Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Yrityksen tietojärjestelmät

2014

Jesse Kesti

TARJOUSPYYNTÖJEN KIRJAAMISJÄRJESTELMÄ

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma | Yrityksen tietojärjestelmät

2014 | 37

Ohjaaja Anne Jumppanen

Jesse Kesti

TARJOUSPYYNTÖJEN KIRJAAMISJÄRJESTELMÄ

Tämän opinnäytetyön tarkoituksena oli luoda tarjouspyyntöjen kirjaamisjärjestelmä, jonka avulla yrityksen on mahdollista kirjata asiakkailta saatuja tarjouspyyntöjä esimerkiksi messuilla tai muissa vastaavanlaisissa tapahtumissa.

Opinnäytetyön teoria koostuu salaustekniikoiden kuvauksesta, jota käytetään sovelluksessa muun muassa asiakkaan tietojen salaukseen. Tietojen salausta käsitellään sekä teoriatasolla että käytännön tasolla, kun tutkitaan, miten salausta käytetään sovelluksessa hyödyksi.

Tarjouspyyntöjen kirjaamisjärjestelmässä asiakkaiden tietoja käsitellään erillisen sovelluksen avulla. Käsiteltävät tiedot talletetaan erilliseen tietokantaan, josta ne myös noudetaan sovellukseen esimerkiksi tiedon muokkaamista varten. Opinnäytetyössä kuvataan tämän tehdyn sovelluksen toimintoja, ja annetaan esimerkkejä sovelluksen käytännön toiminnasta.

ASIASANAT:

Ohjelmointi, salaus, SQL, tietosuoja

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information technology | Information systems

2014 | 37

Instructor Anne Jumppanen

Jesse Kesti

AN OFFER REQUEST SYSTEM

The purpose of this thesis was to create an offer request system that allows a company to collect offer requests from clients during different events.

The theory of this thesis consists of descriptions of different encryption algorithms that are used in the application to encrypt client information and other sensitive data. Encryption is described in both theory and practice as it is explained how the application utilizes encryption.

In the offer request system the client data is managed with a separate software. The managed information that is being handled is stored in a database where it is fetched into the application whenever modifications or other alterations to the data occur. This thesis describes the functions of the application and gives examples of the way the application works in practice.

KEYWORDS:

Programming, encryption, SQL, data protection

SISÄLTÖ

KUVAT

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena oli luoda tarjouspyyntöjärjestelmä, johon voidaan kirjata asiakkailta tulleita tarjouspyyntöjä helposti talletettavaan ja selattavaan muotoon. Sovellus suunniteltiin ensisijaisesti yrityksille, jotka osallistuvat erilaisiin tapahtumiin, ja haluavat kirjata kaikkien asiakkaiden tarjouspyynnöt yhtenäiseen järjestelmään.

Opinnäytetyön teoria-osuudessa kuvataan salaustekniikkaa, jonka avulla sovelluksessa suojataan esimerkiksi asiakkaiden tietoja. Teorian lisäksi opinnäytetyössä annetaan myös käytännön esimerkkejä siitä, miten salausta käytetään sovelluksessa hyödyksi.

Tarjouspyyntöjärjestelmään syötetyt tiedot talletetaan tietokantaan, josta ne haetaan käyttäjän näkyviin esimerkiksi tietoja muokatessa tai tietoja vertaillessa. Tämän vuoksi opinnäytetyössä kuvataan myös lyhyesti niitä perusteita, jolla tietoja haetaan tietokannasta ja joilla niitä talletetaan tietokantaan.

Opinnäytetyön lopuksi kuvataan vielä tehdyn sovelluksen toimintaa sekä teoriatasolla että käytännön tasolla. Teoriatasolla perehdytään muun muassa tietojen lisäämisen, poistamisen ja tallettamisen logiikkaan, kun taas käytännön tasolla perehdytään muun muassa sovelluksen toimintaan ensimmäisellä käynnistyskerralla.

2 TIETOJEN SALAUS

Tietojen salaus on tärkeää, jotta voidaan estää tietojen joutuminen ulkopuolisten henkilöiden käsiin. Salauksen avulla voidaan varmistaa myös se, että vaikka ulkopuoliset henkilöt näkisivät tiedot kun niitä siirretään lähteestä toiseen, nämä tiedot eivät ole ymmärrettävässä muodossa henkilölle, jolla ei ole tietojen purkuun sopivaa avainta (Feldwick 2005, 4).

Ennen tietokoneiden käytön yleistymistä rakennetut verkot olivat yleensä vain sisäisiä verkkoja, eli ne olivat kokonaan suljettuja ulkopuoliselta pääsyltä. (Rhodes-Ousley 2013, 6). Tämä periaate ei päde enää nykymaailmassa, ja esimerkiksi tarjouspyyntöjärjestelmän tapauksessa sovelluksen tietokantaa voidaan ylläpitää palvelimella, johon otetaan yhteyttä sovelluksen avulla palvelimen sisäisen verkon ulkopuolelta.

Tyypillisessä tietomurrossa pyritään pääsemään sisään juuri verkossa olevalle tietoa sisältävälle palvelimelle, kuten SQL-palvelimelle. Mikäli palvelimelle ei ole otettu käyttöön tarvittavia suojatoimenpiteitä esimerkiksi SQL-injektioita vastaan, saattaa hyökkääjä päästä helposti palvelimeen käsiksi ja ottaa haltuunsa palvelimella olevia tietoja (Dorrans 2010, 2-5.) Tietojen salauksella voidaan estää mahdollisten tietomurtojen sattuessa tietojen joutuminen vääriin käsiin ainakin siten, että tiedot olisivat suoraan luettavassa muodossa.

2.1 Sovelluksessa salattavat tiedot sekä niiden salausperiaate

Sovellukseen talletetaan useita tietoja, joiden tallennus selkokielisenä ei ole tietoturvan kannalta hyväksyttävä idea. Tämän johdosta sovelluksessa käytetään salausalgoritmia, jonka avulla tiedot salataan, kun ne talletetaan tietokantaan, ja puretaan luettavaksi, kun ne avataan sovelluksessa.

Salattavia tietoja sovelluksessa ovat käyttäjän salasana sekä useimmat asiakkaan tiedoista. Asiakkaan tietoihin talletetaan asiakkaan nimi, osoite, puhelinnu-

mero sekä sähköpostiosoite, joten näiden tietojen salaus on tärkeää. Mikäli tietoja ei salattaisi, voisi kuka tahansa käydä lukemassa näitä tietoja suoraan tietokannasta, mikäli he pääsisivät kirjautumaan tietokantaan sisään.

Tietojen salaukseen käytetään sovelluksessa symmetristä salausta. Symmetrinen salaus edellyttää, että sekä salattavan tiedon vastaanottajalla ja salattavan tiedon lähettäjällä on käytössä sama salausavain (Chapman 2000, 25). Tarjouspyyntösovelluksen tapauksessa salattavaa tietoa lähetetään kahteen suuntaan: ensinnä tieto salataan sovelluksessa, josta se talletetaan kantaan. Jatkossa salattu tieto haetaan tietokannasta ja salaus puretaan sovelluksessa. Salauksen purku tapahtuu siis aina sovelluksen puolella.

Symmetrisen salauksen vahvuutena voidaan pitää sen nopeutta, kun taas suurimpana ongelman pidetään sitä, miten salausavain saadaan molemmille tahoille ilman, että kolmas osapuoli saa avainta haltuunsa (Chapman 2000, 25). Sovelluksen suhteen tätä ongelmaa ei ole, sillä salausavaimen ei tarvitse olla kahdessa eri paikassa, vaan salaus ja purku tapahtuvat kokonaan sovelluksen puolella.

Mikäli esimerkiksi sovelluksen jatkokehityksessä ilmenisi tarve jakaa purkuavain esimerkiksi useiden käyttäjien kesken, voitaisiin tietoturvaa parantaa käyttämällä Diffie-Hellman-avainta. Diffie-Hellman-avain perustuu matemaattiseen kaavaan, jonka perusteella molemmille avaimille lasketaan omat, toisistaan riippumattomat arvonsa, jonka jälkeen ne voivat jakaa saamansa tuloksen keskenään ja laskea tämän perusteella kolmannen avaimen, jonka perusteella päätellään, olivatko avaimet oikeat (Rhodes-Ousley 2013, 245.)

2.2 Sovelluksessa käytettävä salausalgoritmi

Sovelluksen salausalgoritmia valitessa oli kaksi vaihtoehtoa. Ensimmäinen vaihtoehto oli Advanced Encryption Standard eli AES-algoritmi, ja toisena vaihtoehtona oli Triple Data Encryption Algorithm eli Triple DES-algoritmi. Nämä kaksi olivat molemmat hyviä valintoja, sillä molemmat olivat opinnäytetyön tekohetkellä vielä aktiivisessa käytössä, ja molemmat on helppo sisällyttää Visual Basic .NET-sovelluksiin (Bosworth 2014, 1445.)

AES-algoritmi koostuu kahdesta osasta, tietojenkäsittelyosasta ja avaimen laajennusosasta. Tietojenkäsittelyosan tarkoituksena on vaihtaa rivien ja sarakkeiden järjestystä salausta suoritettaessa sekä tallettaa avainarvo, joka luodaan avaimen laajennusosassa. AES toimii 128-bittisissä lohkoissa, mutta sen avaimen koko voi olla joko 128, 192 tai 256 bittiä (Saini ym. 2014, 34.)

AES-algoritmin rakenne pitää sisällään tilamatriisin, joka on kooltaan 4*Nb, jossa neljä on rivien määrä, ja Nb sarakkeiden määrä. Tämä tarkoittaa käytännössä, että jokaisessa tilamatriisissa on 16 syötettä, joista jokainen pitää sisällään yhden tavun. Yksi tavu on kahdeksan bittiä, joten voidaan laskea 16*8=128 bittiä (Feldwick, 2005, 20-21.) AES-algoritmin tapauksessa on huomattava, että avaimen pituus ei välttämättä ole vain 128 bittiä kuten yllä olevassa esimerkissä, vaan avaimen pituus voi olla myös 192 ja 256 bittiä riippuen siitä, mikä arvo Nb-muuttujalle annetaan.

AES-algoritmin salauskierros koostuu neljästä osasta. Suoritettavien salauskierroksien lukumäärä puolestaan on riippuvainen sekä avaimen että salattavan tiedon koosta. Salauskierroksen ensimmäinen vaihe vaihtaa jokaisen tavun yksitellen sen arvon multiplikatiiviseen käänteislukuun. Toisessa vaiheessa matriisissa vaihdetaan rivien arvojen järjestystä. Arvojen järjestyksen vaihtamisen luku on riippuvainen salattavan tiedon koosta sekä käsiteltävästä rivinumerosta. Kolmannessa vaiheessa vaihdetaan matriisin sarakkeiden järjestystä kertomalla aina kunkin matriisissa olevan koordinaattivektorin arvot alla olevan taulukon mukaisesti. On huomattavaa, että kolmas vaihe jää suorittamatta, kun ollaan viimeisellä salauskierroksella.

$$[b_0] = [2\ 3\ 1\ 1]\ [a_0]$$

$$[b_1] = [1\ 2\ 3\ 1]\ [a_1]$$

$$[b_2] = [1\ 1\ 2\ 3]\ [a_2]$$

$$[b_3] = [3\ 1\ 1\ 2]\ [a_3]$$

Neljännessä vaiheessa lisätään avaimen laajennusosassa luotu kierrosavain matriisiin. Tätä avainta käytetään algoritmia purettaessa eli silloin, kun halutaan kääntää salattu teksti takaisin luettavaan muotoon. (Feldwick, 2005, 22-24.)

Triple DES perustuu IBM:n 1960-luvulla kehittämään Data Encryption Algorithm-algoritmiin eli DES-algoritmiin. DES pohjautuu 64-bittisiin lohkoihin, jotka käyttävät 56-bittisiä avaimia. DES-algoritmissa syötetty selkeäkielinen teksti käsitellään 64-bittisissä lohkoissa, jotka salataan 64-bittisiin lohkoihin 16 salauskierroksen avulla (Rhee 2003, 58.) Tämä tekee Triple DES-algoritmista huomattavasti vaikeamman murtaa verrattuna DES-algoritmiin. Triple DES:n toimintaperiaatetta kuvataan alla olevissa kaavoissa. Näissä kaavioissa käytetään seuraavia kirjaimia kuvaamaan muuttujia:

K = Key eli avain

E = Encrypt eli salaus

D = Decrypt eli salauksen purku

Triple DES-avaimessa avaimen pituus on 168 bittiä. Näitä avaimia voidaan käyttää kahdella tavalla. Ensimmäinen vaihtoehto on käyttää kahta avainta:

K1 = K3, K2

Ylläolevassa esimerkissä avaimet on merkitty numeroittain järjestyksessä, joten K1 on ensimmäinen avain, K3 kolmas avain ja K2 toinen avain. Toinen vaihtoehto on käyttää kolmea avainta:

K1, K2, K3

Ylläolevassa esimerkissä avaimet on myös merkitty numeroittain järjestyksessä, joten K1 on ensimmäinen avain, K2 toinen avain ja K3 kolmas avain. Kolmen avaimen käyttö on suositeltua, sillä se nostaa salauksen varmuutta (Rhee 2003, 71.)

Triple DES-algoritmissa tiedon salaus tapahtuu seuraavalla logiikalla:

$$\text{salattu teksti} = E_{K3}(D_{K2}(E_{K1}(\text{salaamaton teksti})))$$

Ylläolevassa kaavassa teksti salataan ensin avaimella K1, minkä jälkeen salaus puretaan avaimella K2. Viimeiseksi salaus tehdään avaimella K3. Kun salattu lause halutaan purkaa, käytetään logiikkaa:

$$\text{purettu teksti} = D_{K1}(E_{K2}(D_{K3}(\text{salattu teksti})))$$

Ylläolevassa kaavassa puretaan ensin salaus avaimella K3, jonka jälkeen teksti salataan avaimella K2. Viimeiseksi salaus puretaan avaimella K1 (Rhee 2003, 71-72.) Toisin sanoen salatun tekstin purkamiseen tehtävä toiminto on sama kuin salaamistoiminto, mutta se suoritetaan päinvastaisessa järjestyksessä.

DES-algoritmi on nykypäivän standardeilla ja järjestelmillä liian helposti murrettava, joten sen käyttäminen sovelluksessa ei ole hyvä ajatus. On arvioitu, että nykyisillä järjestelmillä DES-algoritmin pystyy murtamaan jo muutamassa tunnissa (Rhodes-Ousley 2013, 244). Nopeutensa puolesta DES-algoritmia voitaisiin pitää sopivana sovelluksessa. DES pystyy salaamaan ja purkamaan tietoa C-ohjelmointikielellä noin 3.8 megatavua sekunnissa, kun käytetään Pentium III/200 MHz-konetta (Welschenbach 2005, 237). Avaimen pituus on DES-algoritmissa 56 bittiä. 8 bittiä avaimesta on pariteettibittejä, jotka ovat aina joka kahdeksas bitti koko 64-bittisestä avaimesta (Rhee 2003, 58.)

Kun verrataan keskenään AES-algoritmia ja DES-algoritmia, voidaan DES-algoritmin ongelmat huomata helposti. Siinä, missä AES-algoritmin murtaminen niin kutsutulla ”brute force”-taktiikalla, jossa yritetään löytää oikea avain yksinkertaisesti käymällä kaikki mahdolliset avainyhdistelmät läpi, on lähes mahdotonta, DES-algoritmin murtaminen käyttämällä samaa taktiikkaa on huomattavasti helpompaa varsinkin nykyisin käytössä olevilla tietokoneilla. AES-algoritmin murtaminen kokeilemalla kaikki yhdistelmät läpi vaatisi 128-bittisten avainten tapauksessa $3.40282367*10^{38}$:n yhdistelmän läpikäyntiä (Feldwick, 2005, 10.)

Triple DES-algoritmi nostaa DES-algoritmin suojausta laskennallisesti jopa 2^{112} verran. Tämä tekee Triple DES-algoritmista myös hyvin suojatun, varsinkin ”brute force”-hyökkäyksiä vastaan. Tarjouspyyntösovelluksen näkökulmasta erot Triple DES-algoritmin ja AES-algoritmin välillä eivät siis ole suuria. Päädyin käyttämään sovelluksessa Triple DES-algoritmia, sillä sen implementointi sovellukseen oli

käytännön syistä helpompaa, eikä se aiheuttanut laskennallisesti suurempaa tietoturvauhkaa erilaisia murtohyökkäyksiä vastaan kuin AES-algoritmin käyttäminen.

3 TIETOKANTA

3.1 Tietokannan luonti ja tietokantaan yhdistäminen

Sovellus käyttää tietokantanaan Microsoft SQL Serveriä. Microsoft SQL Serverin rakenteellisista muutoksista johtuen sovellus toimii ainoastaan Microsoft SQL Server 2008:n tai uudemman version kanssa. Ensimmäisellä sovelluksen käynnistyskerralla sovelluksessa tarkistetaan, löytyykö järjestelmän rekisteristä merkintää tietokantaan yhdistämistä varten. Mikäli merkintää ei löydy, niin käyttäjää kehotetaan luomaan tietokannan taulut sovelluksessa olevan toiminnon avulla.

Sovelluksen Tietokanta-ruudussa voidaan alustaa tietokanta automaattisesti ilman, että käyttäjän tarvitsee tehdä tietokantaa tietokantapalvelimelle manuaalisesti. Lisää tietokannan luonnista Tietokanta-ruudun avulla kerrotaan luvussa 4.

Ennen kuin sovellus pystyy luomaan tietokannan taulut, tulee tietokantapalvelimella olla olemassa tyhjä tietokanta, jonne taulut myöhemmin luodaan sovelluksen avulla. Tyhjään tietokantaan tulee olla määriteltynä käyttäjä, jolla on oikeus lukea ja kirjoittaa tietokantaan tietoja. Luku- ja kirjoitusoikeuksia tarvitaan, kun sovelluksessa luetaan tietoja ruudulle tietokannasta sekä kirjoitetaan sovellukseen kirjattuja tietoja tietokantaan. Lisäksi tietokannan rakenteen luomista varten lisätylle käyttäjälle tulee lisätä tietokannan omistajan oikeudet. Nämä oikeudet mahdollistavat taulujen luomisen tietokantaan. Omistajan oikeudet tulee tietoturvan parantamiseksi ottaa käyttäjältä pois taulujen luomisen jälkeen.

Tietokantaan yhdistetään sovelluksessa käyttämällä ADO-yhdistämisobjektia. Yhteyden luomista varten tarvitaan tietokantapalvelimen nimi, tietokannan nimi, käyttäjätunnus sekä salasana. Käyttäjätunnus sekä salasana ovat talletettuna suoraan sovellukseen, joten niitä ei tarvitse syöttää erikseen connect stringiä talletettaessa. Sen sijaan tietokantapalvelimen nimi ja tietokannan nimi tulee määritellä sovelluksessa. Määrittelyn jälkeen sekä tietokantapalvelimen nimi että tietokannan nimi talletetaan järjestelmän rekisteriin polkuun HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\TarjouspyyntoJarjestelma. Local

Machine-polun alle talletetut tiedot ovat käytettävissä myös esimerkiksi käyttäjän vaihdon jälkeen, joten jokaisen tietokoneen käyttäjän ei tarvitse määrittää tietokannan palvelinta tai tietokannan nimeä erikseen, vaan sovellus voi noutaa ne millä tahansa käyttäjällä ensimmäisen määrityskerran jälkeen. Vaikka käyttäjä käyttäisi 64-bittistä käyttöjärjestelmää, talletetaan tiedot silti 32-bittistä järjestelmää vastaavan Wow6432Node-kansion alle, sillä sovellus toimii oletuksena 32-bittisessä tilassa.

Tietoturvasyistä tietokannan käyttäjätunnus ja salasana haetaan sovelluksen koodista rekisterin sijaan. Vaikka salasana on mahdollista salata ennen rekisteriin kirjoittamista, on turvallisempaa pitää salasana piilotettuna sovelluksen sisällä siten, että ketään ei pääse katsomaan tai kopioimaan salasanaa ja yrittämään mahdollisesti purkaa salausta. Haittapuolena tässä menetelmässä on, että luodun tietokannan käyttäjätunnus ja salasana tulee aina olla sama, ja niiden vaihtaminen vaatii sovelluksen koodin muuttamista.

3.2 Tietokannan taulut

Sovelluksen käyttämä tietokanta sisältää viisi taulua. Tietokanta on siis suhteellisen pieni, ja myös siihen talletettavat tiedot melko yksinkertaisia. Alla on lueteltu tietokannan taulut, niiden tarkoitukset sekä rakenteet.

3.2.1 Käyttäjät

Käyttäjät-taulu pitää sisällään sovelluksen käyttäjät. Käyttäjätietoja luetaan sovellukseen kirjautuessa. Käyttäjät-taululla on neljä saraketta, jotka ovat Käyttäjätunnus, Etunimi, Sukunimi sekä Salasana.

Käyttäjätunnus on maksimissaan 50 merkin mittainen merkkijono, joka tallettaa sovelluksen käyttäjän käyttäjätunnuksen. Käyttäjätunnus-sarake on Käyttäjät-taulun pääavain. Käyttäjätunnuksen käyttäminen pääavaimena pakottaa käyttämään yksilöllisiä käyttäjätunnuksia jokaisella käyttäjällä. Mikäli käyttäjille sallittai-

siin päällekkäisiä käyttäjätunnuksia, aiheuttaisi tämä ongelmia sovellukseen kirjauduttuessa, kun yritetään tunnistaa, mikä salasana kuuluu kyseiselle käyttäjälle.

Etunimi-sarake sekä Sukunimi-sarake pitävät sisällään käyttäjän Etu- ja sukunimen. Molemmat näistä sarakkeista ovat maksimissaan 255 merkin pituisia merkkijonoja. Näiden sarakkeiden tarkoitus on selkeyttää käyttäjien hallintaa tapauksissa, jossa sovellukseen on lisätty useita käyttäjiä.

Salasana-sarake on maksimissaan 50 merkin pituinen merkkijono. Kentän tarkoitus on tallettaa käyttäjän salasana, jota käytetään sovellukseen kirjauduttuessa yhdessä käyttäjätunnuksen kanssa. Kun salasana talletetaan tietokantaan sovelluksen kautta, se ajetaan salausalgoritmin läpi, joten salasana talletetaan tietokantaan salattuna. Sovellukseen kirjauduttuessa ajetaan salausalgoritmin purkumetodi, joka purkaa salauksen ja mahdollistaa kirjautumisen tarkistuksen.

3.2.2 Tarjouspyynnöt

Tarjouspyynnöt-tauluun talletetaan jätetyt tarjouspyynnöt. Tarjouspyyntö-taulussa on yhdeksän saraketta, jotka ovat asiakkaan numero, tapahtumanumero, etunimi, sukunimi, osoite, puhelinnumero, sähköpostiosoite, toteutuskuukausi sekä toteutusvuosi. Tarjouspyynnöt-taulu linkitetään Tarjoustieto-tauluun asiakkaan numeron sekä tapahtumanumeron perusteella, joten vuoksi Tarjouspyynnöt-taulussa käytetään yhdistelmäavainta, joka koostuu asiakasnumerokentästä sekä tapahtumanumerokentästä.

Sekä asiakasnumero että tapahtumanumero ovat integer-tyypin sarakkeita. Asiakasnumero-kenttä saa arvonsa juoksevasti, kun asiakas lisätään, kun taas tapahtumanumeron kenttäarvo saadaan Tapahtumat-taulusta, jossa se on pitänyt numeroida luonnin yhteydessä juoksevasti ennen kuin sitä voidaan käyttää Tarjouspyyntö-taulussa.

Sarakkeet etunimi, sukunimi, osoite, puhelinnumero sekä sähköpostiosoite ovat maksimissaan 255 merkin pituisia merkkijonoja, joihin talletetaan asiakkaan perustiedot. Näiden kenttien arvot voivat olla päällekkäisiä toisten talletettujen tietojen kanssa, eli samalla asiakkaalle voidaan tallettaa useita eri tarjouspyyntöjä, kunhan asiakasnumero-kentän arvo ei ole sama.

Myös toteutuskuukausi ja toteutusvuosi-sarakkeet ovat pituudeltaan maksimissaan 255 merkin pituisia merkkijonoja. Näiden aikojen tallennus ohjataan ohjelmallisesti käyttäjälle siten, että vain sovellukseen valmiiksi luotuja arvoja voidaan tallettaa näihin kenttiin. Toisin sanoen sovellukseen luodaan jokainen vuoden kuukausi ja vuosi nykyisestä vuodesta 100 vuotta eteenpäin, jotka käyttäjä voi valita tarpeen mukaan. Toteutusvuosi-sarakkeessa on siis vuonna 2014 arvot 2014-2114, vuonna 2015 arvot 2015-2115 ja niin edelleen.

3.2.3 Tarjoustiedot

Tarjoustiedot-tauluun talletetaan tarkemmin tarjouspyyntöjen tiedot. Tarjoustietotaulua voi ajatella esimerkiksi lomakepohjana, jota käyttäjä voi tarpeen mukaan muokata tarpeen mukaan. Tarjoustiedot-taulussa on neljä saraketta, jotka ovat asiakkaan numero, tapahtumanumero, parametrinimi sekä parametriarvo. Tarjoustiedot-taulussa sarakkeet asiakkaan numero sekä tapahtumanumero muodostavat yhdistelmäavaimen, jonka perusteella haetaan oikean asiakkaan tiedot.

Sekä asiakasnumero että tapahtumanumero ovat integer-tyypin kenttiä. Molempien kenttien arvot haetaan Tarjouspyyntö-taulusta, kun asiakkaan tietoja talletetaan kantaan. Näin varmistetaan, että kun sovelluksessa haetaan tai viedään tarjouspyyntötietoja, saadaan haettua oikean asiakkaan tiedot.

Parametrinimi-sarake on maksimissaan 255 merkin pituinen merkkijono. Tähän sarakkeeseen talletetaan halutun talletettavan tiedon nimi, esimerkiksi ”kohteen tyyppi” tai ”kohteen valmistumisvuosi.” Parametriarvo-sarakkeen pituus on myös maksimissaan 255 merkkiä, ja siihen on tarkoitus tallettaa parametrinimi-sarakkeessa annetun otsakkeen alle kuuluva tieto. Esimerkiksi ”kohteen tyyppi”-otsikon alle voi tallettaa tiedon ”omakotitalo”, ja ”kohteen valmistumisvuosi”-otsikon

alle voi tallettaa arvon ”1995.” Nämä kentät yhdistämällä voidaan sovelluksessa luoda niin monta erilaista talletettavaa tietoa kuin kulloinkin tarvitaan, ja käyttää näitä arvoja yhdessä Tarjouspyynnöt-taulun tietojen kanssa.

3.2.4 Tapahtumat

Tapahtumat-tauluun talletetaan eri tapahtumat, joissa yrityksen on tarkoitus kerätä tarjouspyyntöjä. Tapahtumat-taulussa on kolme saraketta, jotka ovat tapahtumanumero, tapahtuman nimi sekä tapahtuman vuosi.

Tapahtumanumero on integer-tyypin kenttä, joka numeroidaan juoksevasti aina tapahtuman luonnin yhteydessä. Tapahtumanumero toimii taulun pääavaimena, eli jokaisella tapahtumalla pitää olla oma numeronsa, jotta niiden tietojen haku onnistuu muissa sovelluksen osissa.

Tapahtuman nimi on maksimissaan 255 merkin pituinen merkkijono, johon talletetaan eri tapahtumien nimet. Mikäli yritys osallistuu esimerkiksi joka vuosi järjestettäville messuille, käytetään hyödyksi tapahtuman vuosi-saraketta. Tapahtuman vuosi-sarake on maksimissaan 255 merkin pituinen merkkijono, johon talletetaan tapahtuman vuosi. Sovelluksessa rajataan tapahtuman vuosi-kentän täyttöä siten, että tauluun voidaan kirjata ainoastaan arvoja, jotka ovat joko sata vuotta nykyistä vuotta vanhempia tai sata vuotta nykyistä vuotta uudempia. Esimerkiksi vuonna 2014 voidaan käyttää vuosia 1914-2114, ja vuonna 2015 vuosia 1915-2115. On tärkeää mahdollistaa tallennus myös nykyistä vuotta edellisille vuosille, sillä vuoden vaihtuessa voidaan joutua palaamaan takaisin vanhempiin tietoihin. Mikäli käytettäisiin ainoastaan nykyistä vuotta seuraavia vuosia, ei tietoja osattaisi hakea kannasta enää oikein sovelluksen puolella.

Kun yhdistetään tapahtuman nimi sekä tapahtuman vuosi-sarakkeet, mahdollistetaan saman tapahtuman tallennus eri vuosille. Esimerkiksi Messu-niminen vuosittainen tapahtuma voidaan tallettaa eri vuosille, joten nimeksi voidaan tallettaa Messu 2014, Messu 2015 ja niin edelleen.

3.2.5 Lomake

Lomake-tauluun talletetaan lomakepohja, jota käytetään tarjouspyyntöjä tehdessä. Lomake-taulussa on kaksi saraketta, lomakenimi sekä lomakeparametri. Molemmat näistä sarakkeista ovat maksimissaan 50 merkin pituisia merkkijonotyypin muuttujia.

Lomake-taulussa on käytössä yhdistelmäavain, joka koostuu lomakenimi-sarakkeesta sekä lomakeparametri-sarakkeesta. Yhdistelmäavaimen tarkoituksena on estää päällekkäisten parametriarvojen tallennus yhteen lomakkeeseen.

4 SOVELLUKSEN RUUDUT JA TOIMINTA

Sovelluksen toiminta perustuu kahdeksaan ruutuun, joista jokaisella on oma tarkoituksensa. Nämä ruudut ovat Sisaankirjaus, Paaruutu, Tietokanta, Kayttajat, Tapahtumat, Lomake, Tarjouspyynnot sekä ExcelExport. Lisäksi sovelluksessa on luokka gMuuttujat, johon talletetaan sovelluksessa ruutujen väliset tiedot, joita ei talleteta tietokantaan.

4.1 Virheenkäsittely

Virheen sattuessa on tärkeää antaa käyttäjälle ilmoitus, missä virhe tapahtui ja mikä virhe oli. Tietoturvan ylläpitämiseksi peruskäyttäjälle ei kuitenkaan kerrota tarkemmin virheen laadusta, vaan virheistä kertoessa käytetään teknisiä tietoja sisältämättömiä viestejä, jotka käyttäjä voi välittää eteenpäin ylläpidolle. Virheiden ilmoittamislogiikka sovelluksessa on yksinkertainen. Jokaisen menetelmän tai metodin alussa määritellään virheenkäsittely seuraavalla kutsulla:

```
On Error GoTo Error_Err
```

Ylläolevalla kutsulla kerrotaan sovellukselle, että virheen tapahtuessa sovellus hyppää Error_Err-nimiseen virheenkäsittelytoimintoon. Error_Err-virheenkäsittelytoiminto on seuraavanlainen:

```
Error_Err:

        MsgBox(HaeVirheViesti(Err.Number), MsgBoxStyle.OkOnly,
        Me.Text)
        Resume Next
```

Virheenkäsittelytoiminto määritellään alussa Error_Err: -komennolla, jolloin sovellus tunnistaa, että kyseessä on virheenkäsittelytoiminto. Tämän jälkeen sovellukselle kerrotaan, että se tulostaa ruudulle tekstiä komennolla MsgBox. Tulostettavan tekstin sisältö määritellään MsgBox-komennon sulkeiden sisällä. Virheviestin sisältö saadaan metodilla HaeVirheViesti, joka käyttää hyödykseen ilmenneen virheen numeroa hakiessaan virheviestin. Virhenumero saadaan automaat-

tisesti virheen sattuessa Visual Basic .NET –virhekirjastosta. Virheviestin ulkonäkö määritellään komennolla MsgBoxStyle.OkOnly, jolla tulostetaan laatikko, josta käyttäjä voi mennä eteenpäin painamalla OK-painiketta. Lopuksi komento Me.Text kertoo, mistä ruudusta virheviesti tuli. On huomioitavaa, että sovelluksessa ruudun nimi on eri kuin mitä käyttäjä näkee. Esimerkiksi ruutu Sisaankirjaus näkyy käyttäjälle nimellä Sisäänkirjaus.

4.2 Salausalgoritmin käyttö sovelluksessa

Salausalgoritmin käyttöönotossa käytetään hyödyksi Visual Basicin kirjastoon sisällytettyä System.Security.Cryptography-kirjastoa. Tämän kirjaston avulla voidaan käyttää Triple DES-salausta sovelluksessa, mikä helpottaa tehtävien laskutoimitusten ja tarkistusten määrää (Microsoft.)

Triple DES-algoritmia käytettäessä tulee määrittää avain, jonka perusteella merkkijonoja voidaan salata. Tämä byte-tyypin muuttuja on 24 merkin pituinen, koska yksi bitti on kahdeksan tavua. Tästä voidaan laskea:

DES-avain = 64 bittiä

Triple DES-avain = 192 bittiä (3*64)

Salatun avaimen pituus = 24 tavua (8*24 = 192)

Avaimen arvo voidaan määritellä esimerkiksi seuraavasti:

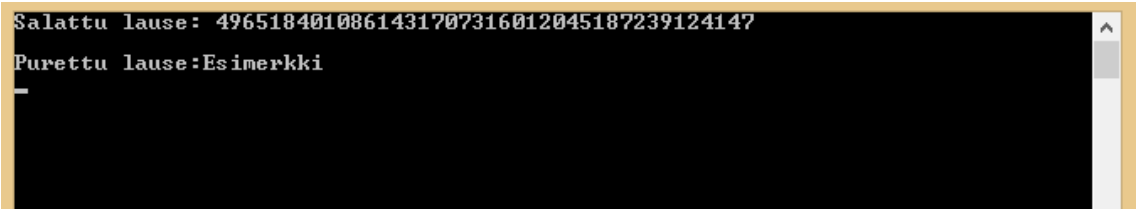
```
Avain() As Byte = {12, 1, 9, 31, 4, 51, 88, 98, 15, 8, 90, 24, 36, 11,
48, 5, 68, 52, 43, 20, 64, 10, 79, 3}
```

Seuraavaksi koodissa määritellään byte-tyypin muuttuja Kaynnistysvektori, jonka pituus on kahdeksan tavua eli yksi bitti. Tämän muuttujan tarkoituksena on toimia osana salaustoimintoa yhdessä aiemmin määritellyn avaimen kanssa. Kaynnistyvektori-muuttujan arvoksi voidaan asettaa esimerkiksi:

```
Private KaynnistysVektori() As Byte = {1, 14, 78, 30, 55, 68, 70, 98}
```

Näiden määriteltyjen avaimien avulla tehdään salaus- ja salauksen purkutoiminnot sovelluksessa. On huomattava, että näitä muuttujia ei käytetä oman luokkansa ulkopuolella, joten niiden nimeämiskäytäntö ei ole sama kuin muiden sovellusten yleisten muuttujien.

Kun sovelluksessa täytyy salata jokin tieto, käytetään hyödyksi julkista gSalaa-funktiota. Tietoja purkaessa sovellukseen on luotu gPuraSalaus-funktio, jonka tarkoitus on purkaa tehty salaus.



Kuva 1. Esimerkki salauksen ja salauksen purun toiminnasta.

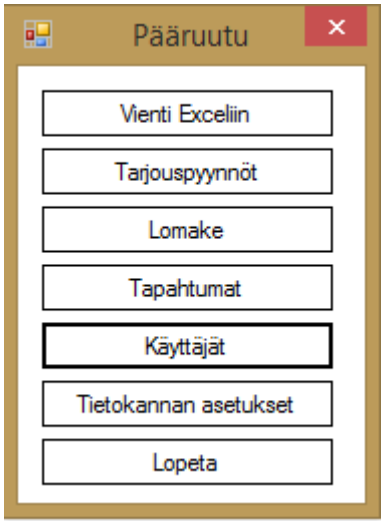
Yllä olevassa kuvassa näkyy ensin kirjoitettu lause salatussa muodossa. Tämän jälkeen kutsutaan gPuraSalaus-funktiota, joka purkaa salauksen ja tuottaa ruudulle tulosteeksi alkuperäisen syötteen, tässä tapauksessa sanan ”Esimerkki.” Tällä tavoin pystytään helposti tietokantaan tallennettaessa salaamaan tieto käyttämällä hyväksi funktiota gSalaa, ja antamalla sille muuttujana tallennettavan tiedon. Kun salattua tietoa haetaan kannasta, ajetaan se gPuraSalaus-funktion läpi, jolloin teksti tulee selkokielisenä versiona ruudulle.

4.3 Sisaankirjaus ja Paaruutu

Sisaankirjaus-ruudussa kysytään käyttäjän käyttäjätunnusta sekä salasanaa. Näiden tietojen täyttämisen jälkeen sovellus yrittää kirjata käyttäjän sisään tarkistamalla, löytyykö tietokannasta käyttäjätunnusta, johon syötetty salasana täsmää. Mikäli tiedot täsmäävät, päästetään käyttäjä Paaruutu-ruutuun. Mikäli tiedot eivät täsmää, annetaan siitä ilmoitus käyttäjälle.

Salasanan täsmätessä käyttäjätunnukseen Sisaankirjaus-ruutu suljetaan, ja käyttäjälle avataan Paaruutu, josta käyttäjä pystyy avaamaan sovelluksen muut ruudut.

Paaruutu-ruutu pitää sisällään painikkeet, joita painamalla käyttäjä pystyy avaamaan sovelluksen eri ruutuja sekä lopettamaan sovelluksen. Paaruutu-ruutu on sisäänkirjautumisen jälkeen aina käyttäjän näkyvillä, ja se sulkeutuu vasta, kun käyttäjä sulkee sovelluksen.



Kuva 2. Sovelluksen pääruutu.

Käyttäjällä on kaksi tapaa lopettaa sovellus. Käyttäjä voi joko painaa Lopeta-painiketta tai ruudun oikeassa yläkulmassa olevaa rastia, jolloin käyttäjältä kysytään varmistus siitä, haluaako tämä varmasti sulkea sovelluksen. Käyttäjän valitessa ”Kyllä” sovellus suljetaan, ja käyttäjän valitessa ”Ei” sovellus pidetään auki ja sen toiminta jatkuu normaalisti.

Käyttäjän avatessa muita ruutuja, ruutu avataan komennolla:

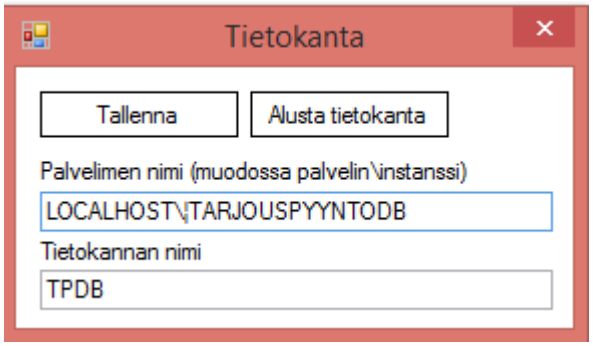
```
Ruutunimi.ShowDialog()
```

Ruutunimi on aina avattavan ruudun nimi, esimerkiksi Kayttajat-ruutua avatessa komento on Kayttajat.ShowDialog(). Komennolla varmistetaan myös se, että avattu ruutu pysyy käyttäjällä aktiivisena, eikä käyttäjä voi siirtyä muihin ruutuihin,

ennen kuin on sulkenut avaamansa ruudun. Näin vältytään esimerkiksi päällekkäisiltä tietojen muokkauksilta, jotka saattaisivat aiheuttaa ongelmia sovelluksen toiminnassa.

4.4 Tietokanta

Tietokanta-ruutua käytetään sovelluksen tietokantapalvelimeen yhdistämiseen sekä sovelluksen tietokannan alustamiseen ja taulujen luomiseen. Tietokanta-ruudun avulla talletetaan tietokannan rekisteriin tietokannan yhdistämiseen tarvittava tietokannan instanssinimi sekä tietokannan nimi, jolloin seuraavilla käynnistyskerroilla sovellus pystyy lukemaan tietokantaan kirjautumiseen tarvittavat tiedot suoraan rekisteristä, eikä Tietokanta-ruutuun tarvitse palata.



Kuva 3. Tietokanta-ruutu, johon on täytetty palvelimen nimi sekä tietokannan nimi.

Tallenna-painiketta painamalla sovellus tallettaa käyttäjän syöttämät arvot rekisteriin seuraavalla komennolla:

```
My.Computer.Registry.SetValue("HKEY_LOCAL_MACHINE\SOFTWARE\Tarjous-
pyyntoJarjestelma", "DataSource", Me.txtPalvelin.Text)
My.Computer.Registry.SetValue("HKEY_LOCAL_MACHINE\SOFTWARE\Tarjous-
pyyntoJarjestelma", "InitialCatalog", Me.txtTietokanta.Text)
```

Ylläolevassa esimerkissä asetetaan tietokoneen rekisteriin paikallisesti arvot DataSource sekä InitialCatalog, joita käytetään hyödyksi seuraavassa sisään kirjauksessa. On huomattavaa, että kun tiedot talletetaan rekisteriin avaimen

HKEY_LOCAL_MACHINE alle nämä tiedot löytyvät, vaikka käyttäjää vaihdettaisiinkin. Tämä tarkoittaa sitä, että kun tietokannan yhdistämisparametrit on kerran asetettu sovelluksessa, niitä ei tarvitse asettaa enää erikseen Windows-käyttäjän vaihtuessa.

Alusta tietokanta-painike alustaa käytössä olevan tietokannan. Ennen alustuksen aloittamista käyttäjältä pyydetään varmistus siitä, haluaako tämä varmasti tehdä uuden tietokannan, sillä uuden tietokannan luomisen yhteydessä vanha tietokanta poistetaan alta, mikäli käytetään samoja tietokannan yhdistämisarvoja.

4.5 Kayttajat

Kayttajat-ruudun avulla hallinnoidaan käyttäjiä, jotka pystyvät kirjautumaan sovellukseen. Sovelluksen käyttäjiltä talletetaan käyttäjätunnuksen ja salasanan lisäksi käyttäjänumero, etunimi sekä sukunimi. Käyttäjänumero määritellään sovelluksessa juoksevasti, joten käyttäjien ei tarvitse itse määritellä tunnusta. Numeron määrittely tapahtuu seuraavasti:

```
lause = "SELECT MAX(Kayttajanumero) as kn FROM Kayttajat"
rs.Open(lause, gMuuttujat.gConnectString.ConnectionString, ADODB.Cur-
sorTypeEnum.adOpenStatic)
If IsDBNull(rs.Fields("kn").Value) = False Then
    Me.txtKayttajanumero.Text = rs.Fields("kn").Value + 1
Else
    Me.txtKayttajanumero.Text = "1"
End If
rs.Close()
```

Ylläolevassa komennossa tehdään seuraavaa: SQL-lauseen avulla valitaan suurin mahdollinen käyttäjänumero taulusta Kayttajat. Mikäli arvoa ei löydy, eli käyttäjiä ei ole lisätty, asetetaan käyttäjänumeron arvoksi yksi. Muussa tapauksessa käyttäjänumeron arvoksi asetetaan kn + 1, jossa kn on suurin käyttäjänumero, joka löytyi tietokannasta.

Käyttäjät erotellaan toisistaan paitsi käyttäjänumeron, myös käyttäjätunnuksen perusteella. Mikäli käyttäjä yrittää lisätä käyttäjätunnuksen, joka on jo lisätty jär-

jestelmään, sovellus ilmoittaa, että samaa käyttäjätunnusta ei voi lisätä. Tällä estetään mahdolliset ongelmat sisäänkirjautumisessa, jossa pitää linkittää salasana käyttäjätunnukseen.

Lisää-painike tyhjentää käyttäjän valinnan sekä laskee uuden käyttäjänumeron järjestelmään. Tämän jälkeen käyttäjä voi tallettaa syöttämänsä tiedot tekstikenttiin ja painaa Tallenna-painiketta, jolloin tietoja yritetään tallettaa kantaan. Ennen tallennusta tarkistetaan, ovatko syötetyt tiedot oikein, eli samaa käyttäjätunnusta ei ole jo olemassa järjestelmässä, eikä syötetyistä tiedoista puutu tietoja. Mikäli tiedot ovat puutteelliset tai käyttäjätunnus on jo lisätty, ilmoitetaan siitä käyttäjälle, ja tallennus keskeytetään.

Poista-painike poistaa kannasta taulukosta valitun käyttäjän. Ennen käyttäjän poistoa sovellus varmistaa käyttäjältä, haluaako tämä varmasti poistaa valitun käyttäjän. Tällä varmistuksella pyritään estämään mahdolliset väärät poistot.

4.6 Tapahtumat ja Lomake

Tapahtumat-ruutu pitää sisällään tapahtumat, joiden alle tarjouspyynnöt talletetaan. Tapahtumiin määritellään numero, nimi sekä vuosi. Vuosi on tärkeä määritellä sen vuoksi, että käyttäjä voi lisätä esimerkiksi vuosittain järjestettävän tapahtuman, ja voi erotella ne toisistaan vuoden avulla. Juoksevan numeron avulla pidetään kirjaa tapahtumien lukumäärästä, ja sitä käytetään myös tapahtumien yksilöinnissä. Tapahtumien numerointi tapahtuu samalla periaatteella kuin esimerkiksi Kayttajat-ruudussa.

Käyttäjän valitessa Tapahtumat-ruudun taulukosta rivin, haetaan ruudun tekstilaatikkoihin kyseisen tapahtuman tiedot. Tässä käyttäjä voi muokata tapahtuman nimeä ja vuotta.

Vuosi-pudotusvalikko täytetään siten, että siihen annetaan arvoja sata vuotta nykyistä vuotta ennen sekä sata vuotta nykyisen vuoden jälkeen. Myös nykyistä

vuotta vanhemmat tapahtumat tulee merkitä, jotta taulukosta rivin valitessa sovellus osaa hakea oikean arvon pudotusvalikosta. Vuosi-pudotusvalikko täytetään seuraavalla logiikalla:

```
For i = -100 To 100
            Me.cmbVuosi.Items.Add(Date.Now.Year + i)
Next
```

Ylläolevassa koodissa muuttuja i on luku, jonka arvoa suurennetaan aina yhdellä numerolla silmukassa. Alussa i:n arvo on -100, ja koodin silmukkaa toistetaan, kunnes i:n arvo on 100. Silmukan täyttövaiheessa vuosi-pudotusvalikkoon lisätään aina Visual Basic .NET –kirjaston päivämääräfunktio Date.Now.Year:n avulla i:n arvo. Date.Now.Year saa arvonsa suoraan järjestelmän kellosta, eli sitä ei tarvitse asettaa erikseen sovelluksessa.

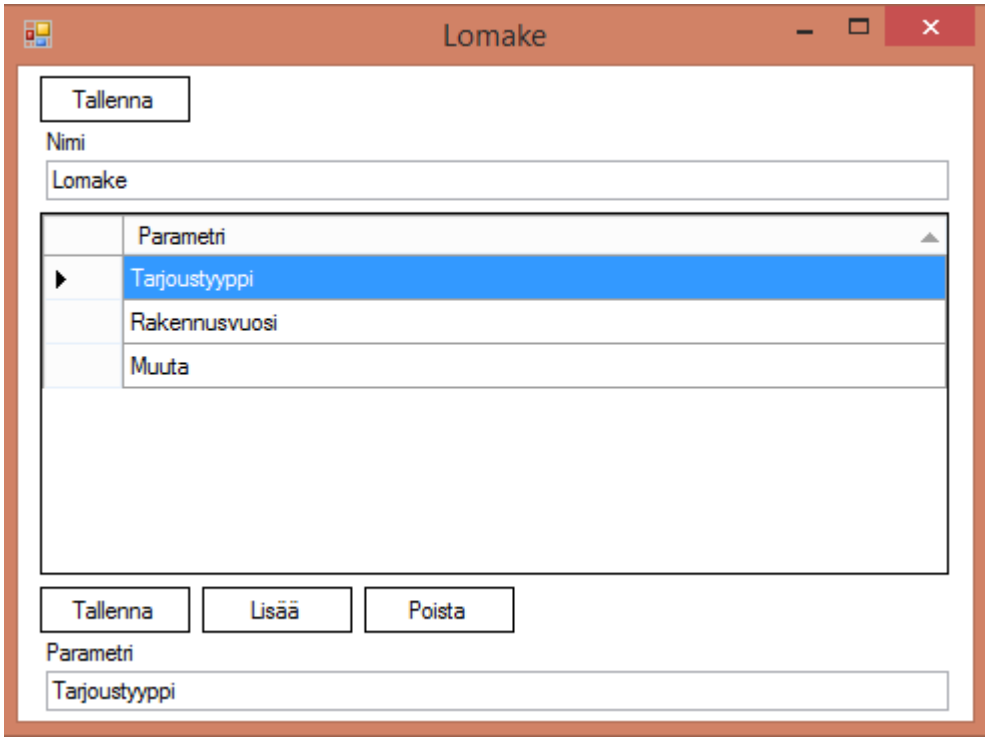
Lisää-painike tyhjentää ruudun alalaidassa olevat tekstikentät, ja laskee automaattisesti suuruusjärjestyksessä seuraavan numeron. Tapahtuman vuodeksi haetaan oletuksena pudotusvalikosta nykyinen vuosi. Nimi-kenttä tyhjennetään, ja käyttäjän annetaan kirjoittaa siihen haluamansa tapahtuman nimi.

Tallenna-painike tallettaa käyttäjän syöttämän tapahtuman, kunhan tämä on antanut tapahtumalle nimen. Mikäli tapahtuman nimi puuttuu, ilmoitetaan siitä käyttäjälle ja tallennus peruutetaan. Käyttäjän ei tarvitse täyttää numero-kenttään arvoa, sillä se luodaan automaattisesti. Myös vuosi-kentän arvo on oletuksena nykyinen vuosi, joten sille ei tarvitse erikseen kirjoittaa arvoa, eikä tämän kentän arvo voi koskaan olla tyhjä, koska siinä olevaa tekstiä ei pääse muokkaamaan.

Poista-painike poistaa käyttäjän valitseman rivin tietokannasta, kunhan taulukon valinta ei ole tyhjä. Mikäli käyttäjä ei ole valinnut riviä taulukosta, ilmoitetaan siitä käyttäjälle ja poisto peruutetaan. Muussa tapauksessa avataan kantayhteys ja tietokannasta poistetaan valittu rivi.

Lomake-ruutua käytetään tarjouspyyntöjen lisätietojen tallettamiseen. Lisätiedoilla tarkoitetaan kaikkia Tarjouspyynto-ruudun perustietojen ulkopuolisia tietoja. Lisätietojen tallettamiseen käytetään yksinkertaista taulukkoa, johon talletetaan lisätiedon nimi. Lisätietojen tietojen tallettamisesta kerrotaan yksityiskohtaisemmin Tarjouspyynnot-ruudun rakenteesta selostettaessa.

Lomake-taulukon rivin valinnan jälkeen ruudun alaosan Parametri-kenttään haetaan taulukon arvo. Käyttäjä voi tässä muokata talletetun parametrin nimeä, ja tallettaa sen uudestaan taulukkoon vanhalle paikalleen.



Kuva 4. Lomake-ruutu, johon on täytettynä kolme esimerkkiparametria.

Lomake-ruudussa on kaksitasoinen tallennusjärjestelmä käytössä. Ylemmän tason Tallenna-painike tallettaa lomakkeelle annetun nimen sekä kaikki taulukon rivit. Alatason painikkeita käytetään taulukon sisällön muokkaamiseen.

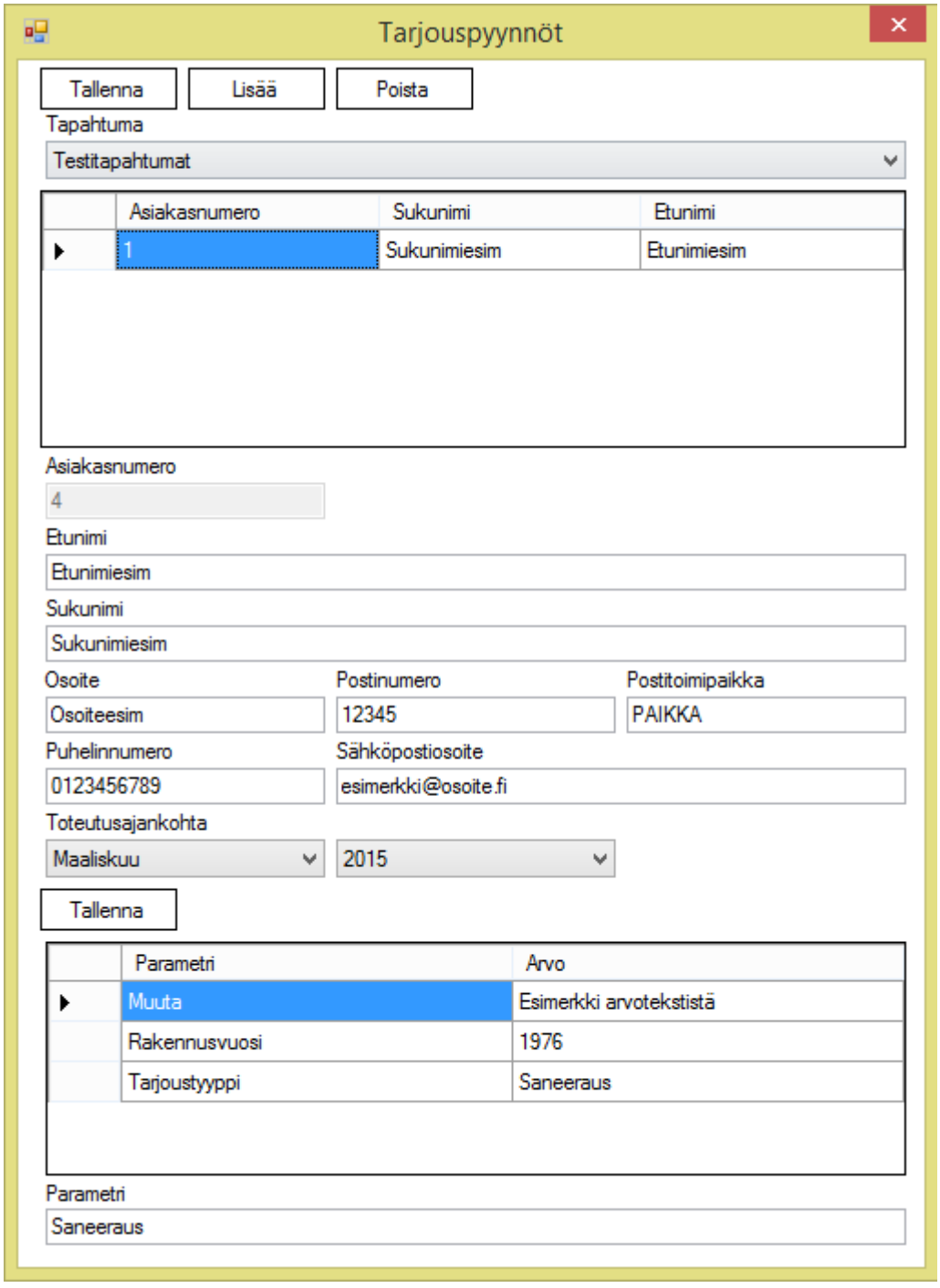
Alatason Lisää-painike lisää taulukkoon uuden rivin, ja tyhjentää Parametri-tekstikentän sisällön, jotta käyttäjä voi syöttää siihen uuden parametrin nimen. Alatason Tallenna-painike tallettaa käyttäjän syöttämän parametrin nimen taulukkoon, kunhan parametrin nimi ei ole tyhjä. Mikäli parametrin nimi on tyhjä, tallennus peruutetaan. Alatason poista-painike poistaa käyttäjän valitseman taulukon rivin taulukosta, kunhan käyttäjä on valinnut taulukosta rivin. Mikäli käyttäjän valinta on tyhjä, poisto peruutetaan.

4.7 Tarjouspyynnot

Tarjouspyynnot-ruutua käytetään tarjouspyyntöjen tallettamiseen eri tapahtumille. Tapahtumat-pudotusvalikkoon haetaan kaikki ne tapahtumat, joita sovellukseen on lisätty Tapahtumat-ruudussa. Käyttäjän valitessa tapahtumaa pudotusvalikosta haetaan ruudussa pudotusvalikon alla olevaan taulukkoon tietokannasta kaikki ne asiakkaat, joiden tarjouspyynnöt on jo lisätty kyseisen tapahtuman alle. Käyttäjän valitessa taulukosta rivin haetaan taulukon alla oleviin kenttiin kaikki käyttäjään liittyvät tiedot, kuten juokseva asiakasnumero sekä asiakkaan perustiedot.

Toteutusajankohdan kuukausivalikkoon haetaan ja täytetään kaikki kuukaudet tammikuusta joulukuuhun. Toteutusajankohdan vuosivalikkoon haetaan ja täytetään vuodet samalla logiikalla kuin esimerkiksi Tapahtumat-ruudussa.

Tarjouspyynto-ruudussa on käytössä kaksitasoinen painikejärjestelmä. Ylätason painikkeilla hallitaan koko tallennettua tarjouspyyntöä. Alatason painikkeilla hallitaan asiakkaan tallennettuja lomakkeen lisätietoja.



Kuva 5. Tarjouspyyntö-ruutu, jossa on talletettuna yksi asiakas.

Ylätason Lisää-painike tyhjentää ruudun tiedot ja laskee uuden juoksevan asiakasnumeron asiakkaalle samalla logiikalla kuin esimerkiksi Kayttajat-ruudussa. Tämän jälkeen käyttäjä voi lisätä asiakkaan tiedot tyhjennettyihin kenttiin.

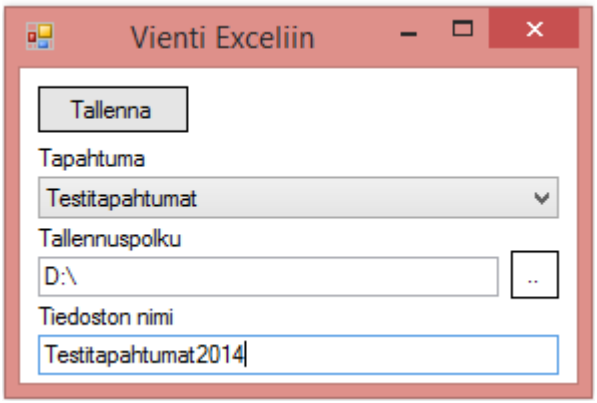
Ylätason Tallenna-painike tarkistaa, että käyttäjä on täyttänyt vähintään kaikki lisätieto-parametrien ulkopuoliset parametrit. Mikäli jokin perustiedoista puuttuu, ilmoitetaan siitä käyttäjälle ja tallennus peruutetaan.

Ylätason Poista-painike poistaa käyttäjän valitseman tarjouspyynnön asiakastaulukosta valitun rivin perusteella. Mikäli käyttäjä ei ole valinnut taulukosta riviä, ilmoitetaan siitä käyttäjälle ja poisto peruutetaan.

Alatason Tallenna-painike tallettaa käyttäjän Parametri-kenttään syöttämän arvon sille riville, jonka käyttäjä on valinnut lisätieto-taulukosta. Mikäli taulukkoon on jo talletettu tieto kyseiselle arvolle, korvataan aiemmin syötetty tieto. Mikäli käyttäjä ei ole valinnut taulukosta riviä, ilmoitetaan siitä käyttäjälle ja tallennus peruutetaan.

4.8 ExcelExport

ExcelExport-ruudun tarkoitus on viedä valitun tapahtuman tarjouspyynnöt Excel-tiedostomuotoon, jonka käyttäjä voi avata sopivalla taulukonkäsittelyohjelmalla. Tätä ruutua voidaan käyttää, kun tarjouspyyntötiedot täytyy viedä sovelluksen ulkopuolelle muokattavaan muotoon.



Kuva 6. ExcelExport-ruutu, jossa on täytettynä tiedot valmiiksi vientiä varten.

Tapahtuma-pudotusvalikkoon haetaan kaikki ne tapahtumat, jotka on tallennettu tietokantaan. Tallennuspolku-kenttään käyttäjä voi tallettaa haluamansa tallennuspolun joko kirjoittamalla käsin, tai painamalla kentän oikealla puolella olevaa

painiketta, jolloin avataan polunvalinta-valikko, josta käyttäjä voi valita haluamansa tallennuspolun. Tiedoston nimeksi käyttäjä voi kirjoittaa haluamansa nimen, jolla tiedosto talletetaan valittuun polkuun.

Tallenna-painike tarkistaa, että valitulle tapahtumalle on valittu tallennuspolku sekä tiedoston nimi. Mikäli jompikumpi näistä arvoista puuttuu, tallennus peruutetaan.

Tiedoston vientiin käytetään hyödyksi Microsoft.Office.Interop.Excel-rajapintaa, jonka avulla voidaan määritellä tarkemmat määritteet eri tiedoston osille, kuten sarakeleveyksille ja rivikorkeuksille.

4.9 gMuuttujat

Luokka gMuuttujat pitää sisällään sellaisia muuttujia, joita käytetään useammassa kuin yhdessä sovelluksen ruudussa. Näitä muuttujia erilaisten tietojen tarkistuksessa sekä yleisesti käytössä olevien muuttujien, kuten kantaan yhdistämisessä käytettävän connection stringin, tallettamiseen. Kaikkien muuttujien nimien alussa on pieni g-kirjain, jolla kuvataan global eli yleisen tason muuttujaa. Alla on lueteltu yleiset muuttujat, jota sovelluksessa käytetään, sekä selitetty niiden käyttötarkoitus.

Muuttuja gConnectString on rajapinnan ADODB.Connection tallettamiseen käytetty muuttuja. ADODB.Connection pitää sisällään tietokannan kirjautumiseen käytettävät tiedot, eli tietokantainstanssin nimen, tietokannan nimen, käyttäjätunnuksen sekä salasanan. Kun tietokantaan talletetaan tai poistetaan tietoja, käytetään gConnectString-muuttujaa yhteyden muodostamisessa.

Muuttuja gKirjautuminenOK tarkistaa, onko käyttäjän kirjautuminen tapahtunut odotetusti. Mikäli esimerkiksi käyttäjän salasana on väärä, tarkistaa sovellus gKirjautuminenOK-muuttujan arvosta, onko kirjautuminen tapahtunut oikein. Kun kirjautuminen ei onnistu, gKirjautuminenOK-muuttujan arvoksi asetetaan 0. Kun kirjautuminen onnistuu, arvoksi asetetaan 1.

Muuttuja gSuljettuKohta tarkistaa, onko sovelluksen sulkeminen aloitettu vasemman yläkulman rastista, vai sovelluksen pääruudussa olevasta Lopeta-painikkeesta. Tätä tarvitaan siksi, että sovelluksen sulkeminen tapahtuu eri tavalla riippuen siitä, miten käyttäjä sulkee sovelluksen. Mikäli käyttäjä sulkee sovelluksen Lopeta-painikkeesta, annetaan gSuljettuKohta-muuttujan arvoksi 0. Mikäli käyttäjä sulkee sovelluksen vasemman yläkulman rastista, laitetaan gSuljettuKohta-muuttujan arvoksi 1.

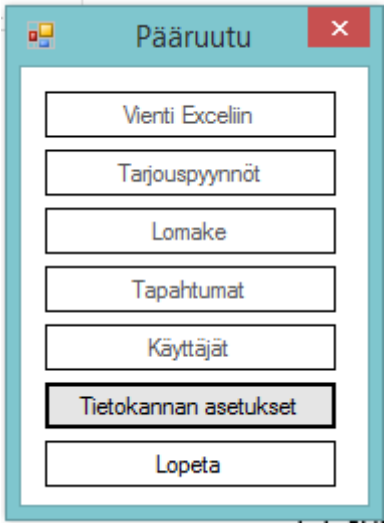
Muuttuja gKayttajaLisatty tarkistaa, onko sovelluksen tietokannassa käyttäjiä. Tämä tarkistus tarvitaan, jotta käyttäjää pystytään ohjeistamaan käyttäjän lisäämisessä tapauksissa, joissa käyttäjiä ei ole vielä lisätty. Mikäli sovelluksesta ei löydy käyttäjiä, annetaan gKayttajaLisatty-muuttujan arvoksi 0. Mikäli tietokannasta löytyy käyttäjiä, annetaan muuttujan arvoksi 1.

Muuttuja gKantaSulkeminen tarkistaa, onko käyttäjällä oikeuksia yhdistää tietokantaan. Tätä muuttujaa käytetään, kun tarkistetaan käyttäjän antamat tietokantaan yhdistämiseen liittyen. Mikäli tietokantaan yhdistäminen ei onnistu, annetaan gKantaSulkeminen-muuttujan arvoksi 0. Mikäli kantaan yhdistäminen onnistuu, arvoksi annetaan 1.

gMuuttuja-luokka pitää sisällään myös funktiot gSalaa() ja gPuraSalaus(). Näitä funktioita käytetään salaamaan ja purkamaan salattavat tiedot sovelluksessa. Tarkempaa tietoa salaus- ja purkufunktioiden toiminnasta on kerrottu kappaleessa salausalgoritmin käyttö sovelluksessa.

4.10 Sovelluksen ensimmäinen käynnistys

Kun sovellus käynnistetään ensimmäisen kerran, tarkistetaan, pystytäänkö tietokantayhteys muodostamaan. Mikäli yhteyttä ei pystytä muodostamaan, ilmoitetaan siitä käyttäjälle, ja järjestelmä avaa suoraan pääruudun siten, että käyttäjä päästetään ainoastaan asettamaan tietokannan asetukset. Käyttäjä voi myös lopettaa sovelluksen niin halutessaan. Kaikki muut painikkeet on poistettu käytöstä tässä vaiheessa. Kun tietokannan parametrit on asetettu, käynnistetään sovellus uudestaan.



Kuva 7. Tietokannan yhdistämisparametreja ei ole asetettu.

Käyttäjän asetettua tietokannan yhdistämiseen tarvittavat tiedot tarkistetaan, onko tietokantaan lisätty käyttäjiä. Mikäli käyttäjiä ei ole lisätty, päästetään käyttäjä suoraan Sisaankirjaus-ruudun ohitse pääruutuun siten, että käyttäjä päästetään vain Kayttajat-ruutuun ja Tietokanta-ruutuun. Käyttäjä voi myös lopettaa sovelluksen niin halutessaan. Kaikki muut painikkeet on otettu pois käytöstä.

5 JATKOKEHITYSIDEAT

Sovellusta on mahdollista kehittää jatkossa eteenpäin kattavammaksi. Muutamia jatkokehitysajatuksia ovat muun muassa käyttäjien tuonti suoraan Active Directorystä, tarjouspyyntöjen sähköpostilähetys, useampien lomakepohjien luominen sekä useampien tietokantojen tukeminen.

Käyttäjien tuonti suoraan Active Directorystä voisi onnistua esimerkiksi käyttämällä Lightweight Directory Access Protocol-verkkoprotokollaa, jonka avulla voidaan etsiä verkossa olevia käyttäjiä esimerkiksi käyttäjätunnuksen tai käyttäjäryhmän perusteella. Tähän tarvittaisiin oma hallintaruutunsa, jonka avulla käyttäjiä lisättäisiin Kayttajat-tauluun tietokantaan. Active Directorystä saataisiin paitsi käyttäjätunnus, sukunimi ja etunimi, myös käyttäjän sähköposti, jota voisi käyttää hyödyksi tarjouspyyntöjen sähköpostilähetyksessä. Tämä vaatisi uuden sähköposti-sarakkeen lisäämisen Kayttajat-tauluun.

Sähköpostilähetyksen avulla voitaisiin tallennetut tarjouspyynnöt lähettää suoraan sähköpostitse haluttuihin osoitteisiin. Tämä vaatisi uuden hallintaruudun, jossa voisi määrittää käyttäjät, mihin sähköposti lähetetään, kuten myös sähköpostipalvelimen asetukset, jota sähköpostilähetyksessä käytettäisiin. Sähköpostilähetyksen käyttöönotto saattaisi vaatia kolmannen osapuolen komponenttia, jonka avulla sähköpostilähetys mahdollistettaisiin.

Jatkossa voitaisiin myös lisätä tuki useammalla eri lomakepohjalle. Tämän avulla voitaisiin käyttää eri asiakkaille tai tapahtumille omia lomakepohjia. Tämä vaatisi juoksevan numeroinnin lisäämisen Lomake-tauluun, jonka avulla erotettaisiin toisistaan eri lomaketyypit. Lisäksi tarjouspyyntö-ruutuun pitäisi lisätä pudotusvalikko, josta käyttäjä voisi valita haluamansa lomakepohjan. Viimeiseksi tarjouspyyntöjen viennin yhteyteen tulisi lisätä, mitä lomakepohjaa käytetään viennin yhteydessä.

Useampien tietokantojen tukeminen voitaisiin lisätä Tietokanta-ruutuun. Eri tietokannat käyttävät erilaisia yhdistämiskeinoja, joten käyttäjä voisi ensin valita pu-

dotusvalikosta, mitä tuettua kantayhteyttä tämä haluaa käyttää. Tällöin tallennuksen yhteydessä talletettaisiin rekisteriin kantatyyppi, ja sovelluksen käynnistyksen yhteydessä kantayhteys muodostettaisiin oikean muotoiseksi rekisteristä löytyvät kantatyypin mukaan.

6 YHTEENVETO

Tässä opinnäytetyössä tutkittiin tarjouspyyntösovelluksen rakentamista alusta loppuun. Teoriaosuus keskittyi salausalgoritmiin, joka on sovelluksessa keskeisessä roolissa talletettavien tietojen luonteen vuoksi. Ilman salausta sovellusta ei voisi käyttää turvallisesti, sillä niin asiakkaiden kuin käyttäjienkin tiedot olisivat ulkopuolisten käsiin joutuessa suoraan luettavissa.

Opinnäytetyön käytännön osuus kuvasi sovelluksen käyttämän tietokannan rakennetta sekä varsinaisen sovelluksen toimintaa. Tietokantaa kuvatessa käytiin tietokannan taulut läpi kentittäin. Kentistä kuvattiin niiden tyyppi sekä tarkoitus sovelluksessa. Myös taulujen avaimet käytiin lävitse tietokannan rakennetta kuvatessa.

Varsinaisen sovelluksen toimintaa kuvattiin ruutukohtaisesti. Ruutuja kuvatessa käytiin läpi niiden tarkoitus ja käytännön toiminta sekä mahdollinen toimintalogiikka muiden sovelluksen ruutujen kanssa. Opinnäytetyön lopuksi kerrottiin vielä kehitysideoista, joita on mahdollista sisällyttää sovellukseen myöhemmin. Nämä ideat olivat sellaisia, jotka ilmenivät kehitystyön aikana, mutta joiden toteuttamiselle ei jäänyt aikaa varsinaisen opinnäytetyöprosessin aikana.

LÄHTEET

Bosworth, S.; Kabay, M.E. & Whyne, E. 2014. Computer Security Handbook. Sixth Edition. Hoboken: Wiley.

Chapman, D. 2000. Developing Secure Applications with Visual Basic. Indianapolis: Sams Publishing.

Dorrans, B. 2010. Beginning ASP.NET Security. Hoboken: Wiley.

Feldwick, C. 2005. Implementation and Analysis of the Advanced Encryption Standard (AES) Algorithm in Different Memory Configurations. Viitattu 9.12.2014. http://www.cs.bath.ac.uk/~mdv/courses/CM30082/projects.bho/2004-5/chris_feldwick-2004-5.pdf

Microsoft. 2014. TripleDES.Create Method (String). Viitattu 20.11.2014. http://msdn.microsoft.com/en-us/library/wxk6k4x8(v=vs.110).aspx

Rhee Young M. 2003. Internet Security Cryptographic Principles, Algorithms and Protocols. Hoboken: Wiley.

Rhodes-Ousley, M. 2013. Information Security The Complete Reference. Second Edition. New York City: McGraw-Hill Education.

Saini, V.; Bangar, P. & Chauhan Singh, H. 2014. Study and Literature Survey of Advanced Encryption Algorithm for Wireless Application. Viitattu 9.12.2014 http://www.ijese.org/attachments/File/v2i6/F0717042614.pdf

Welschenbach, M. 2005. Cryptography in C and C++. Second Edition. New York City: Apress.