Jukka Mäenpää

Ohjelmistokehykset verkkosovelluskehityksessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

11.2.2013

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jukka Mäenpää Ohjelmistokehykset verkkosovelluskehityksessä 32 sivua + 1 liite 21.1.2013
Tutkinto	insinööri (AMK)
Koulutusohjelma	mediatekniikka
Suuntautumisvaihtoehto	digitaalinen media
Ohjaajat	lehtori Ilkka Kylmäniemi

Insinöörityön lähtökohtana oli parantaa olemassa olevaa verkkosivustoa, joka oli rakennettu huonoille ohjelmointikäytännöille. Työllä ei ollut tilaajaa, vaan se perustui pienempään projektiin, joka oli sopivan kokoinen ja sopivassa tilassa paranneltavaksi. Tavoitteeksi otettiin käyttää eri verkkokehitykseen tarkoitettuja ohjelmistokehyksiä sivuston uudelleenrakennuksessa. Sivuston koodissa oli paljon turvallisuusaukkoja, toiminnot oli jaettu eri tiedostoihin epäskaalautuvasti ja toistoa oli paljon. Ohjelmistokehykset tekivät koodista turvallisempaa ja jaottelivat logiikan järkevästi. Käytetyt kehykset ja kirjastot tarjosivat rajapinnan, joka helpotti ohjelmointia ja vähensi inhimillisten virheiden määrää.

PHP:n ohjelmistokehys Symfony 2 järjesti koodin tarkasti eriteltyihin paketteihin, ja ohjelmoinnilla oli tarkat työn vaiheet. Symfonyn mukana tuli useita valmiita toimintoja, kuten pyynnön reititys halutun PHP-luokan ohjelmistologiikkaan ja käyttäjän autentikointi ja sen tiedon ylläpito. Tietokannan kanssa keskusteluun käytettiin Symfonyn mukana tulevaa Doctrine-kirjastoa, joka tarjoaa tietokannan abstraktointikerroksen ja Object Relational Mapper -tuen. ORM-tuella tietokannan rivejä voi käsitellä kuten olioita ohjelmointilogiikassa. Doctrine auttoi tekemään tietokantakyselyistä turvallisempia ja vähensi virheitä, kun kyselyitä ei tarvinnut kirjoittaa itse.

Sivun asettelun puolella Twitterin Bootstrap-ohjelmistokehys tarjosi hyvän kokoelman sivun asettelua, sommittelua ja toiminnallisuutta parantavia ja tehostavia osia. Kokoelma hyödynsi Jquery-Javascript-kirjastoa. Käyttämällä Bootstrapin tarjoamaa pohjaa sivustolle sai helposti responsiivisen, hyvännäköisen valmiin sivurakenteen. Jquery helpotti Javascript-koodin kirjoittamista ja tarjosi rajapinnan tehdä muun muassa DOM-manipulaatioon, animaatioon ja asynkronisiin pyyntöihin liittyviä toimintoja käyttämällä helppokäyttöisiä metodeja, jotka toimivat samalla koodilla kaikissa selaimissa.

Muutosten jälkeen sivuston tiedostorakenne on ohjelmistokehysten standardin mukainen, rakenne on skaalautuvampi ja koodi perustuu ammattilaisten ylläpitämiin, testattuihin ja dokumentoituihin ohjelmistokehyksiin. Sivusto on vakaammalla pohjalla, ja sitä on helpompi ylläpitää ja kehittää projektina.

Avainsanat	PHP, ohjelmistokehys, CSS, Javascript, Symfony, tietokanta

Author Title Number of Pages Date	Jukka Mäenpää Web application development using frameworks 32 pages + 1 appendice 21 January 2013
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructors	Principal Lecturer Ilkka Kylmäniemi

The existing website was built on poorly chosen programming practices. The code had several security issues and the functionality was distributed accross files in a non-scalable way and there was much repetition. I decided to use frameworks to build the new version of the website. Frameworks made the code more secure and arranged the logic better. The frameworks and libraries I used offered an interface that facilitated programming and decreased the amount of human errors

The Symfony2 PHP framework organized the code into separated bundles and there was a special work flow to programming. Symfony came with a lot of built in functionality such as routing the request to a specific PHP class and taking care of user authentication. Doctrine was used for interaction with the database. Doctrine came bundled with Symfony.

Doctrine offers a database abstraction and Object Relational Mapper support. With ORM support it was possible to handle rows from the database tables as objects in the programming logic. Doctrine helped make database queries safer and lessened mistakes made when writing the queries by hand.

On the front end the use of Twitter's Bootstrap framwork brought a collection of components for improving the layout and functionality of the page. The collection made use of the Jquery Javascript library. By using Bootstrap's ready page layouts, it was easy to build a responsive and good-looking user interface.

Jquery made writing Javascript simpler by offering an API with methods for easier cross-browser functionality, including DOM-manipulation, animation and asynchronous requests.

Keywords	PHP, framework, CSS, Javascript, Symfony, database

Sisällys

Lyhenteet

1 Johdanto 1

2 PHP-ohjelmistokehykset 2

2.1 MVC-malli PHP-ohjelmistokehyksissä 2

2.2 Symfony2-ohjelmointikehys 3

3 Javascript-kirjastot 10

4 CSS-ohjelmistokehykset 12

4.1 LESS: dynaaminen tyylimääritelmäkieli 12

4.2 Ruudukkojärjestelmät 14

4.3 Joustava ruudukko 16

4.4 Twitter Bootstrap -kehys 16

4.5 Joustava ruudukko ja reagoiva sivu 17

5 Muutokset sivuston uudessa versiossa 19

5.1 Aiemman version puutteita ja huonoja puolia 19

5.2 Tietoturvan parantaminen Symfonyn avulla 22

5.3 Jquery uudessa versiossa 24

5.4 Mediaquery-sääntöjen käyttö 24

6 Yhteenveto 28

Lähteet 31

Liitteet

Liite 1. Ote TeamController.php-tiedostosta

Lyhenteet

HTML	HyperText Markup Language. Verkkosivujen merkintäkieli.
CSS	Cascading Style Sheets. Tyylinmäärityskieli verkkodokumenteille.
MVC	Model-View-Controller. Ohjelmistoarkkitehtuuri, jossa tiedon tallennus, käyttöliittymä ja näkymät sekä sovelluslogiikka on eroteltu toisistaan.
PHP	Palvelinpuolen verkko-ohjelmointikieli.
YAML	YAML Ain't Markup Language. Ihmisen luettava, määrittelydokumenttien helppoon kirjoittamiseen tarkoitettu kieli.
XML	Extensible Markup Language. Merkintäkieli, jolla voi määritellä sääntöjä ja dokumentin elementtien hierarkiaa useisiin eri tarkoituksiin.
DBAL	Database Abstraction Layer. Rajapinta, joka erottaa tietokantaa käsittelevän koodin riippumattomaksi tietokannan tyypistä.
ORM	Object-Relational Mapping. Tekniikka, jossa tietokannan tieto otetaan koodissa käsittelyyn oliomuodossa.
LESS	Dynaaminen tyylimääritelmäkieli, joka yhdistää CSS:n ja Javascriptin ominaisuuksia.
Ajax	Asynchronous Javascript and XML. Teknologia, joka mahdollistaa tiedonsiirron ladatun verkkosivun ja verkkopalvelimen välillä ilman sivun uudelleenlataamista välissä.
NBA	National Basketball Association. Koripallon ammattilaisliiga Pohjois-Amerikassa.

1 Johdanto

Insinöörityössä on tarkoitus käyttää useita eri ohjelmistokehyksiä verkkosivuston parantamiseen. Koodin puolesta huonolla pohjalla olevaa verkkosivustoa on tarkoitus parantaa käyttämällä verkkosovelluskehitykseen tarkoitettuja ohjelmistokehyksiä ja kirjastoja ja dokumentoida muutokset, joita ohjelmistokehykset tuottavat sivuston koodiin ja rakenteeseen.

Verkkosivusto, joka rakennetaan, on nimeltään NBA Fantasialiiga. Kyseessä on sivusto, jolla käyttäjät voivat seurata NBA-koripalloliigan pelaajista kokoamiensa ”fantasiajoukkueiden” tilaa sarjataulukossa sekä hallita pelaajia. Sivustosta on olemassa vanha versio, jossa on paljon rakenteellisia, ohjelmallisia ja tietoturvallisia puutteita.

Verkkosivuja on luotu ja rakennettu jo 1990-luvulta lähtien, jolloin Tim Berners-Lee loi spesifikaatiot HTML-kielelle sekä selain- että palvelinohjelmistoille. Ei siis ole ihme, että nykypäivänä on jo muodostuneita normeja ja parhaita käytäntöjä, joita verkkosovelluksen kehittämisessä on hyvä seurata. Insinöörityössä käytetään yhtä tämän hetken suosituimmista verkko-ohjelmointikielistä, PHP:ta, kaikkien nykyaikaisten selainten tukemaa JavaScript-kieltä, sekä HTML- ja CSS-verkkotekniikoita. Käyttäjän hakiessa verkkosivua tietystä osoitteesta hänen selaimensa lähettää pyynnön verkkopalvelimelle, joka vastaa tähän pyyntöön tarjoamalla halutun sivun. PHP on palvelinpuolen ohjelmointikieli, jolla kirjoitettuja ohjelmia suoritetaan palvelimella ennen sivun lähettämistä takaisin käyttäjälle. JavaScript-komennot sitä vastoin suoritetaan käyttäjän selaimessa, hyödyntäen käyttäjän oman koneen tehoa. HTML-merkinnät ja CSS-säännöt määrittelevät, miltä verkkosivu näyttää käyttäjän selaimessa.

NBA Fantasialiigan aiemman version kehitys lähti siitä, kun minulta kysyttiin, osaanko tehdä sivun, jonka näkemiseksi tarvitsee salasanan. Totesin osaavani ja tein pienen sivun, joka tarkisti, oliko käyttäjän istuntoon tallennettu tieto, että hän on kirjautunut. Jos tieto löytyi, sivun sisältö paljastettiin, jos tieto puuttui, käyttäjä ohjattiin sisäänkirjautumissivulle. Vähitellen lisäpyyntöjä toiminnallisuudesta kertyi, ja lopulta koossa oli jonkinlainen sivusto, joka vaati käyttäjää kirjautumaan sisään ja antoi kirjautumisen jälkeen mahdollisuuden tarkastella liigan tapahtumia ja omaa joukkuettaan.

Liigan sivusto muodostui lopulta tiedostojen ja ohjelmapätkien sekamelskasta. Ylläpito oli vaikeaa, eikä sivuston suunnittelu tukenut helppoa laajennusta. Rakenne oli toimiva, mutta holistisesti kehno. Nyt sivustoa uusittaessa lähtökohtana on käyttää nykypäivän standardeja vastaavaa teknologiaa ja parempia ohjelmointikäytäntöjä. Lopputuloksen tulee olla skaalautuva, turvallinen ja ohjelmointikehyksiä hyödyntävä. Uutta versiota luotaessa käyttöön otetaan verkkokehityksen ohjelmistokehyksiä, jotka helpottavat kehitysprosessia tekemällä siitä muodollisempaa sekä auttavat tuottamaan kunnollista ja hyvin rakennettua koodia. Käytetyt ohjelmistokehykset ja kirjastot ovat Symfony2 PHP:n kehitykseen, Twitter Bootsrap sivuston rakenteen ja tyylien esittämiseen, Jquery JavaScriptin tuottamiseen ja Doctrine työskentelyyn tietokannan kanssa.

Ohjelmistokehyksen käyttö perustuu siihen, että se tarjoaa helpon tavan toteuttaa usein käytettyjä toimintoja, kuten sisäänkirjautumista tai tietokannan käsittelyä. Verkkokehityksessä ohjelmistokehykset tekevät koodin tuottamisesta virtaviivaista luomalla pohjan sivuston toiminnallisuudelle ja koodin yhtenäistämiselle. Kehyksen luoma kehitysympäristö tarjoaa hyvin skaalautuvan alustan pienten ja suurten verkkosovellusten tekoon ja auttaa vähentämään virheiden määrää.

Ohjelmistokehys tarjoaa siis valmiita työkaluja ja komponentteja sovelluskehitykseen – jolloin koodia ei tarvitse itse kirjoittaa yhtä paljon – ja tietynlaisen lähestymistavan sovelluksen kehittämiseen ja rakenteeseen. Kun jokaista pientä säätöä ei tarvitse erikseen ohjelmoida käsin, koodin kirjoittaminen ohjelmistokehyksen päälle, sen tarjoamia rajapintoja käyttäen, tuntuu oikeastaan miellyttävältä. [1.]

2 PHP-ohjelmistokehykset

2.1 MVC-malli PHP-ohjelmistokehyksissä

PHP-ohjelmointikielen ohjelmistokehykset ovat usein ratkaisuja, jotka perustuvat ”Front Controller”- tai ”Model-View-Controller”-malleihin. Front Controller -mallissa kaikki pyynnöt menevät yhden kontrolleriolion tai tiedoston kautta. Tämä kontrolleri jakaa pyynnön perusteella tehtävän eteenpäin eri luokkien olioille tai ohjelmakoodeille. Front Controller -periaatteella toimiviksi PHP-ohjelmistokehyksiksi voi lukea esimerkiksi Fat Free Framework- ja Silex-ohjelmistokehykset. Molemmat ohjelmistokehykset vastaan-

ottavat pyynnön päätiedostossa ja vertaavat sitä erilaisiin määriteltyihin polkuihin. Kun osuman löytyy, suoritetaan polulle määritelty koodi. [2.]

PHP:lle on tarjolla myös useita eri Model-View-Controller- eli MVC-mallin ohjelmistokehyksiä, jotka laajentavat Front Controller -mallin toimintaa. MVC-mallissa tärkein periaate on erotella eri jakoihin käsiteltävä tieto (model), näkymä, jossa tieto esitetään (view), ja ohjelmointilogiikka (controller). Kontrolleri vastaanottaa pyynnön, poimii ja käsittelee tarvittavasta tietomallista datan ja kutsuu näkymää ja antaa sille muuttujia käyttöön syötteen muodostamista varten. [3; 4.]

MVC tarjoaa voimakasta skaalautuvuutta ja mukautuvuutta. Lähes kaikki verkkosovellukset, jotka tallentavat dataa, käyttävät tietokantaa apuna. Tietokannassa on helppo tallentaa arvoja järjestelmällisesti ja noutaa niitä helposti. Tietomallit antavat rajapinnan noutaa ja käsitellä sovelluksen käyttämää dataa. Tietomalleja on helppo käyttää uudelleen eri kontrollereissa, ja tietomallin lähteen muuttuessa esimerkiksi tietystä tietokantajärjestelmästä toiseen mallia voi muuttaa muokkaamatta kontrollerin koodia ollenkaan. [5.]

Näkymien erottamisen etu on taas siinä, että ohjelmointilogiikka on eroteltu ulkoasun suunnittelusta, ohjelmoija välttää PHP-koodin ja HTML-merkinnän sekamelskan. Jopa PHP:ta hyvin vähän osaava suunnittelija voi muokata näkymiä tuntematta kontrollerin koodia.

2.2 Symfony2-ohjelmointikehys

Symfony2 on Symfony-ohjelmistokehyksen uusin versio, joka ottaa kaiken tehon irti PHP:n versiosta 5.3. Symfony-ohjelmistokehyksessä pyynnöt otetaan vastaan ja katsotaan reititysmäärityksistä, mitä kontrolleria käytetään ja mikä sen toiminto suoritetaan.

Valitsin Symfonyn projektin toteuttamiseen luettuani Internetistä useiden käyttäjien kehuvia kommentteja positiivisesta kokemuksesta. En ollut aiemmin tutustunut Symfonyn toimintaan, ja se eroaa muista suosituista PHP-ohjelmistokehyksistä. Selatessani työpaikkailmoituksia verkossa totesin, että useassa ilmoituksessa toivottiin hakijalta osaamista Symfonyn kaltaisesta ohjelmistokehyksestä, mikä viestii sen suosiosta ja sen osaamisen tarpeellisuudesta työmarkkinoilla.

Symfony2:n asentaminen

Symfony2 vaatii verkkopalvelimelta PHP:n version 5.3.2 tai uudemman. Näissä versioissa on mahdollista käyttää nimiavaruuksia, joita Symfony hyödyntää. Symfonyn asennuksen ensimmäinen vaihe on ladata sen lähdekoodi kehittäjän sivustolta ja purkaa pakatusta tiedostosta Symfonyn hakemistorakenne ja tiedostot verkkopalvelimelle. Kun tiedostot on purettu, suoritetaan Symfonyn ”web”-hakemiston config.php-tiedosto. Tämä tiedosto näyttää sivun, joka kertoo, onko palvelimen tai PHP:n asetuksissa mitään virheitä. Oman asennukseni aikana minun tuli määritellä ”php.ini”-nimisessä PHP:n asetustiedostossa aikavyöhyke ja muuttaa Symfonyn app/cache- ja app/logs-hakemistoihin kirjoitusoikeudet verkkopalvelimelle ja komentoriville. [6.]

Symfonyn omilla sivuilla tämä tyypillinen virhe oli dokumentoitu ja siihen tarjottiin ratkaisut ja niiden ohjeet. Ensin oli ladattava Ubuntu Linux -distribuutiolle paketti, jolla sain tuen käyttöoikeuslistojen laatimiseen Ubuntussa. Seuraavaksi annoin verkkopalvelimen käyttäjäryhmälle oikeudet kirjoittaa app/cache- ja app/logs-hakemistoihin. Kun olin korjannut kaikki löydetyt ongelmat, siirryin Symfonyn asennuksessa verkkokäyttöliittymän kautta eteenpäin. [7.]

Uudelle projektille tuli luoda oma paketti. Symfony tarjoaa työkalut uuden paketin luomiseen komentoriviltä. Komento näytti seuraavalta:

```
php app/console generate:bundle --namespace=Oyvey/NbaBundle --
format=yml
```

Komennon namespace-kohdassa määritin uuden paketin nimiavaruuden. Nimi koostuu osista Oyvey ja NbaBundle. Symfonyn nimikäytännön mukaan Oyvey viittaa yrityksen tai muun vastaavan tahon nimeen. Valitsin nimen Oyvey, koska sivusto tuli sijaitsemaan www.oyvey.fi-palvelimella. NbaBundle vastaa nimeä, jolla paketti tunnistetaan. Samaan projektiin kuuluvan koodin tulisi olla samassa paketissa. NbaBundle oli lyhyt nimi, joka vastasi hyvin projektin aihetta.

Format-kohta määrittelee, millä tavalla projektin asetukset, kuten reititys, autentikointi ja ohjelmistokehyksen asetukset, määritellään. YML tarkoittaa YAML-kieltä, jolla luodaan dokumentteja, joissa arvoja määritellään yksinkertaisesti tekstin asetteluun perus-

tuvalla tavalla. Symfony antaa myös vaihtoehdot käyttää XML-merkintää tai PHP-koodiin upotettuja annotaatioita eli kommenteissa olevia koodeja, jotka määrittelevät, miten sovelluksen eri osien pitäisi toimia. [6.]

Komentorivillä pyydetään määrittämään vielä yleinen nimi, jolla viitata projektiin – valitsin OyveyNbaBundle – ja kysytään, halutaanko paketti rekisteröidä automaattisesti Symfonyn käyttöön. Valitsemalla kyllä automatisoinnille säästin vaivan tehdä sen käsin ja varmistin, että en ainakaan itse riko mitään muuttamalla Symfonyn asetustiedostoja.

Kun Symfony2-asennus oli valmis, Symfonyn omista dokumentaatioista löytyi hyvin tietoa ja ohjeita, joilla päästä alkuun verkkosovelluksen kehittämisessä ohjelmistokehystä käyttäen. Symfonyn tarjoama ohjattu asennus verkkokäyttöliittymän avulla sekä eri tehtävien automatisointi konsolikomennoilla helpottivat suuresti Symfonyn asennuksessa ja käytössä.

PHP-tiedostojen sekamelskasta Symfony2-ohjelmistokehyksen järjestykseen

Tehtävänä oli siirtää useaan eri tiedostoon sekalaisesti hajautettu toiminnallisuus toimimaan Symfonylla. Symfonyn MVC-rakenne mahdollistaa paljon selkeämmän jaottelun tiedostoissa ja niiden toiminnallisuudessa. Kaikki laskenta ja tiedonkäsittely tehdään kontrollereissa, jotka löytyvät OyveyNbaBundle-paketin Controllers-hakemistosta.

Kun kontrollerit ovat käsitelleet tarvittavan datan, ne antavat tarvittavat tiedot eteenpäin näkymille eli Twig-tiedostoille, jotka ovat sivun HTML-lähdekoodin sisältäviä tiedostoja, joissa on käytössä kontrollerilta saadut muuttujat. Twig mahdollistaa tiettyjen toiminnallisuuksien, kuten koodisilmukoiden, konditionaalilausekkeiden ja muuttujien, tulostamisen helpolla tavalla. Tarkoitus on erotella ohjelmointi näkymän suunnittelusta ja antaa kuitenkin hyödyllisiä toimintoja käyttöön ulkoasua luotaessa.

Mietin, mitkä toiminnot haluan tuoda uuteen sivustoon, mitkä haluan jättää pois ja mitä uutta tahdon luoda. Yhdellä kontrollerilla voi olla useita toimintoja, joita kutsutaan eri näkymiä varten. Jaottelin toiminnot eri ryhmiin ja lisäsin ne vastaaviin kontrollereihin osiksi. Esimerkiksi Team-kontrolleriin lisäsin toiminnot liittyen joukkueisiin: sarjataulukkolistaus, joukkueen tarkastelu, oman joukkueen tarkastelu ja muuttaminen sekä pelaajakohtaiset toiminnot.

Useissa muissa PHP-ohjelmistokehyksissä, kuten CodeIgniterissa ja Zend Frameworkissa, verkko-osoitteet muodostuvat tiedostorakenteen mukaan. Osoite saattaa osoittaa ja vihjata suoraan käytetyn kontrollerin nimeen ja pyydettyyn kontrollerin metodiin. Symfony2:ssa käytetään verkko-osoitteiden kartoitukseen reititysmääritelmiä. Jokainen verkko-osoite tulee määrittää erikseen ja sen toiminta ohjata jollekin kontrollerin metodille. Dynaamisia reittejä voi myös määritellä. Verkko-osoitteen tietty osa voidaan määrittää tulkittavaksi muuttujana, joka lähetetään kontrollerille, ja haluttu metodi vastaanottaa sen hyödyntäen sitä määritellessään dataa näkymälle. Reittimääritys, joka on muotoa ”/joukkue/{id}”, vastaa kaikkiin reitteihin, joissa {id}-osan tilalla on jokin arvo, ja käyttää sitä arvoa parametrina. Reitityksmäärityksiä voi myös rajata vastaanottamaan vain tietyntyyppisiä (esimerkiksi GET tai POST) pyyntöjä tai vastaanottamaan vain tietynlaisia parametreja – kuten vain numeroita – muuttujina. [6; 8; 9.]

Aluksi vaikutti oudolta, että kaikki reitit täytyy määritellä erikseen, ja loppua kohden reittimääritelmiä olikin kerääntynyt melkoisesti. Kuitenkin reittien määrittäminen erikseen auttaa tilanteissa, joissa verkko-osoitetta täytyy muuttaa, sillä se ei vaadi koko kontrollerin nimeämistä uudelleen, ainoastaan reitin polun määritelmän muuttamista.

Koodin selaaminen on uudessa ohjelmistokehyksen tarjoamassa ympäristössä helpompaa ja selkeämpää. Vanhassa järjestelmässä kaikki tiedostot olivat yhdessä kansiossa ja nimetty ilman minkäänlaista yhteistä käytäntöä. Usein joutui avaamaan useamman tiedoston, kunnes löysi oikean toiminnallisuuden. Koodia on Symfonyssa helpompi skaalata, käyttää uudelleen ja muuttaa sen toiminnallisuutta rikkomatta mitään.

Tietokannan abstraktointi ja Doctrinen ORM-ominaisuuden käyttö

Lähes kaikissa nykyaikaisissa verkkosovelluksissa tiedon ja arvojen tallennukseen käytetään tietokantaa. Tietokannan käytöllä on etuja esimerkiksi evästeisiin tai käyttäjän paikalliseen välimuistiin tallentamisessa. Evästeet asetetaan usein erääntymään tietyn ajanjakson jälkeen, tai käyttäjä voi manuaalisesti poistaa niitä. Vieraat ohjelmat pääsevät myös helposti urkkimaan, mitä tietoja evästeissä on, kun ne lähetetään HTTP-pyynnön otsaketiedoissa. Paikallinen välimuisti on HTML-kielen viidennen spesifikaatioversion määrittämä tallennustila, johon pääsee JavaScript-rajapinnan kautta. Paikallisen välimuistin ja evästeiden käytön estoksi voi helposti asettua selainasetus, joka määrittää JavaScriptin käytön tai evästeiden hyväksynnän pois käytöstä. [10.]

Tietokannat välttävät paikallisesti tallennettavan tiedon ongelmat toimimalla palvelinpuolella. Tietokantaan saa tallennettua suuria määriä tietoa organisoidusti taulukkoihin tai dokumentteihin, ja SQL-kieltä tukevista tietokannoista on helppo hakea eri sarakkeiden tai rivien arvoja. Koska tietokantoihin lähes aina tallennetaan käyttäjien antamaa tietoa, on sovellukset turvattava SQL-injektioita vastaan. SQL-injektiot ovat hyökkäyksiä, joissa vääränlaista dataa yritetään ujuttaa SQL-kyselyyn. [11.]

Yksi syy SQL-injektioiden onnistumiselle tai muunlaisille tietokantavirheille sovelluksessa ovat huonosti suunnitellut kyselyt. Vanhassa versiossa tietokantakyselyn lausekkeet oli kirjoitettu käsin suoraan koodiin. Lausekkeet ovat täysin koodaajan muistin varassa turvallisuuden suhteen. Onko jokainen lausekkeeseen syötetty arvo muistettu suodattaa vääränlaisista merkeistä ja onko kyselyn parametrit sekä komennot varmasti kirjoitettu oikeaan järjestykseen? Tietoturvan kannalta on paljon turvallisempaa, että kehittäjä käyttää erillistä rajapintaa lausekkeille ja antaa kyselyjen muodostua ja suorittaa koneellisesti. [11; 12; 13.]

Sivuston uudessa versiossa käyttöön otettiin Symfonyn kanssa toimiva Doctrine. Se on PHP-kirjasto, jonka tarkoituksena on tarjota rajapintoja joilla käsitellä tietokantoja epäsuorasti ja turvallisemmin. Sivuston kyselyissä käytettiin Doctrinen ORM-rajapintaa (Object Relational Mapper), eli tietokannan taulukot kartoitettiin olioiksi käytettäväksi PHP:ssä. ORM:n avulla ei tarvitse ajatella SQL-kieltä. Taulukosta voi hakea esimerkiksi joukkueen tiedot yhdeltä tietokannan riviltä ja manipuloida niitä oliona kuten koodiesimerkissä 1. [6; 14.]

```
// haetaan pelaaja tietokannasta
$pid = $request->request->getInt('pid');
$pelaaja = $em->getRepository('OyveyNbaBundle:Pelaaja')
              ->find($pid);
// haetaan kohdejoukkue tietokannasta
$jid = $request->request->getInt('team');
$joukkue = $em->getRepository('OyveyNbaBundle:Joukkue')
              ->find($jid);

// varmistetaan, että pelaajaa muokkaava joukkue
// omistaa pelaajan
if($pelaaja->getTeam()->getJid() == $this
    ->get('security.context')->getToken()->getUser()
    ->getJid()) {

    // muutetaan pelaajan joukkue
    $pelaaja->setTeam($joukkue);
    $em->persist($pelaaja);
    $em->flush();
```

Koodiesimerkki 1. Entiteettihallintaoliolla haetaan taulukoiden rivien arvoja ja käsitellään niitä kuten olioita [liite 1].

Pelaajan tiedot saa käsiteltäviksi, kun ensin hakee Symfonyn entiteettihallintaolion ja lataa sillä ”Pelaaja”-entiteetin. Entiteetit ovat käytettyjä tietokantataulukoita vastaavia olioita, joilla on metodit määrittää ja hakea olion (eli yhden tietokannan rivin) eri arvot. Entiteetit määritellään myös Doctrinea varten YAML-dokumenteissa, joissa määritellään, minkälaisia arvoja tietokannan kentät voivat vastaanottaa.

YAML-dokumenteissa voi myös määritellä tietokantataulukoiden välisiä yhteyksiä. Määritin Pelaaja-taulukon ja Joukkue-taulukon välille yhteyden, jossa pelaajan joukkueen määrittävä kenttä on kartoitettu joukkueiden taulukon id-tunnisteisiin. Kun määritys oli tehty, saatoin antaa pelaajaolion setTeam-funktiolle joukkueolion parametrinä, jolloin Doctrine osasi automaattisesti poimia joukkueoliosta oikean kentän arvon talletettavaksi pelaajaolioon. [6.]

Kun taulukkoa vastaavaan entiteettiolioon on asetettu arvot, sen voi tallentaa tietokantaan. Jos olio luotiin uutena instanssina, Doctrine osasi lisätä taulukkoon uuden rivin. Jos olio oli haettu tietokannasta ja sen tietoja vain muutettiin, tiesi Doctrine päivittää jo olemassa olevan rivin tiedot. Useamman tapahtuman voi tallentaa kerralla resurssien säästämiseksi. Kutsumalla entiteettihallintaolion persist-komentoa tietokantaolio tallennetaan käsiteltäväksi, mutta sitä ei vielä tallenneta tietokantaan. Kun kutsutaan entiteettihallintaolion flush-komentoa, Doctrine luo käsiteltävänä olevista olioista mahdollisimman suorituskykyiset kyselyt. [6.]

Doctrinen kykyä luoda tehokkaasti kyselyitä voisi hyödyntää esimerkiksi uuden joukkueen populoimista (jäsenten valintaa) sovellettaessa. Jos uudelle joukkueelle luotaisiin automaattisesti vaikka 10 pelaajaa ja täytettäisiin ne pelaajatietojen perusarvoilla, jokainen Pelaaja-olio voitaisiin tallentaa käsiteltäväksi persist-komennolla ja lopulta tallentaa kaikki kerralla flush-komennolla. Tällöin Doctrine loisi vain yhden INSERT-lausekkeen, jolla syötettäisiin kaikkien kymmenen pelaajan arvot. [6; 14.]

Twig ja sivupohjien käyttö

Symfony2:n mukana tulee ohjelma nimeltä Twig. Sillä voi luoda dokumenttipohjia tai -runkoja, joiden tarkoitus on tehdä ulkoasun muokkaamisesta helppoa ja vaivatonta. Twigia käytettäessä kehittäjä tai suunnittelija ei joudu käyttämään yhtään PHP:ta, vaan voi sen sijaan hyödyntää Twigin syntaksia, jolla tietyt peruslausekkeet onnistuvat ja muuttujia on helppo upottaa koodin väliin. [6; 15.]

Twigin komenot ovat monipuolisia. Eri sivupohjia voi liittää toisten sisään, ja sivupohjalle voi antaa mahdollisuuden muokata vain tiettyjä kohtia. Sivupohjalla voi olla staattista HTML-merkintää ja väliin määritelty nimen mukaan kappale, jonka sisältöä voi muokata pidemmälle tai korvata myöhemmillä sivuilla, jotka hyödyntävät kyseistä sivupohjaa.

Twigin syntaksilla voi myös ohjelmoida silmukoita, jotka luuppaavat usean taulukossa olevan tiedonpalan läpi. Tämä helpottaa muun muassa tietokannasta näytettävän tiedon esitystä sivulla, kun tietoa tulee usean tietokantarivin verran. [14.]

Twig tarjoaa myös perinteisen konditionaalilausekkeen ohjelmoinnin sivupohjalle: jos muuttujalla on arvo, näytä se, muutoin näytä virhe. Tätä ominaisuutta voi esimerkiksi käyttää esittämään eri sisältöä sivun osissa käyttäjän ollessa kirjautunut tai nimetön käyttäjä. Nämä ominaisuudet ja yksinkertainen syntaksi tekevät Twigistä tehokkaan keinon datan esittämiselle sivulla. Koska Twigin kieli ei vaadi PHP-osaamista, sen käyttäminen on helpompaa myös ohjelmointia osaamattomalle henkilölle. Twigin komentoja käyttäen PHP:n voi jättää pois sivupohjatiedostoista, mikä tekee Twigistä hyvän kehyksen sivusuunnittelussa. Se tuo ohjelmoinnin tehoa HTML-merkinnän keskelle.

3 Javascript-kirjastot

Javascriptia käsiteltäessä ei puhuta niinkään ohjelmistokehyksistä vaan kirjastoista. Kirjastot tarjoamine etuineen vastaavat kuitenkin ohjelmistokehyksiä. Koodin kirjoittaminen helpottuu ja kirjasto tarjoaa useita hyödyllisiä työkaluja toimintojen toteuttamiseen.

Usein JavaScriptiä kirjoittaessa voi huomata keksivänsä samoja ratkaisuja samoihin toistuviin ongelmiin jokaisessa projektissa. Pienten toimintojen koodaaminen voi olla kovan työn takana, mikäli haluaa varmistaa koodin olevan helposti siirrettävissä eri yhteyksiin ja toimivan jokaisessa selaimessa.

JavaScriptin ohjelmointiin on olemassa useita hyödyllisiä ja suosittuja kirjastoja. Muun muassa Mootools, YUI ja Jquery ovat kirjastoja, joilla on laajat kokoelmat toiminnallisuuksia ja hyvät dokumentaatiot niiden rajapinnoista ja käytöstä. Nämä kirjastot tarjoavat funktioita ja metodeja suorittaa erilaisia Javascript-komentoja ja HTML-dokumentin manipulointia. Toimintoihin kuuluvat muun muassa Ajax-pyyntöjen teko, HTML-elementtien löytäminen valitsijoilla, elementtien tyylien ja attribuuttien muuttaminen, tehosteiden luominen ja toiminnallisuuden sitominen tapahtumaan. [16.]

Kirjaston käyttö helpottaa suuresti vaikeidenkin Javascript-toimintojen ohjelmointia. Toiminnot, joiden ohjelmointiin itseltäni menisi useita tunteja aikaa ja joissa virheiden määrä olisi todennäköisesti suuri, voi kirjastojen avulla toteuttaa muutamissa minuuteissa dokumentaation ohjeita seuraten. Esimerkiksi sivulla näytettävät animaatiot, joiden matematiikan miettiminen, ymmärtäminen ja ohjelmallinen toteutus vaatisivat suuren määrän ohjelmointitaitoa, onnistuvat JavaScript-kirjastoa käyttäen muutamalla rivillä ja muutamia funktion parametreja säätämällä.

Javascript-kirjastot eroavat ominaisuuksiltaan ja käyttötarkoituksiltaan. NBA Fantasialiiga-sivustolle valitsin käyttöön Jquery-kirjaston sen hyvän dokumentaation ja helposti opittavan kirjoitustyylin vuoksi. Jquery-tiedosto saadaan 32 kilotavun kokoiseksi, joten se ei tee sivusta liian raskasta, mutta tuo todella paljon toiminnallisuutta ja mahdollisuuksia. Jquery-kirjastoa voi myös laajentaa lataamalla tarkempiin käyttötarkoituksiin räätälöityjä lisäosia. [16.]

Jquery-JavaScript-kirjaston käyttö

Yksi hyödyllisimmistä Jqueryn ominaisuuksista on yhteensopivuus selainten kesken. Varsinkin vanhempien Internet Explorer -selaimen versioiden ja nykyaikaisten selainten välillä on eroavaisuuksia toimintojen kirjoittamisessa. Jqueryn avulla eroavaisuuden piilotetaan rajapinnan taakse eikä ohjelmoijan tarvitse kirjoittaa samaa toiminnallisuutta useaan kertaan tai muistaa itse lisätä konditionaalilausekkeita eri selaimille.

Jqueryn voi lisätä sivustolle eri tavoilla. Koodikirjaston voi ladata dynaamisesti määrittelemällä script-elementin src-attribuutille ulkoisen verkko-osoitteen, josta kirjasto ladataan. Tämä voi olla hyödyllistä, jos haluaa hyödyntää Jqueryn omaa tai Googlen palvelinta, josta kirjasto tarjotaan. Toinen vaihtoehto on ladata Jquery-kirjasto omalle palvelimelle ja liittää se paikallisesti. Kun tiedosto on omalla palvelimella, voi olla varma, että versio on juuri se, jonka on alun perin ladannut. Ulkoisilla palvelimilla verkko-osoite saattaakin yhtäkkiä tarjota uutta versiota, joka ei ole jostain syystä taaksepäin yhteensopiva tai niin turvallinen kuin versio, jolla verkkosivusto on kehitetty. Ulkoiset versiot kuitenkin tarjoavat parempaa kaistakapasiteettia ja tiedostojen tallentamista välimuistiin. [16; 17; 18.]

NBA Fantasialiiga -sivuston vaatimat Javascript-tiedostot liitin lisäämällä script-elementit aivan sivun loppuosaan, muun HTML:n jälkeen. Tämä varmistaa, että sivuston näkyvät elementit latautuvat ja ovat saatavilla nopeammin. Jos Javascript-tiedostot olisi lisätty sivun alkuosassa, muut elementit eivät latautuisi, ennen kuin Javascriptit ovat valmiita, ja käyttäjälle näytettäisiin mahdollisesti tyhjää sivua epämiellyttävän kauan. Samalla, jos jotain koodia olisi tarkoitus suorittaa heti Javascript-tiedoston latauduttua, voisi käydä niin, että koodi kutsuisi dokumentin elementtiä, joka ei vielä ole latautunut, ja koodin suoritus keskeytyisi. [18.]

Jquery-kirjastossa toiminnallisuus on sidottu yhteen muuttujaan. Koska Javascript sallii tietyt merkit muuttujien nimissä, nimetään muuttuja yleensä ytimekkyyden vuoksi yksinkertaisesti dollarimerkiksi: $. Tästä muuttujasta kutsutaan Jqueryn toimintoja. [16.]

Sivun HTML-elementtejä voi hakea olioina Jqueryn kätevän valitsijan avulla, kuten koodiesimerkissä 2.

```
var otsikko = $('#otsikko');
otsikko.html('Uusi otsikko');
$('.kappaleet p').each(function() {
          $(this).addClass('korostettu');
});
```

Koodiesimerkki 2. Koodinpätkä esittelee, miten Jqueryyn perustuvaa Javascript-koodia kirjoitetaan.

Koodin ensimmäisellä rivillä otsikko-nimiseen muuttujaan sidotaan Jqueryn palauttama HTML-olio elementistä, jonka id-attribuutin arvo on ”otsikko”. Ristikkomerkki ennen sanaa otsikko tarkoittaa, että haetaan id-attribuutin perusteella. Piste ennen sanaa viittaa CSS-luokkanimeen. Teksti ilman pistettä tai ristikkomerkkiä viittaa HTML-elementin nimeen. Jqueryssa on sisäänrakennettu valitsinkone, joka hakee elementtejä perustuen erikoisempiinkin valintakriteereihin. Palautetulla oliolla on Jqueryn antamia ominaisuuksia, joten sitä voi helpommin manipuloida kuin natiivin Javascriptin palauttamaa DOM-oliota. [16.]

4 CSS-ohjelmistokehykset

Ulkoasusuunnitteluun on myös omia ohjelmistokehyksiä. CSS-ohjelmistokehykset ovat hieman erityyppisiä perinteisiin ohjelmistokehyksiin verrattuina. CSS:n tarkoitus on olla näkymän tyylejä ja sivuston ulkoasua määrittelevä kokoelma tyylimääritelmiä. CSS:n merkintä on suoraviivaista eikä tarjoa kovinkaan paljon varaa erilaisille toteuttamistavoille. Kuitenkin myös CSS:nkin kirjoittamiseen on apuvälineitä, jotka tehostavat verkkokehitystä.

4.1 LESS: dynaaminen tyylimääritelmäkieli

LESS on CSS:ää laajentava tyylimääritelmäkielen ja Javascript-tiedoston kokoonpano. LESS-kieli on CSS-kielen näköistä, mutta lisää siihen useita puuttuvia ja kaivattuja ominaisuuksia, kuten muuttujien käyttö, yhden tyylin määritelmien sulauttaminen toiseen tyyliin ja muutamat perusfunktiot.

LESS-kielen tarjoamat muuttujat auttavat koodia muutettaessa. Sivustoilla on tavallisesti yksi tai muutama määrittelevä perusväri. LESS:n avulla värin voi tallentaa muuttujaan ja käyttää uudelleen kaikkialla, missä sitä tarvitaan. Koodiesimerkissä 3 väriarvo tallennetaan muuttujaan.

```
@minun-sininen: #2d62db;

.runko {
          color: @minun-sininen;
          background: #fff;
}
h1 {
          background: @minun-sininen;
}
```

Koodiesimerkki 3. Muuttujan määrittäminen ja käyttö LESS-kielellä.

Muuttuja @minun-sininen on määritetty vastaamaan väriarvoa #2d62db. Kun väriä halutaan käyttää useassa eri paikassa uudelleen, muuttujan nimi on paljon helpompi muistaa kuin väriarvon koodi. Jos sinisen sävyä haluaa myöhemmin muuttaa, riittää, että vaihtaa arvon muuttujalle. Ei siis tarvitse erikseen etsiä jokaista kohtaa, jossa väriä on käytetty ja vaihtaa uutta värikoodia tilalle. (Jos myöhemmin haluaa vaihtaa sinisen sävyä, riittää muuttujan arvon vaihtaminen – ei siis tarvitse etsiä ja vaihtaa jokaista värikoodikohtaa.) [19.]

Funktioiden käyttö CSS:n välissä tehostaa vastaavasti sijoittelua ja sivuston elementtien geometrian suunnittelua. Muuttujalle voi suorittaa hyödyllisiä laskutoimituksia, kuten koodiesimerkissä 4.

```
@leveys: 560px;

.artikkeli {
          width: @leveys - 10;
}
```

Koodiesimerkki 4. Muuttujan käyttö laskutoimituksessa.

Leveys-nimiselle muuttujalle on määritelty arvo 560 pikseliä. HTML-elementit, joilla on artikkeli-luokka liitettynä, halutaan aina 10 pikseliä kapeammaksi kuin määritelty leveys. LESS mahdollistaa perus matemaattiset operaatiot, ja artikkeli-luokan leveys voidaan määrittää lausekkeen erotukseksi.

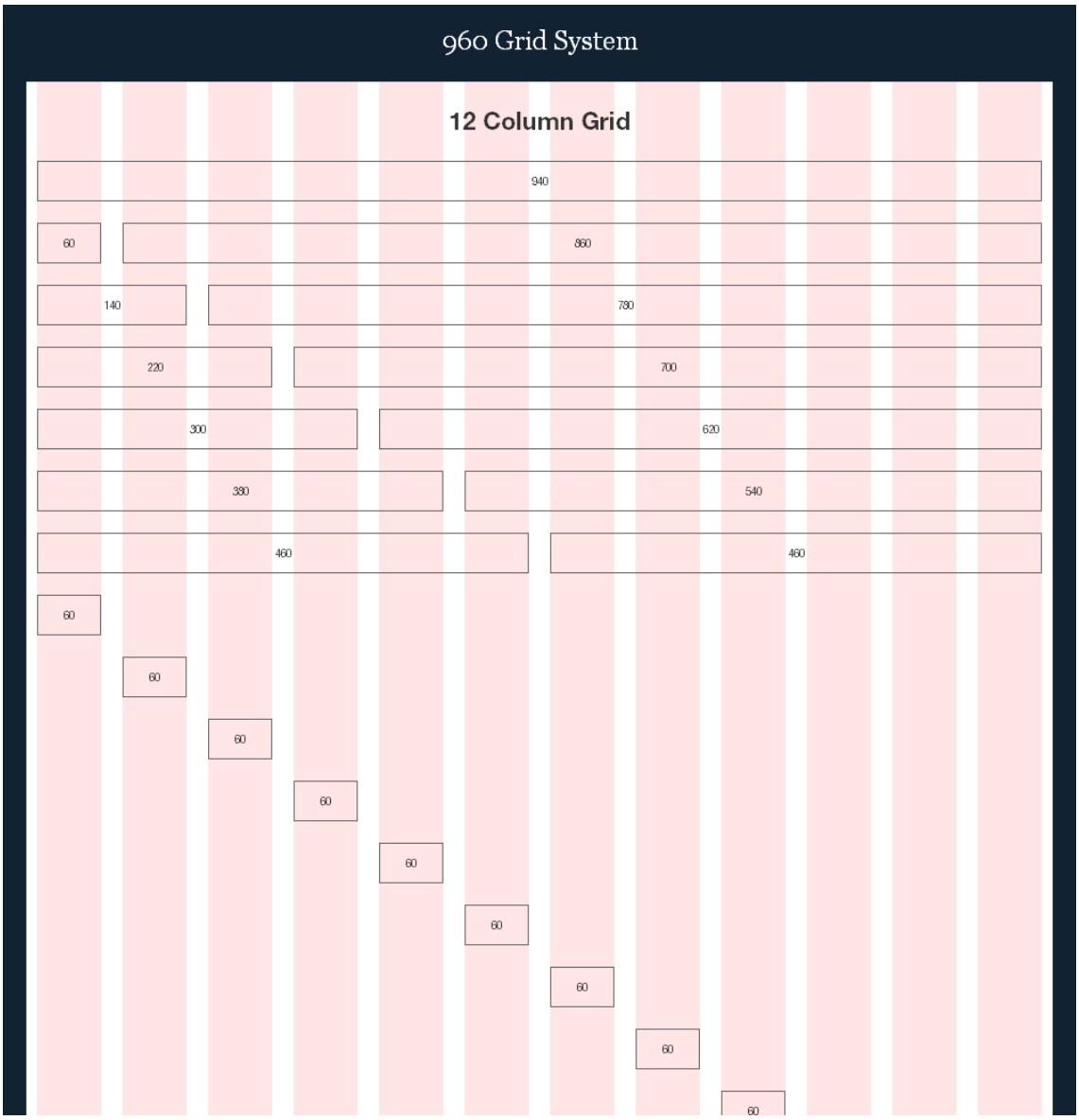
LESS-tiedostot voi muuntaa CSS-tyyleiksi sivun latauksen yhteydessä liittämällä LESS-tyylitiedoson jälkeen sivulle LESS-Javascript-tiedoston, joka hoitaa muuntamisen käyttäjän selaimen tehoa hyödyntäen. LESS-tyylitiedoston voi myös muuntaa CSS-tiedostoksi ennen verkkoon sijoittamista ja näin säästää jonkin verran käyttäjän resursseja. [19.]

4.2 Ruudukkojärjestelmät

Sivuston asetteluun on useita ohjelmistokehyksiä, joita sanotaan ruudukkojärjestelmiksi. Ruudukkojärjestelmä liitetään sivustoon CSS-tiedostona, ja se mahdollistaa elementtien sijoittelun helposti eri kohtiin verkkosivua. Lähtökohtana ruudukkojärjestelmässä on jakaa sivu pystysuuntaisiin palstoihin eli sarakkeisiin, jotka toimivat rakennuspalikoina sivun asettelulle. Sarakkeet ovat tasalevyisiä, ja niissä voi mahdollisesti olla jonkinkokoiset hengitystilaa antavat ”rännit” eli tyhjät tilat sarakkeiden väleissä. [20.]

CSS-tyyliluokkia on kahdenlaisia, leveysluokat ja marginaaliluokat. Leveysluokat määrittävät, kuinka monen sarakkeen leveyden verran HTML-elementti vie tilaa. Marginaaliluokat määrittävät, kuinka monen sarakkeen verran elementti on irti lähimmästä vasemmalla olevasta reunasta, olkoon se sitten edeltävän elementin oikea reuna tai ruudukon sisältävän kehyksen vasen reuna. [20.]

Kuvassa 1 on 960 Grid Systemin 12-sarakkeinen ruudukko. Ruudukon sarakkeet on väritetty vaaleanpunaisella. Sarakkeiden päällä näkyy erikokoisia elementtejä, jotka havainnollistavat hyvin leveys- ja marginaaliluokkia.



Kuva 1. 960 pikseliä leveä ruudukkojärjestelmä, joka jakaa sivun 12 sarakkeeseen [27].

Kuvan 1 ylimmällä rivillä näkyy koko sivun levyinen laatikko. Tällä laatikolla on leveysluokka, joka määrittää sille 12 sarakkeen leveyden. Seuraavat rivit näyttävät erikokoiset laatikot, joilla on leveysluokat yhdestä sarakkeesta yhteentoista.

Porrasmaisesti näkyvät yhden sarakkeen levyiset laatikot esittelevät marginaaliluokkia. Ensimmäisellä laatikolla ei ole yhtään marginaalia, seuraavat alempana olevat esittelevät, kuinka marginaaliluokat työntävät elementtiä oikealle tietyn sarakemäärän verran.

Ruudukkojärjestelmät voivat perustua kiinteään leveyteen, jolloin sarakkeen leveys on vakio, tai ne voivat olla venyviä, jolloin sarakkeen leveys on määritelty suhteessa koko

ruudukon leveyteen. Ruudukoissa sarakkeille on usein määritelty ”kourut” eli tyhjä tila sarakkeiden välillä. Kuvassa 1 kourut näkyvät valkoisena vaaleanpunaisten sarakkeiden välissä. Kourujen tarkoitus on antaa hengitystilaa sivun eri elementeille.

4.3 Joustava ruudukko

Ruudukon ja sarakkeiden leveyksien ei tarvitse olla kiinteitä. 960 pikseliä leveä ruudukko, jossa on kaksitoista 60 pikseliä leveää saraketta, voidaan muuttaa joustavaksi yksinkertaisimmillaan muuttamalla leveydet suhteellisiksi. [21.]

Laskemalla sarakkeen ja kourun suhteellinen leveys ruudukon kokoon nähden saadaan alkeellinen joustava ruudukko. Jos sarakkeen kiinteä leveys on 60 pikseliä 960 pikseliä leveässä ruudukossa, sen suhteellinen leveys on 60/960 eli 6,25 %. Kourun leveyden ollessa 20 pikseliä sen suhteellinen leveys on 20/960 eli 2,0833 %. Koska kourun leveys koostuu kahden sarakkeen sivumarginaalien summasta, sarakkeen sivumarginaaliksi tulee 10/960 eli 1,04166 %. [21.]

Kun muutetaan leveydet suhteellisiksi, ruudukko venyy ja kutistuu leveyssuunnassa, kun ruudukon kokonaisleveyttä muuttaa suuremmaksi tai pienemmäksi kuin 960 pikseliä. Joustava ruudukko on hyödyllinen, kun sivu vaatii joustavuutta leveyden suhteen mutta asettelu halutaan luoda järjestelmällisesti ruudukon päälle.

Joustavaa ruudukkoa käytettäessä on hyvä muistaa, että sivu venyy suhteellisesti vaikka kuinka pieneksi tai suureksi ja jossain vaiheessa sivun tekstin, kuvien ja muiden kiinteäkokoisten elementtien mitat tulevat vastaan. Sivun näkymän rikkoutumisen ehkäisemiseksi on hyvä ottaa käyttöön sivun leveyteen perustuvaa mukautuvuutta ja rajoitteita koon suhteen, yksinkertaisimmillaan ruudukon min-width eli minileveyden CSS-säännön määrittäminen. [21.]

4.4 Twitter Bootstrap -kehys

Suositun sosiaalisen verkoston, Twitterin, kehitysryhmän tuottama Bootstrap on kokoelma CSS-, HTML- ja JavaScript-tiedostoja, jotka lisättyinä verkkosovellukseen tuovat heti saataville useita suosittuja ja hyväksi todettuja toimintoja sekä rakenteellisia komponentteja ja tyylejä [22].

Jquery vaaditaan Twitter Bootstrapin Javascript-liitännäisten toimimiseen. Jotta Bootstrapin kaikki toiminnallisuus olisi käytössä, sivuun tulee liittää Jquery-kirjaston uusin versio ja Bootstrapin mukana tuleva Javascript-tiedosto. Tietyt komponentit eivät toimi ilman Javascript-tukea. [22.]

Bootstrapiin kuuluu 12-sarakkeinen CSS-ruudukkojärjestelmä sekä staattisena että reagoivana versiona. NBA Fantasialiigan sivustoa uusittaessa sivustoon sovellettiin reagoivaa, joustavaa ruudukkoa. Tämä tarkoittaa sitä, että sivusto on automaattisesti pelkkien Bootstrapin CSS-tiedostojen vakiomääritelmien ansiosta valmis monelle alustalle. Rakentaminen aloitettiin Bootstrapin tarjoaman HTML-dokumenttipohjan päälle. Tämä pohja ei sido minkäänlaisiin sisällöllisiin ratkaisuihin, mutta antaa kehittäjälle helpon ja monipuolisen lähtökohdan sivuston luomiseen.

Bootstrapin lataussivulla oli mahdollisuus koota ladattavat tiedostot niistä komponenteista, joita tulee projektissa tarvitsemaan. Jättämällä pois CSS- ja Javascript-koodia tiedostokoko jää pienemmäksi. Lataussivulla oli myös mahdollisuus muokata CSS-tyylejä oman sivun näköiseksi monipuolisen lomakkeen kautta. Lomaketta tai Bootstrapin tarjoamaa LESS-tiedostoa käytettäessä CSS:n perusarvoja, värejä, sarakkeiden määrää ja typografiaa on helpompi muokata kuin etsimällä arvoja CSS-tiedostosta. [22.]

4.5 Joustava ruudukko ja reagoiva sivu

Valitsin joustava ruudukon, jotta heti alusta alkaen voisin varmistaa sivuston näyttävän hyvältä kaikenkokoisilla laitteilla. Liigaan osallistujat ovat kaikki henkilöitä, jotka omistavat älypuhelimen ja hyötyisivät sivustosta, joka muuntautuu pienelle ruudulle kätevästi. Kehittämisen alkuvaiheesta lähtien on hyvä pitää mielessä mobiilikäyttäjät, sillä yhä useampi selaa verkkoa älypuhelimella tai tabletilla. Näiden käyttäjien huomioiminen pakottaa kehittäjän ja suunnittelijan jättämään pois kaiken ylimääräisen, mikä tekee käyttöliittymästä helpomman oppia ja nopeamman käyttää. [23.]

Joustava ja reagoiva ruudukko perustuu CSS-tyylimääritelmien mediaquery-sääntöihin ja suhteellisiin kokomääritelmiin. Kuten aiemmin totesin, joustavassa ruudukossa sarakkeiden arvot perustuvat suhteellisiin yksiköihin, kuten prosentteihin tai typografiseen, tekstin M-kirjaimen pistekokoon perustuvaan mittayksikköön em:iin. Sivuston eri

osien leveydet eivät ole vakioita pikselimääritteisiä kokoja, eli ne muuttuvat esimerkiksi sivun leveyden pienentyessä tai kasvaessa. [21; 23.]

Koodiesimerkkissä 5 on ote Bootstrapin responsiivisesta tyylimääritelmästä, jossa on säännöt vakiokokoisille ruudukoille ja joustaville ruudukoille.

```
/* vakiokoot */
  .span12 {
    width: 724px;
   }
  .span11 {
    width: 662px;
  }
  .span10 {
    width: 600px;
  }
  .span9 {
    width: 538px;
  }

/* muuttuvat koot */
  .row-fluid > .span12 {
    width: 99.999999993%;
  }
  .row-fluid > .span11 {
    width: 91.436464082%;
  }
  .row-fluid > .span10 {
    width: 82.87292817100001%;
  }
  .row-fluid > .span9 {
    width: 74.30939226%;
  }
```

Koodiesimerkki 5. Kiinteät vakiokoot on määritelty pikseleinä, ja muuttuvat koot on määritelty prosentteina.

Pikseleinä merkityt koot ovat aina samanlevyisiä riippumatta selainikkunan leveydestä. Prosentuaaliset koot muuttuvat selainikkunan koon mukaan ja antavat sivustolle mukautuvamman asettelun. [21.]

Ruudukosta saa reagoivan, kun lisää mediaquery-sääntöjä, jotka määrittävät tiettyjä pisteitä, joissa leveys on suosituimpien laitteiden selaimia vastaava. Esimerkiksi yli 1200 pikselin selainikkunan leveys todennäköisesti tarkoittaa sivustoa käytettävän kannettavalla tietokoneella tai pöytätietokoneella. Toisaalta 320 pikseliä tai sitä pienempi selaimen leveys tarkoittaa suurella varmuudella pystyasennossa olevalla älypuhelimella sivustoa tarkkailevaa käyttäjää. Leveyspisteitä käyttämällä voi muuttaa sivuston asettelua mukautumaan vielä paremmin käyttäjän selainikkunan näkymän. [21.]

Jos vähintään 1200 pikseliä leveällä selainikkunalla on mukava katsella neljää saraketta vierekkäin, voi tekstille annettava tila mennä liian kapeaksi, kun leveys onkin 768 pikseliä (iPad pystyasennossa), vaikka joustavalla ruudukolla ne eivät leviäisi yli selainikkunan leveyden. On parempi siirtää kaksi saraketta seuraavalle riville, jolloin teksti mukautuu paremmin leveyteen sopivaksi. Mobiilikäyttäjälle on todennäköisesti parasta näyttää vain yksi sarake kerrallaan ja siirtää seuraavat aina omalle rivilleen.

Mediaquery-merkinnässä määritellään, mitkä ehdot toteuttavat mediaqueryn sisältämät säännöt.

```
@media (min-width: 768px) and (max-width: 980px) {
          .row {
                    margin-left: -20px;
                    *zoom: 1;
          }
          .row:before, .row:after {
                    display: table;
                    content: ”“;
          }
          ...
}
```

Koodiesimerkki 6. Mediaquery-sääntö, joka koskee 768–980 pikseliä leveitä selainikkunoita.

Mediaquery-sääntö määrittää, että selainikkunan leveyden ollessa vähintään 768 pikseliä ja enintään 980 pikseliä, aaltosulkeiden sisällä olevat CSS-säännöt on otettava huomioon. Siis määritelmiä, joita on tehty ”row”-luokalle, voidaan määrittää uudelleen ja ne vaihtelevat selainikkunan leveyden perusteella. [21.]

Tämä kaikki on Bootstrapin mukana tulevaa sisäänrakennettua toiminnallisuutta. Käyttämällä Bootstrap-ohjelmistokehyksen tarjoamia valmiita tyylejä säästin paljon koodausaikaa ja sain kätevät toiminnallisuudet käytännössä asettamalla vain sivupohjan HTML:n paikoilleen.

5 Muutokset sivuston uudessa versiossa

5.1 Aiemman version puutteita ja huonoja puolia

Aiemman sivustoversion ongelmana oli se, että se oli rakennettu vaiheittain lisäten toiminnallisuuksia, jotka välillä olivat ristiriidassa muiden toimintojen kanssa, jolloin piti

palata aiempaan vaiheeseen muuttamaan jo sovellettuja toteutuksia. Tämä johti huonoihin ohjelmointikäytäntöihin, kompromisseihin ja huolimattomaan koodiin. Sivustossa oli todettavissa useita huonoja tietoturvakäytäntöjä.

Verkkokehityksessä on tärkeä muistaa aina suodattaa käyttäjän syöttämä tieto ja muuntaa sivulle tulostettava syöttö turvalliseen muotoon [24]. NBA Fantasialiiga -sivuston vanhassa versiossa oli paljon tietoturvapuutteita. Kun ylläpitäjä esimerkiksi muutti joukkueen pelaajan nimeä, pelaajan uusi nimi vastaanotettiin ja lisättiin tietokantaan koodiesimerkissä 7 esitetyllä tavalla.

```
include_once('db.php');
$pid = (int) $_POST['pid'];
$pname = addslashes($_POST['newname']);
$parvo = (int) $_POST['newprice'];
$query = "update nba_pelaajat set pelaaja = '$pname',  arvo =
'$parvo'  where pid = '$pid';";
$result = mysql_query($query);

if($result) echo "success";
mysql_close();
```

Koodiesimerkki 7. Käsinkirjoitettu SQL-komento, joka päivittää pelaajan tiedot tietokannassa.

Koodiesimerkissä 7 muodostetaan ensin yhteys tietokantaan hakemalla db.php-tiedosto. Tämän jälkeen sijoitetaan tietokantaan menevät arvot muuttujiin $pid, $pname ja $parvo. Muuttujien $pid ja $parvo tulisi olla kokonaislukuja, joten niiden suodatus tapahtuu helposti langettamalla ne kokonaisluvuiksi (int)-komennolla. Nimimuuttuja $pname sen sijaan ei ota ollenkaan suodatusta, vaan siihen ainoastaan lisätään addslashes-komennolla vinoviivat, mikä jonkinasteisesti suojaa tietokantainjektiolta. Jos $pname:n arvo olisi ”Ville'; --” addslashes-komento lisäisi '-merkin eteen vinoviivan \, joka estäisi sitä vaikuttamasta kyselyyn. Ilman vinoviivaa '-merkki sulkisi pelaaja = ' -alkuisen osan, ja tämän jälkeen käyttäjän mahdollisesti nimiarvoon laittamat muut komennot tulisivat osaksi kyselyä, tässä tapauksessa ”; --” eli kyselyn päättäminen puolipisteellä ja loppukyselyn kommentoiminen pois kahdella katkoviivalla. Tuloksena olisi kaikkien pelaajien nimen muuttuminen Villeksi. [11; 13; 24.]

Ongelma pelkän addslashes-komennon käytössä on se, että siinä ei suodateta pois tietoa, jota ei tahtoisi ollenkaan pelaajan nimeen. Ehkä pelaajan nimessä halutaan sallia ainoastaan aakkoset ja välilyönnit. Kehittäjänä en voi olla varma siitä, että tietokannassa oleva tieto on turvallista. Tämän vuoksi on myös varmistettava, ettei tietokannas-

ta haettava tieto voi tehdä mitään ikävää, kun se näytetään sivulla. Vanhalla sivustolla joukkueen pelaajien tiedot tulostettiin sivulle koodiesimerkin 8 esittämällä tavalla.

```
echo "<table><tr>
    <th>Nimi</th><th>Arvo</th><th>Paikka</th></tr>\n";
while($row=mysql_fetch_assoc($result2)) {
    echo "<tr>";
    echo "<td>".stripslashes($row['pelaaja'])."</td>";
    echo "<td>".$row['arvo']."</td>";
    echo "<td>".$row['paikka']."</td>";
    echo "</tr>\n";
}
echo "</table>";
```

Koodiesimerkki 8. Tietokantakyselyn palauttamat rivit tulostetaan sivulle taulukkoon.

Koodiesimerkin 8 koodilla luodaan HTML-merkinnällä taulukko ja täytetään sen solut pelaajan tiedoilla. Esimerkin koodissa on useita tietoturvapuutteita. Kun pelaajan tiedot tulevat tietokannasta, koodissa oletetaan pelaajan arvon ja paikan olevan turvallisesti tulostettavissa, sillä ne tulevat tietokannasta kentistä, joiden arvot voivat olla vain kokonaislukuja (arvo) tai tietyn turvallisen arvojoukon arvoja (paikka). Voidaan kiistellä siitä, onko tarpeen käyttää suoritintehoa arvojen muuttamiseen turvalliseen muotoon, vai voiko luottaa täysin siihen, ettei tietokantaan millään voisi asettaa vaarallisia arvoja. Erityistä huomiota on kiinnitettävä kohdassa, jossa pelaajan nimi tulostetaan $row['pelaaja']-muuttujasta. Muuttujaa ei muokata muutoin kuin poistamalla stripslashes-komennolla ne vinoviivat, jotka lisättiin pelaajan nimeä lisättäessä tietokantaan. Kun pelaajan nimeä ei kerran suodatettu mitenkään tietokantaan lisättäessä, sivusto jää tämän suoran tulostuksen vuoksi avoimeksi erilaisille hyökkäyksille. [24.]

Otetaan tapaus, jossa pelaajan nimeksi olisi asetettu <script src=http://esimerkki.fi/komento.js></script>. Taulukkoon ei tulostuisi pelaajan nimeä, vaan selain käsittelisi nimen omana HTML-elementtinään ja lataisi ja suorittaisi komento.js-tiedoston. Tämänkaltainen tietoturva-aukko on suhteellisen helppo korjata. Kun ei kerran voi luottaa tietokannasta tulevien arvojen olevan turvallisia tulostettavaksi sivulle, tulee muuttaa arvoja turvalliseksi tekstiksi, jota ei käsitellä HTML-merkintänä. PHP:ssä on komento, joka auttaa tekemään juuri tämän; se on htmlspecialchars. Htmlspecialchars muuttaa tietyt erikoismerkit niiden entiteettimuotoihin. Entiteetit ovat erikoismerkintää, joka näytetään tiettynä merkkinä kirjoittamatta sitä merkkiä itseään. Esimerkiksi ”pienempi kuin” -merkki kirjoitetaan entiteettimuodossa <. Kun selain

näkee HTML:n joukossa merkinnän < se tietää näyttää ruudulla <. Tällöin <-merkkiä ei luulla osaksi HTML-merkintää, kuten koodiesimerkki 9 havainnollistaa. [25; 26.]

```
$arvo = ”<script>...</script>”;
echo htmlspecialchars($arvo);
// tulostaa  &lt;script&gt;...&lt;/script&gt;
// näkyy ruudulla <script>...</script>, ei suoriteta HTML:nä
```

Koodiesimerkki 9. Htmlspecialchars-komento muuttaa merkkijonon erikoismerkit entiteeteiksi.

Jos koodiin lisää htmlspecialchars-komennon, se varmistaa sen, että tietokannasta tulevan tiedon ollessa paha-aikeista tiedon uskaltaa tulostaa ruudulle ilman, että se luetaan osana HTML-merkintää.

5.2 Tietoturvan parantaminen Symfonyn avulla

Doctrinen ORM:n käyttäminen tietojen syötössä varmista, etteivät tallennettavat muuttujien arvot voi vaikuttaa kyselyyn. Itse ei tarvitse varmistaa, että tietokantaan menevät arvot on muutettu turvalliseen muotoon. Doctrinen varaan voi jättää kyselyn rakentamisen. Doctrinen kautta arvoja ei aseteta kyselyyn itse. Ne määritellään tietokannan riviä vastaavaan entiteettiolioon, joka tallennetaan tietokantaan. [6; 12.]

Symfonyn mukana tulevassa Twigissa on sisäänrakennettu ominaisuus, jolla voi asettaa kaikkien muuttujien muuntamisen turvalliseen muotoon automaattiseksi. Kaikki sivulla näkyvät muuttujat ovat siis valmiiksi turvallisessa muodossa, eikä tarvitse muistaa jokaisen kohdalla tehdä htmlspecialchars-komentoa erikseen. [6; 15.]

Uudella NBA Fantasialiiga -sivustolla oman joukkueen tarkastelusivulla pelaajat listataan koodiesimerkin 10 esittämällä tavalla.

```
<div class="span6">
  <table class="table" id="myteam">
  <thead><tr>
      <th>Nimi</th>
      <th>Arvo</th>
      <th colspan="2">Paikka</th>
  </tr></thead>
  <tbody>
  {% for p in players %}
      <tr id="{{ p.pid }}">
          <td>{{ p.name }}</td>
          <td>{{ p.value }}</td>
          <td>{{ p.position }}</td>
```

```
            <td style="width:50px">
                <div class="btn-group">
                    <a class="btn dropdown-toggle"
                        data-toggle="dropdown" href="#">
                        <i class="icon-cog" title="Toiminnot"></i>
                        <span class="caret"></span>
                    </a>
                    <ul class="dropdown-menu myteamdrop">
                        <li><a href="#"
    onclick="promptTradePlayer({{ p.pid }});return false;">
                            <i class="icon-share-alt"> </i>
                            Siirrä toiseen joukkueeseen</a></li>
                        <li><a href="#"
    onclick="promptChangePosition({{ p.pid }}); return false;">
                            <i class="icon-edit"> </i>
                            Muuta pelipaikkaa</a></li>
                        <li><a href="#"
    onclick="removePlayer({{ p.pid }});return false;">
                            <i class="icon-minus"> </i>
                            Poista pelaaja</a></li>
                    </ul>
                </div>
            </td>
        </tr>
  {% endfor %}
  </tbody>
  </table>
</div>
```

Koodiesimerkki 10. Pelaajien tietojen tulostaminen taulukkoon Twig-komentoja hyödyntäen.

Koodiesimerkissä 10 Twig-komennot näkyvät kahdella kaarisuluilla ympyröityinä. Esimerkiksi {{ p.name }} tulostaa sivulle pelaajan nimen. Erillistä komentoa, kuten PHP:n htmlspecialchars, ei tarvita tekstin muuttamiseen turvalliseen muotoon. Twig hoitaa tämän osuuden automaattisesti. [6.]

Sivun merkintää ja rakennetta suunniteltaessa ja koodattaessa ei Twigin ansiosta tarvitse yrittää aina muistaa oikeita komentoja ja varmistaa, että niitä on käyttänyt. Twig luo automaattisesti yksinkertaisen, mutta tehokkaan tason suojausta sivulle.

Tarkempien rajoitusten lisääminen vastaanotetulle tiedolle

Vaikka esitettyä tekstiä muokataan siten, että sitä ei voi tulkita HTML-merkintänä, on myös hyvä käytäntö varmistaa, ettei sellaista tietoa alun perin joudu tietokantaan. Teksti saatetaan jossain muussa käyttötapauksessa hakea tietokannasta sivulle, jossa se näytetään raakana. Tällöin tietokannassa oleva teksti pääsee tekemään tuhojaan. Vaikka voisi olla varma, että tekstiä ei ikinä näytetä täysin raakana syöttönä tietokan-

nasta, ei silti ole käyttäjäystävällistä näyttää sivulla epämääräistä erikoismerkeistä ja koodinpätkistä koostuvaa tekstiä.

Pelaajilla on monenlaisia nimiä; joillakuilla on nimessään heittomerkki tai toisilla katkoviiva. NBA:ssa on useita pelaajia maista, joiden aakkosissa on erilaisia aksenteilla tai muilla merkeillä varustettuja kirjaimia. Liigassa on kuitenkin käytäntönä usein muuttaa aksentoidut kirjaimet englanninkielisten aakkosten lähimmäksi vastaavaksi kirjaimeksi. Esimerkiksi ensimmäinen suomalainen NBA:ssa pelannut henkilö Hanno Möttölä listataan liigan rekistereissä nimellä Hanno Mottola ja slovenialainen Saša Vujačić listataan nimella Sasha Vujacic.

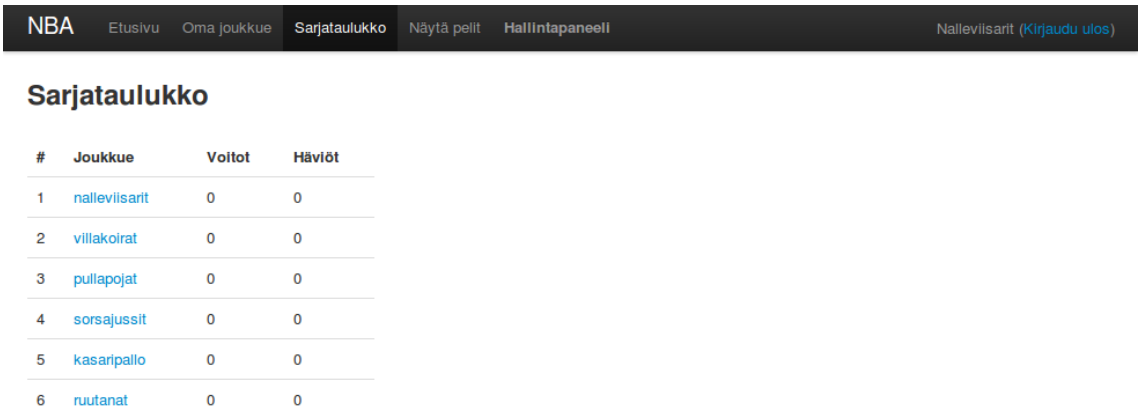
Kun tiedetään, että NBA:ssa pelaavien pelaajien nimet voi rajata englanninkielisiin aakkosiin sekä katkoviivaan ja heittomerkkiin, on mahdollista kontrolloida tiukemmin, mitä tietoja tietokantaan syötetään. [24.]

5.3 Jquery uudessa versiossa

NBA Fantasialiiga -sivuston uudessa versiossa Bootstrapin Jquery-pohjaista JavaScriptia käytetään muun muassa ”oma joukkue” -sivulla. Kun käyttäjä napsauttaa pelaajan rivin päädyssä olevaa toimintopainiketta, toiminnot ilmestyvät pudotusvalikkona Bootstrapin oman koodin avulla. Kun valitaan pelaajan paikan muuttaminen tai siirtäminen toiseen joukkueeseen, Jquery-tehostettu modaali, eli tyylitelty ponnahduselementti, ilmestyy. Itse siirtämis- ja paikanvaihtotoiminnot on myös kirjoitettu Jquery-koodilla, jolla haetaan helposti tarvittava arvo, lähetetään se Ajax-pyynnöllä eteenpäin ja piilotetaan ja vaihdetaan elementtien arvoja vastauksen perusteella.

5.4 Mediaquery-sääntöjen käyttö

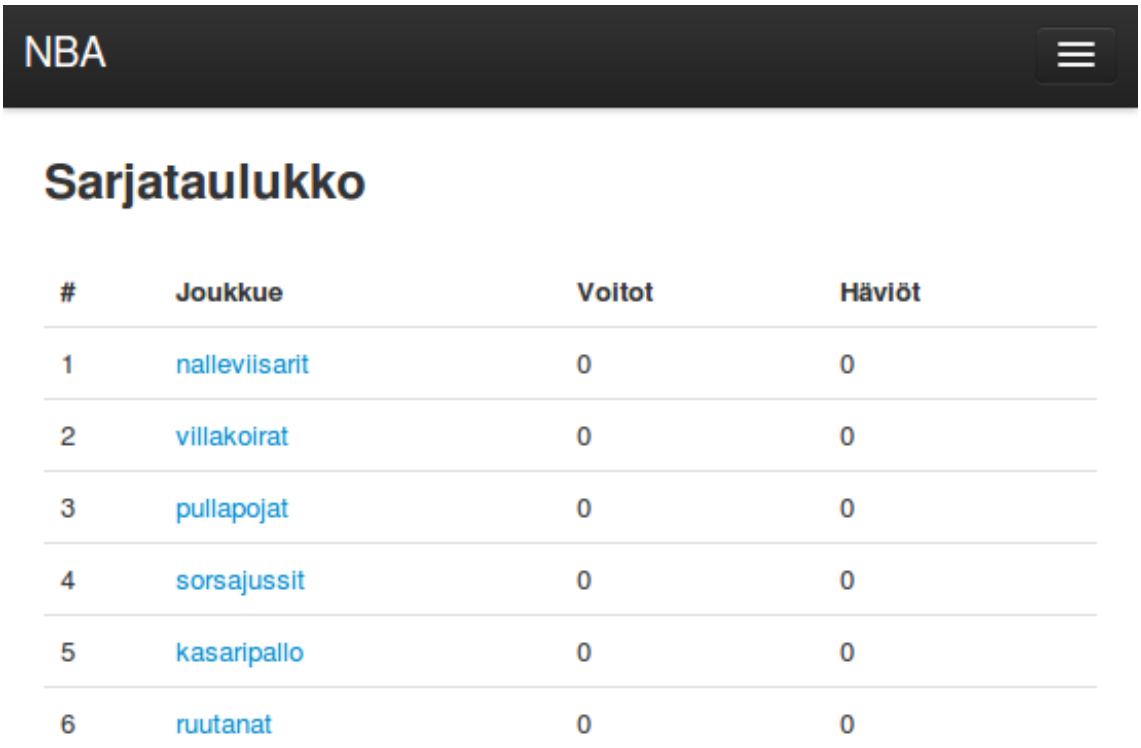
Mediaquery-sääntöjen näkyvimmät vaikutukset uusilla NBA Fantasialiiga -sivuilla ovat sivun yläosan navigaatiopalkissa. Kun sivusto on tyypillisen pöytä- tai kannettavan tietokoneen käyttäjän selaimen mitoissa, navigaatiopalkki pysyy koko ajan näkyvissä sivunäkymässä ylimpänä. Kuvassa 2 näkyy, kuinka navigaatiopalkki asettuu sivun yläreunaan, kun sivua katselee tarpeeksi leveällä selainikkunalla.



Kuva 2. Leveän selainikkunan sivu, jossa navigaatiolinkit näkyvät listattuna vierekkäin.

Kun sivun leveys kapenee, yläpalkin näkyvyys muuttuu siten, ettei se enää olekaan jatkuvasti näkyvillä, vaan vieritettäessä sivua alemmas se pysyy sivun yläosassa ja katoaa näkyvistä. Tämä toiminnallisuus antaa enemmän tilaa pystysuunnan käytölle, sillä kun vaakasuuntaista tilaa on vähemmän, usein myös pystysuuntainen tila vähenee.

Selainikkunan vielä kaventuessa navigaatiopalkissa tapahtuu selkeä muutos [kuva 3]. Vaakasuunnassa näkyneet linkit menevät piiloon ja oikeaan kulmaan ilmestyy painike, jota painamalla saa tuotua esiin sivun linkkivaihtoehdot – tällä kertaa rivin sijaan pinossa. Linkit eivät mahdu kaikki vaakariviin, joten ne on asetettava pinoon. Pino veisi kuitenkin liikaa tilaa ruudulta pystysuunnassa, joten linkit piilotetaan. Tämännäköinen rakenne on tablettitietokoneiden ja älypuhelimien sovelluksien tyypillistä navigaatiorakennetta mukaileva. Monet käyttäjät ovat tottuneet näkymissä siihen, että navigaatio on yhden painalluksen päässä.

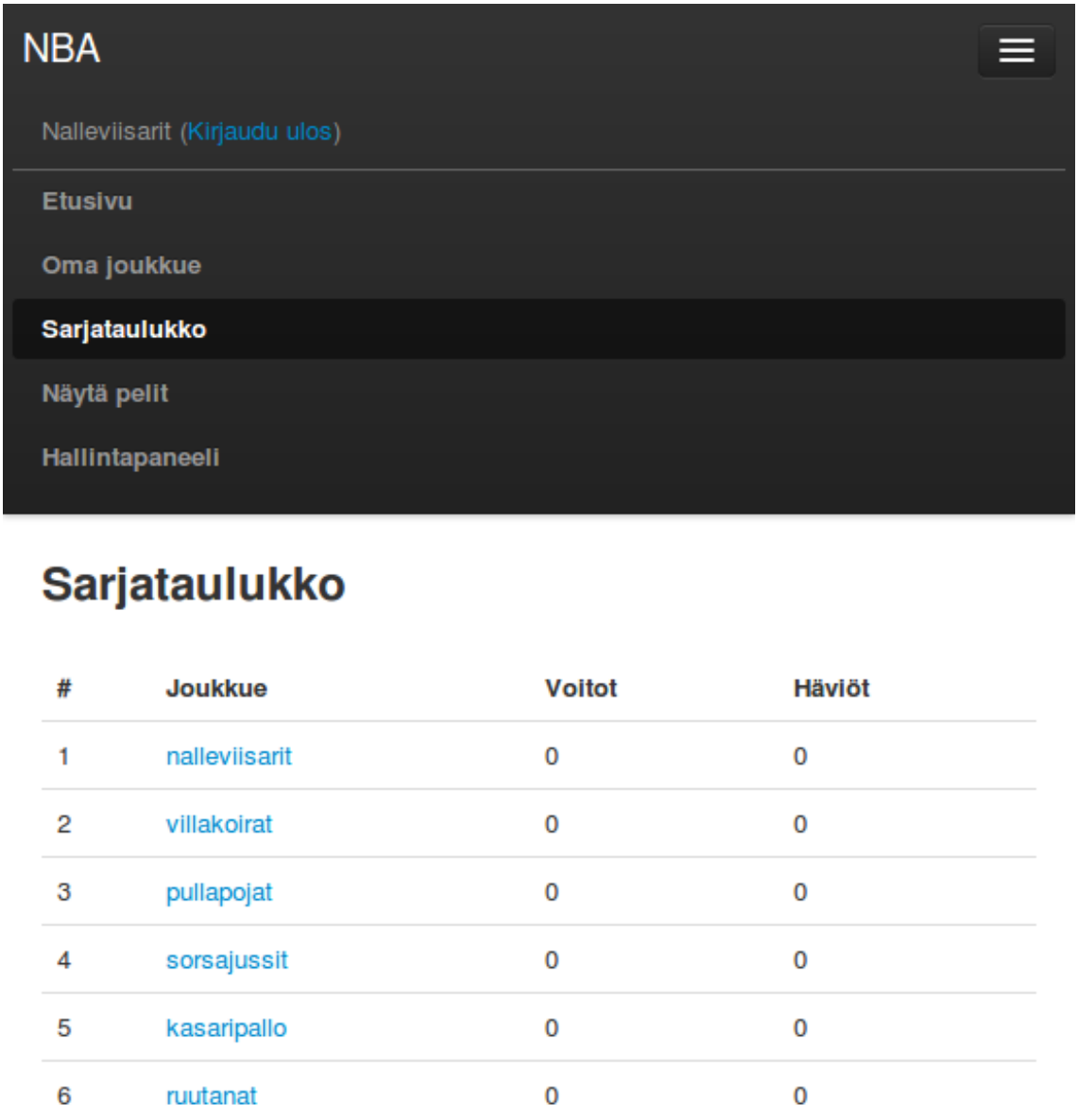


Kuva 3. Kapean selainikkunan navigaatiopalkki. Linkit ovat kadonneet vaakalistauksesta, ja ne on korvattu oikean yläreunan painikkeella, joka avaa navigaatiovalikon.

Navigaation avaavassa oikean reunan painikkeessa ei ole tekstiä, vaan kolme viivaa. Uudet käyttäjät voisivat ihmetellä, missä navigaatiolinkit ovat, ja he joutuisivat erikseen kokeilemaan ilman ennakkokäsitystä, mitä napin painaminen tekee. Koska oikeassa yläkulmassa oleva painike on kuitenkin erittäin vahvasti älypuhelinten ja tablettien käyttöliittymää mukaileva, päätin jättää painikkeen ilman tekstiä. Painikkeen toiminnallisuus ei myöskään ole senhetkistä työtehtävää haittaavaa. Navigaation avaaminen ei vie pelaajaa toiselle sivulle tai muuten haittaa hänen senhetkistä tehtäväänsä.

Kun tekee ratkaisuja verkkosivun elementeistä ja toiminnallisuudesta, on pidettävä mielessä kohdeyleisö ja käyttäjien muistiin kiteytyneet käytännöt. Jos kyseessä olisi aivan uudenlainen verkkosivuston selaustoiminnallisuus tai muu odottamaton navigaation sijoitus, vaihtaisin ehdottomasti kolmen viivan tilalle toimintoa kuvaavan sanan, kuten ”valikko”.

Vaikka Bootstrap tarjosi pitkälle viedyn valmiin pohjan, tein kuitenkin omia muutoksia koodiin, jotta sain navigaatiosta vielä toimivamman [kuva 4].



Kuva 4. Navigaatio avattuna. Linkit ovat nyt listattuna pystysuunnassa ja ”kirjaudu ulos” -linkki on tuotu vasemmalle.

Päätin tuoda ”kirjaudu ulos” -linkin vasemmalle samaan linjaan muiden linkkien kanssa. Siinä oli aluksi vielä tyyli, joka asetti sen oikealle kuten leveässä valikossa. Muutoksella ei ole toiminnallista vaikutusta, mutta se näyttää paremmalta pystynavigaatiossa. Ulos kirjautumisen linkin on joskus hyvä olla erillään muista linkeistä, ettei sitä paina vahingossa, mutta oikealle sijoitettuna se saa koko navigaation näyttämään rikkonaiselta.

Kun toin linkin vasemmalle, sain sen erotettua vielä muista linkeistä lisäämällä sille pienen viivan eristämään sen ikään kuin toiseen osioon.

6 Yhteenveto

Insinöörityönä tehdyn NBA Fantasialiiga -sivuston uuden version lähtökohta oli tuottaa tyylikkäämpi, toimivampi ja paremmin hallittavissa oleva sivusto; tällä kertaa hyväksi todettuja ohjelmointikäytäntöjä käyttäen. Otin sivuston tuotannossa käyttöön erilaisia verkkokehityksen ohjelmistokehyksiä ja kirjastoja. Ohjelmistokehykset tekevät sivun koodista laadukkaampaa tarjoamalla rajapinnan huolellisesti tuotettuihin ohjelmistoratkaisuihin ja ohjelmointimalleihin. Ohjelmistokehykset tekevät myös ohjelmoijan työstä helpompaa. Koodia ei tarvitse tuottaa yhtä paljon, virheitä sattuu harvemmin ja ne ovat helpommin hallittavissa.

Palvelinpuolella valitsin käyttöön Symfony2-ohjelmistokehyksen. Se on PHP-ohjelmointikielellä toimiva ohjelmistokehys, joka tarjoaa ohjelmoijalle useita toiminnallisuuksia valmiina. Verkko-osoitteet voi reitittää polkumääritysten mukaan eri olioille, jotka hoitavat toiminnallisen logiikan erikseen tietorakenteista, kuten tietokannasta, ja esitettävistä näkymistä. Ohjelmoinnissa säästin paljon aikaa useiden muuten haastavien ratkaisujen kanssa hyödyntämällä Symfonyn sisäänrakennettuja toiminnallisuuksia.

Tietokannan käsittelyä helpottava Doctrine ORM tarjoaa mahdollisuuden käsitellä tietokannan dataa kirjoittamatta yhtään SQL-lauseketta itse. Vaikka monimutkaisia tietokantakyselyitä on mahdollista kirjoittaa itse, useimmissa tilanteissa riittää, kun lataa Doctrinen entiteettienhallintaolion, jolla voi etsiä haluamansa rivin tietokannan taulukosta. Taululle voi myös luoda uusia tietokantarivejä olioita käsittelemällä.

Koska Doctrine abstraktoi tietokannan käsittelyn olio-ohjelmointiin, minun ei ollut tarpeen koodatessa kirjoittaa yhtään SQL-kyselyä. Yhden kyselyn päätin kuitenkin kirjoittaa käsin, ja totesin, että sekin toimii Doctrinessa. Kun kyselyjä ei tarvitse kirjoittaa käsin, vaan ne luodaan ohjelmallisesti, on paljon vähemmän todennäköistä, että kyselyiden rakenteessa tapahtuu virheitä. Samalla kyselyyn syötetyt käyttäjältä tulleet arvot lisätään kyselyyn turvallisesti, eivätkä ne voi vaikuttaa lausekkeen rakenteeseen tai muuttaa sen tarkoitettua toiminnallisuutta. Tämä oli selkeä parannus sivuston alkuperäiseen versioon nähden.

Sivupohjien luominen Twigia käyttäen auttaa erottamaan ohjelmointilogiikan sisällön esittämisestä. Kun tieto on MVC-mallille uskollisesti haettu Doctrinen tarjoamista malleista ja käsitelty kontrolleripuolella valmiiksi, se siirretään Twigille esitettäväksi. Twig-tiedostoissa käyttäjä luo HTML-sivupohjan, johon muuttujat upotetaan ennalta määrätyille paikoille.

Twig helpottaa sivujen suunnittelua ja luomista, sillä siinä ei tarvitse ajatella ohjelman logiikkaa, vaan voi keskittyä pelkästään sivun asetteluun. Twig tarjoaa kuitenkin myös puitteet muutamille primitiivisille funktioille, joita asettelussa tarvitaan. PHP-koodia ei kuitenkaan tarvitse käyttää. Minulle ei tuottanut vaikeuksia toteuttaa sivurakenteen suunnittelua Twigin avulla. Twigin funktioita ja sen automaattista muuttujien muuttamista turvalliseen hyödyntämällä sain rakennettua sivuston näkymät tehokkaasti ja pelkäämättä tietoturvahuolia.

Erottamalla tietokannan logiikan ja näkymien logiikan erillisiksi osiksi pohjalla olevasta ohjelmointilogiikasta Symfony toteuttaa onnistuneesti Model-View-Controller-mallia. MVC-malli tarjoaa sovelluksen kehittäjälle mahdollisuudet skaalautuvuuteen ja helpottaa käytettyjen teknologioiden vaihtoa, kuten esimerkiksi siirtymistä yhdestä tietokantapalvelimesta toiseen. Kirjoitettu koodi voi pysyä samana, kun käytössä on abstraktoitu rajapinta.

Symfony eroaa useista muista suosituista MVC-pohjaisista PHP-ohjelmistokehyksistä siinä, että se käyttää manuaalista reititystä. Muissa kirjastoissa, kuten Zendissä ja CodeIgniterissa, on mahdollista määrittää omia reittejä, mutta ensisijaisesti reitit ohjaavat tiettyyn kontrolleriin hakemistorakenteen ja tiedostonimien perusteella. Symfonyssa minun piti määrittää jokainen reitti erikseen. Tämä ei kuitenkaan tuottanut ongelmia, vaan totesin lopulta sen olevan tehokas keino sivuston toiminnallisuuden hallinnassa. Ainoastaan itse kirjoittamani reittimääritykset toteuttaisivat ohjelmakoodia verkkosivustollani.

Ohjelmistokehysten yksi tehtävä on tehdä rutiinitoiminnoista helppoja tai automatisoituja, ja tässä Symfony on erittäin tehokas. Symfony tarjoaa useita eri toimintoja valmiiksi helppoon muotoon paketoituna, kuten käyttäjän sisäänkirjautumisen ja tilan ylläpidon sekä edellä mainitut tietokannan käsittelyn abstraktoinnin.

Symfonyssa on tarjolla laaja valikoima laajennusosia, joita nimitetään ”bundle”-käsitteellä. Käyttäjän omat sovelluksen koostavat projektin osat ovat myös bundle-paketteja. Verkon pakettitietokannoissa on eri tarkoituksiin useita erilaisia paketteja, joita käyttäjä voi liittää omaan projektiinsa rekisteröimällä ne osaksi sovellusta asetus-tiedostoissa. Bundle-laajennuksilla on helppo tuoda muiden kehittämiä sovelluksen osia, joilla voi säästää paljon aikaa ohjelmoinnissa.

Lähteet

1 Sharp, Josh. 2007. Why you should be using a framework. Verkkodokumentti. <http://joshsharp.com.au/blog/view/why_you_should_be_using_a_framework>. Luettu 9.2.2012.

2 McCallum, Q Ethan. 2004. Building a PHP Front Controller. Verkkodokumentti. <http://onlamp.com/ pub/a/php/2004/07/08/front_controller.html>. Luettu 15.5.2012.

3 Stump, Joe. 2005. Understanding MVC in PHP. Verkkodokumentti. <http://oreilly.com/pub/a/php/archive/mvc-intro.html>. Luettu 4.3.2012.

4 Atwood, Jeff. 2008. Understanding Model-View-Controller. Verkkodokumentti. <http://www.codinghorror.com/blog/2008/05/understanding-model-view-controller.html>. Luettu 24.3.2012.

5 Dalling, Tom. 2009. Model View Controller Explained. Verkkodokumentti. <http://tomdalling.com/blog/software-design/model-view-controller-explained/>. Luettu 24.3.2012.

6 The Book for Symfony 2.1. Verkkodokumentti. Sensio Labs. <http://symfony.com/doc/current/book/index.html>. Luettu 5.2.2012.

7 FilePermissionsACLs. Verkkodokumentti. Ubuntu Documentation. <https://help.ubuntu.com/community/FilePermissionsACLs>. 5.2.2012.

8 CodeIgniter User Guide. Verkkodokumentti. EllisLab. <http://codeigniter.com/user_guide/general/routing.html>. Luettu 17.6.2012.

9 Zend Framework User Guide. Verkkodokumentti. Zend Technologies. <http://framework.zend.com/manual/2.0/en/user-guide/routing-and-controllers.html>. Luettu 17.6.2012.

10 What is a cookie? Verkkodokumentti. All About Cookies. <http://www.allaboutcookies.org/cookies/>. Luettu 15.5.2012.

11 SQL Injection: What is it?. Verkkodokumentti. Acunetix. <http://www.acunetix.com/websitesecurity/sql-injection/>. Luettu 11.5.2012.

12 Gilfillan, Ian. 2006. Connecting and prepared statements with the mysqli extension. Verkkodokumentti. <http://www.databasejournal.com/features/mysql/article.php/3599166>. Luettu 11.5.2012.

13 SQL Injection. Verkkodokumentti. The Open Web Application Security Project. <https://www.owasp.org/index.php/SQL_Injection>. Luettu 11.5.2012.

14 Working with Objects. Verkkodokumentti. Doctrine Project. <http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/reference/working-with-objects.html>. Luettu 10.2.2012.

15 Twig Documentation. Verkkodokumentti. Sensio Labs. <http://twig.sensiolabs.org/documentation>. Luettu 20.2.2012.

16 Jquery Documentation. Verkkodokumentti. The jQuery Project. <http://docs.jquery.com/>. Luettu 15.2.2012.

17 Ward, Dave. 2012. 3 reasons why you should let Google host jQuery for you. Verkkodokumentti. <http://encosia.com/3-reasons-why-you-should-let-google-host-jquery-for-you/>. Luettu 10.11.2012.

18 Best Practices for Speeding Up Your Web Site. Verkkodokumentti. Yahoo. <http://developer.yahoo.com/performance/rules.html>. Luettu 10.11.2012.

19 Sellier, Alexsis. Less. Verkkodokumentti. <http://lesscss.org/>. Luettu 17.6.2012.

20 Smith, Nathan. 2008. 960 Grid System. Verkkodokumentti. <http://sonspring.com/journal/960-grid-system>. Luettu 16.6.2012.

21 Marcotte, Ethan. 2011. Responsive Web Design. New York, NY, Yhdysvallat: A Book Apart

22 Bootstrap. Verkkodokumentti. Twitter. <http://twitter.github.com/bootstrap/>. Luettu 11.2.2012.

23 Wroblewksi, Luke. 2011. Mobile First. New York, NY, Yhdysvallat: A Book Apart.

24 Shiflett, Chris. 2005. Essential PHP Security. Yhdysvallat: O'Reilly Media.

25 Guillaumier, Jacques. Cross Site Scripting – XSS – The Underestimated Exploit. Verkkodokumentti. <http://www.acunetix.com/websitesecurity/xss/>. Luettu 10.5.2012.

26 XSS Filter Evasion Cheat Sheet. Verkkodokumentti. The Open Web Application Security Project. <https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet>. Luettu 10.5.2012.

27 12 Column Grid. Verkkodokumentti. 960 Grid System. <http://960.gs/demo.html>. Luettu 12.2.2013.

Ote TeamController.php-tiedostosta

```
public function tradePlayerAction()
{
    // haluamme vain ajax-pyyntöjä
    if($request->isXmlHttpRequest() == false) {
        return new Response(
            json_encode(array(
                'success' => false,
                'debug' => 'Vain Ajax-pyynnöt ovat sallituja'
            ))
        );
    }

    $em = $this->getDoctrine()->getEntityManager();

    // haetaan pelaaja tietokannasta
    $pid = $request->request->getInt('pid');
    $pelaaja = $em->getRepository('OyveyNbaBundle:Pelaaja')
                  ->find($pid);

    // haetaan kohdejoukkue tietokannasta
    $jid = $request->request->getInt('team');
    $joukkue = $em->getRepository('OyveyNbaBundle:Joukkue')
                  ->find($jid);

    // varmistetaan, että pelaajaa muokkaava joukkue
    // omistaa pelaajan
    if($pelaaja->getTeam()->getJid() == $this
        ->get('security.context')->getToken()->getUser()
        ->getJid()) {
        // muutetaan pelaajan joukkue
        $pelaaja->setTeam($joukkue);
        $em->persist($pelaaja);
        $em->flush();

        return new Response(
            json_encode(array('success' => true))
        );

    } else {
        return new Response(
            json_encode(array(
                'success' => false,
                'debug' => 'Pelaaja ei ollut joukkueessasi.'
            ))
        );
    }
}
```