**Windows-ohjelmointi: Jäsenrekisteri**

**TTOS0300**

Miika Avela H4211

Kurssin harjoitustyö

Huhtikuu 2017

Tekniikan ja liikenteen ala

Insinööri (AMK), ohjelmistotekniikan tutkinto-ohjelma

**Sisältö**

[1 Yleiskuvaus 1](#_Toc480931251)

[1.1 Projektin toimeksianto 1](#_Toc480931252)

[1.2 Tekijä 1](#_Toc480931253)

[1.3 Projektin materiaali 1](#_Toc480931254)

[2 Tavoite 2](#_Toc480931255)

[3 Käytetyt teknologiat ja toteutus 3](#_Toc480931256)

[3.1 Projektin suunnittelu 3](#_Toc480931257)

[3.2 Tietokantayhteys ja malli 4](#_Toc480931258)

[3.3 WPF ja käyttöliittymä 6](#_Toc480931259)

[3.3.1 Toiminnallisuudet 7](#_Toc480931260)

[3.4 PDF Export 9](#_Toc480931261)

[4 Aikataulu 11](#_Toc480931262)

[5 Ongelmakohdat ja niiden ratkaisut 12](#_Toc480931263)

[6 Itsearviointi 12](#_Toc480931264)

**Kuvat**

[Kuva 1 Ohjelmiston arkkitehtuuri 3](#_Toc480931265)

[Kuva 2 SQL-taulu 4](#_Toc480931266)

[Kuva 3 Luotu entiteetti 5](#_Toc480931267)

[Kuva 4 connectionString 5](#_Toc480931268)

[Kuva 5 Käyttöliittymä 6](#_Toc480931269)

[Kuva 6 Jäsen poistettu 7](#_Toc480931270)

[Kuva 7 Jäsenten haku postinumerolla 7](#_Toc480931271)

[Kuva 8 Jäsenten haku nimen osalla 8](#_Toc480931272)

[Kuva 9 Jäsenmaksua maksamattomien jäsenten haku 8](#_Toc480931273)

[Kuva 10 Nimilista järjestettynä sukunimen mukaan 9](#_Toc480931274)

[Kuva 11 Postitustarrat järjestettynä postinumeron mukaan 10](#_Toc480931275)

[Kuva 12 Tallennukset 11](#_Toc480931276)

# Yleiskuvaus

Toteutuksena tehtiin opintojakson TTOS0300 Käyttöliittymäohjelmointi-kurssin harjoitustyö. Dokumentti sisältää kuvaukset projektin toiminnasta ja käytetyistä tekniikoista.

## Projektin toimeksianto

Harjoitustyön toimeksiannon antoi Käyttöliittymäohjelmointi-kurssin opettaja, Esa Salmikangas. Työn tavoitteena oli perehtyä Windows-ohjelmoinnissa käytettyihin yleisimpiin teknologioihin ja oppia näistä käytännössä. Aiheeksi valikoitui ennalta määritelty jäsenrekisteri, mikä tultiin toteuttamaan C#-kieltä käyttämällä WPF-sovelluksena.

## Tekijä

Harjoitustyön tekijänä toimi Jyväskylän ammattikorkeakoulun, ohjelmistotekniikan kolmannen vuoden opiskelija Miika Avela (H4211).

Kurssin toimeksiantoa suorittamaan lähtiessä tekijän C# ja WPF-kirjaston taidollinen lähtötaso perustui aiemmin käytyyn IIO13200 NET-ohjelmointi -kurssiin. Kokemusta siis C#:sta oli hieman, mutta WPF tuli uutena aihealueena.

## Projektin materiaali

Projektityön tarkennukseksi annettiin JYU:n *Kerhon jäsenrekisteri* tehtävä.

Tehtävä: <https://tim.jyu.fi/view/2#kerhon-jäsenrekisteri>

Projektin tuotos on katselmoitavissa Githubissa.

Github: <https://github.com/Monni/TTOS0300-Kayttoliittymien-ohjelmointi>

Projektin dokumentaatio löytyy myös Githubista:

Github: <https://github.com/Monni/TTOS0300-Kayttoliittymien-ohjelmointi/tree/master/Docs>

# Tavoite

Kurssin harjoitustyönä oli tarkoitus toteuttaa jonkin kerhon jäsenrekisteri. Tärkeimmät vaatimukset jäsenrekisteriltä olivat:

* Jäsenten lisääminen
* Jäsenten poistaminen
* Jäsenten hakeminen tietyillä ehdoilla
* Jäsenen tietojen muuttaminen
* Postitustarrojen tulostaminen postinumerojärjestyksessä
* Nimilistan tulostaminen nimijärjestyksessä
* Lista jäsenmaksua maksamattomista jäsenistä

Tämän pohjalta luotiin vaatimukset jäsentiedoille tietokannassa, mitkä olivat:

* ID
* Etunimet
* Sukunimi
* Hetu
* Katuosoite
* Postiosoite
* Postitoimipaikka
* Puhelin
* Maksetun jäsenmaksun suuruus
* Lisätietoja

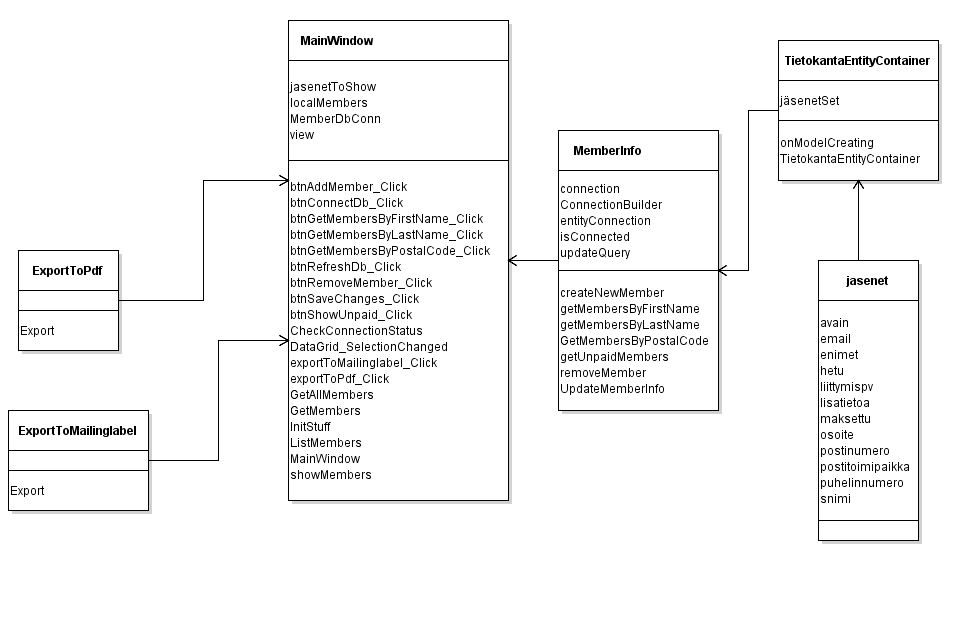
Jäsenrekisterin graafinen ulkoasu valittiin toteutettavaksi Windows Presentation Foundation, eli WPF .NET Framework -kirjastolla, mikä muodostaa uusien Windows-versioiden graafisen rajapinnan. WPF:n pariksi backend-puolelle valittiin kieleksi C#, sillä siitä olen löytänyt selkeästi eniten ohjeita suosionsa vuoksi.

# Käytetyt teknologiat ja toteutus

Kaikki ohjelmointi ja testaus on suoritettu Visual Studio 2017:lla. Projektinhallinta ja -seuranta toteutettiin Github-repositoriolla, mutta suurimmilta osin versionhallinnan virkaa hoiti oman koneen paikalliset kansiot sillä mitään erityisen suurta ei ohjelmiston kehityksessä tarvinnut toteuttaa yhdellä kertaa.

## Projektin suunnittelu

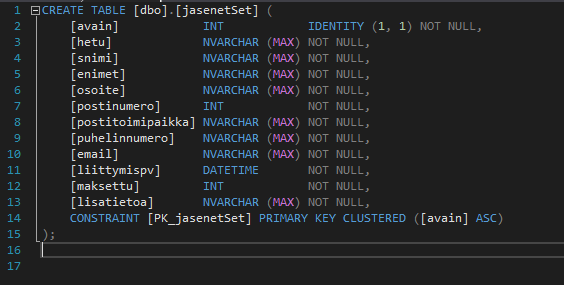
Jäsenrekisterin suunnitteluvaiheessa kirjasin käsin ylös rekisterin tärkeimmät ominaisuudet ja vaatimukset, sekä tein karkean suunnitelman ohjelmiston rakenteesta ja toiminnasta Violet UML Editorilla. Harjoitustyön edetessä myös UML-kaavio tarkentui ja päätyi seuraavan kuvan mukaiseksi.



Kuva 1 Ohjelmiston arkkitehtuuri

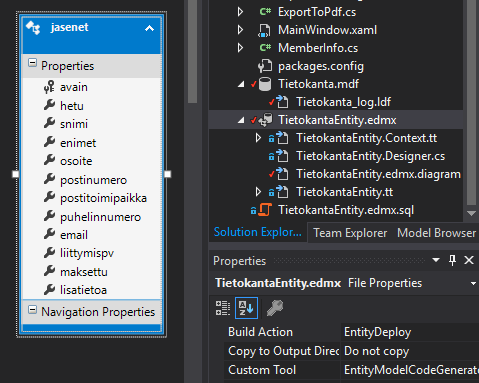
## Tietokantayhteys ja malli

Jäsenrekisterin tietokantana toimi SQL Server, mihin luotiin seuraava taulu.



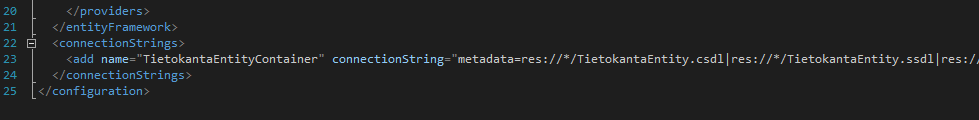
Kuva 2 SQL-taulu

Luodun taulun perusteella Visual Studiolla tehtiin Entity Data Model -tiedosto (.edmx), mikä osasi luoda taululle sopivan entiteetin ja toimia ORM-palveluna näiden välillä.



Kuva 3 Luotu entiteetti

Samalla Visual Studio osasi luoda vaadittavan connectionstringin App.config-tiedostoon käytettävälle tietokantayhteydelle.

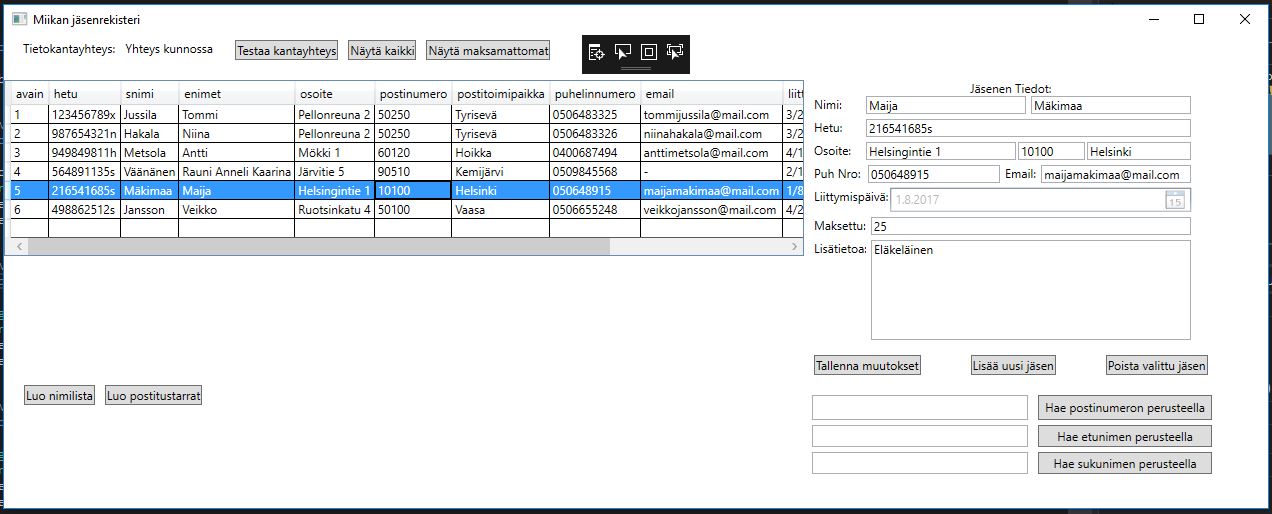


Kuva 4 connectionString

Hyvien oppien mukaisesti ohjelmisto on toteutettu perinteisen MVC-mallin mukaisesti, ja Modelin (jasenet), sekä Viewin väliin on luotu Controller nimeltään MemberInfo.cs. MemberInfon tehtävänä oli kommunikoida tietokannan kanssa ja käyttää tässä apunaan jasenet-mallia, sekä palauttaa kyselyistä palannut data halutulla tavalla Viewille.

## WPF ja käyttöliittymä

Käyttöliittymä on toteutettu WPF-kirjastolla, missä muotoilu ja toiminnallisuuksiin vaadittavat komponentit on kirjoitettu Extensible Application Markup Languagella (XAML). Käyttöliittymän taustalla toimii myös C#:lla kirjoitettu luokka, mikä lasketaan kokonaisuudessaan View-komponentiksi ajateltuna osana MVC-mallia.

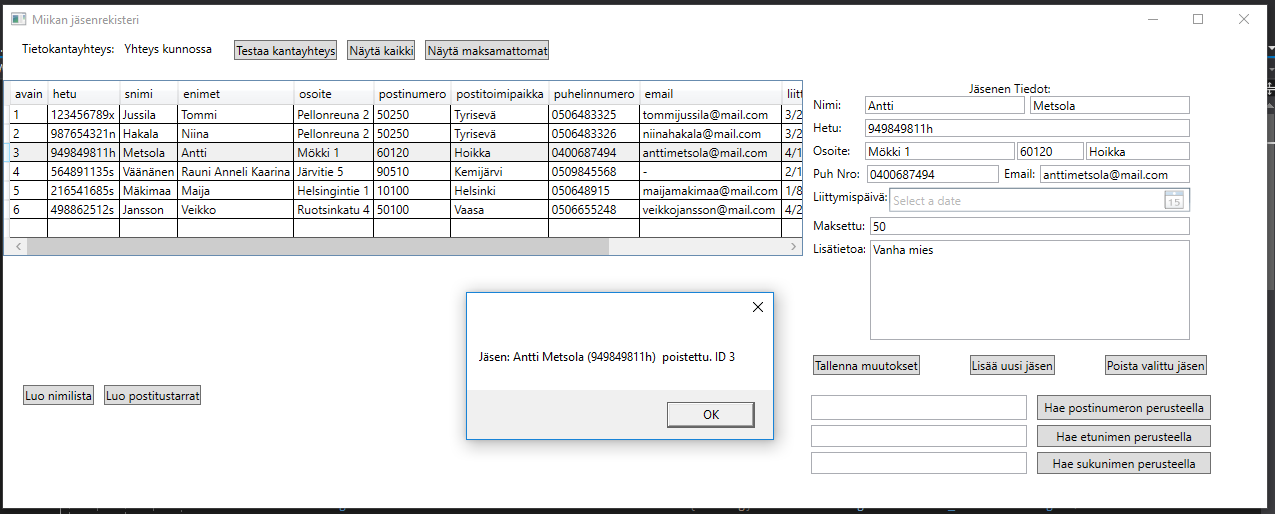


Kuva 5 Käyttöliittymä

View-komponentti ei itsessään osaa kuin ottaa vastaan käyttäjän syötteitä ja välittää niistä pyyntöjä Controllerille, mikä hoitaa varsinaisen kommunikoinnin tietokannan kanssa ja kaiken sovelluslogiikan. View-komponentti osaa välittää myös Controllerilta palaavan syötteen takaisin näytölle loppukäyttäjälle visualisoitavaksi.

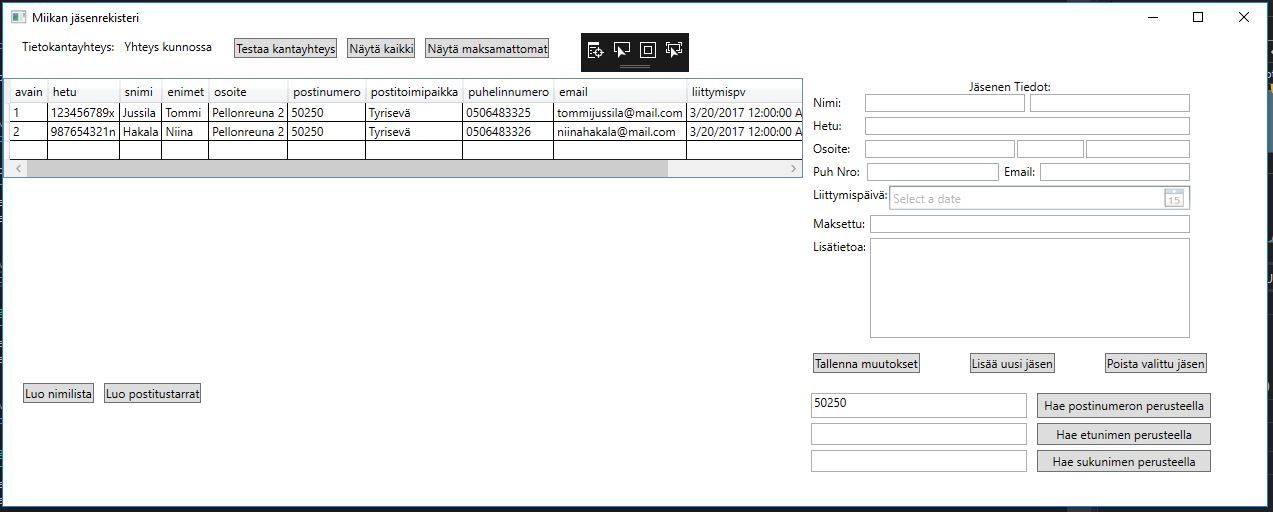
### Toiminnallisuudet

Käyttöliittymässä on mahdollista tarkastella kaikkia jäseniä tietokannasta (Kuva 5). Tämän lisäksi on mahdollista luoda järjestelmään uusia jäseniä, muokata vanhan tietoja, sekä poistaa kerhosta eronneita jäseniä.



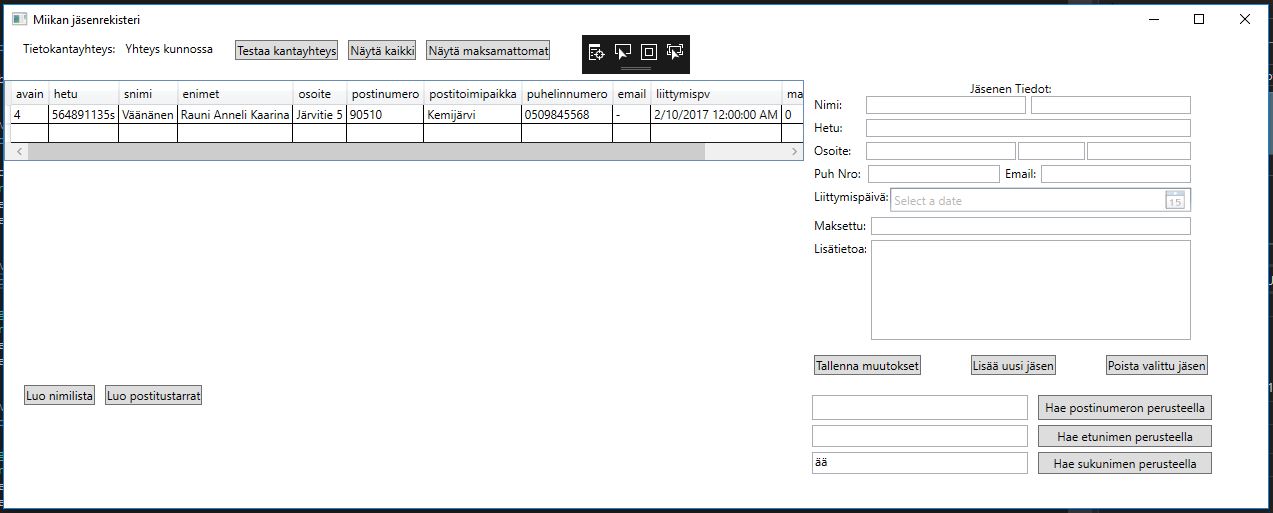
Kuva 6 Jäsen poistettu

Perinteisten CRUD-operaatioiden lisäksi ohjelmisto tukee jäsenten parametrisoituja hakuja, kuten esimerkiksi jäsenten haku postinumeron perusteella



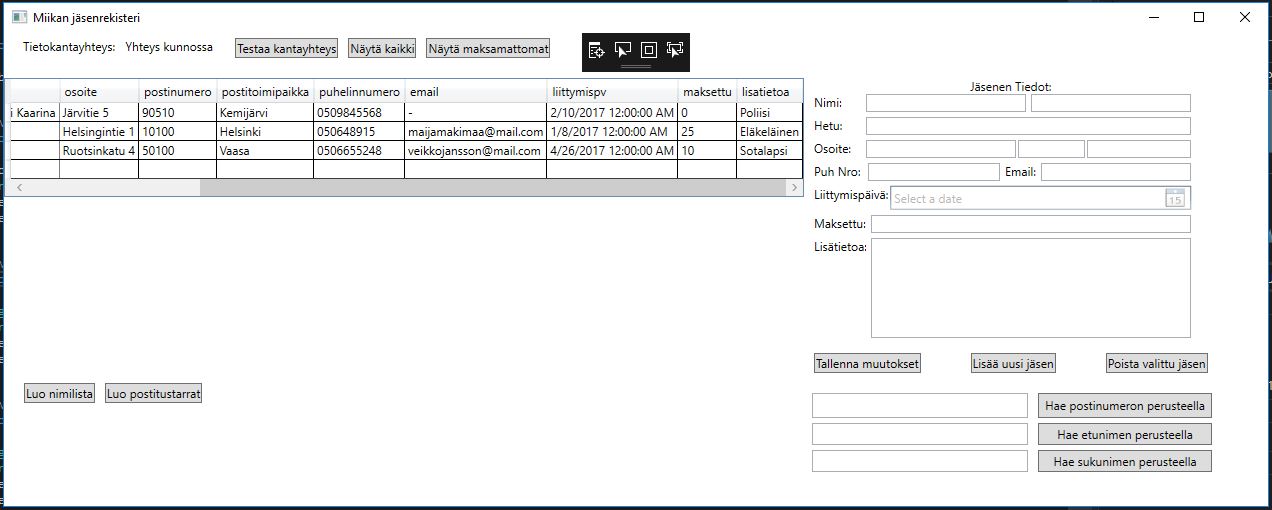
Kuva 7 Jäsenten haku postinumerolla

Lisäksi jäseniä on mahdollista hakea etu- tai sukunimen perusteella joko jäsenen koko nimellä, tai vain nimen osalla.



Kuva 8 Jäsenten haku nimen osalla

Ohjelmistosta on myös mahdollista pyytää lista jäsenmaksua osittain tai kokonaan maksamattomista jäsenistä



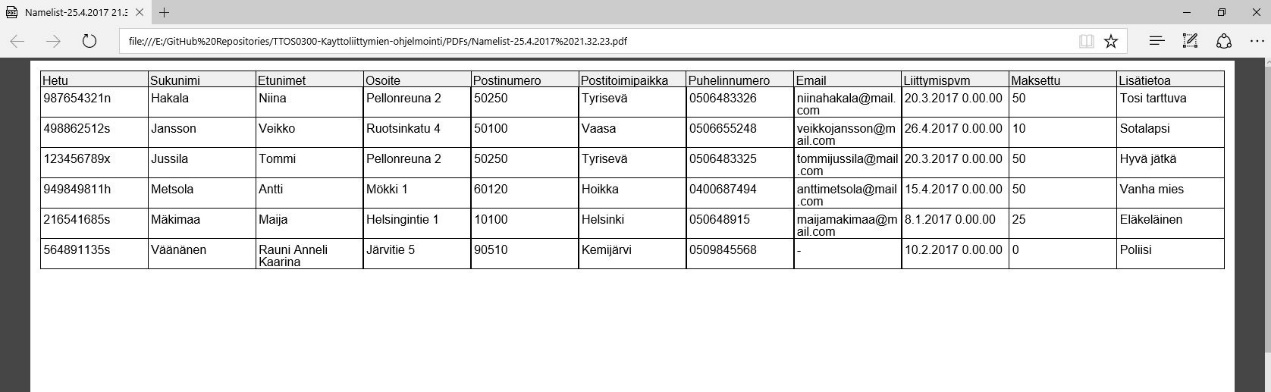
Kuva 9 Jäsenmaksua maksamattomien jäsenten haku

## PDF Export

Harjoitustyön vaatimuksina oli pystyä tulostamaan postitustarrat postinumerojärjestyksessä, sekä nimilistat nimen mukaisessa järjestyksessä.

Käyttöliittymässä voi vapaasti valita Exportattavat listat, esimerkiksi jäsenmaksua maksamattomat jäsenet. Tämän jälkeen käyttäjälle näkyvästä listasta on mahdollista tallentaa PDF:na nimilista yhteystietoineen ja postitustarralista.

Tallentaessa listat nimilistoiksi, järjestyvät ne jäsenen sukunimen mukaan .



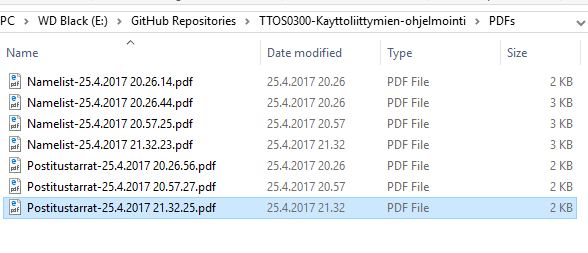
Kuva 10 Nimilista järjestettynä sukunimen mukaan

Listan perusteella luotaessa postitustarrat järjestyvät ne postinumeron mukaan.



Kuva 11 Postitustarrat järjestettynä postinumeron mukaan

PDF:t tallentuvat projektin juuressa sijaitsevaan kansioon ’PDFs’. Mikäli kansiota ei tallennusvaiheessa ole, se luodaan automaattisesti. Tallennuksen nimitietoihin syötetään senhetkinen aika, jolloin vanhempia tallennuksia on mahdollista katselmoida myös jälkikäteen.



Kuva 12 Tallennukset

# Aikataulu

Työtunteja harjoitustyössä kertyi seuraavasti:

Suunnittelu ja ongelmanratkonta: 6h

Ympäristön pystytys: 3h

Tietokannan pystytys ja konfigurointi: 13h

jasenet-entiteetin toteutus: 4h

kommunikaatio tietokantaan ja takaisin: 3h

Controller: 10h

View XAML: 4h

View C#: 4h

PDF Exportit: 7h

Dokumentaatio: 4h

**Yhteensä: 58h**

# Ongelmakohdat ja niiden ratkaisut

Ylivoimaisesti suurimpana ongelmakohtana törmäsin projektin yhteensopivuuteen MySQL-tietokannan kanssa. Neljä päivää yritin yhdistää MySQL:aa Visual Studioon erilaisilla MySQL Installereilla, Connectoreilla ja Visual Studion NuGet-paketeilla. Lopulta saadessani Visual Studion tunnistamaan ajurit ja yhteyden testattua, pysähtyi edistys ongelmaan missä Visual Studio ei osannut luoda Modelia valmiista tietokannasta, eikä tietokantaa valmiista Modelista. Tämä ilmeni Stackoverflow:ssa paljon puhutuksi DbNull -bugiksi, mihin oli kyllä monia ratkaisuja, mutta niin oli myös vähintään yhtä monia onnistumattomia korjausyrituksiä. Lopulta yrityksiin kyllästyttyäni potkaisin projektiin paikallisen SQL Serverin pystyyn ja projekti lähti vihdoin etenemään.

Myöhemmin SQL Serveri aiheuttikin hetkellistä päänvaivaa yritettäessä puskea testidataa kantaan, sillä sen kanssa ei voinutkaan kommunikoida aivan vastaavanlaisilla käskyillä kuin MySQL:ssa, vaan oli käytettävä T-SQL -komentoja. Tämä vaati hetken opiskelua kunnes data alkoi kulkemaan kumpaankin suuntaan.

# Itsearviointi

Mielestäni harjoitustyö on ollut hyvin opettavainen ja mielenkiintoinen kokonaisuus. Tunnilla en ole ollut läsnä kurssin hieman sivuuttaen oman linjani varsinaista opintosuunnitelmaa ja kiireisen aikataulun vuoksi. Onneksi C# ja WPF ovat laajasti käytettyjä tekniikoita ja internet on ohjeita täynnä. Silti suuremman kokonaisuuden hallitseminen on haastavaa ilman aukotonta suunnitelmaa, kun ei vielä omaa vuosien kokemusta aiheista.

Lopputulokseen olen hyvin tyytyväinen, sillä aikaa harjoitustyön tekemiseen oli hyvin rajallisesti rankan kevään takia, silti täyttäen kaikki projektille osoitetut vaatimukset. Kurssin alussa puhuimme Esa Salmikankaan kanssa harjoitustyön kuittaavan Windows-ohjelmointi -kurssin ja mikäli työ määrää kurssista numerollisen arvosanan, on numero 5 mielestäni oikeutettu.