Opinnäytetyö (AMK)

Tietojenkäsittelyn koulutusohjelma

Tietojärjestelmät

2014

Antton Alkio

KEHITYKSENSEURANTA-JÄRJESTELMÄ GOLF-HARJOITTELUUN

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Tietojärjestelmät

Kesäkuu 2014 | 36 sivua

Anne Jumppanen

Antton Alkio

KEHITYKSENSEURANTAJÄRJESTELMÄ GOLF-HARJOITTELUUN

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa virtuaalinen kehityksenseurantajärjestelmä golf-harrastajien ja valmentajien väliseen kanssakäymiseen. Ohjelmaan syötetään tietoa harjoituksista sekä kilpailuista, joista kootaan erilaisia tilastoja niin pelaajalle kuin valmentajalle. Opinnäytetyö tehdään toimeksiantona Auragolf ry:n alaisuudessa toimivalle Ville Sirkiä Tmi:lle.

Työn teoreettinen osuus koostuu keskeisten teknologioiden ja niiden ominaisuuksien läpikäymisestä. Sovelluksen palvelinpuoli toteutetaan Microsoftin ASP.NET MVC –ohjelmistokehyksen päälle ja asiakaspuoli taas jQuery Mobilea hyväksikäyttäen. Sivusto sekä tietokanta sijoitetaan Microsoftin Azure pilvipalveluun. Työssä tutustutaan myös muihin moderneihin teknologioihin ja palveluihin.

Käytännön osuus pitää sisällään itse sovelluksen toteutuksen. Sovellus on jaettu karkeasti kahteen puoleen: valmentajan ja pelaajan. Työssä perehdytään tarkemmin ainoastaan pelaajan puoleen.

Lopputuloksena syntyi halutunlainen ja toimiva sovellus, joka on ollut jo jonkin aikaa käytössä ja jota aiotaan kehittää lisää tulevaisuudessa.

ASIASANAT:

ASP.NET, MVC, jQuery Mobile, EF

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology | Information Systems

June 2014 | 36 pages

Anne Jumppanen

Antton Alkio

DEVELOPMENT OF A MONITORING APPLICATION FOR GOLF

The purpose of this Bachelor's thesis was to design and implement a monitoring application for golf. Information from practices and tournaments is entered into the system which calculates different statistics for players and coaches. This application was commissioned by Ville Sirkiä Tmi which operates under Auragolf ry.

The theoretical part of this thesis consists of explaining crucial technologies used in this project and their properties. The server-side of the application was implemented using the Microsoft ASP.NET MVC -framework and the client-side using the jQuery Mobile -framework. The website and database is located in the Azure cloud-service. Other modern technologies are also introduced in this work.

The practical part includes implementing the application itself. This application is divided in two sections: players and coaches. This work only explains the players' section.

The end result was a working program which has been in use for some time and will be further developed.

KEYWORDS:

ASP.NET, MVC, jQuery Mobile, EF

SISÄLTÖ

KUVAT

KUVIOT

1 JOHDANTO

Opinnäytetyön tavoitteena on tuottaa pelaajan kehitystä seuraava ohjelmisto golf-pelaajille sekä valmentajille. Nykyisellään kehitystä analysoidaan lähinnä valmentajan omien huomioiden sekä kisatulosten perusteella. Seurantamalli jättää paljon tilaa inhimillisille virheille eikä valmentajalla ole aina edes mahdollisuutta päästä seuraamaan suoritusta henkilökohtaisesti, jolloin ainoa analysointi suoritetaan pelaajan omien arviointien mukaan.

Ohjelmiston tilaaja on Auragolf ry:n alaisuudessa toimiva Ville Sirkiä Tmi. Sirkiä on toiminut ammattilaisena vuodesta 2002 ja suorittaa parhaillaan Suomen Golfliiton ja PGA:n 4. vaiheen Pelaajan Kehittäminen -tutkintoa. Sirkiä on erikoistunut juniorivalmennukseen ja hän onkin ollut mukana junioreiden maajoukkueen ringissä vuodesta 2005 lähtien.

Ohjelmistoa on tarkoitus käyttää kentältä käsin, jolloin valmentaja saa lähes reaaliaikaista tietoa pelaajan tilastoista. Tällöin ohjelmistoa on pystyttävä käyttämään luontevasti mobiililaitteilla ja tietojen syöttämisen on oltava yksinkertaista ja nopeaa. Ohjelmiston asiakaspuolen pohjaksi valittiin jQuery Mobile sen mahdollistaman nopean kehityksen ja helppokäyttöisyyden takia. Tällöin myös vältytään siltä, että ohjelma pitäisi tehdä usealla eri ohjelmointikielellä, jotta suurin osa mobiilialustoista saataisiin tuettua. Sovelluksen palvelinpuolen arkkitehtuuriksi valittiin ASP.NET MVC, sen hyvän yhteensopivuuden jQuery Mobilen kanssa ja omien kokemusten perusteella.

Ohjelmistoa on tulevaisuudessa tarkoitus kehittää työpöytäympäristöön paremmin sopivaksi ja mahdollisesti myös muuttaa palvelu maksulliseksi. Tämä vaatimus on otettu koko ajan huomioon ohjelmistoa suunniteltaessa ja toteuttaessa.

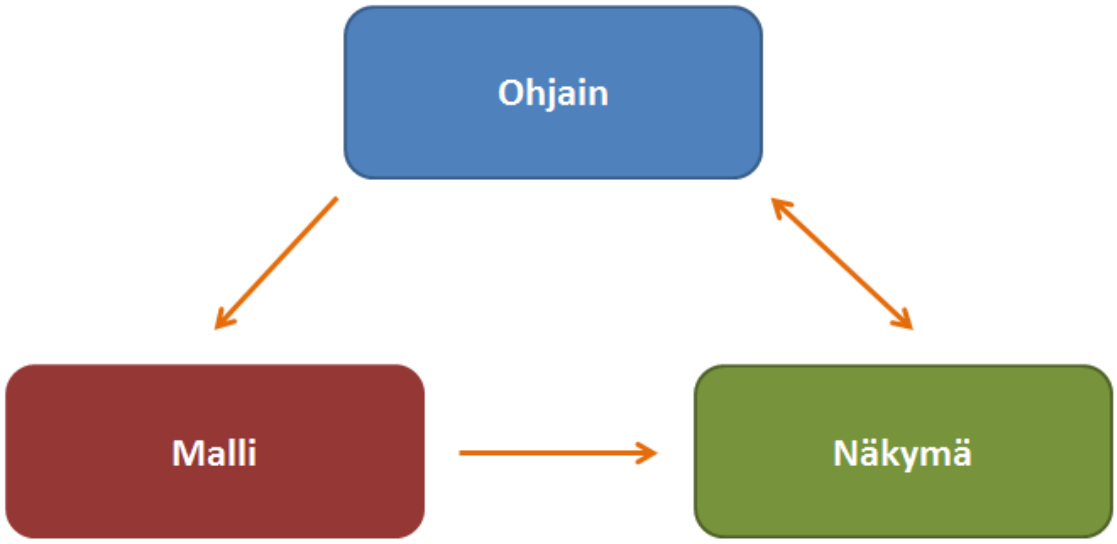
2 KESKEISET TEKNOLOGIAT

2.1 ASP.NET MVC

ASP.NET -ohjelmistokehys mahdollistaa kahden täysin erilaisen ohjelmistorakenteen valinnan: Web Formsin ja MVC:n. Web Forms on vanhempi ja perinteisempi tapa luoda web-sivuja, jossa jokaista näkymää vastaa vähintään yksi staattinen tiedosto palvelimella, joka taas kapsuloi kaiken sivun toiminnallisuuden sisäänsä. Web Forms lähestyy sivujen luomista samalla tavalla kuin työpöytäsovellusten tekemisessä eli sivun toimintalogiikka sekä näkymä sisällytetään kummatkin itse sivuun.

HTTP on tilaton protokolla (The Internet Society 2004), mikä tarkoittaa, että mitään tietoa esimerkiksi käyttäjästä ei siirry sivulatausten välissä. Uusi sivulataus ei ilman apua tiedä mitään ohjelman tilasta ennen sivulatausta. Web Forms kiertää tämän ongelman sisällyttämällä joka sivuun ViewState ominaisuuden, joka sisältää sivun tilan. Sivun koon kasvaessa myös View State kasvaa. Tämä ei ole toivottua mobiilisivuissa, joissa sivu pyritään pitämään mahdollisimman pienenä. jQuery Mobilen hakiessa sivujen sisällön AJAX-kyselyillä ja suodattamalla sivusta vain tarvittavan tiedon syntyy ongelmia sekä sivun tilan että html-elementtien ID-tunnisteiden kanssa.

Muun muassa näitä rajoituksia poistamaan on tehty MVC. MVC käsitteenä on arkkitehtuurimalli, joka tulee sanoista Model, View ja Controller. Mallissa esitetään ohjelmiston rakenne, jossa informaation hallinta ja käyttäjän vuorovaikutus on eriteltynä kolmeen eri osaan:



Kuvio 1. MVC -arkkitehtuuri.

Malli (Model) toimii mallina tiedon tallentamista, käsittelyä sekä ylläpitoa varten. Malli siis vastaa liiketoimintalogiikasta (Microsoft 2009). Malli voi olla esim. tietokannasta haettua tietoa. Malli on ainoa osa ohjelmistoa, joka on yhteydessä tietovarastoon (Kuvio 1). Käytännössä tämä tarkoittaa sitä, että sitä kautta hoidetaan mm. tietokantojen käsittely. Esimerkiksi näkymä ei ota suoraan yhteyttä tietokantaan vaan huolehtii asiasta mallin kautta. Malli pitää sisällään ohjelman kaikki luokat, ominaisuudet, tietotyypit sekä relaatiot, ja sitoo ne vastaaviin tietokannan kenttiin. Malli toimii siis virtuaalisena tietokantana, mutta toteutettuna valitulla ohjelmointikielellä.

```
[Table("Tours")]
public class Tour
{
    public int TourId { get; set; }
    public int UserId { get; set; }

    [Required(ErrorMessage = "Kilpailun nimi ei voi olla tyhjä")]
    public string Name { get; set; }

    [Display(Name = "Alkaa")]
    public DateTime DateFrom { get; set; }

    [Display(Name = "Päättyy")]
    public DateTime DateTo { get; set; }

    [Display(Name = "Sijoitus")]
    public int Placement { get; set; }

    public virtual UserProfile Profile { get; set; }
    public ICollection<TourRound> Rounds { get; set; }
}
```

Kuva 1. Esimerkki mallin luokasta.

Kuvan 1 luokassa ominaisuuksiin on lisätty mallin toimintaa ohjaavia attribuutteja, kuten kentän nimi, joka näytetään näkymässä käyttäjälle sekä kentän pakollisuuden määräävä tieto ja tämän virheilmoitus. Riippuen mallin luomiseen käytettävästä lähestymistavasta, voidaan nämä attribuutit sisällyttää joko suoraan luokkaan tai erilliseen metadata-luokkaan, jolloin nämä ominaisuudet eivät häviä luokkaa uudelleen generoitaessa tietokannasta.

Näkymä (View) pyytää tietoa mallilta ohjaimen kautta ja näyttää informaation käyttäjälle (Kuvio 1). Näkymä myös määrittää käyttöliittymän ulkoasun ja tietojen näytön esitystavan käyttöliittymässä (Microsoft 2009). Näkymä on yleensä konkreettinen websivu (cshtml).

```
@model vgDemo.Models.Practise

<h2>Harjoitus</h2>

@using (Html.BeginForm()) {
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)


        @Html.DropDownListFor(model => model.Type, new SelectList(trainingTypes))
        @Html.ValidationMessageFor(model => model.Type)

        <div class="ui-grid-a">
            <div class="ui-block-a">
                @Html.LabelFor(model => model.Hours)
                @Html.TextBoxFor(model => model.Hours, new { Type = "number" })
                @Html.ValidationMessageFor(model => model.Hours)
            </div>
            <div class="ui-block-b">
                @Html.LabelFor(model => model.Date)
                @Html.TextBoxFor(model => model.Date, new { Type = "datetime" })
                @Html.ValidationMessageFor(model => model.Date)
            </div>
        </div>

        @Html.LabelFor(model => model.Details)
        @Html.TextAreaFor(model => model.Details)
        @Html.ValidationMessageFor(model => model.Details)

        <p>
            <input type="submit" value="Tallenna" />
        </p>

}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Kuva 2. Esimerkki näkymästä.

ASP.NET MVC –ohjelmistokehykseen on sisällytetty paljon apumetodeja, jotka helpottavat muun muassa syötettävien tietojen validointia asiakaspäässä, html-elementtien generoinnissa sekä tarvittavien kirjastojen sisällytyksessä sivulle (Microsoft 2014a).

Näkymä voidaan määritellä joko dynaamisesti tai vahvasti tyypitetyksi. Dynaamisesti tyypitetyssä näkymässä näkymä tietää vasta ajon aikana millaista tietoa se tulee näyttämään. Vahvasti tyypitettäessä näkymä tietää koko ajan

mallin ominaisuudet ja Visual Studio osaa kertoa virheistä jo kehityksen aikana ennen ohjelman kääntämistä tai ajoa.

Ohjain (Controller) vastaanottaa käyttäjältä tulevat käskyt sekä muuttaa mallia ja näkymää vastauksena niihin. Web- sovelluksessa ohjain siis lukee tiedot HTTP-pyynnöstä ja muokkaa sovelluksen tilaa mallin avulla ja esittää lopputuloksen näkymän avulla (Kuvio 1).

ASP.NET MVC noudattaa tätä mallia vaikkakin ohjaimen toimintaa voidaan tarvittaessa siirtää myös näkymään. Tätä pidetään yleisesti kuitenkin huonona ohjelmointitapana, sillä se rikkoo MVC:n perusperiaatetta ja sitä tulisikin mahdollisimman pitkään välttää. ASP.NET:ssä ohjaimet ovat tavallisia luokkia, jotka kaikki periytyvät Controller-luokasta (Microsoft 2009).

```
[Authorize]
public class kalenteriController : Controller
{
    private Entities _context = new Entities();

    public ActionResult Empty()
    {
        return View();
    }

    public ActionResult Json()
    {
        return Json(true);
    }

    public ActionResult File()
    {
        return File(@"C:\dummy.pdf", "application/pdf");
    }

    public ActionResult Int()
    {
        return View(1);
    }
}
```

Kuva 3. Esimerkki ohjaimesta ja eri palautustyypeistä.

ASP.NET käyttää ohjainten etuliitettä myös verkko-osoitteiden reititykseen. Esimerkiksi osoite: http://localhost/kalenteri/_events/ ohjataan

kalenteriController-luokan _events-metodiin. Näitä reitityksiä on mahdollista muuttaa ja niihin on myös mahdollista lisätä staattisia parametreja. Ohjain palauttaa oletuksena ActionResult-luokan olion, joka voi olla näkymän lisäksi esimerkiksi html-koodia, tiedosto tai JSON-objekti. Palautettavan tiedon ollessa näkymä ASP.NET etsii metodin nimellä olevaa näkymää ja sen löytyessä palauttaa sen selaimelle.

2.2 jQuery Mobile

jQuery Mobile on HTML5 standardeihin pohjautuva Javascript- ja CSS-kirjasto, jonka tarkoituksena on helpottaa kehitystä mobiililaitteille. Se on rakennettu mahdollisimman laiteriippumattomaksi ja tukeekin lähes kaikkia suosituimpia mobiilialustoja kuten iOS:ää, Androidia ja Windows Phonea (The jQuery Foundation 2014a). JQM on suunniteltu erityisesti korvaamaan ja jäljittelemään natiiveja mobiiliapplikaatioita. Tarkoituksena on mahdollistaa web applikaatioiden saumaton käyttö ilman että käyttäjä edes huomaa toimivansa oikeasti selaimessa. JQM nopeuttaa mobiilisovellusten luomista useille käyttöjärjestelmille, sillä kehittäjän ei tarvitse opetella kaikkien eri mobiilialustojen ohjelmointikieliä. Tavallisella tietokoneella sivuja selatessa yhteytenä on yleensä nopeampi ja vakaampi yhteys kuin mobiililaitteilla (Nettitutka 2014). Tämä on otettu huomioon kutistamalla kaikki palvelimelle lähetettävät pyynnöt minimiin. JQM pyrkii käyttämään mahdollisimman vähän kuvia ja lähes kaiken grafiikan modernin CSS3-standardin avulla.

jQuery Mobile käyttää HTML5-standardin mukaisia data-attribuutteja mahdollistaakseen elementtien toiminnallisuuden ja ulkoasun muokkauksen ilman javascript osaamista (The jQuery Foundation 2014b). Nämä attribuutit ovat täysin vapaaehtoisia ja vastaavien metodien kutsuminen eri parametrein ajaa täysin saman asian. Sivuston luominen onnistuu myös hyvin vähäisellä javascript ja jQuery osaamisella. Seuraavana on neljä yleisintä data-attribuuttia:

- data-role: määrittelee elementin roolin kuten otsikko tai sisältö

- data-position: määrittelee elementin paikan sekä onko paikka absoluuttinen vai suhteellinen
- data-transition: määrittelee millaista siirtymistä sivujen latauksissa käytetään.
- data-theme: määrittelee yksittäisen elementin tai koko sivun teeman. Oletuksena käytössä on viisi teemaa, joihin viitataan kirjaimilla a, b, c, d, e.

jQuery Mobilessa sivu koostuu elementistä, jossa on määritetty data-role=”page” attribuutti. Elementin sisältö voi koostua mistä tahansa, mutta yleensä tyypillinen sisältö koostuu kolmesta div-elementistä, joiden data-role attribuutit ovat “header”, “main” ja “footer”.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title></title>
    <link href="/Content/jquery.mobile-1.4.2.css" rel="stylesheet" />
</head>
<body>
    <div data-role="page" data-title="Otsikko" data-url="/Home/Index">
        <div data-role="header" data-id="header">
            <h1>Otsikko</h1>
        </div>
        <div data-role="main" class="ui-content">
            <h2>Sivun otsikko</h2>
            <p>Sivun sisältö</p>
        </div>
        <div data-role="footer">
            <h1>Sivun alatunniste</h1>
        </div>
    </div>
    <script src="/Scripts/jquery-1.8.2.js"></script>

    <script src="/Scripts/jquery.mobile-1.4.2.min.js"></script>

</body>
</html>
```

Kuva 4. jQuery Mobilen sivurakenne.

jQuery Mobile käyttää AJAXiin perustuvaa sivunlatausta, jolloin koko sivua ei tarvitse ladata joka kerta uudelleen palvelimelta, vaan pelkkä muuttuva sisältö. Framework kaappaa automaattisesti tavalliset linkkien painallukset ja muuttaa ne AJAX-pyynnöiksi. Tämän jälkeen haettu sivu selataan läpi ja etsitään elementtiä data-role=”page” -attribuutilla, jonka sisältö syötetään vanhaan sivuun. Tällöin vain tarvittava tieto ladataan ja vältytään turhalta renderöimiseltä. Sivun lataus muistuttaa myös enemmän natiivin sovelluksen toimintaa. Sivulatauksen jälkeen jQuery Mobile lisää palvelimelta tulevaan html-koodiin useita lisäkerroksia sekä aputunnisteita ja muuttaa html-elementit kyseiselle laitteelle parhaiten sopiviksi (The jQuery Foundation 2014c).

2.3 Windows Azure

Windows Azure on Microsoftin kehittämä pilvipalvelualusta ohjelmistojen sekä palveluiden rakentamiseen, käyttöönottoon ja hallintaan Microsoftin maailmanlaajuisen datakeskusverkoston kautta. Se tarjoaa sekä alustan palveluna (platform as a service, PaaS) että infrastruktuurin palveluna (Infrastructure as a service, IaaS). Azure tukee monia eri ohjelmointikieliä, työkaluja sekä ohjelmistokehyksiä.

Azuren hinnoittelu on erittäin kilpailukykyinen esimerkiksi kotimaisiin webhotelleihin verrattuna. Azure perustuu siihen, että käyttäjä maksaa vain käytön mukaan, joka mahdollistaa pienet maksut käyttäjämäärien ja siirretyn datan ollessa pieniä. Kävijämäärien kasvaessa myös maksut nousevat samassa suhteessa, jolloin ei ikinä joudu maksamaan turhasta. Azure tarjoaa myös yhden kuukauden ilmaisen kokeilujakson. Ilmaiseen kokeilujaksoon kuuluvat kaikki Azuressa olevat komponentit tiettyine kuukausirajoituksineen (Microsoft 2014b).

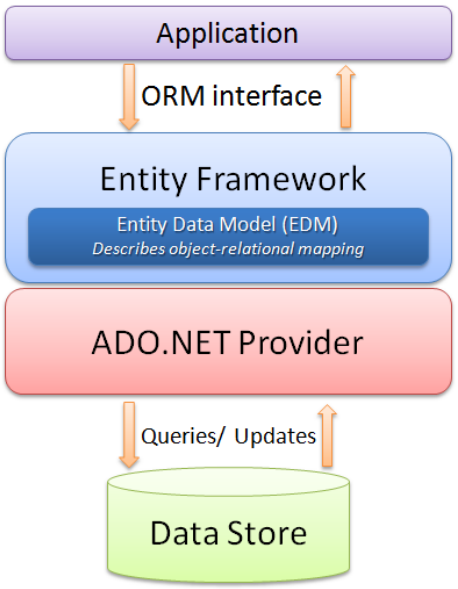
Pilvipalvelua käytettäessä käyttäjä ulkoistaa kaiken ylläpidon sekä hallinnoimisen kolmannelle osapuolelle. Azure tarjoaa palveluilleen erittäin suuren 99,95 %:n saatavuuden sekä kattavat maantieteellisesti hajautetut ja

replikoidut varmuuskopiot (Microsoft 2014c). Käyttäjän ei siis tarvitse itse huolehtia varmuuskopioiden ottamisesta tai päivitysten asentamisesta.

Tässä opinnäytetyössä perehdytään syvemmin ainoastaan Azure Websites sekä Azure SQL -moduuliin. Sivusto tulee sijaitsemaan Azuren web-palvelimella ja Azure SQL toimii sovelluksen tietokantana.

2.4 Entity Framework

Entity Framework (EF) on framework, joka kääntää tiedot eli tässä tapauksessa tietokantataulut ohjelmointikielelle yhteensopiviksi olioiksi. Tällöin tietokantoja voidaan käyttää oliomaisesti .NET sovelluksissa ja palveluissa. Se sijoittuu sovellusarkkitehtuurissa ohjelman ja tietovaraston väliin (Kuva 5). EF siis luo virtuaalisen tietokanta-objektin, jota voidaan lukea ja manipuloida ilman SQL-kielen osaamista (Kuva 6). Tällaista ohjelmaa kutsutaan lyhenteellä ORM (Object Relational Mapping).



Kuva 5. EF sovellusarkkitehtuurissa (Microsoft 2014d).

EF:n suurimmat hyödyt ovat seuraavat:

- Nopea ohjelmistokehitys: EF tarjoaa yleisimmät objektien operaatiot, jolloin kehittäjän ei tarvitse huolehtia SQL-kyselyistä, sillä EF muodostaa ne luodun mallin pohjalta.
- Kehittäjien ei tarvitse huolehtia taustalla olevasta tietokantamoottorista.
- Tietokannan ja mallin välisiä suhteita voidaan tietyissä rajoissa muuttaa ilman ohjelmakoodin muutoksia.
- IntelliSense ilmoittaa kehittäjälle virheistä kyselyissä jo ennen ohjelmakoodin kääntämistä.

```
// EF
_context.Events.Where(e => e.UserId == 1).ToList();

// SQL
"SELECT * FROM EVENTS WHERE UserId = 1;"

// EF
_context.Events.Where(e => e.Duration > 1 && e.Name.StartsWith("Matti")).Take(3).ToList();

// SQL
"SELECT TOP 3 * FROM EVENTS WHERE Duration > 1 AND Name LIKE 'Matti%';"

// EF
_context.Events.OrderBy(e => e.Date).FirstOrDefault();

// SQL
"SELECT TOP 1 * FROM EVENTS ORDER BY Date;";
```

Kuva 6. EF- ja SQL-kyselyiden vastaavuudet.

EF-kyselyjä käytettäessä on syytä ottaa huomioon, että tietokannan ollessa suuri tai kyselyn ollessa hyvin monimutkainen voi kyselyn suorituskyky kärsiä. EF joutuu kääntämään kaikki kyselyt SQL-kielelle sekä suorittamaan useita taustaprosesseja, jotka liittyvät tietokantaan sekä malliin liittämiseen (Microsoft 2014e). Suorituskyvyn heikkenemistä voidaan estää monella tapaa, joista tärkeimpinä mainittakoon valmiiksi käännettyjen kyselyiden käyttö ja näkymien esitäyttö. Tällöin kyselyt käännetään valmiiksi ja ladataan välimuistiin odottamaan käyttöä eikä kyselyä tarvitse kääntää joka kerta. Tätä työtä tehtäessä suorituskykyyn liittyviä ongelmia ei esiintynyt.

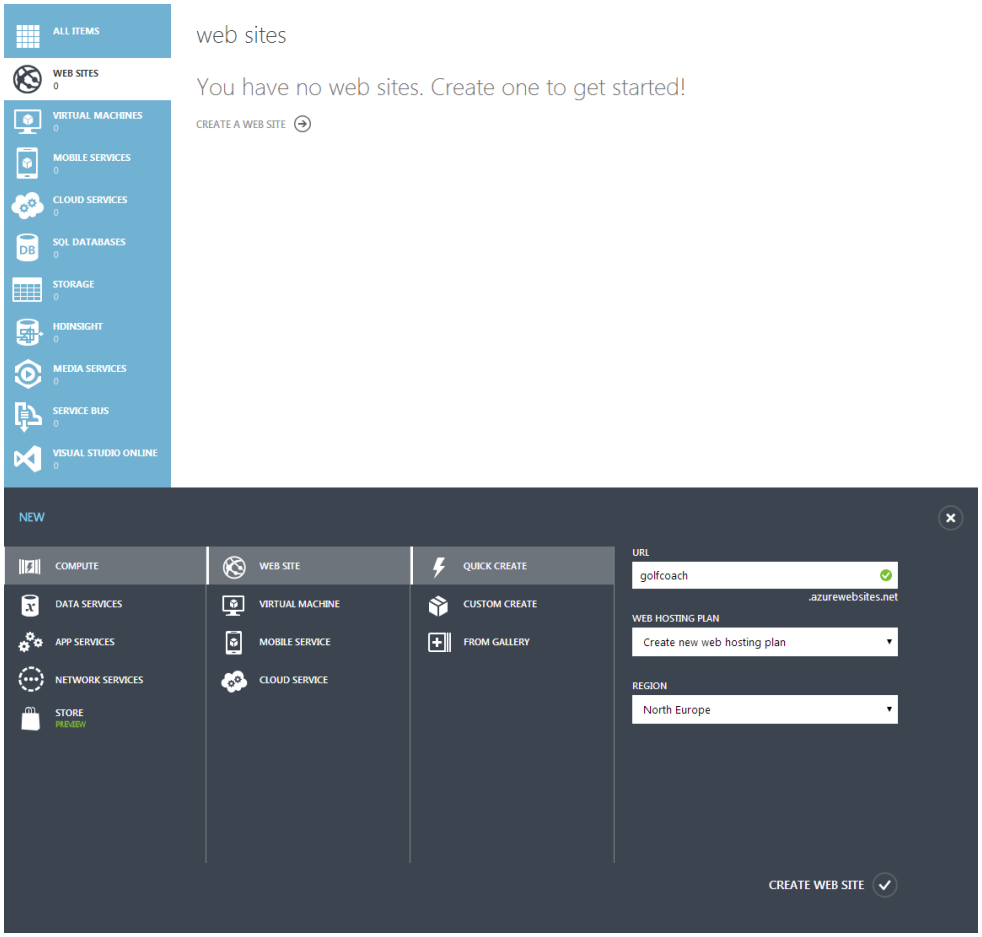
3 TOTEUTUS

3.1 Pohjustus

Sovelluksen suunnittelu aloitettiin yhdessä toimeksiantajan kanssa ja sovittiin sovelluksen pääpiirteittäinen toiminta sekä vaadittavat toiminnot. Samalla kartoitettiin myös nykyinen tapa seurata pelaajien kehitystä ja pohdittiin, miten jo olemassa olevat keinot saadaan siirrettyä sovellukseen ja mitä uusia tapoja sovellus mahdollistaisi.

3.2 Uuden sivuston luominen Azureen

Uuden sivuston perustaminen Windows Azureen on tehty yksinkertaiseksi ja nopeaksi. Uuden sivustun luominen aloitetaan WEB SITES -välilehdeltä klikkaamalla New-painiketta.

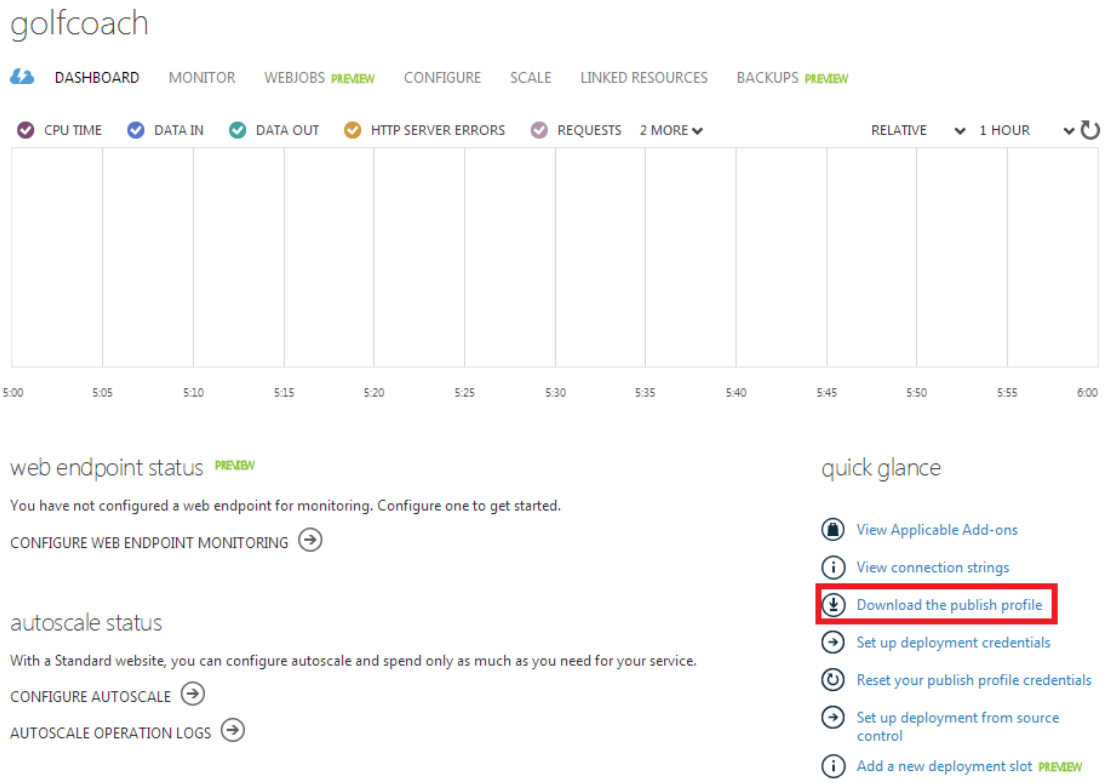


Kuva 7. Uuden sivuston luominen.

Uutta sivua varten tarvitaan vain sivun nimi, josta muodostuu myös sivun osoite sekä maantieteellinen sijainti. Azuressa kaikki sivut luodaan oletuksena domainin azurewebsites.net alidomaineiksi, jolloin käyttäjällä ei tarvitse olla rekisteröitynä omaa domainia testatakseen sivua selaimella. Sivustojen nimien on tämän takia oltava uniikkeja kaikkien Azuressa olevien sivujen kesken. Tässä projektissa sivuston nimeksi valittiin golfcoach, jolloin osoitteeksi muodostuu http://golfcoach.azurewebsites.net. Osoite on tarkoitus myöhemmin vaihtaa omaan domainin alle. Oman domainin käyttö on mahdollista käyttämällä joko CNAME tai A-tietuetta. (Microsoft 2014f).

Sivusto on tässä vaiheessa käytettävissä palvelimella. Sivujen lähetys Azureen on mahdollista myös ftp-ohjelmalla mutta helpoimmaksi tavaksi osoittautui sivun lähetys suoraan Visual Studion omalla julkaisutyökalulla. Julkaisua varten

Azuresta ladataan julkaisuprofiili (publish profile) kuvan 8 osoittamasta kohdasta, jonka jälkeen se asennetaan Visual Studioon. Tämän jälkeen sivun julkaisu ja päivitys onnistuvat muutamalla napinpainalluksella.



Kuva 8. Julkaisuprofiilin lataus Azuresta.

Julkaisuprofiili asennetaan Visual Studioon Publish-valikon ensimmäisen välilehden alta, johon syötetään ladattu profiili. Tämän jälkeen sivusto käyttää kyseistä profiilia ja muutokset päivittyvät Azureen joka julkaisun yhteydessä.

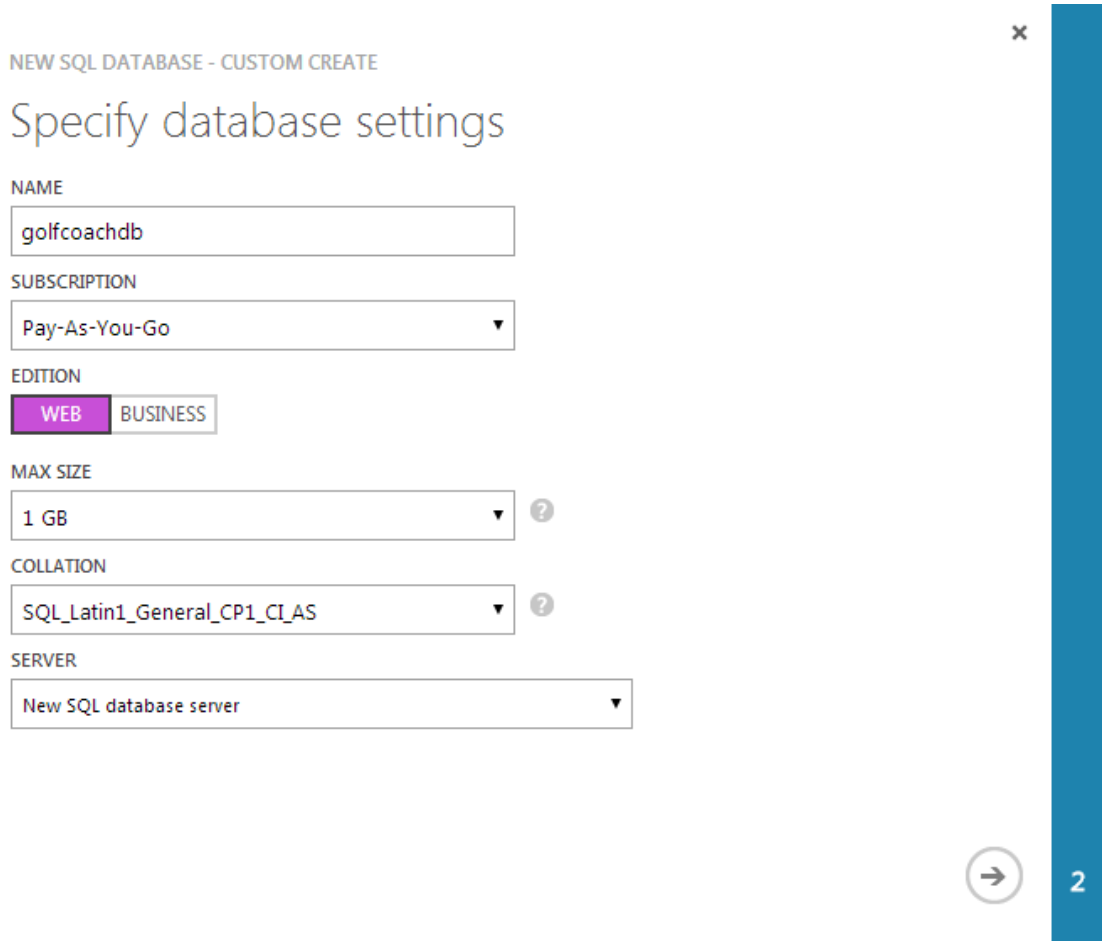
3.3 Tietokannan luomien

Sivuston perustamisen jälkeen luodaan vielä uusi tietokanta, johon tallennetaan ohjelman käyttämät tiedot. Uuden tietokannan perustaminen tapahtuu SQL DATABASES-välilehdeltä.

Tietokannalle määritellään nimi sekä palvelin. Muut asetukset voidaan tässä tapauksessa jättää oletuksena oleviin (Kuva 9). Seuraavalla sivulla asetetaan

vielä käyttäjätunnus sekä salasana. Tämän jälkeen tietokanta on valmis käytettäväksi.

Tietokannan taulujen luominen ja muokkaus onnistuu Azuren selainkäyttöliittymällä, Microsoft SQL Server Management –työkalulla tai suoraan Visual Studiosta.



Kuva 9. Uuden tietokannan luominen.

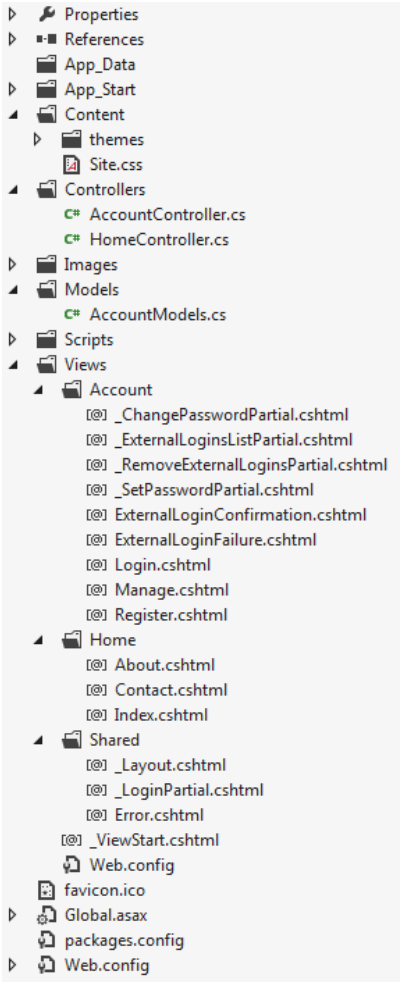
Tietokantaan ei vielä tässä kohdassa luoda yhtäkään taulua, sillä se jätetään myöhemmin tehtäväksi.

3.4 Projektin luominen Visual Studiossa

Sovelluksen teko aloitetaan luomalla uusi projekti Visual Studiolla. Ensimmäisenä projektille valitaan nimi sekä .NET-Frameworkin versio. Tässä projektissa tullaan käyttämään versiota 4.5. Seuraavalta sivulta voidaan valita projektille valmis pohja sekä näkymämoottori, joka määrittelee näkymätiedostoissa käytettävän mallin käsittelyä ohjeistavan koodikielen sekä syntaksin. Vaihtoehtoina ovat ASPX sekä Razor, josta tähän sovellukseen valittiin jälkimmäinen vaihtoehto. Pohjiin kuuluvat seuraavat vaihtoehdot:

- Empty: täysin tyhjä pohja, jossa automaattisesti generoidaan käytännössä vain kansiorakenne.
- Basic: pohja sisältää kansiorakenteen lisäksi yleisimmät JavaScript kirjastot sekä perus css-tyylitiedostot.
- Internet Application: edellisten ominaisuuksien lisäksi projektiin generoidaan proseduurit käyttäjien hallintaan sekä muutama esimerkkisivu ohjaimineen.
- Intranet Application: käyttäjätunnuksina toimii Windows-tunnukset. Muuten pohja on samanlainen kuin edellinen.
- Mobile Application: Internet Application muutettuna paremmin mobiililaitteille sopivaksi.
- Single Page Application: yksisivuinen moderni web-applikaatio, joka perustuu jQuery Mobilea hyvin paljon muistuttavaan KnockoutJS-frameworkiin.

Kyseistä sovellusta varten valittiin Internet Application, joka on hyvä pohja lähes minkä tyyppiselle sovellukselle tahansa. Kehittäjän ei tarvitse itse luoda käyttäjätunnusten hallintaan liittyviä metodeja itse, mikä nopeuttaa huomattavasti uuden projektin aloitusta, sillä kehittäjä pääsee suoraan pureutumaan itse ohjelman logiikkaan.



Kuva 10. Visual Studion generoiman projektin rakenne.

Kuvassa 10 on esitettynä Visual Studion automaattisesti luodun projektin rakenteen, johon kuuluu muun muassa käyttäjätietokanta sekä ohjaimet ja näkymät muun muassa kirjautumiselle, salasanan vaihdolle, sekä rekisteröitymiselle joko paikallisia tunnuksia käyttäen tai OAuth 2.0 -rajapintaa tukevien kolmansien osapuolien tunnuksilla. OAuth -rajapintaa tukeviin yrityksiin kuuluvat muun muassa Google, Facebook, Microsoft sekä Twitter (OAuth 2014). Käytännössä tämä tarkoittaa kirjautumista näiden palveluiden olemassa olevilla tunnuksilla, josta sivusto hakee tarvittavat tiedot sekä liittää ne paikalliseen tietokantaan. Sivuston suljetusta käyttäjäkunnasta johtuen tämä ominaisuus poistettiin kuitenkin käytöstä.

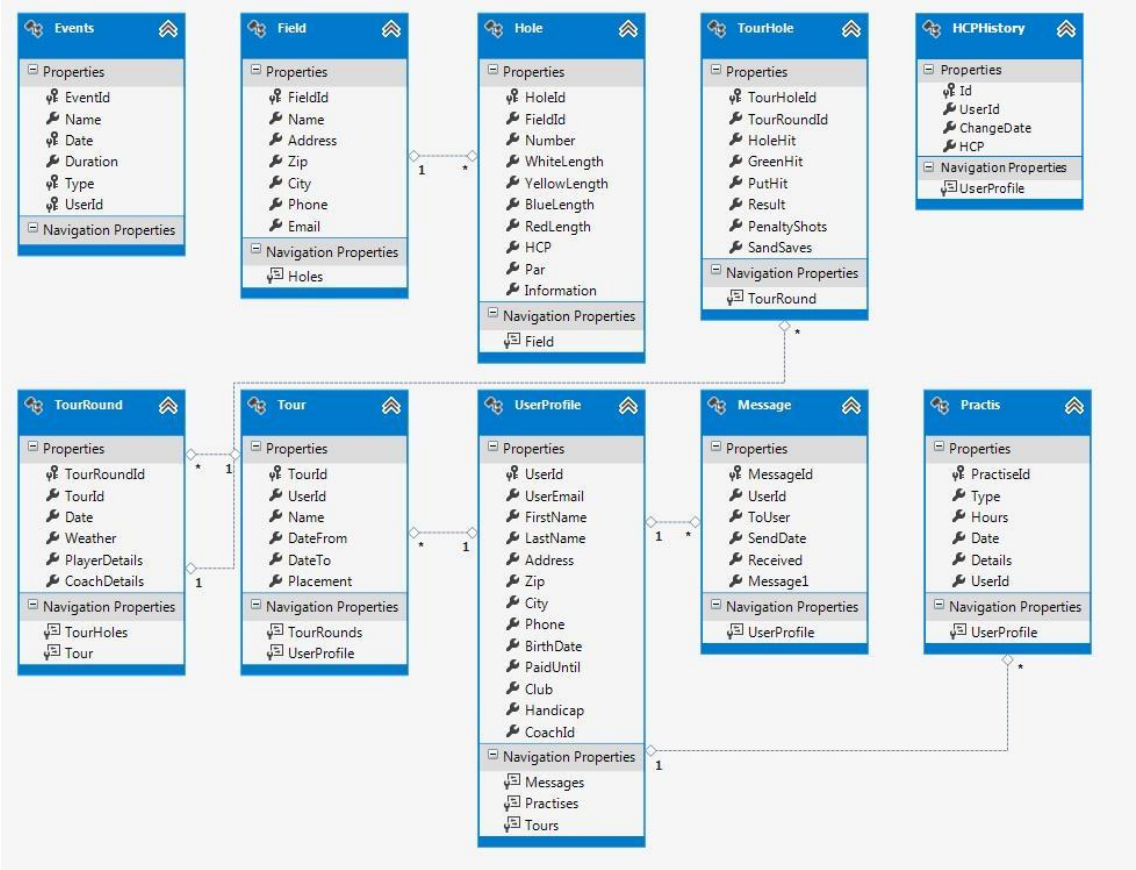
Projekti piti valmiina jo sisällään jQuery sekä jQuery UI –kirjastot, joita jQuery Mobile tarvitsee tuekseen, joten ainoaksi laajennukseksi asennettiin vain jQuery Mobile.

3.5 Sovelluksen malli

Malli voidaan generoida automaattisesti kolmella eri tavalla. Visual Studiossa automaattisesti luodusta mallista käytetään nimeä Entity Data Model. Malli ei rajoitu pelkästään automaattisesti luotaviin olioihin vaan siihen voidaan tarpeen vaatiessa lisätä myöhemmin lisää olioita (Lerman, J. 2011). EDM:n luomiseen käytettävät lähestymistavat ovat:

- Database First (tietokanta ensin)
- Code First (ohjelmakoodi ensin)
- Model First (malli ensin)

Tietokanta ensin-periaatteella määritetään taulut ja relaatiot ensin ja sen perusteella ohjelma luo tietokannalle mallin. Kuvassa 11 esitetään mallin ER-kaavio tietokannasta.



Kuva 11. Tietokannan ER-kaavio.

Tässä projektissa käytettiin Database First-lähestymistapaa, koska haluttiin säilyttää mahdollisuus lisätä tietokantaan uusia kenttiä myöhemmin. Model First- ja Code First-menetelmissä mallin muuttaminen tai uusien attribuuttien lisääminen aiheuttaa oletuksena joka kerta tietokannan rivien tyhjentymisen, sillä mallin on aina vastattava tietokantaa eikä tietokantaa voida tietyissa tapauksissa muuttaa jos taulu pitää sisällään tietoa. Luomalla tietokanta ensin tätä ominaisuutta voidaan hallita helpommin.

3.6 Sivuston rakenne

Sivusto jaetaan kahteen eri alueeseen: valmentajan sekä pelaajan alueeseen. Pelaajan puoli on tarkoitettu harjoitusten sekä kilpailujen tilastojen syöttöön ja

seurantaan. Valmentajan puoli pitää sisällään valmennettavien syöttämien tietojen analysoinnin sekä kehityksen seurannan. Pelaajalla ei välttämättä ole valmentajaa eli samojen tietojen seuranta onnistuu myös pelaajalta. Tässä opinnäytetyössä tarkastellaan tarkemmin vain pelaajan näkymää.

Pelaajan näkymä koostuu neljästä eri pääsivusta sekä navigaatiovalikosta:

- kalenteri
- harjoitteet
- kilpailut
- statistiikat
- valmentaja.

Navigaatiovalikko sijaitsee sivun alareunassa ja se mahdollistaa nopean siirtymisen eri sivujen välillä. Sivulla on aina näkyvissä myös linkki sekä etusivulle että oman profiilin asetuksiin, mistä onnistuu esimerkiksi salasanan vaihto sekä uloskirjautuminen. Navigaatiovalikossa näkyy myös esimerkiksi valmentajalta tulleiden kommenttien ja päivän tapahtumien lukumäärä.

3.6.1 Kalenteri

Kalenterisivun tarkoituksena on pitää pelaaja ajan tasalla omista tapahtumistaan sekä tarjota helppo ja nopea käyttöliittymä omien suoritusten merkitsemiseen. Pelaaja voi halutessaan selata sekä muokata tapahtumia. Kuvasta 13 ilmenee kalenterisivun käyttöliittymä.

Kuva 12. Ohjelman kalenterinäkymä.

Kilpailut ja harjoitukset on värikoodattu, jolloin käyttäjä erottaa helposti ohjelmastaan eri tapahtumat. Tyhjän päivän painaminen avaa valikon, josta valitaan halutaanko syöttää uusi tapahtuma kilpailulle vai harjoitteelle. Valinnasta riippuen käyttäjä ohjataan oikealle sivulle, jossa täytetään tapahtuman tiedot ja harjoituksen tai kilpailun tilastointiin liittyvät tiedot. Tapahtuman painaminen taas ohjaa käyttäjän kilpailun tai harjoitteen sivulle, josta tietoja pääsee tarkastelemaan ja tarvittaessa muokkaamaan.

Kalenterin asiakaspään toimintalogiikasta huolehtii jQuery-lisäosa FullCalendar. Myös FullCalendarissa on jQuery Mobilen tavoin mahdollisuus hakea tapahtumat AJAX-pyynnöillä. Kaikkia tapahtumia ei haeta kerralla, vain kalenterissa näkyvät. FullCalendar liittää jokaiseen pyyntöön alku- sekä loppuajan, jolloin palvelin osaa palauttaa oikeat tapahtumat. Tällöin

tapahtumien lataus pysyy nopeana eikä hidasta sivun toimintaa tapahtumien määrän ollessa suuri. Kuukauden vaihto lähettää joka kerta uuden AJAX-pyynnön palvelimelle. Koska jQuery tallentaa GET-pyynnöt oletuksena välimuistiin, lisää kalenteri jokaiseen pyyntöön vielä ylimääräisen aikaleimaparametrin estämään pyynnön välimuistikopion lataamisen.

```
<script>
    $(document).on('pageload', function () {
        calendar = $('#calendar').fullCalendar({
            lang: 'fi',
            dragRevertDuration: 0,
            header: {
                left: 'prev,next today',
                center: 'title'
            },
            editable: true,
            events: {
                url: '/kalenteri/_events',
                type: 'GET',
                traditional: true,
                error: function () {
                    alert('Tapahtumia haettaessa tapahtui virhe');
                }
            },
            eventClick: function (calEvent, jsEvent, view) {
                $.mobile.changePage(calEvent.url, { });
            },
            dayClick: function (date, jsEvent, view) {
                $.mobile.changePage('/tapahtumat/uusi', {
                    data: { startdate: date.format('DD.MM.YYYY HH:mm') }
                });
            }
        });
    });
</script>
```

Kuva 13. Kalenterin alustus.

Kuvassa 13 alustetaan kalenteri sekä määritellään metodit päivän ja tapahtuman painamiselle. Palvelimella määritellään tapahtuman muokkausta varten url-attribuutti, joka normaalisti aiheuttaisi selaimen synkronisen ohjauksen määritellylle sivulle. Tapahtuman painalluksen käsittelevä metodi ylikirjoitetaan ohjaamaan samalle sivulle asynkronisesti eli jQuery Mobilen suosimalla tavalla. Päivää painettaessa käyttäjä ohjataan samalla tavalla uuden tapahtuman sivulle.

Javascript -funktiot sisällytetään yleensä jQueryn ready() -metodin sisään, jolloin jQuery odottaa, että koko sivu on ladattu ennen kuin muita funktioita suoritetaan, jolloin voidaan olla varmoja, että kaikki sivun DOM-elementit ovat käytettävissä. jQuery Mobilessa tätä metodia ei kuitenkaan suoriteta joka sivulatauksen yhteydessä, sillä sisältö haetaan asynkronisesti. jQuery Mobilessa vastaava toiminnallisuus saavutetaan odottamalla pageload-tapahtumaa.

```
public JsonResult _events(string start, string end)
{
    int userId = WebSecurity.CurrentUserId;
    DateTime dtStart, dtEnd;

    if (!DateTime.TryParse(start, out dtStart) || !DateTime.TryParse(end, out dtEnd))
        throw new ArgumentException("Datetime parsing failed");

    var events = _context.Events
        .Where(e => e.UserId == userId && (e.Date >= dtStart || e.Date.AddHours(e.Duration) <= dtEnd))
        .AsEnumerable()
        .Select(e => new {
            title = e.Name,
            start = e.Date.ToString("yyyy-MM-ddTHH:mm"),
            end = e.Date.AddHours(e.Duration).ToString("yyyy-MM-ddTHH:mm"),
            allDay = false,
            url = Url.Action("muokkaa", e.Type == "practise" ? "harjoitukset" : "kilpailut", new { id = e.EventId }),
            className = "type-" + e.Type,
        })
        .ToList();

    return Json(events, JsonRequestBehavior.AllowGet);
}
```

Kuva 14. Kalenterin ohjaimen toimintalogiikka.

Ohjain validoi palvelimella sekä alku- että loppupäivämäärän virheiden varalta ja hakee käyttäjän tapahtumat kyseiseltä aikaväliltä. Entity Framework huolehtii kyselyn kääntämiseksi SQL-kielelle. Tietoja haettaessa on muistettava, että EF ei suorita kyselyitä ennen kuin niitä konkreettisesti pyydetään ohjelmassa. Tällaista konseptia kutsutaan nimellä Lazy Loading . EF ei pysty muuntamaan SQL-kielen avulla päivämäärää ja kellonaikaa muuhun muotoon. Tällöin pitää kutsua metodia, joka käskee ohjelmaa suorittamaan kyselyn tietokantaan. Näitä metodeita on useita eri tapauksia varten. Tässä tapauksessa AsEnumerable()-metodi suorittaa kyselyn, jolloin jälkimmäisiä metodeja ajaessa tiedot ovat jo oliomuodossa ja päivämäärät voidaan muuttaa merkkijonoiksi haluttuun muotoon. Tapahtuman päättymispäivä lasketaan tietokantaan tallennetusta tiedosta, joka on tunteina alkamisesta. Tapahtuma-objektiin lisätään vielä

className-attribuutti, joka auttaa tapahtuman värin asettamisessa css-tyylien avulla (Kuva 14).

Ohjain osaa yleensä automaattisesti serialisoida oliot JSON-muotoon, jolloin kehittäjän ei esimerkiksi tarvitse huolehtia syntaksin tarkastamisesta.

3.6.2 Harjoitteet

Harjoite-sivu koostuu listasta, josta käyttäjä näkee kaikki tulevat sekä menneet harjoitteet. Sivulta löytyvät myös napit harjoitteen poistolle, muokkaukselle sekä uuden harjoitteen lisäämiselle, joka johtaa samalle sivulle kuin kalenterista päivää painamalla, mutta ilman päivämäärän esitäyttöä.

Uuden harjoitteen lisääminen on yksinkertaista. Harjoitukseen valitaan ennalta määritetystä listalta harjoituksen tyyppi, joka voi olla esimerkiksi range- tai fysiikkaharjoitus. Harjoitukseen täytetään myös päivämäärä, harjoituksen kesto sekä vapaaehtoinen lisätietokenttä, johon pelaaja ja valmentaja voivat kirjoittaa mahdollisia muistiinpanoja ja lisätietoja. Tallennuksen jälkeen harjoite ilmestyy kalenteriin sekä listaan, josta harjoitusta päästään tarvittaessa muokkaamaan.

3.6.3 Kilpailut

Golfturnaus voi kestää useita päiviä sekä koostua useasta kierroksesta. Tästä johtuen yksi kilpailu on jaettu kolmeen osaan:

- kilpailu
- kierros
- reikä.

Kilpailu voi sisältää useita kierroksia. Kierros voi sisältää useita reikiä, johon tallennetaan kaikki tilastolliset merkinnät. Uutta kilpailua lisätessä käyttäjän on ensin luotava kilpailulle uusi tapahtuma, jonka alle kierrokset ja reiät luodaan. Tietokannassa tauluihin on luotu relaatiot, jolloin oikeat reiät ja kierrokset saadaan haettua suoraan kilpailun perusteella.

3.6.4 Statistiikat

Statistiikkasivu on ohjelman yksi tärkeimmistä sivuista, sillä sen avulla pelaaja saa palautteen omista suorituksistaan. Pelaajan on tarkoitus seurata omaa kehitystään myös itse, jolloin ohjelman käyttämiseen ei välttämättä tarvita valmentajaa. Pelaaja näkee nopeasti itse omat puutteensa ja kehityssuunnan havainnollistavista kuvaajista ja taulukoista. Kilpailuista tallennetaan aina myös kentän tiedot, jotka sisältävät muun muassa reikien par-arvot sekä kentän vaikeutta kuvaavan Slope-arvon.

Statistiikoissa oleellisena asiana on myös tasoitusjärjestelmä, jonka avulla voidaan laskea, kuinka hyvin pelaaja on pelannut kierroksen omiin taitoihinsa ja omaan tasoitukseensa nähden. Tasoitusjärjestelmän ansiosta tulokset ovat keskenään vertailukelpoisia pelaajien tasoeroista ja kentän vaikeustasosta riippumatta (Tasoitusjärjestelmä (golf) 2013). Näiden tietojen lisäksi tilastoissa otetaan huomioon myös sää, jolloin saadaan mahdollisimman tarkka arvio pelaajan kehityksestä ilman, että yksittäiset tai ulkopuolisesta syystä johtuvat tapahtumat muuttavat arviota rajusti.

Kuva 15. Harjoitusmäärän kuvaaja.

Kuvaajien piirtämiseksi selaimessa käytetään jQueryn lisäosaa jqplot, joka on erittäin kattava ja monipuolinen ohjelma kuvaajien piirtämiseksi sekä myös yhteensopiva mobiilialustojen kanssa. Kuvassa 16 oleva kaavio on yksi monista, joita pelaaja pääsee tarkastelemaan. Kyseinen kaavio on generoitu satunnaisluvuista oikean datan puutteen vuoksi. Käyttäjä voi myös vaihtaa tarkastelemaansa vuotta alasvetovalikosta.

```
[Authorize]
public class statistiikkaController : Controller
{
    private UsersContext _repository = new UsersContext();

    //
    // GET: /statistiikka/
    public ActionResult Index(int? y)
    {
        int year = y ?? DateTime.Now.Year;
        return View(year);
    }

    //
    // POST: /statistiikka/_trainingHours?y=2014
    public JsonResult _trainingHours(int y)
    {
        List<WeeklyHours> hours = new List<WeeklyHours>();

        var practisehours = _repository.Practises
            .Where(p => p.Date.Year == y && p.UserId == WebSecurity.CurrentUserId) // Haetaan vuoden kaikki harjoitukset käyttäjälle
            .GroupBy(p => Math.Floor(p.Date.DayOfYear / 7.0)) // Ryhmitellään harjoitukset viikon mukaan ja pyöristetään alaspäin seuraavaan kokonaislukuun
            .OrderBy(p => p.Key) // Järjestetään viikon mukaan
            .ToDictionary(p => p.Key, p => (int)p.Sum(h => h.Hours)); // Summataan harjoitukset jokaisella viikolla Dictionary tyypin muuttujaan

        for (int i = 1; i <= 52; i++)
        {
            var weeklyhours = new WeeklyHours();
            weeklyhours.week = i;
            weeklyhours.hours = practisehours.ContainsKey(i) ? practisehours[i] : 0; // Lisätään myös viikot jolloin tietokannasta ei löydy merkintää viikolle
            hours.Add(weeklyhours);
        }

        var viewmodel = new { hours = hours, weeks = Enumerable.Range(1, 52).ToArray() };

        return Json(hours);
    }
```

Kuva 16. Harjoitusmäärän palauttavan ohjaimen toimintalogiikka.

Sivun kokonaan ladattuaan jQuery Mobile lähettää AJAX-pyynnön osoitteeseen /statistiikka/_trainingHours joka palauttaa kyseisen vuoden yhteenlasketun harjoitusmäärän viikoittain ryhmiteltynä JSON-muodossa. Onnistuneen pyynnön jälkeen kaavio renderöidään ruudulle.

3.6.5 Valmentaja

Valmentaja-sivu on tarkoitettu pelaajan ja valmentajan väliseen vuorovaikutukseen, kuten viestien lähettämiseen sekä harjoitusten ja kilpailujen kommentointiin. Valmentajasta on näkyvissä perustiedot, kuten puhelinnumero ja sähköpostiosoite. Tällöin pelaaja voi helposti soittaa tai lähettää viestin valmentajalle. Pelaaja voi myös poistaa valmentajan, minkä jälkeen sivulla on mahdollisuus lisätä uusi valmentaja itselleen. Pelaajalla ei kuitenkaan voi olla kuin yksi valmentaja. Valmentajan on hyväksyttävä pyyntö ennen kuin kaikki toiminnallisuudet tulevat käyttöön.

4 JATKOKEHITYS JA PALAUTE

Palvelulle on tarkoitus rakentaa tulevaisuudessa täysin itsenäinen suurille näytöntarkkuuksille suunnattu sivusto. Statistiikkasivua suunniteltaessa huomattiin, että kaikkea haluttua tietoa ei ole mahdollista näyttää järkevästi mobiililaitteilla. Vaikka mobiililaitteiden näytön resoluutio ja koko on viime vuosina kasvanut, asettavat ne silti tietyissä tapauksissa rajoitteita tiedon esitystavalle. Tämä ongelma tuli vastaan erityisesti kuvaajia suunniteltaessa, jossa kaikki tieto ei vain yksinkertaisesti mahtunut ruudulle halutulla tavalla. Näytön fyysinen koko tulee ratkaisevasti esiin, kun halutaan esimerkiksi vertailla useaa pelaajaa samanaikaisesti. Tällöin pienellä ruudulla sivua joudutaan rullaamaan edestakaisin, jolloin vertailu on hankalaa. Pöytäkoneympäristö mahdollistaa myös raa'an tiedon lataamisen muodossa, jota ei mobiililaitteilla välttämättä ole mahdollista avata. Tällöin esimerkiksi valmentaja voi itse luoda haluamiaan raportteja ja kaavioita.

Kummatkin sivustot käyttäisivät samaa tietovarastoa niin käyttäjätunnuksille kuin tilastoillekin. Tällöin kummankin sivuston tulisi käyttää yhteistä rajapintaa tietojen hakemiselle, jolloin säästytään toiminnallisuuden kirjoittamiselta moneen kertaan ja muutokset heijastuvat kumpaankin sivustoon. MVC-arkkitehtuurimallissa tämä tarkoittaa sitä, että ohjain ei ole enää suoraan yhteydessä tietovarastoon vaan suorittaa kyselyt erillisen rajapinnan kautta, joka taas palauttaa halutut tiedot tietokannasta. Tätä ei ole otettu sivuston suunnittelussa alusta alkaen huomioon ja rajapinnan toteutus aiheuttaisi myös nykyisen sivuston ohjainten osittaisen uudelleensuunnittelun.

Sivustoa on tulevaisuudessa mahdollisesti tarkoitus tarjota myös muille valmentajille sekä seuroille maksullisena palveluna. Ohjelmisto pysyisi pelaajille ilmaisena ja olisi kuukausimaksullinen vain seuroille sekä valmentajille. Tämä on otettu huomioon ohjelmaa tehtäessä ja tietokannassa ovat nykyisellään käyttämättömät kentät esimerkiksi ilmaiselle kokeilujaksolle sekä tilauksen loppumispäivämäärälle, jolloin valmentaja voi halutessaan lopettaa ohjelman

käytön tai jatkaa tilaustaan. Maksupalvelun toteutusta ei vielä kirjoitushetkellä ole mietitty.

Tarkoitus on myös lisätä kävijäseuranta sivulle Google Analytics-työkalun avulla. Tällöin saadaan yksityiskohtaista tietoa sivuston käytöstä sekä mahdollisista epäloogisuuksista ja puutteista käyttöliittymässä.

Golfin peluu ja harjoittelu tapahtuu pääsääntöisesti kesällä, joten sovellusta ei vielä ole päästy testaamaan kovinkaan laajasti. Tämä hankaloittaa erityisesti statistiikkojen toiminnallisuuden arvioimista, sillä tarvittavaa tietoa ei vielä ole tarpeeksi. Vuorovaikutus toimeksiantajan kanssa kuitenkin jatkuu koko ajan ja kesän jälkeen ohjelman arviointi on jo helpompaa. Alustavat arviot ja pelaajien antama palaute ohjelmasta ovat kuitenkin olleet pääasiassa positiivisia. Pelaajat ovat olleet tyytyväisiä päästessään konkreettisesti seuraamaan omaa kehitystään.

Muutamia bugeja ja käyttöliittymän epäloogisuuksia on kuitenkin jo löytynyt ja ne on korjattava tulevaisuudessa. Esiin on tullut myös offline-tilan puute, sillä joissain kentillä voi yhteys olla heikko. Tällaisissa tilanteissa käyttäjät on ohjeistettu syöttämään tiedot siirryttyään kentältä paremman verkon alueelle.

5 YHTEENVETO

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa virtuaalinen kehityksenseurantajärjestelmä golf-harjoitteluun. Järjestelmä toteutettiin pääasiassa uusimmilla Microsoftin tarjoamilla työkaluilla sekä palveluilla. Työn avulla tutustuttiin moderneihin teknologioihin ja pilvipalveluihin.

Työssä nähdään käytännössä, miten helppoa ja nopeaa nykyaikainen ohjelmistokehitys näitä työkaluja hyväksikäyttäen parhaimmillaan on. Valmiita pohjia käyttämällä ja niitä omiin tarpeisiin räätälöimällä kehittäjän ei tarvitse toteuttaa ohjelmia alusta alkaen tyhjästä, vaan pohjat tarjoavat hyvän pohjan monimutkaisillekin sovelluksille.

Suurin haaste kehittäjän näkökulmasta työssä oli opetella perusasiat golfista sekä ymmärtää toiminnallisuus valmentajan näkökulmasta. Muita haasteita olivat tiedon esitys mobiililaitteiden vaatimusten mukaisesti sekä käyttöliittymän suunnittelu mahdollisimman yksinkertaiseksi ja helpoksi. Itse työn toteutus sujui jouhevasti hyvän suunnitelman pohjalta. Käytetyt teknologiat olivat jo pääsääntöisesti muista projekteista ennestään tuttuja, joten perusasioiden opetteluun ei tuhlaantunut aikaa. Lopputuloksena syntyi haluttu sovellus, jota on tarkoitus jatkokehittää eteenpäin ja mahdollisesti myös julkaista yleiseen käyttöön.

LÄHTEET

The Internet Society 2004. Hypertext Transfer Protocol -- HTTP/1.1. Viitattu 15.5.2014 http://www.w3.org/Protocols/rfc2616/rfc2616.html

The jQuery Foundation 2014a. jQuery Mobile 1.4 Supported Platforms. Viitattu 13.2.2014 http://jquerymobile.com/gbs/1.4/

The jQuery Foundation 2014b. Pages. Viitattu 11.6.2014 http://demos.jquerymobile.com/1.4.2/pages/

The jQuery Foundation 2014c. Ajax Navigation. Viitattu 11.6 http://demos.jquerymobile.com/1.4.2/navigation/

Lerman, J. 2011. Demystifying Entity Framework Strategies: Model Creation Workflow. Viitattu 11.6.2014 http://msdn.microsoft.com/en-us/magazine/hh148150.aspx

Microsoft 2009. ASP.NET MVC Overview. Viitattu 11.6.2014 http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview

Microsoft 2014a. HtmlHelper Methods. Viitattu 11.6.2014 http://msdn.microsoft.com/en-us/library/system.web.mvc.htmlhelper_methods(v=vs.100).aspx

Microsoft 2014b. Pricing. Viitattu 2.6.2014 http://azure.microsoft.com/en-us/pricing/calculator/?scenario=web

Microsoft 2014c. Service Level Agreements. Viitattu 2.6.2014 http://azure.microsoft.com/en-us/support/legal/sla/

Microsoft 2014d. ADO.NET Entity Framework At-a-Glance. Viitattu 13.5.2014 http://msdn.microsoft.com/en-us/data/aa937709.aspx

Microsoft 2014e. Performance Considerations for Entity Framework 4, 5, and 6. Viitattu 3.6.2014 http://msdn.microsoft.com/en-US/data/hh949853

Microsoft 2014f. Configuring a custom domain name for an Azure cloud service. Viitattu 2.6.2014 http://azure.microsoft.com/en-us/documentation/articles/cloud-services-custom-domain-name/

Nettitutka 2014. Tilastot. Viitattu 1.6.2014 http://www.netradar.org/fi/statistics/

OAuth 2014. OAuth 2.0. Viitattu 11.6.2014 http://oauth.net/2/

Tasoitusjärjestelmä (golf) 2013. Wikipedia. Viitattu 3.6.2014 http://fi.wikipedia.org/wiki/Tasoitusjärjestelmä_(golf)