Joel Mitrunen

Monikäyttöisen itsenäisen Discord-ohjelman suunnittelu ja toteutus

Insinööri (AMK)

Tietotekniikka

Kevät 2018

KAJAANIN
AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tiivistelmä

Tekijä(t): Mitrunen Joel

Työn nimi: Monikäyttöisen itsenäisen Discord-ohjelman suunnittelu ja toteutus

Tutkintonimike: Insinööri (AMK), tietotekniikka

Asiasanat: Discord, Itsenäinen ohjelma, Web-kaavinta, .NET, C#

Työn tavoitteena luoda itsenäinen ohjelma Discord sovellukselle käyttäen C#-ohjelmointikieltä. Itsenäisen ohjelman tulisi myös käyttää hyödyksi web-kaavintaa internetsivuilta tietojen saamiseksi. Ohjelman tulee myös toimia itsenäisesti virtuaalisessa ympäristössä kolmannella osapuolella.

Ohjelma tulee hyväksikäyttämään Microsoftin .NET kehittäjä-alustaa Visual Studiolle, joka takaa alustariippumattomuuden sekä useita kolmannen osapuolen paketteja eri osa-alueitten toteuttamiseksi.

Lopputuloksena saatiin valmiiksi itsenäinen C# ohjelma, jossa on helposti muokattavissa oleva runko ja komennot, joka käyttää hyväksi internetsivujen kaavintaa ja ääniominaisuuksia. Ohjelma toimii itsenäisesti virtuaalikoneella, ja sitä ei tarvitse valvoa.

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Abstract

Author(s): Mitrunen Joel

Title of the Publication: Design and Implementation of Multifunctional Independent Discord Program

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: Discord, Independent program, Web Scraping, .NET, C#

The aim of this thesis was to create an independent program for the Discord application using the C# programming language. The standalone program should also make use of web scraping from websites to get information. The program should also work autonomously in a third party virtual environment.

The program will be using Microsoft's .NET developer platform in Visual Studio, which ensures the implementation of platform independence, as well as a number of third-party packages for completing different segments of the program.

As the result, an independent C# program was created, that has easy to modify body, and commands that utilize web scraping and sound features. The program works independently on a virtual machine and does not need to be monitored.

Sisällys

Symboliluettelo

- EC2 — Amazonin pilvi-tietojenkäsittelyalusta.
- NuGet — Pakettienhallintaohjelma.
- RID — Alustan tunnistamisnumero.
- VoIP — Teknologiaryhmä puheviestin toimittamiseksi.
- Wrapper — Aliohjelma, joka kutsuu toista aliohjelmaa.

1 JOHDANTO

Työn tarkoituksena on tutkia ja käydä läpi vaatimuksia itsenäisen ohjelman luontiin Discord VoIP-sovellukselle käyttämällä C#-ohjelmointikieltä.

Tavoitteena on luoda itsenäinen ohjelma, jonka voi kutsua mihin tahansa kanavalle ja sen käyttäjät voivat kutsua ja hyödyntää erilaisia komentoja. Komentoihin on luotu yksi esimerkki tiedoston lukemisen ja kirjoittamisen avulla tehdyistä muokattavista komennoista, web-kaavinnan avulla luotu uutisten hakeminen sekä äänikirjastojen avulla luotu musiikin toisto Youtube-sivuston kautta.

Tämä opinnäytetyö on tarkoitettu avuksi ohjelmoinnista kiinnostuneille henkilöille. Tästä johtuen eri osa-alueiden tekemisistä on nähtävänä useita esimerkkejä.

2 TARVITTAVAT TYÖKALUT

2.1 Discord

Discord on kehitysstudion ”Hammer&Chisel” kehittämä sosiaalista mediaa edustava ilmainen VoIP-sovellus, joka on suunniteltu peliyhteisöille. Discord toimii Windows-, macOS-, Android- ja Linux-käyttöjärjestelmillä sekä näiden lisäksi myös nettiselaimella.

Ensimmäisen ohjelman julkaisu tapahtui 5.3.2015, ja vakaa, varsinainen versio julkaistiin 5.10.2017. Discordin toimitusjohtaja sai idean ohjelmalle, kun hän huomasi kommunikoinnin vaikeuksia tietyissä edustavissa peleissä kuten ”League of Legends” ja ”Final Fantasy XIV” ja näiden pelien VoIP mahdollisuudet, jotka tarvitsivat erilaisia IP-osoitteita vain kanavan liittymiseen. Muut ohjelmistot kuten Skype ja TeamSpeak olivat resursseiltaan työläitä ajaa ja niissä oli tunnettuja turvallisuusongelmia. Vuonna 2017 toukokuusta lähtien Discordilla on yli 45 miljoonaa käyttäjää ympäri maailman. Discord tarjoaa käyttäjille VoIP-kommunikaatio-, pikaviestintä- sekä videokeskustelumahdollisuudet. [1]

Discordin ohjelmointirajapinta on tehty valtuuttamaan ja helpottamaan discord-komentojen käyttöä ohjelmien koodeissa.

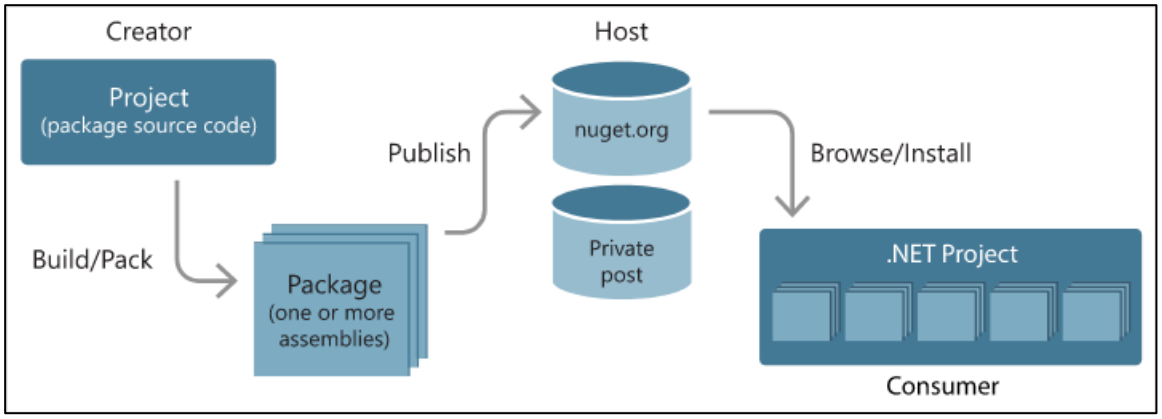
2.1 NuGet-pakettimanageri

Tämän ohjelman toteutuksessa tullaan käyttämään apuna ohjelmointipaketteja. Ohjelmointipaketit ovat kuin kirjastot. Niitä käytetään koodaamisen avustamisena ja nopeuttamisena, koska ne yksinkertaistavat tai ajavat osan koodeistaan sisällänsä.

NuGet on keskeinen työkalu mille tahansa modernille kehitystoiminnalle. Se on mekanismi, jonka avulla kehittäjät voivat luoda, jakaa ja käyttää hyödyllisiä kirjastoja ja paketteja NET Frameworkissa. NuGet määrittelee, miten paketit ovat NET:ille luotu, ylläpidetty ja käytetty, ja se tarjoaa myös työkalut kaikkiin näihin rooleihin. [2]

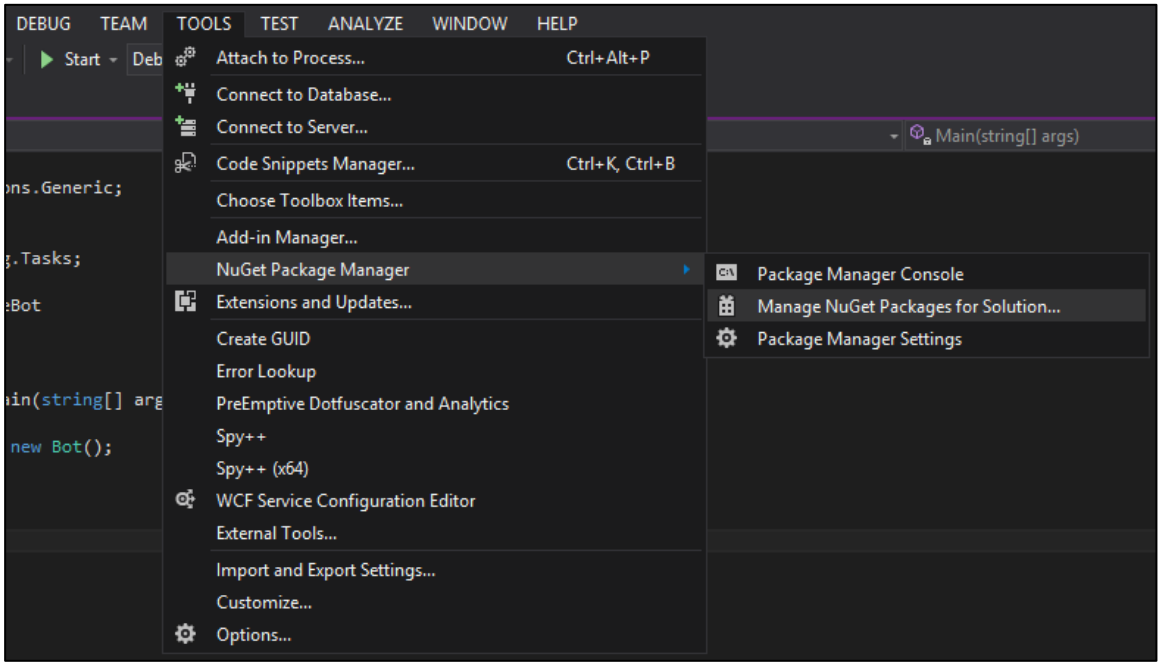
NuGet-isäntänä toimii säilytyspaikkana yli 60.000 ainutlaatuisille, avoinna oleville paketeille. Näitä paketteja käyttävät miljoonat NET-kehittäjät joka päivä. Näitä paketteja voi myös jakaa ja käyttää yksityisesti pilvessä, yksityisessä verkossa. Oli käyttötarkoitus mikä tahansa, isännän tarkoitus on jakaa tekijöiden paketit niiden käyttäjille. Pakettien

luojat jakavat ja julkaisevat pakettinsa eri isännille. Sieltä käyttäjät voivat etsiä ja ladata omiin projekteihinsa hyödyllisiä paketteja. Paketin asentamisen jälkeen projektiin paketin rajapinta on avoinna koko projektin koodille kuvan 1 mukaisesti.



Kuva 1. NuGet-roolit ja niiden toiminta. [2]

Visual Studio 2012 ja sitä myöhemmät versiot sisältävät NuGet pakettimanageri UI:n ja pakettimanageri-konsolin. Tässä tapauksessa Visual Studio 2017, päästään pakettien manageroimisikkunaan työkalupalkista kuvan 2 mukaisesti: Tools > NuGet Package Manager > Manage NuGet Packages For Solution.



Kuva 2. NuGet-pakettien manageroimisikkunan sijainti.

2.2 DSharpPlus

DSharpPlus on epävirallinen C# wrapper, joka on tehty Discord- sovellusta varten. Tämä tarjoaa käyttäjille menetelmiä tehdä automaattisia operaatioita ja itsenäisiä ohjelmia. [3]. Itsenäisen ohjelman tekeminen ei ole monimutkaista, sillä kaikki komennot on dokumentoitu Discordin sivuilla. Tämä on joka tapauksessa hyvä apu vaikeampia komentoja kuten esimerkiksi äänentoistoa varten.

Tähän projektiin tullaan käyttämään DsharpPlus-versiota 3.2.3, ja saadaan nämä paketit projektiin avaamalla NuGet-pakettimanagerikonsolin työkalupalkista Package Manager Console. Konsoliin kirjoitamme seuraavat komennot: *Install-Package DsharpPlus –Version 3.2.3* ja *Install-Package DsharpPlus.CommandsNext –Version 3.2.3*

2.3 Html Agility Pack

Html Agility Pack on C#-projekteihin tarkoitettu paketti, jonka käyttötarkoituksena on auttaa käyttäjiä hakemaan tietoja internetsivuilta koodiin. Yksinkertaisuudessaan se on NET-koodikirjasto, jonka avulla käyttäjät voivat jäsentää ”verkon ulkopuolella” olevia tiedostoja (olkoon se HTML, PHP tai aspx). Kyseinen projekti tulee käyttämään versiota 1.7.1. [4]

2.4 .NET Core

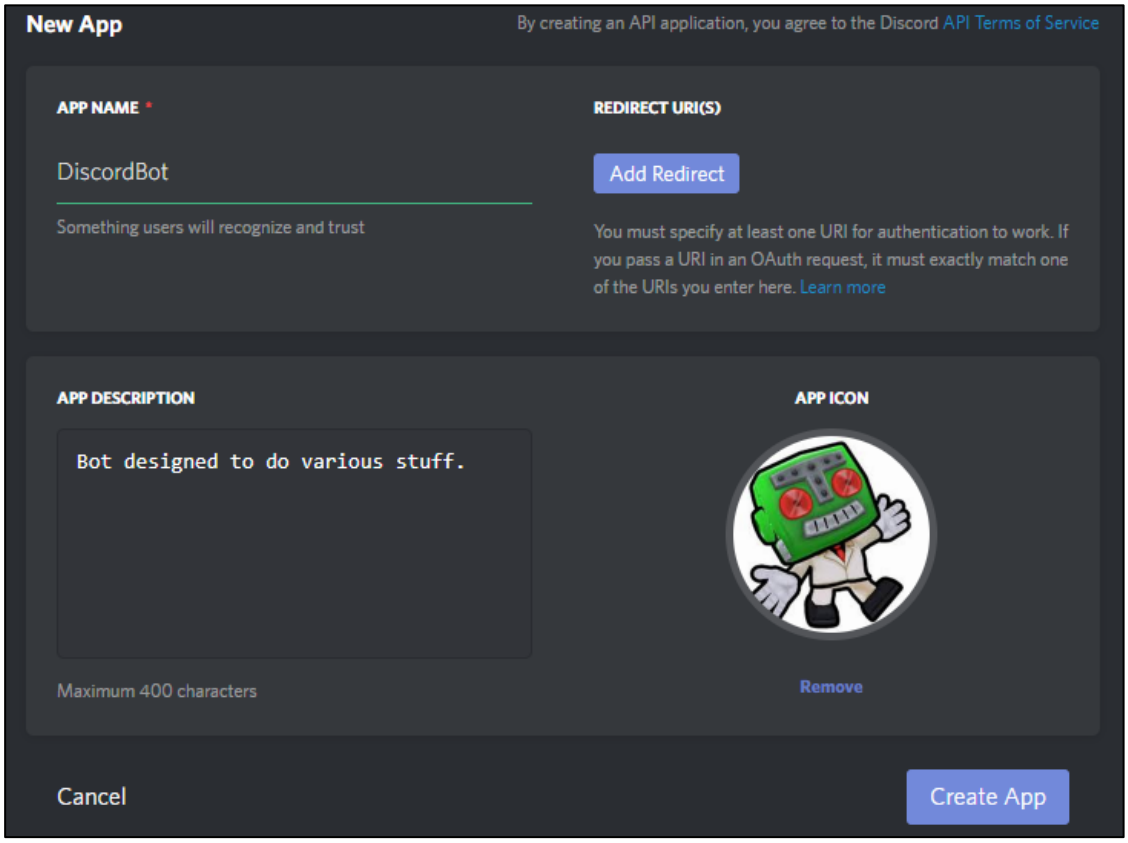
.NET on ilmainen, alustariippumaton, avoimen lähdekoodin kehittäjä-alusta monien erilaisten ohjelmien ja projektien rakentamiseen. Tämän lisäksi se tukee monia kieliä, editoreja ja kirjastoja, joilla voi rakentaa webbiin, mobiilille, tietokoneisiin ja pelaamiseen. NET tukee C#, F# ja Visual Basic- koodikieliä. Julkaisijana toimii Microsoft. [5]

3 ITSENÄISEN OHJELMAN TOTEUTUS

3.1 Itsenäisen ohjelman luonti

Aloitetaan luomalla tyhjän itsenäisen ohjelman käyttäjä. Kutsutaan se kanaville, jonka jälkeen rakennetaan ohjelmalle erilaisia kutsuja.

Itsenäisen ohjelman luonti tapahtuu käymällä Discordin kehittäjäsivustolla. Kirjautumisen jälkeen ohjelman voi luoda painamalla ”New App”-nappia, jonka jälkeen käyttäjä valitsee ohjelmalle nimen, kuvauksen ja kuvan. Kuvassa 3 on ikkuna mistä edellä mainitut asiat löytyvät.

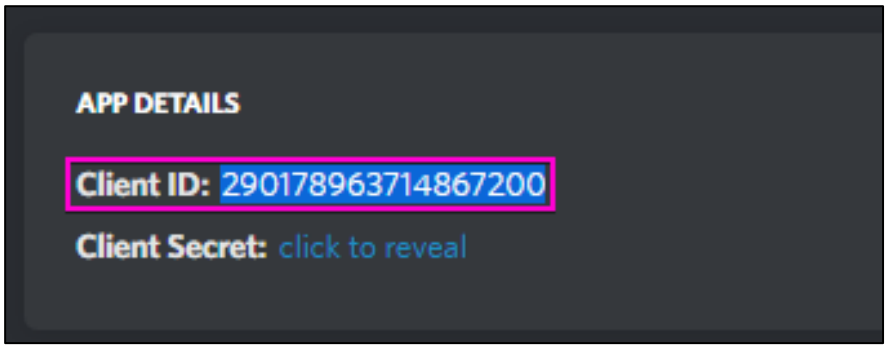


Kuva 3. Itsenäisen ohjelman luonti -ikkuna.

Tämän jälkeen ohjelman esikatselu-ikkunassa painetaan ”Create a Bot User”- nappia ja asetamme ohjelman julkiseksi ruksittamalla ”Public Bot” -valintaruudun. Tämän jälkeen on mahdollisuus luoda ohjelmasta käyttäjä, jonka voi kutsua kanaville.

3.2 Ohjelman lisääminen kanavalle

Kun itsenäinen ohjelma on luotu, voidaan se kutsua kanavalle hakemalla ”Client ID”-sarjanumeron ohjelman esikatselu-ikkunasta kuvan 4 mukaisesti.



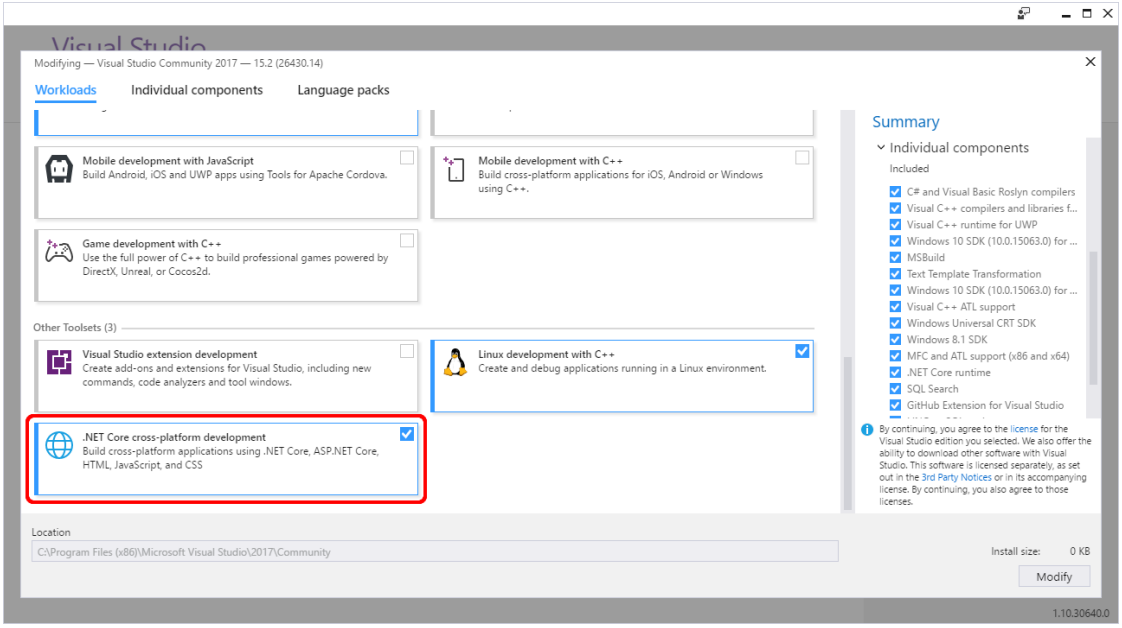
Kuva 4. Client ID- sarjanumeron paikantaminen.

Kyseisen sarjanumeron saatua mennään nettiselaimella osoitteeseen: https://discordapp.com/oauth2/authorize?client_id=<CLIENT ID>&scope=bot, mutta muuttamalla <CLIENT ID> -kohdan saadulla sarjanumerolla. Kyseinen linkki vie ikkunaan, josta voidaan kutsua ohjelma kanaville, joihin käyttäjillä on oikeudet.

Kuva 5. Ohjelman yhdistäminen Discord-kanavalle.

3.3 Visual Studio 2017 ja .NET Core

On tärkeää, että tämän ohjelman luontiin käytetään Visual Studio 2017, koska apuna tullaan käyttämään Microsoftin .NET Corea. .NET Core on ilmainen ja avoimen lähdekoodin web-rakenne, joka tarjoaa monia etuja projektiin, kuten ohjelman järjestelmäriippumattomuuden. .NET Core on avoinna Visual Studio Code- sekä Visual Studio 2017-kehitysympäristöille. Kun Visual Studio 2017 asennetaan, valitaan ”.NET Core cross-platform development” kuvan 6 mukaisesti.



Kuva 6. .NET Coren asennus.

3.4 Ohjelman päärakenteen toteutus

Ohjelman toimimiseen tehdään C#-sovellus. Aloitetaan tekemällä tyhjä ”Console App (.NET Core)”-projekti. Tämä löytyy C#-valintojen alta, jos .NET Core on asennettu Visual Studioon.

Aloitetaan lisäämällä DSharpPlus-paketit projektiin komennoilla. Pakettien lisäysten jälkeen lisätään projektin alkuun tarvittavat kirjaston komponentit käyttämällä avainsanaa ”using”.

Seuraavaksi lisätään Main-luokan yläpuolelle staattisen muuttujan *static DiscordClient discord;*. Tämä muuttuja tulee pitämään DiscordClient-instanssin, jota tarvitaan, kun

halutaan käyttää Discordin ohjelmointirajapintaa projektissa. Token muuttujalle on annettava ohjelman henkilökohtainen merkkisarja, joka löytyy kyseisen ohjelman sivulta. Kuvassa 7 on esimerkki toteutus ohjelman rungosta.

```
using DSharpPlus;

namespace CSharpDiscordBot
{
    2 references
    class Program
    {
        public static DiscordClient discord;

        0 references
        static void Main(string[] args)
        {
            MainAsync(args).ConfigureAwait(false).GetAwaiter().GetResult();
        }

        1 reference
        static async Task MainAsync(string[] args)
        {
            // Discord configuration.
            discord = new DiscordClient(new DiscordConfiguration
            {
                Token = "",
                TokenType = TokenType.Bot,
                UseInternalLogHandler = true,
                LogLevel = LogLevel.Debug

            });


            await discord.ConnectAsync();
            await Task.Delay(-1);
```

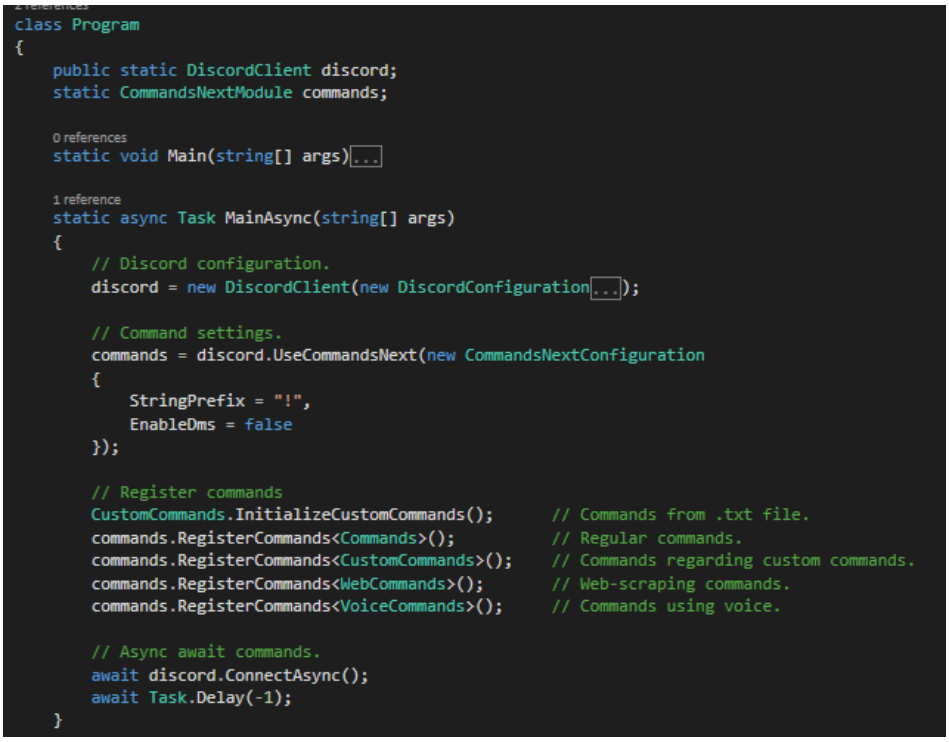
Kuva 7. Itsenäisen ohjelman yhdistämiseen tarvittava runko.

3.5 Komentojen kutsuminen

Komennot ovat käyttäjiä ajatellen itsenäisen ohjelman tärkein asia, joten on tärkeä tietää mihin niiden kutsuminen perustuu.

Komentojen toteutukseen tarvitaan NuGet-paketti nimeltä ”DsharpPlus.CommadsNext”, jonka jälkeen tämän kirjaston komponentit lisätään projektin alkuun käyttämällä ”using”-avainsanaa.

Lisäämällä projektiin paketin staattinen muuttuja, voidaan asetuksia muokata kuvan 8 mukaisella tavalla. Komennoille on suositeltavaa tehdä oma luokka, jonka jälkeen luokan kaikki komennot voi rekisteröidä kerralla. Kuvassa 8 on esimerkillinen tapa komentoluokkien rekisteröimisestä.



```
class Program
{
    public static DiscordClient discord;
    static CommandsNextModule commands;

    0 references
    static void Main(string[] args)...

    1 reference
    static async Task MainAsync(string[] args)
    {
        // Discord configuration.
        discord = new DiscordClient(new DiscordConfiguration...);

        // Command settings.
        commands = discord.UseCommandsNext(new CommandsNextConfiguration
        {
            StringPrefix = "!",
            EnableDms = false
        });

        // Register commands
        CustomCommands.InitializeCustomCommands();      // Commands from .txt file.
        commands.RegisterCommands<Commands>();          // Regular commands.
        commands.RegisterCommands<CustomCommands>();    // Commands regarding custom commands.
        commands.RegisterCommands<WebCommands>();       // Web-scraping commands.
        commands.RegisterCommands<VoiceCommands>();     // Commands using voice.

        // Async await commands.
        await discord.ConnectAsync();
        await Task.Delay(-1);
    }
```

Kuva 8. Esimerkki komentoluokkien rekisteröimisestä.

4 MUKAUTETUT KOMENNOT

Yksi pääominaisuuksista, joka tullaan tekemään tälle itsenäiselle ohjelmalle on mukautetut komennot tai englanniksi ”Custom Commands”. Tällä tarkoitetaan komentojen tekemistä ohjelmalle ilman, että ne kovakoodataan koodiin. Tämä antaa muille käyttäjille mahdollisuuden tehdä ohjelmalle omia yksinkertaisia komentojaan ilman koodaamista.

4.1 Alustus ja luonti

Mukautetut komennot toteutetaan kirjoittamalla ja lukemalla tekstitiedostoa, johon ne säilytetään. Kun ohjelma käynnistetään, halutaan, että ohjelma lukee tekstitiedoston ja parsii komennot sekä vastaukset tiedostosta ja lisää sen listaan, josta ohjelma sen jälkeen tekee komennot. Kun uusi komento on luotu, halutaan siitä tehdä komento ohjelman sisällä sekä myös kirjoittaa ne ylös tekstitiedostoon ohjelmaa käynnistettäessä seuraavan kerran. Komentojen luomisen lisäksi annetaan käyttäjille myös mahdollisuus poistaa komento. Kuvassa 9 on kuvitus tavasta miten tälläiset komennot voidaan luoda.

Kuva 9. Kuvitus mukautetun komentojen toiminnasta.

Komentolistan alustaminen aloitetaan avaamalla tekstitiedosto (tai luomalla se, jos sitä ei ole olemassa). Seuraavaksi se luetaan ja sen sisältämät komennot lisätään listaan. Esimerkillisesti komennot voidaan jakaa riveihin, joissa ensimmäinen sana varataan komennolle ja sitä seuraavat sanat ohjelman vastaukselle. Kun ohjelma on

vastaanottanut tekstit, joista komennot luodaan, voidaan ne toteuttaa. Kuvassa 10 on yksinkertainen komentojen rekisteröintiprosessi.

```
Program.discord.MessageCreated += async e =>
{
    if (e.Message.Content.ToLower().StartsWith( Program.prefix + commandName))
        await e.Message.RespondAsync(botReply);
};
```

Kuva 10. Komennon rekisteröinti.

4.2 Komentojen lisäys ja poisto

Seuraavassa esimerkissä luodaan komento, jonka avulla kanavan esimiehet voivat tehdä komentoja huoneissa, joissa ohjelma sijaitsee.

Komento tarvitsee kaksi parametria toimiakseen; komennon nimen ja ohjelman vastauksen komennon kutsumiseen. On suositeltavaa, että tätä komentoa voi kutsua vain käyttäjät, joilla on mahdollisuus muokata rooleja kanavalla. Kun uutta komentoa luodaan, on hyvä tehdä tarkistuksia kuten esimerkiksi varmistaa, että onko kyseinen komento jo olemassa.

Komentojen poistaminen tapahtuu samalla tyylillä kuin lisääminen. Poistamisessa tarvitaan tietoon vain komennon nimi. Komennon tulisi kysyä komennon nimeä, jonka jälkeen esimerkiksi voidaan lukea ja kirjoittaa komennot listaan, josta voidaan tarkistaa onko komento olemassa. Jos komento on olemassa, niin komentorivi voidaan poistaa ja lista voidaan kirjoittaa uudelleen tiedostoon. Koska ohjelma rekisteröi komennot käynnistäessä, tämä komento vaatii toimiakseen ohjelman uudelleenkäynnistämisen.

5 VERKON KAAVINTA

Verkon kaavinnalla tarkoitetaan datan kaavintaa, jolla otetaan talteen dataa sivustoilta. Nettisivun kaavintaan liittyy datan noutaminen ja talteenotto. Noutamisella tarkoitetaan nettisivun lataamista. Kun nettisivun noutaminen on valmista, voidaan jäsentää, etsiä tai muotoilla sivun sisältöä uudelleen. Verkon kaavintaa käytetään yleensä silloin, kun nettisivulta otetaan tietoja ja niitä käytetään jossain muussa tarkoituksessa. [6]

5.1 Xpath

XPath (lyhenne sanoista *XML Path Language*) on ei-XML-pohjainen kieli XML-dokumenttien osien osoittamiseen ja XML-dokumentin rakenteeseen perustuvan tiedon luontiin. XPath-kieli perustuu XML-dokumentin puumuotoiseen esitystapaan ja antaa siten mahdollisuuden poimia eri osia dokumentista tietyillä valintakriteereillä. XPath kehitettiin halusta saada yhtenäinen syntaksi ja käyttäytymismalli XPointerin ja XSL:n välille. Kehittäjät ovat nopeasti ottaneet XPathin käyttöönsä pienenä kyselykielenä. [7]

Taulukosta 1 tulee ilmi, mitä syntaksia käyttää, kun etsii haluamaansa solmua xpathia käyttäen.

Valinnan tyyppi	Lyhennetty syntaksi	Täysi syntaksi
Valitse kontekstisolmun lapsielementti nimeltä "nimi"	nimi (lapsi on oletussuunta)	child::nimi
Valitse nykyisen puun juuri ja kaikki sen jälkeläiset	//	/descendant-or-self::node()/
Valitse nykyinen solmu	.	self::node()
Valitse ylemmän tason solmu	..	parent::node()
Valitse attribuutti nimeltä "nimi"	@nimi	attribute::nimi

Taulukko 1. Lyhennetyn ja täyden syntaksin vastaavuudet. [7]

5.2 Uutisten hakemisen komennon toteutus

Päädyin tekemään esimerkki web-kaavinta -komennon, jonka kutsuttua ohjelma menisi iltasanomien sivuille ja tulostaisi viisi uusinta uutista. Tähän sisältyy uutisen nimi, aihealue sekä kellonaika, milloin uutinen oli sivulle laitettu. Web-kaavintaan tulee käyttämään HtmlAgilityPack-pakettia. Projektiin tulee lisätä paketin viittaukset käyttämällä ”using” -avainsanaa projektin alussa.

Yksinkertaisuudessaan luodaan paketin mukana tullut WebClient- muuttuja, jonka avulla voidaan lukea verkkosivua lataamalla se WebClientissa olevalla tekstinlataus-komennolla. Tämä tarvitsee parametriksi luettavan verkkosivun. Kun sivu on ladattu merkkijonoksi, voidaan siitä ladata html-dokumentin ja valita tarvittava solmu kuvan 11 mukaisesti.

```
HtmlAgilityPack.HtmlDocument document = new HtmlAgilityPack.HtmlDocument();
document.LoadHtml(result);

HtmlNode[] nodes = document.DocumentNode.SelectNodes("//div[@id='tuoreimmat']//p").ToArray();
string[] text = new string[5];
```

Kuva 11. Esimerkki solmun hakemisesta web-sivulta.

Jotta saadaan haluttu tieto, täytyy etsiä oikea solmu ladatuilta internet-sivulta. Oikean solmun etsiminen voi olla hankalaa riippuen sivusta, ja tähän on myös tehty omia työkaluja. Tähän käytetään Chrome-nettiselaimen ”Inspect”-ominaisuutta, joka löytyy, kun painetaan nettisivulla hiiren oikeaa painiketta. Tämän avulla saadaan hyvä idea siitä mitä solmuja ollaan sivulta etsimässä.

Kyseisellä komennolla halutaan hakea viisi tuoreinta uutisotsikkoa ja tulostaa ne kanavalle. Solmut valitaan ”SelectNodes()”-funktiolla, joka tarvitsee xpath-lausekkeen. Kyseisessä tapauksessa xpath löytyi kuvan 12 mukaisesti.

Kuva 12. Xpathin löytäminen elementeistä.

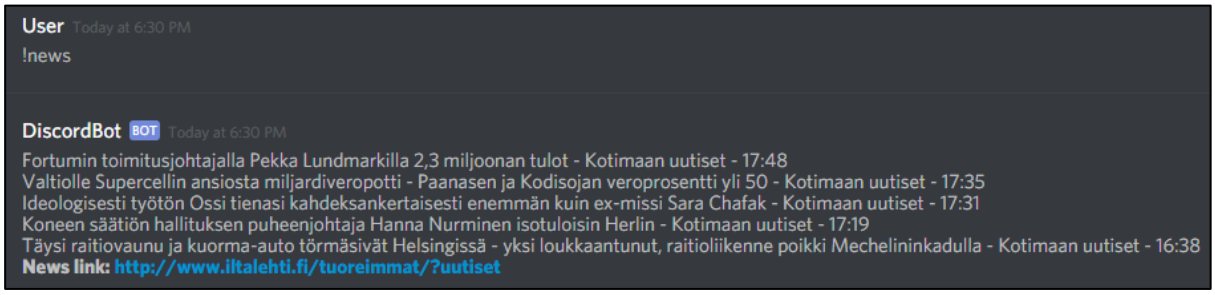
Kun xpath on löydetty, solmuista otetaan haluttu tieto. Tässä tapauksessa teksti, joka saadaan solmun ”InnerText” avulla kuvan 13 mukaisesti.

```
for (int i = 0; i < text.Length; i++)
    text[i] = RemoveHTML(nodes[i].InnerText);
```

Kuva 13. Tekstin parsiminen InnerText solmun avulla.

Joskus tekstiin voi tulla mukaan html-tunnisteita. Näistä pääsee eroon käymällä tekstin läpi yleisimmistä tunnisteista ja poistamalla ne tekstistä käyttämällä HTML-tunnisteiden poistamiseen tehtyä funktiota, joka käyttää hyväksi stringin korvausfunktiota.

Lopullinen tulos laittaa ohjelman näyttämään viisi tuoreinta uutista, kun komentoa kutsutaan sekä toiminnon lopuksi hyperlinkin sivulle, josta uutiset voi halutessaan lukea. Kuvassa 14 on tämän komennon kutsuminen käytännössä.



Kuva 14. Ohjelma tulostaa uutiset kanavalle.

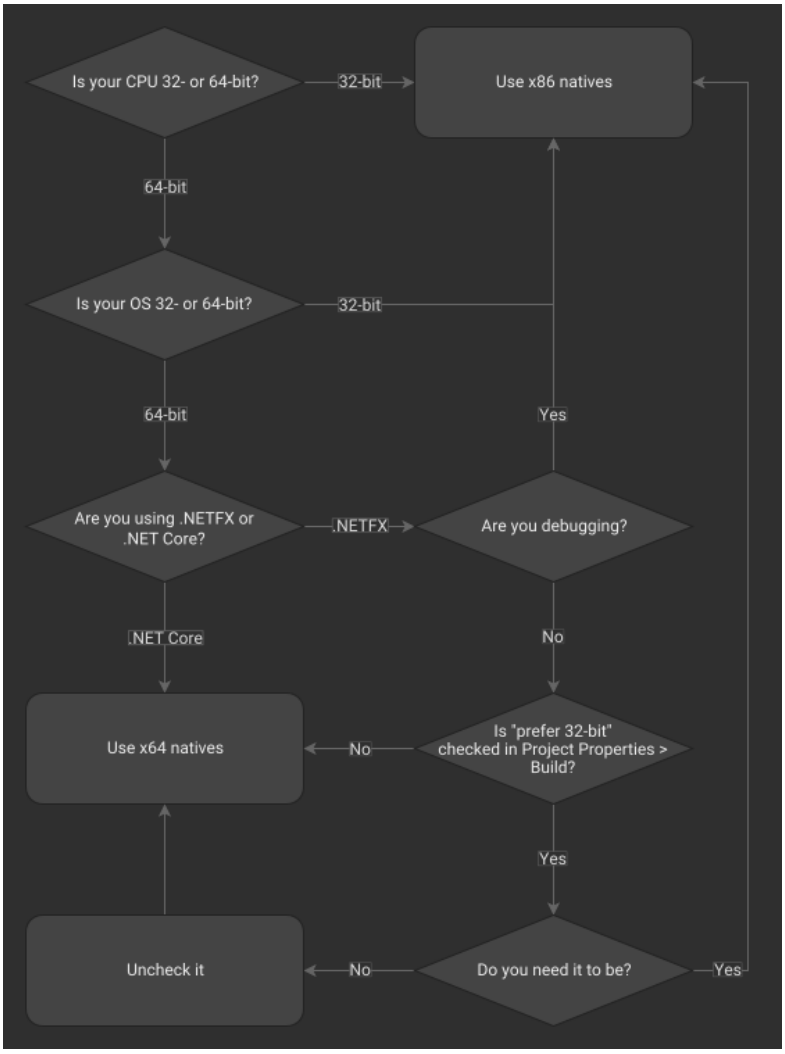
6 ÄÄNEN TOISTAMINEN

Discord sallii käyttäjien puhua kanavilla ääntä käyttäen. Tätä voi myös hyödyntää monin tavoin itsenäisillä ohjelmilla. Tätä hyödyntäen tehtiin komento DsharpPlussan tarjoamaa VoiceNext-pakettia apuna käyttäen. Jotta ääni saadaan toimimaan tarvitaan projektiin myös natiivikirjastoja.

Äänen toistoa hyödyntäen tehtiin komento, joka vaatii käyttäjältä Youtube URL-osoitteen. Komennon ja osoitteen saatuaan ohjelma soittaa äänen äänikanavalla, jossa sen kutsuja on. Tämän lisäksi lisätään komento äänen pysäytykseen.

6.1 Tarvittavat asennukset

Projektiin tulee lisätä DsharpPlussan tarjoama VoiceNext-paketti. Lisäksi tulee projektiin lisätä “Opus”- ja “Sodium”-kirjastot. Nämä kirjastot löytyvät helposti hakukoneita käyttämällä. Oikean kirjastoversion lataaminen riippuu käyttöjärjestelmästä ja prosessorista. Oikean kirjaston valintaan voi konsultoida kuvan 15 mukaista vuokaaviota. Äänen saamiseksi Youtube-sivulta käytössä on youtube-dl komentorivi-apuohjelma. Sen avulla äänen saaminen videoista eri sivuilta kuten Youtube, Dailymotion ja Vimeo on helpompaa. [8] Tämän ohjelman löytää myös helposti hakukonetta käyttämällä. Toimiakseen .NET Core -projekti vaatii nämä kirjastot sekä apuohjelman sijoittamisen projektin juureen. Kuvan 15 vuokaavion avulla tiedät, mitä natiiveja tulee käyttää.



Kuva 15. Vuokaavio oikean version valintaan. [8]

6.2 Musiikin toiston toteutus

Komennon kutsumiseen tarvitaan “kutsumisnimi” sekä youtube URL- linkki. Siitä otetaan ääni, jota ohjelma sitten toistaa.

Käytännössä komento tulee käynnistämään youtube-dl ohjelman, jonka avulla se hakee äänen videosta, jota sitten edelleen käytetään äänikirjastoihin. Näiden avulla äänikanavat puskevat äänen ohjelman kautta käyttäjille.

Komennon aloittamisessa tulee tehdä VoiceNextClient-muuttuja, jonka avulla voi yhdistää äänikanaville. Tämän saa VoiceNext-paketin asennuksen jälkeen DiscordClient-muuttujasta. Ennen kuin ohjelman yhdistää kanavalle on suotavaa

tarkistaa asioita, jotka voivat haitata komentoa. Esimerkiksi, onko ohjelma jo soittamassa ääntä tai onko komennon kutsuja äänikanavalla, mihin liittyä. Tarkistusten jälkeen käytetään hyväksi prosessiluokan funktiota, jolla voidaan käynnistää ulkopuolinen ohjelma. Näin käynnistetään ohjelma, joka muuttaa youtube-videon äänidataksi. Saatu äänidata lisätään puskuriin, jota sitten pusketaan ja lähetetään käyttäjille. Kuvassa 16 on esimerkkikoodi komennosta, joka soittaa musiikkia Discordin äänikanavalla youtube-linkin saatuaan.

```
public async Task PlayUrl(CommandContext ctx, [RemainingText] string url)
{
    var vnext = ctx.Client.GetVoiceNextClient();

    var vnc = vnext.GetConnection(ctx.Guild); // If theres an audio already playing, dispose of it
    if (vnc != null)
        if (vnc.IsPlaying)
            vnc.Dispose();

    var chn = ctx.Member.VoiceState.Channel;
    if (chn == null)
        await ctx.RespondAsync("You need to be in a voice channel.");

    vnc = await vnext.ConnectAsync(chn);

    await vnc.SendSpeakingAsync(true); // send a speaking indicator

    var psi = new ProcessStartInfo
    {
        FileName = "cmd.exe",
        Arguments = $"/C youtube-dl.exe -o - {url} | ffmpeg -i pipe:0 -ac 2 -f s16le -ar 48000 pipe:1",
        UseShellExecute = false,
        RedirectStandardOutput = true,
        CreateNoWindow = true
    };


    Process ffmpeg = Process.Start(psi);
    Stream ffout = ffmpeg.StandardOutput.BaseStream;

    byte[] buff = new byte[3840];
    int br = 0;
    while ((br = ffout.Read(buff, 0, buff.Length)) > 0)
    {
        if (br < buff.Length) // not a full sample, mute the rest
            for (var i = br; i < buff.Length; i++)
                buff[i] = 0;

        await vnc.SendAsync(buff, 20);
    }

    await vnc.SendSpeakingAsync(false); // we're not speaking anymore


    if (!vnc.IsPlaying) // leave if audio is not playing
        vnc.Dispose();
}
```

Kuva 16. Esimerkki youtuben kuuntelu komennosta.

Ohjelman musiikin pysäyttämiseen tarvittavaan komentoon voi vain yksinkertaisesti kertoa VoiceNextClient-muuttujalle neuvon poistua kanavalta ja sulkea äänikanavat komennon saatuaan.

7 OHJELMAN ISÄNNÖINTI

Ohjelmaa ei haluta ajaa ja pitää päällä omalla kotikoneellamme pitkiä aikoja, koska se vaatisi koneen kokoaikaista päällä olemista ja se veisi koneen käyttöresursseja.

Tämän projektin tarkoitukseen tullaan käyttämään Amazonin tarjoamaa www-sovelluspalvelua. Tullaan siihen tekemään virtuaalikoneen, jossa isännöidään ohjelmaa.

7.1 Amazon Web Services

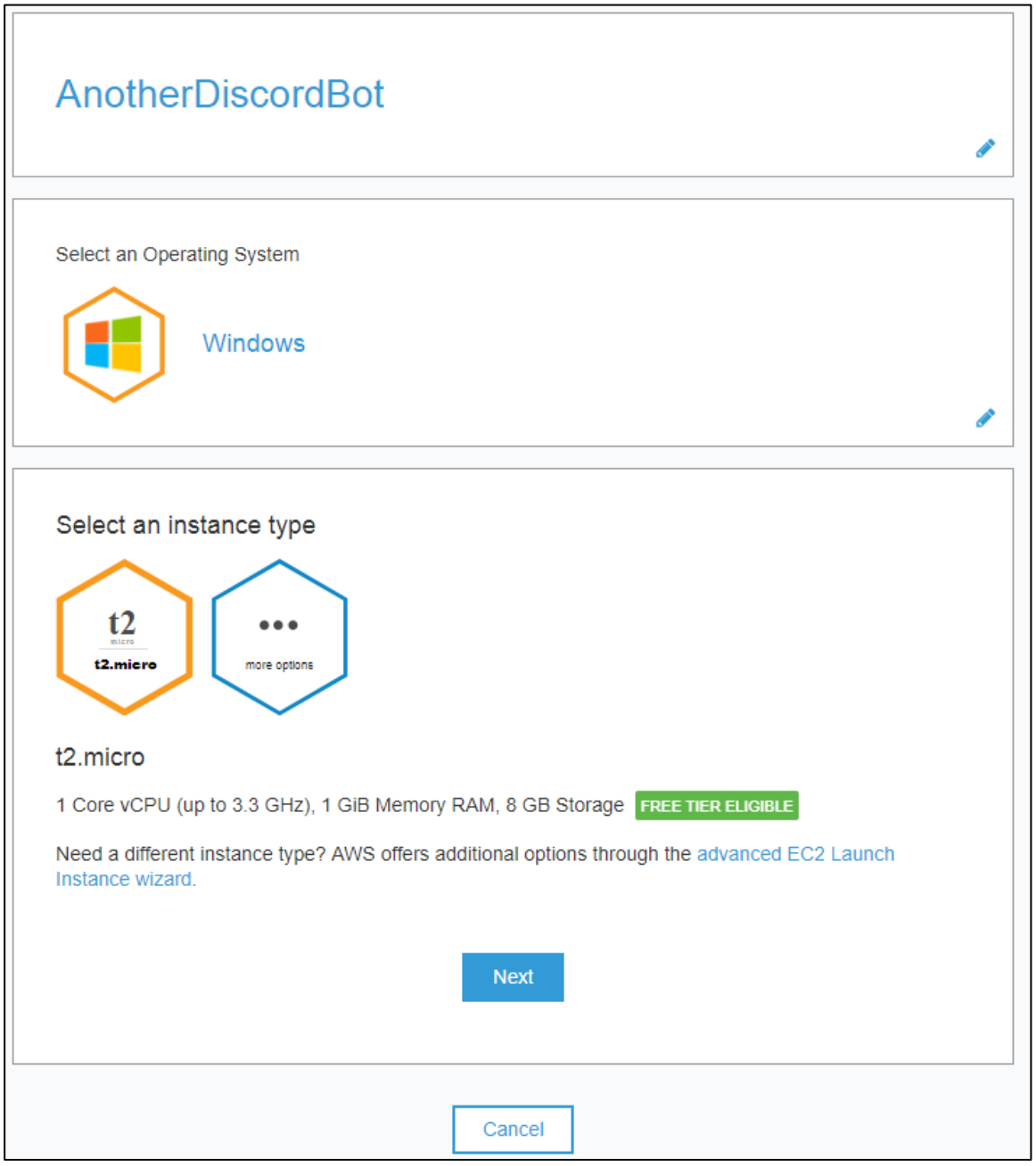
Amazonin pilvipalvelu antaa helppokäyttöisen, nopean ja halvan tavan isännöidä ohjelmamme.

7.2 Palvelimen rekisteröiminen ja asentaminen

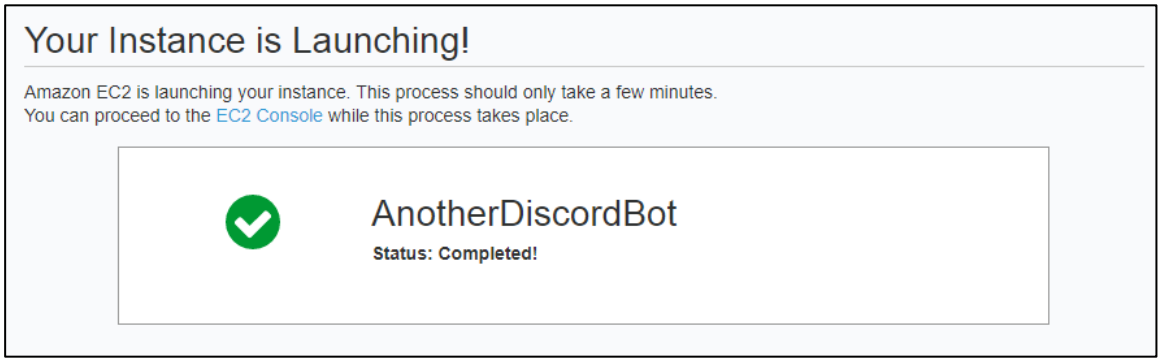
Rekisteröiminen tapahtuu “Amazon Web Servicen” sivuilla, napista ”Sign Up” [9]. Rekisteröiminen tulee vaatimaan käyttäjänimen ja salasanan lisäksi kotiosoitteen sekä laskutustiedot. Sivusto laskuttaa käyttäjiä vain, jos he ostavat sivustolta maksullisia palveluja.

Rekisteröimisen jälkeen voidaan yläpalkista valita, missä päin maailmaa virtuaalikone sijaitsee. Virtuaalikoneen luomista varten mennään etusivulle, josta valitaan ”Build a solution” -otsikon alta ”Launch a virtual machine” -painike. Tulevalta sivukkeelta valitaan ”EC2 Instance” -vaihtoehto, koska se on käyttötarkoitukseen sopiva. Tämän jälkeen valitaan virtuaalikoneelle sopiva nimi sekä käyttöjärjestelmä. Käydään läpi kuinka virtuaalikone laitetaan toimimaan ”Windows Server 2016” -käyttöjärjestelmällä. Windowsin valinnan jälkeen valitaan kuvan 17 mukaisesti instanssityypiksi ainoa ilmainen vaihtoehto, eli t2.micro. T2.micro sisältää 1-virtuaalisen prosessorin (jopa 3,3 GHz), 1 GIB RAM-muistia sekä 8 GB tallennustilaa, enemmän kuin tarpeeksi tähän projektiin. Lopuksi sivu antaa ladattavaksi avainparin, joka tulee olemaan myöhemmin tärkeä eli sitä ei saa hävittää. Asennus päätetään ”Create Instance”-napilla. Ohjelman luomiseen voi mennä muutamia minuutteja, jonka jälkeen saadaan viesti sen valmiudesta. Tämän jälkeen mennään EC2-konsoliin alhaalla olevan napin kautta tai

yläpalkista Services > EC2 > Running Instances -nappien kautta. Kuvassa 18 on virtuaalikoneen valmistumisen saatu viesti.

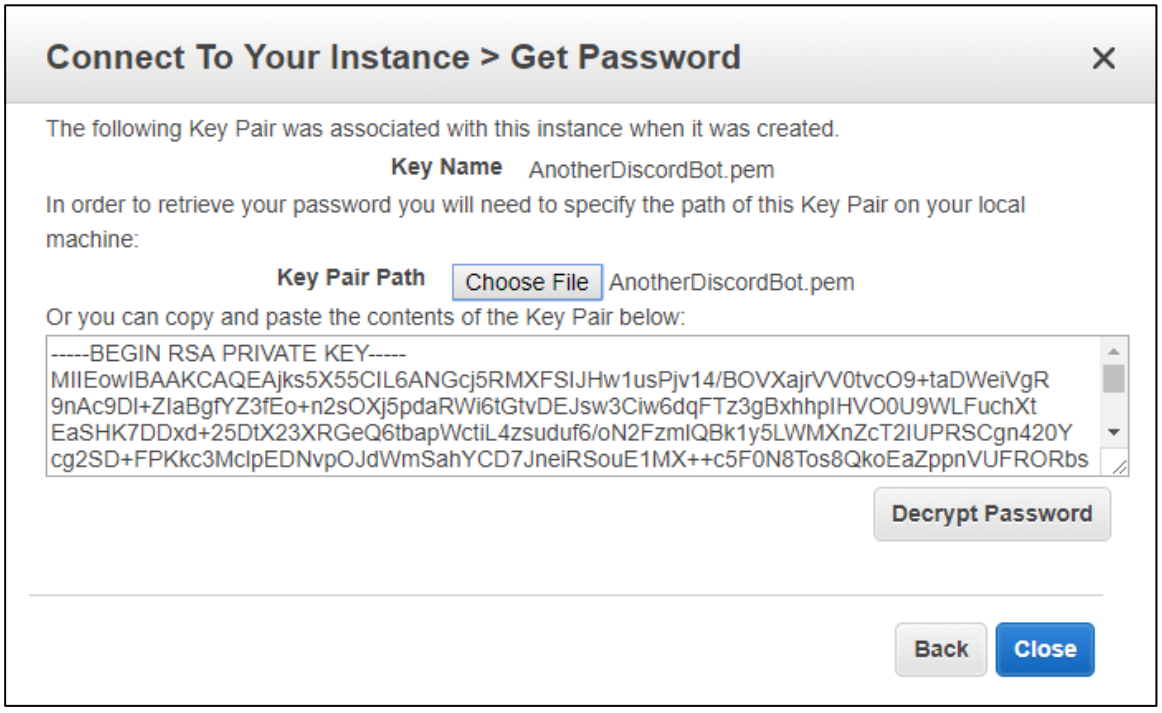


Kuva 17. Virtuaalikoneen luonti -ikkuna asetuksineen.

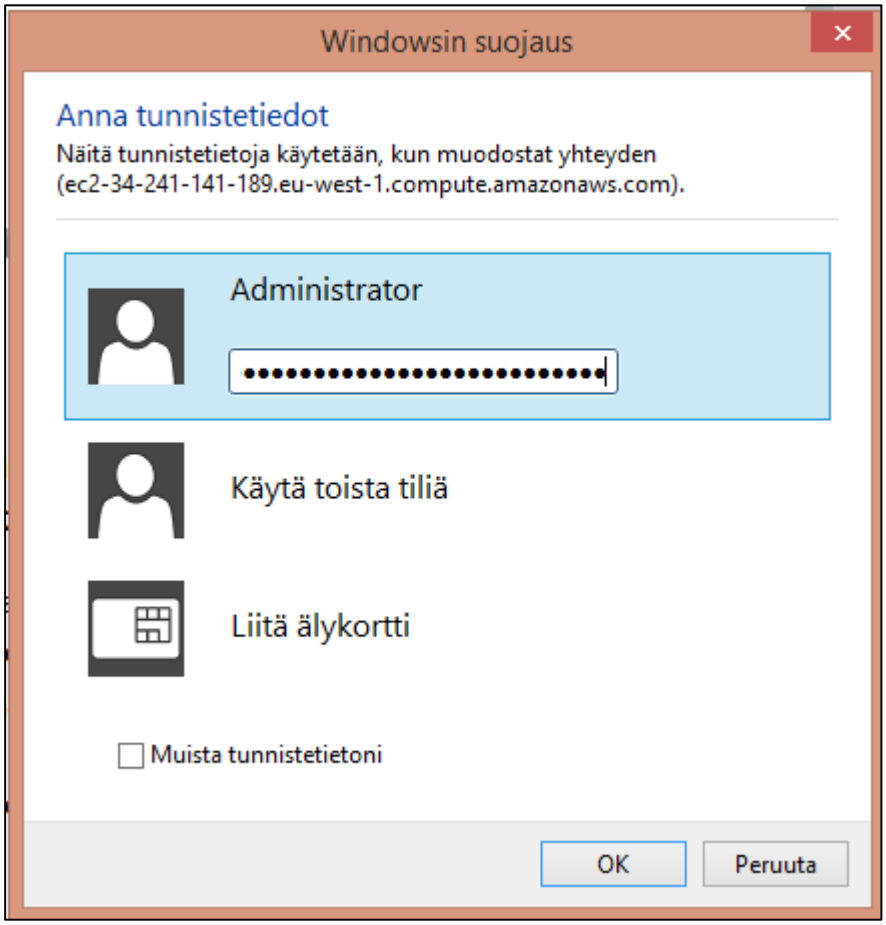


Kuva 18. Virtuaalikoneen valmistuminen.

EC2 konsolissa valitaan virtuaalikone ja painetaan sivun ylälaidassa olevaa ”Connect”-nappia. Tämä aukaisee kuvan 19 mukaisen ikkunan jossa voi tehdä kaksi asiaa: ensimmäiseksi ladataan etätyöpöytä –tiedosto, jolla yhdistetään virtuaalikoneeseen ja toiseksi haetaan virtuaalikoneen yhdistämiseen tarvittava salasana. Salasanan saamiseksi tarvitaan virtuaalikoneen luomisessa saatu ”.pem”- tiedosto, jonka avulla puretaan salasana. Virtuaalikoneeseen voidaan yhdistää aikaisemmin saadulla salasanalla käyttäen etäpöytä-tiedostoa kuvan 20 mukaisesti.



Kuva 19. Salasan saaminen tiedoston purkamisella.



Kuva 20. Windowsin etätyöpöydällä virtuaalikoneeseen yhdistäminen.

7.3 Ohjelman julkaisu ja palvelimella ajaminen

Yksi tapa ajaa ohjelmaa on julkaista projektista ajettava versio. Koska projekti käyttää .NET Corea ja kolmannen osapuolen kirjastoja, on otettava huomioon muutama lisäaskel. Projektin asetuksiin tulee lisätä, mihin käyttöjärjestelmään ajettava versio ohjelmasta tulee. Tämän asetus lisätään oikeaklikkaamalla ”csproj.” -tiedostoa ”Solution Explorer”- näkymästä, ja valitsemalla ”Edit”-valikon. Muokkausvalikossa lisätään uusi rivi <runtimeidentifiers>, <propertygroups> merkkijonon sisälle kuvan 21 mukaisesti. Runtimeidentifiers vaatii tiedon mihin käyttöjärjestelmään ohjelma julkaistaan, tämä löytyy kätevästi taulukon 2 avulla. [10]

Nimi	RID
Portable	win-x86 win7-x86
Windows 7 / Windows Server 2008 R2	win7-x64 win7-x86
Windows 8 / Windows Server 2012	win8-x64 win8-x86 win8-arm
Windows 8.1 / Windows 2012 R2	win81-x64 win81-x86 win81-arm
Windows 10 / Windows Server 2016	win10-x64 win10-x86 win10-arm win10-arm64

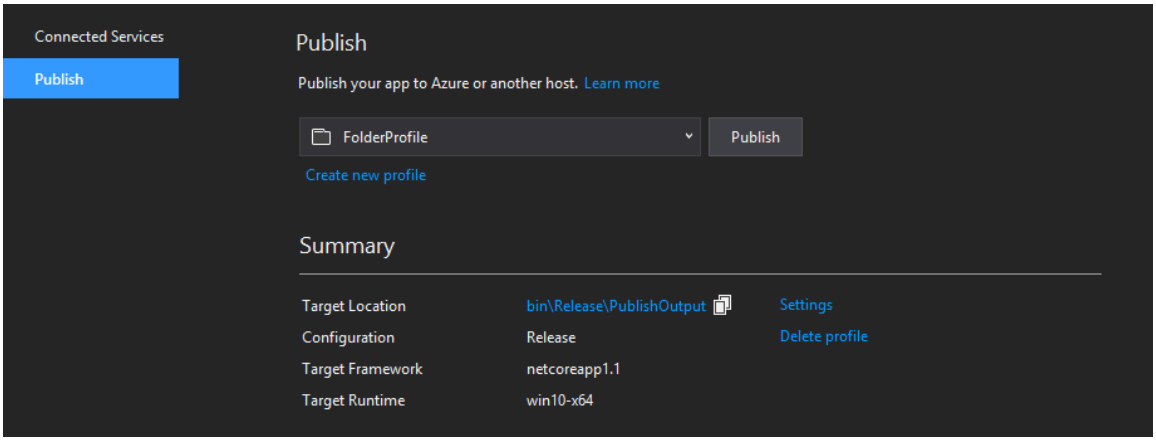
Taulukko 2. Runtime-tunnisteet Windowsin eri versioille. [11]

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>netcoreapp1.1</TargetFramework>
  <RuntimeIdentifiers>win10-x64</RuntimeIdentifiers>
  </PropertyGroup>
```

Kuva 21. Esimerkki RID-merkkijonon lisäämisestä projektiasetuksiin.

Kun RID-merkkijono on lisätty, on hyvä testata, toimiiko projekti varmasti ja sen jälkeen ajaa ohjelma debugissa. Kun kaikki toimii, niin ratkaisukokoonpano muutetaan Debugista Releasiin työkaluvalikosta. Publish-valikkoa käyttäen avautuu luomis-ikkuna kuvan 22 mukaisesti.



Kuva 22. Julkaisunäkymä.

Oletusarvoisilla asetuksilla julkaisuversio tulee projektikansion bin > Release > PublishOutput -kansion sisälle. Tähän kansioon tulee lisätä kolmannen osapuolen äänikirjastot sekä youtube-dl-ohjelma.

Jos kaikki on mennyt ilman ongelmia, kansiossa tulisi olla ajettava tiedosto, josta ohjelma menee päälle ja luo itsenäisen käyttäjän discord-kanavalle kaikkine ominaisuuksineen. Tämän jälkeen nämä tiedostot voidaan viedä kolmannen osapuolen koneelle ja käynnistää sieltä, jolloin ohjelma voi olla käynnissä pitkiä aikoja.

.

8 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnitella ja käydä läpi itsenäisen ohjelman ja sille erilaisten komentojen tekemistä käyttäen C#-ohjelmointikieltä ja hyväksi käyttäen kolmannen osapuolen kirjastoja ja wrapperiä. Tarkoituksena oli tehdä itsenäinen ohjelma, joka sisältää vähintään yhden tiedoston lukemisen avulla tehtyjä mukautettuja komentoja, äänikirjastoja hyväksikäyttäen komennon, joka toistaa musiikkia internetistä sekä web-kaavinnan avulla tehdyn komennon, joka tuo uusimmat uutiset web-sivulta suoraan “discord chat” -ohjelmaan.

Työn lopputuloksena päästiin tavoitteeseen, jossa saatiin valmiiksi .NET Corea käyttäen itsenäinen ohjelma Discord-kanavalle. Ohjelma käyttää ääneen, web-kaavintaan ja tiedoston lukuun käyttäviä funktioita. Lisäksi ohjelma on toiminnassa virtuaalikoneella kellon ympäri.

.

LÄHTEET

1 Wikipedia: Discord (software). Avaible at: https://en.wikipedia.org/w/index.php?title=Discord_(software)&oldid=834393923. Accessed Apr 5, 2018.

2 Microsoft: An introduction to NuGet. Avaible at: https://docs.microsoft.com/en-us/nuget/what-is-nuget. Accessed Mar 10, 2018

3 DSharpPlus: Homepage, Avaible at: https://dsharpplus.emzi0767.com/. Accessed Mar 10, 2018

4 Nitesh Luharuka: Getting Started With HTML Agility Pack. Avaible at: http://www.c-sharpcorner.com/UploadFile/9b86d4/getting-started-with-html-agility-pack. Accessed Jan 10, 2018

5 What is .NET?. Avaible at: https://www.microsoft.com/net/learn/what-is-dotnet. Accessed Jan 10, 2018

6 Wikipedia: Web Scraping, Avaible at: https://en.wikipedia.org/w/index.php?title=Web_scraping&oldid=841548834. Accessed May 21, 2018

7 Wikipedia: Xpath. Avaible at: http://fi.wikipedia.org/w/index.php?title=XPath&oldid=15725590. Accessed Apr 5, 2018

8 DsharpPlus: Setting up VoiceNext. Avaible at: https://dsharpplus.emzi0767.com/articles/vnext_setup.html. Accessed Apr 5, 2018

9 Amazon: Amazon Web Service. Avaible at: https://aws.amazon.com. Accessed Apr 6, 2018.

10 Microsoft: Deploying .NET Core apps with Visual Studio. Avaible at: https://docs.microsoft.com/en-us/dotnet/core/deploying/deploy-with-vs. Accessed Apr 6, 2018.

11 Microsoft: .NET Core RID Catalog. Avaible at: https://docs.microsoft.com/en-us/dotnet/core/rid-catalog. Accessed Apr 6, 2018