

Treenikalenteri.net – kuntoilijoiden verkkopalvelu

Tommi Paavola

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2013



Tietojenkäsittelyn koulutusohjelma

Tekijä tai tekijät Tommi Paavola	Ryhmätunnus TIM08I
Opinnäytetyön nimi Treenikalenteri.net – kuntoilijoiden verkkopalvelu	Sivu- ja liitesivumäärä 51 + 10
Opettajat tai ohjaajat Sirpa Marttila	
<p>Tämän opinnäytetyön taustalla olivat aktiivisten liikkujien tarpeet sekä tekijän oma kiinnostus liikuntaan ja ohjelmointiin. Verkossa ei ole juurikaan saatavilla helppokäyttöisiä, ilmaisia ja tuotemerkkiriippumattomia palveluita, jotka innostavat liikkumaan sekä edistävät kehittymisen ja harjoittelun seuraamista.</p> <p>Työn tavoitteena oli toisaalta tuottaa prototyyppi edellä mainitusta verkkopalvelusta, toisaalta selvittää kirjallisuuden ja Internet-lähteiden avulla sekä tutkia käytännön ohjelmoinnin kautta, miten voidaan luoda tietoturvallinen verkkopalvelu, jossa sovelluslogiikka on täysin erillään käyttöliittymästä.</p> <p>Työ toteutettiin kevään 2013 aikana tekijän omalla testipalvelimella ja sen tuloksena syntyi PHP:llä ohjelmoitu oma toteutus MVC -arkkitehtuurista. Työssä todetaan lopupäätelmänä, että MVC -arkkitehtuuri mahdollisti sovelluslogiikan erottamisen käyttöliittymästä. Tietoturvan osalta oli päätelmänä, että tietoturva on ensisijaisesti kiinni kehittäjän viitseliäästä asenteesta. Tähän liittyy toisaalta uhkien seuraaminen ja toisaalta käyttäjän syötteiden ja muiden tarpeellisten tarkistusten huolellinen ohjelmoiminen sovellukseen.</p>	
Asiasanat Sovelluskehikset, tietoturva, PHP, SQL	

Degree Programme in Information Technology

<p>Author or authors Tommi Paavola</p>	<p>Group TIM08I</p>
<p>The title of thesis Treenikalenteri.net – web service for amateur sport</p>	<p>Number of pages and appendices 51 + 10</p>
<p>Supervisor or supervisors Sirpa Marttila</p>	
<p>The background of this thesis was the needs of active amateur sportsmen and sports-women as well as the author's own interest in sports and programming. There are not so many web services available that are easy-to-use, free, and independent of a brand that would encourage exercise and help in monitoring own training and progression.</p> <p>The aim of this thesis was to produce a prototype of the above-mentioned web service. Secondly, the aim was to examine the literature and Internet sources, as well as to explore by practical programming how to create a secure service where the application logic is completely separated from the user interface.</p> <p>The thesis was carried out during spring 2013 in the author's development environment and it resulted in PHP-programmed own implementation of MVC architecture. The conclusion was that the MVC architecture enabled the separation of application logic and interface. Regarding the information security in a web service it was concluded that security is a matter of attitude. It involves, on the one hand, being aware of common threats in the web and on the other it requires meticulous checking of the user's input and implementing all the necessary controls in the application.</p>	
<p>MVC-architecture, information security, PHP, SQL</p>	

Sisällys

1 Johdanto	1
2 MVC-arkkitehtuuri.....	4
2.1 Malli.....	4
2.2 Näkymä.....	5
2.3 Ohjain	5
2.4 MVC-arkkitehtuurin toiminta.....	6
2.5 MVC-arkkitehtuurin etuja ja haittoja	8
3 Verkkopalvelun rakentaminen.....	9
3.1 Palvelinpään tekniikat	10
3.2 Selainpään tekniikat.....	18
3.3 Verkkopalvelun tietoturva.....	20
4 Toteutus.....	27
4.1 Treenikalenterin ominaisuudet	28
4.2 Treenikalenterin arkkitehtuuri	32
4.3 Treenikalenterin toteuttaminen	37
5 Yhteenveto, pohdinta ja jatkokehitysehdotukset	43
Lähteet.....	45
Liitteet.....	52
Liite 1. Rekisteröintilomake	52
Liite 2. Asetukset.....	53
Liite 3. Harjoituksen tallentaminen.....	54
Liite 4. Indeksien tallentaminen	55
Liite 5. Muokkausikkunat.....	56
Liite 6. Listanäkymät.....	57
Liite 7. Kilometridiagrammi.....	58
Liite 8. Tietokannan taulut indeksit ja käyttäjät.....	59
Liite 9. Tietokannan taulut käyttäjät_lajit ja lajit	60
Liite 10. Tietokannan taulut asetukset ja harjoitukset	61

1 Johdanto

Yhä useammat ihmiset ovat ymmärtäneet liikunnan merkityksen terveydelle ja hyvinvoinnille. Kansallisen liikuntatutkimuksen (SLU 2010, 6) mukaan vuonna 2009 aikuisväestöstä 55 % harrasti liikuntaa vähintään neljästi viikossa. Aktiivisia liikkujia on siis paljon ja monet heistä asettavat myös tavoitteita harrastukselleen. Omasta harjoittelusta tulee motivoivampaa ja palkitsevampaa, kun voi seurata kehittymistään ja arvioida tavoitteidensa toteutumista. Työkalu, joka auttaa tässä, kiinnostaakin varmasti monia liikkujia. Toki tällaisia välineitä on jo olemassa, mutta jos asettaa vaatimukseksi, että niiden tulee olla ilmaisia, tuotemerkkiriippumattomia, selkeitä, helppokäyttöisiä ja suomenkielisiä, käy joukko jo melko vähiin.

Produktinäkökulmasta tarkasteltuna opinnäytetyöni tavoitteena on tuottaa prototyyppi verkkopalvelusta, joka innostaa liikkumaan, edistää kehittymisen ja harjoittelun seuraamista sekä auttaa harjoittelun suunnittelussa. Treenikalenteri.net on ilmainen, tuotemerkkiriippumaton ja suomenkielinen. Sen pyrkimyksenä on olla selkeä ja helppokäyttöinen. Prototyypissä toteutetaan palvelun arkkitehtuuri sekä sovelluksen keskeisimmät osat. Valmistuttuaan työ siirretään toimimaan julkiselle www-palvelimelle osoitteeseen www.treenikalenteri.net.

Prosessi- ja oppimisenäkökulmasta tarkasteltuna opinnäytetyöni päätavoitteena on selvittää kirjallisuuden ja Internet-lähteiden avulla sekä tutkia käytännön ohjelmoinnin kautta, miten voidaan luoda tietoturvallinen verkkopalvelu, jossa sovelluslogiikka on täysin erillään käyttöliittymästä. Tavoitteenani ovat myös sovellusarkkitehtuurin selkeä jako itsenäisiin komponentteihin ja koodin hyvä organisointi. Tällä kaikella pyrin siihen, että tulevaisuudessa palvelua olisi helppo ylläpitää ja siihen voisi kohtuullisella työllä lisätä uusia ominaisuuksia.

Tässä opinnäytetyössä tutkitaan MVC-arkkitehtuuria, joka tarkoittaa sovelluksen jakamista itsenäisiin malli-, näkymä- ja ohjainluokkiin. Tämä valinta sopii hyvin edellä mainitun päätavoitteen toteuttamiseen. Markkinoilla on lukuisia MVC-sovelluskehyskiä, mutta tässä työssä pyritään luomaan oma toteutus. Oman järjestelmän luominen mah-

dollistaa syvemmän perehtymisen ohjelmointiin ja on näin oppimisprosessin kannalta perusteltua.

Tämän työn ohjelmointikieleksi valittiin PHP, koska se on ilmainen, olio-ohjelmointia tukeva, varsin yleinen ja sopii erityisesti web-kehitykseen. Lisäksi se sisältää turvallisen ohjelmointirajapinnan tietokantojen käytölle. Treenikalenteri käyttää tietovarastonaan MySQL -relaatiotietokantaa, joka tukee käytetyimpiä oliokieliä. Se on myös ilmainen, paljon käytetty ja sopii hyvin web-projekteihin. PHP:llä ja MySQL:llä on myös varsin laaja käyttäjien ja kehittäjien yhteisö, mikä takaa sen, että on helppo löytää apua ja ideoita erilaisiin tilanteisiin omassa kehitystyössä.

Tämän opinnäytetyön tutkimusmenetelminä ovat kirjallisuuskatsaus ja käytännön ohjelmointityö. MVC-arkkitehtuuri muodostaa tietoperustan keskeisen osan, jota käsitellään luvussa 2. Tietoturva on välttämätöntä ottaa huomioon verkkopalvelua rakennettaessa. Tietoturvaa ja käytettyjä tekniikoita selvitetään tietoperustan luvussa 3. Näihin lukuihin tukeutuen kuvataan palvelun toteutus luvussa 4. Tietoperusta ja käytännöllinen toteutusosa yhdessä pyrkivät vastaamaan tämän opinnäytetyön tutkimusongelmiin, jotka ovat:

- 1) Miten MVC-arkkitehtuuria voi käytännössä hyödyntää luotaessa verkkopalvelu, jossa sovelluslogiikka on erillään käyttöliittymästä?
- 2) Miten PHP:tä voi käyttää tietoturvalisellä tavalla luotaessa sovelluslogiikka edellä mainittuun verkkopalveluun?
- 3) Miten luodaan käyttöliittymä edellä mainittuun verkkopalveluun?

Verkkopalvelun luominen kokonaisuudessaan on erittäin laaja projekti. Taustalla vaikuttaa aina yrityksen tai yhteisön toiminta-ajatus. Kuntoilijoiden tarpeet ja harjoittelukäytänteet ovat Treenikalenterin liiketoimintalogiikan taustalla. Näiden asioiden kartoittaminen ja pohtiminen ovat tärkeitä suunniteltaessa sovelluksen toimintoja, mutta tämän taustan käsittely ei mahdu opinnäytetyöni rajoihin. Verkkopalvelun toteuttamiseen kuuluu runsaasti myös käytettävyyden, käyttöliittymän ja ulkoasun suunnittelua. Vaikka nämä seikat liittyvätkin hyvin läheisesti kolmanteen tutkimusongelmaan, rajataan niiden käsittely laajuuden vuoksi tämän työn ulkopuolelle.

Verkkopalvelun tekniseen toteuttamiseen liittyy lukuisia erilaisia palvelinpään ja selainpuolen tekniikoita. Kuviossa 3 rajataan tämän projektin tekninen viitekehys. Sekin on vielä hyvin laaja kokonaisuus, mutta sen käsitteleminen on perusteltua, jotta lukija voisi hahmottaa kokonaisuuden. Myös tietoturva on laaja käsite, mutta se rajataan tässä opinnäytetyössä käsittämään tietoturvaa, joka riippuu PHP-ohjelmoijan tekemistä ratkaisuksista.

2 MVC-arkkitehtuuri

Arkkitehtuuri on järjestelmän perusrakenne. Siihen kuuluu järjestelmän osat ja niiden väliset suhteet sekä osien suhteet ympäristöön. Arkkitehtuuriin kuuluvat myös periaatteet, jotka ohjaavat järjestelmän suunnittelua ja kehittämistä. (IEEE 2000, 3.) Koskimiehen ja Mikkosen (2005, 18–19) mukaan voidaan puhua järjestelmän perustuslaista, jota noudatetaan järjestelmää tehtäessä ja muutetaan vain erittäin painavin perustein.

MVC-arkkitehtuurissa järjestelmä jakaantuu kolmentyyppisiin osiin: malleihin (model), näkymiin (view) ja ohjaimiin (controller). Perusajatus on erottaa käyttöliittymä sovelluslogiikasta ja -datasta. (Koskimies & Mikkonen 2005, 142.) MVC:n kuvasi ensimmäisenä norjalainen Trygve Reenskaug vuonna 1979 työskennellessään Xeroxilla Palo Alto-tutkimus-laboratoriossa. Idea levisi nopeasti ja on nykyään hyvin yleinen erityisesti olioperustaisissa järjestelmissä. (Deacon 2009, 1; Reenskaug 2007). MVC:n yleistettyä siitä on kehitetty erilaisia muunnelmia. Eri implementaatioiden erot ilmenevät siinä, miten komponenttijako em. osiin on tehty, miten itsenäisiä osat ovat tai miten osat kommunikoivat keskenään. (Qiu, Pallickara & Uyar 2004, 3-4).

2.1 Malli

Malli vastaa pysyvän tietosisällön varastoinnista ja muokkaamisesta. Malli peittää tietovaraston teknisen toteutuksen sovelluksen muilta osilta ja on myös sovelluksen ainoa osa, joka on yhteydessä tietovarastoon. (Huhtamäki 2010, 6). Deacon (2009, 2) mainitsee mallin koostuvan useista luokista, joiden kunkin tehtävänä on mallintaa sovelluksen toiminnon taustalla olevaa ilmiötä ja tuottaa ratkaisu toiminnon toteuttamiseen. Käsitteellisesti voidaan ajatella, että malli on se taso arkkitehtuurista, jolla toteutetaan sovelluksen liiketoimintalogiikka ja tiedon varastointi. Mallitasoon kuuluvia luokkia on sovelluksessa tyypillisesti useita. Malleja voisivat olla esimerkiksi käyttäjä, ostoskori, tietokantaan pääsy tai transaktioiden hallinta. (Huhtamäki 2010, 6; Ping, Kontogiannis & Lau 2004).

Malli on itsenäinen eikä tyypillisesti tiedä muista osista mitään. (Ping, Kontogiannis & Lau 2004; Deacon 2009, 2.) Mallin tehtäviin kuuluu kuitenkin vastata sen tilaa koske-

viin kyselyihin sekä pyyntöihin muuttaa tuota tilaa (Burbeckin 1992). Koskimiehen ja Mikkosen mukaan malli tarjoaa operaatiot, joilla kiinnostuneet voivat ilmoittautua tarkkailemaan sen muutoksia, kuten myös toimenpiteet datan muuttamiseksi. Malli ilmoittaa muutoksen tapahtuttua siitä kiinnostuneille, eli tilanteeseen liittyville näkymille ja ohjaimille (Koskimies & Mikkonen 2005, 142). Kuten tämän pääluvun alussa mainittiin, MVC:n toteutustavat eroavat kuitenkin sinä, miten eri osat kommunikoivat keskenään. Monissa asiakas-palvelin-ympäristöön toteutetuissa ratkaisuisa kaikki kommunikaatio mallin ja näkymän välillä kulkee ohjaimen kautta. Tällöin mallin ja näkymän välillä ei ole minkäänlaista yhteyttä. (Qiu 2004, 2; Salihefendic 2011).

2.2 Näkymä

Näkymä on datan esityskerros, se arkkitehtuurin taso, jolla mallin data esitetään käyttäjälle. Näkymä voi esimerkiksi generoida sivuja tai lomakkeita. (Mitchell, Shafik & Turland 2011, 156.) Verkkopalvelussa näkymä rakentuu tyypillisesti HTML-rungolle, mutta tietoa voidaan tuottaa myös XML-muodossa, tekstinä tai kuvina (Huhtamäki 2010, 7.) Näkymät edustavat jotain osaa käyttöliittymästä ja yhteen malliin voi liittyä useita erilaisia näkymiä (Koskimies & Mikkonen 2005, 142)

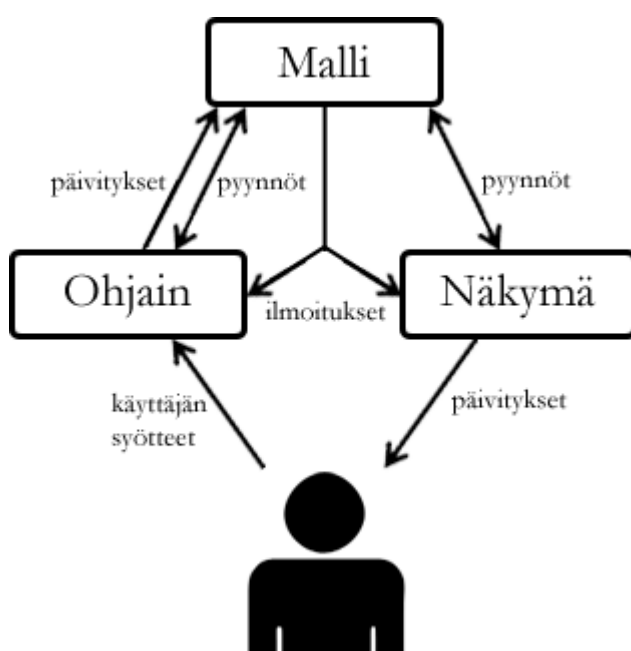
Näkymää pitää huomauttaa, kun mallissa tapahtuu muutoksia. (Ping, Kontogiannis & Lau 2004.) Miten se tapahtuu, riippuu taas MVC:n toteutustavasta. Koskimiehen ja Mikkosen mukaan näkymä voi ilmoittautua eli rekisteröityä yhdelle tai useammalle mallille tarkkailemaan sen muutoksia. Tässä käytetään hyväksi nk. tarkkailijaa (ks. seuraava luku). Muutosten jälkeen näkymä tietää kysyä mallilta sen muuttunutta dataa. (Koskimies & Mikkonen 2005, 142.) Kuten edellä mainittiin, asiakas-palvelin-ympäristössä taas on tyypillistä, että ohjain kokoaa kaiken tarvittavan datan mallista ja välittää sen näkymälle. (Mitchell ym. 2011, 158; Qiu 2004, 2; Salihefendic 2011.)

2.3 Ohjain

Ohjain tulkitsee käyttäjän hiiri- ja näppäimistösyötteet, muuttaa ne loogiseksi sovellustoiminnoiksi ja komentaa mallia ja näkymää muuttumaan sen mukaisesti (Burbeck 1992; Koskimies & Mikkonen 2005, 143). Ohjain hoitaa kaikkien niiden mallissa olevien olioiden luomisen, joita näkymä kulloinkin käyttää. (Ping, Kontogiannis & Lau

2004.) Näin ohjaimet toimivat sovittimina mallien ja näkymien välissä ja pitävät huolta siitä, että ne vastaavat toisiaan. Kuten näkymä, myös ohjain voi rekisteröityä yhdelle tai useammalle mallille tarkkailemaan sen muutoksia. Rekisteröinnissä käytetään hyväksi tarkkailija-suunnittelumallia. (Koskimies & Mikkonen 2005, 142–143.) Sen tarkoitus on määritellä olioiden välille yksi moneen riippuvuussuhde: kun yhden olion tila muuttuu, siitä riippuvat oliot saavat ilmoituksen ja muuttavat itseään automaattisesti. Näin yhteistyössä toimivat luokat pysyvät yhtenäisinä eikä niitä tarvitse sitoa kiinteästi toisiinsa. (Gamma, Helm, Johnson & Vlissides 2001, 293.)

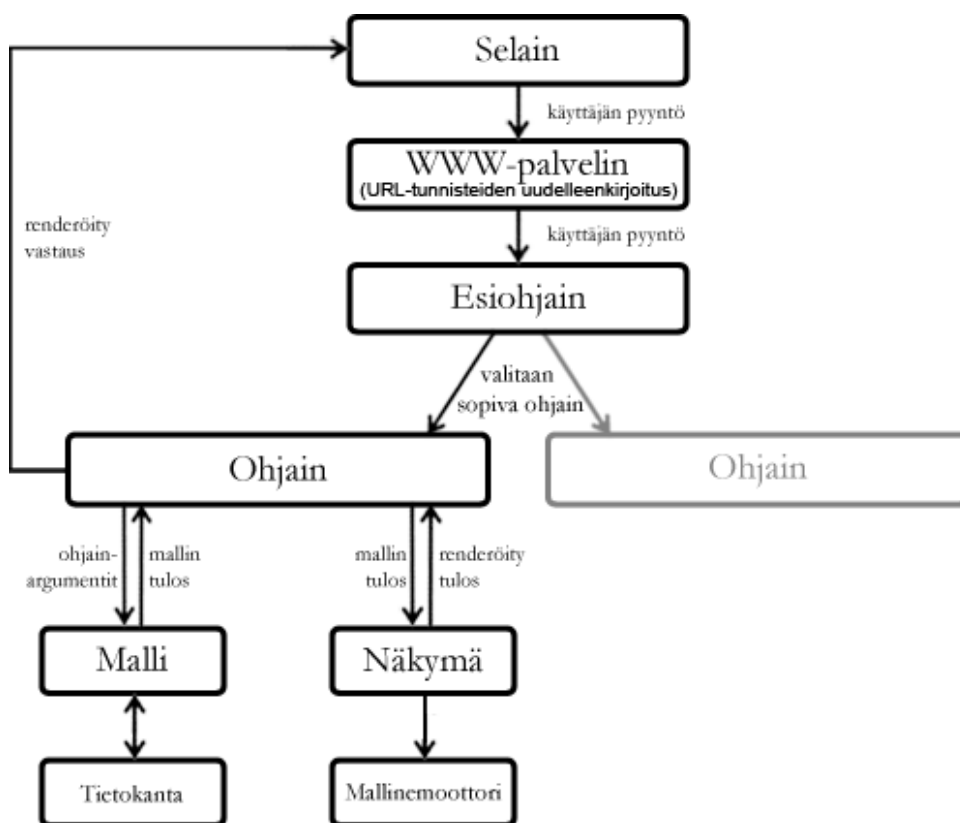
2.4 MVC-arkkitehtuurin toiminta



Kuvio 1. Perinteisen MVC-arkkitehtuurin toiminta (Graham, Urnes & Nejabi 1996, 2)

Perinteisessä, klassisessa MVC-arkkitehtuurin mukaisessa järjestelmässä käyttäjän komento käynnistää vuorovaikutuksen kulun. Ohjain havaitsee tämän ja tulkitsee sen loogiseksi sovellustoiminnoksi. Se päivittää mallia komentamalla toimeenpanemaan vastaavan sovelluspalvelun. Mallin tila muuttuu ja se ilmoittaa tästä sitä tarkkaileville näkymille ja ohjaimille. Näkymä tietää nyt käydä kysymässä mallilta muuttunutta tietoa, jonka mukaan näyttö päivitetään. Vastaavasti ohjain tiedustelee muuttunutta tietoa mallilta ja muuttaa tarpeen mukaan omaa tilaansa. Ohjaimen täytyy esimerkiksi tietää, mitkä komennot ovat nyt sallittuja, mitkä kiellettyjä. (Koskimies & Mikkonen 2005, 143–144.)

MVC-arkkitehtuuri syntyi työpöytäsovellusten käyttöön. Verkkopalvelussa asiat kuitenkin hieman vaikeutuvat, koska näkymä ei voi asiakas-palvelin-ympäristössä helposti tarkkailla mallia. Tällöin on luontevaa korostaa ohjaimen roolia, jolloin näkymään ei jää paljon logiikkaa. Monesti ohjaimen edessä käytetään esiohjainta, eräänlaista portinvartijaa, jonka tehtävä on määrittellä asiakkaan valintojen perusteella oikea ohjain. (Mitchell ym. 2011, 157; Salihefendic 2011; Qiu 2004.)



Kuvio 2. MVC-arkkitehtuurin toiminta verkkopalvelussa (Mitchell ym. 2011, 158)

Verkkopalvelun MVC-arkkitehtuurissa kaikki käyttäjän syötteet ovat selaimella palvelimelle lähetettäviä HTTP-pyyntöjä. Pyyntön saapuessa palvelimelle on yleistä, että URL-tunnisteet kirjoitetaan uudelleen ihmiselle luettavampaan ja myös hakukoneille edullisempaan muotoon. URL:n perusteella esiohjain valitsee varsinaisen ohjaimen, joka mallien avulla muuttaa sovelluksen tilaa ja valitsee asiakkaalle lähetettävän näkymän. (Huhtamäki 2010, 8; Mitchell ym. 2011, 158–160; Rantala 2005, 99)

MVC-arkkitehtuurissa toimintojen jako mallin ja ohjaimen kesken ei aina ole yksiselitteinen. Esimerkiksi käyttäjän syötteen tarkistaminen voi kuulua jommankumman tehtäviin. Verkkopalvelussa mallin ja näkymän suhde on kuitenkin melko selkeä: näkymä ei

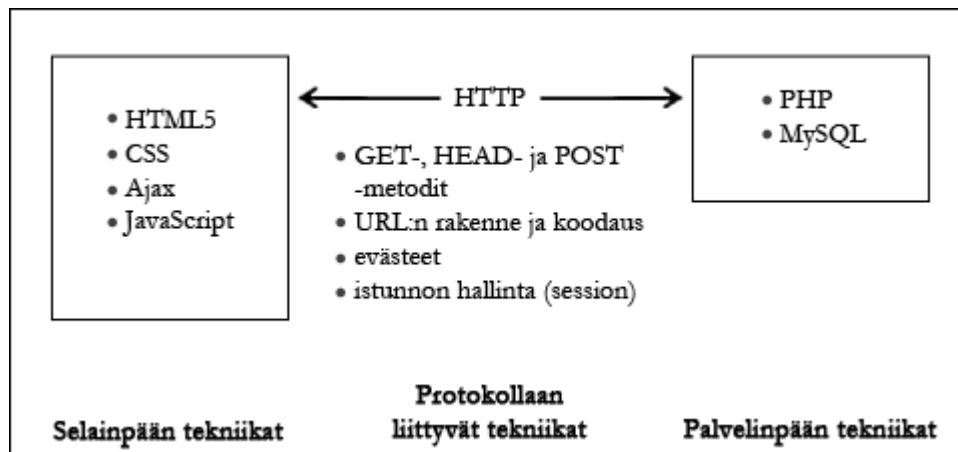
koskaan sisällä tietovaraston suoraa käsittelyä eikä malli sisällä HTML-merkkäusta. (Huhtamäki 2010, 10.)

2.5 MVC-arkkitehtuurin etuja ja haittoja

MVC-arkkitehtuuri mahdollistaa mallin uudelleenkäytön lähes kaikissa tilanteissa. Se onkin luonnollinen valinta graafisen käyttöliittymän toteutusmekanismiksi. MVC:n haittapuolena voidaan kuitenkin pitää sitä, että se tyypillisesti lisää luokkien määrää ja näin monimutkaistaa järjestelmää. Haitta voi muodostua myös näkymän ja ohjaimen suhteesta: näkymä- ja ohjainluokat ovat kiinteästi yhteydessä toisiinsa ja sen vuoksi niitä on vaikea käyttää uudelleen toisistaan irrallaan muissa yhteyksissä. Lisäksi tarkkailijoiden käyttö on mahdollinen tehottomuuden lähde: mallin yleisten operaatioiden käyttö voi olla hidasta ja samantapaisia kyselyitä joudutaan tekemään useita. (Koskimies & Mikkonen 2005, 144.)

3 Verkkopalvelun rakentaminen

Verkkopalvelun toiminta perustuu asiakas-palvelin-malliin. Tämä malli voidaan jakaa yleisellä tasolla kolmeen osaan: asiakasohjelmaan, palvelinohjelmaan ja yhteyskäyttöön eli protokollaan. Asiakasohjelma on tyypillisesti selain, joka lähettää palvelimelle pyyntöjä käyttäjän ohjaamana. Palvelinohjelma vastaa pyyntöön esimerkiksi lähettämällä selaimelle web-sivun katseltavaksi. Asiakkaan ja palvelimen välistä kommunikointia ohjaa HTTP-protokolla. (Heinisuo & Rauta 2007, 12; Rantala 2005, 2.) Edellä esitetyn mukaisesti verkkopalvelussa käytettävät tekniikat voidaan jakaa karkeasti selaimessa suoritettaviin tekniikoihin, palvelimessa suoritettaviin tekniikoihin ja protokollaan liittyviin tekniikoihin (Rantala 2005, 5). Alla oleva pelkistetty kuvio havainnollistaa tätä sisältäen tässä työssä käytetyt keskeiset tekniikat.



Kuvio 3. Verkkopalvelun asiakas-palvelin-malli (Rantala 2005, 2)

HTTP (Hyper Text Transfer Protocol) on yhteyskäytäntö, jolla verkkopalvelun käyttöösi liittymä selaimessa ja palvelimella oleva sovelluslogiikka viestivät keskenään (Rantala 2005, 8). Nimestään huolimatta HTTP:llä voidaan siirtää monenlaista dataa. Siirtotapahtuma koostuu kahdesta HTTP-viestistä: asiakaspyynnöstä ja palvelimen vastauksesta. (Rantala 2005, 99–100.) Seuraavassa kahdessa kappaleessa kuvataan HTTP-siirtotapahtuma.

Asiakas (selain) ottaa yhteyttä palvelimeen lähettäen sille pyyntöriivin, HTTP-otsakkeet ja mahdollisen datan. Pyyntöriivi kertoo käytettävän metodin. (Rantala 2005, 100.) GET-metodia käytetään datan pyytämisen eli resurssin hakuun tilanteissa, joissa ei

muuteta sovelluksen tilaa. Tiedot välitetään tällöin selaimen osoiterivin kautta. POST-metodia käytetään datan lähettämiseen eli tilanteissa, joissa ollaan muuttamassa sovelluksen tilaa, esimerkiksi lähetettäessä selaimen lomaketietoja tietokantaan. (Heinisuo & Rauta 2007, 223–224.) Metodina voi olla myös HEAD, jolloin palvelin ei lähetä vastauksessaan dataa, vaan ainoastaan tilarivin ja otsakkeet. Pyyntöriiviin sisältyy myös tieto pyydettyvästä dokumentista tai resurssista. Pyyntöriivin jälkeen asiakas lähettää otsakkeet, joissa se voi kertoa esimerkiksi nimensä, hyväksymänsä dokumenttityypit tai toivomansa kielen. Myös evästeet lähetetään otsaketiedoissa. Tämän jälkeen asiakas voi lähettää dataa, esimerkiksi HTML-lomakkeilla. Mahdollisen datan lähettämisen jälkeen pyyntö loppuu.

Palvelin vastaa asiakkaan pyyntöön. Palvelimen vastaus sisältää tilarivin, HTTP-otsakkeet ja datan. Tilarivi kertoo, onnistuiko pyyntö ja syyn mahdolliseen epäonnistumiseen. Otsakkeissa palvelin voi esimerkiksi kertoa vastauksen luontiajan, antaa tietoa itsestään, ilmoittaa palautettavan dokumentin sisältötyypin, asettaa evästeitä tai uudelleenohjata pyynnön. Pyyntö onnistuessa palvelin lähettää mahdollisesti halutun datan, joka voi olla esimerkiksi staattinen HTML-dokumentti, kuvatiedosto tai PHP:n generoima dokumentti. (Rantala 2005, 99–110.)

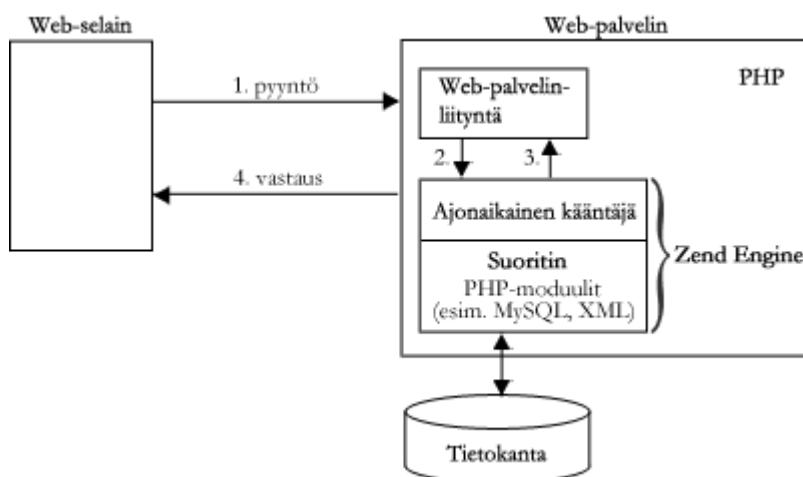
Verkkopalvelu rakentuu edellä kuvatuista peräkkäisistä HTTP-siirtotapahtumista. HTTP on tilaton protokolla, mikä tarkoittaa sitä, että jokainen pyyntö käsitellään erillisenä ilman tietoa edellisistä tai myöhemmistä pyynnöistä (Gilmore 2005, 381). Sen vuoksi tilatietoa täytyy ylläpitää HTML-lomakkeissa, hyperlinkeissä URL:n kyselyosassa, evästeissä tai istunnoissa (session). (Rantala 2005, 103–104; Heinisuo & Rauta 2007, 272–273.) Seuraavassa kahdessa luvussa kuvataan tarkemmin nämä ja muut kuvion 3 esittämät verkkopalvelun rakentamiseen liittyvät keskeiset tekniikat.

3.1 Palvelinpään tekniikat

PHP on yleinen avoimen lähdekoodin ohjelmointikieli, joka sopii erityisesti webkehitykseen. Kirjainyhdistelmä on rekursiivinen lyhenne sanoista ”PHP: Hypertext Preprocessor”. PHP on skriptikieli eli komentosarjakieli, jossa ohjelmakoodi tulkitaan palvelimella ohjelman suoritusvaiheessa. (The PHP Group 2013a; Rantala 2005, 9.)

PHP on erittäin laajalti käytetty. W3Techs:in (W3Techs 2013) säännöllisesti päivittyvien tilastojen mukaan lähes 80 % sivustoista, joiden kieli on heidän tiedossaan, käyttää PHP:tä.

PHP:n ensimmäisen version loi Rasmus Lerdorf vuonna 1995. Luomisen taustalla oli tarve seurata www-sivun kävijämääriä. Pieni kieli saavutti nopeasti suosiota ja sen ympärille kasvoi maailmanlaajuinen kehittäjien yhteisö. Tarpeiden lisääntyessä paranivat myös kielen ominaisuudet versio versiolta. Andi Gutmans ja Zeev Suraski uudistivat kieltä erittäin merkittävästi kirjoittamalla PHP:n jäsentimen kokonaan uudelleen: syntyi Zend Engine (tulee nimistä Zeev ja Andi), joka esiteltiin versiossa 4 vuonna 2000. PHP 5 julkaistiin heinäkuussa 2004 ytimenään Zend Engine 2. PHP 5 toi parannuksia erityisesti laajojen järjestelmien toteuttamiseen tarjoten kehittyneet oliopuurteet, poikkeusten käsittelyn ja paremman tietoturvan. (Gilmore 2005, 1–2; Rantala 2005, 9–14; Schafer 2005, 471–472.) PHP 5 on tätä kirjoitettaessa kielen viimeisin tuotantokäyttöön tarkoitettu pääversio (The PHP Group 2013b).



Kuvio 4. PHP:n toiminta (Rantala 2005, 14)

Laajasti ymmärrettynä PHP viittaa sekä varsinaiseen kieleen, että niihin teknisiin ratkaisuihin, joiden ansiosta PHP-kielisten ohjelmien suorittaminen on mahdollista. Voidaan puhua PHP-ympäristöstä. (Rantala 2005, 9.) Dynaamisen web-dokumentin muodostaminen käynnistyy tässä ympäristössä, kun web-palvelin huomaa selaimen pyynnön tarkenteesta (.php), että dokumentin PHP-ohjelmakoodia sisältävät osat pitää antaa käsiteltäväksi Zendin ajonaikaiselle kääntäjälle (Run-Time Compiler). Kääntämisen jälkeen

Zendin suoritin (Executor) suorittaa ohjelman ja käyttää tarvittavia PHP:n moduuleja, esim. MySQL-moduulia ottaessaan yhteyttä MySQL-tietokantaan. Suoritin palauttaa luomansa koosteen web-palvelinliittymälle, joka vuorostaan palauttaa dokumentin selaimelle käyttäjän tarkasteltavaksi. (Rantala 2005, 14.) PHP:llä on mahdollista tulostaa paitsi HTML:ää ja muita tekstitiedostoja, myös kuvia, PDF-tiedostaja ja Flash-esityksiä (The PHP Group 2013c).

PHP sisältää kaikki ohjelmointikielille tyypilliset ohjausrakenteet, muuttujat ja funktiot (Heinisuo & Rauta 2007, 13). PHP mahdollistaa myös olio-ohjelmoinnin, jonka idea on pakata muuttujien ja funktioiden muodostamia mielekkäitä kokonaisuuksia luokiksi (Heinisuo & Rauta 2007, 86). Olio-ohjelmoinnin etuna on koodin parempi organisointi, laajennettavuus ja uudelleenkäytettävyys. PHP 5:ssä olio-ominaisuudet on laajennettu muiden nykyaikaisten oliokielten tasolle. Oliomalli on lainattu monin osin Java-kielestä. (Rantala 2005, 64.) Seuraavassa käydään lyhyesti läpi PHP:n keskeiset olio-ohjelmoinnin piirteet.

Luokka määritellään *class*-avainsanalla. Luokassa on attribuutteja eli jäsenmuuttujia ja metodeita eli jäsenfunktioita. Perinnässä kantaluokasta voidaan *extends*-avainsanalla muodostaa uusi aliluokka, joka perii kantaluokan attribuutit ja metodit. PHP sisältää tyypilliset näkyvyysmääreet, joilla voidaan määritellä, mistä luokan sisältämiä metodeita (ja attribuutteja) voidaan käyttää: *public* (käytettävissä kaikkialta), *private* (käytettävissä vain siinä luokassa, jossa määritelty, ei aliluokissakaan) ja *protected* (käytettävissä siinä luokassa, jossa määritelty sekä kaikissa aliluokissa). (Rantala 2005, 64–68.)

Luokasta luodaan sen ilmentymiä eli olioita *new*-avainsanalla. Olion ominaisuuksia tulisi aina käsitellä metodien kautta. (Rantala 2005, 64–65.) Staattinen metodi eli luokkametodi poikkeaa normaaleista metodeista siinä, että sen kutsua ei osoiteta millekään luokan oliolle. Se on siis luokka- eikä oliokohtainen. Samoin luokkamuuttuja eli staattinen muuttuja on luokkakohtainen; sillä voi olla vain yksi arvo. (Rantala 2005, 68.) PHP:llä voidaan toteuttaa myös abstraktit luokat ja rajapintaluokat. Abstrakteista luokista ei voi sellaisenaan luoda olioita, vaan niistä periytetään uusia luokkia. Abstraktit luokat kuvaavat aliluokkiensa yhteisiä attribuutteja ja metodeja. Rajapintaluokka luettelee metodit,

joille tulee löytyä toteutus kyseisen rajapinnan toteuttavista luokista. (Rantala 2005, 69–70.)

PHP toteuttaa poikkeusten käsittelyn muiden ohjelmointikielien tapaan (The PHP Group 2013d). Poikkeuksien ajatuksena on helpottaa virheiden käsittelyä monimutkaisissa sovelluksissa. Poikkeusten käsittelyyn kuuluu kolme osaa: 1) poikkeus, joka heitetään virheen tapahtuessa; 2) try-lohko, jossa suoritetaan mahdollisesti virheen heittävä koodia ja 3) catch-lohko, jossa heitetty poikkeus otetaan kiinni, ja se voidaan käsitellä. (The PHP Group 2013d; Heinisuo & Rauta 2007, 93.)

Kuviossa 3 mainittu URL (Uniform Resource Locator) on osoiterakenne, jota käytetään viittaamaan mihin tahansa Internetissä olevaan resurssiin (Rantala 2005, 127). URL:n rakenne muodostuu siten, että ensin on protokollaosa (esim. http://), jonka jälkeen tulee palvelinosa (esim. www.treenikalenteri.net). Sitä seuraa kauttamerkillä (/) erotettuna hakupolku eli polku varsinaiseen resurssiin. Hakupolun jälkeen voi tulla vielä kysymysmerkillä erotettu ns. kyselyosa. (Rantala 2005, 127–128; Schafer 2005,4.)

Dataa voidaan välittää sovelluksen osalta toiselle hyperlinkkien ja niihin liittyvien URL:ien avulla. URL:ssa sallittujen merkkien määrä on kuitenkin rajoitettu, siksi useita merkkejä tulee koodata tiettyjen sääntöjen mukaan. (Rantala 2005, 127). PHP dekoodaa automaattisesti URL:n kyselyosuuden, jonka jälkeen nimi-arvo-parien arvot liitetään nimiä vastaaviin muuttujiin. Jos URL:n koodaamista tarvitaan enemmän, sitä varten voi käyttää PHP:n sisäänrakennettuja funktioita. (Rantala 2005, 133.)

PHP tarjoaa joukon sisäänrakennettuja taulukkomuotoisia muuttujia, jotka ovat käytävissä kaikkialta läpi koko suoritettavan ohjelmakoodin. Näitä kutsutaan superglobaaleiksi. Niihin talletettu tieto liittyy kuviossa 3 mainittuun HTTP-siirtotapahtumaan. Paljon käytettyjä superglobaaleja ovat `$_GET`, `$_POST`, `$_SERVER`, `$_COOKIE` ja `$_SESSION` (Mitchell ym. 2011, 90–91; The PHP Group 2013e.) Seuraavaksi selostetaan lyhyesti näiden merkitys.

Kuten sivulla 9 mainittiin, selaimen pyyntöriivi palvelimelle sisältää tiedon mm. käytävästä HTTP-metodista, joka on yleensä GET tai POST. `$_GET`-superglobaaliin talletuu GET-metodilla välitetyt parametrit (Gilmore 2005, 65; The PHP Group 2013f).

Nämä nimi-arvo-parit välitetään URL:n kyselyosassa. Jos esimerkiksi käyttäjä kirjoittaa osoiteriville `http://esimerkki.fi/?kirjain=a`, on taulukkomuuttujan `$_GET [”kirjain”]` arvo `a`. (The PHP Group 2013f.) POST-metodia käytetään tyypillisesti silloin, kun HTML-lomakkeilla lähetetään tietoa sovellukselle. `$_POST`-superglobaaliin tallentuu kaikki POST-metodilla välitetyt lomakkeen nimi-arvo-parit, ja niihin päästään kiinni `$_POST`-taulukon muuttujilla. (Gilmore 2005, 66; The PHP Group 2013g.)

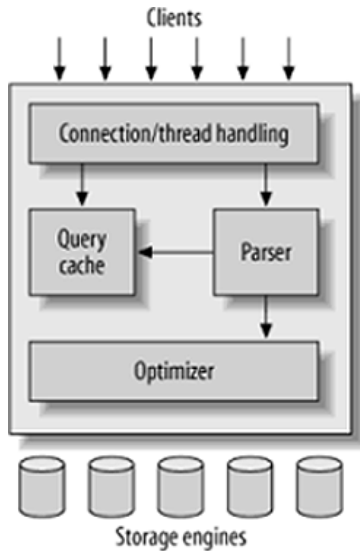
`$_SERVER` sisältää runsaasti palvelin- ja suoritussympäristöön liittyvää tietoa. Tämän taulukon merkinnät ovat web-palvelimen luomia ja niiden saatavuus riippuu palvelimesta. (Gilmore 2005, 65; The PHP Group 2013h.) `$_SERVER`-taulukkomuuttujilla on mahdollista lukea asiakaspyynnön http-otsakkeiden sisältämää tietoa. Esimerkiksi `$_SERVER[”REQUEST_METHOD”]` sisältää tiedon käytetystä siirtometodista. Muita käyttökelpoisia `$_SERVER`-taulukon sisältämiä tietoja ovat mm. asiakaskoneen IP-osoite, URL-osoitteen polkukomponentti (se osa URL-osoitteesta, joka on palvelinimen jälkeen) ja asiakaskoneen käyttäjäagentti (user agent), joka tarjoaa tietoa sekä käyttäjärjestelmästä, että selaimesta. (The PHP Group 2013h.) PHP:ssä on välineet myös otsaketietojen kirjoittamiseen. Palvelin voi lähettää otsaketietoja asiakkaalle PHP:n sisäänrakennetulla funktiolla `header()`. Tällä funktiolla voidaan suorittaa myös selaimen uudelleenohjaus. (Mitchell ym. 2011, 90–91.)

Evästeiden käyttäminen on yksi tapa tunnistaa verkkopalvelun käyttäjä. Eväste (cookie) on tiedosto, jonka selain tallentaa käyttäjän tietokoneelle. Evästeet ovat nimettyjä ja niihin voidaan tallentaa suhteellisen pieni määrä tekstimuotoista dataa. (Heinisuo & Rauta 2007, 271.) Evästeet luodaan PHP:n `setcookie()`-funktion avulla. Funktion pakolliset parametrit ovat evästeen nimi ja arvo, mutta sille voidaan antaa muitakin parametreja, kuten esimerkiksi evästeen vanhentumispäivämäärä. (Gilmore 2005, 66; The PHP Group 2013i.) Eväste lähetetään selaimelle palvelimen vastauksen HTTP-otsakkeissa. Selaimen uudessa pyynnössä palvelimella on nyt käytössään tuo yksilöivä eväste ja näin käyttäjä voidaan tunnistaa. (Heinisuo & Rauta 2007, 271.) `$_COOKIE`-superglobaali tallentaa tietoa, joka on välitetty suoritettavalle ohjelmakoodille HTTP-evästeiden kautta (Gilmore 2005, 66; The PHP Group 2013j). Jos on esimerkiksi asetettu eväste komennolla `setcookie (”nimi”, ”Maija”)`, silloin muuttujan `$_COOKIE [”nimi”]` arvo on `Maija` (The PHP Group 2013i).

PHP:n istunnot (session) ovat keino säilyttää käyttäjäkohtaisia tietoja sivunlatauksesta eli HTTP-siirtotapahtumasta toiseen. Istunnot käyttävät sisäisesti evästeitä käyttäjän tunnistamiseen. Kun istunto aloitetaan komennon `session_start()` avulla, käyttäjän selaimelle lähetetään eväste, jonka nimi on PHPSESSID. Evästeen arvo on satunnainen jono kirjaimia ja numeroita. Palvelimella on tuota merkkijonoa vastaava tiedosto, johon tallennetaan istunnon aikana luodut tiedot nimi-arvo-pareina. Näihin tietoihin pääsee käsiksi `$_SESSION`-superglobaalin avulla. (Heinisuo & Rauta 2007, 273; The PHP Group 2013k.) Esimerkiksi komento `$_SESSION [”testi”] = 42` tallentaa `$_SESSION [”testi”]` -muuttujan arvoksi 42 (The PHP Group 2013k). Edellisessä kappaleessa mainittujen evästeiden tieto säilytetään verkkopalvelun käyttäjän tietokoneella. Istunnon varsinaiset tiedot taas säilytetään turvallisemmin palvelimella. Tähän tietoon vain viitataan satunnaisen tunnisteen avulla, joka on tallennettu evästeessä. (Heinisuo & Rauta 2007, 272.)

MySQL on relaatiotietokantoja käsittelevä tietokannanhallintajärjestelmä (Heinisuo & Rauta 2007, 37). Relaatiotietokannoissa todellisuutta mallintava tieto esitetään tauluina. Kussakin taulussa on tunnisteenä uniikki perusavain (primary key) sekä viiteavain (foreign key), joka viittaa toisen taulun perusavaimeen. Taulujen välille luodaan näin suhteita eli relaatioita yhdistämällä perusavain toisen taulun viiteavaimeen. (Hovi, Huotari & Lahdenmäki 2005, 8–9.) MySQL on ilmainen ja hyvin sorituskykyinen tietokantaohjelma ja sitä voidaan käyttää useista ohjelmointikielistä (Heinisuo & Rauta 2007, 38). Schwartzin, Zaitsevin ja Tkachenkon (2012,1) mukaan MySQL soveltuu hyvin toimimaan sangen vaativissa ympäristöissä, kuten web-sovelluksissa.

MySQL käyttää asiakas-palvelin-arkkitehtuuria. Tämä tarkoittaa sitä, että MySQL toimii palvelinkoneella omana erillisenä sovelluksenaan, mutta myös palvelinohjelmana toisille sovelluksille. Kuten aikaisemmin kuviossa 4 esitettiin, tietokantaa ei käsitellä suoraan PHP:stä käsin, vaan PHP:ssä otetaan yhteys MySQL-tietokantapalvelimeen. PHP lähettää tietokantapalvelimelle SQL-kielisiä komentoja ja palvelin vastaa niihin esimerkiksi lähettämällä sovellukselle tietoa tietokannasta. SQL (Structured Query Language) on standardoitu relaatiotietokantojen kysely- ja määrittelykieli, mutta MySQL ei noudata sen normeja peruskomentoja lukuun ottamatta kovin täydellisesti. (Heinisuo & Rauta 2007, 39–40; 97.)



Kuvio 5. MySQL-palvelimen arkkitehtuuri (Schwartz, Zaitsev & Tkachenko 2012, 2)

MySQL:n arkkitehtuurin ylin taso sisältää palvelimelle tyypilliset palvelut, kuten yhteydenhallinnan, autentikoinnin ja turvallisuudesta vastaavat palvelut. Toinen taso on MySQL:n varsinaiset aivot. Tällä tasolla tapahtuu mm. kyselyiden jäsentäminen, erittely, optimointi ja välimuistin käyttö. Myös kaikki sisäänrakennetut funktiot ja varastointimoottoreiden toiminnallisuus ovat tällä tasolla. Kolmannella tasolla ovat itse varastointimoottorit. Ne ovat vastuussa datan säilyttämisestä ja palauttamisesta. Ne eivät kommunikoi keskenään eivätkä jäsennä SQL-lauseita, vaan ainoastaan vastaavat palvelimelta tuleviin pyyntöihin. (Schwartz, Zaitsev & Tkachenko 2012, 1–2.)

MySQL:n merkittävin piirre on sen muista tietokannoista poikkeava arkkitehtuuri, joka erottaa kyselyiden prosessoinnin ja muut palvelintehtävät varastointimoottoreista eli datan säilyttämisestä ja palauttamisesta. (Schwartz, Zaitsev & Tkachenko 2012, 1–2.) Tämä merkitsee sitä, että erityistilanteissa tietokannan jokaiselle taululle voidaan valita toimintojen ja erityispiirteiden puolesta sopivin varastointimoottori (Gilmore 2005, 565–566; MySQL 5.6 Reference Manual 2013). MySQL:n versiosta 5.5.5 lähtien sen oletusvarastointimoottorina on InnoDB. Sitä suositellaan käytettäväksi yleisesti kaikissa tauluissa, jollei ole jotain painavaa syytä valita toisin. InnoDB tukee transaktioita. (MySQL 5.5 Reference Manual 2013; Schwartz, Zaitsev & Tkachenko 2012, 15.) Transaktio on joukko tehtäviä, joita pidetään yhtenä työyksikkönä. Jos kaikki tehtävät onnistuvat, silloin tietokannan taulujen muutokset suoritetaan. Jos taas yksikin tehtävä epäonnistuu, silloin kaikki edeltävät tehtävät perutaan eikä muutoksia tulla tallenta-

maan. (Gilmore 2005, 565.) InnoDB suojaa dataa myös toipumalla automaattisesti mahdollisista kaatumisista (automatic crash recovery). Lisäksi InnoDB pakottaa taulujen väliseen viihte-eheyteen. (MySQL 5.6 Reference Manual 2013.)

MySQL:n versioon 5.5.5 asti sen oletustallennusmoottorina on MyISAM, joka toimii kaikissa MySQL:n asennuksissa, mutta ei tue transaktioita. (MySQL 5.5 Reference Manual 2013.) MyISAM on kompakti ja suunnittelultaan yksinkertainen. Siksi se voi olla suorituskykyinen ja kustannustehokas vaihtoehto joidenkin taulujen tallennusmoottoriksi. (Schwartz, Zaitsev & Tkachenko 2012, 19.)

PDO (PHP Data Objects) on tietoturvallinen tapa käyttää tietokantaa (Heinisuo & Rauta 2007, 401–402). PDO määrittelee kevyen ja yhtenäisen rajapinnan tietokantojen käsittelyä varten. Se käyttää erityistä tietokantakohtaista ajuria saadakseen yhteyden kyseiseen tietokantaan ja käyttääkseen sen ominaisuuksia. PDO:n avulla voidaan käyttää samoja metodeita kyselyiden tekemiseen ja datan noutamiseen riippumatta siitä, mikä tietokanta on kyseessä. (The PHP Group 2013L.) PDO on siis hyvin tietokantariippumaton tapa käyttää tietokantoja. Täysin riippumaton se ei kuitenkaan ole, sillä jotkin SQL-lauseet saattavat vaatia sopeuttamista käytettyyn tietokantajärjestelmään. (Heinisuo & Rauta 2007, 274.)

PDO:ssa on mahdollista käyttää tietokantahauissa ns. valmisteltuja lausekkeita. Tämä tarkoittaa sitä, että ensiksi luodaan SQL-lauseke, jossa ei ole mukana mitään tiettyjä arvoja, vaan kaksoispisteellä alkavat apumerkinnät. Tästä lausekkeesta tehdään valmisteltu lauseke siihen tarkoitetulla PDO:n metodilla. Valmistellun lausekkeen kaksoispisteellä merkittyihin kohtiin voidaan sijoittaa toisen metodin avulla varsinaiset arvot. (Heinisuo & Rauta 2007, 274; Mitchell ym. 2011, 49–51.) Edellä kuvattu menettely on tietoturvallinen tapa syöttää arvoja tietokantaan, sillä silloin syöte puhdistuu automaattisesti (Heinisuo & Rauta 2007, 401–402; Mitchell ym. 2011, 49). Toinen hyöty on se, että valmisteltuja lausekkeita voidaan käyttää uudelleen saman PHP-tiedoston ajon aikana (Heinisuo & Rauta 2007, 274–275).

3.2 Selainpään tekniikat

HTML (Hypertext Markup Language) on kieli, jonka tehtävä on kuvata selaimen palvelimelta pyytämän dokumentin rakenne. Tätä varten HTML sisältää lukuisia elementti- ja attribuuttimäärittäjiä ja mm. hypertekstilinkkien ja sivuille upotettavien skriptien rajapintamäärittäjiä. HTML on tarkoitettu käytettäväksi yhdessä muiden web-tekniikoiden kanssa. Huomattava osa sen hyödyllisyydestä perustuu sen kykyyn liittää yhteen erityyppisiä Internet-palveluita ja ohjelmistoja. HTML-dokumentilla ei ole täsmällistä ulkoasua, vaan siihen merkataan ensisijaisesti dokumentin eri osien merkitys. (Peltomäki & Nykänen 2006, 9–10.)

HTML:n käytön perinteisiä pulmia on ollut standardista lipsuminen. Tämä tarkoittaa paitsi kehittäjien huolimattonta merkkausta, myös selainten rönsyilevää HTML-implementointia. (Peltomäki & Nykänen 2006,10.) HTML:n uusin versio, HTML5, saikin alkunsa ajatuksesta, että selainten valmistajat kehittävät yhteistyössä HTML:ää ja selaimia rinnakkain. (Korpela 2011, 24). Vuonna 2008 julkistettiin ensimmäinen W3C:n (World Wide Web Consortium) HTML5-luonnos. HTML5:n kehittäminen jatkuu edelleen W3C:ssä ja WHATWG:ssä (Web Hypertext Applications Technology Working Group). (Korpela 2011, 58.) HTML5 sisältää joukon uusia ominaisuuksia, joita voi hyödyntää verkkopalvelun selainpään tekniikoissa. Kaikki merkittävät selaimet (Chrome, Firefox, Internet Explorer, Safari ja Opera) jatkavat HTML5:n ominaisuuksien lisäämistä uusimpiin versioihinsa. (W3schools.com 2013a.)

HTML5 määrittelee dokumentit ensisijaisesti olioina ja sen pohjalta niiden esityksen merkkauksielellä (Korpela 2011, 286.) Dokumenttioliomalli eli DOM (Document Object Model) tarkoittaa kuvausta siitä, miten dokumentti ymmärretään rakenteiseksi olioksi, jolla on ominaisuuksia. (Korpela 2011, 42). DOM on W3C:n luoma standardi ja se määrittelee ohjelmointikieliriippumattoman ohjelmointirajapinnan HTML-dokumentteihin (Peltomäki & Nykänen 2006, 98). Jokaisella merkatulla elementillä HTML:ssä on vastineensa DOM:ssa. DOM:ssa tämä elementti on olio, jolla on nimettyjä ominaisuuksia ja suhteita muihin olioihin (vanhempi-, lapsi- ja sisäsuhteet). Käytännössä on merkittävää se, että DOM on standardoitu rajapinta, jonka kautta JavaScriptillä voi muuttaa, poistaa ja lisätä HTML-dokumentin elementtejä. (Schafer 2005,

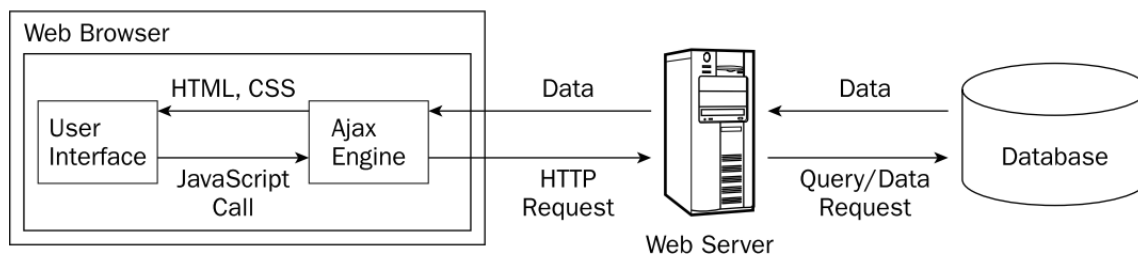
291; W3schools.com 2013b.) DOM tarjoaa JavaScriptille myös tavan rekisteröidä erilaisia HTML-elementteihin liittyviä tapahtumankäsittelijöitä, mikä mahdollistaa vuorovai-
kutuksen luomisen verkkopalvelun käyttöliittymään. Tapahtuma on esimerkiksi se, kun
käyttäjä klikkaa jotain elementtiä tai painaa jotain näppäintä. (W3schools.com 2013c.)

HTML esittää dokumentin rakenteen ja sisällön, mutta esitystapa toteutetaan CSS-
tyyleillä. CSS (Cascading Style Sheets) -tyyliohje koostuu säännöistä, joilla voidaan halli-
ta HTML-dokumentin eri elementtien esittämiseen ja ulkoasuun liittyviä piirteitä.
CSS tarkoittaa laskeutuvaa (cascading) tyylimallia, jossa elementtien tyylien määrittelys-
sä otetaan huomioon aiemmat tyylimäärittelyt. Tällöin määrittelyssä laskeudutaan yleis-
estä tapauksesta yksityiseen tapaukseen. (Peltomäki & Nykänen 2006, 260–261.) Tyy-
liohjeet voidaan liittää HTML5-dokumenttiin erillisenä tiedostona (.css) link-
elementillä, mikä on joustava käytäntö, koska se tarjoaa mahdollisuuden muokata hel-
posti HTML-dokumentin ulkoasua. (Korpela 2011, 46; W3schools.com 2013d.)

JavaScript on muodostunut lähes ainoaksi kieleksi, jolla ohjelmoidaan selain suoritta-
maan haluttuja toimintoja (Korpela 2011, 47). Flanaganin (2011,1) mukaan JavaScript
on vakaa ja tehokas kieli, jota käyttää ylivoimainen enemmistö nykyaikaisista web-
sivustoista. JavaScript sisältää kaikki ohjelmointikielille tyypilliset piirteet ja se tukee
myös olioita (W3schools.com 2013e). DOM-malli antaa JavaScriptille tavan viitata ele-
mentteihin. Elementtien ominaisuuksiin taas voi viitata JavaScriptissä pistemerkin-
nällä, joka on muotoa *elementti.ominaisuus*. DOM-mallin mukaisesti HTML-elementillä on me-
todeina joukko valmiiksi määriteltäviä JavaScript-funktioita. Näitä metodeja voi käyttää
niin ikään pistemerkin-
nällä, joka on muotoa *elementti.metodi*. (Korpela 2011, 48–51.) Ja-
vaScriptin tuki selaimissa on erittäin laaja, mutta yksityiskohdissa on eroja (Korpela
2011, 47). Näitä peittämään on kehitetty lukuisia JavaScript-kirjastoja, joista yksi laajalti
käytetty on jQuery. Se helpottaa HTML-dokumentin elementtien käsittelyä ja tarjoaa
työkalut myös Ajaxin käyttämiseen. (Flanagan 2011, 523.)

Ajax (Asynchronous JavaScript and XML) tarkoittaa joukkoa tekniikoita, joilla web-
sovellus voi hakea tietoa palvelimelta taustatoimintona. (Korpela 2011, 295). Ilman
Ajax-tekniikkaa on ladattava uudelleen koko web-sivu aina, kun halutaan sisällön päi-
vittyvän, mikä voi olla hidasta. Ajax mahdollistaa sivun päivittämisen asynkronisesti

siten, että päivitetään vain tarvittavat osat sivusta. (Peltomäki & Nykänen 2006, 296; W3schools.com 2013f.) Alla oleva kuvio havainnollistaa Ajaxin toimintaa.



Kuvio 6. Ajaxin toiminta (Zakas, McPeak & Fawcett 2007, 5)

Selain käynnistää Ajax-tapahtuman kutsumalla jotain tapahtumankäsittelijää. Sen jälkeen luodaan JavaScriptillä XMLHttpRequest-olio, joka on Ajaxin moottori (kuvassa Ajax Engine): se vastaa datan asynkronisesta vaihdosta palvelimen kanssa soveltuvine metodeineen ja attribuutteineen. XMLHttpRequest-olio alustaa yhteyden palvelimeen, jonka jälkeen se lähettää palvelimelle HTTP-pyyntöä. JavaScriptin ei tarvitse odottaa palvelimen vastausta (asynkroninen pyyntö), vaan se voi suorittaa sillä aikaa muuta koodia. Palvelin tekee tarvittavat toiminnot (usein ottaa yhteyden tietokantaan) ja palauttaa vastauksen selaimelle. Selainpäässä XMLHttpRequest-olio tarkistaa pyynnön tilan ja sen, onko kaikki data saapunut. Kun pyyntö on valmis ja kaikki data saapunut, JavaScriptin ja DOM:n avulla voidaan käsitellä data XMLHttpRequest-oliosta. Lopuksi tarvittavat elementit HTML-dokumentista päivitetään. (Peltomäki & Nykänen 2006, 300–303; W3schools.com 2013f; Zakas, McPeak & Fawcett 2007, 5–7.)

3.3 Verkkopalvelun tietoturva

Tietoturvallisuuden klassiseen määritelmään kuuluvat tiedon luottamuksellisuus, käytettävyys ja eheys. Luottamuksellisuus tarkoittaa sitä, että järjestelmän tiedot ovat vain niihin oikeutettujen henkilöiden käytettävissä. Tiedon luottamuksellisuus pyritään saavuttamaan suojaamalla järjestelmän laitteet ja tietovarastot tunnuksin ja salasanoihin sekä käyttämällä salakirjoitusmenetelmiä erityisen arvokkaiden tietojen kohdalla. Käytettävyys taas merkitsee sitä, että tiedot on mahdollista saada järjestelmästä riittävän nopeasti ja oikeassa muodossa. Tähän pyritään oikeilla laite- ja ohjelmistoratkaisuilla sekä tiedon jalostuksen hyvällä automatisoinnilla. Eheydellä tarkoitetaan sitä, että järjestelmän

tiedot pitävät paikkansa eivätkä sisällä tahallisia tai tahattomia virheitä. Tiedon eheys saavutetaan lähinnä ohjelmointiteknisillä ratkaisuilla: sovelluksiin ohjelmoidaan syötteiden tarkistuksia, syöttörajoitteita sekä tallennus- ja tiedonsiirto-operaatioiden varmistuksia. (Hakala, Vainio & Vuorinen 2006, 4-5.)

Tietojen syötön aikana tapahtuvat inhimilliset virheet aiheuttavat valtaosan tietojärjestelmien virheellisistä tiedoista. Ohjelmoijan pitäisi aina varautua näihin virheisiin ja syötteiden tarkistuksiin. (Hakala ym. 2006, 320.) Inhimillisiä virheitä voidaan ehkäistä pitämällä käyttöliittymä mahdollisimman yksinkertaisena. Käyttäjän kannalta vain tarpeelliset ohjauselementit kannattaa pitää näkyvissä. Käyttöliittymä antaa virhetilanteissa selkeän ilmoituksen ja mahdollistaa virheen korjaamisen tai toiminnon hallitun keskeyttämisen. Sovelluksen on aina tarkistettava, onko kaikki tarpeelliset tiedot syötetty ennen tietojen käsittelyä. Sen jälkeen sovellus käyttää syötteen oikeellisuuden ja täydellisyyden tarkistamiseksi soveltuvia merkki-, raja-arvo ja tarkistusrutiineja, esimerkiksi säännöllisiä lausekkeita. (Hakala ym. 2006, 321–323.)

Verkkopalvelussa käyttäjän syötteet käsittävät kaiken selaimen lähettämän tiedon: otsakkeet, URL-osoitteet ja varsinaisten sivujen kautta lähetettävät tiedot. (CERT-FI 2008; Heinisuo & Rauta 2007, 400; Mitchell ym. 2011, 174.) PHP:ssä syötteeksi voidaan laskea `$_GET`-, `$_POST`- ja `$_COOKIE`-taulukot, kuten myös niiden kaikkien tiedot sisältävä `$_REQUEST`-taulukko. Näiden taulukoiden sisältämät arvot ovat vääristeltävissä, joten niihin pitää suhtautua sen mukaisesti. Selainpään rajoituksiin ja tarkistuksiin ei pidä luottaa, vaan mainittujen superglobaalien tiedot tulee aina tarkistaa PHP-koodissa ennen kuin niitä päästetään järjestelmään. (Heinisuo & Rauta 2007, 399–403.) Verkkopalvelussa käyttäjän syötettä voi tulla joskus yllättävältäkin taholta. On hyvä huomata, että myös PHP:n `$_SERVER["HTTP_HOST"]` superglobaali voi sisältää käyttäjän manipuloimaa dataa. Tuon muuttujan arvona on palvelimelle tulevan pyynnön host-otsakkeen sisältö, jonka tarjoaa selain eli viime kädessä käyttäjä. (Mitchell ym. 2011, 174.)

Kun tarkastellaan palvelimen vastausta tietoturvan kannalta, on hyvä huomata, että kaikki sisältö mitä palvelin lähettää selaimelle, on käyttäjän luettavissa ja mahdollisesti väärinkäytettävissä. On harkittava tarkkaan, mitä tietoa antaa sovelluksen tulostaa.

(Heinisuo & Rauta 2007, 402–404.) Tämän laiminlyömisestä seuraa kahdentasoisia haittoja. Toisaalta voidaan paljastaa sovelluksen arkkitehtuurista tai teknologiasta jotain sellaista, joka auttaa hyökkääjää. Hyökkääjä voi saada tietoa virheilmoituksista tai tutkimalla sivun lähdekoodia. (Lehtonen & Siljander 2010, 7; Heinisuo & Rauta 2007, 404.) Toisaalta voidaan suoraan altistaa sivu hyökkäyksille, jos sivulle tulostuvaa dataa ei käsitellä siinä mahdollisesti olevan HTML-merkkauksen varalta. (Mitchell ym. 2011, 176–178). Ongelman ydin on siinä, että palvelimet, jotka luovat dynaamisia sivuja, eivät pysty täysin hallitsemaan sitä, miten selain tulkitsee tulosteen ja jos data esitetään HTML:n seassa, voi hyökkääjä sisällyttää haitallista koodia sivulle HTML-merkkauksen avulla. (CERT.) Tällöin sivu ei ole validia HTML:ää, mutta saattaa toimia hyökkääjän haluamalla tavalla (Mitchell ym. 2011, 177). Dynaamisesti tulostettava data tulee aina siivota kaikesta HTML-merkkauksesta ja erikoismerkeistä. Ohjelmoitaessa kannattaa myös eksplisiittisesti merkitä joka sivulle käytettävä merkistököödaus, jotta voidaan olla varmoja siitä, mitkä merkit ovat erikoismerkkejä. (CERT.)

WWW:ssä toimivan verkkopalvelun tietoturvaan kohdistuu erityisiä haasteita, koska sovellus on koko Internetin nähtävillä ja periaatteessa kuka tahansa voi yrittää käyttää palvelua väärin tarkoituksiin. Heikkoudet palvelun tietoturvassa voivat myös paljastaa luottamuksellista tietoa ulkopuolisille. (Heinisuo & Rauta 2007, 399.) Luottamuksellisuus, käytettävyys ja eheys eivät siis ole vaarassa ainoastaan ihmisten erehdysten vuoksi, vaan myös aktiivisen hyökkääjän toimenpiteiden vuoksi. Seuraavassa kartoitetaankin vakavimpia uhkia, joita kohdistuu verkkopalvelun tietoturvaan ja käsitellään niitä ohjelmoijan kannalta.

Brittiläisen pilvihostauspalveluja tarjoavan FireHost-yhtiön uhkatilastot vuoden 2012 loppupuolelta käsittävät yli 64 miljoonaa torjuttua hyökkäystä USA:ssa ja Euroopassa. Yhtiön mukaan vuoden 2012 viimeisellä neljänneksellä Cross-Site Scripting (XSS) ja SQL-injektio pysyivät merkittävimpinä hyökkäyksinä. Muina merkittävinä hyökkäystyypeinä mainitaan Cross-Site Request Forgery (CSRF) ja Directory Traversals. (SC Magazine UK 2013.) The Next Web (2012) tutki FireHost-yhtiön tilastoja aiemmin samalta vuodelta mainiten samat neljä hyökkäystyyppiä kaikkein vaarallisimmiksi hyökkäyksiksi. Myös tietoturvayhtiö Imperva (2012) mainitsee raportissaan edellä mainitut hyökkäykset CSRF:ää lukuun ottamatta. Open Web Application Security Project

(OWASP) on maailmanlaajuinen voittoa tavoittelematon tietoturva- ja tietosuojaprojekti (OWASP 2013). OWASP (2010) kartoitti kymmenen kriittisintä turvallisuusriskiä, jotka kohdistuvat web-sovelluksiin ja viiden ensimmäisen joukossa olivat edellä mainitut SQL-injektio, XSS ja CSRF. Muita raportoituja verkkopalvelun ohjelmoijan kannalta merkittäviä uhkia ovat Broken Authentication and Session Management sekä Insecure Direct Object References (OWASP 2010a).

SQL-injektiossa hyökkääjä lähettää epäluotettavaa dataa osana hakulausetta. Järjestelmä ei osaa tunnistaa vihamielistä koodia, vaan suorittaa sen ja tämän jälkeen hyökkääjä pääsee käsiksi luottamukselliseen tietoon. (OWASP 2010a.) SQL-injektio perustuu SQL:n kommenttimerkkien hyväksikäyttöön siten, että hyökkääjä pääsee muokkaamaan kyselyä haitallisella tavalla. Kommentit katkaisevat lauseen, jolloin loppuosa siitä jää suorittamatta. SQL-injektio on erittäin tyypillinen esimerkki siitä, että syöte jätetään tarkistamatta. Tehokas keino suojautua sitä vastaan on käyttää PDO:n valmisteltuja lausekkeita yhdessä kaksoispisteellä alkavien parametrien kanssa, jolloin tietokantaan menevät arvot puhdistuvat automaattisesti. (Heinisuo & Rauta 2007; Mitchell ym. 2011, 190–191.)

Cross site scripting (XSS) on hyökkäystyyppi, jonka perussyynä on se, että verkkopalvelu ei tarkista kunnolla käyttäjän syötteitä eikä niiden pohjalta tehtäviä tulosteita. (Imperva 2012, 6; OWASP 2010a.) XSS voidaan jakaa tallennettuihin, heijastettuihin ja DOM-pohjaisiin hyökkäyksiin. (OWASP 2010a.) Tallennetuissa hyökkäyksissä haavoittuvalle sivustolle injektoidaan osoiterivin tai lomakkeiden kautta haitallinen koodi ja se tallennetaan palvelimelle esimerkiksi tietokantaan tai keskustelupalstalle. Vierailijan käydessä sivulla koodi ajetaan, koska se tulee selaimen kannalta luotettavalta taholta. (Lehtonen & Siljander 2010, 10–11; OWASP 2011a.) Heijastetuissa hyökkäyksissä uhri houkutellessaan painamaan hyökkääjän manipuloimaa linkkiä tai käyttämään lomaketta esimerkiksi sähköpostiviestissä tai toisella sivustolla. Uhri tulee tätä kautta haavoittuvalle sivustolle, joka heijastaa haittakoodin takaisin uhrin selaimeen. Selaimen kannalta koodi tulee luotettavalta taholta ja koodi suoritetaan. (Lehtonen & Siljander 2010, 11–12; Mitchell ym. 2011, 176–178; OWASP 2011a.) DOM-pohjaisissa hyökkäyksissä ohjelmointivirheet ovat sivun omassa asiakaspuolen koodissa. Sivu itse ei muutu, mutta sen sisältämä asiakaspuolen koodi ajetaan haitallisella tavalla, koska DOM-ympäristöä

on manipuloitu. Muuten nämä hyökkäykset toimivat samoin kuten heijastetut hyökkäykset. (OWASP 2011b.) XSS-hyökkäystä voidaan käyttää evästeiden varastamiseen ja sitä kautta istuntojen kaappaamiseen ja käyttäjän tunnistetietojen saamiseen. Sillä voidaan vahingoittaa ja muunnella sivustoa tai ohjata toisille vihamielisille sivustoille. (OWASP 2010a; Lehtonen & Siljander 2010, 12; Imperva 2012, 6.) Koska XSS:n perussyynä on syötteiden ja tulosteiden puutteellinen tarkistaminen, on ensisijainen suojauskeino niiden kunnollinen tarkistaminen. XSS-hyökkäyksiä voidaan ehkäistä siivoamalla tulosteiden HTML-kontekstissa esiintyvä data. Vastaavasti myös syötteistä tulee siivota HTML-merkkaukset ja muut erikoismerkit. (OWASP 2010a.)

Cross-Site Request Forgery (CSRF) on hyökkäys, jonka tarkoitus on saada uhri tekemään väärennetty HTTP-pyyntö tietylle sivustolle ja hyötyä sinne perustetusta uhrin identiteetistä. (Mitchell ym. 2011, 180). CSRF toteutetaan asettamalla haavoittuvalle sivustolle HTML-merkkaukset, jonka avulla pyynnöt toteutetaan (Lehtonen & Siljander 2010, 14). Pyyntö sisältävät istuntoevästeen ja kaiken muun autentikointiin tarvittavan informaation, joka liitetään automaattisesti mukaan. Tämä mahdollistaa sen, että haavoittuva sivusto tulkitsee uhrilta tulevat pyynnöt täysin sallituiksi. Näin käyttäjä saadaan suorittamaan hyökkääjän haluamia toimintoja. (OWASP 2010a.) CSRF perustuu siihen, että hyökkääjä on päässyt käsiksi uhrin evästeisiin, usein edellä mainitun XSS:n avulla. Tehokas tapa välttää CSRF on estää ensikädessä XSS-hyökkäys. CSRF voidaan torjua myös liittämällä jokaiseen istuntoon ja jokaisen GET- ja POST pyynnön mukaan salattu satunnainen tokeni, jonka avulla pyynnöt voidaan yksilöidä ja varmistaa, että ne kuuluvat oikeaan istuntoon. (Lehtonen & Siljander 2010, 14–15; OWASP 2010a.) OWASP (2012) suosittaa tokenin liittämistä HTML-dokumentin lomakkeen piilokenttään. Tokeni tallennetaan istuntomuuttujaan ja sitä verrataan joka kerta pyynnön mukana tulevaan arvoon. Jos arvot poikkeavat toisista, voidaan epäillä CSRF -hyökkäystä. (OWASP 2012.)

Directory Traversal tarkoittaa tietoturvauhkaa, jossa hyökkääjän HTTP -pyynnöt saavat palvelinohjelmiston paljastamaan kielletyissä hakemistoissa olevaa dataa. Hyökkääjä pääsee myös suorittamaan haluamia haitallisia komentoja palvelimella. Directory Traversal -haavoittuvuus voi sijaita joko itse palvelimella tai palvelimella suoritettavan sovelluksen koodissa. Verkkopalvelun ohjelmoijan kannalta on olennaista huomioida

jälkimmäinen seikka. Tällöin on taas huomattava se seikka, että dynaamiset web-sivut saavat monenlaista syötettä käyttäjältä, jonka kunnolliseen tarkistamiseen täytyy kiinnittää erityistä huomiota. (Imperva 2013.)

OWASP:n raportoima Broken Authentication and Session Management on laaja-alainen uhka, joka aiheutuu toisaalta verkkopalvelun puutteellisesta autentikoinnista ja toisaalta turvattomasta istuntojen käsittelystä. Se voi johtaa tilanteeseen, jossa hyökkääjä pääsee murtautumaan järjestelmään tai saa haltuunsa käyttäjään liittyvää tunnistetietoa. (OWASP 2010a.) Tätä riskiä voidaan ehkäistä toisaalta vahvoilla salasanoilla, jotka ovat riittävän pitkiä ja sisältävät kirjainten lisäksi numeroita ja erikoismerkkejä. Kirjautumisyritysten lukumäärää voidaan rajoittaa, mikä pienentää hyökkääjän mahdollisuuksia murtautumiseen. Käyttäjille pitää tarjota toimiva mekanismi salasanojen vaihtamiseen ja salasanat tulee tallentaa tietovarastoon turvallisesti salattuina. Autentikointidataa ei tule koskaan lähettää GET-metodilla, vaan aina tulee käyttää POST-metodia. Toisaalta taas täytyy kiinnittää huomiota istuntodatan turvallisuuteen. Kun käyttäjä on kirjautunut järjestelmään, tulee istuntotunnus säilyttää turvallisesti. Sitä ei saa koskaan liittää osaksi URL:ää. (OWASP 2010b.) Istuntotunnusten turvallisuuteen liittyy hyökkäys, joka tunnetaan myös nimellä session hijacking. Se tarkoittaa istuntotunnuksen paljastamista esimerkiksi arvattavuuden vuoksi, HTTP-siirron salakuuntelun avulla tai XSS:n avulla. (OWASP 2011c.) Kun hyökkääjä on saanut tämän tiedon käsiinsä, voi seuraava vaihe olla edellä mainittu CSRF -hyökkäys. Kuten sen yhteydessä mainittiin, voidaan turvallisuutta lisätä pienentämällä tunnusten ennustettavuutta. PHP:ssä on mekanismi (session_regenerate_id) istuntotunnuksien vaihtamiseen istunnon aikana, millä voidaan vaikeuttaa potentiaalisen hyökkääjän toimia. (The PHP Group 2013n). Lisäksi täytyy muistaa, että istuntodata on aina tuhottava palvelimelta istunnon jälkeen (Heinisuo & Rauta 2007, 284).

Insecure Direct Object References tarkoittaa turvatonta viittausta johonkin järjestelmän sisäiseen objektiin, joka voi olla esimerkiksi tiedosto, hakemisto tai tietokannan avain. Tämä voi johtaa tilanteeseen, jossa hyökkääjä pääsee muokkaamaan viittauksen kohdetta omiin tarpeisiinsa sopivaksi. Tätä vastaan voi suojautua välttämällä suorita viittauksia objekteihin ja käyttäjä sijasta voi käyttää mapattuja epäsuoria viittauksia. Olen-

naista on, että aina kun suoria viittauksia järjestelmän resursseihin käytetään, tarkistetaan myös, että käyttäjällä on oikeus kyseiseen resurssiin. (OWASP 2010a.)

Ohjelmoinnin kannalta verkkopalvelun tietoturva voitaisiin kiteyttää FIEO-periaatteeksi: Filter Input, Escape Output. Samaa yleistä lähestymistapaa pitää soveltaa kaikkiin sovelluksen syötteisiin ja tulosteisiin: niitä pitää käsitellä niin, ettei niitä voida missään tilanteessa tulkita muuksi kuin puhtaaksi dataksi, joka ei pääse vaikuttamaan haitallisesti sovelluksen toimintaan. (Mitchell ym. 2011, 174–175.)

4 Toteutus

Tässä pääluvussa kuvataan opinnäytetyön tuloksena syntynyt verkkopalvelun prototyyppi sekä sen toteutusprosessi. Lähtökohtana oli luoda kuntoilijoiden todellisiin tarpeisiin vastaava, julkisella palvelimella Treenikalenteri.net toimiva verkkopalvelu, joka voisi myöhemmin laajentua ominaisuuksiltaan ja omata myös liiketoiminnallista potentiaalia. Tähän liittyen on jo käyty neuvotteluja ammattivalmentajan kanssa. Tämä opinnäytetyö on keskittynyt sovelluksen MVC-arkkitehtuurin ja perustoimintojen luomiseen. Koska opinnäytetyön puitteissa ei voida kaikkia ideoita ja ominaisuuksia vielä toteuttaa, käytetään tässä sanaa prototyyppi. Tähän liittyy kolme olennaista seikkaa:

- 1) Prototyyppi eli opinnäytetyön produkti on luotu välitavoitteena laajemmalle kokonaisuudelle. Prototyyppi on täysin toimiva, mutta ominaisuuksiltaan riisuttu verrattuna Treenikalenteri-projektin myöhempiin tavoitteisiin. Prototyypin tavoitteena on olla testiversio ja suunnannäyttävä myöhemmille parannuksille ja laajennuksille, joita tehdään käyttökokemusten ja palautteen perusteella.
- 2) Prototyyppi on pyritty rakentamaan MVC-arkkitehtuurin ja olio-ohjelmoinnin periaatteita noudattaen siten, että uusien ominaisuuksien tuottaminen onnistuu muuttamatta muuta järjestelmää.

Treenikalenterin kohderyhmänä ovat aktiiviset kuntoilijat, jotka ovat kiinnostuneita suunnittelemaan ja seuraamaan harjoitteluaan. Palvelun tavoitteena on innostaa ja kannustaa liikkujiä ylläpitämään tervettä elämää tukevaa harrastustaan. Päästäkseen tähän tavoitteeseen, on Treenikalenterin ensinnäkin oltava saatavilla ja helppokäyttöinen. Tähän Internetissä toimiva palvelu pääsee hyvällä käyttöliittymäsuunnittelulla, johon liittyy erilaisten käyttöliittymien luominen erilaisia päätelaitteita varten. Tämä aihe ei kuitenkaan mahdu opinnäytetyöni rajoihin. Toiseksi Treenikalenterin on oltava käyttäjilleen hyödyllinen. Tähän se pyrkii seuraavilla tavoilla:

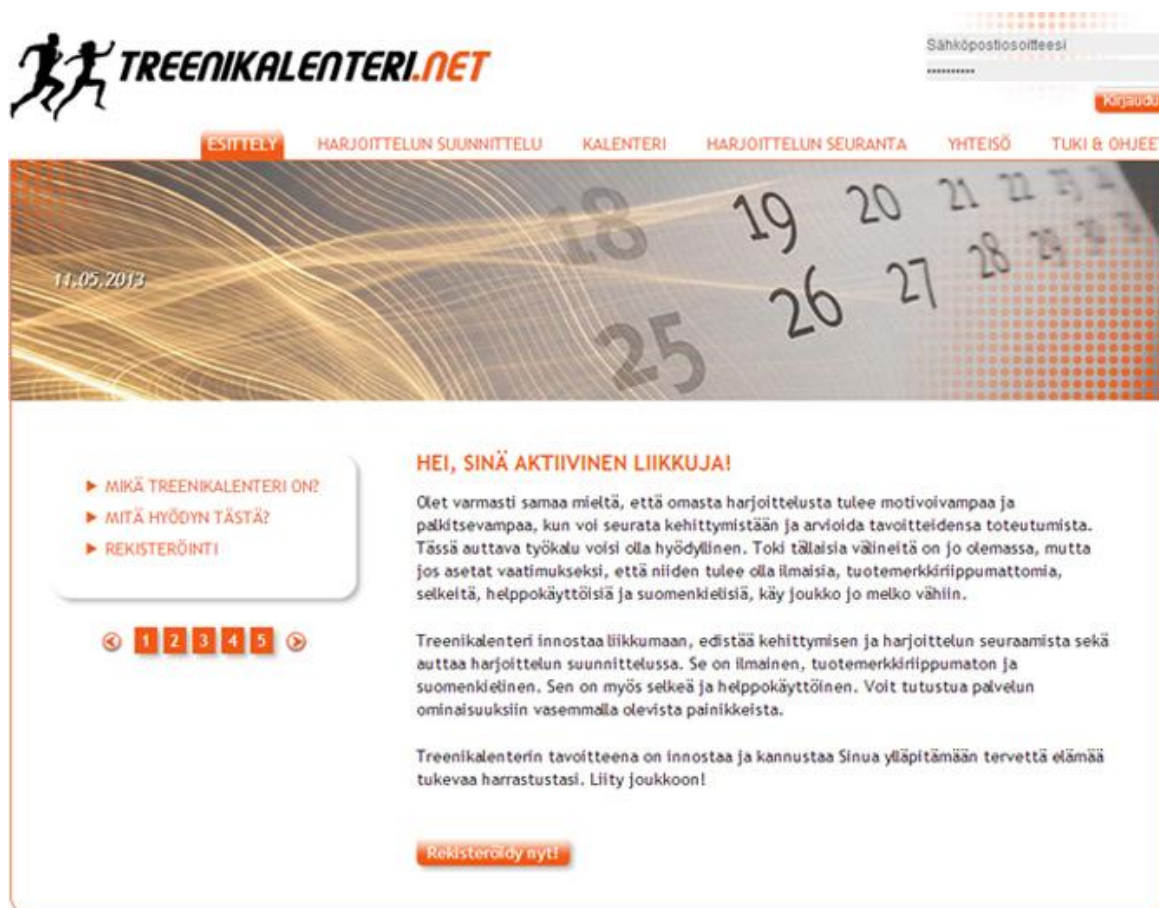
- 1) auttamalla asettamaan tavoitteita
- 2) ohjaamalla harjoittelun suunnittelussa
- 3) tekemällä harjoitusten kirjaamisen helpoksi
- 4) esittämällä havainnollisella tavalla käyttäjän kirjaamia harjoituksia
- 5) luomalla erilaisia indikaattoreita, joiden avulla käyttäjä voi havainnollisesti seurata kehittymistään

- 6) tuomalla liikuntaharrastukseen mukaan sosiaalisen aspektin.
- 7) muistuttamalla ja raportoimalla sähköpostitse käyttäjän niin halutessa.

Treenikalenteri-projektin taustalla on näkemys siitä, että markkinoilla ei ole kovin paljon edellä mainittuja hyötyjä tuottavia verkkopalveluja, jotka ovat ilmaisia, tuotemerkki-riippumattomia, selkeitä, helppokäyttöisiä ja suomenkielisiä.

4.1 Treenikalenterin ominaisuudet

Tässä aluvuussa kuvataan opinnäytetyönä syntyneen verkkopalvelun prototyypin ominaisuudet. Prototyyppi ei riisuttujen ominaisuuksiensa vuoksi vielä tavoita kaikkia edellä mainittuja hyötyjä.



Kuvio 7. Esittelysivu

Käyttäjän tullessa sivustolle aukeaa palvelun esittelysivu (kuvio 7). Kirjautumattomalle käyttäjälle palvelusta aukenee vain tämä esittelyosio. Esittelyosio jakautuu kolmeen

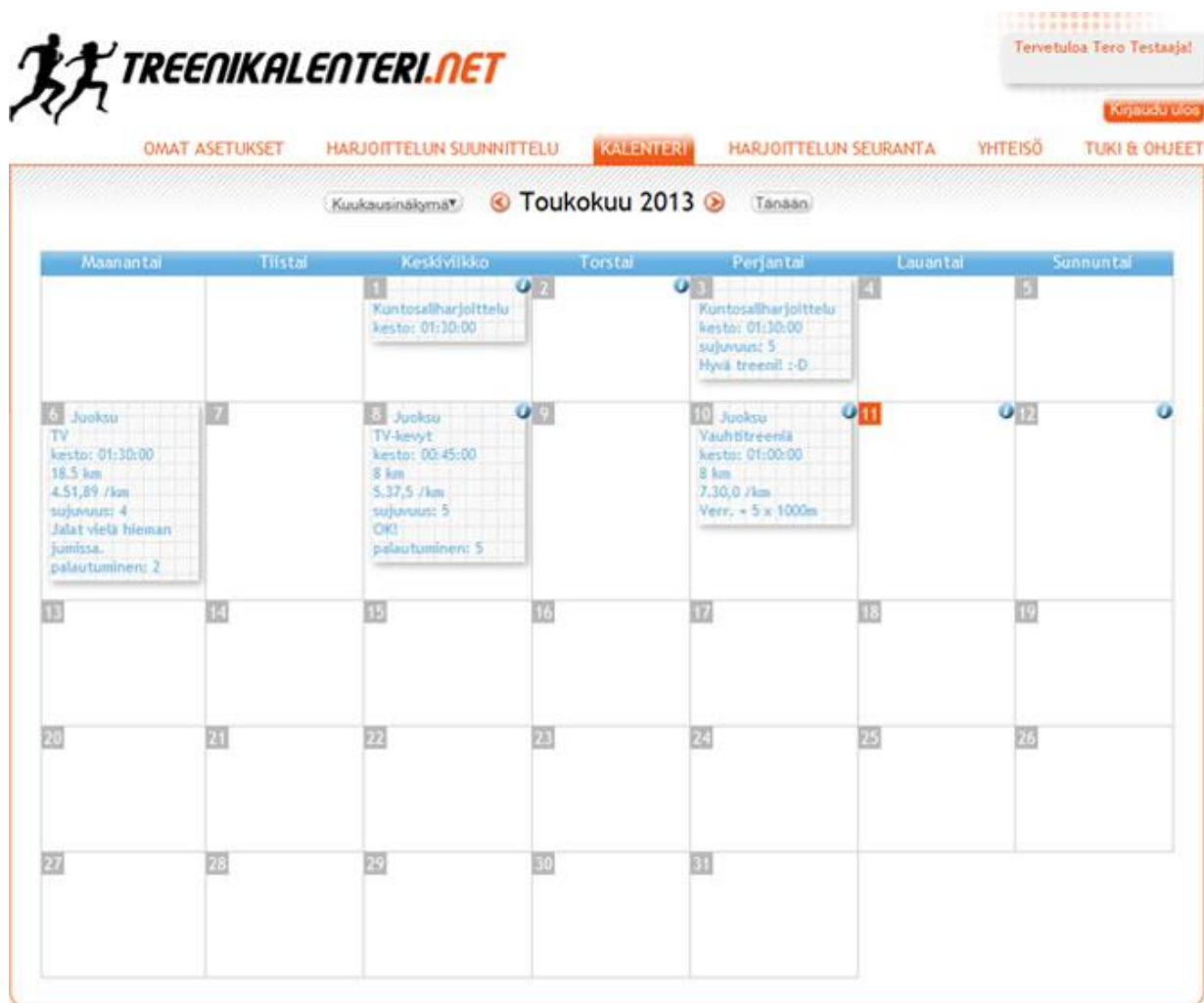
osaan: varsinaiseen esittelyyn, hyötyjen perusteluun ja rekisteröintilomakkeeseen. Näitä vastaavat linkit ovat sekä ylhäällä päänavigoinnissa, että näkymän vasemmassa reunassa. Varsinaisen esittelyn vasemmassa reunassa on nuolipainikkeet ja myös numeroidut painikkeet, joista käyttäjä voi selailta varsinaisen esittelyn alasivuja. Jokaisen näkymän alaosassa on selkeästi näkyvillä painike, josta käyttäjä pääsee rekisteröitymään palveluun.

Kun käyttäjä painaa rekisteröidylinkkiä tai -painiketta, aukeaa HTML-lomake tietojen syöttämistä varten (liite 2). Tällä lomakkeella käyttäjä ilmoittaa palvelun tarvitsemat yhteystiedot sekä muuttumattomat fysiologiset perustiedot. Käyttäjälle ilmoitetaan, että yhteystiedot ja fysiologiset perustiedot eivät näy muille käyttäjille. Käyttäjää kehoitetaan antamaan myös käyttäjänimi, joka näkyy palvelun muille käyttäjille. Lisäksi käyttäjä voi halutessaan kirjoittaa lyhyen kuvauksen harjoittelustaan ja tavoitteestaan.

Kun käyttäjä painaa rekisteröidy-painiketta, lomake lähetetään palvelimelle, joka tarkistaa, että annetut tiedot ovat valideja ja määrämuotoisia. Mikäli jokin syötetty tieto ei kelpaa, palautetaan rekisteröintilomake syötetyillä tiedoilla ja virheelliset kohdat on merkitty näkyvästi, jotta käyttäjä voi ne korjata. Kun käyttäjän antamat syötetiedot ovat kunnossa, järjestelmä lähettää viestin käyttäjän antamaan sähköpostiosoitteeseen ja avaa käyttäjälle näkyviin yksinkertaisen tarkistuslomakkeen, johon käyttäjän tulee syöttää sähköpostissa lähetetty tarkistuskoodi. Näin voidaan varmistaa, että rekisteröityvä käyttäjä todella omistaa sen sähköpostiosoitteen, jonka ilmoitti palveluun. Kun käyttäjä on syöttänyt oikean tarkistuskoodin tarkistuslomakkeelle, pääsee hän käyttämään palvelua.

Kirjautumislomake näkyy sivun oikeassa yläreunassa (kuvio 7). Käyttäjätunnus on järjestelmään ilmoitettu sähköpostiosoite, jonka oikeellisuus on tarkistettu edellä kuvatulla tavalla. Salasanan käyttäjä on määrittänyt rekisteröitymisessä ja sen voi myöhemmin muuttaa asetuksista. Kun kirjautuminen onnistuu, vaihtuu kirjautumislomakkeen sisältö (prototyypissä esitetään vain tervetuloivotus) ja käyttäjä ohjataan aloitussivulle. Se on oletusarvoisesti kalenterin kuukausinäkyvä, mutta aloitussivun voi valita asetuksista tarpeidensa mukaisesti. Kirjautuneelle käyttäjälle palvelun navigointi on erilainen: esittelyosio on vaihtunut asetuksiksi ja lisäksi navigoinnin kaikki linkit ovat nyt käytävissä.

Asetus -osiossa käyttäjä voi muokata joitakin palvelun ominaisuuksia itselleen sopivaksi (liite 3). Prototyypissä käyttäjä pääsee vaihtamaan aloitussivunsa ja salasansa. Aloitus-sivun valinnassa käyttäjälle esitetään yksinkertainen pudotusvalikko, johon listautuu palvelun kaikki sivut. Näistä käyttäjä voi valita haluamansa aloitussivun. Salasanan va-linnassa on kaksi kenttää, joihin syötettyjen salasanojen täytyy olla identtiset. Järjestelmä hyväksyy vain riittävän pitkiä salanoja, joissa tulee olla isojen ja pienten kirjainten li-säksi myös numeroita ja erikoismerkkejä. Mikäli salasana ei täytä vaatimuksia, järjestel-mä ilmoittaa siitä asianmukaisesti.



Kuvio 8. Kalenterin kuukausinäköymä

Kalenteriosio on palvelun keskeinen komponentti (kuvio 8). Se jakautuu kolmeen osaan: kuukausi-, viikko- ja päivänäköymään. Ne voi valita ylhäällä olevasta navigoinnista tai kalenterin yläosassa olevasta painikkeesta. Osoiden toiminnot ovat samanlaiset. Ylhäällä lukee näkyvästi valittu kuukausi ja vuosi sekä viikkonäkymässä lisäksi valittu

viikko ja päivänäkymässä vastaavasti valittu päivä. Kuukausiotsikon ympärillä ovat painikkeet, joista voi selata vuosia, kuukausia, viikkoja ja päiviä. Tänään -painiketta napsauttamalla pääsee kuluvaan kalenteripäivään.

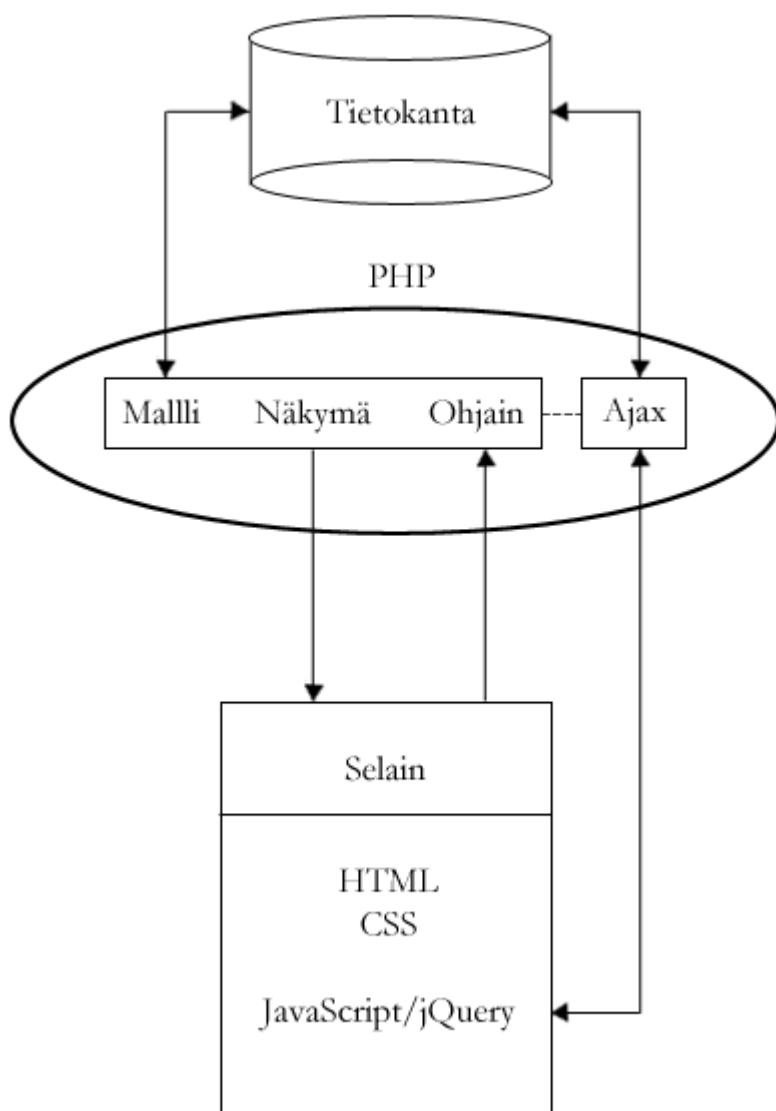
Napsauttamalla kalenteriruudun tyhjää kohtaa käyttäjä saa näkyviin ikkunan, jossa hän voi syöttää uuden harjoituksen (liite 4). Ikkunassa valitaan laji pudotusvalikosta, johon oletusarvoisesti listautuu yleisimmät liikuntalajit (SLU 2010, 16). Käyttäjä valitsee harjoituksen keston liukusäätimellä. Nämä ovat pakollisia valintoja. Lisäksi käyttäjä voi syöttää vapaamuotoisen harjoitusnimikkeen siihen varattuun kenttään, aloitusajan, kilometrit, sujuvuuden, palautumisen sekä lyhyen kuvauksen harjoitteluun liittyvistä tuntemuksista. Kaksoisnapsauttamalla kalenteriruutua käyttäjälle avautuu myös indeksisyöttöikkuna (liite 5). Siinä käyttäjä voi tallentaa järjestelmään painonsa ja leposykkeensä sekä lisätä vapaamuotoisia muistiinpanoja. Molemmat ikkunat ovat raahattavia, joten käyttäjä voi muuttaa niiden paikkaa haluamaansa kohtaan ruudulla. Ikkunat sulkeutuvat joko kaksoisnapsautuksella tai oikeassa yläkulmassa olevasta rastista.

Tallennetut harjoitukset näkyvät kalenterissa vihkоруutuisina lappuina ja tallennetut indeksit näkyvät sinisinä palloina (kuvio 8). Kun käyttäjä napsauttaa olemassa olevaa merkintää, avautuu hänelle samanlainen ikkuna kuin uutta tietoa syötettäessä. Erona on nyt vain, että ikkunoissa näkyvät vastaavan harjoituksen tai indeksin tallennetut tiedot (liite 6). Lisäksi muokkausikkunan yläosa on väriltään oranssi, kun taas uutta tietoa syötettäessä ikkunan yläosa on väriltään sininen. Muokkausikkunassa käyttäjä voi muuttaa tallennetun merkinnän kaikkia tietoja. Harjoituksen ja indeksin päivämäärän voi muuttaa myös raahaamalla kyseisen merkinnän uuteen ajankohtaan. Mikäli harjoitus menee aikaisemmin tallennetun päälle, järjestelmä antaa siitä ilmoituksen. Merkintöjen poistaminen tapahtuu kalenterinäkymissä siten, että käyttäjä raahaa merkinnän kalenterin sivuun, sen ulkopuolelle.

Harjoittelun seuranta -valikkoon (kuvio 8) kuuluu prototyypissä viisi toimintoa: kaikkien harjoitusten listaaminen, kaikkien indeksien listaaminen sekä syke-, paino ja kilometridiagrammit. Listanäkymissä käyttäjä voi tarkastella kaikkia merkintöjään taulukkoina (liite 7). Taulukon sarakkeina ovat syötettyjen merkintöjen tiedot. Käyttäjä voi järjestää taulukoiden tietoja nousevaan tai laskevaan järjestykseen kunkin sarakkeen mukaan.

Taulukon vasemman puoleisin sarake sisältää toimintopainikkeet, joilla käyttäjä voi poistaa kyseisen merkinnän tai muokata sitä samaan tapaan kuin kalenterinäkyvässä. Lisäksi käyttäjä voi ruksata rivin valituksi ja poistaa alhaalla olevalla painikkeella kaikki valitut merkinnät. Diagramminäkymissä käyttäjä valitsee ensiksi piirrettävän diagrammin ylhäällä olevasta valikosta. Sen jälkeen sovellus piirtää diagrammin kyseisistä luke- mista kyseisen kuukauden aikana (liite 8). Käyttäjä voi selaila kuukausia samoin kuin kalenterin kuukausinäkyvässä.

4.2 Treenikalenterin arkkitehtuuri

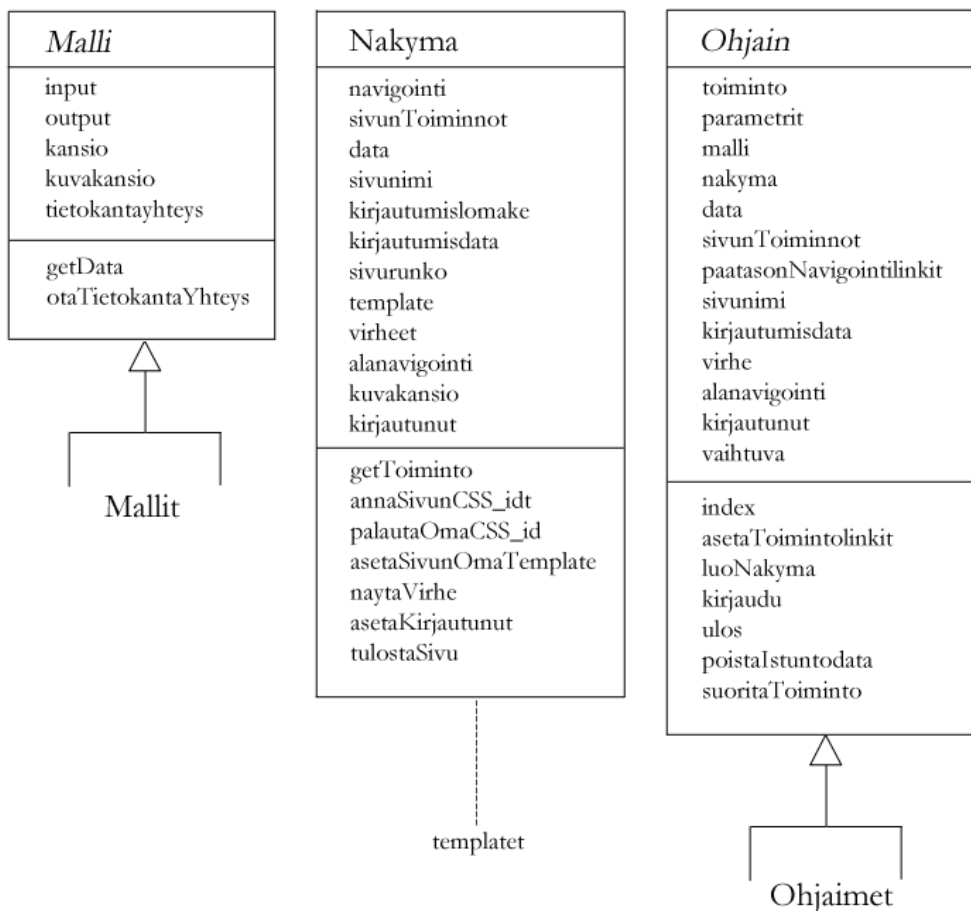


Kuvio 9. Treenikalenterin arkkitehtuuri

Treenikalenterin arkkitehtuuri voidaan karkealla tasolla jakaa kolmeen osaan: selaimessa toimivaan JavaScript -sovellukseen, palvelimella toimivaan PHP -sovellukseen sekä

tietokantaan (kuvio 9). Myös sovelluksen tiedostohierarkia on järjestetty tämän arkkitehtuurin mukaisesti ja nimeämisessä on pyritty käyttämään kuvaavia nimiä.

Selainpään JavaScript -sovellus on käytännössä toteutettu jQuery -kirjastoilla. Se mahdollistaa käyttömukavuutta lisäävät toiminnot, kuten harjoitustietojen ja indeksien raahaamisen. Selainpään sovellustiedostot käyttävät yhteisiä HTML -elementtejä, joiden ulkoasu on muotoiltu CSS -tiedostojen avulla. Jäljempänä kuvattavan näkymän eri sivupohjat ovat myös näitä samoja HTML -elementtejä. Selainpään sovellus tekee Ajax -metodeilla asynkronisia pyyntöjä palvelimelle (luku 3.2). Palvelimella näitä pyyntöjä käsittelevät sovelluksen ajax -kansioista löytyvät PHP -tiedostot, joita ei ole yhtä luokkaa lukuun ottamatta toteutettu oliopohjaisesti. Tämä poikkeuksen tekevä luokka on Testaaja -luokka, joka käyttää Malliin kuuluvaa Kayttaja_Malli -luokkaa testatessaan käyttäjän uusia salasanoja. Ajax-tiedostot tekevät tietokantakyselyjä, päivittävät tietokantaa tarvittaessa ja palauttavat vastauksen selaimen Ajax -metodille, joka tarvittaessa päivittää näkymän vastauksen mukaiseksi.



Kuvio 10. Treenikalenterin abstraktit Kantaluokat sekä Nakyma -luokka

Palvelimella toimiva, PHP:llä kirjoitettu MVC -arkkitehtuuri muodostaa Treenikalenterin rungon. Sen luokat jakautuvat MVC -mallin mukaisesti malli-, näkymä- ja ohjainluokkiin, jotka muodostavat sovellukseen kolme loogisesti erottuvaa toiminnallista tasoa (luku 2). Seuraavassa kuvataan lyhyesti nämä osat.

Selain lähettää myös synkronisia pyyntöjä palvelimelle. Ne ottaa vastaan esiohjain, joka määrittelee varsinaisen ohjaimen uudelleen kirjoitetun HTTP -pyynnön mukaisesti (kuvio 2). Esiohjaimesta muodostettu olio on käytännössä ensimmäinen olio, joka syntyy käyttäjän ladatessa sivun selaimensa. Esiohjainta lukuun ottamatta ohjainluokat periytetään kantaluokasta *Ohjain* (kuvio 10). Luvussa 3.1 esitetyn mukaisesti alaluokat perivät kantaluokkansa attribuutit ja metodit. Täten kaikilla Treenikalenterin ohjainluokilla on esimerkiksi ominaisuudet malli, data ja nakyma. Niillä kaikilla on myös metodi nimeltä luoNakyma. Muun muassa näillä attribuuteilla ja metodeilla ohjaimet nimensä mukaisesti ohjailevat sovellusta. Ohjaimet huolehtivat navigoinnista ja käyttäjän syötteiden muuntamisesta sovellustoiminnoiksi. Ne välittävät käyttäjän toiminnot eteenpäin siten, että ne luovat sopivan malliolion ja käyttävät sen ominaisuuksia pyytäessään dataa näkymälle. Saatuaan datan ne käyttävät metodia luoNakyma ja komentavat näkymää valitsemaan sopivan sivupohjan ja lähettävät datan sille.

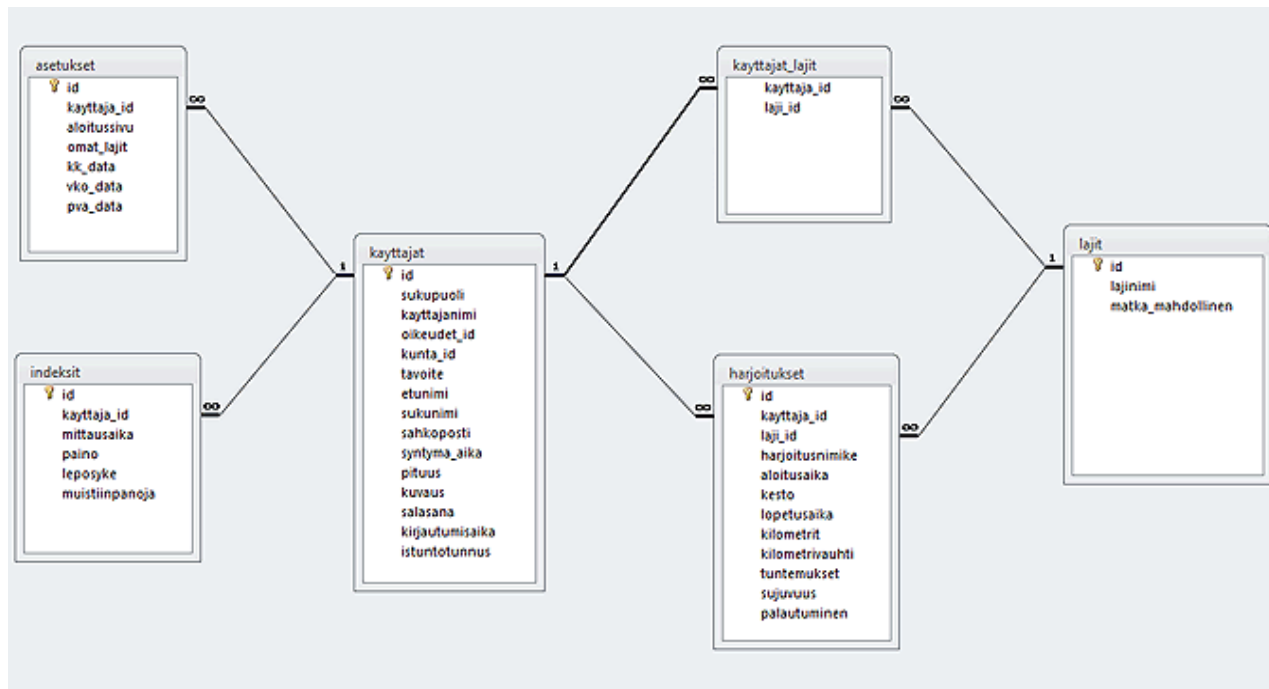
Kaikki malliluokat periytetään abstraktista kantaluokasta *Malli* (kuvio 10). Siten kaikilla malliluokilla on mm. attribuutit input ja output sekä metodit getData ja otaTietokantaYhteys. Olennaisesti näillä attribuuteilla ja metodeilla mallit huolehtivat luvussa 2 esitetyn mukaisesti Treenikalenterin toimintalogiikasta ja tietokannan päivittämisestä. Kun ohjain luo mallin, saa syntyvä malliolo syötteenä inputin, jonka mukaisesti se osaa tehdä tietokantaoperaatioita ja muita loogisia sovellustoimintoja. Ohjain saa datan mallilta getData -metodin avulla. Data on useimmiten matriisina, mutta voi olla myös staattinen HTML-dokumentti. Mallit ovat täysin riippumattomia näkymästä ja kommunikoivat ainoastaan ohjainluokkien kanssa. Niin malleilla kuin ohjaimillakin on perittyjen attribuuttien ja metodien lisäksi omia, omaan tehtävään erikoistuneita ominaisuuksia.

Treenikalenterin MVC-mallissa näkymä eroaa muista tasoista siten, että luokkia on vain yksi. Näkymän keskeisimmät piirteet ovat attribuutti data ja metodi tulostaSivu. Olennaisesti niiden avulla näkymä esittää ohjaimen mallilta pyytämän datan käyttäjälle. Nä-

kymä saa dataa ohjainluokilta, mutta ei muuten kommunikoi muiden luokkien kanssa. Treenikalenterissa näkymäluokka käyttää useita kulloinkin tilanteeseen sopivia sivupohjia, templateja. Se ei kuitenkaan itse päättelä, mitä sivupohjaa tulee käyttää, vaan saa pyynnön ohjaimelta esittää tietty data tietyssä sivupohjassa.

Jotta luokkahierarkia olisi mahdollisimman selkeä, on periytyvät luokat nimetty päätteillä `_Malli` ja `_Ohjain` sen mukaisesti, mille arkkitehtuurin tasolle ne kuuluvat. Pääteen edessä on luokan tehtävää kuvaava nimitys. Siten esimerkiksi `Kirjautu_Malli` hoitaa käyttäjän kirjautumisen ja tarpeelliset muutokset tietokannassa ja `Piirtaja_Malli` vastaa dynaamisten graafisten kuvaajien piirtämisestä. Ohjainluokat on nimetty vastaavan logiikan mukaisesti. Koska ohjainluokat huolehtivat sovelluksen ohjailusta käyttäjän navigoinnin mukaan, ohjainten nimen alkuosassa näkyy sivuston osaa kuvaa nimitys. Siten esimerkiksi `Kalenteri_Ohjain` ohjailee metodeillaan tapahtumia, kun käyttäjä on kalenteriosiossa ja `Seuranta_Ohjain` käskyttää sopivia malleja käyttäjän liikkeessa `Seuranta` -osiossa. Näkymän käyttämät sivupohjat on nimetty siten, että nimen alkuosan muodostaa konkreettinen näkymää kuvaava nimitys ja loppuosan pääte `_template`. Siten esimerkiksi käyttäjän tallentaessa tai muokatessa harjoitusta näkymäolio käyttää `syottokentta_template` -nimistä sivupohjaa, kun taas käyttäjän ollessa rekisteröitymässä hänelle tarjotaan `rekisterointi_template` näkyville.

Treenikalenteri käyttää tietovarastonaan MySQL -tietokantaa. Kuten luvussa 3.1 esitettiin, relaatiotietokannoissa todellisuutta mallintava tieto esitetään tauluina. Treenikalenterin käsitelmän keskeisin käsite on käyttäjä, jonka attribuutit esitetään `kayttajat` -taulussa. Siihen tallennetaan kaikki rekisteröityneen käyttäjän uniikit tiedot, joita samalla käyttäjällä voi samana hetkenä olla vain yksi. Käyttäjällä voi esimerkiksi olla vain yksi salasana tai vain yksi sähköpostiosoite (tässä palvelussa). Tiedot, jotka liittyvät käyttäjään, mutta eivät ole käyttäjän ominaisuuksia, esitetään tauluissa `asetukset`, `indeksit`, `harjoitukset` ja `kayttajat_lajit`. Taulun nimitys kertoo, mitä käyttäjään liittyvää tietoa se mallintaa. Tarkemmin taulujen sisällöt sarakkeineen ja tietotyypeineen on kuvattu liitteissä 9 – 11. `Kayttajat_lajit` on ns. välitaulu. Tämä taulu mahdollistaa sen, että käyttäjä voi valita omat lajinsa, jotka hän näkee pudotusvalikossa syöttäessään tai muokatessaan harjoituksia. Lajitieto tulee omasta taulusta `lajit`, johon on koottu harrastetuimpia liikuntalajeja.



Kuvio 10. Treenukalenterin tietokanta

Treenukalenterin tietokannan jokaisen taulun tallennusmoottorina on InnoDB, joka mahdollistaa, kuten aikaisemmin luvussa 3.1 todettiin, transaktioiden käytön. Transaktioita käytetäänkin kaikissa Treenukalenterin tietokantaoperaatioissa, joissa lisätään tai muutetaan tietoa tauluun.

Edellä esitetystä yhteenvedosta voidaan esimerkkinä tarkastella kahta käyttötapausta siltä kannalta, mitä Treenukalenterin arkkitehtuurissa tapahtuu käyttäjän konkreettisten toimintojen seurauksena.

Tapaus 1: käyttäjä haluaa lisätä uuden leposykelukeman tietylle päivälle. Käyttäjä kirjautuu järjestelmään. Hän syöttää tunnuksensa ja salasansa ja painaa kirjaudu -painiketta. Esiohjaimen synnyttämä ohjainolio käyttää Ohjain -kantaluokan kirjaudu -metodia, joka antaa käyttäjän syötetä parametrinä muodostamalleen Kirjaudu_Malli -oliolle. Tämä olio ottaa kantaluokansa otaTietokantaYhteys -metodin avulla yhteyttä tietokannan tauluun kayttajat ja katsoo löytyykö oikeaa tunnus- salasana-paria. Mikäli yksi rivi löytyy, malliolio tarkistaa vielä, onko käyttäjällä oikeudet voimassa. Jos on, malliolio lukee tietokannan asetukset -taulusta käyttäjän aloitussivun. Jos tietokantaoperaatiot onnistuvat, malliolio lisää kayttajat -tauluun istuntotunnuksen oikean rivin kohdalle ja ohjaa käyttäjän aloitussivulle. Mikäli tietokantaoperaatiot epäonnistuvat transaktio pe-

rutaan ja käyttäjälle annetaan virheilmoitus. Onnistuneen kirjautumisen jälkeen käyttäjä navigoi kalenteriosioon tai hänet ohjataan sinne automaattisesti, mikäli se on hänen aloitussivunsa. Käyttäjä navigoi oikean kalenteripäivän kohdalle. Kalenteri_Ohjain -olio käyttää nyt Kalenteri_Malli -oliota, joka hakee pyydetyn datan eli kalenteritiedot ja käyttäjän merkinnät. Ohjainolio käskee näkymäoliota päivittämään näkymän saamansa datan mukaiseksi. Käyttäjä kaksoisklikkaa kalenteriruutua syöttääkseen uuden indeksin. Tällöin selainpään JavaScript -sovellus avaa syöttöikkunat, jotka ovat HTML -elementtejä. Käyttäjä antaa sykearvonsa liukusäätimen avulla ja painaa tallenna. Tällöin JavaScript -sovelluksen ajax -metodi ottaa yhteyttä palvelimen ajax -kansiossa olevaan PHP -tiedostoon, joka taas ottaa yhteyttä tietokantaan ja lisää indeksit tauluun rivin, jonka kayttaja_id -sarakkeen arvoksi tulee käyttäjän id ja leposykesarakkeen arvoksi tulee käyttäjän liukusäätimellä syöttämä arvo. Mikäli tietokantaoperaatio onnistuu, PHP antaa Ajaxille vastaukseksi uuden rivin tunnuksen ja Ajax päivittää sivun uudella indeksipallukalla, joka on dynaamisesti synnitetty CSS:llä muotoiltu HTML -elementti. Tämän elementin tunnukseksi tulee Ajaxin vastauksena saama id, jotta käyttäjä voi saman tien muuttaa uutta tietoa ilman sivunlatausta ja päivityksen kohde voidaan tunnistaa. Mikäli tietokantaoperaatio ei onnistu, transaktio perutaan.

Tapaus 2: käyttäjä haluaa katsella edellisen kuukauden syke-diagrammia. Käyttäjä kirjautuu palveluun kuten edellä. Käyttäjä navigoi Harjoittelun seuranta -osioon, jolloin esi-ohjain synnyttää uuden Seuranta_Ohjain -olion. Kun käyttäjä navigoi alasivulle syke-diagrammi, äsken synnitetty ohjainolio käyttää syke-diagrammi -metodiaan. Tämä metodi luo uuden Kalenteri_Malli -olion, joka palauttaa kalenteridatan ohjaimelle. Kalenteridataan kuuluu olennaisesti mm. se, mikä kuukausi on kyseessä ja montako päivää kyseisessä kuukaudessa on. Kalenteri_Malli -olio ottaa lisäksi yhteyttä tietokannan indeksit -tauluun ja lukee sieltä käyttäjän leposykkeet kyseisen kuukauden ajalta. Myös tämän datan se palauttaa ohjainoliolle, joka synnyttää nyt uuden Piirtaja_Malli -olion. Ohjainolio välittää tälle malliolille sykedatan, joka perusteella tämä piirtää diagrammin.

4.3 Treenikalenterin toteuttaminen

Tämä opinnäytetyö toteutettiin tekijän omalla testipalvelimella. Palvelinohjelmistona oli Apache/2.4.2 (Win64). Tietokantaohjelmistona oli MySQL 5.5.24. Työssä käytetty

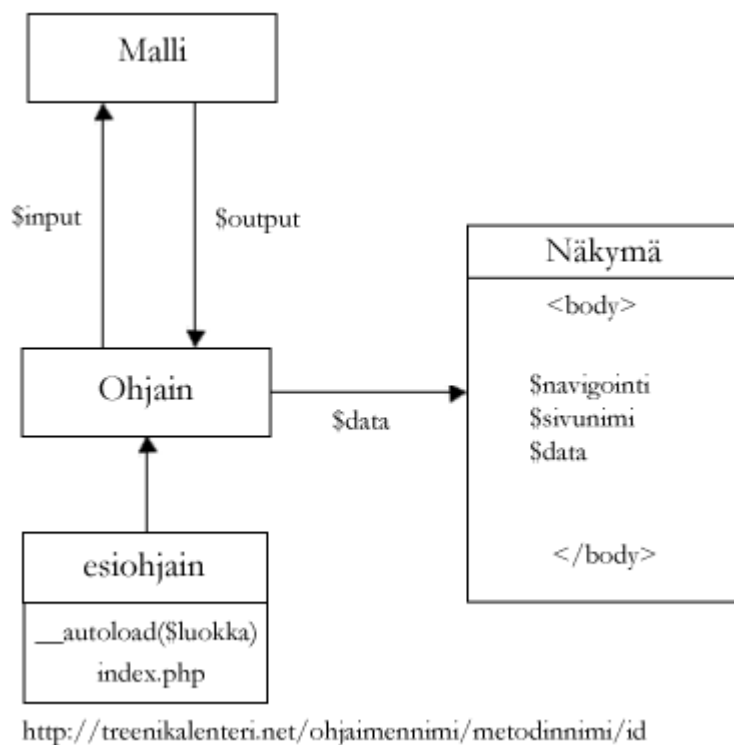
PHP:n versio oli 5.4.3. Layouttien ja käyttöliittymän suunnitteluun käytetty ohjelma oli Adoben Photoshop CS5. Tietokantasuunnittelun apuna käytettiin phpMyAdmin 3.5.1 -ohjelmaa. Varsinaiseen koodaamiseen käytettiin ohjelmia Adobe Dreamweaver CS5 sekä Notepad++.

Tutkimusongelmien valossa tarkasteltuna Treenikalenterin toteuttaminen voidaan jakaa kolmeen tärkeään osa-alueeseen: MVC-arkkitehtuurin toteuttamiseen, tietoturvan toteuttamiseen ja käyttöliittymän toteuttamiseen. Tässä aluvuossa kuvataan tarkemmin Treenikalenterin toteutusprosessi noudattaen yllä mainittua järjestystä.

Treenikalenterin työstäminen lähti liikkeelle vaadittavien ominaisuuksien suunnittelusta. Hyvin pian kävi selväksi, että verkkopalvelu on tuote, joka ei varsinaisesti koskaan ole valmis: palvelua on ylläpidettävä ja on hyvin mahdollista, että se tarvitsee myöhemmin uusia ominaisuuksia. Sen vuoksi oli tärkeää löytää sovellukselle sellainen rakenneratkaisu, jota olisi helppo ylläpitää ja täydentää. Lisäksi Treenikalenterin kaltaisen verkkopalvelun ollessa kyseessä on tärkeää ainakin tulevaisuudessa huomioida myös erilaiset päätelaitteet, joilla palvelua mahdollisesti käytetään. Tavoitteena oli siis arkkitehtuuri, joka on ylläpidettävä, laajennettava ja jossa käyttöliittymä on erillään sovelluslogiikasta.

Lähdekirjallisuuden mukaan vaikutti ilmeiseltä, että MVC-arkkitehtuuri sopisi hyvin edellä mainittuun tilanteeseen, koska siinä malliluokat ovat varsin itsenäisiä ja käyttöliittymä erillään sovelluslogiikasta. Seuraava vaihe työssä olikin käytännön perehtyminen MVC-arkkitehtuuriin ohjelmoimalla. Ennen varsinaisen Treenikalenterin ohjelmoimista luotiin pelkistetty MVC-arkkitehtuuria noudattava runko, jonka avulla testattiin ajatuksia, joita voisi soveltaa varsinaisen työn toteuttamisessa. Kun perusidea tuntui toimivalta, sen päälle lähdettiin rakentamaan varsinaista Treenikalenteria. Seuraavassa esitetään, osittain kerraten, ne ratkaisut, joihin testivaiheen perusteella päädyttiin.

Treenikalenterin aloitussivu on `index.php`. Tällä tiedostolla käynnistyy Treenikalenterin MVC-arkkitehtuurin toiminta (kuvio 11). Tähän tiedostoon liittyy olennaisesti `htaccess`-tiedosto, joka kirjoittaa uudelleen URL-osoitteen ihmisille luettavampaan ja hakukoneille edullisempaan muotoon pyynnön saapuessa palvelimelle. Kaikki Treenikalenterin URL-osoitteet ovat täten muotoa: `treenikalenteri.net/ohjaimen_nimi/metodin_nimi`.



Kuvio 11. MVC-arkkitehtuurin toiminta Treenikalenterissa

Aloitussivulla sovelletaan PHP:n autoload -funktioita, joka lataa automaattisesti oikean luokkatiedoston oikeasta kansioista aina, kun se huomaa, että ollaan muodostamassa uutta oliota. Näin lukuisia luokkatiedostoja ei tarvitse liittää erikseen käsin kirjoittamalla.

Ensimmäinen luotava olio on esiohjain, jonka metodi osaa luoda käyttäjän syöttämän URL:n perusteella sopivan varsinaisen ohjainolion. Sen jälkeen äsken luotu olio käynnistää oikean metodinsa käyttäjän antaman URL-syötteen perusteella. Ohjaimet siis hoitavat navigoinnin ja sovelluksen ohjailun käyttäjän URL-syötteiden perusteella. Sen lisäksi niiden tulee käynnistää tilanteeseen sopivia sovellusoperaatioita. Tähän ne käyttävät malleja, kuten luvussa 4.2. kuvattiin. Mallit saavat ohjaimilta syötettä, toimivat sen perusteella ja palauttavat dataa takaisin ohjaimelle. Tämä data on usein matriisimuodossa. Kun ohjain on saanut datan mallilta, se käskää näkymää asettamaan sopivan sivupohjan ja sen jälkeen näkymä esittää datan sivupohjallaan. Nämä sivupohjat käyttävät kaikki yhteistä HTML-runkoa. Sen rakenne muodostuu yhteisestä yläosasta, kesiosasta ja alaosasta. Yläosassa näkyvät sisään- ja uloskirjautumislomakkeet sekä navigointi. Keskiosa on se osa sivurungosta, johon näkymä sijoittaa kulloinkin vaihtuvan templa-

ten. Alaosa on myös sivurungon yhteinen osa. Se ei prototyypissä tee muuta kuin esitä kulloinkin näytettävän templatien nimen, mikä voi kehitysvaiheessa olla vieraille kehittäjälle jonkinlainen etu.

Alusta alkaen tuntui selvältä, että tietoturva on niin tärkeä osa-alue verkkopalvelun kehittämisessä, että se täytyy jollain tasolla ottaa mukaan tarkasteluun. Aihe tuntui erittäin laajalta ja vaikealta rajata ainakin opinnäytetyön raporttiosuudessa. Aluksi päädyttiin melko suppeaan ratkaisuun, mutta sitten aivan loppuvaiheessa raportin tietoturvaosuutta laajennettiin ja itse sovellukseen tehtiin vielä joitakin merkittäviä parannuksia. Seuraavassa selostetaan, miten tietoturva on huomioitu Treenikalenterin ohjelmoinnissa.

Ylivoimaisesti tärkein periaate tietoturvan toteuttamisessa on käyttäjän syötteiden tarkistaminen. Tämä kävi jo lähdekirjallisuuden perusteella hyvin selväksi. Selainpuolen tarkistuksilla ei tässä ole juuri merkitystä, joten kaikki Treenikalenterin syötetarkistukset tehdään palvelinpuolella. Kaikki mallit ja ajax -kansion PHP -tiedostot tarkistavat jokaisen syöteen. Tarkistamisessa käytetään seuraavia keinoja: 1) Syöteen tulee olla jokin rajatuista vaihtoehdoista. 2) Syöteen tulee olla validi päivämäärä tai validi luku. 3) Syöteen määrämuotoisuuden tarkistamiseen käytetään säännöllisiä lausekkeita. 4) Syöte puhdistetaan HTML-merkkauksesta ja muista tarpeettomista merkeistä. 5) Käytetään PDO:n valmisteltuja lausekkeita ja parametrien sidontaa kaikissa tietokantaoperaatioissa. Jos syöte ei pääse tarkistuksista läpi, se luonnollisesti hylätään, toimenpide keskeytetään, tarvittaessa käytetään tietokannan päivitys- ja lisäysoperaatioissa transaktioita ja käyttäjälle annetaan virheilmoitus. Dynaamisesti tulostettava data on myös pyritty siivoamaan HTML-merkkauksesta ja erikoismerkeistä. Treenikalenterin sivurunkoon on myös eksplisiittisesti merkattu käytettävä merkistökooodaus, jotta voidaan olla varmoja siitä, mitkä merkit ovat erikoismerkkejä.

Ohjain tarkistaa käytännössä myös URL-syötteet ja reagoi ”laittomiin” syötteisiin hie-man eri lailla riippuen siitä, onko asetuksissa asetettu tuotantoversion arvoksi true vai false. Tämä koskee kaikkiin muihinkin virhetilanteisiin liittyviä ilmoituksia. Jos tuotantoversio on false, antaa järjestelmä tarkkoja ilmoituksia, mikä voi olla kehittäjän kannalta hyödyllistä. Jos taas tuotantoversio on true, ei kannata virheilmoituksilla paljasta juuri mitään järjestelmästä.

Transaktiot ja viite-ehyettä tukevat tietokannan taulut ovat tärkeä osa tiedon eheyden vaalimisessa (luku 3.3). Tämä on Treenikalenterin tietoturvassa huomioitu ensinnäkin siten, että on valittu MySQL -tietokanta jossa nämä ominaisuudet on mahdollista toteuttaa. Toiseksi transaktiot on myös toteutettu ohjelmoinnin keinoin kaikkiin tietokantaoperaatioihin, joissa muutetaan kantaa tai lisätään kantaan jotain. Kolmanneksi on varmistettu, että kaikissa tauluissa on viite-ehyettä ja transaktioita tukeva tallennusmoottori (luku 3.1).

Yksi Treenikalenterin tärkeä tietoturvaominaisuus liittyy rekisteröitymiseen. Kuten luvussa 4.1 mainittiin, Treenikalenteri tarkistaa käyttäjän ilmoittaman sähköpostiosoitteen lähettämällä tähän osoitteeseen uniikin satunaisen varmistuskoodin. Käyttäjä pääsee rekisteröitymään vasta, kun on lähettänyt tuon koodin takaisin järjestelmään. Näin voidaan olla jokseenkin varmoja, että sähköpostiosoite kuuluu juuri tuolle ihmiselle, joka sen rekisteröitymisessä ilmoitti ja samoin hänen henkilöydelleen saadaan jonkinlainen todellisuus pohja.

Treenikalenteri huomioi tietoturvan myös sallimalla ainoastaan vahvoja salasanoja. Tämä salasanoille asetettu vaatimus koskee niin rekisteröitymistä kuin salasanan vaihtoa-kin. Lisäksi salasanoja ei säilytetä selväkielisenä missään, vaan kohtalaisen vahvasti salattuina.

Treenikalenteri pyrkii suojautumaan istuntotunnusten kaappaamista vastaan vaihtamalla koko ajan käyttäjän istuntotunnusta. PHP tarjoaa tähän välineen nimeltä `session_regenerate_id`. Kuten kirjallisuuskatsauksessa selvisi, kannattaa istunnon aitoudesta varmistua muillakin keinoilla (luku 3.3). Myös Treenikalenteri soveltaa tätä turvallisuustekijää: jokaisen pyynnön mukaan liitetään oma salainen tunnus, jolla pyritään varmistamaan, että pyyntö todella kuuluu oikeaan istuntoon. Tämä salainen tunnus kulkee myös jokaisen Ajax -pyynnön mukana. Palvelin tarkistaa pyynnön mukana tulevan tunnuksen ja vertaa sitä istunnon tunnukseen. Jos tunnukset ovat samoja, voidaan olla kohtalaisen varmoja siitä, että pyyntö kuuluu oikeaan istuntoon. Muuten on jonkinlainen syy epäillä potentiaalista CSRF -hyökkäystä ja palvelin hylkää pyynnön. Lisäksi istuntoihin liittyen on mainittava, että ohjain tuhoaa istunnon lopuksi kaiken istuntotiedon palvelimelta.

Treenikalenterin käyttöliittymäsuunnittelu ja visuaalinen suunnittelu aloitettiin rinnakkain prosessin kanssa, jossa pyrittiin hahmottamaan palvelun ominaisuuksia. Aluksi työssä ei koodattu yhtään mitään, vaan sen sijaan piirrettiin ja kokeiltiin Photoshopilla erilaisia layoutteja. Koska on kuitenkin aika erikoinen ajatus ajatella MVC -mallia ilman käyttöliittymää, koodattiin edellä mainittuun MVC:n testiversioon saman tien pelkistetty käyttöliittymä. Käytännössä käsitteet näkymä, sivupohja, navigointi ja käyttöliittymä menevät päällekkäin toistensa kanssa. Käytännössä voidaan kuitenkin ajatella, että kaikki näkyvä, joilla käyttäjä jotenkin ohjailee sovellusta, on käyttöliittymää. Aluksi oli toiveissa liittää sovellukseen oma käyttöliittymä myös mobiililaitteille, mutta siitä jouduttiin aikataulun puitteissa luopumaan.

Treenikalenterin käyttäjä käsittelee sovelluslogiikkaa selaimessaan näkyvän käyttöliittymän avulla. Sovellus tuottaa katseltavaksi ja muokattavaksi dynaamisia näkymiä, joiden rakenne kuvataan HTML-kuvauskielellä. HTML on pitkään ollut standardi verkkosivujen rakenteen kuvaamisessa. Sen viimeisin versio, HTML5, valittiin toteutukseen uusien mielenkiintoisten ominaisuuksiensa vuoksi. Tulevaisuuden kannalta tämä on mielenkiintoinen ja oikea ratkaisu, mutta tämän prosessin edetessä todettiin, ettei HTML5:lle sinänsä kannata asettaa muita odotuksia kuin, että pyrkii huolehtimaan siitä, että dynaamiset näkymät tuottavat validia HTML:ää. Sen sijaan JavaScriptin jQuery -kirjasto osoittautui aluksi suunniteltua merkittävämmäksi tekijäksi käyttöliittymän toteuttamisessa.

Treenikalenterissa lähes jokainen näkymä käyttääkin jQueryä käyttöliittymän toteutuksessa jollain tavoin. Sen avulla toteutettiin indeksipallukoiden ja harjoituslappujen raahaaminen uuteen aikaan ja ikkunoiden raahaaminen sopivaan sijaintiin. Myös merkintöjen tuhoaminen tehtiin mahdollisimman helpoksi, kun ne voi tuhota raahaamalla oikealle tai vasemmalle kalenterin sivuun.

5 Yhteenveto, pohdinta ja jatkokehitysehdotukset

Koskien ensimmäistä tutkimusongelmaa voidaan yhteenvetona todeta, että Treenikalenterin MVC -arkkitehtuurin luomisessa onnistuttiin kohtalaisen hyvin. Sovelluksessa on runsaasti erilaisia näkymiä, jotka on onnistuttu luomaan sovelluslogiikasta riippumattomaksi. Toisaalta näkymiä tuotettiin koko ajan omalle pöytäkoneelle eikä päästy oikeasti testaamaan täysin erilaisen käyttöliittymän luomista sovellukseen. Tämä onkin ensimmäisiä jatkokehitysideoita.

Yksi sovelluksen ongelma voi olla se, että näkymät saavat ohjaimilta varsin isoja matriiseja tulostettavakseen. Järjestelmää ei ole testattu tositilanteessa, mutta matriiseja voi pitää toisaalta potentiaalisena suorituskykyongelmana joissain tapauksissa ja toisaalta kehitystyötä vaikeuttavana tekijänä. Toinen arkkitehtuuriin liittyvä heikkous on se, ettei järjestelmä noudata täysin MVC -mallia, vaan sen rinnalle luotiin selain-palvelinpään Ajax-systeemi (kuvio 9). Muutenkin JavaScript -koodi on sovelluksessa paljon heikommin organisoitua kuin PHP -koodi. Tämä johtuu ensisijaisesti siitä, että JavaScript -koodausta ei toteutettu oliopohjaisesti. JavaScript -koodin oliopohjaistaminen on myös yksi myöhempien aikojen jatkokehitysehdotus. Toisaalta Ajax parantaa kyllä sovelluksen käytettävyyttä, mutta sen toteutusta pitää vielä myöhemmin tarkistaa.

Koskien toista tutkimusongelmaa, voidaan yhteenvetona todeta, että PHP:ssä on monia hyviä välineitä tietoturvan ylläpitämiseen. Tietoturvaongelmia löytyy varmasti paljon enemmän kielen käyttäjistä kuin itse kielestä. Olennaista tietoturvan luomisessa verkkopalveluun onkin kielestä riippumatta huolellisuus käyttäjien syötteiden tarkistamisessa. Tärkeää on myös tietoisuus verkossa liikkuvista vaaroista ja viitseliäs varautuminen uhkiin. Treenikalenterin syötteiden tarkistamisessa pyrittiin olemaan huolellisia ja johdonmukaisia. Hyvä ominaisuus on myös sähköpostiosoitteen tarkistaminen rekisteröitymisvaiheessa. Lisäksi projektin loppupuolella vielä parannettiin palvelun tietoturvaa istunnon suojaamisella, mikä voi olla merkittävääkin.

Palvelun tietoturvatoteutusta raportoitaessa tuli mieleen ajatus, että millä tasolla sovelluksen tietoturvaominaisuuksia oikeastaan kannattaa selostaa. Jos nimittäin pitää lähtökohdana, että sekä raportti, että sovellus ovat julkisia, ei välttämättä ole omalta kannalta

viisasta kertoa kaikkea juurta jaksan. Tämä koskee myös sovelluksen muita osia. Treenikalenteri tuskin tuo mitään mullistavaa uutta esille, mutta ainakin periaatteessa tämä on tärkeä juttu, ja sovelluksen arkkitehtuuria selostettaessa päätettiin alkuperäisestä tarkkuudesta luopua.

Liittyen kolmanteen tutkimusongelmaan voidaan yhteenvetona todeta, että sovelluksen työpöytäversioon onnistuttiin luomaan tyylikäs ja helppokäyttöinen käyttöliittymä, mikä oli tavoitteenakin. Toisaalta käyttöliittymän toteutuksessa ollaan melko riippuvaisia JavaScript/jQuery -koodista, joka ei ole, kuten edellä todettiin, tekijää täysin tyydyttävästi organisoitua. Tässä on odotettavissa kenties vaikeuksia, kun järjestelmään lähdetään tekemään mobiikäyttöliittymää. Tässä yhteydessä todettakoon, että JavaScriptin ja jQueryn kanssa käytiin tämän projektin suurimmat taistelut, joiden jälkeen keskustelu PHP:llä palvelimen kanssa tuntui varsin miellyttävältä. Suurin ongelma noissa taisteluisa oli se, että ne olivat vaarantaa projektin aikataulun.

Näillä ominaisuuksilla Treenikalenteri ei vielä täytä sille asettamiani vaatimuksia, mutta uskoisin, että nyt on käsissä hyvä pohja, jota kannattaa kehittää. Kuten raportin johdannossa ja luvussa neljä perustelin, tämän kaltaisella sovelluksella voisi hyvinkin olla hyödynnettävyyttä. Oma domain, treenikalenteri.net, on jo olemassa ja seuraava iso etappi projektissa onkin sen julkaiseminen. Uudet ominaisuudet odottavat koodaamista ja tunnustelevat neuvottelut valmentajan kanssa on aloitettu.

Oman MVC -järjestelmän luominen tyhjästä oli työläs, mutta erittäin antoisa ja opettavainen urakka. Suurimmat vaikeudet projektissa oli keksiä alussa se, miten järjestelmä oikeastaan toimii, mutta alkuvaiheen tutkimustyö testiversiolla kannatti. Tämän jälkeen työ oli palvelinpuolella oikeastaan sujuvaa. Työtä on tehty useita tunteja suurella intohimolla ja tinkimättömällä asenteella. Mutta vaikka olen myös PHP -ohjelmoijana tämän projektin jälkeen paljon rutinoituneempi, olen silti edellä mainituista taisteluista, tai niiden ansioista, kehittynyt selainpuolen ohjelmoinnissa huomattavasti enemmän. Nyt osaan ottaa huomioon sen, että tapahtumat kuplivat DOM -puussa ylöspäin tai että dynaamisesti luotaville elementeille pitää jQueryssä delegoida tapahtumat.

Lähteet

Burbeck, s. 1992. Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). Luettavissa: <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.htm>. Luettu: 10.1.2013.

CERT (Computer Emergency Response Team at Carnegie Mellon University). Understanding Malicious Content Mitigation for Web Developers. Luettavissa: http://www.cert.org/tech_tips/malicious_code_mitigation.html. Luettu: 5.2.2013.

CERT-FI 2008. Tietoturva nyt! Luettavissa: http://www.cert.fi/tietoturvanyt/2008/03/P_6.html. Luettu: 6.2.2013.

Deacon, J. 2009. Model-View-Controller (MVC) Architecture. Luettavissa: <http://techsimplified2.com/Uploads/Agendas/October28,2011.pdf>. Luettu: 9.1.2013.

Flanagan, D. 2011. JavaScript: The Definitive Guide. O'Reilly Media. Sebastopol.

Gamma, E., Helm, R., Johnson R. & Vlissides J. 2001. Olio-ohjelmointi. Suunnittelumallit. Edita. Helsinki.

Graham, T.C., Urnes, T. & Nejabi, R.1996. Efficient Distributed Implementation of Semi-Replicated Synchronous Groupware. Department of Computer Science York University. Luettavissa: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.34.4453&rep=rep1&type=pdf>. Luettu: 11.1.2013.

Gilmore, W. 2005. PHP & MySQL. Tehokas hallinta. Readme.fi. Gummerus. Jyväskylä.

Hakala, M., Vainio, M., & Vuorinen, O. 2006. Tietoturvallisuuden käsikirja. Docendo. Jyväskylä.

Heinisuo, R. & Rauta I. 2007. PHP ja MySQL. Tietokantapohjaiset verkkopalvelut. Talentum. Helsinki.

Hovi, A., Huotari, J. & Lahdenmäki, T. 2005. Tietokantojen suunnittelu & indeksointi. Docendo. Jyväskylä.

Huhtamäki, J. 2010. Luento 6. Kohti edistyneempää PHP-ohjelmointia. Tampereen teknillinen yliopisto, hypermedialaboratorio (HLab). Luettavissa: <http://matriisi.ee.tut.fi/hmopetus/hm-ohj/2011/pruju/06/esitys.php>. Luettu: 4.3.2013.

IEEE 2000. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE standard 1471–2000. Luettavissa: https://docs.google.com/viewer?a=v&q=cache:cVaWoHE-upMJ:www.win.tue.nl/~wsinmak/Education/2II45/software-architecture-std1471-2000.pdf+&hl=fi&gl=fi&pid=bl&srcid=ADGEESi_j3JVbt3VNxp5-NVY7dSv62WnnAeiCECP-aI2RKR7b5Bq5_zgo-Ue_hhB3oFjY1bWlwRxOgfXNheSjgyamh0y84rixFvJfXxiVXhY0oviv-T1zdfLZf2i4qQd-9jIrl0LKdu&sig=AHIEtbT04fT4YGquf71H7zVVhe6FbSf9Ag. Luettu: 3.1.2013.

Imperva 2012. Imperva's Web Application Attack Report. Luettavissa: http://www.imperva.com/docs/HII_Web_Application_Attack_Report_Ed3.pdf. Luettu: 12.2.2013.

Imperva 2013. Directory Traversal. Luettavissa: http://www.imperva.com/resources/glossary/directory_traversal.html. Luettu: 10.5.2013.

Korpela, J. 2011. HTML5. Uudet ominaisuudet. Docendo. WSOYpro. Jyväskylä

Koskimies, K. & Mikkonen, T. 2005. Ohjelmistoarkkitehtuurit. Talentum. Helsinki.

Lehtonen, J. & Siljander, K. 2010. Web-palveluiden tietoturva. Luettavissa:
<https://koppa.jyu.fi/kurssit/96852/luento/web-tietoturva>. Luettu: 6.2.2013.

Microsoft 2013. How to prevent cross-site scripting security issues. Luettavissa:
<http://support.microsoft.com/kb/252985>. Luettu: 5.2.2013.

Mitchell, L., Shafik, D. & Turland M. 2011. PHP Master: Write Cutting-Edge Code. SitePoint. Collingwood.

MySQL 5.5 Reference Manual 2013. 14 Storage Engines. Luettavissa:
<http://dev.mysql.com/doc/refman/5.5/en/storage-engines.html>. Luettu: 31.1.2013.

MySQL 5.6 Reference Manual 2013. 14 Storage Engines. Luettavissa:
<http://dev.mysql.com/doc/refman/5.6/en/storage-engines.html>. Luettu: 31.1.2013.

OWASP 2010a. OWASP Top 10 - 2010. The Ten Most Critical Application Security Risks. Luettavissa:
<http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202010.pdf>.
Luettu: 9.5.2013.

OWASP 2010b. Broken Authentication and Session Management. Luettavissa:
https://www.owasp.org/index.php/Broken_Authentication_and_Session_Management. Luettu: 9.5.2013.

OWASP 2011a. Cross-site Scripting (XSS). Luettavissa:
https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29.
Luettu: 12.2.2013.

OWASP 2011b. DOM Based XSS. Luettavissa:
https://www.owasp.org/index.php/DOM_Based_XSS. Luettu: 12.2.2013

OWASP 2011c. Session hijacking attack. Luettavissa:
https://www.owasp.org/index.php/Session_hijacking_attack. Luettu: 10.5.2013.

OWASP 2012. Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet. Luettavissa: https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29_Prevention_Cheat_Sheet. Luettu: 10.5.2013.

OWASP 2013. The Open Web Application Security Project. Luettavissa: https://www.owasp.org/index.php/Main_Page. Luettu: 9.5.2013.

Peltomäki, J. & Nykänen, O. 2006. Web-selainohjelmointi. Docendo. Jyväskylä.

Ping, Y., Kontogiannis, K. & Lau, T. 2004. Transforming Legacy Web Applications to the MVC Architecture. Luettavissa: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.8980&rep=rep1&type=pdf>. Luettu: 10.1.2013.

Qiu, X. 2004. Building Desktop Applications with Web Service in a Message-based MVC Paradigm, to appear. Electrical Engineering and Computer Science. Paper 63. Luettavissa: http://surface.syr.edu/cgi/viewcontent.cgi?article=1062&context=eecs&sei-re-dir=1&referer=http%3A%2F%2Fscholar.google.fi%2Fscholar%3Fstart%3D10%26q%3Dmvc%2Barchitecture%26hl%3Dfi%26as_sdt%3D0#search=%22mvc%20architecture%22. Luettu: 11.1.2013.

Qiu, X., Pallickara, S. & Uyar, A. 2004. Making SVG a Web Service in a Message-based MVC Architecture. Electrical Engineering and Computer Science. Paper 122. Luettavissa: http://surface.syr.edu/cgi/viewcontent.cgi?article=1121&context=eecs&sei-re-dir=1&referer=http%3A%2F%2Fscholar.google.fi%2Fscholar%3Fq%3Dvariations%2Bof%2Bmvc%2Barchitecture%26btnG%3D%26hl%3Dfi%26as_sdt%3D0&sei-re-dir=1&referer=http%3A%2F%2Fscholar.google.fi%2Fscholar%3Fq%3Dvariations%2Bof%2Bmvc%2Barchitecture%26btnG%3D%26hl%3Dfi%26as_sdt%3D0#search=%22variations%20mvc%20architecture%22. Luettu 10.1.2013.

Rantala, A. 2005. Web-ohjelmointi. Docendo. Jyväskylä.

Reenskaug, T. 2007. The original MVC reports. Luettavissa: <https://www.duo.uio.no/bitstream/handle/123456789/9621/Reenskaug-MVC.pdf?sequence=1>. Luettu: 9.1.2013.

Salihefendic, A. 2011. Model View Controller: History, theory and usage. Luettavissa: <http://amix.dk/blog/post/19615>. Luettu: 12.1.2013.

Schafer, S. 2005. Web Standards. HTML, CSS, JavaScript, Perl, Python and PHP. Programmer to Programmer. Wiley Publishing Inc. Indianapolis.

SC Magazine UK 2013. Use of cross-site scripting attacks massively increased at end of 2012. Luettavissa: <http://www.scmagazineuk.com/use-of-cross-site-scripting-attacks-massively-increased-at-end-of-2012/article/277959/>. Luettu: 6.2.2013.

Schwartz, B., Zaitsev, P. & Tkachenko, V. 2012. High Performance MySQL. 3. Painos. O'Reilly Media. Sebastopol. Luettavissa: http://www.google.fi/books?id=D0b_Xg3UeXEC&printsec=frontcover&hl=fi&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false. Luettu: 31.1.2013.

SLU 2010. Kansallinen liikuntatutkimus 2009 - 2010. Aikuisliikunta. SLU:n julkaisusarja 6/2010. Luettavissa: http://slu-fi-bin.directo.fi/@Bin/2ca228c132a632c63a22c145c8f09010/1357475287/application/pdf/3244706/Liikuntatutkimus_aikuiset_2009_2010.pdf. Luettu 6.1.2013.

The Next Web 2012. Hackers have a new favorite attack vector: Cross-site scripting up 69%. Luettavissa: <http://thenextweb.com/insider/2012/10/22/hackers-have-a-new-favorite-attack-vector-cross-site-scripting-up-69/>. Luettu: 6.2.2013

The PHP Group 2013a. What is PHP? Luettavissa: <http://php.net/manual/en/intro-what-is.php>. Luettu: 15.1.2013.

The PHP Group 2013b. Luettavissa <http://php.net/>. Luettu: 15.1.2013.

The PHP Group 2013c. What can PHP do? Luettavissa:
<http://www.php.net/manual/en/intro-whatcando.php>. Luettu 16.1.2013.

The PHP Group 2013d. Exceptions. Luettavissa:
<http://php.net/manual/en/language.exceptions.php>. Luettu: 28.1.2013.

The PHP Group 2013e. Superglobals. Luettavissa:
<http://php.net/manual/en/language.variables.superglobals.php>. Luettu: 29.1.2013.

The PHP Group 2013f. `$_GET`. Luettavissa:
<http://php.net/manual/en/reserved.variables.get.php> . Luettu: 29.1.2013.

The PHP Group 2013g. `$_POST`. Luettavissa:
<http://php.net/manual/en/reserved.variables.post.php>. Luettu: 29.1.2013.

The PHP Group 2013h. `$_SERVER`. Luettavissa:
<http://php.net/manual/en/reserved.variables.server.php>. Luettu: 29.1.2013.

The PHP Group 2013i. `setcookie`. Luettavissa:
<http://www.php.net/manual/en/function.setcookie.php>. Luettu 29.1.2013.

The PHP Group 2013j. `$_COOKIE`. Luettavissa:
<http://php.net/manual/en/reserved.variables.cookies.php> . Luettu: 29.1.2013.

The PHP Group 2013k. `$_SESSION`. Luettavissa:
<http://php.net/manual/en/reserved.variables.session.php> . Luettu: 29.1.2013.

The PHP Group 2013L. PHP Data Objects. Introduction. Luettavissa:
<http://www.php.net/manual/en/intro.pdo.php>. Luettu: 30.1.2013.

The PHP Group 2013m. PDO Drivers. Luettavissa:

<http://www.php.net/manual/en/pdo.drivers.php>. Luettu: 30.1.2013.

The PHP Group 2013n. session_regenerate_id. Luettavissa:

<http://www.php.net/manual/en/function.session-regenerate-id.php>. Luettu: 10.5.2013.

W3schools.com 2013a. HTML5 Introduction. Luettavissa:

http://www.w3schools.com/html/html5_intro.asp. Luettu: 2.2.2013.

W3schools.com 2013b. HTML DOM Tutorial. Luettavissa:

<http://www.w3schools.com/html/dom/default.asp>. Luettu: 1.2.2013.

W3schools.com 2013c. HTML DOM Events. Luettavissa:

http://www.w3schools.com/jsref/dom_obj_event.asp. Luettu: 3.2.2013.

W3schools.com 2013d. CSS Tutorial. Luettavissa:

<http://www.w3schools.com/css/default.asp>. Luettu: 1.2.2013.

W3schools.com 2013e. JavaScript Tutorial. Luettavissa:

<http://www.w3schools.com/js/default.asp>. Luettu: 2.2.2013.

W3schools.com 2013f. AJAX Introduction. Luettavissa:

http://www.w3schools.com/ajax/ajax_intro.asp. Luettu: 3.2.2013.

W3Techs 2013. World Wide Web Technology Surveys. Luettavissa:

http://w3techs.com/technologies/overview/programming_language/all. Luettu: 30.1.2013.

Zakas, N., McPeak, J. & Fawcett, J. 2007. Professional Ajax. 2. Painos. Programmer to Programmer. Wiley Publishing Inc. Indianapolis.

Liitteet

Liite 1. Rekisteröintilomake

TREENIKALENTERI.NET

Sähköpostiosoitteesi

Kirjaudu

ESITTELY HARJOITTELUN SUUNNITTELU KALENTERI HARJOITTELUN SEURANTA YHTEISÖ TUKI & OHJEET

REKISTERÖINTI

Yhteystietosi

Etunimi: Sukunimi:

Sähköposti: Salasana:

Kunta:

Fysiologiset perustiedot

Sukupuoli: mies nainen

Syntymäaika (pp.kk.vvvv): Pituus (cm):

Mulle näkyvät omat kuvaukset

Käyttäjänimi:

Lyhyt kuvaus harjoittelustani (alle 256 merkkiä, valinnainen):

Tavoitteeni (alle 256 merkkiä, valinnainen):

Liite 2. Asetukset

TREENIKALENTERI.NET

Tervetuloa Tero Testasjal

Kirjaudu ulos

OMAT ASETUKSET HARJOITTELUN SUUNNITTELU KALENTERI HARJOITTELUN SEURANTA YHTEISÖ TUKI & OHJEET

Salasanan vaihto

Muuta aloitussivua

Sykegrammi Muuta

- Et ole vielä valinnut aloitussivua
- Harjoitele säännöllisesti
- Muista levätä
- Älä harjoitele sairaana
- Kuukausinäkymä
- Viikkonäkymä
- Paivänäkymä
- Näytä kaikki harjoitukset
- Näytä kaikki indeksit
- Sykegrammi**
- Painogrammi
- Kilometrogrammi
- Kimppalenkit
- Ilmoitustaulu
- Ohjeita käyttäjälle

Liite 3. Harjoituksen tallentaminen

The screenshot shows the TREENIKALENTERI.NET website interface. At the top, there is a navigation menu with options: OMAT ASETUKSET, HARJOITTELUUN SUUNNITTELU, KALENTERI (highlighted), HARJOITTELUUN SEURANTA, YHTEISÖ, and TUKI & OHJEET. A user greeting "Tervetuloa Tero Testaaja!" is visible in the top right corner, along with a "Kirjaudu ulos" button.

The main content area features a calendar for the month of Toukokuu 2013. A modal window titled "TORSTAI 23.5.2013" is open, titled "Uusi harjoitus". The form contains the following fields and controls:

- Laji:** A dropdown menu with "Valitse tästä" selected.
- Harjoitusnimike:** A text input field.
- Alkoi:** A time selection field set to "Tunnit 15" and "Minuutit 00".
- Kesto:** A time selection field set to "00:00:00" with a "min" unit indicator.
- Kilometrit:** A distance selection field with a "km" unit indicator.
- Sujuvuus (1-5):** A rating selection field.
- Palautuminen (1-5):** A rating selection field.
- Harjoituksen kuvaus / tuntemukset:** A large text area with a character limit of "Korkeintaan 255 merkkiä".

At the bottom right of the form, there are two buttons: "Tallenna" (Save) and "Tyhjää" (Clear).

Liite 4. Indeksin tallentaminen

The screenshot shows the TREENIKALENTERI.NET website interface. At the top left is the logo with a running figure. The navigation menu includes: OMAT ASETUKSET, HARJOITTELUUN SUUNNITTELU, KALENTERI (highlighted), HARJOITTELUUN SEURANTA, YHTEISÖ, and TUKI & OHJEET. A user greeting 'Tervetuloa Tero Testaaja!' and a 'Kirjaudu ulos' button are in the top right. The main content area shows a calendar for 'Toukokuu 2013'. A modal window is open for 'TORSTAI 23.5.2013' with the title 'Uusi indeksi'. The modal contains the following fields and controls:

- Mittausaika:** Tunnit 15, Minuutit 00
- Paino:** A slider control with a red triangle and a '+' sign.
- Leposyke:** A slider control with a red triangle and a '+' sign.
- Muistiinpanoja:** A text input field with a character count 'Kokonaan 255 merkkiä' and 'Tallenna' and 'Tyhjää' buttons.

On the left side of the modal, there is a sidebar with a list of activities for the month of May, including 'Juoksu' (running) with details like 'kesto: 01:30:00', '18.5 km', '4.51,81 /km', 'sujuvuus: 4', 'Jalat vielä hieman jumissa.', and 'palautuminen: 2'.

TREENIKALENTERI.NET

Tervetuloa Tero Testaajel

Kirjaudu ulos

YHTEISÖ TUKEA & OHJEET

KESKIVIikko 8.5.2013 Harjoituksen muuttaminen

Laji: Juoksu Harjoitusnimike: TV-kevyt Alkoi: Tunnit 15 Minuutit 00

Kesto: 00:45:00 min Kilometrit: 8 km

Sujuvuus (1-5): 5 Palautuminen (1-5): 5

Harjoituksen kuvaus / tuntemukset:
OK!

Korkeintaan 255 merkkiä

KESKIVIikko 8.5.2013 Indeksien muuttaminen

Mittausaika: Tunnit 10 Minuutit 00

Paino: 59 kg

Leposyke: 51

Muistiinpanoja:

Korkeintaan 255 merkkiä

Tallenna Tyhjää

Liite 6. Listanäkymät



Tervetuloa Tero Testaaja!

Kirjaudu ulos

OMAT ASETUKSET HARJOITTELUUN SUUNNITTELU KALENTERI HARJOITTELUUN SEURANTA YHTEISÖ TUKE & OHJEET

	Laji	Harjoitusnimi	Aloitusajka	Kesto	Kilometrit	Kilometrivahti	Tunteukset	Sujuvuus	Palausminen
<input type="checkbox"/> x m	Juoksu	Vauhtitreeniä	10.05.2013 Mo 13:00:00	01:00:00	8	7.30,0	Verr. + 5 x 1000m		
<input type="checkbox"/> x m	Juoksu	TV-kevyt	08.05.2013 Mo 15:00:00	00:45:00	8	5.37,5	OK!	5	5
<input type="checkbox"/> x m	Juoksu		02.04.2013 Mo 19:00:00	01:15:00	15	5.00,0	Pää		
<input type="checkbox"/> x m	Juoksu	TV	06.05.2013 Mo 18:15:00	01:30:00	18,5	4.51,89	Jalat vielä hieman jumissa.	4	2
<input type="checkbox"/> x m	Kuntosaliharjoittelu		01.05.2013 Mo 13:00:00	01:30:00					
<input type="checkbox"/> x m	Kuntosaliharjoittelu		03.05.2013 Mo 13:00:00	01:30:00			Hyvä treeni! :-D	5	

Valitse kaikki



Tervetuloa Tero Testaaja!

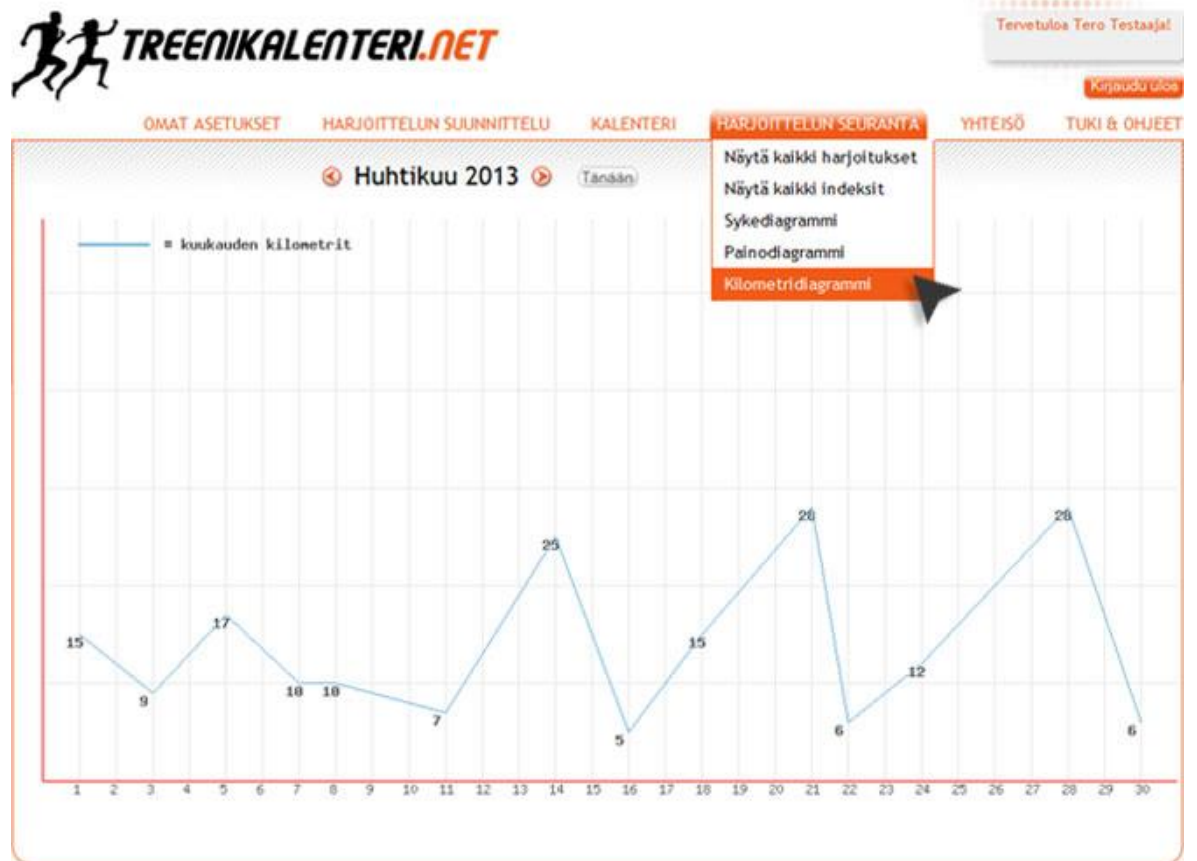
Kirjaudu ulos

OMAT ASETUKSET HARJOITTELUUN SUUNNITTELU KALENTERI HARJOITTELUUN SEURANTA YHTEISÖ TUKE & OHJEET

	Mittausajka	Paino	Leposyke	Muistiinpanoja
<input type="checkbox"/> x m	08.04.2013 Mo 08:00:00	75	50	
<input type="checkbox"/> x m	10.04.2013 Mo 10:00:00	76	54	
<input type="checkbox"/> x m	14.04.2013 Mo 08:08:00	75	51	Ok
<input type="checkbox"/> x m	01.05.2013 Mo 09:09:00	75	51	
<input type="checkbox"/> x m	08.05.2013 Mo 10:00:00	74.2	49	
<input type="checkbox"/> x m	11.05.2013 Mo 00:00:00	73.6	48	

Valitse kaikki

Liite 7. Kilometridiagrammi



Liite 8. Tietokannan taulut indeksit ja käyttäjät

indeksit

Taulun kommentit: indeksit

Sarake	Tyyppi	Tyhjä	Oletusarvo	Kommentit
id	int(11)	Ei		
kayttaja_id	int(11)	Ei		
mittausaika	datetime	Ei		
paino	float	Kyllä	NULL	
leposyke	int(11)	Kyllä	NULL	
muistiinpanoja	varchar(255)	Kyllä	NULL	

kayttajat

Taulun kommentit: oikeudet (0):käyttökiellossa; sukupuoli(0)=mies

Sarake	Tyyppi	Tyhjä	Oletusarvo	Kommentit
id	int(11)	Ei		
status	int(11)	Ei		0=kirj vahvistamatta, 1=peruskäyttt
sukupuoli	tinyint(4)	Ei		
kayttajanimi	varchar(20)	Ei		
oikeudet_id	tinyint(1)	Ei		
kunta_id	smallint(6)	Ei		
tavoite	varchar(255)	Ei		
etunimi	varchar(100)	Ei		
sukunimi	varchar(100)	Ei		
sahkoposti	varchar(200)	Ei		
syntyma_aika	varchar(10)	Ei		
pituus	int(3)	Ei		
kuvaus	varchar(255)	Ei		
salasana	varchar(100)	Ei		
kirjautumisaika	varchar(100)	Ei		
istuntotunnus	varchar(100)	Ei		

Liite 9. Tietokannan taulut kayttajat_lajit ja lajit

kayttajat_lajit

Taulun kommentit: kayttajat_lajit

Sarake	Tyyppi	Tyhjä	Oletusarvo	Kommentit
kayttaja_id	int(11)	Ei		
laji_id	int(11)	Ei		

lajit

Taulun kommentit: lajit

Sarake	Tyyppi	Tyhjä	Oletusarvo	Kommentit
id	int(11)	Ei		
lajinimi	varchar(100)	Ei		
matka_mahdollinen	tinyint(1)	Ei		

Liite 10. Tietokannan taulut asetukset ja harjoitukset

asetukset

Taulun kommentit: asetukset

Sarake	Tyyppi	Tyhjä	Oletusarvo	Kommentit
id	int(11)	Ei		
kayttaja_id	int(11)	Ei		
aloitussivu	varchar(100)	Kyllä	NULL	
omat_lajit	tinyint(1)	Kyllä	NULL	onko käyttäjällä omat lajit välitaulussa
kk_data	varchar(120)	Kyllä	NULL	kk-datan sisältö ja järjestys
vko_data	varchar(120)	Kyllä	NULL	vko-datan sisältö ja järjestys
pva_data	varchar(120)	Kyllä	NULL	pva-datan sisältö ja järjestys

harjoitukset

Taulun kommentit: harjoitukset

Sarake	Tyyppi	Tyhjä	Oletusarvo	Kommentit
id	int(11)	Ei		
kayttaja_id	int(11)	Ei		
laji_id	int(11)	Ei		
harjoitusnimike	varchar(30)	Kyllä	NULL	
aloitusaika	datetime	Ei		
kesto	time	Ei		
lopetusaika	datetime	Ei		
kilometrit	float	Kyllä	NULL	
kilometrivahti	varchar(10)	Kyllä	NULL	
tuntemukset	varchar(255)	Kyllä	NULL	
sujuvuus	tinyint(4)	Kyllä	NULL	
palautuminen	tinyint(4)	Kyllä	NULL	