Heikki Hyvönen

SUUNNITELMA LIIKENNETIEDOTEJÄRJESTELMÄN TIEDONSIIRRON JA JÄRJESTELMÄÄ HYÖDYNTÄVÄN SOVELLUKSEN TOTEUTTAMISEKSI

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikan ja liikenteen ala
Tietotekniikan koulutusohjelma
Kevät 2009

Kajaanin ammattikorkeakoulu

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Tekniikan ja liikenteen ala	Koulutusohjelma Tietotekniikka
Tekijä(t) Heikki Hyvönen	
Työn nimi Suunnitelma liikennetiedotejärjestelmän tiedonsiirron ja järjestelmää hyödyntävän sovelluksen toteuttamiseksi	
Vaihtoehtoiset ammattiopinnot Sulautetut järjestelmät	Ohjaaja(t) Jukka Heino
	Toimeksiantaja Ebsolut Oy, Kajaani
Aika Kevät 2009	Sivumäärä ja liitteet 30

Tämän insinöörityön tilaajana toimi kajaanilainen ohjelmistoalan yritys Ebsolut Oy. Työn tarkoituksena oli tehdä suunnitelma liikennetiedotejärjestelmän tiedonsiirron ja järjestelmää hyödyntävän sovelluksen toteuttamiseksi. Ebsolut Oy:llä on oma kuljetusyrityksille suunnattu tuote, joka on kokonaissovellus kuljetustenhallintaan. Sovellus sisältää kuljetusyritysten ajoneuvoihin ohjelmiston, jossa liikennetiedotejärjestelmä tulee ottaa käyttöön.

Suomessa Destia Oy tarjoaa ajantasaista tietoa liikenteestä. Destialla on ympärivuorokautisesti toimiva liikenteen palvelukeskus, joka kerää tietoja liikenteestä eri tahoilta. Liikenteen palvelukeskus käsittelee tiedot ja välittää ne eteenpäin palvelua tarjoaville yrityksille tai suoraan loppukäyttäjille. Destialla on oma radiotaajuuksilla toimiva liikennetiedotteita välittävä RDS-TMC-palvelu, jossa tiedotteet välitetään RDS-tiedonsiirtokanavaa pitkin. Radioverkko on yksisuuntainen järjestelmä, jolloin koko Suomea kattavat tiedot välittyvät jokaiselle käyttäjälle. Ebsolut Oy:n liikennetiedotejärjestelmän on suunniteltu toimivan siten, että liikennetiedotteet välitetään vain sille alueelle, jolla liikutaan. Tätä varten Destia tarjoaa liikennetiedotteet myös XML-muotoisena lähetteenä.

Ebsolut Oy.lle suunniteltu liikennetiedotejärjestelmä hakee liikennetiedotteet Destian palvelimelta XML-muodossa. Tiedot tallennetaan Ebsolut Oy:n palvelimelle, josta ne ovat haettavissa liikennetiedotteita tarvitsevien sovellusten käyttöön. Liikennetiedotteiden hakemista varten suunniteltiin oma itsenäinen komponentti, joka hoitaa liikennetiedotteiden hakemisen palvelimelta, tietojen käsittelyn sekä välittämisen sovellukselle. Tiedonsiirtokanavaksi liikkuvien päätelaitteiden ja Ebsolut Oy:n palvelimen välille valittiin GPRS-yhteys.

Tässä työssä on kuvattu järjestelmien integroimiseen liittyviä yksityiskohtia sekä yksi arkkitehtuurimalli, palvelukeskeinen arkkitehtuuri, jolla järjestelmien välinen teknologiasta ja alustasta riippumaton integroiminen voidaan suorittaa. Lisäksi työssä on kuvattu Web Services -tiedonsiirtotekniikka, joka on yksi tekniikka, jolla tätä arkkitehtuurimallia noudattavia sovelluksia voidaan toteuttaa. Web Services valittiin tiedonsiirtotavaksi liikennetiedotteiden välittämiseen palvelimelta sovellukselle. Työssä on esitetty myös suunnitelmat liikennetiedotekomponentin toteuttamiseksi sekä suunnitelma liikennetiedotejärjestelmän hyödyntämiseksi ajoneuvosovelluksessa. Ajoneuvosovellus sisältää kartan, jolla liikennetiedotteet tullaan näyttämään.

Kieli	suomi
Asiasanat	TMC, liikennetiedotteet, Web Services
Säilytyspaikka	☒ Kajaanin ammattikorkeakoulun Kaktus-tietokanta ☒ Kajaanin ammattikorkeakoulun kirjasto

Kajaanin ammattikorkeakoulu
University of Applied Sciences

THESIS
ABSTRACT

School School of Engineering	Degree Programme Information Technology
Author(s) Heikki Hyvönen	
Title Designing the Data Transmission of a Traffic Information System and an Application Using the System	
Optional Professional Studies Embedded Systems	Instructor(s) Jukka Heino
	Commissioned by Ebsolut Oy, Kajaani
Date Spring 2009	Total Number of Pages and Appendices 30

This Bachelor's thesis was commissioned by Ebsolut Oy. The purpose of this study was to create a design for the use of a traffic information service implemented by the company. This study describes a traffic information application handling the data transmissions between the client application and the traffic information server of Ebsolut Oy. This study also describes one way to implement a client application which uses traffic information and the application mentioned above.

First, the architecture of the traffic information application was designed. The architecture was designed by using the service oriented architecture (SOA) model. Secondly, various methods for data transmission were considered. The selection for the data transmission was made by choosing the most suitable method implementing the idea of service oriented architecture. The chosen method was Web Services. Thirdly, the traffic information and its interfaces were designed using the Unified Modeling Language (UML). Finally, the graphical user interface (GUI) and the business logic of the application using the traffic information application were designed using the UML.

As a result of this thesis, the design for the traffic information application and one client application was created. The data transmission using the Web Services was tested and it was found out that the Web Services met all the requirements set on the architectural design.

This study includes the design for the traffic information application and one client application. These applications will be implemented later.

Language of Thesis	finnish
Keywords	Traffic Information, Web Services, Service Oriented Architecture
Deposited at	☒ Kaktus Database at Kajaani University of Applied Sciences ☒ Library of Kajaani University of Applied Sciences

ALKUSANAT

Tämä työ on toteutettu Ebsolut Oy:lle, jota haluan kiittää mielenkiintoisesta aiheesta. Kiitos myös Jukka Heinolle työn ohjauksesta sekä Eero Soiniselle ja Kaisu Korhoselle kieliasun tarkastuksesta. Erityiskiitos insinööriopiskelija Jukka Korhoselle loistavasta yhteistyöstä sekä avovaimo Johannalle tuesta ja kärsivällisyydestä.

Kajaanissa 9.4.2009

Heikki Hyvönen

LYHENTEET

API	Application Programming Interface
DCOM	Distributed Component Object Model
DLL	Dynamic Link Library
JAX-WS	Java API for XML Web Services
PDA	Personal Digital Assistant
RMI	Remote Method Invocation
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XML	eXtensible Markup Language

SISÄLLYS

1 JOHDANTO

Navigointia helpottavat laitteet ja sovellukset ovat yleistyneet viime vuosina. Navigaattoreita on tarjolla kirjava valikoima eri valmistajien laitteita, ja navigoinnista on tullut myös pysyvä ominaisuus matkapuhelimiin. Navigoinnin lisäksi navigaattoreihin on nykyään saatavilla liikennetiedotteita, jotka helpottavat matkantekoa. Autoilija voi muun muassa välttää pahat liikenneruuhkat, tietyöt ja onnettomuudet, mikä lisää matkanteon sujuvuutta ja nopeuttaa päämäärän saavuttamista.

Suomessa Destia Traffic, joka on osa Destia Oy:tä, tarjoaa ajantasaista tietoa liikenteestä, sääoloista sekä liikennettä haittaavista häiriöistä. Destia Trafficin ympärivuorokautisesti toimiva liikenteen palvelukeskus kerää tietoja liikenteestä useista eri lähteistä ja kokoaa niistä tiedotteita eteenpäin lähetettäväksi. Ennen tietojen lähettämistä asiakkaille tai yhteistyökumppaneiden palveluihin tietojen oikeellisuus vahvistetaan. [1.]

Tämän työn tilaajana toimii Ebsolut Oy, joka on kajaanilainen ohjelmistotuotannon palveluyritys. Ebsolut Oy suunnittelee ja toteuttaa asiakaskohtaisia tietojärjestelmähankkeita sekä tarjoaa osaavaa alihankintapalvelua. Näiden lisäksi Ebsolut Oy:llä on tarjolla omia tuotteita muun muassa kuljetusyritysten käyttöön. Yritys on perustettu vuonna 2000 ja on tätä nykyä osa KPO-konsernia. [2.]

Työn tarkoituksena on toteuttaa Ebsolut Oy:lle liikennetiedotejärjestelmä kevään 2009 aikana. Työ on jaettu kahteen osaan, ja se on toteutettu yhdessä insinööriopiskelija Jukka Korhosen kanssa. Ensimmäisessä osassa kuvataan suunnitelma liikennetiedotejärjestelmän perustamiseksi ja toisessa osassa laaditaan suunnitelma liikennetiedotejärjestelmän tiedonsiirron ja järjestelmää hyödyntävän sovelluksen toteuttamiseksi. Tässä dokumentissa on kuvattu suunnitelma asiakassovelluksen ja palvelimen välisestä tiedonsiirrosta sekä esitelty liikennetiedotejärjestelmää hyödyntävän asiakassovelluksen suunnittelu sekä rakenne. Enemmän asiaa liikennetiedotteista, niiden välittämiseen käytettävistä tiedonsiirtokanavista sekä palvelinpuolen suunnittelusta ja rakenteesta löytyy Jukka Korhosen insinöörityöstä ”Suunnitelma liikennetiedotejärjestelmän perustamiseksi”.

Luvussa 2 esitellään palvelun suunnittelun lähtökohtia sekä taustaa liikennetiedotteista. Siinä kuvataan järjestelmien integroimiseen liittyviä yksityiskohtia sekä palvelun toteuttamiseen käytettävää arkkitehtuuria. Lisäksi tarkastellaan liikennetiedotteiden välittämiseen soveltuvia

tiedonsiirtotekniikoita, joista valittiin sopivin järjestelmän toteuttamiseen. Lopuksi kuvataan liikennetiedotejärjestelmä yleisellä tasolla.

Luvussa 3 käsitellään tarkemmin tiedonsiirtoon käytettävää Web Services -tekniikkaa. Tekniikasta kuvataan sen rakenne sekä tuodaan esille tekniikan hyötyjä, joita sillä on järjestelmän toteutuksen kannalta. Lopuksi kuvataan yksinkertaisen Web Services -palvelun toteuttaminen Java-ohjelmointikielellä.

Luvussa 4 testataan Web Services -palvelun toimintaa. Testausta varten luodaan itse palvelu sekä kaksi palvelua käyttävää asiakassovellusta. Palvelua ajetaan työpöytäkoneelle asennetulla palvelimella. Palvelua testataan sekä kiinteästi verkossa olevalla pöytäkoneella sekä liikkuvalla päätelaitteella.

Luvussa 5 esitellään liikennetiedotekomponentti, joka hoitaa tietoliikenteen palvelimen ja sovelluksen välillä. Siinä kuvataan komponentin ja sovelluksen välinen kommunikointi sekä liikennetiedotteiden päivittäminen ja hakeminen palvelimelta. Komponentti hyödyntää edellä kuvattua Web Services -tekniikkaa liikennetiedotteiden hakemiseen. Luvussa on kuvattu vastausviestin käsittely sekä tiedotteiden välittäminen sovelluksen käyttöön.

Luvussa 6 esitellään suunnitelma yhdestä tavasta toteuttaa sovellus, joka hyödyntää toteutettavaa liikennetiedotejärjestelmää. Sovellus toteutetaan osaksi Ebsolut Oy:n omaa tuotetta, EMobile Kuljetuksen ajoneuvosovellusta. Lopuksi luvussa 7 analysoidaan suunnitelman onnistumista sekä suunnitelman toteuttamiskelpoisuutta käytännössä.

2 LIIKENNETIEDOTEJÄRJESTELMÄN SUUNNITTELU

Liikennetiedotejärjestelmän suunnittelussa lähdettiin ajatuksesta, jossa liikennetiedotteet tuli saada välitettyä loppukäyttäjien erilaisiin, eri tekniikoita käyttäviin päätelaitteisiin mahdollisimman yksinkertaisesti ja vähällä vaivalla. Järjestelmän toteuttaminen itsenäiseksi sovellukseksi sekä integroiminen jo olemassa oleviin sovelluksiin tuli onnistua ilman isompia laite- tai alustakohtaisia toteutuksia. Toisin sanoen järjestelmän tuli olla sekä alusta- että laiteriippumaton. Tässä luvussa esitellään liikennetiedotejärjestelmän suunnittelun lähtökohtia sekä teknisiä ratkaisuja järjestelmän toteuttamiseksi.

2.1 Taustaa

Tässä luvussa on kerrottu taustaa liikennetiedotteista, ja sen sisältö on koottu pääosin Jukka Korhosen insinöörityöstä [3].

Liikennetiedotteet eivät ole uusi asia Suomessa, vaan niiden historia alkaa jo 1990-luvun puolestavälistä, jolloin liikennetiedotteita alettiin lähettää muun muassa Yleisradion Radio Suomen taajuudella. Alkuvaiheessa liikennetiedotteet välitettiin vain ULA-radioverkon RDS-tiedonsiirtokanavalla toimivan RDS-TA-palvelun kautta. Palvelun ollessa toiminnassa liikennetiedotteet välittyvät kuulijoille radiotoimittajan lukemana.

Vuosina 1998–2006 Tiehallinto pilotoi RDS-TMC-palvelua Yleisradion ja Digita Oy:n kanssa. RDS-TMC-palvelussa liikennetiedoteviestit lähetetään digitaalisesti RDS-tiedonsiirtokanavaa hyödyntäen. Vuoden 2007 alusta RDS-TMC-palvelu kaupallistettiin, ja sitä hoitaa Destia Oy. Palvelussa liikennetiedotteet lähetetään suoraan Destian liikenteen palvelukeskuksesta palvelua tukeviin vastaanottimiin.

Destia kerää liikennetietoja useista eri lähteistä. Kerätyt tiedot käsitellään, ja niistä muodostetaan liikennetiedotteita. Liikennetiedotteet välitetään eteenpäin tietolajeittain palvelun tarjoajille sekä suoraan TMC-palveluun soveltuville päätelaitteille, esimerkiksi navigaattoreihin. Liikennetiedot jaetaan tieliikenteen osalta tietolajeihin, joita ovat liikenteen häiriötiedot, tietyötiedot, sujuvuus- ja mittalaitetiedot.

Radioverkon kautta välitettävien liikennetiedotteiden paikannus tapahtuu TMC-pisteiden avulla. Suomessa maanteiden (numerot 1–999) sekä suurten kaupunkiseutujen (Helsinki, Tampere, Turku, Oulu) risteykset on merkitty TMC-pisteellä.

Destia tarjoaa liikennetiedot myös XML-muotoisina tiedotteina. Ne sisältävät radioverkon kautta lähetettäviin tiedotteisiin verrattuna enemmän ja tarkempaa tietoa liikenteestä. Lisäksi XML-muotoisia tiedotteita käytettäessä tiedotteita voidaan suodattaa halutuilla hakuehdoilla. Näistä syistä liikennetiedotejärjestelmän toteuttamiseen valittiin käytettäväksi XML-muotoisia tiedotteita.

Tiedonsiirtokanavaksi liikennetietojen lähettämiseen valittiin GPRS-yhteys. Radioverkkoon verrattuna GSM-verkossa tapahtuva tiedonsiirto on huomattavasti nopeampaa, ja GSM-verkko on kaksisuuntainen. Kaksisuuntaista verkkoa käyttävästä palvelusta saadaan kysely-pohjainen, jolloin kyselyyn voidaan sisältää maantieteellinen alue, jolle liikennetiedotteet halutaan. Näin kyselyn tekijälle saadaan välitettyä vain olennaisin tieto sekä siirrettävän datan määrää saadaan pudotettua.

2.2 Järjestelmäintegraatio

Tässä luvussa on kerrottu järjestelmien integroimiseen liittyvistä yksityiskohdista, ja sen sisältö on pääosin koottu Sami Tähtisen kirjasta Järjestelmäintegraatio: tarve, vaihtoehdot, toteutus [4].

Järjestelmäintegraatio on ajattelutapa, jolla voi hahmottaa yrityksen tietoteknistä arkkitehtuuria. Se ei ole tuote tai teknologia vaan kokoelma erilaisia ajatus- ja suunnittelumalleja sekä käytäntöjä, joiden avulla yrityksen tietotekniset järjestelmät saadaan valjastettua mahdollisimman hyvin yrityksen liiketoiminnan tarpeisiin. Se on tapa hahmottaa yrityksen omaa teknistä järjestelmäarkkitehtuuria suhteessa yrityksen liiketoiminnan tarpeisiin. Integraatio on kokoelma toimintatapoja, jotka voivat tehostaa merkittävästi yrityksen toimintaa ja lisätä sen joustavuutta samoin kuin parantaa monitorointia ja raportointia.

Yrityksen liiketoimintaprosessit jakautuvat lähes aina useiden eri sovellusten alueelle. Järjestelmäintegraation avulla muutoin keskenään yhteensopimattomat tietotekniset sovellukset saadaan kommunikoimaan automatisoidusti keskenään. Useista yksittäisistä sovelluksista ra-

kennetaan yrityksen liiketoiminnan kannalta mahdollisimman tarkoituksenmukainen kokonaisuus.

Järjestelmäintegraatiossa on yksinkertaisimmillaan kysymys informaation siirtämisestä integroitavien järjestelmien välillä, tietomuunnoksista näiden järjestelmien sisäisten esitysmuotojen välillä sekä kokonaisprosessin kontrolloinnista sekä tähän liittyvästä valvonnasta ja raportoinnista. Informaation siirto eri järjestelmien välillä on mahdollista ainoastaan, mikäli integroitavat järjestelmät tarjoavat jonkinlaiset rajapinnat, joiden välityksellä järjestelmästä voidaan hakea ja jota kautta järjestelmiin voidaan syöttää informaatiota. Sovellukset eroavat toisistaan informaation esitystavan perusteella, jonka vuoksi informaatiota voidaan siirtää järjestelmien välillä ainoastaan erityyppisiä tietomuunnoksia käyttämällä. Integraatioratkaisuissa tärkein asia on tiedonsiirron ja tietomuunnosten tehokas hallinta.

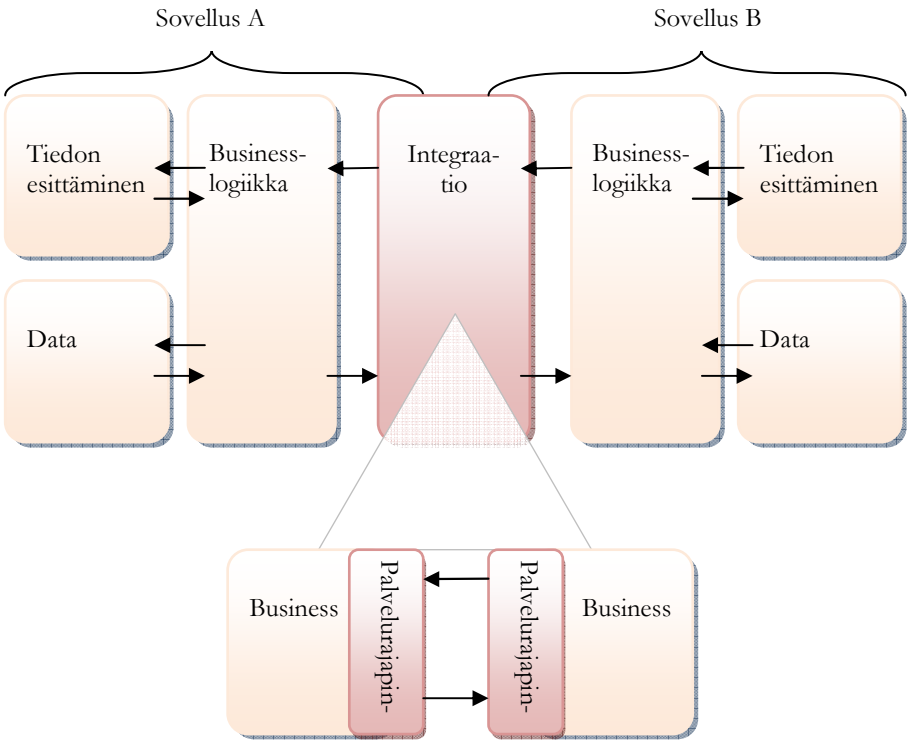
Järjestelmien välisessä kommunikaatiossa informaation siirtoon käytetään usein jotain verkkokerroksen päällä toimivaa etäkutsu- tai sanomansiirtoarkkitehtuuria. Tällainen toiminnallisuus mahdollistaa ohjelmallisten etäkutsujen suorittamisen (synkroninen tiedonsiirto) tai sanomien välittämisen (asynkroninen tiedonsiirto) eri käyttöjärjestelmillä toimivien ja eri tekniikoilla rakennettujen ohjelmistojen välillä.

2.3 Arkkitehtuuri

Yrityksen tietotekninen arkkitehtuuri on kuvaus järjestelmien välisestä yhteistoiminnasta, eli siinä on kysymys järjestelmäintegraatiosta. Tietojärjestelmien kokonaisarkkitehtuurin suunnittelussa voidaan kerätä, dokumentoida ja hyväksikäyttää yleiskäyttöisiä arkkitehtuurimalleja. Erityyppisten integraatiomallien tarkoituksena on nopeuttaa tietoteknisen arkkitehtuurin kehitystä ja tehostaa projektinhallintaa. [4.]

Yksi arkkitehtuurimalleista on SOA- (Service Oriented Architecture) eli palvelukeskeinen arkkitehtuuri. Palvelukeskeisen arkkitehtuurin tarkoituksena on mahdollistaa eri ohjelmistojen tarjoamien palveluiden ja informaatiosisältöjen mahdollisimman joustava ja monipuolinen hyväksikäyttö yrityksen liiketoiminnan apuna. Sen lähtökohtana on alustariippumattomuus, jossa liiketoiminta luo tarpeet. Palvelukeskeisessä arkkitehtuurissa lähdetään ajatuksesta, että eri ohjelmistot tarjoavat erityyppisiä palveluita yrityksen kokonaisratkaisun kokoamiseksi. Yksittäisiä palveluita sopivasti yhdistelemällä voidaan toteuttaa tietyn yrityksen liike-

toimintaa mahdollisimman hyvin palvelevia kokonaisuuksia. Palvelukeskeisessä arkkitehtuurissa ei ole teknisesti mitään uutta, vaan kyse on yksinkertaisesti eri ohjelmistojen lisääntyvästä keskinäisestä vuoropuhelusta. Tärkeintä palvelukeskeisessä arkkitehtuurissa on sen tiedostaminen. Ohjelmistoille tulee luoda yhä avoimempia rajapintoja ja tehokkaampia palveluita, joiden kautta ohjelmistojen toiminnallisuutta voidaan käyttää hyväksi. Kuvassa 1 on kuvattu kahden sovelluksen palvelukeskeisen arkkitehtuurin mukainen integrointi. Sovellusten integrointi tapahtuu molempien sovellusten tarjoamien palvelurajapintojen avulla. Niiden avulla sovellukset tarjoavat toisilleen palveluita, joita käyttämällä sovellukset voivat kommunikoida eli vaihtaa informaatiota keskenään. Sovellukset näkevät toisistaan ainoastaan rajapinnoissa määritetyt palvelut, joiden kautta vuoropuhelu tapahtuu. Toistensa sisäisestä toiminnasta tai toteutuksesta sovellukset eivät tiedä mitään, eikä tarvitsekaan tietää. [4, 5, 6, 7.]



Kuva 1. Palvelukeskeisen integraation arkkitehtuuri [6].

2.4 Tiedonsiirto

Valittaessa ratkaisua hajautetun sovelluksen kommunikoinnin toteuttamiseksi tulee harkita monia tekijöitä, kuten helppokäyttöisyyttä, ylläpidettävyyttä sekä suorituskykyä. Toteutettava liikennetiedotejärjestelmä tulee olemaan luonteeltaan reaaliaikainen järjestelmä, jossa tiedot

voivat päivittyä useasti päivän mittaan. Kuten aikaisemmin mainittiin, palvelusta tehdään kyselymuotoinen. Liikennetietoja haettaessa tehdään kysely, johon saadaan vastaus. Tästä johtuen palveluun ei tarvitse muodostaa viestijonoa, johon liikennetiedotteet kerätään sovelluksen haettavaksi. Epäonnistunut kysely voidaan tehdä aina uudestaan, jotta tiedot saadaan haetuksi. Liikennetiedotteet haetaan Destian palvelimelta aina minuutin välein, joten liikennetiedotepalvelimella on aina ajantasaisin tieto liikennetiedotteista [3.].

Järjestelmien väliseen kommunikointiin on tarjolla useita erilaisia teknologioita. Hyvin useat teknologiat ovat kuitenkin joko sidottuna tiettyyn valmistajaan tai ovat hankalia käyttää. Tiedonsiirtoa valittaessa vastaan tulivat muun muassa DCOM ja RMI. DCOM on Microsoftin teknologia, joka on tarkoitettu käytettäväksi lähinnä Microsoftin omien tuotteiden ja teknologioiden kanssa [5]. RMI on taas puolestaan Javan tarjoama teknologia, joka mahdollistaa oliokutsujen ja paluuarvojen välittämisen eri päätelaitteissa ajettavien Java-ohjelmien välillä [5]. Nämä kaksi edellä mainittua teknologiaa ovat liian tiukasti sidoksissa tiettyyn toimittajaan, jolloin alusta- ja teknologiariippumattoman järjestelmän rakentaminen ei onnistu.

Ebsolut Oy:llä on olemassa oma tuote tiedonsiirron toteuttamiseksi. EMobile CS on yleinen tiedonsiirtoratkaisu sovelluksille, jotka käyttävät keskinäiseen kommunikointiin joko kiinteää tai langatonta verkkoa. EMobile CS on ratkaisu, joka on suunniteltu liikkuvien sekä perinteisten päätelaitteiden tuottaman tiedon siirtämiseen toisille tietojärjestelmille. EMobile CS tarjoaa rajapinnan tiedon välitykseen sekä tiedonsiirron statuksen seurantaan. EMobile CS tiedonsiirtoratkaisuna on turhan raskas tämän palvelun toteutukseen, koska EMobile CS:ssä on oma viestijono, joka vaatii tietokannan päätelaitteeseen. Viestijono sekä sen käsittely vaativat päätelaitteelta huomattavasti resursseja. Palvelu on tarkoitus pystyä ottamaan käyttöön myös kevyissä mobiililaitteissa, kuten esimerkiksi matkapuhelimissa. Viestijonoa sekä sen käsittelyä ei myöskään tarvita liikennetiedotejärjestelmän reaaliaikaisuuden takia. [2.]

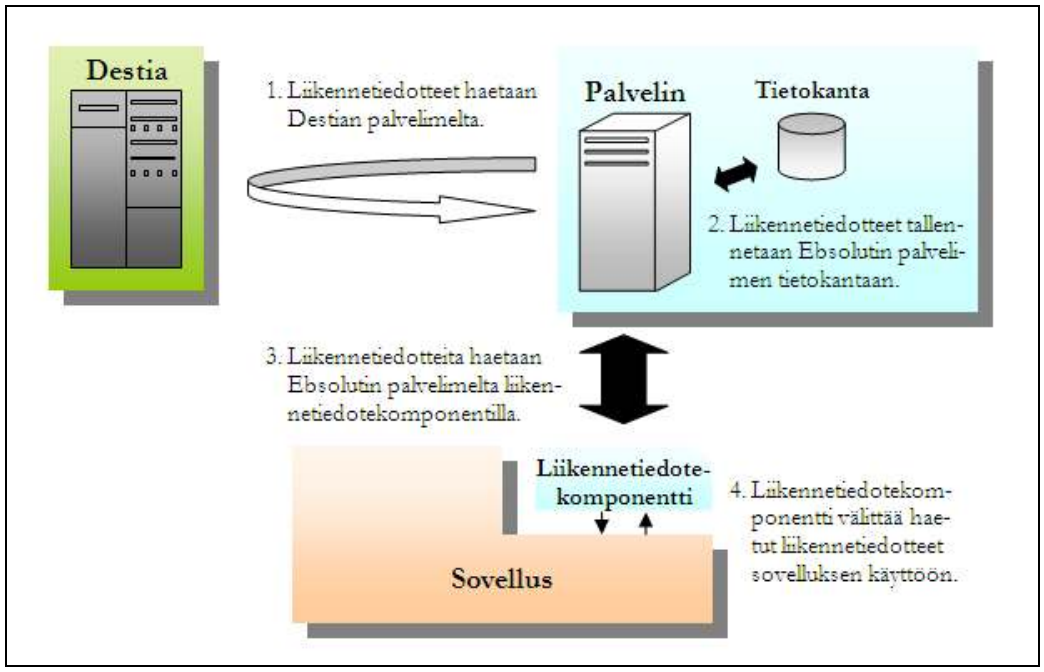
Tiedonsiirtotavaksi valittiin Web Services eli www-sovelluspalvelu, joka on sovelluksen toiselle sovellukselle tarjoama palvelu. Se on tapa, jonka avulla erityyppiset, eri tekniikoilla toteutetut ohjelmistot kykenevät välittämään tietoa keskenään. Siinä yhdistyvät yleisimmät teknologiat, kuten internet, HTTP ja XML. Web Services -teknologiaa käyttäen on mahdollista toteuttaa suurin osa palvelukeskeiseen arkkitehtuuriin pohjautuvista järjestelmistä. Web Services perustuu yksinkertaisiin XML-sanomiin, jotka välitetään yleisimmin HTTP-protokollan välityksellä. Muutkin siirtotiet ovat tosin mahdollisia. Näihin tekniikoihin perustuva sanomanvälitys on helppo toteuttaa lähes millä tahansa tekniikalla tai ohjelmointikielel-

lä. Web Services -palvelut ovat käytettävissä riippumatta käyttöjärjestelmästä tai kääntäjästä. [4, 5]

2.5 Järjestelmän kuvaus

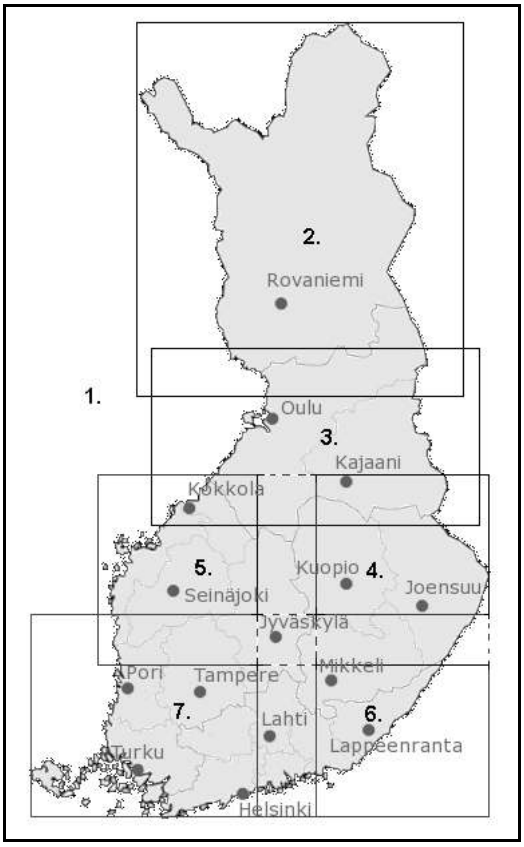
Toteutettava liikennetiedotejärjestelmä sisältää palvelinsovelluksen sekä päätelaitteeseen tulevan liikennetiedotekomponentin. Näiden lisäksi järjestelmään kuuluu liityntä Destian palvelimeen sekä päätelaitteessa yhteys liikennetiedotekomponentin ja liikennetiedotteita tarvitsevan sovelluksen välillä.

Liikennetiedotteet haetaan Destian palvelimelta Ebsolutin palvelimelle ajastetusti noin minuutin välein. Näin Ebsolutin palvelimella on aina ajantasaiset liikennetiedotteet. Liikennetiedotteet saadaan Destian palvelimelta XML-tiedostoina, joista tiedot parsitaan ja tallennetaan palvelimella olevaan tietokantaan. Tämän jälkeen liikennetiedotteet ovat haettavissa palvelimelta. Päätelaitteeseen tulevan liikennetiedotekomponentin tehtävä on hoitaa tiedonsiirto sovelluksen ja palvelimen välillä. Komponentti suorittaa hakupyynnöt palvelimelle ja välittää haetut liikennetiedotteet sovelluksen käyttöön. Liikennetiedotekomponentti on nimensä mukaisesti erillinen komponentti, joka voidaan liittää osaksi liikennetiedotteita tarvitsevaa sovellusta. Järjestelmän rakenne on esitetty kuvassa 2. [3.]



Kuva 2. Liikennetiedotejärjestelmän rakenne.

Liikennetiedotteiden hakemiseksi Suomi on jaettu kuuteen osaan: Pohjois-, Etelä-, Itä-, Länsi-, Kaakkois- sekä Lounais-Suomeen. Näiden lisäksi liikennetiedotteet on mahdollista hakea koko Suomen alueelta. Web Services -palvelun on suunniteltu toimivan siten, että liikennetiedotteet on mahdollista hakea joko koordinaattien tai alueen avulla. Haut eivät palvelinpäässä eroa toisistaan, vaan aina palautuvat tietyn tai tiettyjen alueiden tiedot. Koordinaatein tehtävässä haussa liikennetiedotekomponentti saa koordinaatit asiakassovellukselta. Koordinaatit kysytään, ja ne lisätään hakuun aina ennen liikennetiedotteiden hakupyynnön suorittamista. Haun vastauksena palautuvat sen alueen tiedot, jolle koordinaatit osuvat. Vastauksena voi tulla myös useamman alueen tiedot, koska aluejaon on suunniteltu menevän osittain limittäin. Alustava aluejako on esitetty kuvassa 3. [3.]



Kuva 3. Aluejako.

3 WEB SERVICES

Tässä luvussa määritellään, mitä Web Services -termillä tarkoitetaan, mitä eri osa-alueita tekniikkaan kuuluu sekä mitä hyötyä Web Services -palveluiden käyttämisestä on.

3.1 Määrittely

The World Wide Web Consortium (W3C) on määritellyt suosituksia www-sovelluspalvelujen toteuttamiseen. Web Services on yksinkertaiseen sanomapohjaiseen kommunikaatioon perustuva tekniikka, johon liittyy kaksi perusmäärittelyä, SOAP ja WSDL. WSDL on kuvauskieli, jota käytetään Web Services -palvelun kuvaamiseen, ja SOAP on varsinainen etäkutsuprotokolla. Kommunikoinnissa käytettävät viestit ovat riippumattomia, joten niiden tulee sisältää kaikki tarvittavat tiedot, jotka ovat tarpeen viestin sisällön ymmärtämiseksi. [4, 5, 8.]

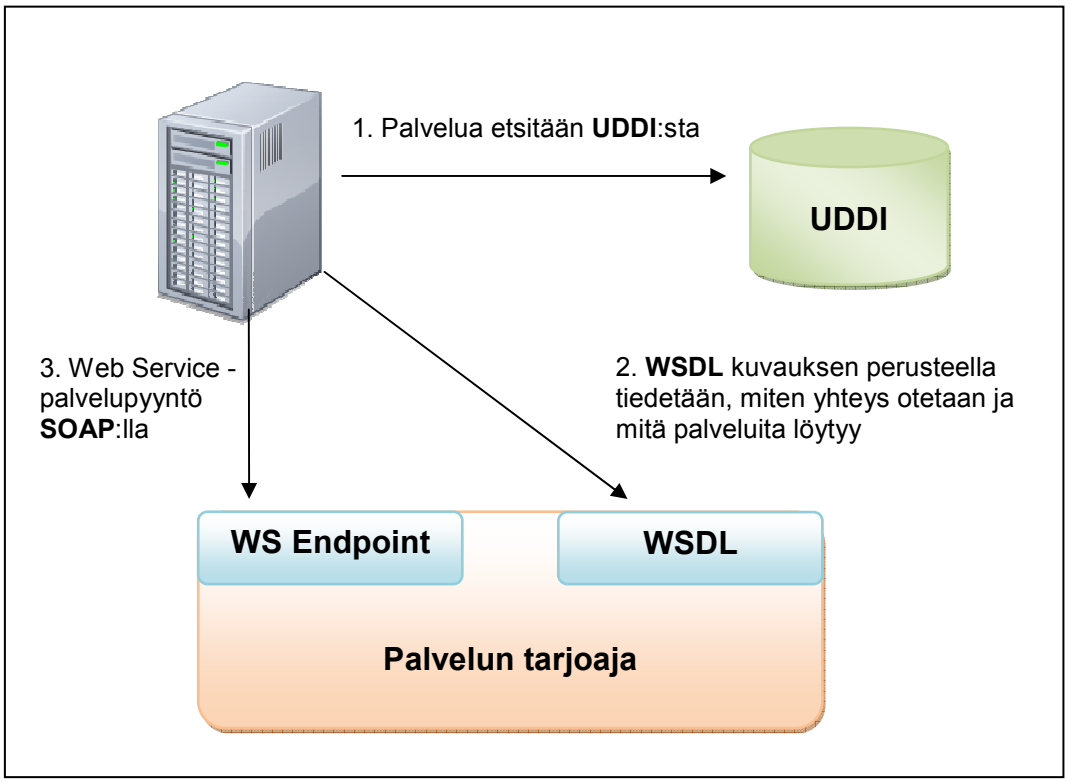
Näiden lisäksi Web Services -määrittelyyn luetaan usein erilaisia palveluita luetteloiva UDDI. UDDI on julkinen rekisteri, johon palvelun tarjoajat rekisteröivät palvelunsa. Tästä rekisteristä kuluttajat voivat hakea palveluita eri kriteerein. Mikäli palvelu löytyy, rekisteri tarjoaa kuluttajalle palvelun käyttöehdot ja palvelun osoitteen. [4, 5, 8.]

Näiden kolmen edellä mainitun tekniikan avulla Web Services voidaan yksinkertaisesti määritellä palveluna, jossa UDDI auttaa löytämään tietyntyyppisen palvelun, WSDL kuvaa tämän palvelun ja SOAP mahdollistaa palvelun kutsumisen (kuva 4). [4, 5.]

3.1.1 XML (eXtensible Markup Language)

XML on W3C:n suositus rakenteisten dokumenttien kuvaamiseen. Se on kuvauskieli, jota käytetään tiedon esittämiseen laitteistoriippumattomana ja itsensä kuvaavana tekstimuodossa. Kielen avulla esitetään dokumentin tietosisältö sekä tietosisällön rakenne käyttämällä XML-elementtejä ja -attribuutteja. XML on perusluonteeltaan metakieli, eikä esimerkiksi valmiiksi sisällä yhdenkään elementin määrittelyä. Tämän vuoksi täytyy olla jotkin sovitut säännöt, jotka vasta antavat merkityksen eri elementeille. Yleensä XML-dokumenttiin onkin liitetty erillinen skeema, jossa on määritetty, mitä tunnuksia dokumentissa on sallittu käyttää sekä näi-

den tunnusten rakenteet ja niihin liittyvät säännöt. Skeeman perusteella sovellus ja laitealusta osaavat muuntaa tiedot alustakohtaisiksi tietotyypeiksi. Kuvassa 5 on esitetty yksinkertainen XML-dokumentti. [4, 8, 9.]



Kuva 4. Web Servicen toimintaidea [6].

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookself>
   <book>
      <title>Da Vinci -code</title>
      <author>Brown, Dan</author>
      <price>39,90</price>
   </book>
</bookself>
```

Kuva 5. Yksinkertainen XML-dokumentti.

3.1.2 WSDL (Web Services Description Language)

WSDL on Web Service -palvelun kuvaamiseen käytettävä kieli. WSDL määrittelee XML-pohjaisen web-palvelun sekä palvelulle ne viestit, joita se ottaa vastaan ja joita se lähettää. Toisin sanoen se kertoo, mitä palveluita on tarjolla ja miten niitä voidaan käyttää. WSDL-

dokumentissa määritetään myös, kuinka viestit tulee lähettää ja missä verkko-osoitteessa palvelu sijaitsee. WSDL-dokumentin sisältö on kuvattu taulukossa 1. [4, 8.]

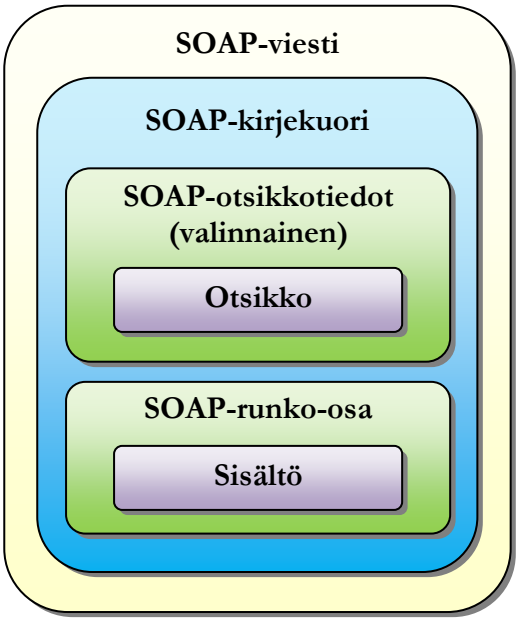
Taulukko 1. WSDL-dokumentin kuvaus [1].

Elementti	**Kuvaus**
tyypit (types)	määrittää viesteissä käytettävät XML-tietotyypit
viestit (message)	määrittää yksisuuntaisen operaation (kutsu- tai vastausviestin) ja sen mukana kuljetettavan types-elementin
porttityypit (portType)	määrittää abstraktilla tasolla operaatiossa vaihdettavat viestit
sidonnat (binding)	määrittää yksityiskohdat (tiedonsiirtoprotokollan ja viestien muodostamistyylin) portType-elementissä kuvattujen viestien vaihtoon
portti (port)	määrittää palvelun konkreettisen päätepisteen (endpoint), sen osoitteen ja binding-elementin, jossa on määritetty kutsuttavat operaatiot
palvelut (service)	määrittää päätepisteiden joukon, joiden kautta port- ja portType-elementeissä kuvatut operaatiot ovat käytettävissä

3.1.3 SOAP (Simple Object Access Protocol)

Kuten WSDL, myös SOAP on W3C:n standardoima XML-pohjainen protokolla, joka määrittelee yksinkertaisen kirjekuoren, jossa tieto liikkuu verkon yli alustariippumattomasti [5]. SOAP koostuu kolmesta pääkomponentista: pakollisesta SOAP-kirjekuoresta, valinnaisista SOAP-otsikkotiedoista sekä pakollisesta SOAP-viestistä [8].

SOAP-viesti (kuva 5) alkaa kirjekuori-elementillä (envelope), joka määrittelee XML-dokumentin SOAP-viestiksi. Sen sisälle tulevat valinnaiset SOAP-otsikkotiedot (header) sekä viestin runko-osa (body). Otsikkotiedot sisältävät palvelusta tietoja, joiden avulla eri tahot voivat tehdä eri toimenpiteitä, kuten esimerkiksi muuntaa tietoja tai suorittaa tietoturvaan liittyviä toimenpiteitä. Viestin runko-osa sisältää tärkeimmän osan eli itse viestin, joka on tarkoitettu ainoastaan viestin vastaanottajalle. Runko-osan sisään on mahdollista lisätä valinnainen virhe-elementti (fault) ilmaisemaan viesti virheviestiksi. Virhe-elementtien sisään tulevat virhettä kuvaavat tiedot. SOAP Messages with Attachments -spesifikaation mukaisen SOAP-viestin mukana voi olla myös liitetiedostoja. Liiteosa tulee varsinaisen SOAP-viestin perään. [8, 10.]



Kuva 6. SOAP-viestin rakenne [11].

3.1.4 UDDI (Universal Description, Discovery and Integration)

UDDI on Web Services -palveluita luetteloiva rekisteri, josta voi etsiä palveluita tarjoavia organisaatioita, niiden tarjoamia palveluita sekä teknisiä rajapintoja, joiden kautta näitä palveluita voidaan käyttää. Palvelun etsijä saa UDDI:n kautta WSDL-tiedoston, jonka perusteella asiakassovellus voidaan toteuttaa. UDDI-rekisteristä voi hakea palveluja esimerkiksi niiden nimen, sijainnin tai tyypin mukaan. [6.]

3.2 Lisäosat

Peruskomponenttien lisäksi Web Serviceen voi yhdistää lisäosia. Koska pelkästään Web Services -tekniikoiden peruskomponentteja käyttämällä palvelukeskeisen arkkitehtuurin toteutus ei ole mahdollista palveluiden muodostamisen, merkinnän ja palveluiden laadunvarmistuksen osalta, eri osapuolet ovat kehittäneet tekniikoita – lisäosia – kattamaan nämä alueet [8]. Ne ovat joukko standardiehdotuksia, jotka pyrkivät tuomaan Web Service -palveluihin ratkaisuja muun muassa tietoturvan, transaktioiden, reitityksen sekä luotettavuuden parantamiseksi [6]. Seuraavissa luvuissa on esitelty muutama tällainen lisäosa.

3.2.1 WS-Security

WS-Security ottaa kantaa XML-tietoturva-asioihin, kuten muun muassa digitaalisiin allekirjoituksiin ja XML-osien salaamiseen [6]. Se tarjoaa laajennuksen SOAP-protokollaan, jonka avulla voidaan toteuttaa viestien sisällön koskemattomuus ja luottamuksellisuus tietoturvallisten Web Services -palveluiden toteuttamiseksi. WS-Security ei varsinaisesti määritä uusia turvallisuusmalleja vaan mahdollistaa olemassa olevien mallien ja tekniikoiden yhteensopivan käytön Web Services -palveluissa. [8.]

3.2.2 WS-ReliableMessaging

Tietoverkkoja käytettäessä viestit saattavat kadota matkan varrella tai yhteydet katkeilla. WS-ReliableMessaging määrittää mekanismin, jonka avulla Web Services -palvelut voivat varmistua viestien perille menosta. Spesifikaatio määrittää viestitysprotokollan, jolla tunnistetaan, jäljitetään ja hallitaan viestien luotettavaa välitystä lähetys- ja vastaanottopäiden välillä. [8.]

3.2.3 WS-Addressing

Käytettäessä Web Services -palveluissa HTTP-protokollaa viestin lähettäjän ja vastaanottajan tiedot kulkevat HTTP-protokollan otsikkotiedoissa. Varsinaisessa SOAP-viestissä näitä tietoja ei ole lainkaan, jolloin käytettäessä välityspalvelimia tai yhteyden katkeamisen yhteydessä tämä tieto voi muuttua tai puuttua kokonaan. WS-Addressing tarjoaa toimivan ja riippumattoman tavan identifioida viestin lähettäjä ja vastaanottaja. [8.]

3.3 Web Services -tekniikan hyödyt

Järjestelmien integroimisen suhteen suurin hyöty on, että Web Services -palvelut perustuvat itsensä kuvaaviin XML-sanomiin, joten teknologian käyttö on alustariippumatonta. Toisin sanoen palveluntarjoaja ja asiakassovellus voivat olla ohjelmoitu eri ohjelmointikielillä, ne

voivat toimia eri käyttöjärjestelmissä sekä sovelluksen suorittavat laitteet voivat olla erilaisia. [8.]

SOAP-viestit välitetään yleisimmin HTTP-protokollan välityksellä, jolloin tieto reititetään saman tietoliikenneportin lävitse kuin verkkoselaintenkin liikenne. Tällöin vältytään ongelmallisten TCP/IP-porttien käytöltä. Lisäksi SOAP-protokolla on yksinkertainen ja kevyt käyttää, eikä sen pitäisi vaatia suuria prosessointitehoja laitteilta. [8.]

3.4 Web Services -palvelun ja -asiakassovelluksen toteuttaminen Javalla

JAX-WS (Java API for XML Web Services) on Javan tarjoama yksinkertainen tekniikka Web Services -palvelujen sekä palveluja kutsuvien asiakassovellusten toteuttamiseen. Se tarjoaa ohjelmistorajapinnan, joka helpottaa Web Services -tekniikoiden käyttämistä. Kuten edellä kuvattiin, SOAP-viestien rakenne on varsin monimutkainen. JAX-WS:n ohjelmistorajapinta piilottaa tämän monimutkaisuuden sovelluskehittäjältä eikä sovelluskehittäjän tarvitse huolehtia SOAP-viestien generoinnista tai parsimisesta. JAX-WS:n ajonaikainen järjestelmä muuntaa SOAP-viestit ohjelmistorajapintakutsuiksi ja rajapintavastaukset SOAP-viesteiksi. JAX-WS:llä luodut Web Services -palvelut ja asiakassovellukset ovat alustariippumattomia. Tämä tarkoittaa, että JAX-WS:llä luodut asiakassovellukset voivat hyödyntää Web Services -palveluja, joita ei ajeta Java-alustalla ja päinvastoin. Tämän mahdollistaa se, että JAX-WS käyttää W3C:n määrittelemiä teknologioita kuten, HTTP, SOAP ja WSDL. [11.]

Yksinkertaisen Web Services -palvelun luominen alkaa tuomalla Javan JAX-WS-kirjasto (javax.jws.WebService) import-komennolla palvelun toteuttavaan Java-luokkaan. Kirjaston lisäämisen lisäksi ennen luokan esittelyä tulee lisätä @WebService-lisäys. Lisäys määrittelee luokan automaattisesti Web Services -palvelun päätepisteeksi (endpoint). Metodit, jotka julkaistaan palvelun kautta asiakassovelluksille, vaativat javax.jws.WebMethod-kirjaston. Julkaistavat metodit vaativat lisäksi @WebMethod-lisäyksen. Yksinkertainen esimerkki Web Services -palvelun luomisesta on esitetty kuvassa 7, joka selventää lisäysten käyttöä palvelun luomisessa. [11.]

Ennen asiakassovelluksen luomista edellä kuvattu palvelu tulee kääntää ja julkaista jollakin Web Services -palveluita tukevalla palvelimella. Tämän jälkeen palvelun WSDL-tiedosto, joka kuvaa palvelun, on haettavissa palvelimelta. Tiedoston perusteella useat ohjelmistokehi-

tysalustat osaavat luoda valmiin rungon palvelun käyttämiseksi. Tiedoston hakemisen jälkeen palvelun käyttö ei eroa normaalista olio-ohjelmoinnista.

```
package helloservice.endpoint;

import javax.jws.WebService;
import javax.jws.WebMethod;

@WebService()
public class Hello
{
   private String message = new String("Hello, ");

   public void Hello() {}

   @WebMethod()
   public String sayHello(String name)
   {
      return message + name + ".";
   }
}
```

Kuva 7. Yksinkertainen Web Services -palvelu [11].

Kuvaan 8 on kerätty edellä kuvattua palvelua käyttävän asiakassovelluksen palvelun käyttämiseen tarvittavat rivit.

```
import helloservice.endpoint.HelloService;
import helloservice.endpoint.Hello;

…

// Uuden ilmentymän luonti palvelusta
HelloService service = new HelloService();

// Palvelun portin hakeminen
Hello port = service.getHelloPort();

// Palvelun julkisten metodien kutsuminen portin kautta
String response = port.sayHello(“World”);

// Vastauksen tulostaminen
System.out.println(response);

…
```

Kuva 8. Asiakassovelluksessa palvelun käyttämiseen tarvittavat rivit.

Palvelun toimivuutta voi testata ajamalla asiakassovelluksesta käännetyn tiedoston komentokehotteessa. Palvelusta tulee löytyä ohjelman käynnistävä metodi, josta asiakassovellus käynnistetään. Toimiakseen asiakassovellus vaatii, että palvelin on käynnissä ja että palvelu on

ladattu palvelimelle. Kuvassa 9 on tuloste, joka tulostuu, jos edellä kuvatut palvelu sekä asiakassovellus on saatu luotua oikein ja yhdistäminen palveluun on onnistunut.

Kuva 9. Vastauksen tuloste.

4 TIEDONSIIRRON TESTAUS

Tiedonsiirto testattiin luomalla Web Services -palvelu ja kaksi palvelua käyttävää asiakassovellusta. Palvelu luotiin Javalla, ja sen luomiseen käytettiin NetBeans 6.5 -kehitysalustaa, joka on ilmainen avoimen lähdekoodin ohjelmisto. Asiakassovelluksista toinen luotiin Javalla Windows-työpöytäympäristöön ja toinen C#-kielellä PDA-laitteeseen. PDA-laitteen käyttöjärjestelmänä oli Windows Mobile 6.1.

4.1 Palvelun perustaminen

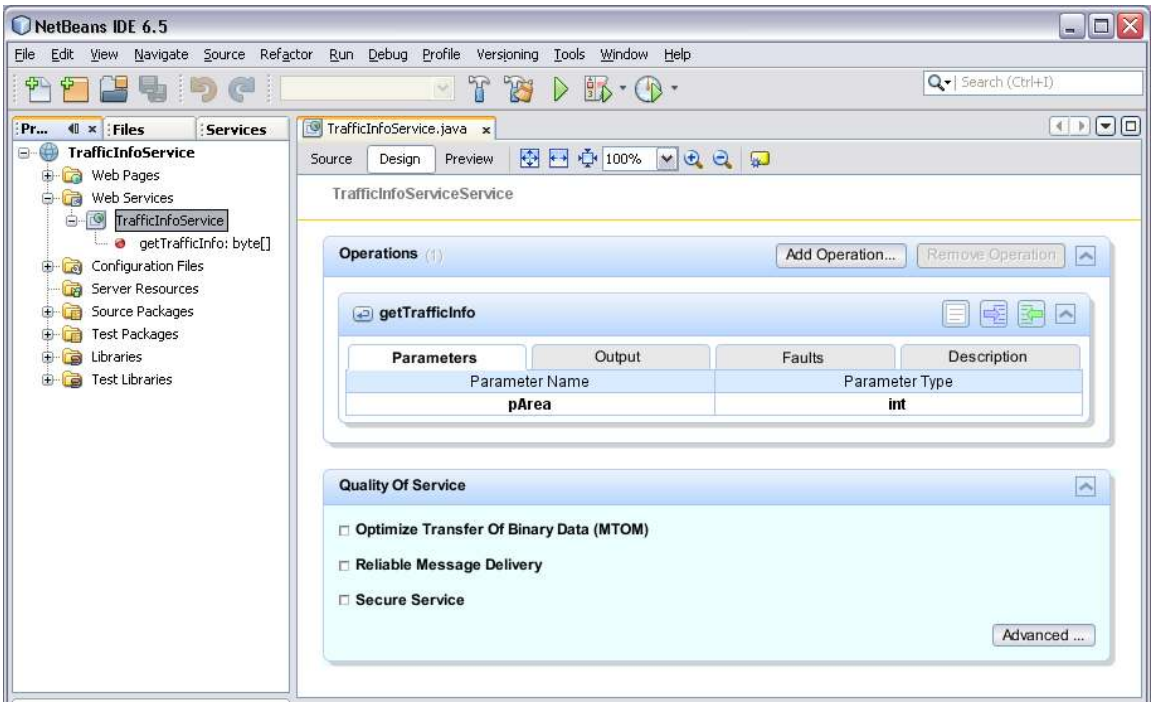
Ennen Web Services -palvelun testaamista piti valita palvelin, jolla palvelu saatiin ajettua. Palvelimeksi valittiin ilmainen avoimen lähdekoodin Apache Tomcat 6.0. Palvelin valittiin lähinnä sen ilmaisuuden, keveyden ja varmatoimisuuden vuoksi, mutta myös sen takia, koska valittu palvelin on Ebsolut Oy:llä osassa projekteista käytössä, joten osaamista ja tietotaitoa löytyi tarvittaessa.

Palvelun luominen NetBeans-ohjelmistolla aloitettiin luomalla uusi Web Application -projekti. Projektin asetuksiin tuli määrittää palvelin, jolle projekti julkaistaan. Palvelimen määrittämiseksi projektin asetuksiin määritettiin edellä mainitun Apachen Tomcat 6.0:n asennushakemisto. Tämän jälkeen projektiin luotiin uusi Web Services -palvelu. Palvelun asetuksia sekä palvelussa asiakassovelluksille julkaistavia metodeja pääsee NetBeansissä muokkaamaan graafisen käyttöliittymän kautta (kuva 10).

Palveluun perustettiin yksi palvelussa julkaistava metodi. Metodi otti parametrina vastaan pArea-nimisen int-tyyppiä olevan muuttujan, jolla testausvaiheessa mallinnettiin tuotantoon tulevaa ratkaisua. Parametria ei testauksessa käytetty muuhun kuin todentamaan, että tieto välittyy asiakassovelluksilta palvelulle. Paluuarvona metodi palautti tavutaulukon (byte []), jossa liikennetiedotteet ovat XML-muotoisena syötteenä. Graafisessa käyttöliittymässä määritetyistä asetuksista NetBeans generoi koodipohjan Web Services -palvelulle. Ohjelmoijalle jäi toteutettavaksi ainoastaan metodien toteutus.

Liikennetiedotteet tullaan välittämään palvelimelta liikennetiedotekomponentille XML-muotoisena syötteenä, joten tiedonsiirron testausta varten luotiin XML-muotoinen liikenne-

tiedotteita muistuttava sanoma. Javalta löytyy valmiit kirjastot XML-muotoisten dokumenttien luomiseen. Tiedot häiriöstä kirjoitettiin vielä testausvaiheessa suoraan koodin sekaan, mutta tuotantoon siirryttäessä tiedot haetaan tietokannasta, ja tiedoista generoidaan palvelupyyntöön vastaus.



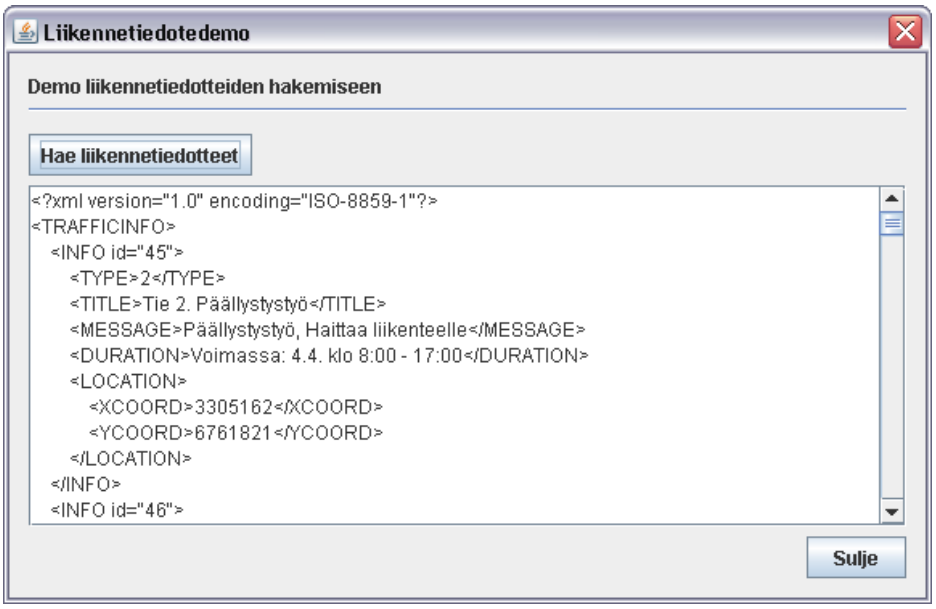
Kuva 10. Web Services -palvelun luominen.

XML-muotoisen dokumentin generoinnin jälkeen sanoma muunnettiin sarjamuotoiseksi merkkijonoksi (serialisointi). Serialisointia varten projektiin tuli ottaa mukaan yksi kolmannen osapuolen toteuttama Java-kirjasto. Kirjasto oli saatavilla Ebsolut Oy:n aikaisemmista projekteista, joissa sitä oli käytetty XML-muotoisten sanomien serialisointiin. Lopuksi ennen vastauksen lähettämistä merkkijono muunnettiin tavutaulukoksi ja palautettiin palvelupyynnön vastauksena asiakassovellukselle. Kun palvelu oli saatu luotua, palvelin käynnistettiin ja toteutettu palvelu sijoitettiin palvelimelle.

4.2 Asiakassovellusten toteutus ja tiedonsiirron testaus

Palvelun teknologia- ja alustariippumattomuutta testattiin luomalla kaksi asiakassovellusta, toinen Java-kielellä Windows-työpöytäympäristöön ja toinen C#-kielellä PDA-laitteeseen Windows Mobile -ympäristöön.

Javalla tehdyn asiakassovelluksen toteutus aloitettiin luomalla uusi Java-sovellus. Sovellukselle luotiin yksinkertainen käyttöliittymä (kuva 11) sekä Web Services -asiakassovellus, jolla edellä kuvatun palvelun toimivuutta testattiin. Asiakassovelluksen luomisen yhteydessä palvelimelta haettiin palvelun WSDL-tiedosto, jossa on kuvattu palvelun nimi, päätepiste sekä tiedonsiirtotavat. Tiedoston avulla NetBeans osasi generoida asiakassovellukselle valmiin rungon palvelun käyttämiseksi.



Kuva 11. Tiedonsiirron testaamiseen käytetty käyttöliittymä.

Palvelun käyttämiseksi palvelusta luotiin sovellukselle uusi ilmentymä new-operaattorin avulla. Luodulta ilmentymältä kysyttiin portti, jonka kautta palvelun metodeja pystyy kutsumaan. Tämän jälkeen palvelun käyttäminen näkyi ohjelmoijalle tavallisina metodikutsuina. Portin kautta kutsuttiin palvelun getTrafficInfo-metodia, ja kutsun paluuarvona saatiin tavutaulukko, joka muunnettiin merkkijonoksi. Merkkijono tulostettiin käyttöliittymään luotuun tekstikenttään (kuva 10), josta oli luettavissa XML-muodossa olevat liikennetiedotteet.

Palvelun alustariippumattomuutta testattiin luomalla C#-kielellä toteutettavaa liikennetiedotekomponenttia muistuttava Windows Mobile -sovellus. Komponentista luotiin dynaaminen luokkakirjasto (DLL), joka on mahdollista ottaa käyttöön halutuissa sovelluksissa lisäämällä viite kyseiseen tiedostoon. Samalla luotiin yksinkertainen Windows Mobile -sovellus, jossa luotu luokkakirjasto otettiin käyttöön ja joka hyödynsi komponentin tarjoamaa liikennetiedotepalvelua. Komponentti sekä sovellus luotiin Visual Studio 2005 -kehitysalustalla. [3.]

C#-kielellä toteutettu komponentti saatiin luotua siten, että sen avulla saatiin haettua liikennetiedotteet palvelimelta. Tiedot saatiin virheettöminä perille ja ne saatiin esitettyä PDA-laitteeseen luodussa käyttöliittymässä. Tarkka kuvaus komponentin rakenteesta ja toiminnasta löytyy Jukka Korhosen insinöörityöstä. [3.]

5 LIIKENNETIEDOTTEITA HAKEVAN KOMPONENTIN SUUNNITELMA

Liikennetiedotteita hakevasta komponentista tehdään palvelu, joka voidaan integroida osaksi mitä tahansa liikennetiedotteita tarvitsevaa sovellusta tai järjestelmää. Komponentti toimii sovelluksen ja palvelimen välillä hoitaen tiedonsiirron sekä liikennetiedotteiden välittämisen sovellukselle. Komponenttiin toteutetaan Web Services -asiakassovellus, joka hyödyntää Ebsolut Oy:n palvelimelle toteutettavaa Web Services -palvelua liikennetiedotteiden hakemiseen.

5.1 Liikennetiedotekomponentin ja sovelluksen välinen kommunikointi

Liikennetiedotekomponentin ja sovelluksen välinen kommunikointi tapahtuu normaaleilla metodikutsuilla. Liikennetiedotekomponentti tarjoaa sovellukselle julkiset metodit liikennetiedotteiden hakemiseen, hakuintervallin asettamiseen sekä komponentin ajastetun haun käynnistämiseen ja pysäyttämiseen. Sovelluksen tulee puolestaan toteuttaa rajapinta, jonka kautta komponentti välittää haetut liikennetiedotteet sovelluksen käyttöön sekä kysyy uuden paikkatiedon aina ennen uutta ajastettua hakua. Sovelluksen alustaessa liikennetiedotekomponentin komponentille viedään parametreina tiedot siitä, mikä on ajastetun haun aikaväli sekä mikä komponenttia käyttävän sovelluksen luokka toteuttaa rajapinnan, jonka kautta komponentti voi kommunikoida sovelluksen kanssa. Rajapinta tarjoaa metodit liikennetiedotteiden välittämiseen sekä paikkatiedon kysymiseen.

5.2 Liikennetiedotteiden päivittäminen ja hakeminen

Liikennetiedotteet on mahdollista päivittää joko ajastetusti tai manuaalisesti. Ajastetusti tapahtuvaa hakua käytetään tiellä liikuttaessa, jolloin liikennetiedotteiden jatkuva päivittyminen on tärkeää. Ajastetusti tapahtuvan haun aikaväli ilmoitetaan minuuttimääräisenä kokonaislukuna. Näin ollen yksi minuutti tulee olemaan lyhin aikaväli, jonka välein tiedot voi päivittää. Minuuttia pienempää aikaväliä ei hyödytä edes käyttää, koska liikennetiedotteet päivitetään Destian palvelimelta Ebsolut Oy:n palvelimelle minuutin välein. Jos ajastettua hakua ei käytetä, aikaparametrina viedään nolla. Aikaväli on mahdollistaa alustaa myös myöhemmin, ja

sitä voi tarvittaessa muuttaa. Komponentin alustuksen jälkeen ajastettu haku voidaan käynnistää ja tarvittaessa pysäyttää. Manuaalisesti tapahtuva haku tapahtuu liikennetiedotekomponenttia käyttävän sovelluksen kautta.

Liikennetiedotteiden haku palvelimelta komponentille toteutetaan suoritettavaksi joko koordinaattien tai alueen perusteella. Alustava aluejako on esitetty luvussa 2.5. Haut eivät vastausten osalta poikkea toisistaan, vaan vastauksena saadaan aina tietyn alueen kattavat tiedot. Koordinaatteja käytettäessä on mahdollista myös saada useamman alueen tiedot, jos koordinaatit osuvat alueiden päällekkäin meneville osille. Koordinaatteja hyödynnetään ajastetusti tapahtuvassa haussa, jossa komponentti kysyy sovellukselta koordinaatit aina ennen haun suoritusta. Näin liikennetiedotteet saadaan aina sille alueelle, missä päin liikutaan.

5.3 Vastauksen käsittely

Haun vastauksena palvelimelta saadaan tavutaulukossa olevat liikennetiedotteet. Tavutaulukko muunnetaan ensin merkkijonoksi, josta on mahdollista muodostaa XML-muodossa oleva dokumentti. Dokumentista parsitaan saadut liikennetiedotteet ja tallennetaan liikennetiedotekomponentin tietorakenteisiin. Yksittäisestä liikennetiedotteesta luodaan olio, johon tallennetaan häiriön tyyppi, otsikko, tarkka kuvaus, kesto, alku- ja loppupiste, suunta pisteiden välillä, alueen kuvaus sekä tien nimi ja numero. Yksittäiset liikennetiedotteet tallennetaan taulukkoon, joka välitetään sovellukselle.

6 SUUNNITELMA LIIKENNETIEDOTEJÄRJESTELMÄÄ HYÖDYNTÄVÄN SOVELLUKSEN TOTEUTTAMISEKSI

Tässä luvussa esitetään yksi tapa toteuttaa sovellus, joka hyödyntää liikennetiedotejärjestelmää. Sovellus toteutetaan osaksi EMobile Kuljetuksen ajoneuvosovellusta. EMobile Kuljetus on Ebsolut Oy:n ja Sunit Oy:n yhdessä kehittämä kuljetusyrityksille suunnattu kokonaissovellus, joka sisältää työkaluohjelmistot ajoneuvoon sekä kuljetusyrityksen toimistoon [2].

6.1 Määrittely

EMobile Kuljetuksen ajoneuvosovellukseen luodaan osasovellus liikennetiedotteita varten. Osasovellus hoitaa liikennetiedotekomponentin alustuksen sekä kommunikoinnin liikennetiedotekomponentin kanssa. Osasovellukselle luodaan yksinkertainen käyttöliittymä, jonka avulla liikennetiedotteiden haku voidaan suorittaa manuaalisesti, asettaa liikennetiedotteet näkyviin ajoneuvosovelluksen kartalle sekä selata ja lukea liikennetiedotteita. Liikennetiedotteiden selausta ja lukemista varten käyttöliittymässä tulee olemaan taulukko, johon tiedotteet listataan. Tietojen näyttämiseen kartalla sekä manuaalisesti suoritettavaa alueellista hakua varten käyttöliittymään luodaan painikkeet.

Ajettaessa ennalta suunniteltua reittiä liikennetiedotteet päivitetään automaattisesti ajastetusti tapahtuvalla haulla. Liikennetiedotekomponentti alustetaan päivittämään liikennetiedotteet minuutin välein. Kun reittiä lähdetään ajamaan, liikennetiedotteiden ajastettu haku käynnistetään ja liikennetiedotteet esitetään kartalla. Jos häiriö osuu ajettavalle reitille, käyttäjälle esitetään toteutuksen ensimmäisessä vaiheessa siitä huomautus. Häiriön kiertäminen jää käyttäjän vastuulle. Jatkossa sovellukseen tullaan toteuttamaan toiminto, jolla käyttäjältä kysytään reitin uudelleen reititystä häiriön välttämiseksi. Jos käyttäjä valitsee uudelleen reitityksen, ajoneuvosovellus laskee uuden reitin, joka kiertää häiriön.

6.2 Liikennetietojen esittäminen kartalla

Liikennetiedotteet esitetään ajoneuvosovelluksen kartalla taulukossa 3 esitetyillä kuvakkeilla. Kuvakkeet ovat Destian määrittelemiä suosituksia kunkin häiriötyypin kuvaamiseen, ja niitä

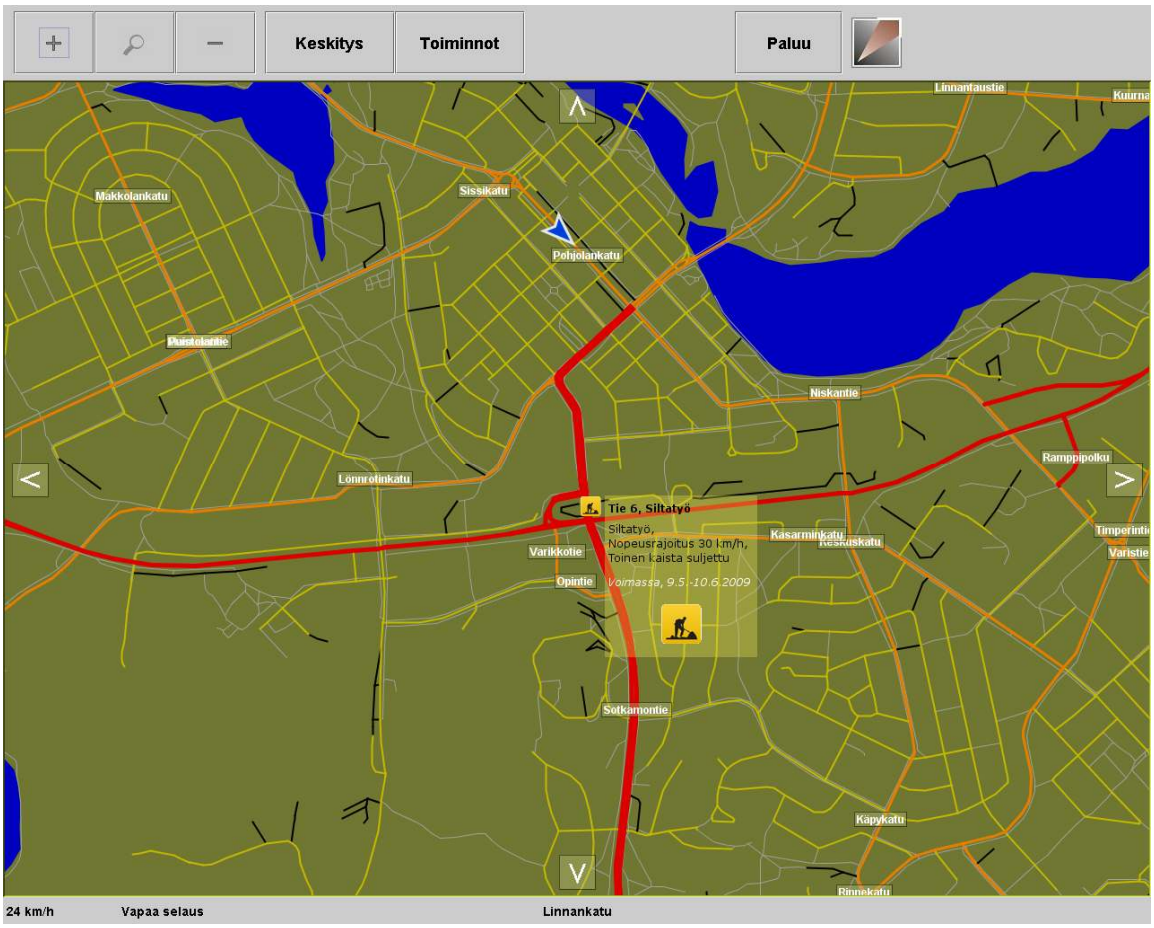
käytetään muun muassa osoittamaan häiriön paikka kartalla [13]. Kuvake valitaan jokaiselle liikennetiedotteelle sen tyypin mukaan. Kuvaketta napsauttamalla kartalla esitetään viesti, jossa on tarkka kuvaus häiriöstä. Liikennetiedotteesta esitetään häiriön tyyppi, otsikko, tarkka kuvaus sekä kesto. Häiriön vaikutusalue ja suunta esitetään kartalla graafisesti.

Taulukko 3. Häiriötyyppien kuvakkeet [12].

 tapahtuma	 liukas tie
 onnettomuus tai suuri tapahtuma	 rankka lumisade
 kapeat kaistat tai kaista suljettu	 rankkasade
 tietyö	 sumua tai usvaa
 tietyö, jolla väliaikainen nopeusrajoitus	 esteitä tiellä
 räjäytystyö	 eläimiä tiellä
 hidas jonoutuva liikenne	 lauttarajoitukset
 tie suljettu	

Karttapohjana EMobile Kuljetuksen ajoneuvosovelluksessa käytetään Java-pohjaista OpenMapia. OpenMap on Java Beans -pohjainen työkalu maantieteellistä informaatiota tarvitsevien sovellusten ja sovelmien luomiseksi. Sen ydin on joukko Swing-komponentteja, jotka ymmärtävät maantieteellisiä koordinaatteja. Nämä komponentit mahdollistavat karttaaineiston esittämisen sekä käyttäjän syötteiden hallitsemisen aineiston käsittelemiseksi. OpenMap tarjoaa yksinkertaisen tekniikan ikonien sekä viestien esittämiseen kartalla. Kuvassa 12 on kuva ajoneuvosovelluksessa käytettävästä kartasta. Kuvaan on hahmoteltu piirto-

ohjelmaa avuksi käyttäen alustava suunnitelma ikonin sekä viestin näyttämisestä karttapohjalla. Viesti on luotu mallintamaan mahdollista häiriötä. [13.]



Kuva 12. Ikonien ja viestien näyttäminen karttapohjalla [14].

7 ANALYSOINTI

Liikennetiedotejärjestelmän suunnittelu Ebsolut Oy:lle lähti ajatuksesta, jossa liikennetiedotteita pystyttäisiin välittämään Ebsolut Oy:n asiakkaille. Asiakkaina on muun muassa kuljetusyrityksiä, joilla on käyttöä liikennetiedotteille. Aktiivinen tiedottaminen liikenteessä esiintyvistä häiriöistä auttaa välttämään häiriöt, mikä nopeuttaa päämäärän saavuttamista ja lisää liikenteen sujuvuutta. Suomessa liikennetiedotteet tarjoaa Destia.

Tämän työn tarkoituksena oli laatia suunnitelma liikennetiedotejärjestelmän tiedonsiirron ja järjestelmää hyödyntävän sovelluksen toteuttamiseksi. Tiedonsiirtotekniikaksi valittiin Web Services -tekniikka, joka osoittautui yksinkertaiseksi ja helpoksi teknologiaksi toteuttaa sekä alusta- että teknologiariippumaton järjestelmä. Näin ollen liikennetiedotejärjestelmä voidaan ottaa käyttöön kirjavassa valikoimassa eri tekniikoilla toteutetuissa päätelaitteissa, jotka käyttävät kiinteää verkkoa tai ovat liikkuvia langatonta yhteyttä käyttäviä päätelaitteita. Tiedonsiirto saatiin testattua välittämällä liikennetiedotteita muistuttavia sanomia. Siirrettävän tiedon määrä oli vielä tässä vaiheessa vähäistä, eikä järjestelmää kuormitettu kunnolla. Jatkossa järjestelmää tulee testata välittämällä esimerkiksi koko Suomea koskevat liikennetiedotteet Web Services -palvelun kautta ja todentaa järjestelmän toimivuus suurella kuormituksella. Työssä ei myöskään otettu kantaa Web Services -tekniikan tietoturvallisuuteen eikä palvelua käyttävien päätelaitteiden autentikointiin. Nämä asiat tulee ottaa huomioon, kun järjestelmää toteutetaan tuotantoon.

Liikennetiedotejärjestelmän suunnittelun lähtökohtina oli toteuttaa liikennetiedotteiden välittäminen päätelaitteisiin alueittain. Järjestelmästä suunniteltiin kyselypohjainen, jolloin tarvittavat tiedot saadaan haettua päätelaitteeseen silloin, kun niitä tarvitaan. Käyttäjän kannalta katsottuna saatiin suunnittelua järjestelmä, jossa käyttäjä saa haettua vain oleellisimmat tiedot silloin, kun niille on tarve. Näin esimerkiksi etelässä liikkujan ei tarvitse vastaanottaa ja käsitellä pohjoiseen sijoittuvia tietoja. Teknisestä näkökulmasta katsottuna siirrettävän informaation määrää saatiin pudotettua, kun välitettävänä on vain tietyn alueen tiedot koko Suomen sijasta. Tässä vaiheessa suunnittelua Suomi jaettiin kuuteen osaan, joille liikennetiedotteet tarjotaan. Aluejako on tässä vaiheessa varsin riittävä. Suunnitelmaa toteutettaessa tuli esille jatkokehitysajatus, jossa näiden kuuden alueen lisäksi liikennetiedotteet tarjottaisiin myös suuremmille yksittäisille kaupungeille.

Liikennetiedotejärjestelmän määrittely saatiin kattamaan koko toteutettava järjestelmä palvelinpuolelta aina järjestelmää hyödyntävään sovellukseen saakka. Järjestelmää määritettäessä sivuttiin jo hieman teknistä suunnittelua, mikä oli tarpeen käytettävien tekniikoiden testauksen osalta. Testaukseen rakennetut sovellukset tehtiin hyvin pitkälle tuotantoon toteutettavien sovellusten kaltaisiksi. Järjestelmää hyödyntävän sovelluksen suunnitelmasta saatiin luotua kattava, ja sen pohjalta on helppo toteuttaa liikennetiedotejärjestelmän käyttöönotto. Suunnitelma on luonteeltaan yleisluontoinen, jolloin sitä voi hyödyntää myös tulevien sovellusten osalta.

8 YHTEENVETO

Tämän työn tarkoituksena oli laatia suunnitelma liikennetiedotejärjestelmän tiedonsiirron sekä järjestelmää hyödyntävän sovelluksen toteuttamiseksi. Työssä kuvattiin järjestelmien integrointiin liittyviä yksityiskohtia sekä palvelukeskeistä arkkitehtuurimallia, joka on yksi ajattelutapa toteuttaa järjestelmien teknologia- ja alustariippumaton integrointi. Seuraavaksi kuvattiin tiedonsiirtotapoja, joita harkittiin palvelun tiedonsiirron toteuttamiseksi. Tiedonsiirtotapaa valittaessa lähdettiin palvelukeskeiseen arkkitehtuuriin sisältyvistä ajatusmalleista. Tiedonsiirtotavaksi valittiin Web Services, joka osoittautui käteväksi teknologiaksi toteuttaa palvelukeskeistä arkkitehtuuria noudatteleva palvelu. Työssä kuvattiin Web Services -tekniikkaan liittyviä ominaisuuksia sekä tekniikoita, ja tekniikka saatiin testattua välittämällä ensimmäiset sanomat palvelulta asiakassovelluksille. Työssä on esitelty myös suunnitelmat liikennetiedotekomponentin sekä liikennetiedotejärjestelmää hyödyntävän sovelluksen toteuttamiseksi.

Liikennetiedotejärjestelmän suunnittelu toteutettiin yhteistyössä insinööriopiskelija Jukka Korhosen kanssa. Suunnitelmat saatiin laadittua kattaviksi, ja niiden pohjalta sekä liikennetiedotejärjestelmä että sitä käyttävä sovellus tullaan toteuttamaan kevään ja kesän 2009 aikana.

LÄHTEET

1 Destia Trafficin kotisivut. [WWW-dokumentti] <http://www.destiatraffic.fi> (Luettu 14.3.2009)

2 Ebsolut Oy:n kotisivut. [WWW-sivut] <http://www.ebsolut.fi/> (Luettu 7.2.2009)

3 Korhonen, J. Suunnitelma liikennetiedotejärjestelmän perustamiseksi. Kajaani. Kajaanin ammattikorkeakoulu. 2009. Julkaistaan 9.4.2009

4 Tähtinen, Sami. Järjestelmäintegraatio: tarve, vaihtoehdot, toteutus. Jyväskylä. Talentum Media Oy. 2005. 217 s. ISBN 952-14-0854-5

5 Vesterholm, Mika, Kyppö, Jorma. Java-ohjelmointi. 7. painos. Helsinki. Talentum Media Oy. 2008. 606 s. ISBN 978-952-14-1356-8.

6 Olli, Sakari. SOA – ajattelutapa vai teknologia. Tieturi Oy. 2005. [PDF-dokumentti] <http://www.pcuf.fi/sytyke/syysseminaarit/SS_2005/SOA_ajattelutapa_teknologia_Tieturi.pdf> (Luettu 9.2.2009)

7 FCS Sovelto Oyj. SOA-arkkitehtuuri. 2006. [PowerPoint-dokumentti] <http://seminaarit.codezone.fi/video/jyu2006/11/SOA-arkkitehtuuri-ja-WebService-ohjelmointi.ppt> (Luettu 9.2.2009)

8 Nevala, Lauri. Web Services -tekniikoiden hyödyntäminen Java Platform, Micro Edition -ympäristössä. [PDF-dokumentti] <http://urn.fi/URN:NBN:fi:jyu-200810145810> (Luettu 6.2.2009)

9 Nykänen, Ossi. XML. Jyväskylä. Docendo Finland Oy. 2001. ISBN 951-846-096-5

10 Refsnes Data, w3school.com. SOAP Tutorial. [WWW-dokumentti]. <http://www.w3schools.com/soap/default.asp> (Luettu 8.3.2009)

11 Sun Microsystems. The Java Web Services Tutorial. 2006. [WWW-dokumentti] <http://java.sun.com/webservices/docs/2.0/tutorial/doc/> (Luettu 16.3.2009)

12 Eloranta, Tuomo. Destia Traffic: Data Model Description. Destia. 2007. [PDF-dokumentti] <http://www.liikennetieto.com> (Luettu 3.5.2009)

13 OpenMap-kotisivut. [WWW-dokumentti] <http://openmap.bbn.com> (Luettu 29.3.2009)

14 Ebsolut Oy. EMobile Kuljetuksen ajoneuvosovellus. 2009