Roobert Björkbacka

Telegram API -viestintäsovellus

Opinnäytetyö
Kevät 2019
SeAMK Tekniikka
Insinööri (AMK), Tietotekniikka

SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Tutkinto-ohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Roobert Björkbacka

Työn nimi: Telegram API -viestintäsovellus

Ohjaaja: Markku Lahti

Vuosi: 2019 Sivumäärä: 36 Liitteiden lukumäärä:1

Opinnäytetyön aiheena oli Telegram API -rajapintaa hyödyntävä viestintäsovellus. Projektin toimeksiantaja on Alajärveläinen JAPO Palvelut Oy. Projektin tavoitteena oli rakentaa sovelluskokonaisuus, joka sisältää kaksi osaa. Telegram Bot -rajapintaa hyödyntävän sovelluksen, joka tallentaa Telegram-keskustelut tietokantaan, sekä ASP.NET MVC pohjaisen web-sovelluksen, joka hakee keskustelut tietokannasta ja näyttää keskustelut MVC-käyttöliittymässä.

Työn teoriaosuudessa tutkittiin verkkoviestintää ja eri pikaviestintäsovelluksien ominaisuuksia. Teoriaosuudessa tutkittiin myös robottiohjelmia eli botteja sekä sovelluskehityksessä käytettyjä tekniikoita. Työn sovellusosuudessa käytiin läpi viestintäsovelluksen kehitysprosessi ja sovelluksen testaus. Työn lopputuloksena saatiin toimiva sovelluskokonaisuus.

Avainsanat: Telegram API, Viestintäsovellus, SQL-tietokanta, Bot

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Programming

Author: Roobert Björkbacka

Title of thesis: Telegram API Communication Software

Supervisor: Markku Lahti

Year:2019 Number of pages:36 Number of appendices:1

The subject of this thesis was to develop a communication software that utilizes Telegram API. It was made at the request of JAPO Palvelut Oy. The goal of this project was to build an application which would consist of two elements. The first element was planned to utilize the Telegram Bot API to save the conversations within the Telegram application into a database. The second element was an ASP .NET MVC based web-application, the task of which was to retrieve the saved conversations from the database and show them on an MVC-user interface.

In the theory section of this thesis, online communications, different instant messaging applications, robot programs known as bots, and all the technologies used in the thesis were analyzed. The implementation and testing of the application were explained in the fifth and sixth chapter of this thesis. The end result was an application working as intended.

Keywords: Telegram API, communication software, SQL-database, Bot

SISÄLTÖ

Opinnäytetyön tiivistelmä 2
Thesis abstract 3
SISÄLTÖ 4
Kuvioluettelo 6
Käytetyt termit ja lyhenteet 8
1 JOHDANTO 9
1.1 Työn tausta 9
1.2 Työn tavoite 9
1.3 Työn rakenne 9
2 VERKKOVIESTINTÄ 11
2.1 Yleistä 11
2.2 Pikaviestimet 11
3 BOTTI 13
3.1 Mikä on botti? 13
3.2 Telegram Bot 13
4 KÄYTETYT TEKNIIKAT 14
4.1 Rajapinta (API) 14
4.2 .NET-alusta 14
4.3 SQL-tietokanta 16
5 TELEGRAM-SOVELLUS 17
5.1 Yleistä 17
5.2 Telegram-client 17
5.3 Telegram Bot 19
5.4 MVC-käyttöliittymä 24
5.4.1 Keskusteluryhmien hakunäkymä ja viestiketjun lukunäkymä 24
5.4.2 Keskusteluryhmän perustaminen 29
6 SOVELLUKSEN TESTAUS 32
7 POHDINTA 33
LÄHTEET 34

LIITTEET 36

Kuvioluettelo

Kuvio 1 .NET arkkitehtuuri (DeveloperIn [Viitattu 11.2.2019]) 15

Kuvio 2. Telegram-todennuskoodi 17

Kuvio 3. Lista keskusteluryhmää varten. 18

Kuvio 4. Ryhmän luominen 19

Kuvio 5. TelegramFile-taulun lisääminen tietokantaan. 19

Kuvio 6. Tietokannan luokkakaavio. 20

Kuvio 7. Botin alustus 21

Kuvio 8. Dokumenttiviestin käsittely ja tallennus 21

Kuvio 9. Tiedostojen tallennus. 22

Kuvio 10. Tarkistuskysely tietokantaan 23

Kuvio 11. Viestien tallennus tietokantaan. 24

Kuvio 12. Viestien hakunäkymä 25

Kuvio 13. Viesti-ikkuna 25

Kuvio 14. Keskusteluryhmien hakumetodi. 26

Kuvio 15. Keskusteluryhmän haku 27

Kuvio 16. Ryhmälistan purkaminen näkymään. 27

Kuvio 17. Viestilistan käsittely. 28

Kuvio 18. Viestilistan purkaminen Foreach-lauseessa 29

Kuvio 19. Keskusteluryhmälomake. 30

Kuvio 20. Telegram-ryhmän luominen ja sovelluksen todennus. 31

Kuvio 21. Virhesivu 32

Käytetyt termit ja lyhenteet

Asynkroninen ohjelmointi

Asynkronisessa ohjelmoinnissa ohjelmalausekkeita ei välttämättä suoriteta siinä järjestyksessä, missä ne esiintyvät ohjelmakoodissa.

Botti Botti tarkoittaa robottiohjelmaa, joka suorittaa yleensä monia toistoja vaativia tehtäviä itsenäisesti.

Metodi Metodi on olio-ohjelmointiluokan aliohjelma, jota voidaan kutsua pääohjelmasta.

Pikaviestin Pikaviestin on tietokoneohjelma, jonka avulla kaksi käyttäjää tai useampi voi keskustella keskenään reaaliajassa.

Rajapinta Rajapinta eli API (Application Programming Interface) on standardin mukainen yhtymäkohta, jonka avulla tietoja voidaan siirtää ohjelman ja käyttäjän välillä.

Todentaminen Todentaminen on menetelmä, jonka avulla sovelluksen todenmukaisuus varmennetaan.

1 JOHDANTO

1.1 Työn tausta

JAPO Palvelut Oy yrityksellä oli toimeksiantona kehittää yritys X:lle sovellus, jonka avulla asiakasviestintä voidaan tallentaa tietokantaan sekä lukea tarvittaessa myöhemmin.

1.2 Työn tavoite

Työn tavoitteena oli luoda toimiva viestintäsovelluskokonaisuus Telegram API -rajapintaa käyttäen. Sovelluskokonaisuuteen kuului robottiohjelma, joka tallentaa Telegramissa käydyt keskustelut tietokantaan ja ASP.NET MVC -tekniikalla toteutettu sovellus, joka hakee Telegram-keskustelut tietokannasta ja näyttää ne käyttäjälle MVC-käyttöliittymässä.

Työssä oli tarkoituksena myös tutustua rajapintojen hyödyntämiseen sovelluskehityksessä, tietokantojen rakentamiseen ja MVC-sovelluksien kehittämiseen.

1.3 Työn rakenne

Opinnäytetyössä kartoitettiin asiakasyrityksen toiveet sovelluksen toiminnasta ja käyttöliittymän ulkonäöstä.

Luvussa kaksi käydään läpi verkkoviestinnän perusteita ja tarkastellaan eri pikaviestinsovelluksien ominaisuuksia.

Luku kolme käsittelee robottiohjelmia eli botteja, sekä kuinka botteja käytetään Telegramissa.

Luvussa neljä käydään läpi opinnäytetyössä käytetyt tekniikat.

Luvussa viisi käsitellään läpi opinnäytetyön käytännön osio eli sovelluskehitys. Luvun alussa on projektin yleiskatsaus, jonka jälkeen käydään läpi ohjelmakoodit ja tietokantojen alustaminen. Luvun lopussa käsitellään sovelluksen käyttöliittymän rakentaminen.

Luku kuusi käsittelee sovelluksen testaamista ja testituloksista laadittua testiraporttia. Testiraportti on liitteenä työn lopussa.

Luvussa seitsemän on työn yhteenveto ja pohdinta työn tavoitteista ja toteutuksesta.

2 VERKKOVIESTINTÄ

2.1 Yleistä

Verkkoviestintä on koko ajan kehittyvä viestintämuoto, jossa viestit liikkuvat tietoverkon välityksellä. Yritykset hyödyntävät verkkoviestintää monella eri tavalla, kuten markkinoinnissa, asiakasviestinnässä ja yrityksen sisäisessä viestinnässä. Markkinointi on siirtynyt viime vuosina enemmän sosiaalisen median kanaville, kuten esimerkiksi Facebookiin, Instagramiin sekä verkkosivuille mainosten muodossa. Asiakaspalvelunumeroista vastaa usein robottisovellus, ei ihminen. Perinteiset paperikirjeet ovat muuttuneet sähköposteiksi ja pikaviestimien ilmoituksiksi. (Isokangas & Vassinen 2010,16-18, 27.)

Pikaviestintä. Pikaviestintä tarkoittaa ohjelmiston toimintoa, joka mahdollistaa kahden tai useamman käyttäjän keskustelun reaaliaikaisesti (TSK 2007.) Pikaviestintää on pidetty vain yksityisessä käytössä tapahtuvana vapaamuotoisena viestittelynä, mutta se on yleistynyt myös yrityskäytössä. Pikaviestintä on nimensä mukaisesti nopeampaa kuin perinteinen sähköpostiviestittely. Monissa pikaviestimissä on myös ominaisuus, joka näyttää onko käyttäjä läsnä palvelussa. Moni yritys käyttää pikaviestintää esimerkiksi tukipalveluissa, jossa asiakasta voidaan neuvoa reaaliajassa. (Laitila 2017.)

2.2 Pikaviestimet

Pikaviestin on ohjelma, jonka avulla käyttäjä voi keskustella yhden tai useamman käyttäjän kanssa (Jyväskylän yliopisto 2009). Pikaviestimiä on moneen käyttötarkoitukseen ja eri laitteistoille. Yleisimpiä yrityskäytössä olevia pikaviestimiä tietokoneella ovat esimerkiksi Slack ja Microsoft Teams (Laitila 2017). Nykyisin pikaviestintä tapahtuu useimmiten mobiililaitteilla. Tunnettuja mobiililaitteille suunnattuja pikaviestimiä ovat esimerkiksi WhatsApp, Snapchat, Facebook Messenger ja Telegram. (Hill 2019.)

WhatsApp. WhatsApp on Facebookin omistama pikaviestinsovellus, jolla on yli miljardi käyttäjää. WhatsApp on ilmainen sovellus, se on saatavilla Google Storesta sekä Apple App Storesta. WhatsApp-sovellusta voi käyttää myös tietokoneella web-selaimen kautta tai työpöytäsovelluksella. WhatsApp tarjoaa myös mahdollisuuden ääni- ja videopuheluihin. (WhatsApp 2018.)

WhatsApp käyttää end to end -salausteknologiaa. Sovelluksessa lähetetyt viestit salataan salausavaimella. Salausavain on vain viestin lähettäjällä ja vastaanottajalla, joten viestejä ei voi lukea kukaan muu. Jokainen lähetetty viesti salataan eri salausavaimella. Viestejä ei tallenneta WhatsApp-sovelluksen tietokantaan vaan ainoastaan käyttäjän laitteelle. WhatsApp käyttää samaa salausta myös WhatsApp-sovelluksessa käytyihin puheluihin. (WhatsApp 2019.)

Snapchat. Snapchat on pikaviestintäsovellus, joka on etenkin nuorten suosiossa. Snapchatin suosion perustuu monipuolisiin kamerasuodattimiin, joiden avulla kasvokuvia voi muokata. Toinen suosittu ominaisuus on Snapchat-tarinat. Tarinat näkyvät kaikille käyttäjän lisäämille kavereille vuorokauden ajan, jonka jälkeen ne katoavat. Tarinoita voi jakaa myös julkisesti, jolloin ne näkyvät kaikille Snapchat-käyttäjille. (Snapchat 2018.)

Telegram. Telegram on 2013 julkaistu pilvipohjainen pikaviestintäsovellus, joka keskittyy nopeaan ja turvalliseen viestintään. Telegram on saatavilla kaikille yleisimmille käyttöjärjestelmille (Android, iOS, Windows, OSX ja Linux). Telegram on ulkoisesti saman tyyppinen sovellus kuin WhatsApp, mutta Telegramiin on luotu mahdollisuus lisätä keskusteluryhmään Botteja eli eräänlaisia virtuaalikäyttäjiä, joita voidaan ohjelmoida tekemään erilaisia tehtäviä. (Telegram 2019.)

Telegram käyttää WhatsAppin kaltaista end to end -salausteknologiaa, mutta Telegram sisältää myös secret chat -ominaisuuden, joka salaa viestit kahdesti. Secret chat -keskustelussa viestejä ei voi välittää. Viestit voi määrittää katoamaan tietyn ajan kuluttua, ja keskustelu toimii vain laitteessa, jossa keskustelu on aloitettu. (Telegram 2019.)

3 BOTTI

3.1 Mikä on botti?

Botti on robottiohjelma, joka on suunniteltu suorittamaan helppoja tehtäviä esimerkiksi asiakaspalvelutilanteessa. Botti voi esimerkiksi ottaa vastaan tilauksia ravintolan verkkosivulla. Botit lukevat käyttäjän antamia käskyjä, hakevat tietoja käyttäen eri rajapintoja ja antavat tiedot käyttäjälle nopeasti ja vaivattomasti. (Microsoft 2018.)

Internet on täynnä botteja. Botit hakevat käyttäjille tietoja hakukoneen muodossa. Esimerkiksi hintarobotit etsivät käyttäjälle sivun, josta haluttu tuote löytyy halvimpaan hintaan. Botti voidaan myös ohjelmoida tarkkailemaan käyttäjän internetkäyttäytymistä, ja saatujen tietojen perusteella botti voi näyttää käyttäjälle kohdennettua mainontaa. (Symantec [Viitattu 19.2.2019].) Kirjassa digitaalinen jalanjälki Isokangas ja Vassinen (2010, 27) korostavat, että nykyisin modernin yrityksen on kannattavaa pysyä mukana teknologian kehityksessä ja hyödyntää uusia tapoja viestiä asiakkaiden kanssa, kuten esimerkiksi bottien avulla.

3.2 Telegram Bot

Telegramissa botti tarkoittaa kolmannen osapuolen laatimaa ohjelmaa, jota ajetaan telegramin ulkopuolelta. Botti-käyttäjä lisätään Telegram-keskusteluryhmään, jonka jälkeen botti lukee käyttäjän komentoja ja vastaa niihin. Bottikomentoja pystyy käyttämään liittämällä botin nimen Telegram-keskuteluun ja lisäämällä perään komennon nimen, esimerkiksi: ”@EsimBotti /help”. Esimerkin komennossa mainitaan botin nimi @-merkin avulla ja sen perään lisätään komento /help. Botti voisi esimerkiksi palauttaa käyttäjälle ohjeita botin käyttöön. (Telegram [Viitattu 5.1.2019].)

4 KÄYTETYT TEKNIIKAT

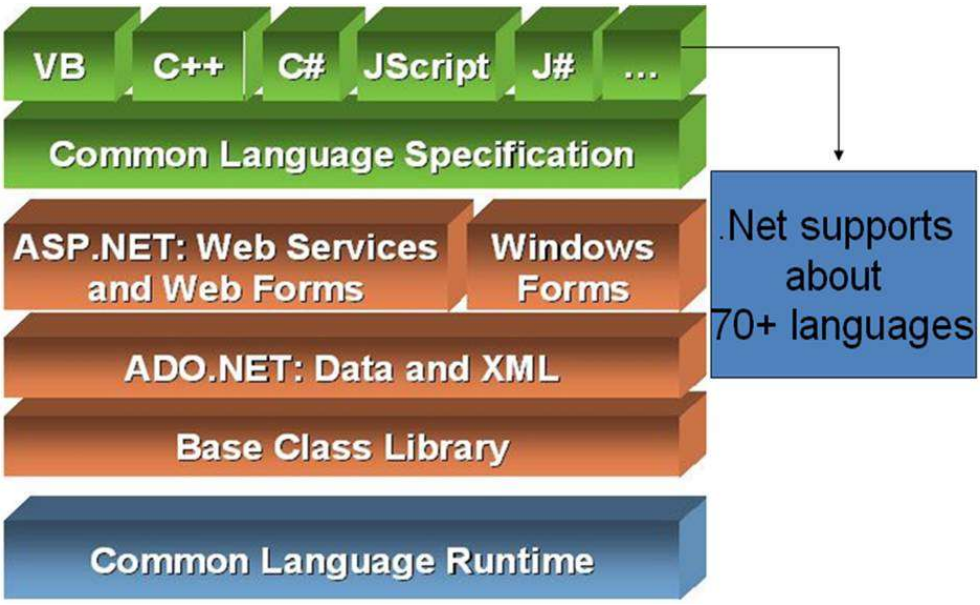
4.1 Rajapinta (API)

Rajapinta tarjoaa datan vapaan käytön palvelussa tai sovelluksessa. Käytännössä palvelu voi avata rajapinnan, jota kuka tahansa voi hyödyntää omassa sovelluksessaan. Esimerkiksi karttasovellus voisi käyttää hyödykseen Googlen karttarajapintaa. Joidenkin rajapintojen käyttö on maksullista. Rajapinnat helpottavat valmiin datan käyttöä, koska dataa on helpommin saatavilla yhdenmukaisessa muodossa. (DataBusiness [Viitattu 8.1.2019].)

Telegram API. Telegramilla on kaksi avoimen lähdekoodin rajapintaa (Telegram API ja Bot API). Telegram API -rajapinnan avulla voidaan rakentaa räätälöityjä Telegram-asiakasohjelmia. TelegramBot API -rajapinnan avulla ohjelmoituja botteja voidaan yhdistää telegram-verkostoon. (Telegram [Viitattu 9.1.2019].)

4.2 .NET-alusta

.NET on Microsoftin kehittämä Windows-pohjainen ohjelmistokomponenttikirjasto. Yleisimpinä ohjelmointikielinä toimivat Visual Basic, C++ ja C#, mutta .NET tukee myös monia muita ohjelmointikieliä. (Microsoft 2019a.) Kuviossa 1 on kuvattu .NET-alustan osat. Opinnäytetyöprojektissa käytettiin C#-ohjelmointikieltä ja ASP.NET-teknologiaa.



Kuvio 1 .NET arkkitehtuuri (DeveloperIn [Viitattu 11.2.2019])

ASP.NET. ASP.NET on Microsoftin kehittämä viitekehys, jonka avulla voidaan luoda verkkosivuja ja sovelluksia käyttäen HTML-, CSS- ja JavaScript-ohjelmointikieliä. ASP.NET sisältää kolme eri viitekehystä: Web Forms, MVC ja Web Pages. (Microsoft 2019b.)

MVC. MVC on yksi ASP.NET-alustan kolmesta viitekehyksestä, joka on suunniteltu mobiili- tai yhden sivun sovelluskehitykseen. Lyhenne MVC tulee sanoista model, view ja controller. Näiden kolmen eri moduulin avulla ohjelmakoodi voidaan erotella. (Microsoft 2019b.)

Model eli malli vastaa tietojen käsittelysta, ylläpidosta ja tallentamisesta. Esimerkiksi tietojen tallentamisesta tietokantaan. Controller eli käsittelijä käsittelee käyttäjiltä tulevat käskyt ja näyttää käyttäjälle halutun näkymän. Käsittelijä sisältää yleensä suurimman osan ohjelman logiikasta. Näkymä eli view on tiedon esitystapa, jonka käsittelijä valitsee käyttäjän pyyntöjen mukaan. Näkymään rakennetaan ohjelman käyttöliittymän ulkoasu ja kuvataan, miten ja missä muodossa tiedot esitetään käyttäjälle. Näkymän määrittelyyn käytetään pääosin HTML-kuvauskieltä, mutta sovelluksissa tarvitaan usein myös muita ohjelmointikieliä, esimerkiksi JavaScript-ohjelmointikieltä. (Lahtonen 2016.)

C#-ohjelmointikieli. C#-ohjelmointikieli on Microsoftin kehittämä olio-orientoitunut ohjelmointikieli, joka on julkaistu vuonna 2000. Kielen kehityksen pohjana käytettiin muita olio-orientoituneita kieliä, kuten C++ ja Java. C# muistuttaa kirjoitusasultaan paljon C- ja C++-ohjelmointikieliä, koska tavoitteena oli säilyttää niiden hyvät puolet ja muokata toimivuutta vain tarvittaessa. (Liberty 2005, 3-7.)

4.3 SQL-tietokanta

SQL-tietokanta on tietokoneelle asennetulla palvelimella toimiva tietojen kokoelma, jota voidaan muokata ja käyttää erilaisilla hakumenetelmillä. Lähes kaikissa tietokannoissa käytetään SQL-ohjelmointikieltä. Neljä isointa tietokantasovellusta ovat Oracle, DB2, SQL Server ja MySQL. (Hovi 2008, 2-4.) Tässä opinnäytetyössä käytettiin Microsoftin SQL-server-tietokantaa.

Relaatiomalli. Tietokantojen relaatiomalli on E.F Coddin vuonna 1970 kehittämä malli, joka määrittelee pohjan relaatiotietokannoille. Relaatiomalli syrjäytti aiemmin käytetyt tietokantatekniikat ja samalla SQL-kieli standardoitui kaikkien relaatiokantojen kieleksi. Relaatiokannat rakentuvat tauluista, jotka sisältävät sarakkeita ja rivejä. Sarakkeille annetaan nimi ja tietotyyppi, eli millaista tietoa sarakkeeseen voidaan tallentaa, ja tiedot tallennetaan taulun riveille. Jokaisessa relaatiomallin taulussa on oltava perusavain, jonka täytyy olla yksilöivä eli sarakkeen eri riveillä ei voi olla samaa arvoa. Tietokannan tauluja liitetään yhteen viiteavaimen avulla. Viittaava taulu eli lapsitaulu sisältää viiteavaimen, joka viittaa viitattavan taulun eli isätaulun perusavaimeen ja näin taulujen välille syntyy liitos. (Hovi, 2008, 5-7.)

Upotettu SQL. SQL-käskyjä voidaan upottaa suoraan ohjelmointikieleen. Tietokantakyselyt voidaan sijoittaa suoraan sovelluksen ohjelmakoodiin sellaisenaan tai kyselyt voidaan suorittaa rajapintaliittymän kautta. (Hovi 2008, 16.) Tämän opinnäytetyön tietokantakyselyt on toteutettu upottamalla SQL-kyselyt suoraan C#-ohjelmakoodiin.

5 TELEGRAM-SOVELLUS

5.1 Yleistä

Sovellus toteutettiin JAPO Palvelut Oy:n ASP.NET MVC -alustaan, johon oli valmiiksi alustettu tarvittavat kirjastot. Sovelluskehitys jakautui kolmeen vaiheeseen: Telegram-client, Telegram Bot ja MVC-käyttöliittymä. Sovellus rakennettiin Microsoftin Visual Studio 2017 -kehitysympäristössä ja tietokanta lisättiin Microsoftin SQL Server 2017 -tietokantapalvelimelle käyttäen Microsoft SQL Server Management Studio (SSMS) -ohjelmaa.

5.2 Telegram-client

Sovellyskehitystä varten alustettiin Telegram API. Telegramin verkkosivuille kirjautumalla saatiin tarvittavia tietoja, kuten API-avain sekä palvelimen IP-osoite, joiden avulla viestintäsovellus yhdistettiin Telegramin järjestelmään. Ennen kuin ohjelmisto voidaan yhdistää telegramin palvelimeen, käyttäjän on todennettava sovellus omalla laitteellaan. Todennus tehtiin syöttämällä Telegram API -rajapinnassa olevalle AskAuthCode-metodille (kuvio 2) puhelinnumero, jonka jälkeen Telegram lähettää puhelimeen koodin. Koodi syötetään Telegramin MakeAuthentication-metodiin, jonka jälkeen Telegram API on valmiina käytettäväksi.

```
public async System.Threading.Tasks.Task<OpenTl.Schema.Auth.TSentCode> AskAuthCode(string phone)
{
    NumberToAuthenticate = phone;
    OpenTl.Schema.Auth.ISentCode codeTask = await this.client.AuthService.SendCodeRequestAsync(phone);
    var sentCode = (OpenTl.Schema.Auth.TSentCode)codeTask;
    return sentCode;
}
```

Kuvio 2. Telegram-todennuskoodi

Todennuksen jälkeen Telegram-ryhmä perustetaan Telegram API -rajapinnan kautta. Ryhmän perustamiseen tarvitaan lista käyttäjistä, jotka ryhmään lisätään, sekä ryhmän nimi. Listaa varten tehtiin GetList-metodi (kuvio 3) , jolle syötetään käyttäjän puhelinnumero. Puhelinnumeron avulla haetaan käyttäjän yksilöllöinen ID-

numero, jonka avulla Telegram API löytää käyttäjän, ja lisää sen luotuun ryhmään. Samalla lisätään ryhmään myös Botti-käyttäjä.

```
public async System.Threading.Tasks.Task<List<TelegramUser>> GetList(string phonenumber)
{
    var result = await client.ContactsService.GetContactsAsync();
    phonenumber = phonenumber.Trim('+');
    var user = result.Cast<OpenTl.Schema.Contacts.TContacts>().Users.Items
                        .OfType<TUser>()
                        .FirstOrDefault(x => x.Phone == phonenumber);

    List<TelegramUser> lista = new List<TelegramUser>();
    long bothash = 1;
    int botID = 1;
    TelegramUser bot = new TelegramUser(botID, bothash);
    lista.Add(bot);
    if (user!=null)
    {
        long hash = user.AccessHash;
        int id = user.Id;
        TelegramUser User = new TelegramUser(id, hash);
        lista.Add(User);
    }
    return lista;
}
```

Kuvio 3. Lista keskusteluryhmää varten.

Lista ja keskusteluryhmän nimi annetaan CreateGroup-metodille (kuvio 4) ja keskusteluryhmä lisätään Telegramin CreateChatAsync-metodin avulla. Toiminnallisuus testattiin ensin kovakoodaamalla tiedot suoraan ohjelmakoodiin, mutta myöhemmin ohjelmiston autentikointi ja ryhmän luominen liitettiin ASP.NET MVC -käyttöliittymään.

```
public async System.Threading.Tasks.Task<TChat> CreateGroup(string groupName, List<TelegramUser> dbUsers)
{
    TVector<IInputUser> users = new TVector<IInputUser>();

    foreach (TelegramUser rawUser in dbUsers)
    {
        IInputUser readyuser = new TInputUser() { UserId = rawUser.TelegramUserID, AccessHash = rawUser.AccessHash };
        users.Items.Add(readyuser);
    }

    IUpdates result = await client.MessagesService.CreateChatAsync(groupName, users).ConfigureAwait(false);

    TUpdates updatesResult = result as TUpdates;
    if (updatesResult != null)
    {
        IChat createdChat = updatesResult.Chats.Items.FirstOrDefault();
        if (createdChat != null)
        {
            TChat chat = createdChat as TChat;
            if (chat != null)
            {
                return chat;
            }
        }
    }

    return null;
}
```

Kuvio 4. Ryhmän luominen

5.3 Telegram Bot

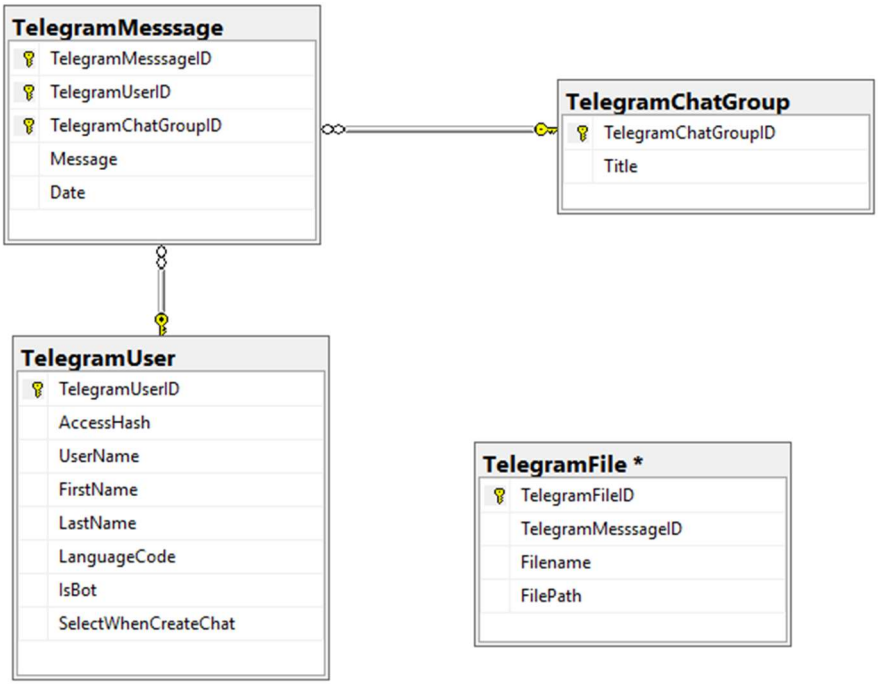
Bottia varten perustettiin erillinen Visual Studio -projekti, koska bottia ei tarvitse yhdistää suoraan ASP.NET MVC -projektiin, vaan sitä ajetaan itsenäisenä sovelluksena. Bottia varten perustettiin myös tietokanta. Botti tallentaa viestit, keskusteluryhmät, tiedostot ja käyttäjät edellä mainittuun tietokantaan.

Projektissa käytetty relaatiotietokanta alustettiin Microsoft SQL Server Management Studiolla (SSMS) ja kantaan lisättiin neljä taulua: TelegramChatGroup, TelegramFile, TelegramMessage ja TelegramUser. Kuviossa 5 on SQL-kysely, jolla TelegramFile-taulu lisättiin tietokantaan.

```
CREATE TABLE [dbo].[TelegramFile1](
    [TelegramFileID] [int] NOT NULL,
    [TelegramMesssageID] [int] NOT NULL,
    [Filename] [nvarchar](256) NOT NULL,
    [FilePath] [nvarchar](max) NOT NULL,
 CONSTRAINT [PK_TelegramFile] PRIMARY KEY CLUSTERED
(
    [TelegramFileID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

Kuvio 5. TelegramFile-taulun lisääminen tietokantaan.

Tietokannan taulujen TelegramMesssage, TelegramChatGroup jaTelegramUser väliset yhteydet on määritetty taulujen avainkenttien avulla. Taulujen relaatiot on kuvattu kuviossa 6.



Kuvio 6. Tietokannan luokkakaavio

Botti-sovellus. Botti-sovellus sisältää kolme pääelementtiä (kooditiedostoa): Program, TelegramBot ja Database. Program-koodi sisältää pelkästään botin alustusmuuttujat sekä alustusmetodin kutsun. Botti vaatii toimiakseen vain yhden yksilöllisen merkkijonon: accessToken, joka syötetään TelegramBot-olioon parametrinä. Tämän jälkeen botti käynnistetään olion omalla StartListeningToGroups-metodilla (kuvio 7).

```
class Program
{
    static void Main(string[] args)
    {
        string accessToken = "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
        TelegramBot bot = new TelegramBot(accessToken);
        bot.StartListenToGroups();
    }

}
```

Kuvio 7. Botin alustus

Botti-sovelluksen TelegramBot-koodi käsittelee keskusteluryhmiin tulleet viestit ja erittelee ne viestityypin mukaan, esimerkiksi: kuvaviestit ja videoviestit. Samalla mahdolliset viestien sisältämät tiedostot ladataan projektin kansioon ja tiedostojen polut tallennetaan tietokantaan. Kuviossa 8 on esimerkki tavasta, jolla viestejä jaotellaan viestityypin mukaan. Koodi hakee viestistä tiedostotyypin, formatoi siitä tiedostopäätteen ja tallentaa tiedoston oikealla nimellä (ryhmän nimi, tiedoston yksilöllinen numero ja tiedostopääte esimerkiksi .mp3 tai .jpg). Lopuksi viestin tiedot tallennetaan tietokantaan.

```
//Dokumentit
else if (message.Type == Telegram.Bot.Types.Enums.MessageType.DocumentMessage)
{
    string type = message.Document.MimeType;
    string[] splitss = type.Split('/');
    string format = splitss[1];
    if (message.Document == null)
    {
        System.Diagnostics.Debug.WriteLine("File missing.");
        return;
    }

    System.Diagnostics.Debug.WriteLine("File downloaded");
    string filepath = message.Chat.Title + message.MessageId + "."+format;
    DownloadFile(message.Document.FileId, "C:/tmp/telegram" + filepath);
    this.DataHandler.InsertPhoto(message, filepath, out exceptionMessage);
}
```

Kuvio 8. Dokumenttiviestin käsittely ja tallennus

Tiedostoja varten TelegramBot-koodiin on lisätty DownloadFile-metodi (kuvio 9), jonka avulla keskusteluryhmiin lähetetyt tiedostot ladataan projektin kansioon.

DownloadFile-metodille annetaan tiedoston yksilöllinen ID-numero sekä polku, johon tiedosto tallennetaan.

```
private async void DownloadFile(string fileId, string path)
{
    try
    {
        var file = await this.client.GetFileAsync(fileId);

        using (var saveFileStream = new System.IO.FileStream(path, System.IO.FileMode.Create))
        {
            await file.FileStream.CopyToAsync(saveFileStream);
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine("Error downloading: " + ex.Message);
    }
}
```

Kuvio 9. Tiedostojen tallennus.

Telegram-botti ohjelman kolmas osa eli Database vastaa keskusteluryhmien tietojen tallentamisesta tietokantaan. Koodi sisältää tarkistusmetodeja, joiden avulla varmistetaan, että tiedot ovat oikein ja tietokannan taulut ovat eheät. Tarkistusmetodien lisäksi koodi sisältää myös erilaisia insert-metodeja, joilla tiedot tallennetaan tietokantaan. Kuviossa 10 on esimerkkinä tarkistuskysely, jossa ohjelma tekee SQL-kyselyn tietokantaan ja tarkistaa, onko viesti olemassa.

```
private bool? CheckMessageExist(Message message, out string exceptionMessage)
{
    exceptionMessage = string.Empty;

    string sql = "Select count(1) as value from TelegramMesssage where TelegramMesssageID = @param1 and TelegramUserID = @param2 and TelegramChatGroupID = @param3;";
    using (SqlConnection connection = new SqlConnection(this.ConnectionString))
    using (SqlCommand command = new SqlCommand(sql))
    {
        command.Connection = connection;
        command.CommandType = System.Data.CommandType.Text;

        command.Parameters.Add("@param1", System.Data.SqlDbType.Int).Value = message.MessageId;
        command.Parameters.Add("@param2", System.Data.SqlDbType.Int).Value = message.From.Id;
        command.Parameters.Add("@param3", System.Data.SqlDbType.BigInt).Value = message.Chat.Id;

        try
        {
            connection.Open();
            using (SqlDataReader reader = command.ExecuteReader())
            {
                reader.Read();
                int result = (int)reader["value"];
                if (result > 0)
                {
                    return true;
                }
                else
                {
                    return false;
                }
            }
        }
        catch (Exception ex)
        {
            exceptionMessage = ex.ToString();
            return null;
        }
        finally
        {
            connection.Close();
        }
    }
}
```

Kuvio 10. Tarkistuskysely tietokantaan

Tietokantaosion insert-metodit tallentavat parametreinä syötetyt tiedot tietokannan tauluihin. Aina kun keskusteluryhmään lähetetään viesti, botti tallentaa viestin tekstiosan TelegramMessage-tauluun. Jos viesti sisältää kuvia, videoita tai muita tiedostoja, ne ladataan ohjelman kansioon ja tiedostopolku tallennetaan TelegramFile-tauluun. Tiedostoviestit käsitellään siten, että viestitekstin tilalle sijoitetaan tiedostomuotoa kuvaava teksti, jota käytetään projektin MVC-osiossa tunnistamaan tiedostoja sisältävät viestit. Kuviossa 11 on esimerkki insert-metodista, jossa viesti tallennetaan tietokantaan.

```
private bool InsertMessage(Message message, out string exceptionMessage)
{
    exceptionMessage = string.Empty;

    string sqlInsert = "Insert into TelegramMesssage(TelegramMesssageID,TelegramUserID, TelegramChatGroupID, Message, Date) VALUES(@param1,@param2,@param3,@param4,@param5)";

    try
    {
        using (SqlConnection connection = new SqlConnection(this.ConnectionString))
        using (SqlCommand command = new SqlCommand(sqlInsert))
        {
            command.Connection = connection;
            command.CommandType = System.Data.CommandType.Text;

            connection.Open();


            command.Parameters.Add("@param1", System.Data.SqlDbType.Int).Value = message.MessageId;
            command.Parameters.Add("@param2", System.Data.SqlDbType.Int).Value = message.From.Id;
            command.Parameters.Add("@param3", System.Data.SqlDbType.BigInt).Value = message.Chat.Id;

            if (message.Photo != null && message.Text==null)
            {
                if (message.Caption!=null)
                {
                    command.Parameters.Add("@param4", System.Data.SqlDbType.NVarChar, -1).Value = "_TelegramPhoto_ " + message.Caption;
                }
                else
                {
                    command.Parameters.Add("@param4", System.Data.SqlDbType.NVarChar, -1).Value = "_TelegramPhoto_";
                }

            }
```

Kuvio 11. Viestien tallennus tietokantaan.

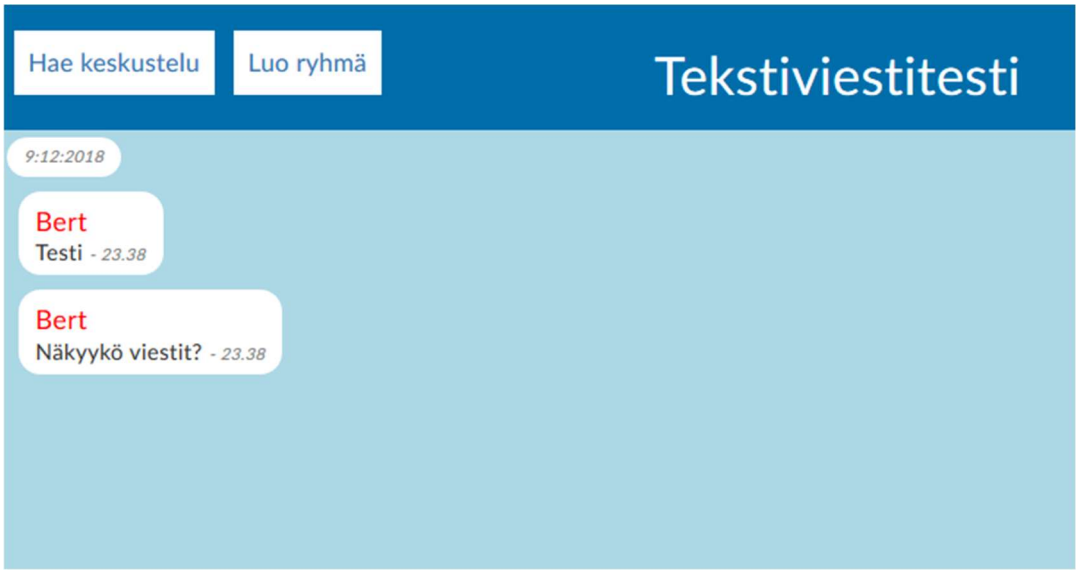
5.4 MVC-käyttöliittymä

Projektin viimeinen vaihe oli käyttöliittymän kehittäminen käyttäen ASP.NET MVC -tekniikkaa. Käyttöliittymässä on kolme päänäkymää: keskustelujen hakunäkymä, viestiketjun lukunäkymä ja keskusteluryhmän luontinäkymä.

5.4.1 Keskusteluryhmien hakunäkymä ja viestiketjun lukunäkymä

Viestien hakeminen toteutettiin lisäämällä projektiin kaksi MVC-näkymää. Ensimmäisessä näkymässä (kuvio 12) ohjelma hakee tietokannasta kaikki tallennetut keskusteluryhmät ja lisää ne näkymässä olevaan listaan. Kun käyttäjä valitsee listasta keskusteluryhmän, ohjelmisto hakee sen ryhmän koko viestihistorian ja näyttää sen käyttäjälle toisessa näkymässä (kuvio 13)

Kuvio 12. Viestien hakunäkymä



Kuvio 13. Viesti-ikkuna

Keskusteluryhmät tuodaan näkymään Search-metodilla (kuvio 14), joka hakee keskusteluryhmät GetGroup-metodilla (kuvio 15), joka lähettää tietokantaan SQL-kyselyn. Kyselyn tuloksena saadaan kaikkien keskusteluryhmien nimet sekä yksilölliset ID-numerot, jotka puretaan listaan While()-lauseella. Lopuksi lista palautetaan ja siirretään näkymään. Näkymässä lista puretaan Foreach()-lauseella suoraan HTML-syntaksiin (kuvio 16).

```
public ActionResult Search()
{

    this.DataHandler = new Database(@"Server=localhost; database=master; Integrated Security=true;");
    List<Group> list = new List<Group>(DataHandler.GetGroup());
    ViewData["MyProduct"] = list;


    return View();
}
```

Kuvio 14. Keskusteluryhmien hakumetodi.

```
public List<Group> GetGroup()
{
    List<Group> lista = new List<Group>();
    string sql = "Select TelegramChatGroupID as ID, Title as title from TelegramChatGroup";
    using (SqlConnection connection = new SqlConnection(this.ConnectionString))
    using (SqlCommand command = new SqlCommand(sql))
    {
        command.Connection = connection;
        command.CommandType = System.Data.CommandType.Text;
        try
        {
            connection.Open();
            using (SqlDataReader reader = command.ExecuteReader())
            {
                if (reader.HasRows)
                {
                    while (reader.Read())
                    {
                        Group group = new Group();
                        group.GroupID = (long)reader["ID"];
                        group.GroupName = (string)reader["title"];
                        lista.Add(group);


                    }
                }
            }
        }
        catch (Exception ex)
        {
            throw;
        }
        finally
        {
            connection.Close();
        }
        return lista;
    }

}
```

Kuvio 15. Keskusteluryhmän haku

```
<div class="messagebox1">
    @foreach (var item in prod)
    {
        <ul>
            <li>
                <a style="font-family:sans-serif; font-size:200%;"
                   href="@Url.Action("Message", null,
                        new {id= item.GroupID,groupname=item.GroupName,
                            area = string.Empty, controller = "Search" }, Request.Url.Scheme)">
                    <div class="groupbox1">
                        <span>@item.GroupName</span>

                    </div>

                </a>
            </li>
        </ul>

    }

</div>
```

Kuvio 16. Ryhmälistan purkaminen näkymään.

Kun käyttäjä valitsee listasta keskusteluryhmän, ohjelmisto hakee keskusteluryhmän viestit tietokannasta. Jokaisella keskusteluryhmällä on yksilöllinen ID-numero. ID-numero syötetään Message-metodille (kuvio 17), joka tekee SQL-kyselyn tietokantaan, ja siirtää kyselyn tulokset listaan, joka annetaan viestinäkymälle. Viestinäkymä jäsentelee tiedot oikeaan paikkaan. Ennen listan siirtämistä näkymälle, tiedostoja sisältäviin viesteihin liitetään tiedostopolut ja nimet, jotta näkymä voi näyttää ne käyttäjälle.

```
public ActionResult Message(long id,string groupname)
{

    this.DataHandler = new Database(@"Server=localhost; database=master; Integrated Security=true;");
    List<Viesti> list = new List<Viesti>(DataHandler.GetMessages(id));
    System.DateTime _Now = DateTime.Now;
    foreach (var item in list)
    {
        if (item.message1.StartsWith("_TelergramPhoto_"))
        {
            Kuva kuva1 = new Kuva();
            kuva1 = DataHandler.GetPhoto(item.ID);
            item.PhotoPath = kuva1.FilePath;
            item.PhotoName = kuva1.Filename;

        }
        else if (item.message1.StartsWith("_TelergramVideo_"))
        {
            Kuva kuva1 = new Kuva();
            kuva1 = DataHandler.GetPhoto(item.ID);
            item.PhotoPath = kuva1.FilePath;
            item.PhotoName = kuva1.Filename;

        }
        else if (item.message1.StartsWith("_TelergramVoice_"))
        {
            Kuva kuva1 = new Kuva();
            kuva1 = DataHandler.GetPhoto(item.ID);
            item.PhotoPath = kuva1.FilePath;
            item.PhotoName = kuva1.Filename;

        }
        else if (item.message1.StartsWith("_TelergramDocument_"))
        {
            Kuva kuva1 = new Kuva();
            kuva1 = DataHandler.GetPhoto(item.ID);
            item.PhotoPath = kuva1.FilePath;
            item.PhotoName = kuva1.Filename;

        }
    }
    ViewData["MessageList"] = list;
    ViewData["Groupname"] = groupname;

    return View(list);
}
```

Kuvio 17. Viestilistan käsittely.

Kun viestlista on siirretty näkymälle, lista puretaan foreach-lauseella. Listan jokainen viesti käydään yksitellen läpi. Samalla tarkastetaan, onko viestissä tiedostoja. Viesti sekä mahdolliset tiedostot sijoitetaan näkymän HTML-syntaksiin

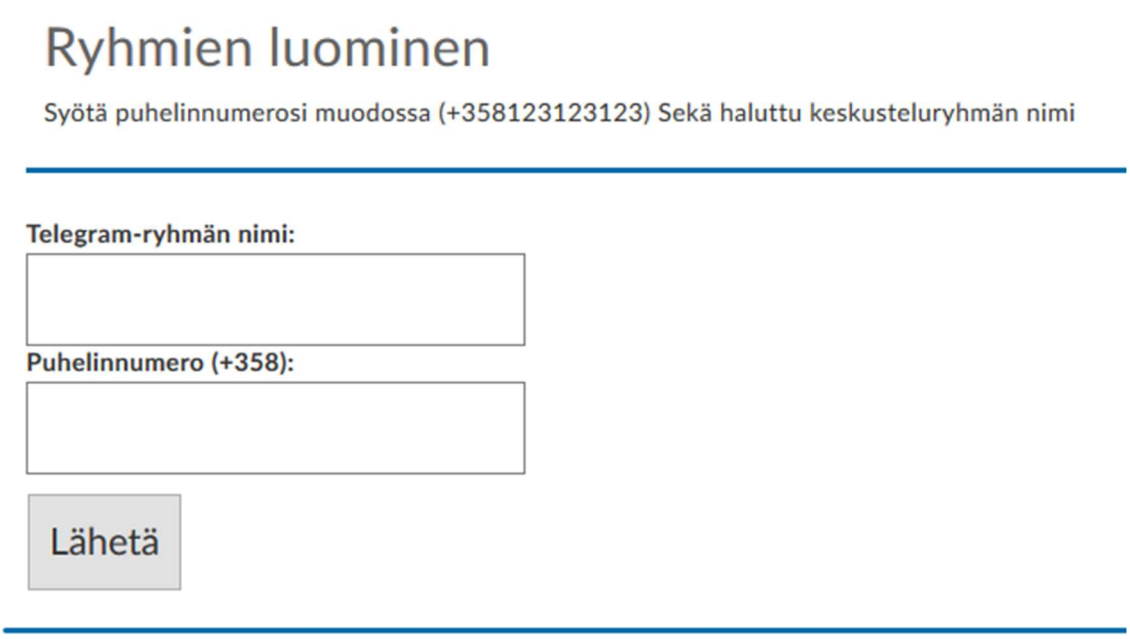
(kuvio 18). Viestiobjekti sisältää myös viestin päivämäärän ja kellonajan, ne myös lisätään viestinäkymään. Tavoitteena oli luoda Telegramin käyttöliittymää muistuttava näkymä.

```
{
    string format = "HH:mm";
    string kello = " " + item.Date.ToString(format);
    string datepvm = item.Date.Day + ":" + item.Date.Month + ":" + item.Date.Year;
    <div class="viestiblock">

        <h4 class="messageusername" style="color:@vari[index]">@item.Username</h4>
        <span class="messagetext">@item.message1  </span>

        <span class="messageTime">-@kello</span>

    </div>
    <br />
}
```

Kuvio 18. Viestilistan purkaminen Foreach-lauseessa

5.4.2 Keskusteluryhmän perustaminen

Keskusteluryhmien lisäämistä varten rakennettiin yksinkertainen lomake (kuvio 19), johon käyttäjä syöttää puhelinnumeronsa, sekä halutun keskusteluryhmän nimen. Ennen ryhmän perustamista sovellus tarkistaa, onko sovellus todennettu. Jos sovellusta ei ole todennettu, ohjataan käyttäjä toiseen lomakkeeseen, johon käyttäjä sijoittaa puhelimeensa toimitetun varmistuskoodin, jonka jälkeen Telegram-keskusteluryhmä perustetaan.



Kuvio 19. Keskusteluryhmälomake.

Keskusteluryhmien käsittely ohjelmoitiin pääosin käyttäen asynkronisia metodeja, joiden ansiosta autentikointi ja ryhmien lisääminen tapahtuu nopeasti, eikä sovellus jää odottamaan eri vaiheiden suorittamista. Sovelluskoodi on kuviossa 20.

```
[AcceptVerbs(HttpVerbs.Post)]
public async System.Threading.Tasks.Task<ActionResult> CreateTelegramGroupAsync(FormCollection Form)
{
    string groupName = Form["Groupname"];
    string Phone = Form["PhoneNumber"];
    if (Phone.StartsWith("+") == false || groupName=="")
    {
        return View("~/Views/Create/Error.cshtml");
    }
    Utility.TelegramInterface client = null;
    bool IsAuthorized = false;

    if ( groupName != null && groupName.Length > 5)
    {
        client = new Utility.TelegramInterface();

        try
        {
            IsAuthorized = await client.Connect();


        }
        catch (Exception ex)
        {
            IsAuthorized = false;

        }

        if (!IsAuthorized)
        {
            var returnCodeObject = await client.AskAuthCode(Phone);
            ViewData["Phonenumber"] = Phone;
            ViewData["Groupname"] = groupName;
            ViewData["Phonehash"] = returnCodeObject.PhoneCodeHash;

            return View("~/Views/Create/AskAuth.cshtml");

        }
        else
        {

            List <Utility.TelegramUser> lista= new List<Utility.TelegramUser>();
            lista =await client.GetList(Phone);
            if (lista.Count<=1)
            {
                return View("~/Views/Create/Error.cshtml");

            }
            await client.CreateGroup(groupName, lista);
            return View("~/Views/Home/MakeAuth.cshtml");
        }
    }
```

Kuvio 20. Telegram-ryhmän luominen ja sovelluksen todennus.

6 SOVELLUKSEN TESTAUS

Projektin lopuksi sovellusta testattiin ja testituloksista tehtiin raportti JAPO Palvelut Oy:lle. Sovellusta testattiin eri selaimilla ja testeistä kävi ilmi, että sovellukseen on lisättävä virhesivu, jos käyttäjä syöttää esimerkiksi virheellisen puhelinnumeron. Kuviossa 21 on esimerkki virheenkäsittelytilanteesta, jossa sovellus hakee puhelinnumerolla listan käyttäjän kontakteista. Puhelinnumeron ollessa väärä listan pituudeksi tulee 0, jonka jälkeen If-lause palauttaa käyttäjälle virhesivunäkymän.

```
lista =await client.GetList(Phone);
if (lista.Count<=1)
{
    return View("~/Views/Create/Error.cshtml");

}
```

Kuvio 21. Virhesivu

Viestien hakeminen testattiin kolmella erillisellä testikeskusteluryhmällä: tekstiviestiryhmä, kuvaviestiryhmä ja videoviestiryhmä. Ominaisuudet testattiin ryhmien nimien mukaisesti, esimerkiksi kuvaviestiryhmässä testattiin kuvaviestien toimivuutta. Jokainen testiryhmä toimi testaustilanteessa ongelmitta. Koko testiraportti on liitteenä työn lopussa.

7 POHDINTA

Työn valmistuttua tarkasteltiin, onko projektin alussa asetetut tavoitteet saavutettu. Sovelluskehityksessä ei tullut suuria ongelmia ja lopputulos täytti asiakkaan vaatimukset. Sovelluksen käyttöliittymästä tuli Telegramia muistuttava ja toiminnallisuus oli odotuksia vastaava.

Työn tavoitteina oli myös kirjoittajan taitojen kehittäminen. Lisätaitoja haettiin rajapintojen ja tietokantojen kanssa työskentelemiseen, koska työn alkuvaiheessa näistä aiheista ei ollut paljon kokemusta. Myös MVC-sovelluksien kehittämiseen haluttiin lisäkokemusta. Kehitystä tapahtui varsinkin MVC-sovelluskehityksessä ja tietokantojen kanssa työskentelemisessä, mikä tulee helpottamaan tulevaisuuden projektien etenemistä. Myös työn kehittävät tavoitteet siis saavutettiin.

LÄHTEET

DataBusiness. Ei päiväystä. Rajapinnat. [www-dokumentti]. DataBusiness. [Viitattu 8.1.2019]. Saatavissa: https://www.databusiness.fi/fi/avoindata/avoimet-rajapinnat/

DeveloperIn. Ei päiväystä. Components of .Net Framework. [www-dokumentti]. Developerin.Net. [Viitattu 11.2.2019]. Saatavissa: http://www.developerin.net/a/39-Intro-to-.Net-FrameWork/23-Components-of-.Net-Framework

Hill, S. 2019. The best text messaging apps for Android and iOS. [www-dokumentti]. Digital Trends. [Viitattu 19.2.2019]. Saatavissa: https://www.digitaltrends.com/mobile/best-text-messaging-apps/

Hovi, A. 2008. SQL-opas. 5. painos. Jyväskylä: Docendo.

Isokangas, A. & Vassinen, R. 2010. Digitaalinen jalanjälki. Helsinki: Talentum.

Jyväskylän yliopisto. 2009. Pikaviestin. [www-dokumentti]. Jyväskylän yliopisto Digipalvelut. [Viitattu 19.2.2019]. Saatavissa: https://www.jyu.fi/digipalvelut/fi/ohjeet/sanasto/pikaviestin

Lahtonen, T. 2016. Model-view-controller (MVC) -arkkitehtuuri. [www-dokumentti]. Jyväskylän yliopiston informaatioteknologian tiedekunta. [Viitattu 5.1.2019]. Saatavissa: http://appro.mit.jyu.fi/tiea2080/luennot/mvc/

Laitila, T. 2017. Pikaviesti toimistossa. [verkkoartikkeli]. Talouselämä. [Viitattu 19.2.2019]. Saatavissa: https://www.talouselama.fi/uutiset/pikaviesti-toimistossa/27b681f1-ea56-380c-8621-5a4bb2338e88

Liberty, J. 2005. Programming C#. 4.painos. Sebastopol: O'Reilly Media, Inc.

Microsoft. 2019a. What is .NET?. [www-dokumentti]. Microsoft Corporation. [Viitattu 5.1.2019]. Saatavissa: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet

Microsoft. 2019b. ASP.NET overview. [www-dokumentti]. Microsoft Corporation. [Viitattu 5.1.2019]. Saatavissa: https://docs.microsoft.com/fi-fi/aspnet/overview

Microsoft. 2018. About Azure Bot Service. [www-dokumentti]. Microsoft Corporation. [Viitattu 8.1.2019]. Saatavissa: https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overview-introduction?view=azure-bot-service-4.0

Snapchat. 2018. What is snapchat?. [www-dokumentti]. Snap Inc.[Viitattu 18.12.2018]. Saatavissa: https://whatis.snapchat.com/

Symantec. Ei päiväystä. What are bots?. [www-dokumentti]. Symantec corporation. [Viitattu 19.2.2019]. Saatavissa: https://us.norton.com/internetsecurity-malware-what-are-bots.html

Telegram. Ei päiväystä. Bots: An introduction for developers. [www-dokumentti]. Telegram Messenger LLC. [Viitattu 5.1.2019]. Saatavissa: https://core.telegram.org/bots

Telegram. 2019. Telegram FAQ. [www-dokumentti]. Telegram Messenger LLC. [Viitattu 5.1.2019]. Saatavissa: https://telegram.org/faq

Telegram. Ei Päiväystä. Telegram FAQ. [www-dokumentti]. Telegram Messenger LLC. [Viitattu 9.1.2019]. Saatavissa: https://core.telegram.org/api

TSK. 2007. Internetpuhelusanasto. [www-dokumentti]. Sanastokeskus TSK ry. [Viitattu 18.12.2018]. Saatavissa: http://www.tsk.fi/tiedostot/pdf/internetpuhelusanasto.pdf

WhatsApp. 2018. Tietoja WhatsAppista. [www-dokumentti]. WhatsApp Inc. [Viitattu 18.12.2018]. Saatavissa: https://www.whatsapp.com/about/

WhatsApp. 2019. WhatsApp Security. [www-dokumentti]. WhatsApp Inc. [Viitattu 11.12.2019]. Saatavissa: https://www.whatsapp.com/security/

LIITTEET

Liite 1. Telegram-viestintäsovelluksen testiraportti

Liite 1. Telegram-viestintäsovelluksen testiraportti

1 Testialustat

Ohjelmisto on testattu neljällä eri selaimella (Mozilla Firefox [versio 63.0.3], Internet Explorer Firefox [versio 11.406.17134.0], Google Chrome [versio 70.0.3538.110], Microsoft Edge[versio 42.17134.1.0]). Ohjelmisto toimi jokaisella selaimella ongelmitta viimeisessä testivaiheessa.

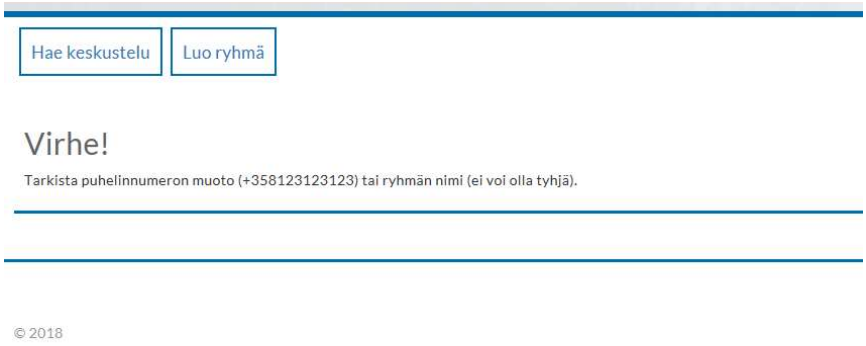
Testattu vain yhdellä tietokoneella ja käyttöjärjestelmällä (Windows 10). Testipuhelimena toimi Applen iPhone 8-puhelin, jossa käyttöjärjestelmänä iOS 12.0.0. Telegrammin vastaavaa Android-sovellusta ei ole testattu.

2 Virhetilanteet

Virhetilanteen sattuessa, esimerkiksi väärä puhelinnumero, ohjelma ohjaantui oletusvirhesivulle. Tämä korjattiin lisäämällä virhenäkymä, joka ilmoittaa tapahtuneesta virheestä.

2.1 Ryhmän luominen

Ryhmän luonnissa tapahtuvien ongelmien varalle on rakennettu virhe-sivu (kuvio 1) , jolle käyttäjä ohjataan, jos puhelinnumero kirjoitetaan väärässä muodossa tai jos puhelinnumeroa ei ole olemassa. Sama virheilmoitus toistuu, jos ryhmän nimi jätetään tyhjäksi. Luodessa ryhmää syötettävän puhelinnumeron pitää olla sama, kuin puhelinnumeron jolla autentikointi on tehty.



Kuvio 1. Virhe-sivu

2.2 Kuvaviestit ja videot

Kun ryhmään lähetetään kuva, botti tallentaa viestiin tunnuksen, jonka avulla kuvat tunnistetaan tietokannasta luettaessa. (kuvio 2)

```
if (message.Photo != null && message.Text==null)
{
    if (message.Caption!=null)
    {
        command.Parameters.Add("@param4", System.Data.SqlDbType.NVarChar, -1).Value = "_TelergramPhoto_ " + message.Caption;
    }
    else
    {
        command.Parameters.Add("@param4", System.Data.SqlDbType.NVarChar, -1).Value = "_TelergramPhoto_";
    }

}
```

Kuvio 2. Kuvaviestin käsittely

Samalla tavalla tallennetaan myös videot.

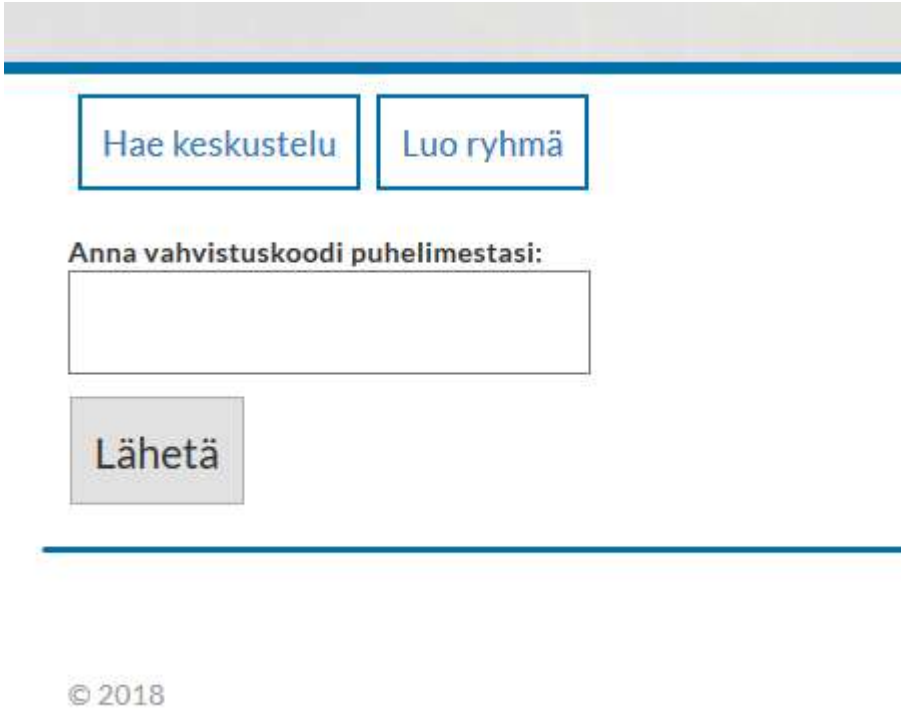
Sivun videosoitin toistaa vain .mp4-tiedostoja, joita Telegram lähettää sovelluksesta oletuksena. Jos videoformaatti ei ole mp4, selaimen videosoitin ei toista sitä.

2.3 Botti

Botti on tehty C#-ohjelmointikielellä. Bottia on testattu Telegram-sovelluksella (versio 5.0.17). Botti on toiminut yhdellä järjestelmällä testattuna ilman virheitä.

2.4 Autentikointi

Telegram API-autentikointi on testattu Applen iPhone 8-puhelimella ja ongelmia ei ole ilmennyt. Ohjelma kysyy autentikointikoodia samalla koneella vain ensimmäisellä käyttökerralla. Alla olevassa kuviossa (kuvio 3) on sovelluksen autentikointilomake johon puhelinkoodi syötetään.



Kuvio 3. Autentikointilomake

3 Testiryhmät

Testiryhmä koostui kolmesta ryhmästä: tekstiviestiryhmä, kuvaviestiryhmä ja videoryhmä. Jokaisessa ryhmässä oli läsnä botti, joka tallentaa keskustelut ja kaksi muuta henkilöä.

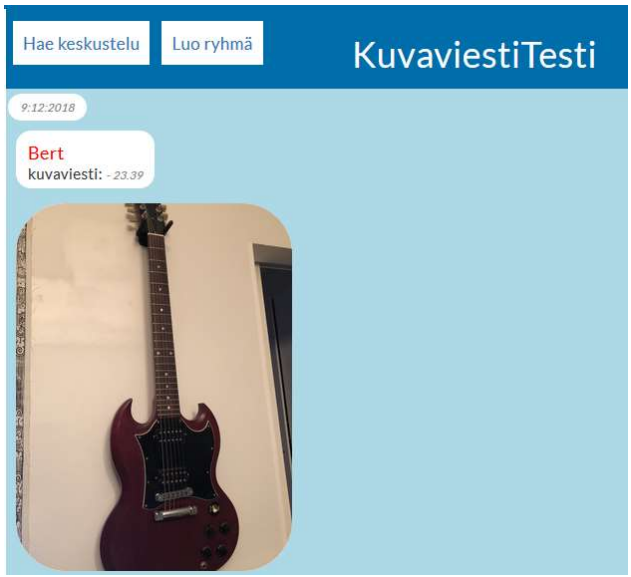
3.1 Tekstiviestit

Tekstiviestiryhmä oli ensimmäinen testiryhmä (kuvio 4). Ryhmän tarkoituksena oli saada viestintäsovelluksen runko testattua ennen sovelluksen laajentamista.

Kuvio 4. Tekstiviestitesti

3.2 Kuvaviestit

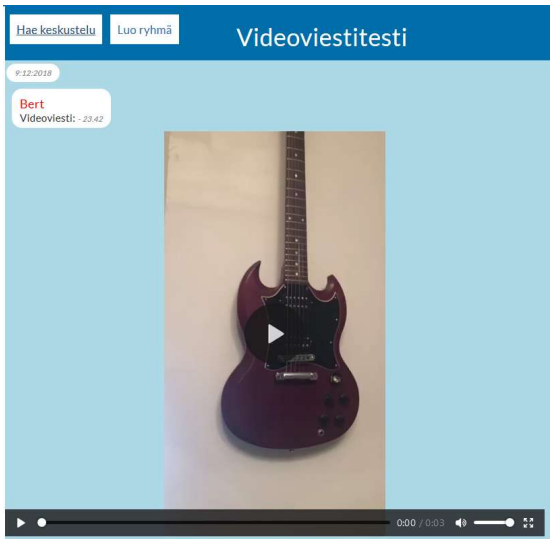
Kuvaviestiryhmä oli toinen testiryhmä. Kuvaviestiryhmässä testattiin kuvien lähetyksen toimivuutta. Alla olevassa kuviossa (kuvio 5) on sovelluksen viesti-näkymä johon keskusteluryhmän viestit tulevat näkyviin.



Kuvio 5. Kuvaviestitesti

3.3 Videoviestit

Videotestiryhmä oli kolmas testiryhmä. Videoryhmässä testattiin videon lähettämistä puhelimesta selaimelle (kuvio 6) . Videosoitin toistaa mp4-tiedostoja ongelmitta.



Kuvio 6. Videoviestitesti