Opinnäytetyö (AMK)

Tietojenkäsittely

Yrityksen tietojärjestelmät

2013

Petteri Kallanmaa

INFONÄYTTÖJÄRJESTELMÄN TOTEUTUS

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietojenkäsittely | Yrityksen tietojärjestelmät

Joulukuu 2013 | 41 sivua

Ohjaaja: Anne Jumppanen

Petteri Kallanmaa

INFONÄYTTÖJÄRJESTELMÄN TOTEUTUS

Opinnäytetyön tavoitteena on toteuttaa Turun ammattikorkeakoulun Lemminkäisenkadun toimipisteen käyttöön uusi infonäyttöjärjestelmä. Järjestelmän tarkoituksena on tehostaa yksikön sisäistä viestintää ja tarjota muuta hyödyllistä tietoa toimipisteen käyttäjille.

Opinnäytetyöraportissa selvitetään infonäyttöjärjestelmien perusteita, järjestelmän toteutukseen tarvittavia tekniikoita ja niiden valintaa sekä kehitysprosessin aikana ilmenneitä haasteita, käyttäen apuna useista eri lähteistä löytynyttä tietoa. Raportin lopussa pyritään myös selventämään toteutuneen järjestelmän toimintaa.

Työn tuloksena on valmis infonäyttöjärjestelmä. Toteutus on tietokantapohjainen web-sovellus, jonka toiminta pohjautuu paljolti Ajax-tekniikan käyttöön. Järjestelmä käsittää tietosisältöjen muokkaamiseen tarkoitetun hallintapaneelin sekä näistä sisällöistä koostuvan infonäyttöesityksen.

Järjestelmä saatiin valmiiksi ja se tullaan ottamaan käyttöön mahdollisimman pian. Työn tuloksista on hyötyä sekä toimeksiantajalle valmiin tuotteen muodossa että tekijälle arvokkaana työkokemuksena.

ASIASANAT:

infonäyttöjärjestelmä, infonäyttö, verkko-ohjelmointi, JavaScript, Ajax

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Business Information Technology | Business Information Systems

December 2013 | 41 pages

Instructor: Anne Jumppanen

Petteri Kallanmaa

THE IMPLEMENTATION OF A DIGITAL SIGNAGE SYSTEM

The objective of the present bachelor's thesis is to implement a new digital signage system for the Lemminkäisenkatu Campus of Turku University of Applied Sciences. The purpose of the system is to improve internal communication at the campus and provide other useful information for the students and personnel.

The thesis describes the basic principles of digital signage systems, techniques needed for the implementation and challenges appeared during the development process. Information found from varying sources was used as the basis for the study. The finished system is discussed at the end of the thesis.

The result is a complete digital signage system i.e. a database driven web application, which is largely based on the usage of Ajax-technique. The system comprises a control panel for information editing and a digital signage presentation that uses this information for creating the presentation.

The system was completed and it will be put into operation as soon as possible. The results will be useful both for the client as a functional product and for the creator as a valuable work experience.

KEYWORDS:

digital signage, web programming, JavaScript, Ajax

SISÄLTÖ

LIITTEET

KUVAT

KUVIOT

KÄYTETYT LYHENTEET JA SANASTO

Ajax	Ajax-tekniikka (Asynchronous JavaScript and XML) mahdollistaa selaimen ja palvelimen välisen kommunikoinnin ilman sivun uudelleenlatausta (Mozilla Developer Network 2013b).
CSS	Cascading Style Sheets on HTML-dokumenttien visuaalisen esityksen määrittelyyn käytettävä standardi (Flanagan 2011, 413).
DOM	Document Object Model on HTML-sisällön esittämiseen ja muokkaamiseen käytettävä rajapinta (Flanagan 2011, 361).
HTTP	Hypertext Transfer Protocol määrittelee, kuinka web-selaimet vastaanottavat ja lähettävät tietoa sekä miten palvelimet näihin pyyntöihin vastaavat (Flanagan 2011, 491).
Infonäyttöesitys	Infonäyttöesitys kierrättää näyttölaitteella hallintapaneelin kautta näytettäväksi asetettuja sisältösivuja.
JavaScript	JavaScript on web-ympäristössä käytettävä ohjelmointikieli, jonka avulla voidaan määritellä sivujen käyttäytymistä (Flanagan 2011, 1).
jQuery	jQuery on JavaScript-kirjasto, joka helpottaa muun muassa HTML-dokumentin elementtien manipuloimista ja Ajax-pyyntöjen lähettämistä (Flanagan 2011, 523).
MySQL	MySQL on relaatiotietokantoihin perustuva avoimen lähdekoodin tietokannan hallintajärjestelmä (Oracle 2013).
PDO	PHP-kielen PDO-kirjasto (PHP Data Objects) tarjoaa eri tietokantojen käyttämistä varten yhtenäisen abstraktiotason (Tatroe ym. 2013, 203).
PHP	PHP (PHP: Hypertext Preprocessor) on suosittu ohjelmointikieli, joka on suunniteltu erityisesti web-ohjelmointia varten (The PHP Group 2013).
REST	REST-arkkitehtuurityylillä (Representational State Transfer) resurssia käsitellään käyttäen sen URL-osoitetta sekä HTTP-metodeja (Vaswani 2012).
Sisältösivu	Sisältösivu on yksittäinen, infonäyttöesityksessä vuorollaan esitettävä sisältö.
SPA-sovellus	SPA-sovellukset (Single-Page Application) ovat web-sovelluksia, jotka toimivat yhdellä ja samalla sivulla tarvitsematta navigoimiseen uudelleenlatauksia (Osmani 2011).
XMLHttpRequest	JavaScript-kielen XMLHttpRequest-oliota käytetään palvelimen kanssa kommunikointiin osana Ajax-tekniikkaa (Mozilla Developer Network 2013b).

1 JOHDANTO

Nykypäivän teknologialla viestintää voidaan harjoittaa monenlaisten välineiden ja laitteiden avustuksella. Yksi tällaisista viestintävälineistä on nopeasti yleistyvä infonäyttö. Näyttöjä käytetään vaihtelevissa ympäristöissä, kuten kauppakeskuksissa, hotelleissa, yritysten toimitiloissa ja oppilaitoksissa välittämään informaatiota organisaation tärkeinä pitämistä asioista. Sisällöntuotantovälineiden tullessa yhä helppokäyttöisemmiksi ja näyttölaitteiden yhä edullisemmiksi, kiinnostus tämänkaltaista viestintämuotoa kohtaan kasvaa. Tällaiseen tarpeeseen myös opinnäytetyöni pyrkii vastaamaan.

Sain syyskuussa 2013 toimeksiannon Turun ammattikorkeakoululta Lemminkäisenkadun toimipisteen infonäyttöjärjestelmän uudistamiseksi. Käytössä olleen edeltävän järjestelmän käyttö koettiin joiltain osin hankalaksi ja näyttölaitteille kaivattiin myös uudenlaisia sisältöjä. Toteutettavan järjestelmän tarkoituksena on tarjota kaikille toimipisteen käyttäjille tietoa tämänpäiväisistä ja tulevista tapahtumista sekä muuta hyödyllistä informaatiota.

Projekti sai alkunsa kouluni T&K-projektikurssilla, jossa meille esiteltiin erilaisia opintojakson puitteissa toteutettavia projekteja. Valitsin niistä tämän. Koin aiheen mielenkiintoiseksi, koska se sisälsi web-ohjelmointia hieman erilaisesta näkökulmasta. Tein opintojakson yhteydessä aiheesta esiselvityksen, jossa kartoitin käyttäjävaatimuksia ja rajoitteita järjestelmälle sekä arvioin vaatimusten toteuttamiskelpoisuutta. Tämä selvitys on toimitettu sähköisenä projektin ohjausryhmän jäsenille. Itse järjestelmän toteutus laajeni opinnäytetyöksi.

Tässä raportissa käsittelen infonäyttöjärjestelmän toteutusta selainkäyttöisenä, Ajax-tekniikkaan vahvasti pohjautuvana web-sovelluksena. Sovellus asennetaan koulun verkossa olevalle palvelimelle, ja infonäyttöinä toimivat TV-ruudut saavat sisällön niihin liitetyistä tietokoneista. Nämä osat kokonaisuudesta ovat jo ennalta olemassa ja toimiviksi todettuja, joten ne on rajattu tämän tarkastelun ulkopuolelle.

1.1 Tavoitteet

Opinnäytetyön tavoitteena on tuottaa Turun ammattikorkeakoulun Lemminkäisenkadun toimipisteen käyttöön uusi infonäyttöjärjestelmä, joka parantaa toimipisteen sisäisen viestinnän tehokkuutta sekä tarjoaa opiskelijoita ja henkilökuntaa kiinnostavaa sisältöä. Järjestelmän esittämän tietosisällön muokkaaminen tulisi onnistua helppokäyttöisen hallintapaneelin kautta. TV-ruuduilla näytettävän infonäyttöesityksen selkeyteen ja näkyvyyteen tulee kiinnittää huomiota, jotta sisältö olisi helposti luettavaa.

Omana tavoitteenani on saada aikaan mahdollisimman hyvin käyttäjien toiveita palveleva lopputulos. Toivon, että työstäni on hyötyä myös muille opiskelijoille ja mahdollisille järjestelmän jatkokehittäjille. Omien taitojen kehittäminen ja kokemuksen kartuttaminen ovat myös tärkeitä tekijöitä.

1.2 Tutkimusongelma ja -menetelmät

Työni tutkimusongelma on selvittää, miten tuottaa edellä kerrottuja tavoitteita ja käyttäjien toiveita vastaava lopputulos, ja lopuksi toteuttaa sellainen. Ongelmaan pyritään vastaamaan perehtymällä infonäyttöjärjestelmien toimintaan ja toteutuksiin sekä pohtimalla, miten ohjelma kannattaisi laatia; minkälaisia tekniikoita ja ohjelmakirjastoja toteutuksessa tulisi käyttää, ja miten niitä sovelletaan käytäntöön.

Pääongelman lisäksi olen käsitellyt muutamia toteutukseen liittyneitä pienempiä tutkimuksen osaongelmia. Nämä ovat kehitysprosessin aikana ilmenneitä teknisiä haasteita, jotka olivat toteutuksen ja oppimisen kannalta oleellisia.

Opinnäytetyön tutkimusmenetelmänä on tapaustutkimus, tutkimusote on konstruktiivinen. Selvitys toteutetaan käyttäen valmiita aineistoja: verkko- ja kirjalähteitä. Aiemmat infonäyttöihin ja web-ohjelmointiin liittyvät opinnäytetyöt ovat olleet hyödyllisiä lähdemateriaalin etsimisessä.

1.3 Työn sisältö ja rakenne

Projektin laajuuden vuoksi opinnäytetyöraportin rakennetta ja sisältöä täytyi erityisen tarkasti pohtia. Järjestelmän toteutusta ei jokaiselta toiminnoltaan ja ohjelmatiedostoltaan pysty tässä yhteydessä selventämään.

Päädyin lopulta käsittelemään projektin sisältöä oppimisprosessin kannalta keskeisten sisältöjen ja ongelmien näkökulmasta. Keskityn raportissani näihin osaongelmiin sen sijaan, että pureutuisin tarkasti järjestelmän rakenteeseen. Nyt esiteltävät asiat ovat myös sellaisia, joilla oli itselleni eniten uutuusarvoa, ja toivottavasti lukijallekin.

Opinnäytetyön teoriaosuus koostuu infonäyttöjärjestelmien yleisten toimintaperiaatteiden ja käyttötilanteiden selvittämisestä sekä järjestelmän toteutukseen tarvittujen tekniikoiden ja ohjelmakirjastojen kuvauksesta. Soveltavassa osuudessa esittelen edellä käsitellyn tutkimustyön pohjalta tuotetun valmiin järjestelmän toimintaa. Raportin lopuksi esitän ideoita järjestelmän jatkokehitykselle ja päätän pohdintaan työn onnistumisesta.

2 INFONÄYTTÖJÄRJESTELMÄT

Infonäyttöjärjestelmät ovat organisaation viestinnässä käytettäviä välineitä, joiden avulla voidaan välittää muun muassa tiedotteita, mainoksia ja aikataulutietoja (InfoSign Oy 2008). Näyttöjä käytetään myynnin kasvattamiseen, yrityskuvan parantamiseen, mainostilan myymiseen kolmansille osapuolille, viihdykkeen tarjoamiseen sekä sisäiseen ja julkiseen viestintään (Yackley 2011, 6–7). Järjestelmän käyttökohteina voivat olla yritysten toimitilat, oppilaitokset, hotellit, julkiset kulkuneuvot ja kauppakeskukset (InfoSign Oy 2008).

Toisin sanoen, infonäytöissä on kyse viestintämuodosta, jossa näyttölaitteilla esitetään tiedotusmateriaalia jonkinlaista hallintajärjestelmää apuna käyttäen. Näyttölaitteita ovat esimerkiksi LCD-televisiot, videotykit, pöytätietokoneiden näytöt tai vaikka valokuvakehykset. Hallintajärjestelmän avulla näytölle saadaan siirrettyä sisältöjä, kuten kuvia, videoita, hinnastoja tai opasteita. (First Technology Oy 2011.) Infonäytöt voivat myös hakea päivittyvää tietoa automaattisesti tietokannoista ja internetistä. Sisältöjä voidaan tuoda esimerkiksi RSS-syötteinä. (Nordlund 2008, 27.)

Viestintävälineelle ei ole muodostunut yhtä vakiintunutta suomenkielistä käsitettä, joten asiasta puhuttaessa käytetään monenlaisia termejä. Nimityksiä ovat muun muassa sähköiset tiedotusjärjestelmät, digital signage, info-TV, infojärjestelmät, mainosnäytöt, infonäytöt, älynäytöt sekä digitaaliset julisteet. Myös monet kosketus- ja aikataulunäytöt voidaan määritellä samaan kategoriaan. (First Technology Oy 2011.)

Viime vuosiin asti tekniikka ei ole ollut kovinkaan toteuttamiskelpoinen tai taloudellisesti kannattava. Näytöt ovat olleet liian kalliita, liian kookkaita ja ne kuluivat loppuun liian nopeasti. Investointiin nähden hyöty ei ole ollut tarpeeksi suuri. LCD- ja plasmaruutujen tuoma muutos on aiheuttanut sen, että näytöt ovat riittävän edullisia, jotta ne voivat kilpailla kustannuksissa tulostettujen julisteiden kanssa. Modernit näytöt ovat ohuita, niitä voidaan ripustaa seinille, ja ne voivat kommunikoida tietoverkoissa ja hakea niistä sisältöä. (Yackley 2011, 1.)

Infonäytöille on olemassa monenlaisia käyttökohteita. Näyttöjä käytetään muun muassa

- vähittäismyynnissä välittämään asiakkaille tietoa myymälän tarjouksista, ohjaamaan asiakasvirtoja sekä esittämään yritystä koskevia ilmoituksia
- yrityksissä aulatilojen tervetulotoivotuksiin, tuoteilmoituksiin, työntekijäviestintään ja sisäiseen koulutukseen
- pankeissa esittämään korkotasoja, tuoteinformaatiota ja yritystietoa
- lentokentillä ja linja-autoasemilla pitämään matkustajat tietoisina lähtö- ja saapumisajoista tarjoten samalla mainosvälineen liikkeille ja ravintoloille
- hotelleissa tarjoamaan vierailijoille hyödyllistä tietoa tilojen sijainneista ja aukioloajoista sekä lähialueen palveluista, vähentäen näin asiakaspalvelun tarvetta
- viihdetilaisuuksissa luomaan asiakaskokemuksia, jotka ovat muuhun ilmapiiriin ja tunnelmaan sopivia
- terveysasemilla tarjoamaan opastusta sekä viihdykettä potilaille odotustiloissa
- oppilaitosten ruokaloissa, auloissa ja muissa tiloissa tarjoamaan tietoa tulevista tapahtumista sekä tiedottamaan poikkeustilanteista (Yackley 2011, 1–4).

Kuvassa 1 on esitetty joitain infonäyttöjen todellisia käyttökohteita: suuri ulkonäyttö mainosten välittämiseen ja tiedotukseen, mainosnäytöt kauppakeskuksessa sekä linja-autojen pysäkkiaikataulut.

Kuva 1. Esimerkkejä infonäyttöjen käyttökohteista.

Infonäyttöjärjestelmien käyttökohteet voidaan karkeasti jakaa mainosviestintään sekä yritys- ja organisaatioviestintään. Mainosnäyttöjärjestelmät keskittyvät nimensä mukaisesti mainosten välittämiseen. Näiden käyttö on yleistynyt etenkin kauppakeskuksissa. Sisällöt ovat video- ja kuvasisältöjä, jotka saadaan usein valmiina mainostoimistoilta ja mainostajilta. (Nordlund 2008, 26.)

Yritys- ja organisaatioviestinnässä infonäyttöjärjestelmien käyttö keskittyy tavallisesti sisäiseen tiedonvälitykseen. Sisältö ei useinkaan ole mainostoimistojen tuottamaa, joten sisältöjen luomiseen käytettävien työkalujen helppokäyttöisyyteen tulee kiinnittää erityistä huomiota. (Nordlund 2008, 26.)

Infonäyttöjä voidaan jaotella myös käyttötilanteiden mukaan. Ne määräävät laitteiden sijoittelun ja viestinnän tyylin. Myyntitilanteisiin (POS, point of sale) tarkoitettuja näyttöjä käytetään myytävien tuotteiden ja palveluiden läheisyydessä. Tämäntyyppisten järjestelmien vahvuus on toimintapyynnön välittömyys: näytöt on sijoitettu sinne missä kuluttajat tekevät ostopäätöksensä. Sisältö on huomiota herättävää ja relevanttia. (Kelsen 2010, 3.)

Ohikulkutilanteisiin (POT, point of transit) käytettäviä näyttöjä on tienvarsilla, myymälöiden näyteikkunoissa ja ylipäätään paikoissa, joissa on paljon ohikulkuliikennettä. Tavoitteena on visuaalisesti houkuttelevia sisältöjä esittämällä viedä ohikulkijoiden huomio hetkeksi. (Kelsen 2010, 4.)

Odotustilanteisiin (POW, point of wait) suunnatut laitteet sijoitetaan tyypillisesti paikkoihin, joissa ihmiset odottavat jotakin palvelua tai joihin muodostuu jonoja. Näyttöjä voidaan asentaa esimerkiksi hissien yhteyteen. (Kelsen 2010, 4.)

Tämän toteutuksen kohteessa, Lemminkäisenkadun toimipisteessä, on kolme infonäyttöinä käytettävää televisiota. Järjestelmän tarkoituksena on keskittyä ensisijaisesti organisaatioviestintään. Näytöt on sijoitettu pää- ja sivusisäänkäynnin sekä rakennuksen keskellä olevan käytävän yhteyteen. Edellistä jaottelua soveltaen, näytöt on tarkoitettu pääasiassa ohikulkijoille ja osin myös odotustilanteisiin. Yksi näytöistä ei ole aivan kulkuväylän vieressä, eikä sitä myöskään pääse helposti tarkastelemaan lähempää, joten sisällön näkyvyyteen liittyvä haaste on oleellinen tässäkin mielessä.

3 KÄYTETYT TEKNIIKAT

Infonäyttöjärjestelmiä on olemassa sekä web-sovelluksina että paikallisella tietokoneella toimivina ohjelmistoina. Ensin mainitun etuna on se, ettei käyttäjien koneille tarvitse asentaa erillisiä ohjelmia. Tällöin myös järjestelmän käyttöönotto ja päivittäminen onnistuvat helpommin. (InfoSign Oy 2008.)

Tämä toteutus on tehty web-sovelluksena, koska se koettiin luontevimmaksi vaihtoehdoksi ja minulla oli ohjelmoinnin alalta paras osaaminen nimenomaan verkkopalveluista. Myös edeltävä järjestelmä oli web-sovellus. En kuitenkaan käyttänyt tätä projektini pohjana, vaan aloitin puhtaalta pöydältä.

Aluksi oli itsellenikin suurelta osin epäselvää minkälaisilla tekniikoilla ja ohjelmakirjastoilla lähtisin viemään projektia eteenpäin. Palvelinpuolen ohjelmointikielen ja tietokannan valinta oli vielä melko yksinkertaista, koska näistä nyt valituksi tulleista minulla oli eniten kokemusta.

Projektin edetessä tuli eteen monia tilanteita, joissa piti pystyä tekemään rajanvetoa sen suhteen, mitä on järkevää toteuttaa itse ja mihin ongelmiin kannattaisi ottaa käyttöön valmiita ratkaisuja. Ohjelmakirjastoja ja muita lisäosia valittiin projektiin siis sitä mukaa kun niille tuli tarve. Erilaisten kirjastojen käyttöönotto vaatii kuitenkin paljon selvitystä ja opettelua, joten aina ei ole suinkaan selvää, tuottavatko ne halutun ratkaisun nopeammin. Kaikkea ei myöskään voi yhden opinnäytetyön puitteissa opetella.

Selvitin muun muassa tietokantakyselyjen yksinkertaistamiseen käytettävien ORM-kirjastojen (Object-relational mapping) sekä erilaisten sivupohjajärjestelmien käyttöönottoa. Nämä olisivat kuitenkin heikentäneet järjestelmän ylläpidettävyyttä, koska kullakin on omanlaisensa syntaksi, joka vaatii jonkin verran opettelua. Kyseiset lisätoiminnot eivät myöskään ole mitenkään välttämättömiä; ne vain helpottavat joidenkin tiettyjen ominaisuuksien toteuttamista, syntaksin tultua tutuksi. Samaa asiaa pohdin ohjelmistokehyksen käyttöönottoa harkitessani. Päädyin siinä kuitenkin erilaiseen ratkaisuun, koska koin siitä olevan enemmän hyötyä.

Olen pyrkinyt käyttämään suosittuja ja hyvin dokumentoituja ratkaisuja. Kaikki käyttämäni tekniikat ja kirjastot ovat ilmaisia ja avoimen lähdekoodin lisenssillä lisensoituja. Täydellinen lista käyttämistäni ohjelmakirjastoista ja muista lisäosista on liitteessä 1.

Tämä järjestelmä on toteutettu käyttäen PHP-ohjelmointikieltä ja MySQL-tietokantaa. Projektin alkuvaiheessa otin käyttöön Slim Framework -ohjelmistokehyksen, joka helpottaa muun muassa URL-osoitteiden reititystä. Toteutus ei noudata MVC-ohjelmistoarkkitehtuuria, mutta käyttöliittymä ja toimintalogiikka on pyritty pitämään erillään. Varsin luonteva oli myös valinta Ajax-tekniikan käytöstä järjestelmän hallintapaneelin ja infonäyttöesityksen yhteydessä.

Seuraavissa alaluvuissa käsittelen tarkemmin ohjelman toteutukseen valitsemiani tekniikoita; mitä ne ovat, mihin niitä käytetään ja miksi olen näihin valintoihin päätynyt.

3.1 PHP ja PDO-kirjasto

PHP on suosittu ohjelmointikieli, joka on suunniteltu web-ohjelmointia varten, mutta jota voidaan käyttää myös yleiskäyttöisenä ohjelmointikielenä (Tatroe ym. 2013, 1). PHP-kielellä voidaan ohjelmoida sekä olioperustaisesti että proseduraalisesti (Tatroe ym. 2013, 147). W3Techs-tutkimuslaitoksen mukaan PHP:n käyttöosuus palvelinpuolen ohjelmointikielistä on 81,4 prosenttia (W3Techs 2013a).

Sisällön luomiseksi tarvitaan PHP-prosessointimoduuli sekä web-palvelin, joka palauttaa pyydetyt tiedot. Useimmiten tuloksena on HTML-dokumentti, mutta PHP:n avulla voidaan tuottaa myös XML-dokumentteja, grafiikkaa, PDF-tiedostoja ja jopa Flash-animaatioita. (Tatroe ym. 2013, 1–2.)

Yksi kielen tärkeimmistä ominaisuuksista on sen laaja tuki erilaisille tietokannoille. PHP tukee kaikkia suosituimpia tietokantoja. Näitä ovat muun muassa MySQL, PostgreSQL, Oracle, MS-SQL, SQLite ja MongoDB. PHP:lla dynaami-

sen sisällön hakeminen tietokannasta web-sivulle onkin tehty erityisen helpoksi. (Tatroe ym. 2013, 2.)

PHP voi käsitellä tietokantoja tietokantakohtaisten laajennosten tai tietokanta-riippumattoman PDO-kirjaston (PHP Data Objects) avulla. PDO tarjoaa abstraktiotason, jonka avulla tietokannan hallintajärjestelmän vaihtaminen onnistuu yhtä riviä muuttamalla. Tämä ei kuitenkaan tee SQL-kyselyistä siirrettäviä, jos niissä on käytetty tietokantakohtaisia piirteitä. (Tatroe ym. 2013, 203–204.)

Yksi mielenkiintoinen PDO:n mahdollistama menetelmä on olioiden tuottaminen suoraan tietokantataulun rakenteeseen perustuen. Tätä varten hakumetodille annetaan parametreina PDO::FETCH_CLASS-vakiomuuttuja ja halutun luokan nimi, jolloin tiedot palautetaan tässä muodossa. Taulun sarakkeiden nimien tulee vastata luokan olion attribuutteja. (Wurzer 2012.)

Olen käyttänyt edellä kuvattua menetelmää hyödyksi toteutuksessani. Kuvan 2 mukainen esimerkkikoodi palauttaa result-muuttujaan tuloksen User-luokan mukaisena oliona. Tämän voi myös asettaa oletushakumenetelmäksi, jolloin sitä ei tarvitse hakumetodille erikseen ilmoittaa.

```
1 $query  = $pdo->query($sql);
2 $result = $query->fetch(PDO::FETCH_CLASS, 'User');
```

Kuva 2. PDO:n fetch-metodi palauttaa tuloksen oliona.

PHP on hyvin vakiintunut ja suosittu kieli web-ohjelmoinnissa. Suuren suosion takia sille on olemassa suuri määrä erilaisia ohjelmistokehyksiä, kirjastoja ja muita lisäosia. Aiheesta on myös olemassa runsaasti opastusmateriaaleja sekä valmiita ratkaisuja erilaisiin ongelmiin. Tämän lisäksi minulla oli eniten kokemusta juuri kyseisestä ohjelmointikielestä, joten oli luontevaa päätyä käyttämään sitä tässäkin projektissa.

3.2 MySQL

MySQL on relaatiotietokantoihin perustuva ilmainen, avoimen lähdekoodin tietokannan hallintajärjestelmä. Tietokannan hallintajärjestelmillä lisätään, haetaan ja prosessoidaan tietokannoissa säilytettävää tietoa. Relaatiotietokannoissa oleva tieto on järjestetty tauluihin, joilla on keskinäisiä suhteita eli relaatioita. (Oracle 2013.) MySQL:n arvioidaan olevan toiseksi suosituin tietokannan hallintajärjestelmä, ja ilmaisista vaihtoehdoista kaikkein suosituin (DB-Engines 2013).

Järjestelmään syötetty tieto on myöhempää käyttöä varten tallennettava johonkin, ja tietokanta on aivan ilmeinen vaihtoehto tässä. Tietokannat tallentavat tiedot semanttisessa muodossa, josta ne on helppo ottaa käyttöön. Edellisessä alaluvussa kuvattuun tapaan taulujen sisältöjä voidaan helposti käsitellä olioina.

Kuten PHP:n tapauksessa, päädyin MySQL:ään, koska se oli ennestään tuttu ja suuren suosion takia sen käyttöön on olemassa paljon dokumentaatiota. Toinen vaihtoehto olisi voinut olla SQLite-tietokanta. Yksittäiseen tiedostoon perustuvana sillä olisi ollut etuja siirrettävyyden suhteen, mutta eroavaisuudet tehokkuuden, ominaisuuksien ja SQL-syntaksin suhteen vaatisivat tarkempaa selvittämistä ja hyötyjen arvioimista.

3.3 JavaScript ja jQuery

JavaScript on web-ympäristössä käytettävä ohjelmointikieli, joka on käytössä suurimmassa osassa verkkosivuista. Kaikki modernit web-selaimet sisältävät JavaScript-tulkin. JavaScript tuo sivun sisältöä ja rakennetta kuvaavan HTML-kielen ja tyylejä kuvaavien CSS-tyyliohjeiden rinnalle keinon määritellä sivujen käyttäytymistä. (Flanagan 2011, 1.)

JavaScript tarjoaa välineitä selainohjelman hallintaan ja sivun elementeistä koostuvan DOM-dokumenttipuun muokkaukseen. Kielen avulla voidaan muun muassa vastata hiiren napsautuksiin, lomakesyötteisiin, sivunavigointiin ja muihin käyttäjän toimiin. JavaScriptiä voidaan käyttää myös selainympäristöjen ul-

kopuolella, palvelinympäristöissä, mutta tätä mahdollisuutta en käsittele työssäni. (Mozilla Developer Network 2013a.)

JavaScript kärsii selainten eroavaisuuksien aiheuttamasta epäyhteensopivuudesta, erityisesti vanhojen selainten osalta. Usein tähän käytetään ratkaisuna JavaScript-ohjelmistokehyksiä tai -työkalukirjastoja, jotka yksinkertaistavat toistuvia tehtäviä ja piilottavat selainten välisiä eroja. (Flanagan 2011, 523.)

Suosituin tähän tarkoitukseen soveltuva kirjasto on jQuery. W3Techs-tutkimuslaitoksen mukaan sen osuus käytetyistä JavaScript-kirjastoista on tällä hetkellä 92,6 prosenttia, ja kaiken kaikkiaan se on käytössä 57 prosentissa kaikista web-sivuista. (W3Techs 2013b.)

jQuery-kirjaston avulla on vaivatonta etsiä dokumentista halutut elementit ja manipuloida niitä lisäämällä sisältöä tai muokkaamalla HTML-attribuutteja ja CSS-tyylejä. Sen avulla voidaan myös määritellä tapahtumankäsittelijöitä ja esittää animaatioita. Lisäksi jQuery tarjoaa yksinkertaisen ja selainyhteensopivan syntaksin Ajax-pyyntöjen tekemiseen. (Flanagan 2011, 523.)

Nimensä mukaisesti jQuery-kirjaston käyttö perustuu kyselyihin. Tyypillinen kysely hyödyntää CSS-tyyliohjekielestä tuttuja valitsimia haluttujen elementtien paikallistamiseen. Kysely palauttaa olion, joka esittää löydettyjä elementtejä. Tämä olio tarjoaa monia hyödyllisiä metodeja, joilla löydettyjä elementtejä voidaan käsitellä ryhmänä. (Flanagan 2011, 523.)

Käyttäjäystävällisten ja sovellusmaisten sivustojen rakentamiseen JavaScript-kielen käyttö on lähes välttämättömyys, ja jQuery helpottaa sen tuottamista huomattavasti korjaten samalla selainyhteensopivuuksien aiheuttamia ongelmia. Edellä mainituista syistä olen päätynyt näitä myös tässä käyttämään.

Lisäksi verkosta löytyy hyvin paljon jQuery-kirjastoa käyttäviä lisäosia. Näitä ovat esimerkiksi erilaiset lomakkeenkäsittelijät, ajan- ja värinvalitsimet sekä muut pienet työkalut. Tässäkin projektissa on käytetty muutamia jQuery-lisäosia. Projektin laajuuden takia näitä ei ole mahdollista käsitellä tarkemmin tämän työn puitteissa, ne ovat kuitenkin listattuina liitteessä 1.

3.4 Ajax

Ajax on akronyymi sanoista Asynchronous JavaScript and XML. Käytännössä termillä tarkoitetaan JavaScript-kielen XMLHttpRequest-olion käyttämistä kommunikointiin palvelinpuolen ohjelmointikielen kanssa. Nimestään huolimatta menetelmän avulla voidaan lähettää ja vastaanottaa tietoa palvelimen ja selaimen välillä XML-muodon lisäksi monissa muissa formaateissa, kuten JSON- ja HTML-muodoissa sekä tekstitiedostoina. (Mozilla Developer Network 2013b.)

Ajaxin kiinnostavuus liittyy sen asynkroniseen luonteeseen, jolla tarkoitetaan selaimen ja palvelimen välisen kommunikoinnin tapahtumista ilman sivujen uudelleenlatausta. Tämä antaa sivustolle mahdollisuuden päivittää vain joitakin tiettyjä osiaan käyttäjien toimiin perustuen. (Mozilla Developer Network 2013b.) Sivun uudelleenlatausten välttäminen mahdollistaa web-sovellukset, joiden käyttökokemus muistuttaa enemmän perinteisiä työpöytäsovelluksia (Flanagan 2011, 491).

jQueryn käyttö yksinkertaistaa Ajax-pyyntöjä lyhentämällä merkittävästi tarvittavan ohjelmakoodin määrää. Kuvan 3 esimerkissä lähetetään GET-pyyntö palvelimelle, napsautetun linkin mukaiseen osoitteeseen. Linkille on asetettu tapahtumankäsittelijä, joka käynnistää pyynnön. Kuvan koodilohko asetetaan tapahtumankäsittelijän sisään. Palautuksena saatu sisältö sijoitetaan dokumentin main-elementtiin.

```
1 $.ajax({
2     type: 'GET',
3     url: $(this).attr('href'),
4     success: function(data) {
5         $('#main').html(data);
6     }
7 });
```

Kuva 3. Ajax-pyyntö jQuery-kirjastoa käyttäen.

Ajax-tekniikkaa käytetään monenlaisiin käyttötarkoituksiin. Sen avulla voidaan toteuttaa

- reaaliaikainen lomakkeentarkistus esimerkiksi käyttäjätunnusten, sarjanumeroiden, tarjouskoodien tai postinumeroiden osalta ennen lomakkeen lähetystä
- tietojen automaattinen täydennys lomakekenttään käyttäjän kirjoittaessa esimerkiksi sähköpostiosoitettaan, nimeään tai paikkakuntaa
- selaintapahtumiin, kuten sivun vieritykseen, perustuva sisällön lataus, jolloin voidaan hakea lisää tietoja taustalla sallien selaimen ladata aluksi sivut nopeammin
- kehittyneemmät käyttöliittymätoiminnot ja -efektit, kuten puurakenteet, valikot, tietotaulukot, kalenterit ja edistymispalkit, jotka sallivat paremman vuorovaikutuksen käyttäjän kanssa
- sivun reaaliaikainen päivitys, jolloin erilaiset tiedot, kuten pistetilanteet, pörssikurssit ja säätiedot, voidaan noutaa ilman sivun uudelleenlatausta
- lomakkeiden lähetykset ilman tarvetta koko sivun päivitykselle
- tietojen lataus sovellukseen kolmansien osapuolten tarjoamista sisällöistä, kuten esimerkiksi Google Maps -karttapalvelusta
- yhden sivun sovelluksia (Single-Page Application), jotka muistuttavat käyttökokemukseltaan tavallisia työpöytäohjelmia (Murray 2005).

3.5 SPA-sovellukset

Yhden sivun sovellukset (Single-Page Application) eli SPA-sovellukset ovat web-sovelluksia tai -sivustoja, jotka toimivat yhdellä ja samalla sivulla tarvitsematta uudelleenlatausta sivulla navigoimiseen. Tarvittava sisältö haetaan joko paikallisesta muistista tai pyydetään web-palvelimelta. (Osmani 2011.)

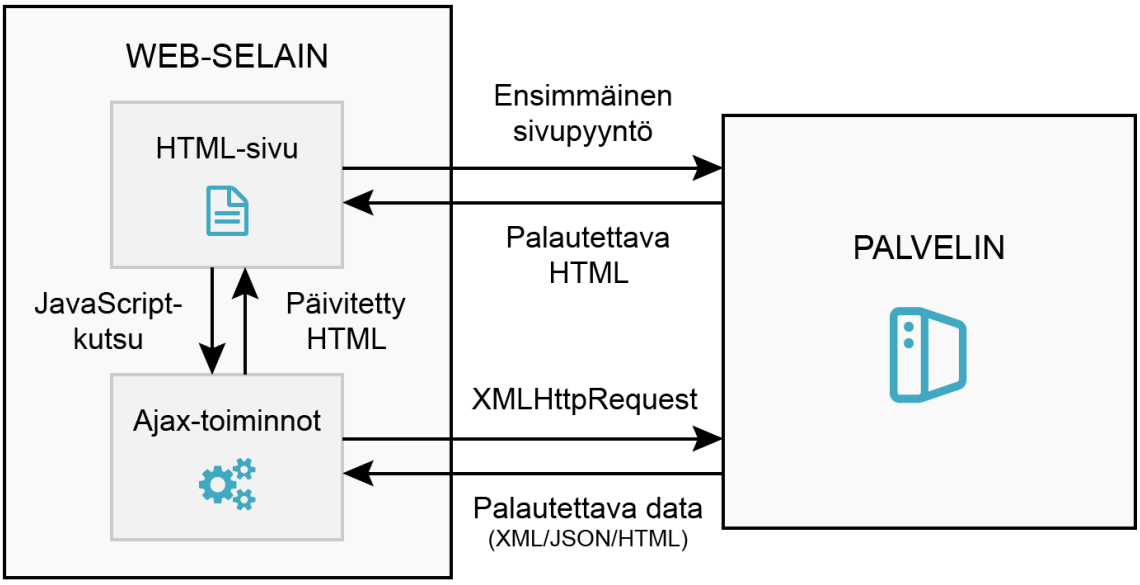
Usein SPA-sovellukset rakennetaan käyttäen valmiita JavaScript MVC -ohjelmistokehyksiä ja palvelinkommunikointiin REST-rajapintaa (Representational State Transfer). Nämä eivät kuitenkaan ole välttämättömiä. Tar-

koitukseen soveltuvista kehyksistä suosituimpia ovat Backbone.js, AngularJS ja Ember.js. (Heiskanen 2013.)

Perinteisten, useista yksittäisistä sivuista koostuvien, web-sovellusten haittana on käyttökokemuksen häiriintyminen sivulta toiselle siirryttäessä. Tällöin sivun koko sisältö latautuu aina uudelleen valikkoineen sekä ylä- ja alaotsakkeineen. SPA-sovelluksissa muutokset ohjelman tilaan hoidetaan yleensä käyttäen aiemmin esiteltyjä XMLHttpRequest-pyyntöjä. Näin saadaan käyttökokemuksesta huomattavasti sulavampi. (Osmani 2011.)

Pyynnöt palauttavat tiedon HTML-muodossa tai raakadatana eli XML- tai JSON-muodossa. Jos palvelin palauttaa tiedon HTML-muodossa, JavaScript-koodi sijoittaa sen suoraan paikalleen DOM-dokumenttipuussa. Kun palautetaan raakadataa, JavaScript prosessoi ja muuntaa datan ensin HTML-koodiksi, jota sitten käytetään DOM-puun päivitykseen. (Single-page application 2013.)

Kuviossa 1 on selitetty SPA-sovelluksen toimintaperiaate. Se vastaa myös Ajax-tekniikan toimintaa yleisemminkin, sillä erotuksella, että tätä SPA-sovellusten erikoistapausta lukuun ottamatta, koko dokumentti ladataan aina uudestaan sivulta toiselle siirryttäessä.



Kuvio 1. SPA-sovelluksen toimintaperiaate.

Kun sivustolle ensimmäisen kerran saavutaan, pyydetään palvelimelta koko sivua. Palvelin palauttaa täydellisen HTML-dokumentin. Tämän jälkeen sivulta toiselle siirtyminen tapahtuu valikon linkkejä napsauttamalla, kuten tavallisesti. Linkit eivät kuitenkaan ohjaa selainta uudelle sivulle vaan käynnistävät JavaScript-kutsun, jonka perusteella ohjelma lähettää XMLHttpRequest-pyynnön palvelimelle. Palautettu, yksittäistä sivua vastaava, sisältö sijoitetaan dokumentin DOM-puurakenteeseen JavaScriptin avulla.

Päätös käyttää Ajax-tekniikkaa oli varsin selvä jo heti alusta. Sen mahdollistamien SPA-sovellusten suurin hyöty on sovelluksen käyttöliittymän sulavampi toiminta verrattuna perinteisiin verkkosivuihin. Sivusto ei missään vaiheessa päivitä itseään, jolloin käyttäjälle muodostuu parempi käyttökokemus.

En kuitenkaan ottanut käyttöön mitään edellä mainituista JavaScript-ohjelmistokehyksistä. Tutkin kyllä tätäkin vaihtoehtoa. Opettelu olisi luultavasti vaatinut liian paljon aikaa ja vaivannäköä tässä yhteydessä. Sitä paitsi pääsin haluamaani lopputulokseen käyttämällä PHP-ohjelmistokehystä ja HTML-muotoa tiedon paluutyyppinä. Nämä palautettavat valmiit koodinpalaset sijoitellaan dokumenttiin kukin omalle paikalleen.

3.6 Slim Framework

Tarvitsin sovellukselle osoiterakenteen, jonka halusin myös olevan selkeä ja havainnollinen. Olin aiemmin tehnyt itse joitakin yksinkertaisia URL-osoitteiden reititykseen tarkoitettuja toimintoja, mutta tämä olisi ollut monimutkaisempi toteutettava. Aloin etsimään valmiita ratkaisuja ongelmaan. Juuri tähän tarkoitukseen tehtyjä kirjastoja löysinkin, mutta päädyin kuitenkin käyttämään web-ohjelmistokehystä.

Web-ohjelmistokehykset on suunniteltu auttamaan kehittäjiä web-sovellusten rakentamisessa. Ohjelmistokehykset tarjoavat perustoiminnallisuudet, jotka ovat yhteisiä useimmille web-sovelluksille. Ohjelmistokehyksiä käyttämällä voidaan säästää huomattavasti aikaa sivustojen rakentamisessa. Kehittäjien ei tällöin

tarvitse toteuttaa samoja ominaisuuksia uudestaan kaikkiin luomiinsa sovelluksiin. (Web application framework 2013.)

Uusien ohjelmistokehysten opettelu vaatii kuitenkin runsaasti aikaa. Itselläni ei ennen tätä projektia ollut kovinkaan paljon kokemusta aiheesta. Olisi ollut ajankäytön suhteen arvaamatonta lähteä opettelemaan täysverisen ohjelmistokehyksen käyttöä. En edes tulisi tarvitsemaan läheskään kaikkia niiden sisältämistä toiminnallisuuksista; aluksi olin kiinnostunut lähinnä osoitteiden reitityksestä. Lisäksi ne heikentävät ohjelman suorituskykyä. Tästä syystä päädyin käyttämään mikrokokoista ohjelmistokehystä.

Slim Framework on mikro-ohjelmistokehys (micro framework) PHP-ohjelmistokehitykseen. Se tarjoaa muun muassa kehittyneen URL-reitityksen (Kuva 4), tuen omille sivupohjille, salatut evästeet sekä mahdollisuuden asettaa reittikohtaisia funktioita. (Vaswani 2012.) Slim Framework noudattaa Front Controller -mallia ohjatessaan kaikki HTTP-pyynnöt yhteen tiedostoon, usein index.php-nimiseen. Tässä tiedostossa haetaan käyttöön kehyksen kirjastotiedosto, määritetään reitit ja käynnistetään sovellus. (Lockhart 2011.)

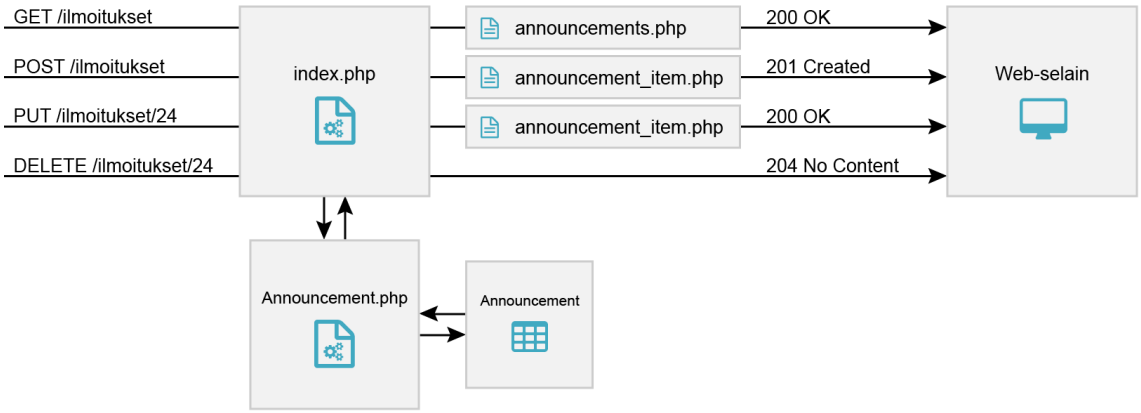
```
1 $app = new \Slim\Slim();
2 $app->get('/hello/:name', function ($name) {
3     echo "Hello, $name";
4 });
5 $app->run();
```

Kuva 4. Slim Framework -ohjelmistokehyksellä tehty yksinkertainen reititys.

Slim Framework käyttää reitityksessä REST-arkkitehtuurityyliä (Representational State Transfer). Tällöin resurssia käsitellään käyttäen apuna sen URL-osoitetta sekä HTTP-protokollan GET-, POST-, PUT- ja DELETE-metodeja. Metodit vastaavat resurssin palauttamisesta, luomisesta, muokkaamisesta ja poistamisesta. (Vaswani 2012.)

Kuviossa 2 on selitetty tarkemmin Slim Frameworkin ja REST-pohjaisen reitityksen periaate. Kaikki HTTP-pyynnöt on asetettu ohjautumaan index.php-

tiedostoon. Tämä tiedosto valitsee suoritettavat toiminnot perustuen pyynnön osoiterakenteeseen ja HTTP-metodiin.



Kuvio 2. Reititys esimerkkisisällölle.

Kuviossa on käytetty esimerkkinä ilmoitusten käsittelyä järjestelmässä. Sovelluksen index.php-tiedosto kutsuu Announcement-luokkaa, joka käsittelee pyynnön ja suorittaa tarvittavat muutokset tai lisäykset sitä vastaavassa tietokantataulussa. Luokka palauttaa tarvittavat tiedot kutsujalle. Saaduista tiedoista tuotetaan selaimelle esitys käyttäen HTTP-metodin mukaan valittua sivupohjaa.

Esimerkin GET-metodi palauttaa esityksen kaikista järjestelmän ilmoituksista, POST- ja PUT-metodit vain juuri lisätystä tai muokatusta ilmoituksesta. Esitys on HTML-koodilohko, joka asetetaan JavaScriptin toimesta oikealle paikalleen dokumentissa. DELETE-metodia käytettäessä ei tuoteta minkäänlaista sisältöä, koska esitettävää kohdetta ei ole. Kaikki pyynnöt palauttavat myös onnistumisesta kertovan HTTP-vastauskoodin.

Päädyin käyttämään juuri Slim Framework -ohjelmistokehystä, koska se tuntui helposti sisäistettävältä. Dokumentaatio oli selkeä, joskin kehityksen aikana kävi selväksi, että osin myös puutteellinen. Slim Framework on yksinkertaisuutensa takia myös järjestelmän mahdollisten jatkokehittäjien helpompi omaksua.

4 TUTKIMUKSEN OSAONGELMAT

Pitkän kehitysprosessin aikana tulee vastaan monenlaisia ongelmia ja haasteita. Näistä voisi varmasti kertoa enemmänkin kuin yhteen opinnäytetyöhön mahtuu. Tässä luvussa käsittelen tarkemmin muutamia tutkimuksen osaongelmia, jotka olivat erityisen haasteellisia tai muuten aikaa vieviä sekä sisällöltään tälle projektille ja osin myös infonäyttöjärjestelmille ominaisia. Esittelen ongelman ja kerron minkälaiseen ratkaisuun päädyin.

4.1 SPA-sovellusten selainhistoria

Yksi SPA-sovellusten haasteista on historianhallinnan toteuttaminen. Koska sovellus lataa pyydetyn sisällön asynkronisesti, sivu ei missään vaiheessa päivity. Tällöin ei myöskään selainhistoriaan kerry merkintöjä eikä osoiterakenne muutu. Tämän takia selaimen takaisin- ja eteenpäin-painikkeet eivät toimi halutulla tavalla. (Flanagan 2011, 671.)

Historianhallinnan toteuttamiseen on olemassa kaksi HTML5-määrittelyn mukanaan tuomaa menetelmää. Yksinkertaisempi historianhallintatekniikka käyttää location.hash-ominaisuutta ja hashchange-tapahtumaa. Useimmissa selaimissa location.hash-ominaisuuden asettaminen päivittää sijaintipalkin URL-osoitetta ja tallentaa historiatiedon selaimen sivuhistoriaan käyttäen hyväksi osoitteen ristikkomerkillä erotettua fragmenttiosaa. (Flanagan 2011, 671–672.)

Tallennetun tilan palauttamiseksi HTML5-tekniikkaa tukevat selaimet käynnistävät hashchange-tapahtuman aina kun fragmenttiosa muuttuu. Selaimissa, jotka tukevat hashchange-tapahtumaa, voidaan asettaa window.onhashchange-tapahtuma käsittelyfunktioon, jota kutsutaan muutoksen tapahtuessa. Funktio palauttaa halutun sisällön fragmenttiosan sisältämän tiedon perusteella. (Flanagan 2011, 671–672.)

HTML5-määrittely tarjoaa myös toisen tavan historianhallintaan. Tähän käytetään history.pushState()-metodia ja popstate-tapahtumaa. Kun sovellus siirtyy

uuteen tilaan, se kutsuu history.pushState()-metodia lisätäkseen tilan selainhistoriaan. Metodi saa ensimmäisenä argumenttinaan palautukseen tarvittavat tilatiedot sisältävän olion, toisena argumenttina sivun otsikon ja kolmantena dokumentin nykyistä sijaintia kuvaavan URL-osoitteen. Kun käyttäjä navigoi selainhistoriassa, selain käynnistää popstate-tapahtuman window-oliolle. (Flanagan 2011, 672–673.)

Käytin aluksi URL-osoitteen fragmenttiosan mahdollistamaa hashchange-ratkaisua, mutta halusin tätä vaihtoehtoa selkeämmän osoiterakenteen. Lopullisena ratkaisuna ongelmaan oli edellä kuvatuista jälkimmäinen menetelmä, jota kutsutaan myös nimellä HTML5 History API (Pilgrim 2013).

Käytän menetelmää sovelluksessani kuvan 5 osoittamalla tavalla. Tilaobjektille ei ole tarvetta, eivätkä nykyselaimet tue otsikkoargumenttia, siksi näille ei ole asetettu todellista arvoa. Viimeisenä argumenttina on päävalikosta napsautetun linkin osoite, joka tallennetaan selainhistoriaan.

```
1 history.pushState(null, null, $(this).attr('href'));
```

Kuva 5. pushState-metodin käyttö historiatiedon tallentamiseen.

Tilan palautusta varten tulee popstate-tapahtumalle määrittää funktio, joka reagoi selainhistorian muutoksiin. Käyttämäni funktio lataa pyydetyn sisällön jQueryn Ajax-pyynnöllä, muuttuneeseen URL-osoitteeseen perustuen (Kuva 6).

```
1 window.onpopstate = function() {
2     $.ajax({
3         type: 'GET',
4         url: document.location,
5         success: function(data) {
6             $('#main').html(data);
7         }
8     });
9 };
```

Kuva 6. Tilan palautus selainhistoriasta.

4.2 Sisältöeditorin räätälöinti

Järjestelmän hallintapaneeliin tarvittiin työkalu infonäytöillä esitettävien vapaamuotoisten ilmoitusten tuottamiseen. Sisältö tulisi pystyä kirjoittamaan valmiin sivupohjan päälle. Etsin valmista, JavaScript-pohjaista ja PowerPoint-ohjelmaa toiminnoiltaan muistuttavaa lisäosaa järjestelmään integroitavaksi. Tällaista en kuitenkaan onnistunut löytämään, joten piti keksiä jokin toinen ratkaisu.

Päätin lopulta yrittää JavaScript-pohjaisen WYSIWYG-tekstieditorin muokkaamista tähän käyttötarkoitukseen soveltuvaksi. WYSIWYG-editorilla tarkoitetaan ohjelmaa, jonka avulla voidaan tarkastella sisällön lopullista ulkoasua jo muokkauksen aikana (Rouse 2011).

Ratkaisu ei ole mitenkään erityisen omaperäinen, onhan tekstieditorin varaan rakennettu monestikin käyttäjien tiedonsyöttöön tarkoitetut ratkaisut. Haasteena kuitenkin oli saada editori käyttäytymään esitysgrafiikkaohjelmien tavoin: editorin muokkausalue pitäisi voida rajoittaa tiettyyn kokoon ja kuvasuhteeseen, ja lisäksi sen tulisi mahdollistaa ennalta luotujen sivupohjien käyttö.

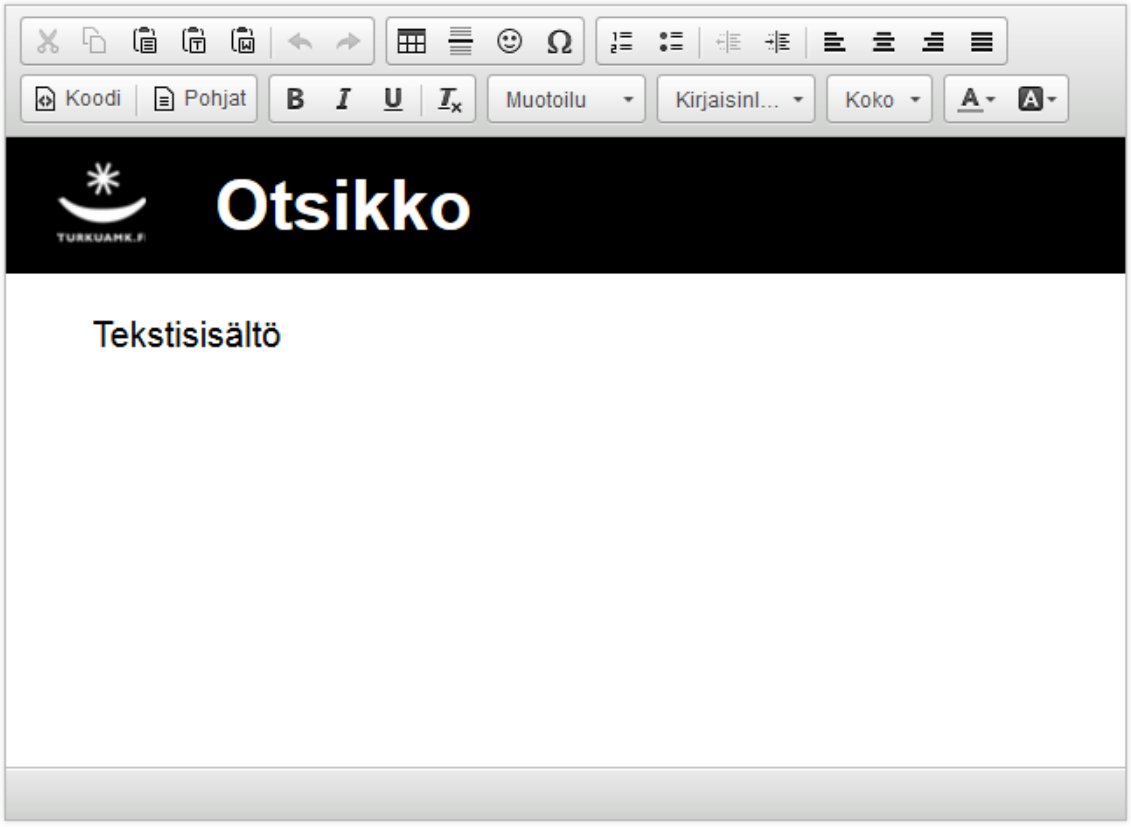
Vertailin kahtena vaihtoehtona CKEditoria ja TinyMCE:tä. Huolellisen harkinnan ja testaamisen jälkeen päädyin käyttämään ensimmäistä. Tämä tuntui olevan paremmin muokattavissa haluttuun käyttötarkoitukseen, lisäksi sen ulkoasu oli miellyttävämpi.

CKEditor upotetaan web-sivulle omana tekstinmuokkausalueenaan, joka sisältää yleiset tekstinkäsittelyominaisuudet. Se mahdollistaa myös omien sivupohjien käytön. Sivupohjat ovat tavallisia HTML-muotoisia rakenteita, joita käytetään editorissa muokattavan sisällön pohjana. CKEditor on hyvin kustomoitavissa, ja siihen saa myös runsaasti lisäosia. (CKSource 2013a; 2013b.)

Tämäkään ratkaisu ei lopulta osoittautunut aivan yksinkertaiseksi. Keino editorin muokkausalueen koon lukitsemiseen oli yllättävän hankala oivaltaa, vaikka lopulta ratkaisu olikin varsin yksinkertainen. Oletuksena editorin muokkausalue laajenee sisällön pituuden mukaan. Tämän rajoittamiseen tarvittiin CSS-tyyliohjeiden position- ja overflow-ominaisuuksia.

Muitakin ongelmia ilmeni. Yksi oli Internet Explorer -selaimen poikkeava tapa esittää kirjoitusalustan ympärillä venytystyökalut, jos sivupohjan sisällä oleville lohkoelementeille oli asetettu kiinteät mitat. Ongelma esti sisällön suoran muokkauksen ja vaati käyttäjältä kaksoisnapsautuksen sisältöalueelle. Tämä paljastui myöhemmin selaimen ominaisuudeksi. Ratkaisuksi löytyi tyylipohjien elementtien kiinteiden mittojen poistaminen ja korvaaminen täytteillä, vastaavan ulkoasun tuottamiseksi.

Myös editorin oletusasetusten muokkaaminen vaati paljon aikaa ja vaivaa. Työkalupalkeista piti karsia kaikki tähän käyttötarkoitukseen sopimattomat toiminnot pois. Lopulta käyttöliittymästä saatiin kuitenkin varsin selkeä (Kuva 7).



Kuva 7. CKEditor kustomoituna käyttötarkoitukseen soveltuvaksi.

Yksi erityinen ongelma oli saada editorilla luotu sisältösivu skaalautumaan koko näytön suuruiseksi infonäyttöesitystä varten. Editori itsessään ei voi olla resoluutioltaan valmiiksi FullHD-kokoinen, koska tällöin muokkausalue ei mahtuisi kokonaan useimmille näyttötarkkuuksille. Tämä täytyi siis vielä ratkaista.

4.3 Esityksen ja sisältöjen skaalaus

Edellisessä alaluvussa esitettyyn ongelmaan oli kehitettävä ratkaisu, joka skaalaisi editorilla luodut sisältösivut koko ruudun kokoiseksi. Ominaisuudelle olisi hyötyä myös muiden sisältöjen skaalaamisessa. Näin esitystä voitaisiin ajaa ongelmitta erikokoisilla näytöillä mahdollistaen samalla esityksen esikatselun tietokoneilla.

Päädyin ratkaisuun, jossa asetin ensin tälle sisältöeditorissa luodulle sivulle kiinteät ulkomitat: leveydeksi 640 pikseliä ja korkeudeksi 360 pikseliä. Tällöin se vastaisi sisältöeditorin muokkausalustan kokoa. Käytettävä 16:9 kuvasuhde säilyisi skaalauksessa vakiona. Lopulta päätin asettaa muutkin luomani sisältösivut tähän kokoon, jotta fonttikoot, täytteet ja muut mitat olisivat yhtenäisiä.

Toteutus perustuu CSS-tyyliohjeiden ja jQuery-kirjaston yhteiskäyttöön. Kyseessä on pieni jQuery-lisäosa (Kuva 8), joka voidaan liittää haluttuun elementtiin, tässä tapauksessa yhteen sisältösivuun. Funktio laskee sisältöelementin ja selainikkunan mitat, muodostaa näistä kertoimen ja skaalaa sisältösivun selainikkunan levyiseksi käyttäen CSS:n transform-ominaisuutta. Lisäosaa kutsutaan uudelleen aina selainikkunan koon muuttuessa. Esitettävästä koodista on selkeyden vuoksi jätetty pois yhteensopivuutta parantavat CSS:n selainetuliitteet.

```
1  $.fn.scaleToFit = function() {
2      return this.each(function() {
3          var width = $(this).outerWidth();
4          var windowWidth = $(window).width();
5          var value = windowWidth/width;
6          $(this).css({'transform': 'scale(' + value + ')'});
7          $(this).css({'transform-origin': 'left top'});
8      });
9  };
```

Kuva 8. Skaalaus selainikkunan leveyden mukaan.

Skaalauksen tarve ei rajoitu pelkästään itse esitykseen. Infonäytöt ovat laitteita, joissa esitettävää sisältöä ei voi järkevästi vierittää vierityspalkin avulla syöttölaitteiden puuttuessa. Näin ollen, jos esityksen sivun sisältö ylittää näytön pys-

tysuuntaisen resoluution, ylimääräinen osa leikkautuu pois. Tätä ongelmaa varten jatkokehitin edellä esittelemääni ratkaisua.

Tällä kertaa algoritmi (Kuva 9) vertaa sisältölohkon korkeutta ympäröivän elementin korkeuteen ja laskee kertoimen, jonka mukaisesti sisältöä on pienennettävä. Skaalaus suoritetaan vain siinä tapauksessa, että ylivuotoa tapahtuu, eli lapsielementti on vanhempaansa korkeampi.

```
1  $.fn.scaleByHeight = function() {
2      return this.each(function() {
3          var height = $(this).outerHeight();
4          var parentHeight = $(this).parent().height();
5          if(height > parentHeight) {
6              var value = parentHeight/height;
7              $(this).css({'transform': 'scale(' + value + ')'});
8              $(this).css({'transform-origin': 'left top'});
9          }
10     });
11 };
```

Kuva 9. Skaalaus sisällön korkeuden mukaan.

Ratkaisussa on kuitenkin tietty ongelma. Koska menetelmä pienentää valittua sisältöä alkuperäisen kuvasuhteen säilyttäen, myös sisältöalueen leveys kapenee. Ei ole kuitenkaan järkevää asettaa skaalausta vain pystysuunnassa, koska tällöin teksti litistyisi lukukelvottomaksi. Vaakasuuntainen tila jää siis käyttämättä, koska rivinvaihdot on tehty alkuperäisen, skaalaamattoman, sisällön pohjalta (Kuva 10). Menetelmä soveltuukin parhaiten tilanteisiin, joissa sisällön pituus on suurin piirtein ennustettavissa ja ylivuotoa tapahtuu vain satunnaisesti.

Kuva 10. Uutissyötteen näkyvyys esityksessä neljällä ja kymmenellä otsikolla.

Tässä mielessä jokin menetelmä, joka olisi vastaavalla tavalla pienentänyt sisältöä, mutta muokaten elementin koon sijaan kirjasinkokoa, olisi saattanut olla parempi. Tällaistakin ratkaisua yritin saada toimimaan, mutta en onnistunut siinä siihen käyttämässäni ajassa.

4.4 Esitysalgoritmi

Yksi keskeisimpiä haasteita järjestelmän toteutuksessa oli infonäyttöesityksen sisältösivuja kierrättävän algoritmin suunnittelu ja toteutus. Sen toiminnan tulisi olla mahdollisimman vakaata ja häiriötöntä. Algoritmin pitäisi myös tunnistaa ei-aktiiviset ja tyhjät sisällöt ja osata jättää nämä pois esityskierrosta.

Alkuperäisessä ratkaisussani esitys haki uuden sivun aina ajastimen umpeuduttua. Tämä aiheutti ongelman, jossa sisältöjen latausaikojen vaihdellessa myös sisältöjen vaihtuminen oli epätasaista. Muutin algoritmia sellaiseksi, että se lataa tarvittavat sisällöt aina kunkin esityskierroksen aluksi taulukkoon, josta ne esitetään ajastimen osoittaman ajan mukaisesti. Näin saadaan tasaisemmat näyttöajat kaikille sisältösivuille.

Täytyi vielä ratkaista, kuinka käsitellä sisältöjä, joiden haluttiin näkyvän vain tiettyinä ajankohtina tai jotka olivat toisinaan sisällön puutteesta johtuen kokonaan tyhjiä. Päädyin käyttämään tähän HTTP-virhekoodeja. Järjestelmä palauttaa virhekoodin pyydettäessä sisältöä, joka ei ole tällä hetkellä näytettävissä jostain edellä mainitusta syystä. Ajax-koodi lataa sisällön taulukkoon vain pyynnön onnistuttua, joten nämä sisällöt eivät näin päädy esitykseen.

Kehittelemäni ratkaisu on esitetty hieman lyhennettynä kuvassa 11. Esityksen suoritus käynnistyy initPresentation-funktion kutsulla. Funktiota kutsutaan aina jokaisen esityskierroksen aluksi. Tässä yhteydessä tyhjennetään myös esitettäviä sisältöjä kuvaava pages-taulukko. getPages-funktio palauttaa Ajax-pyynnön, joka noutaa esityksessä esitettävät sisällöt. Pyyntö suoritetaan, ja kullekin tulokselle haetaan HTML-sisältö, joka asetetaan pages-taulukkoon indeksin mukaiseen alkioon. Näin järjestelmässä tallennettu esitysjärjestys säilyy samana infonäyttöesityksessä.

Tämän jälkeen kutsutaan run-funktiota sisällön vaihtumisnopeutta kuvaava aika parametrina. run-funktion sisällä oleva repeatedCall-funktio suoritetaan parametrina saadun sekuntimäärän kuluttua. repeatedCall hakee pages-taulukosta seuraavan määritellyn indeksin mukaisen sisällön. Tämä sisältö asetetaan näkyville esitykseen, ja se skaalataan sopivan kokoiseksi. repeatedCall-funktiota kutsutaan rekursiivisesti, kunnes koko taulukko on käyty läpi. Tämän jälkeen kutsutaan initPresentation-alustusfunktiota uudestaan.

```
initPresentation();

function initPresentation() {
    pages = [];
    getPages().done(function(result) {
        $.each(result[1], function(index) {
            getData(this.url).done(function(data) {
                pages[index] = data;
            });
        });
        run(result[0].time);
    }).fail(function() {
        initPresentation();
    });
}

function run(time) {
    var interval = parseInt(time) * 1000;
    var index = 0;
    function repeatedCall() {
        index = getNext(index);
        $('body').empty();
        $('body').append(pages[index]);
        initScale();
        index++;
        if(index >= pages.length) {
            initPresentation();
        } else {
            setTimeout(repeatedCall, interval);
        }
    }
    setTimeout(repeatedCall, interval);
}
```

Kuva 11. Infonäyttöesityksen esitysalgoritmi.

4.5 Ulkopuolisten syötteiden tuonti

Järjestelmään haluttiin tuoda sisältöjä myös ulkopuolisista lähteistä, jotta sisällöntuotanto ei olisi pelkästään oppilaitoksen oman viestinnän varassa. Mahdollisia järjestelmään tuotavia tietosisältöjä pohdittiin projektin ohjausryhmän kanssa käydyissä tapaamisissa.

Tietoa on tarjolla verkossa erilaisina syötteinä. Haluttujen syötteiden parsiminen järjestelmän ymmärtämään muotoon on oma haasteensa. Eri syötetyypit ja joskus myös yksittäiset syötteet erikseen vaativat omanlaisensa käsittelyn.

Olen käyttänyt RSS-syötteiden hakemiseen PHP-pohjaista SimplePie-kirjastoa. Tämä kykenee automaattisesti parsimaan halutuista RSS-syötteistä tiedot, jotka on vaivatonta sijoittaa ohjelmakoodiin kirjaston tarjoaman rajapinnan avulla.

Toimipisteen ruokalan lounaslista oli yksi sisällöistä, joita infonäyttöesityksen toivottiin näyttävän. Tämä oli saatavana yrityksen sivuilta JSON-muotoisena syötteenä. JSON-syötteet ovat yleensä sisällöltään niin erilaisia, ettei niiden jäsentämiseen ole mielekästä käyttää valmiita kirjastoja. Tietojen parsiminen täytyi tehdä siis itse. Tämä kuitenkin onnistui tässä tapauksessa melko vaivattomasti, koska syötteen rakenne oli helposti ymmärrettävässä muodossa.

Myös säätiedot olivat esillä tuotavia sisältöjä ideoitaessa. Ilmatieteen laitos tarjoaa säätietojaan avoimena datana kaikkien saataville. Käyttö vaatii kuitenkin API-avaimen. Tiedon muoto on varsin vaikeaselkoista XML-dataa, ja halutun tiedon parsiminen syötteestä onkin melko työlästä. Syötteen tarjoaman sääsymbolitiedon käyttö vaatisi myös jatkokäsittelyä, koska se ei sellaisenaan tue eri vuorokaudenaikoja. Tämä sisältötyyppi jäi nyt tällä erää toteuttamatta.

Turun kaupungin linja-autojen pysäkkiaikataulut olivat saatavilla Brahe-matkainfosta, eivät tosin syötteenä vaan valmiina sivuna. Tämän voi upottaa esitykseen HTML:n iframe-elementtinä tai käyttäen tätä varten luomaani pohjaa, joka hakee saman sivun sisällön, mutta muokkaa sen ulkoasun esityksen tyyliin paremmin soveltuvaksi. Näin myös ikävät pienet viiveet esityksessä vältetään, kun selain ei muodosta iframe-elementtiä jokaisella näyttökerralla.

5 JÄRJESTELMÄN TOIMINTA

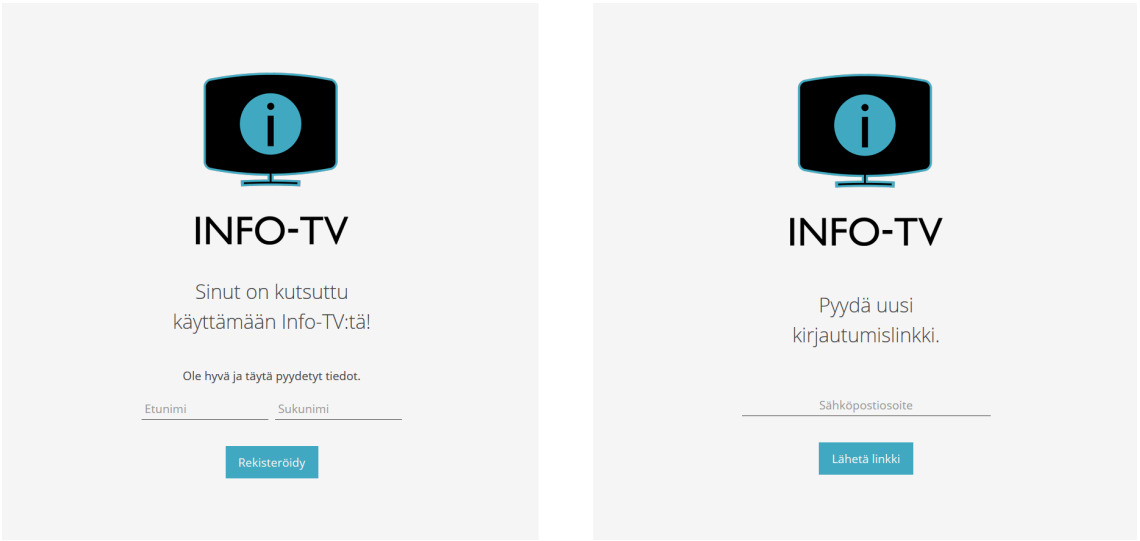
Järjestelmän rakenne koostuu kolmesta osasta: sisääntulosivuista, hallintapaneelista ja itse infonäyttöesityksestä. Samaa jakoa olen käyttänyt osoiterakenteen ja ohjelmakoodien, kuten JavaScriptin ja CSS-tyylien jaottelussa.

Järjestelmän hallintapaneeli on oma SPA-sovelluksensa. Myös infonäyttöesitys toimii ilman varsinaisia sivupäivityksiä, joskaan siinä ei voida navigoida käyttäjän toimesta; sisältömuutokset tapahtuvat ajastimen mukaisesti. Sisääntulosivut käyttävät nekin Ajax-tekniikkaa, tosin ainoastaan esittäessään palautustiedon toimenpiteen onnistumisesta.

Toteutetut toiminnot on määritelty yhteistyössä asiakkaan kanssa. Toiveita kartoitettiin ohjausryhmän kanssa käydyissä palavereissa ja sovelluksesta julkaistujen esittelyversioiden pohjalta.

5.1 Sisääntulosivut

Sisääntulosivut vastaavat käyttäjän rekisteröitymisestä ja kirjautumisesta. Kuvassa 12 on esitetty kuvankaappaukset näistä järjestelmän sivuista. Lomakkeet on pyritty toteuttamaan mahdollisimman yksinkertaisina.



Kuva 12. Järjestelmän rekisteröitymis- ja kirjautumissivut.

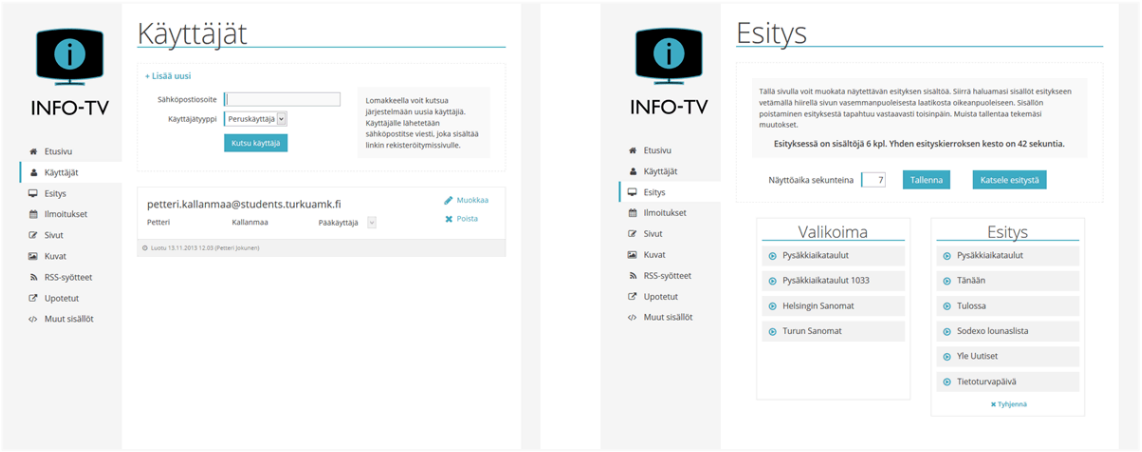
Järjestelmään lisätään uusia käyttäjiä kutsumalla heidät hallintapaneelin kautta. Kutsu lähetetään sähköpostina halutulle henkilölle. Sähköpostien hallintaan olen käyttänyt SwiftMailer-kirjastoa, joka tarjoaa monipuolisen oliopohjaisen ratkaisun sähköpostien lähettämiseen.

Vastaanotetussa sähköpostiviestissä olevan linkin kautta pystyy tekemään lopullisen rekisteröitymisen. Tämän jälkeen käyttäjälle lähetetään sähköpostiin kirjautumislinkki järjestelmään. Jatkossa käyttäjä pyytää järjestelmältä uuden kirjautumislinkin aina, kun hänen kirjautumisensa on päättynyt.

5.2 Hallintapaneeli

Järjestelmän hallintapaneelin toteutukseen ja toimintojen viilaamiseen kului suuri osa työajasta. Se on järjestelmän käytön kannalta keskeinen osa, joka sisältää paljon käyttäjävuorovaikutusta. Tämän takia siinä on myös paljon yksityiskohtia, jotka täytyi ottaa huomioon.

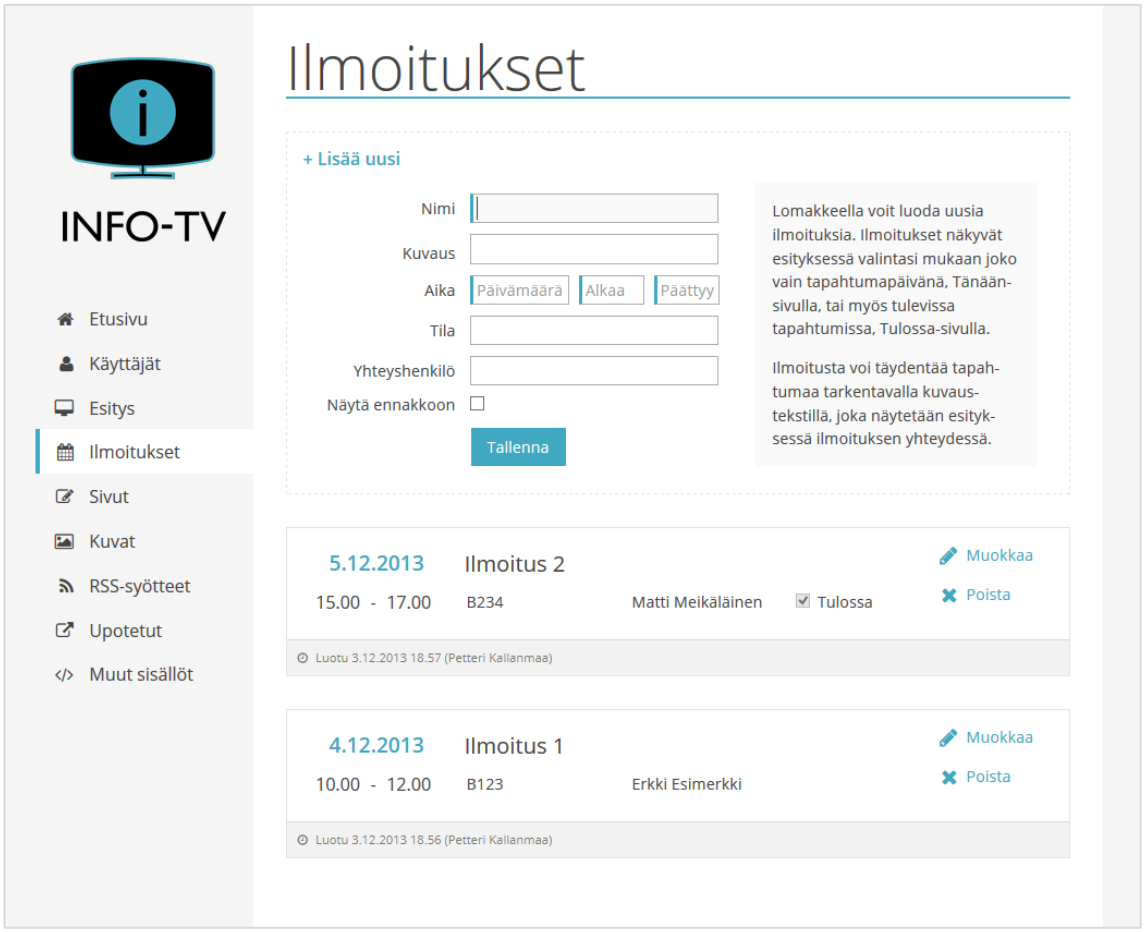
Järjestelmän hallintapaneelin kautta pystyy lisäämään uusia käyttäjiä, poistamaan ja muokkaamaan vanhoja sekä muuttamaan käyttäjän valtuuksia (Kuva 13). Järjestelmässä on kahden tasoisia käyttöoikeuksia. Ainoastaan pääkäyttäjällä on lupa tehdä edellä mainittuja käyttäjiin kohdistuvia toimia.



Kuva 13. Hallintapaneelin Käyttäjät- ja Esitys-sivu.

Hallintapaneelin kautta voidaan järjestellä infonäyttöesitystä: mitä sisältöjä esityksessä näytetään, missä järjestyksessä ja kuinka pitkä on sisältöjen näyttöaika. Sisältösivuja siirretään esitykseen vetämällä sisältö hiirellä valikoimasta esitys-laatikkoon (Kuva 13). Sivulta löytyvän linkin kautta voidaan katsella esitystä.

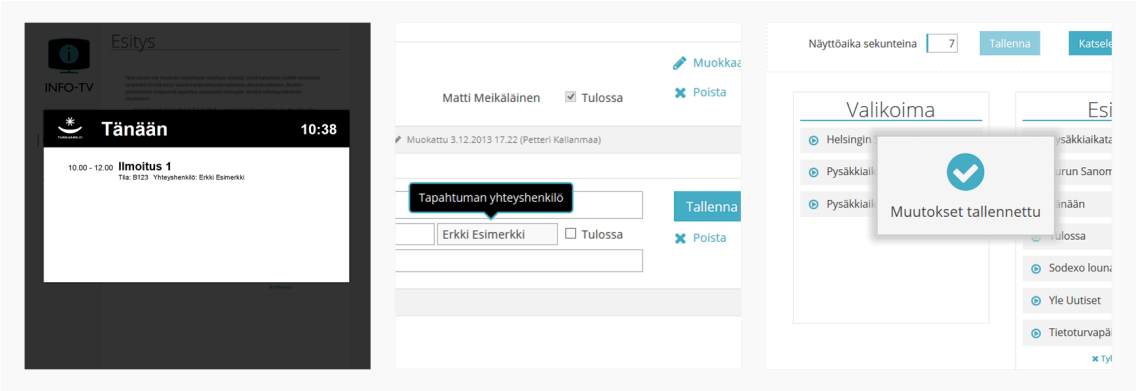
Järjestelmä tukee kuutta esityksessä näytettävää sisältötyyppiä. Näitä ovat ilmoitukset (Kuva 14), joita esitetään tapahtumapäivänä Tänään-sivulla tai mahdollisesti poimintoina Tulossa-sivulla. Sivut ovat editorin avulla valmiin mallipohjan päälle käyttäjän itse luomia yksittäisiä sisältösivuja. Kuvia voidaan ladata palvelimelle hallintapaneelin kautta esitystä varten. Sekä ilmoituksille, sivuille että kuville pystyy asettamaan voimassaoloajan, jolloin sisältö näkyy esityksessä. Infonäyttöesitykseen voidaan tuoda myös RSS-syötteitä sekä yksittäisiä HTML:n iframe-elementtiä käyttäviä upotettavia web-sivuja.



Kuva 14. Hallintapaneelin sivu, jolla käsitellään järjestelmään syötettyjä ilmoituksia.

Edellä mainittujen lisäksi järjestelmässä on muut sisällöt -sisältötyyppi. Tätä käytetään lisäosina luotujen sisältöjen rekisteröimiseksi järjestelmään. Lisäosat ovat sisältöjä, jotka eivät ole niin yleistettävissä, jotta niitä varten olisi ollut luontevaa luoda omaa sisältötyyppiänsä. Hallintapaneelin kautta näille voidaan antaa niiden toimintaa muokkaavia parametreja. Tällä hetkellä tätä sisältötyyppiä käyttäen on toteutettu toimipisteen ruokalan lounaslista sekä paikallisliikenteen linja-autojen pysäkkiaikataulut.

Olen pyrkinyt käyttöliittymän toteutuksessa helppokäyttöisyyteen ja selkeyteen. Kuvassa 15 on esitetty joitakin hallintapaneelin käyttökokemusta parantavia yksityiskohtia. Infonäyttöesityksen sisältösivuja voidaan esikatsella järjestelmän kautta. Nämä esikatseluikkunat aukeavat suoraan auki olevan sivun päälle. Työkaluvihjeitä käytetään lomakekenttien täyttöä helpottamaan. Tehdyt muutokset ja lisäykset järjestelmä vahvistaa pienellä ilmoitusikkunalla.

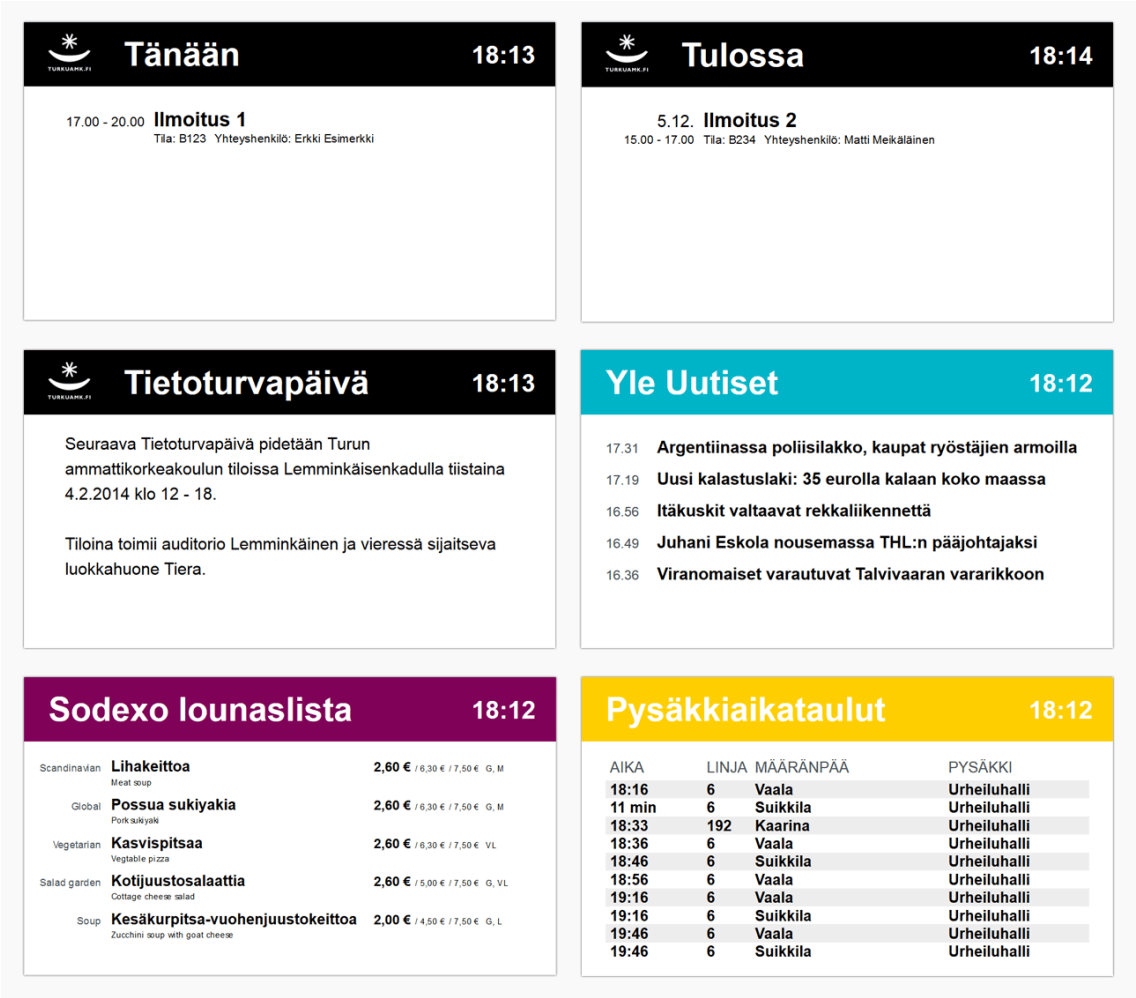


Kuva 15. Hallintapaneelin käyttöliittymän yksityiskohtia.

Yhtenä yksityiskohtana mainittakoon sisältöjen muokkaus hallintapaneelissa. Se onnistuu paikallaan, ilman uusia ikkunoita. Tämä on toteutettu siten, että kaikki sivulla listatut sisällöt ovat jo sivua ladattaessa valmiiksi lomakkeita. Nämä on kuitenkin muotoiltu siten, ettei niitä pysty suoraan muokkaamaan, eivätkä ne muistuta ulkoasultaan lomakekenttiä. Kun sisältö valitaan muokattavaksi, lomakekentät avautuvat käsiteltäviksi ja tietojen vaihto onnistuu elementin omalla paikalla. Muutosten tallentamisen jälkeen tämä yksittäinen sisältö ladataan uudestaan dokumenttiin, jolloin se vastaa ulkoasultaan alkuperäistä.

5.3 Infonäyttöesitys

Infonäyttöesitys kierrättää näyttölaitteella hallintapaneelin kautta esitykseen asetettuja sisältösivuja. Yhden sivun näkyvyysaikaa pystyy muokkaamaan hallintapaneelista käsin. Kuvassa 16 on joitain esimerkkejä esityksessä näytettävistä sisältösivuista.



Kuva 16. Esimerkkejä esityksessä näytettävistä sisältösivuista.

Infonäyttöesityksen ulkoasua suunniteltaessa sisällön selkeys ja erottuvuus olivat oleellisia tekijöitä. Ne asetettiin myös työn keskeisiksi tavoitteiksi. Näihin tavoitteisiin päästään monia pieniä yksityiskohtia käyttäen: iso kirjasinkoko, otsikon taustavärien käyttö sekä sisällön kirjasinten koon, paksuuden ja värien pieni vaihtelu.

6 JATKOKEHITYS

Vaikka projektiin kuluikin suuri määrän työtunteja, ei kaikkea silti voi saada täysin valmiiksi ja viimeistellyksi, ainakaan yhdessä opinnäytetyössä. Tämän takia erityisesti projektin loppuvaiheessa piti tehdä rajausta toteutettavien ominaisuuksien suhteen.

Projektia suunniteltaessa pidettiin esillä vaihtoehtoa järjestelmän käyttämisestä myöhemmin muissakin ympäristöissä. Pyrin ottamaan tämän huomioon, enkä lisännyt kiinteästi mitään tälle projektille ominaisia sisältöjä, kuten lounasravintolan ruokalistaa. Tämä on tuotu järjestelmään yhteen tiedostoon perustuvana lisäosana, jolloin sen poistaminen on helppoa.

Mikäli järjestelmää tullaan tulevaisuudessa käyttämään julkisessa verkossa, tietoturvaa tulisi pohtia nykyistä perusteellisemmin. Autentikointimenetelmää voisi vahvistaa ja harkita SSL-yhteyden käyttämistä. Tämän toteutuksen osalta tilannetta helpottaa se, että kyseessä on vain osassa koulun sisäverkkoa toimiva järjestelmä. Kehityksessä on kuitenkin otettu perinteiset haavoittuvuudet, kuten SQL-injektiot ja XSS-hyökkäykset huomioon.

Vanhempien selainten tukemista voidaan käyttöympäristöstä riippuen joutua harkitsemaan. Internet Explorer 7 -tuesta olisi ollut hyötyä tässäkin ympäristössä, mutta jotkin käyttämäni lisäosat vaativat vähintään Internet Explorer 8 -version. Näin ollen tämä ei ollut järkevästi toteutettavissa, ja siksi järjestelmä tukee nykyisellään tästä selaimesta vain versiota 10 tai sitä uudempia.

Kaikenlaisten poikkeustilanteiden käsittely ja hallinta kaipaisi viimeistelyä. Järjestelmän tuottamat virheilmoitukset eivät ole useimmissa tapauksissa kovinkaan informatiivisia. Myös mahdollisten välimuistiratkaisujen kehittäminen saattaisi olla hyödyllistä, jos jatkuvat palvelinpyynnöt koetaan ongelmallisiksi.

Lisätyötä riittäisi myös tällä erää toteutumattomien sisältötyyppien, kuten sään ja videoiden toteuttamisessa. Järjestelmän pidempiaikaisen käytön myötä saatetaan hyvin havaita jotain muutakin lisättävää tai muutettavaa.

7 POHDINTA

Opinnäytetyöni tavoitteena oli selvittää, miten tuottaa viestinnän tehokkuutta parantava ja muokkaustoiminnoiltaan helppokäyttöinen infonäyttöjärjestelmä. Aiheen laajuus aiheutti haasteita rajaukselle, niin itse toteutusprojektin kuin opinnäytetyöraportin suhteen. Tavoitteisiin nähden työ kuitenkin onnistui mielestäni hyvin.

Tätä kirjoittaessani järjestelmää ei ole vielä otettu käyttöön, mutta se tullaan tekemään mahdollisimman pian. Olen saanut positiivista palautetta toteutuksesta ja erityisesti ulkoasun ja käyttöliittymän selkeydestä. Näiden viimeistelemiseen olenkin käyttänyt paljon aikaa, ehkä osin muiden toimintojen kustannuksella. Järjestelmän kehitystä saatetaan kuitenkin jatkaa tulevaisuudessa muissa opiskelijaprojekteissa.

Projektin haastavimpia osia oli erilaisten lisäosien muokkaaminen järjestelmän tarpeisiin soveltuviksi. Myös Ajax-tekniikan laaja käyttö tarjosi oman haasteensa. Projektin aikana tulikin opeteltavaksi monia uusia asioita, joista jouduin tekemään perusteellista selvitystyötä.

Suurimpana yllätyksenä pidän käytettyä työmäärää. Projektiin olisi saanut kulutettua kaiken ajan, ja sitä olisi voinut jatkaa vielä pitkään tämän jälkeenkin. Itselleni jäi pitkä lista asioita, joita olisi voinut vielä lisätä tai viimeistellä. Täysin valmista järjestelmää ei näillä resursseilla pysty tekemään, eikä se varmaan ole opinnäytetöiden tarkoituskaan.

Projekti oli kuitenkin antoisa ja opettavainen. Web-ohjelmointi on minua kiinnostava aihe, ja nyt siihen avautui vielä hieman erilaisempi näkökulma infonäyttöjen muodossa. Näyttöjä tuleekin luultavasti tarkasteltua aivan toisella silmällä, kun niitä jatkossa, niiden monissa käyttöympäristöissään, huomaan. Kaiken kaikkiaan projektista kertyneellä kokemuksella on varmasti paljon hyötyä tulevaisuudessa.

LÄHTEET

CKSource 2013a. About. Viitattu 10.12.2013 http://ckeditor.com/about.

CKSource 2013b. Features. Viitattu 10.12.2013 http://ckeditor.com/about/features#user-rich-content.

DB-Engines 2013. DB-Engines Ranking. Viitattu 1.12.2013 http://db-engines.com/en/ranking.

First Technology Oy 2011. Rakkaalla lapsella on monta nimeä. Viitattu 22.11.2013 http://www.firstview.fi/rakkaalla-lapsella-on.

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

Heiskanen, H. 2013. Yhden sivun web-sovellukset tulevat, oletko valmis? Viitattu 3.12.2013 http://gofore.com/asiantuntijoilta/yhden-sivun-web-sovellukset-tulevat-oletko-valmis/.

InfoSign Oy 2008. Info-TV hankkijan ja suunnittelijan opas 2008. Viitattu 22.11.2013 http://www.infokanava.info/.

Kelsen, K. 2010. Unleashing the Power of Digital Signage. Burlington, MA: Focal Press.

Lockhart, J. 2011. Say Hello World with Slim. Viitattu 10.12.2013 http://www.slimframework.com/news/hello-world.

Mozilla Developer Network 2013a. JavaScript Overview. Viitattu 23.11.2013 https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/JavaScript_Overview.

Mozilla Developer Network 2013b. Getting Started. Viitattu 23.11.2013 https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started.

Murray, G. 2005. Asynchronous JavaScript Technology and XML (Ajax) With the Java Platform. Viitattu 27.11.2013 http://www.oracle.com/technetwork/articles/javaee/ajax-135201.html.

Nordlund, M. 2008. Infonäyttöjärjestelmä osana yritys- ja organisaatioviestintää. Sähköala 5/2008, 26–28. Viitattu 22.11.2013 http://www.infosign.fi/images/xsahkoala.pdf.

Oracle 2013. What is MySQL? Viitattu 30.11.2013 http://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html.

Osmani, A. 2011. Building Single Page Applications With jQuery's Best Friends. Viitattu 3.12.2013 http://addyosmani.com/blog/building-spas-jquerys-best-friends/.

Pilgrim, M. 2013. Manipulating History for Fun & Profit. Viitattu 3.12.2013 http://diveintohtml5.info/history.html.

Rouse, M. 2011. WYSIWYG (what you see is what you get). Viitattu 10.12.2013 http://whatis.techtarget.com/definition/WYSIWYG-what-you-see-is-what-you-get.

Single-page application 2013. Wikipedia. Viitattu 3.12.2013 http://en.wikipedia.org/wiki/Single-page_application.

Tatroe, K.; MacIntyre, P. & Lerdof, R. 2013. Programming PHP. Third Edition. Sebastopol, CA: O'Reilly Media.

The PHP Group 2013. What is PHP? Viitattu 18.12.2013 http://www.php.net/manual/en/intro-whatis.php.

Vaswani, V. 2012. Create REST applications with the Slim micro-framework. Viitattu 3.12.2013 http://www.ibm.com/developerworks/opensource/library/x-slim-rest/.

W3Techs 2013a. Usage of server-side programming languages for websites. Viitattu 26.11.2013 http://w3techs.com/technologies/overview/programming_language/all.

W3Techs 2013b. Usage statistics and market share of JQuery for websites. Viitattu 23.11.2013 http://w3techs.com/technologies/details/js-jquery/all/all.

Web application framework 2013. DocForge. Viitattu 3.12.2013 http://docforge.com/wiki/Web_application_framework.

Wurzer, E. 2012. Why you Should be using PHP's PDO for Database Access. Viitattu 29.11.2013 http://net.tutsplus.com/tutorials/php/why-you-should-be-using-phps-pdo-for-database-access/.

Yackley, B. 2011. A Beginner's Guide to Digital Signage. Viitattu 23.11.2013 http://global.networldalliance.com/downloads/white_papers/Black-Box_WP_Beginners-Guide-to-DS_To-Launch.pdf.

Käytetyt ohjelmakirjastot ja muut lisäosat

PHP

Slim Framework
PHP-ohjelmistokehys (micro framework).

SimplePie
PHP-kirjasto RSS-syötteiden jäsentämiseen.

SwiftMailer
PHP-kirjasto sähköpostien lähetykseen.

JavaScript

jQuery
JavaScript-kirjasto, joka helpottaa muun muassa DOM-rakenteen muokkausta ja Ajax-kutsuja.

CKEditor
WYSIWYG-tekstieditori.

pickadate.js
jQuery-pohjainen ajanvalitsin.

jQuery Form Plugin
Lomakkeiden käsittelyä ja lähettämistä helpottava jQuery-kirjasto, jota käytetään sovelluksessa kuvien lataamiseen palvelimelle.

jQuery UI Sortable
jQuery-pohjainen ja jQuery UI -kirjastoa hyödyntävä työkalu listaelementtien uudelleenjärjestämiseen.

Tooltipster
jQuery-pohjainen lisäosa omien, ulkoasultaan kustomoitujen, työkaluvihjeiden luomiseen.

colpick
jQuery-pohjainen värinvalitsin.

Autosize
jQuery-pohjainen työkalu, joka suurentaa textarea-elementit sisällön koon mukaan.

Fontit

Open Sans
Vapaasti käytettävä sans-serif-fontti.

Font Awesome
Vapaasti käytettävä kuvakefontti.