Lauri Mattila

TYÖKALU SARJAOHJELMIEN ANALYSOINTIIN

Tietojenkäsittelyn koulutusohjelma
Sovellusohjelmoinnin suuntautumisvaihtoehto
2009

TYÖKALU SARJAOHJELMIEN ANALYSOINTIIN

Mattila, Lauri
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Toukokuu 2009
Kyngäs, Jari
UDK: 004.42
Sivumäärä: 24

Asiasanat: ammattiurheilu, Ajax-ohjelmointi, JavaScript, ASP.NET

Opinnäytetyön tarkoituksena oli kehittää analysointityökalu sovellukseen, jonka avulla voidaan muokata sarjaohjelmia. Kehitetyn työkalun avulla voidaan analysoida sarjaohjelmiin tehtyjä muutoksia sekä nähdä, miten muutokset vaikuttavat sarjaohjelmaan.

Analysointityökalu tuli Satakunnan ammattikorkeakoulun käyttöön. Opinnäytetyö on osa Satakunnan ammattikorkeakoulun tutkimusprojektia. Tutkimusprojektin tuloksena syntyi ohjelma, jonka avulla voidaan laskea sarjaohjelmia joukkuelajien urheilusarjoille.

Opinnäytetyö laajentaa aikaisemmin tehtyä selainkäyttöistä sarjaohjelman muokkaussovellusta. Sovellus on jaettu kahteen osaan, palvelimella olevaan ASP.NET-sovellukseen sekä selaimessa toimivaan JavaScript-sovellukseen. Opinnäytetyön toteutuksessa käytettiin myös SQLite-tietokantaa, ja tiedonsiirto hoidettiin pääasiassa Ajaxin avulla.

Opinnäytetyö sisälsi käyttöliittymäsuunnittelua, ohjelmointia, testausta ja dokumentointia.

A TOOL FOR ANALYZING SPORT SCHEDULES

Mattila, Lauri
Satakunta University of Applied Sciences
Degree Programme in Business Information Systems
May 2009
Kyngäs, Jari
UDC: 004.42
Number of pages: 24

Key words: professional sport, Ajax-programming, JavaScript, ASP.NET

The purpose of this thesis was to develop an analysis tool for a software that is used for editing sport schedules. With the analysis tool, one can analyze the sport schedules and the affects of changing the schedules.

The analysis tool was made for the use of Satakunta University of Applied Sciences. This thesis is part of a research project of Satakunta University of Applied Sciences. As the product of the research project a program was made with which sport schedules can be constructed for sport leagues.

This thesis extends a web-based software that was made earlier. The software has been divided into two parts: the server-side ASP.NET program and the client-side JavaScript program. SQLite database was also used in implementing this thesis. The data was transferred mainly with Ajax.

The thesis contained designing the user interface, programming, testing and documentation.

SISÄLLYS

1 SYMBOLI- JA TERMILUETTELO

Urheilusarja: Tässä opinnäytetyössä urheilusarjalla tarkoitetaan tietyn urheilulajin joukkueiden muodostamaa ryhmää. Esimerkiksi jääkiekon SM-liiga on urheilusarja.

Sarjaohjelma: Sarjaohjelma tarkoittaa listaa, johon on kirjattu jonkin urheilusarjan tietyn kauden pelipäivät ja pelipäivien pelit.

Kierros: Tarkoittaa pelipäivää.

Jääkiekon SM-liiga: Jääkiekon SM-liiga on Suomen jääkiekon pääsarja.

Breikki (break): Tarkoittaa peräkkäisiä koti- tai vierasotteluita. Esimerkiksi jos Tappara pelaa kaksi kertaa peräkkäin kotona, syntyy yksi kahden pelin mittainen breikki.

Ohjelmakirjasto: Ohjelmakirjastot koostuvat aliohjelmista, joita käytetään apuna tietokoneohjelmien kehittämisessä sekä ohjelmien suorittamisen aikana.

Dynaaminen linkitys: Linkittäminen tarkoittaa sitä, kun ohjelmakirjaston resurssit ladataan sitä tarvitsevan ohjelman käyttöön. Dynaamisessa linkityksessä ohjelmakirjasto linkitetään ohjelman käynnistämisen yhteydessä tai vasta silloin, kun tiettyä aliohjelmaa tarvitaan. Staattisessa linkityksessä kirjasto tai kirjastot linkitetään silloin, kun ohjelma käännetään.

2 JOHDANTO

Opinnäytetyöni aiheena oli toteuttaa työkalu sarjaohjelmien analysointiin. Opinnnäytetyö laajentaa harjoittelussa ja myöhemmin töissä tekemääni selainkäyttöistä sarjaohjelman muokkaussovellusta. Sovelluksen tein Jari Kyngäksen ja Kimmo Nurmen tutkimuksen käyttöön. Kyngäksen ja Nurmen kehittämä algoritmi pyrkii laskemaan parhaan mahdollisen sarjaohjelman joukkuelajeille. Tekemäni sovelluksen avulla voidaan muokata Kyngäksen ja Nurmen algoritmin laskemaa sarjaohjelmaa.

Laadin sovelluksen työharjoitteluni kahden viimeisen kuukauden sekä yhden työkuukauden aikana. Ennen kuin aloitin tämän opinnäytetyön tekemisen, sovelluksen avulla oli mahdollista siirtää otteluita kierrokselta toiselle, poistaa ja lisätä kierroksia sekä tallentaa muutettu sarjaohjelma. Sovelluksessa oli myös tuki monikielisyydelle sekä useille eri urheilulajeille ja urheilusarjoille. Sovelluksen käyttäjäkuntaa ovat urheilusarjojen ja urheiluseurojen johtajat.

Sovelluksesta puuttui kuitenkin vielä mahdollisuus analysoida sarjaohjelma. Analysointityökalu näyttää, miten käyttäjän tekemät muutokset vaikuttavat sarjaohjelmaan. Ilman tietokoneen laskemaa analyysia on erittäin työlästä tutkia tehtyjen muutosten vaikutusta. Toisin sanoen analysointityökalu on ratkaisevan tärkeä sovelluksen hyödyllisyyden kannalta.

Aluksi minun oli tarkoitus tehdä pelkästään analysaattorin käyttöliittymä sekä rajapinta sen käyttöä varten, mutta päädyin toteuttamaan myös itse analysaattorin.
Aloitin siis suunnittelemalla analyysityökalun käyttöliittymän ja ulkoasun sekä rajapinnan, jolla analyysi lisätään käyttöliittymään. Tämän jälkeen aloin kehittää varsinaista analysaattoria.
Tässä raportissa käyn ensin läpi työssä käytetyt tekniikat. Siitä siirryn sarjaohjelman laskemiseen ja muokkaussovelluksen toimintaan. Luvussa 5 käsittelen ana-

lyysin sisällön ja analysaattorin toiminnan. Luvussa 6 selvitän, miten dokumentoin työni ja testasin analysaattorin toiminnan.

3 KÄYTETYT TEKNIIKAT

Koska tämä opinnäytetyö laajentaa jo valmiina olevaa järjestelmää, käytettävät tekniikatkin valittiin jo kun järjestelmää ruvettiin tekemään. Siksi en käy perusteellisesti läpi eri tekniikoiden valintaprosessia ja vertailua, vaan tyydyn esittelemään opinnäytetyössä käytetyt tekniikat.

3.1 ASP.NET

Palvelinpuolella käytetään ASP.NET-tekniikkaa. ASP.NET koostuu erilaisista palveluista, joiden avulla luodaan esimerkiksi HTML-sivu, joka lähetetään selaimelle. ASP.NET on osa Microsoftin .NET-sovelluskehystä. ASP.NET ei sido kehittäjää tiettyyn ohjelmointikieleen, vaan ohjelmoijan on mahdollista valita jokin ohjelmointikieli, joka on yhteensopiva .NET-sovelluskehyksen ajoympäristön eli CLR:n (Common Language Runtime) kanssa. Tässä opinnäytetyössä käytettiin ASP .NET:n kanssa C#-ohjelmointikieltä.

ASP.NET-sovellukset käännetään CIL-tavukoodiksi (Common Intermediate Language). Tavukoodi käännetään sovelluksen suorituksen aikana konekielelle CLR:n toimesta. Idea on siis hyvin samanlainen kuin Javassa. (Msdn 2009; Wikipedia 2009/a)

3.2 Ajax

Ajax (Asynchronous JavaScript and XML) -tekniikka tarkoittaa useiden eri tekniikoiden yhdistelmää. Ajax-termiä käytetään melko löyhästi, ja erilaisissa Ajax-toteutuksissa käytettävät tekniikat voivat vaihdella. Tavallisesti ainakin

seuraavat tekniikat mielletään osaksi Ajaxia: (X)HTML, DOM (Document Object Model) ja JavaScript. (Wikipedia 2009/b)

Ajaxin avulla voidaan siirtää tietoa selaimen ja palvelimen välillä ilman, että koko sivua tarvitsee ladata uudelleen. Käytännössä Ajax toimii niin, että JavaScriptin avulla otetaan synkronisesti tai asynkronisesti yhteyttä palvelimeen ja palvelimelta saadun vastauksen avulla voidaan päivittää jotain tiettyä osaa web-sivusta. Koska palvelimeen voidaan ottaa yhteyttä asynkronisesti, muiden skriptien suorittaminen jatkuu normaalisti. Ajaxin avulla saadaan sulavuutta sovellusten toimintaan nopeampien vasteaikojen ansiosta, kun sivua ei tarvitse pienten muutosten takia ladata kokonaan uudestaan. Käyttäjä voi esimerkiksi arvostella Youtube-videon samaan aikaan, kun katsoo sitä. Tämän kaltaisen toiminnallisuuden toteuttaminen ilman Ajaxia katkaisisi videon katselun, kun sivu jouduttaisiin lataamaan uudelleen. (Wikipedia 2009/b)

3.3 Ext JS -JavaScript-kirjasto

Ext JS on JavaScript-kirjasto, jonka avulla voidaan helposti luoda selainriippumattomia sovelluksia JavaScriptin avulla. Ext JS -kirjasto perustuu olio-ohjelmointimalliin (Extjs.com 2009). Erilaisia JavaScript-kirjastoja on suhteellisen paljon. Ext JS eroaa muista siinä, että sen mukana tulee kattava käyttöliittymäkomponenttikirjasto. Valmiiden käyttöliittymäkomponenttien avulla voidaan varsin helposti ja nopeasti luoda käyttöliittymiä, joiden ulkoasu ja käyttö vastaavat työpöytäsovellusten käyttöliittymiä. Valmiiden, muokattavien käyttöliittymäkomponenttien ansiosta ohjelmoijan ei välttämättä tarvitse kirjoittaa HTML-koodia eikä CSS-tyylimäärittelyjä. Uusien komponenttien luominen on helpompaa ja nopeampaa, kun voi periyttää ominaisuuksia ja metodeja toisista komponenteista sen sijaan, että täytyisi kirjoittaa jokainen komponentti alusta asti.

3.4 JSON

JSON (JavaScript Object Notation) on kevyt tiedonsiirtoformaatti (Json.org 2009). JSON on yksinkertaisempi merkintäkieli kuin XML, ja sitä on helppo lukea, parsia ja muodostaa.

JSON-koodatun tiedon käyttämisessä JavaScript-sovelluksessa on se etu esimerkiksi XML:ään verrattuna, että JSON on JavaScriptin osajoukko. Toisin sanoen JSON-muotoon koodatusta tiedosta on helppo muodostaa JavaScript-olio. (Wikipedia 2009/c)

Kaikki Ajaxin avulla lähetetty tieto palvelimen ja selaimen välillä on sarjaohjelman muokkaussovelluksessa koodattu JSON-muotoon.

3.5 SQLite

SQLite on pieni ja suorituskykyinen relaatiotietokantajärjestelmä. Verrattuna perinteisiin tietokantajärjestelmiin SQLite:n etuna on, että sen käyttämiseen ei tarvita tietokantapalvelinta. SQLiten moottori on pieni (n. 180−500 kt) ANSI-C:llä kirjoitettu ohjelmakirjasto. Koska SQLite on ohjelmakirjasto, se voidaan dynaamisesti linkittää sitä tarvitsevaan sovellukseen. SQLite tarvitsee toimiakseen hyvin vähän palveluita käyttöjärjestelmältä, ja se käyttää ainoastaan seitsemää funktiota C-kielen perusohjelmakirjastosta. Tästä johtuen SQLite on helppo kääntää erilaisille järjestelmille. (SQLite.org 2009a; SQLite.org 2009b; SQLite.org 2009c)

SQLite tukee lähes kokonaan SQL-92-standardia (SQLite.org 2009d). SQLite-tietokanta on yksittäinen tiedosto, joka pitää sisällään tietokannan rakenteen, tietokantaan tallennetun tiedon sekä tietokannan käyttöä ohjaavat asetukset.

SQLite sopii hyvin pieniin ja keskikokoisiin sovelluksiin ja ylipäänsä sellaisiin tilanteisiin, joissa usea käyttäjä ei käytä tietokantaa samanaikaisesti. Jos käyttäjiä on erittäin paljon tai jos suoritetaan todella isoja operaatioita ja tietoa on paljon, SQLite ei ole paras mahdollinen vaihtoehto. Kun lisätään, muokataan tai

poistetaan tietoa SQLite-tietokannasta, niin koko tietokanta joudutaan lukitsemaan muutosten ajaksi. Monissa muissa tietokantajärjestelmissä on mahdollista lukita vain se osa tietokannasta, johon suoritettavat toimenpiteet vaikuttavat. Tämän kaltainen tietokannan osittainen lukitseminen nopeuttaa tietokannan käyttöä, silloin, kun useampi käyttäjä käyttää tietokantaa samaan aikaan. Tämän opinnäytetyön tarpeisiin SQLite kuitenkin sopii hyvin, koska samanaikaista käyttöä ei ole kovin paljon eivätkä suoritettavat operaatiotkaan vie paljon aikaa. (SQLite.org 2009c)

4 SARJAOHJELMAN MUOKKAUSSOVELLUS

Tässä luvussa kerron, mitä sarjaohjelmien laskemisessa tulee ottaa huomioon, ja esittelen sarjaohjelman muokkaussovelluksen perustoiminnot.

4.1 Sarjaohjelman laskeminen

Niin kauan kun on ollut kilpaurheilua, on ollut myös tarvetta sarjaohjelmille. Viimeisen kolmenkymmenen vuoden aikana sarjaohjelmien laskemisesta on tullut oma tieteenalansa. Ammattiurheilusta on tullut koko ajan organisoidumpaa, ja siinä liikkuu paljon rahaa. Nykyään sarjaohjelmalta vaaditaan enemmän kuin koskaan aiemmin, joten hyvän sarjaohjelman laatiminen käsin on erittäin vaikeaa. Tietokoneiden teho on vihdoin kasvanut sellaisiin mittoihin, että on ylipäätään mahdollista laskea sarjaohjelmia todellisiin tarpeisiin. (Kyngäs & Nurmi 2009)

Suurin osa urheilusarjoista käyttää niin sanottua round-robin-järjestelmää, joka tarkoittaa sitä, että urheilusarjan joukkueet pelaavat toisiaan vastaan tietyn monta kertaa. Yleisimmät turnausmuodot ovat 2RR tai 4RR, eli joukkueet pelaavat kaksi kertaa tai neljä kertaa toisiaan vastaan. (Kyngäs ym. 2009)

Ei ole kovin vaikeaa kehittää algoritmia, joka laskee sarjaohjelman, jos ei tarvitse ottaa huomioon mitään ulkopuolisia tarpeita. Mutta kun pitäisi ottaa huomioon

kaikki sarjaohjelmaa koskevat toiveet, niin algoritmin vaativuustaso ja monimutkaisuus kasvavat jyrkästi. Esimerkiksi seuraavat asiat täytyy ottaa huomioon muodostettaessa sarjaohjelmaa jääkiekon SM-liigalle:

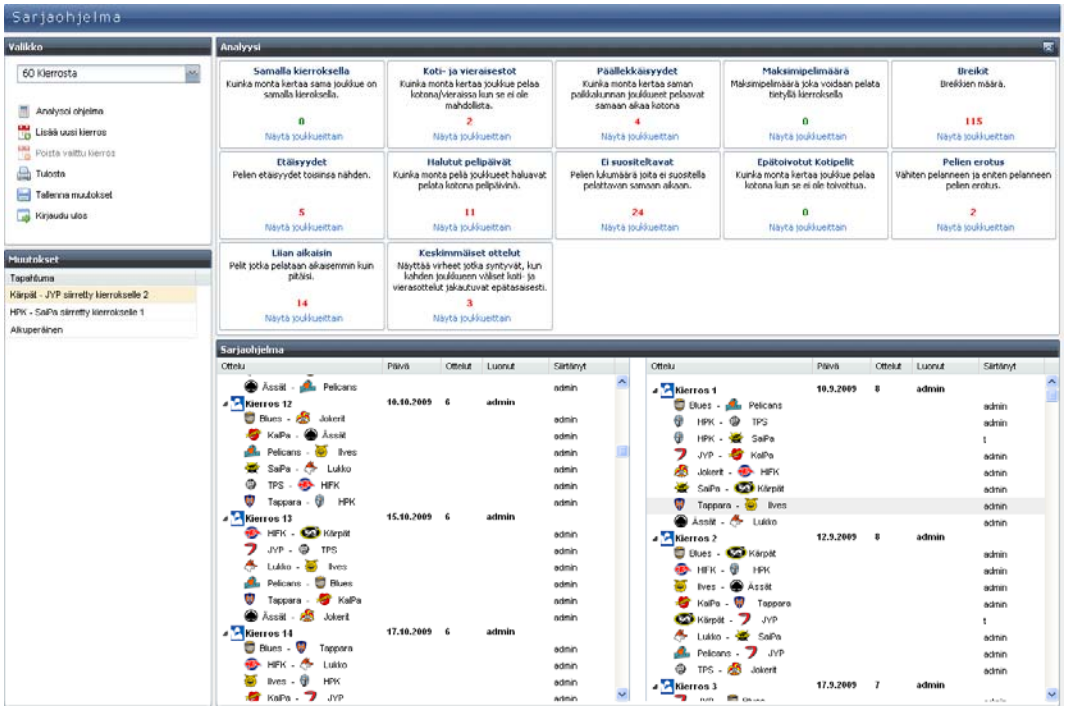
- joukkue voi pelata korkeintaan yhden pelin jokaisella kierroksella
- joukkue ei pysty pelaamaan kotona milloin vain, koska jäähalleissa järjestetään paljon muutakin toimintaa
- joukkueiden pitäisi pelata vuorotellen kotona ja vieraissa, breikkien minimoimiseksi
- tietyn joukkueparin kohtaamisten välissä pitäisi olla tietty määrä kierroksia
- joukkueet haluvat pelata tietyn määrän pelejä tiettyinä viikonpäivinä
- jos samalla paikkakunnalla on useita joukkueita, niin joko joukkueet eivät pysty pelaamaan kotona samaan aikaan tai sitten se ei ole toivottua
- joitain pelejä ei toivota pelattavan ennen määrättyä kierrosta
- kaikilla joukkueilla pitäisi olla yhtä paljon otteluita pelattuna koko ajan
- tietyt pelit täytyy pelata ennalta määrättynä päivänä
- tietyille kierroksille voidaan määritellä maksipelimäärä.

Yllä mainittujen kohtien lisäksi eri urheilusarjoilla saattaa olla vielä paljon muitakin ehtoja. Esimerkiksi pitkien välimatkojen maissa joukkueen peräkkäiset pelipaikkakunnat eivät saa olla liian kaukana toisistaan. Kaikkien ylhäällä lueteltujen ehtojen samanaikainen toteutuminen on käytännössä mahdotonta. Tästä johtuen pyritäänkin laskemaan vain paras mahdollinen sarjaohjelma. Koska sarjaohjelma vaikuttaa suoraan katsojalukuihin, joka vaikuttaa taas suoraan pelien tuottoon, hyvä sarjaohjelma on jokaiselle urheilusarjalle erittäin arvokas.

4.2 Sarjaohjelman muokkaussovellus

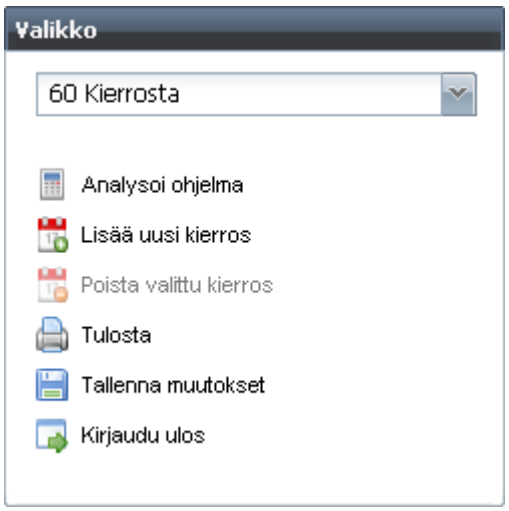
Seuraavaksi käyn läpi sarjaohjelman muokkaussovelluksen perustoiminnot. Käsittelen lyhyesti järjestelmän, jonka tein harjoittelujaksoni aikana. Pyrin antamaan peruskäsityksen siitä, minkälainen järjestelmä on ja mitä sillä voidaan tehdä. Tar-

kempaan käsittelyyn otan tämän opinnäytetyön aikana syntyneen analysointityökalun. Kuvassa 1. on yleisnäkymä sovelluksesta.



Kuva 1. Yleisnäkymä sarjaohjelman muokkaussovelluksesta

4.2.1 Valikko



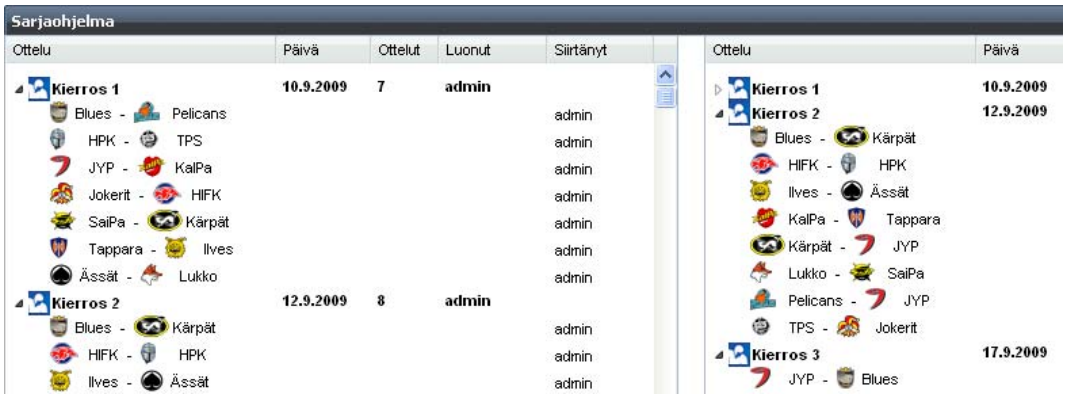
Kuva 2. Valikkonäkymä

Kuvassa 2 näkyy sovelluksen päävalikko. Ylhäällä olevasta alavetovalikosta näkee, että käyttäjä on valinnut 60 Kierrosta -nimisen sarjaohjelman.

Analysoi ohjelma -nappia painamalla sarjaohjelma analysoidaan ja analyysi näytetään käyttäjälle. Lisää uusi kierros -nappia painamalla avautuu pieni kalenteri, johon on merkitty nykyiset pelipäivät ja ottelut. Kalenterista käyttäjä voi valita pelipäivän uudelle kierrokselle. Kun pelipäivä on valittu ja käyttäjä on hyväksynyt sen, kalenteri sulkeutuu ja uusi kierros lisätään sarjaohjelmaan. Poista valittu kierros -napista käyttäjän valitsema kierros poistetaan. Tällä hetkellä nappi on merkitty harmaaksi, koska yhtään kierrosta ei ole valittu eikä kierroksia voi siis poistaa. Tulosta-napista aukeaa tulostusikkuna. Käyttäjä voi valita, haluaako hän tulostaa koko sarjaohjelman vai kenties pelkästään jonkin joukkueen ottelut. Käyttäjä voi tallentaa muutokset tietokantaan valitsemalla Tallenna muutokset. Tällöin sarjaohjelmasta luodaan uusi versio, jota taas toiset käyttäjät voivat tarkastella ja muokata. Muutoksia ei siis koskaan tehdä alkuperäisen sarjaohjelman päälle, vaan aina luodaan uusi versio. Kun käyttäjä painaa Kirjaudu ulos -nappia, hänet kirjataan ulos sovelluksesta..

4.2.2 Sarjaohjelman muuttaminen

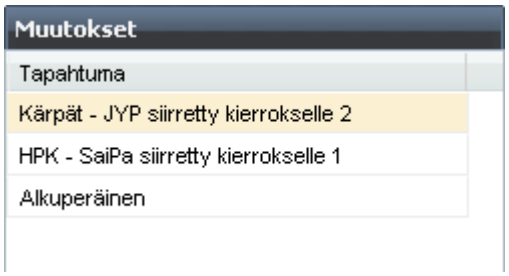
Kuvassa 3 näkyy, miten sarjaohjelma näytetään käyttäjälle. Kierrokset ja pelit on listattu kahteen identtiseen puurakenteeseen: oikealla ja vasemmalla puolella on näkymä samasta sarjaohjelmasta. Kumpaakin listausta voidaan selata toisistaan riippumatta. Kun listoja on kaksi, pelejä on helpompi siirrellä. Pelien siirtäminen kierrokselta toiselle tapahtuu samalla tavalla kuin esimerkiksi Windowsissa tiedostojen siirtäminen kansioihin, eli raahaamalla. Tässä tapauksessa kierrokset edustavat kansioita ja pelit tiedostoja. Vaikka pelejä raahattaisiin kierrospuusta toiseen, järjestelmä huolehtii siitä, että puut pysyvät identtisinä. Kierroksen ottelut voidaan myös piilottaa, jolloin ne vievät vähemmän tilaa. Piilottaminen onnistuu painamalla kierroksen vasemmalla puolella olevaa nuolta. Kuvassa 3 oikean puoleisessa kierrospuussa Kierroksen 1 pelit on piilotettu, kun taas vasemmalla puolella Kierroksen 1 pelit ovat näkyvillä.



Kuva 3. Sarjaohjelman ottelut ja kierrokset näytetään kahtena identtisenä puurakenteena

4.2.3 Muutosten hallinta

Kuvassa 4 näkyy muutosikkuna. Muutosikkuna sisältää taulukon, johon kirjataan sarjaohjelmaan tehdyt muutokset. Alimmaisena näkyy sarjaohjelman alkuperäinen tila. Tehtyjä muutoksia voidaan kumota tai toteuttaa uudelleen valitsemalla hiirellä taulukosta haluttu tila. Sarjaohjelman nykyinen tila on merkitty keltaisella värillä.



kuva 4. Sarjaohjelmaan tehdyt muutokset kirjataan ylös taulukkoon

5 ANALYSAATTORI JA ANALYYSI

Toteuttamani analysaattori laskee seuraavat asiat sarjaohjelmasta:

- Lasketaan, kuinka monta kertaa sama joukkue on samalla kierroksella. Esimerkiksi jääkiekon SM-liigassa joukkue ei saa eikä voi pelata kuin yhden pelin per kierros.
- Lasketaan, kuinka monta kertaa joukkueelle on merkitty koti- tai vieraspeli, silloin kuin se ei jostain syystä ole mahdollista. Joukkueen kotihalli voi olla esimerkiksi varattu jonkun muun tapahtuman käyttöön.
- Tarkistetaan, montako kertaa saman paikkakunnan joukkueet pelaavat samaan aikaan kotona. Esimerkiksi Tappara ja Ilves, jotka jakavat saman jäähallin, eivät voi pelata samaan aikaan kotona.
- Jos tietylle kierrokselle on asetettu maksimipelimäärä, tarkistetaan ettei se ylity ja listataan mahdolliset virheet.
- Lasketaan, kuinka monta breikkiä sarjaohjelmassa on. Listataan löydetyt breikit.
- Tarkistetaan, että kun kaksi joukkuetta pelaa vastakkain, joukkueiden kohtaamisten välissä on tietty määrä kierroksia. Jääkiekon SM-liigassa pelien välissä tulee olla vähintään seitsemän kierrosta.
- Joukkueet saavat esittää toiveita siitä, kuinka paljon pelejä he haluasivat pelata tiettyinä viikonpäivinä. Jos pelejä on enemmän tai vähemmän kuin mitä toivottiin, siitä kirjataan virhe. Analysaattori sietää yhden pelin heiton suuntaan tai toiseen. Esimerkiksi jos Ässät haluaisi pelata torstaisin yhdeksän peliä mutta sille on laitettu vain kahdeksan, niin siitä ei tule virhettä koska eroa on vain yhden pelin verran. Mutta jos pelejä olisikin torstaisin vain seitsemän, siitä kirjattaisiin virhe.
- Listataan pelit, joita ei suositella pelattavan samaan aikaan. Tällä tarkoitetaan joukkuepareja. Esimerkiksi jos Blues ja Jokerit pelaavat kotona samalla kierroksella, niin siitä kirjataan virhe.
- Lasketaan, kuinka monta kertaa joukkue pelaa kotona silloin, kun se ei ole toivottua. Päivät, jolloin ei ole toivottavaa, että joukkue x pelaa kotona, on listattu analysaattorille annettaviin asetuksiin.
- Listataan pelattujen pelimäärien erotus kierroksittain. Esimerkiksi: kierrokset 1-4, ero 1. Tämänkaltainen merkintä tarkoittaisi, että kierroksilla 1-4 eniten pelanneet joukkueet ovat pelanneet yhden ottelun enemmän kuin vähiten pelanneet.

- Tarkistetaan, ettei tiettyjä pelejä pelata liian aikaisin. Analysaattorin asetuksissa voisi esimerkiksi olla määritelty, että Tappara ja Ilves saavat pelata vastakkain aikaisintaan kierroksella kolme. Mutta jos Tappara ja Ilves pelaavatkin vastakkain jo kierroksella kaksi, niin tästä kirjataan virhe.
- Lasketaan virheet, jotka syntyvät, kun kahden joukkueen väliset koti- ja vieraspelit jakautuvat epätasaisesti. Esimerkiksi jos Blues ja Ässät pelaisivat ensin kaksi kertaa Porissa ja sitten vasta Espoossa, siitä merkittäisiin virhe.

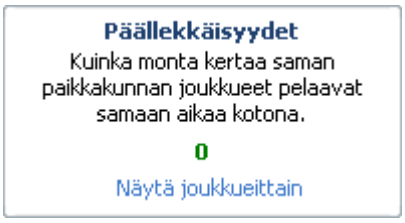
5.1 Analyysin ulkoasu ja käyttöliittymä

Pyrin suunnittelemaan analyysin ulkoasun niin, että käyttäjä näkisi mahdollisimman nopeasti, missä on virheitä ja mitä virheet tarkoittavat. Kun suunnittelin analyysin käyttöliittymää, yritin etsiä aikaisempia toteutuksia samasta aiheesta saadakseni apua ja vertailupohjaa suunnitteluun. Huomasin kuitenkin, ettei aikaisempia toteutuksia ole tai ainakaan niistä ei ole julkista tietoa saatavilla. Ilman valmista mallia suunnitteluun kului enemmän aikaa.

Kuvassa 5 näkyy yleiskatsaus analyysista. Jokainen analysoitu osa-alue on sijoitettu omaan paneeliinsa. Paneelit on sijoitettu analyysi-ikkunan sisälle. Yksittäisen paneelin sisällä on kuvaus virheestä, virheiden lukumäärä sekä linkki jota painamalla aukeaa tarkempi kuvaus mahdollisista virheistä.

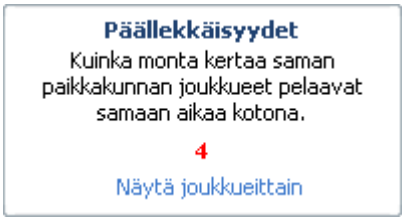
Kuva 5. Analyysipaneelit analyysi-ikkunassa

Kuvassa 6 näkyy esimerkki päällekkäisyyksien analysointipaneelista. Sarjaohjelmassa, jonka analyysistä kuva 6 on otettu, ei ole yhtään päällekkäisiä pelejä. Virheettömät tulokset merkitään vihreällä värillä.



Kuva 6. Esimerkki analyysipaneelista, jossa ei ole virheitä.

Kuva 7 taas on otettu sarjaohjelmasta, jossa on neljä päällekkäistä peliä. Virheiden lukumäärä merkitään paneelinäkymässä punaisella numerolla. Jos käyttäjä on kiinnostunut tarkemmasta analyysistä, hän pääsee siihen käsiksi Näytä joukkueittain -linkkiä painamalla. Tällöin aukeaa ikkuna, joka sisältää selitteen virheestä. Tämän lisäksi näytetään myös taulukko, johon on listattu virheiden syyt.



Kuva 7. Esimerkki analyysipaneelista, joka näyttää että analysaattori löysi neljä virhettä tästä kohdasta.

Kuvassa 8 on esimerkki seliteikkunasta. Kuvasta näkyy, että Jokerit ja HIFK pelaavat samaan aikaan kotona kierroksilla 11, 16, 21 ja 37.

Kuva 8. Esimerkki ikkunasta, joka aukeaa Näytä joukkuettain -linkistä

5.2 Käyttöliittymän muodostaminen

Analyysin käyttöliittymän muodostamiseen tarvitaan kolme eri komponenttia, AnalysisPanel, AnalysisDetailPanel ja DetailTable.

5.2.1 AnalysisPanel

AnalysisPanelista luotava olio on ikkuna, johon virhepaneelit sijoitetaan. Kun sarjaohjelma on analysoitu, analysaattorin palauttama tieto annetaan syötteenä AnalysisPanel-olion loadContent-metodille. Metodissa loadContent muodostetaan AnalysisDetailPanel-oliot. Luotaville AnalysisDetailPanel-olioille annetaan syötteenä virhetekstit, virhetaulukoiden sarakkeiden otsikot sekä analysaattorin laskema tieto mahdollisista virheistä. Virhe- ja otsikkotekstit haetaan erillisestä kielitiedostosta, joten käyttöliittymän ja ulkoasun kieli on helppo vaihtaa.

5.2.2 AnalysisDetailPanel

AnalysisDetailPanelista luotavat oliot ovat virhepaneeleja. AnalysisDetailPanel näyttää yksittäisen virheen kuvauksen sekä samantyyppisten virheiden määrän. Kun käyttäjä painaa Näytä joukkueittain -linkkiä, muodostetaan liikuteltava

ikkuna jonka sisälle sijoitetaan kuvaus virheestä sekä DetailTable-olio eli taulukko, jossa on yksityiskohtaisempi selostus löydetyistä virheistä.

5.2.3 DetailTable

DetailTable-olio näytetään käyttäjälle HTML-taulukkona. Taulukko muodostetaan DetailTablen loadGrid-metodissa. Luotavan taulukon ulkoasu määritellään erillisessä template-tiedostossa; näin yhden taulukon ulkoasua voidaan vaihtaa ilman, että se vaikuttaisi toisiin. Metodissa LoadGrid on määritelty perusalgoritmi taulukon muodostamiselle, mutta jos taulukon muodostaminen on monimutkaisempaa, template-tiedostossa voidaan määritellä taulukolle uusi render-metodi, jonka avulla taulukko muodostetaan. Template-tiedostossa määritellään muun muassa taulukon sarakkeiden leveydet, solujen muotoilut sekä käytettävät CSS-luokat.

5.3 Analysaattorin toiminta

Alkuperäisten suunnitelmien mukaan analysointi oli takoitus suorittaa niin, että Ajaxin avulla lähetettäisiin tiedot palvelimelle, jossa analyysi laskettaisiin, ja laskennan tulos palautetaisiin takaisin selaimelle. Myöhemmin päätimme kuitenkin, että analyysi olisi parempi laskea JavaSriptillä: se on nopeampaa, koska aikaa ei kulu tiedon serialisoimiseen eikä tiedon lähettämiseen palvelimen ja selaimen välillä. Analysaattorin toteutin kirjoittamalla Analyzer nimisen JavaScript-luokan, joka sisältää sarjaohjelman analysoimiseen tarvittavat metodit.

Kun käyttäjä on kirjautunut järjestelmään, hänen täytyy valita, mitä sarjaohjelmaa hän haluaa katsella tai muokata. Sen jälkeen sarjaohjelma ja sarjaohjelman analyysin asetukset ladataan palvelimelta Ajaxin avulla. Analyzer-oliolle annetaan syötteenä ColumnTree-olio ja analysaattorin ohjaamiseen tarvittavat asetukset. ColumnTree-olioon on listattuna sarjaohjelman kierrokset ja pelit. ColumnTreen avulla luodaan Round- ja Game-oliot. Round-oliot vastaavat kierroksia ja Game-oliot pelejä. Round-olioihin lisätään sille kuuluvat Game-oliot eli pelit, jotka kuu-

luvat kyseiselle kierrokselle. Näiden lisäksi muodostetaan vielä Team-oliot, jotka vastaavat joukkueita. Team-olio pitää sisällään joukkueen nimen sekä joukkueen pelaamat pelit.

Analyzer-luokka sisältää metodeja, jotka analysoivat sarjaohjelman eri osa-alueita. Kun Analyzer-olio on luotu, sarjaohjelma voidaan analysoida kutsumalla Analyzer-olion analyze-metodia. Analyze-metodille annetaan syötteeksi käytettävä ulkoasu, eli template. Analyze-metodi kutsuu laskentametodeja ja tallentaa metodien palauttamat tiedot tulosolioihin. Tulosolioihin tallennetaan myös template, jota käytetään graafisen näkymän muodostamisessa tuloksista.

6 DOKUMENTOINTI JA TESTAUS

Tässä luvussa käsittelen sovelluksen testaamisen, ja sovelluksesta tehdyn dokumentaation.

6.1 Dokumentointi

Dokumentoin analysaattorin laajentamalla sovelluksen dokumentaatiota. Sovelluksen dokumentaatio koostuu kahdesta eri osuudesta, yleisestä käyttöohjeesta sekä jatkokehittäjälle suunnatusta ohjelmoijan oppaasta.

Yleisessä käyttöohjeessa neuvotaan ohjelman asennus, käyttöönotto, tietojen lisääminen sekä järjestelmän ylläpito. Tähän osioon lisäsin ohjeen uusien sarjaohjelmien lisäämiseen tietokantaan sekä ohjeen analysaattorin asetustiedostojen luomiseen.

Ohjelmoijan opas koostuu kolmesta eri osasta. Ensimmäinen osa on sovelluksen arkkitehtuurin kuvaus. Toinen on palvelinpuolen C#-ohjelmakoodin kommenteista muodostettu chm-tiedosto. Kolmas osuus koostuu JavaScript-koodin sekaan kirjoitetuista kommenteista. JavaScript-ohjelmakoodista ei ole olemassa erillistä dokumentaatiota. Analysaattorin JavaScript-koodin dokumentaatio koostuu siis pelkästään

koodissa olevista kommenteista. Laajensin ohjelmoijan oppaan jokaista osaa analysaattorin osalta. Jos aikaa olisi, dokumentaatiota voisi vielä parantaa huomattavasti.

6.2 Analysaattorin testaus

Analysaattorin testaaminen, virheiden etsiminen ja korjaus vei paljon aikaa. Testauksessa suurena apuna toimi Firefoxin Firebug-lisäosa. Firebugin avulla voidaan esimerkiksi jäljittää virheitä (debug), tutkia CSS-tyylejä sekä JavaScriptin avulla dynaamisesti muodostettua HTML-koodia.

Testausta helpotti myös se, että Kyngäs oli tehnyt Java-ohjelmointikielellä analysaattorin. Sain Kyngäkseltä esimerkkisarjaohjelmia ja niistä laskettuja analyysejä. Pystyin näin vertaamaan tekemäni analysaattorin analyyseja Kyngäksen analyyseihin.

Ensin testasin jokaista analysoitavaa kohtaa erikseen. Testasin esimerkiksi sen, lasketaanko breikit oikein. Tein testisarjaohjelmia, siirtelin pelejä ja aiheutin tahallani virheitä. Kun kaikki kohdat oli testattu ja löydetyt virheet korjattu, oli aika testata analysaattoria oikeilla sarjaohjelmilla. Niillä virheitä löytyi vielä lisää.

Testasin analysaattorin seuraavilla selaimilla:

- Google Chrome v. 1
- Mozilla Firefox v. 2 ja v. 3
- Internet Explorer v. 7
- Opera v. 9

Oli mielenkiintoista havaita valtavat erot JavaScriptin suoritusnopeudessa eri selainten välillä. Google Chrome -selain pystyi suorittamaan tätä ohjelmistoa sulavimmin. Sen jälkeen tulivat Firefox, Opera ja Internet Explorer. Käytännössä sovellus vaatii niin paljon tehoa JavaScript-moottorilta, että vain Chrome suoriutui tehtävästä kiitettävästi. Suorituskyky kuitenkin vaihteli sovelluksen eri osa-alueiden välillä. Jotkin asiat toimivat esimerkiksi nopeammin Firefoxilla kuin Chromella. Kierrospuiden lataaminen ja muokkaaminen tuntui vaativan eniten tehoa.

Testikoneen suorituskyky ei, yllättävää kyllä, vaikuttanut odotetulla tavalla suoritusnopeuteen. Tietokone, jolla kehitin sovellusta, sisälsi vanhan Pentium 4 530 -prosessorin, 1Gt:n DDR-keskusmuistia ja 32-bittisen Windows XP Professional SP2 -käyttöjärjestelmän. Testasin sovellusta myös koulumme uusilla koneilla, jotka sisältävät Core 2 Duo E8300 -prosessorin, 4Gt DDR2-keskusmuistia ja 32-bittisen Windows Vista SP1 -käyttöjärjestelmän. Vaikka koneiden ikäero oli noin neljä vuotta, nopeudessa ei ollut merkittävää eroa. En mitannut nopeuseroja tarkasti, totesin ne vain silmämääräisesti. Eri koneiden välisen testauksen suoritin Chromella ja Firefoxilla.

Kattavan testauksen tekeminen ilman kunnollista testaussuunnitelmaa on hankalaa. Testauksen kattavuuden todistaminenkin on silloin mahdotonta. Ideaalitilanteessa tehtäisiin suunnitelma ja kunnollinen dokumentaatio testauksen aikana. Olisi hyvä, jos ainakin osa testaajista olisi ulkopuolisia eli henkilöitä, jotka eivät ole osallistuneet sovelluksen tekemiseen.

Tämän kaltaisessa yhden kehittäjän projektissa kunnollinen testaaminen on vaikeaa. Kun testaaja tietää, miten sovellus toimii ja miten se on tehty, niin hän tiedostamattaan käyttää sitä ”oikein” ja ehkä jopa välttelee virheitä. Kiireellisen aikataulun takia sovimme, että riittää kun tekemäni analysaattori päätyy samoihin lopputuloksiin kuin Javalla tehty analysaattori, käyttäen Kyngäkseltä saamiani esimerkkisarjaohjelmia.

7 YHTEENVETO

Opinnäytetyön tekemiseen kului yllättävän paljon aikaa. Osaltaan tämä johtui suunnitelmien muuttumisesta työn tekemisen aikana. Jouduin myös korjaamaan joitain virheitä, joita löysin aikaisemmin tekemästäni sarjaohjelman muokkaussovelluksesta. Suunnittelin analyysikomponentin ulkoasun kaksi kertaa, koska analysoitavia kohtia tuli suunniteltua enemmän. Tämän lisäksi työmäärää kasvatti se, että analysointi päätettiin tehdä JavaScriptillä. Aikaa kului myös erilaisten pienten työkalujen tekemiseen: Algoritmin luomat sarjaohjelmat tallennetaan tekstitiedostoon. Jotta niitä voi-

taisiin käyttää sovelluksessa, ne täytyy ensin siirtää tietokantaan. Sarjaohjelman asetukset taas täytyy muuttaa JSON-muotoon, että niitä voitaisiin käyttää. Siksi kirjoitin pienet apuohjelmat LUA-skriptikiellä.

Erityisen tyytyväinen olen JavaScript-ohjelmointitaitoni kehittymisestä. Arvostan paljon myös Ext JS -JavaScript-kirjaston käyttö- ja muokkauskokemusta. Yhä useammat sovellukset suunnitellaan nykyään selainkäyttöisiksi, ja ne käyttävät runsaasti JavaScriptiä. Uskon että minulle on paljon hyötyä tämän projektin aikana opituista taidoista.

Opinnäytetyön lopputulokseen olen suhteellisen tyytyväinen. En voinut suunnitella toiminnallista osuutta kovin pitkälle, koska työn tilaajakaan ei tiennyt tarkkaan, mitä halusi. Jos jotain pitäisi tehdä toisin, niin dokumentointia voisi parantaa. Samoin kunnollinen testausvaiheen suunnittelu olisi ollut hyvä idea.

Tulevaisuuden tarpeisiin vastaaminen on vaikeaa. Yritin suunnitella analysaattorin siten, että sitä olisi helppo laajentaa ja siihen voisi lisätä uusia urheilulajeja ilman, että ohjelmakoodiin tarvitsisi tehdä juurikaan muutoksia. Toivon että analysaattori vastaisi tarpeita ja että sen elinkaari olisi pitkä.

LÄHTEET

Extjs.com 2009. Ext JS – Manual [verkkodokumentti]. [viitattu 25.3.2009]. Saatavissa: http://extjs.com/learn/Manual:Intro

Json.org 2009. JSON [verkkodokumentti]. [viitattu 25.3.2009]. Saatavissa: http://www.json.org

Kyngäs J. & Nurmi K. 2009. Scheduling the Finnish Major Ice Hockey League. IEEE Symposium Series on Computational Intelligence in Scheduling, 84-89

Msdn 2009. ASP.NET Overview [verkkodokumentti]. [viitattu 25.3.2009]. Saatavissa: http://msdn.microsoft.com/en-us/library/4w3ex9c2.aspx

SQLite.org 2009a. About SQLite [verkkodokumentti]. [viitattu 11.3.2009]. Saatavissa: http://www.sqlite.org/about.html

SQLite.org 2009b. SQLite Is Self-Contained [verkkodokumentti]. [viitattu 11.3.2009]. Saatavissa: http://www.sqlite.org/selfcontained.html

SQLite.org 2009c. SQLite Is Serverless [verkkodokumentti]. [viitattu 11.3.2009]. Saatavissa: http://www.sqlite.org/serverless.html

SQLite.org 2009d. SQLite Features [verkkodokumentti]. [viitattu 11.3.2009]. Saatavissa: http://www.sqlite.org/features.html

SQLite.org 2009e. Appropriate Uses For SQLite [verkkodokumentti]. [viitattu 11.3.2009]. Saatavissa: http://www.sqlite.org/whentouse.html

Wikipedia 2009/a. Common Intermediate Language [verkkodokumentti]. [viitattu 25.3.2009]. Saatavissa: http://en.wikipedia.org/wiki/Common_Intermediate_Language

Wikipedia 2009/b. Ajax [verkkodokumentti]. [viitattu 25.3.2009]. Saatavissa: http://en.wikipedia.org/wiki/Ajax_(programming)

Wikipedia 2009/c. JSON [verkkodokumentti]. [viitattu 25.3.2009]. Saatavissa: http://en.wikipedia.org/wiki/Json