Ville Viitaharju

JULKAISUJÄRJESTELMÄ FINNRUNS-JÄRJESTÖLLE

Tietojenkäsittelyn koulutusohjelma

2015

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

JULKAISUJÄRJESTEMÄ FINNRUNS-JÄRJESTÖLLE

Viitaharju, Ville
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Elokuu 2015
Ohjaaja: Nieminen, Hans
Sivumäärä: 34
Liitteitä:

Asiasanat: Julkaisujärjestelmä, Web, Sovelluskehitys, Projektinhallinta

---

Opinnäytetyön aiheena oli julkaisujärjestelmän toteutus Finnruns-järjestölle. Opinnäytetyössä käydään läpi projektin toteutusta ja lopputulosta.

Opinnäytetyössä kuvataan projektia kokonaisuudessaan ja perehdytään eri selaintekniikoiden toimintaan. Lisäksi opinnäytetyössä kerrotaan projektissa esiintyvistä tärkeistä sekä hankalista tilanteista.

Työn tuloksena syntyi nopeakäyttöinen verkossa toimiva julkaisujärjestelmä joka palvelee Finnruns-järjestön tarpeita. Julkaisujärjestelmä on helppo käyttää ja sisältää järjestölle tärkeitä työkaluja.

PUBLISHING PLATFORM FOR FINNRUNS ORGANIZATION

Viitaharju, Ville
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Business Information Systems
August 2015
Supervisor: Nieminen, Hans
Number of pages: 34
Appendices:

Keywords: Publishing platform, Web, Software development, Project Management

---

The purpose of this thesis was publishing system implementation to Finnruns-organization. The thesis goes through the implementation and outcome of the project.

The thesis describes the project as a whole and focuses on different browser-techniques. In addition, the thesis describes project most important and difficult situations.

The work resulted in fast to use an online publishing system that serves the needs of Finnruns organization. Publishing system is easy to use and contains important tools for the organization.

SISÄLLYS

1 JOHDANTO

Opinnäytetyönä toimiva web-pohjainen julkaisujärjestelmä on luotu Finnruns-järjestölle. Finnruns-järjestö on pelaajien oma keskittymä joka pyrkii saattamaan yhteen ns. ”speedrunnaajia”, eli pelaajia jotka pelaavat pelejä nopeasti ja hyödyntäen pelien jokaista ominaisuutta sekä ohjelmointivirhettä minimoidakseen käytettävän ajan pelien läpipelaamiseen.
Järjestelmä on eritäin tärkeä työkalu järjestölle tiedon julkaisemisessa. Järjestöllä ei ole tällä hetkellä käytössä mitään julkaisujärjestelmää, joka toimisi järjestön toimintaperiaatteiden mukaan. Tarve oli saada muokattavissa oleva järjestelmä, jolla järjestö voi hallita aikatauluja sekä julkaista tietoa tulevista tapahtumista. Opinnäytetyö yrittää täyttää tämän aukon mahdollisimman käyttäjäystävällisellä tapaa, sekä olemalla selkeä ja helppokäyttöinen.

Julkaisujärjestelmää jatko kehitetään järjestön toiveiden mukaisesti eteenpäin.
Sivun logon on suunnitellut graafikko. Allekirjoittanut on tehnyt sivustoon sopivan tyylitiedoston ja käyttänyt logon mukaista väriteemaa koko sivustolla.

Julkaisujärjestelmää kehittää yksi ohjelmoija. Ohjelmoija myös hallitsee julkaisujärjestelmän palvelinratkaisusta sekä siihen liittyvistä asennuksista.

2 WEB-JÄRJESTELMÄN MÄÄRITTELY JA SUUNNITTELU

2.1 Vaatimusmäärittely

Vaatimusmäärittely vastaa kysymykseen mitä järjestelmältä vaaditaan. Järjestelmässä käytössä olevat toiminnallisuudet sekä ominaisuudet kuvataan vaatimusmäärittelydokumentissa. Vaatimusmäärittelydokumentista on hyvä löytyä sovellusta kuvaavia kaavoita joilla voidaan seurata sovelluksen toimintatapaa ja tämän jälkeen saadaan myös käsitys sovelluksen toiminnasta, joka voi auttaa sovelluksen yksityiskohtaisem-

paa suunnittelua ja toteutusta. Kaavioista saadaan sovellettua tietoa suunnittelun myöhemmässä vaiheissa. Tallaista tietoa on mm. miten sovelluksen kokonaisuus muodostuu ja minkälaisia moduuleita sovelluksessa käytetään. (Haikala, 2011)

Vaatimusmäärittelydokumentista olisi hyvä löytyä ainakin otsikko, tekijä, asiakas, vaatimuksen tyyppi ja kuvaus, suhde muihin vaatimuksiin, tarpeellisuus, testattavuus ja aika-arvio. Otsikkona toimii vaatimuksen nimi. Tekijänä toimii vaatimuksen kirjaaja, joka on yleensä asiakaspalavereissa mukana oleva projektipäällikkö tai sovelluksen kehittäjä. Asiakkaan nimi on hyvä myös merkitä ylös, jotta tiedetään keneltä tai mistä vaatimus on peräisin. Vaatimuksen tyyppi on yleisesti joko lisäys, muutos tai korjaus. Vaatimuksen kuvaus on vaatimusmäärittelydokumentissa yksi tärkeimpiä kohtia, koska tässä kerrotaan vaatimuksen käyttötapaukset, kaaviot, käyttäjätarinat, sekvenssikaaviot ja määrittely. Suhde muihin vaatimuksiin on hyvä tutkia etukäteen, jolloin voidaan ohjelmointiprojekti toteuttaa tehokkaimmalla mahdollisella tavalla eikä tehdä ylimääräsitä työtä mm. toteuttamalla sama ominaisuus useampaan eri paikkaan. Tarpeellisuus mittaa määrityksen tarpeellisuutta sekä prioriteettia. Jotkut vaatimukset voivat olla tärkeämpiä kuin toiset, jolloin vaatimuksen prioriteetti on myös suurempi. Testattavuudella tarkoitetaan miten vaatimuksen täyttäminen voidaan testata. Aika-arvio on karkea työarvio vaatimuksen valmistumisesta. (Haikala, 2011)

2.2 Suunnittelu

2.2.1 Tietokanta

Tietokannan suunnittelussa tulee ottaa huomioon mitä tietokannalta halutaan. Halutaanko säilyttää yksinkertaisia tietoja vai luoda relaatiorakenteinen tietokanta jossa tieto on sidoksissa toisiinsa. Tietokantaa suunnitellessa pitää ottaa myös huomioon sovelluksen rakenne. Jos sovellusta käytetään yksinkertaisilla kyselyillä niin toteutukseen riittää yleensä hyvin yksinkertainen tietokantarakenne tai vaihtoehtoisesti vain tekstitiedostoon kirjoitettava tieto joka luetaan sovellukseen ja josta tämän jälkeen tieto voidaan ylikirjoittaa. Jos sovelluksessa on monta eri osa-aluetta jotka ovat toisiinsa yhteyksissä, on tällöin relaatiotietokanta parempi ratkaisu. Relaatiotietokantaa

luodessa on hyvä suunnitella tietokannan rakenne siten, ettei tietoa esitetä useampaan otteeseen jos sille ei ole tarvetta. Esimerkiksi osoiteluetteloa luodessa voidaan vähentää turhan tiedon lisäämistä tekemällä mm. kaupungista oma taulu joka sen jälkeen linkitetään jokaiseen osoiteluettelon tietokantariviin. Tällöin ei tarvitse kirjoittaa jokaiselle riville kaupunkia erikseen, vaan voidaan karsia turha tieto pois käyttämällä relaatiota.

Tietokantasovellusta valittaessa kannattaa tutkia internetistä omaan tarpeeseen sopivaa tuotetta, koska erilaisia tietokantasovelluksia on hyvin monta ja ne sopivat erilailla erilaisiin tarpeisiin. Tietokantaa valittaessa tulee myös ottaa huomioon hinta. Mm. MariaDB on ilmainen kaupallisessa käytössä, mutta mm. Mysql, Oracle sekä Microsoftin SQL Server ovat maksullisia kaupallisessa käytössä (ks. kuva 1) (Parthian Systems, 2014).

Database Licensing

Database	Version	Vendor	Software License	Cost
DB2	10.5 Enterprise Server Edition	IBM	Proprietary	**$122,000**
Informix	Version 12 Enterprise Edition	IBM	Proprietary	**$7,232** (annually)
MariaDB	MariaDB 10	MariaDB	GPL - Open Source	**$0**
MySQL	Enterprise Edition	Oracle	Dual Licensing Model	**$5,000 - $15,000**
Oracle	12c Standard Edition	Oracle	Proprietary	**$70,000**
Oracle	12c Enterprise Edition	Oracle	Proprietary	**$190,000**
PostgreSQL	9.3.4	PostgreSQL	PostgreSQL License (Open Source)	**$0**
SQL Server	SQL Server 2012 Standard Edition	Microsoft	Proprietary	**$28,688**
SQL Server	SQL Server 2012 Enterprise Edition	Microsoft	Proprietary	**$109,984**
Sybase	Adaptive Server Enterprise (ASE)	SAP	Proprietary	**$66,000** (core/base without options)

Kuva 1. Kuvankaappaus lisenssihinnoittelusta

2.2.2 Käyttöliittymä

Käyttöliittymää luodessa tulee huomioida sovelluksen tarve ja käyttökohde. Nykypäivänä moni verkkokäyttäjistä selailee verkkosivuja erilaisilla mobiililaitteilla mm. tableteilla ja matkapuhelimilla. Nykypäivänä sivuston käyttöliittymä kannattaa rakentaa mobiilikäyttäjiä ajatellen. Tämä voidaan toteuttaa tekemällä käyttöliittymä jonka eri elementit siirtyvät ja suurentuvat jos sivustolle tullaan mobiililaitteilla. Tähän hyvänä esimerkkinä on mm. valikoiden rakentaminen. Tietokoneella selaillessa sivustoa on helppo käyttää isommankin valikon kanssa, mutta esimerkiksi matkapuhelimella leveän valikon käyttö voi olla hankalaa. Tämän voi ratkaista esimerkiksi rakentamalla valikkoratkaisun joka aukeaa ja sulkeutuu nappia painamalla. Valikon elementit voidaan tämän jälkeen listata allekkain jolloin käytetään korkeussuunnassa mobiililaitteen mittoja hyväksi. (Usability, 2015)

Käyttöliittymää rakentaessa pitää myös ottaa huomioon käyttöystävällisyys. Toiminnot pitäisivät olla mahdollisimman yksinkertaisesti saatavilla ilman että käyttäjä joutuu pohtimaan kuinka jokin toiminto toimii. Yleisesti kaikki napit jotka kuuluvat samaan ryhmään, pitäisivät olla ryhmitettynä mieluusti samassa paikkaa, eikä esimerkiksi eripuolella sivustoa. (Usability, 2015)

2.2.3 Ohjelmistoarkkitehtuuri

Ohjelmistoarkkitehtuuri määrää ohjelman toteutustavan ja osittain myös käytettävät ohjelmistokirjastot. Ohjelmistoarkkitehtuuri kattaa ohjelmiston jaon tärkeimpiin osiin ja näiden osien väliset suhteet, ohjelmiston rakenteen ja käyttäytymiset, erilaiset näkökulmat ja tasot, korkeamman tason suunnitteluratkaisut ja niiden perustelut sekä ohjelmiston kehittämisen säännöt ja periaatteet. Vaikka ohjelmistoarkkitehtuuria ei määriteltäisi erikseen, on kuitenkin jokaisessa sovelluksessa olemassa oma ohjelmistoarkkitehtuuri, eli ohjelmiston toimintatavat ja eri komponenttien rakenne. (Ohjelmistoarkkitehtuurit, 2015)

Ohjelmistoarkkitehtuuriin voidaan liittää myös ulkopuolisia ohjelmakirjastoja. Ohjelmistokirjastot eivät ole pakollisia ohjelmassa, mutta yleensä helpottavat toteutusta.

Ohjelmaan kannattaa valita hyvät ohjelmistokirjastot joiden kanssa työskentely on nopeaa ja tehokasta. Ohjelmistokirjastoja valittaessa kannattaa huomioida ovatko ne liian monimutkaisia ohjelmaan vai jopa liian suppeita. Kannattaa myös tehdä vertailua erilaisten ohjelmistokirjastojen välillä. Muuan vertailussa oli projektissa käytettävä TinyButStrong sisältänyt paljon ominaisuuksia mitä kaikki kirjastot eivät täyttäneet (Wikipedia, 2015). Ohjelmistokirjastoja löytyy myös täysin valmiina paketteina, kuten CakePHP, joka sisällyttää ohjelmistokirjastoja tietokantahakuihin, HTML-sivujen luontiin sekä monia muita lisäosia (CakePHP, 2015).

3 SELAINTEKNIIKAT

3.1 PHP

PHP (Hypertext Preprocessor) on web-ohjelmointiin suunnattu ohjelmointikieli. PHP on vapaan lähdekoodin ohjelmointikieli. Koska PHP on vapaan lähdekoodin ohjelmointikieli, sillä kehitys on ilmaista ja ei vaadi lisenssiä edes kaupallisessa tarkoituksessa. PHP on hyvin laajasti käytetty ohjelmointikieli, joten uusia kehittäjiä on helppo löytää. (Pitts, 2015). Tästä syystä myös ohjeita on hyvin helppo löytää internetistä laajan yhteisön takia. Verkkosivu http://php.net on PHP:n virallinen sivusto dokumentointia varten, josta löytyvät kaikki PHP:n tukemien funktioiden dokumentoinnit sekä lähes kaikille funktioille on hyvät käyttäjäkommentit, joissa on esimerkkejä ja joskus suoraan valmiita ohjelmointipätkiä. PHP tukee monia tietokantoja ilman erilliskirjastoja (MySQL, Oracle, MariaDB). Projektissa käytössä on PHP versio 5.5, joka oli projektin aloitushetkellä uusin vakaa versio.

PHP on myös hyvin laajalti tuettu, lähes jokaiselle käyttöjärjestelmälle löytyy PHP-ohjelmointikieleen vaadittavat palvelinohjelmistot. PHP:tä tukevia käyttöjärjestelmiä ovat mm. eri Linux jakelut, Microsoftin Windows sekä Applen OSX. PHP on kolmanneksi suosituin ohjelmointikieli tällä hetkellä (Programming Language Popularity, 2015).

3.1.1 php-activetecord

Kirjaston oman verkkosivuston on osoitteessa http://www.phpactiverecord.org/. php-activerecord on samannimisen ruby-kirjaston php-ohjelmointikielellä toteutettu oliopainoitteinen ohjelmointikirjasto tietokantahakuja ja -käsittelyitä varten. Tietokantahakuja yksinkertaistetaan tekemällä jokaisesta tietokannan taulusta olio, jota voidaan käsitellä kuin normaalia oliota. Esimerkiksi, kun haetaan kaikki games–nimisen taulun pelit, pitää avata tietokantayhteys, määritellä SQL-lause hakemiseen ja siirtää tulos muuttujaan. Edellisen asemesta asia voidaan tehdä yksinkertaisesti kuvan 2 osoittamalla tavalla.

```
$games = \Game::all(array("order"=>"name asc"));
```

Kuva 2. Tietokantahaku taulusta games

Ilman kirjastoa tietokantahaku on huomattavasti monimutkaisempi, kuten kuvasta 3 voidaan todeta. Kyselyssä joudutaan määrittelemään erikseen tietokantayhteys, jonka jälkeen kirjoittaa käsin tietokantakysely. Esimerkissä on kirjoitettu pelkästään nopea hakukysely joka ei ota huomioon tietokantarelaatioita. Kyselyn jälkeen joudutaan tulokset käymään läpi yksi kerrallaan ja lisäämään ne lista muuttujaan nimeltä $list. Viimeisenä joutuu sulkemaan tietokantayhteyden erikseen.

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT * FROM Games";
$result = $conn->query($sql);
$list = array();
if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        $list[] = $row;
    }
}

$conn->close();
?>
```

Kuva 3. Kuvankaappaus tietokantakyselystä ilman kirjastoa

Ohjelmistokirjasto nopeuttaa tietokantakäsittelyjä ohjelmoijan näkökulmasta poistamalla ylimääräisiä vaiheita. Kirjaston tulevaisuus on tosin kysymyksen alla, koska kirjaston vakaa versio on päivitetty viimeksi 27.05.2010. Projektissa on käytössä ns. ”nightly build” versio, joka on päivitetty 16.04.2013.

3.1.2 TinyButStrong

Kirjaston oman verkkosivuston on osoitteessa http://www.tinybutstrong.com/. TinyButStrong (lyh. TBS) on kirjasto jolla voi toteuttaa dynaamisia HTML- ja XML-rakenteita. Tässä projektissa käytetään tätä pelkästään HTML-rakenteiden muodostamiseen. Perinteisesti PHP-ohjelmointikieltä kirjoittaessa, liitetään näytettävälle sivulle PHP-ohjelmointikieltä, mutta TBS avulla voidaan ohjelmoida toiminnallisuus muualle ja luoda pelkän näytettävän sivun ilman PHP-ohjelmointikieltä. Tämä helpottaa dynaamisen rakenteen luomista ja selventää ohjelmoijan työtä, koska sillä saadaan lajiteltua ohjelmointikokonaisuuksia eri tiedostoihin.

3.2 Front end

3.2.1 Bootstrap

Verkkosivu http://getbootstrap.com/. Bootstrap on suosittu tyyliasu/Javascript runko web-pohjaisia sivuja varten. Bootstrapin avulla voidaan luoda helposti dynaamisia sivuja jotka skaalautuvat hyvin riippuen käytettävän näytön koosta. Bootstrap pienentää ja suurentaa elementtejä automaattisesti sen mukaan minkäkokoisella näyttöpäätteellä sivustoa katsellaan. Tämä on hyvin tärkeää nykypäivänä koska tänä päivänä käytetään paljon mobiililaitteita (matkapuhelimet, tabletti-laitteet yms.) internetissä (Bosomworth, 2015).

3.2.2 JavaScript

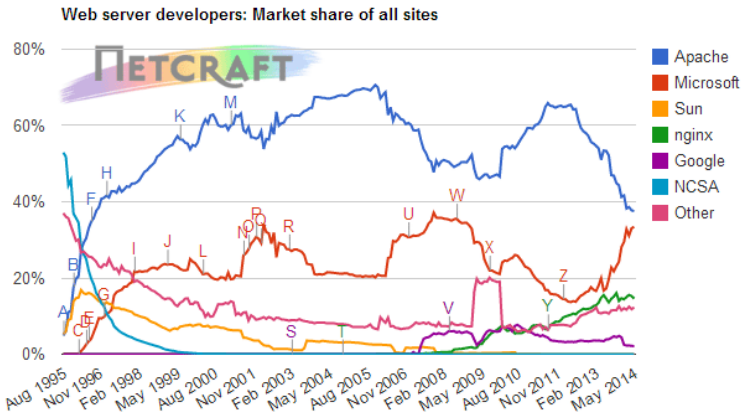
Projektissa käytetään erilaisia Javascript-pohjaisia kirjastoja. Tunnetuin näistä on jQuery (https://jquery.com/) sekä jQuery UI (https://jqueryui.com/). JQuery helpottaa JavaScriptin kirjoittamista tarjoamalla yksinkertaisempia ilmaisuja. Tämä helpottaa ohjelmoijan työtä sekä saattaa jossain tapauksissa nopeuttaa sivuston käyttöä, jos ohjelmoija on tehnyt saman ominaisuuden huonommin kuin mitä jQuery tarjoaa. JQuery UI on nimensä mukaisesti keskitetty enemmän käyttöliittymän ohjelmointiin. Kirjastolla saadaan tehtyä animaatioita ja erilaisia käyttöliittymän ominaisuuksia helposti mm. vetolaatikoita, päivämäärän valitsimia, erilaisia toimintanappeja ja paljon muuta.

4 KÄYTETTÄVÄT VÄLINEET

4.1 Palvelinympäristö

4.1.1 Nginx

Palvelinohjelmistoa valittaessa oli vaihtoehtoina joko Apache Software Foundationin tarjoama Apache- tai nginx-palvelinohjelmisto. Apache on suosittu palvelinohjelmisto, mutta uudempi nginx on kerännyt hyvin suosiota vuosien varrella, kuten kuva 4 osoittaa.



Developer	April 2014	Percent	May 2014	Percent	Change
Apache	361,853,003	37.74%	366,262,346	37.56%	-0.18
Microsoft	316,843,695	33.04%	325,854,054	33.41%	0.37
nginx	146,204,067	15.25%	142,426,538	14.60%	-0.64
Google	20,983,310	2.19%	20,685,165	2.12%	-0.07

Kuva 4. viitattu http://www.itworld.com/article/2695421/choosing-a-linux-web-server-nginx-vs-apache.html

Syy nginxin valintaan löytyy henkilökohtaisista keskusteluista IRC-verkossa muiden ohjelmistokehittäjien kanssa, sekä faktasta että alusta on uusi minulle, joka tarjoaa samalla uutta opittavaa.

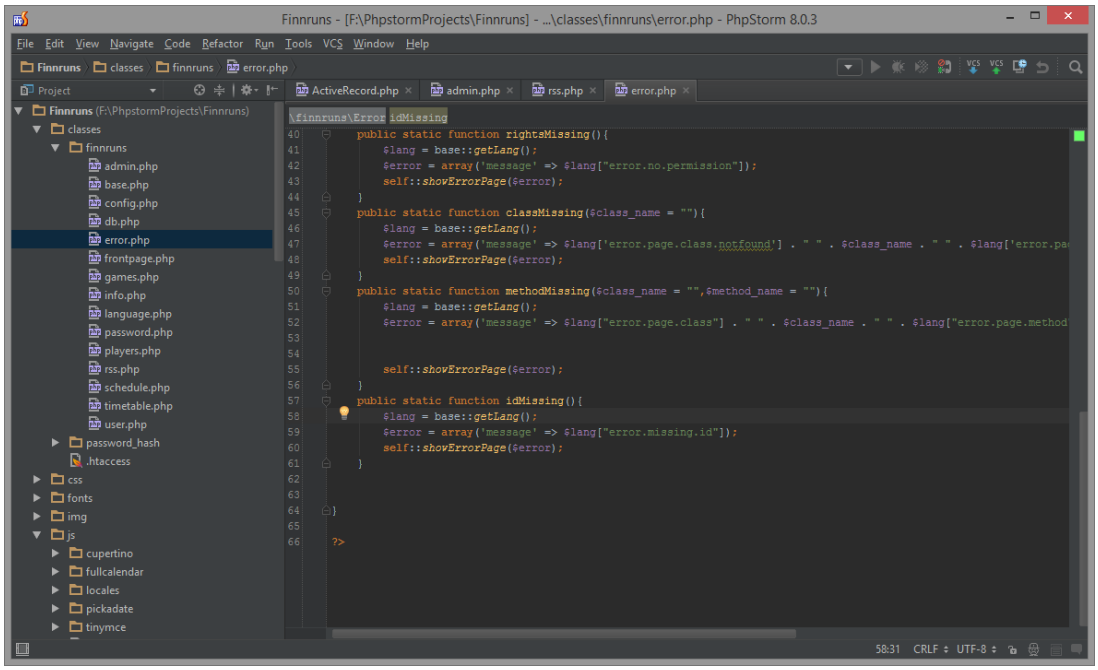
4.1.2 MariaDB

MariaDB:n verkkosivusto löytyy osoitteesta https://mariadb.org/. Vaihtoehtoina projektille oli käyttää joko tuttua ja turvallista MySQL-tietokantaa, tai uudempaa MariaDB-tietokantaa. MariaDB-tietokanta on pohjautunut MySQL-tietokantaan ja kumpaakin on kehittänyt sama henkilö nimeltä Monty Widenius. MySQL myytiin Sun Microsystemsille 1 miljardin kauppahintaan. Monty oli sitä mieltä että Sun oli parasta mitä olisi MySql:lle tapahtua koska hän uskoi että Sun voisi auttaa avoimen lähdekoodin kanssa vielä enemmän. Myöhemmin Oracle osti Sun Microsystemsin. Tämän seurauksena Monty päätti rakentaa uuden tietokannan ja nimesi sen hänen toisen tyttärensä Marian mukaan. (MUELLER, 2013). MariaDB on myös hiukan nopeampi kuin MySQL, sitä kehitetään aktiivisemmin ja vakaita versioita testataan tarkemmin ennen julkaisua. MariaDB on myös ilmainen tietokanta jopa kaupallisesti käytettynä (MariaDB, 2015).

4.2 Työkalut

4.2.1 PhpStorm IDE

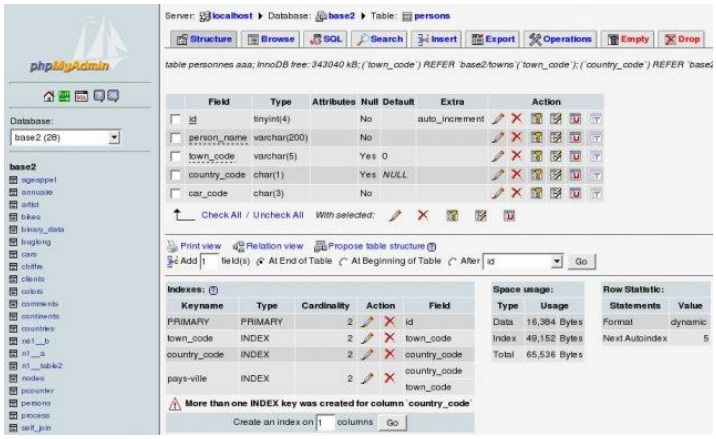
PhpStorm:in verkkosivusto löytyy osoitteesta https://www.jetbrains.com/phpstorm/. JetBrainssin PhpStorm on kaupallinen monella alustalla toimiva ohjelmistokehitystyökalu. Se tarjoaa tehokkaan reaaliaikaisen koodin analyysin, virheiden eston sekä automaattisen refaktoroinnin PHP, HTML ja JavaScript ohjelmointikielille. PhpStorm on saman yrityksen tuote kuin suosittu ohjelmointiympäristö Javalle nimeltä IntelliJ IDEA. PhpStorm tarjoaa myös edullisemman lisenssin opiskelijoille. Syy PhpStormin hankintaan oli sen kehittyneet reaaliaikaiset koodin analyysit. Saman tyyppiset ominaisuudet ovat myös avoimen lähdekoodin Eclipsessä, mutta Eclipse ei sisällä suoraan näin kattavia työkaluja. Kuvassa 3 nähdään PhpStormin käyttöliittymä.



Kuva 5. kuvankaappaus PhpStorm:ista

4.2.2 Phpmyadmin

Phpmyadminin verkkosivusto löytyy osoitteesta http://www.phpmyadmin.net/. Phpmyadmin on web-pohjainen tietokannan visuaaliseen käyttöön tarkoitettu ilmainen ohjelmisto. Tämän ohjelmiston avulla tietokantaan liittyvät nopeat testaukset ja muutokset ovat helppo tehdä, kuten myös tietokannan varmuuskopio ja palautus ovat nopeita. Phpmyadmilla on myös helppo hallita tietokantayhteyksiä erilaisten taulujen välillä. Phpmyadmin on myös suoraan asennettavissa useimmilla Linux-käyttöjärjestelmissä. Kuvassa 6 nähdään phpmyadminin käyttöliittymä.



Kuva 6. kuvankaappaus phpmyadmin:ista

4.2.3 WAMP

WAMP:in verkkosivusto löytyy osoitteesta http://www.wampserver.com/en/. Lokaalissa testiympäristössä on käytössä WAMP-palvelinympäristö (Windows Apache, MySQL, PHP). WAMP:in valinta tähän käyttöön oli helppo, koska ohjelmointiympäristö toimii myös Apachen ja MySQL:n kanssa lähes saman lailla kuin Nginx ja MariaDB:n. Ainoa käytännön ero on Apachen ja Nginxin välillä, koska projektissa käytetään uudelleensiirtoa, niin joudutaan se määrittelemään erilailla palvelinalustasta riippuen. Apache käyttää pääkansioon sijoitettua htaccess tiedostoa taas kun nginx palvelinympäirstössä pitää uudelleensiirtoa varten lisätä muutama rivi palvelinsovelluksen asetustiedostoon. WAMP sisältää myös pienen hallintaliittymän johon pääsee käsiksi Windowsissa oikean kulman ikoneista.

5 TOTEUTUS

5.1 Pohjan rakentaminen

5.1.1 Bootstrap

Käyttöliittymä rakennetaan Twitterin kehittäjien luoman Bootstrap HTML, CSS ja Javascipt rungon päälle, koska kyseinen runko on erittäin hyvin dokumentoitu ja monessa paikkaa käytössä. Käyttöliittymä tehdään suurimmaksi osaksi käsin ohjelmoimalla tyyliasut sekä sivustossa käytettävät elementit. Bootstrap liitetään projektiin lataamalla ja lisäämällä Bootstrapissä käytettävät tyylitiedosto sekä Javascript-tiedosto projektissa käytettävään HTML-tiedostoon kuvan 5 osoittamalla tavalla.

```
<link rel="stylesheet" href="[var.htmldir]/css/bootstrap.min.css"/>
<script src="[var.htmldir]/js/bootstrap.min.js"></script>
```

Kuva 7. Kuvankaappaus projektista

Kuvassa 7 ensimmäisellä rivillä lisätään Bootstrapissä käytettävän tyylitiedoston projektiin ja seuraavalla rivillä lisätään Bootstrapissä käytetävän Javascript-tiedoston projektiin. Tämän lisäksi kirjoin omaa CSS–tyylitiedostoa rinnalla jota käytin projektissa.

Bootstrap ei tarjoa suoraan muokattavaa teemaa vaan tämä pitää joko tehdä itse tai ladata netistä valmis teema, joka tässä tapauksessa ei käy. Bootstrapissä tuleva Javascript-kirjasto on vain käyttöliittymän kannalta hyödyllinen, joten se ei tarjoa kaikkia tarvittavia työkaluja, joten nämäkin pitää rakentaa itse.

5.1.2 TinyButStrong integrointi

TinyButStrong, eli lyhyesti TBS, tarjoaa hyvän ympäristön rakentaessa PHP-ohjelmointikielellä MVC mallin mukaisia sivustoja.
MVC eli model-view-controller (suom. malli-näkymä-ohjain) on ohjelmistoarkkitehtuurityyli jonka tarkoituksena on erottaa ohjelman vaadittava koodi sekä käyttöliittymä. Tämä helpottaa jälkeenpäin sivustojen käyttämistä ja muokkaamista kun luoduilla HTML-sivuilla ei ole käytössä riviäkään PHP-ohjelmointikielellä tehtyjä rivejä. TBS integrointia varten täytyy ladata TBS kotisivuilta asennuspaketti ja siirtää se projektissa käytettävään kansioon, tässä tapauksessa alikansio /tbs/. Tämän jälkeen voidaan tarvittaessa ladata TBS käyttöön kun sitä tarvitaan. Jokainen sivu ei tätä tarvitse, joten se otetaan käyttöön vasta tarvittaessa eikä ladata sitä joka kerta. TBS otetaan käyttöön kuvan 6 osoittamalla komennolla.

```
require_once(__BASEDIR__ . "/tbs/tbs_class.php");
require_once(__BASEDIR__ . "/tbs/plugins/tbs_plugin_plus.php");
$TBS = new \clsTinyButStrong();
```

Kuva 8. Kuvankaappaus projektista

Kuvassa 8 ensimmäinen rivi lataa TBS luokkatiedoston. Tämän jälkeen ladataan TBS plus-lisäosa käyttöön, tämä antaa lisää ohjelmointivaihtoehtoja HTML-sivuja tehdessä, kuten ope-operaattorin jolla voi toteuttaa erilaisia nopeita kyselyitä. Viimeisellä rivillä alustetaan TBS muuttujaan $TBS, jotta voidaan käyttää TBS-luokkaa.

5.1.3 ActiveRecords integrointi

Projektissa käytettävä ActiveRecords auttaa huomattavasti tietokantahakuja tehdessä. Tämä nopeuttaa ohjelmointia ja antaa ohjelmoijalle paremmat työkalut tietokantayhteyttä luodessa. ActiveRecords ladataan joka sivunlatauksella koska tietokantaa käy-

tetään lähestulkoon jokaisessa sivussa. Latauksen jälkeen jokainen tietokantataulu pitää määritellä omaksi luokaksi. Tämä määrittely on pienimmillään muutaman rivin mittainen, kuten kuva 7 osoittaa.

```
class Page extends ActiveRecord\Model
{
    static $primary_key = "id";
}
```

Kuva 9. Kuvankaappaus projektista

Kuvassa 9 ensimmäisellä rivillä annetaan luokalle nimi, jonka jälkeen luokka laajentaa luokkaa ActiveRecod\Model. Tämä antaa tässä tapauksessa luokalle Page kaikki ohjelmistokomennot mitä luokka ActiveRecord\Model pitää sisällään. Kolmannella rivillä määritellään tietokannassa löytyvän pääavaimen nimi. Muuta ei tarvitse ohjelmoijan tehdä tässä tapauksessa, vaan ActiveRecord hoitaa loput käsittelyt.

5.1.4 Monikielisyyden rakentaminen

Projektia on tarkoitus käyttää myös muilla kielillä, joten ns. kielipaketin tekeminen oli paras ratkaisu. Projekti sisältää sekä kielet suomi ja englanti. Valittua kieltä voi nopeasti vaihtaa käyttöliittymässä painamalla kyseisen maan lippua. Kieli tallentuu selaimessa käytettävään evästeeseen, jolloin se myös on tallessa seuraavan kerran sivulle tullessa. Oletuskielenä toimii Suomi. Kielipaketteja luodessa oli useita vaihtoehtoja miten nämä luodaan. Pienen pohdinnan jälkeen päädyin tulokseen käyttää palvelimelle sijoitettavien tiedostoja, koska tiedostojen lukunopeus on suuri ja kielitiedostoja joudutaan lukemaan lähes joka sivunlatauksen yhteydessä. Toisena vaihtoehtona olisi ollut käyttää tietokantaan sijoitettavaa taulua jossa olisi eri kielille omat rivinsä, mutta tämän päivitys ja lataaminen olisi ollut paljon työläämpi operaatio kuin mitä suoraan tiedostosta luku. Tiedostot ovat projektissa nimellä fi.lang sekä en.lang. Näiden tiedostorakenne on seuraavanlainen:

avain=kielikohtainen muunnos

toinen_avain=kielikohtainen muunnos

jne.

Avain tässä tapauksessa on ohjelmoinnin aikana käytettävä sana joka luetaan HTML-tiedostossa TinyButStrongia käyttämällä ja palautetaan sen kielikohtainen muutos.

Tämä mahdollistaa sen että HTML-tiedostoa luodessa ei tarvitse tehdä useampaa tiedostoa eri kielille, vaan voidaan käyttää samaa tiedostoa ja vaihtaa vain ohjelmallisesti käytettävä kieli. Tämä ratkaisu auttaa myös kirjoitusvirheiden kanssa, koska jos tekisin useamman tiedoston jossa käyttäisin samaa kirjoitusvirhettä, joutuisin muokkaamana jokaista tiedostoa erikseen. Nykyisellä järjestelmällä voidaan muokata kirjoitusvirhettä suoraan tekstitiedostosta ja se muuttuu jokaiselle sivulle jossa kyseistä käännöstä on käytetty.

5.1.5 Tyylin rakentaminen

Vaikka Bootstrap tarjoaa hyvän pohjan, on se silti vain pohja. Oman tyylin rakentaminen tuo sivulle oman ammattimaisemman ilmeen ja persoonallisuuden tunteen. Minulla oli projektia tehdessä vapaat kädet toteuttaa haluamani tyyliasu. Mielessäni oli ollut jo pitkän aikaa idea teemasta, jossa vasemmalla puolella olisi isot napit navigaatiota varten ja jolloin jäisi yläosa sekä koko oikeapuoli sivujen käyttöön. Teeman kuvakkeet on luotu käyttämällä Fontawesome nimistä kuvake kirjastoa, joka toteuttaa kuvakkeet erittäin järkevällä tavalla. Fontawesome on lyhyesti kerrottuna oma fontti, jossa kirjainten tilalla on ikoneita. Näitä ikoneita voi tämän takia vapaasti suurentaa, pienentää ja värittää ilman että niiden laatu kärsii. Tämä oli erittäin hyvä apuväline teemaa luodessa.

Teemaa suunnitellessa ja luodessa tuli rakennettua monia prototyyppejä erilaisista ratkaisuista. Jotkut ratkaisut olivat parempia kuin toiset, mutta eivät silti toimineet halutulla tavalla. Ongelmia oli mm. teeman rikkoutumienne mobiilialustoilla sekä eri selaimilla saattoivat viivat ja kuvakkeet liikkua väärään suuntaan. Toteutin teemaa pääasiassa käyttämällä Googlen Chrome-selainta ja testasin sivuston Firefox sekä Internet Explorer selaimilla. Koska eri selaimet lukevat ja käyttävät sivuston tyyliasua hieman eritavalla, ovat sivustot pakko testata useammalla selaimella jos haluaa täydellistä tulosta. Tuntien pohdinnan, kokeilun ja korjauksien jälkeen päädyin nykyiseen teemaan joka toimii mobiililaitteilla sekä tietokoneilla. Sivuston ulkoasuista löytyi myös myöhemmässä vaiheessa pieniä vikoja, mutta ne olivat nopeita korjata ja liittyivät lähinnä alustoihin joita en voinut itse testata kuten Windows Phone ja sen käyttämä Internet Explorer selaimeen.

Teeman toimittua oli seuraavana ongelmana värimaailma. Testasin useita eri värejä keskenään ja lopulta päädyin rakentamaan värimaailmani oranssin sekä harmaan ympärille. Värimaailmaa auttoi myös miettimään kaverini, joka toteutti myös grafiikkaa Finnruns-tapahtumalle. Tämä ratkaisu miellytti myös loppukäyttäjiä ja sain rakennettua samalla myös erittäin kevyen ja selvän ulkoasun käyttäjiä varten. Samaa värimaailmaa käytettiin myös Finnruns-tapahtumassa muualla ja se sai paljon kiitosta selkeydestään sekä yksinkertaisuudestaan. Kuvassa 10 on kuvankaappaus valmiista projektista.

Kuva 10. Kuvankaappaus projektista

5.1.6 Haasteet mobiililaitteilla

Nykypäivänä internettiä selaillaan paljon mobiililaitteilla (Bosomworth, 2015). Sivuston ulkoasun ja käyttöliittymän pitää olla myös mobiiliystävällinen ja toimia samaan aikaan täyskokoisella tietokoneella. Tähän auttaa Bootstrap huomattavasti, koska siinä

käytettävät elementit mukautuvat sen mukaan minkäkokoisella laitteella käytetään sivustoa, jolloin ei tarvitse tehdä erillistä mobiiliteemaa vaan siihen riittää yksi ja sama teema joka toimii mobiililaitteilla sekä pöytätietokoneilla. Tämä oli yksi syistä minkä takia halusin tehdä käyttöliittymän jossa olisi isot napit vasemmalla ja kaikki tieto oikealla. Isot napit auttavat mobiilikäyttäjiä navigoimaan sivustolla helposti. Ongelmana tässä oli se, että sivupalkki vei ison osan ruudusta jolloin ei sisällölle jäänyt tilaa. Tämän ratkaisin piilottamalla sivupalkin automaattisesti kun sivustoa katsellaan laitteella jonka resoluutio on tarpeeksi pieni. Sivupalkin voi myös piilottaa tietokoneella klikkaamalla tapahtuman logoa (FINNRUNS2) vasemmasta yläkulmasta. Tällä samalla napilla saadaan myös mobiilikäyttöliittymässä sivupalkki avattua ja suljettua. Sivupalkin aukioloa indikoi oranssi nuoli logon vieressä, joka siirtyy vasempaan reunaan kun sivupalkki on kiinni, kuten kuvassa 11 voidaan nähdä.

Kuva 11. Kuvankaappaus projektista

5.1.7 JavaScript

Projektissa käytetään käyttöliittymässä runsaasti Javascriptiä nopeuttamaan käyttöä sekä antamaan parempi käyttökokemus. Javascriptin kanssa käytetään pääosin Jquery-kirjastoa joka helpottaa ja nopeuttaa Javascriptin kirjoittamista. Projektissa käytetään myös monia muita kirjastoja helpottamaan ohjelmointia.

Picker	Päivämäärien sekä kellonaikojen valintaan käytettävä kirjasto
Moment	Päivämääräobjektin luontiin tarkoitettu kirjasto.
Jquery Tablesorter	Taulujen järjestelyyn tarkoitettu kirjasto
Jquery UI	Käyttöliittymän parannukseen tarkoitettu kirjasto
Selectize	Valintoihin tarkoitettu kirjasto (käyttö hallintaympäristössä)

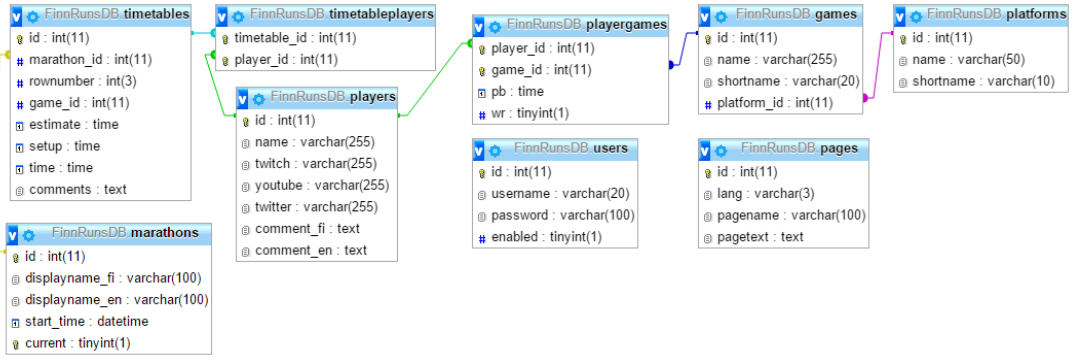
Taulukko 1. Lista käytettävistä kirjastoista

Javascriptistä löytyy natiivisti päivämääräobjektin luontiin tarkoitettu luokka, mutta tämä luokka vaihtelee selaimesta riippuen ja ei anna samaa tulosta jokaisella selaimella esim. Chrome-selaimella luodessa päivämääräobjektia, antaa selain päivämäärän kellonajaksi nykyisen aikavyöhykkeen ajan, kun taas Firefox selaimella selain antaa aikavyöhykkeen +0 kellonajan.

5.1.8 Tietokanta

Timetables	Sisältää maratooniin liittyvät rivitiedot mm. Pelien ID sekä käytettävän ajan ja lopullisen ajan.
Players	Sisältää pelaajatiedot. mm. Nimi ja sosiaalisen median linkit.
Games	Sisältää pelien tiedot mm. Nimi ja lyhytnimi. Sisältää myös platform tauluun linkityksen
Pages	Sisältää sivujen tekstin. Omat rivinsä jokaiselle kielelle. (näitä pystyy muokkaamaan käyttöliittymässä)
Platforms	Pelialustojen tiedot. Nimi ja lyhytnimi.
Playergames	Linkitys pelaajien ja pelien välillä. Tämän avulla voidaan linkittää pelaajia ja pelejä yhteen. Tälle välitaululle tallentuu myös pelaajien omat henkilökohtaiset peliennätykset.
Users	Käyttäjätaulu. Tässä on käyttäjien tunnukset ja salasanat
Timetableplayers	Linkitys aikataulun ja pelaajien välillä
Marathons	Maratoonit yksilöivä taulu

Taulukko 2. Tietokannan rakenne



Kuva 12. Tietokannan rakenne

Taulukosta 2 käy selväksi tietokannan rakenne. Tietokannan linkitykset (kuva 12) hoidetaan ActiveRecord-ohjelmistokirjastolla eri taulujen välillä. Tietokanta on suunniteltu ja rakennettu tätä projektia varten, joten tämä sama tietokantarakenne ei ole missään muualla käytössä. Timetables-taulu on suurin taulu joka linkittää kaikki sarakkeet

aikatauluja varten yhteen. Tästä luetaan ja sen jälkeen luodaan sivuilla käytettävä aikataulu. Players-taulusta saadaan kaikkien pelaajien tiedot pelaajasivuille. Pages-taulusta saadaan sisältö etu- sekä tietoja-sivulle. Pages-taulusta löytyy suomenkielinen sekä englanninkielinen rivi kummastakin sivusta.

5.2 Sivut

5.2.1 Etusivu- ja Tietoja-sivut

Etusivu ja Tietoja sivujen rakenne on hyvin samanlainen. Kummankin sivuston sisältö ladataan tietokannasta käyttäen hakuehtoina joko ”frontpage_view” tai ”info_about” sekä antamalla oikea käyttöliittymän kieli. Kumpaakin sivua pystyy hallintaliittymässä muokkaamaan joten jokainen jolla on sivustolle tunnukset pystyvät muokkaamaan sivujen sisältöä.

```php
<?php
namespace finnruns;
class Frontpage
{
    public $TBS;

    public function view()
    {
        $TBS = Config::loadTBS();

        $page = \Page::find(1,
            array('conditions' => array("pagename=? AND lang=?", "frontpage_view", __LANGUAGE__),
                'select' => "pagetext"));

        $TBS->MergeField("dbtext", $page);
        $TBS->Show();
    }

}
?>
```

Kuva 13. Kuvankaappaus projektista

Kuvasta 13 käy ilmi etusivun-luokka. Luokka ladataan etusivua hakiessa (tietoja-sivun luokka on lähes samanlainen). Luokkaan määritellään nimiavaruus ja luokan nimi. Tämän jälkeen ladataan TBS muuttuja asetusluokasta, haetaan $page muuttujaan tietokannasta sivun sisältö, liitetään sisältö sen jälkeen HTML-sivuun komennolla

$TBS->MergeField ja näytetään sivu komennolla $TBS->Show. Jälkimmäinen komento kasaa sivuston ja tulostaa käyttäjälle lopullisen sivun.

```
<ul class="breadcrumb">
    <li><a href="[var.htmldir]/">[lang.frontpage]</a></li>
</ul>
<div class="row">
    <div class="col-lg-12">
            [dbtext.pagetext;noerr;strconv=no]
        </div>

</div>
```

Kuva 14. Kuvankaappaus projektista

Kuvasta 14 on esitettynä etusivun HTML-tiedosto kokonaisuudessaan. Tämä liitetään main.html-tiedostoon, jossa on loput sivustosta. Ylimmäisessä UL-elementissä on määriteltynä etusivun osoite sekä annettu linkille kielipaketista teksti avaimella ”frontpage”. Tämän alapuolella on kaksi yksinkertaista div-elementtiä joiden sisällä on PHP-tiedostossa annettu sivuston sisältö. Tämä tulostetaan sivulle käyttäen TinyButStrongia. Tämä on hyvä esimerkki kuinka pienellä työllä saadaan luotua kokonaisia sivuja käyttäjälle ilman että täytyy luoda jokaista sivua erikseen tai sekoittaa PHP-ohjelmointikieltä HTML-sivuun.

5.2.2 Pelaajat-sivu

Pelaajat sivu listaa yksinkertaisuudessaan kaikki pelaajat siistiin listaan josta voi nähdä kaikki maratooniin osallistuvat pelaajat. Pelaajanimeä painamalla voidaan avata pelaajanimen alapuolelle pieni informaatio laatikko, josta löytyy pelaajan Twitch-käyttäjätunnus, Youtube-käyttäjätunnus sekä Twitter-käyttäjätunnus. Informaatiolaatikon avaus on toteutettu Javascriptillä. Nämä helpottavat mainostamaan pelaajia sekä uudet fanit voivat käydä katsomassa pelaajien sosiaalisen median sivustoja helposti ja nopeasti. Pelaajanimen vieressä on harmaa kuvake, joka indikoi onko pelaajalla ns. pelistriimi päällä, eli lähettääkö hän pelikuvaa internetin välityksellä. Kuvake muuttuu vihreäksi jos pelistriimi on päällä. Tämä on toteutettu käyttämällä Twitch.tv:n API komentoa jolla voidaan yhdellä kyselyllä hakea kaikkien pelaajien tila Twitch.tv palvelussa. Tämä toteutetaan suoraan loppukäyttäjän selaimella Javascript-komennolla.

```
var nick_list = "[nick_list]";

$.ajax({
    url: 'https://api.twitch.tv/kraken/streams?channel=' + nick_list,
    dataType: 'JSONP',
    jsonpCallback: 'callback',
    type: 'GET',
    success: function (data) {
        $.each(data.streams, function () {
            $("[data-playerid='" + this.channel.name + "']").addClass("player-online");
        });
    }
});
```

Kuva 15. Kuvankaappaus projektista

Kuvassa 15 olevalla Javascript komennolla saadaan tietoon pelaajien tilan. [nick_list] on TBS muuttuja jonka arvona on kaikkien pelaajien Twitch käyttäjätilin tunnus pilkuilla eroteltuna. Tämä tieto annetaan Ajax kyselyssä Twitch.tv:n API komennolle parametrinä jolloin saadaan kaikki nykyiset pelaajat listaan. Tämän jälkeen ”success” komennossa käydään kaikki ”streamit” läpi jonka jälkeen lisätään harmaalle kuvakkeelle lisäluokka ”player-online” jos pelaajalla on pelistriimi päällä, jolloin kuvake muuttuu vihreäksi. Tämä onnistuu helposti kun jokaiselle kuvakkeelle on määritelty HTML5 syntaksin mukana tullut data-arvo, tässä tapauksessa data-playerid. Jqueryä käyttämällä saadaan helposti oikean kuvakkeen käsittelyyn kyselyllä, joka etsii data arvosta oikeaa pelaajatunnusta.

Pelaajasivun HTML-rakenne on tehty dynaamiseksi, eli ohjelmoidaan vain yhden kerran rivi, eikä siten jokaista pelaajaa varten tarvitse luoda uutta riviä. Tämä on toteutettu käyttämällä TBS syntakseja. Jotta tämä onnistuisi on PHP-ohjelmointikielen puolella on pelaajalista oltava lisättynä sivulle komennolla $TBS->MergeBlock. Tämän jälkeen voidaan käyttää listaa HTML-sivussa seuraavalla tavalla

```
[player.name;block=fieldset;magnet=fieldset;sub1=playergames]
```

Kuva 16. Kuvankaappaus projektista

Kuvassa 16 on lista nimetty nimellä ”player”. Player listasta löytyy kenttä nimeltä ”name”, joka sisältää pelaajan nimen. Puolipisteellä eroteltu Block-argumentti ilmaistaan mihin elementtiin on kyseinen rakenne sidottu. Tässä tapauksessa elementti on Fieldset. Seuraavana puolipisteellä on eroteltuna magnet. Magnet kertoo mikä elementti piilotetaan jos ”player” niminen lista on tyhjä. Tämä on hyvä tapa lisätä erilaista

tietoa sivulle ja poistaa elementti näkyvistä käyttäjälle jos siellä ei ole mitään informaatiota. Viimeisenä puolipisteellä eroteltuna on sub1 argumentti. Tällä voidaan ilmaista onko listassa jotain ns. alilistaa, eli lista joka on listan sisällä. Tässä tapauksessa on olemassa alilista ”playergames”, joka sisältää jokaisen pelaajan henkilökohtaisen pelilistan sekä parhaat peliaikansa. Jos halutaan näyttää pelaajanimen lisäksi jotain muuta, ei silloin tarvitse koko riviä, vaan riittää pelkästään esimerkiksi seuraavanlainen komento: *[player.twitch]*. Tuolla komennolla saadaan ”player” listasta ”twitch” arvo tulostettua sivulle, jos tämä on kirjoitettu ylemmän rivin ”block” argumentissa annetun elementin sisällä, joka tässä tapauksessa on Fieldset. Lopputulos näyttää yhden pelaajan kohdalla kuvan 17 mukaiselta.

Kuva 17. Kuvankaappaus projektista

```
public function view()
{
    $TBS = Config::loadTBS();

    $players = \Player::all( array( "order"=>"LOWER(name) asc" ) );
    $nick_list = "";

    foreach($players as $p) {
        $nick_list .= strtolower($p->twitch) . ",";
    }

    $nick_list = rtrim($nick_list, ",");
    $TBS->MergeBlock("player", $this->playerArrayToTBS($players));
    $TBS->MergeField("nick_list", $nick_list);
    $TBS->Show();
}
```

Kuva 18. Kuvankaappaus projektista

Kuva 18 on kuva pelaajasivun PHP-ohjelmointipuolesta. Kuvasta nähdään että tämä on hyvin yksinkertaisesti rakennettu sivu, vaikka sivulla on paljon tietoa. Ensin alustetaan TBS $TBS nimisen muuttujaan, jotta voidaan käyttää sitä myöhemmässä vaiheessa. Seuraavaksi haetaan tietokannasta kaikki pelaajat ja pelaajien pelit $players

nimiseen muuttujaan. Tämä tuottaa ActiveRecords objektin joka sisältää myös pelitiedot. Tämän jälkeen luodaan pelaajien Twitch.tv käyttäjätunnuksista erillinen lista, joka upotetaan HTML-sivulle. Rtrim-komennolla saadaan poistettua ylimääräisen pilkun tekstipätkän, joka tässä tapauksessa on muuttuja $nick_list, perästä. Tämän jälkeen komennolla $TBS->MergeBlock lisätään komennon ”playerArrayToTBS” tuottaman listan sivulle. Komento ”playerArrayToTBS” luo erillisen listan jonka TBS tunnistaa. Ohjelmointivaiheessa tämä oli suuri ongelma, koska TBS ei käyttänyt ActiveRecordsista saatua objektia niin kuin olisi pitänyt käyttää, vaan antoi virheilmoituksen. Komento luo ActiveRecordsin objektia vastaavan listan, joka sisältää myös pelaajien pelaamat pelit, jotta tämä saadaan lisättyä HTML-tiedostoon ilman ongelmia. Komento käy jokaisen pelaajan läpi erikseen ja luo sen perusteella oikeanlaisen listan.
Seuraavana Lisätään HTML-sivuun komennolla $TBS->MergeField tekstirivin joka sisältää Twitch-käyttäjätunnukset pilkulla eroteltuna. Tämän jälkeen $TBS->Show TBS luo sivun ja näyttää lopullisen version käyttäjälle.

5.2.3 Aikataulu-sivu

Aikataulu sivu on projektin lähtökohta. Tätä varten on koko muu järjestelmä kehitetty. Aikataulu sivulla nähdään pelimaratoonin aikataulun, joka toimii käyttäjän omalla aikavyöhykkeellä. Tämä on yksinkertaisesti lista, jossa on päivämäärä, kellonaika, peli, pelaaja, pelin alusta sekä erinäisiä aikoja kestoista. Listan rivien vasemmalla puolella on valkoinen ympyrä, joka muuttaa väriä aikataulusta riippuen. Valkoinen väri indikoi että peliä ei ole vielä pelattu vaan on tulossa. Sininen väri indikoi sitä että peliä pelataan juuri nyt ja vihreä väri kertoo että peli on jo pelattu. Tämä on toteutettu Javascriptillä joka minuutin välein katsoo aikataulun rivistä kellonajan ja vertaa sitä nykyiseen kellonaikaan.

Suurin ongelma sivua tehdessä oli saada aikavyöhykkeet toimimaan oikein. Tätä vaikeutti eri selainten käyttämä Javascriptin DateTime-objekti, jonka myöhemmin korvasin moment.js kirjastolla. Sivun rivit ovat toteutettu samalla tavalla kuin pelaaja sivun pelaajarivit, tällä kertaa elementtinä toimii taulukon (table) sisällä oleva tr-elementti. Jokaiseen riviin on myös lisätty HTML5 mukana tuleva data arvo, tässä tapauksessa ”data-datetime”, johon annetaan rivin kellonaika ja päivämäärä muodossa ”

2015-08-08T16:00”, joka on selkokielelle käännetty 8.8.2015 kello 16:00. Tämä arvo annetaan Javascriptiä varten ja sitä varten että voidaan määrittää vasemmalla olevan pallon värin oikein. Lopputulos on kuvassa 19.

	Päivämäärä	Peli	Pelaajat
●	09.08.2015 15:00	Mirror's Edge	MrWalrus
●	09.08.2015 15:50	Darksiders II	Nikneim
○	09.08.2015 17:25	Amnesia	Edward

Kuva 19. Kuvankaappaus projektista

```
public function view()
{
    date_default_timezone_set('Europe/Helsinki');
    if(__USE_CACHE__) {
        $TBS = Config::loadTBS(__TEMPLATEDIR__ . "Cached_templates/" . __LANGUAGE__ . "_timetable_view.html");
    } else {
        $TBS = Config::loadTBS();
        $timetable = \Timetable::find('all', array('order' => 'rownumber asc'));
        $timetableArray = base::timetableParse($timetable);
        $TBS->MergeBlock("timetable",$timetableArray);
        $TBS->MergeField("timetable_first",$timetableArray[0]);
    }
    $TBS->Show();
}
```

Kuva 20. Kuvankaappaus projektista

Kuvassa 20 on kyseisen sivun tarvittava PHP-ohjelmointi. Kyseessä on jälleen yksinkertainen rakenne. Ensin asetetaan aikavyöhyke, joka tässä tapauksessa on ”Europe/Helsinki”, jolloin saadaan käyttöön Helsingin aikavyöhyke. Oletuksena PHP käyttää järjestelmän tai php.ini:ssä (palvelimella käytettävän PHP kääntäjän asetustiedosto) määriteltyä aikavyöhykettä, joka saattaa olla ongelma jos projekti siirretään palvelimelle joka sijainti ei ole suomessa.

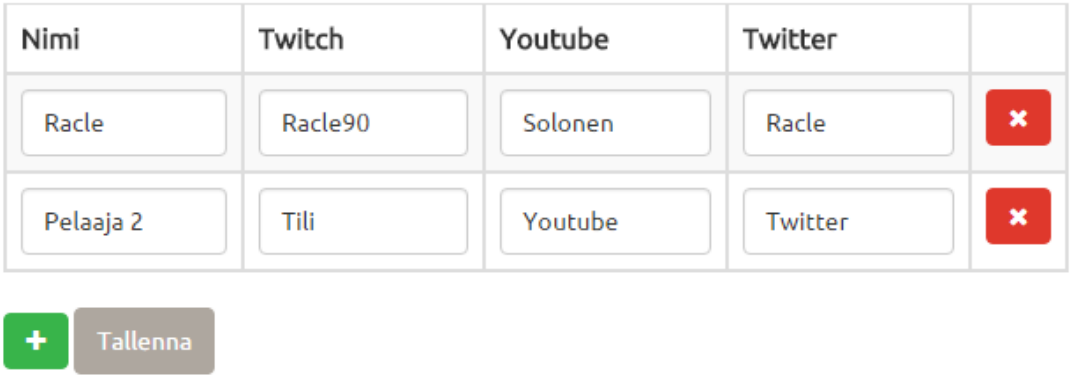
Seuraavana kuvassa 17 on kysely, jolla määritellään käytetäänkö ns. välimuistiversiota vai rakennetaanko sivu uudelleen. Muuttuja ”__USE_CACHE__” on määriteltynä asetukset luokassa. Välimuistiversiota käyttämällä saadaan sivuston latausajat minimaaliseksi, koska sivu on luotu jo kertaalleen, niin ei kantakyselyä tarvitse tehdä uudestaan. Sivun välimuistiversio luodaan muokkauksen tallennuksen yhteydessä. Tämä on hyvä tapa jos käyttäjä on useampi sata kerralla, niin saadaan käyttäjille sivusto toimimaan paljon nopeammin. Sivu löytyy kansiosta ”Cached_templates” ja se sisältää

omat versiot englanninkieliselle sivulle kuin myös suomenkieliselle sivulle. Tämä ladataan samalla tavalla kuin sivut normaalisti, ja näytetään normaalisti komennolla $TBS->Show. Tämä periaatteessa käyttää HTML-pohjana ennalta määriteltyä ja luotua HTML-tiedostoa, jolloin sivuun ei tarvitse enää lisätä mitään, vaan voidaan se näyttää suoraan. Jos kuitenkin käytetään ns. live versiota, eli nykyhetkistä, niin sivun luonti tapahtuu lähestulkoon saman lailla miten muidenkin sivujen luonti. Eli ensin alustetaan $TBS muuttujaan TBS. Tämän jälkeen haetaan $timetable muuttujaan tietokannasta kaikki aikataulun rivit. Tämän jälkeen luodaan erikseen $timeTableArray-muuttuja käyttämällä komentoa base::timetableParse. Tämä luo oman listan aikatauluista, jolloin saadaan yksilöityä tuloksia, kuten luotua riveille tarvittavat päivämäärä ja kellonaika tekstipätkät erikseen. Myös pelit ja pelien alustat kirjoitetaan erillisiksi, koska tietokannassa on vain linkitykset joita TBS ei osaa käyttää. Tämä on samanlainen ongelma kun pelaajalistaa luodessa. Listan luotua voidaan liittää se HTML-sivuun komennolla $TBS->MergeBlock. Komennolla $TBS->MergeField lisätään myös ensimmäinen tulos listasta, jolloin saadaan aloituskellonaika sivulle näkyviin. Viimeisenä tulostetaan sivu käyttämällä komentoa $TBS->Show, jota myös välimuistissa oleva versio käyttää.

5.2.4 Ylläpito-sivu

Ylläpito-sivu pitää sisällänsä seuraavat toiminnot: Oman käyttäjätunnuksen muokkaaminen, pääkäyttäjien muokkaaminen, aikataulujen muokkaaminen, sivujen muokkaaminen, pelaajien muokkaaminen, pelien muokkaaminen sekä pelialustojen muokkaaminen. Hallintapaneelissa käyttäjä voi muokata lähes kaikkea mitä sivuilla julkaistaan. Oman käyttäjätunnuksen muokkaussivulla voi vaihtaa käytettävän käyttäjätunnuksen ja salasanan. Muita tietoja ei sovellus tarvitse käyttäjältä. Salasanaa vaihtaessa tarvitsee käyttäjän syöttää nykyinen salasana ja uusi salasana kahteen kertaan. Pääkäyttäjien muokkaussivulla käyttäjä voi vaihtaa samoja tietoja. Tässä on estetty salasanan vaihto muille käyttäjille. Sivulla voi myös lisätä uuden käyttäjän sivuille tai poistaa nykyisen käyttäjän. Poisto nappi ei poista käyttäjää tietokannasta, vaan asettaa sen tilan ”ei käytössä”, jolloin käyttäjätunnuksella ei voi kirjautua sisään ylläpito portaaliin. Pääkäyttäjien muokkaussivu on graafisesti hyvin samannäköinen kuin mm. pelaajien muokkaussivu, joka on kuvattuna alapuolella.

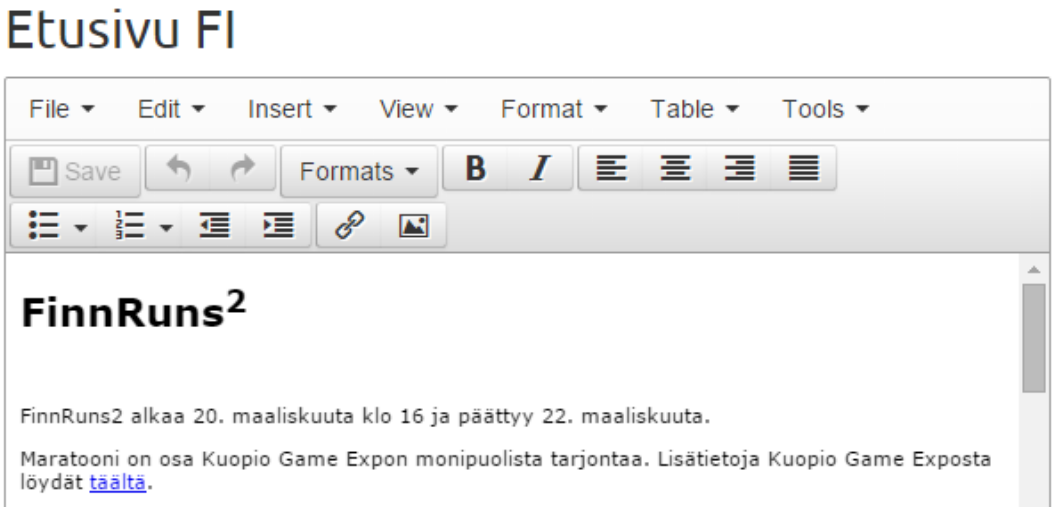
Pelaajien muokkaussivulla voit muokata pelaajien kaikkia tietoja kerralla. Sivulla on iso lista jossa on jokaisen pelaajan tiedot. Vihreästä plus napista voidaan lisätä uusi pelaaja, ja punaisesta x-napista voidaan poistaa jo olemassa oleva pelaaja. Sivun ulkoasu on esitettynä kuvassa 21.



Kuva 21. Kuvankaappaus projektista

Pelien muokkaussivu sekä alustojen muokkaussivu käyttävät samanlaista tyyliä ja logiikkaa muokkauksessa. Graafisesti ei muutu kuin näytettävien kenttien määrä ja nimet. Muuten sivut toimivat samalla tapaa kuin pelaajien muokkaussivu.

Sivujen muokkaussivulla voi muokata etu- ja tietoja-sivuja. Tämä tapahtuu samaan aikaan. Sivulla on graafiset käyttöliittymät (ns. wysiwyg, what you see is what you get eli mitä näet, sitä saat) joilla voidaan muokata etu- ja tietoja-sivuja. Sivulle on ohjelmoitu erikseen editorit englanninkieliselle ja suomenkieliselle sivulle. Alla kuva käytettävästä editorista nimeltään tinymce joka on esitettynä kuvassa 19. http://tinymce.com



Kuva 22. Kuvankaappaus projektista

Kuvassa 22 käytetään tekstiä FI ja EN ilmaisemaan onko muokattava osio suomenkielinen (FI) vai englanninkielinen (EN). Koska kaikki sivujen sisällönhallinta löytyy yhdeltä sivulta, ei tarvitse mennä useaan paikkaan muokkaamaan tekstejä, vaan ne voidaan muokata nopeasti samasta sijainnista.

Aikataulujen muokkaus on suurimpia ja tärkeimpiä kohteita joita pääkäyttäjä pääsee muokkaamaan. Se on koko projektin ydin. Rivejä voi myös siirrellä hiirellä vetämällä, jolloin saadaan järjestettyä pelit oikeaan järjestykseen jos aikatauluun tulee muutoksia alkuperäisen suunnitelman jälkeen. Riveille ei tallenneta tarkkaa aloituskellonaikaa ja päivämäärää vaan se määritellään erikseen. Tämän jälkeen ohjelma laskee jokaiselle maratonin kohteelle oman aloitusajan ottaen huomioon suunnitellun ajankäytön sekä edellisen rivin järjestely ajan. Tallentaessa sivua sivusto luo samalla välimuistiversion aikataulusta, jolloin saadaan nopeutettua sivunlatausta huomattavasti loppukäyttäjällä. Graafisesti ulkoasu on hyvin samantapainen pelaajien muokkauksen kanssa, mutta tähän on tarvittu nopeita työkaluja muokkaamaan sisältöä. Aikaa voi vaihtaa monelle riville samanaikaisesti minuuttikohtaisesti eteenpäin, jolloin voidaan lisätä jokaiseen riviin esimerkiksi järjestelyaikaa 5 minuuttia lisää. Kuvassa 23 voimme nähdä lopullisen graafisen ulkoasun.

Kuva 23. Kuvakaappaus projektista

Nopea ajanvaihto on toteutettu käyttämällä Javascriptiä. Kun on valittu halutaanko muokata suunniteltua vai järjestelyaikaa, valitaan halutaanko lisätä vai poistaa aikaa. Tämän jälkeen painetaan nappia ”Vaihda ajat”. Tämä ajaa Javascript komennon jokaiselle aikataulun riville ja lisää ja poistaa minuutteja sen mukaan mitä käyttäjä on toivonut. Tämä oli hyödyllinen työkalu järjestelykenttää tehdessä, koska alkuperäisen suunnitelman mukaan ei ollut suunniteltu järjestelyaikaa pelien välille, mutta työkalun avulla saadaan nopeasti laitettua esimerkiksi 5 minuuttia pelien vaihdon väliin, jona

aikana ehtii mm. vaihtamaan konsolin tai tietokoneen ja asettamaan uudelle pelaajalle mikrofonin.

6 YHTEENVETO

Julkaisujärjestelmän luominen ja suunnittelu on valtava operaatio, vaikka loppukäyttäjälle tämä on hyvinkin yksinkertaisen oloinen järjestelmä. PHP on ohjelmointikielenä helppo opetella, mutta vaatii huomattavasti ammattitaitoa jos sitä haluaa hyödyntää tehokkaasti. PHP:ta ohjelmoidessa on tärkeää osata käyttää internetin hakukoneita sekä PHP:n omaa, ja erittäin kattavaa, dokumentaatiota (http://php.net). PHP:n dokumentointi on ensiluokkaista, ja sitä parantaa käyttäjien lisäämät kommentit sivustolle, jotka välillä antavat suoran ohjelmointiratkaisun tuloksen.

Itse projekti oli edennyt hitaammin kuin oletin. Itse tapahtumaa siirrettiin yli puolella vuodella eteenpäin, joten en itse pitänyt kiirettä ohjelmoinnin kanssa. Projekti oli kuitenkin erittäin hyvä, koska pääsin keskustelemaan erilaisten ihmisten kanssa ja pohtimaan loppukäyttäjän kanssa toteutettuja ratkaisuja. Projektin eri vaiheissa on hyvä kysyä muilta mielipiteitä, koska he saattavat nähdä asioita mitä itse ei enää huomaa, koska on niin syvällä jo projektissa.
Projekti jatkaa eloaan vielä valmistumisen jälkeen, koska lisäohjelmointipyyntöjä on tullut ja uudet ominaisuudet ovat tekeillä.

Projektissa tuli sopivasti ongelmatilanteita vastaan ja niiden ratkaiseminen vaati ammattitaitoa. Minulle uusia ongelmia tuli mm. aikavyöhykkeiden kanssa. Olen mielestäni tehnyt sivuston ammattimaisesti päivätyön ja koulun ohella omalla ajallani. Pystyn käyttämään opinnäytetyössä oppimiani taitoja omassa ammatissani tulevaisuudessa. Jos tekisin projektin uudestaan, saattaisin käyttää hieman erilaista pohjaa ohjelmointiin, koska osa ohjelmistokirjastoista ovat jo jääneet ”orvoiksi”. Nykyisellään projekti kuitenkin toimii vielä pitkään koska se on tehty käyttämään tämänhetkisiä uusia palvelinympäristöjä.

Projektia ei näillä näkymin ohjelmoi tulevaisuudessa kuin allekirjoittanut, koska uusille ohjelmoijille ei ole tarvetta. Ohjelman lähdekoodia ei ole suunniteltu levitettäväksi laajemmalle yleisöllä näillä näkymin, joten se säilyy Finnruns-järjestön käytössä.

LÄHTEET

ActiveRecord. (2015). http://www.phpactiverecord.org/projects/main/wiki/Quick_Start

Arrington, M. (Tammikuu 2008). http://techcrunch.com/2008/01/16/sun-picks-up-mysql-for-1-billion-open-source-is-a-legitimate-business-model/

Bosomworth, D. (Tammikuu 2015). http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

CakePHP. (2015). http://api.cakephp.org/3.0/

Usability. (2015). http://guidelines.usability.gov/

Haikala, M. (2011). Ohjelmistotuotannon käytännöt. Talentum.

MariaDB. (2015). https://mariadb.com/kb/en/mariadb/mariadb-vs-mysql-features/

MUELLER, D. (Elokuu 2013). https://blog.openshift.com/why-mariadb-matters-the-openshift-interview-with-monty/

Ohjelmistoarkkitehtuurit. (2015). http://www.cs.tut.fi/~ohar/luennot.shtml

Parthian Systems. (2014). http://parthiansystems.com/indepth/

Pitts, M. (Toukokuu 2015). http://www.mayecreate.com/2013/12/5-reasons-php-great-programming-language/

Programming Language Popularity. (2015). http://langpop.com/

Wikipedia. (Heinäkuu 2015) https://en.wikipedia.org/wiki/Comparison_of_web_template_engines