KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Olli Tähtinen

TILAVARAUSJÄRJESTELMÄ

Opinnäytetyö 2012

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka

TÄHTINEN, OLLI	Tilavarausjärjestelmä
Insinöörityö	33 sivua
Työn valvoja	Laboratorioinsinööri Marko Oras
Työn ohjaaja	Toimitusjohtaja Mikko Maja
Toimeksiantaja	Profimill Engineering Oy
Maaliskuu 2012	
Avainsanat	asp.net, tilavaraus, web-ohjelmointi, tietokanta

Työn tavoitteena oli toteuttaa Profimill Engineering Oy:lle tilavarausjärjestelmä, jonka avulla voidaan seurata ja hallita kiinteistön tilavarauksia. Sovellus on selainpohjainen ja tietojen tallentamiseen käytetään tietokantaa. Päänäkymänä on kalenteri, josta voidaan seurata ja suorittaa tilojen varauksia. Sovelluksen muilla sivuilla hallitaan asiakkaiden, tilojen, ryhmien ja laitteiden tietoja.

Sovellus toteutettiin ASP.NET-tekniikalla ja tietojen tallentamiseen käytetään Microsoft SQL Server –tietokantaa. Sovelluksessa hyödynnetään kolmannen osapuolen ASP.NET-komponentteja. Sivuston dynaamiset osat toteutettiin Ajax-tekniikalla, käyttämällä Microsoft Ajax Extensions -komponentteja. Sovellukseen toteutettiin kaksi käyttäjätasoa: pääkäyttäjä ja vierailija. Pääkäyttäjällä on täydet luku- ja kirjoitusoikeudet, vierailijalla vain kalenterin lukuoikeus. Sovellus lokalisoitiin suomen ja englannin kielille, ASP.NET-resurssitiedostoja käyttämällä.

Ohjelmaa kehitettiin noin viikon mittaisissa jaksoissa, niin että koko ajan oli ajan tasalla oleva ja toimiva versio käytettävissä. Lopputuloksena syntyi toimiva ohjelma, johon tullaan toteuttamaan lisäominaisuuksia myöhemmin. Ohjelma liitetään osaksi ohjelmistokokonaisuutta, jolla voidaan monipuolisesti seurata ja hallinnoida kiinteistön energiankulutusta.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology

TÄHTINEN, OLLI	Space reservation system
Bachelor's Thesis	33 pages
Supervisor	Marko Oras, Laboratory Engineer
Instructor	Mikko Maja, Managing Director
Commissioned by	Profimill Engineering Oy
March 2012	
Keywords	asp.net, space reservation, web programming, database

The purpose of the study was to create a space reservation system for Profimill Engineering Oy that enables the viewing and management of space reservations in the facility. The application is browser-based and uses database for data storing. The main view is a calendar that can be used for viewing and performing space reservations. Other pages of the application are used for managing information about customers, spaces, groups and equipment.

The application was created with the ASP.NET technology and Microsoft SQL Server is used for data storing. Third party ASP.NET components are utilized in the application. The dynamic parts of the website were created using the Ajax technology by using Microsoft Ajax Extensions components. Two user levels were created in the application: main user and visitor. The main user has full read and write permissions, the visitor only has the permission to read the calendar. The application was localized in the Finnish and English languages, using ASP.NET Resource files.

The application was developed in about one week-long periods so that all the time there was-up-to-date and functional version of the application available. The end result was a functional product that will have additional features added later. The application will be integrated into a software package that can be used for wide range monitoring and managing of facility energy consumption of the facility.

SISÄLLYS

TIIVISTELMÄ

ABSTRACT

SISÄLLYS

TERMIT JA LYHENTEET 6

1 JOHDANTO 8

2 TEKNIIKKA JA KOMPONENTIT 8

2.1 ASP.NET 8

2.2 T-SQL 11

2.3 Ajax 11

2.4 Microsoft ASP.NET Ajax Extensions 13

2.5 Intersoft WebUI Studio for ASP.NET 13

3 TYÖKALUT 14

3.1 Microsoft Visual Studio 2010 14

3.2 Microsoft SQL Server Management Studio 15

4 TYÖN VAIHEET 16

4.1 Lähtötilanne 16

4.2 Ohjelmistokehitysmenetelmä 17

4.3 Testaus 17

4.4 Dokumentointi 18

5 TILAVARAUSJÄRJESTELMÄN ESITTELY 18

5.1 Käyttöliittymä 18

5.1.1 Kalenteri 19

5.1.2 Asiakkaat 20

5.1.3 Tilat 21

5.1.4 Laitteet 22

5.1.5 Laitevaraukset 23

5.2 Tietokanta 24

5.2.1 Taulut 24

5.2.2 Kyselyt 26

5.3 Käyttäjätasot 29

5.4 Lokalisointi 30

5.5 Käyttöohje 30

6 YHTEENVETO JA LOPPUPÄÄTELMÄ 30

LÄHTEET 32

TERMIT JA LYHENTEET

.NET Framework	Microsoftin kehittämä ohjelmistokomponenttikirjasto, joka toimii runkona sen päälle rakennettavalle sovellukselle.
ASP.NET	Microsoftin kehittämä .NET ohjelmistokomponenttikirjastoa hyödyntävä, verkkosovellusten kehittämiseen tarkoitettu tekniikka.
T-SQL	Transact Structured Query Language. Microsoftin ja Sybasen kehittämä tietokantojen hallintaan tarkoitettu, SQL-ohjelmointikielestä tehty laajennus.
Microsoft SQL Server	Microsoftin kehittämä, T-SQL relaatiotietokannan sisältävä palvelinohjelmisto.
IIS	Internet Information Services. Microsoftin kehittämä web-palvelin, jota käytetään ASP.NET-sivustojen isännöimiseen.
Ajax	Asynchronous JavaScript And XML. Ajax on joukko web-sovelluskehityken tekniikoita, joiden avulla web-sovelluksista voidaan tehdä vuorovaikutteisempia.
Lokalisointi	Lokalisoinnilla tarkoitetaan toimia, joiden avulla tuote sovitetaan eri maiden kulttuureihin. Lokalisointi tässä opinnäytetyössä tarkoitti käyttöliittymän kääntämistä suomenkielisestä englanninkieliseksi.
Ketterä menetelmä	Ohjelmistonkehitysmenetelmä, jossa täydellisen dokumentaation, projektinohjaustyökalujen käytön ja raskaan prosessin noudattamisen sijaan pääpaino on toimivalla ohjelmistolla, tyytyväisellä asiakkaalla ja henkilökohtaisella vuorovaikutuksella.
XML	Extensible Markup Language. XML on rakenteellinen kuvauskeli, jossa tiedon merkitys on kuvattavissa tiedon sekaan. XML auttaa jäsentämään laajoja tietomääriä selkeästi ja sitä käytetään niin tiedon esittämisen kuin tiedon välittämisen formaattina.

Synkroninen	Synkronisella tapahtumalla tarkoitetaan toimintoa, joka suoritetaan kokonaisuudessaan, ennen kuin seuraavan tapahtuman suorittaminen voidaan aloittaa.
Asynkroninen	Asynkronisella tapahtumalla tarkoitetaan toimintoa, joka voidaan suorittaa samanaikaisesti muiden toimintojen suorittamisen kanssa.
Olio	Oliolla tarkoitetaan ohjelmoinnin lähestymistapaa, jossa ohjelmointiongelmien ratkaisut jäsennetään olioiden yhteistoiminnalla. Oliot sisältävät toisiinsa loogisesti liittyvää tietoa ja toiminnallisuutta.
iFrame	Kehys, joka sijoitetaan web-sivulle ja jonka sisään sijoitetaan toinen web-sivu. Iframe-kehykseen voidaan sijoittaa web-sivun muuttuva sisältö, jolloin koko web-sivua ei tarvitse ladata uudestaan sisällön päivittämiseksi.
DLL	Dynamic Link Library. DLL on jaettu ohjelmakoodikirjasto, joka mahdollistaa ohjelmakoodin jakamisen useiden eri sovellusten kesken.
ViewState	ViewState on ASP.NET-tekniikka, joka mahdollistaa web-sivun tilan säilymisen päivittämisen yhteydessä.

1 JOHDANTO

Tämä opinnäytetyö on tehty Profimill Engineering Oy:n toimeksiantona ja ohjeistamana. Projektin toteutuksen valvojana toimi Mikko Maja Profimill Engineering Oy:ltä ja ohjaajana Marko Oras Kymenlaakson ammattikorkeakoulusta. Suurena apuna työn tekemisessä oli koko Profimill Engineering henkilökunta, erityisesti Niko Sipolaa haluan kiittää hyvistä neuvoista ja ohjauksesta.

Työn tavoitteena oli kehittää työkalu kiinteistön tilojen varaamista ja varaustilanteiden seurantaa varten. Varaustietoja tullaan käyttämään myös asiakkaiden laskutuksessa, markkinoinnissa, tilojen käyttöasteiden ja tuoton raportoinnissa sekä mahdollisesti myös kiinteistön automaatiojärjestelmän ohjauksessa.

Profimill Engineering Oy on kotkalainen vuonna 2010 perustettu yritys, joka suunnittelee ja toimittaa teollisuudelle tietojärjestelmäkokonaisuuksia tuotannon ja logististen prosessien ohjaukseen. Yrityksen erityisosaamista on tuotannon- ja valmistuksen ohjaus, tiedonkeruu- ja raportointijärjestelmät, tietojärjestelmien integrointipalvelut ja asiakaskohtaisten ohjelmistojen valmistaminen. Profimill Engineering Oy työllistää neljä henkilöä.

2 TEKNIIKKA JA KOMPONENTIT

2.1 ASP.NET

ASP.NET (Active Server Pages .NET) on Microsoftin kehittämä, vuonna 2002 julkaistu ohjelmointiteknologia dynaamisten web-sivujen (Web Forms) ja web-palveluiden (Web Services) luomiseen Web Forms tarkoittaa dynaamisia web-sivuja, joiden tiedostopääte on aspx. Web Forms -sivut käännetään ennen selaimelle lähettämistä Microsoft Internet Information Services -palvelimella (IIS-palvelin). Web Forms -sivut sisältävät selaimelle lähetettävän HTML-sivun lisäksi palvelimella suoritettavan ohjelmakoodisivun. Nämä palvelimella suoritettavat koodisivut peritään aspx-sivusta ja tiedostopäätteenä on aspx.vb käytettäessä Visual Basic -ohjelmointikieltä. Käyttämällä ASP.NET Client Script –tekniikkaa myös selaimessa suoritettava JavaScript-ohjelmakoodi voidaan sijoittaa samaan tiedostoon palvelinpuolen ohjelmakoodin kanssa (1). Web-sivujen lisäksi toinen ASP.NET-tekniikan pää-

ominaisuus on Web Services -palvelut. Tällä palvelulla tarkoitetaan sovellusta, joka toimii verkossa rajapintana muille sovelluksille ja mahdollistaa tiedon välittämisen näiden välillä (2).

ASP.NET on ilmainen teknologia web-sivujen kehittämiseen. Opinnäytetyön kehittämisessä käytettiin maksullista Visual Studio 2010 -ohjelmistokehitystyökalua ASP.NET-sivujen luomisessa, mutta vaihtoehtojakin on, esimerkiksi ilmainen Microsoft WebMatrix, joka on erityisesti ASP.NET-sivujen kehittämiseen tarkoitettu työkalu. Tilavarausjärjestelmän ASP.NET versio on yhteensopivuussyistä uusimman 4.0 Framework -version sijaan 3.5, koska käytetyt Intersoftin komponentit on toteutettu tällä versiolla. ASP.NET tuo monia hyötyjä ohjelmistokehittäjälle .NET Framework-kehyksen ansiosta.

ASP.NET-tekniikan hyötyjä ovat esimerkiksi:

- Kun ASP.NET web-sivu julkaistaan, se käännetään DLL-tiedostoksi ja näin ollen se toimii nopeammin kuin lähdekoodista tulkittava web-sivu, jollainen on esimerkiksi PHP-ohjelmointikieltä käyttävä web-sivu. (3.)

- ASP.NET tukee kaikkia .NET olio-ohjelmointikieliä (C#, VB.NET, JScript, Managed C++). Eri ohjelmointikieliä voidaan käyttää samassa web-sovelluksessa, jolloin valmista ohjelmakoodia voidaan hyödyntää, eikä sitä tarvitse kirjoittaa uudestaan toisella ohjelmointikielellä.

- Sovelluksen lokalisointi eri kielille onnistuu kätevästi ASP.NET-resurssitiedostojen avulla. Eri kielille luodaan omat tiedostonsa ja ASP.NET ottaa käyttöön sen, joka vastaa parhaiten käyttäjän koneelle asetettua kielivalintaa.

- Käyttäjän tunnistuksessa voidaan kätevästi käyttää Active Directoryä ja näin myös monen käyttäjän samanaikainen sovelluksen käyttö on mahdollista. (4.)

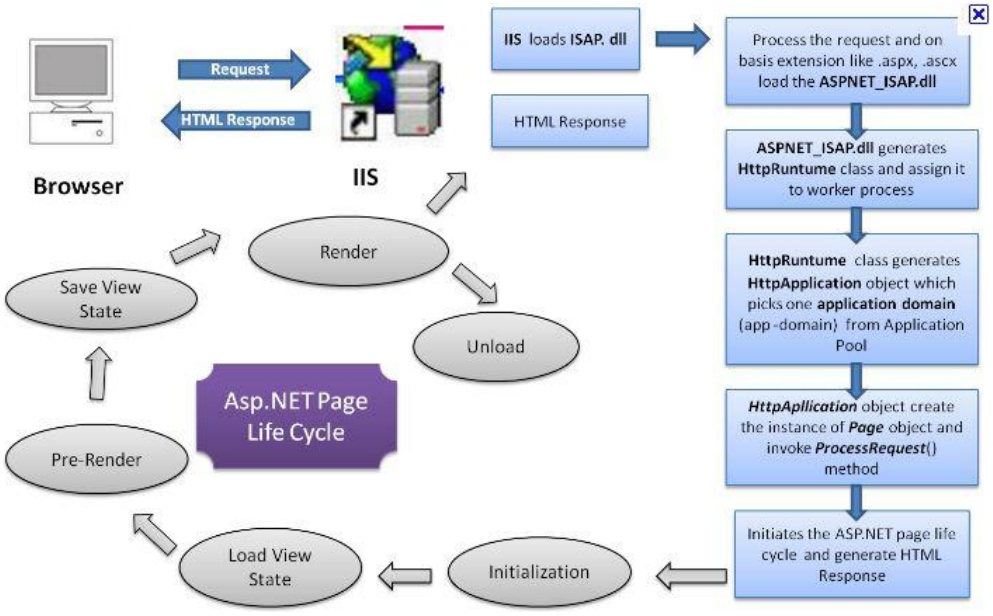
- Tietokannan käyttö ASP.NET-komponenttien kanssa on tehty todella helpoksi ja luodut tietokantakyselyt ovat turvallisempia kuin perinteiset, dynaamiset SQL-lauseet.

- Normaalien HTML-komponenttien lisäksi ASP.NET tarjoaa mahdollisuuden suorittaa HTML-komponentteja palvelimella, jolloin näihin voidaan viitata palvelinpuolen ohjelmakoodista. Lisäksi ASP.NET tarjoaa monipuolisempia palvelinpuolella suoritettavia ASP-komponentteja, joilla sivustolle on helppo luoda kehittyneempiä toimintoja ja dynaamisuutta. (5, 36 – 51.)

- Palvelimella suoritettavat HTML-komponentit ja ASP-komponentit säilyttävät käyttäjän valinnat web-sivun päivityksen yhteydessä ViewState -tekniikan ansiosta. Tämä helpottaa ohjelmoijan työtä, koska erillistä koodia ei tarvitse kirjoittaa muistamaan kontrollien tilaa. (5, 210.)

ASP.NET-tekniikan käyttämisen vaatimukset:

ASP.NET-tekniikan käyttäminen vaatii ohjelmallisen Microsoft Internet Information Services -palvelimen, jossa web-sivut käännetään internetselaimen ymmärtämään XHTML-muotoon. Toinen vaatimus koskeekin tästä syystä käytettävää web-sivuston palvelinalustaa, jonka täytyy tukea .NET kehystä ja jossa on tällä hetkellä oltava siis Microsoftin käyttöjärjestelmä. Kuva 1 selventää ASP.NET-sivun luomisprosessia.



Kuva 1. ASP.NET-sivun elinkaari. (6.)

2.2 T-SQL

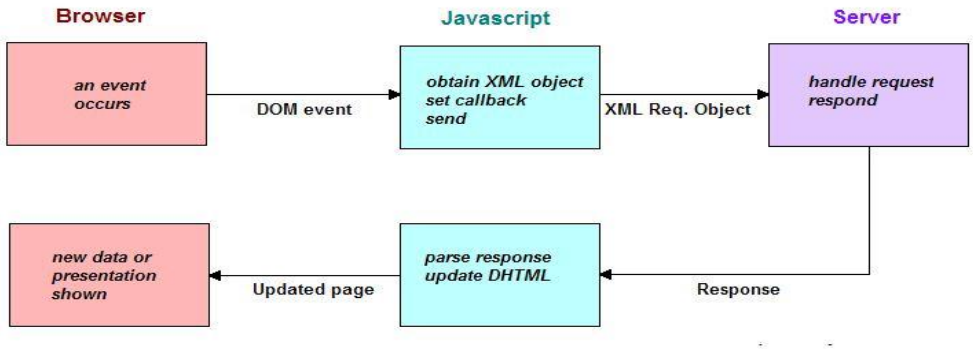
Tilavarausjärjestelmän tiedot tallennetaan tietokantaan, Microsoft SQL Server 2008 R2 -tietokantapalvelimelle. Käytettävä kieli SQL Server -tietokannassa on Transact-SQL (T-SQL). T-SQL on perinteistä SQL-kieltä monipuolisempi ja mahdollistaa muun muassa silmukoiden ja ehtolauseiden käyttämisen tietokantaan sijoitettavissa kyselyissä, eli tallennetuissa proseduureissa (Stored Procedures). (7.)

Tilavarausjärjestelmässä käytetään parametrisoituja, tietokantaan tallennettuja proseduureja. Tietokantaan tallennettujen proseduurien uudelleen käyttäminen on helppoa ja kyselyiden tietoturvaa saadaan parannettua. Parametrisoidulla kyselyllä tarkoitetaan sitä, että web-sivulta kyselyyn liitettävät syötteet sijoitetaan muuttujiin eikä suoraan SQL-lauseeseen. Parametrisoidut kyselyt ovat vahvasti tyypitettyjä, jolloin käyttäjän syöttämä tieto pitää ensin muuttaa parametrin tyyppiä vastaavaan muotoon, esimerkiksi päivämääräksi tai luvuksi. Tämä tietotyyppimuunnos vähentää SQL-injektion mahdollisuutta, jossa väärällä syötteellä yritetään manipuloida kyselyn rakennetta.

2.3 Ajax

Ajax on lyhenne sanoista Asynchronous JavaScript and XML. Ajax-tekniikka parantaa web-sivun käyttökokemusta mahdollistamalla web-sivun päivittämisen osissa. Ilman Ajax-tekniikkaa toteutetuissa web-sivuissa koko web-sivu lähetetään palvelimelle, kun sisältöä on muutettu. Tämä koko web-sivun päivittäminen näkyy näytön välähtämisenä valkoisena ja web-sivun latautumisena taas kokonaan uudestaan. Ajax mahdollistaa halutun komponentin tai web-sivun osan päivittämisen ilman koko web-sivun välittämistä palvelimelle ja takaisin. Ajax-tekniikkaa hyödyntämällä web-sivun päivitys tapahtuu nopeammin ja web-sivu pysyy paikoillaan. Ajax-toiminnallisuuden pääosassa toimii XMLHttpRequest-olio, joka mahdollistaa internetselaimen keskustelun, eli kommunikoinnin web-palvelimen kanssa. XMLHttpRequest-olio voi kommunikoida web-palvelimen kanssa joko synkronisesti tai asynkronisesti. Asynkroninen kommunikointi tarkoittaa web-sivun sisällön päivittämistä niin, että muu osa web-sivua on samanaikaisesti käytettävissä. Ilman Ajax-tekniikkaa toteutetut web-sivut kommunikoivat web-palvelimen kanssa synkronisesti. Synkronisessa kommunikointitavassa joudutaan odottamaan kommunikoinnin päättymistä, ennen kuin web-sivu on jälleen käytettävissä. (8.)

Kuvassa 2 näkyy Ajaxin toimintaperiaate. Kuvan vasemmassa reunassa punaiset laatikot kuvaavat internetselainta. Keskellä olevat turkoosin väriset laatikot kuvaavat JavaScriptillä, internetselaimessa suoritettavaa osuutta. Oikeassa reunassa oleva violetin värinen laatikko kuvaa web-palvelinta, jossa selaimelta välitetty pyyntö käsitellään. Lähtötilanteena on siis internetselaimen tapahtuma jonka seurauksena internetselaimessa muodostetaan XMLHttpRequest-olio. XMLHttpRequest-olio suorittaa pyynnön web-palvelimelle, web-palvelin käsittelee tulleen pyynnön ja lähettää päivitetyn internetsivun osan XML-muodossa takaisin internetselaimelle. Internetselaimessa web-palvelimelta palautettu XML-muotoinen vastaus muutetaan takaisin web-sivulla näytettäväksi sisällöksi. (8.)



Kuva 2. Ajax. (8.)

Tilavarausjärjestelmässä on Ajax-tekniikkaa hyödynnetty käyttämällä Microsoftin Ajax Extensions Update Panel -komponenttia, josta tässä kappaleessa jäljempänä käytetään nimeä paneeli. Paneeli sijoitetaan web-sivulle ja sillä ympäröidään komponentti tai komponentit, jotka halutaan päivittää. Paneeleja voidaan sivulla käyttää lukuisia ja niitä voidaan käyttää myös sisäkkäin, jolloin ulomman paneelin päivittyessä päivittyy myös sisemmät paneelit. Paneelin päivittymisen eli palvelimella suoritettavan ohjelmakoodin suorittamisen käynnistää esimerkiksi jonkin painikkeen OnClick-tapahtuma, eli painikkeen klikkaaminen. Yleensä päivittyvä paneeli on se, jossa tapahtuman liipaiseva komponentti sijaitsee. Paneelin voi kuitenkin määrittää päivittymään myös sen ulkopuolella sijaitsevan komponentin tapahtumasta lisäämällä paneelin liipaisinkokoelmaan eli Triggers-kokoelmaan halutun komponentin ID-tunniste. (9.)

2.4 Microsoft ASP.NET Ajax Extensions

Microsoft ASP.NET Ajax Extensions -komponentteihin kuuluu edellä mainitun Update Panel -komponentin lisäksi muita web-sivuston kehittämistä helpottavia komponentteja. Tilavarausjärjestelmässä hyödynnetään ASP.NET Validator -komponentteja käyttäjän syötteiden tarkistuksessa. Syötteen tarkistus tapahtuu käyttäjän selaimessa, jolloin web-sivua ei lähetetä palvelimelle, jos jokin lomakkeen kenttä on täytetty väärin. Tästä saatava hyöty on etenkin web-sivun käyttökokemuksessa ja se myös osaltaan vähentää web-palvelimen kuormitusta. ASP.NET Validator -komponenttien ominaisuuksia voidaan lisätä käyttämällä ASP.NET Ajax Control Toolkit -komponenteista löytyvää laajennusta. Edellä mainittuja komponentteja hyödyntämällä voidaan esimerkiksi vaihtaa väärin täytetyn tekstikentän väri tai ilmoittaa käyttäjälle ponnahdus-ikkunassa virheellisestä syötteestä. (10.)

2.5 Intersoft WebUI Studio for ASP.NET

Tilavarausjärjestelmässä on hyödynnetty kolmannen osapuolen, Intersoftin valmistamia WebUI Studio -komponentteja. Intersoftin komponenteissa on normaaleihin ASP.NET-komponentteihin verrattuna enemmän toimintoja ja ulkoasun muokkausmahdollisuuksia. Komponenteissa käytetään Ajax-tekniikkaa tehokkaasti hyväksi, se parantaa web-sivun käyttökokemusta ja vähentää ohjelmoijan työtä.

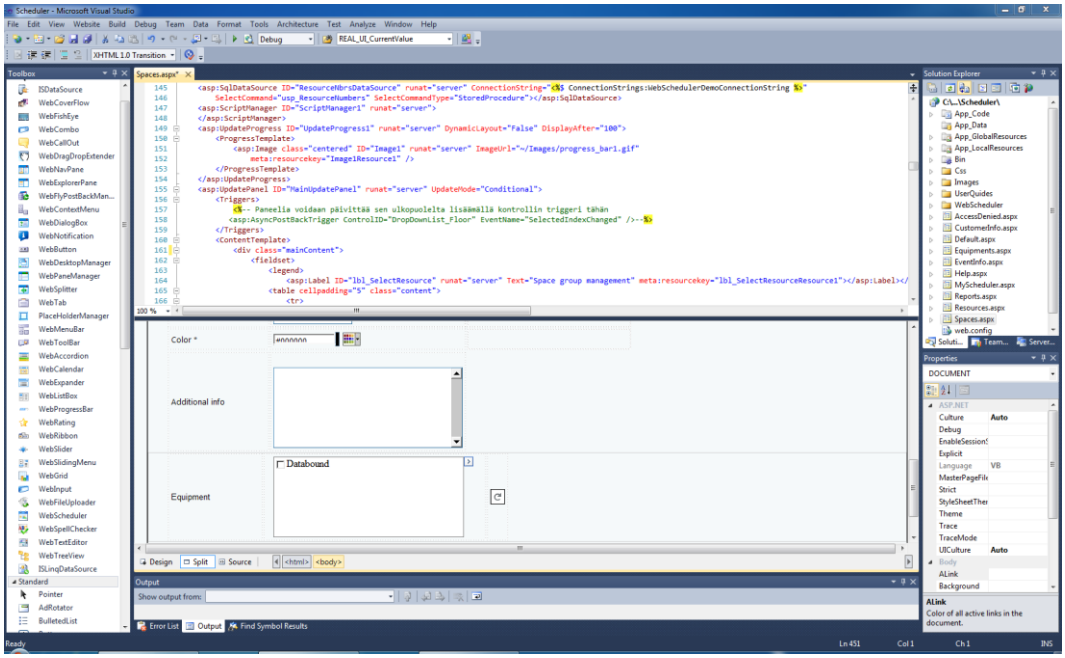
Suurinta osaa käytetyistä Intersoftin komponenteista näyttelee pääsivulla oleva Web-Scheduler 3 -kalenterikomponentti. Kalenterin käyttöönotto oli helppoa, mutta sen muokkaaminen tilavarausjärjestelmän vaatimuksiin osoittautui melko vaativaksi. Intersoftin asiakaspalveluun oltiin useaan otteeseen yhteydessä ja sieltä saatiinkin nopeasti hyviä neuvoja, miten haluttuja muutoksia pystytään toteuttamaan. Kalenterin ulkoasun ja toimintojen muokkauksessa pääpaino oli JavaScript-ohjelmoinnissa. Moneen kalenterin muokkaustarpeeseen löytyi valmistajalta valmiita ohjelmakoodeja ohjelmointityötä helpottamaan. Tietokantaan tehdyt muutokset ja lisäkentät vaativat runsaasti perehtymistä tietokannan käyttöön ASP.NET-ympäristössä. Opiskeltavaksi tässä kohtaa projektia tuli Intersoftin ISDataSource-tietolähteen hyödyntäminen tietokantayhteydessä, sekä Data Access Layer -tekniikan ymmärtäminen.
Data Access Layer -tekniikasta kerrotaan tarkemmin luvussa 5.2.2.

3 TYÖKALUT

3.1 Microsoft Visual Studio 2010

Tilavarausjärjestelmän ohjelmointityökaluna toimi kouluprojekteista entuudestaan tuttu Microsoftin Visual Studio. Visual Studio -ohjelmointityökalulla voidaan luoda ASP.NET, Windows Phone, SharePoint sekä Windows Forms -sovelluksia, useilla eri ohjelmointikielillä (11). Visual Studio -ohjelmointityökalun käyttäminen helpotti sovelluksen kehittämistä, koska kehitystyökalun käyttöä ei tarvinnut opetella alusta asti. ASP.NET-sovelluksen kehittäminen toi kuitenkin paljon uusia asioita opittavaksi aiempaan Windows-lomakeohjelmointiin verrattuna, Pidänkin ASP.NET-sovelluksen kehittämistä selvästi vaativampana ohjelmointityönä.

Palvelinpuolen ohjelmakoodin virheiden etsimiseen on Visual Studiossa hyviä työkaluja tarjolla ja tämä nopeuttaa ja helpottaa sovelluskehitystä. Ilmaisessa ASP.NET-sovelluksien kehittämiseen tarkoitetussa Microsoftin WebMatrix-ohjelmointityökalussa virheiden etsimiseen tarkoitetut työkalut puuttuvat ja se vaikeuttaa kehitystyötä melkoisesti. Kuvassa 3. on työssä käytetyn Visual Studion kehitysnäkymä.

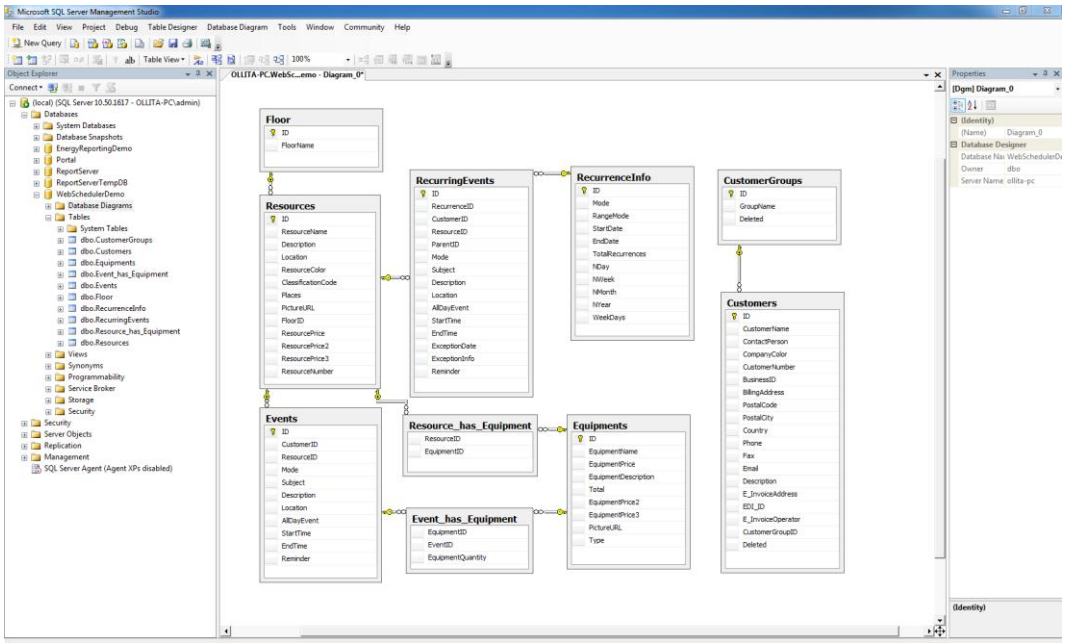


Kuva 3. Microsoft Visual Studio 2010.

3.2 Microsoft SQL Server Management Studio

Tilavarausjärjestelmän tietokantapalvelimena toimii Microsoft SQL Server 2008 R2. SQL Server -tietokantapalvelimen mukana tulee tietokantojen hallintatyökalu, SQL Server Management Studio (kuva 4). Management Studio on monipuolinen työkalu tietokantojen luomiseen, ylläpitoon, käyttäjäoikeuksien määrittämiseen sekä tietokantojen käyttöönottoon. Management Studio -työkalussa on mukana SQL Server Agent -toiminto, jolla voidaan ajastaa tietokannan huoltotoimenpiteitä tai ajaa haluttuja proseduureja automaattisesti tiettyyn kellonaikaan. Tietokantakyselyjen nopeutta voidaan testata sisäänrakennetulla suorituskykymittarilla, jonka avulla kyselyt saadaan helposti optimoitua asettamalla indeksoinnit kohdalleen. Tietokantarakenteesta saadaan myös havainnollistava kuva, josta voidaan myös muokata tietokantaa. Kuvassa 4 näkyy tilavarausjärjestelmän tietokantarakenne Management Studio -työkalulla luotuna diagrammina. (12.)

Kuvassa 4. on työssä käytetyn SQL Server Management Studion kehitysnäkymä.



Kuva 4. SQL Server Management Studio.

4 TYÖN VAIHEET

4.1 Lähtötilanne

Tilavarausjärjestelmän kehitys aloitettiin määrittelemällä ohjelman tarvittavat toiminnot ja käyttäjätasot, joihin toiminnot jaetaan. Muita vaatimuksia ohjelmalle oli lokalisointi sekä ohjelmassa olevan tiedon käytön mahdollistaminen toisista järjestelmistä. Lokalisoinnilla tarkoitetaan ohjelman kotouttamista eri kulttuureihin. Tässä tilanteessa lokalisointi tarkoitti käytännössä käyttöliittymän tekstien kääntämistä kahdelle eri kielelle, suomeksi ja englanniksi. Mitään tarkempaa suunnitelmaa ohjelman ulkoasusta tai toimintojen sijoittamisesta ei ollut. Ohjelmointitekniikkana päätettiin käyttää ASP.NET-tekniikkaa, ohjelmointikielenä Visual Basic.NET -ohjelmointikieltä ja tietokantana SQL Server -tietokantapalvelimella suoritettavaa T-SQL-tietokantaa. Näistä edellä mainituista tekniikoista yrityksessä oli ennestään vankka kokemus, joka helpottaisi työn suorittamista. Käytettävien ohjelmistokomponenttien valinnassa vertailtiin alan lehdistä ja internetistä löytyviä komponentteja ja valittiin näistä omaan tarkoitukseen sopivimmat. Myöhemmin ohjelmaa kehittävän henkilön ohjeeksi tehtiin 27-sivuinen toteutussuunnitelma (Toteutussuunnitelma_Tilavarausjärjestelmä.docx), jossa ohjelman toiminnot, tietokanta ja proseduurit ovat tarkasti kuvattuna. Toteutussuunnitelmaa täydennettiin ja tarkennettiin työn edetessä ja asioita opiskellessa.

Kun käytettävät komponentit ja ohjelman vaatimukset oli määritetty, alkoi ASP.NET- ja SQL Server -ympäristöön tutustuminen. Aiempaa kokemusta web-ohjelmoinnista ei ollut koulussa käytyjen kurssien lisäksi, joten ensin kerrattiin web-ohjelmoinnin perusteita ennen ASP.NET-tekniikkaan siirtymistä. Tietokantakokemusta oli jonkin verran jo aikaisemmista projekteista, mutta SQL Server ja T-SQL -ympäristöstä ei, joten tässäkin riitti runsaasti opiskelemista. Käytettävien tekniikoiden perusasioiden selkiydyttyä tuli seuraavaksi opiskelun aiheeksi kolmannen osapuolen eli komponenttien hyödyntäminen ohjelmoinnissa. Ohjelmassa päätettiin käyttää yrityksessä jo aiemmin käytettyjä Intersoft WebUI Studio -komponentteja, näiden käytön tullessa tutuksi oli mukavaa huomata kolmannen osapuolen komponenteista saatava hyöty ohjelmistokehityksessä.

4.2 Ohjelmistokehitysmenetelmä

Sovelluksen kehittämisessä käytettiin niin sanottua ketterää menetelmää, jossa sovelluksesta on koko ajan käytettävissä toimiva versio. Ketterässä menetelmässä ei pyritä täydelliseen dokumentointiin, raskaiden etukäteen suunniteltujen prosessien noudattamiseen tai projektityökalujen käyttöön. Ketterän menetelmän tärkeimpinä ideoina ovat tyytyväinen asiakas, toimiva sovellus ja henkilökohtainen vuorovaikutus sekä ohjelmistokehittäjien että asiakkaan välillä. (13, 43 – 46.)

Ketterässä menetelmässä sovelluksen kehitys jaetaan sykleihin, joiden pituus on normaalisti 2 – 3 viikkoa (13. 47 – 48). Jokaisen syklin lopulla on sovelluksesta tarkoitus olla käytössä toimiva versio, johon on toteutettu uusia ominaisuuksia tai tehty parannuksia. Koska sovellus on käyttökuntoinen ja tuore versio on testattavissa koko projektin ajan, on helppoa kysyä muilta työntekijöiltä mielipiteitä sovelluksesta ja tehdä mahdollisia muutoksia ja korjauksia. Tällä tavalla sovellusta kehitettäessä säästetään aikaa, koska muutokset eivät ole kovin suuria, jos sovelluksen jokin osa-alue päätetäänkin toteuttaa toisella tavalla.

4.3 Testaus

Ohjelmistotestaus on tärkeä ja aikaa vievä alue ohjelmistokehityksessä. Ohjelmiston järjestelmällinen testaus ja virheiden korjaaminen on työlästä ja suunnitelmallista työtä, mutta vain tällä tavalla saadaan kehitettyä tuote, johon asiakas voi olla tyytyväinen. Samanaikaisesti ohjelmistoa kehitettäessä suoritettava testaus on tehokasta, koska virheiden korjauskustannukset kasvavat nopeasti, jos virheen löytyminen siirtyy myöhäisempään vaiheeseen (13. 198). Virheetöntä ohjelmistoa ei ole olemassa, mutta on pyrittävä löytämään ja korjaamaan mahdollisimman paljon virheitä ennen ohjelmiston luovuttamista asiakkaalle. Tilavarausjärjestelmän osalta testausta suoritettiin samalla ohjelmaa kehitettäessä ja etenkin projektin loppuvaiheessa, jossa kaikki vaaditut toiminnot oli jo saatu toteutettua. Loppuvaiheen testaus paljasti monia erikoistilanteita, joihin ei osattu varautua sovellusta kehitettäessä. Erikoistilanteiden hallitseminen vaati muutosten tekemistä sekä käyttöliittymän toimintojen logiikkaan että ohjelman sisäisen toteutukseen.

4.4 Dokumentointi

Ohjelmistoprojekteihin kuuluu oleellisena osana erilaisten dokumenttien kirjoittaminen. Tilavarausjärjestelmästä tehtiin 27-sivuinen toteutussuunnitelma (Toteutussuunnitelma_Tilavarausjärjestelmä.docx) sekä 12-sivuinen käyttöohje (Tilavarausjärjestelmä_Käyttöohje.docx). Toteutussuunnitelma toimii ohjeena sovellusta jatkossa kehittävälle henkilölle. Toteutussuunnitelma sisältää tiedot käytetyistä komponenteista, käyttäjätasojen toteutuksesta, lokalisoinnista, tietokannasta ja proseduureista, sekä ohjeita sovelluksen asentamista varten. Toteutussuunnitelma tehdään yleensä ennen sovelluksen tekemistä, mutta koska kaikki vaatimukset ja tekniset toteutustavat eivät olleet alkuvaiheessa tiedossa, kirjoitettiin ja täydennettiin toteutussuunnitelmaa projektin edetessä. Toteutussuunnitelma on tehty opinnäytetyön toimeksiantajalle, Profimill Engineering Oy:lle ja on kyseisen yrityksen omaisuutta, eikä sitä näin ollen voida julkaista opinnäytetyöraportin liitteenä. Tilavarausjärjestelmästä tehty käyttöohje sijoitettiin avautuvaksi tilavarausjärjestelmän päävalikosta. Näin käyttöohje on helposti saatavilla ja käyttäjä voi nopeasti tarkistaa halutun toimenpiteen oikean suoritustavan.

5 TILAVARAUSJÄRJESTELMÄN ESITTELY

Tilavarausjärjestelmä on ulkoasultaan ja toiminnoiltaan melko perinteinen lomakkeita käyttävä sovellus, jonka käyttö on helppoa omaksua. Käyttöliittymän toiminnot sijoitettiin ja nimettiin niin, että sovellusta ensi kertaa käyttävän henkilön ei tarvitsisi välttämättä turvautua käyttöohjeeseen. Käyttöliittymästä pyrittiin tekemään informatiiviseksi ja sen käyttölogiikka helposti ymmärrettäväksi. Seuraavissa luvuissa kerrotaan tilavarausjärjestelmän päätoiminnot lyhyesti.

5.1 Käyttöliittymä

Käyttöliittymä koostuu pääsivusta, jossa sijaitsevalle iFrame-kehykselle avataan sovelluksen eri sisältösivut. Iframe-kehys on käytännössä web-sivu upotettuna toisen web-sivun sisään. Iframe-kehykseen voidaan sijoittaa web-sivun muuttuva osuus, jolloin koko web-sivua valikkoineen ei tarvitse päivittää sisältöä päivitettäessä. Iframe-kehyksen käytössä ongelmana on kirjanmerkkien tallentaminen. Iframe-kehyksessä sijaitsevan web-sivun osoite on eri kuin internetselaimen osoiterivillä näkyvä osoite. Tästä syystä ei web-sivusta tehty kirjanmerkki avaa käyttäjälle oikeaa Iframe-

kehykseen sijoitettua sisältösivua. Tilanteessa, jossa web-sivuston luonteva käyttö ei häiriinny kirjanmerkkiongelman takia, on Iframe-kehysten käyttö kätevä ja helppo tapa rakentaa web-sivu.

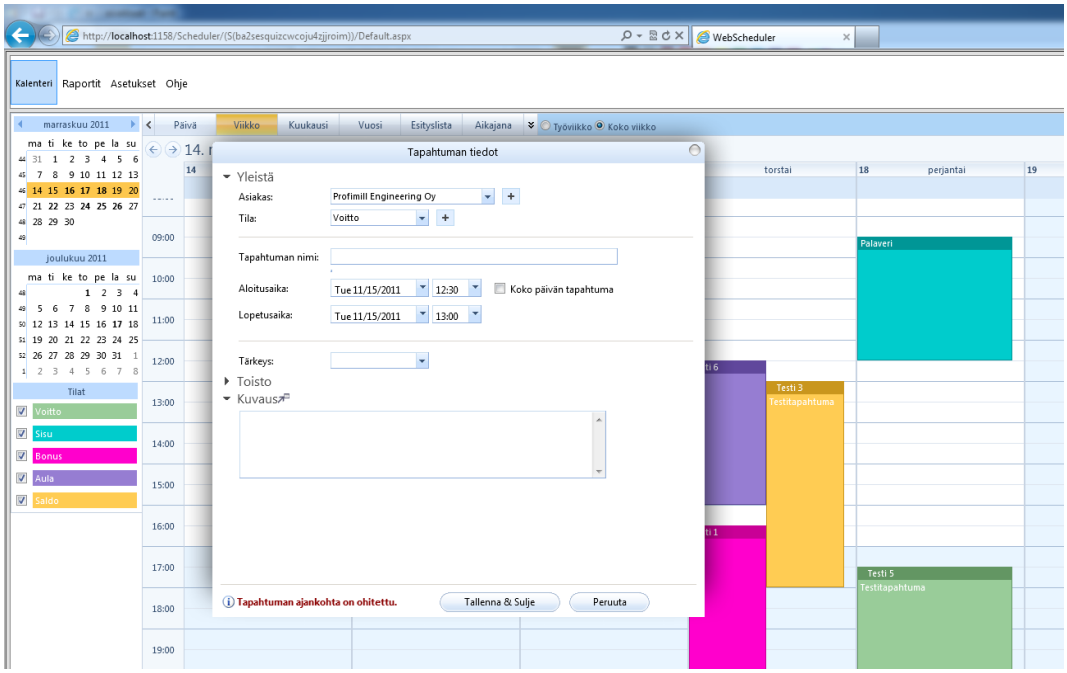
Pääsivun yläreunassa on valikko, josta siirrytään sovelluksen eri web-sivujen välillä. Sisältösivut ovat kuvassa 5 näkyvää kalenteria lukuun ottamatta lomakkeita, joilla hallitaan asiakkaiden, tilojen sekä tiloissa ja tapahtumissa olevien laitteiden tietoja. Sivujen valikko- ja sisältöosiot on toteutettu Intersoft WebPanemanager -komponentilla.

5.1.1 Kalenteri

Sovelluksen päänäkymänä on kalenteri (kuva 5), josta voidaan tarkastella tilojen varaustilannetta sekä suorittaa tilavarauksia. Kalenterista voidaan myös syöttää uusia asiakkaita järjestelmään ja näiden täydelliset yhteystiedot voidaan täydentää myöhemmin, asiakastietojen hallinnointiin tarkoitetulla lomakesivulla (kuva 6). Kalenteri on Intersoftin WebScheduler 3 -komponentti. WebScheduler 3 -komponenttiin täytyi tehdä melko paljon muutoksia käyttöliittymään sekä tietokantaan, jotta se saatiin vastaamaan tilavarausjärjestelmän tarpeisiin.

Tilavaraukseen liittyy asiakas- ja varattava tila. Nämä tiedot valitaan kalenteria klikkaamalla avautuvassa lomakkeessa, joka näkyy kuvassa 5. Tilavaraus voi olla luonteeltaan kertaluonteinen tai toistuva ja toistojen väli voidaan määrittää päivän, viikon, kuukauden tai vuoden tarkkuudella. Varauksen ajankohta ja kesto valitaan samalla lomakkeella kuin asiakas ja tila. Varauksen voi poistaa ja sen kestoa voidaan muuttaa kätevästi suoraan kalenterinäkymästä.

Kuvassa 5. on tilavarausjärjestelmän päänäkymä, tilanteessa jossa käyttäjä on avannut kalenterista tilavarauslomakkeen.



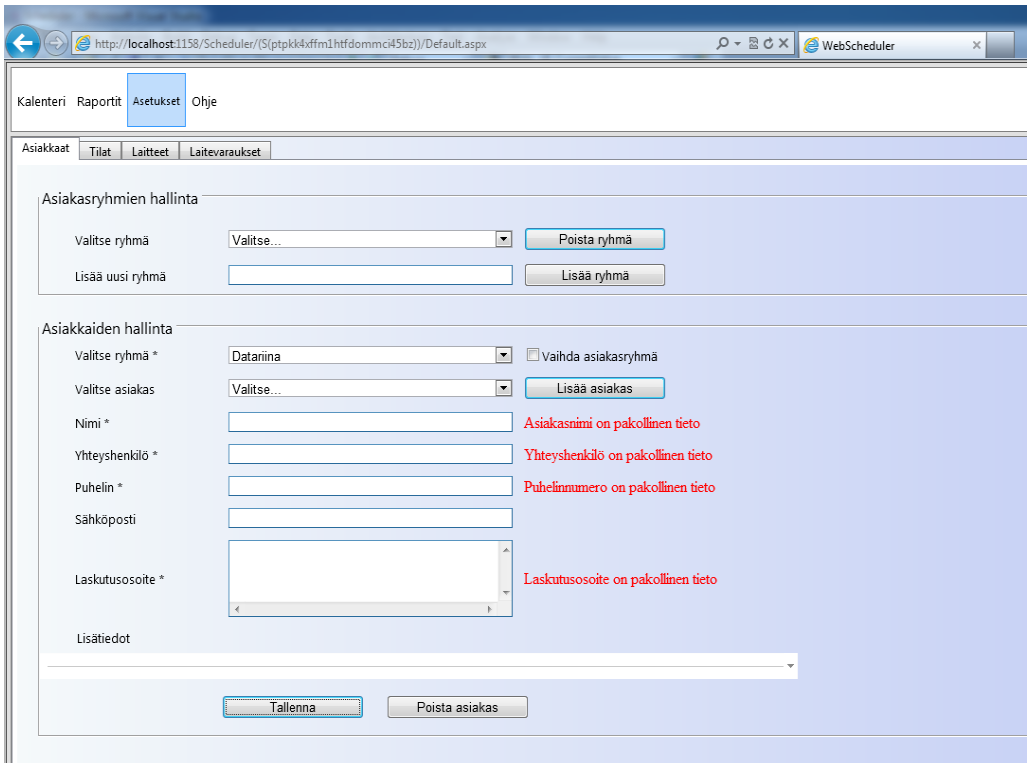
Kuva 5. Päänäkymän kalenteri ja tilavarauslomake.

5.1.2 Asiakkaat

Tilavarauksiin liitettävien asiakkaiden yhteystiedot lisätään ja niitä muokataan omalla lomakkeellaan (kuva 6). Järjestelmään määritetään asiakasryhmiä, joihin asiakkaat ryhmitellään. Ryhmittelyn ansiosta asiakasrekisterin selaaminen on helpompaa ja tilojen varaushinnoissa voidaan käyttää eri hinnoittelua eri asiakasryhmille.

Kuvan 6 yläreunassa näkyy asiakasryhmien hallintaosio, jossa ryhmiä lisätään, muokataan ja poistetaan. Asiakasryhmäosion alapuolella hallitaan asiakkaiden ryhmä-, yhteys- ja laskutustietoja. Lomakkeella on myös piilotettu lisätieto-osio, joka avautuu kuvassa 6 näkyvää nuolenpäätä klikkaamalla. Lisätieto-osio sisältää e-laskutustiedot sekä muita valinnaisia lisätietoja. Lisätieto-osio on toteutettu Intersoft WebExtender -komponentilla. Käyttäjän syötteiden tarkistus on toteutettu selainpäässä Microsoftin ASP.NET Validator -komponentteja hyödyntämällä. Jos asiakas on täydentänyt jonkin kentän väärin, tai jokin tähdellä(*) merkityistä pakollista kentistä on täyttämättä, näytetään väärin täytetyn kentän vieressä virheilmoitus sekä esimerkki kentän oikeasta täyttötavasta. Kun lomakkeen syötteet on hyväksytty ja lomake lähetetään palvelimelle, suoritetaan syötteille tyyppimuunnos tietokantakyselyn parametreihin siirtämistä varten. Jos lomakkeelle on selainpäässä suoritetusta tarkistuksesta huolimatta jäänyt

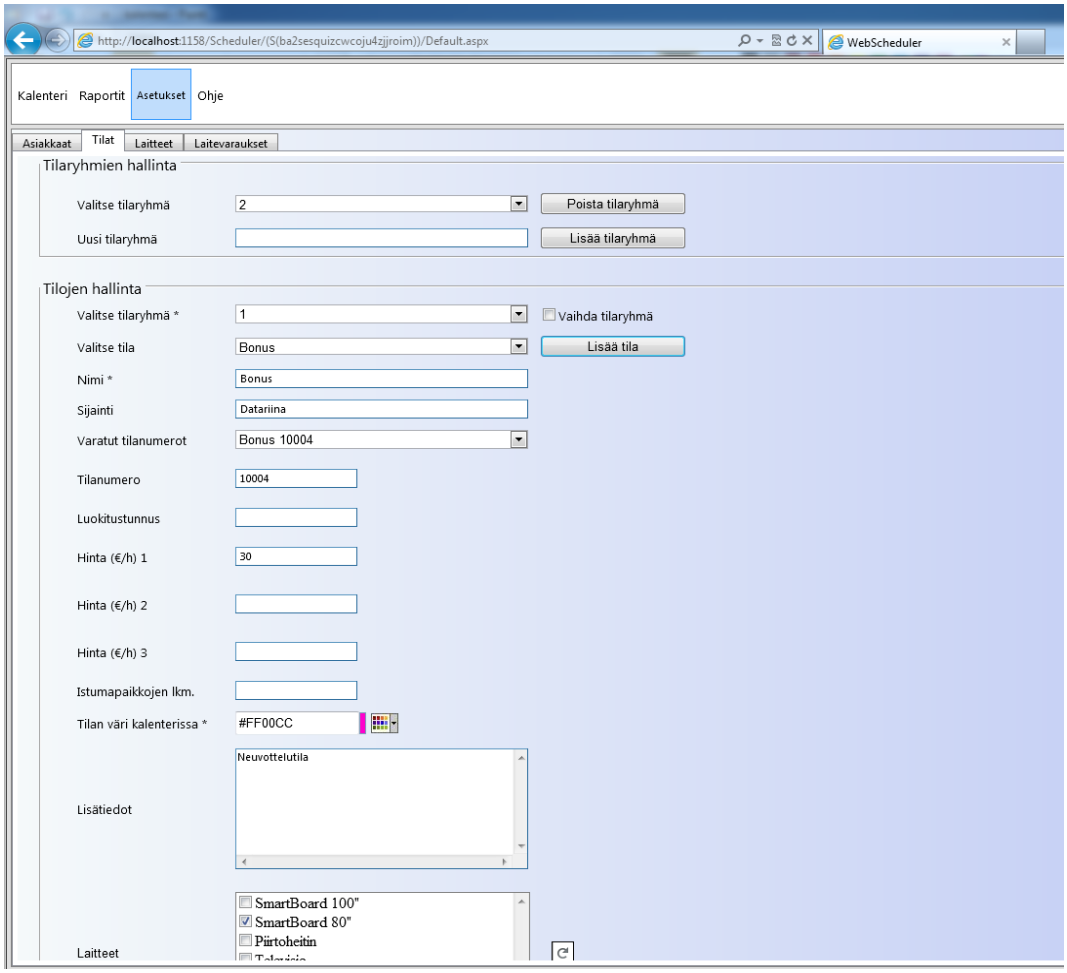
vääriä syötteitä, ei tietoja tallenneta tietokantaan vaan näytetään virheilmoitus sivun alareunassa.



Kuva 6. Asiakastietolomake.

5.1.3 Tilat

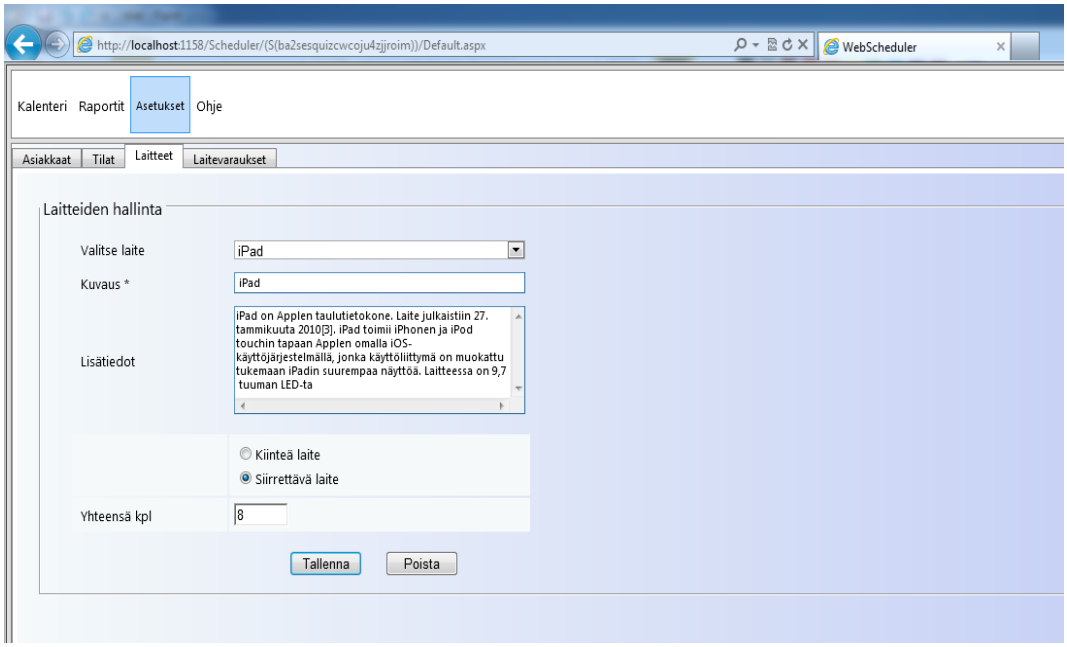
Tilojen tietoja hallitaan lomakkeella (kuva 7), jonka ulkoasu ja toiminta noudattaa asiakaslomakkeen tyyliä. Kuten asiakkaat, ovat tilatkin omissa ryhmissään. Tilaryhmä voi olla esimerkiksi kerrosnumero tai tilatyyppiä kuvaava nimi. Sivun yläreunassa olevassa osiossa hallitaan tilaryhmien tietoja ja sen alapuolella olevassa osiossa tilojen tietoja. Suurin ero asiakaslomakkeeseen on tilalomakkeella oleva laitelista, josta valitaan tilan varustukseen kuuluvia laitteita sekä värinvalitsin, jolla valitaan tilalle kalenterilla näkyvä väri. Tilalle on valittavissa yksilöllinen tilanumero, joka näkyy Varatut tilanumerot pudotusvalikossa. Yksilöllistä tilanumeroa tarvitaan sovellukseen myöhemmin toteutettavassa liitännässä, jossa tilojen varaustietoja näytetään asiakasyrityksen kiinteistön aulamonitorissa.



Kuva 7. Tilatietolomake.

5.1.4 Laitteet

Sovelluksessa on kahdenlaisia laitteita: kiinteitä ja siirrettäviä. Näiden tietoja hallitaan kuvassa 8 näkyvällä lomakkeella. Kiinteät laitteet ovat tilojen varustukseen kiinteästi kuuluvia ja tulevat järjestelmään lisäämisen jälkeen näkyviin tilatiedot sivulla sijaitsevalle valintalistalle (kuvan 7 alareunassa). Siirrettävät laitteet ovat tilavarauksen yhteydessä tapahtumiin varattavia laitteita tai tarvikkeita. Siirrettäville laitteille merkitään järjestelmään lisäämisen yhteydessä määrä ja määrää voidaan päivittää myöhemmin laitteiden määrän muuttuessa. Määrä-tietoa tarvitaan laitevarauksia tehdessä (kuva 9), jotta nähdään, onko laitteita vapaana tai ovatko ne jo varattuina samana ajankohtana johonkin toiseen tapahtumaan.



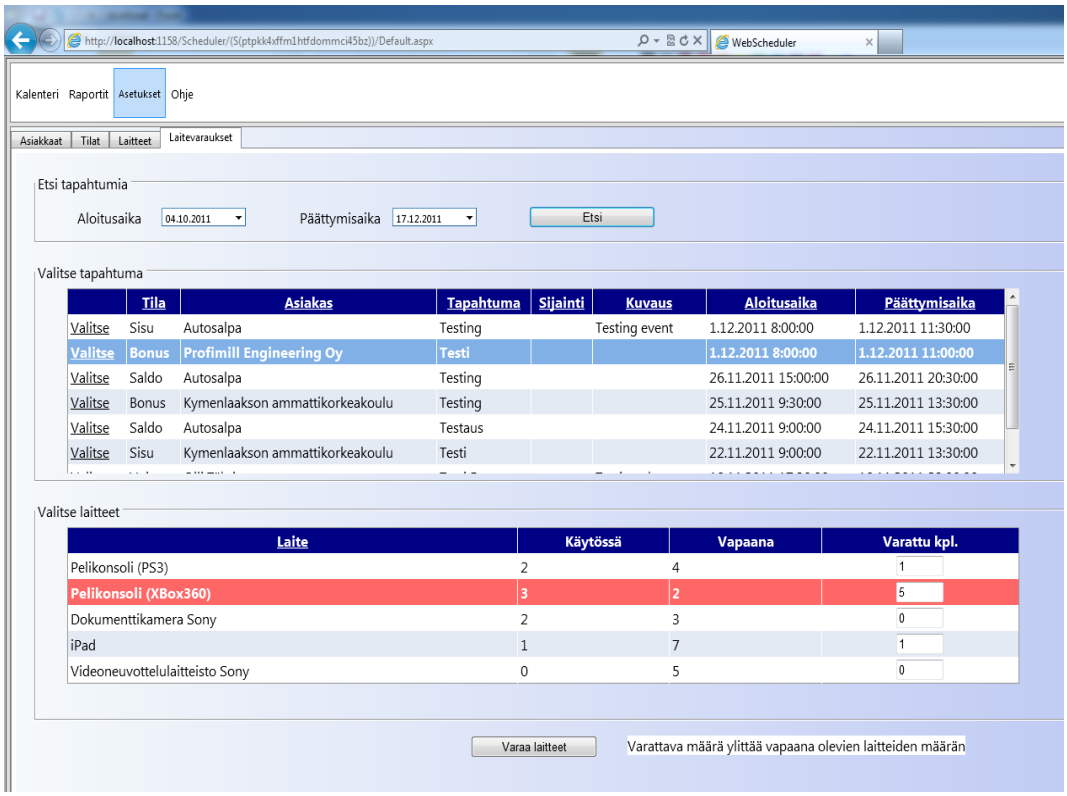
Kuva 8. Laitetietolomake.

5.1.5 Laitevaraukset

Tilavarauksiin eli tapahtumiin voidaan tarvita laitteita tai tarvikkeita. Tätä tarvetta varten sovellukseen tehtiin laite- ja laitevaraussivut. Kuvassa 8 näkyvällä laitesivulla voidaan järjestelmään lisätä laitteita, jotka ovat tyypiltään siirrettäviä ja joille voidaan antaa kappalemäärä.

Laitevarauksen tekeminen voidaan suorittaa heti sen jälkeen, kun uusi tilavaraus tallennetaan sovelluksen päänäkymän kalenterissa. Kun tilavaraus tallennetaan, voidaan samassa ikkunassa valita, halutaanko laitteita varata tapahtumaan. Jos laitteita halutaan varata, laitevaraussivu avautuu uuteen selainikkunaan. Avautuneessa selainikkunassa on juuri lisätty uusi tilavaraus valmiiksi valittuna tapahtumalistalta. Laitevarauksen voi tehdä myös myöhemmin. Kuvan 9 yläreunassa näkyy päivämäärävalitsimet, joilla tapahtumia voidaan etsiä. Kun haluttu tapahtuma on löydetty, valitaan se listalta, jonka jälkeen sivun alareunassa olevalta laitelistalta voidaan varata laitteita kyseiseen tapahtumaan. Laitelistalla olevat laitemäärät päivittyvät ajankohdan mukaisesti, eikä laitteita siis voida varata tapahtumaan, jos niitä ei ole vapaana. Tapahtuman ajankohdan mennessä ohi varattu laitemäärä päivittyy automaattisesti ja laitteet ovat taas varattavissa uusiin tapahtumiin. Sovellus estää laitteiden varaamisen tapahtu-

maan, jos varattava määrä ylittää laitteen vapaana olevan määrän. Tästä tilanteesta näkyy esimerkki kuvan 9 alareunassa.



Kuva 9. Laitevarauslomake.

5.2 Tietokanta

Tietokannan suunnittelun ja toteutuksen tulisi olla ohjelmistoprojektin ensimmäisiä töitä, heti ohjelmistolle asetettavien vaatimusten määrityksen jälkeen. Ennen tietokannan rakenteen valmistumista onkin käyttöliittymän suunnittelu ja toteuttaminen hankalaa ellei mahdotonta. Sovelluksen päänäkymässä käytettävän WebScheduler 3 -kalenterikomponentin mukana tuli viisi taulua sisältävä T-SQL–tietokanta. Lähtötilanteena oli siis tietokannan rakenteen tutkiminen ja suunnittelu, miten sitä voidaan muokata ja laajentaa tilavarausjärjestelmän tarpeisiin.

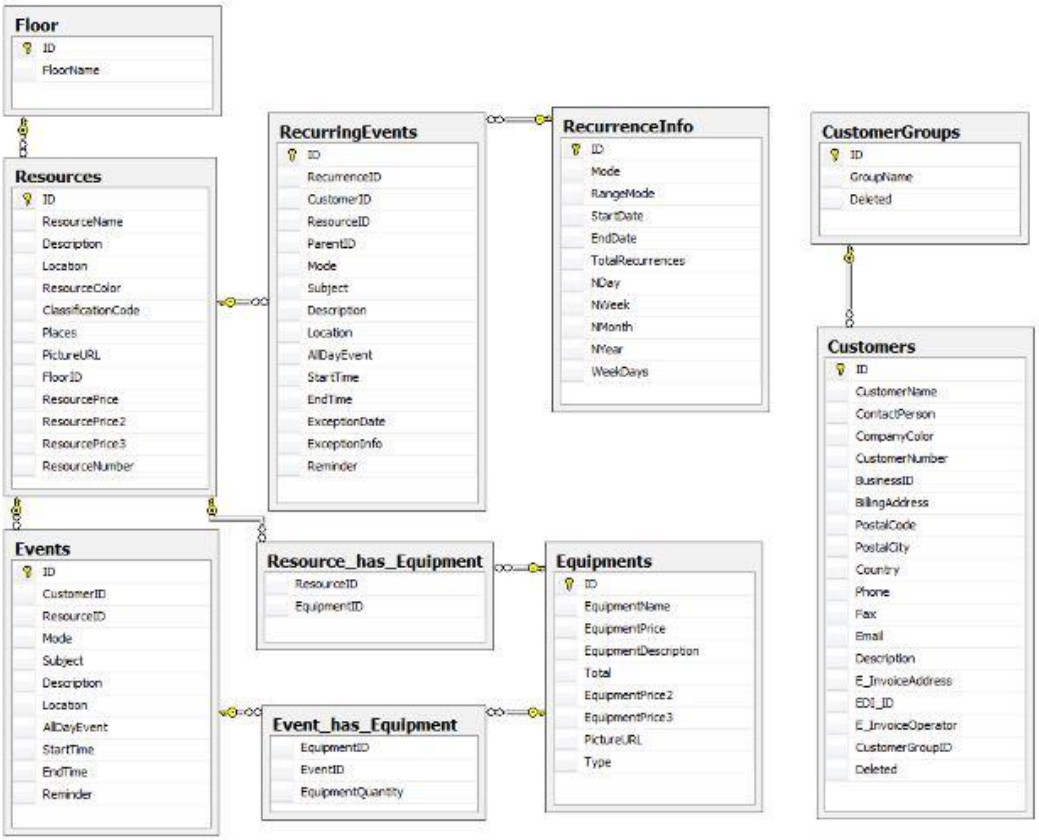
5.2.1 Taulut

Tietokannan suunnittelu ja taulujen luominen oli tämänkin ohjelmistoprojektin ensimmäisiä töitä ja vasta sen jälkeen päästiin käyttöliittymän ja lomakkeiden suunnitte-

luun. Tietokannan rakenne kuitenkin muuttui projektin edetessä ja lopullinen tietokanta poikkeaakin melko paljon ensimmäisestä versiosta. Alkuperäisen tietokannan taulujen kenttiä uudelleen nimeämällä ja kenttiä lisäämällä päästiin työssä hyvin alkuun. Käyttöliittymän ensimmäiset osat luotiin tietokannan välttämättömien osien valmistuttua. Tietokantaa laajennettiin sovellukselle asetettavien vaatimusten selkiytyessä. Uusien web-lomakkeiden ja tietokantataulujen lisääminen olikin jo nopeampaa, koska valmis lomakerunko ja tietokantayhteydet oli luotuna. Laitteiden, laiteryhmien, asiakasryhmien ja tilaryhmien vuoksi tietokantaan lisättiin alkuperäisten viiden taulun lisäksi vielä viisi taulua. Tietokannan taulut ja kentät, kuten sovelluksen ohjelmakoodikin, on kirjoitettu Profimill Engineeringin ohjeistamana englanniksi. Kun kaikki tarvittavat taulut oli tietokantaan luotu ja rakenne vaikutti toimivalta, lisättiin taulujen välille relaatiot.

Relaatioiden käyttämisellä on tärkeä osa tietokantojen eheyden säilyttämisessä. Relaatiot kytketään taulujen välille pää- ja vierasavaimilla. Esimerkkinä tilavarausjärjestelmän käyttämästä relaatiomallista olkoon asiakasryhmä ja asiakas, jolloin asiakkaalla on vierasavaimena asiakasryhmän pääavain. Tilavarausjärjestelmän relaatioissa hyödynnetään cascade-ominaisuutta, jolloin asiakasryhmää poistettaessa myös asiakkaat, joilla on kyseisen asiakasryhmän vierasavain, poistuvat tietokannasta. Sama suunnitteluperiaate toistuu koko tietokannassa. Tämä toimintamalli selkiyttää järjestelmän toimintaideaa sekä pitää yllä tietokannan eheyttä. Relaatioiden cascade-ominaisuuden hyödyntäminen helpottaa ohjelmoijan työtä, koska erillisiä delete-lauseita ei tarvita ja vierasavaimen sisältävät rivit poistuvat automaattisesti pääavaimen poistuessa.

Tässä tietokannan ylläpitämisessä tietoja poistamalla on kuitenkin huonona puolena historian häviäminen. Esimerkiksi asiakkaat tai tilat ja tilavaraukset voitaisiin haluta säilyttää tietokannassa historiatietoina. Historian luomiseksi voisi käyttää tietojen poistamisen sijaan tietokantatauluissa päivämääräkenttää, joka päivitetään update-lauseella ja merkitään päivämäärä, jolloin rivi on poistettu. Jos rivillä olisi päivämäärätieto, voisi sen poistaa halutun ajan päästä myöhemmin tai ottaa uudelleen tilavarausjärjestelmän käyttöön.



Kuva 10. Tilavarausjärjestelmän tietokantataulut.

5.2.2 Kyselyt

Tilavarausjärjestelmän lomakkeet käyttävät tiedon siirtämiseen tietokannan ja sovelluksen välillä Microsoftin SQLDataSource-komponenttia. Lomakkeiden kentistä luettavat tiedot tarkistetaan ensin selaimella ASP.NET Validator –komponenteilla. Tarkistuksen jälkeen suoritetaan syötteille tietotyyppimuunnokset jokaisen kentän tietotyypin mukaan, jonka jälkeen syötteet sijoitetaan tietokantakyselyn parametrijoukkoon. Tietokantakyselyinä käytetään tallennettuja proseduureja tietokannassa, joita SQLDataSource-komponentti suorittaa. Tietokantakyselyt sijoitettiin aluksi web-sivuille, jossa parametritkin sijaitsevat, mutta päätettiin myöhemmin siirtää tietokantaan, tallennetuiksi proseduureiksi. Tallennettujen proseduurien käyttäminen helpottaa sovelluksen, ylläpitoa, koska kaikki kyselyt ovat keskitetysti samassa paikassa ja ne myös toimivat nopeammin kuin lomakkeella sijaitsevat kyselyt. Seuraavassa esimerkki tallennetun proseduurin määrittämisestä .aspx-tiedostoon ja tallennetun proseduurin luomisesta SQL Server –tietokantaan. Esimerkin tallennettua proseduuria käytetään tilavarausjärjestelmässä laskemaan siirrettävän laitteen varausmäärä, parametreinä välitettyjen ta-

pahtuman aloitus- ja päättymisaikojen (@StartTime, @EndTime) ja siirrettävän laitteen ID-tunnisteen (@EquipmentID) perusteella.

SQLDataSource-tietolähteen ja käytettävän Select-tietokantakyselyn määrittäminen .aspx tiedostossa:

```
<asp:SqlDataSource ID="CountDevicesDataSource" runat="server"
 ConnectionString=
"<%$ ConnectionStrings:WebSchedulerDemoConnectionString%>"
SelectCommand=
"usp_CountDeviceNumber" SelectCommandType="StoredProcedure">
<SelectParameters>
 <asp:ControlParameter ControlID="GridView_Equipment" Name="EquipmentID"
PropertyName="SelectedValue" />
 <asp:Parameter Name="StartTime" Type="DateTime" />
 <asp:Parameter Name="EndTime" Type="DateTime" />
 </SelectParameters>
</asp:SqlDataSource>
```

Tallennetun proseduurin luominen SQL Server –tietokantaan:

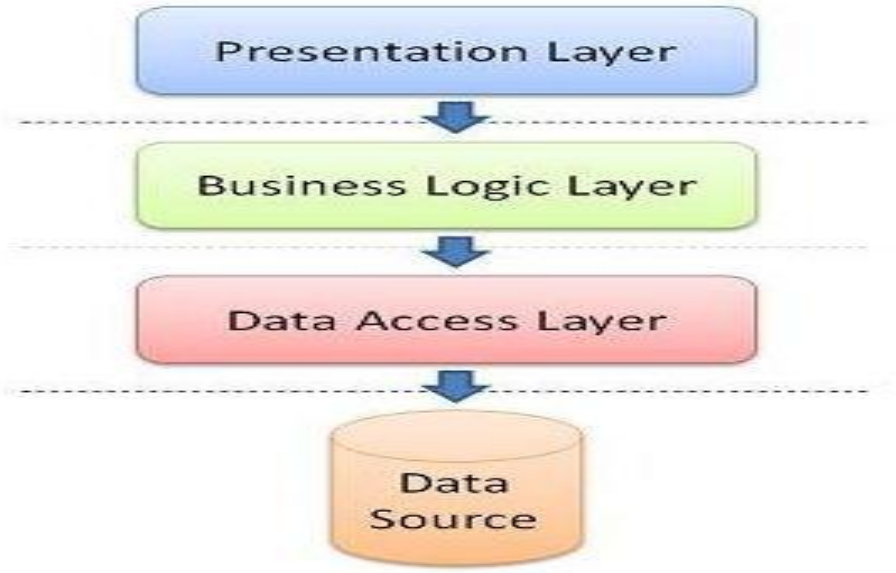
```
USE [WebSchedulerDemo]
GO
/****** Object:  StoredProcedure [dbo].[usp_CountDeviceNumber]
Script Date: 02/18/2012 15:10:40 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE  PROCEDURE [dbo].[usp_CountDeviceNumber]
@EquipmentID int,
@StartTime datetime,
@EndTime datetime
AS
SELECT SUM(EquipmentQuantity) AS sum1
FROM Event_has_Equipment
```

```
WHERE (EquipmentID = @EquipmentID)
AND (EventID  IN (SELECT ID FROM Events
WHERE (Events.EndTime > @StartTime) AND (Events.StartTime < @EndTime)))
```

Tietokantayhteyden luomiseen tarvittava yhteyslause eli connection string on sovelluksen hakemistossa sijaitsevassa web.config-tiedostossa, jossa on myös muut sovelluksen asetuksiin ja komponentteihin liittyvät tiedot. Asetusten keskittäminen web.config-tiedostoon helpottaa ylläpitoa, koska connection stringin muuttuessa tarvitsee se päivittää vain yhteen paikkaan.

Tietokannan käyttämisessä pääsivun WebScheduler 3-kalenterikomponentti hyödyntää Intersoftin ISDataSource-komponenttia. ISDataSource käyttää tietokantaa Data Access Layer -kerrokselle sijoitettujen funktioiden avulla, jotka sisältävät tietokannan kyselyt. Kuvassa 11 on kuvattu eri kerrokset, joihin sovellus voidaan jakaa Layer- eli kerrosmallia käyttäen. Data Access Layer eli yhteyskerros toimii rajapintana Presentation Layer eli esityskerroksen ja Data Layer eli tietokerroksen välillä. Yhteyskerroksessa suoritetaan tietokantayhteyden luominen sekä tietokantakyselyiden välittäminen tietokerroksella sijaitsevalle tietokantapalvelimelle. Esityskerroksella tarkoitetaan sovelluksen ylintä osaa, eli osaa, jossa sijaitsevat käyttöliittymän komponentit. Tietokerroksella tarkoitetaan sovelluksen alinta osaa, eli osaa, jossa sijaitsee sovelluksen tietojen tallentamiseen käytettävä tekniikka. Tilavarausjärjestelmän tapauksessa tietojen tallentamiseen käytettävänä tekniikkana toimii SQL Server -tietokantapalvelin ja siinä suoritettava T-SQL-tietokanta. Tilavarausjärjestelmä sisältää kuvassa 11 näkyvistä kerroksista Presentation-, Data Access- ja Data Layer -kerrokset. Kuvassa 11 näkyvä Business Logic Layer tarkoittaa sovelluslogiikkakerrosta. Sovelluslogiikkakerroksen tehtävänä on tiedon valmistelu ja välitys esityskerroksen ja yhteyskerroksen välillä. Koska tilavarausjärjestelmä on melko yksinkertainen toteutukseltaan, eikä ohjelmakoodirivejä ole erityisen runsaasti, sovelluslogiikkakerrosta ei katsottu tarpeelliseksi toteuttaa. Tietojen valmisteluun ja välittämiseen esityskerroksen ja yhteyskerroksen välillä päätettiin käyttää ASP.NET-web-sivuissa perinteisesti käytettävää ohjelmakooditiedostoa. Ohjelmakooditiedostosta käytetään alan artikkeleissa usein nimitystä Code Behind -tiedosto. Jokaiselle HTML-merkintäkielen sisältävälle ASP.NET-web-sivulle on oma ohjelmakoodisivunsa. Ohjelmakoodi-sivu periytyy HTML-

merkintäkielen sisältävästä .aspx-tiedostosta ja on tiedostopäätteeltään .aspx.vb, Visual Basic .NET –ohjelmointikieltä käytettäessä.



Kuva 11. Layer-rakenne. (14.)

5.3 Käyttäjätasot

Sovellukseen toteutettiin kaksi käyttäjätasoa: pääkäyttäjä ja vierailija. Pääkäyttäjällä on täydet luku- ja kirjoitusoikeudet, vierailijalla vain kalenterin lukuoikeus. Tilavaraukset tekee tilavarauksia hoitava henkilö, esimerkiksi aulapalvelu, joka on siis pääkäyttäjä. Tilavarausta tai alustavaa tilavarausta, jonka aulapalvelu myöhemmin vahvistaisi, ei asiakas voi tehdä. Tilavarausten keskittäminen vain aulapalvelu-henkilökunnan tehtäväksi oli sovelluksen asiakkaan toivomus.

Sovellus tullaan liittämään ohjelmistoportaaliin osaksi ohjelmistokokonaisuutta, jonka käyttäjätunnistuksen toteutus on vielä tällä hetkellä toteuttamatta. Nyt sovellus hakee käyttäjätietonsa projektin hakemistossa sijaitsevasta, sovelluksen asetukset sisältävästä web.config-tiedostosta. Web.config-tiedostosta noudettua käyttäjätietoa säilytetään Session-muuttujassa, eli muuttujassa, joka on voimassa yhden istunnon ajan. Pitkän istunnon ollessa kyseessä voidaan Session-muuttujan tietojen säilyttämisaika määrittää halutun mittaiseksi. Kalenterin toiminnot ja valikoiden rakenne luodaan dynaamisesti, eli ajonaikaisesti eri käyttäjätasoille. Jos käyttäjä yrittää internetselaimen osoite-

rivin kautta päästä sovelluksen muille web-sivuille, annetaan ilmoitus käyttöoikeuksien puuttumisesta ja ohjataan käyttäjä takaisin pääsivulle.

5.4 Lokalisointi

Tilavarausjärjestelmä ottaa kieliasetuksensa käyttäjän internetselaimen asetuksista. Tuettuja kieliä ovat tällä hetkellä suomi ja englanti. Lokalisointi toteutettiin ASP.NET-tekniikkaan sisäänrakennettuja resurssitiedostoja käyttämällä. Resurssitiedostot, eli komponenttien tekstit sisältävät tiedostot ovat XML-tiedostoja, jotka sijaitsevat Visual Studio -ohjelmointityökalun automaattisesti luomissa App_GlobalResources- ja App_LocalResources-kansioissa. App_GlobalResources-kansioon sijoitetaan sovelluksen yhteiset käännökset, käyttäjän ilmoitukset ja esimerkiksi valikoiden tekstit. App_LocalResources-kansioon sijoitetaan jokaisen sivun omat käännökset. Nämä resurssi-tiedostot Visual Studio -ohjelmointityökalu luo lokalisointia käyttöönotettaessa automaattisesti ja ohjelmoijan tehtäväksi jää ainoastaan kirjoittaa jokaiselle komponentille haluamansa teksti, haluamallaan kielellä. Resurssitiedostot voidaan päivittää kopioimalla uudet kielitiedostot projektiin IIS-palvelimelle. Kuten ASP.NET-web-sivuja eli .aspx-tiedostoja päivitettäessä, myöskään resurssitiedostoja päivitettäessä sovellusta ei tarvitse kääntää ja julkaista uudestaan, vaan nämä osat web-sivustoa käännetään ajonaikaisesti IIS-palvelimella. (15.)

5.5 Käyttöohje

Tarkemmat kuvaukset tilavarausjärjestelmän toimintojen suorittamisesta ja tilannekohtaiset kuvat sijaitsevat sovellukseen tehdyssä ja sovelluksen menusta avautuvassa 12-sivuisessa käyttöohjeessa. Käyttöohjeen tekeminen olikin opettavainen ja melko työläs vaihe, mutta ohjetta tehdessä löytyi käyttöliittymään ja toimintojen logiikkaan kehitysideoita. Samalla kun käyttöohjetta kirjoitti, joutui käymään jokaisen toiminnon uudestaan läpi. Toimintoja läpikäydessä tuli samalla suoritettua ohjelmistotestausta ja sen avulla muutama ohjelmointivirhe saatiin vielä löydettyä sekä korjattua.

6 YHTEENVETO JA LOPPUPÄÄTELMÄ

Projekti valmistui ajallaan ja täyttää sille asetetut vaatimukset. Lähtötilanteessa oli tarkoitus toteuttaa sovellukseen myös Crystal Reports -raportointi, mutta asiakkaan

sovellukselle asettamien lisäominaisuuksien toteuttaminen vei raportoinnin toteuttamiseen varatun ajan. Sovellukseen tuli asiakkaan suunnalta myös muutamia muita toteuttamatta jääneitä lisätoiveita, esimerkiksi laskutustietojen luonti ja tulostus. Näiden lisäominaisuuksien suunnitteleminen, toteuttaminen ja testaaminen vaatii melko paljon aikaa ja ne päätettiin toteuttaa omana projektinaan myöhemmin.

Työ oli mielestäni teknisesti vaativa ja vaati ohjelmointiosaamista laajalta alalta. Projektia tein Profimill Engineering Oy:n toimitiloissa ja sain hyviä neuvoja ja tukea yrityksen työntekijöiltä sovelluksen kehitystyölle. Koska aiempaa kokemusta minulla ei tämänkaltaisen sovelluksen toteuttamisesta ollut, ei yrityksessä sovelluksen kehittämiseen käytetty aika riittänyt vaan lähes yhtä paljon aikaa vietin myös kotona tämän sovelluksen parissa.

Lopputulokseen voin olla tyytyväinen ja vaikka joitain sovelluksen osa-alueita haluaisinkin nyt toteuttaa toisella tavalla, on se vain merkki siitä, että ohjelmoinnin parissa oppii koko ajan uutta ja asiat voi tehdä kokemuksen karttuessa aina paremmin. Tätä kirjoittaessa olen jo siirtynyt kyseisessä yrityksessä muihin ohjelmointitehtäviin, mutta palaan myös opinnäytetyönä kehittämäni sovelluksen pariin myöhemmin, toteuttamaan siitä pois jääneet lisäominaisuudet. Tämä opinnäyteraportti on kirjoitettu suurimmaksi osaksi projektin toteuttamisen jälkeen ja olenkin sitä mieltä, että parasta olisi kirjoittaa raporttia mahdollisuuksien mukaan samaan aikaan opinnäytetyön toteuttamisen aikana.

LÄHTEET

1. MSDN. Client Script in ASP.NET Web Pages. Saatavissa: http://msdn.microsoft.com/en-us/library/3hc29e2a.aspx [viitattu 11.1.2012].

2. MSDN. Web Services. Saatavissa: http://msdn.microsoft.com/en-us/library/ms950421.aspx [viitattu 11.1.2012].

3. Naspinski, Stan. 2009. Asp.Net vs php: speed comparison. Saatavissa: http://naspinski.net/post/AspNet-vs-php--speed-comparison.aspx [Viitattu 26.12.2011]

4. MSDN. Active Directory Authentication from ASP.NET. Saatavissa: http://msdn.microsoft.com/en-us/library/ms180890%28v=vs.80%29.aspx [viitattu 26.12.2011].

5. Liberty ja Hurwitz. 2002. Programming ASP.NET. O'Reilly & Associates. 1. painos. ISBN: 0-596-00171-1

6. Tiwari, Rajeev. 2010. ASP.NET Page Life Cycle. Saatavissa: http://rajeevdotnet.blogspot.com/2010/07/aspnet-page-life-cycle.html [viitattu 26.12.2011].

7. Chapple, Mike. SQL Server Stored Procedures. Saatavissa: http://databases.about.com/od/sqlserver/a/storedprocedure.htm [viitattu 26.12.2011].

8. Google Code University. AJAX Tutorial. Saatavissa: http://code.google.com/edu/ajax/tutorials/ajax-tutorial.html [viitattu 26.12.2011].

9. Cate, Scott. 2008. Understanding ASP.NET AJAX UpdatePanel Triggers. Saatavissa: http://www.asp.net/web-forms/tutorials/aspnet-ajax/understanding-asp-net-ajax-updatepanel-triggers [viitattu 26.12.2011].

10. AjaxToolkit. ASP.NET Ajax Control Toolkit. Saatavissa: http://www.ajaxtoolkit.net/ValidatorCallout/ValidatorCallout.aspx [viitattu 13.2.2012].

11. MSDN. Microsoft Visual Studio 2010. Saatavissa: http://msdn.microsoft.com/en-us/library/52f3sw5c.aspx [viitattu 10.1.2012].

12. Microsoft. Microsoft SQL Server Management Studio 2008 R2. Saatavissa: http://www.microsoft.com/download/en/details.aspx?id=22985#overview [viitattu 10.1.2012].

13. Haikala, I & Mikkonen, T. 2011. Ohjelmistotuotannon käytännöt. 12., uudistettu painos. ISBN: 978-952-14-1754-2.

14. Graig, S. 2010. Saatavissa: http://stackoverflow.com/questions/2011698/3-tier-architecture-in-need-of-an-example [viitattu 19.1.2012].

15. MSDN. ASP.NET Web Page Resources Overview. Saatavissa: http://msdn.microsoft.com/en-us/library/ms227427.aspx [viitattu 12.2.2012].