan aikaan kuin botti on käynnissä, saattaa muut tietokoneen prosessit hidastua tai toisin päin, botin toiminnallisuus saattaa hidastua, koska tietokone vie liikaa resursseja muihin prosesseihin. Jos botilla on jonkinlaisia kuuntelijoita, jotka vaativat sen, että botti on päällä vuorokauden ympäri, kannattaa harkita botin ajamista jollakin muulla alustalla.

4.2. Botin hostaus

Jotta Discord-botti voisi toimia luotettavasti ja olla jatkuvasti saatavilla, sen ajaminen omalta koneelta ei ole pitkällä aikavälillä järkevin ratkaisu. Tämän takia on suositeltavaa siirtää botin ajaminen omalta tietokoneelta ulkoiselle palvelimelle, joko palveluntarjoajan kautta tai self-hostaamalla. Palvelinta valittaessa kuitenkin pitää ottaa huomioon omat tarpeet, mihin tarkoitukseen on bottia kehittää. Jos tarkoituksena on luoda mahdollisimman monimutkainen Discord-botti, jonka haluaa ottaa käyttöön sadoilla eri servereillä, palvelimelta täytyy löytyä laskentatehoa prosessoimaan satojen eri servereiden viestejä tai tapahtumia, sekä sen tarvitsee olla hyvin skaalautuva. Pieneen harrastelijakäyttöön, omalle tai muutamalle Discord-serverille, riittää usein kevyempi ratkaisu, kuten ilmainen tai halpa bottien hostaamiseen tarkoitettu hostingpalvelu tai self-hostaaminen.

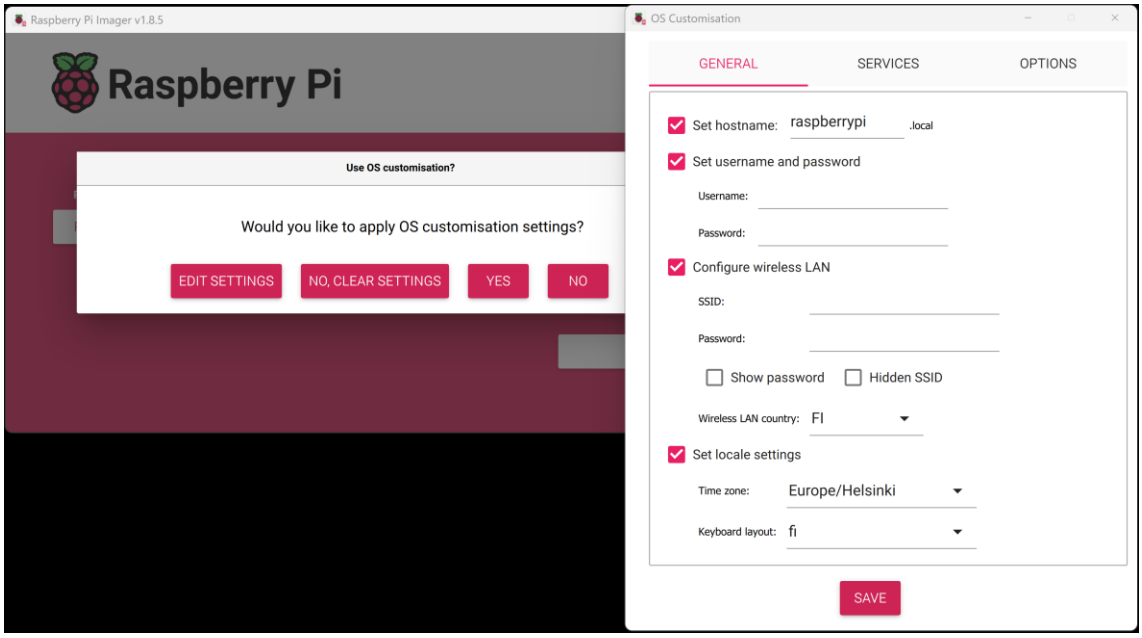
Täysin ilmaisia hostingpalveluita on erittäin harvassa ja mahdollisesti niidenkin ilmaisversiot ovat rajoitettuja jossain määrin. Tämän takia self-hostaaminen on hyvinkin järkevä vaihtoehto. Kynnyksenä self-hostaamisessa kuitenkin tulee vastaan osaaminen ja opetteleminen (Fifthdread, 2025). Tämä voi aluksi lannistaa ja työntää poispäin oman palvelimen pyörittämisestä, mutta self-hostaamisesta saadut hyödyt ovat lyhyen oppimiskäyrän arvoisia. Mielessä kannattaa pitää, että myöskään self-hostaaminen ei ole täysin ilmaista, mutta se on paljon halvempaa pidemmällä aikavälillä. Oman palvelimen pyörittämiseen tarvitaan tietokone, oli se sitten pelaamiseen tarkoitettu tehomylly tai halpa, yksiytiminen Raspberry Pi pientietokone, sekä vakaan sähkö- ja internetyhteyden.

Self-hostaaminen on järkevää myös muissakin tilanteissa kuin Discord-botin ajamisessa. Yksiä suurimpia on hyötyjä on yksityisyys. Oman palvelimen käyttäminen auttaa välttelemään kolmannen osapuolien palveluiden käyttöä, jotka saattavat kerätä ja analysoida dataasi. Data keräämisen ja analysoinnin lisäksi kolmannen osapuolen palveluiden käyttökokemus ja tarjottavat palvelut heikentyvät laadultaan ja nousevat hinnoiltaan ajan myötä. Tätä kutsutaan internetin enshittificaatioksi. Vaikka täysin ei pysty välttymään kaikilta verkkopalveluilta, self-hostaamalla olet kuitenkin itse hallinnoimassa käyttämisiä sovelluksia ja palveluita ja näin ollen olet itse vastuussa niiden laadusta (Fifthdread, 2025).

4.3. Raspberry Pi

Yksinkertaisten palveluiden self-hostaamiseen suosittuja laitteistoja ovat pientietokoneet niistä tarkemmin Raspberry Pi pientietokoneet (Heinzman, 2024). Raspberry Pi -tietokoneet, usein lyhennettynä RasPi tai RPI, ovat yhden piirilevyn tietokoneita, jotka ovat edullisia, pienkokoisia ja energiatehokkaita, mikä tekee niistä loistavan vaihtoehdon Discord-botin ajamiseen. Raspberry Pi tukee natiivisti Python-ohjelmointiympäristöä, jonka takia botti on helppo saada pyörimään Raspberry Pi:lle. Vaikein osuus Raspberry Pi:n alustamisessa Discord-botin ajamiseen tulee olemaan Raspberry Pi:n tarjoaman Raspberry Pi OS käyttöjärjestelmän käyttäminen headless:inä, joka tarkoittaa käyttöjärjestelmän käyttöä komentoriviltä ilman graafista käyttöliittymää. Raspberry Pi OS asentaa itsensä ilman mitään etäyhteysmahdollisuuksia, mutta ne ovat helppo saada päälle ilman näyttöä tai näppäimistöä (Piltch, 2023).

Kaikki alkaa muistikortista ja Raspberry Pi Imager -sovelluksesta. Raspberry Pi Imager -sovellus on kehitetty Linux-pohjaisten käyttöjärjestelmien lataamiseen muistikortille tai muulle asennuslevylle, jota Raspberry Pi loppujen lopuksi käyttää käyttöjärjestelmänään. Omistetun Raspberry Pi -laitteen, käyttöjärjestelmän ja asennuslevyn valinnan jälkeen on tärkeää muistaa asettaa käyttäjänimi ja salasana, WiFi-asetukset, sekä SSH-asetukset (kuva 12). Nämä ovat erittäin tärkeitä asettaa tässä kohtaa asennusprosessia, sillä ne takaavat sen, että Raspberry Pi pystyy yhdistämään internettiin ja että Raspberry Pi:hin pystyy yhdistämään toisesta tietokoneesta SSH-protokollan avulla. Jos joitain näistä asetuksista ei muista asettaa tai jokin näistä asetuksista on väärin, pahimmassa tapauksessa Raspberry Pi:hin ei saa mitenkään yhteyttä, jolloin tarvitsee aloittaa koko asennusprosessi uudestaan. Jos on taas kokemusta Linux-pohjaisten käyttöjärjestelmien asentamisesta, on vielä WiFi-asetukset ovat vielä mahdollista asettaa, ennen kuin laittaa asennuslevyn Raspberry Pi:hin kiinni (Piltch, 2023).



KUVA 12. Raspberry Pi Imager -sovelluksen tärkeät asetukset

4.4. Raspberry Pi:n käyttö etäyhteydellä

Raspberry Pi:n käyttö etäyhteydellä on olennainen osa sen hyödyntämistä self-hostaamisessa, erityisesti silloin, kun laitetta käytetään ilman näyttöä, näppäimistöä tai hiirtä. Yleisin tapa muodostaa etäyhteys Raspberry Pi:hin on SSH-protokollan käyttö, joka mahdollistaa turvallisen pääsyn laitteen komentoriville. Tämä vaatii joko kokemusta tai opettelua tietoverkkojen toiminnasta, sillä SSH-protokollalla yhdistämisessä tarvitsee tietää Raspberry Pi:n käyttäjänimi, salasana sekä IP-osoite. Laitteen IP-osoitteen pystyy löytämään monella eri tapaa, mutta helpoiten sen löytää oman reitittimen hallintapaneelista, josta näkyy kaikkien laitteiden IP-osoitteet, jotka ovat yhdistettynä reitittimeen. Tämä tietenkin vaatii, että asennusvaiheessa käyttöjärjestelmälle annettiin oikeat WiFi-pääsytiedot.

Raspberry Pi:hin muodostetaan yhteys toiselta tietokoneelta käyttämällä SSH-asiakasohjelmaa, kuten PuTTY:ä tai Windowsin sisäänrakennettua SSH-komentoa. Tähän tarvitaan käyttäjänimi ja salasana, jotka asetettiin käyttöjärjestelmän asennuksen yhteydessä, sekä Raspberry Pi:n IP-osoitteen. Tämän jälkeen käyttäjän eteen aukeaa Raspberry Pi:n komentorivi, jolta pystyy täysin hallitsemaan koko tietokonetta. Tämä sisältää esimerkiksi ohjelmistojen asentamisen, tiedostojen hallinnan ja palveluiden käynnistämisen. Tässä vaiheessa on hyvä päivittää

koko käyttöjärjestelmä, jotta käytössä olisi viimeisimmät versiot käyttöjärjestelmän ohjelmista ja paketeista. Koska Discord-botin pyörittäminen ei periaatteessa tarvitse muuta kuin Pythonin, on käyttäjällä mahdollisuus muokata Raspberry Pi:n käyttöjärjestelmää asentamalla haluamiaan paketteja ja ohjelmia.

Etäyhteyden muodostamisen jälkeen Discord-botin lähdekoodi pitää siirtää Raspberry Pi:lle, jotta sitä voisi ajaa. Tähän löytyy monia eri tapoja. Lähdekoodin voi siirtää vanhanaikaisella tavalla, jossa tiedostot siirretään ensiksi muistikortille tai -tikulle, jonka jälkeen ne siirretään tikulta Raspberry Pi:lle. Tämä kuitenkin luo ongelmia lähdekoodin pitämisessä ajan tasalla, erityisesti silloin, kun bottia halutaan kehittää aktiivisesti tai päivittää usein. Varsinkin jos mielessä on Discord-botin jatkokehitys, kannattaa käyttää versionhallintajärjestelmiä, kuten Git:iä. Tämä mahdollistaa vaivattoman botin päivittämisen, jossa muutokset bottiin kirjoitetaan omalla koneella, ladataan projektin Git-repositorioon ja lopuksi ladataan Raspberry Pi:lle repositoriosta käyttäen komentoriviä.

4.5. Pythonin ajaminen Raspberry Pi:llä

Discord-botin ajaminen omalta palvelimelta, joka pyörii Raspberry Pi -pientietokoneella, käy samalla lailla, kun botin ajaminen omalla koneella: kutsumalla python-komentoa. Raspberry Pi:n käyttöjärjestelmään on valmiiksi asennettu Python ohjelmointiympäristö, mutta kannattaa ottaa huomioon Python-versio, jolla Discord-botti on kirjoitettu. Raspberry Pi:lle asennettuna oleva Python voi mahdollisesti olla versioltaan vanhempi, joten on tärkeä varmistaa, että käytössä oleva Python ohjelmistoympäristö on versioltaan uudempi tai vähintään sama version, jonka botti vaatii.

Oikean Python-version lisäksi botti tarvitsee toimiakseen samat ohjelmistokirjastot, mitä kehittämisessä on käytetty. Python-projekteissa käytettyjen ulkoisten kirjastojen määrä riippuu paljon projektista, mutta hyvin monimutkaisissa projekteissa niitä voi olla useita kymmeniä. Joskus voi olla hyvinkin epäselvää, mitä kirjastoja projektissa on käytetty, ja näiden kirjastojen asentaminen yksi kerrallaan komentoriviltä on erittäin aikaa vievää ja tyhmää. Siksi pip-paketinhallintajärjestelmä tarjoaa mahdollisuuden tallentaa kaikki projektissa käytetyt kirjastot ja niiden versiot tekstiksi ”requirements.txt”-nimiseen tiedostoon (kuva 13). Tämä

tiedosto toimii eräänlaisena ohjeistuksena, jonka avulla projektissa tarvittavat kirjastot voidaan asentaa yhdellä komennolla. Käyttämällä asennuskomentoa (kuva 13) pip-paketinhallintajärjestelmä asentaa kaikki projektissa käytetyt kirjastot täsmälleen samoilla versioilla, joita käytettiin kehitysvaiheessa.

```
pip freeze > requirements.txt # luo listan projektissa käytetyistä ohjelmistokirjastoista
pip install -r requirements.txt # asentaa projektissa käytetyt ohjelmistokirjastot
```

KUVA 13. Komennot kirjastojen listaamiseen ja asentamiseen

Kun ohjelmistokirjastot on saatu asennettua, Discord-botti on valmis käyttöön. On kuitenkin tärkeää varmistaa, että botti pysyy käynnissä myös silloin, kun SSH-yhteys katkeaa tai Raspberry Pi käynnistyy uudelleen. Tämä voidaan saavuttaa käyttämällä prosessinhallintatyökaluja, kuten tmux tai systemd. Nämä prosessinhallintatyökalut mahdollistavat käyttäjän luomaan ”pseudo-komentorivejä” yhdeltä pääkomentoriviltä, jolloin käyttäjä voi ajaa Discord-bottia yhdeltä komentoriviltä ja yhdistää SSH-protokollan avulla toiseen komentoriviin, keskeyttämättä Discord-botin prosesseja (Gerardi, 2022).

4.6. Jatkokehitys

Discord-botin pyöriessä Raspberry Pi:llä, on mahdollisuus jättää kehittäminen siihen. Discord-botti on päällä niin kauan kuin Raspberry Pi pysyy käynnissä ja sen ohjelmisto toimii moitteettomasti. Harrastelijakäyttöön yhdelle tai muutamalle serverille tarkoitetulle botille alkuperäiset ominaisuudet ovat usein riittäviä ja jatkokehitys riippuu paljon kehittäjän omasta halusta kehittää uusia ominaisuuksia tai parantaa nykyisiä.

On kuitenkin mahdollista, että kehittäjä haluaa laajentaa kehittämänsä botin käyttäjäkuntaa ja tarjota bottinsa ominaisuuksia suuremmalle määrälle Discord-servereistä. Tällaisen tilanteen edessä tarvitsee suunnitella ja arvioida jatkokehityksen vaikutuksia botin kykyyn vastaanottaa viestejä ja kuunnella tapahtumia, sekä palvelimen suorituskykyyn palvella suurempaa käyttäjäkuntaa. Suurempi käyttäjämäärä tulee varmasti vaikuttamaan omiin self-hostaus-ratkaisuihin ja uusien ominaisuuksien ja kuuntelijoiden kehittäminen tulee vaikuttamaan Discord-botin responsiivisuuteen.

Jatkokehitykseen lähtiessä tarvitsee huomioida pari asiaa: botin skaalautuvuus ja tietoturva. Tähänastisen Discord-botin kehittämisessä ei ole otettu huomioon skaalautuvuutta, vaan keskitytty juurikin kavereiden kesken käytettävän botin kehittämiseen. Suurempi käyttäjämäärä rasittaa enemmän palvelinta, jolla Discord-bottia ajetaan, jolloin Raspberry Pi:n suorituskyky ja resurssit alkavat tulla vastaan. Tällöin on tarpeen siirtää botin hostaaminen tehokkaammalle alustalle, kuten pilvipalveluihin tai virtuaalipalvelimille. Tällaiset palvelut tarjoavat skaalautuvia ratkaisuja, jotka mahdollistavat resurssien lisäämisen käyttäjämäärän kasvaessa, mutta botin siirtäminen vaatii koodin mukauttamista palvelun mukaiseksi, sekä monet pilvipalvelut eivät ole ilmaisia.

Tietoturvaan ei myöskään harrastelijaprojektissa tarvitse kovinkaan paljoa keskittyä. Botin kanssa keskustelee ainoastaan kehittäjä ja hänen valitsemansa ihmiset. Tähänastisessa Discord-botissa on kuitenkin käytetty joitakin yleisiä tietoturvaan liittyviä menetelmiä, kuten kaikenlaisten tokenien piilottaminen ympäristömuuttujiin, jotta ne eivät leviäisi, kun koodi esimerkiksi ladataan julkiseen Git-repositorioon. Käyttäjämäärän kasvaessa myös mahdollisten palvelunestohyökkäysten ja luvattomien pääsy-yritysten määrä kasvaa. Tämän takia Discord-botin skaalaamisessa, sekä uusien ominaisuuksien kirjoittamisessa, tulee ottaa huomioon, kuinka Discord-botti esimerkiksi käsittelee käyttäjien syötteitä ja suojaa arkaluonteista dataa.

5 POHDINTA

Discord-botit ovat tärkeä osa Discordin ekosysteemiä ja käyttäjäkokemusta. Ne rikastuttavat Discord-servereiden käyttöä laajoilla toiminnallisuuksillaan ja tukevat yhteisöjen arkea automatisoimalla serverin hallintaa, hoitamalla käyttäjien moderointia, sekä tarjoamalla viihdettä käyttäen ulkoisia palveluita. Discordin API ja sen tarjoamat työkalut mahdollistavat laajan skaalan erilaisia käyttötarkoituksia, mikä tekee oman Discord-botin kehittämisestä houkuttelevan sekä harrastajille, että kokeneemmille ohjelmistokehittäjille.

Tämä opinnäytetyö lähti liikkeelle tarpeesta rakentaa oma Discord-botti omalle kavereiden keskeiselle Discord-serverille, ja näin ollen tutustua paremmin Discord-botin rakenteeseen, toiminnallisuuteen ja kehitysprosessiin erityisesti Python-ohjelmointikielellä. Työssä käsiteltiin myös ohjelmistokirjastojen ja ohjelmointikielien merkitystä bottikehityksessä. Toteutetun projektin perusteella Python ja Pythonin ohjelmistokirjastot osoittautuivat erittäin tehokkaiksi ja joustaviksi työkaluiksi Discord-botin kehittämisessä. Esimerkiksi discord.py-kirjasto, joka toimi projektin pääohjelmistokirjastona, tarjosi selkeän ja hyvin dokumentoidun tavan kommunikoida Discordin API:n kanssa, mikä mahdollisti laajan joukon toiminnallisuuksia, kuten komentoihin reagoiminen, tapahtumien käsittely ja ulkoisten API-yhteyksien hyödyntäminen.

Työn tarkoituksena oli luoda toimiva Discord-botti ja saada se pyörimään itsenäisesti Linux-pohjaiselle pientietokoneelle. Oman Discord-botin kehittäminen tarjoaa monia konkreettisia hyötyjä, jotka ulottuvat teknisestä osaamisen kartuttamisesta aina käyttäjäkokemuksen parantamiseen asti. Oman botin rakentaminen mahdollistaa juuri omien toiveiden mukaisen ratkaisun luomisen ja antaa mahdollisuuden muokata omaa ja muiden käyttökokemusta Discord-palvelussa käyttäjien tarpeisiin sopivaksi ilman kompromisseja tai turhia toimintoja.

Työn onnistumista voidaan pitää erittäin hyvänä, sillä emme luoneet ainoastaan teknistä ratkaisua, vaan käyttökelpoisen ja itse ylläpidettävän palvelun. Työ palvelee erityisesti harrastajia tai pienkehittäjiä, joilla on kiinnostusta tai tarvetta kehittää oma Discord-botti esimerkiksi ystäväporukan serverille tai oppia Pythonin käytännön soveltamista. Projektin aikana saavutettu lopputulos osoittaa, kuinka

suhteellisen pienellä vaivalla on mahdollista rakentaa täysin toimiva, omaan käyttötarkoitukseen suunniteltu järjestelmä, joka tuo lisäarvoa käyttäjilleen päivittäisessä käytössä.

Jatkokehityksenä olisi kiinnostaa tutkia Discord-botin skaalaamista suuremmalle käyttäjämäärälle, kuin yhdelle serverille, jossa on muutama käyttäjä. Tämä vaatisi lähemmin tutkimista siitä, mitä suorituskyky- ja turvallisuusratkaisuja se edellyttäisi, sekä miten se vaikuttaisi nykyiseen self-hostaus-ratkaisuun. Kiinnostavaa olisi myös selvittää, kuinka nimenomaan Pythonia voisi paremmin hyödyntää botin toiminnassa ja esimerkiksi olisiko mahdollista hyödyntää koneoppimisen menetelmiä botin päätöksenteossa.

LÄHTEET

Chng, Roy. 6.6.2023. Python Decorators Explained For Beginners. freeCodeCamp. Verkkosivu. Viitattu 22.7.2024. https://www.freecodecamp.org/news/python-decorators-explained/

Diazbeltran, Jaime, 18.4.2022. Splitting Code Into Multiple Files. Medium. Verkkosivu. Viitattu 23.4.2025. https://medium.com/@jaime.diazbeltran/splitting-code-into-multiple-files-1b4d70001a18

Discord.py. n.d. A Primer to Gateway Intents. Verkkosivu. Viitattu 24.4.2025. https://discordpy.readthedocs.io/en/stable/intents.html

Fifthdread. 13.2.2025. Why Self-Hosting Is Important. Fifthdread. Verkkosivu. Viitattu 29.4.2025. https://fifthdread.com/why-self-hosting-is-important/

Ganesh, Abhijith. 3.9.2021. End of Discord.Py. DEV. Verkkosivu. Viitattu 10.7.2024. https://dev.to/abhijithganesh/end-of-discord-py-58pc

Gerardi, Ricardo. 13.9.2022. A beginner's guide to tmux. Red Hat Blog. Verkkosivu. Viitattu 30.4.2025. https://www.redhat.com/en/blog/introduction-tmux-linux

Gupta, Aryan. 10.7.2024. 20 Most Popular Python IDEs in 2024: Code Like a Pro. Simplilearn. Verkkosivu. Viitattu 17.7.2024. https://www.simplilearn.com/tutorials/python-tutorial/python-ide

Guru Insights. 28.9.2022. How does a Discord bot work?. Verkkosivu. Viitattu 9.7.2024. https://www.guru.com/blog/how-does-a-discord-bot-work/

Heinzman, Andrew. 21.12.2024. 2024 Was Raspberry Pi's Biggest Year to Date. How-To Geek. Verkkosivu. Viitattu 29.4.2025. https://www.howtogeek.com/2024-was-raspberry-pis-biggest-year-to-date/

Marks, Tom. 13.5.2016. One year after its launch, Discord is the best VoIP service available. PC Gamer. Verkkosivu. Viitattu 6.7.2024. https://www.pcgamer.com/one-year-after-its-launch-discord-is-the-best-voip-service-available/

Nelly, 7.4.2020. The future of bots on Discord. Discord Blog. Verkkosivu. Viitattu 8.7.2024. https://discord.com/blog/the-future-of-bots-on-discord

Nelly, 24.3.2021. Slash Commands are Here. Discord Blog. Verkkosivu. Viitattu 24.4.2025. https://discord.com/blog/slash-commands-are-here

Piltch, Avram. 23.10.2023. How to Set Up a Headless Raspberry Pi, Without Ever Attaching a Monitor. Tom's Hardware. Verkkosivu. Viitattu 29.4.2025. https://www.tomshardware.com/reviews/raspberry-pi-headless-setup-how-to,6028.html

Rothwell, Lauren. 13.7.2023. Must know: Top 5 Discord libraries every developer should know. Blaze. Verkkosivu. Viitattu 10.7.2024. https://www.withblaze.app/blog/must-know-for-discord-top-5-discord-libraries

Ronquillo, Alex. n.d. "How to Make a Discord Bot in Python". Real Python. Verkkosivu. Viitattu 16.7.2024. https://realpython.com/how-to-make-a-discord-bot-python

Sidak, Kyryl, 1.6.2024. Python Libraries Written in C++. Codefinity. Verkkosivu. Viitattu 24.4.2025. https://codefinity.com/blog/Python-Libraries-Written-in-C-plus-plus

Tan, Siew Ann. 6.10.2023. Discord Integrations: A Guide for Beginners. Make. Verkkosivu. Viitattu 26.4.2025. https://www.make.com/en/blog/discord-integrations-guide

Ullah, Rakib. 1.4.2023. Unleashing the Power of Discord: The Role of BOTS in Boosting Your Business. Syscomatic. Verkkosivu. Viitattu 9.7.2024. https://medium.com/@syscomatic/unleashing-the-power-of-discord-the-role-of-bots-in-boosting-your-business-c5546019d340