# LIIKUNTAPAIKKOJEN WEB-POHJAINEN MALLINNUSOHJELMA

Juhana Oukka
2008
Oulun seudun ammattikorkeakoulu

# LIIKUNTAPAIKKOJEN WEB-POHJAINEN MALLINNUSOHJELMA

Juhana Oukka
Opinnäytetyö
12.12.2008
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

OULUN SEUDUN AMMATTIKORKEAKOULU **TIIVISTELMÄ**

| Koulutusohjelma | Opinnäytetyö | Sivuja | + Liitteitä |
|---|---|---|---|
| Tietotekniikan koulutusohjelma | Insinöörityö | 43 | + 0 |

| Suuntautumisvaihtoehto | Aika |
|---|---|
| Sulautetut ohjelmistot | 28.10.2009 |

| Työn tilaaja | Työn Tekijä |
|---|---|
| OAMK | Juhana Oukka |

Työn nimi
Web-pohjainen mallinnusohjelma

Asiasanat
ASP.NET, C#, verkko-ohjelmointi, internetsivut, JavaScript, MySQL, relaatiotietokannat

Insinöörityössä suunniteltiin ja toteutettiin web-pohjainen tilanmallinnustyökalu. Ohjelmalla on tarkoitus mallintaa eri toimiin tarkoitettuja tiloja ja helpottaa tilan käyttöönottoa antamalla valmiita osia, joita sijoittaa haluttuun tilaan. Ohjelmaan voidaan lisätä haluttuja toimipisteitä, esimerkiksi tenniskenttä tai myyntiautomaatti, ja määritellä näiden tarvitsema alue.

Mallinnettava tila voidaan suunnitella suoraan ohjelmalla käyttäen valmiiksi luotuja toimipisteitä ja tilaan varattua kokoa. Työkalulla on tarkoitus tehdä tilasta kartta, jolla helpotetaan tilan ottamista käyttöön. Käyttäjät tallentavat tietokantaan suunnittelemansa tilat sekä mahdolliset omat toimipisteensä. Näitä voidaan uudelleen käyttää useissa muissakin tiloissa sekä muokata myöhempää tarvetta vastaaviksi.

Työ tehtiin Microsoftin Visual Web Developer -työkalulla ja sen tarjoamilla serverioptioilla. Koodi toimii kehitysversiossa suoraan, mutta sen vieminen toimivaan serveriin vaati joitain muutoksia koodissa tapahtuvaan tietokantayhteyteen.

# SISÄLTÖ

# KÄSITTEET

ASP — ASP tulee sanoista Active Server Page. Tällä tarkoitetaan verkkosivua, johon on sisällytetty toiminnallisuuksia, jotka ajetaan serverillä ennen sivun lähettämistä käyttäjän selaimelle.

AJAX — AJAX tulee sanoista Asynchronous JavaScript and XML. AJAX on tekniikka, jota hyödynnetään verkkosivujen toiminnallisuuksien nopeuttamiseen ja käyttäjä kokemuksen parantamiseen.

CLR — Common Language Runtime. Ohjelman koodia ajava ympäristö, joka kääntää ohjelman koodin käyttöjärjestelmän ymmärtämään muotoon ajon aikana.

Disconnected protocol ja tilaton yhteys

Disconnected protocol tarkoittaa yhteyttä, joka ei ole koko ajan auki. Yhteys luodaan palvelimen ja käyttäjän välille vain siksi aikaa, että käyttäjä lähettää pyynnön ja pyyntöön vastataan, jonka jälkeen yhteys katkaistaan. Tämän tarkoitus on tarjota vähän kaistaa ja palvelimen resursseja vievä yhteystyyppi, koska palvelimen ei tarvitse koko ajan pitää yhteyttä auki ja muistaa mitään käyttäjästä. (Esimerkiksi HTTP.)

HTML — Hypertext Markup Language. HTML on kuvauskieli, jolla kuvataan, miltä tiedoston asiasisällön tulisi näyttää.

HTTP — Hypertext Transfer Protocol. HTTP on verkossa tiedon ja tiedostojen siirtoon käytetty protokolla.

Skriptikieli

Skriptikieli on kuin tulkattava ohjelmointikieli, mutta sitä ei käännetä miksikään välikieleksi ennen ajoa, vaan se käännetään suoraan ajon aikana koneen ymmärtämään muotoon (kuten esimerkiksi JavaScript, Python ja PHP). Tällaiseksi lasketaan myös pelkkä jono käyttöjärjestelmän omia komentoja (Bash ja Perl tai Windowsin bat-päätteiset tiedostot).

Tulkattava ohjelmointikieli

Tulkattavalla ohjelmointikielellä tarkoitetaan ohjelmointikieltä, jota ei käännetä binääriksi (konekielelle), mutta se yleensä käännetään joksikin välikieleksi (kuten esimerkiksi Java ja ASP.NET). Välikieli ajetaan tietokoneella olevan ajoympäristön (engl. Runtime Environment) läpi, joka muuntaa kielen binääriksi tietokoneelle.

WYSIWYG (What You See Is What You Get)

Tämän tyyppisellä ohjelmalla voidaan tehdä HTML-sivuja ja muita käyttöliittymäkomponentteja kirjoittamatta koodia suoraan.

# 1 JOHDANTO

Nykyään on paljon erilaisia urheilutiloja ja niiden rakennetta voidaan muuttaa helposti vastaamaan haluttua tarvetta. Useilla urheiluhalleilla ja uimahalleilla on mahdollisuudet muuttaa sisätilojaan vastaamaan erilaisiin urheilutapahtumiin. Esimerkiksi Ouluhallissa voidaan järjestää sulkapallokisoja, salibandykisoja tai yleisurheilukisat. Jokainen näistä tapahtumista vaatii erilaiset järjestelyt sisätiloille. Tästä johtuen jokainen urheilutapahtuma tarvitsee omanlaisensa tilan. Tähän tällä projektilla lähdettiin työstämään ratkaisua.

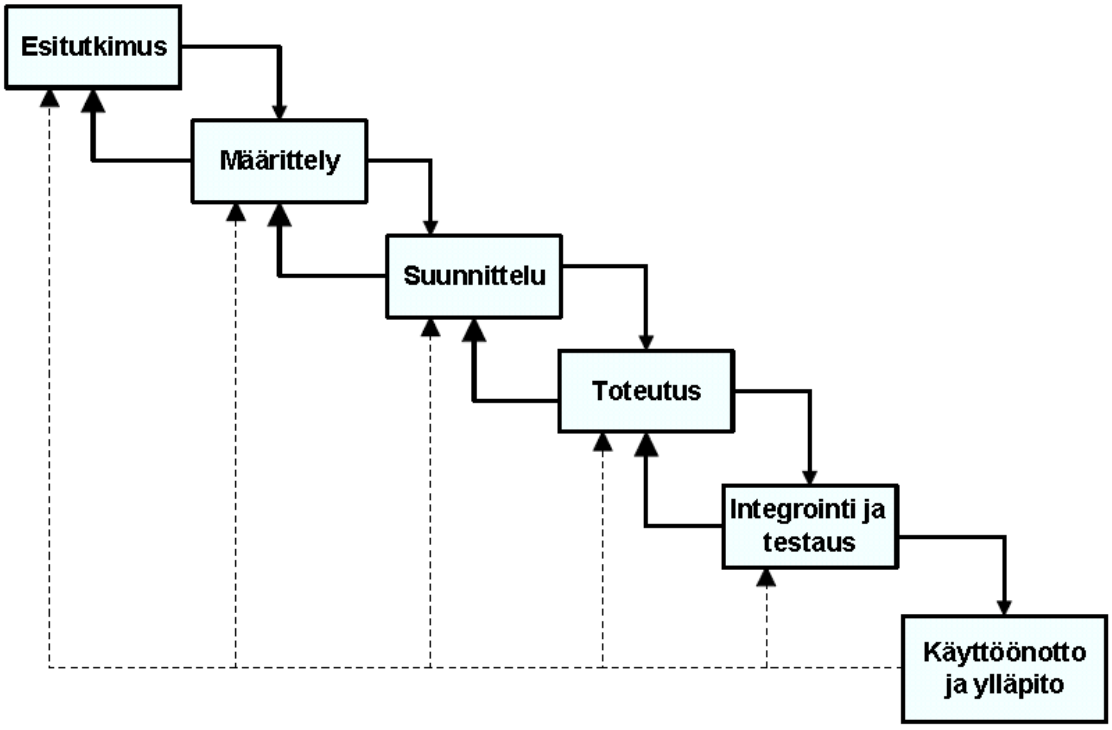
Työn tilaajana on OAMK:n Live-projekti ja Oulun kaupungin liikuntavastaavana toimiva Liikuntasihteeri. Liikuntasihteeri toimenkuvaan kuuluu järjestää Oulun alueella koululiikuntatapahtumia ja -kilpailuja.

Tavoitteena on tehdä tilanmallinnusohjelma, jolla voidaan tehdä karttoja liikuntapaikoista. Käyttäjät tekevät haluamastaan paikasta ensin pohja-piirustuksen ja lisäävät ohjelmaan tarvitsemansa urheilupisteet, kuten esimerkiksi sulkapallokentän, sählykaukalon tai korkeushyppypaikan. Käyttämällä pohjapiirustuksia ja urheilupisteitä saadaan tapahtumalle suunniteltua ja rakennettua sopivat tilat ja samalla kartta tapahtumalle. Käyttäjien tehdessä lisää pohjapiirustuksia ja urheilupisteitä ohjelmaan heidän työtaakkansa vähenee ja muiden käyttäjien ohjelman käyttöönotto helpottuu.

# 2 OHJELMISTON KEHITYSPROSESSI

Ohjelmiston kehitysprosessilla tarkoitetaan ohjelman elinkaarta eli sitä aikaa, kun ohjelmiston kehitys aloitetaan, siihen asti, kunnes se poistetaan käytöstä. Ohjelmiston kehitysprosessin lopputuloksena on järjestelmä, joka täyttää tilaajan toiveet ja odotukset laadittujen aikataulujen ja kustannusarvioiden puitteissa. Vaihejakomallilla, josta näkyy esimerkki kuvassa 1, ohjelmiston elinkaari voidaan jakaa vaiheisiin. (Haikala - Märijärvi 2000, 23 - 26.)

## Vesiputous-malli




*KUVA 1. Vaihejakomalli (Haikala - Märijärvi 2000, 24)*

**Esitutkimuksessa** haetaan, mitkä ovat ratkaistavat ongelmat (asiakkaan toiveet) sekä miten niitä olisi helpointa lähteä ratkaisemaan. Tähän vaiheeseen sisältyy useita haastatteluja asiakkaan kanssa, että saadaan mahdollisimman tarkkaan rajattua ongelma sekä reunaehdot ja asiakasvaatimukset tulevalle ohjelmistolle. (Haikala - Märijärvi 2000, 25 - 26.)

**Määrittelyvaiheessa** analysoidaan saadut asiakasvaatimukset ja tarkennetaan mitkä niistä tullaan toteuttamaan kyseessä olevassa projektissa. Asiakasvaatimuksista tehdään ohjelmiston vaatimuksia. Tähän vaiheeseen myös liittyy paljon haastatteluja asiakkaan kanssa, että toteutustapa ja tulokset vastaavat mahdollisimman hyvin asiakkaan toiveita. Määrittelyvaiheen tuloksena syntyvää dokumentaatiota kutsutaan toiminnalliseksi määrittelyksi. (Haikala - Märijärvi 2000, 26 - 27.)

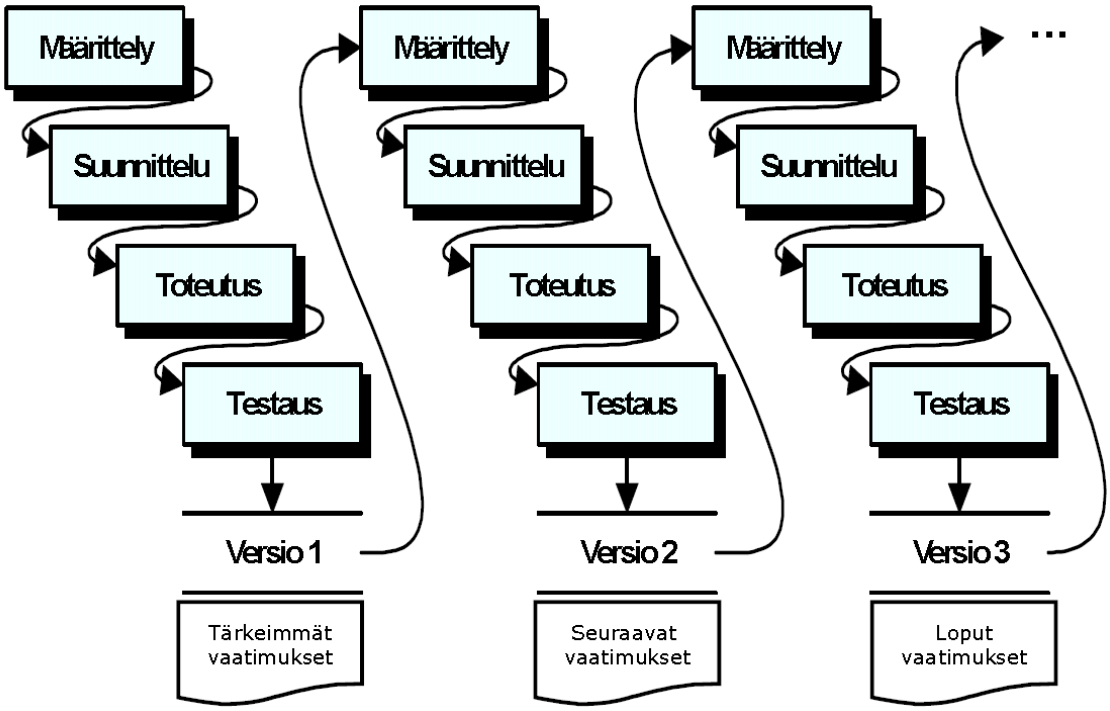
**Suunnitteluvaiheessa** suunnitellaan määrittelyvaiheessa kuvatut toimintojen toteutustavat. Tässä vaiheessa lyödään lukkoon funktioiden nimiä sekä tehdään luokkakaaviot ja luokkien ja funktioiden yhteydet. Työ jaetaan mahdollisimman pieniin osiin, että se on helpommin hallittavissa ja paremmin jaettavissa työryhmän kesken. Myös ohjelman käyttöliittymä suunnitellaan tässä vaiheessa, mutta se tulee muuttumaan toteutuksen ja testauksen aikana käyttäjäpalautteen myötä. Tämä vaihe tuottaa dokumentin nimeltään tekninen määrittely. (Haikala - Märijärvi 2000, 27 - 28.)

**Toteutusvaiheessa** toteutetaan suunnitteluvaiheessa suunnitellut komponentit ja moduulit mahdollisimman tarkasti. Ongelmana voi olla tekijöiden liiallinen improvisointi suunnitelmiin nähden, joka voi hankaloittaa kokonaisuuden yhteen keräämistä vaiheen loppupuolella. Toteutusvaiheesta saadaan mahdollisimman loppuun asti tehty tuote, jota vielä hiotaan testausvaiheessa ennen julkaisua. (Haikala - Märijärvi 2000, 28 - 29.)

**Testausvaiheessa** testataan toteutusvaiheessa luodut osat ja etsitään niistä mahdolliset virheet. Virheet voidaan korjata tässä vaiheessa, mutta jos niitä on löytynyt liikaa voidaan palataan takaisin toteutusvaiheeseen tai jopa suunnitteluvaiheeseen. (Haikala - Märijärvi 2000, 28 - 29.)

Vaihejakomalleja on olemassa useita. Edellinen oli niin sanottu vesiputousmalli, joka on muodossa tai toisessa perustana lähes kaikille muille vaihejakomalleille.

## Evo-malli




*KUVA 2. Evo-malli (Haikala - Märijärvi 2000, 30)*

Evo-mallissa (tunnetaan myös nimellä *iteratiivinen prosessi* tai *inkrementaalinen prosessi*) rakennetaan ensin projektin ydinjärjestelmä, jota lähdetään jatkokehittämään seuraavissa iteraatioissa. Tämä vaihejakomalli koostuu useasta vesiputousmallista, kuten kuvasta 2 voidaan todeta. Niistä jokainen iteraatio kasvattaa järjestelmää. Ensimmäisen iteraation jälkeen järjestelmä pysyy toimivana, koska jokaisessa iteraatiossa rakennetaan jo valmiin perustan päälle. Tämä mahdollistaa jälkeenpäin useamman yhtäaikaisen ominaisuuden rakentamisen järjestelmään, koska jo tehdyt iteraatiot ovat vakaita. (Haikala - Märijärvi 2000, 29 - 32.)

Tässä projektissa käytettiin Evo-mallia, koska Evo-malli sopi projektiin hyvin. Alkuun suunniteltiin erittäin pieni ydinjärjestelmä, joka olisi mahdollisimman nopea rakentaa. Valmis ydinjärjestelmä helpottaa suurelta osin tulevien ominaisuuksien rakentamista ja testaamista. Ohjelma on helpommin

hallittavissa, koska ohjelman osat saadaan rakennettua suoraan jo toimivan järjestelmän päälle. Koska tekijöitä oli vain yksi, haluttiin tällä varmistaa, että tekijä pysyy koko ajan selvillä ohjelman tilanteesta eli siitä, mitkä osat ovat valmiit ja mitkä eivät. Suunnittelu oli vaikeaa, koska tekijällä ei ollut aikaisempaa tuntemusta kaikista projektissa käytössä olleista tekniikoista. Suunnittelu ja ohjelmointi tehtiin, kun oli opittu sopivat tekniikat.

Vaikka tekijällä ei ollut aikaisempaa kokemusta tässä työssä käytetyistä tekniikoista, auttoi hänen aikaisempi kokemuksensa verkko-ohjelmoinnista PHP-kielellä, koska suurin osa verkko-ohjelmoinnista ratkaisee samantyylisiä ongelmia. Tästä johtuen suunnittelu pystyttiin tekemään yllättävän hyvin. Suunnitelmat pidettiin pieninä hallittavina palasina, että tekijän olisi helpompi opiskella ja hallita tietty alue tai tekniikka ja toteuttaa se. Tällä tavoin pystyi ohjelmointikielen ja tekniikoiden idiomien opiskelun pitämään hallittavana ja helposti lähestyttävänä.

# 3 PROJEKTIN TEKNIIKAT

## 3.1 .NET Framework ja ASP.NET

Tässä työssä käytettiin ASP.NET 3.5 -versiota. ASP.NET 3.5 eroaa merkittäviltä osin klassisesta ASP:stä ja näiden koodauskäytännöt eivät vastaa suoraan toisiaan. Uusin ASP.NET toi mukanaan paljon uusia parannuksia edeltäviin ASP.NET-versioihin verrattuna. Klassinen ASP oli Microsoftin ensimmäinen verkko-ohjelmointiin suunniteltu järjestelmä, joka suurilta osin kirjoitettiin vain HTML-sivujen sekaan. ASP.NET on osa Microsoftin kehittämää isompaa .NET-alustaa. Tästä syystä kerrotaan tässä myös .NET Framework -alustasta.

### .NET Framework

.NET alustan tarkoitus on antaa kehittäjille seuraavat edut:

- tarjota yhtenäinen olio-ohjelmointiympäristö riippumatta, onko ohjelma tarkoitettu tietokoneella ajettavaksi, tietokoneella ajettavaksi siten, että ulostulo (output) lähetetään verkon kautta jollekin muulle järjestelmälle, tai ajetaanko ohjelma jossain muualla kuin käytössä olevalla koneella (Executed remotely)
- tarjota koodin ajoympäristö, joka minimoi versio-ongelmat sekä antaa turvallisen tavan ajaa koodia ja parantaa suorituskykyä verrattuna skriptattuun tai tulkattuun kieleen
- tarjota sisäänrakennettuna kaikki viestintästandardit (verkkoviestintään eri protokollien yli ja koneen sisäiseen ohjelmien väliseen viestintään), siten, että .NET:llä ohjelmoitu koodi integroituu muihin järjestelmiin suoraan (.NET Framework Conceptual Overview 2008).

.NET sisältää kaksi pääkomponenttia, jotka ovat yleiskieli-ajoympäristö, tästä lähtien CLR (Common Language Runtime), ja .NET-luokkakirjasto. Kehittäjä voi valita, käytetäänkö CLR:ää vai halutaanko ohjelmoida suoraan käyttöjärjestelmä komponenteille. CLR:ää käyttävää koodia kutsutaan *hallituksi koodiksi (managed code)* ja sitä käyttämätöntä *ei-hallituksi koodiksi (unmanaged code).* Ei-hallitun koodin kehittäjällä on vastuu pitää koodi toimivana. Tällä tarkoitetaan, että mahdollisten käyttöjärjestelmä päivityksien ja muutoksien jälkeen ohjelman tulisi edelleen toimia. CLR tarjoaa kehittäjälle helpottavia ominaisuuksia, kuten muistinhallinnan ja säikeidenhallinnan, ja pakottaa tyyppiturvalliset muuttujat. Luokkakirjasto tarjoaa paljon ohjelmointia helpottavia uusiokäyttöisiä funktioita ja olioita, joilla voidaan rakentaa sekä verkkopohjaisia että Windows-pohjaisia ohjelmia. (.NET Framework Conceptual Overview.)

.NET-alusta on pyritty tekemään mahdollisimman käyttöjärjestelmä- ja ohjelmointikieliriippumattomaksi, eli se tukee useampaa käyttöjärjestelmää ja ohjelmointikieltä. Tämä riippumattomuus saadaan juuri CLR:llä. Kaikki .NET-alustalla kirjoitetusta koodista käännetään ensin tulkattavaksi yleiskieleksi (Common Intermediate Language, CIL), jonka CLR sitten kääntää binääriksi koodin ajon aikana. Toisin kuin joissakin tulkattavissa kielissä .NET-alustan CLR pitää myös käännetyt (tulkatut) binäärit muistissa, jotta niiden käyttö olisi nopeampaa. (.NET Framework Conceptual Overview.)

## ASP.NET

ASP.NET on Microsoftin jatkokehittämä versio klassisesta ASP:sta. ASP.NET on tämän kirjoituksen hetkellä versiossa 3.5, jolla tämä työ on tehty. ASP.NET on yksi osa Microsoftin kehittämää .NET-alustaa. ASP on lyhenne sanoista ”Active Server Pages”.

ASP.NET on verkkokehitykseen suunniteltu alusta, joka käyttää Microsoftin .NET Frameworkiä. ASP.NET -toiminnot sisältävät palvelimella pyörivää tulkattavaa ohjelmakoodia, jonka tulkkauksen hoitaa .NET Frameworkin CLR, ja käyttäjän selaimelle lähetettäviä skriptejä. CLR:n ansiosta ASP.NET-ohjelmointia voi tehdä Microsoft Visual Basic-, C#-, JScript-, .NET-, ja J#-ohjelmointikielillä. Näillä ohjelmointikielillä on tarkoitus tehdä tapahtuman käsittelyt ja toiminnot palvelimella pyörivään ohjelmaan. Ohjelmoija voi silti kirjoittaa itse suoraa HTML:ää, joka tulee käyttäjän näkyviin, mutta ASP.NET tarjoaa siihen myös omat työkalut. (ASP.NET Overview.)

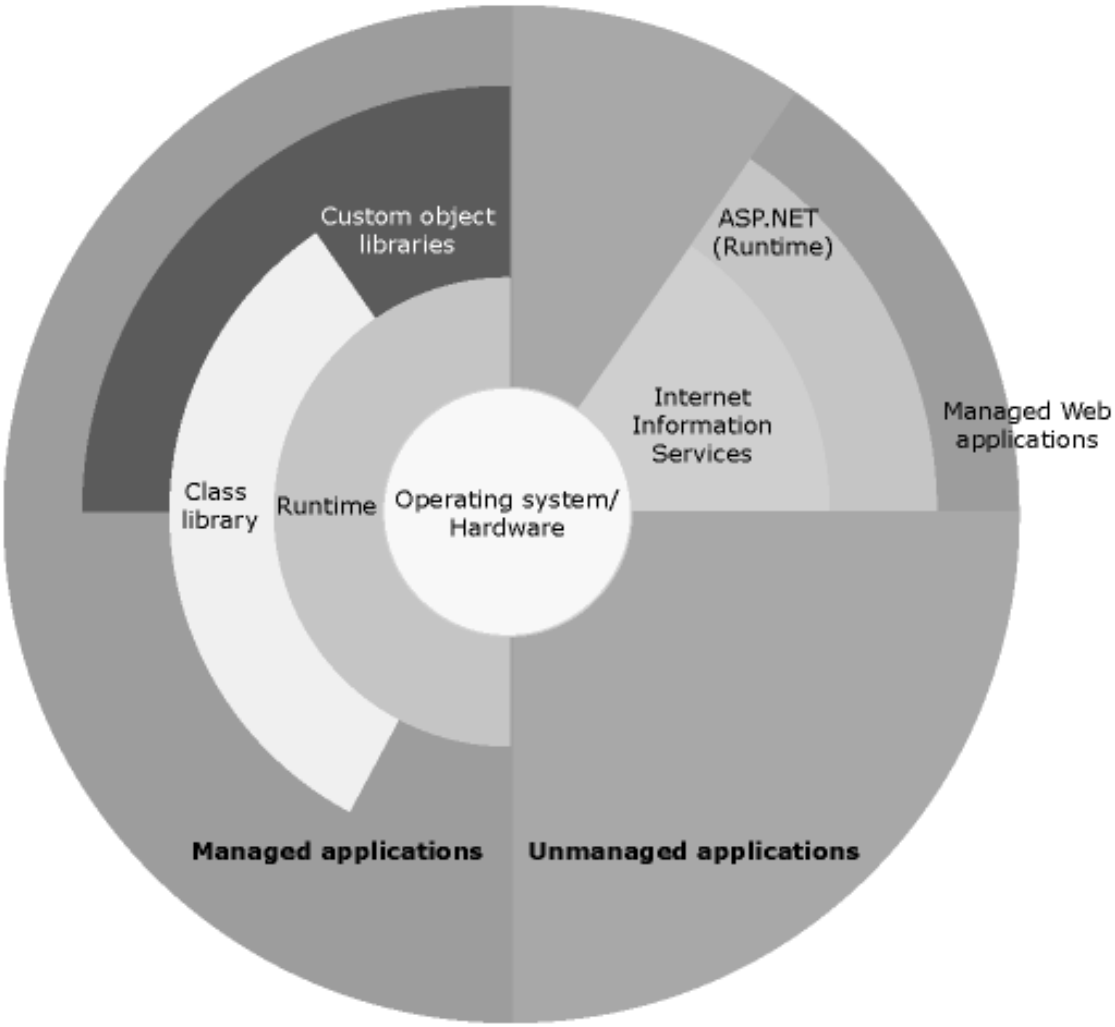
ASP.NET käyttää verkon yli HTTP (Hypertext Transfer Protocol) -yhteyttä, joka on niin kutsuttu tilaton yhteys (disconnected protocol). Yhteystyypin johdosta käyttäjän tilan kanssa tulee olla erittäin tarkka. HTTP toimii pelkkänä viestikanavana verkkojen yli. ASP.NET tarjoaa tähän valmiita ratkaisuja, jotka helpottavat ohjelmien ja järjestelmien kehittämistä HTTP:n päälle. Näitä ovat sivunhallinta- ja tilanhallintajärjestelmä, jotka tarjoavat valmiita osia, joilla pidetään palvelimella olevan järjestelmän tiedossa. mitä käyttäjä on tehnyt ja mitä käskyjä hän antaa tai on antanut. (ASP.NET Overview.)

ASP.NET käyttää JavaScriptiä ottamaan vastaan käyttäjän tekemiä muutoksia ja tapahtumia sivuilla tai käyttäjän selaimen tukemaa versiota siitä. Tätä voidaan edelleen tehostaa käyttämällä ASP.NET AJAX -luokkakirjastoa, joka tuo mukanaan mahdollisuuden osittaiselle sivupäivitykselle (partial page update). Osittaisella sivupäivityksellä on tarkoitus vähentää turhaa liikennettä käyttäjän ja palvelimen välillä ja luoda enemmän työpöytäohjelmaa vastaava kokemus käyttäjälle, kun jokaisen muutoksen yhteydessä ei sivu ”välähdä”, kun se päivitetään.

ASP.NET luo käyttäjälle näkyvän ohjelman, tai sivun, ulkoasun HTML:llä (Hypertext Markup Language). Koska ohjelman käyttöliittymä on tehty HTML:llä, sen käyttämiseen kelpaa mikä tahansa internet-selain. ASP.NET tarjoaa HTML-näkymän tekemiseen valmiita työkaluja, jotka sisältävät valmiita osia tiedon

näyttämiseen käyttäjälle sekä käyttäjän syötteiden käsittelyyn. Käsittelyt ja tapahtumat voidaan koodata millä tahansa ASP.NET:n CLR:n tukemista ohjelmointikielistä. (ASP.NET Overview.)

Kuvassa 3 näkyvät ASP.NET:n sijoittuminen .NET Frameworkissä sekä hallitun ja ei-hallitun ohjelmakoodin erot.




*KUVA 3. .NET Framework (.NET Framework Conceptual Overview)*

## 3.2 JavaScript

JavaScript oli alun perin Netscapen 1990-luvulla kehittämä niin kutsuttu tulkattava skriptikieli. Vuonna 1995 Netscape julkaisi Netscape Navigatorin, joka oli ensimmäisiä JavaScriptiä tukevia selaimia. Microsoft teki tästä myös oman versionsa omiin selaimiinsa vuonna 1996 ja kutsui sitä JScriptiksi. Kummatkin skriptikielet olivat lähes identtisiä. Myöhemmin vuonna 1996 Netscape esitti JavaScriptin standardoitavaksi Ecma Internation:lle, joka sai siitä ensimmäisen version julkaistua kesäkuussa vuonna 1997, joka nimettiin ECMAScriptiksi. JavaScriptiä tai JScriptiä, joka toteuttaa Ecma Internationaalin tekemän standardin, kutsutaan ECMAScript-yhteensopivaksi. (JavaScript Mozilla.)

JavaScript, kuten ASP.NET, on tulkattava ohjelmointikieli, mutta toisin kuin ASP.NET se on tarkoitus ajaa käyttäjän koneella, tarkemmin käyttäjän selaimessa, ei palvelimella. Jokaisessa nykyisessä internet-selaimessa on sisällytetty jokin JavaScript- tai JScript-tulkki. Käyttäjän selaimen tulkki kääntää koodin, kun käyttäjä saapuu sivuille ja selain saa koodin. Koska selainkohtaisia tulkkieroja on, yleensä ensimmäisessä JavaScript-osiossa tarkastetaan, mikä selain käyttäjällä on käytössä ja otetaan huomioon mahdolliset eroavaisuudet koodissa tämän pohjalta. (JavaScript Mozilla.)

JavaScript on suunniteltu oliopohjaiseksi ohjelmointikieleksi. Tästä syystä se on helposti omaksuttava, jos on kokemusta muista oliopohjaisista ohjelmointikielistä. Täten JavaScript tukee ohjelmointityyliltään normaalia tapahtuma-, luokka- ja funktiotyyppistä-ohjelmointitapaa. (JavaScript Mozilla.)

JavaScript kirjoitetaan joko suoraan itse HTML-tiedostoon script-elementtien väliin tai omaan, yleensä .js-päätteiseen tiedostoon, josta se voidaan sisällyttää HTML-sivuille script-elementin sisään. Script-elementit HTML:ssä myös vaativat kyseessä olevan skriptikielen tiedon, joka kirjoitetaan language-tietueeseen script-elementin sisällä. (JavaScript Mozilla.)

Huomaa alla olevassa esimerkissä käytössä olevat HTML-kommentti merkinnät, joilla koodi piilotetaan pois näkyvistä selaimilta, jotka eivät käytä tai tunnista JavaScriptiä. Esimerkki suoraan HTML-tiedostoon kirjoitetusta JavaScriptistä:

```
<script language=”JavaScript”>
<!-- piilotetaan koodi
     itse koodi tulee tähän
     ...
piilotus päättyy -->
</script>
```

Esimerkki jossa scriptin koodi tuodaan tiedostosta sivulle:

```
<script language=”JavaScript” src=”kansio/JavaScripttiedosto.js” >
</script>
```

JavaScriptillä on tarkoitus luoda käyttäjälle saumaton selauskokemus, jota on vaikea saada aikaan pelkällä HTML:llä, sekä tarjota kokemus, joka vastaisi käytöltään työpöytäsovellusta. JavaScriptiä voidaan myös käyttää vähentämään palvelimen ja käyttäjän välistä liikennettä, esimerkiksi tarkistamalla käyttäjän antamat syötteet ennen niiden lähettämistä palvelimelle, jossa ne myös varmennetaan tietoturvan ja käytännön syistä. (JavaScript Mozilla.)

JavaScript on käytössä tapahtumapohjaisesti (engl. event based) suurimmassa osassa internetsivustoja, joissa se sidotaan osaksi jotain tapahtumaa, joka laukaisee halutun skriptin. Nämä tapahtumat on sidottu W3-säätiön HTML standardeihin, jotka tukevat eri elementeissä olevia tapahtumia, mutta eivät itsessään tee mitään. (JavaScript Mozilla.)

Esimerkki tapahtuman laukaisemisesta HTML-sivulla, kun syötekenttä tulee aktiiviseksi:

```
<input type=”text” onfocus=”JavaScriptFunctio()” id=”IDnimi” />
```

## 3.3 Microsoft SQL Server

Opinnäytetyössä oli käytössä Microsoftin SQL Server 2008. Microsoftin SQL Server 2008 on Microsoftin kehittämä SQL-tietokantaserverijärjestelmä, joka on suunnattu yrityksille. Palvelinohjelmisto käyttää Microsoftin omaa versiota SQL:stä (Structured Query Language), jota kutsutaan MS-SQL:ksi. Kaikki SQL-kielet perustuvat IBM:n kehittämään SQL-standardiin, mutta alustasta riippuen kielissä voi silti olla joitain alustalle ominaisia funktioita ja kirjoitusasuja. (SQL Server 2008 Overview.)

Ensimmäinen Microsoftin SQL Server -palvelinohjelmisto kehitettiin vuonna 1989 Microsoftin ja Sybasen yhteistyönä. Tämän jälkeen vuoteen 1994 asti Microsoftin SQL Server pohjautui Sybasen kanssa kehitettyyn ohjelmistoon. Vuonna 1995 Microsoft julkaisi uuden palvelinohjelmistonsa, jonka koodi oli kirjoitettu puhtaalta pöydältä. Sen jälkeen Microsoft on itse jatkanut palvelinohjelmiston koodin ja sen sisältämien tietokantojen hallinnoimiseen tarkoitettujen työkalujen kehittämistä ja parantamista. (SQL Server 2008 Overview.)

Microsoft SQL Server käyttää tietokantatyyppinä relaatiomallista tietokantaa. Palvelinohjelmisto pystyy tekemään tietokannan hakutuloksista ja tallennuksista XML (Extensible Markup Language) -tyyppisiä muuttujia. (SQL Server 2008 Overview.)

## 3.4 Visual Web Developer

Visual Web Developer on osa Microsoftin kehittämää Visual Studiota, joka sisältää lähes kaikki työkalut Microsoftin eri tekniikoihin ja ohjelmointikieliin. Visual Web Developer, nimensä mukaan, sisältää ainoastaan Microsoftin verkkokehitykseen tehdyt työkalut. Näihin lukeutuvat ASP.NET, ADO.NET, eri ohjelmointikielien kääntäjät CIL:ksi, IDE (engl. Interface Designing

Environment), johon kuuluu WYSIWYG-editori ja tekstieditori, sekä kehittäjäversiot tietokantaserveristä ja verkkoserveristä. (Spaanjaars 2000, 2 - 62.)

Kehittäjäversiot servereistä on tehty hoitamaan pienempää käyttöastetta kuin niiden kokoversiot. Kehittäjäversiot kuitenkin vastaavat toiminnaltaan täysin oikeita järjestelmiä, joten yhteensopivuusongelmia ei toteutusvaiheessa tule. Kehittäjäversiot on suunniteltu viemään mahdollisimman vähän resursseja koneelta, koska serveri pyörii kehittäjän omalla koneella. (Spaanjaars 2008, 5 - 14.)

Visual Web Developer sisältää suurimman osan niistä eduista, joita Visual Studiossakin on. Näihin lukeutuvat tärkeimpinä projektin hallintatyökalut sekä Microsoft IntelliSense. (Spaanjaars 2008, 5 - 14.)

Projektinhallintatyökaluissa Visual Web Develepor ei välitä siitä, missä koodi sijaitsee, sillä normaalin tiedostojen hallinnan lisäksi on ohjelmaan laitettu mukaan tuki FTP:lle sekä IIS:n virtuaalisille hakemistoille. Tämä tarvitsee vain kerran tehdä projektin luontivaiheessa kuntoon, minkä jälkeen ne ovat koko ajan käytössä Visual Web Developerin tiedostojen ja projektien hallintatyökaluissa. Koodin tallentaminen hoituu myös automaattisesti normaalilla tallennusoperaatiolla, joka osaa tulkita, mikä protokolla tiedostojen hallinnassa on käytössä, ja toimia sen mukaan. (Spaanjaars 2008, 33 - 46.)

Toisena erittäin tärkeänä työkaluna on Microsoft IntelliSense (tästä lähtien vain IntelliSense). IntelliSense tarjoaa kehittäjälle nopean tavan selata kirjoitettujen olioiden, funktioiden, muuttujien ja metodien nimiä koodissa. IntelliSense osaa katsoa, mitkä kirjastot kehittäjällä on tuotuna projektiin, ja osaa tarjota kyseisien kirjastojen tarjoamat toiminnot suoraan. IntelliSense osaa myös katsoa käyttäjän omasta koodista olioiden, muuttujien ja funktioiden nimiä sekä pitää kirjaa niiden ulottuvuudesta (engl. scope) ja tarjota näitä kehittäjälle. IntelliSense tulee esille, kun kirjoittaa ensimmäisen muutaman kirjaimen tai

merkin koodia. IntelliSense tarjoaa ensimmäiseksi valinnaksi yleisimmin tai viimeksi käytettyä vaihtoehtoa ja tuo esille aakkosellisen hakemiston muista valinnoista. IntelliSense näyttää myös funktioiden ja olioiden parametrit eli sen, mitä ne ottavat sisäänsä ja palauttavat. Muuttujista IntelliSense kertoo niiden tyypin. Kaikista edellä mainituista tulee esille myös lyhyt kuvaus ja käyttötarkoitus. (Spaanjaars 2008, 50 - 51.)

# 4 TIETOKANNAT

## 4.1 Tietokantojen tyypit

Nykyisissä tietokantojen hallintajärjestelmissä käytetään kahden tyyppisiä tietokantoja, jotka ovat käyttötietokantoja ja analyyttisiä tietokantoja. Tietokantatyyppejä ei tule sekoittaa tietokantamalleihin; näistä kerrotaan enemmän Varhaiset tietokantamallit -luvussa. (Hernandez 2000, 3 - 4.)

Käyttötietokantoja käytetään pääasiassa, kun pitää kerätä, säilyttää ja muokata tietoa. Tämän tyyppiseen tietokantaan tallennetaan dynaamista dataa eli dataa, joka muuttuu jatkuvasti ja heijastaa ajan tasalla olevia tietoja. Esimerkkeinä ovat inventaariot, kuten kirjastoilla lainakirjojen tilanne. (Hernandez 2000, 3 - 4.)

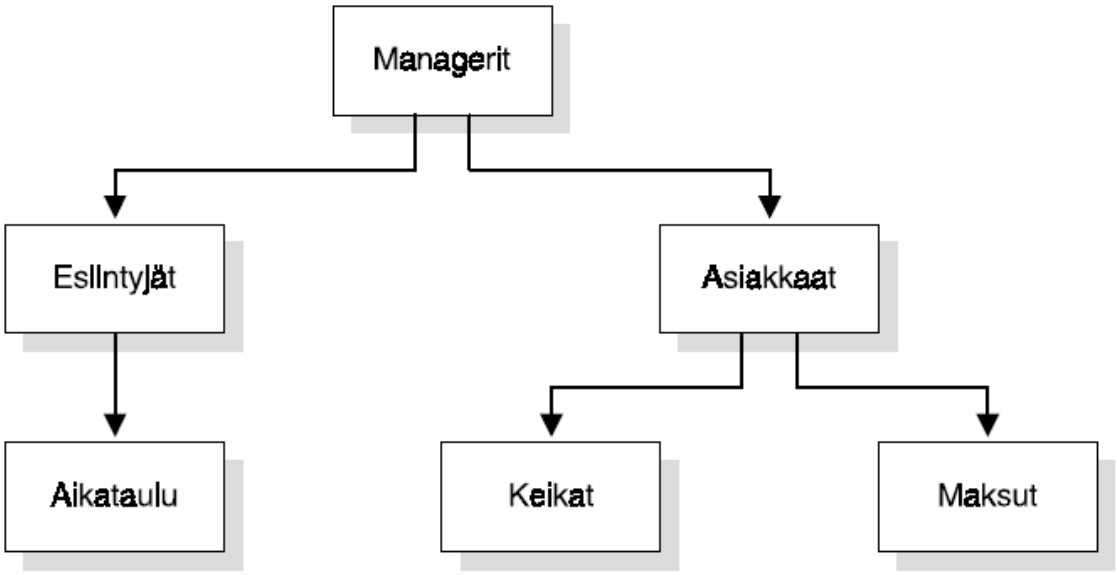
Analyyttisiä tietokantoja käytetään keräämään ajasta ja historiasta riippuvaista tietoa. Näitä tietoja seuraamalla tehdään pitkäaikaisia suunnitelmia ja seurataan kehitystä. Analyyttisten tietokantojen tiedot ovat staattisia eli ne eivät koskaan muutu (tai ainakaan usein) ja tietokanta koskee tiettyä ajankohtaa. Esimerkkinä ovat kyselytutkimus- ja myyntitietokannat. (Hernandez 2000, 3 - 4.)

## 4.2 Varhaiset tietokantamallit

Varhaisia tietokantamalleja oli kaksi, hierarkkinen ja verkkotietokantamalli. Kummankin suurimmat ongelmat olivat tietokannan jäykkyys ja saman tiedon useampaan kertaan tallennus eri paikkaan. (Hernandez 2000, 4 - 11.)

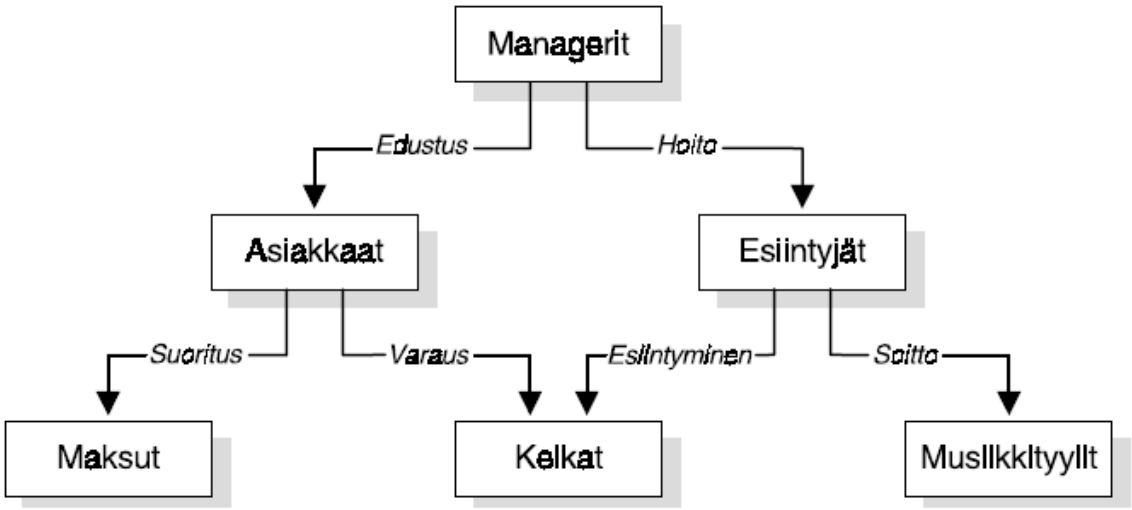
Hierarkkisessa tietokantamallissa, esimerkki näkyy kuvassa 4, jokaisella taululla pystyi olemaan vain yksi isätaulu, mutta isällä useampi lapsitaulu. Taulujen

suhteet olivat aina yhden suhde moneen tai yhden suhde yhteen. (Hernandez 2000, 4 - 8.)



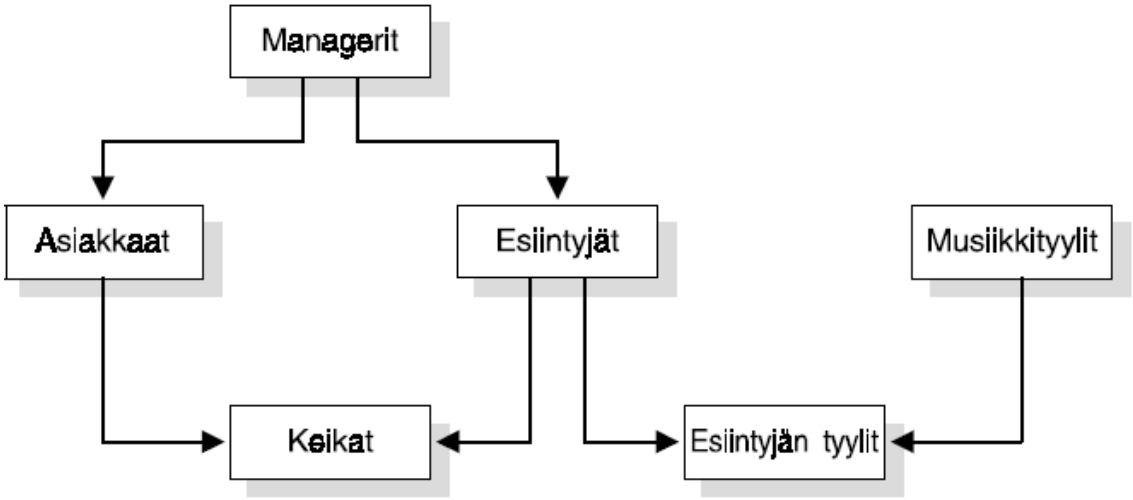

*KUVA 4. Hierarkkinen tietokantamalli (Hernandez 2000, 5.)*

Verkkotietokantamalli, esimerkki näkyy kuvassa 5, kehitettiin korjaamaan juuri tätä ongelmaa. Verkkotietokantamallin suunnittelussa oli tarkoituksena vähentämään saman tiedon useampaan kertaan tallentamista. Verkkotietokantamallissa yhdellä taululla saattoi olla useampi isätaulu. (Hernandez 2000, 8 - 11)




*KUVA 5. Verkkotietokantamalli (Hernandez 2000, 9.)*

1960-luvun loppupuolella matemaatikko E. F. Codd, joka toimi tutkijana IBM:llä, alkoi kehittää relaatiotietokantoja ja julkaisi kesäkuussa 1970 työnsä nimeltä ”A Relation Model of Data for Large Shared Databanks”. Terminä relaatio tuleekin matematiikan joukkoteoriasta eikä niinkään siitä, että tietokannassa tauluilla on ”suhteita”. Kuvassa 6 näkyy, millainen relaatiotietokantamalli on. (Hernandez 2000, 11 - 17.)




*KUVA 6. Relaatiotietokantamalli (Hernandez 2000, 13.)*

## 4.3 Relaatiotietokannat

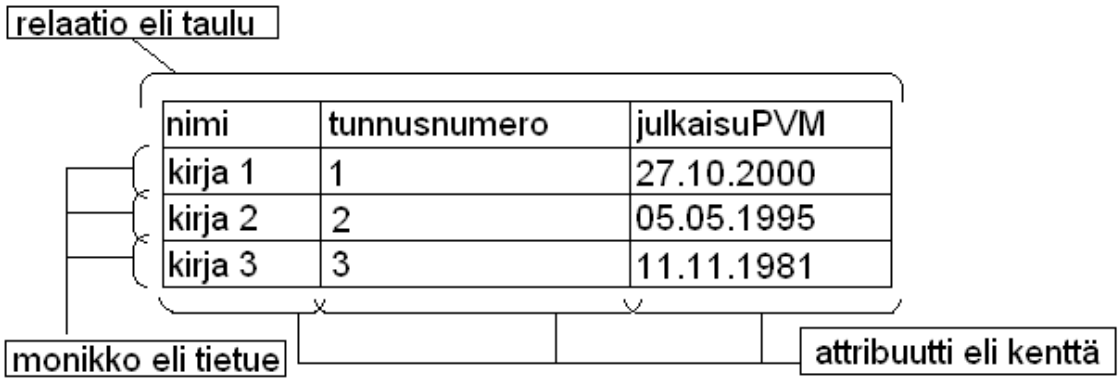
Nykyään lähes kaikki tietokannat ovat relaatiotietokantoja, koska ne ovat hyvin muokkautuvia ja erittäin virheitä kestäviä. Relaatiotietokantojen etuihin lukeutuvat seuraavat:

- sisäänrakennettu monitasoinen eheys
- datan looginen ja fyysinen riippumattomuus tietokantasovelluksista
- taattu tietojen yhtäpitävyys ja oikeellisuus
- tietojen haun helppous.

(Hernandez 2000, 11 - 20.)

### 4.3.1 Terminologiaa

Relaatiotietokantamallissa tiedot tallennetaan *relaatioina*, jotka käyttäjä näkee *tauluina*. Jokainen relaatio muodostuu *monikosta* eli *tietueista* ja *attribuuteista* eli *kentistä.* (Hernandez 2000, 12.)




*KUVA 7. Relaatiotietokannan termit*

Relaatiotietokannassa yhteydet toisiin tauluihin voivat olla jokin seuraavista: yhdestä yhteen, yhdestä moneen ja monesta moneen. Taulujen välinen yhteys luodaan epäsuorasti asettamalla niiden kesken jaetun kentän arvot samoiksi. (Hernandez 2000, 12 - 17.)

Avaimet ovat taulun erikoiskenttiä, jotka palvelevat juuri tätä tarkoitusta. Pääavain on kenttä, jonka avulla taulussa oleva tietue tunnistetaan yksiselitteisesti eli sen sisältämä arvo on uniikki kyseisen taulun sisällä. Viiteavain on kenttä, jonka avulla kahden taulun välille luodaan yhteys. Viiteavaimeen tallennetaan arvo, joka vastaa jonkin muun taulun halutun kentän arvoa. Yleisessä käytännössä on, että isäntätaulussa oleva pääavain on lapsitaulussa samaa tyyppiä oleva viiteavain. (Hernandez 2000, 12 - 17.)

### 4.3.2 Relaatiotietokannan hallintajärjestelmät

Relaatiotietokannan hallintajärjestelmä (engl. RDBMS, Relational Database Management System) on ohjelma, jolla luodaan, ylläpidetään, muokataan ja käsitellään relaatiotietokantaa ja sen sisältämää dataa. Yleisesti puhekielessä, kun puhutaan relaatiotietokannasta, tarkoitetaan sillä relaatiotietokannan hallintajärjestelmää. Tällaisia ohjelmia ovat esimerkiksi Oracle, Microsoft Access ja MySQL. Jotkin hallintajärjestelmät tarjoavat graafisen käyttöliittymän tietokannan rakentamiseen ja hallinnointiin sekä mahdollisesti muitakin lisätyökaluja. Suurimmassa osassa hallintajärjestelmiä on silti edelleen komentorivipohjainen konsoli, jonka kautta voidaan suoraan kirjoittaa SQL-komentoja järjestelmälle. (Hernandez 2000, 17 - 19.)

### 4.3.3 Kyselykieli

Lähes kaikissa nykyisissä relaatiotietokannan hallintajärjestelmissä käytetään kyselykielenä SQL:ää (Structured Query Language). SQL oli alunperin IBM:n kehittämä kyselykieli, nimeltä SEQUEL, jolla voitiin hallita ja käyttää sen omia tietokantoja. SQL sai ANSI-standardin (American National Standards Institute) vuonna 1986 ja ISO-standardin (International Organization for Standardization) vuonna 1987. (Stephens 2001, 4 - 6.)

Kyselykielen tarkoituksena on antaa relaatiotietokannan hallintajärjestelmälle tapa hallinnoida tietokantaa. Riippuen relaatiotietokannan hallintajärjestelmästä voi kehittäjä kirjoittaa kyselykielen komentoja suoraan komentokehotteeseen, sillä lähes jokaisessa hallintajärjestelmässä on tämä ominaisuus, tai antaa graafisesti hallintajärjestelmän kautta haluttuja komentoja. (Stephens 2001, 4 - 6.)

Esimerkki SQL hakulauseesta:

```
SELECT taulu1.sarake1, taulu2.sarake1, sarake2 FROM taulu1, taulu2
WHERE taulu1.liitos = taulu2.liitos;
```

# 5 OPINNÄYTETYÖPROJEKTI

## 5.1 Tausta ja lähtökohdat

LIVE-hankekokonaisuuteen kuuluu neljä erillistä rinnakkaishanketta, joiden koordinaattoreita ovat Oulun seudun ammattikorkeakoulu, SunCom Systems Oy, Healthy Exersice Holding Oy ja Jyväskylän ammattikorkeakoulu. Muita hankkeeseen osallistuvia tahoja ovat muun muassa Oulun kaupungin liikuntavirasto, Jyväskylän seudun kehittämisyhtiö JYKES, Polar Elektro Oy ja Virpiniemen liikuntaopisto. Aiemmin Oulun seudun ammattikorkeakoulun osana hankkeessa on ollut toteuttaa LIVE Liikuntaympäristöjen geneerinen toiminnan hallintajärjestelmä.

Tämä projekti sai alkunsa eräässä LIVE-hankkeen palaverissa esille tulleesta ongelmasta. Työtä lähdettiin tekemään palaverissa syntyneistä ideoista sekä myöhemmistä jatkokehitysideoista.

ASP.NET valittiin ohjelmointialustaksi, koska siinä oli paljon valmiita komponentteja, jotka vähentävät koodaustarpeita useasti tapahtuville tarpeille, kuten tietojen esille tuomista tietokannasta ja tietokannan tietojen muokkausta.

## 5.2 Toteutussuunnitelma

Projekti aloitettiin 1.9.2008 ja sen sovittiin päättyvän 28.2.2009. Projektin aikana kalenteriviikkoja oli 26, johon lukeutuu useampi lomaviikko. Jokaiselle työviikolle merkattiin käytettäväksi 3 päivää opinnäytetyön tekemiseen, josta tulee 50 päivää eli 400 tuntia. Projektisuunnitelmasta poikettiin tekemällä projekti Evo-mallin mukaan.

## 5.3 Työvälineet

Tässä työssä käytettiin taulukossa 1 esitettyjä työkaluja. Lisenssien haltijana on OAMK.

*TAULUKKO 1: Työkalut*

| Kehitysympäristön ohjelmistot | Omistaja |
|---|---|
| Microsoft Vista x86 Business SP1 | OAMK |
| Microsoft .NET Framework 3.5 SP1 | OAMK |
| Microsoft Visual Web Developer 2008 Express SP1 | OAMK |
| Microsoft SQL Server 2008 Express SP1 | OAMK |

# 6 PROJEKTIN TOTEUTUS

## 6.1 Aloitusvaihe

Aloitusvaiheeseen kuului palavereita, joissa pohdittiin ohjelmaa ja mitä sillä pitäisi pystyä tekemään sekä rajattiin ja yleistettiin ohjelmaa samalla. Projektin toteutus tehtiin Evo-vaihejakomallin mukaan, koska kehittäjiä oli vain yksi. Tämä helpotti alkuun perusasioiden opiskelua ja implementointia ohjelmaan. Näiden päälle sitten voitiin rakentaa järjestelmää.

### 6.1.1 Ohjelmointityökalujen ja ohjelmointikielen valinta

Ohjelmointikieleksi ja ympäristöksi valittiin Microsoftin kehittämä ASP.NET ja C#. ASP.NET valittiin ohjelmointiympäristöksi sen sisältämien valmiiden komponentin takia. Komponentit sisältävät paljon automatiikkaa, joka helpottaa useasti toistuvien asioiden toteuttamista järjestelmässä. Näistä päällimmäisinä olivat AJAX-komponentti, tietokantaan yhteyden ottava SQLDataSource-komponentti ja tietokannan tietojen esittämiseen keskittyvät ”view”-komponentit (Detailsview, Gridview, Dataview muutamana esimerkkinä).

Ohjelmointiin sekä tietokannan tekemiseen käytettiin Microsoftin Visual Web Developer 2008 Express Edition -ohjelmaa. Microsoftin Visual-ohjelmasarja on jo vakioitunut Microsoftin Windows-ohjelmointiin sekä Microsoftin kehittämien tekniikoiden ohjelmointiympäristöksi

### 6.1.2 Projektisuunnitelman laatiminen

Vaikka ASP.NET olikin toteuttajalle uusi tekniikka, auttoi aikaisempi kokemus PHP:llä ja SQL:llä tehdystä verkkopohjaisesta ohjelmistosta tämän työn tekemistä ja suunnittelua. Alustavista palavereista saadulla vaatimuslistalla voitiin suunnitella, mitkä toiminnallisuudet tultaisiin tekemään opinnäytetyössä.

Projektisuunnitelma sisältää opinnäytetyölle suunnitellut työtehtävät jaoteltuna mahdollisimman pieniin osiin. Projektisuunnitelman suunnitelmat oli tehty liian positiivisesti ajan suhteen. Suunnitelmassa ei ollut otettu huomioon tarpeeksi hyvin, kuinka kauan uuden ohjelmointikielen ja ympäristöön tutustumiseen ja opiskeluun voi mennä. Myös webohjelmien rajoitukset HTML:llä ja JavaScriptillä tulivat tekijälle vastaan.

### 6.1.3 Työympäristön asennus ja konfigurointi

Microsoftin Visual Web Developer -asennuspaketti sisältää kaiken, mitä työn aloittamiseen tarvittiin. Konfiguroinnilta vältyttiin, koska asennuspaketti osasi hoitaa kaiken tarvitsemansa itse. Tämä yllätti positiivisesti, koska yleensä niin kutsutut Development kitit vaativat paljon konfigurointia ja tiedostojen muokkausta toimiakseen. Huomattavaa on kuitenkin, että asennuspaketti ei sisällä loppujärjestelmässä ajettavia ohjelmia ja resursseja vaan näistä erikseen tehdyt kehitysversiot, jotka on suunniteltu kevyemmiksi, mutta jotka samalla antavat enemmän palautetta kehittäjälle. Tämän palautteen ansiosta kehittäjien on helpompi löytää tekemiensä ohjelmistojen ja järjestelmien ongelmakohdat ja virheet koodissa ennen niiden julkaisua.

### 6.1.4 Työympäristöön tutustuminen

Työympäristöön tutustuminen hoitui ongelmitta, koska Microsoftin Visual Studio sisältää lähestulkoon identtiset käyttöliittymät riippumatta ohjelmointikielestä ja kehitettävästä ympäristöstä. Kehittäjä oli aikaisemmin käyttänyt vain Visual

Studion tekstieditoria koodin kirjoittamiseen, joten WYSIWYG-editoriin sekä toolbox- ja properties-valikoihin tutustumiseen meni jonkin aikaa.

### 6.1.5 Ohjelmointikielen opiskelua

Ohjelmointikielenä oli C#, johon tekijä oli tutustunut aikaisemmin Windows-ohjelmoinnin yhteydessä. ASP.NET myös toi omat idiominsa HTML-sivujen koodaamiseen.

Opinnäytetyön alkuvaiheessa etsittiin hyvää opiskelumateriaalia, jota voitaisiin käyttää suoraan lopputuotteessa. Opiskelu aloitettiin Microsoftin sivuilla olevista opetusvideoista, joissa selvitetään selkeästi ja käytännön kautta kuinka komponentit toimivat. Microsoftin sivujen opetusvideo valikoimassa oli hyvät kuvaukset videoista. Kuvauksissa kerrottiin, mistä komponenteista videossa on kyse ja mitä se tekee ASP.NET ympäristössä. Tämä helpotti videoiden valintaa opiskelussa.

Suurin ongelma ohjelmointikielen opiskelussa oli kehittäjän kokemus PHP:stä, jossa sai, tai joutui riippuen näkökannasta, itse kirjoittaa kaiken HTML-koodauksen ja tietojen esittämiseen. ASP.NET sitä vastoin tarjoaa WYSIWYG-editorinsa ja luokkakirjastojensa kautta omat erikoiset tapansa tuoda taulukoita ja tiedot tietokannasta esille.

## 6.2 Määrittelyvaihe

### 6.2.1 Toiminnallisten vaatimusten lista

Alkuun tehtiin toiminnallisista vaatimuksista lista, jonka tiedot kerättiin palavereissa, joissa oli alkuun esitetty tarve ohjelmalle, jolla saataisiin tehtyä uimahallin kartta, joka sisältäisi uimaradat, joille kilpailijat voitaisiin ohjata.

Näin tarkkaan suunniteltu ohjelma olisi ollut turhan rajoittunut ja tarkkaan räätälöity. Tästä syystä lähdettiin ohjelman perusosia yleistämään siten, että sitä voitaisiin mahdollisesti käyttää muissakin tilanteissa.

Alussa mietittiin, minkälainen tila uimahalli on ja mitä osia siitä voitaisiin yleistää. Uimaradat yleistettiin urheilupisteiksi, joita voitaisiin lisätä ohjelmaan urheilutapahtuman ja urheilutilan tarpeiden mukaan. Urheilupisteitä ovat esimerkiksi tenniskenttä ja sulkapallokenttä palloiluhalliin tai pituushyppypaikka, kolmiloikka ja seiväshyppy yleisurheilutapahtumaan. Urheilupisteestä alkuun tehtäisiin tietokantataulukko, joka sisältäisi urheilupisteen fyysisen koon, pisteen nimen ja piirretyn yleiskuvan eli karttakuvakkeen. Karttakuvake pystytään sitten urheilupisteen tietojen perusteella mitoittamaan oikean kokoiseksi tapahtumapaikan kartalle. Eri urheilupisteitä pystytään kokoamaan kuvaan haluttu määrä. Tällä tavalla saadaan urheilupaikasta tietyssä mittakaavassa oleva kartta. Urheilupisteille tulisi myös saada omat aikataulunsa, jotka sisältäisivät kilpailijoiden nimet ja heidän suoritustensa kellonajan. Urheilupisteiden aikatauluilla helpotettaisiin urheilijoiden saamista oikeaan aikaan oikeaan paikkaan vähemmällä ohjauksella.

### 6.2.2 Käyttöliittymän määrittely

Alkuun käyttöliittymän määrittely -vaiheelle oli varattu enemmän aikaa, mutta ajan puutteen vuoksi käyttöliittymän määrittelyä ja suunnittelua vähennettiin minimiin. Minimillä tarkoitetaan sitä, että kaikkiin toimintoihin pääsee käsiksi, mutta esteettisesti järjestelmä saattaa olla käyttäjien mielestä ruma.

Myös koska tekijä ei ole käyttöliittymien suunnittelija eikä käyttöliittymä ole elintärkeä ohjelmistoa luodessa, tämä oli helppo jättää vähemmälle huomiolle. ASP.NET:n ominaisuuksiin kuuluu helpot muutokset ulkoasuun joko CSS:n kautta tai suoraan Visual Studion WYSIWYG-editorin kautta. Tähän tekijä yritti

ottaa kantaa lisäämällä mahdollisimman moneen ASP.NET- ja HTML-osaan CSS:n luokkaominaisuuden (engl. class).

## 6.3 Suunnitteluvaihe

### 6.3.1 Tietokannan suunnittelu

Tässä työssä käytettiin Microsoftin Visual Web Developer -ohjelmaan sisällytettyä MS-SQL-ohjelmistoa. Tietokanta rakennettiin Visual Web Developerin graafisella ohjelmalla.

Tietokannan jokaisella taululla tulisi olla oma pääavain, jolla tietue voidaan tunnistaa yksiselitteisesti. Tietokantaan suunniteltiin pitämään sisällään omat taulut seuraaville asioille.

Urheilupiste-taulu on taulu, johon kerätään urheilulajien suorituspaikkoja, esimerkiksi tenniskenttä, salibandykenttä, pituushyppyrata sekä hietikko ja moukarinheitto. Taulun tulisi sisältää vähintään seuraavat kentät:

- *Tunniste* toimii pääavaimena.
- *Nimi* on urheilupisteen helposti tunnistettava nimi.
- *Kuva* sisältää kuvan, tai kuvan osoitteen palvelimella, jota voidaan käyttää tilanmallinnuksessa kuvaamaan paikkaa.
- *Koko* sisältää alueen viemän fyysisen tilan senttimetreinä pituudelle ja leveydelle. Tätä käytetään piirtämään kuva oikeassa mittasuhteessa.
- *Lyhyt kuvaus* sisältää urheilupisteen lyhyen kuvauksen, kuten mitä urheilupisteessä voidaan tehdä. Esimerkiksi tenniskentällä voi verkkoa vaihtamalla pelata sulkapalloa.

*Urheilupaikka*-taulu on taulu, joka on kuva lopullisesta tilasta, jota ohjelmalla on mallinnettu. Taulun tulisi sisältää vähintään seuraavat kentät:

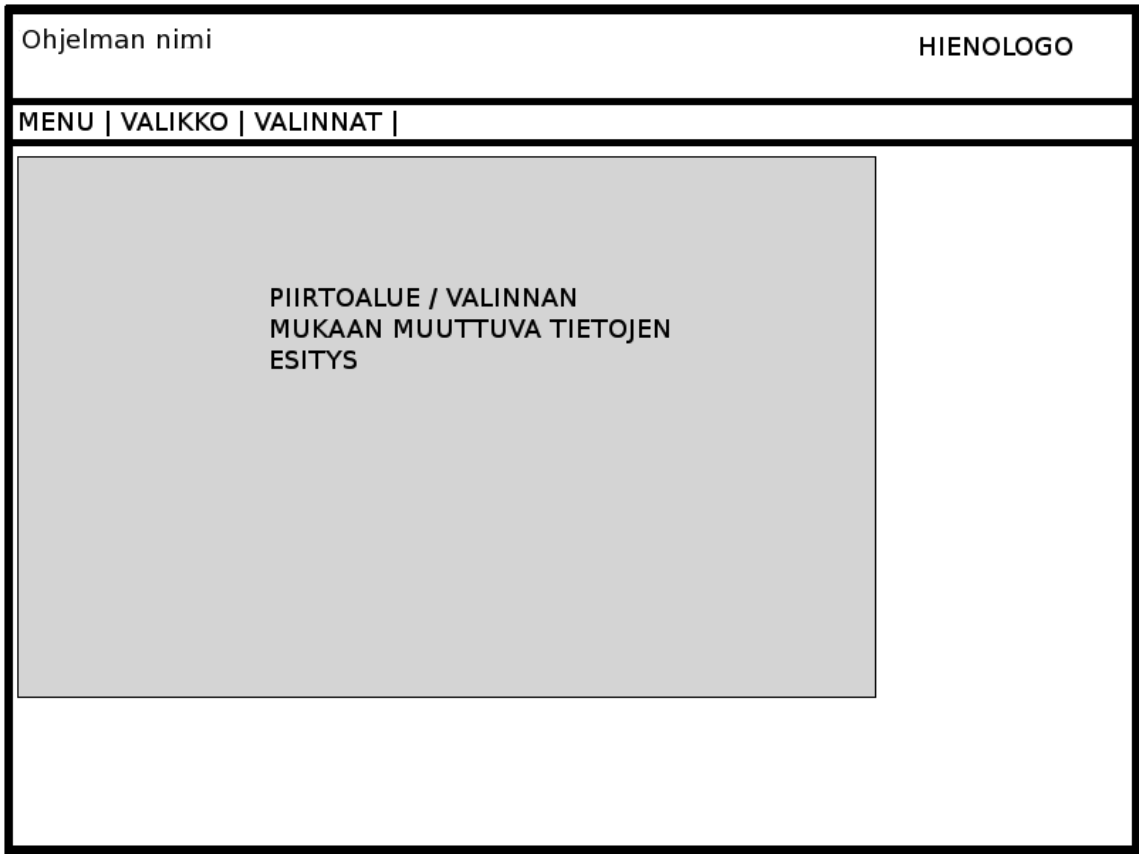
- *Tunniste* toimii pääavaimena.
- *Nimi* on urheilupaikan nimi.
- *Lyhyt kuvaus* sisältää lyhyen kuvauksen urheilupaikasta.
- *Mittakaava* on mittakaavan suhdeluku verrattuna yhteen senttimetriin. Esimerkiksi arvo 100 tarkoittaa suhdetta 1:100 eli yksi senttimetri kuvassa vastaisi maastossa sataa senttimetriä.

Pala-taulu on taulu, joka sisältää viittaukset urheilupiste-taulun tietueisiin, joita Urheilupaikka-taulussa käytetään. Pala-taulu toimii viitetauluna, josta poimitaan tiettyyn kuvaan (urheilupaikkaan) kuuluvat kuvakkeet (urheilupisteestä) ja esitetään ne oikeassa paikassa ja oikein päin. Taulu sisältää vähintään seuraavat kentät:

- *Tunniste* toimii pääavaimena.
- *Urheilupaikan tunniste* toimii viiteavaimena tämän taulun ja Urheilupaikka-taulun yhteyttäjänä.
- *Urheilupisteen tunniste* toimii viiteavaimena tämän taulun ja Urheilupiste-taulun yhteyttäjänä.
- *Paikka* kertoo kyseisen kuvakkeen paikan kuvassa.
- *Kiertokulma* kertoo, missä kulmassa kyseinen kuvake tulee piirtää kuvaan.
- *Kerros* kertoo, missä järjestyksessä kuvat tulisi piirtää käyttäjälle.

### 6.3.2 Ulkoasun suunnittelu

Ohjelman ulkoasu suunniteltiin useammaksi eri sivuksi, joita Masterpagessa sijaitsevan menun kautta voitaisiin vaihtaa. Opinnäytetyössä on vain kaksi sivua, tilanmallinnus-ohjelman sivu ja tilanmallinnuspalikoiden hallinnointisivu. Tilanmallinnus-ohjelman sivulla on myös tallennetut kuvat sekä kuvaan sijoitetut elementit. Suunnitelman ulkoasun näkee kuvasta 8.




*KUVA 8: Sivun ulkoasu*

### 6.3.3 Toimintojen suunnittelu

Toiminnot suunniteltiin pohjalta ylöspäin. Alkuun tehtiin ydinjärjestelmän olennaisimmat osat ja niiden päälle rakennettiin järjestelmään lisää ominaisuuksia.

Ensimmäinen järjestelmän osa, joka tarvitsi toiminnallisuudet, olivat urheilupisteet. Urheilupisteet ovat järjestelmän pohjusta, jolla on vaikutusta kaikkialla muualla järjestelmää ja jonka päälle loppuosa ohjelmistosta rakennetaan. Urheilupaikat, eli itse karttakuvat, koostuvat useammasta urheilupisteestä. Tässä osassa järjestelmää pitäisi voida lisätä uusia, poistaa vanhoja ja muokata jo olemassa olevien urheilupisteiden tietoja. Tälle osiolle tehdään oma sivunsa järjestelmässä.

Samanlaiset toiminnallisuudet tarvittaisiin myös itse kartalle ja siihen lisätyille palasille. Tämä toteutettaisiin suoraan itse mallinnuksessa, jossa voitaisiin valita tila, jota mallinnettaan sekä siihen lisätyt yksittäiset osat.

### 6.3.4 Tietokannan integraation suunnittelu

Aikaisemmin luvussa 6.3.1 kerrottiin tämän työn tietokannan suunnittelusta. Tässä luvussa kerrotaan, kuinka itse tietokantaa käytettiin järjestelmässä.

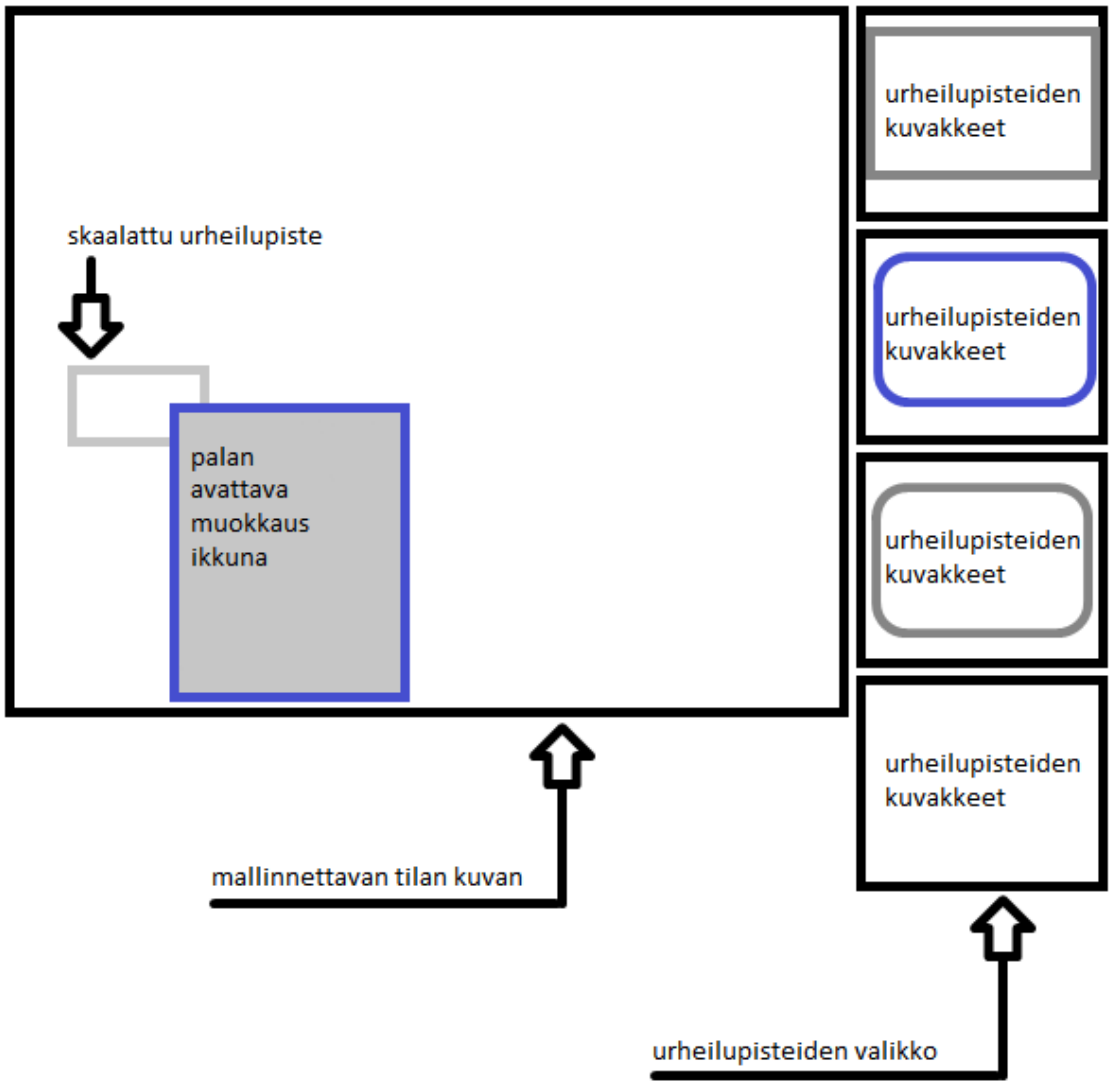
Kaikki taulut tietokannasta tultaisiin sijoittamaan omiin taulukkoihinsa, joista niitä voidaan tarkastella. Periaatteelliset päätaulukot, joiden sisällön ei tarvitse muuttua tilanmallinnusosassa ohjelmaa tehtyjen valintojen perusteella, ovat urheilupaikka ja urheilupiste. Näihin tauluihin tulisi päästä suoraan käsiksi ja niitä pitää voida muokata ja siis luoda uusia suoraan taulukosta. Jokaista tietueessa olevaa saraketta, lukuun ottamatta pääavainta, joka luodaan automaattisesti, kun uutta tietuetta luodaan tauluun, tulisi voida muokata taulukosta tavalla tai toisella.

Pala-taulua käytetään mallinnustyökalun kautta, josta voidaan lisätä, poistaa ja muokata urheilupaikan karttaa. Pala-taulua siis käytettäisiin pääsääntöisesti kyseisessä osassa ohjelmistoa. Kuitenkin pala-taululle annettaisiin oma taulukko, jonka sisältö muuttuu valittuna olevan urheilupaikan mukaan. Taulukosta nähdään koko ajan ne palat, jotka ovat urheilupaikassa käytössä ja mahdolliset ongelmat voitaisiin korjata tätä kautta. Esimerkiksi jos kaksi pala-taulun kuvaa on päällekkäin eikä alempaan päästä itse mallinnustyökalussa napauttamaan.

### 6.3.5 Mallinnustyökalujen suunnittelu

Mallinnustyökalun tarkoitus on tarjota graafinen näkymä tilasta, jota ollaan mallintamassa. Kuvassa 9 näkyy suunnitelma, millainen ohjelman ulkoasu voisi

olla. Mallinnustyökalulla muokataan pala-taulua, jossa sijaitsevat valittuna olevan urheilupaikan osaset. Ensin luodaan uusi urheilupaikka tai ladataan jokin jo tehdyistä urheilupaikoista. Valinnan jälkeen ohjelma hakee kyseisen urheilupaikan tietojen perusteella pala-taulusta ne tietueet, jotka tilaan tulisi piirtää ja mihin kohtaan. Tämän jälkeen käyttäjä voi aloittaa mallintamaan ohjelmalla, lisäämällä uusia urheilupisteitä ja muokkaamalla tai poistamalla jo kuvassa olevien urheilupisteiden tietoja.



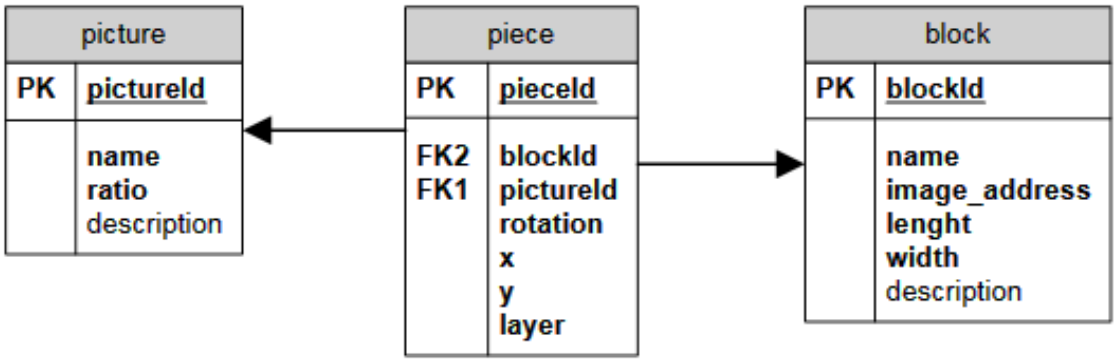

*KUVA 9. Tilanmallinnusohjelman käyttöliittymäsuunnitelma*

## 6.4 Toteutusvaihe

Toteutusvaiheen aikana suunnitelmat yritettiin toteuttaa mahdollisimman tarkasti vastaamaan tehtyjä päätöksiä.

### 6.4.1 Tietokannan toteutus

Tietokannan nimet vaihdettiin englannin kielelle ja sellaisiksi, joka kuvaa paremmin sitä, mitä ne ohjelmallisesti toteuttavat. Urheilupiste-taulun nimi muutettiin *block*-tauluksi, koska sen tiedot toimii rakennuspalikoina urheilupaikalle. Urheilupaikka-taulun nimeksi vaihdettiin *picture*-taulu, koska se on lopputulos, joka saadaan valmiiksi ohjelmalla. Pala-taulu nimettiin *piece*-tauluksi, koska kuva koostuu yksittäisistä paloista. Myös taulujen kenttien nimet muutettiin englanninkielisiksi. Kuvassa 10 näkyy tietokannan lopullinen muoto.




*KUVA 10. Tämän työn tietokanta*

### 6.4.2 Ulkoasun toteutus

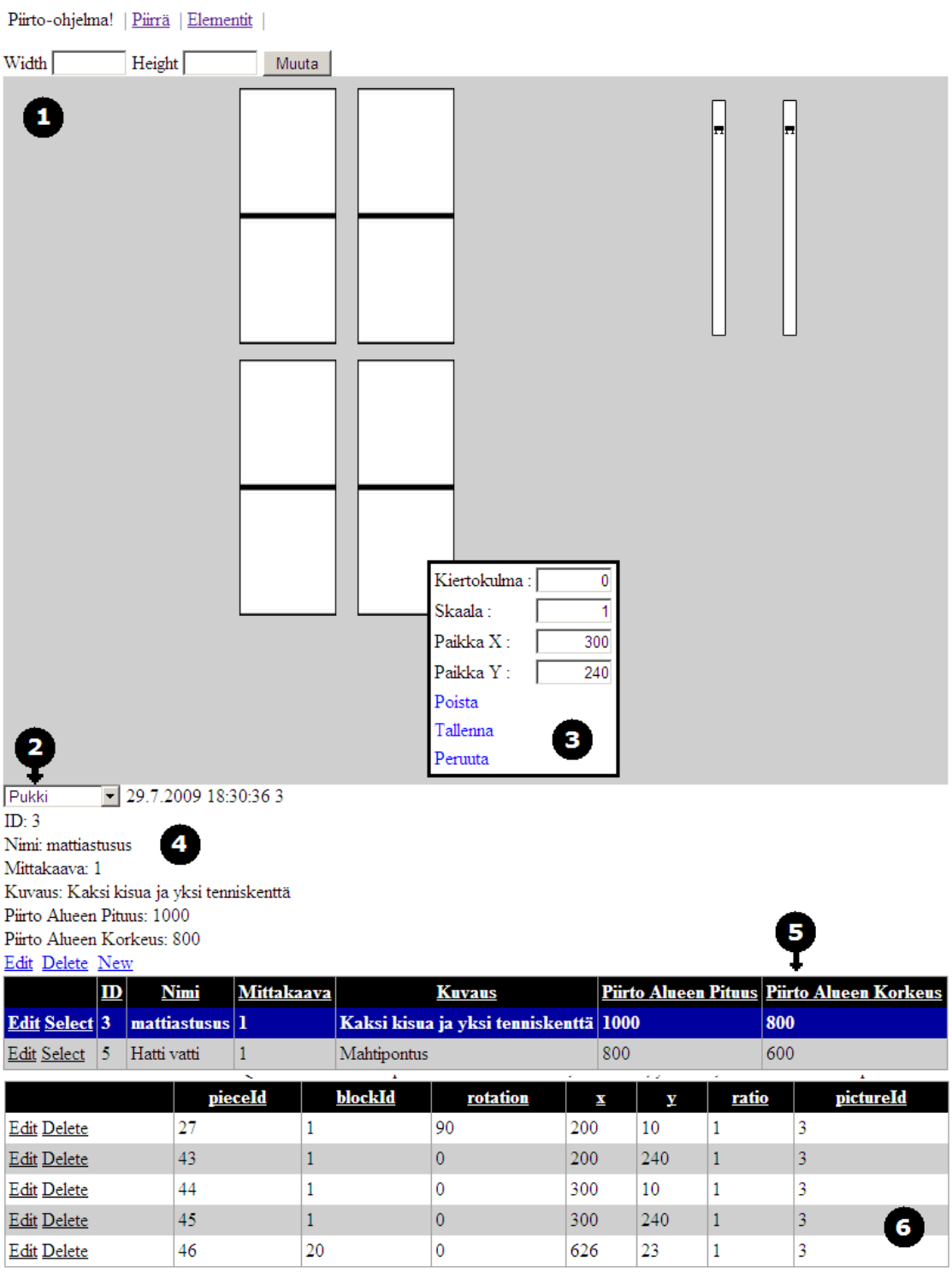
Ohjelmistoon tehtiin suunnitelmien mukaan Masterpage, johon tehtiin erilliset linkit tilanmallinnus- ja urheilupisteiden lisääminen tietokantaan -osaan ohjelmaa. Vaikka Masterpagen hyöty tässä tapauksessa oli vielä vähän

epämääräinen, se helpottaa jatkokehitystä ja ohjelmiston laajentamista. Masterpageen voidaan myöhemmin lisätä esimerkiksi sisäänkirjautuminen, jolla voidaan estää ohjelmiston sisäänpääsy ja rajoittaa näkymiä. Tällä hetkellä Masterpage tarjoaa yhteisen päänäkymän ja linkit ohjelman eri osiin.

Tilanmallinnussivun, jonka toteutunut ulkoasu näkyy kuvassa 11, sisältää itse ohjelman, jolla voidaan luoda kuva halutusta tilasta sekä muokata sitä. Suurin osa työstä tulee keskittymään tähän osaan ohjelmaa. Tilanmallinnussivu sisältää seuraavat kohdat, jotka on merkattu kuvaan ympyröidyillä numeroilla:

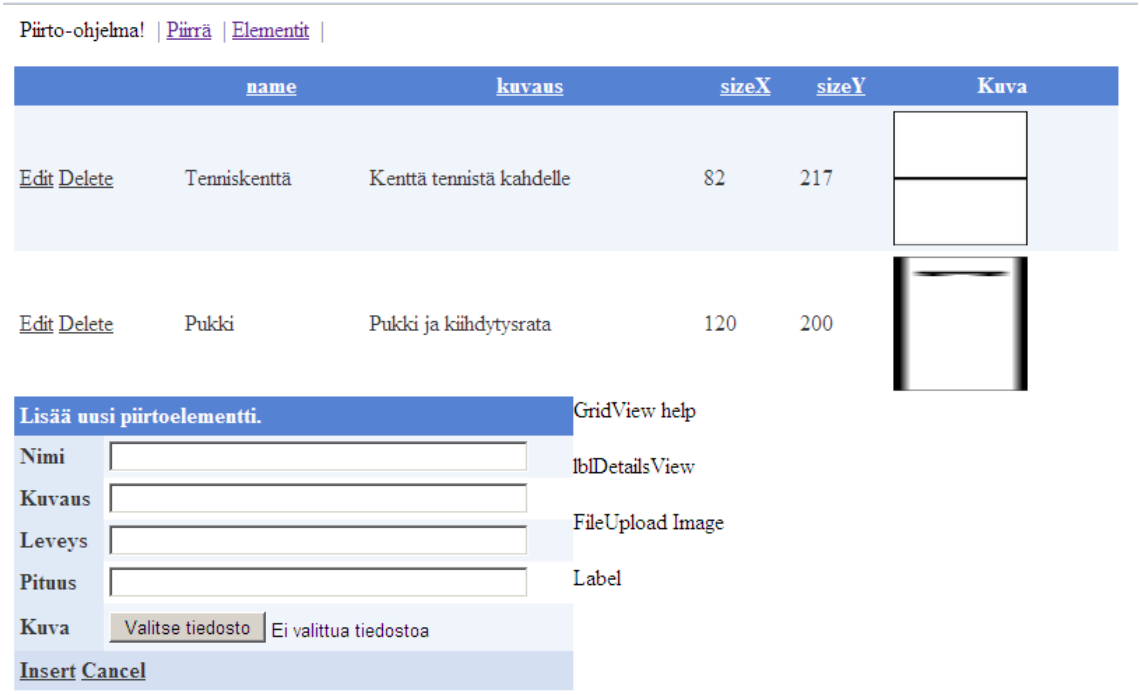
1. Tilanmallinnusalue, tai niin sanottu piirtotila
2. Urheilupisteen alasvetovalikko
3. Urheilupisteen muokkausvalikko
4. Urheilutilan Formview
5. Urheilutilojen Gridview
6. Urheilutilan sisältämien urheilupisteiden Gridview.

Tilanmallinnusalueesta lisätään urheilupisteitä ja luodaan tila. Urheilupisteiden alasvetovalikosta valitaan, mikä urheilupiste liitetään tilaan, kun tilanmallinnusalueella napautetaan hiirellä tyhjää kohtaa. Urheilupisteen muokkausvalikko tulee esiin, kun hiirellä napautetaan jotain jo tilanmallinnusalueella olevaa urheilupistettä. Urheilutilan Formviewissä näytetään Urheilutilojen Gridviewissä valittuna oleva elementti tai luodaan uusi elementti, jos mitään tilaa ei ole valittu. Urheilutilan urheilupisteet Gridviewissä näkyy, mitä urheilupisteitä on jo lisätty valittuna olevaan urheilutilaan.




*KUVA 11. Tilanmallinnussivu*

Urheilupisteiden lisäyssivu, joka näkyy kuvassa 12, sisältää Gridview-elementin (ensimmäinen taulukko), jolla voidaan poistaa vanhoja ja muokata olemassa olevia urheilupisteitä. Urheilupisteiden Gridview-taulukon alla on Formview, jolla lisätään uusia urheilupisteitä ohjelmaan. Urheilupisteiden kuvia ei ohjelmalla ole tarkoitus piirtää. Urheilupisteiden kuvat on tarkoitus tehdä jossain piirtämiseen tarkoitetussa ohjelmassa kuten GIMP- tai Corel Draw -ohjelmilla.




*KUVA 12. Urheilupisteiden lisäyssivu*

### 6.4.3 Urheilupisteiden lisäystoiminnot

Urheilupisteiden lisäyssivu sisältää Gridview-elementin, joka käyttää sqlDataSourcea, jonka kautta haetaan pala-taulun tiedot. ASP.NET osaa tehdä suoraan view-tauluihin muokkaus-, poisto- ja lisäystoiminnot tekstikentille ilman, että tarvitsee itse kirjoittaa jokaista SQL-lausetta ja koodata toimintoa C#:lla.

Gridviewiin jouduttiin tekemään muutoksia perusasetuksiin, koska se ei tue kuvien näyttämistä tai lisäämistä ja muokkaamista suoraan. Näille kirjoitettiin omat lisäys- ja muokkaustoiminnot sekä kuvan esittäminen Gridviewissä. Lisäykseen ja muokkaukseen tehtiin tiedoston haku ja tallennus palvelimelle, jotka koodattiin C#:lla. Tiedosto tallennetaan itse palvelimen tiedostojärjestelmä ja tiedoston osoite ja nimi tietokantaan. Kuvan esittäminen lisättiin Gridviewin soluun HTML-image osioon, johon kirjoitetaan tietokannassa olevan kuvan osoite.

Kun perussivut olivat valmiita, tarvittiin tietokanta ja sinne taulukot, joihin voitaisiin alkaa kerätä dataa. Alkuun tietokantaan tehtiin taulu nimeltä block, joka sisältää urheilupisteen tarvitsemat tiedot, joita ovat nimi, kuva, mittasuhde, paikka (kuvan paikka ruudulla) ja kuvaus. Tämän jälkeen urheilupisteiden lisäyssivulle laitettiin sqlDataSource paketti, joka yhdistettiin tietokantaan ja siellä olevaan tauluun. Seuraavaksi sivulle laitettiin Gridview, joka mahdollistaa suoraan muutamalla valinnalla sen ominaisuuksista tietokannan tietojen esittämisen sekä muokkauksen ja poiston.

### 6.4.4 Tilanmallinnustyökalun sivu

Piirtotyökalun toteutus aloitettiin lisäämällä sivulle tila-taulun ja pala-taulun tietojen näyttämiseen, tallentamiseen, lisäämiseen, poistamiseen ja muokkaukseen tarvittavat Gridviewit. Tila-taulun Gridviewiin lisättiin myös valinta ominaisuus (engl. select), joka antaa mahdollisuuden valita jonkin tietueen käyttöön. Sivuille lisättiin myös Formview, jota käytetään lisäämään uusia tiloja sekä näyttämään tämän hetken valittuna olevan tilan tiedot isommassa muodossa.

Tila-taulun Gridview-valinta ei itsessään tee mitään, mutta antaa mahdollisuuden ohjelmoida toimintoja tapahtumalle. Valinnalle koodattiin toiminto, joka vaihtaa pala-taulun Gridview-näkymää sen mukaan, mikä tila-taulun tietue on valittu, sekä tila-taulun Formview näyttämään valittuna olevan

tilan tiedot. Näkyviin tulevat tiettyyn tilaan sidotut pala-taulun tietueet. Tämä mahdollistaa pala-taulun suoran muokkaamisen, jos esimerkiksi jotain käytössä olevaa palaa ei voida suoraan tilanmallinnusalueelta valita ja muokata. Kun valinta on tehty, haetaan pala-taulusta kaikki tilaan kuuluvat palat ja näiden tietojen perusteella lisätään tilanmallinnusalueelle ne urheilupisteiden kuvat, joihin pala-taulussa viitataan.

Urheilupisteiden kuviin lisätään myös oma piilotettu div-elementti, joka sisältää palan muokkaus form-elementin. Menu div-elementti tuodaan esille JavaScript funktiolla, joka laukeaa, kun urheilupisteen kuvaa painetaan hiirellä. JavaScript funktio muuttaa div-elementin style display -ominaisuuden inlineksi hiddenistä.

### 6.4.5 Tilanmallinnustyökalun toteutus

Tilanmallinnustyökalu oli projektin ainoa osa, jossa itse ohjelmoitiin JavaScriptiä. Ohjelman muissa osissa annettiin ASP.NETin tehdä selaimen JavaScript-toiminnot itse. JavaScriptilla ja ASP.NET AJAX -työkaluilla tehtiin käyttäjälle mahdollisimman työpöytäsovellusta vastaava kokemus.

Sivulle tehtiin div-elementti, joka tulisi toimimaan tilanmallinnusalueena. Sivu-elementti laitettiin ASP.NET AJAXin osion sisään, jotta saataisiin tehtyä alueelle osittainen sivunpäivitys (partial page update). Tälle alueelle tultaisiin lisäämään JavaScriptilla urheilupisteiden lisäys sekä urheilupisteiden ominaisuuksien muokkaus menu.

Urheilupisteen valinta tehtiin vetovalikoksi tilanmallinnusalueen vasempaan alakulmaan. Tämä oli helpoin ja nopein tapa saada valinnat suoraan luettavaksi JavaScriptillä ja serverillä pyörivälle C#:lle.

Seuraavaksi ohjelmoitiin sivuille hiiren osoitinta seuraava JavaScripti. Tätä tultaisiin käyttämään tilanmallinnussivulla katsomaan mihin kohtaan on hiirellä painettu ja sen koordinaatit, jonka origona on mallinnusalueen vasen yläkulma.

Koordinaatit ovat positiivisia oikealle- ja alaspäin mentäessä origosta. Koordinaattien paikkaa käytetään joko valitsemaan urheilupisteen lisäyspaikka tai avaamaan urheilupisteen muokkausvalikko valittuun kohtaan, sen mukaan painettiinko hiirellä tyhjää aluetta vai urheilupistettä.

JavaScriptillä ohjelmoitiin toiminnot, jolla saataisiin tehtyä tilanmallinnus sivulle ASP.NETin DoPostback-funktiota käyttävä jatke, jolla viedään tietoja serverille. DoPostback on ASP.NETin oma JavaScript-toiminto, jolla mahdollistetaan automaattinen submit-ominaisuus lomakkeilla ja syötteille sivuilla, ohjelmoijan koodaamin perustein. DoPostBack sijaitsee HTML-elementin scriptin tapahtuman laukaisu-elementissä. DoPostBack toimintoa käytetään lisäämään pala-tauluun uuden palan tiedot, kun käyttäjä on valinnut halutun urheilupisteen vetovalikosta ja painanut tilanmallinnusalueella olevaa tyhjää kohtaa. Samaa jatketta käytettiin pala-taulun muokkaus valikon tietojen tuontiin ja vientiin palvelimelle.

# 7 YHTEENVETO

Opinnäytetyön kohteena oli rakentaa tilanmallinnusohjelma, jota voitaisiin käyttää internet-selaimella mistä tahansa. Ohjelma on tarkoitettu helpottamaan tilojen suunnittelua urheilutapahtumaan, koska ohjelma sisältää halutun mallinnettavan tilan koon sekä urheilupisteiden kokoja. Tämä eliminoi tarpeen paikanpäällä olemiselle ja mittanauhoilla suunnittelun. Ohjelmalla mallinnettuja tiloja pystytään käyttämään karttoina tulostamalla nämä paperille tai kuvaksi nettisivuille.

Ohjelman rakentaminen sekä Microsoftin ASP.NET:n ja .NETin opettelu oli melkoisen raskas kokemus samaan aikaan. Jostain syystä .NET-alustassa kaikille on omat olionsa ja tavat käydä tiedot läpi muuttujista. Verrattuna .NET:iin PHP:ssä käytettiin erittäin vähän olioita sekä lähes kaikki funktiot palauttivat normaalin arrayn tai muuttujan, jonka pystyi käymään läpi normaalissa foreach-silmukassa. .NET-alusta tarvitsi jokaiselle oman kontrollerin tai olion, jolla niihin tallennetut tiedot voitiin käydä läpi.

Ohjelmaan oli myös tarkoitus sisällyttää aikataulutukset jokaiselle urheilupisteelle. Tällä olisi saatu myös suunniteltua ajastukset urheilijakohtaisesti tapahtumalle. Urheilijat olisi voitu myös luokitella johonkin järjestöön, kuten kouluun, ja olisi voitu tulostaa järjestökohtainen aikataulutus. Ominaisuus jäi toteuttamatta ajan puutteen vuoksi.

ASP.NET on ohjelmointialustana nopeasti käyttöön otettavissa ja sillä saa nopeasti tehtyä perusnettisivut. Alusta sisältää myös hyvät laajennusmahdollisuudet, mutta tämä vaatii paljon harjoittelua ja opiskelua asiaan eikä ole läheskään yhtä helposti lähestyttävä kuin perussivujen toteutus on.

# LÄHTEET

.NET Framework Conceptual Overview. 2008
Saatavissa: http://msdn.microsoft.com/fi-fi/library/zw4w595w(en-us).aspx.
Hakupäivä 1.9.2008.

ASP.NET Overview. 2008
Saatavissa: http://msdn.microsoft.com/fi-fi/library/4w3ex9c2(en-us).aspx.
Hakupäivä: 1.9.2008.

Haikala, Ilkka - Märijärvi, Jukka 2000. Ohjelmistotuotanto.
Helsinki: Talentum.

JavaScript Mozilla. 2008
Saatavissa: https://developer.mozilla.org/en/JavaScript.
Hakupäivä 1.9.2008.

Martin, Joe - Tomson, Brett 2002. ASP.NET Trainer Kit. Suom. Arola Jussi.
Helsinki: Edita.

Hernandez, Michael 2000. Tietokannat – Suunnittelu Käytännössä.
Helsinki: IT-Press

Spaanjaars, Imar 2008. Beginning ASP.NET 3.5: In C# and VB. US, Indianapolis: Wiley Publishing, Inc.

Shepherd, George 2008. Microsoft ASP.NET 3.5: Step by Step.
Redmond, Wash: Microsoft Press

Stephens, Ryan - Plew, Ronald - Morgan, Morgan - Perkins, Jeff 2001.
SQL Tietokantaohjelmointi Trainer Kit
Helsinki: IT-Press

SQL Server 2008 Overview. 2008.
Saatavissa: http://www.microsoft.com/sqlserver/2008/en/us/default.aspx
Hakupäivä: 1.9.2008

The Official Microsoft ASP.NET Site. 2008.
Saatavissa: http://www.asp.net/
Hakupäivä 1.9.2008.