OULUN AMMATTIKORKEAKOULU

Toni Heikkala

RESTFUL-RAJAPINNAN KÄYTTÖ WORDPRESS-LISÄOSAN TOTEUTUKSESSA

RESTFUL-RAJAPINNAN KÄYTTÖ WORDPRESS-LISÄOSAN TOTEUTUKSESSA

Toni Heikkala
Opinnäytetyö
Syksy 2014
Tietojenkäsittelyn koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma, Internet-palvelut ja digitaalinen media

---

Tekijä(t): Toni Heikkala
Opinnäytetyön nimi: RESTful-rajapinnan käyttö WordPress-lisäosan toteutuksessa
Työn ohjaaja: Teppo Räisänen
Työn valmistumislukukausi- ja vuosi: Syksy 2014 Sivumäärä: 35

Tämän opinnäytetyön tarkoituksena oli kehittää lisäosa (plugin) WordPress-julkaisualustalle, joka lisää Tooltip Oy:n kehittämän FitnessBooker-verkkosovelluksen toiminnallisuuden asiakkaan WordPress-sivustolle. Lisäosan toiminnallisuus on jaettu WordPress-sivuston ja Tooltipin kehittämän FitnessBooker-ohjelmontirajapinnan välille. Lisäosan pääohjelmointikielenä oli PHP ja editorina toimi Sublime Text 2.

Teoriaosuudessa käydään läpi lyhyesti läpi WordPressin historiaa ja ominaisuuksia, lisäosan kehityksessä suuressa roolissa toimivia WordPressin ohjelmointirajapintoja, sekä REST arkkitehtuurityyliä. Raportin käytännön osuudessa käydään läpi lisäosan rakennetta, toimintaa, ja tarkastellaan miten lisäosan eri osiot kytkeytyvät toisiinsa. Erilaiset kuviot auttavat hahmottamaan koodin rakennetta ja lisäosan toimintaa.

Lisäosan ideana on mahdollistaa WordPress-sivuston teeman ostaminen kolmannelta osapuolelta, minkä jälkeen FitnessBooker voidaan integroida osaksi sivustoa vaivatta. Lisäosan avulla sivuston käyttäjä voi muun muassa rekisteröityä ja kirjautua FitnessBooker-palveluun, selata tulevia ja menneitä liikuntatapahtumia sekä tehdä ja perua varauksia tapahtumiin.

Työn tuloksena syntyi WordPress-lisäosa, joka saadaan vaivatta käyttöön asiakkaan WordPress-sivustolle muutamalla klikkauksella ja tekstikentän täytöllä. Lisäosan avulla palvelun käyttäjä pääsee käsiksi FitnessBookerin tärkeimpiin toimintoihin.

Asiasanat: PHP, WordPress, API, REST, HTTP, plugin

ABSTRACT

Oulu University of Applied Sciences
Degree programme of Business Information Systems, Internet Services and Digital Media

---

Author(s): Toni Heikkala
Title of thesis: Creating a WordPress plugin using a RESTful application programming interface
Supervisor(s): Teppo Räisänen
Term and year when the thesis was submitted: Autumn 2014 Number of pages: 35

The purpose of this thesis was to develop a WordPress plugin that adds the functionality of FitnessBooker web-application developed by Tooltip Oy to client's WordPress website. The functionality of the plugin is split between the WordPress website and Tooltip's FitnessBooker application programming interface. The main programming language of the plugin was PHP and the editor used was Sublime Text 2.

In the theory section of this thesis is a quick overview of WordPress' history and features, the WordPress application programming interface and REST architecture style. The practical section goes through the plugin's structure, functionality and goes in-depth about how different parts of the plugin code come together. Several figures assist the reader in perceiving the structure of the program code and the functionality of the plugin.

The idea of the plugin is to give Tooltip the possibility of purchasing WordPress site theme from a third party, after which FitnessBooker can be integrated to the website with ease. With the plugin an user can for example registrate and login to FitnessBooker service, browse upcoming and past fitness events and also book and unbook seats to these events.

The result of the thesis was a WordPress plugin that can be added to client website with a few clicks and editing a text field. With the plugin user gains an access to all the most important features of FitnessBooker application.

Keywords: PHP, WordPress, API, REST, HTTP, plugin

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheena on FitnessBooker WordPress-lisäosa. Työn toimeksiantaja on Tooltip Oy, jonka käyttöön lisäosa kehitetään. Työn kehittämistehtävän päämääränä on luoda WordPress-lisäosa pääasiassa PHP-ohjelmointikielellä. Työn kehityksessä ovat oleellisena osana sekä Tooltipin FitnessBooker-ohjelkointirajapinta että WordPressin omat ohjelmointirajapinnat. Opinnäytetyön tavoitteena on kehittää omia ohjelmointitaitoja ja oppia kehittämään laatuvaatimukset täyttävä WordPress-lisäosa, jota alan yritys voi hyvillä mielin tarjota asiakkailleen. Raportin tietoperustassa kerrotaan lyhyesti WordPressistä ja sen historiasta, avataan REST-käsitettä, sekä syvennytään tarkemmin julkaisualustan ominaisuuksiin kuten sen tarjoamiin lukuisiin ohjelmointirajapintoihin.

FitnessBooker eli FiBo on Tooltip Oy:n kehittämä toiminnanohjausjärjestelmä liikuntakeskuksille ja kuntosaleille, jonka tarkoituksena on helpottaa yrittäjän arkea. FitnessBooker ohjaa liikuntakeskuksen toimintaa ja tarjoaa yrittäjälle reaaliaikaista tietoa keskuksen tapahtumista sekä raportit liiketoiminnan seurantaa varten. FitnessBookerin avulla yrittäjä voi hallinnoida asiakasrekisteriä, raportointia, sosiaalisen median näkyvyyttä, verkkosivujaan, laskutusta, kulunvalvontaa ja ryhmäliikuntatunteja. Tässä työssä tuotettu lisäosa integroi WordPress-sivustolle FitnessBookerin verkkopalvelun tietyt ominaisuudet, jotka ovat uuden asiakkaan rekisteröityminen, sisään- ja uloskirjautuminen, ryhmäliikuntakalenteri ja –tapahtumat sekä ryhmäliikuntavarauksien tekeminen ja peruminen.

Raportin keskeisin aihe on FitnessBooker-lisäosan toteutus ja siihen liittyvissä kappaleissa käydään yksityiskohtaisesti läpi miksi ja miten jokainen lisäosan toiminto on toteutettu. Lisäosan toiminta on jaoteltu osioihin kuten kirjautumiseen, käyttäjätietojen esittämiseen ja varauksien tekemiseen. Jokaisessa osiossa käsitellään kyseisen toiminnon toteuttamiseen käytettyjä funktioita ja rajapintoja. Toteutusosiossa syvennytään erityisesti FitnessBooker-rajapinnan tarjoamiin resursseihin ja niiden kanssa kommunikointiin käytettyihin HTTP-pyyntöihin.

2 WORDPRESS

2.1 Mikä on WordPress?

WordPress on ilmainen, PHP-ohjelmointikielen ja MySQL-tietokannan avulla kehitetty avoimen lähdekoodin julkaisualusta, joka sai alkunsa vuonna 2003 ja on tähän päivään mennessä kasvanut maailman suurimmaksi omalla palvelimella hostattavaksi bloggaus-työkaluksi. Vaikka WordPress alkoikin juuri bloggaus-järjestelmänä, se on kehittynyt käytettäväksi muun muassa julkaisujärjestelmänä, ja muita käyttötarkoituksia on lähes rajattomasti tuhansien lisäosien ja pienohjelmien (widget) tarjotessa erilaisia toimintoja järjestelmään. (WordPress 2014, Viitattu 30.10.2014)

2.2 WordPressin ominaisuudet

WordPress tarjoaa useita erilaisia työkaluja bloggaukseen ja verkkosivustojen sekä –sovellusten kehitykseen. Seuraavaksi käsitellään muutamia, tämänkin työn toteutuksessa hyödynnettyjä, WordPressistä asennuksen jälkeen valmiiksi löytyviä ominaisuuksia. (WordPress 2014, Viitattu 30.10.2014)

2.2.1 Teemat

WordPress-teemat ovat tiedostoja, jotka yhdessä luovat sivuston ulkoasun sekä toiminnallisuuden. Teemoja kehitetään yleisimmin omiin tarpeisiin, asiakasprojekteihin tai lähetettäväksi WordPressin yleiseen teemakirjastoon. Teemat erottavat sivuston ulkoasun ja sisällön WordPressin järjestelmätiedostoista, mikä mahdollistaa sivuston päivityksen ilman näkyviä muutoksia. Vaikka FitnessBooker-sovelluksen toiminnallisuuden olisi voinut rakentaa suoraan johonkin yksittäiseen teemaan, lisäosan kehittäminen mahdollistaa FitnessBookerin käyttöönottamisen mille tahansa WordPress-sivustolle teemasta riippumatta. (WordPress Codex 2014, Viitattu 31.10.2014)

2.2.2 Lisäosat

Työn toteutuksen mahdollisti WordPressin lisäosajärjestelmä, jonka avulla voidaan kehittää erilaisia lisäosia laajentamaan WordPressin toiminnallisuutta. Kuten aiemmassa kappaleessa todettiin, lisäosat voidaan ottaa käyttöön mille tahansa WordPress-sivustolle jolloin ne lisäävät tarjoamansa toiminnallisuuden kyseiselle sivustolle.

2.2.3 Ohjelmointirajapinta

WordPress tarjoaa ohjelmistokehittäjille kattavan ohjelmointirajapinnan, joka voidaan jakaa useisiin eri rajapintoihin. Jokainen rajapinta sisältää omat funktionsa tuovat jonkin tietyn toiminnallisuuden sovellukseen. Yhdessä nämä rajapinnat muodostavat niin kutsutun WordPress-rajapinnan. Tämän työn teossa käytettiin HTTP-, Options-, Settings-, Rewrite-, Shortcode-, Database- sekä tietysti Plugin-rajapintoja, joihin syvennytään myöhemmin tässä raportissa. (WordPress Codex 2014, Viitattu 31.10.2014)

2.2.4 Skriptikirjastot

WordPress tarjoaa oletuksena verkkosivustojen- ja sovellusten kehitykseen oleellisena osana kuuluvia skriptikirjastoja. Muutamia näistä ovat Plupload, Underscore.js, Backbone.js sekä laajasti käytetty JavaScript-kirjasto jQuery jota on hyödynnetty tässäkin työssä. jQuerya käytettiin lisäosassa käyttäjän suorittamien toimintojen tapahtumaviestien näyttämiseen, lomakkeiden validointiin jQueryn Validation-lisäosan avulla sekä salasanan vaihtamislomakkeen näyttämiseen ja piilottamiseen. (WordPress Codex 2014, Viitattu 31.10.2014)

3 TYÖSSÄ KÄYTETYT WORDPRESSIN RAJAPINNAT

3.1 HTTP API

HTTP-rajapinta lisättiin WordPressiin versiossa 2.7 ja sitä laajennettiin versiossa 2.8. PHP-ohjelmointikielessä on lukuisia tapoja lähettää HTTP-pyyntöjä. Ongelmia voi ilmetä kun jotkin web-palvelimet tukevat vain joitakin näistä tavoista, tai pahimmassa tapauksessa eivät tue yhtäkään. WordPressin HTTP-rajapinnan tarkoituksena on tukea mahdollisimman montaa näistä eri tavoista yksinkertaisella rajapinnalla, jota voidaan standardina käyttää sekä lisäosien että WordPressin ydintoimintojen kehityksessä. (WordPress Codex 2014, Viitattu 4.11.2014)

3.2 Options API

WordPressin options-rajapinta tarjoaa yksinkertaisen ja standardoidun tavan säilöä dataa WordPressin tietokantaan. Rajapinta tekee helpoksi muokattavien vaihtoehtojen (options) luomisen, päivittämisen ja poistamisen. Rajapinnan avulla tallennettava data on tallennettuna uniikeilla nimillä wp_options-nimisen tietokantataulun sisälle, josta sitä voidaan tarvittaessa käyttää. (WordPress Codex 2014, Viitattu 4.11.2014)

3.3 Settings API

WordPressin versiossa 2.7 lisätty asetukset-rajapinta mahdollistaa asetussivujen luomisen ja hallitsemisen WordPress-hallintasivulla. Asetukset-rajapintaa käytetään tässä työssä yhdessä options-rajapinnan kanssa. Yhdessä nämä rajapinnat antavat sivustolle hallintaoikeudet saaneelle käyttäjälle mahdollisuuden muokata joitakin lisäosan toimintoja, kuten esimerkiksi seuraavat tapahtumat –listassa näytettävien tapahtumien lukumäärää. (WordPress Codex 2014, Viitattu 4.11.2014)

3.4 Rewrite API

Uudelleenkirjoitus-rajapinta antaa teema- ja lisäosa-kehittäjille mahdollisuuden määrittää uusia, mukautettuja URL-uudelleenkirjoitussääntöjä. Rajapintaa on käytetty työssä Tapahtuma-sivujen

URL:ien siistimiseen. Esimerkiksi selaimen osoiterivillä näkyvä URL http://toniheikkala.fi/wordpress/ryhmaliikunta/0/6021/Jooga/ on ennen uudelleenkirjoittamista muodossa http://toniheikkala.fi/wordpress/?page_id=3&calendar_id=0&event_id=6021. URL:ien siistiminen on tärkeä osa verkkosivuston hakukoneoptimointia ja käytettävyyden parantamista. (WordPress Codex 2014, Viitattu 4.11.2014)

3.5 Shortcode API

WordPressin 2.5-versiossa lisätty shortcode-rajapinta mahdollistaa esimerkiksi ohjelmakoodia sisältävien, niin kutsuttujen makrokoodien käyttämiseen sivujen tai artikkeleiden sisällössä. Shortcodeja on työssä käytetty sisällön ja toiminnallisuuden lisäämiseen lisäosan luomille sivuille. Shortcodejen toiminnasta kerrotaan tässä raportissa työn toteutusta käsittelevässä osiossa. (WordPress Codex 2014, Viitattu 4.11.2014)

3.6 Database API

WordPressin tietokantarajapinta voidaan katsoa koostuvaksi kolmesta eri tietokantaan liittyvästä rajapinnasta, jotka ovat Options API, Transients API ja Metadata API. Yhdessä nämä rajapinnat muodostavat koko WordPress-projektin kattavan käyttöliittymän, joka mahdollistaa tietokantaan tallennetun datan käsittelemisen lisäosa- ja teemakehityksessä. (WordPress Codex 2014, Viitattu 4.11.2014)

3.7 Plugin API

Lisäosat perustuvat koukkuihin (hooks) joilla lisäosa ”kiinnittyy” muuhun WordPressiin, eli kutsuvat funktioita määritettyyn aikaan ja täten saavat lisäosan toiminaan. Koukkuja on kahdenlaisia: toiminta- (action) ja suodatinkoukkuja (filter). Vaikka kummallakin näistä on omat funktionsa, joskus saman lopputuloksen voi saavuttaa koukkutyypistä riippumatta. (WordPress Codex 2014, Viitattu 4.11.2014)

3.7.1 Toimintakoukut

Toimintakoukut käynnistyvät yhdessä tiettyjen WordPress-tapahtumien kuten artikkelin julkaisun, teeman vaihdon tai sivun latauksen yhteydessä. Yleisimmin toimintakoukut suorittavat jonkin seuraavista toiminnoista:

1. Muokkaavat tietokantadataa.
2. Lähettävät sähköpostiviestin.
3. Muokkaavat teeman tai lisäosan tuottamaa selaimessa näkyvää sivua.

Työssä on eniten käytetty init-koukkua, joka ajetaan heti WordPress-sovelluksen sivulle latautumisen jälkeen, eli ennen kuin mitään sivuston näkyvää sisältöä on ladattu selaimen näytettäväksi. Tämän koukun avulla ajettavia funktioita ovat esimerkiksi PHP-sessionin aloittava ja sessiomuuttujiin halutut arvot lisäävä funktio (fibo_session), shotcodet rekisteröivä funktio (register_shortcodes), sekä halutut lisäosan luomat sivut WordPressin valikosta piilottava fibo_filter_menu. (WordPress Codex 2014, Viitattu 6.11.2014)

Muita työssä käytettyjä koukkuja ovat muun muassa wp_footer, wp_enqueue_scripts, admin_enqueue_scripts, admin_menu sekä admin_init. Wp_footer-koukkua käytetään teemaan HTML-sisältöä tulostavien funktioiden kanssa. Wp_enqueue_scripts-koukun avulla lisätään WordPressiin lisäosan oma CSS-tyylitiedosto ja JavaScript-tiedosto, ja admin_enqueue_scripts lisää samaan tapaan CSS-tiedoston WordPressin hallintasivuille. Admin_menu lisää WordPressin hallinnan valikkoon lisäosan asetussivun ja admin_init:illä tälle lisätylle sivulle tulostetaan tarvittavat lomakkeet ja muu sisältö. (WordPress Codex 2014, Viitattu 4.11.2014)

3.7.2 Suodatinkoukut

Suodattimet ovat funktioita joiden kautta WordPress kuljettaa dataa tietyissä pisteissä juuri ennen tämän datan käsittelemistä esimerkiksi lisäämällä sen tietokantaan tai lähettämällä sen selaimen näytettäväksi. Suodatinkoukut ovat tietokannan ja selaimen välissä esimerkiksi WordPressin luodessa näytettäviä sivuja, ja selaimen ja tietokannan välissä esimerkiksi WordPressin lisätessä uusia artikkeleita ja kommentteja tietokantaan; lähes kaikki WordPressiin syötettävä ja sen tuottama data kulkee ainakin yhden suodattimen läpi. WordPress tekee osan suodatuksesta oletusarvoisesti, ja lisäosat voivat lisäksi lisätä omia suodattimiaan. Työn lisäosassa on käytetty kahta eri suodatinkoukkua: wp_list_pages_excludes ja the_title. Kumpaakin näistä suodatinkoukuista

käytetään yhdessä aiemmassa kappaleessa mainitun init-toimintakoukun kanssa, init-koukun ajaman funktion sisällä. Wp_list_pages_excludes kirjaimellisesti suodattaa halutut sivut WordPressin sivuvalikosta. The_title-suodatinkoukkua on käytetty hakemaan Ryhmäliikunta-sivulla esitettävän tapahtuman tyyppi ja esittämään se sivun otsikkona. (WordPress Codex 2014, Viitattu 4.11.2014)

4 REST

REST on HTTP-protokollaan perustuva arkkitehtuurimalli web-sovellusten ja ohjelmointirajapintojen toteuttamiseen. Lyhenne REST tulee englannin kielen sanoista Representational State Transfer, ja sitä käytti ensimmäisenä vuonna 2000 Roy Fielding tohtorin tutkinnon väitöskirjassaan Architectural Styles and the Design of Network-based Software Architectures. Termillä kuvataan verkottuneiden sovelluksien suunnittelun arkkitehtuurityyliä. (Costello, Viitattu 25.11.2014; Fielding, Viitattu 25.11.2014)

Web koostuu resursseista, jotka ovat URL:ien (uniform resource locator) kohteita. Esimerkiksi kuvitteellisilla Example.com-verkkosivuilta voi olla resurssi, johon asiakasohjelmat eli verkkoselaimet pääsevät käsiksi URL:illa http://www.example.com/items/item1/. Kyseinen URL palauttaa resurssin representaation (representation), joka voisi olla esimerkiksi item1.html. Tämä representaatio asettaa asiakasohjelman tilaan (state). Item1.html:n representaatiossa toiselle sivulle johtavan linkin avaus asettaa asiakasohjelman jälleen uuteen tilaan, eli asiakasohjelman tila muuttuu (transfer) jokaisen resurssin representaation välillä. (Costello, Viitattu 25.11.2014)

REST-arkkitehtuurityylin mukaisesti toteutettuja sovelluksia kutsutaan RESTful-sovelluksiksi. RESTful-sovellukset ja –rajapinnat käyttävät HTTP-pyyntöjä datan lähettämiseen, lukemiseen, ja poistamiseen. REST on kevyt vaihtoehto aiemmin käytetyille mekanismeille ja web-palveluille, joita ovat esimerkiksi RPC, SOAP ja WSDL. Yksinkertaisuudestaan huolimatta REST sisältää samat ominaisuudet kuin nämä raskaammat menetelmät. (Elkstein, Viitattu 27.11.2014)

5 FITNESSBOOKER-LISÄOSA

5.1 Lähtokohta, tavoite ja suunnittelu

Toimeksianto WordPress-lisäosan kehittämiseen tuli Tooltip Oy:lta otettuani heihin yhteyttä opinnäytetyön aihetta miettiessäni. Tavoitteeksi määriteltiin lisäosa, joka tuo FitnessBookerin toiminnallisuuden asiakkaan WordPress-sivustolle. Lisäosan avulla käyttäjän piti olla kykenevä rekisteröitymään ja kirjautumaan FitnessBooker-palveluun, näkemään ja päivittämään omia tietojaan, selaamaan ryhmäliikuntakalenteria ja yksittäisiä tunteja, sekä tekemään ja perumaan varauksia ryhmäliikuntatunneille. Puhetta oli myös verkkokaupan toteutuksesta lisäosaan, mutta se jäi pois kun huomattiin että työtä on perustoiminnoissa riittävästi. Lisäosan ominaisuuksien tuli myös olla muokattavissa sivustolta. Tavattuani toimeksiantajat palaverissa sain heiltä tehtävälistan, jossa oli listattu lisäosaan toteutettavat toiminnallisuudet, sekä lisäosalle haluttua toiminnallisuutta avaavan use casen.

5.1.1 FitnessBooker-rajapinta

Tooltip oli aloittanut RESTful-arkkitehtuurityyliä noudattavan FitnessBooker-ohjelmointirajapinnan kehityksen ja se jatkui heidän päässään aina lisäosan valmistumiseen asti. Sain toimeksiantajalta rajapinnan dokumentoinnin, jossa oli kuvattu rajapinnan resurssit sekä ohjeet niiden käyttöön. (Kuvio 1) Resurssit palauttavat erilaisia arvoja käytetyn HTTP-protokollan verbin mukaan. Esimerkiksi oheisessa ruudunkaappauksessa on dokumentoitu mitä /calendars/[id]/events/[id]/bookings –resurssi palauttaa, kun sitä käytetään GET-verbillä. Kyseinen resurssi palauttaa tietyn kalenterin tietyn tapahtuman kaikki varaukset ja niiden arvot JSON-formaatissa, ja nämä arvot voidaan lisäosan avulla esittää WordPress-sivustolla.

GET /calendars/[id]/events/[id]/bookings

Tapahtuman varaukset

Parametrit

Parametri	Oletus	Formaatti	Kuvaus

Palautusarvot

Arvo	Kuvaus
booking_id	Varauksen id, `booking id`
event_id	Tapahtuman id, `event id`
calendar_id	Kalenterin id, `calendar id`
present	Onko varaus lunastettu, boolean
event_type	Tapahtuman tyypin nimi

Esimerkki

```
GET /calendars/1/events/5783/bookings
```

Vastaus:

```
[
  {
    "booking_id": 2890,
    "event_id": 5783,
    "calendar_id": 0,
    "present": false,
    "event_type": "Spinning",
    "event_begin": "2014-10-03T20:00:00+03:00"
  },
  {
    "booking_id": 2889,
    "event_id": 5783,
    "calendar_id": 0,
    "present": false,
    "event_type": "Spinning",
    "event_begin": "2014-10-03T20:00:00+03:00"
  }
```

KUVIO 1. FitnessBooker-rajapinnan dokumentaatio – /calendars/[id]/events[id]/bookings-resurssi

Huomatessani rajapinnassa silloin tällöin puutteita otin sähköpostilla yhteyttä Tooltipin Jari Ylimäiseen, joka teki tarvittavat lisäykset tai korjaukset. Jari myös auttoi ohjelmointityössä aina kun vastaan tuli ongelma johon en löytänyt ratkaisua kohtuullisessa ajassa tai kun en ollut varma

käyttämästäni menetelmästä toteuttaa jokin lisäosan ominaisuus. (Ylimäinen, J. 2014, Viitattu 31.10.2014)

5.2 Lisäosan toteutus

Koska WordPress-lisäosien kehitys oli minulle uusi asia, aloitin työn tutustumalla WordPressin Codexin lisäosien kehitykseen liittyvää materiaalia keskittyen erityisesti WordPressin HTTP-rajapinnan käyttöön lisäosakehityksessä.

5.2.1 Aktivointi, deaktivointi

WordPress-lisäosa otetaan käyttöön ensiksi lataamalla se palvelimelle WordPress-asennuksen plugins-kansioon ja tämä jälkeen aktivoimalla sen WordPressin lisäosien hallintasivulla. Aktivoinnissa lisäosa käynnistyy eli sen koodi otetaan käyttöön WordPressin ydintoimintojen rinnalle. Aktivoinnin yhteydessä voidaan ajaa haluttuja funktioita rekisteröimällä ne käyttäen WordPressin register_activation_hook-funktiota. Rekisteröity funktio ajetaan jokaisen hallintasivulla suoritetun lisäosan aktivoinnin yhteydessä. Tämän työn lisäosan aktivoinnin yhteydessä ajetaan funktio, joka määrittää lisäosan luomien sivujen sisällön ja lisää ne WordPressin tietokantaan. Lisäksi aktivointifunktio lisää URL-uudelleenkirjoitussäännön jota käytetään myöhemmin tässä raportissa käsiteltävien Tapahtuma-sivujen URL:ien muodostuksessa. Register_activation_hook-funktion vastakohta on register_deactivation_hook jolla rekisteröity funktio ajetaan kun lisäosa poistetaan käytöstä. Tämän lisäosan deaktivoinnista vastaava funktio etsii aktivoinnissa luodut sivut WordPressin tietokannasta ja poistaa ne. Tämä toimenpide on erittäin tärkeä, sillä useiden aktivointien jälkeen luodut samannimiset sivut aiheuttaisivat ongelmia lisäosan toiminnassa.

5.2.2 Tyylit, skriptit

Kuten verkkosivujen toteutuksessa myös WordPress-lisäosien kehityksessä tarvitaan usein CSS-tyylitiedostoja ja JavaScript-skriptejä. Tässä työssä lisäosan omalla tyylitiedostolla muotoillaan lisäosan tuottamien HTML-elementtien ulkonäkö selaimessa. JavaScriptin jQuery-kirjaston avulla suoritetaan raportissa jo aiemmin mainitut toimenpiteet, jotka ovat rekisteröitymis- ja asiakastietolomakkeiden validointi, salasanan vaihtolomakkeen näyttäminen ja piilotus, sekä virhe- ja onnis-

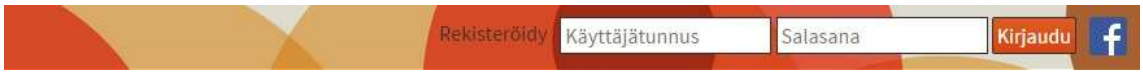
tumisilmoitusten näyttö yläpalkissa. Koska WordPress luo automaattisesti HTML:än <head>-elementin johon tyyli- ja skriptitiedostot yleensä sisällytetään, tulee lisäosan omat tiedostot lisätä käyttäen WordPressin koukkuja. Wp_enqueue_scripts-toimintakoukulla ajetaan funktio, jossa wp_register_style- ja wp_register_script-funktioilla rekisteröidään normaalin käyttäjän näkemän sivun tyylitiedosto, lisäosan oma jQuery-skriptiä sisältävä JavaScript-tiedosto, ja jQuery Validation-lisäosa. Admin_enqueue_scripts-koukulla ajetaan vastaavasti funktio, joka rekisteröi WordPressin hallintapuolella käytössä oleva CSS-tyylitiedosto. Kummankin funktion sisältä löytyy rekisteröintifunktioiden lisäksi wp_enqueue_style- ja wp_enqueue_script-funktiot, jotka lisäävät rekisteröidyt tyylit ja skriptit sivulle (Kuvio 2).

```
// Include admin CSS file
function enqueue_fibo_admin_styles() {
    wp_register_style(
        'fibo-admin-styles',
        plugins_url('css/fibo-admin-style.css', __FILE__));

    wp_enqueue_style('fibo-admin-styles');
}
add_action('admin_enqueue_scripts', 'enqueue_fibo_admin_styles');

// Include front-end CSS, Javascript and jQuery
function enqueue_fibo_scripts() {

    wp_register_style(
        'fibo-styles',
        plugins_url('css/fibo-style.css', __FILE__));

    // Register jQuery validation
    wp_register_script(
        'jquery-validation-plugin',
        plugins_url('js/jquery.validate.min.js', __FILE__),
        array('jquery'));

    // Register general scripts
    wp_register_script(
        'fibo-scripts',
        plugins_url('js/fitnessbooker.js', __FILE__),
         array('jquery'));

    wp_enqueue_style('fibo-styles');
    wp_enqueue_script('jquery-validation-plugin');
    wp_enqueue_script('fibo-scripts');
}
add_action('wp_enqueue_scripts', 'enqueue_fibo_scripts');
```

KUVIO 2. Tyyli- ja JavaScript-tiedostot rekisteröivät ja käyttöönottavat funktiot

5.2.3 Yläpalkki

Useimpien lisäosan toimintojen aloituspisteenä on lisäosan aktivoinnin jälkeen sivuston yläreunaan ilmestyvä palkki (Kuvio 3), josta löytyy käyttäjän ollessa sisäänkirjautuneena linkki asiakastietoihin ja uloskirjautumiseen, ja muussa tapauksessa linkki rekisteröitymissivulle, kirjautumislomake, ja Facebook-nappi josta käyttäjä joko rekisteröityy tai kirjautuu palveluun. Kiinteiden elementtien lisäksi yläpalkissa esitetään käyttäjää informoivat virhe- ja onnistumisviestit, jotka näytetään väliaikaisesti jQueryn avulla. Vaikka yläpalkki sijaitseekin sivun yläosassa, se tulostetaan sivulle wp_footer-toimintakoukun avulla HTML-dokumentin alaosaan ennen </body>-tagia, ja se sijoitetaan CSS-tyyleillä selaimen yläosaan.



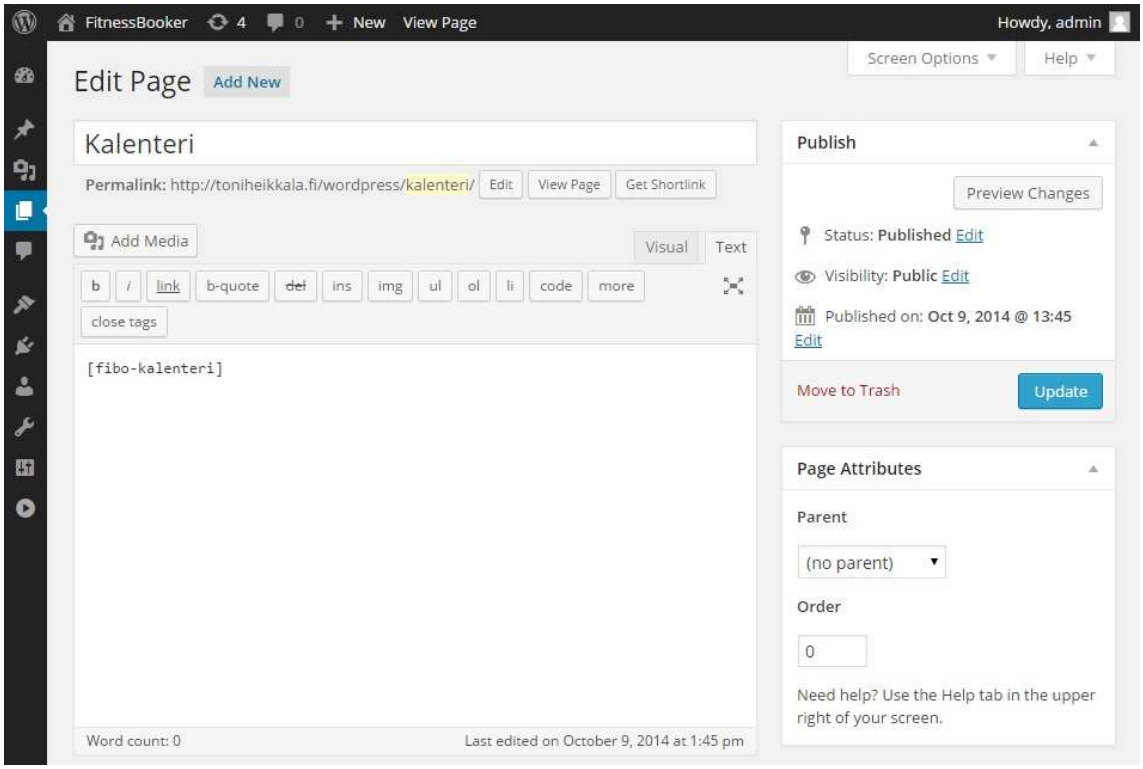
KUVIO 3. Yläpalkki ennen sisäänkirjautumista

5.2.4 Sivut ja shortcodet

Seuraavaksi lähdin kehittämään lisäosaa toimeksiantajan suosittelemassa järjestyksessä aloittamalla asiakastietojen esityksestä. Tätä varten piti kehittää kirjautuminen, sivu asiakastiedoille ja mahdollisuus muokata näitä tietoja. Lisäosan aktivoinnin yhteydessä ajettavat wp_insert_post-funktiot lisäävät WordPressin tietokantaan lisäosan tarvitsemat sivut joita ovat Kalenteri, Omat tiedot, Rekisteröityminen ja Ryhmäliikunta. Sivujen sisältö ja sisältöä tuottava ohjelmakoodi lisätään jokaiselle sivulle WordPressin Shortcode-rajapintaa käyttäen. WordPressin artikkeleiden tai sivujen sisältöön voidaan lisätä monimutkaista sisältöä kuten PHP-koodia shortcodeja käyttämällä. Shortcodet ovat sivun sisältöeditorissa muodossa ”[shortcode]”, jossa hakasulkujen sisällä oleva teksti on tietylle shortcodelle määritelty tunniste. (Leiniö, T. 2013, Viitattu 31.10.2014)

Shortcodeja käytetään lisäosassa siten, että oletuksena tyhjänä luodulle sivulle annetaan sisällöksi sille tarkoitettu shortcode. Esimerkiksi Kalenteri-sivun WordPressin hallintasivulla nähtävä tekstisisältö on vain ”[fibo-kalenteri]”, mutta tämä hakasuluissa oleva shortcode tuo sisällöksi calendar.php-tiedoston, joka sisältää kyseisellä sivulla näkyvän tekstisisällön ja toiminnallisuuden. Lisäosan sivujen sisällön tuottamisen lisäksi shortcodea käytetään työssä myös hieman tavanomaisemmin Tulevat tapahtumat –listan lisäämiseen halutulle sivulle. Lisäämällä WordPress-sivun tai artikkelin sisältöeditoriin (Kuvio 4) koodin ”[fibo-tulevat-tapahtumat id="0"]”, jossa id on

halutun kalenterin id-luku, näytetään lisäosan asetuksissa määritelty lukumäärä tapahtumia kyseisestä kalenterista, sekä ”Näytä kaikki”-linkki, joka ohjaa Kalenteri-sivulle.



KUVIO 4. Kalenteri-sivun ohjelmakoodin lisäys WordPress-sivulle sisältöeditorin kautta

5.2.5 HTTP-rajapinta, sessiot ja kirjautuminen

Kun oli selvillä miten koodilla tuotettua sisältöä saadaan näkymään lisäosan luomilla WordPress-sivuilla ja seuraavana tehtävälistalla oli sisään- ja uloskirjautuminen, piti saada yhteys kirjautumisen mahdollistavaan FitnessBooker-rajapintaan. Kuten aiemmin mainittu, rajapinnan kanssa keskustellaan HTTP-protokollan verbejä käyttäen. Vaikka WordPress tarjoaakin oman HTTP-rajapintansa, minun piti sisään- ja uloskirjautumisen toteuttamiseksi käyttää cURL-projektin libcurl-kirjastoa, sillä WordPressin HTTP-rajapinta toimii vain WordPressin ”sisällä” olevassa ohjelmakoodissa (Kuvio 5) ja kirjautumistiedostot eivät ole suoraan yhteydessä WordPressiin toisin kuin sivujen sisällön tuottavat tiedostot. Sisäänkirjautuminen toimii siten, että lisäosan sivustolle luomaan kirjautumis-lomakkeeseen syötetty käyttäjänimenä toimiva sähköpostiosoite ja käyttäjän salasana lähetetään login.php-tiedostoon, jossa nämä tiedot lähetetään FitnessBooker-rajapinnan /auth/login/-resurssille HTTP:n POST-verbiä käyttäen. Mikäli rajapinta löytää käyttäjänimi ja salasana –parin, se palauttaa jokaisen onnistuneen kirjautumisen yhteydessä luotavan

poletin (token), jota tarvitaan joidenkin rajapinnan resurssien näyttämiseen. Poletti tallennetaan PHP sessiomuuttujaan, jossa se säilyy käytettävissä lisäosan toiminnassa kunnes poletti vanhentuessaan tuhoutuu tai käyttäjä kirjautuu ulos. Uloskirjautuminen tapahtuu antamalla FitnessBooker-rajapinnan /auth/logout/-resurssille base64-enkoodattu kirjautumispoletti, jolloin kyseinen poletti tuhoutuu ja käyttäjä kirjataan ulos palvelusta.

```php
// Username and password from login form
$uname = $_POST["fibo-login-email"];
$pass = $_POST["fibo-login-password"];

// API url from plugin settings
$api_url = $_SESSION['api_url'];

$ch = curl_init($api_url . '/auth/login');

curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_POSTFIELDS,
            'username=' . $uname . '&password=' . $pass);

$result = curl_exec ($ch);
$decoded = json_decode($result, true);

// Login succeeded
if( $decoded["success"] === true ) {
    // Token from result
    $token = $decoded["token"];
    // Token into session variable
    $_SESSION['token'] = $token;
    // Login success into session variable. Used to show login status message
    $_SESSION['message'] = 'login_success';
}
// Login failed
else {
    // Login fail into session variable. Used to show login status message
    $_SESSION['message'] = 'login_fail';
}

curl_close ($ch);
```

KUVIO 5. Sisäänkirjautuminen käyttäen sähköpostiosoitetta ja salasanaa

Ohjelmointityön loppuvaiheessa lisättiin lisäosaan käyttäjälle mahdollisuuden kirjautumiseen Facebook-tilinsä avulla. Facebook-autentikointi tapahtuu ohjaamalla käyttäjä osoitteeseen, jossa joko Tooltipin facebook-register-api.php- tai facebook-login-api.php-tiedosto hoitaa autentikoinnin ja palauttaa poletin WordPress-lisäosalle. Mikäli käyttäjä kirjautuu Facebook-tunnuksillaan ensimmäistä kertaa palveluun, hänet ohjataan rekisteröitymään, tai jos rekisteröityminen on jo kertaalleen suoritettu, käyttäjä ohjataan kirjautumiseen. Kummassakin tapauksessa käyttäjää pyydetään kirjautumaan tiedoillaan Facebookiin, jonka jälkeen rajapinta palauttaa käyttäjän poletin kanssa WordPress-sivustolle, rekisteröitymistilanteessa kehottaen täyttämään Facebookin antamilla tiedoilla osalta valmiiksi täytetty rekisteröitymislomake. Jokaisen sisään- tai uloskirjautumi-

sen jälkeen, kirjautumistavasta riippumatta, selain uudelleenohjataan PHP:n header-funktiota käyttäen WordPress-sivustolle joko etusivulle, tai mikäli mahdollista, edelliselle vieraillulle sivulle, ja käyttäjälle näytetään sessiomuuttujaan tallennetun viestin perusteella tieto onnistuiko kirjautuminen (Kuvio 6).

KUVIO 6. Käyttäjä kirjautuu lisäosan kautta onnistuneesti

5.2.6 Käyttäjätiedot

Kirjautumisen ja uloskirjautumisen jälkeen vuorossa oli palveluun onnistuneesti kirjautuneen käyttäjän tietojen tulostus tämän henkilökohtaisella Omat tiedot-sivulla. Sivulla esitettävät tiedot ovat seuraavat:

- Etu- ja sukunimi
- Asiakasnumero
- Jäljellä ovat kuntosalikäyntikerrat
- Asiakkuuden alkamis- ja loppumispäivämäärät
- Sähköpostiosoite
- Puhelinnumero
- Katuosoite, postinumero ja postitoimipaikka
- Syntymäaika
- Sukupuoli
- Viestintäasetukset, joita muuttamalla käyttäjällä on mahdollisuus valita haluaako hän vastaanottaa sähköposti- ja tekstiviestejä, pelkkiä sähköposteja tai pelkkiä tekstiviestejä, vai eikö hän halua vastaanottaa mitään viestejä.
- Listat tulevista ja menneistä varauksista sekä kuntosalikäynneistä

Näistä tiedoista on käyttäjän muokattavissa sähköpostiosoite, puhelinnumero, osoitetiedot, syntymäaika, sukupuoli ja viestintäasetukset. Lisäksi jos käyttäjä on rekisteröitynyt palveluun sivus-

ton eikä Facebookin kautta, hänellä on Omat tiedot-sivulla mahdollisuus vaihtaa kirjautumiseen käytettävä salasanansa. Tietojen näyttämiseen tarvitaan kirjautumisen yhteydessä FitnessBooker-rajapinnalta palautuvapoletti. Poletin löytyessä sen oikeellisuus tarkastetaan rajapinnalta lähettämällä se /auth/valid_token/-resurssille. Mikäli poletti on käypä, sen avulla voidaan HTTP GET-pyyntö /profile/-resurssille, joka palauttaa asiakkaan tiedot JSON-muodossa. Saadut tiedot muutetaan luettavaan muotoon ja tulostetaan sivun HTML-elementteihin. Tiedot tallennetaan lisäksi PHP-sessiomuuttujaan tietojen päivityksessä käytettäväksi. Käyttäjän varaukset ja kuntosalikäynnit saadaan tekemällä poletilla GET-pyyntö /profile/bookings/-resurssille. Saatu data jaotellaan tuleviin varauksiin, menneisiin varauksiin, ja kuntosalikäynteihin. Jaottelu tapahtuu ensiksi tarkastamalla onko varauksen tapahtuman id-tunnusnumero negatiivinen, missä tapauksessa kyseessä on kuntosalikäynti, ja tämän jälkeen tarkastamalla onko tapahtuman alkamispäivämäärä ja -aika jo mennyt sivun latauksen ajankohtana.

5.2.7 Käyttäjätietojen päivitys

Käyttäjä voi halutessaan päivittää kaikkia rekisteröidyttäessä tallennettuja tietojaan lukuun ottamatta nimeä ja asiakasnumeroa. Tietojen päivitys tapahtuu Omat tiedot-sivun lomakkeella, josta päivitetyt tiedot lähetetään profile-update.php-tiedoston käsiteltäväksi. Jottei FitnessBooker-rajapinnalle lähetettäisi turhaan muokkaamattomia tietoja, tehdään päivityksen aluksi GET-pyyntö /profile/-resurssille, josta saadaan rajapinnassa ennestään löytyvät tiedot. Näitä tietoja verrataan juuri lomakkeella lähetettyihin ja mahdolliset päivitykset kerätään merkkijonoon lähetettäväksi rajapinnalle. HTTP:n PATCH-verbillä lähetetään päivitykset sisältävä merkkijono rajapinnalle ja käyttäjä ohjataan takaisin Omat tiedot-sivulle päivityksen onnistumisesta tai epäonnistumisesta kertovan viestin kanssa. Käyttäjä voi myös vaihtaa salasanansa Omat tiedot-sivulla klikkaamalla ”Vaihda salasanasi”-linkkiä, jolloin sivulle tulostetaan kolme uutta kenttää. Ensimmäinen lomakekenttä vaatii käyttäjän nykyisen salasanan, toinen uuden salasanan ja kolmas uuden salasanan toiseen kertaan. Nykyisen salasanan oikeellisuus tarkistetaan POST-pyynnöllä rajapinnan /auth/valid_password/:ille, mikäli /valid_password/ palauttaa ”true”-arvon uusi salasana lisätään ylempänä mainittuun merkkijonoon. Käyttäjän halutessa keskeyttää salasanan vaihto hänen tarvitsee vain painaa ”En halua vaihtaa salasanaani”-linkkiä jolloin kentät piilotetaan eikä niiden sisältöä lähetetä eteenpäin päivitykseen.

Jotta päivitys toimisi mahdollisimman sujuvasti käyttäjän virheistä riippumatta, tietojen päivityksen sekä rekisteröitymisen lomakkeissa on käytössä jQuery JavaScript-kirjaston Validation-lisäosa (Kuvio 7). Lisäosalle kerrotaan JavaScript-koodilla tarkastettavaksi haluttavan lomakkeen id-attribuutti, lomakkeen tarkastettavien kenttien nimet ja niille suoritettavat tarkastukset, sekä virhesyötteiden ilmetessä esitettävät viestit. Koska käyttäjä voi myös vaihtaa sähköpostiosoitettaan, jonka tulisi olla jokaisella asiakkaalla yksilöllinen, rajapinta palauttaa vastaavan sähköpostin jo löytyessä virheen ja käyttäjä ohjataan takaisin Omat tiedot-sivulle ja virheestä ilmoitetaan yläpalkissa.

```
// Profile form validation
$( "#fibo-profile-form" ).validate({
  errorClass: 'fibo-form-error',
  rules: {
    fibo_pass: {
        required: true,
        minlength: 8
    },
    fibo_pass_check: {
      equalTo: "#fibo_pass"
    },
    phone: {
        number: true
    },
    zip: {
        number: true
    }
  },
  messages: {
    phone: {
        required: "Puhelinnumero on pakollinen tieto.",
        number: "Numero on virheellinen. Käytä vain numeroita."
    },
    address: "Katuosoite puuttuu.",
    zip: {
        required: "Postinumero puuttuu.",
        number: "Numero on virheellinen. Käytä vain numeroita."
    },
    city: "Paikkakunta puuttuu.",
    email: "Sähköpostiosoite on virheellinen.",
    fibo_pass: "Salasana on liian lyhyt. (väh. 8 merkkiä)",
    fibo_pass_check: "Salasanat eivät täsmää."

  }
});
```

KUVIO 7. Lomakkeen tarkistus jQuery Validation-lisäosan avulla

5.2.8 Rekisteröityminen

Koska uuden asiakkaan rekisteröityminen onnistuu myös Tooltipin FitnessBooker-palvelun esittelyyn tarkoitetulla sivulla, rekisteröitymisen lisääminen WordPress-lisäosaan onnistui myös kirjau-

tumisen ja asiakastietojen esittämisen jälkeen. Kuten aiemmin mainittua, käyttäjällä on mahdollisuus rekisteröityä joko Facebook-tunnuksillaan tai WordPress-sivuston kautta. Facebook-rekisteröitymisen tapauksessa rekisteröitymislomake on esitäytetty käyttäjän Facebookista löytyvällä nimellä ja sähköpostiosoitteella. Käyttäjän tulee täyttää kentät, jotka löytyvät myös Omat tiedot-sivulta ja lisäksi syöttää haluamansa salasana kahteen kertaan. Rekisteröitymislomakkeessa käytetään samaa virheentarkastusta kuin tietojen päivityksessä eli jQuery Validation-lisäosaa sekä mahdollisesta sähköpostivirheestä kertovaa viestiä. Koska täytettäviä tietoja on kohtuullisen paljon, käyttäjän syöttämät tiedot tallennetaan väliaikaisesti sessiomuuttujiin lomakkeen lähetyksen yhteydessä. Jos FitnessBooker-rajapinta palauttaa sähköpostivirheen, käyttäjä ohjataan takaisin rekisteröitymissivulle, jossa kentät ovat valmiiksi täytetty sessiomuuttujiin tallennetuilla tiedoilla. Uuden käyttäjän tiedot lähetetään rajapinnalle tekemällä POST-pyyntö /profile/-resurssille. Onnistuneen rekisteröitymisen jälkeen käyttäjä ohjataan WordPress-sivuston etusivulle, jossa hän voi kirjautua antamillaan tiedoilla.

5.2.9 Kalenteri

Yksi FitnessBooker-lisäosan päätoiminnoista on näyttää käyttäjälle asiakasyrityksen tapahtumakalenteri johon on listattu valitun viikon ryhmäliikuntatapahtumat (Kuvio 8). FitnessBooker-palvelun tilanneella yritysasiakkaalla on mahdollisuus luoda useita eri tapahtumia sisältäviä kalentereita, joista WordPressin hallintasivun lisäosa-asetuksissa on valittavissa Kalenteri-sivulla esitettävä ryhmäliikuntakalenteri. Kalenteri-sivulla näytetään siis valittu kalenteri tai jokaisen kalenterin kaikki tapahtumat samalla sivulla. Kalenterin tapahtumat saadaan tekemällä GET-pyyntö WordPressin HTTP-rajapinnan wp_remote_get-funktiolla FitnessBooker-rajapinnan /calendars/[id]/events/-resurssille, jolle annetaan parametreina valitun viikon alkamis- ja päättymisajat. Lisäosan asetuksissa määritettävä kalenterin id-numero määrittää minkä kalenterin tapahtumat rajapinnalta pyydetään. Jos id on -1, näytetään kaikki kalenterit, mikä edellyttää kalenterien lukumäärän tietämistä. Tämä luku saadaan tekemällä GET-pyyntö /calendars/-resurssille ja laskemalla sen palauttamien tulosten lukumäärä. Tapahtumat tulostetaan viikonpäivittäin (Kuvio 9) listoihin PHP:n for-silmukassa käymällä /calendars/[id]/events/-resurssilta saadut tapahtumat läpi tulostaen ne aina oikean viikonpäivän osuessa kohdalle. Tapahtumien tulostettavat tiedot ovat sen alkamisaika, tapahtuman tyyppi ja jokaisen tapahtuman kohdalla ainutlaatuinen URL, joka ohjaa tapahtuman Ryhmäliikunta-sivulle. Tapahtuman URL määräytyy tapahtuman kalenterin ja itse tapahtuman id-tunnistenumeroiden mukaan.

Kalenteri

Valitse viikko 46 / 2014 (Nyt) ▼

Maanantai

12:00 Vauva-aikuinen -jumppa
16:00 ZUMBA®
18:00 LesMills BODYPUMP
19:20 Jooga

Keskiviikko

09:00 Core-training
11:00 LesMills BODYPUMP
16:00 ZUMBA®
19:00 Spinning

Perjantai

11:00 Spinning
12:00 Jooga
13:00 ZUMBA®
16:00 ZUMBA®
17:00 Spinning
20:00 Spinning

Sunnuntai

08:00 ZUMBA®
09:10 Spinning
10:00 Jooga
17:00 Vauva-aikuinen -jumppa
19:00 LesMills BODYPUMP

Tiistai

08:00 ZUMBA®
13:00 Jooga
17:00 Spinning
18:00 LesMills BODYPUMP

Torstai

14:00 Core-training
18:00 LesMills BODYCOMBAT
18:45 LesMills BODYJAM

Lauantai

12:00 Core-training

KUVIO 8. Ryhmäliikuntakalenteri

```
$result = wp_remote_get ( $api_url . '/calendars/' . $calendar_id .'/events?from=' . $week_start_final . '&to=' . $week_end );

if ( !is_wp_error ($result) ) {
$result_body= json_decode($result['body'], true);

    foreach ($result_body as $event) {
        // Find out the weekday
        $date = date_format(date_create($event['begin']), "d.m.Y");
        $time = date_format(date_create($event['begin']), "H:i");
        $weekday = date('l', strtotime($date));

        if ($weekday == $weekdays[$j]) {
            echo '<li>';
            echo $time . ' <a href="' . get_bloginfo('url') . '/ryhmaliikunta/' . $calendar_id . '/' . $event['event_id'] . '/'
                . $event['type'] . '">' . $event['type'] . '</a>';
            echo '</li>';
        }
    }
}
```

KUVIO 9. Yhden viikonpäivän tapahtumien tulostus

5.2.10 Tapahtuma

FitnessBooker-rajapinnan /calendars/[id]/events/[id]/-resurssi palauttaa yksittäisen tapahtuman tiedot. Resurssin vaatimat id:t löytyvät sivulle ohjaavasta URL:stä, joka on ennen aiempaa käsiteltyä URL-uudelleenkirjoitusta muodossa http://toniheikkala.fi/wordpress/?page_id=1&calendar_id=1&event_id=1. Tästä URL:stä saadaan PHP:n GET-metodia vastaavaa WordPressin wp_query_var-funktiota käyttämällä talteen kalenterin id ja tapahtuman id.

Resurssi palauttaa seuraavat tapahtuman tiedot:

- Tapahtuman id
- Alkamisaika
- Päättymisaika
- Liikuntamuoto
- Liikuntamuodon id
- Kalenterin id
- Ohjaaja
- Huomautus tai lisätiedot
- URL tapahtuman varaussivulle
- Vapaina olevien paikkojen lukumäärä

Näistä lisäosa käyttää kaikkia paitsi URL-tietoa, sillä lisäosa luo tapahtumien URL:ät niin että ne eroavat Tooltipin alkuperäisestä URL-rakenteesta, jota rajapinta tarjoaa. Ryhmäliikunta-sivun otsikkona toimii kyseessä olevan tapahtuman tyyppi. Koska tyyppi ja täten myös otsikko vaihtelee tapahtumasta riippuen, täytyy otsikkoon käyttää WordPressin the_title-suodatinkoukkua (Kuvio 10), joka hakee jokaisella Ryhmäliikunta-sivun latauskerralla kyseisen tapahtuman tyypin rajapin-

nasta ja asettaa sen sivun otsikoksi. Sivulla esitetään otsikon lisäksi, viikonpäivä jona tapahtuma on, tapahtuman alkamispäivämäärä ja –aika, tapahtuman loppumisaika, tapahtuman ohjaaja, sekä tapahtuman kuvaus. Tapahtuman kuvausta ei ole suoraan tapahtuman tiedoissa vaan se on saatu tekemällä kyseessä olevan tapahtuman liikuntamuodon id:llä GET-pyyntö /types/-resurssille, joka palauttaa muun muassa tarvittavan liikuntamuodon kuvauksen.

```php
function fibo_filter_title() {

    add_filter('the_title', 'fibo_event_title');
    function fibo_event_title($title){

        $event_page = get_page_by_title( 'ryhmaliikunta' )->ID;

        //Return new title if called inside loop and the page is Event correct
        if ( in_the_loop() && is_page( $event_page ) ) {

            // Calendar and event IDs from query_var
            $event_id = get_query_var( 'event_id' );
            $calendar_id = get_query_var( 'calendar_id' );

            // Get event data from API
            $result = wp_remote_get ( $_SESSION['api_url'] . '/calendars/' . $calendar_id . '/events/' . $event_id );

            if ( !is_wp_error ($result) ) {

                $results = json_decode($result['body'], true);
                return $results['type'];

            }
        }

        // If not Event page, return page title as normal
        return $title;
    }
}
add_action( 'init', 'fibo_filter_title');
```

KUVIO 10. Ryhmäliikunta-sivun otsikon muutos the_title-suodatinkoukun avulla

5.2.11 Varaukset

Varausten tekeminen ryhmäliikuntatapahtumiin tapahtuu halutun tapahtuman Ryhmäliikunta-sivulla. Käyttäjän täytyy olla kirjautunut sisään nähdäkseen varauslomakkeen, sillä varausten tekeminen ja poistaminen tapahtuu resurssin /calendars/[id]/events/[id]/bookings avulla, joka vaatii validin kirjautumispoletin. Mikäli käyttäjä ei ole sisäänkirjautuneena saapuessaan Ryhmäliikunta-sivulle, hänelle näytetään Varaaminen-otsikon alla viesti ”Ole hyvä ja kirjaudu sisään” ja mahdollisuus kirjautua joko lomakkeella tai Facebookin kautta. Sivun kirjautumisosion alta löytyy myös kehotus rekisteröitymään, sekä linkki rekisteröitymissivulle, mikäli käyttäjällä ei ole vielä tunnuksia. Kirjautuneelle käyttäjälle näytetään tieto, montako paikkaa tapahtumassa on vapaana, montako varausta käyttäjällä on jo kyseiseen tapahtumaan, ja kuinka monta varausta yksittäinen käyttäjä voi tehdä tapahtumaan. Vapaiden paikkojen lukumäärä tulevat /calendars/[id]/events/[id]/-resurssilta. Mikäli vapaita paikkoja ei löydy, näytetään viesti ”Tapahtuman paikat ovat täynnä.” Käyttäjän itse tekemien varausten lukumäärä saadaan tekemällä GET-

pyyntö käyttäjän poletilla /calendars/[id]/events/[id]/bookings-resurssille, joka palauttaa varauksen tiedot, jotka ovat varauksen id, tapahtuman id, kalenterin id, tieto onko varaus lunastettu, ja tapahtuman liikuntamuodon nimi. Resurssin palautuksen perusteella käyttäjälle esitetään hänen tekemiensä varausten lukumäärä. Varausten enimmäismäärä tulee WordPress-sivuston hallinnasta lisäosan asetussivulta, jossa tämä lukumäärä voidaan määrittää. Käyttäjä voi tehdä varauksia tapahtumaan kun siihen on vapaita paikkoja, hän ei ole tehnyt maksimimäärää varauksia, eikä tapahtuma ole vielä mennyt. Varaaminen tapahtuu valitsemalla alasvetovalikosta haluttu määrä varauksia ja painamalla ”Varaa”-painiketta. Rajapinnan /calendars/[id]/events/[id]/bookings-resurssille tehdään POST-pyyntö tapahtuman id:llä kirjautumispoletin kanssa niin monta kertaa kuin käyttäjä haluaa varauksia tehtävän ja käyttäjä ohjataan takaisin tapahtuman sivulle. (Kuvio 11) Jos tapahtuman paikat täyttyvät varauksen aikana, käyttäjälle näytetään virheilmoitus yläpalkissa. Varausten peruminen tapahtuu vastaavasti tekemällä /calendars/[id]/events/[id]/bookings/[id]/-resurssille DELETE-pyyntö kirjautumistokenilla niin monta kertaa kuin käyttäjä haluaa varauksia peruttavan. Peruttavan varauksen id saadaan sessiomuuttujaan tallennetusta taulukosta, joka sisältää käyttäjän tekemät varaukset kyseessä olevaan tapahtumaan. Varausten peruminen aloitetaan aina viimeiseksi tehdystä varauksesta.

```php
session_start();

// API URL from plugin settings
$api_url = $_SESSION['api_url'];
// Token from session
$token = $_SESSION['token'];
// User's bookings amount from session variable
$user_bookings = $_SESSION['user_bookings'];

$event_id = $_POST['event-id'];
$calendar_id = $_POST['calendar-id'];

// How many bookings should be deleted
$delete_bookings = $_POST['bookings-delete-amount'];

// Delete the number of bookings user wants
for ($i=0; $i<$delete_bookings; $i++) {

    // ID of the booking to be deleted
    $booking_id = $user_bookings[$i]['booking_id'];

    $ch = curl_init($api_url . '/calendars/' . $calendar_id . '/events/' . $event_id . '/bookings/' . $booking_id);

    curl_setopt_array($ch, array(
        CURLOPT_CUSTOMREQUEST => "DELETE",
        CURLOPT_RETURNTRANSFER => 1,
        CURLOPT_HTTPAUTH => CURLAUTH_BASIC,
        CURLOPT_USERPWD => $_SESSION['token'] . ":"
    ));

    curl_exec($ch);
}

curl_close ($ch);

// Redirect
if ($_SESSION['prevloc'] == NULL) {
    header('Location: ' . $_SESSION['site_url']);
    die();
}
else {
    header('Location: ' . $_SESSION['prevloc']);
    die();
}
```

KUVIO 11. Varausten peruminen

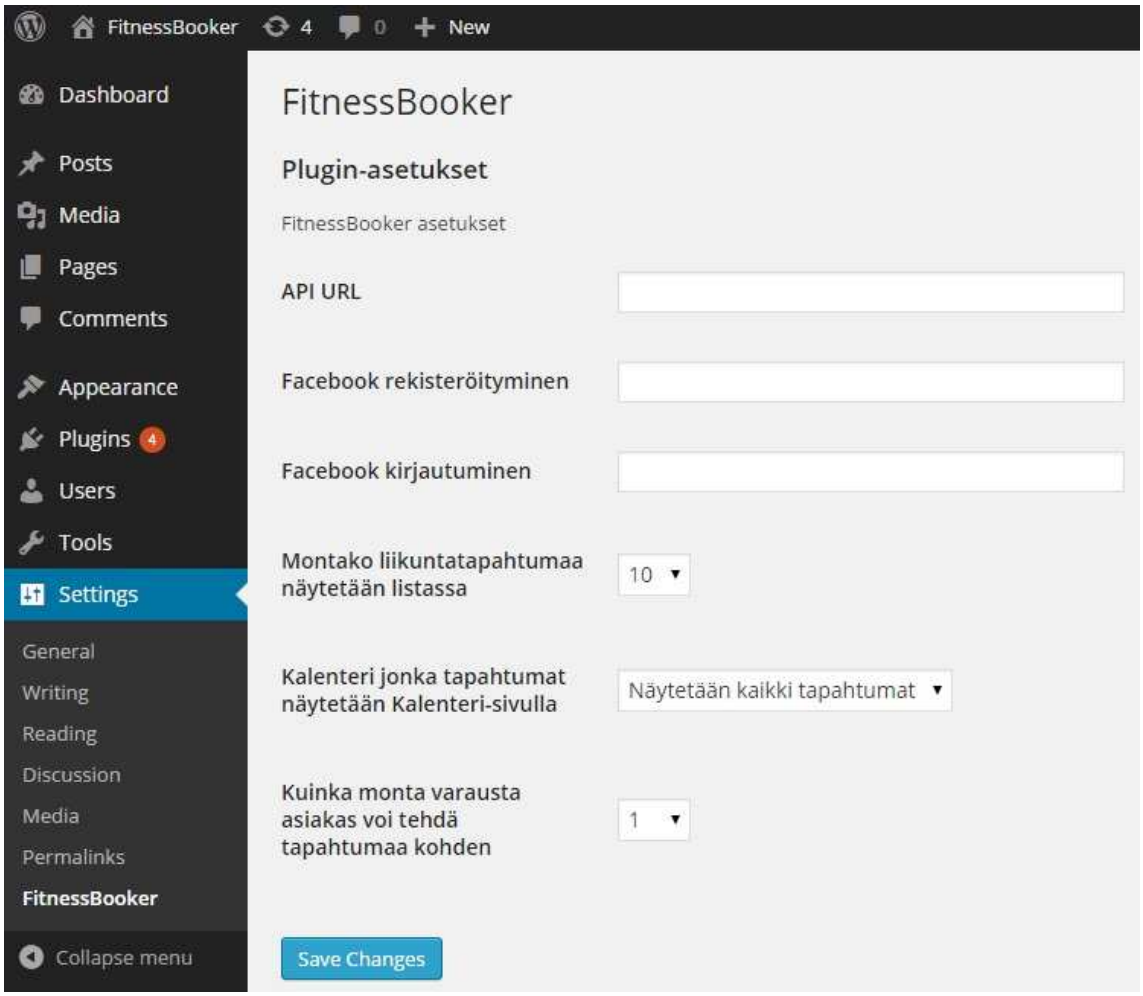
5.2.12 Asetukset

Linkki lisäosan asetuksiin on WordPressin hallintapuolen sivupalkissa WordPressin muiden asetussivujen kanssa. Asetussivulla (Kuvio 13) on lisäosan toiminnan kannalta välttämättömien URL-kenttien lisäksi sivuston ylläpitäjälle muutamia lisäosaa koskevia muokkausmahdollisuuksia. API URL-kenttään syötetään rajapinnan URL-osoite johon kohdistuvat kaikki lisäosan HTTP-pyynnöt. Facebook-kenttiin syötetään URL-osoitteet, joista löytyvät Tooltipin Facebook-kirjautumisen ja –rekisteröitymisen mahdollistavat tiedostot joihin käyttäjä ohjataan hänen tunnistautuessaan Facebook-tilillään. Nämä kolme kenttää tullaan luultavasti täyttämään valmiiksi Tooltipin toimesta lisäosan asennuksen yhteydessä, jolloin asiakkaalle jäisi vain sivuston sisältöön ja ulkoasuun liittyvien asetusten säätäminen haluamikseen. Ensimmäinen alasvetovalikko tarjoaa mahdollisuuden päättää kuinka monta tapahtumaa näytetään Seuraavat tapahtumat-listassa, toinen määrittelee minkä kalenterin tapahtumat näytetään Kalenteri-sivulla, ja kolmannesta valitaan kuinka monta varausta asiakas voi tehdä yhteen liikuntapahtumaan. Asetusten toteutukseen on käytetty sekä asetukset- että options-rajapintaa, jotka mahdollistavat asetussivulla tehtyjen valintojen kulkemisen muiden lisäosan osioiden käytettäviksi, asetussivuston lomakkeella tehtyjen valintojen tallentuessa options-rajapinnan avulla tietokantaan.

WordPressin asetukset-rajapinnan register_setting-funktiota käyttämällä asetukset rekisteröidään määrittämällä niille ryhmä ja nimi. Jokaiselle rekisteröidylle asetukselle luodaan omat lomakekentät add_settings_field-funktiolla (Kuvio 12), joka ottaa parametrina muun muassa lomakekenttien HTML-koodin tulostavan funktion. Funktiolla määritetään myös nimi, jolla kyseisen kentän määrittämä vaihtoehto (option) tallennetaan WordPressin tietokantaan ja jolla vaihtoehtoa kutsutaan ohjelmakoodissa. Esimerkiksi get_option('api_url') palauttaa asetukset-sivun lomakkeella syötetyn FitnessBooker-rajapinnan URL-osoitteen. Alasvetovalikolla valittaville asetuksille on määritetty valmiiksi oletusarvot ohjelmakoodissa käyttäen add_option funktiota. Tulevat tapahtumat –listassa näytettyjen tapahtumien määrä on rajattu kymmeneen, käyttäjän tekemien varausten määrä yhteen kappaleeseen ja Kalenteri-sivulla esitetään oletuksena kaikkien kalenterien tapahtumat. Register_setting-, add_settings_field- ja add_option-funktiot ovat sisällytetty yhteen funktioon, joka ajetaan admin_init-koukun avulla ennen hallintapuolen lopullista latautumista.

```
// Add API URL field
add_settings_field(
    'api_url', 'API URL', 'api_url_input', 'fibo_settings', 'fibo_settings_section'
);

// Set default values for the fields
add_option( 'events_amount', '10' );
add_option( 'allowed_bookings', '1' );
add_option( 'calendar_id', '-1');
```

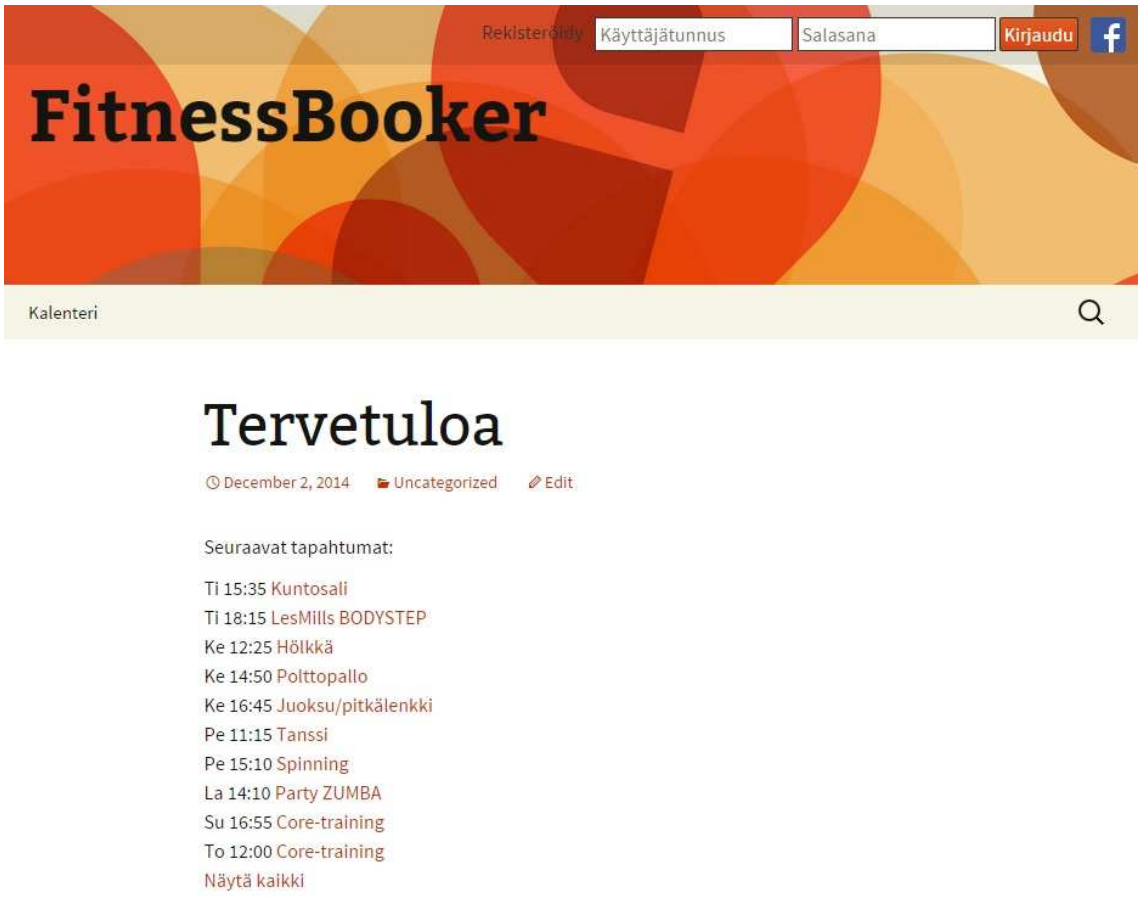
KUVIO 12. Esimerkki add_settings_field- ja add_option-funktioista.



KUVIO 13. Lisäosan asetussivu ja sen sijainti WordPressin hallinnassa

5.3 Tulokset

Työn tuloksena syntyi toimeksiantajan kriteerit täyttävä WordPress-lisäosa, jonka toteutuksessa on käytetty useita web-ohjelmointitekniikoita ja WordPressin sekä Tooltipin tarjoamia ohjelmointirajapintoja. WordPress-sivuston (Kuvio 14) ylläpitäjä pystyy lisäosan avulla integroimaan vaivattomasti FitnessBooker-verkkopalvelun toiminnot sivustolleen ja voi halutessaan tehdä muutoksia lisäosan toimintaan asetussivulta.



KUVIO 14. Esimerkki WordPress-sivustosta, jossa lisäosa on käytössä

POHDINTA

Lähtiessäni miettimään aihetta opinnäytetyölleni toivoin löytäväni front-end kehitystaitojani kehittävän ja mahdollisesti WordPressiin liittyvän aiheen. Kiinnostukseni WordPressiä kohtaan kumpusi työharjoittelusta, jonka suoritin yrityksessä, joka käytti WordPressiä asiakasyritysten verkkosivujen julkaisualustana. Tooltip Oy:n tarjoama kehitystehtävä ei vastannut täysin suunnitelmiani keskittyen enimmäkseen back end -kehitykseen PHP:lla, mutta päätin tarttua tilaisuuteen kohentaa PHP-ohjelmointitaitojani ja oppia lisää WordPress-lisäosakehityksestä. Tarjottu tehtävä houkutti myös siksi, että pääsisin työskentelemään kasvavan ohjelmistoyrityksen kanssa. Näin tämän erittäin positiivisena seikkana, sillä vaihtoehtona oli monen kanssaopiskelijani valitsema kehitystehtävä ilman toimeksiantajaa, tai jonkin sortin tutkimus, joka ei välttämättä tule edes käyttöön.

Ennen opinnäytetyötä olin ennestään ohjelmoinut PHP:llä sekä koulun opintojaksoilla että työharjoittelussa, jossa tutustuin myös syvemmin WordPress-julkaisujärjestelmään. PHP-ohjelmointitaitoni olivat kuitenkin rajoitteelliset, enkä ollut tutustunut missään määrin WordPress-lisäosien kehitykseen, joten epäilin saisinko työtehtävän suoritettua alustavasti sovittuun ajankohtaan mennessä. Alkukangertelujen jälkeen sain työn kuitenkin etenemään hyvällä tahdilla ottaen huomioon, että olin lähes koko kesän töissä ja kerkesin kehittämään lisäosaa vain muutaman tunnin iltaisin. Työn sovitun valmistumispäivämäärän eli heinäkuun lopun lähestyessä kävi kuitenkin ilmi, ettei lisäosan valmistumisella ollutkaan niin kiire kuin aluksi luulin. Lisäosan tiedostojen toimitus venyikin lokakuun loppupuolelle.

Lisäosan kehitys oli antoisa kokemus niin ohjelmoimisen kuin myös oman ajankäytön hallinnan opettelun saralla. Lähes jokaisen lisäosan eri osion toteutus vaati uuden ohjelmointimenetelmän tai -rajapinnan opiskelun, joten uutta tuli opittua valtaisasti. Ajankäytön ja itsekurin opettelun tärkeys ilmeni erityisesti kesällä, jolloin piti osata tasapainottaa päivätyö, opinnäytetyö ja vapaa-aika. Loppujen lopuksi lisäosa kuitenkin valmistui ja olen tyytyväinen lopputulokseen.

LÄHTEET

Costello, R. Building Web Services the REST Way. Viitattu 25.11.2014, http://www.xfront.com/REST-Web-Services.html.

Elkstein, M. What is REST? Viitattu 27.11.2014. http://rest.elkstein.org/2008/02/what-is-rest.html

Fielding, R. Architectural Styles and the Design of Network-based Software Architectures. Viitattu 25.11.2014, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Leiniö, T. 2013. Mitä ovat shortcodet ja kuinka käytät niitä? Viitattu 3.11.2014, http://wpopas.fi/mita-ovat-shortcodet-ja-kuinka-kaytat-niita/.

WordPress. 2014. About WordPress. Viitattu 30.10.2014, https://wordpress.org/about/.

WordPress. 2014. Features. Viitattu 30.10.2014, https://wordpress.org/about/features/.

WordPress Codex. 2014. Database API. Viitattu 4.11.2014, http://codex.wordpress.org/Database_API.

WordPress Codex. 2014. HTTP API. Viitattu 4.11.2014, http://codex.wordpress.org/HTTP_API.

WordPress Codex. 2014. Options API. Viitattu 4.11.2014, http://codex.wordpress.org/Options_API.

WordPress Codex. 2014. Plugin API. Viitattu 4.11.2014, http://codex.wordpress.org/Plugin_API.

WordPress Codex. 2014. Plugin API/Action Reference/init. Viitattu 6.11.2014, http://codex.wordpress.org/Plugin_API/Action_Reference/init.

WordPress Codex. 2014. Rewrite API. Viitattu 4.11.2014, http://codex.wordpress.org/Rewrite_API.

WordPress Codex. 2014. Settings API. Viitattu 4.11.2014,
http://codex.wordpress.org/Settings_API.

WordPress Codex. 2014. Shortcode API. Viitattu 4.11.2014,
http://codex.wordpress.org/Shortcode_API.

WordPress Codex. 2014. Theme Development. Viitattu 31.10.2014,
http://codex.wordpress.org/Theme_Development.
WordPress Codex. 2014. WordPress APIs. Viitattu 31.10.2014,
http://codex.wordpress.org/WordPress_APIs.

Ylimäinen, J. 2014. FitnessBooker API. Ei julkinen. Viitattu 31.10.2014.