Juho Kuivalainen

TYÖKALU TYÖNTEKIJÖIDEN TIETOJEN HALLITSEMISEEN JA HENKILÖPASSIEN TULOSTAMISEEN

Tietojenkäsittelyn koulutusohjelma
Sovellusohjelmoinnin suuntautumisvaihtoehto
2010

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

TYÖKALU TYÖNTEKIJÖIDEN TIETOJEN HALLITSEMISEEN JA HENKILÖPASSIEN TULOSTAMISEEN

Kuivalainen, Juho
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Kesäkuu 2010
Ohjaaja: Stenfors Juha
Sivumäärä: 27
Liitteitä: 1

Asiasanat: ADO.NET, C#-ohjelmointi, SQLite, iTextSharp, Microsoft Excel

---

Opinnäytetyön tarkoituksena oli kehittää sovellus, jonka avulla hallitaan Maata Näkyvissä nimisen tapahtuman työntekijöiden tiedot sekä työntekijöiden henkilöpassien luominen ja tulostaminen. Sovelluksen avulla voidaan lisätä, muokata ja poistaa henkilöitä ja heidän henkilötietojaan tietokantaan. Kehitetyllä sovelluksella vältetään saman passin tulostamista moneen kertaan sekä nopeutetaan tulostusprosessia.

Opinnäytetyön sovellus tulee Maata Näkyvissä-tapahtuman käyttöön. Sovellus korvaa vanhan samaa tehtävää hoitavan sovelluksen, joka ei hoida enää tehtäväänsä riittävän tehokkaasti. Uusi sovellus on kehitetty nopeuttamaan ja helpottamaan työtehtävää.

Opinnäytetyön toteutuksessa käytettiin ADO.NET tekniikkaa ja ohjelmointikielenä C#-kieltä. Tietokantana toimii SQLite-tietokanta. Työ sisältää ohjelmointia, testausta, dokumentointia sekä tietokannan ja käyttöliittymän suunnittelua.

A TOOL FOR CONTROLLING EMPLOYEES INFORMATION AND PRINTING IDENTITY CARDS

Kuivalainen, Juho
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Systems
Juni 2010
Supervisor: Stenfors, Juha
Number of pages: 27
Appendices: 1

Key words: ADO.NET, C#-ohjelmointi, SQLite, iTextSharp, Microsoft Excel

---

The purpose of this thesis was to develop software that helps controlling employees' personal data and to create and print personal passports for a festival called Maata Näkyvissä. The employees are possible to add one at a time or with Microsoft Excel datasheet to database. With this software it is able to avoid printing the same personal passport twice.

This developed software will come in use of Maata Näkyvissä festival. The software will replace old software which has been used before to fill this job. The old software didn't serve its job well enough anymore. The new software is developed to ease this job and to make it done more quickly than before.

The technique that was used in implementation of this thesis was ADO.NET and programming language was C#. The software used SQLite-database engine as a database. This thesis contains programming, software testing, documenting and also database and user interface planning.

SISÄLLYS

1 TERMIT JA NIIDEN MÄÄRITELMÄT

Ohjelmakirjasto: Ohjelmakirjastot ovat kokoelmia aliohjelmia, luokkia tai ohjelmia, joita käytetään tietokoneohjelmien kehittämisessä sekä ohjelmien suorituksen aikana.

Henkilöpassi: Tässä opinnäytetyössä henkilöpassi tarkoittaa työntekijälle henkilökohtaista kulkulupapassia, josta käy ilmi työntekijän nimi, tehtävä festivaalien aikana sekä passin vahvuus.

Konekieli: Konekieli on tietokoneen suorittama kieli. Konekieli koostuu sarjasta konekielisiä käskyjä. Konekieliset käskyt koostuvat biteistä eli ykkösistä ja nollista.

SQL-92 standardi: Tämä standardi on SQL tietokantakielen standardi vuodelta 1992. Standardi määrää tietokantakielen syntaksin sekä toiminnallisuuden.

Standalone- sovellus: Standalone- sovellus on yleiskuvaus sovellukselle, joka ei toimi minkään sovelluksen sisällä, eikä lisänä, eikä myöskään tarvitse Internetiä toimiakseen.

Tietokantatapahtuma: Tarkoittaa tietokantaan kirjoittamista tai tietokannasta lukemista.

CLR: Common Language Runtime on Microsoft .NET –ympäristön ydinkomponentti. CLR kääntää ohjelman ajovaiheessa CIL -koodin käyttöjärjestelmälle sopivaksi.

DataSet: DataSet on xml-kielellä kirjoitettu välimuisti tietokantaan ja tietokannasta siirtyvälle datalle. Sitä käytetään tietokannan ja itse ohjelman välisenä rajapintana.

Tableadapter: Tableadapter toimii ohjelman ja tietokannan välissä. Se on yhteydessä tietokantaan ja suorittaa tietokantaan kyselyitä, jonka jälkeen se palauttaa tiedot tie-

tokannasta joko uutena datatauluna tai jo valmiiseen datatauluun. Tietokantaan päivitetään tietoja myös tableadapterin avulla.

Datagridview: Datagridview on erittäin käytännöllinen komponentti esittämään tietoja tietokannasta käyttäjälle. Sen avulla käyttäjä voi myös päivittää tietoja tietokantaan. Se on käytännössä taulukko, joka esittää kokonaisia tauluja tai niiden osia.

2 JOHDANTO

Opinnäytetyöni aiheena oli toteuttaa sovellus Maata Näkyvissä nimisen tapahtuman työntekijöiden tietojen hallitsemiseen ja heidän henkilöpassien tulostamiseen. Sovellus tuli Maata Näkyvissä henkilökunnan ja työntekijöiden käyttöön, jolloin ohjelmasta vaadittiin kaksi eri versiota. Versiot eroavat toisistaan lähinnä eri rajoituksilla. Tapahtuman työntekijöiden mahdollisuudet rajoittuvat passien tulostamiseen ja tietokannasta löytyvien työntekijöiden tietojen etsimiseen. Henkilökunnan versiossa on täydet oikeudet työntekijöiden lisäämiseen ja poistamiseen sekä henkilötietojen muokkaamiseen. Lisäksi henkilöpasseja oli tarve saada tulostettua Adoben pdf-muotoon, jolloin henkilöpassit voitiin viedä tulostettavaksi mihin tahansa. Pdf-tulostaminen onnistuu vain henkilökunnan versiossa.

Laatimani sovellus tuli vanhan samaa tehtävää hoitavan ohjelman tilalle. Tein sovelluksen helpottamaan vanhan sovelluksen ongelmia ja yksinkertaisuutta. Uusi sovellus tarjoaa huomattavan määrän parannuksia ja palvelee käyttäjää työn helppoudella ja nopeudella. Opinnäytetyössä ei ole hyödynnetty vanhaa ohjelmaa millään tavalla.
Olen tehnyt ja suunnitellut sovelluksen täysin itse. Opinnäytetyö oli tarkoitus saada valmiiksi kuudessa kuukaudessa. Projektin ensimmäinen kuukausi menikin kokonaan tarvittavien tietojen hankkimiseen ja tulevan ohjelman käyttäjien haastattelemiseen. Koko sovelluksen laatiminen yksin on hyvin haastavaa, koska ohjelmointiympäristöjä on tällä hetkellä todella kattava määrä tarjolla ja tehtävänä oli valita juuri tähän sopiva ratkaisu. Käsittelen tekemiäni ohjelmointiympäristön päätöksiä luvussa 3.

Päätettyäni ohjelmointiympäristön aloin suunnittelemaan tietokantaa ja sovelluksen käyttöliittymää, sekä ulkoasua. Tietokannan suunnittelua ja siihen liittyviä ongelmia käsittelen luvussa 3.2 ja käyttöliittymää ja sen ulkoasua luvussa 3.4. Tämän jälkeen aloin kehittää itse sovellusta.

3 KÄYTETYT TEKNIIKAT

Koska suunnittelin ohjelman täysin itse, oli käytetyissä tekniikoissa valinnanvaraa runsaasti. Tässä luvussa esittelen käyttämäni tekniikat ja perustelen miksi päädyin juuri näihin tekniikoihin.

3.1 ADO.NET

ADO.NET on yksi osa .NET Framework ohjelmointiympäristöä. ADO.NET koostuu erilaisista luokista joiden avulla päästään muun muassa käsiksi tiedonlähteisiin, kuten tässä tapauksessa tietokantaan. Näin tietokantaa pystytään helposti täyttämään, päivittämään rivejä ja poistamaan rivejä.

ADO.NET ei sido kehittäjää tiettyyn ohjelmointikieleen vaan antaa vapauden käyttää muita .NET- sovelluskehykseen yhteensopivia kieliä. ADO.NET-sovellukset käännetään CIL- tavukoodiksi (Common Intermediate Language), jonka jälkeen CLR kääntää tavukoodin sovelluksen suorituksen aikana konekielelle. .NET-sovelluskehyksen ajoympäristöön eli CLR:n kuuluvista ohjelmointikielistä tässä opinnäytetyössä on käytetty C#- ohjelmointikieltä. (Msdn 2010a; Msdn 2010b)

Valitsin ADO.NET- ohjelmointiympäristön ja C#- ohjelmointikielen yksinkertaisesti sen takia, koska se oli minulle jo ennestään tuttu. Lisäksi tiesin mitä yhdistelmällä pystytään tekemään, jolloin asiakkaalle on helpompi ehdottaa ja luvata mitä mahdollisuuksia uusi ohjelma voisi tuoda mukanaan.

3.2 SQLite- tietokantajärjestelmä

SQLite on pieni ja nopea relaatiotietokantajärjestelmä. Sen käyttämiseen ei tarvita tietokantapalvelinta niin kuin moniin muihin tietokantajärjestelmiin. Lisäksi SQLite tukee lähes kokonaan SQL-92-standardia. SQLiten moottori on ANSI C:llä kirjoitettu ohjelmakirjasto ja se käyttää vain seitsemää C-kielen funktiota perusohjelmakirjastosta. Ohjelmakirjastona SQLite voidaan helposti kiinnittää dynaamisesti sitä ympäröivään sovellukseen. SQLite- tietokanta on yksittäinen tiedosto, joka sisältää tie-

tokannan rakenteen, tietokannan käyttöä ohjaavat asetukset sekä tietokantaan tallennetut tiedot. Erittäin tärkeänä etuna verrattuna moniin muihin tietokantajärjestelmiin SQLite on täysin ilmainen yksityisiin ja kaupallisiin tarkoituksiin. (SQLite.org 2010a)

SQLite- tietokantajärjestelmä soveltuu erittäin hyvin keski- ja pienikokoisiin sovelluksiin. Mikäli suoritettavana on hyvin suuri määrä tietokantatapahtumia ja suuri määrä tietoa, SQLite ei ole paras mahdollinen vaihtoehto. SQLiten heikkoutena onkin sen lukitseminen tietokantaan tehtävien muutosten ajaksi. Eli kun tietokantaan lisätään, muokataan tai poistetaan tietoja, koko tietokanta lukitaan niin, ettei kukaan muu pääse sieltä hakemaan tai kirjoittamaan tietoja. Monissa muissa tietokantajärjestelmissä on mahdollista lukita vain ne taulut, joita muutokset koskevat. Tietokannan osittainen lukitseminen on huomattavasti nopeampaa ja näin myös tehokkaampaa.(SQLite.org 2010b)

Koska opinnäytetyöni sovellus oli tarkoitus toimia Standalone - sovelluksena, on ajo ilman tietokantapalvelinta lähes välttämätöntä. Vaikka SQLite jäädyttääkin tietokannan muutosten ajaksi, ei sillä tähän tarkoitukseen ole juurikaan merkitystä, koska samanaikaista käyttöä ei käytännössä ole lainkaan, eivätkä suoritettavat tietokantatapahtumat ole nimeksikään pitkiä. Koska asiakas piti sovelluksen edullisuutta tärkeänä, oli SQLite täydellinen valinta opinnäytetyön sovellukseen.

3.3 iTextSharp

iTextSharp on käännös iText- nimisestä Java- kirjastosta käännettynä C#- kielelle. iText on kirjasto, jonka avulla pystytään lennossa tekemään Adoben .pdf- muotoisia dokumentteja. Sitä ei ole tarkoitettu loppukäyttäjälle, vaan sillä on tarkoitus muodostaa .pdf- tiedostot dynaamisesti sen ympärillä olevan ohjelman sisällä. Sillä voidaan tehdä automaattisesti hyvin pitkälti samoja toimintoja, joita Adoben omilla työkaluilla pystytään tekemään.(iTextpdf 2010)

iText on erittäin käytännöllinen kirjasto, koska usein juuri halutaan automatisoida pdf- dokumentin tekeminen. Sillä voidaan tehdä esimerkiksi valmiita pohjia ja täytet-

täviä kaavakkeita, sekä täyttää niitä automaattisesti. Automatisoidut tiedostot halutaankin usein lähettää selaimelle. iTextSharp osoittautuukin melko hyväksi, mutta todella huonona puolena on dynaamisen dokumentin tulostamisen puuttuminen. Koska tällä luokkakirjastolla tiedoston tulostaminen ei ole mahdollista, on tehtävään pakko käyttää ulkoista ohjelmaa.(Bruno Lowagie 2006)

Päätin käyttää iTextSharp- kirjastoa sen hyvien ominaisuuksien vuoksi. Oli vaikea löytää muita samoilla tai paremmilla ominaisuuksilla olevaan kirjastoa tähän tarkoitukseen. Saatavilla oli suuri määrä kokonaisia ohjelmistoja, joilla saman tehtävän olisi pystynyt tekemään. Yritin kuitenkin tässä opinnäytetyössä sisällyttää kaikki tarvittavat komponentit oman ohjelmani sisällä, niin hyvin kuin se oli mahdollista. Tämän vuoksi iTextSharp oli ehdottomasti paras vaihtoehto opinnäytetyöhöni.

4 SOVELLUKSEN SUUNNITTELU

4.1 Ohjelmiston suunnitteluun liittyvät ongelmat

Ohjelmiston suunnittelun alkuvaiheessa oli tärkeää saada loppukäyttäjän ajatus siitä, millainen sovelluksen tulisi olla ollakseen mahdollisimman pätevä tulevaan tehtäväänsä. Haastatellessani muutamia henkilöitä loppukäyttäjistä, jotka ovat käyttäneet vanhaa samaan tehtävään tarkoitettua ohjelmaa, tuli esille muun muassa seuraavia uudistusehdotuksia:

- Ohjelman täytyy olla todella helppokäyttöinen ja nopeasti opittava,
- ohjelman sisällä täytyy pystyä siirtymään tabulaattorin avulla objektista toiseen, jotta se on nopea käyttää,
- mitä nopeammin passeja saadaan tulostettua sen parempi,
- ohjelmasta täytyy käydä ilmi onko henkilölle tulostettu ennen passia,
- jokaisen tietokannassa olevan henkilön vastaava henkilö tulisi näkyä ohjelman käyttäjälle,
- passit täytyy pystyä tulostamaan joko neljä passia yhdellä sivulla tai yhdeksän passia yhdellä sivulla,

- tulostettavat passit tulee voida tallentaa pdf- muotoon, jotta ne voidaan tulostaa jossain muualla,
- sekä työntekijät lisätään tietokantaan Microsoft Excel- taulukosta niin, että sovellus lukee työntekijät annetusta taulukosta.

Saatuani selville uudelta ohjelmalta vaaditut uudistukset, otin selvää, että uudistukset olivat mahdollista toteuttaa. Seuraavaksi valitsin käytettävät työkalut, jotka sopivat parhaiten sovelluksen kehittämiseen.

4.2 Tietokannan suunnittelu ja toteutus sekä siihen liittyvät ongelmat

Tietokannan suunnittelun alkuvaiheessa valitsin itselleni apuohjelman tietokannan luomiseen. SQLiten kehittäjien sivuilta löytyi helppokäyttöinen ja ilmainen ohjelma tietokannan luomiseen ja muokkaamiseen: SQLite2009 Pro Enterprise Manager.

4.2.1 SQLite2009 Pro Enterprise Manager

SQLite2009 Pro Enterprise Manager on erinomainen työkalu SQLite3-tietokantojen luomiseen ja muokkaamiseen. Se sisältää muun muassa seuraavia asioita:

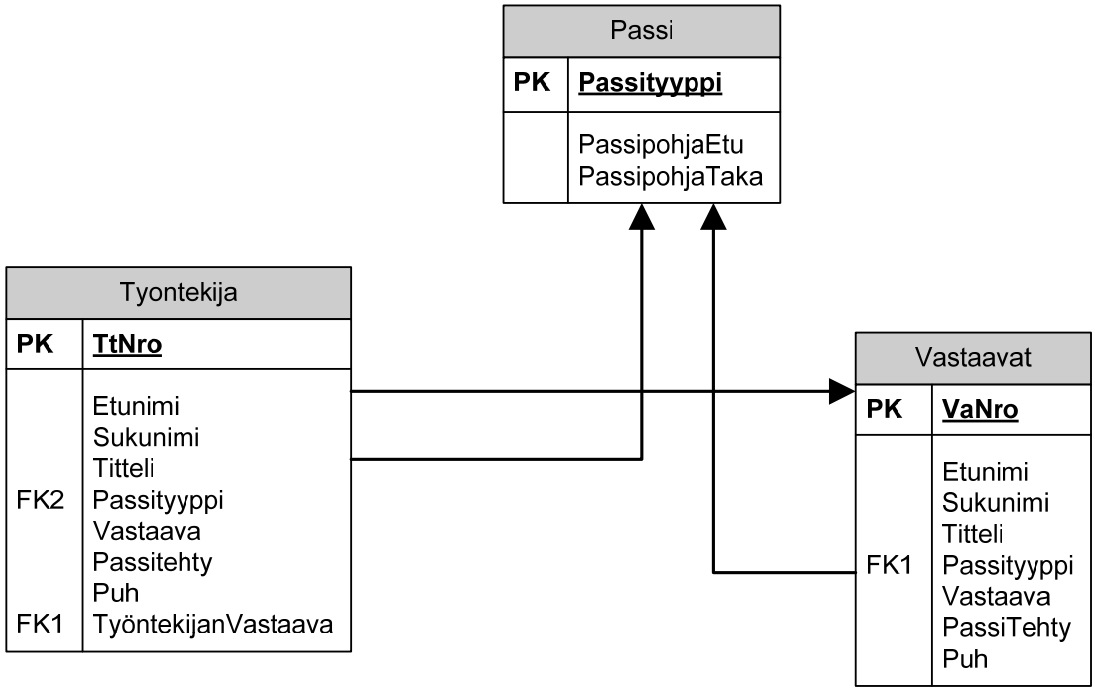
- Visuaalisen apuvälineen kyselyiden tuottamiseen,
- heksadesimaaliluku-näyttäjä, jolla on mahdollista katsoa tietokannassa olevaa dataa heksadesimaalilukutaulukkona,
- kuvan tai blob- tiedoston näyttäjä,
- tietokanta on mahdollista salata vxSQLite3 (AES-128 bittiä) tekniikalla tai SQLite3 ADO.NET Provider- (RSA-MS Crypt) yhteensopivalla tekniikalla,
- tietokannan tietuejoukko on mahdollista viedä Microsoft Excel- taulukkoon tai kääntää xml- tai html- muotoon,
- sekä tietokantaan voidaan tuoda tietoja suoraan MS Access, MS SQL- server ja MySQL-server tietokannoista.

SQLite2009 Pro Enterprise Manager antaa myös mahdollisuuden työskennellä Lua-ohjelmointikielen kanssa.(SQLiteManagementTools)

Koska itse koodauksen apuvälineenä toimii Microsoft Visual Studio 2008 ja ympäristönä ADO.NET, on käytettävä SQLite3 ADO.NET Provider- työkalua tietokannan liittämiseen sovellukseen. Myös tietokannan salaus tehtiin SQLite3 ADO.NET provider- yhteensopivalla salauksella.

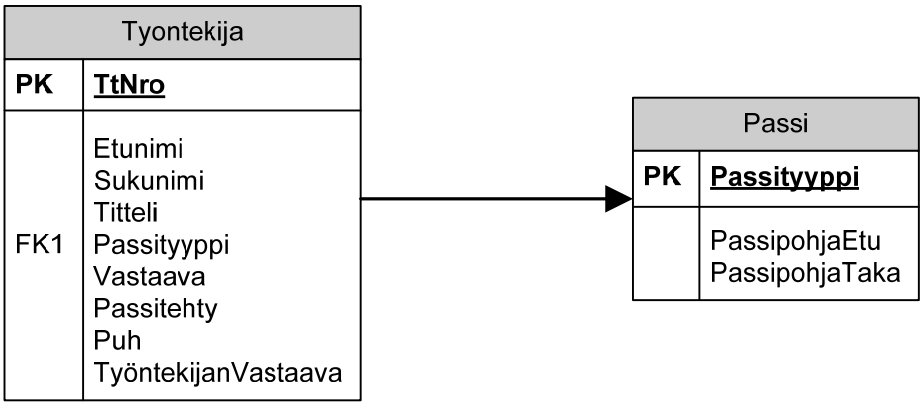
4.2.2 Tietokantataulujen ja tietotyyppien suunnittelu ja toteutus

Suunniteltuani tietokannan ensin paperille, huomasin että tarvittavia tietokantatauluja tulee vain kolme kappaletta. Tietokantataulut asettuivat kuvan mukaisesti.



Kuva 1 Tietokannan taulujen kuvaus työn alussa

Tietokanta on hyvin suppea ja pieni, joten sen nopeaan tuottamiseen ei kulunut montaa tuntia. Myöhemmin kuitenkin tietokantaan tuli iso muutos, joka helpotti ratkaisevasti ohjelmiston toteutuksessa. Muutos koski käytännössä Vastaavat taulua, joka poistui kokonaan. Näin tietokannasta tuli hyvin yksinkertainen kahden taulun tietokanta kuvan 2 mukaisesti.



Kuva 2 Lopullinen tietokannan kuvaus

Lopulliseen tietokantaan yhdistettiin Tyontekija- ja Vastaavat taulu niin, että Tyontekijat-taulun sarake TyontekijanVastaavat muutti merkityksensä. Jokaisella työntekijällä on vastaava ja jokaisella vastaavalla on myös vastaava. Näin ollen Vastaavat-taulua ei tarvita.

4.2.3 Tietokantakaavat ja tietokannan luominen

Tein tietokantakaavat yksinkertaisesti tekstinkäsittelyohjelmalla, koska on huomattavasti helpompi kopioida ohjelmasta, joka ei sisällä automaattista muotoilua tekstille. Tietokantakaavojen avulla luodaan tietokantataulut, taulujen sarakkeet, tietueiden tietotyypit, taulujen pää- ja viiteavaimet sekä null- arvot. Suunniteltaessa tietokantaa on mietittävä tarkkaan mitä tarvitaan ja mikä on liikaa. Jokaisella taulun sarakkeella on tietotyyppi, joka on rajattu sen tarpeiden mukaisesti. Luomani tietokantakaavat näyttävät seuraavalta:

```
CREATE TABLE TYONTEKIJA(
TtNro INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
Etunimi VARCHAR(30),
Sukunimi VARCHAR(30),
Titteli VARCHAR(30) NOT NULL,
Passityyppi VARCHAR(10) NOT NULL,
Vastaava VARCHAR(10),
PassiTehty VARCHAR(10) NOT NULL,
```

```
Puh VARCHAR(20) NOT NULL,
TyontekijanVastaava VARCHAR(30),
CONSTRAINT FK_TYONTEKIJA_PASSI
FOREIGN KEY (Passityyppi) REFERENCES PASSI(Passityyppi));

CREATE TABLE PASSI(
Passityyppi VARCHAR(10) NOT NULL,
Passipohja BLOB NOT NULL,
CONSTRAINT PK_PASSI PRIMARY KEY (Passityyppi);
```

Näillä tietokantakaavoilla pystytään luomaan tietokanta. Käytännössä olen tehnyt tietokantakuvauksen mukaisesti kaksi taulua, tyontekija ja passi. Jokaisella taululla tietokannassa täytyy olla yksi yksilöivä sarake, joka ei voi saada samoja arvoja. Tämä sarake ei voi myöskään saada null- arvoa, joka tarkoittaa tyhjää, mutta ei kuitenkaan nollaa. Tämän tyyppinen sarake asetetaan usein pääavaimeksi, koska se on yksilöivä.

Eri tietotyyppejä on lukuisia määriä. Tässä kuitenkin on tarvittu vain kolmea erilaista tietotyyppiä. Integer- tietotyyppi tarkoittaa lukua ja varchar merkkijonoa. Varchar-tietotyypin perään merkataan sulkuihin, kuinka monta merkkiä on mahdollista tallentaa. Tämä rajoittaa käyttöä, mutta samalla pitää tietokannan pienenä. Erityinen huomion kohde on tietotyyppi blob. Blob- tietotyyppi on kasa tietoa, joka on varastoitu tietokantaan juuri sellaisena, kuin se on sinne laitettukin. Tässä tapauksessa blob-tietotyyppiä käytetään kuvien tallentamiseen tietokantaan.(SQLite2010c)

Työntekija- taulussa TtNro on taulun pääavaimena ja sillä on ominaisuutena autoincrement. Tämä tarkoittaa automaattisesti nousevaa lukua. Eli aina kun uusi työntekijä lisätään tietokantaan, asetetaan sille uusi työntekijänumero, joka on yhden suurempi kuin suurin edellinen työntekijänumero. Tietokantakaavoissa on merkattuna vielä tyontekija-taulun viiteavain jolla se on liitetty passi-tauluun. Loin tietokannan pohjan SQLite2009 Pro Enterprise Manager-ohjelmalla. Tietokannan luontivaiheessa lisäsin sinne myös vähän testaukseen tarvittavaa dataa.

4.3 Sovelluksen koodin suunnittelu

Tässä luvussa käsittelen sovelluksen koodaustavan suunnittelua, sekä ratkaisen muutamia koodaukseen liittyviä valintoja. Esittelen myös tapani tehdä opinnäytetyöni kaltaisia sovelluksia.

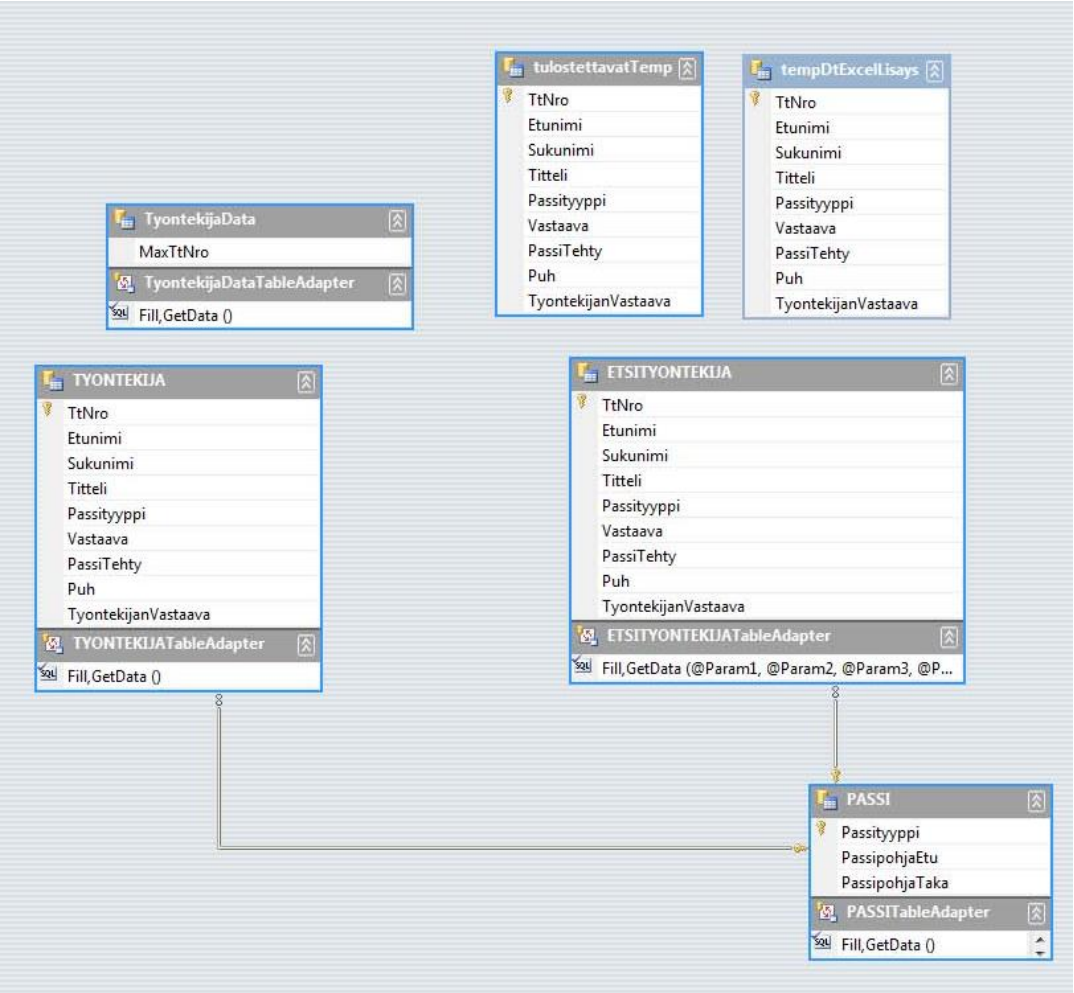
4.3.1 Olioajattelu

Lähdin toteuttamaan sovellusta olioajattelun mukaisesti selvittämällä millaisia olioita ja olioiden ominaisuuksia työni sisältää. Esimerkiksi työntekijästä tulisi hyvä olio, koska olihan tietokannassa suunniteltu Tyontekijat- taulu. Tyontekijat- olion ominaisuuksiksi tuli kaikki taulun sarakkeet. Kun olio on yhtenevä tietokantataulun kanssa, on niitä helppo käyttää keskenään. Näin tietokantaan lisättäviä henkilöitä on hyvin helppo käsitellä olioina.

4.3.2 Data Set:n käyttö

Suunnitellessani sovelluksen ja tietokannan rajapintaa huomasin, että muutamalle uudelle datataululle olisi käyttöä. Sovelluksen DataSet näyttää kuvan 3 kaltaiselta.

Tein neljä datataulua ja kahdelle niistä tableadapterit. Data-taulujen avulla saan näkymät halutusta tiedosta datagridview- kontrollille, jonka avulla näytän tiedot käyttäjälle. Toiseen tauluun kerätään henkilöt, jotka ovat menossa tulostettavaksi ja toiseen kerään henkilöt, jotka ovat lisätty Microsoft Excel taulukosta ja ovat menossa tietokantaan. Kahdessa muussa taulussa kerään tietokannasta etsittäviä tietoja. Toisessa haetaan suurin työntekijänumero, lisättäessä uutta työntekijää ja toisessa toimii työntekijöiden hakuprosessi.



Kuva 3 Dataset:n taulut ja niiden sarakkeet sekä taulujen väliset liitokset

4.3.3 Tarvittavat vakiot ja niiden esittely

Sovellusta tehdessäni olen yleensä yrittänyt selvittää kaikki sovelluksen vakiot ja määrittänyt ne johonkin omaan paikkaansa. Vakio on arvo, joka ei muutu koko ohjelman ajamisen aikana. C#- kielessä vakion määrittäminen onnistuu esittelyvaiheessa const- etuliitteellä. Vakioiden määrittäminen helpottaa myös ohjelman testaamista, koska arvo ei voi muuttua ohjelman käytön aikana.

4.3.4 Tietojen haku Microsoft Excel- dokumentista tietokantaan

Tietojen siirtämiseen Excel- dokumentista tietokantaan voitiin tehdä ainakin kolmella eri tavalla. Tehtävänä olikin etsiä juuri tähän tarkoitukseen sopivin tapa. Päätin

käyttää Excelin omaa kirjastoa, jonka liitin sovellukseen. Toinen vähintään yhtä tehokas tapa on kääntää Excel- dokumentti xml-muotoon ja lukea tiedot siitä. Halusin kuitenkin lukutavan olevan mahdollisimman yksinkertainen ja nopea. Testasin Excel- kirjaston käytännössä ja totesin sen riittävän tehokkaaksi keinoksi siirtää tekstityyppistä dataa. Mikäli lisättävä Excel- tiedosto olisi sisältänyt erityyppisiä tietoja ja esimerkiksi laskukaavoja ja makroja, olisi tehtävä ollut moninkertaisesti haastavampi. Luulisinkin, että tällaisessa tapauksessa olisin käyttänyt xml-muotoa hyväkseni.

4.4 Sovelluksen ulkoasun suunnittelu

Lähdin suunnittelemaan sovelluksen ulkoasua silmällä pitäen sen helppokäyttöisyyttä ja viitteellisyyttä. Sovelluksen yhtenä toimivuusehtona oli tehdä siitä mahdollisimman helppokäyttöinen. Päättelin, että sovelluksessa tulisi olemaan komponentteja, joiden täytyi näkyä kokoajan ja komponentteja, joita tarvittiin vain, kun niitä käytettäisiin. Päätin tehdä sovelluksen eri toiminnot välilehdille, koska ne ovat hyvin helppokäyttöisiä ja ymmärrettäviä. Välilehtiö ei peitä koko ikkunaa vaan sovelluksen oikeassa reunassa on aina näkyvissä tarpeelliset tiedot ja toiminnot. Suunniteltuani ohjelman ulkoasun, lopputulos näytti liitteen 1 kaltaiselta.

Suunnitteilla oli ensiksi tehdä ohjelmalle etusivu joka tulisi aina näkyviin ohjelman käynnistyessä. Huomasin kuitenkin, ettei se palvellut mitään tai ketään, jonka jälkeen päätin jättää sen kokonaan pois. Ohjelma meneekin suoraan välilehtiön ensimmäiselle sivulle, josta pääsee suoraan töihin.

5 MNPASSIOHJELMA2010

Sovelluksen oli tarkoitus hoitaa neljä tehtävää: Henkilötietojen lisääminen, muokkaaminen ja poistaminen, sekä henkilöpassien tulostaminen. Tässä luvussa käyn läpi näiden tehtävien tapahtumat ja toiminnan.

5.1 Henkilötietojen lisääminen

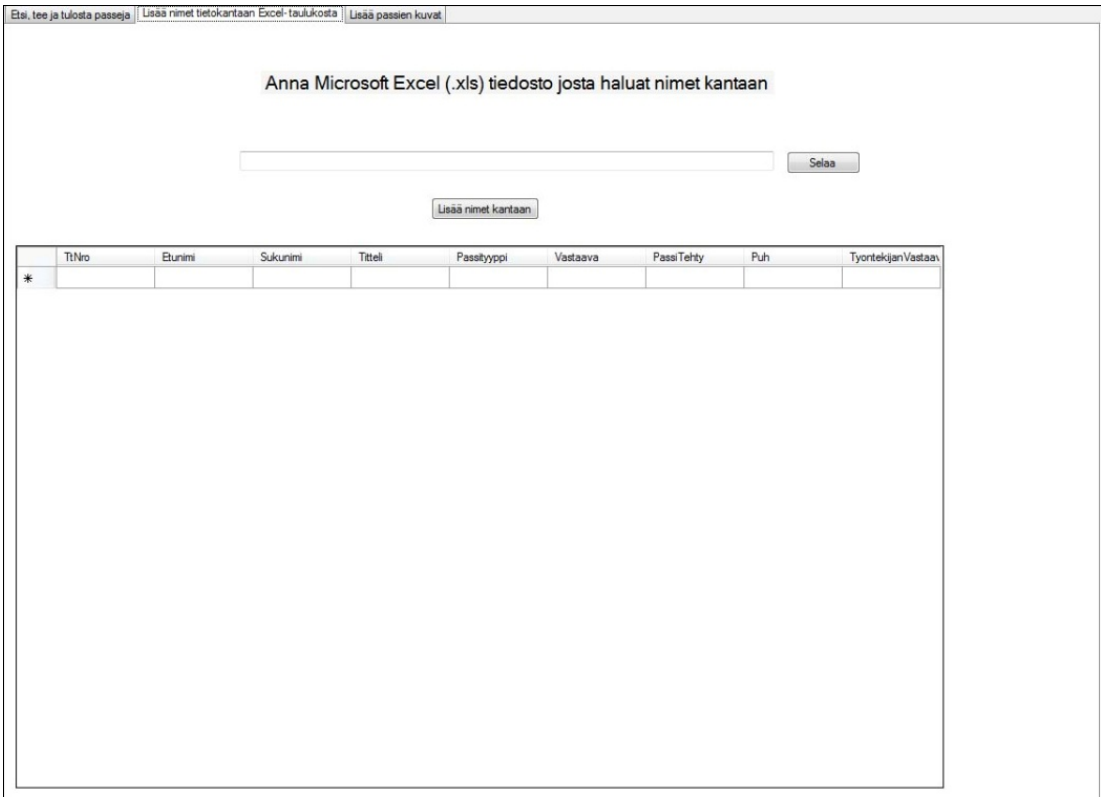
Henkilötietoja oli tarve pystyä lisäämään kahdella eri tavalla: Henkilöiden lisääminen tietokantaan Microsoft Excel- taulukko tietolähteenä ja henkilöiden lisääminen käsin kirjoitettuna.

5.1.1 Henkilötietojen lisääminen Microsoft Excel- taulukosta

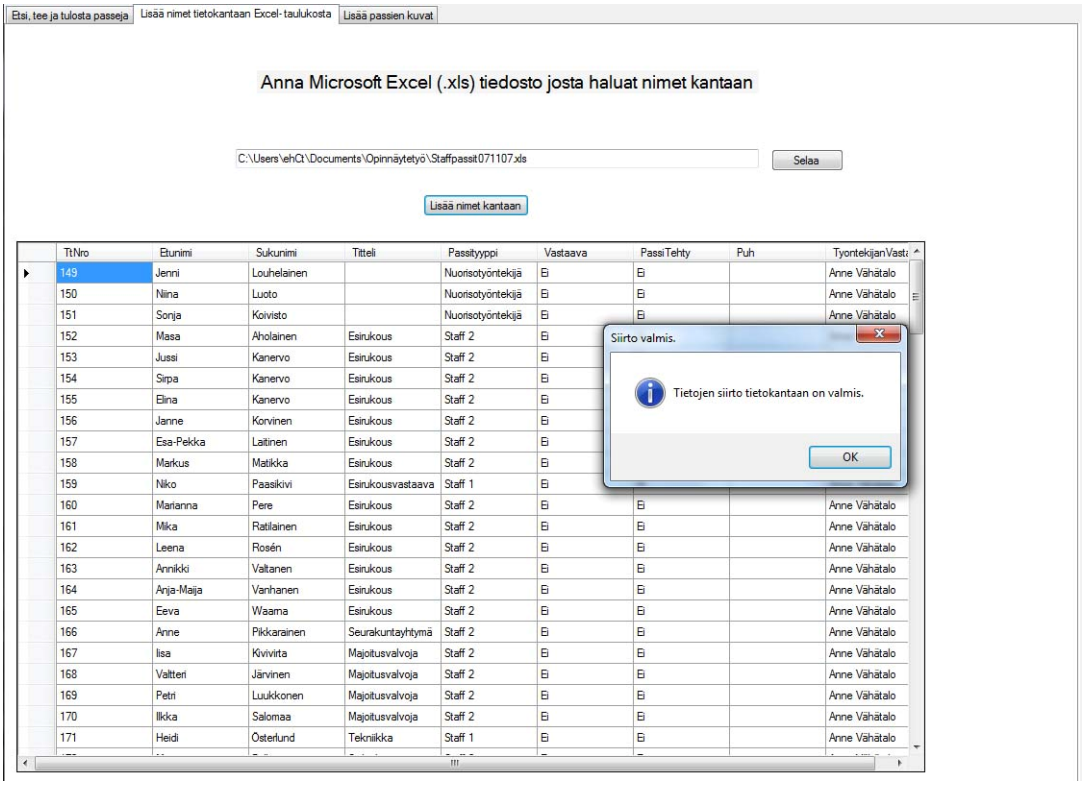
Päätin suunnitelmassani käyttää Microsoft.Office.Core- ja Microsoft.Office-Interop.Excel- kirjastoja annetun Excel- taulukon lukemiseen. Tein itselleni Excel -taulukkomallin, jonka avulla kokeilin koodin toimivuutta. Taulukkomallista tulikin pohja sovelluksen käyttäjälle täytettäväksi vuosittain, josta henkilötiedot siirretään kantaan. Excel- taulukkomallin täytyy olla aina samanlainen, jotta ohjelma toimisi. Tästä johtuen lukitsin taulukosta kaikki muut solut paitsi ne, joita täytyy päästä muokkaamaan. Käytännössä tapahtuma näyttää kuvan 4 ja kuvan 5 kaltaiselta.

Haluttu Excel taulukko annetaan painamalla Selaa, jolloin ohjelma pyytää näyttämään tiedoston sijainnin. Kun haluttu taulukko on annettu ja painettu lisää nimet kantaan- painiketta, löydetyt nimet tulevat taulukkona näkyviin kuvan 3 kaltaisesti. Tiedonsiirron ollessa valmis tulee näyttöön lukemaan ”Tietojen siirto tietokantaan on valmis”.

Tietojen esittäminen näytöllä on hidasta, mutta joissain tapauksissa pakollista. Olikin pitkään suunnitteilla, että tietojen siirrosta tulee näkyviin vain palkki joka näyttää prosessin kulun. On kuitenkin mielestäni huomattavasti havainnollisempaa ohjelman käyttäjälle, että hän näkee käytännössä, mitä tietoja on siirtynyt tietokantaan. Taulukosta oli myös erittäin suuri apu ohjelman testausvaiheessa.



Kuva 4 Näkymä Excel–taulukon lisäämisestä kantaan



Kuva 5 Näkymä hyväksytystä tietojen siirrosta tietokantaan

5.1.2 Henkilötietojen lisääminen tietokantaan manuaalisesti

Tietokantaan oli tarve päästä muuttamaan tietoja myös manuaalisesti. Yhdistin kolme toimintoa yhdelle sivulle sovellukseen: Etsi, tee ja muokkaa henkilöitä. Henkilön lisääminen tapahtuu käytännössä kirjoittamalla henkilön tiedot niille osoitetuille paikoille ja painamalla ”Lisää uusi työntekijä”-painiketta. Tämän toiminnon jälkeen henkilö löytyy tietokannasta.

5.2 Henkilötietojen etsiminen ja muokkaaminen tietokantaan

Työntekijän tietojen etsiminen tietokannasta on hyvin yksinkertainen tapahtuma. Etsitystä henkilöstä annetaan mikä tahansa tieto, mitä henkilöstä tiedetään. Ohjelma hakee viereiseen taulukkoon lähimpänä hakusanoja olevat henkilöt automaattisesti. Mikäli etsittyä henkilöä ei löydy tietokannasta, se voidaan sinne lisätä kirjoittamalla henkilön tiedot loppuun ja painamalla ”Lisää uusi työntekijä”-painiketta.

Henkilötietojen muokkaaminen on tehty hyvin yksinkertaiseksi. Ensin etsitään työntekijä, jota halutaan muokata. Kun haluttu henkilö on löydetty, pystytään henkilön kaikkia muita tietoja muuttamaan paitsi työntekijänumeroa suoraan taulukkoon, johon etsinnän tulokset tulevat. Muutokset tallentuvat tietokantaan, kun muutosten tekeminen on lopetettu.

5.3 Henkilöpassien tulostaminen

Henkilöpassien tulostaminen onnistuu raahaamalla hakutaulukosta henkilöitä tulostustaulukkoon. Henkilöitä voidaan tulostaa lähes rajaton määrä. Ohjelma asemoi passit automaattisesti ja tekee uusia sivuja niin kauan kuin tulostettavia passeja riittää. Käytännössä ohjelma tekee passeista Adoben pdf- tiedoston, jonka se tallentaa haluttuun paikkaan myöhempää tarkastelua varten. Tulostettavaa passitiedostoa on mahdollista esikatsella ennen tulostamista, jonka avulla pystytään varmistamaan, että kaikki tulostettavat komponentit ovat halutuilla paikoillaan. Kun henkilön passi on

tulostettu, asettaa ohjelma kullekin tulostetulle henkilölle passin tulostetuksi tietokantaan. Tämä on tärkeää, ettei passeja tulosteta monia kappaleita.

Henkilöpasseja oli tarve pystyä muodostamaan myös yhdeksän passin pdf- tiedosto. Kyseinen tulostus toimii eri nappulasta, kuitenkin muuten täsmälleen samalla tavalla. Ohjelma kysyy käyttäjältä mihin uusi tiedosto tallennetaan. Ohjelma ei tulosta tiedostoa automaattisesti. Tehdyt tiedostot on tarkoitus viedä kopiointifirmoille tulostettavaksi.

5.4 Henkilöpassien lisääminen, muokkaaminen ja poistaminen tietokantaan

Koettiin tarpeelliseksi, että passeja täytyisi pystyä lisäämään tietokantaan. Passien tiedot ja kuvat saattavat muuttua vuosittain, jolloin passeja pitää pystyä lisäämään, muokkaamaan ja poistamaan. Passien lisääminen onnistuu painamalla tähden kuvaa passityyppitaulukosta. Passityyppitaulukko kuvan 6 kaltainen. Tällöin taulukkoon tulee uusi rivi, jolloin passille voi kirjoittaa nimen. Kuvat voidaan lisätä kaksoisklikkaamalla passipohjaetu- tai passipohjatakasoluun, jolloin ohjelma kysyy vahvistuksen muutokseen. Vastattaessa kyllä, ohjelma pyytää hakemaan kansiodialogista halutun passin kuvan. Tämän jälkeen passipohja on tallennettu tietokantaan ja siitä on nähtävissä pieni layoutkuva taulukossa.

Passien muokkaaminen voidaan tehdä kaksoisklikkaamalla muokattavaa passia tai solua. Tapahtuman jälkeen ohjelma kysyy vahvistuksen muutokseen, jonka jälkeen muokkaaminen on mahdollista. Jokaisesta passista pystytään muokkaamaan mitä tahansa tietoa.

Henkilöpassin poistaminen onnistuu valitsemalla taulukosta rivin ja painamalla delete-nappia näppäimistöstä. Passi poistuu välittömästi tietokannasta ja passien lista päivittyy ruudulle.

Kuva 6 Yleisnäkymä passitaulukosta johon on lisätty ylimääräinen passi

6 DOKUMENTOINTI JA TESTAUS

Tässä luvussa käsittelen sovelluksesta tehdyn dokumentoinnin sekä sovelluksen testaamisen eri muodot.

6.1 Sovelluksen dokumentointi

Dokumentoin sovelluksen kahdella eri tavalla. Yleinen dokumentaatio, joka toimii käyttöohjeena, sekä sovelluksen jatkokehittäjille tarkoitettu opas. Yleinen dokumentaatio menee loppukäyttäjille sellaisenaan.

Yleisessä käyttöohjeessa neuvotaan ohjelman toiminnallisuus, asennus, käyttöönotto, tietojen lisääminen sekä järjestelmän ylläpito. Suuri osa toiminnallisuudesta neuvotaan jo itse sovelluksessa.

Jatkokehittäjien opas on käytännössä dokumentoitu ohjelmakoodin joukkoon kommentteina. Jokaiseen luokkaan ja toiminnallisuuteen on kommentoitu mitä sillä tehdään. Vaikka kommentoiminen on hyvin työlästä, se on erittäin tärkeää sovelluksen jatkokehittämisen kannalta.

6.2 Sovelluksen toimivuuden testaus

Sovelluksen virheiden etsiminen ja niiden korjaaminen on vienyt hyvin paljon aikaa opinnäytetyöstäni. Tein itselleni hyvin yksinkertaisen testaussuunnitelman, jonka avulla halusin selventää testausta. Testaussuunnitelma sisälsi seuraavanlaisia asioita:

- Dynaaminen Black box- testaus,
- staattinen White box- testaus,
- testaus testihenkilöiden avulla.

Dynaaminen Black box testaus tarkoittaa sovelluksen testaamista niin, ettei ohjelman koodia tunneta. Sovellusta testataan suorittamalla sitä useaan kertaan. Black box -testaukseen kuuluu myös apinatestit, tyhjät arvot sekä syöteavaruuden analysoiminen.(Patton Ron 2005a)

Suoritin Black box – testauksen muutaman koehenkilön avulla, koska en tietenkään itse voinut testausta tehdä. Ongelmana on usein myös se, että testatessa tulee omille virheilleen sokeaksi, jonka takia usein käytetään ulkoisia testaajia. Suurin osa sovelluksen ongelmista löytyikin juuri tällä keinolla.

Staattinen White box testaus tarkoittaa sovelluksen testaamista niin, että ohjelmakoodi tunnetaan. Ohjelmadokumentteja tutkitaan ja tarkastetaan lukemalla suoraan koodia. Testaajan tulee tuntea ohjelmakoodi, jolloin on helpompaa huomata, että oh-

jelma tekee juuri sen mitä on ohjelmoitu. Sovelluksesta pyritään löytämään ongelmia analysoimalla ohjelmalogiikkaa.(Patton Ron 2005b)

Koska teen koko sovelluksen itsenäisesti opinnäytetyönäni, olin ainoa joka pystyi tekemään staattisen White box – testaamisen. Teinkin testausta kokoajan samalla, kun koodasin sovellusta. Yhden kokonaisuuden jälkeen testasin ohjelmaa ja pyrein korjaamaan kaikki esille tulleet virheet.

Testasin sovellusta kolmessa eri tehoisella tietokoneella. Testattavien tietokoneiden suoritusnopeudet olivat hieman toisistaan poikkeavia testattaessa opinnäytetyöni sovellusta. Tietokone, jolla kehitin sovelluksen sisälsi vanhahkon Intel Core2 4300 prosessorin, 4Gt:n DDR2-keskusmuistia ja 64-bittisen Windows 7 Professional-käyttöjärjestelmän. Lisäksi testasin sovellusta Intel Core2 E8500 4Gt DDR2-keskusmuistilla 32-bittisellä Windows Vista SP1-käyttöjärjestelmä ja huomattavasti hitaammalla AMD Athlon XP 2400+-prosessorilla, 1Gt:n DDR-keskusmuistilla ja 32-bittisellä Windows XP SP3-käyttöjärjestelmällä varustetulla koneella. Jokaisella testattavalla koneella sovellukseni toimi riittävän nopeasti. Vasta suurien passitiedostojen käsittelyssä tuli hitaimmalla koneella hiipumisen merkkejä. Olen kuitenkin hyvin tyytyväinen sovelluksen suoritusnopeuteen. Suoritusnopeus oli tärkeä siksi, koska sovelluksen piti toimia hieman keskivertoa hitaammallakin tietokoneella.

Testasin sovellukseni helppokäyttöisyyttä muutaman testihenkilön avulla. Testihenkilöt eivät olleet käyttäneet vanhaa sovellusta, jonka opinnäytetyöni korvaa, joten he eivät osanneet kommentoida onko ohjelma helppokäyttöisempi kuin vanha ohjelma. Kaikki testihenkilöt totesivat ohjelman helppokäyttöiseksi ja yksinkertaiseksi, mutta sain kuitenkin erinomaisia uudistusehdotuksia sovelluksen ulkonäköön. Sovellus sai myös suorituskyvystään hyvän arvosanan, mikä oli asiakkaalle hyvin tärkeää. Ainoana vaikeana ongelmana huomattiin kuitenkin tietokannan ylikirjoittaminen, joka käyttäjän täytyi tehdä käsin, kopioimalla tyhjä tietokanta vanhan tietokannan päälle. Lopuksi todettiin kuitenkin ongelman olevan niin pieni, ettei sille lähdetty etsimään parempaa ratkaisua, koska tietokanta ylikirjoitetaan kerran vuodessa.

Testaamisen aikana tuli enää nykyään harvoin esille tuleva ongelma. Ohjelman ikkuna, joka on huomattavasti leveämpi kuin korkea, oli hankala käsitellä näytöllä joka

oli kuvasuhteeltaan 3:4. Tein ohjelman isolla 24 tuuman laajakuvanäytöllä, joten en osannut ottaa kyseistä ongelmaa huomioon. Päätin ratkaista tämän ongelman tekemällä sovelluksesta kokonäyttösovelluksen. Tämän seurauksena huomasimme, että kokonäyttösovellus palvelisi jopa paremmin tapahtuman aikana käytettävässä ohjelmaversiossa, kuin ikkunasovellus. Tämä johtui siitä, ettei tietokoneella jolla passiohjelmaa käytetään, ollut tarkoitus päästä tekemään mitään muuta, kuin käyttämään passisovellusta. Tällöin pakotettu kokonäyttösovellus oli erinomainen ratkaisu.

7 SOVELLUKSEN KEHITYSIDEAT

Tehdessäni opinnäytetyöni sovellusta, tulee väkisinkin mietittyä sovellukselle kehitysideoita. Jos olisin itse saanut päättää millainen sovelluksesta olisi tullut, olisin ehdottomasti laittanut tietokannan palvelimelle, jolloin sitä olisi voinut käyttää mistä tahansa. Tällöin kaikki päivitykset olisivat menneet samaan tietokantaan. Tietokantapalvelimen käyttö olisi laajentanut jonkin verran opinnäytetyötäni. Tietokannan ollessa palvelimella, tulee helposti samanaikaisia tietokantakyselyjä eri käyttäjiltä, jolloin tulisi tehdä sovellus, joka järjestää kyselyt tulojärjestykseen ja suorittaa niitä heti kuin mahdollista. Uskon, että tämä uudistus tulee tapahtumaan vielä jossain vaiheessa ohjelman eliniän aikana.

Sovelluksen ylläpidettävyydessä on vielä parannettavaa. Jotta sovellus toimisi lähes täydellisesti, tulisi erilaisia passeja pystyä vielä lisäämään ja poistamaan sekä muokkaamaan erilaisia ulkoasuja eri passeille. Tätä ei kuitenkaan toistaiseksi vielä haluttu, koska opinnäytetyöni olisi paisunut mahdottomaksi tehdä kuuden kuukauden aikana. Tätä uudistusta ei kuitenkaan olisi vaikea lisätä tämän hetkiseen sovellukseen, mutta sen suunnitteleminen toimivaksi veisi kuitenkin jonkin verran aikaa.

Mikäli tietokanta olisi sijoitettu palvelimelle, olisi tietysti myös koko ohjelma voinut olla kokonaan verkkosovellus. Selainpohjaiset sovellukset ovat yleistyneet huomattavasti ja niiden mahdollisuudet ovat lähes rajattomat. Tämä vaatisi kuitenkin huomattavasti enemmän resursseja mitä nykyinen ohjelmisto. Mikäli tulevaisuudessa

lähtökohdat ovat erilaiset, pitäisin tätä erittäin hyvänä ratkaisuna juuri tämän tyyppiseen tehtävään.

8 YHTEENVETO

Opinnäytetyöni lopputulokseen olen suhteellisen tyytyväinen. Asiakkaan toivomukset tuli täytettyä hyvin ja ohjelma toimii vakaasti. On kuitenkin mahdotonta sanoa varmasti, toimiiko ohjelma täydellisesti tehtävässään, koska sovellusta käytetään pääosiltaan vasta vuoden 2010 lopussa jolloin festivaali järjestetään. Jos jotain haluaisin tehdä toisin, niin sovelluksen sekä koodin suoritustehoa voisi parantaa. Myös dokumentointia ja ajanhallintaa voisi parantaa.

Opinnäytetyöni tekemiseen kului hyvin paljon aikaa. Jo aloittaessani opinnäytetyötäni tiesin, että aikataulutus tuli olemaan hyvin tiukka. Saatuani suunnitteluosuuden tehtyä ja tietokannan ollessa valmis, huomasin, että aikataulutus tuli olemaan vieläkin tiukempi. Tämä johtui osaltaan suunnitelmien muuttumisessa työn tekemisen aikana. Päätin kuitenkin ottaa muutokset vastaan ja tehdä sovelluksesta juuri halutun laisen.

Olen hyvin tyytyväinen C#- kielen ohjelmointitaitojeni kehittymisestä. Vaikka osa opinnäytetyöni sovelluksesta olikin vanhan kertaamista, oli kuitenkin hyvin mielenkiintoista laajentaa tietoja ja taitoja tämän sovelluksen myötä. Erityisesti tietokannan hallitsemisesta ja suunnittelusta kerääntyneestä tietotaidosta opinnäytetyöni myötä, olen hyvin tyytyväinen. Opinnäytetyö oli kaikin puolin mielenkiintoinen ja haastava kokonaisuus.

LÄHTEET

Msdn 2010a ADO.NET [verkkodokumentti]. [viitattu 23.3.2010].
Saatavissa: http://msdn.microsoft.com/en-us/library/aa286484.aspx

Msdn 2010b Managed Execution process [verkkodokumentti].[viitattu 2.6.2010]
Saatavissa: http://msdn.microsoft.com/en-us/library/k5532s8a.aspx

SQLite 2010a About SQLite [verkkodokumentti]. [viitattu 23.3.2010]
Saatavissa: http://www.sqlite.org/about.html

SQLite 2010b SQLite Is Serverless [verkkodokumentti].[viitattu 5.4.2010]
Saatavissa: http://www.sqlite.org/serverless.html

SQLite 2010c Datatypes In SQLite Version 3 [verkkodokumentti]. [viitattu 5.4.2010]
Saatavissa: http://www.sqlite.org/datatype3.html

SQLiteManagementTools Active and Supported [verkkodokumentti]. [viitattu 30.3.2010]
Saatavissa: http://www.sqlite.org/cvstrac/wiki?p=ManagementTools

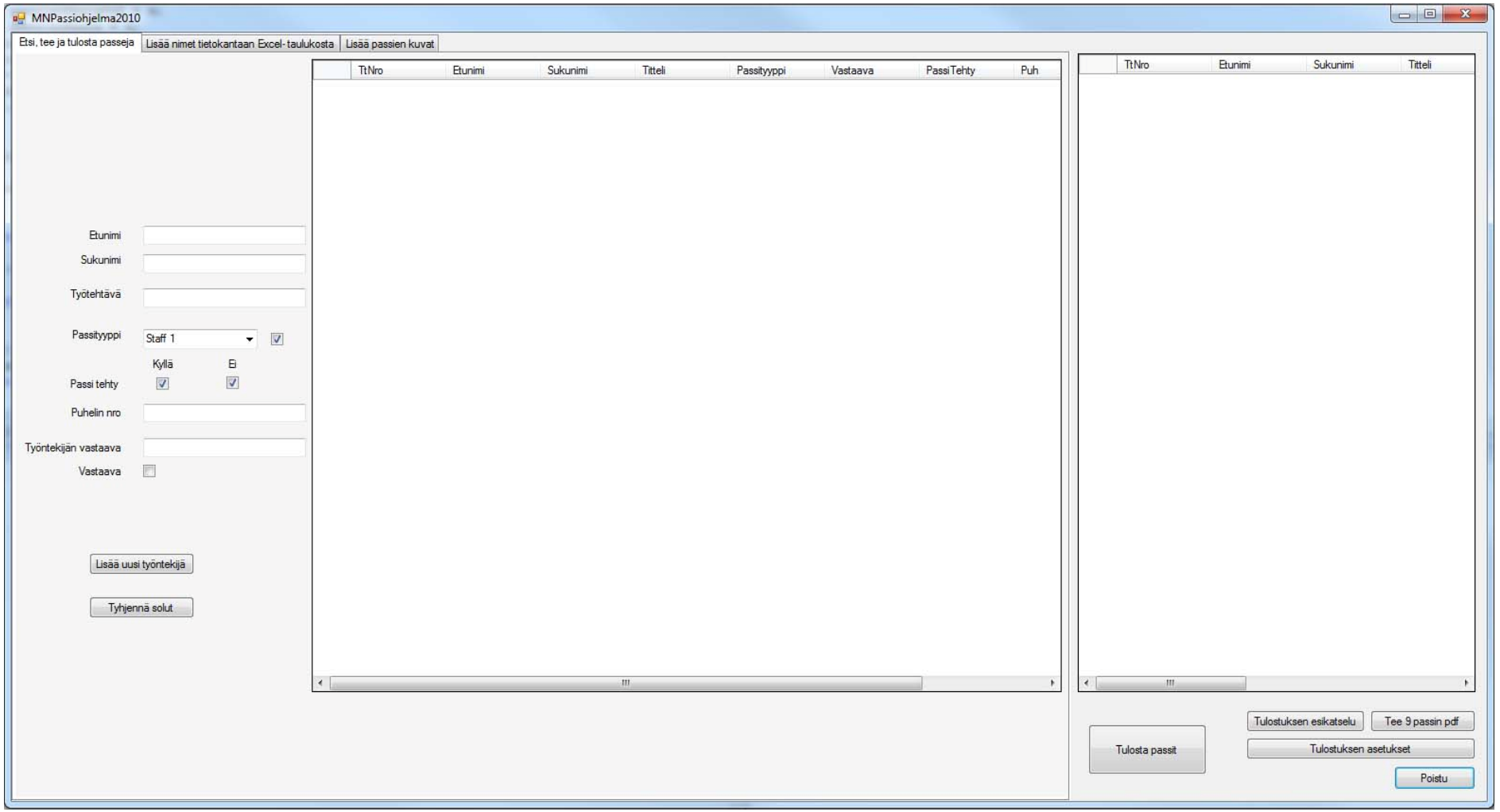
iTextPdf 2010 Project description [verkkodokumentti]. [viitattu 25.3.2010]
Saatavissa: http://itextpdf.com/index.php

Bruno Lowagie 2006 iText In Action Creating and Manipulating PDF. Manning Publications Co. Chapter 18.3 Rendering PDF

Patton Ron 2005a. Software Testing. Sams Publishing. Chapter 5, Testing the Software with Blinders On.

Patton Ron 2005b. Software Testing Sams Publishing Chapter 6, Examining the Code

LIITE 1


MNPassiohjelma2010
Etsi, tee ja tulosta passeja
Lisää nimet tietokantaan Excel-taulukosta
Lisää passien kuvat
TtNro
Etunimi
Sukunimi
Titteli
Passityyppi
Vastaava
PassiTehty
Puh
Etunimi
Sukunimi
Työtehtävä
Passityyppi
Staff 1
Kyllä
Ei
Passi tehty
Puhelin nro
Työntekijän vastaava
Vastaava
Lisää uusi työntekijä
Tyhjennä solut
Tulosta passit
Tulostuksen esikatselu
Tee 9 passin pdf
Tulostuksen asetukset
Poistu