Kimmo Hannila

REAALIAIKAINEN FRISBEEGOLFTULOSSEURANTA

REAALIAIKAINEN FRISBEEGOLFTULOSSEURANTA

Kimmo Hannila

Opinnäytetyö

Syksy 2015

Tietotekniikan koulutusohjelma

Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma / Ohjelmistokehityksen suuntautumisvaihtoehto

Tekijä: Kimmo Hannila
Opinnäytetyön nimi: Reaaliaikainen frisbeegolftulosseuranta
Työn ohjaaja: Kari Jyrkkä
Työn valmistumislukukausi ja -vuosi: Syksy 2015
Sivumäärä: 30

Opinnäytetyön tavoitteena oli rakentaa sovellus frisbeegolfkisojen tulosten reaaliaikaiseen syöttämiseen ja seurantaan. Sovelluksella voidaan helpottaa kisajärjestäjien työtä korvaamalla paperiset tuloskortit mobiililaitteella tapahtuvalla tuloskirjauksella ja tarjota yleisölle kanava kisatulosten seuraamiseen internetin kautta sitä mukaa, kun tuloksia kirjataan.

Sovellus toteutettiin Javalla ja sen ympärille rakentuvilla teknologioilla NetBeans-kehitysympäristössä.

Projektin tuloksena syntyi tehokas ja luotettava sovellus, joka sisältää kisojen järjestämiseen tarvittavan perustoiminnallisuuden kuten pelaajien hallinnan sekä tulosten kirjaamisen ja katselun. Sitä voidaan myös laajentaa tulevaisuudessa tunnistettavien tarpeiden perusteella.

Asiasanat: Java, HTML, web-sovellus, frisbeegolf

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology / Software Development

---

Author: Kimmo Hannila
Title of thesis: Real time disc golf scorecard
Supervisor: Kari Jyrkkä
Term and year when the thesis was submitted: Autumn 2015
Pages: 30

Thesis work's objective was to build an application for disc golf competitions' real time scoring. The application eases the work of organizers by replacing paper scorecards with mobile devices and offers the audience a channel to follow the scoring.

The application was implemented in Java and related technologies, using NetBeans integrated development environment.

Outcome of the project was efficient and reliable system that includes the basic functionality needed to organize a competition like player management, scoring and leaderboard. It can be extended in future if needed.

Keywords: Java, HTML, web application, disc golf

ALKULAUSE

Kiitos Jori Löytynojalle mielenkiintoisesta aiheesta. Tämän työn parissa minulla oli mahdollisuus sekä syventää osaamistani että kokeilla uusia tekniikoita.

13.8.2015

Kimmo Hannila

SISÄLLYS

TIIVISTELMÄ 3
ABSTRACT 4
ALKULAUSE 5
SISÄLLYS 6
SANASTO 8
1 JOHDANTO 9
2 JÄRJESTELMÄN VAATIMUKSET 10
3 TYÖKALUT JA TEKNOLOGIAT 11
3.1 Java EE 11
3.2 WildFly 11
3.3 H2 11
3.4 Hibernate 12
3.5 JavaServer Faces 12
3.6 PrimeFaces 12
3.7 NetBeans 12
4 TOTEUTUS 14
4.1 Tietokanta 14
4.2 Palvelukerros 15
4.3 Käyttöliittymät 16
4.4 Suorituskyky 17
4.5 Tietoturva 18
5 TOIMINNALLISUUS 20
5.1 Käyttäjien ja pelaajien hallinta 20
5.2 Kisatapahtuman luonti 21
5.3 Pelaajien liittäminen kisatapahtumaan 22
5.4 Kierrosten ja väylien määrittely 23
5.5 Pelaajien sijoittaminen kierroskohtaisiin ryhmiin 24
5.6 Tulosten kirjaaminen 25
5.7 Tulosten reaaliaikainen seuranta 26

6 JATKOKEHITYSMAHDOLLISUUDET 28
7 YHTEENVETO 29
LÄHTEET 30

SANASTO

AJAX	Asynchronous JavaScript and XML, joukko web-sovelluskehityksen tekniikoita
CSS	Cascading Style Sheets, www-dokumenttien tyyliohjeiden laji
Frisbeegolf	Golfia muistuttava urheilulaji, jota pelataan liitokiekoilla
H2	Relaatiotietokanta
Hibernate	Ohjelmistokehys olio-relaatiomallinnukseen
Java EE	Ohjelmistokehitysalusta Java-sovelluksille
JavaScript	Dynaaminen komentosarjakieli web-ympäristöihin
JSF	Ohjelmistokehys web-käyttöliittymille
jQuery	Avoimen lähdekoodin JavaScript-kirjasto
MVC	Model-View-Controller, ohjelmistoarkkitehtuurityyli
NetBeans	Integroitu kehitysympäristö
PDGA	Professional Disc Golf Association, frisbeegolfin kansainvälinen kattojärjestö
PrimeFaces	Komponenttikirjasto JSF:lle
SQL	Structured Query Language, kyselykieli relaatiotietokannoille
Web-sovellus	Selaimella käytettävä ohjelmisto
WildFly	Java EE -spesifikaation toteuttava sovelluspalvelin

1 JOHDANTO

Frisbeegolf on perinteistä golfia muistuttava urheilulaji, jossa radalla edetään heittämällä kiekko maalikoriin ja vähiten heittoja käyttänyt pelaaja on voittaja. Lajin kasvu on ollut viime vuosina räjähdysmäistä, erityisesti Yhdysvalloissa ja Pohjoismaissa. (1.)

Frisbeegolfkisoissa rata kierretään 3–5 pelaajan ryhmissä. Rata koostuu yleensä 18 väylästä ja kierroksia pelataan yksi tai useampia, kisan tasosta riippuen.

Tulosseuranta tapahtuu kirjaamalla pelaajien heitot paperisiin tuloskortteihin, joista ne siirretään kisatoimistossa digitaaliseen muotoon, tavallisesti kerran tai kaksi kierroksen aikana. Tämä menettely ei ole kisojen seuraamisen kannalta kovin käytännöllistä. Usein lopulliset tulokset ovat nähtävissä netissä vasta useita tunteja kisan päättymisen jälkeen.

Eri mobiilialustoille on olemassa runsaasti henkilökohtaiseen käyttöön suunnattuja tuloskorttisovelluksia, mutta täysimittaisen kisan tarpeet täyttävää sovelluksia tai järjestelmiä on hyvin vähän, yhtenä esimerkkinä virolainen Skoorin.com. Opinnäytetyön tavoitteena olikin tehdä tulosseurantajärjestelmä, joka helpottaisi järjestäjien työtaakkaa korvaamalla paperiset tuloskortit ja osallistujalistat sekä mahdollistaisi yleisölle tulosten reaaliaikaisen seuraamisen internetin kautta.

2 JÄRJESTELMÄN VAATIMUKSET

Sovellusta hahmoteltaessa oli selvää, että sen tulisi olla alustariippumaton web-sovellus, sillä sen haluttiin toimivan kaikenlaisilla laitteilla.

Sovellus piti jakaa kahteen osioon, julkiseen tulosseurantasivustoon ja kisajärjestäjille suunnattuun sivustoon, joka vaatii sisäänkirjautumisen. Tietoturvamielessä oli toteutettava roolipohjainen käyttäjienhallinta ja huolehdittava, että käyttäjillä on pääsy ainoastaan roolin oikeuttamiin näkymiin ja toimintoihin.

Suorituskykyyn piti kiinnittää erityistä huomiota, sillä kisoissa voi olla jopa viisisataa pelaajaa, viisi kierrosta ja kahdeksantoista reikää kierrosta kohti. Näin ollen yhden kisan aikana syntyy 45 000 tulosta, jotka täytyy pystyä esittämään tulosseurantasivulla ilman järjestelmän hidastumista.

Käyttöliittymän tuli olla helppokäyttöinen ja ominaisuuksiltaan moderni. Suurin osa sovelluksen toiminnoista on kisajärjestäjien käytössä, mutta tulosten kirjaaminen on enimmäkseen satunnaisten kisa-apulaisten tai pelaajien varassa. Koska kisojen puitteissa ei ole aikaa eikä resursseja järjestää laajamittaista koulutusta sovelluksen toimintaan, on tärkeää, että sen käyttö on intuitiivista ja onnistuu minimaalisella perehdytyksellä.

3 TYÖKALUT JA TEKNOLOGIAT

Toteutuksessa hyödynnettiin osittain tuttuja välineitä, joilla työskentely on minulle tuttua, kuten JavaEE ja NetBeans, sekä sellaisia, joiden osaamisen syventäminen oli paikallaan (JSF, PrimeFaces ja WildFly), ja muutamia minulle ennestään tuntemattomia välineitä (Hibernate ja H2). Kaikki työssä käytetyt välineet ovat ilmaisia ja osa niistä perustuu avoimeen lähdekoodiin.

3.1 Java EE

Java EE on ohjelmistokehys, joka laajentaa Java SE -alustaa tarjoamalla useita verkkoympäristössä ja monitasorakenteisissa sovelluksissa tarvittavia ohjelmointirajapintoja (2, s.1–6). Tässä projektissa hyödynnettiin erityisesti Java EE:n JPA-rajapintaa tietokannan käsittelyssä.

Sovelluksen olisi voinut toteuttaa myös muilla alustoilla, esimerkiksi PHP:llä tai ASP.NETillä, mutta valitsin Javan, koska halusin tutustua tarkemmin JSF:ään ja Hibernateen.

3.2 WildFly

Red Hatin kehittämä WildFly on moderni sovelluspalvelin joka toteuttaa Java EE –spesifikaation (3).

Javalle on saatavilla yli kaksikymmentä sovelluspalvelinta, joista yhdeksällä on vapaan käytön mahdollistava lisenssi. Näistä ainoastaan kaksi toteuttaa uusimman Java EE 7 -spesifikaation kokonaisuudessaan: GlassFish ja WildFly. Arvioinnin perusteella valituksi tuli WildFly, koska siinä on mukana H2-tietokanta ja Hibernate-kirjastot. Myös sen käyttöliittymä ja konfigurointi ovat GlassFishiin verrattuna miellyttävämpiä.

3.3 H2

H2 on erittäin kevyt ja nopea relaatiotietokanta. Ominaisuuksiin kuuluvat mm. transaktiot, muistinvaraiset taulut, tuki klusteroinnille eli hajautetulle tiedon käsit-

telylle ja tiedonsalaus sekä levyllä että verkon yli. H2 asentuu WildFlyn mukana ja on siten helppo ottaa käyttöön. Muilta ominaisuuksiltaan se ei häviä yleisimmille vapaille tietokannoille, kuten Derby, MySQL tai PostgreSQL. (4.)

3.4 Hibernate

Red Hatin Hibernate on yksi yleisimpiä ohjelmistokehyksiä relaatiomallinnukseen. Se automatisoi olioiden tallennusta relaatiotietokantaan ja eliminoi SQL:n kirjoittamisen tarpeen käytännössä kokonaan. (5.)

3.5 JavaServer Faces

JavaServer Faces on JavaEE:n ohjelmistokehys web-käyttöliittymien rakentamiseen. JSF:n sisäänrakennettu MVC (Model-View-Controller) –malli ja komponenttikeskeinen lähestymistapa tekevät siitä sekä yksinkertaisen että ilmaisuvoimaisen. (6.)

3.6 PrimeFaces

Prime Technologyn kehittämä PrimeFaces on JQueryyn pohjautuva komponenttikirjasto JSF:lle, joka tarjoaa runsaan valikoiman ominaisuuksiltaan monipuolisia käyttöliittymäkomponentteja sekä AJAX- ja teematuen (7). Muita vastaavia kirjastoja ovat muun muassa ICEFaces, RichFaces ja OpenFaces. Valitsin näistä PrimeFacesin, koska sillä on vahva yhteisö, joka helpottaa ongelmatilanteiden ratkomista.

3.7 NetBeans

NetBeans on Integroitu ohjelmointiympäristö (integrated development environment, IDE), joka tukee Javan lisäksi useita ohjelmointikieliä, mm. PHP, C/C++ ja HTML5. Ympäristö itsessään on tehty Javalla ja tarvitsee siten JRE:n toimiakseen. NetBeans on modulaarinen ja sille on saatavissa runsaasti kolmannen osapuolen liitännäisiä. (8.)

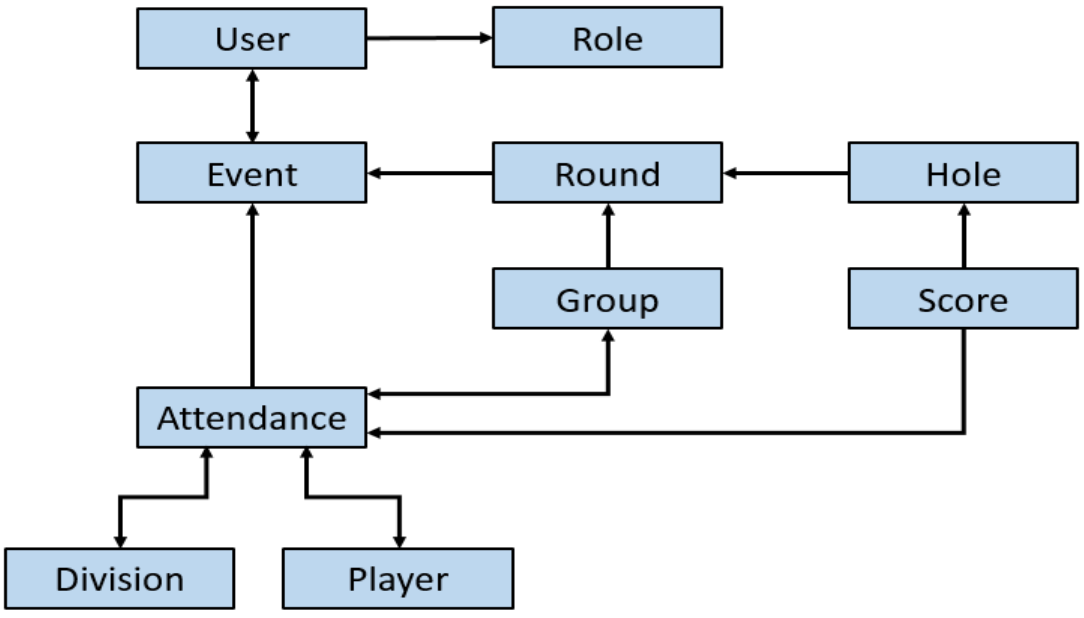
NetBeans on kypsä ja monipuolinen sekä soveltuu hyvin monenlaisiin ohjelmistoprojekteihin. Hyvä vaihtoehto sille on Eclipse, ja valinta niiden välillä on lähinnä makuasia.

4 TOTEUTUS

Sovelluksen toteutuksessa lähdettiin liikkeelle tietomallin ja samalla tietokannan suunnittelusta. Sen jälkeen rakennettiin liiketoimintalogiikan runko johon kuuluivat muun muassa palvelukerroksen ja käyttöliittymien kantaluokat sekä näyttöjen sommittelu. Kun runko oli valmis, voitiin siirtyä yksittäisten toimintojen toteuttamiseen kisatapahtuman luonnista aloittaen. Toimintoja voi ajatella eräänlaisina toisistaan riippumattomina putkina, jotka alkavat käyttöliittymästä ja päättyvät tietokantaan, ja niitä toteuttaessa täytyikin kirjoittaa koodia kaikkiin sovelluksen kerroksiin.

4.1 Tietokanta

Tietokantana toimii H2, joka on WildFlyn mukana toimitettava relaatiotietokanta. Projektin iteratiivisesta luonteesta johtuen tietomalli muuttui sovellusta kehitettäessä useaan otteeseen kunnes saavutti lopullisen muotonsa (kuva 1). Hibernate helpotti työtaakkaa huomattavasti, ja ilman sitä kaikki tietomallin muuttuvia osia koskevat SQL-lausekkeet olisi joutunut kirjoittamaan uudelleen.



KUVA 1. Sovelluksen tietomalli

Tietomalli noudattaa kolmatta normaalimuotoa eli taulujen ei-avainkentät riippuvat avainkentistä (9). Suorituskyvyn kannalta harkitsin osallistujien (Attendance) tulosten (Score) käsittelemistä kierroskohtaisesti atomisena eli jakamattomana arvona, jolloin ne olisi nopeampi ladata tietokannasta. Päädyin kuitenkin sijoittamaan yksittäiset tulokset omaan tauluunsa kahdesta syystä:

- Suorituskykyä voidaan optimoida muilla keinoin ja säilyttää kolmas normaalimuoto, ks. luku 4.4 Suorituskyky.
- Tulosten pitäminen omassa taulussaan mahdollistaa muiden tietojen helpon lisäämisen kyseiseen tauluun ja sitä vastaavaan entiteettiin, esimerkiksi puttien lukumäärän.

4.2 Palvelukerros

Sovelluksen toimintalogiikan keskeisimmät toiminnot ovat palvelukerroksen luokissa. Keskeisiksi luokiteltavia toimintoja ovat kaikki sellaiset metodit, joita kutsutaan useammalta kuin yhdeltä näytöltä, sekä ne, jotka ovat vähänkään monimutkaisempia, kuten tuloslaskennan algoritmit sekä tiedon hakuun ja tallennukseen liittyvät metodit. Tyypillisenä esimerkkinä voidaan pitää sisäänkirjautumisen yhteydessä suoritettavaa käyttäjän hakua. Metodi readByLoginname suorittaa Hibernatea hyödyntäen tietokantakyselyn käyttäjätauluun käyttäjätunnuksen perusteella ja palauttaa sovelluksessa käyttäjää mallintavan luokan, jos sellainen löytyy (kuva 2).

```
package dgscore.repository;

import dgscore.entity.UserEntity;
import javax.ejb.Stateless;
import javax.enterprise.context.Dependent;
import javax.persistence.NoResultException;

@Stateless
@Dependent
public class UserRepository extends Repository<UserEntity> {

    public UserRepository() {
        super(UserEntity.class);
    }

    public UserEntity readByLoginname(String loginname) {
        try {
            return (UserEntity) getEntityManager()
                    .createQuery("SELECT u FROM UserEntity u WHERE u.loginname = :loginname")
                    .setParameter("loginname", loginname)
                    .getSingleResult();
        } catch (NoResultException nre) {
            return null;
        }
    }
```

KUVA 2. Esimerkki palvelukerroksen luokasta

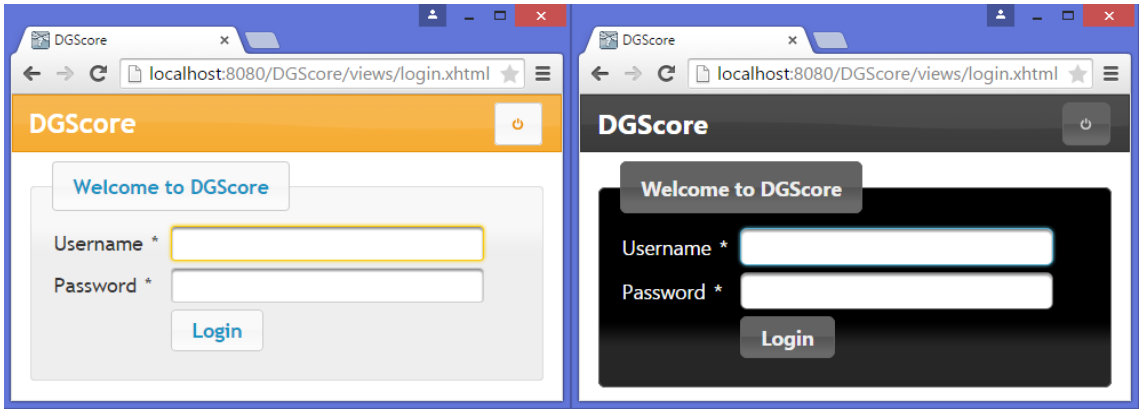
4.3 Käyttöliittymät

Käyttöliittymät toteutettiin JSF:llä ja PrimeFaces-komponenttikirjastoa vahvasti hyödyntäen. Suunnittelussa kiinnitettiin erityistä huomiota selkeyteen ja helppokäyttöisyyteen. Vaikka käyttöliittymän ulkonäkö oli tässä sovelluksessa toissijainen seikka, valitut teknologiat eivät rajoita sen muokkaamista kaupallisempaan suuntaan. PrimeFaces tukee jQuery UI:n ThemeRoller-teemoja, ja osaava graafikko pystyy muokkaamaan ulkoasua vapaasti CSS:llä.

JSF yhdessä PrimeFacesin kanssa on varsin ilmaisuvoimainen tapa toteuttaa käyttöliittymiä. Puhtaaseen HTML:ään verrattuna koodia tarvitaan varsin vähän (kuva 3), ja käyttöliittymän ilme voidaan helposti vaihtaa toisenlaiseksi joko valmiilla tai itse tehdyillä teemoilla (kuva 4).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
      xmlns:p="http://primefaces.org/ui">
    <h:body>
        <ui:composition template="/views/common/layout.xhtml">
            <ui:define name="caption">Welcome to DGScore</ui:define>
            <ui:define name="main">
                <h:form>
                    <p:focus/>
                    <h:panelGrid columns="2" class="separated">
                        <p:outputLabel for="loginname" value="Username"/>
                        <p:inputText id="loginname" value="#{loginView.loginname}" required="true"/>
                        <p:outputLabel for="password" value="Password"/>
                        <p:password id="password" value="#{loginView.password}" required="true"/>
                        <p:outputLabel for="loginCommand" value=""/>
                        <p:commandButton id="loginCommand" value="Login" action="#{loginView.login()}" update="@form"/>
                    </h:panelGrid>
                </h:form>
            </ui:define>
        </ui:composition>
    </h:body>
</html>
```

KUVA 3. Kirjautumisnäytön lähdekoodi



KUVA 4. Kirjautumisnäyttö kahdella eri teemalla

4.4 Suorituskyky

Merkittävimmän haasteen sovelluksen suorituskyvylle tuo kisatuloksista kiinnostuneiden seuraajien määrä ja sitä kautta tietokannan lukuoperaatioiden tiheys. Tätä kuormitusta vähennettiin välimuistin kaltaisella tietorakenteella joka säilyy palvelimen muistissa. Rakenne sisältää kokonaisten kisojen tulostaulukoita ja niihin on laskettu valmiiksi pelaajien raakatuloksista kisa- ja kierroskohtaiset tulokset ja erot ihannetuloksiin. Välimuistissa olevaa kisatulostaulukkoa päivitetään aina kun tuloksia syötetään ja se poistetaan, mikäli sitä ei päivitetä tai kysytä valitun ajan, esimerkiksi tunnin, kuluessa.

Tässä ratkaisussa lähdettiin siitä olettamuksesta, että kisatuloksia luetaan moninkertaisesti useammin kuin niitä tallennetaan. On siis edullisempaa laskea tulokset tallennuksen yhteydessä, ja samalla vältetään luvun aiheuttamat raskaat tietokantaoperaatiot.

4.5 Tietoturva

Sovelluksen tunnistus- ja valtuutusmenetelmät toteutettiin Java EE –standardien mukaisesti (10). Alusta tarjoaa palvelut käyttäjien tunnistamiseen tietokantaa vasten sekä URL-pohjaisen pääsynhallinnan, ja niiden lisäksi sovelluksen sisälle on ohjelmoitu yksittäisten toimintojen suorittamiseen liittyvät tarkistukset.

Esimerkkinä yksittäisen toiminnon tarkistuksesta voidaan ajatella seuraavaa tilannetta. Käyttäjä K (tunnistettu) on sivulla S (valtuutettu) muokkaamassa kisaa A. Hän huomaa selaimen URL-kentässä entiteetin tunnisteen muodossa ”eventId=A”. Käyttäjä kokeilee päästä muokkaamaan toista kisaa, joka ei näkynyt hänelle edellisen sivun valintalistassa, vaihtamalla URLissa olevan tunnisteen muotoon ”eventId=B”, arvellen että sellainen saattaa olla olemassa. Kisa on olemassa, mutta koska käyttäjää ei ole lisätty kisaan, hänet ohjataan virhesivulle ja laittomasta toiminnosta jää merkintä sovelluksen lokiin (kuva 5).

Vastaavia tarkistuksia tehtiin kaikkiin sovelluksessa käsiteltäviin objekteihin, metodeihin ja sivuihin.

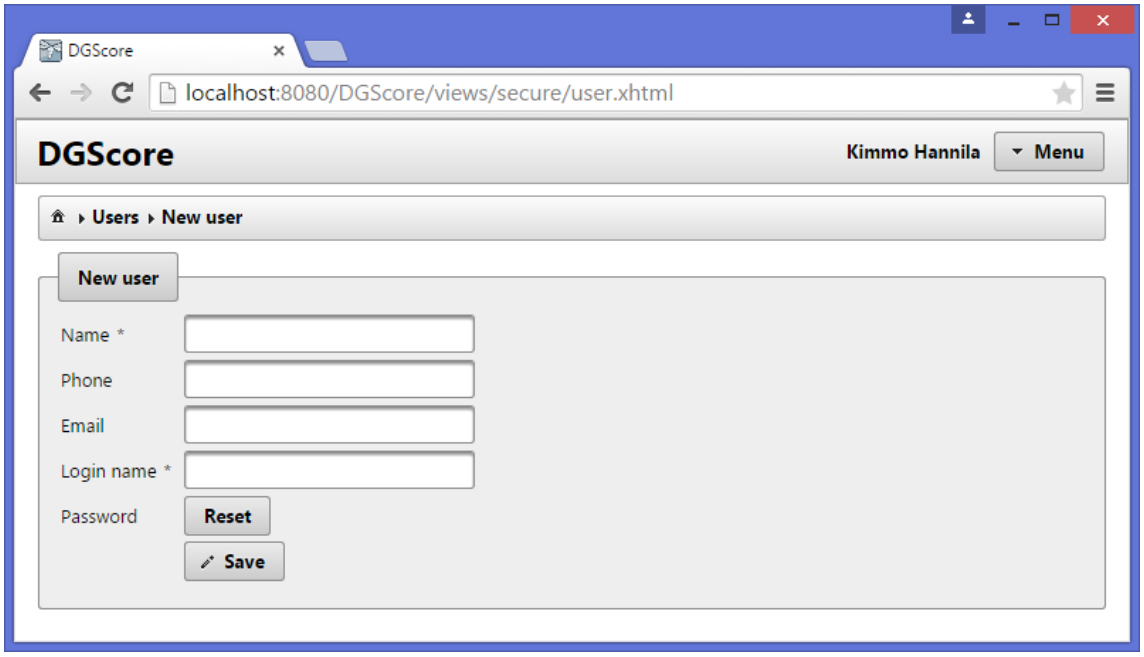
```
@PostConstruct
public void initializeCallback() {
    try {
        String view = this.getClass().getName();
        if (!currentUser.hasAccess(view)) {
            //Throw exception if no access.
            throw new UnauthorizedException("View '" + view + "'");
        }
        initialize();
    } catch (Exception e) {
        String errorPage = "error.xhtml";
        if (e instanceof NotFoundException) {
            errorPage = "notfound.xhtml";
        } else if (e instanceof UnauthorizedException) {
            errorPage = "unauthorized.xhtml";
        }
        logger.warn("User '{}' caused {}", currentUser.getUserLoginname(), e.toString());
        try {
            redirect("/views/error/" + errorPage);
        } catch (IOException ioe) {
            growl(ioe);
        }
    }
}
```

KUVA 5. Pääsynhallinnan tarkistus näkymien kantaluokassa

5 TOIMINNALLISUUS

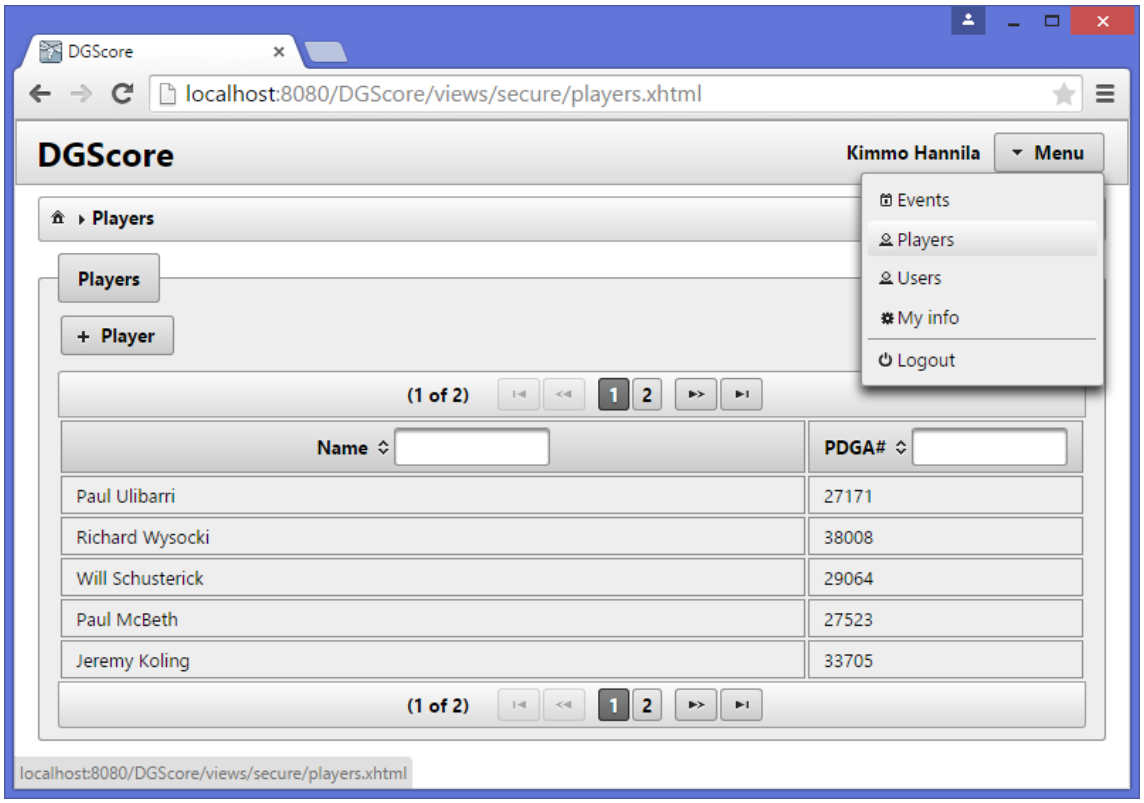
5.1 Käyttäjien ja pelaajien hallinta

Sovelluksen pääkäyttäjä luodaan tietokantaan järjestelmän asennuksen yhteydessä. Muut käyttäjät luodaan järjestelmään käyttöliittymän kautta (kuva 6).



KUVA 6. Käyttäjän luonti ja muokkaus

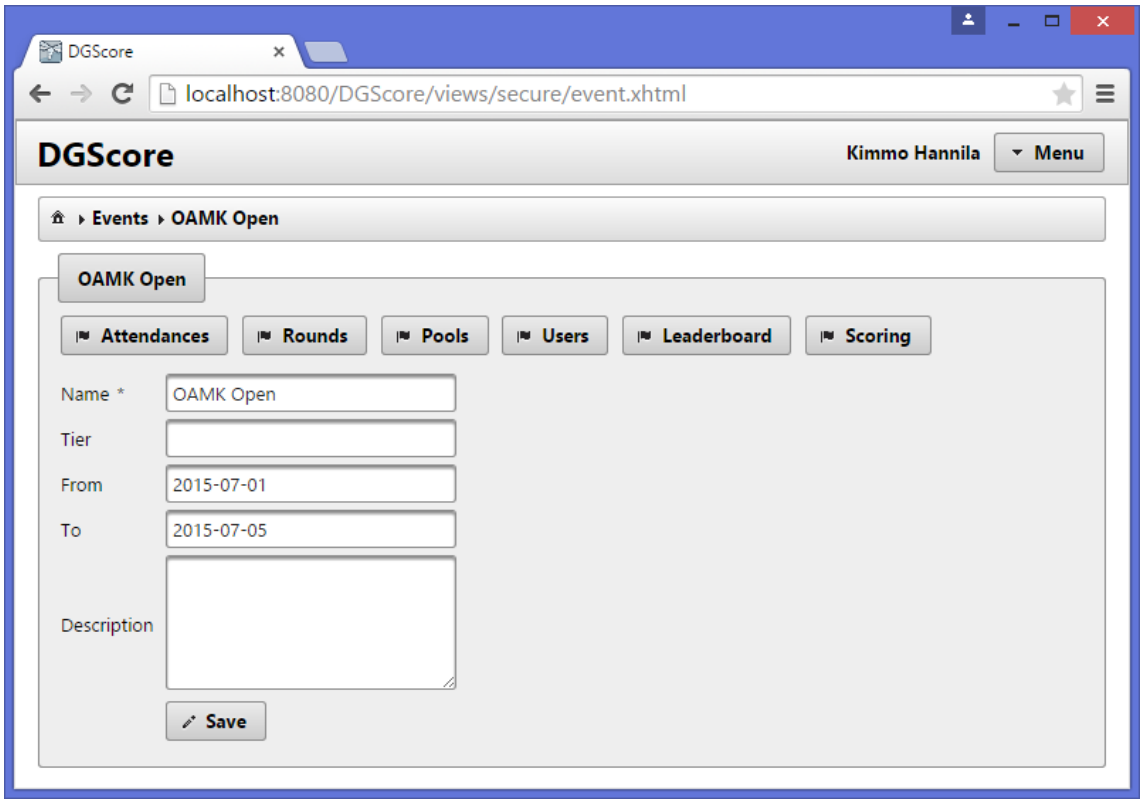
Pelaajat voitaisiin tuoda pääkäyttäjän käynnistämänä ajona muista järjestelmistä, kuten PDGA:n sivustolta, mutta sovellus mahdollistaa nykyisellään yksittäisten pelaajien luonnin ja muokkauksen (Kuva 7). Ominaisuudesta voi olla hyötyä esimerkiksi yksittäisen seuran tarpeisiin tehdyssä pienemmässä asennuksessa, jossa pelaajilla ei ole kisalisenssejä eikä heitä siten löydy PDGA:n tietokannasta.



KUVA 7. Pelaajalista

5.2 Kisatapahtuman luonti

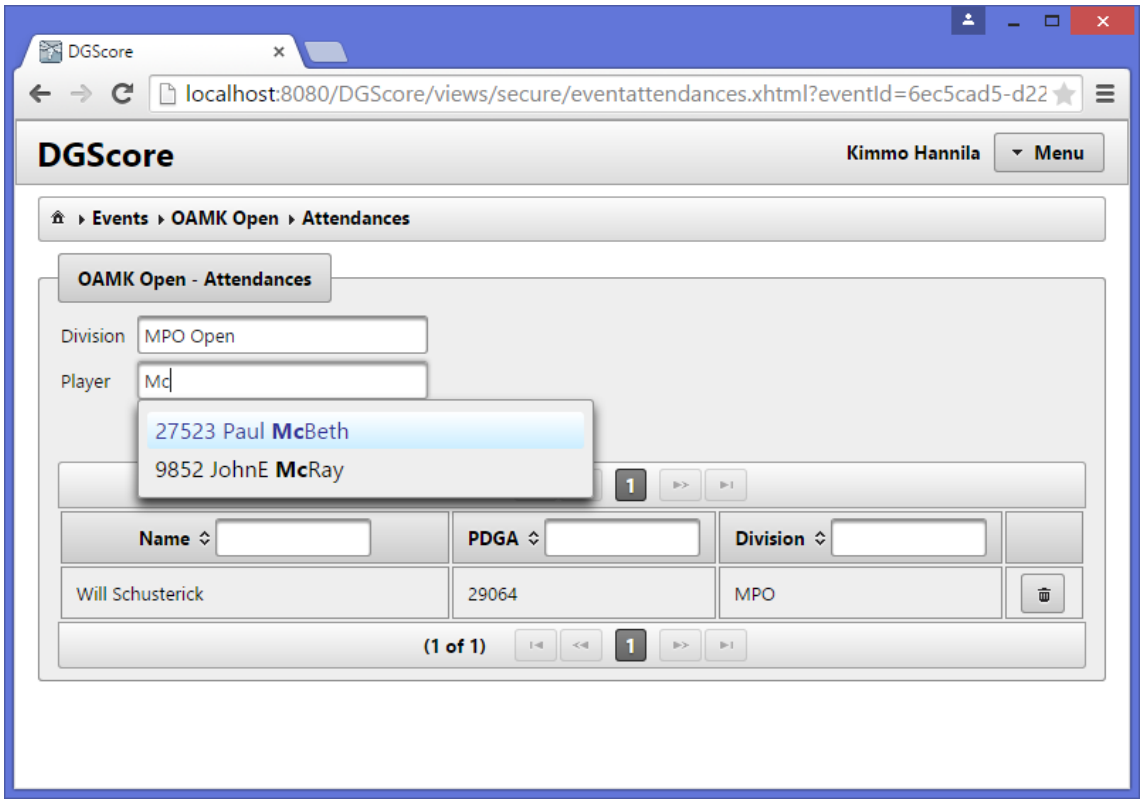
Perustietojen syöttäminen on ensimmäinen toiminto, joka tehdään kisaa luotaessa. Kun perustiedot on syötetty ja tallennettu, avautuvat muut kisaan kohdistuvat toiminnot. (Kuva 8.)



KUVA 8. Kisatapahtuma

5.3 Pelaajien liittäminen kisatapahtumaan

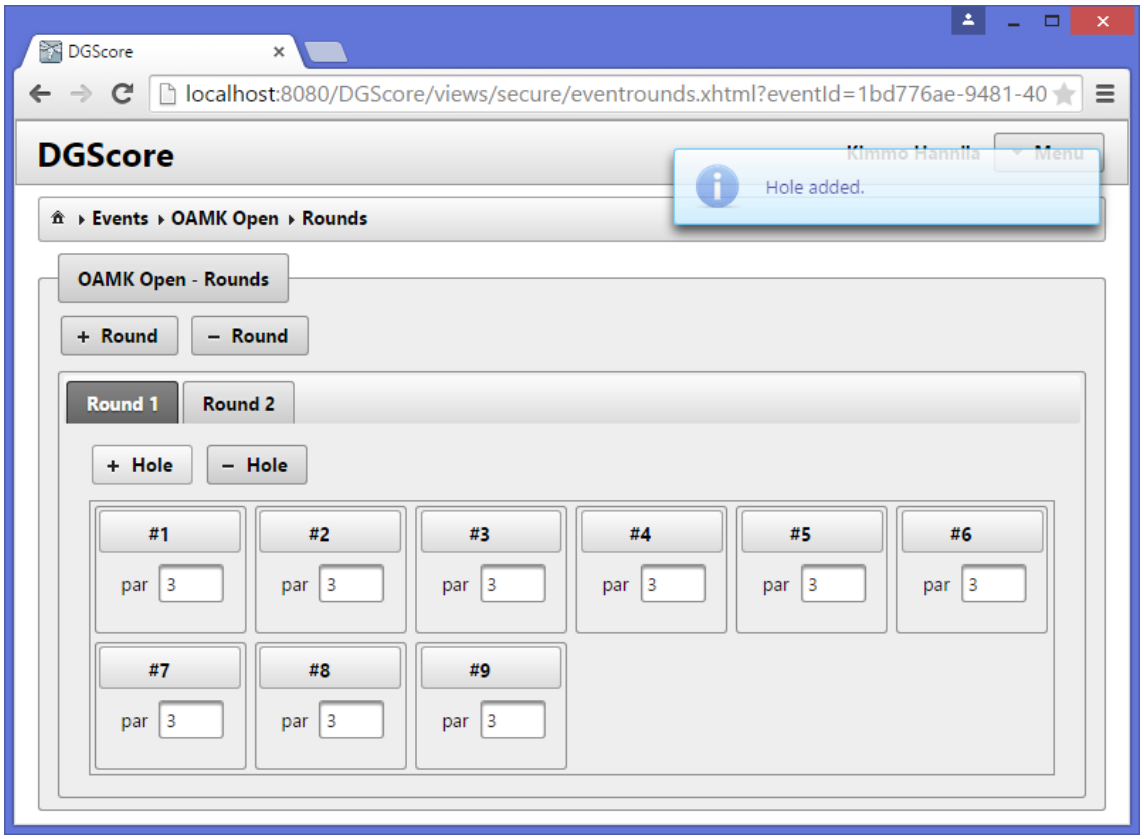
Pelaaja osallistuu kisaan aina jossain divisioonassa. Pelaajat, divisioonat ja kisat ovat erillisiä entiteettejä, ja niiden yhdistämiseksi luodaan ”osallistuminen”, joka tallennetaan niin ikään omaan tauluunsa (kuva 9). Tietokannasta voi näin ollen selvittää, mihin kisoihin tietty pelaaja on osallistunut, missä divisioonissa ja millä lopputuloksella.



KUVA 9. Pelaajien liittäminen kisatapahtumaan

5.4 Kierrosten ja väylien määrittely

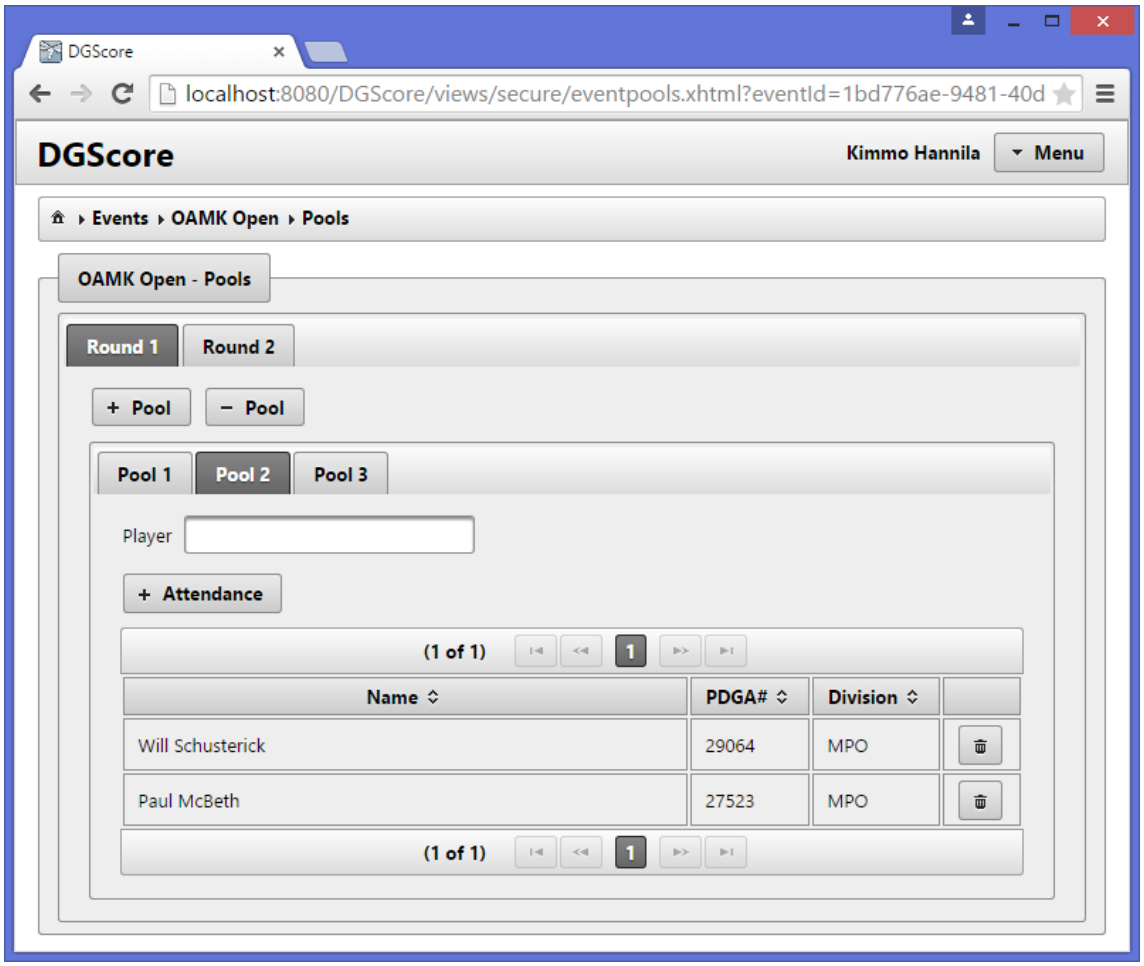
Kisaan määritellään kierrokset ja niihin väylät ihannetuloksineen (par). Väylien ihannetuloksia käytetään tuloslaskennassa tuloksien vertailulukuina, ja pelaajan tulos suhteessa ihannetulokseen (to par) on lajissa olennainen ja kiinnostava tieto. (Kuva 10.)



KUVA 10. Kierrokset ja väylät

5.5 Pelaajien sijoittaminen kierroskohtaisiin ryhmiin

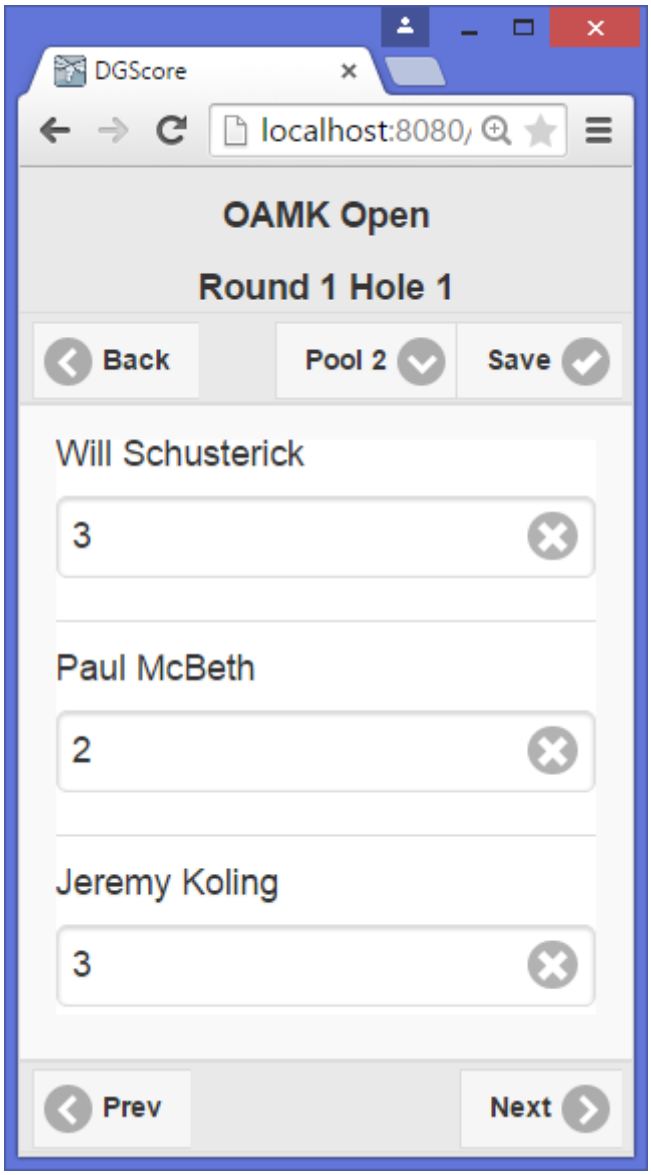
Jokaiselle kierrokselle pelaajat jaetaan ryhmiin, joissa he kiertävät radan. Ryhmien luonti ja pelaajien sijoittaminen tapahtuu manuaalisesti, mutta sitä voitaisiin jatkossa automatisoida, ks. luku 6 Jatkokehitysmahdollisuudet. (Kuva 11.)



Kuva 11. Pelaajien sijoittaminen

5.6 Tulosten kirjaaminen

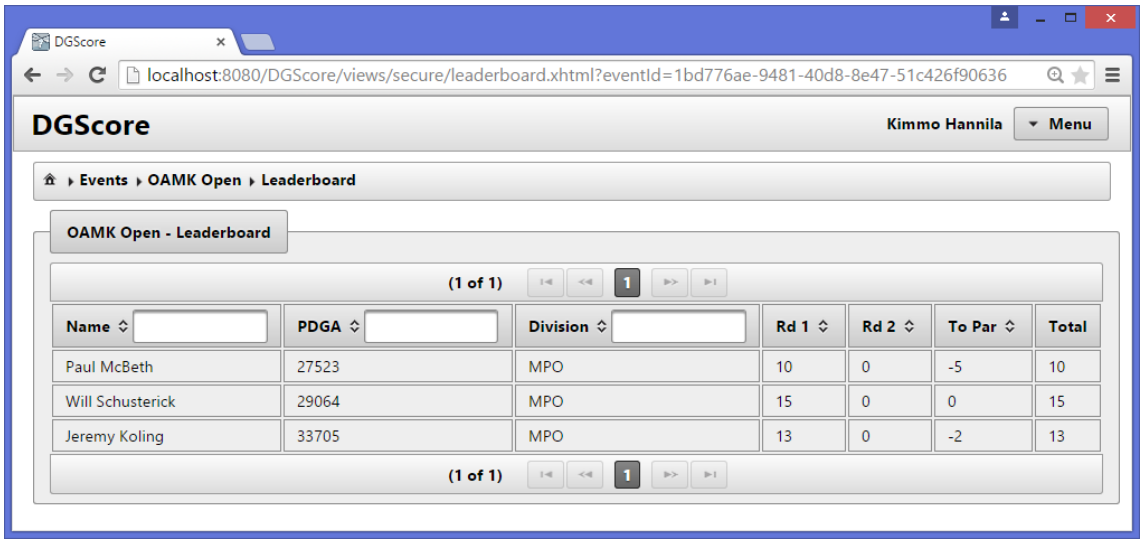
Tulosten kirjaamiseen tarkoitetut näytöt on optimoitu erityisesti pienille näytöille jotta kirjaaminen onnistuisi helposti esimerkiksi älypuhelimella. Käyttäjä valitsee kierroksen, väylän ja ryhmän joille tuloksen kirjataan. Käyttäjä voi seurata ryhmää radalla ja siirtyä sovelluksessa helposti väylältä seuraavalle tai pysyä yhdellä väylällä ja vaihtaa ryhmää sen mukaan, mikä ryhmä kyseiselle väylälle saapuu. (Kuva 12.)



KUVA 12. Tulosten kirjaaminen

5.7 Tulosten reaaliaikainen seuranta

Sovelluksessa on julkinen sivusto, jolta kisan etenemistä voi seurata. Sivustolla on esillä sovellukseen perustetut kisat tietoineen sekä jokaiselle kisalle reaaliaikaisesti päivittyvä tulostaulukko. Tulostaulukon voi siten jättää selaimeen auki ja tarkkailla kisan etenemistä ilman että sivua tarvitsee ladata uudestaan. (Kuva 13.)



KUVA 13. Reaaliaikainen tulostaulukko

6 JATKOKEHITYSMAHDOLLISUUDET

Toteutetun sovelluksen oli tarkoitus sisältää kisojen järjestämiseen tarvittava perustoiminnallisuus. Projektin loppupuolella tilaajan kanssa käytyjen keskustelujen perusteella tunnistettiin muutamia selkeästi hyödyllisiä ominaisuuksia joita sovellukseen olisi hyvä lisätä.

Pelaajat olisi hyvä pystyä tuomaan sovellukseen massana, joko migraationa eli sisään lukuna valmiista aineistosta tai hakuna jostain ulkopuolisesta järjestelmästä. PDGA:lla on kisalisenssin omaavista pelaajista ajantasainen tietokanta, josta haku voitaisiin ehkä tehdä.

Pelaajien jakaminen kierroskohtaisiin ryhmiin tapahtuu yksitellen ja on varsin työlästä mikäli pelaajia on paljon. Jakoa voitaisiin automatisoida lisäämällä toiminto, joka jakaa pelaajat ryhmiin käyttäjän tekemien valintojen mukaan, esimerkiksi ensimmäiselle kierrokselle satunnaiseen järjestykseen ja sen jälkeen divisioonittain ja tuloksen mukaan joko nousevaan tai laskevaan järjestykseen. Myös lähtölistojen generointi tulostinystävälliseen muotoon voitaisiin liittää tähän yhteyteen.

Sovelluksessa on nykyisellään kaksi käyttäjäroolia, pää- ja tavallinen käyttäjä. Tulosten kirjaamista varten olisi hyvä olla kolmas rooli, ”kirjuri”, sillä suurissa kisoissa toimitsijoiden apuna on talkoolaisia, joille ei haluta antaa pääsyä muualle kuin tulosten kirjaamiseen.

7 YHTEENVETO

Opinnäytetyön tavoitteena oli rakentaa sovellus, joka mahdollistaa frisbeegolf-kisojen tulosten syöttämisen mobiililaitteella ja tarjoaa mahdollisuuden seurata tuloksia reaaliaikaisesti.

Projektin edetessä kävi nopeasti ilmi, kuinka paljon muita ominaisuuksia ja toimintoja tulosten syötön lisäksi tarvitaan, että palvelua pystyy käyttämään suunniteltuun tarkoitukseen. Siinä mielessä työn laajuus pääsi hieman yllättämään.

Käyttöliittymien toteutuksessa vastaan tuli muutamia JSF:n ja PrimeFacesin piirteitä, joiden kanssa kului arvioitua enemmän aikaa. Apua ongelmien selvittämiseen sai PrimeFacesin foorumilta.

Työn haastavin ja samalla antoisin alue oli Hibernaten sisäisen toiminnan ymmärtäminen. Olen aiemmin käsitellyt tietokantoja puhtaalla SQL:llä, ja siihen verrattuna erilaisen toimintamallin omaksuminen teetti töitä. Uskon että tästä tietämyksestä on jatkossa paljon hyötyä, sillä Hibernate on varsin laajasti käytössä.

LÄHTEET

1. What is Disc Golf? 2015. PDGA. Saatavissa: http://www.pdga.com/introduction. Hakupäivä 19.1.2015.
2. Gupta, Arun. 2013. Java EE 7 Essentials. Sebastopol, CA: O'Reilly Media, Inc.
3. What is WildFly? 2015. WildFly.org. Saatavissa: http://wildfly.org/about. Hakupäivä 20.2.2015.
4. H2 Database Engine. 2014. H2Database.com. Saatavissa: http://www.h2database.com/html/main.html. Hakupäivä 26.12.2014.
5. Hibernate ORM. 2014. Hibernate.org. Saatavissa: http://hibernate.org/orm. Hakupäivä 27.12.2014.
6. JavaServer Faces. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/JavaServer_Faces. Hakupäivä 10.4.2015.
7. PrimeFaces. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/PrimeFaces. Hakupäivä 10.4.2015.
8. NetBeans. 2015. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/NetBeans. Hakupäivä 10.4.2015.
9. Third normal form. 2014. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Third_normal_form. Hakupäivä 20.11.2014.
10. Securing Web Applications. 2015. Oracle. Saatavissa: http://docs.oracle.com/javaee/7/tutorial/security-webtier002.htm. Hakupäivä 1.6.2015.