VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Henri Nousiainen

AJAX-POHJAISEN WEB-SOVELLUKSEN TOTEUTTAMINEN

Liiketalouden yksikkö

2014

VAASAN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma

TIIVISTELMÄ

Tekijä	Henri Nousiainen
Opinnäytetyön nimi	Ajax-pohjaisen web-sovelluksen toteuttaminen
Vuosi	2014
Kieli	suomi
Sivumäärä	50
Ohjaaja	Raija Tuomaala

Opinnäytetyössä luodun sovelluksen keskeinen tarkoitus oli toteuttaa selain-pohjainen, ergonominen, tehokas ja selkeä muistiinpanosovellus, joka on saatavilla kaikille laitteille, joissa on web-selain.

Työssä selvitettiin ja toteutettiin selain-pohjaisen muistiinpanosovelluksen vaatimuksia ja standardeja. Työ toteutettiin HTML-, CSS- Javascript-, PHP- ja MySQL-kielillä ja yhdistävänä tekniikkana käytettiin Ajaxia. Työn aikana selvitettiin myös käyttöliittymäsuunnittelun tieteellistä pohjaa ja käyttöliittymän ohjelmoinnin vaikutusta näyttöpääte-ergonomiaan.

Työn lähtökohdat olivat ongelmat, jotka ilmenivät tehdessä muistiinpanoja eri laitteilla. Eri muistiinpanosovellusten tallenteet tallentuivat kaikki omiin sijainteihinsa, mikä aiheutti muistiinpanojen hajaantumista ja saatavuuden heikkenemistä.

Jatkokehitys- ja loppusanat-osioissa esitellään näkemyksiä siitä, mihin sovellus kykenee tulevaisuudessa ja kuinka sovellusta aiotaan laajentaa.

Avainsanat Ajax, web-sovellus, ergonomia

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietojenkäsittelyn koulutusohjelma

ABSTRACT

Author	Henri Nousiainen
Title	Creating an Ajax-based Web-application
Year	2014
Language	Finnish
Pages	50
Name of Supervisor	Raija Tuomaala

The application created for this thesis was intended to be a browser-based, ergonomic, powerful and clear-looking web-application, which would be available for all the devices that have a web-browser. The application should be capable of creating, modifying and deleting notes.

The thesis itself researched the requirements and standards for a browser-based note-taking web-application. The application executed in this thesis was made with HTML-, CSS-, Javascript-, PHP- and MySQL-languages, with Ajax as the combining technique.

The work also covers the user interface design's scientific side and the effects of programming on the ergonomics of monitor-work. The starting points of this thesis were the problems appearing during school courses, in making notes with different devices. All the notes created with different applications saved the notes onto different locations, which caused disperse and difficulties in the availability of the notes.

In the final sections of the thesis I present views on the capability of the application in the future and also the ways in which the application will be developed further.

Keywords Ajax, web-application, ergonomics

SISÄLTÖ

1 JOHDANTO

Tässä opinnäytetyössä kehitetään Ajax-pohjaista muistiinpanosovellusta, joka on samalla helppokäyttöinen, tehokas sekä käyttömahdollisuuksiltaan laaja.

1.1 Taustaa ja kehityksen lähtökohdat

Työn lähtökohdat olivat ongelmalähtöiset. Koulun aikana kursseilla tekemäni muistiinpanot olivat kirjoitettu siihen laitteeseen mikä sillä hetkellä oli saatavilla. Välillä oli käytettävissä työpöytätietokone, välillä kannettava tietokone ja välillä pelkästään älypuhelin. Tästä johtui muistiinpanojen tallentuminen useaan eri paikkaan, jolloin niitä ei välttämättä ollut saatavilla juuri silloin kun niitä olisi tarvinnut.

Osin tästä syystä ja osin mielenkiinnosta aihetta ja tekniikoita kohtaan, aloin kehittämään muistiinpanosovellusta, joka on täysin selain-pohjainen. Päätin myös, että koko sovellus tullaan toteuttamaan Ajax-tekniikkaa käyttäen. Aloin kehittämään tätä sovellusta toisen kouluvuoteni aikana ja johtuen ajoittaisista viivästyksistä (joista enemmän "Haasteet"-kappaleessa), sain siitä hyvän kokeiluversion valmiiksi tätä opinnäytetyötä tehdessä.

Kehitystyön aikana olen ottanut huomioon pääasiassa itse tekemiäni tarpeita ja vaatimuksia, joita olen havainnut tarvittavan hyvän muistiinpano-ohjelman ominaisuuksissa. Tällaisia ominaisuuksia ovat esimerkiksi jatkuva tallentaminen muistiinpanoja tehdessä, palaute käyttäjän toimista, yksinomaan näppäimistöllä sovelluksen käyttäminen, ergonomia sekä mm. sovelluksen tavoitettavuus laitteesta riippumatta, jotka kaikki kuuluvat tämän sovelluksen ominaisuuksiin.

Sovellusta kehittäessäni havaitsin myös, että Ajax-pohjaiset web-sovellukset ovat mitä todennäköisimmin lähitulevaisuudessa pääasiallinen tapa tehdä web-sovelluksia. Web-sovellusten ominaisuuksien ja standardien kehittyessä niistä on

tulossa vahva haastaja niin työpöytäsovelluksille kuin myös mobiililaitteiden ns. natiivi-sovelluksille.

1.2 Työn tavoitteet

Tämän työn tavoitteena on toteuttaa web-sovellus Ajax-tekniikkaa hyödyntäen ja samalla tutkia web-kehitykseen liittyviä ajan tasalla olevia standardeja ja suosituksia. Tavoitteena on myös tutkia web-sovellusten kehitystä ja taitoja, joita tulevaisuudessa tarvitaan, jotta voidaan kehittää ajan tasalla olevia, niin web-, kuin mobiiliweb-sovelluksiakin.

1.3 Työn rajaukset

Web-sovelluksen toteuttaminen on laaja kokonaisuus sisältäen suuren määrän eri osa-alueita. Tässä työssä keskitytään nimenomaan sujuvan käyttöliittymän omaavan sovelluksen toimintoja suorittaviin osiin, kuten muistiinpanojen luomiseen, muokkaamiseen ja poistamiseen dynaamisesti Ajaxia käyttäen.

1.4 Seloste tekniikoista

- AJAX

  Ajax, akronyymi sanoista Asynchronous Javascript And XML, on yhdistelmä web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista saadaan vuorovaikutteisempia. Nykyisin Ajax-termiä käytetään viittaamaan toimintatapaan, jossa selainohjelma vaihtaa pieniä määriä tietoa palvelimen kanssa taustalla siten, ettei koko verkkosivua tarvitse ladata uudelleen käyttäjän tehdessä muutoksia sivulla.

  Ajax-tekniikan päämääränä on lisätä verkkopalvelun vuorovaikutteisuutta, nopeutta ja käytettävyyttä. (Shah 2007: 4-6)

  Tässä opinnäytetyössä Ajaxilla viitataan pääasiassa Javascript-funktioon, joka suorittaa toiminnon palvelimelle (tallennuksen, haun tms.).

- PHP

  PHP, lyhenne sanoista Hypertext Preprocessor, on ohjelmointikieli, jota käytetään erityisesti web-palvelinympäristössä dynaamisten web-sivujen luonnissa. PHP on ns. komentosarjakieli, jossa ohjelmakoodi tulkitaan vasta ohjelman suoritusvaiheessa. (PHP 2014)

- CSS

  CSS, kirjainlyhenne sanoista Cascaded Style Sheets, on WWW-dokumenteille kehitetty tyyliohjeiden laji. Pääasiassa CSS:llä muotoillaan web-sivujen eri elementtien ulkoasua. CSS:llä annetut säännöt ehdottavat, kuinka dokumentti voidaan esittää, se ei ole siis ehdoton, joten käyttäjä voi vaikka poistaa sen käytöstä halutessaan. (W3Schools 2014)

- HTML

  HTML (Hyper Text Markup Language) on ohjelmointikieli, jota käytetään web-sivujen rakenteen kuvaamiseen. Web-selain voi lukea HTML-tiedostoja ja muuntaa ne visuaalisiksi tai kuuluvaksi web-sivuksi.

(W3Schools 2014) Tässä opinnäytetyössä HTML on melko staattisessa osassa, sillä on pääasiassa tehty ainoastaan pääsivun HTML-rakenne, jota dynaamisesti muokataan Ajaxin avulla.

2 WEB-SOVELLUSTEN KEHITTYMINEN

W3C, eli World Wide Web Consortium on pääasiallinen, kansainvälinen standardiorganisaatio, joka hallinnoi World Wide Web:in standardeja ja suosituksia. (W3 2014) W3C kehittää web-sovellusten ominaisuuksia ja tekniikoita, jotka sallivat web-kehittäjille jatkuvasti enemmän työkaluja ja keinoja luoda web-sovelluksia, jotka ovat käytettävyydeltään aivan yhtä hyviä kuin työpöytäsovelluksetkin.

W3C kehittää myös mobiiliwebsovellusten puolella samoja ominaisuuksia, jolloin myös mobiiliwebsovellukset ovat käytettävyydeltään huomattavan paljon parempia aiempaan verrattuna. Tämä on helpoiten havaittavissa esimerkiksi mobiililaitteella kirjoitettaessa web-sivulla oleviin eri kenttiin tietoja, mikäli web-kehittäjä on asettanut input-elementin tyypiksi jonkin näistä uusista kenttätyypeistä.

Jos kyseessä on esimerkiksi kenttä, johon on tarkoitus kirjoittaa etunimi (input-tyyppi text), laite antaa täyden näppäimistön käytettäväksi. Sen sijaan, jos kyseessä on sähköpostiosoite (email), laite antaa täyden näppäimistön, mutta myös @-merkin ja esimerkiksi .com päätteen pikanäppäiminä samassa näytössä.

Suurimman eron huomaa, kun kentän määritelmäksi on annettu *number*, eli se hyväksyy numerotietoa, tällöin laite antaa pelkän numeronäppäimistön, joka on paljon parempi käytettävyydeltään, kuin täysi näppäimistö numerotietoja annettaessa, joka on suuri mukavuustekijä pientä näyttöä käytettäessä.

Sama koskee tietysti pc-tietokoneiden selaimia, niissäkin nämä ominaisuudet toimivat samalla tavalla ja esimerkiksi kenttä, joka on määritelty *date*-tyyppiseksi, avaa kentän alle pienen kalenterin, josta voi kätevästi valita halutun päivämäärän. Nämä ominaisuudet eivät ole vielä ulottuneet kaikkiin selaimien työpöytäversioihin. (W3C 2014)

Näilläkin keinoilla saadaan web-sovelluksista erittäin hyviä käytettävyydeltään.

3 SOVELLUKSEN ONGELMALÄHTÖINEN KEHITYSTYÖ

Koko aihe lähti liikkeelle ongelmalähtöisesti. Opiskeluaikanani muistiinpanojen tekeminen oli moneen eri ohjelmaan tehtyjen merkintöjen sekamelskaa. Ongelmana oli se, että esimerkiksi Microsoftin OneNotella sai tehtyä muistiinpanot hyvin, mutta se vaatii oman sovelluksensa asennettuna ja kaikille laitteille sitä ei ollut mahdollista asentaa. Toinen vaihtoehto taas oli DokuWiki, ilmainen wiki-tyyppinen muistiinpanosovellus selaimessa. DokuWikin ongelma taas oli se, että sen rakenteesta ja ominaisuuksista oli aika ajanut tyystin ohi eikä siinä ollut mobiililaitteille optimoitua tilaa.

Web-sovelluksiin perehdyttäessä tutustuin Ajax-tekniikkaan ja sen lisäksi opiskelimme myös PHP:tä sekä CSS:n käyttöä web-sovelluksia rakennettaessa. Tästä lähti idea luoda oma muistiinpanosovellus, joka toimisi täysin selain-pohjaisesti, on oma mobiililaitteille optimoitu tilansa ja jolla on OneNoten ja DokuWikin edut ilman haittoja.

Itse sovelluksen luominen oli sinänsä myös erittäin hyvä harjoitus koulun ohelle, koska siinä käytettiin juuri samoja elementtejä, kuin mitä koulussa kursseilla käytiin. Ennen kaikkea siinä on kaikki normaalin web-järjestelmän elementit, sisällön luomista tietokantaan, sen hakemista ja muokkaamista sekä poistoa. Sovellukseen kirjautumisenkin olen tehnyt, mutta sen rajaan tämän opinnäytetyön ulkopuolelle, koska pidän olennaisempana sovelluksen päätoimia.

Sovelluksen pääpiirteitä ovat muistiinpanojen luominen, muokkaaminen ilman erillistä muokkaa-tilaa ja poistaminen dynaamisesti Ajaxia käyttäen. Sovelluksesta on myös samaan järjestelmään kytkeytyvä mobiililaitteille optimoitu versionsa.

Mobiililaitteille optimoitua versiota ei käsitellä tässä opinnäytetyössä, koska tämä työ käsittää pääasiassa Ajax-sovelluksen toteuttamista tekniseltä kannalta. Mobiilipuolen versio käyttää kuitenkin samoja funktioita kuin työpöytäversiokin, vain

käyttöliittymää on yksinkertaistettu ja skaalattu soveltumaan pieninäyttöiselle, kosketusnäyttöiselle laitteelle.

3.1 Alkuvaihe

Koko sovelluksen periaatteena oli saada aikaiseksi mahdollisimman sujuva ja vakaa sovellus yksinkertaisella ja selkeällä ohjelmoinnilla. Sovelluksen kehittämisen alkuaikoina etsin ja tutustuin erilaisiin tekniikoihin, joita on käytetty nykyisten web-pohjaisten muistiinpano/tekstinkäsittelysovellusten tekemiseen, eli kieliin ja tekniikoihin, joita käytän tässä sovelluksessa. Näiden lisäksi myös oikoteitä löytyi.Yksi tällaisista oli Javascript/Ajax-kirjasto Jquery, joka on kyllä kätevä apu web-sovelluksia luodessa, mutta sen käyttö ei ollut tarkoituksenmukaista omassa kehitystyössäni.

Alusta alkaen päätin, että haluan tutkia kielien koodit ja rakenteet alusta alkaen ja oppia mahdollisimman paljon matkan varrella, enkä halua käyttää sovelluksessani ohjelmointikoodia tai -rakennetta, jota en täysin ymmärrä. Tuo päätös pätee vielä tänä päivänäkin, valmiita (kopioi/liitä) koodinpätkiä, joiden periaatetta en täysin ymmärrä, tai kirjastoja (Jquery yms.) en ole sovellusta rakentaessani käyttänyt.Muistiinpanosovellus tarjoaa loistavan tavan oppia normaalin web-sovelluksen toimintaperiaatteet, koska tässä sovelluksessa käydään läpi tiedon syöttämistä sivustolle, sen kuljettamista Javascriptin avulla PHP:lle ja PHP:n avulla syöttämistä MySQL-tietokantaan. Sen jälkeen, kun tiedot on syötetty, niitä muokataan ja poistetaan, joten myös nuo toimenpiteet tuli opetella.

3.2 Pikanäppäimet

Koulussa tehtyjen opintojaksojen aikana työskenneltiin luokkahuoneessa työpisteellä ja niiden aikana kirjoitettiin sekä käytettiin paljon web-selainta. Monen sivuston kohdalla olisi tärkeää kyetä käyttämään koko sivustoa halutessaan pelkällä näppäimistöllä.

Lähes kaikki sivustoja toki voi käyttää pelkästään näppäimistöllä, mutta mikäli tabindex-arvot eivät ole kohdallaan, eikä muistakaan pikanäppäimistä ole huolehdittu oikein, sivuston käyttäminen on, vaikkakin mahdollista, aivan liian epäkäytännöllistä. Osaltaan tästä syystä olen kehittänyt oman sovellukseni siten, että sitä voi käyttää pelkästään näppäimistöllä.

3.3 Sovelluksen koodirakenne

Sovelluksen koodirakenne on suhteellisen yksinkertainen, ja seuraavaksi selostan sen olennaisimmat toiminnot.

```
41  <body                                                    onkeyup='OlikoESC(event)'>
```

Edellä näytetyssä koodissa osoitetaan, että mikäli mitä tahansa painiketta painetaan, laukaistaan funktio *OlikoESC*.

```
332 function OlikoESC(e){
333  if(e.keyCode==27)
334  {document.getElementById("Etsi").focus();}}
```

Javascript-funktio määrittää ehdon avulla, onko kyseessä esc-painikkeen painallus.

Mikäli esc-painikkeen painallus havaittiin, kohdistus asetetaan hakupalkkiin, jonka jälkeen päästään kirjoittamaan hakupalkkiin, haetaan hakutulokset näytölle ja sen jälkeen tab-painikkeella voi siirtyä järjestyksessä hakutulosten välillä.

Kun haluttu sivu on valittu, se saadaan näkyviin enteriä painamalla.

Sivun otsikon ja sisällön välillä voi siirtyä tab-painikkeen ja shift-tab-yhdistelmän avulla, jonka mahdollistaa tabindex-arvot, jotka näkyvät seuraavassa kuvassa.

```
46 <div id='Otsikko' onmouseover='KorostaY(this.id)' onmouseout='PoistaKorostusY(this.id)' onclick='SuljeTyokalut()'
   tabindex='1'></div>
47 <div id='OtsikkoAika' onclick='SuljeTyokalut()'></div>
48 <div id='Sisalto' onmouseover='KorostaY(this.id)' onmouseout='PoistaKorostusY(this.id)' onclick='SuljeTyokalut()'
   tabindex='2'></div>
```

Edellä oleva koodi esittää, miten tabindex-arvot on asetettu.

Huomioitavaa on, että tabindex-arvoa ei ole asetettu aika-määrelle ollenkaan, koska sitä ei voi muokata. Sivustolla on käytetty myös accesskey-toimintoa sivustolla siirtymisen helpottamiseksi. Accesskeyn toiminnallisuus vaihtelee suuresti eri selainten välillä, joten se on enemmänkin lisätoiminto kuin varsinainen ominaisuus. Keskityn opinnäytteessä niihin ominaisuuksiin, jotka ovat saatavilla kaikissa selaimissa.

4 OHJELMOINNIN VAIKUTUS NÄYTTÖPÄÄTETYÖSKENTELYN ERGONOMIAAN

Kouluvuosien aikana olen jatkuvasti kehittänyt omaa laitteistoani, varsinkin näppäimistöä ja hiirtä, ergonomisempaan suuntaan johtuen siitä, että olen toistuvasti viettänyt pidempiä aikoja näyttöpäätteen edessä työskennellessä ja silloin huomaa työergonomian merkityksen, oli se sitten hyvä tai huono.

Tämä opinnäytetyö käsittelee Ajax-sovelluksen toteutusta lähinnä ohjelmoinnin ja suunnittelun osalta, mutta olen ottanut huomioon myös työergonomiaan liittyviä asioita niin paljon, kuin sovelluksen suunnittelulla voi siihen vaikuttaa. Olen tullut siihen tulokseen kouluaikani perusteella, että käytettäessä pelkästään näppäimistöä mahdollisimman paljon näppäimistön ja hiiren sijasta, olkavarsi ja ranteet rasittuvat vähemmän.

Web-sovellukset on usein rakennettu siten, että hiirtä joudutaan käyttämään paljon. Ranne ja olkapää joutuvat hiirellä työskennellessä olemaan pitkiäkin aikoja keskiasennosta poikkeavassa asennossa, joka aiheuttaa staattista jännitystä yläraajaan. (Ketola 2007, 72-74.) Siitä huolimatta, että hiirimalleja on pyritty kehittämään muotoilun, painikkeiden määrän ja tekniikan suhteen, ainoastaan kannettavassa tietokoneessa olevien hiirten kaltaisten ratkaisujen avulla kyetään välttämään olkavarren turhaa sivuloitonnusta sekä käden nostoa hiirelle ja takaisin näppäimistölle.

Unohtamatta kunnollisen näppäimistön, hiiren ja työasentoon vaikuttavien tekijöiden (työtuoli, työpöytä, näyttöjen korkeussäätö) merkitystä, viimeistelyn ergonomiaan tekee kuitenkin web-käyttöliittymän suunnittelija ja pikanäppäinten ohjelmoija, kun työskennellään käyttäen web-sovelluksia. Näiden seikkojen vuoksi myös kehittäessäni sovelluksia jatkossa, aion kiinnittää erityistä huomiota mahdollisimman ergonomiseen ja käytettävyydeltään parhaaseen tapaan käyttää niitä.

5 MUISTIINPANO-SOVELLUKSEN PERIAATTEET

Sovellus tallentaa käyttäjän syöttämän tekstin joka lyönnillä. Kun sivun otsikkoa tai sisältöä muokataan, sivun sisältö tallennetaan periaatteessa joka lyönnillä. Käytännössä joka lyönnin jälkeen käynnistyy 0,75 sekunnin viive, jonka jälkeen tallennetaan sisältö. Jos käyttäjä tämän ajastimen kuluessa painaa jotakin painiketta, ajastin lähtee jälleen alusta. Tällä pyritään estämään tarpeeton liikenne asiakkaan ja palvelimen välillä ja mahdolliset ongelmat, jotka aiheutuvat, kun yhden Ajax-pyynnön keskeyttää välittömästi seuraava toinen pyyntö.

Kaikki toimenpiteet asiakkaan ja palvelimen välillä on pyritty minimoimaan ja optimoimaan siten, että asiakkaan ja palvelimen välillä liikkuisi mahdollisimman vähän dataa kerrallaan ja suhteellisen usein. Tällä saadaan aikaiseksi sujuva käyttökokemus.

Sovellus on myös suunniteltu siten, että kaikki toimet, jotka käyttäjä suorittaa, kuten sivun luonti, poistaminen ja vastaavat kertaluonteiset tapahtumat ilmoitetaan käyttäjälle erillisellä inforuudulla, joka pyrkii varmistamaan sen, että käyttäjä tietää käynnistämänsä toiminnon suoriutuneen loppuun onnistuneesti. Sovellus ilmoittaa käyttäjälle lataus-animaatiolla työskentelevänsä pyydetyn toiminnon parissa, joten käyttäjän ei tarvitse miettiä, onko sovellus jämähtänyt tai muuten vain hidas. Animaation ilmaisee lopputilallaan myös sen, koska pyydetty toiminto on valmis.

Sovelluksen käyttöliittymää suunniteltaessa on keskitytty yksinkertaiseen ja selkeään ulkoasuun, jolloin käyttäjälle on mahdollisimman selvää, mistä mikäkin toiminto löytyy. Käyttäjä voi aina halutessaan käydä ohje-sivulla katsomassa, mitä mikäkin symboli tarkoittaa. Suunnittelulla on pyritty siihen, ettei sovelluksen käyttöä varten tarvitse käyttöohjetta, mutta mikäli haluaa saada käyttökokemuksestaan kaiken irti, ohje-sivulla on selitetty myös kaikki mahdolliset oikotiet sivuston käyttöön.

Koko sovellusta on mahdollista käyttää pelkästään näppäimistöä käyttäen. Tämä ei kuitenkaan vaikuta käyttökokemukseen, mikäli haluaa käyttää myös hiirtä normaalisti. HTML-ominaisuuksien osalta näppäimistön "integroiminen" sovellukseen on ollut suhteellisen mutkatonta. Accesskey- ja TabIndex-ominaisuudet helpottavat näppäimistöllä työskentelyä.

Accesskey-ominaisuus on HTML-ominaisuus, jonka voi asettaa esimerkiksi tietylle kirjaimella, jolloin esimerkiksi Internet Explorerissa seuraavan esimerkin ensimmäiseen linkkiin saa kohdistuksen käyttämällä näppäinyhdistelmää ALT+h. Tällöin asetetaan kohdistus linkkiin ja enteriä painamalla linkkiä voidaan käyttää.

```
<a href="http://www.w3schools.com/html"
accesskey="h">HTML tutorial</a><br>
<a href="http://www.w3schools.com/css"
accesskey="c">CSS tutorial</a>
```

Javascriptin voi tuoda tuohon esimerkkiin mukaan esimerkiksi asettamalla HTML-elementille onfocus-toiminto, jolloin kohdistus suorittaa vaikkapa Javascript-funktion, jolloin erillistä enter-painallusta ei tarvita.

Eri selaimen käsittävät nuo ominaisuudet eri lailla, kuten Internet Explorerin Accesskey-ominaisuuden painikkeet eroavat esimerkiksi Mozilla Firefoxin Accesskey-painikkeista, Explorer käyttää Accesskey-näppäimenä ALT-näppäintä ja Firefox ALT+SHIFT-yhdistelmää.

Näistä huolimatta olen integroinut noita komentoja sovellukseen ja näiden lisäksi olen lisännyt erilaisia toimintoja, kuten ESC-painikkeen käytön Javascriptillä. ESC-painiketta painaessa selaimen focus kohdistuu hakupalkkiin, jolloin hakupalkkia voidaan käyttää helposti ja nopeasti. Tässäkin toiminnossa tosin täytyy muistaa, että yleensä ESC-painike keskeyttää web-sivun latauksen.

Hieman vanhemmilla (alle 2 vuotta vanhoilla) selainversioilla sovelluksen pääominaisuudet toimivat, mutta esimerkiksi etsi-palkin selite-teksti ja vastaavat pienet ominaisuudet saattavat puuttua. Tätä vanhemmilla selaimilla sovellusta ei voi

joissain tapauksissa täysin käyttää, koska mikäli selain on tarpeeksi vanha, siltä saattaa puuttua tuki div-elementin muokkausta varten. Tällöin sovellusta voi käyttää ainoastaan selaamiseen, ei muokkaamiseen. Tietoturvaankin on panostettu sovellusta kehitettäessä ja erityisen tärkeää on se, että käyttäjän tekemät muistiinpanot varmasti tallentuvat.

Tämä sovellus on nimenomaan muistiinpanojen tekemistä varten, kyseessä ei ole tekstinkäsittelyohjelma. Jatkokehitysvaiheessa sovellukseen on suunnitteilla yksinkertaiset tekstin muotoiluun tarkoitetut työkalut (lihavointi, kursivointi, alleviivaus yms.), mutta sen pidemmälle ei tekstinkäsittelyominaisuuksia todennäköisesti tule.

6 PUHTAASTI AJAX-SOVELLUS

AJAX on tekniikka, jonka ympärille koko sovellus on rakennettu. Muistiinpanosovelluksia, jotka toimivat osittain Ajaxin avulla on paljon, esimerkkinä Google Docs. Sen sijaan täysin Ajax-pohjaisia sovelluksia en ole toistaiseksi löytänyt ja varsinkin web-kehittäjien foorumeilta tukea hakiessani, kun kehitin omaa sovellustani, kävi ilmi, että puhtaat Ajax-sovellukset koetaan toistaiseksi sen verran haasteellisiksi, ettei niitä kannata tehdä, tai jos niitä tehdään, ne tehdään Jqueryllä.

Olen sovellustani kehittäessä törmännyt erinäisiin ongelmiin, joista suurimpaan osaan olen joko löytänyt ratkaisun suoraan tai muovannut oman ratkaisun. Puhtaasti Ajax-sovelluksella tarkoitan tässä sitä, että sivulle saapumisen jälkeen sivua ei enää ladata missään vaiheessa uudelleen päätoimintoja käytettäessä. Näin saadaan mahdollisimman sujuva käyttökokemus.

6.1 Miksei työpöytäsovellusta?

Työpöytäsovelluksen periaate, varsinkin muistiinpanosovelluksen osalta alkaa olla jossain määrin vanhentunut web-tekniikoiden kehittyessä. Sovelluksen tietoja, kuten tässä tapauksessa muistiinpanoja saattaa tarvita odottamattomassa paikassa ja silloin on suuri etu, että muistiinpanot ovat internetin välityksellä saatavilla ilman erikseen asennettavaa sovellusta laitteella kuin laitteella.

6.2 Mobile apps - web-sovellus, miksen kehitä app-sovellusta?

Yksi suurista eduista web-sovelluksella on se, että kun sen ohjelmistoa päivitetään, päivitys tapahtuu automaattisesti, eikä asiakkaan tarvitse huolehtia ohjelmistonsa päivittämisestä.

Mobiililaitteissa suuressa suosiossa olevat 'appsit' ovat hyviä vaihtoehtoja nekin,mutta ne ovat käyttöjärjestelmäkohtaisia. Web-sovellus on sen sijaan saatavilla jokaisella laitteella, jossa on web-selain, joka käytännössä tarkoittaa, että kaikkien käyttöjärjestelmien käyttäjät ovat potentiaalisia sovelluksen käyttäjiä.

Kehitän sovelluksesta myös erillisen mobiililaitteille soveltuvan käyttöliittymän, joka käyttää samoja funktioita ja toimintoja kuin työpöytäversiokin, mutta se on optimoitu pienemmille näytöille ja siitä on karsittu pois esimerkiksi animaatiot, jotka saattavat heikkotehoisemmassa tabletissa tuottaa esimerkiksi tulkinnan hitautta. Mobiililaitteille soveltuvan version kehittäminen ei kuitenkaan kuulu tämän opinnäytetyön sisältöön.

6.3 Kuvien selitteet

Tässä opinnäytetyössä olevissa kuvissa on tietoturvaan liittyviä kohtia, jotka on kuvista poistettu. Tämän vuoksi kuvissa on valkoisia palkkeja. Koodin periaate on kuitenkin tästä huolimatta sama.

7 SOVELLUKSEN NOPEUS

Mittasin sovelluksen nopeutta asettamalla ajastimen siihen vaiheeseen, kun asiakkaan pyyntö tulee palvelimelle. PHP-toiminnon alusta aina siihen asti, kun tietokantakysely on valmis palautettavaksi. Tämä selite on olemassa siksi, että webkehittäjänä en voi vaikuttaa asiakaspäätteen käsittelynopeuteen Javascriptin osalta (turhan monimutkaista koodausta lukuun ottamatta) enkä aikaan, joka kuluu asiakaspäätteen ja palvelimen välillä kulkemiseen.

```
35 $AloitaAjanLasku = microtime(true);
```

Edellä näytetyssä koodissa esitetään, kuinka ajan laskeminen aloitetetaan. Ensin asetetaan muuttujaksi $AloitaAjanLasku Microtime-funktiolla aikamääreen alku.

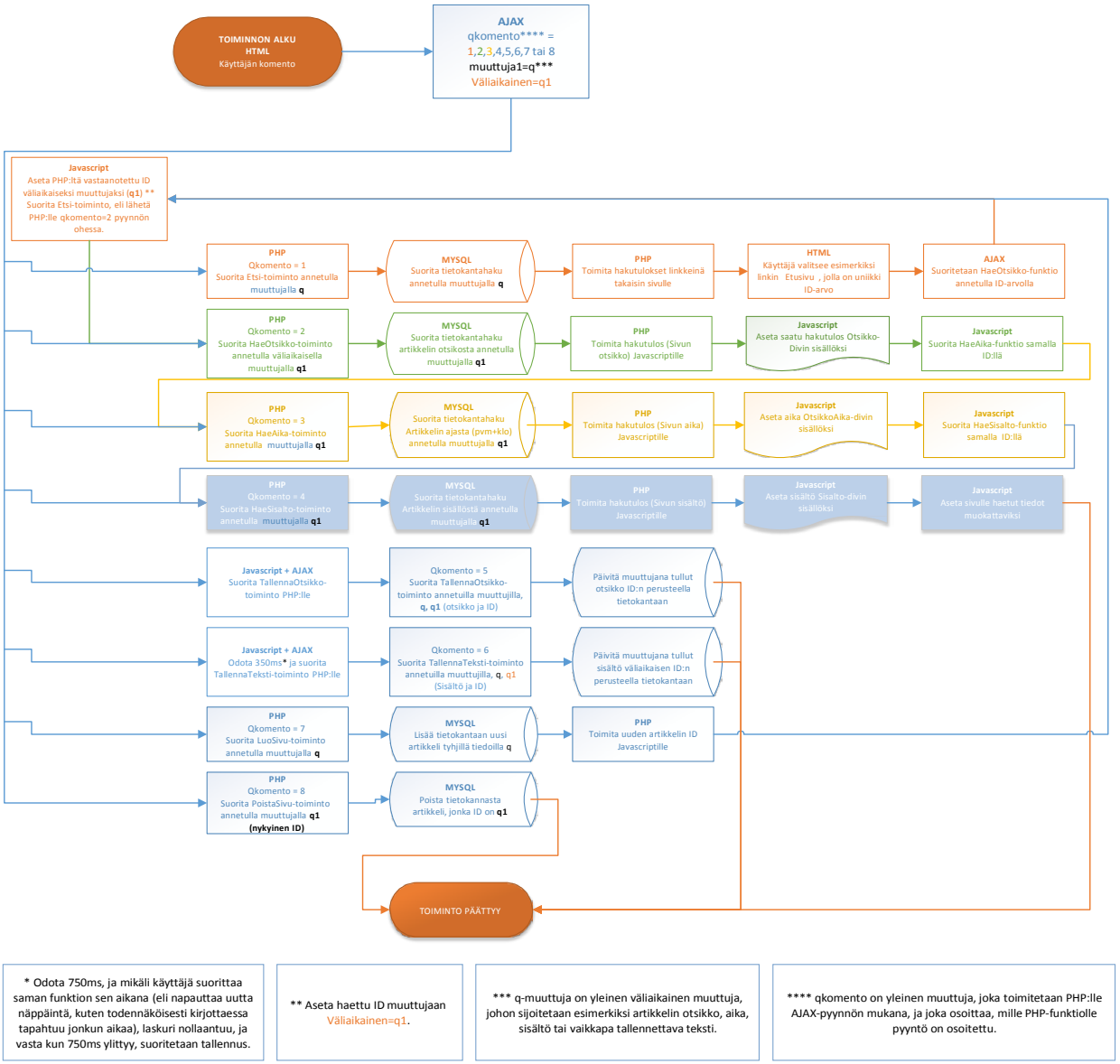
```
55 $LopetaAjanLasku = microtime(true);
56 $Kesto = $LopetaAjanLasku - $AloitaAjanLasku;
57 echo "<br><p title='Kyselyn kesto.' class='KeskitysOikeaAlas'>" . round($Kesto, 3) .  "s</p>";}
```

Tietokantakyselyn jälkeen otetaan uusi aikamääre samalla funktiolla muuttujaksi $LopetaAjanLasku, jonka jälkeen vähennetään muuttujan $LopetaAjanLasku arvosta muuttujan $AloitaAjanLasku arvo, jolloin saadaan tulokseksi kyselyssä kestänyt aika.

Lopuksi ennen ajan näyttämistä arvo käsitellään näyttämään kolme desimaalia kokonaisarvon jälkeen komennolla round(). Nämä tiedot ovat olennaisia webkehittäjälle, mutta myös käyttäjästä on hyvä tietää, kuinka kauan varsinaiseen toimintoon meni.

8 SOVELLUKSEN RAKENNE JA TOIMINNOT

Seuraavassa kaaviossa on selitetty koko sovelluksen toiminta vuokaaviona.



Kuva 1. Sovelluksen toiminta

Koko sovelluksen toimintaperiaatteeseen perehtyessä on hyvä tietää kaksi olennaista tekniikkaa, joita käytän.

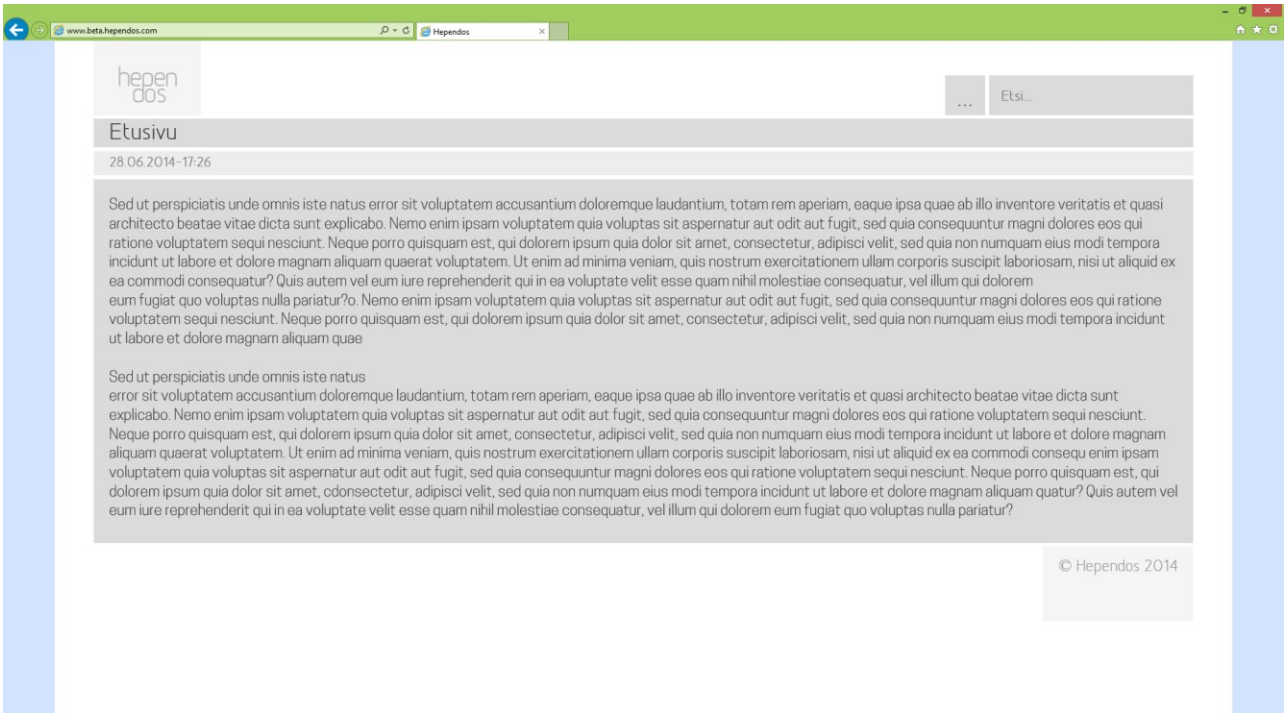
1. Javascriptiltä lähetettäessä pyyntöä PHP-tiedostolle, mukana lähtee muuttuja nimeltä *qkomento,* joka määrittelee toiminnon, joka PHP-tiedostossa on tarkoitus suorittaa. Tästä aiheesta lisää luvussa "**Sovelluksen rakenteiden selkeyttäminen ja yhtenäistäminen**".

2. Kun haetaan sivun tietoja, olen ketjuttanut kolme funktiota yhteen, otsikon hakemisen, ajan hakemisen ja sisällön hakemisen. Javascript-funktioissa voi nähdä kahden ensinnä mainitun funktion lopussa funktio-kutsun aina seuraavaan funktioon.

8.1 Sovelluksen HTML-rakenne

Sovelluksen HTML-rakenne on yksinkertainen, useissa tapauksissa samoja elementtejä käytetään eri käyttötarkoituksiin, kuten sisältö-elementissä näytetään sekä sivun sisältöä että esimerkiksi hakutuloksia.

Näkymä, joka näkyy ilman että käyttäjä on kirjautunut on pelkistetty versio kirjautuneen käyttäjän näkymästä, jossa kirjautuneen käyttäjän toiminnot on poistettu käytöstä. Kaikki ruutukaappaukset selaimesta on otettu Internet Explorer-selaimella, versiolla 11.



Kuva 2. Sovelluksen etusivu

```
<body                          onkeyup="OlikoESC(event)">
<div id="Info">[tässä infoteksti]</div>
<div id="Hependos" onclick="HaeOtsikko(1)" title="Siirry takaisin pääsivulle"></div>
<div id="Tallennetaan" onclick="SuljeTyokalut()">
</div><input type="text" onclick="SuljeTyokalut()" tabindex="-1" id="Etsi" class="EtsiPalkki" onkeyup="Etsi(event,this.value)" onmouseover
="KorostaY(this.id)" onmouseout="PoistaKorostusY(this.id)"  placeholder="Etsi..." />
<div id="Otsikko" onclick="SuljeTyokalut()" tabindex="1"></div>
<div id="OtsikkoAika" onclick="SuljeTyokalut()"></div>
<div id="Sisalto" onclick="SuljeTyokalut()" tabindex="2"></div>
<div id="Kayttaja" onclick="SuljeTyokalut()">                          </div>
<div id="TyokalutIkoni" title="Avaa työkalut" onclick="AvaaTyokalut()" onmouseover="KorostaY(this.id)" onmouseout=
"PoistaKorostusY(this.id)">...</div>
<div id="UusiSivu" onclick="LuoSivu();SuljeTyokalut()" title="Luo uusi sivu" onmouseover="KorostaY(this.id)" onmouseout=
"PoistaKorostusY(this.id)">+</div>
<div id="TyokalutDiv" ><p id="julkinen" class="TekstiKeskitysVasen" onmouseover="KorostaPainike(this.id)" onmouseout=
"PoistaPainikkeenKorostus(this.id)">Julkinen-ruutu</p><hr><p id="poista" onmouseover="KorostaPainike(this.id)" onmouseout=
"PoistaPainikkeenKorostus(this.id)">Poista tämä sivu </p><p onclick="KirjauduUlos()" id="kirjauduUlos" onmouseover=
"KorostaPainike(this.id)" onmouseout="PoistaPainikkeenKorostus(this.id)">Kirjaudu ulos </p></div>
<div id="Poisto"></div>
<div id="Copyright" title="Hependos Engine 0.73 (pre-alpha)">&copy; Hependos<br>2014 </div>
<div id="VasenPalkki"></div>
<div id="OikeaPalkki"></div>
</body>
```

Sivun elementit ovat (kuten alla seuraavasta kuvasta ja edellä näytetystä koodista havaitaan):

- Otsikko (Div-elementti, jossa näytetään otsikko)
- Sisalto (Div-elementti, jossa näytetään esimerkiksi muistiinpanon sisältö tai hakutuloksia)
- Etsi (Input-elementti, johon syötetään hakusanoja tai komentoja)

- Aika (Div-elementti, joka näyttää muistiinpanon luontipäivämäärän ja ajan)
- Tyokalut (Div-elementti, jota klikatessa avataan saatavilla olevat toiminnot, joilla voi muokata avoinna olevaa muistiinpanoa)
- Luosivu (Div-elementti, jota klikatessa luodaan uusi muistiinpano ja avataan se otsikko- ja sisalto-elementteihin.)
- Kayttaja (Div-elementti, joka näyttää kirjautuneen käyttäjän käyttäjänimen.)
- Info (Div-elementti, jossa näytetään sovelluksen ilmoituksia, esim. "Sivu luotu")
- Copyright (Div-elementti, jossa näytetään tekijänoikeustietoa sekä toiminnolla "onmouseover" näytetään sovelluksen versiotietoja)
- VasenPalkki (Div-elementti, joka muotoilee sivun reunan)
- OikeaPalkki (Sama kuin edellinen, oikealla puolen sivua)

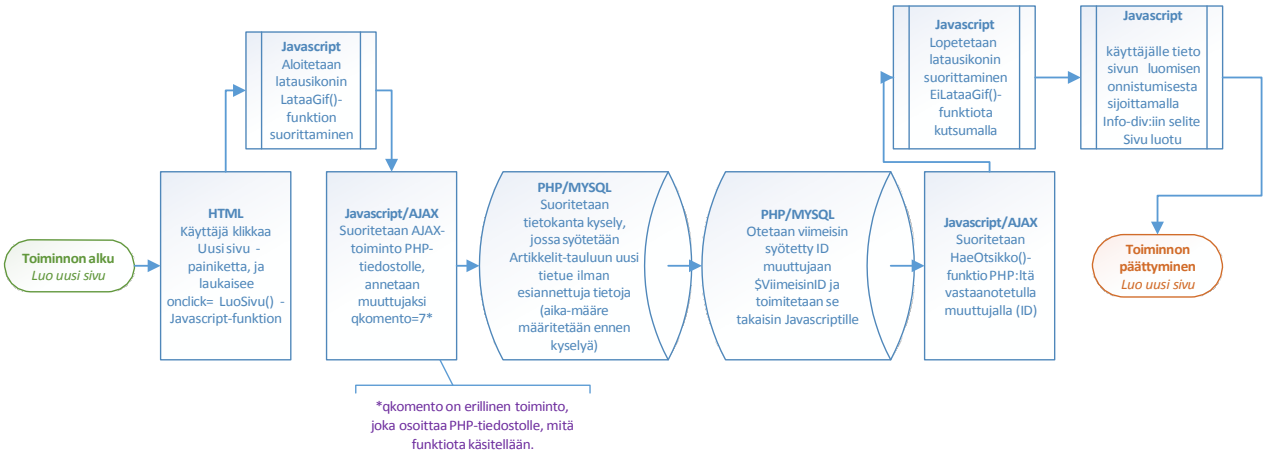
Kuva 3. Kirjautuneen käyttäjän näkymä

8.2 Muistiinpanon luominen

Muistiinpanojen luominen aloitetaan klikkaamalla sivulla olevaa uusi sivu-ikonia (+-merkki), joka suorittaa seuraavan Ajax-komennon (nyt käsiteltävänä olevat rivit on korostettu).

Seuraavassa koodissa luodaan yksinkertainen Ajax-kutsu, jonka mukana lähetetään palvelimelle parametri "qkomento=7", joka vastaa komentoa luoda uusi sivu. Pyyntö lähetetään POST-metodilla keskus.php-tiedostolle.

```
317 function LuoSivu(){LataaGif();
318 xmlhttp=new XMLHttpRequest();
319 xmlhttp.onreadystatechange=function(){
320 if (xmlhttp.readyState==4 && xmlhttp.status==200){
321 document.getElementById("Info").innerHTML="  &#10003; Sivu luotu.";
322 NaytaInfo("ilmoitus");
323 HaeOtsikko(xmlhttp.responseText);
324 document.getElementById("Otsikko").focus();
325 EiLataaGif();
326 }}
327 xmlhttp.open("POST","keskus.php",true);
328 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
329 xmlhttp.send("qkomento="+7);}
```



Kuva 4. Muistiinpanon luominen

```
177 else if($qkomento==7){
178
179
180 $date=date_create();
181 date_add($date,date_interval_create_from_date_string("3 hours"));
182 $aika=date_format($date,"Y.m.d.H.i.s");
183 $sql="INSERT INTO Artikkelit (ID, Sisalto, Aika, Nimi,                    )
184 VALUES
185 ('', '', '$aika', '', '$q5', '$q6', ' ')";
186 $result = mysql_query($sql,$con);
187 $ViimeisinID = mysql_insert_id();
188 echo $ViimeisinID;
189 }
```

Suoritetaan SQL-komento, joka luo sivun annetuilla arvoilla, jonka jälkeen haetaan mysql_insert_id-komennolla viimeisin ID-arvo, joka tauluun on syötetty (eli juuri syötetty arvo) ja lähetetään se takaisin Javascriptille. Kun sivu on luotu, käyttäjälle annetaan vahvistus sivun luonnista ja ohjataan käyttäjä juuri luodulle sivulle HaeOtsikko-funktiolla, jonka arvo saatiin PHP:ltä, kuten seuraava koodirakenne osoittaa.

```
317 function LuoSivu(){LataaGif();
318 xmlhttp=new XMLHttpRequest();
319 xmlhttp.onreadystatechange=function(){
320 if (xmlhttp.readyState==4 && xmlhttp.status==200){
321 document.getElementById("Info").innerHTML="  &#10003; Sivu luotu.";
322 NaytaInfo("ilmoitus");
323 HaeOtsikko(xmlhttp.responseText);
324 document.getElementById("Otsikko").focus();
325 EiLataaGif();
326 }}
327 xmlhttp.open("POST","keskus.php",true);
328 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
329 xmlhttp.send("qkomento="+7);}
```

NaytaInfo-funktio, jonka kutsu on nähtävissä yllä olevan kuvan rivillä 322, näyttää tietoa juuri tapahtuneesta toiminnosta käyttäjälle, kuten seuraavana olevassa kuvassa kutsutun funktion *NaytaInfo*-koodi osoitetaan.

```
290 function NaytaInfo(str){LataaGif();
291 if(str=="virhe"){
292 document.getElementById("Info").style.visibility="visible";
293 document.getElementById("Info").style.backgroundColor="red";
294 setTimeout(function(){document.getElementById("Info").style.visibility="hidden";},4000);
295 EiLataaGif();
296 }
297 else if(str=="ilmoitus"){
298 document.getElementById("Info").style.backgroundColor="#D7FAD7";
299 document.getElementById("Info").style.visibility="visible";
300 setTimeout(function(){document.getElementById("Info").style.visibility="hidden";},4000);
301 EiLataaGif();
302 }}
```

Funkiota kutsutaan ja sille annetaan arvo "virhe" tai "ilmoitus", riippuen onnistuiko tähän ilmoitukseen liittyvä toiminto vai ei. Kutsuvassa koodin osassa näytetään myös, että haluttu ilmoitus sijoitetaan Info-elementin sisällöksi ennen NaytaInfo-funktion kutsumista.

Molemmissa tapauksissa, oli kyse sitten ilmoituksesta tai virheilmoituksesta, div-elementti on näkyvillä 4000 millisekuntia, jonka jälkeen sen tyylitiedostossa määriteltyyn ominaisuuteen *visibility* asetetaan hidden.

Kuva 5. Sivun luonnin selite

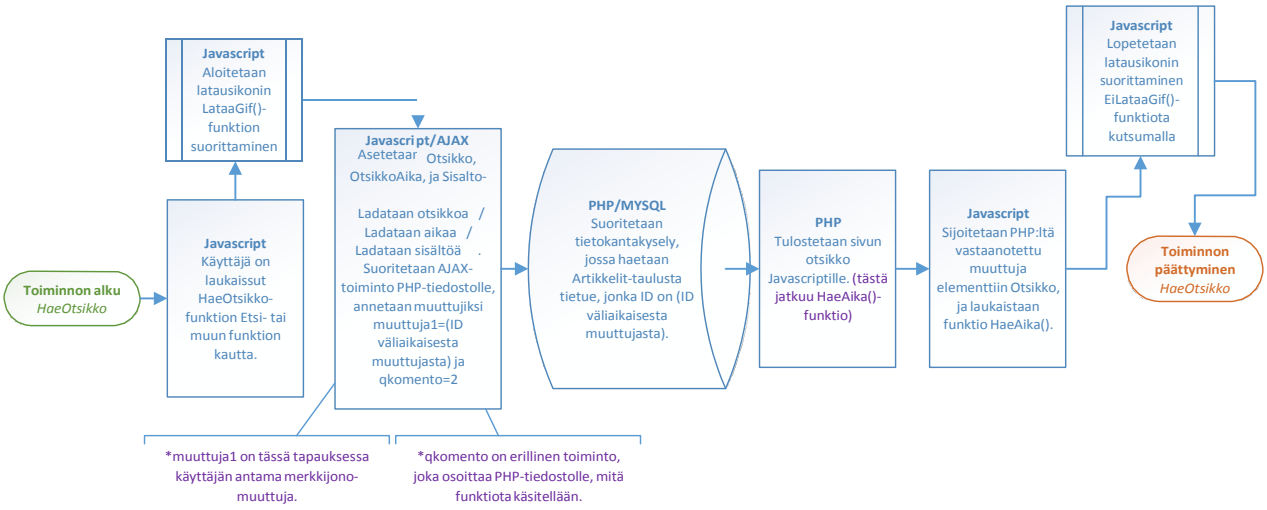
Edellinen kuva esittää, että sivun luonti onnistui ja käyttäjälle annetaan siitä tieto info-elementin tekstillä "Sivu luotu". Tämä tieto on siis esillä 4000 millisekuntia, samoin kuin alempikin virheilmoitus.

Seuraavassa kuvassa näytetään virheilmoitus toiminnon epäonnistuessa.

Kuva 6. Epäonnistuneen toiminnon ilmoittaminen

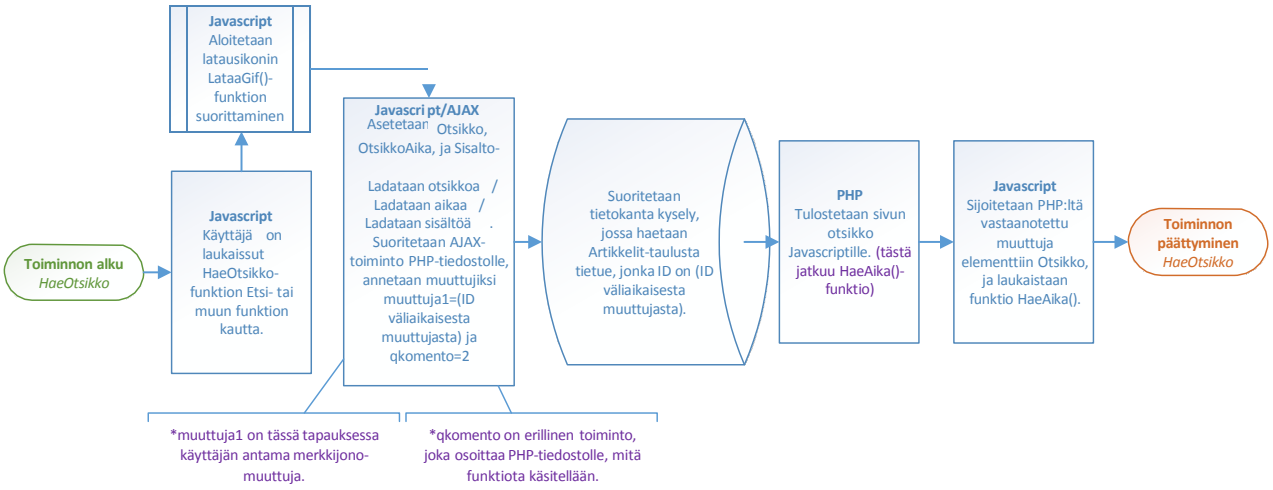
8.3 Muistiinpanojen etsiminen

Seuraavassa kaaviossa osoitetaan, kuinka käyttäjä laukaisee Javascript-funktion kirjoittamalla hakusanan etsi-palkkiin, josta käynnistyy funktio-sarja suorittaen haku-toiminnon.



Kuva 7. Muistiinpanojen etsiminen

Seuraava kaavio näyttää, kuinka Etsi-funktiolta saadulla muuttujalla suoritetaan Otsikon haku.

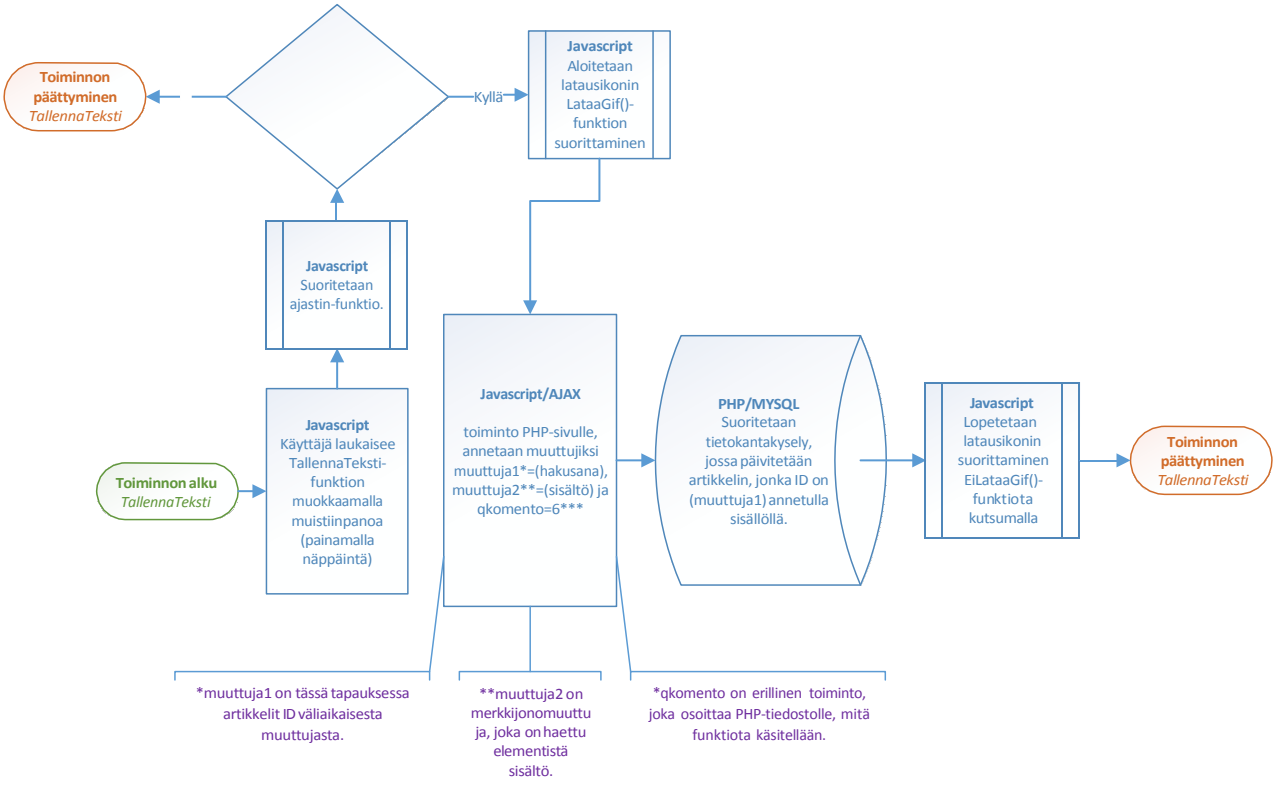


Kuva 8. Otsikon haku

HaeAika- ja HaeSisalto-funktiot ovat rakenteeltaan samankaltaisesti toteutettuja kuin HaeOtsikko-funktiokin, joten selkeyden vuoksi en niitä erikseen näytä.

8.4 Muistiinpanon muokkaaminen

Seuraavassa kaaviossa on esitetty toiminnot, jotka suoritetaan, kun käyttäjä muokkaa muistiinpanoa. Otsikon muokkaamisessa käytetään samankaltaista toimintoa, joten sitä en erikseen esitä.



Kuva 9. Tekstin muokkaaminen

Muistiinpanojen muokkaaminen on yksi keskeisimpiä toimintoja koko järjestelmässä ja se onkin kehittynyt jatkuvasti paremmaksi. Alun perin sivun tallentaminen tapahtui joka lyönnillä, eli heti näppäimen ylös tultua Javascript-funktio alkoi suorittaa tallennusta.

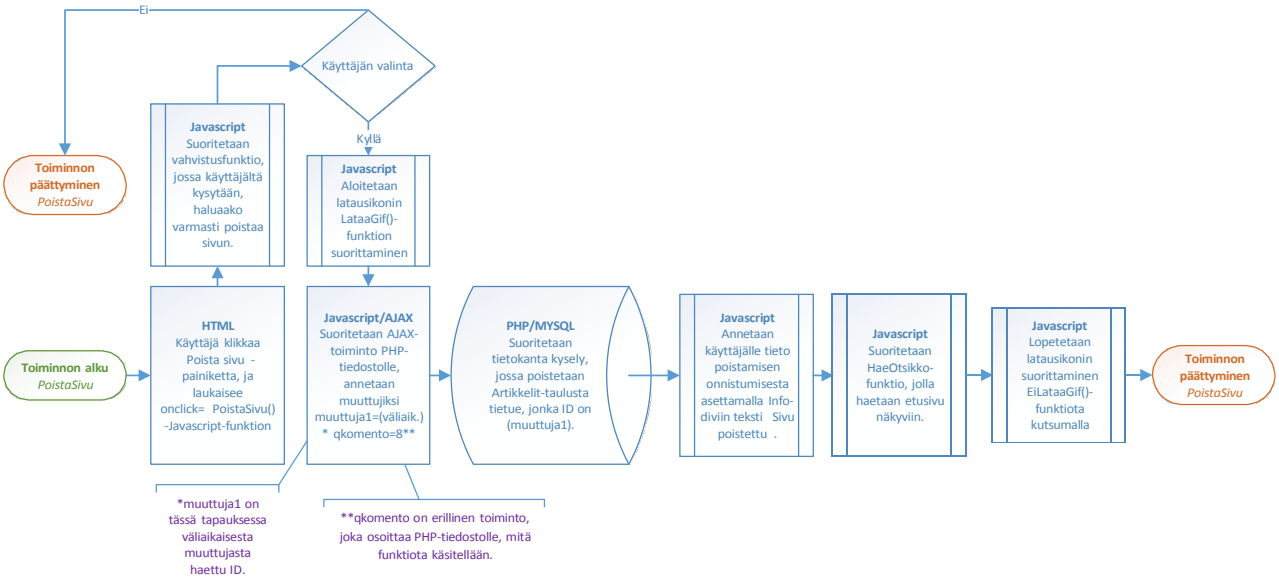
Tästä jatkojalostettu versio on nykyinen versio, jossa teksti tallennetaan joka lyönnillä, mikäli 750ms ehti kulua käyttäjän näppäimenpainallusten välillä. Tämä viive ehkäisee aiheetonta liikennettä asiakkaan ja palvelimen välillä.

```
229 var delay = (function () {
230         var timer = 0;
231  return function (callback, ms) {
232           clearTimeout(timer);
233                timer = setTimeout(callback, ms);
234      };
235 })();
```

```
238 function TallennaTeksti(sisalto0){
239 delay(function(){
240 LataaGif();
241
242 var sisalto
243 var id = NykyinenID;
244 xmlhttp=new XMLHttpRequest();
245 xmlhttp.onreadystatechange=function(){
246 if (xmlhttp.readyState==4 && xmlhttp.status==200){
247 EiLataaGif();
248 }}
249 xmlhttp.open("POST","keskus.php", true);
250 xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
251 xmlhttp.send("muuttuja1="+id+"&muuttuja2="+sisalto+"&gkomento="+6);
252 },750);
253 }
```

Edellisessä koodissa, riveillä 229–235 on ajastin-funktio, jota kutsutaan Tallenna-Teksti-funktiosta ja mikäli käyttäjä siis kutsuu TallennaTeksti-funktiota uudelleen ennen kuin 750ms on ehtinyt kulua, TallennaTeksti-funktiota ei suoriteta. Muuttuja, jonka funktio ottaa vastaan, sisalto0, on sisällöltään sama kuin sisalto-muuttuja, joka lähetetään PHP:lle send-osiossa.

8.5 Muistiinpanon poistaminen



Kuva 10. Muistiinpanon poistaminen

Kun muistiinpano poistetaan, se poistetaan samankaltaisella kyselyllä kuin artikkelin luominenkin, vain koodin rakenne hieman muuttuu, tämän vuoksi en sitä tähän laita.

9 KÄYTTÖLIITTYMÄN MUOTOILU: SUURIN HAASTE

Käyttöliittymän suunnittelu on ollut ehdottomasti suurin haaste sovellusta suunniteltaessa. Olen hakenut inspiraatiota muista sovelluksista. Niistä vaikutteita saaneena en silti ollut tyytyväinen suunnittelun lopputulokseen.

Suurin ongelma on ollut puolittainen kopiointi toisen sovelluksen ulkoasusta, joka ei ole, vaikka lopputulos olisikin ollut siisti, haluttua. Sivun alaosaan ei voisi tällaisessa tapauksessa hyvällä eikä huonolla omallatunnolla laittaa tekijänoikeuteen viittaavaa merkintää. Siksi olen kehittänyt ulkoasun itse sekä värimaailman, muotoilun ja animoinnin osalta.

Ulkoasun suunnittelu on myös rakenteellinen kysymys. Mikäli painikkeiden paikkoja päätetään vaihtaa tai esimerkiksi jokin valintalaatikko kokonaan poistaa, se aiheuttaa varsinkin Javascript-tiedostoon suuriakin muutoksia. Tästä syystä koko sovelluksen kehitys on ajoittain täysin pysähtynyt ja välillä jopa siirtynyt kehityksessä taaksepäin päiviä, jollei viikkojakin.

Tämän lisäksi suuressa osassa tapauksista, jolloin on tehty suurempia muutoksia koko sovellukseen, joitakin päiviä myöhemmin on kuitenkin palattu aiempaan versioon, joka taas aiheutti sovelluskehityksen hidastumista. Muutoksien ansioista on kuitenkin tapahtunut kehitystä niin ohjelmoinnin kuin varmuuskopioinnin kannalta. Edellä mainitun kaltaisissa tilanteissa on tärkeää, että ns. alkutilasta on varmuuskopio, jolloin kehitystä voidaan jatkaa suoraan ilman, että tarvitsee suorittaa koodin "palauttamista" aiempaan tilaan. Siksi tallennankin aina ennen suurempia muutoksia koko sovelluksesta varmuuskopion talteen.

Ulkoasusta on alusta lähtien ollut tarkoitus tulla selkeä sekä mahdollisimman pelkistetty. Käyttöliittymää hallitsevat otsikko sekä sisältö. Ylhäällä ovat kolme painiketta, työkalut-, luo sivu- sekä käyttäjänimi-painikkeet.

Näiden painikkeiden osalta päädyin tekemään grafiikan ja hiiren liikkeisiin vastaavan animoinnin puhtaasti ohjelmoimalla. Painikkeiden pelkistetyt merkit, kolme pistettä ja plus-merkki ovat normaalilla, hieman suurennetulla tekstikoolla tehdyt.

Hiiren liikkeeseen vastaavan toiminnon olen tehnyt muokkaamalla Javascriptillä painikkeiden taustan väriä.

```
419 function KorostaY(str){
420 document.getElementById(str).style.backgroundColor="#A1B6F6";
421 }
```

Kun hiiri siirretään pois elementin päältä, suoritetaan korostuksen poisto, kuten alla olevassa koodissa esitetään.

```
423 function PoistaKorostusY(str){
424 document.getElementById(str).style.backgroundColor="#EDEDED";
425 }
```

Kuva 11. Painikkeen korostus

Edellinen kuva näyttää, miltä korostus näyttää itse sivulla. HTML-elementin title-ominaisuuskin näkyys samassa kuvassa kertomassa, mitä painike painettaessa tekee. Lataus-ikonin olen tehnyt CSS-animaatiolla, joka on kaksiosainen. Ensimmäisessä osassa animaation pyörii loputtomasti, jonka katkaisee toinen animaatio, joka häipyy punaisesta vihreään.

```
@keyframes VaihdaVari {
0% {background: transparent;}
25% {background: red;}
50% {background: red;}
75% {background: red;}
100% {background: transparent;}
}

@keyframes Loppu {
0% {background: transparent;}
25% {background: green;}
50% {background: green;}
75% {background: green;}
100% {background: transparent;}
}
```

Itse sivuston ulkoasu on ns. palikkamainen, koska tällainen ulkoasu luo vaikutelman yksinkertaisesta käyttöliittymästä ja yksinkertainen usein mielletään helpoksi käyttää. Googlen tutkimuksen mukaan (elokuussa 2012) web-sivuston käyttäjä päättää sekunnin murto-osissa sen, onko sivusto näyttävä vai ei. Samassa tutkimuksessa havaittiin, että käyttäjät pitävät selvästi miellyttävämpinä yksinkertaisen näköisiä sivustoja verrattuna monimutkaisen näköisiin sivustoihin. Lyhyesti sanottuna tutkimustulos kertoo, että mitä yksinkertaisempi ulkoasu, sen parempi.

Googlen tutkimuksessa havaittiin myös, että käyttäjät kokevat esimerkiksi verkkokaupan käytön helpommaksi, mikäli se on "samankaltainen kuin vastaavat verkkokaupat", joka tarkoittaa pääasiassa esimerkiksi eri elementtien sijoittelua, missä hakupalkki sijaitsee ja niin edelleen. Olennainen osa tässä huomiossa on se, että ihmisen aivot haluavat mielummin käsitellä asioita, jotka ovat helppoja käsitellä. Tämän vuoksi esimerkiksi web-sivuston käyttäjä käy sivuilla, joilla hän vaistomaisesti tietää, missä mikäkin kytkin tai valikko on. (Walker 2013: www)

Olen ottanut näitä asioita huomioon sivuston ulkoasua suunnitellessani ja siksi esimerkiksi hakupalkki on jatkuvasti näkyvissä sivun oikeassa yläreunassa ja valikot, joilla sivua voi muokata, ovat näkyvillä sivun yläosassa. Näistä asioista huolimatta sovellukseni lähtökohta on ollut ongelmapohjainen, joten etusijalla ovat olleet omat tarpeeni. Ulkoasu on kuitenkin yksi niistä tekijöistä, joka elää jatkuvasti.

9.1 Värimaailma (näytön kalibrointi, värien merkitys)

Käyttöliittymän suunnittelussa väreillä on merkitystä. Sovellusta kehittäessäni omassa työpisteessäni on kolme eri ikäistä näyttöä, yksi LCD-näyttö ja kaksi LED-näyttöä. Kaikkien kolmen värien lämpöisyys on erilainen huolimatta siitä, että olen yrittänyt kalibroida ne mahdollisimman samankaltaiseksi. Tästä johtuen värien sävyjen asettaminen on erittäin haasteellista, koska jokaisella näytöllä värit olivat todella eri sävyisiä.

Tummat sävyt ovat selkeämpiä eri näytöillä, mutta vaaleammat ja todella vaaleansävyiset, kuten harmaanvalkoinen väri, näytti yhdellä näytöllä selkeästi erottuvan taustasta ja taas toisella se oli melkein taustan värinen. Näitä värejä ei luonnollisesti voi saada aina näkymään käyttäjälle sellaisena kuin haluaa, koska eri käyttäjien näytöt ovat todennäköisesti eri tavalla säädetty kirkkauden, kontrastin ja muiden säädösten osalta.

Kompromissina käytin sellaisia värejä ja sävyjä, jotka erottuvat joka tapauksessa taustasta, vaikka näytössä olisi enemmän kirkkautta tai kontrastia. Käytin myös apuna kannettavan tietokoneen ja tabletin näyttöjä, koska niissä ei ole erillisiä säätimiä näytön asetuksille.

9.2 Värit sivustolla

Käytin sivustolla värimaailmaa, joka sisältää harmaata, vaaleanharmaata, valkoista sekä vaalean sinistä. Korostevärinä (esimerkiksi painikkeiden korostuksena) käytin tummempaa sinistä. Koko väri-ilmeen on tarkoitus olla samalla luotettavan, puhtaan, selkeän ja asiallisen oloinen. Valkoinen väri ilmaisee puhtautta ja aitoutta, sininen luotettavuutta, harmaa selkeyttä ja asiallisuutta. (Nolan 2014: www.)

9.3 Käyttöliittymän sujuvuus

Käyttöliittymän vuorovaikutteisuus on suuressa osassa sivustollani, käyttäjälle on olennaista tietää, koska sovellus työskentelee, koska se on valmis ottamaan vastaan uusia komentoja ja niin edelleen. Näillä ehkäistään kuvitelmat siitä, että sovellus on pysähtynyt, tai että se ei ottanut komentoa vastaan. Näitä toimenpiteitä joudutaan tekemään, koska AJAX-tekniikassa ei ole varsinaista virheenhallintaa ollenkaan, joten ne joudutaan tekemään itse. (Purewal 2014: 256)

Käyttöliittymän vuorovaikutteisuus on myös suuressa osassa silloin, kun päätelaitteen suorituskyky tai internet-nopeus eivät ole parhaimmillaan. Yksinkertainen ohjelmointi mahdollistaa pienten datamäärien tehokkaan siirtämisen, joten päätelaitteen tehokkuus ei ole suuressa osassa sovellusta käytettäessä. Painikkeiden korostuksia käytetään sen takia, että käyttäjä saa helposti selville, mitä tällä hetkellä voi muokata.

9.4 Sovelluksen testaus

Tähänastisista testaustuloksista ovat olleet yllättävimpiä vanhemmalla ikäpolvella tehdyt testaamiset. Vanhempi ikäpolvi, tässä tapauksessa yli 50-vuotiaat henkilöt ovat tottuneet käyttämään näppäimistöä ja hiirtä eri tavoin ja sen myötä sovelluksessani on esiintynyt puutteita.

Esimerkkinä voidaan pitää sovelluksen alkuvaiheessa olemassa ollutta tapaa luoda uusi sivu. Sivun luonti oli liitettyä hakupalkin elementtiin, ondblClick-toimintoon, eli elementin tuplaklikkaus-ominaisuuteen. Kuvittelin tuolloin, että jos kirjoittaa sivun nimen hakupalkkiin ja tuplaklikkaa hakupalkkia, saadaan uusi sivu annetulla otsikolla. Tämä osoittautui huonoksi tavaksi luoda uutta sivua.

Vanhempi ikäpolvi on tottunut tuplaklikkaamaan lähes kaikkea, joten heti alkuunsa ongelmaksi tuli se, että ensinnäkin luotiin uusi sivu, jolla ei ollut nimeä ja koska olin myös unohtanut laittaa otsikko-elementille vähimmäiskorkeuden, otsikko-

elementtiä ei siis näkynyt ollenkaan. Toki siihen pääsisi käsiksi vaikkapa tab-painikkeella, mutta normaalille käyttäjälle moinen voisi olla liian haasteellista.

9.5 Mobiilikäyttöliittymän luominen

Mobiilikäyttöliittymän luominen on hyvä keino vastata nykypäivän päätelaitekantaan, joka on erittäin laaja. Cisco ennustaa, että vuoden 2014 loppuun mennessä mobiiliyhteyteen kykeneviä laitteita on yli 7 miljardia (Cisco 2014.). Tämän vuoksi on tärkeää, että web-sovelluksesta on oma versionsa, joka on optimoitu pienille kosketusnäyttöisille laitteille.

Mobiilikäyttöliittymässä on pelkistetty ulkoasua entisestään ja vain olennaisimmat toiminnot ovat sijoitettu aloitusnäyttöön. Mobiililaitteen tunnistus on tehty automaattiseksi, koska olen kokenut sivulle tullessa pop-up-ikkunakyselyn liian epäkäytännölliseksi varsinkin sellaisille käyttäjille, jotka tyhjentävät selaimen välimuistin aina istunnon päätteeksi. Mobiilikäyttäjä siis ohjataan suoraan mobiilisivustolle, josta on mahdollisuus siirtyä työpöytä-versioon, mikäli käyttäjä itse haluaa.

10 SOVELLUKSEN RAKENTEIDEN SELKEYTTÄMINEN JA YHTENÄISTÄMINEN

Sovelluskehityksen alkuvaiheessa loin jokaiselle palvelimella tehtävälle toiminnolle oman PHP-tiedostonsa, eli jokaisesta otettiin erikseen tietokantayhteys ja jokaisessa oli oma koodinsa, joten PHP-tiedostoja alkoi olla melkoisesti (etsi.php, luosivu.php yms). Myöhemmin tutkin keinoja, joilla saisin keskitettyä kaikki muistiinpanojen selaamiseen ja muokkaamiseen liittyvät toiminnot samaan PHP-tiedostoon. Suurimmat edut koin tästä järjestelmästä siten, että tietokantayhteyksiä ei tarvitse avata kuin yksi, jonka jälkeen suoritetaan haluttu toiminto ja lopuksi tietokanta suljetaan.

Loin PHP-tiedoston, johon laitoin kaikki funktiot peräkkäin ja Javascript pääsee niihin käsiksi Javascriptin puolelta tulevalla komentonumerolla (qkomento). Esimerkiksi jos qkomento on 2, komento tarkoittaa artikkelin nimen hakemiseen tarkoitettua funktiota.

Pyydetty funktio päätellään PHP:n puolella ehtolauseella.

```
60 else if($qkomento==2){
61 $q
62 $sql="SELECT * FROM Artikkelit WHERE ID = '".$q."'";
63 $result = mysql_query($sql);
64 while ($row = mysql_fetch_array($result)){echo $row['Nimi'];}}
```

Edellä olevassa koodissa on demonstraatio siitä, miten PHP päättelee, mikä funktio on tarkoitus toteuttaa. Javascriptiltä lähtee komento $qkomento=2, jonka ehtolauseella määritellään tarkoittavan otsikon hakemista ID:n perusteella.

Tämän tavan kehitin itse, pääasiassa vastaamaan siihen ongelmaan, että eri PHP-funktiot olisivat eri tiedostoissa, joka vaati suuren määrän tiedostoja. Näin voi-

daan keskittää funktioita ja jatkokehitys on helpompaa ja kätevämpää, koska tarvitse siirtyä tiedostosta toiseen, jos kaikkiin pitää tehdä muutos.

11 HAASTEET

Yksi olennaisista haasteista, varsinkin kun verrataan web-sovelluksen ja työpöytäsovelluksen kehittämistä on se, että web-sovellusta voi käyttää usealla eri selaimella, samoin kuin usealla eri selainversiolla. Tämän vuoksi selainyhteensopivuus on ollut suuri asia sovellusta suunnitellessa ja toteuttaessa.

Olen testannut sovellusta viidellä yleisimmällä selaimella, eli Internet Explorerilla, Mozillan Firefoxilla, Googlen Chromella, Operalla sekä Applen Safarilla. Näiden lisäksi olen testannut sitä myös Googlen Android-pohjaisten laitteiden natiiviselaimilla, joiden palaute on myös tärkeää. Selainten välillä on pääasiassa pieniä ulkoasullisia eroja, mutta myös suurempia eroja, esimerkiksi HTML-symbolien näyttämisen osalta selainten välillä on eroja, osa selaimista näyttää tietyt symbolit, osa taas ei.

Kuten olen aiemmin maininnut, ulkoasun muokkaaminen on ollut yksi haasteellisimmista tehtävistä. Se ei ole haasteellinen sen vuoksi, että se olisi vaikeaa ohjelmoida, vaan kyseessä on sellaisen ulkoasun luomisesta, mikä on siisti ja toiminnallinen ja ennen kaikkea siihen olisi itse tyytyväinen.

Sovelluksen ulkoasun muokkaaminen on jarruttanut ehdottomasti koko sovelluksen kehittämistä, koska Javascript-tiedostoa, samoin kuin CSS-tiedostoa joutuu muokkaamaan paljon, mikäli suurempia muutoksia aikoo tehdä. Osasyyllisenä sovelluksen kehityksen hidastumiseen voidaan pitää myös uusien ominaisuuksien kehittämistä ja niiden sovittamista nykyiseen koodirakenteeseen.

Yleensä uusien ominaisuuksien sovittamisen mielekkyys osoittautui suuremmaksi, kuin olemassa olevien ominaisuuksien loppuun hiominen. Tästä johtuen sovelluksen kehitys on ollut ripeää, mutta samasta syystä sovelluksen vakaudesta ei voida puhua samassa lauseessa.

Ennen kun aloitin opintoni, minulla ei ollut minkäänlaista kokemusta tässä työssä käytetyistä kielistä tai niiden käyttämisestä. HTML-kielen kanssa olen aiemmin

tehnyt yksinkertaisia sivuja, mutta kaikkeen muuhun sain oppini koulun kurssien aikana.

Suurena kehityksen nopeuttajana on tosin ollut se, että minulla on ollut ideoita, joista yksi on tämä muistiinpanosovellus, joka on johtanut siihen, että olen vapaa-ajallanikin perehtynyt kaikkiin mainitsemiini kieliin tarkemmin. Koulun kurssien aikana kieliä opiskellessani ja muistiinpanoja tehdessäni aloin havainnoida tapoja ja rakenteita, miten sovellukset oli rakennettu. Samalla koin myös kehitystarpeita silloin käyttämieni sovelluksien osalta.

Näistä syistä johtuen aloin rakentaa omaa muistiinpanosovellustani, joka olisi viimeisimmillä web-tekniikoilla toteutettu ja räätälöity omiin tarpeisiini. Sovelluksen kehittyessä mietin kyllä useaan otteeseen, pitäisikö siitä tehdä sellainen, jotta kaikki, tai ainakin tuttavapiirini voisi sitä käyttää ja jo pelkästään tämän asian miettiminen loi kehitystarpeita sivuston eri elementeille.

Kysyin tuttaviltani asiasta ja suurin osa oli innostunut ideastani ja sain samalla myös uusia kehitysehdotuksia. Aloin jalostaa ja sovittaa noita ehdotuksia sovellukseeni ja kun otetaan huomioon vielä ulkoasun jatkuvan kehittyvä luonne, sovelluksen valmistumisen aikataulun saikin venyttää kauas tulevaisuuteen.

12 VIRHEIDEN ETSINTÄ

Jokaiselle web-kehittäjälle tulee todennäköisesti vastaan tilanteita, missä "juuri mitään" ei ole muutettu, mutta jostakin syystä sovellus ei vain toimi. Tai koodia on muutettu siten, että "sen pitäisi teoriassa toimia". Näissä tilanteissa on tärkeää olla muutama hyvä keino löytää nuo viat ilman, että koodia tarvitsisi palauttaa alkutilaan.

Tämän sovelluksen kehittämisessä on 7 kohdan ketju (HTML-Javascript-PHP-MySQL-PHP-Javascript-HTML) ja mikäli yhdenkin lenkin välillä tieto ei kulje, loppulenkki ei toimi kuten pitää ja sama koskee todennäköisesti koko sovellusta. Pahimmillaan vianetsintä on silloin, kun Javascript-tiedoston uumenissa on jokin pielessä siten, että "se ei vaan toimi". Javascript on muista kielistä poiketen tässä tapauksessa hankala vikaselvitettävä.

HTML näyttää yleensä pelkästään jonkin pätkän rakennetta sivulla, mikäli siellä on kirjoitusvirhe, CSS jättää sen elementin (mahdollisesti myös loppumääritelmät) muotoilematta missä virhe on, PHP:lle on helppo asettaa virheilmoituksia ja sillä on myös omia virheilmoituksiaan ja MySQL:lle on myös helppo asettaa vikailmoituksia tai testata annettua kyselyä vaikkapa PHPMyAdminissa.

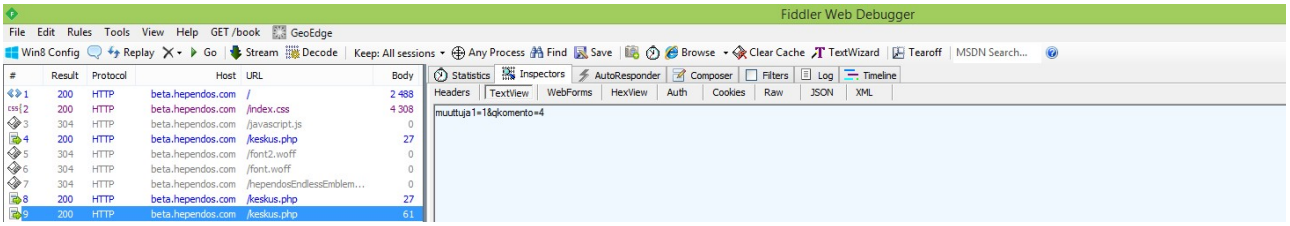
Javascriptin osalta olen käyttänyt eliminointitekniikkaa vikaa selvittäessäni ja siitä on seuraavaksi esimerkki.

```
function PoistaSivu(){
alert("Virhe1");
LataaGif();
xmlhttp=new XMLHttpRequest();
xmlhttp.onreadystatechange=function(){
if (xmlhttp.readyState==4 && xmlhttp.status==200){
document.getElementById("Info").innerHTML="  &#10003; Sivu poistettu. Siirryt&auml;&auml;n etusivulle.";
alert("Virhe3");
NaytaInfo("ilmoitus");
HaeOtsikko(1);
SuljeTyokalut();
document.getElementById("Poisto").style.visibility="hidden";
EiLataaGif();}}
alert("Virhe2");
xmlhttp.open("POST","keskus.php", true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("muuttuja1="+NykyinenID+"&qkomento="+8);
}
```

Tällä pyritään selvittämään, mihin kohtaan pyyntö pysähtyy, olettaen että ylipäätään menee Javascriptille asti. Tämänkin voi selvittää laittamalla Javascript-tiedoston alkuun globaalin funktion *alert("Testi");*, jolloin saadaan selville sivua ladattaessa, toimiiko Javascript-tiedosto ylipäätään. Tämän kaltaisia väli-ilmoituksia olen laittanut joka tiedostoon ja mikäli niitä tarvitaan, poistan niiden ympäriltä kommenttimerkinnät ja alan selvittämään vikaa.

Näiden lisäksi käytän Fiddler Web Debugger-nimistä ohjelmaa, joka tarkkailee asiakas-palvelin välistä liikennettä ja sen avulla näkee todella hyvin, mitä palvelimelle meni ja mitä tulee takaisin.

Seuraavassa kuvassa näkyy kaikki toiminta, kun www.beta.hependos.com ladataan.



Kuva 12. Fiddler-ohjelman näkymä

Vasemmalla olevassa palkissa näkyy ensinnäkin dokumentin juuri, mistä sivu ladataan, eli beta.hependos.com.

Tämän jälkeen järjestyksessä:

1. index.css, eli tyylitiedosto ladataan ja HTML-dokumentti muotoillaan sen perusteella.
2. javascript.js, eli javascript-tiedosto, jolla suoritetaan kaikki Ajax-kutsut PHP-tiedostolle.
3. keskus.php, ensimmäinen kutsu PHP-tiedostolle, haetaan etusivu.
4. Ladataan ensimmäinen fontti.
5. Ladataan toinen fontti.
6. Ladataan logo, joka näkyy vasemmalla ylhäällä.
7. keskus.php, ladataan etusivun aikatiedot.
8. keskus.php, ladataan etusivun sisältö.

Fiddlerillä näkee kaiken, mitä asiakkaan ja palvelimen välillä liikkuu, joten varsinkin vianetsintä helpottuu, kun näkee mihin se tieto jää. Tällä ei näe skriptien ajoa, mutta sitä varten esimerkiksi Mozilla Firefoxin Firebug-lisäosa on erittäin hyvä vaihtoehto, jolla näkee myös mm. skriptien viemät ajat.

Fiddlerillä helposti myös sen, jos esimerkiksi jokin komento ajetaan tiedostolle kaksi kertaa. Myös ennen kaikkea turvallisuusaukot nähdään olennaisesti, koska mikäli asiakkaan ja palvelimen välillä olisi kolmas osapuoli Fiddlerin kaltaisella ohjelmistolla, se osapuoli näkee kaiken, mitä Fiddlerilläkin näkee.

13 JATKOKEHITYS

Aiemmin käyttämäni muistiinpanosovellus Dokuwiki toimi samalla myös tietynlaisena portaalina perheen kesken. Sinne voi kirjata esimerkiksi joululahjatoiveita ja vastaavia ja kaikilla on sinne omat käyttäjätunnuksensa. Muun muassa tästä syystä johtuen tämän sovelluksen kehitys ei tule jäämään muistiinpanosovellukseksi, vaan aion myös tästä sovelluksesta luoda omanlaisensa portaalin, missä voi mainittujen kaltaisia asioita jakaa.

Koko sovelluksen kehitys on tarjonnut niin paljon kokemusta web-sovelluksen rakentamisesta, että olen alkanut listaamaan erilaisia "moduuleja", joita aion sovellukseeni kehittää. Näitä ovat esimerkiksi DVD-tietokanta, auton polttoaineseurantaan tarkoitettu sovellus, johon syötetään tietoja joka tankkauksen yhteydessä ja vastaavan kaltaisia ominaisuuksia.

Olen listannut näitä paljon ja niitä aion myös toteuttaa uuden sovellukseni yhteyteen. Itse sovelluksen kehittämistä olen ajatellut, kun olen luonut ns. uudelleenkäytettäviä funktioita mm. Javascriptiin.

Yksinkertaisena esimerkkinä voi mainita esimerkiksi painikkeen onmouseover-attribuutin muokkaamisen Javascriptillä elementin ID:n perusteella, eikä erillisen funktion perusteella, eli Javascript-funktio ottaa vastaan elementin ID:n ja vaihtaa väriä sen perusteella. Tämänkaltaisilla toimilla saadaan aikaan sovellus, jota on myös jatkossa helppo kehittää eteenpäin.

14 POHDINTA

Verkostoituva maailma on siirtymässä tilaan, jossa käytännössä kaikkeen, mitä kotikoneella on, pitää päästä käsiksi myös muualta.Tämä herättää kysymyksiä luonnollisesti varsinkin tietoturvasta. Web-sovellusten osalta nämä kysymykset ovat suuria ja ajankohtaisia, koska toistaiseksi mobiilimaailmassa käytetään jokaiselle käyttöjärjestelmälle räätälöityjä sovelluksia. Web-sovelluksen muuttavat tuota ajatusmallia sen verran, että ne eivät ehkä olekaan enää niin laitekohtaisia. Kaikilla laitteilla on teoreettinen mahdollisuus päästä käsiksi kaikkiin samoihin tietoihin, kuin sillä ainoalla laitteella, jolla sovellusta on tarkoitus käyttää.

Tietoturva on ollut myös asia, josta olen ollut pitkään kiinnostunut, mutta en ole saanut mahdollisuutta tutustua siihen ns. 'backendin' puolelta. Kuten aiemmin mainitsin, koulu tarjosi oivan tilaisuuden tutustua kaikkiin näihin kieliin, joita tässä opinnäytetyössä käytettiin ja otin siitä kaiken irti. Varsinaista tietoturvaan kohdistettua kurssia en ole käynytkään, mutta tätäkin sovellusta kehittäessä tuli ilmi useita eri tapoja ja keinoja, joilla tällaiseenkin järjestelmään pääsisi luvattomasti kirjautumaan.

Tällaiset asiat ovat luonnollisesti erittäin kriittisiä sovelluksen mahdollisille käyttäjille, joista itse olen yksi, joten perehdyin tietoturvaan eri sivustoilta löytyvien oppaiden pohjalta ja etsin parhaita keinoja murtautua sivustoille, jotta voin niitä vastaan sitten kehittää keinot. Näitä vastaan olen järjestelmääni kehittänyt, mutta varsinkin web-kehittäjille kovaksi vastukseksi on ilmaantunut ns. *bottiverkko* eli kaapattujen tietokoneiden suma, jonka avulla voidaan suorittaa suuria määriä esim. kirjautumisyrityksiä sivustolle. Uhkakuvia on paljon, mutta mielikuvitusta käyttämällä ja koodia yksinkertaistamalla päästään suurimmasta osasta eroon.

Tämän työn valmistuessa sovelluksesta on sellainen versio lähes valmis, jonka voisi periaatteessa vaikkapa kaupallistaa. En todennäköisesti kuitenkaan sitä tee, koska tämän työn lähtökohdat olivat ongelmapohjaiset ja periaatteessa tämä sovellus on räätälöity nimenomaan omaan käyttööni, vaikka tuttavat ja perhe kertovat-

kin sen olevan "ihan normaalin oloinen" sovellus. Jatkossa voi harkita tietysti uudelleen näitäkin asioita.

Web-ohjelmoinnin tulevaisuus tulee pyörimään laajalti web-järjestelmien luonnissa ja ylläpidossa, joihin on oma mobiilikäyttöliittymänsä, jotta kaikilla mobiililaitteilla pääsee siihen myös käsiksi. Juuri näitä työtoimenkuvia ajatellen, tämän opinnäytetyön tekeminen on ollut erittäin ajankohtaista ja olennaista.

Olen tyytyväinen lopputulokseen.

LÄHTEET

Cisco Visual Networking Index 2014: Global Mobile Data Traffic Forecast Up-date,2013–2018. 2014. Cisco Systems Inc. Viitattu 24.7.2014. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

Ketola, R. 2007. Toimiva toimisto. Tammer- Paino Oy. Tampere. Työterveyslaitos.

Nolan, K. Värien käyttäminen tehokkaasti: Kuinka värit vaikuttavat käyttäjään. Viitattu 24.7.2014. http://office.microsoft.com/fi-fi/frontpage-help/varien-kayttaminen-tehokkaasti-kuinka-varit-vaikuttavat-kayttajaan-HA001042937.aspx

PHP Documentation Group. What is PHP? WWW-dokumentti. Viitattu 24.7.2014. http://php.net/manual/en/intro-whatis.php

Purewal S. 2014. Learning Web App Development. First Edition, Sebastopol California, O'Reilly Media Inc.

Shah, S. 2007. Web 2.0 Security : Defending Ajax, RIA, and SOA. 1st edition, Massachusetts, Charles River Media.

W3Schools 2014: Cascading Style Sheets. CSS. WWW-dokumentti. Viitattu 24.7.2014. http://www.w3schools.com/css/css_intro.asp

W3Schools 2014: HyperText Markup Language. HTML. WWW-dokumentti. Viitattu 24.7.2014. http://www.w3schools.com/html/html_intro.asp

Walker T., 2013. Why "Simple" Websites Are Scientifically Better. Viitattu 24.7.2014. http://conversionxl.com/why-simple-websites-are-scientifically-better/

World Wide Web Concortium 2014. W3C. WWW-dokumentti. Viitattu 24.7.2014. http://www.w3.org

World Wide Web Concortium 2014. Standards for Web Applications on Mobile: current state and roadmap. WWW-dokumentti. Viitattu 24.7.2014. http://www.w3.org/Mobile/mobile-web-app-state/