MICROSOFT .NET -SOVELLUSKEHITYSARKKITEHTUURI
Case: UPM IT Client Security

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2006
Petri Puustinen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

PUUSTINEN, PETRI: Microsoft .NET-sovelluskehitysarkkitehtuuri
Case: UPM IT Client Security

Ohjelmistotekniikan opinnäytetyö, 41 sivua

Kevät 2006

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee Microsoft .NET -sovelluskehitysarkkitehtuuria ja erityisesti siihen kuuluvaa ASP.NET-teknologiaa. Työn case-osiossa suunnitellaan ja toteutetaan UPM-Kymmenen tietohallinnossa toimivan työasemaryhmän käyttöön selainpohjainen raportointisovellus. Työn tavoitteena on selvittää case-toteutuksen kautta, voidaanko UPM:n työasemaympäristön hallintasovellukset toteuttaa tulevaisuudessa .NET-teknologian tarjoamin keinoin.

Työasemaryhmä on käyttänyt pääasiassa Lotus Notes -ympäristöön toteutettuja hallintasovelluksia ja myös työssä uusittavasta sovelluksesta on Lotus Notes -versio. Tämä raportointisovellus haluttiin uusia siinä ilmenneiden käytettävyysongelmien takia, joista suurimpana oli sovelluksen hitaus jokapäiväisessä käytössä.

Case-osiossa suunniteltava ja toteutettava sovellus perustuu ASP.NET-teknologiaan. Sitä käytetään työasemien ja palvelimien käyttöjärjestelmä-, virustorjunta- ja palomuuripäivitysten raportointiin sekä päivitysten asentamiseen. Sovellus tulee toimimaan yrityksen intranetissä ja sen tietokantana käytetään SQL Server 2005 Standard -versiota.

Case-sovellukseen määritetyt toiminnot oli mahdollista toteuttaa .NET-sovelluskehitysympäristössä, joten sovellukselle asetetut toiminnalliset tavoitteet saavutettiin. Sovellusta ei otettu vielä tuotantokäyttöön, joten käyttökokemuksia sovelluksen käytettävyydestä ei ollut saatavilla.

Avainsanat: www-ohjelmointi, tietokannat, .NET Framework, ASP.NET

Lahti University of Applied Sciences
Faculty of Technology

PUUSTINEN, PETRI: Microsoft .NET application development architecture
Case: UPM IT Client Security

Bachelor's Thesis in Software Engineering, 41 pages

Spring 2006

ABSTRACT

This thesis deals with the Microsoft .NET software development architecture and especially ASP.NET technology which is a part of the .NET architecture. The thesis also covers the designing and implementing of a browser based reporting application for the workstation management team of UPM-Kymmene. The goal of the thesis was to determine if it could be possible to implement all UPM-Kymmene's workstation management applications with the techniques of the .NET technology in the future.

The workstation management team has used management applications which are mainly implemented to the Lotus Notes environment and also the reporting application has the Lotus Notes version. The reason why the company wanted to renew this reporting application was the issues concerning its usability. The main problem has been its slowness in daily usage.

The reporting application which was designed and implemented in the practical part of the thesis is based on the ASP.NET technology. It is used for reporting the updates of the operating system, antivirus software and firewall software of the workstations and servers, as well as for installing the updates. The application will operate in the company's intranet and the database for this application is Microsoft SQL Server 2005 Standard.

It was possible to implement the functionality which was defined for the practical application with .NET software development architecture, so the functional goals of the thesis were reached. The application has not been implemented to the production environment yet, so no results about usability are available.

Keywords: www programming, databases, .NET Framework, ASP.NET

SISÄLLYS

TERMIT JA LYHENTEET

UPM IT Configuration Management
UPM IT:n työasemaryhmän käytössä oleva laiterekisteri

UPM IT Client Security
UPM:n työasemaympäristön raportointisovellus liittyen tietoturvaan

IP – Internet Protocol (IP-osoite)
Tietokoneverkkoon liitetyn tietokoneen tai aktiivilaitteen looginen yksilöivä osoite

DHCP – Dynamic Host Configuration Protocol
Verkkoprotokolla, jonka avulla voidaan Ethernet-verkkoon liitetylle koneelle / aktiivilaitteelle jakaa dynaamisesti IP-osoite

DNS – Domain Name Service (nimipalvelu)
Palvelu, jonka tarkoituksena on muuntaa tietoverkoissa käytettävät selkokieliset nimet IP-osoitteiksi ja päinvastoin

DC – Domain Controller (toimialueen ohjauspalvelin)
Windows-toimialueella toimiva palvelin, jonka tehtävänä on mm. tehdä

AD – Active Directory (aktiivihakemisto)
Tietokanta Windows-toimialueella toimiville käyttäjä- ja laitetileille ja käyttäjäryhmille

CLR – Common Language Runtime
Common Language Runtime on .NET-ohjelmien ajoympäristö

MSIL – Microsoft Intermediate Language
Microsoft Intermediate Language on Microsoftin kehittämä välikieli, jota kaikki .NET-ohjelmointikielien kääntäjät tuottavat

JIT – Just-In-Time

Ohjelmointikielenkääntäjä, jonka avulla MSIL-välikieli käännetään konekieleksi koodiksi

FCL – Framework Class Library

.NET-sovelluskehitysympäristön luokkakirjasto

CTS – Common Type System

Määrittelee tiettyjä sääntöjä .NET-sovelluksissa käytettäville tyypeille

CLS – Common Language Specification

Asettaa sääntöjä .NET-ympäristössä käytettäville ohjelmointikielille

IIS – Internet Information Services

Microsoftin tarjoama selainpohjaisten sovellusten suoritusympäristö

HTML – Hyper-Text Markup Language

Merkkauskieli, jota käytetään www-sivujen käyttöliittymien luomiseen

Script-kieli

Ohjelmointikieli mm. yksinkertaisten käyttöjärjestelmätoimintojen ja sovellusten automatisointiin

Mono-projekti

Tarjoaa tarvittavat ohjelmistot .NET-sovellusten rakentamiseen Linux-, Unix-, Solaris- ja MacOS-käyttöjärjestelmille

Kernel

Käyttöjärjestelmän ydin. Sisältää käyttöjärjestelmän alimman tason palveluita

1 JOHDANTO

Tämä opinnäytetyö käsittelee Microsoft .NET -sovelluskehitysarkkitehtuuria ja erityisesti siihen kuuluvaa ASP.NET-teknologiaa. Työn case-osiossa suunnitellaan ja toteutetaan UPM-Kymmenen tietohallinnossa toimivan työasemaryhmän käyttöön selainpohjainen raportointisovellus. Sovellusta käytetään työasemien ja palvelimien käyttöjärjestelmä-, virustorjunta- ja palomuuripäivitysten raportointiin sekä päivitysten asentamiseen. Työn tavoitteena on selvittää case-sovelluksen kautta, voidaanko UPM:n työasemaympäristön hallintasovellukset toteuttaa tulevaisuudessa .NET-teknologian tarjoamin keinoin.

Microsoft .NET on uusin Microsoftin tarjoama sovelluskehitysarkkitehtuuri. Arkkitehtuurista julkaistiin ensimmäisen version vuonna 2002 ja työn tekohetkellä uusin saatavilla ollut versio oli 2.0. Työssä toteutettava sovellus tehdään .NET 2.0 -version tarjoamin keinoin.

.NET-teknologian avulla voidaan rakentaa monia erityyppisiä sovelluksia, joita ovat mm. Windows-käyttöjärjestelmässä toimivat asiakas- ja selainsovellukset, verkko- ja Windows-palvelut sekä Microsoft Office tuotteisiin liittyvät sovellukset ja laajennukset. Kaikkia näitä sovelluksia voidaan ohjelmoida monilla eri .NET-ympäristöön toteutetulla ohjelmointikielellä.

UPM-Kymmene työasemaryhmä on käyttänyt pääasiassa Lotus Notes-ympäristöön toteutettuja hallintasovelluksia ja myös työssä suunniteltavasta ja toteutettavasta sovelluksesta on käytössä Lotus Notes -versio. Tämä sovellus haluttiin uusia siinä ilmenneiden käytettävyysongelmien takia, joista ongelmallisimpana oli sovelluksen hitaus jokapäiväisessä käytössä.

Työssä suunnitellusta ja toteutetusta sovelluksesta tehtiin ASP.NET 2.0 -selainsovellus. Sovellukseen liittyy myös muita .NET-teknologian avulla toteutettuja komponentteja, kuten Windows Service –sovellus. Sovelluksen tietokannaksi valittiin SQL Server 2005 Standard, joka oli työn tekohetkellä uusi saatavilla ollut versio SQL Server -tietokantapalvelimesta. Sovellus toimii yrityksen intranetissä.

Työ rajattiin käsittelemään Microsoft .NET -sovelluskehitysarkkitehtuuria. Tarkemmin tästä arkkitehtuurista käsitellään sen web-sovelluksien toteutukseen tarkoitettua ASP.NET-teknologiaa.

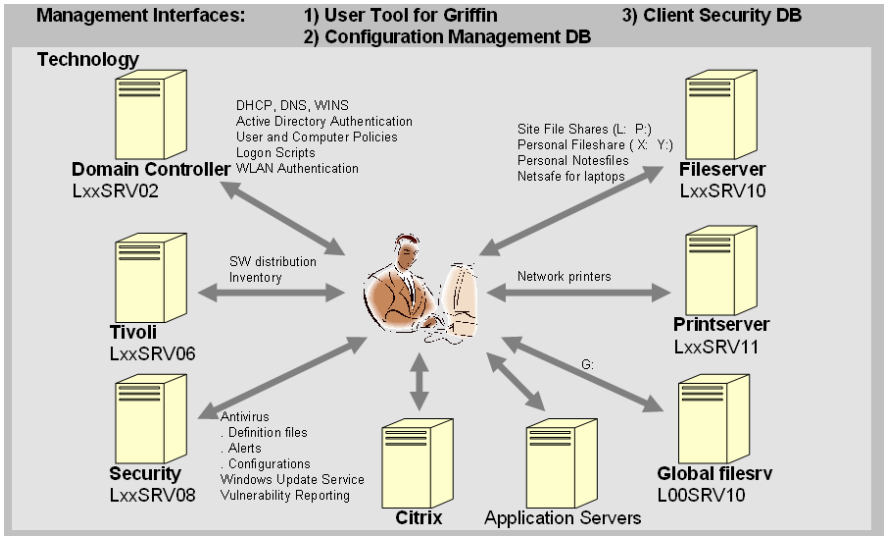
UPM on yksi maailman johtavia paperiyhtiöitä, jonka liiketoiminta keskittyy aikakauslehtipapereihin, sanomalehtipapereihin, hieno- ja erikoispapereihin, jalostusmateriaaleihin sekä puutuotteisiin. Tuotantolaitoksia yhtiöllä on 15:sta maassa ja kattava myynti- ja jakeluverkosto yli 70 maassa, yhteensä yli 170 edustajaa. UPM:n liikevaihto vuonna 2005 oli lähes 9,3 miljardia euroa ja sen palveluksessa työskentelee noin 31 000 henkilöä. Yhtiön osakkeet on listattu Helsingin ja New Yorkin pörsseissä. (UPM-Kymmene, 2006.)

2 NYKYTILANNE

2.1 UPM-Kymmenen työasemaympäristö

UPM-Kymmenen työasemaympäristön kehitys ja ylläpito kuuluvat yrityksen IT-organisaation Workstation Management tiimille. Tiimi on vienyt läpi projektin, jonka tärkeimpinä tavoitteina on ollut yhtenäistää ja standardisoida UPM:n työasemaympäristö ja sen hallinta. Projektin myötä UPM-Kymmenen yksiköillä on ollut mahdollista ottaa käyttöön WIS-palvelu (Workstation Infrasructure Services), joka tarjoaa toimivan konseptin työasemaympäristöjensä hallintaan. Palvelun kautta yksiköt saavat käyttöönsä erilaisia ympäristöön räätälöityjä työkaluja mm. käyttöjärjestelmä- ja sovellusasennuksille sekä työasema- ja käyttäjähallintaan. Palvelun käyttöönottavat yksiköt ovat tämän jälkeen myös osa suurempaa, keskitetysti hallittua työasemaympäristöä. Maaliskuussa 2006 n. 90 %:a kaikista yksiköistä kuului WIS-palvelun piiriin.

Yrityksen työasemaympäristö toimii pääasiassa Microsoft Windows-ympäristössä. Lähes kaikissa työasemissa on käytössä joko Windows XP- tai 2000-käyttöjärjestelmä ja suurin osa palvelimista on asennettu joko Windows Server 2000- tai 2003-ympäristöön.



KUVIO 1. Griffin-toimialueen palvelin ympäristö

Kaikki työasemat ja palvelimet kuuluvat Griffin-toimialueeseen, joka on kuvattu kuviossa 1. Ympäristön työasemahallintaan liittyviä palvelimia ovat toimialueen ohjauskoneet, tiedosto-, tulostus-, sovellus-, tietoturva-, citrix- ja tivolipalvelimet. Ohjauskoneet tarjoavat toimialueelle Active Directory-hakemistopalvelun, jonka kautta ylläpidetään käyttäjien, työasemien ja palvelimien tilitietoja. Muita ohjauskoneen tarjoamia palveluita ovat mm. DHCP (Dynamic Host Configuration Protocol)-, DNS (Domain Name Service)- ja autentikointipalvelut. Tiedostopalvelimilla sijaitsevat kaikkien käyttäjien kotihakemistot sekä yleiset ja yksikkökohtaiset tiedostojaot. Verkkotulostuspalvelut on toteutettu tulostuspalvelimilla, joissa on määriteltyinä kaikkien verkkotulostimien tulostusjonot. Tulostusjonot ovat käyttäjäkohtaisia ja niitä voidaan lisätä käyttäjille erillisen työkalun avulla. Sovelluspalvelimet toimivat joidenkin selainpohjaisten intranetsovellusten ja Lotus Notes-sovellusten suoritusympäristönä. Ohjelmistojen etäkäyttömahdollisuudet on toteutettu Citrix-ympäristön tarjoamilla palveluilla, jossa sovellukset ovat asennettuina palvelimelle ja niitä käytetään työasemaan asennetun etäkäyttösovelluksen kautta. Tietoturvapalvelimien avulla asennetaan työasemien käyttöjärjestelmäpäivitykset sekä virustorjunta- ja palomuurisovellusten päivitykset. Työasemiin asennettavat sovellukset jaetaan Tivoli-palvelimen avulla.

Griffin-toimialueelle asennettavien työasemien käyttöjärjestelmä- ja ohjelma-asennukset on automatisoitu ns. vastaustiedostojen avulla. Kaikki asennusmediat sijaitsevat asennuspalvelimella, josta automaattiasennus hakee tarvittavat osat työasemaa asennettaessa. Tällä asennusmetodilla työaseman käyttöjärjestelmä, ohjelmat ja monet työasemakohtaiset asetukset saadaan asennettua ilman että asentajan tarvitsee puuttua asennukseen sen aikana.

Työasemien ja palvelimien elinkaaren aikana niitä hallitaan UPM IT Configuration Management-sovelluksella, jonka kautta voidaan suorittaa lähes kaikki hallintaan liittyvät tehtävät.

2.2 Sovellusympäristö

Työasemahallintaan liittyy muutamia sovelluksia, joista tärkeimpinä ovat Lotus Notes-ympäristöön rakennetut UPM IT Configuration Management- ja UPM IT Client Security-sovellukset. Tässä luvussa käsitellään Configuration Management-sovelluksen ja muiden pienempien työasemahallintaan liittyvien sovellusten toimintaa. Client Security-sovelluksen toimintaa käsitellään tarkemmin seuraavassa luvussa.

UPM IT Configuration Management-sovelluksen pääasiallisena tarkoituksena on toimia Griffin-toimialueella toimivien laitteiden laiterekisterinä. Tämän sovelluksen tietokantaan tallennetaan jokaisesta yritykselle hankitusta laitteesta laitteen perustiedot, joita työasemaryhmä ylläpitää. Kun laitteen perustiedot on lisätty kantaan, on se mahdollista ottaa käyttöön luomalla siitä aktiivinen komponentti, jota voidaan käyttää esimerkiksi työaseman konfiguraatiotiedoissa.

Jokaiselle työasemalle luodaan oma konfiguraatiolomake, josta on esimerkki kuviossa 2. Konfiguraatiolomakkeelta selviää mm. työaseman rooli, omistaja, yksikkö, siihen liitetyt komponentit ja asennettavat ohjelmat sekä muut työasemaan liittyvät perustiedot. Valitun yksikön perusteella työasemaan asennetaan yksikkökohtaiset sovellukset sekä asetetaan muita yksikkökohtaisia asetuksia. Kun kaikki tarvittavat asetukset ja tiedot on täytetty, luodaan näistä tiedoista työasemakohtaiset vastaustiedostot. Asennustiedostojen luonnin jälkeen työasema voidaan asentaa.

Konfigurointiasetusten ja laitteiden perustietojen lisäksi sovelluksella hallitaan myös yksikkö-, paikkakunta-, ja käyttäjätietoja sekä ohjelmistoasennuspaketteihin liittyviä pyyntöjä. Käyttäjistä sovellukseen tallennettavia tietoja ovat esim. käyttäjän Windows-ryhmät, tulostusjonot ja sähköpostipalvelin sekä sähköpostilaatikon koko. Yksiköistä ja paikkakunnista tallennetaan mm. yksiköiden vastuuhenkilöt, verkko-osoiteavaruudet, osastot ja ympäristöön liittyvät palvelimet.

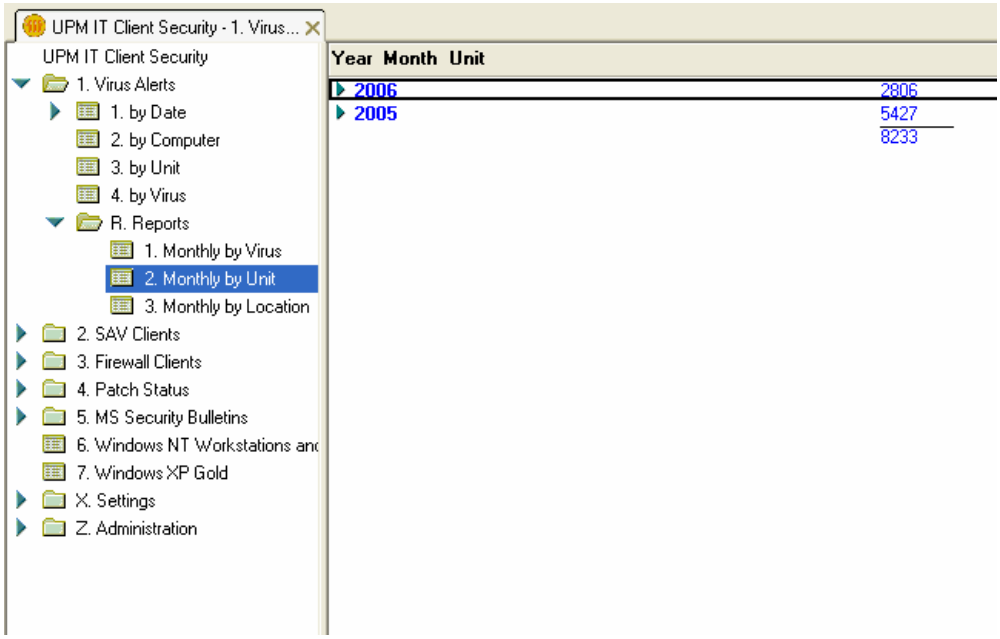
KUVIO 2. Työaseman konfiguraatiolomake

Configuration Management-sovellus toimii myös työasematietojen raportointisovelluksena. Sovelluksen kautta voidaan lukea mm. raportteja laitemääristä, asennetuista ohjelmista, tulostimista ja verkkolaitteista.

Muita ympäristön hallintaan käytettäviä työkaluja ovat mm. Active Directory Users and Computers, jolla hallitaan toimialueeseen kuuluvien työasemien ja palvelimien tilitietoja. Tällä työkalulla voidaan ylläpitää myös käyttäjätilejä ja Windows-ryhmiä. Asennettavien ohjelmistojen asennukset automatisoidaan erilaisten vastaustiedostojen avulla, ja näitä paketteja hallitaan Tivoli-ympäristöön kuuluvalla sovelluksella. Ohjelma asennetaan työasemaan automaattisesti, ja asennuksen aikana työaseman käyttäjä voi työskennellä normaalisti.

2.3 UPM IT Client Security

Client Security-sovellus toimii ensisijaisesti raportointityökaluna työasemien ja palvelimien käyttöjärjestelmäpäivityksille, virushälytyksille sekä virustorjunta- ja palomuurisovelluksien versio-/ päivitystiedoille. Sovelluksen kautta on mahdollista asentaa työasemiin ja palvelimiin päivityksiä sekä hallita näiden virustorjunta- ja palomuurisovelluksia. Suurin osa näistä toiminnoista on toteutettu ns. komentotiedostoilla, jotka ovat asennettuina jokaiseen työasemaan ja palvelimeen. Nämä komentotiedostot sisältävät tarvittavat Windows-komennot tehtävien suorittamiseksi ja niitä voidaan käynnistää kohdekoneessa tämän sovelluksen kautta. Loput näistä toiminnoista on toteutettu IBM:n tarjoamalla komentorivipohjaisella postemsg-sovelluksella. Tätä sovellusta ajetaan paikallisesti koneelta, jossa Client Security-sovellusta käytetään ja sille välitetään parametreina tiedot suoritettavista komennoista ja sovelluksista kohdekoneessa. Client Security-sovelluksen päänäkymä on esitetty kuviossa 3.



KUVIO 3. UPM IT Client Security

Sovelluksessa on myös ns. virushälytin, jonka avulla työasemassa tai palvelimessa havaitusta viruksesta saadaan tieto sähköpostilla Client Security-sovellukseen. Ilmoituksesta lähetetään tieto eteenpäin yksikön vastuuhenkilöille, jotta tarvittavat toimenpiteet virustartunnan saaneessa koneessa voidaan suorittaa. Tästä viestistä selviävät mm. työaseman nimi, yksikkö ja vastuuhenkilö sekä viruksen nimi, suoritetut toimenpiteet ja tiedosto, johon virus on tallentunut.

Tiedot työasemien ja palvelimien päivitystilanteista tulevat sovellukseen myös sähköpostilla. Nämä sähköpostit lähetetään ajastetusti tietoturvapalvelimilta, ja niissä on koottuna kaikki tietoturvapalvelimelle lähetysaikaan mennessä työasemilta ja palvelimelta lähetetyt tilanneraportit. Sovelluksessa sähköpostin sisältö parsitaan ja tiedot päivitetään sovelluksen tietokantaan.

2.4 Ongelmat nykysovelluksissa

Configuration Management- ja Client Security-sovelluksissa suurimmaksi ongelmaksi on osoittautunut hitaus johtuen suurista dokumenttimääristä. Client Security-sovelluksessa dokumentteja on jo lähes 200000 kappaletta, ja Configuration Management-sovelluksessakin on jo ylitetty 100000 dokumentin raja. Tästä syystä sovelluksien käyttö on ajoittain erittäin hidasta. Yhtenä ongelmana on myös se, että samaa tietoa on tallennettuna kumpaankin sovellukseen. Tästä ongelmasta päästäisiin eroon yhdistelemällä sovelluksia.

Työasemaympäristön hallintaa varten on tehty monia työkaluja. Täten hallintaan vaaditaan monenlaista osaamista ja monia eri tekniikoita, joiden kautta kokonaisuuden hallinta monimutkaistuu huomattavasti. Näiden ongelmien ratkaisemiseksi käytettävien tekniikoiden määrää pyritään minimoimaan ja myös yhdistelemään tiettyjä sovelluksia suuremmiksi kokonaisuuksiksi.

2.5 Lotus Notes -sovelluskehitys

2.5.1 Yleistä

Lotus Notes on työryhmäohjelmisto, jonka avulla työryhmien jäsenet voivat jakaa tietoa keskenään ajasta ja paikasta riippumatta. Työryhmiä varten tähän ympäristöön voidaan toteuttaa erillisiä sovelluksia Lotuksen omilla sovelluskehitysvälineillä. Työryhmäohjelmistojen juuret ulottuvat vuoteen 1989, jolloin Lotus Notes versio 1.0 ilmestyi markkinoille, joten Lotus Notesia voidaan pitää eräänlaisena pioneerina tällä alueella. Lotus Notes-työryhmäohjelmistoa ja tähän liittyvää Domino-palvelinympäristöä tarjoaa nykyään IBM, joka osti Lotuksen vuonna 1995. Ympäristöön on mahdollista luoda sovelluksia, joita käytetään Notes-ohjelmiston ja/tai internetselaimen kautta. (Virtala, 2003, 2.)

Lotus Domino-arkkitehtuurin yhtenä suurimpana etuna on sovellusten replikointiominaisuus. Notes-sovellus voidaan julkaista monelle eri Domino-palvelimelle, jonka jälkeen kaikki sovelluksessa tapahtuvat muutokset kopioidaan automaattisesti näiden ilmentymien välillä. Tämän ominaisuuden avulla sovelluksen käyttö on nopeampaa, koska käyttökuorma jakautuu eri palvelimelle ja verkkokuorma pienenee käyttäjien käyttäessä lähintä palvelinta. Sovelluksesta saadaan tämän ominaisuuden myötä myös vikasietoisempi, koska sovelluksen käyttö ei ole riippuvainen vain yhden palvelimen toiminnasta. (Virtala, 2003, 9.)

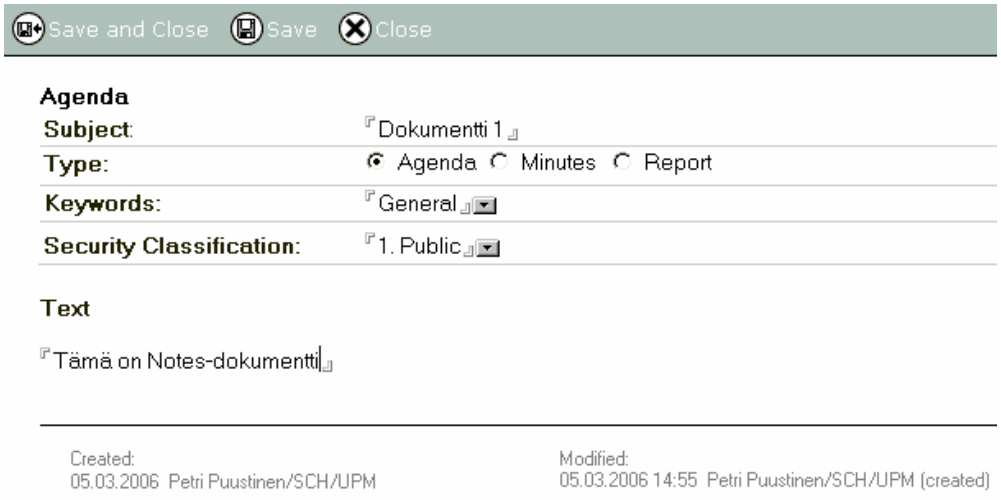
UPM:llä on käytössä satoja Notes-sovelluksia. Suurinta osaa näistä sovelluksista käytetään Notes-ohjelmiston kautta, mutta käytössä on myös joitakin internetselaimen kautta käytettäviä ratkaisuja.

2.5.2 Ohjelmointi

Lotus Notes-sovelluksia tehdään Lotus Designer -sovelluskehitysvälineellä. Sovelluksia Notes-ympäristöön voidaan tehdä monilla eri ohjelmointikielellä. Formula ja LotusScript ovat pääkieliä ohjelmoitaessa joko Notes-ohjelmiston kautta käytettäviä asiakassovelluksia tai internetselaimella käytettäviä selainsovelluksia.

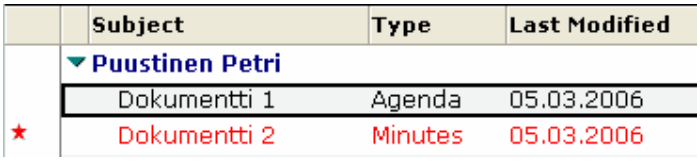
Muita mahdollisia kieliä ovat Java ja JavaScript, joita käytetään pääkielten lisäksi pääasiassa selainsovellusten ohjelmointiin. Notes-sovelluskehityksen etuina ovat mm. helposti opittava sovelluskehitysmalli sekä mahdollisuus käyttää yhtä ja samaa sovellusta sekä Notes-ohjelmistolla että internetselaimella.

Notes-sovelluksen pääkomponentteina toimivat lomakkeet ja näkymät. Lomakkeiden avulla käyttäjiltä kerätään haluttu tieto, minkä jälkeen lomakkeiden tietoihin päästään käsiksi näkymien kautta. Notes-lomakkeesta on esitetty esimerkki kuviossa 4. Lomakkeille voidaan sijoittaa erilaisia kontrolleja, joilla tieto saadaan kerättyä käyttäjältä sovelluksen tietokantaan. Kontrolleja ovat mm. tekstikentät, valintalistat, nappulat, linkit ja upotetut näkymät.



KUVIO 4. Esimerkki Notes-lomakkeesta

Näkymissä näytetään yleensä tärkeimmät lomakkeen identifioivat tiedot rivitasolla ja tätä kautta haluttuun lomakkeeseen päästään käsiksi. Lomakkeita voidaan myös hakea tekstihauilla tietokannasta näkymien kautta. Kuviossa 5 on esitetty esimerkki Notes-näkymästä.



KUVIO 5. Esimerkki Notes-näkymästä

UPM:n Notes-sovellukset rakennetaan UPM:lle räätälöidyn mallipohjan mukaan, jossa suurin osa sovelluksen peruselementeistä on määritelty ja toteutettu valmiina. Tämän mallipohjan avulla sovellusten kehitys nopeutuu ja kaikille sovelluksille saadaan yhtenäinen ulkoasu sekä sovellusten perustoiminnot ovat yhtenäisiä. Tätä kautta käyttäjien on helpompi omaksua uudet sovellukset, ja sovellusten hallinta ja ylläpito helpottuvat.

2.5.3 Tietokanta

Notes-sovelluksien tietokantana käytetään dokumenttipohjaista tietokantaa ja se eroaa paljon relaatiotietokannoista. Dokumenttipohjaisessa ratkaisussa tietokanta on sidottuna tiukasti sovelluksen toimintaan ja tiedon esittämiseen käyttöliittymässä toisin kuin relaatiotietokantaa käyttävien sovelluksien tapauksissa, joissa esitys- ja tietokantakerros ovat selkeästi erotettuina toisistaan. Tiedon tallennuksen perusyksikkönä tässä tekniikassa on dokumentti. Notes-tietokannan vahvuuksia ovat:

- kenttien kokoja ja tietotyyppejä ei tarvitse erikseen määritellä
- tyhjät kentät eivät vie tilaa tietokannassa
- sisäänrakennettu tekstihaku (Lotus Notes 2006).

3 MICROSOFT .NET -SOVELLUSKEHITYSARKKITEHTUURI

3.1 Yleistä

Microsoftin uusin sovelluskehitysarkkitehtuuri on nimeltään .NET. Arkkitehtuuri esiteltiin vuonna 2000, ja Microsoft julkaisi ensimmäisen version tästä vuonna 2002. Marraskuussa 2005 julkaistu versio 2.0 toi mukanaan monia uusia ominaisuuksia ja parannuksia.

.NET-teknologian avulla voidaan rakentaa monia erityyppisiä sovelluksia, joita ovat mm. Windows-käyttöjärjestelmässä toimivat asiakas- ja selainsovellukset, verkko- ja Windows-palvelut sekä Microsoft Office tuotteisiin liittyvät sovellukset ja laajennukset. Tässä työssä toteutettavasta sovelluksesta tehdään ASP.NET-selainsovellus, joten työssä tullaan käsittelemään tarkemmin juuri ASP.NET-teknologiaa. .NET-sovelluksia voidaan ohjelmoida monilla eri kielillä. Microsoftin tarjoamat .NET-ohjelmointikielet ovat: C#, VB.NET, Visual C++.NET, J# ja JScript. Näistä C# on tullut uutena kielenä .NET:n myötä ja on syntaksiltaan pitkälti Javan kaltainen olio-ohjelmointikieli. Visual Basic.NET on uudistettu Visual Basic ja on muuttunut puhtaaksi olio-ohjelmointikieleksi. C++:aan on lisätty .NET:n vaatimat ominaisuudet. J# on Microsoftin versio Java-kielestä ja JScript on käännettävä script-kieli, jolla voidaan automatisoida käyttöjärjestelmän ja ohjelmien toimintoja. Lisäksi yhtenä ohjelmointikielenä voidaan mainita vielä MSIL, joka on ns. välikieli, johon kaikki .NET-ohjelmointikielillä toteutetut lähdekoodit käännetään. Kolmansien osapuolten toteuttamia kieliä on n. 80 kappaletta, ja niitä tulee lisää koko ajan. Näitä kieliä ovat mm. Cobol.NET, Eiffel ja SmallTalk. (Richter 2003, xxi, 4.)

3.2 .NET-sovelluksen osat

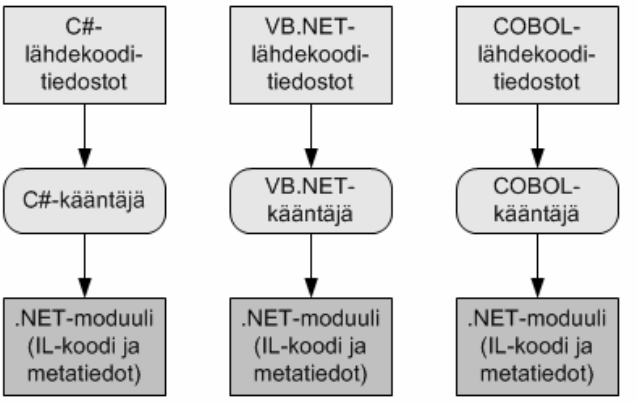
Tyypit, joita kutsutaan myös luokiksi, ovat keskeinen osa .NET-sovellusta. .NET-sovellus muodostuu pääsääntöisesti aina ohjelmoijan omista tyypeistä ja ohjelmistotalojen, kuten Microsoft, toteuttamista tyypeistä. Tyypit voivat sisältää erilaisia jäseniä, kuten mm. kenttiä, metodeja, ominaisuuksia ja tapahtumia. Kuviossa 6 on

esitetty yksinkertainen .NET-konsolisovellus, jossa on toteutettuna Testi-niminen tyyppi. Se sisältää ainoastaan yhden metodin, joka tulostaa komentoriville tekstin. (Richter 2003, 35.)

```
class Testi
{
        public static void Main(System.String[] args)
        {
            System.Console.WriteLine("Tuloste komentoriville");
        }
}
```

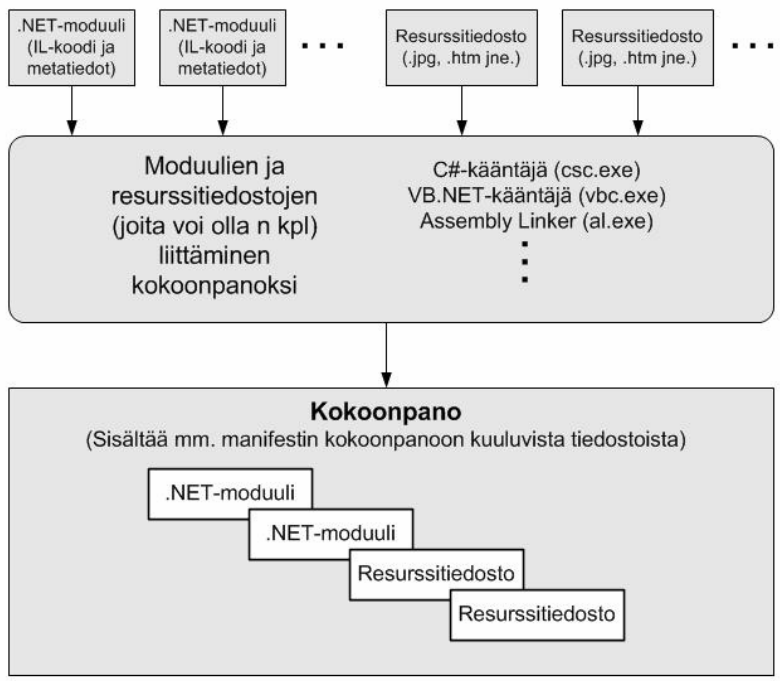
KUVIO 6. Yksinkertainen .NET-konsolisovellus

.NET-sovellusta/-komponenttia kutsutaan kokoonpanoksi (assembly). Kokoonpano on pienin yksittäinen suoritettava osa ja se voi koostua joko yhdestä tai useasta ns. hallittavasta moduulista. Yksittäinen moduuli taas voi koostua yhdestä tai useasta IL-kooditiedostosta, jotka ovat käännetty ohjelmointikielen kääntäjällä. IL-koodin muodostaminen lähdekoodista on esitetty kuviossa 7. IL-koodin lisäksi moduulit sisältävät myös ns. otsikko tietoja, joista selviää mm. sovelluksen tyyppi ja CLR-ajoympäristön versio, jolle moduuli on rakennettu. Metatiedot ovat myös yksi iso osa moduulia ja niistä selviää, mitä tyyppejä ja jäseniä lähdekoodissa on määritelty ja mihin ulkopuolisiin tyyppeihin ja jäseniin koodissa viitataan. (Richter 2003, 4-5.)



KUVIO 7. Lähdekoodin kääntäminen .NET-moduuliksi

Kokoonpanoon liittyy aina myös ns. manifest-tiedosto (rahtikirja). Manifest-tiedostosta sisältävät kokoonpanon versio- ja kulttuuritiedot. Se sisältää myös listat kokoonpanoon kuuluvista tiedostoista (moduulit, resurssit ym.) ja tiedot kokoonpanoon kuuluvista tyypeistä. Siihen voidaan liittää myös resurssitiedostoja kuten kuvia ym. (.NET Framework FAQ, 2001; Richter 2003, 7.)



KUVIO 8. .NET kokoonpanon muodostaminen

Kun kaikki moduulit ja resurssitiedostot ovat selvillä, niin ne yhdistetään Assembly Linker-ohjelmalla (AL.exe) yhdeksi kokoonpanoksi. Kokoonpanon muodostaminen on esitetty kuviossa 8.

Kokoonpanot voivat olla joko yksityisiä tai jaettuja. Yksityinen kokoonpano on yksittäisen sovelluksen käytössä, ja se on asennettuna johonkin sovelluksen omista hakemistoista. Jaettua kokoonpanoa voivat käyttää monet sovellukset ja ne sijaitsevat yleensä Global Assembly Cache-hakemistossa (GAC), joka asennetaan .NET Frameworkin mukana. Esimerkiksi kaikki .NET Framework-luokkakirjastoon kuuluvat tyypit ovat jaettuja tyyppejä, ja ovat asennettuina GAC-hakemistossa. Jotta kokoonpano voidaan asentaa GAC-hakemistoon, se pitää alle-

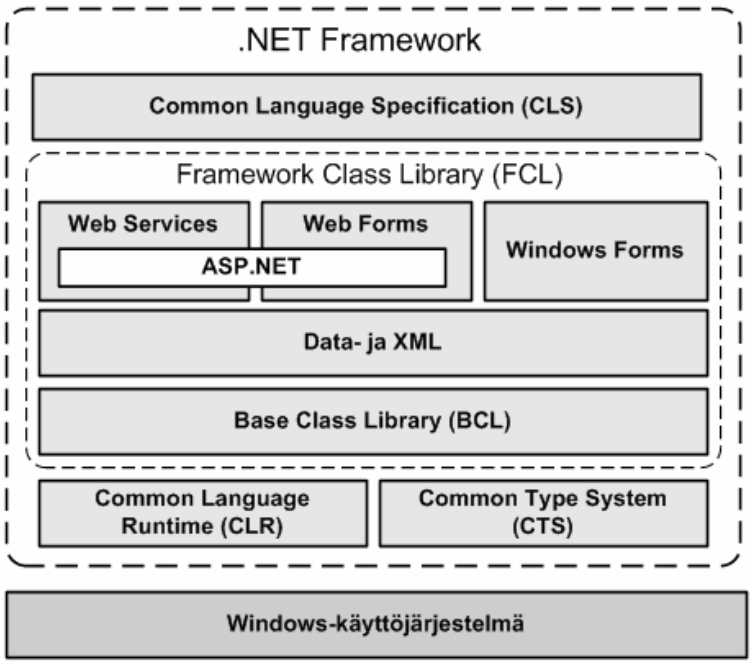
kirjoittaa ns. vahvalla nimellä. Tämä toimenpide voidaan tehdä sn.exe ohjelmalla, joka tulee Microsoft:n uusien sovelluskehitysohjelmistojen mukana. (.NET Framework FAQ 2001)

3.3 .NET Framework

3.3.1 Yleistä

.NET Framework on .NET-sovelluksien kehitys- ja ajoympäristö ja on arkkitehtuurin keskeisin kokonaisuus. Framework koostuu monesta eri osasta, joista kaksi tärkeintä ovat CLR (Common Language Runtime) ja FCL (Framework Class Libraries). Näitä osia käsitellään tarkemmin omissa luvuissaan, ja tässä luvussa käydään lyhyesti läpi muita Framework:n tarjoamia palveluita ja ominaisuuksia. .NET Frameworkin rakenne on esitetty kuviossa 9. (Richter 2003, xxi.)

.NET Framework on osa laajempaa konseptia, jonka tavoitteena on tehdä verkkopalveluiden ja tietojärjestelmien rakentaminen helpoksi. Se asennetaan automaattisesti Windows Server 2003-käyttöjärjestelmän mukana. Muihin Microsoftin vanhempiin Windows-käyttöjärjestelmiin, alkaen Windows 98:sta, se voidaan asentaa erikseen Microsoft:n sivuilta ladattavalla asennuspaketilla, joka on ilmainen. .NET Framework:stä on toteutettu avoimen lähdekoodin versiot myös seuraaviin käyttöjärjestelmiin: Linux, Unix, Mac OS ja Solaris. Nämä versiot ovat Mono-projektin aikaansaannoksia, jota Novell sponsoroi. Mono-projekti tarjoaa kyseisille käyttöjärjestelmille tarvittavat ohjelmistot .NET-sovelluksien kehittämiseksi ja näiden ajamiseksi. (What is mono? 2006; Richter 2003, xxii.)



KUVIO 9. .NET Framework

CTS (Common Type System) määrittelee tiettyjä sääntöjä .NET-sovelluksissa käytettäville tyypeille. Se määrittelee mm. seuraavanlaisia asioita:

- tyypille toteutettavat mahdolliset jäsenet
  - vakiot, kentät, olion- ja tyypin alustajat, metodit, operaattorien ylikuormitukset, muunnosoperaattorit, ominaisuudet, tapahtumat ja sisäiset tyypit
- näkyvyys- ja suojaussäännöt tyypille ja sen jäsenille
  - kuinka muut sovellukset / tyypit voivat käyttää tätä tyyppiä ja sen jäseniä. Esimerkiksi, jos tyypin metodi määritellään avainsanalla internal, niin sitä voi käyttää vain saman kokoonpanon sisällä olevat muut tyypit.

(Richter 2003, 25-26.)

CLS (Common Language Specification) sisältää joukon sääntöjä ja asetuksia, joiden avulla .NET-ympäristössä toimivat ohjelmointikielet saadaan toimimaan yhteistyössä keskenään. Koska yhtenä .NET Framework:n tavoitteista on tukea mo-

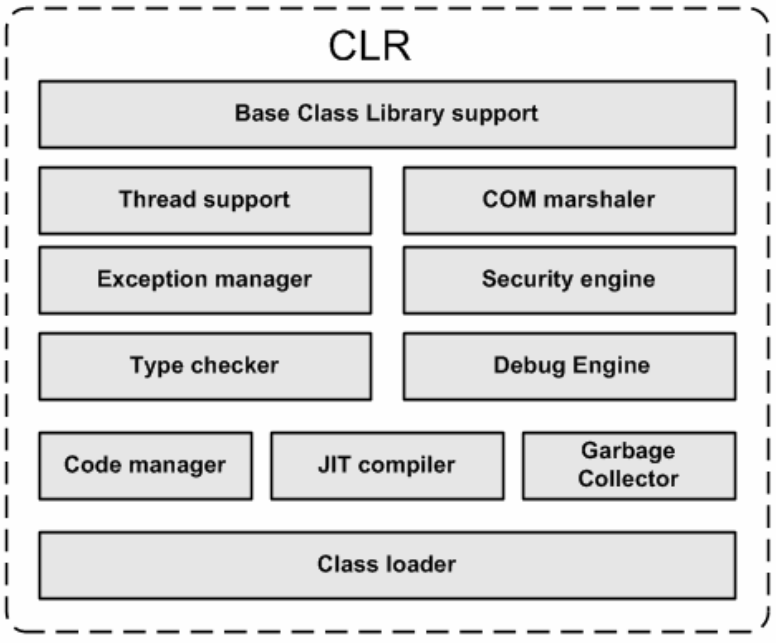
nia ohjelmointikieliä, on näiden sääntöjen olemassaolo tällaiselle ympäristölle välttämätöntä sen toimivuuden kannalta. (Understanding .NET 2005)

3.3.2 Common Language Runtime

CLR:ää voidaan pitää .NET Framework:n ytimenä, ja se on ajonaikainen ympäristö erilaisilla ohjelmointikielillä toteutetuille .NET-sovelluksille. Se sisältää palveluita ja ominaisuuksia, jotka ovat käytettävissä kaikissa siihen soveltuvissa ohjelmointikielissä. Näiden avulla .NET-sovelluksia suoritetaan ja hallitaan. CLR:n arkkitehtuuri on esitetty kuviossa 10. CLR:n tehtäviin kuuluu mm. sovellusten suorittaminen, poikkeusten hallinta, muistin- ja koodinhallinta sekä tietoturvaan liittyvät tarkistukset. Seuraavissa kappaleissa käydään läpi CLR:n keskeisiä toimintoja. (Richter 2003, 4.)

CLR ymmärtää vain MSIL-koodia, joka on Microsoftin toteuttama suoritinriippumaton välikieli. Kun .NET-sovellusta ajetaan CLR:n alaisuudessa, niin CLR:n sisäinen kääntäjä kääntää MSIL-koodin kyseisen suorittimen ymmärtämäksi konekieleksi ja tässä yhteydessä se myös tarkistaa koodin turvallisuuden. Käännös on niin sanottu JIT-käännös (Just-In-Time). Kun käännös on suoritettu, niin konekielinen koodi ladataan sovelluksen muistiavaruuteen ja sovelluksen ajo alkaa. (Richter 2003, 12.)

GC (Garbage Collector) on CLR:ään kuuluva automaattinen muistin siivoustoiminto, jonka avulla .NET-sovelluksen käyttämät resurssit voidaan automaattisesti vapauttaa. Tämä tarkoittaa sitä, että sovellukselle varatulta muistialueelta on mahdollista vapauttaa tämän automatiikan avulla kaikki ne resurssit (oliot), joihin sovelluksesta ei enää viitata. Tämä vapauttaa ohjelmoijan muistinhallintatehtävistä. (Richter 2003, 451.)



KUVIO 10. Common Language Runtime

3.3.3 Framework Class Libraries

FCL sisältää useita tuhansia Microsoftin toteuttamia tyyppejä, joita ohjelmoijat voivat käyttää omissa sovelluksissaan. Tyypit on ryhmitelty niin, että kaikki samaan aiheeseen liittyvät tyypit sijaitsevat samassa nimiavaruudessa (namespace). Esimerkiksi nimiavaruudessa System.Web sijaitsee kaikki web-kehitykseen tarvittavat tyypit. Datan käsittelyyn liittyvät tyypit sijaitsevat omassa Data- ja XML-osiossa ja BCL (Base Class Libraries) sisältää tyyppejä, joita eri tyyppiset .NET-sovellukset voivat käyttää. FCL:n sisältämistä kirjastoista on mahdollista periyttää myös omia tyyppejä, jolloin on mahdollista toteuttaa omia räätälöityjä ominaisuuksia näihin tyyppeihin. (Richter 2003, 22-23)

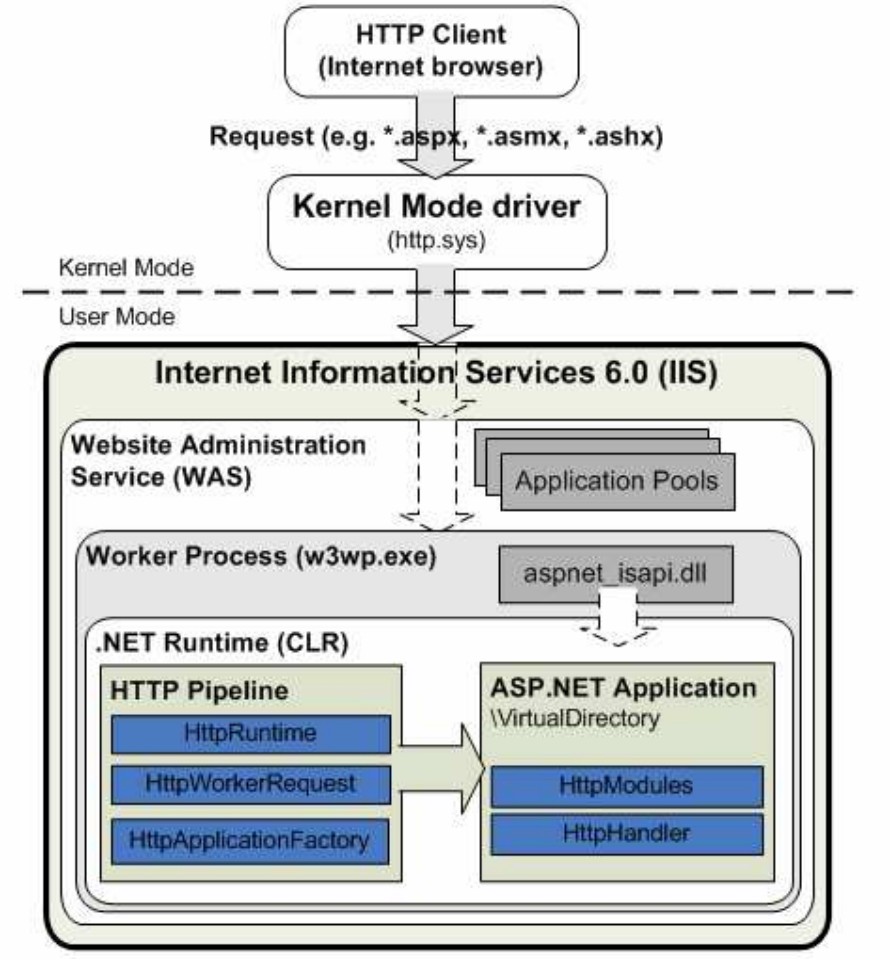
3.4 ASP.NET

3.4.1 Yleistä

ASP.NET on Microsoftin tarjoama teknologia dynaamisten web-sovellusten toteuttamiseksi, ja se pitää sisällään seuraavat yksittäiset teknologiat: web-lomakkeet (Web Forms) ja verkkopalvelut (Web Services). ASP.NET ei ole vain uusi versio Microsoftin vanhalle ASP-teknologialle, vaan on täysin uuden sukupolven web-sovelluskehitysympäristö ja siinä sovelluskehitys perustuu mm. olio-ohjelmointimalliin. Se on tiukasti integroitu .NET Framework-ympäristöön, ja on yksi tämän tärkeimmistä osista. Tästä syystä ASP.NET-sovelluksissa voidaan käyttää hyväksi kaikkia .NET Frameworkin tarjoamia palveluja ja ominaisuuksia. (MacDonald & Szpuszta, 2005, xxix.)

ASP.NET toimii monelta osin eri tavalla verrattuna vanhempiin web-sovellusten toteutusteknologioihin kuten mm. ASP:iin, joka on Microsoftin vanhempi teknologia web-sovelluksien toteuttamiseksi. Eroavaisuuksia näiden teknologioiden välillä on monia, joten tässä kappaleessa käydään näistä läpi vain oleellisimmat. Ensimmäinen suuri ero on käytettävä ohjelmointikieli. ASP.NET-sovelluksia voidaan ohjelmoida monella .NET CLS yhteensopivalla ohjelmointikielellä kuten C#, VB.NET ja J#, kun taas esim. ASP-sovelluksia ohjelmoidaan ainoastaan VBScript-kielellä. Toinen merkittävä ero on ohjelmakoodin suorittaminen. ASP.NET:ssä kaikki palvelimella suoritettava ohjelmakoodi käännetään ensimmäisellä sivupyynnöllä konekieliseksi koodiksi. Kaikki tämän jälkeen tulevat pyynnöt suorittava tätä jo kertaalleen käännettyä ohjelmakoodia. ASP-sovellukset toimivat tässä tapauksessa hieman eri tavalla, ja siinä palvelimella suoritettava VBScript-koodi joudutaan tulkkaamaan joka kerta uudestaan kun ASP-sivupyyntö tehdään selaimesta. Yhtenä suurena erona voidaan pitää myös HTML-koodin ja sovelluskoodin erottamista toisistaan. Tämä tarkoittaa sitä, että käyttöliittymän rakentamiseen käytettävä HTML-koodi ja sovelluksen businesslogiikka sijaitsevat omissa fyysisissä tiedostoissaan, ja näiden välillä on vain linkki toisiinsa. Tästä tekniikasta on mm. se hyöty, että nyt ohjelman lähdekoodi on selkeästi omana osanaan eikä ole sekoitettuna HTML-koodin sekaan kuten ASP-sivuissa. (MacDonald & Szpuszta, 2005, 7-10, 47.)

ASP.NET-sovellusten suoritusympäristönä toimivat yhteistyössä Microsoftin IIS-palvelin ja .NET Framework. Kuviossa 11 on kuvattuna prosessi, kuinka ASP.NET-sovellukseen tuleva pyyntö käsitellään IIS-palvelimella. Kun sovellusta kutsutaan ensimmäisen kerran, niin CLR kääntää sen lähdekoodin konekieliseksi koodiksi ASP.NET:n omaan väliaikaishakemistoon. Tämän jälkeen muiden pyyntöjen yhteydessä sovellusta suoritetaan tästä väliaikaishakemistosta.



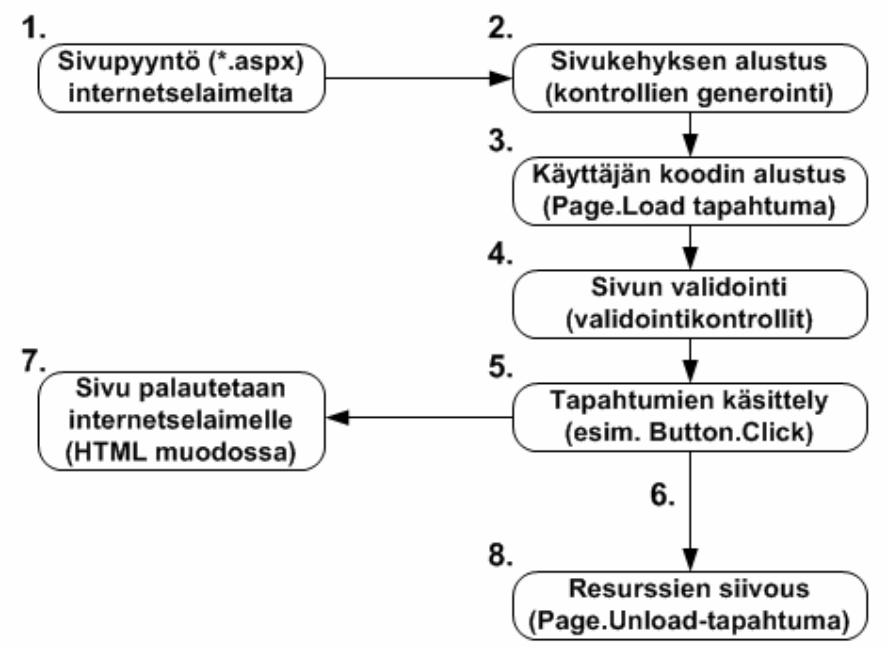
KUVIO 11. IIS 6.0-prosessimalli

Kun IIS 6.0-palvelimella suoritettavaan ASP.NET-sovellukseen lähetetään sivupyyntö internetselaimelta, niin palvelimella käyttöjärjestelmän kernel-tasolla toimiva http.sys-ajuri vastaanottaa pyynnön. Http.sys-ajuri välittää pyynnön eteen-

päin IIS 6.0:n Application Pool-osiolle, johon sovellus kuuluu. Application Pool-osion tehtävänä on luoda ns. W3C Worker Process (w3wp.exe-prosessi) sovelluksen suoritusta varten. Näitä prosesseja ohjaa WAS (Website Administration Service) –palvelu, jonka tehtävänä on valvoa w3wp.exe-prosessien tilaa, ja se mm. pystyy käynnistämään w3wp.exe-prosessin uudestaan virhetilanteessa. Jokaisessa w3wp.exe-prosessissa voidaan suorittaa yhtä tai useampaa ASP.NET-sovellusta ja/tai muilla teknologioilla toteutettuja web-sovelluksia, kuten ASP-sovelluksia. Sovelluksen w3wp.exe-prosessin muodostamisen jälkeen se kutsuu ASP.NET-sovelluksen tapauksessa aspnet_isapi.dll-komponenttia, jonka tehtävänä on ladata .NET CLR tämän prosessin yhteyteen, jotta hallittua .NET-ohjelmakoodia voidaan suorittaa. Sovelluksen prosessi suorittaa ohjelmakoodin ja muodostaa tästä HTML-esityksen, jonka palvelin lähettää vastauksena asiakkaan internetselaimeen. (The ASP.NET HTTP Runtime 2003)

3.4.2 Web-lomakkeet

ASP.NET-sivut eli web-lomakkeet ovat keskeinen osa ASP.NET-sovellusta, ja ne perustuvat internetpalvelimella suoritettaviin dynaamisiin web-sivuihin. Dynaamisissa web-sivuissa ei sinällään ole mitään uutta, mutta mullistavaa ASP.NET:n tapauksessa sen sijaan on tekniikka, jolla näitä dynaamisia web-sivuja toteutetaan. Web-lomakkeiden ohjelmointi ja käsittely perustuvat tapahtumapohjaiseen arkkitehtuuriin, joka on samanlaista kuin perinteisten Win32-työpöytäsovellustenkin ohjelmointi. Kun ASP.NET web-lomaketta suoritetaan palvelimella, niin se käy läpi tiettyjä sivuun liittyviä tapahtumia, jotka ovat oleellinen osa tapahtumapohjaista arkkitehtuuria. Suurin ero vanhempiin palvelimella suoritettaviin dynaamisiin web-sivuihin onkin juuri tekniikka, jolla sivulla tapahtuvia tapahtumia käsitellään. Esimerkiksi perinteiset ASP-sivut suoritetaan jokaisella pyynnöllä palvelimella kokonaisuudessa, kun taas ASP.NET:n tapauksessa suoritetaan vain tarvittava(t) tapahtumat, kuten nappulan painallukseen liittyvä tapahtuma. Kuviossa 12 on esitetty ASP.NET-sivun elinkaari ja siihen liittyvät tapahtumat ja niiden järjestys. (MacDonald & Szpuszta, 2005, 63.)



KUVIO 12. ASP.NET-sivun elinkaari (MacDonald & Szpuszta, 2005, 77)

Yhtenä oleellisena osana ASP.NET-tekniikkaa ovat sen tarjoamat internetpalvelimella suoritettavat kontrollit, eli ns. palvelinkontrollit. Ne ovat .NET Frameworkiin kuuluvia luokkia, jotka pystyvät esittämään visuaalisia elementtejä weblomakkeella. Palvelinkontrollit ovat yksilöllisiä ASP.NET-teknologialle, ja ne voidaan jakaa muutamaan eri kategoriaan:

- HTML-palvelinkontrollit ovat kontrolleja, joilla on suora vastine standardeissa HTML-kontrolleissa. Jotta HTML-kontrolli toimisi palvelinkontrollina, pitää kontrollin lisätä runat=”server”–attribuutti. Esimerkkiksi HTML Button-palvelinkontrolli määritellään seuraavasti:
  <input type=”button” runat=”server” value=”Button text” />
- Standardit Web-kontrollit ovat toiminnoiltaan vastaavia kuin HTML-palvelinkontrollit, mutta niihin on lisätty ominaisuuksia ja metodeja, jotka helpottavat niiden määrittelyä ja käyttöä. Esimerkkiksi tekstikenttä Web-kontrolli määritellään seuraavasti:
  <asp:TextBox id="TextBox1" runat="server"></asp:TextBox>

- Monipuoliset Web-kontrollit ovat kontrolleja, jotka muodostavat paljon HTML- ja JavaScript-koodia ja ovat toiminnoiltaan hyvin monipuolisia. Tähän kategoriaan kuuluu mm. kalenteri-, navigointi-, validointi- ja kirjautumiskontrolleja.
- Web Part-kontrollit ovat peräisin web-portaaleista ja ne ovat tullee mukaan ASP.NET-sovelluskehitykseen .NET 2.0-version myötä. Näiden kontrollien avulla voidaan sovellukselle luoda käyttöliittymä, jonka käyttäjä voi muokata oman näköisekseen tietyissä rajoissa.
- Käyttäjäkontrollit. Ohjelmoija voi myös itse rakentaa palvelinkontrolleja. Kontrolleja on mahdollista luoda joko tyhjästä tai periyttämällä olemassa olevasta kontrollista ja lisäämällä tähän omia ominaisuuksia. Palvelinkontrolleja on myös mahdollista yhdistellä yhdeksi yhdistelmäkontrolliksi.
- Mobiilikontrollit ovat erilaisiin mobiililaitteisiin suunnattuja erikoiskontrolleja. (MacDonald & Szpuszta, 2005, 103.)

Web-lomakkeen HTML-koodi tallennetaan .aspx-tiedostoon, josta on esitetty esimerkki kuviossa 13. Web-lomakkeeseen liittyvä liiketoimintalogiikka on mahdollista ohjelmoida tähän samaan tiedostoon erottamalla se muusta koodista <script>-lohkolla. Toinen vaihtoehto liiketoimintalogiikka toteuttamiseksi on käyttää jo aiemmin mainittua taustakooditekniikka, jossa ohjelmakoodi erotetaan omaan fyysiseen tiedostoonsa, josta on esitetty esimerkki kuviossa 14. Kuviossa 15 on esitetty ASP.NET-sivu, johon on toteutettu kaksi palvelinkontrollia. (MacDonald & Szpuszta, 2005, 47.)

```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>ASP.NET Sivu</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

    <asp:Button ID="Button1" runat="server" Text="Button" OnClick="Button1_Click" />
    <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

    </div>
    </form>
</body>
</html>
```

KUVIO 13. Default.aspx ASP.NET-sivun HTML-esitys

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "Button1_Click";
    }
}
```

KUVIO 14. Default.aspx-sivuun liittyvä Default.aspx.cs taustakooditiedosto

KUVIO 15. Default.aspx-sivun näkymä internetselaimessa

3.4.3 Web-palvelut

Web-palvelut ovat verkon kautta käytettäviä itsenäisiä hajautettuja komponentteja, jotka ovat aidosti alusta- ja ympäristöriippumattomia. Web-palvelut tarjoavat erilaisia palveluita, joita voivat olla esimerkiksi internethakupalvelu, säätiedotepalvelu, uutispalvelu jne. Näillä komponenteilla ei ole omaa käyttöliittymää, joten näitä palveluita käyttävät erityyppiset sovellukset eivätkä niinkään suoranaisesti ihmiset. Web-palveluiden avulla on myös mahdollista jakaa tietoa tietojärjestelmien välillä paljon helpommin kuin ennen johtuen käytettävästä XML-pohjaisesta viestimuodosta. Tämä teknologia ei ole yksinomaan Microsoftin keksimä ja kehittämä, vaan se on monien suurten IT-alan yritysten yhteistyössä kehittämä teknologia. Microsoft tarjoaa tosin hyviä sovelluskehitysvälineitä mm. juuri Web-palvelujen tuottamiseksi, jotka tekevät näiden toteuttamisen yllättävän helpoksi. (.NET Tekee sen yksinkertaisemmin)

Web-palveluiden konseptiin kuuluu monia eri teknologioita ja standardeja, joista tärkeimmät ovat:

- SOAP (Simple Object Access Protocol) on protokolla ja alustariippumaton standardi, jonka avulla XML-tyyppisiä viestejä siirretään tietokoneverkossa. Se määrittelee kommunikoinnissa käytettävän XML-viestin rakenteen. (SOAP 2001)
- WSDL (Web Services Description Language) on XML:aan perustuva kuvauskieli, jonka avulla kuvataan web-palvelun julkiset rajapinnat sekä kommunikointiprotokollat joita palvelu tukee. WSDL-dokumenttia voidaan pitää eräänlaisena sopimuksena, jonka kertoo palvelua käyttävälle sovellukselle mitä sen tarvitsee tietää palvelusta pystyäkseen käyttämään sitä oikein. (WSDL 2006)
- UDDI (Universal Description, Discovery and Integration) on hakemisto web-palveluille. Se voi olla joko julkinen kaikkien saatavilla oleva palvelu tai yksityisen organisaation käytössä oleva palvelu. (.NET Tekee sen yksinkertaisemmin.)

3.5 ADO.NET

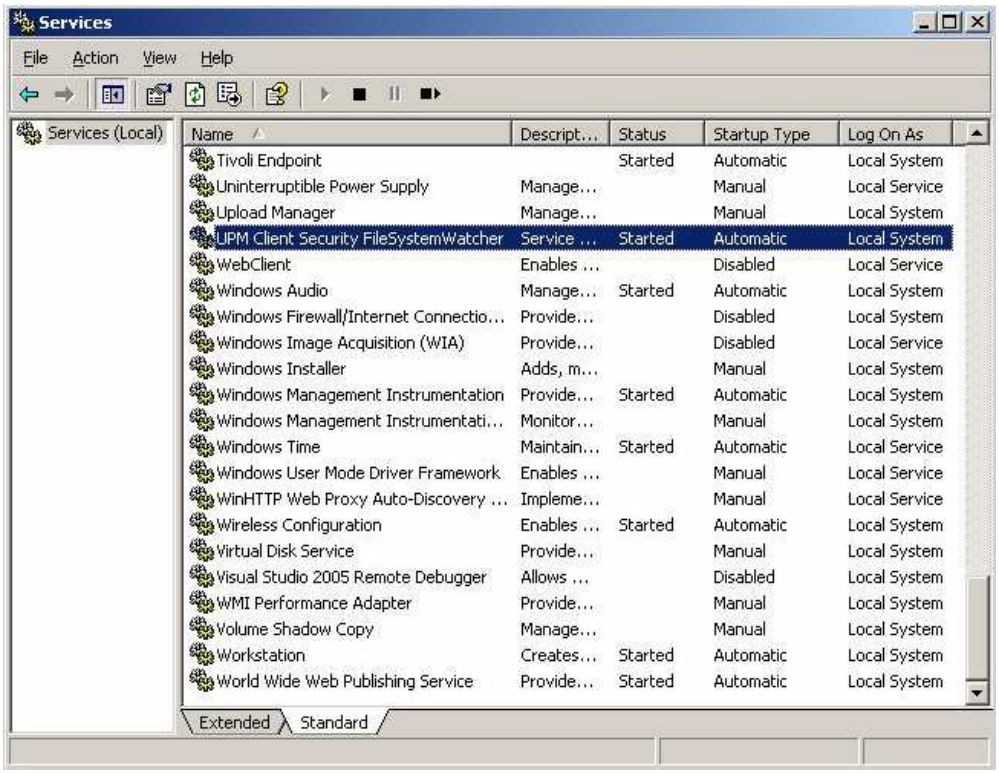
ADO.NET on teknologia tietokantakäsittelyjä varten ja se on osa Microsoft .NET-konseptia. Sen avulla voidaan tietoa hakea monesta erilaisesta lähteestä. Tuettuja tietokantoja ovat mm. SQL Server, Oracle, Access sekä tietokannat, jotka tukevat ODBC- ja OleDB-yhteyksiä. ADO.NET-teknologian tuoma oleellisin uudistus tietokantakäsittelyihin on sen kyky käsitellä tietokannan tietoa ns. offline-tilassa. Tämä tarkoittaa sitä, että tietokannasta voidaan haluttaessa hakea vaikka koko tietokannan sisältö yhteen sovellustasolla käsiteltävään DataSet-tyyppiseen olioon, johon se tallennetaan XML-muodossa. DataSet toimii näin siis muistinvaraisena tietokantana, joka sisältää vastaavaa tietoa kuin oikeat tietokannat. Näitä ovat mm. avaimet, rajoitteet sekä taulut ja niiden väliset suhteet. Kun DataSet-olio on luotu, niin sen sisältämää tietoa voidaan hakea ja muokata ilman, että tietokantayhteyttä joudutaan välillä avaamaan. Huonona puolena tässä tekniikassa on sovelluksen suorituskyvyn heikentyminen, mikä johtuu mahdollisesta suuresta sovellustasolla käsiteltävä datamäärästä. Tietokantahakua varten luodaan DataSetin tapauksessa DataAdapter-olio, jonka tehtävänä on hakea ja täyttää DataSet-olio tietokannasta haettavalla tiedolla. DataAdapterin kautta voidaan myös päivittää mahdolliset DataSet-olioon tehdyt muutokset tietokantaan. (MacDonald & Szpuszta, 2005, 229-232, 276.)

ADO.NET tarjoaa paljon muitakin ratkaisuja tiedon käsittelyyn. Esimerkiksi DataReader-olion avulla tietoa voidaan lukea tietokannasta ja se mahdollistaa ns. kevyen yhdensuuntaisen lukuyhteyden ilman tiedon päivitysominaisuuksia. ADO.NET 2.0-version myötä on tullut muutama uudistus, joista esimerkkinä mainittakoon SqlBulkCopy-tyyppi. Tämän kautta on mahdollista tallentaa tietokantaan suuria määriä tietoa kerrallaan. Esimerkiksi kopioitaessa tuhansia rivejä tietokannasta toiseen SqlBulkCopy:n avulla kohdetietokantaan suoritetaan vain yksi lisäystoiminto. (MacDonald & Szpuszta, 2005, 240-242.)

3.6 Windows-palvelut

Windows-palvelut ovat taustalla suoritettavia prosesseja, joilla ei ole omaa käyttöliittymää. Tällaisia palveluita ovat mm. sovellus- ja tietokantapalvelimet, Windows-tapahtumalogit jne. Käyttöjärjestelmä voi käynnistää näitä palveluita jo ennen kuin käyttäjä on kirjautunut koneelle, mikä onkin yksi niiden vahvuuksista. Windows-palveluita hallitaan käyttöjärjestelmän mukana tulevalla ylläpitosovelluksella, joka on esitetty kuviossa 16. (Windows Services overview 2006)

Microsoft Visual Studio .NET 2002 ja tätä uudemmat versiot tarjoavat valmiin projektipohjan Windows-palvelujen toteuttamiseksi. Näitä palveluita voidaan sekä asentaa että poistaa työasemaan .NET Frameworkin mukana tulevalla installutilkomentorivisovelluksella.



KUVIO 16. Windows-palveluiden ylläpitosovellus

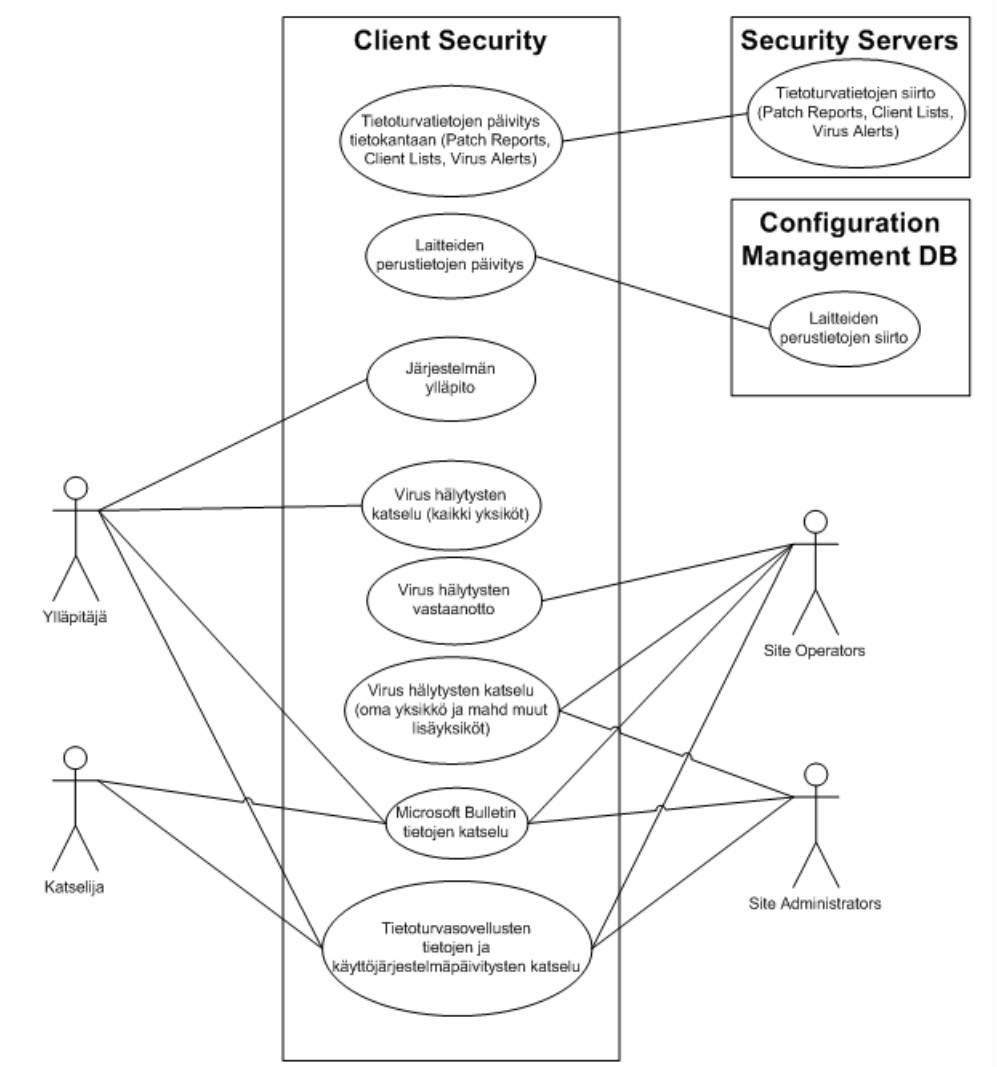
4 SOVELLUS

4.1 Määrittely

Ohjelmistojen määrittelyprosessi alkaa normaalisti vaatimusten kartoituksella. Tässä vaiheessa on tarkoituksena selvittää asiakkaan ohjelmistoon haluamien toimintojen perimmäiset syyt, arvioida näiden tärkeys (priorisointi) ja sovittaa yhteen mahdolliset ristiriitaiset vaatimukset. Kaikki vaatimustenkartoitusvaiheessa esiin tulleet toiminnot kirjataan ylös, minkä jälkeen tämän vaiheen lopputuloksena syntyy toiminnallinen määrittely. Toiminnallinen määrittely on tärkein ohjelmiston määrittelyyn liittyvä dokumentti, jonka luontiin on syytä käyttää aikaa ja toteuttaa se erittäin huolellisesti. Toimintojen lisääminen ja muokkaaminen ohjelmistoon jälkikäteen tehtynä on lähes aina suuritöisempää ja kalliimpaa, kuin että ne olisi mahdollisesti otettu huomioon jo määrittelyvaiheessa. (Haikala & Märijärvi 2002. 78, 96)

Toiminnallisen määrittelyn jälkeen seuraa normaalisti arkkitehtisuunnittelu, jossa järjestelmä ja sen toiminnot jaetaan sellaisiin moduuleihin, että ne voidaan antaa yksittäisten suunnittelijoiden tehtäväksi. Moduulit pyritään suunnittelemaan toisistaan riippumattomaksi, jotta yksittäiseen moduuliin tehtävät muutokset eivät vaikuta muihin moduuleihin. Arkkitehtisuunnittelun tuloksena syntyy tekninen määrittely, jossa on kuvattuna teknisellä tasolla kaikki ohjelmistoon liittyvät moduulit. Tämän jälkeen yksittäisten moduulien suunnittelu ja toteutus voi alkaa. (Haikala & Märijärvi 2002. 81)

Sovelluksen toiminnallista määrittelyä tehtäessä käytettiin hyväksi vanhaa Lotus Notes-sovellusta, josta saatiin pohjatiedot uuden sovelluksen toiminnoille, käyttäjärooleille ja käyttötapauksille, jotka ovat esitetty kuviossa 16. Toiminnallisen määrittelyn jälkeen laadittiin sovelluksen tekninen määrittely, jossa kuvattiin perustiedot kaikista ohjelmistoon liittyvistä moduuleista.



KUVIO 16. Client Security-sovelluksen käyttötapauskaavio

4.2 Sovellusympäristö

Sovellusta varten hankittiin oma palvelin, jonka käyttöjärjestelmäksi valittiin Windows 2003 Standard. Koska sovelluksesta tehtiin www-pohjainen, tarvitsi se myös toimiakseen internetpalvelimen. Tätä varten otettiin käyttöön käyttöjärjestelmän mukana tuleva IIS 6.0-internetpalvelin, joka oli sillä hetkellä paras valinta ASP.NET-sovellusten suoritusympäristöksi. Sovelluksen tietokantana toimi Microsoftin SQL Server 2005-tietokantapalvelin.

SQL Server 2005-tietokantaohjelmistoa asennettaessa voidaan valita mitä erityisominaisuuksia halutaan asentaa perusasennuksen lisäksi. Sovellusta varten piti asentaa Reporting- ja Integration Services-ominaisuudet. Reporting Services-palvelun avulla luodaan raportteja, joiden suoritusympäristönä toimii IIS 6.0-internetpalvelin. Integration Services-palvelulla voidaan mm. siirtää dataa eri tietokantojen välillä. Tietokannan asetuksista piti sovellusta varten ottaa käyttöön myös sähköpostien lähetystä varten ns. Database Mail-ominaisuus. (SQL Server 2005 Overview 2005)

Sovellus toteutettiin kokonaisuudessaan Microsoftin Visual Studio 2005-sovelluskehittimellä. Visual Studio Microsoftin tarjoama sovelluskehitin mm. ASP.NET-sovellusten toteuttamiseksi, josta löytyy valmiita projektipohjia erityyppisten .NET-sovellusten toteuttamiseksi.

4.3 Tietokanta

4.3.1 Yleistä

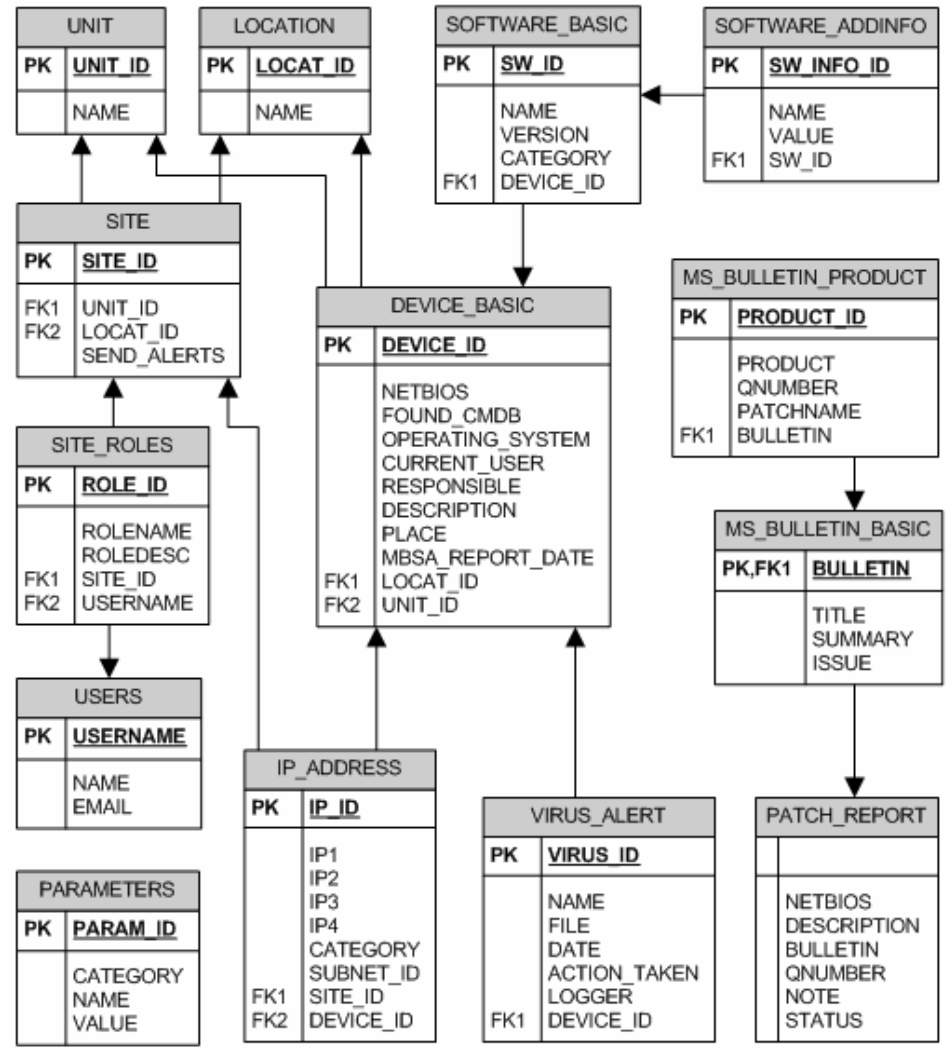
Sovelluksen tietokannan alustava suunnittelu tehtiin toiminnallisen määrittelyn yhteydessä. Tässä vaiheessa hahmottui tietokannan perusrakenne ja suurin osa käytettävistä tauluista ja tallennetuista proseduureista. Proseduurit ovat tietokantaan tallennettavia SQL-kielellä toteutettuja pieniä aliohjelmia, joiden avulla tietokannan dataa voidaan käsitellä tehokkaasti. Sovelluksen tietokantakäsittelyt tehdään näiden proseduurien kautta ja tietokantayhteydet on toteutettu ADO.NET-teknologian tarjoamin keinoin. Tietokannan rakenne muokkautui vielä sovellusta rakennettaessa ja lopulliseen muotoonsa se saatiin vasta vähän ennen sovelluksen valmistumista.

Sovelluksen tietokantakuvaus on esitetty kuviossa 17. Tietokannan device_basic-taulu pitää sisällään kaikkien laitteiden perustiedot. Tähän tauluun tuodaan Configuration Management-sovelluksesta laitteiden nimi, yksikkö ja paikkakuntatiedot. Laitteisiin liittyvät IP (Internet Protocol)-osoitteet ovat tallennettuina omassa taulussaan, koska yhdellä laitteella voi olla useampia IP-osoitteita. Yksikkö- ja paik-

kakuntatiedot tallennetaan omiin tauluihinsa, ja myös nämä tiedot tuodaan tietokantaan Configuration Management-sovelluksesta. Virustorjunta-, palomuuri- ja käyttöjärjestelmäpäivityssovelluksiin liittyvän perustiedot tallennetaan software_basic-tauluun. Jokaiseen sovelluksella raportoitavaan ohjelmistoon liittyy perustietojen lisäksi myös sovelluskohtaisia tietoja, jotka tallennetaan software_addinfo-tauluun. Tiedot asennetuista sekä asentamattomista käyttöjärjestelmäpäivityksistä tallennetaan patch_report-tauluun. Patch_report-tauluun liittyy myös ms_bulletin_basic- ja ms_bulletin_product-taulut, joihin on tallennettuna yksityiskohtaista tietoa jokaisesta Microsoftilta tulevasta päivityksestä. Virus_alert-taulu toimii työasemissa havaittavien virusten tallennuspaikkana. Tauluun on tehty yksi sql-trigger, joka aktivoituu tauluun tehtävästä insert-toimenpiteestä. Triggerin tarkoituksena on lähettää tietyille henkilöille tieto sähköpostilla tietokantaan tulleesta virushavainnosta. Sähköpostin lähetyksessä käytetään SQL Server 2005:n uutta Database Mail-ominaisuutta. Tietokantaan tallennetaan myös eri sijainteihin, joita kutsutaan myös saiteiksi, liittyvää tietoa. Saitit ovat yksikkökohtaisia ja ne muodostuvat yksikkö ja paikkakunta yhdisteistä. Yhteen yksikköön voi kuulua monia saitteja, ja ne sisältävät tietoa mm. niihen kuuluvista rooleista ja roolien jäsenistä. Saitin tiedoista ilmenee myös mahdollisten muiden saittiin kuuluvien toimialueiden kuin Griffin-toimialueen tietoja. Näitä tietoja ovat toimialueen IP-verkot ja toimialueella toimivien laitteiden nimeämisstandardit. Parameters-tauluun voidaan tallentaa sovelluksen yleisiä parametreja, kuten sovelluksen työkaluihin liittyviä asetuksia.

4.3.2 Windows-palvelu tietojen tallennukseen

Sovellusta varten toteutettiin UPM Client Security FileSystemWatcher Windows-palvelu. Tämän palvelun avulla kaikki virustorjunta-, palomuuri- ja käyttöjärjestelmäpäivityssovelluksiin sekä virushälytyksiin liittyvät tapahtumat tallennetaan tietokantaan. Windows-palvelua ajetaan sovelluksen omalla palvelimella ja se valvoo tiettyjä verkkohakemistoja, jotka sijaitsevat yhdellä yrityksen Lotus Domino palvelimista. Kun joihinkin näistä hakemistoista siirretään tietoturvapalvelimilta raportteja, niin Windows-palvelu aktivoituu ja parsii tiedoston sisällön sovelluksen tietokantaan.



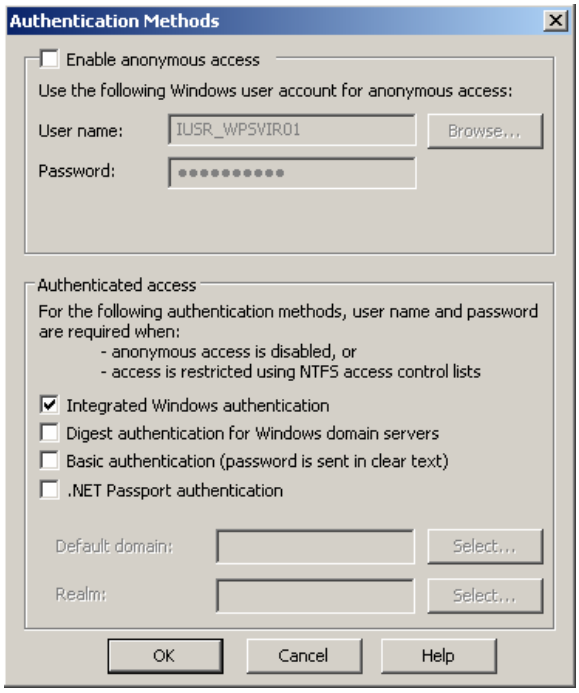
KUVIO 17. Sovelluksen tietokantakuvaus

4.4 Käyttäjien tunnistus ja käyttöoikeudet

ASP.NET-sovelluksissa käyttäjäntunnistus voidaan toteuttaa monella eri tavalla, joita ovat mm. anonyymi-, Forms-, Windows- ja Passport-tunnistus. Forms-tunnistus on käytössä yleisesti sovelluksissa, joita käytetään Internetin kautta. Näissä sovelluksissa käyttäjätunnukset ja salasanat tallennetaan yleensä sovelluksen omaan tietokantaan. Windows-tunnistus perustuu Windows-verkon käyttäjätileihin, joita Windows 2003-toimialueella hallitaan aktiivihakemiston avulla. Windows-tunnistusta käytetään usein yritysten sisäverkossa esimerkiksi intranet-soveluksissa. Passport-tunnistus on Microsoftin tarjoama käyttäjäntunnistuspalvelu. Sen tarkoituksena on yhdistää web-sovellusten käyttäjäntunnistus, joissa Pas-

sport-palvelu on käytössä, yhden käyttäjähallinnan taakse. (MacDonald & Szpuszta, 2005, 671.)

Koska kaikki sovelluksen käyttäjät ja siihen liittyvät laitteet kuuluvat Windows-toimialueeseen ja sovellus toimii yrityksen intranetissä, niin paras vaihtoehto käyttäjien tunnistamiseksi on käyttää Windows-tunnistusta. Windows-tunnistusta varten internetpalvelimen sovelluskohtaisista asetuksista otettiin käyttöön vain integroitu Windows-tunnistus, joka on esitetty kuviossa 18.



KUVIO 18. IIS 6.0:n käyttäjätunnistusasetukset

Jotta Windows-tunnistus ja tähän liittyvät käyttöoikeudet toimivat oikein myös sovellustasolla, pitää sovelluksessakin ottaa Windows-tunnistus käyttöön. Tämä tehdään asettamalla sovelluksen web.config-tiedostoon authentication-asetuksen mode-attribuutin arvoksi Windows, joka on esitetty kuviossa 19.

```
<authentication mode="Windows" />
```

KUVIO 19. Sovelluksen web.config-tiedoston käyttäjäntunnistus asetus

Sovelluksessa käytetään ns. roolipohjaista käyttöoikeuksien hallintaa, joka perustuu käyttäjäryhmiin. Sovellusta varten luotiin kaksi Windows-käyttäjäryhmää katselijoita ja sovelluksen ylläpitäjiä varten. Näiden ryhmien avulla sovelluksen käyttäjät tunnistetaan, ja niillä hallitaan osittain myös sovellukseen liittyviä käyttöoikeuksia.

.NET-sovelluksissa roolipohjainen käyttöoikeuksien tarkistus on tehty helpoksi siihen tarkoitettujen .NET Frameworkiin kuuluvien luokkien avulla. Esimerkiksi staattisen User-luokan IsInRole-metodilla voidaan ohjelmakoodissa tarkistaa kuuluko käyttäjä haluttuun Windows-ryhmään seuraavanlaisella koodilla: User.IsInRole(@"toimialue\windows-ryhmän nimi"). Käytettäessä roolipohjaisessa tunnistuksessa Windows-ryhmiä, joudutaan ASP.NET-sovelluksen web-config-tiedostoon asettamaan rivi <identity impersonate="true" />. Tämä tarkoittaa sitä, että nyt ASP.NET-sovelluksen käyttäjäkohtaista säiettä suoritetaan käyttäjän oikeuksin ja käyttäjän Windows-ryhmäoikeudet voidaan tarkistaa.

Käyttöoikeuksia varten sovellukseen on toteutettu myös sovelluksen sisäisiä ryhmiä, jotka ovat yksikkö- ja paikkakuntakohtaisia ns. saitti-rooleja. Näiden ryhmien avulla hallitaan sovellukseen tallennettavien virushälytysten lukuoikeuksia sekä sitä, ketkä saavat tiedon sähköpostitse havaituista viruksista.

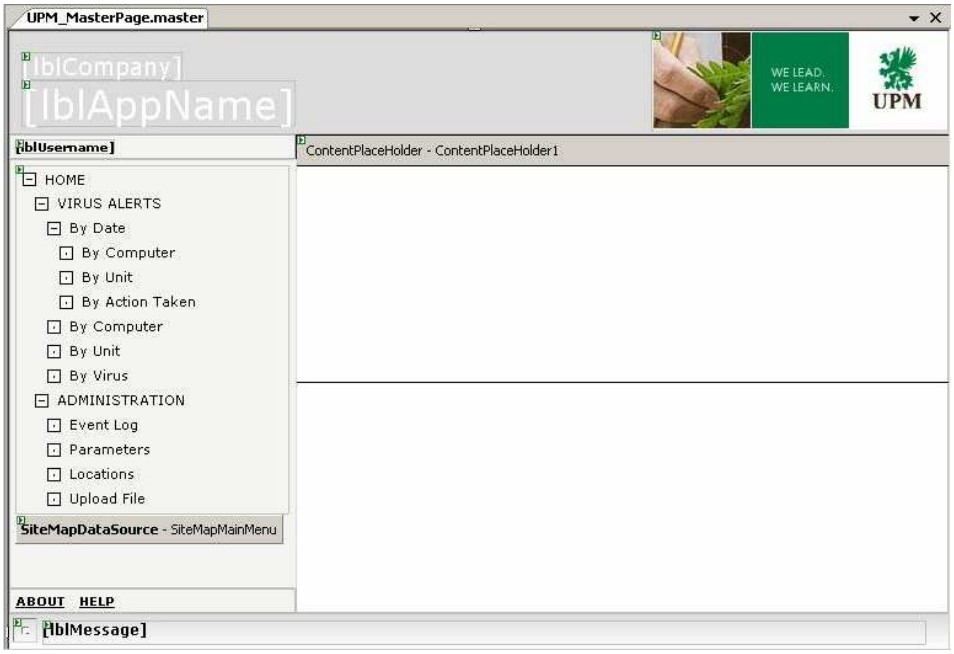
4.5 Sovellukseen liittyvät työkalut

Kaikki vanhaan sovellukseen toteutetut työkalut otettiin käyttöön myös uudessa sovelluksessa. Näihin työkaluihin kuului mm. työasemien virustorjuntaohjelmistoon liittyvät toimenpiteet ja käyttöjärjestelmäpäivityksien tarkistukset ja niiden asentaminen. Lähes kaikki työkalut olivat toteutettu kutsumalla sovellukseen liittyviä ulkoisia komponentteja. Näitä komponentteja olivat mm. jo aiemmin maini-

tut IBM:n postemsg.exe-sovellus ja Windows-komentotiedostot. Näille komponenteille välitetään kutsun yhteydessä parametreina, mitä toimintoja kohdekoneessa halutaan suorittaa. Uudessa sovelluksessa toimintoihin liittyvät parametrit ovat tallennettuina sovelluksen tietokantaan parameters-tauluun, josta ne haetaan toimintoja suoritettaessa.

4.6 Käyttöliittymä

Käyttöliittymä on toteutettu ASP.NET-version 2.0 tarjoamilla palveluilla. Sovellusta varten luotiin oma ns. MasterPage-pohja, joka on tullut uutena ominaisuutena ASP.NET 2.0:n mukana. MasterPage-ominaisuuden avulla sovellukselle voidaan helposti luoda haluttu mallipohja, johon toteutetaan sovelluksen logot, menut jne. Sovellukseen liittyvä MasterPage-pohja on esitetty kuviossa 20. MasterPage-pohjaan liittyy yhtenä tärkeimpänä osana ns. ContentPlaceHolder-komponentti, johon on tarkoitus lisätä sovelluksessa esitettävää ja käsiteltävää sisältöä. Tätä sovellusta varten luotua pohjaa käytetään hyväksi kaikissa sovellukseen liittyvissä näkymissä.



KUVIO 20. Sovelluksen MasterPage-pohja

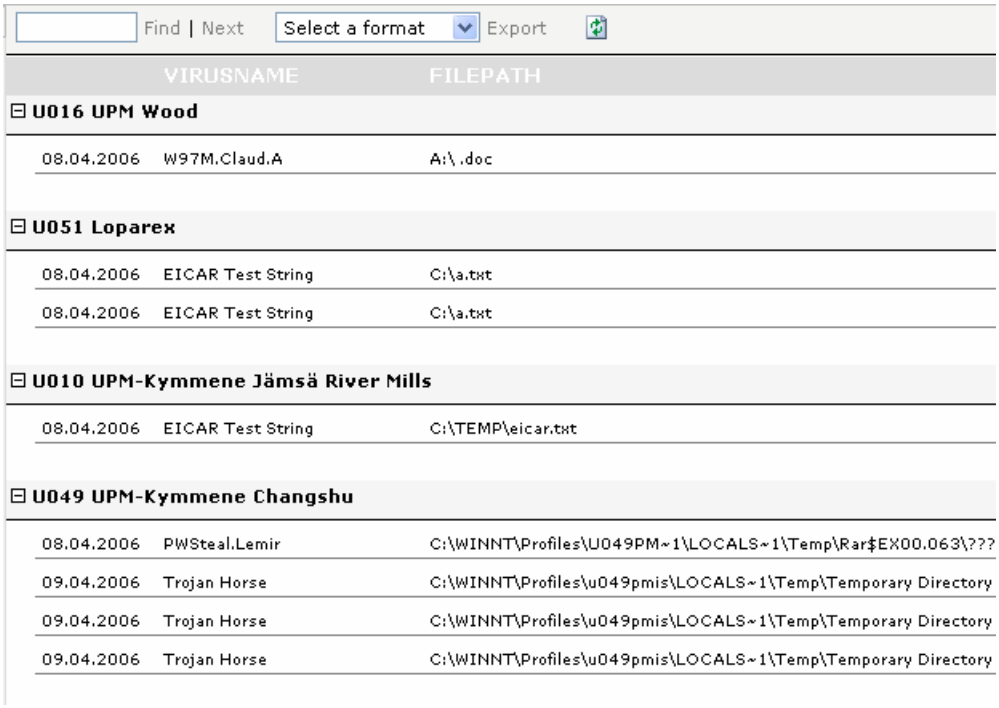
Sovelluksen päävalikko on toteutettu ASP.NET 2.0:n TreeView-kontrollilla. Valikon kaikki tasot ja niiden linkit ovat tallennettuina web.sitemap-tiedostoon, joka on XML-pohjainen tiedosto ja on tarkoitettu erilaisten valikkojen sisällön tallentamiseen. Web.sitemap-tiedostosta on esitetty malli kuviossa 21. TreeView-kontrollin data voidaan tallentaa haluttaessa myös tietokantaan.

Web.sitemap

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >

    <siteMapNode url="/UPM.clientSecurity/Default.aspx" title="HOME">

    <siteMapNode title="VIRUS ALERTS">
      <siteMapNode url="MainView.aspx?report=VirusAlert\ByDate_Computer" title="By Date">
        <siteMapNode url="MainView.aspx?report=VirusAlert\ByComputer" title="By Computer" />
        .
        .
        .
      </siteMapNode>
    </siteMapNode>

    <siteMapNode title="ADMINISTRATION">
      <siteMapNode url="Eventlog.aspx" title="Event Log"/>
      <siteMapNode url="Parameters.aspx" title="Parameters" />
      <siteMapNode url="Locations.aspx" title="Locations" />
      <siteMapNode url="UploadFile.aspx" title="Upload File" />
    </siteMapNode>
  </siteMapNode>

</siteMap>
```

KUVIO 21. Web.Sitemap-tiedosto

Vanhassa Lotus Notes-sovelluksessa olevat näkymät, joilla esitetään tietoa rivitasolla, toteutettiin uudessa sovelluksessa raporttien avulla. Raporttien luonnissa käytettiin Microsoftin Reporting Services-ympäristöä, ja yksittäiset raportit toteutettiin Visual Studio 2005-sovelluskehittimeen kuuluvalla raporttienluontityökalulla. Vaikka näiden raporttien teko onkin suhteellisen yksinkertaista, oli tämä sovelluksen suuritöisin osuus, johtuen raporttien suuresta määrästä. Sovellukseen tehtiin n. 70 raporttinäkymään, joista yksi on esitetty kuviossa 22.



KUVIO 22. Malli sovelluksen raporttinäkymästä

Kaikki sovelluksen sivut, joiden kautta tietoa tallennetaan sovelluksen tietokantaan, on tehty käyttäen ASP.NET:n web-lomakkeita. Näitä sivuja ovat mm. sovelluksen ylläpitoon liittyvät sivut. Myös kaikki yksittäiseen laitteeseen tai virushälytykseen liittyvät tarkemmat tiedot esitetään web-lomakkeiden kautta.

5 YHTEENVETO

Tässä opinnäytetyössä tutkittiin yleisellä tasolla Microsoft .NET -sovelluskehitysarkkitehtuuria ja tarkemmin siihen kuuluvaa ASP.NET-teknologiaa. Työn tavoitteena oli selvittää, voidaanko UPM-Kymmenen työasemaryhmän käytössä olevat työasemaympäristön Lotus Notes -pohjaiset hallintasovellukset toteuttaa Microsoft .NET -sovelluskehitysarkkitehtuurin tarjoamin keinoin.

Työn case-osiossa suunnitellulle ja toteutetulle sovellukselle määritetyt toiminnot oli mahdollista toteuttaa .NET -ympäristössä, joten sovellukselle asetetut toiminnalliset tavoitteet saavutettiin. Sovellusta ei saatu testattua työn aikana tuotantoympäristössä, joten käyttökokemuksia sovelluksen käytettävyydestä ei ollut saatavilla. Sovellus on tarkoitus ottaa lähitulevaisuudessa laajempaan testikäyttöön, jonka jälkeen yrityksessä tehdään päätös siirretäänkö työasemaympäristön hallintasovellukset Microsoft .NET -ympäristöön.

Case-sovelluksen suunnittelu oli melko suoraviivaista vanhan Lotus Notes-sovelluksen ansiosta. Kaikki Lotus Notes-sovellukseen toteutetut toiminnot, näkymät ja käyttäjäroolit toteutettiin myös case-sovellukseen, joten määrittelyä varten vaaditut toiminnot olivat jo ennen työn alkua selvillä. Sovelluksen vaatima tietokanta hahmottui lähes lopulliseen muotoonsa jo määrittelyvaiheessa ja siihen tarvitsikin tehdä vain pieniä muutoksia sovellusta rakennettaessa.

Sovelluksen toteutusvaiheessa törmättiin muutamaan ongelmaan käyttöliittymän osalta. Notes-tyyppisten näkymien toteuttaminen, joissa on käytetty monitasoista kategorisointia, ei ASP.NET:n tarjoamin keinoin ollut helppoa, koska ASP.NET ei tarjoa valmista kontrollia tähän tarkoitukseen. Ongelma ratkaistiin Microsoft Reporting Services-raporttien avulla, joissa tietojen kategorisointiominaisuus on toteutettuna.

Sovelluskehitys Lotus Domino- ja Microsoft .NET -ympäristöissä eroaa monilta osin merkittävästi. Lotus Domino-ympäristön etuja ovat mm. helposti opittava arkkitehtuuri ja sovelluskehitys. Ympäristöön on helppo luoda yksinkertaisia dokumenttienhallintasovelluksia, joita voidaan toteuttaa ilman ohjelmointitaitoja.

Tästä syystä Domino-ympäristöön luotavat Notes-sovellukset sopivatkin hyvin erilaisten työryhmien dokumenttienhallintaan. Dominon suurimpina heikkouksina ovat sen vanhentunut dokumenttipohjainen tietokanta-arkkitehtuuri sekä olio-ohjelmointituen puuttuminen Notes-sovelluskehityksestä.

Microsoft .NET-arkkitehtuuri tarjoaa nykyaikaisen ympäristön monien erityyppisten sovellusten toteuttamiseksi, monilla eri ohjelmointikielillä. Tähän ympäristöön on mahdollista rakentaa suuria ja toiminnoiltaan monipuolisia sovelluksia. Näiden sovellusten toteuttaminen edellyttää olio-ohjelmointiosaamista sekä relaatiotietokantaosaamista sovelluksissa, joissa käytetään tietokantaa.

UPM-Kymmenen kannalta Lotus Dominon etuja ovat yrityksen pitkä kokemus tästä ympäristöstä sekä laaja osaamisalue Notes-sovelluskehityksessä. Lotus Notes-sovelluksissa tulee kuitenkin nopeasti rajat vastaan, niin sovelluskehityksessä kuin tehokkuudessakin toteutettaessa esimerkiksi juuri työasemaympäristön hallintaan liittyviä sovelluksia. Microsoft .NET -arkkitehtuuri tarjoaa sovelluskehitykseen paljon enemmän mahdollisuuksia ja yrityksen toimiessa Windows-ympäristössä antaa se selkeästi paremmat lähtökohdat hallintasovellusten toteuttamiseksi. Huonona puolena .NET-arkkitehtuurin käyttöönotossa yrityksen kannalta on tarve uuden ympäristön ja siinä käytettävän ohjelmointikielen sekä sovelluskehittimen opiskelu. Todennäköisesti myös relaatiotietokantaosaamista sovelluskehityksen näkökulmasta tarvitaan lisää.

Sovellukseen tehtyjen testien perusteella tiedon siirto sovellukseen toimii moitteetta, joka on paljolti tähän tarkoitukseen toteutetun toimivan Windows-palvelun ansiota. Heikkona osana sovelluksessa on Reporting Services-tekniikalla toteutetut näkymät. Testeissä nämä osoittautuivat ajoitta hitaiksi käyttää, johtuen tietojen kategorisoinneista.

LÄHTEET

ASP.NET HTTP Runtime [verkkodokumentti]. heinäkuu 2003 [viitattu: 25.3.2006]. Saatavissa: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/SecNetAP04.asp, 10.7.2003

Haikala, I. & Märijärvi, J. 2002. Ohjelmistotuotanto. 8. uudistettu painos. Talentum Media Oy. RT-Print, Pieksamäki.

Internetin tulevaisuudennäkymiä [verkkodokumentti]. tammikuu 2003 [viitattu: 26.3.2006]. Saatavissa:http://www.cs.helsinki.fi/new/InternetinTulevaisuusArtikkeli2002-12-31.pdf

Lotus Notes [verkkodokumentti]. heinäkuu 2006 [viitattu: 18.3.2006]. Saatavissa: http://en.wikipedia.org/wiki/Lotus_Notes

MacDonald, M. & Szpuszta, M.2005. Pro ASP.NET 2.0 in C# 2005. Apress. Berkeley CA.

MSSQL Server Reporting Services [verkkodokumentti]. helmikuu 2004 [viitattu: 10.4.2006]. Saatavissa: http://www.databasejournal.com/features/mssql/article.php/3317461

.NET Framework FAQ [verkkodokumentti]. heinäkuu 2001 [viitattu: 25.3.2006]. Saatavissa: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/faq111700.asp

.NET Tekee sen yksinkertaisemmin [verkkodokumentti]. [viitattu: 26.3.2006]. Saatavissa: http://www.microsoft.com/finland/net/simple.asp

Richter, J. 2003. Microsoft .NET-ohjelmointi. Edita Publishing Oy IT Press. Helsinki.

SOAP [verkkodokumentti]. joulukuu 2001 [viitattu: 26.3.2006]. Saatavissa: http://en.wikipedia.org/wiki/Simple_Object_Access_Protocol

SQL Server 2005 Overview [verkkodokumentti]. marraskuu 2005 [viitattu: 18.4.2006]. Saatavissa: http://www.microsoft.com/sql/prodifo /overview/default.mspx

Understanding .NET [verkkodokumentti]. marraskuu 2005 [viitattu: 25.3.2006]. Saatavissa: http://www.c-sharpcorner.com/Code/2005/Feb/ Understanding.NET.asp

UPM-Kymmene [verkkodokumentti]. [viitattu: 2.4.2006]. Saatavissa http://w3.upm-kymmene.com/upm/internet/cms/upmcmsfi.nsf/$all/ C72CD61380E5B556C2256E370047BFD8?OpenDocument&qm= menu,1,0,0

Virtala, A. 2003. Lotus Notes työryhmäohjelmiston peruskirja. Docendo Finland Oy. Porvoo.

What is mono? [online]. tammikuu 2006 [viitattu: 18.4.2006]. Saatavissa: http://www.mono-project.com/Main_Page

Windows Services overview [verkkodokumentti]. maaliskuu 2006 [viitattu: 10.4.2006]. Saatavissa: http://www.bozemanblog.com/ PermaLink,guid,10695561-a9bf-4567-b819-b5dec93e30d6.aspx

WSDL [verkkodokumentti]. helmikuu 2006 [viitattu:26.3.2006]. Saatavissa: http://en.wikipedia.org/wiki/WSDL