VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Aki Perttu

Muistiinpano-ohjelma Ajaxia käyttäen

Web-sovellus

Liiketalous
2014

VAASAN AMMATTIKORKEAKOULU
Tietojenkäsittely

TIIVISTELMÄ

Tekijä	Aki Perttu
Opinnäytetyön nimi	Muistiinpano-ohjelma Ajaxia käyttäen
Vuosi	2014
Kieli	suomi
Sivumäärä	54
Ohjaaja	Sirkka Hellman

Tämän opinnäytetyön tarkoituksena oli tutkia, kuinka saadaan luotua muistiinpano-ohjelma käyttäen Ajax-tekniikkaa.

Ajax-tekniikalla pystytään luomaan nopeita ja dynaamisia sivustoja, koska Ajaxin avulla tieto selaimen ja palvelimen välillä siirretään ilman web-sivun päivittämistä. Opinnäytetyön muistiinpano-ohjelman vaatimuksiin kuuluu tekstin tallennus, muistiinpanojen haku, uuden muistiinpanon luonti sekä muistiinpanon poistaminen.

Teoriaosuudessa käydään läpi erilaisia ohjelmantekoon tarvittavia ohjelmointikieliä.

Käytännön toteutuksena tein oman muistiinpano-ohjelman, jonne käyttäjä voi käydä tekemässä helposti ja nopeasti omat muistiinpanonsa. Ohjelmassa on myös käyttäjänhallinta. Käytännön toteutuksen osuudessa esittelen tavat, jolla olen saanut muistiinpano-ohjelman toimimaan, käyttäjän kirjautumisen hallinnan sekä tietokantaratkaisun. Käsittelen toteutuksen yhteydessä vastaan tulleita ongelmia ja itse kehittämiäni ratkaisuja. Opinnäytetyö on tehty puhtaalla koodilla eli mitään kielikirjastoja ei ole käytetty. Käytännön toteutuksessa on myös kiinnitetty huomiota ulkoasuun ja käytettävyyteen.

Muistiinpano-ohjelman luonti onnistui täydellisesti. Ajax-tekniikan avulla sain ohjelmasta nopean ja yksinkertaisen käyttää. Ulkoasullisesti ohjelma on yksinkertainen ja selkeä. Opinnäytetyössäni päädyin siihen lopputulokseen, että Ajax-tekniikkaa käyttämällä voidaan luoda dynaamisia, responsiivisia ja ennen kaikkea nopeita web-sovelluksia.

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Processing

ABSTRACT

Author	Aki Perttu
Title	Note Program using Ajax-technique.
Year	2014
Language	Finnish
Pages	54
Name of Supervisor	Sirkka Hellman

The aim of this thesis was to explore how to create a note program using Ajax-technique.

Ajax technology makes it possible to create speedily and dynamic web sites, because Ajax allows the data to be moved between the browser and the server without updating the web page itself. The features of the note program includes the ability to save your own notes, ability to search for your own notes, ability to create a new note and ability to delete your own notes.

The theoretical section of the thesis covers a variety of programming languages used in the making of the program.

As a practical implementation I made my own note program where a user can write his/her notes quickly and easily. The program also includes user management. In the practical implementation part of the thesis the methods which made the program to work were introduced. The user management and database solution was also explained. The problems I encountered during the process are discussed and self-developed solutions to the problems are provided. This thesis was made with pure code which means that no language libraries were used in the making of this thesis. I was also paid attention to the visual appearance and usability of the program.

Note program was a complete success. With Ajax, the program was fast and easy to use. The appearance of the program is simple and clear. In my thesis the conclusion was that with the use of Ajax-technology, it is possible to create dynamic, responsive and, above all, speed web applications.

SISÄLTÖ

1 JOHDANTO

1.1 Tavoitteet ja tutkimuskysymykset

Tämän opinnäytetyön tarkoituksena on tutkia, kuinka saadaan luotua muistiinpano-ohjelma käyttäen Ajax-tekniikkaa.

Tavoitteena on, että ohjelmalla pystytään tallentamaan tekstiä, luotuja muistiinpanoja voidaan hakea hakupalkilla, uusia muistiinpanoja voidaan luoda sekä omia muistiinpanoja voidaan poistaa.

Käyn opinnäytetyössä myös läpi ohjelman tekoon liittyviä tekniikoita ja niiden historiaa.

Käytännön osuuden ohjelman toteutan web-sovelluksena eli ohjelmaa käytetään selainpohjaisesti. Käytän ohjelman teossa uusimpia tekniikoita ja niiden uusimpia versioita. Työssä käytetyt kielet ovat HTML5, CSS3, JavaScript, PHP ja MySQL. Ohjelman tulee toimia vain uusimmilla selainversioilla, joten ohjelmaa ei ole tarkoitus käyttää vanhemmilla selaimilla, jotka eivät tule Ajax-tekniikkaa.

1.2 Raportin rakenne

Opinnäytetyön toisessa luvussa esittelen web-ohjelmointiin yleisimmin liittyviä tekniikoita. Käyn läpi HTML5- ja CSS3-kielet, joista kerron hieman niiden perusteista. Kerron molempien uusista ominaisuuksista, joita olen myös käyttänyt opinnäytetyössäni. Lisäksi kerron JavaScript-, PHP- ja MySQL-kielistä, jotka olivat pääroolissa tämän opinnäytetyön teossa ja luonnollisesti käyn myös Ajax-tekniikan perustiedot läpi.

Opinnäytetyön kolmannessa luvussa käyn lyhyesti läpi jo olemassa olevia selain- ja työpöytäpohjaisia muistiinpanoohjelmia ja kerron niiden ulkoasuista sekä ominaisuuksista.

Opinnäytetyön neljännessä luvussa käyn läpi käytännön toteutuksen eri vaiheet. Käytännön toteutuksen osuudessa esittelen tavat, joilla olen saanut muistiinpano-ohjelman toimimaan, käyttäjän kirjautumisen hallinnan sekä tietokantaratkaisun.

Esittelen varsinkin oman ratkaisuni Ajaxin käyttöön kyseisessä ohjelmassa. Käsittelen toteutuksen yhteydessä vastaan tulleita ongelmia ja itse kehittämiäni ratkaisuja.

Päätännössä kerron, millaisen ohjelman sain tehtyä opinnäytetyössäni ja kuinka hyvin ohjelman teko onnistui. Käsittelen käyttämieni toteutustapojen hyviä ja huonoja puolia. Lopuksi kerron mahdollisista jatkokehitysideoista, joita aion toteuttaa tulevaisuudessa ohjelmaan.

1.3 Opinnäytetyön toteutus

Opinnäytetyössäni tehty ohjelma on muistiinpanoohjelma, jota olen ideoinut ystäväni kanssa kesällä 2013 ja jonka toteutuksen olen tehnyt itse yksin opinnäytetyöhöni. Muistiinpanoohjelmalle ei ole virallista asettajaa, vaan halusin oppia eri ohjelmointitekniikoiden ja varsinkin Ajax-tekniikan käyttöä ohjelman teossa. Opinnäytetyö on tehty puhtaalla koodilla eli mitään kielikirjastoja ei ole käytetty kuten esim. jQuerya.

2 TEKNIIKAT

Tässä luvussa käsittelen ohjelman teossa käytettyjä tekniikoita. HTML5:n sekä CSS3:n yleistymisen myötä web-sovellusten luominen on muuttunut helpommaksi ja sisällöltään monipuolisemmaksi. HTML5-kielellä pystytään nykyään luomaan suoraan multimediaelementtejä, jolloin paljon käytettyä Flashia ei tulevaisuudessa enää tarvita. CSS3 tuo varsinkin animointiin paljon uusia ominaisuuksia sekä parantaa elementtien hallintaa.

2.1 HTML5

HTML5 on uusin standardi HTML-kielelle. Viimeisin versio HTML:stä, versio 4.01, ilmestyi vuonna 1999 ja HTML5 onkin muuttanut paljon HTML-standardia. HTML5 oli tarkoitettu korvaamaan HTML4 , XHMTL ja HTML:n tason 2 DOM. Se suunniteltiin erityisesti tuomaan monipuolisen sisällön käyttäjälle ilman ulkoisten lisäosien tarvetta, kuten Flashin. Nykyinen versio mahdollistaa animoinnin monipuolisen käytön, graafisen materiaalin luonnin, musiikin ja videon toistamisen sekä monimutkaisten web-sovellusten luonnin. HTML5 mahdollistaa myös samalla yhteensopivuustuen monelle laitteelle mm. PC, tablet-tietokone, älypuhelin ja Smart TV. (W3Schools 2014.)

Uusia ominaisuuksia HTML5:ssä ovat mm. <canvas> elementti, joka mahdollistaa 2D-piirtämisen selaimessa, <video> ja <audio> elementit, jotka ovat median toistoa varten, tuki paikalliselle tallennustilalle eli localStorage, joka tulee korvaamaan evästeet tulevaisuudessa, uudet sisältökohtaiset elementit, kuten <article>, <footer>, <header>, <nav> ja <section> ja uudet lomakekontrollit, kuten calendar, date, time, email, url ja search. (W3Schools 2014.)

Kaikki yleisimmät selaimet (Chrome, Firefox, Internet Explorer, Safari, Opera) tukevat uusia HTML5-elementtejä ja ohjelmointirajapintoja (API) ja jotka jatkuvasti päivittävät uusia HTML5-ominaisuuksia uusimpiin selainversioihinsa.

Taulukko 1. HTML5-tuki eri elementeille (Caniuse 2014)

Elementti /Selain	Internet Explorer	Google Chrome	Mozilla Firefox	Opera	Safari	Android	iOS
<canvas>	9.0+	4.0+	15+	12+	5.1+	1.5+	3.2+
<video>	9.0+	4.0+	3.5+	10.5+	4.0+	2.1+	3.2+
<audio>	9.0+	4.0+	3.5+	10.5+	4.0+	2.3+	4.0+
Lomakeomin.	10.0+	4.0+	4.0+	9.0+	4.0+	4.4+	4.0+
localStorage	8.0+	4.0+	3.5+	10.6+	5.1+	2.1+	3.2+
WebGL	11.0+	8.0+	4.0+	15+	5.1+	-	-

Kuten taulukosta 1 nähdään HTML5-elementit ovat nykyään erittäin hyvin tuettuja jokaisella selaimella. Ainoastaan WebGL eli 3D-animointi JavaScriptiä käyttäen ei ole vielä yleistynyt. Älypuhelimissa HTML5-tuki on parantunut vuosi vuodelta ja eri käyttöliittymät ovatkin jo ottaneet uusimpia HTML5-tekniikoita uusimpiin versioihinsa.

2.2 CSS3

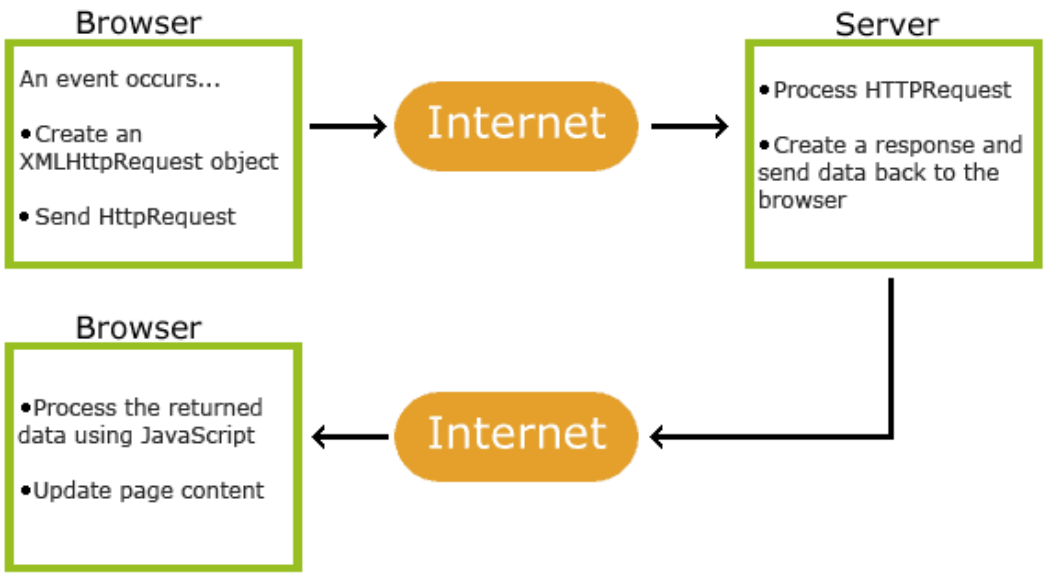
Cascading Style Sheetsin eli CSS:n avulla web-ohjelmoija voi muokata web-sivujen tyylejä, kuten värejä ja fonttien kokoja, marginaaleja ja erilaisten sisältöjen kokoa ja muotoilua. CSS:llä määritellään säännöt, jotka määräävät miltä elementtien sisällön tulisi näyttää sivustolla. Esimerkiksi voidaan tarkentaa, että taustan väri on vaaleanruskea, sisällön kaikkien <p>-elementtien tulee olla harmaita ja käyttää Arial-fonttia ja että kaikki <h1>-elementit tulee olla punaisia ja käyttää Times New Roman-fonttia. (Duckett 2010, 243-244.)

CSS3 tuo monta uutta ominaisuutta elementtien muokkaukseen. Näitä uusia ominaisuuksia on mm. animaatiot, border-radius, box-shadow, gradientit ja web-fontit. Animaatiot korvaavat Flash-animaatioiden ja animoitujen kuvien käytön web-sivuilla mutta ominaisuutta haittaa vielä se, että animaatio pitää määrittää ainakin kahdella eri tavalla, että ominaisuus toimisi kaikissa selaimissa. Border-radius ja box-shadow tuovat sisältöelementteihin pyöristetyt kulmat sekä

varjostukset. Gradienttien avulla pystytään siirtymään pehmeästi kahden eri taustavärin välillä joko lineaarisesti tai radiaalisesti. Web-fonttien avulla ohjelmoijat pystyvät käyttämään itseluotuja fontteja sivustoillaan eli ohjelmoijien ei tarvitse enää käyttää ns. ”web-safe”-fontteja, jotka toimivat kaikilla selaimilla. Web-fontit ladataan palvelimelle, josta se otetaan automaattisesti käyttöön, kun käyttäjä tulee sivustolle. Web-fontteja saa ladattua ilmaiseksi tai maksullisesti eri palveluista. (W3Schools 2014.)

2.3 Ajax

Asynchronous JavaScript and XML lyhennettynä Ajax on termi, jolla viitataan monen eri tekniikan kokoelmaan. Nämä tekniikat antavat web-ohjelmoijalle mahdollisuuden luoda sovelluksia, jotka vuorovaikuttavat dynaamisesti käyttäjien kanssa ja joka samalla kulissien takana hakee palvelinpuolella sovelluksen dataa. Tämä data pystytään näyttämään käyttäjälle selaimessa ilman sivun päivitystä, joka nopeuttaa sovelluksen käyttämistä. Lopputuloksena saadaan web-sovellus, joka näyttää ja tuntuu työpöytäsovellukselta. (Ford 2008, 5-6.)



Kuvio 1. Ajax:in toiminta kaaviona (W3Schools 2014)

Kuviosta 1 nähdään Ajaxin toimintaperiaate kaaviona. Kun tapahtuma eli event esim. napin painallus tapahtuu web-sivulla, niin luodaan XMLHttpRequest-objekti, joka lähettää tiedon palvelimelle. Palvelimella HTTPRequest tieto käsitellään esim. PHP-koodin mukaisesti, jonka jälkeen palvelin lähettää vastauksen takaisin selaimelle. Palvelimelta tuleva tieto käsitellään JavaScriptissä, jossa vastaus sijoitetaan haluttuun paikkaan ja samalla päivitetään sivun sisältö, jolloin käyttäjä huomaa muutoksen. Tämä kaikki tapahtuu niin, että web-sivua ei tarvitse päivittää uudelleen vaan Ajax toimii taustalla jatkuvasti.

Ajax-sovelluksen ovat paljon nopeampia ja responsiivisempia kuin perinteiset web-sovellukset. Ajax:in avulla dataa siirretään selaimen ja palvelimen välillä asynkronoidusti kulissien takana. Tämä tarkoittaa sitä, että Ajax-sovellukset voivat lähettää pyyntöjä palvelimelle ilman sovelluksen suorituksen lopettamista ja samalla Ajax prosessoi pyydetyn datan heti, kun vastaus tulee palvelimelta. Ajax lähettää pyynnön perinteisen lomakkeen sijasta palvelimelle käyttämällä erityistä selainobjektia nimeltä XMLHttpRequest. Tämä on Ajax:in pääkomponentti, joka mahdollistaa asynkronoidun yhteyden. (Ford 2008, 5-6.)

Ajax koostuu joukosta eri tekniikoita, jotka ovat olleet jo käytössä vuosia. Web-ohjelmoijat huomasivat, että kun näitä tekniikoita käytetään yhdessä, niin voidaan luoda vankka ja tehokas ympäristö web-sovellusten tekemiselle. Tekniikat, joista Ajax koostuu, ovat JavaScript, XML, XMLHttpRequest-objekti, HTML, CSS ja DOM eli Document Object Model. JavaScript yhdistää kaikki muut tekniikat ja mahdollistaa vuorovaikutuksen tekniikoiden välillä. XML mahdollistaa jäsennellyn datan välityksen selaimen ja palvelimen välillä. XMLHttpRequest-objekti mahdollistaa datan välityksen asynkronoidusti selaimen ja palvelimen välillä. HTML:llä ja CSS:llä pystytään muokkaamaan ja tyylittämään sivun tekstin ulkonäköä. DOM:illa voidaan dynaamisesti vuorovaikuttaa ja muuttaa sivun ulkoasua ja sisältöä. (Ford 2008, 6.)

```
var xmlhttp;
if (window.XMLHttpRequest)
  {// tämä koodi toimii kaikilla nykyisillä selaimilla
  xmlhttp=new XMLHttpRequest();
  }
else
  {// koodi, jos selain on IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.open("POST","testi.txt",true);
xmlhttp.send();
```

Kuvio 2. Esimerkki XMLHttpRequestobjektin luonnista

Kuviosta 2 nähdään esimerkki XMLHttpRequestobjektin luonnista. Ensiksi luodaan xmlhttp-muuttuja, jolle luodaan uusi XMLHttpRequest-olio. Esimerkissä on myös varauduttu siihen, jos käyttäjä käyttää vielä IE5 tai IE6-selaimia, johon tarvitaan erilainen objekti nimeltä ActiveXObject. Tätä ei enää nykyajan sivuilla käytetä, koska kyseisten selainversioiden osuus on nykyään jo lähes olematon. Kun objekti on luotu, niin xmlhttp.open-kohdassa määritellään ensin lähetystapa eli POST tai GET, toisessa kohdassa kerrotaan kutsulle, mihin tiedostoon kutsu suunnataan ja kolmannessa kohdassa määritellään, onko Ajax-kutsu true vai false.

Jos valitaan false, niin Ajax-kutsu odottaa funktion suorituksen loppuun ennen kuin tulostaa vastauksen web-sivulle. Tällöin Ajax:in suoritus ei tapahdu taustalla jolloin suoritusnopeus kärsii. Falsea kannattaa kuitenkin käyttää, jos halutaan varmistaa kyseisen funktion loppuun ajo ennen kuin vastaus tulostetaan näytölle ja tätä voidaan käyttää esim. kirjautumisessa.

Ajax:illa on myös ongelmansa. Vanhoissa HTML4-selaimissa Ajax-kutsuja käsitellään usein väärin aiheuttaen toimintahäiriöitä, hitaalla internet-yhteyksellä Ajax:in toiminta muuttuu todella epävakaaksi, mikä aiheuttaa virheitä ja hakukoneet eivät löydä JavaScriptillä tehtyjä web-sovellusten sisältöjä, jonka vuoksi sisältö pitää antaa lisäksi muun kuin JavaScriptin kautta hakukoneille.

2.4 JavaScript

JavaScript on ohjelmointikieli, jolla voidaan luoda dynaamista toiminnallisuutta web-sivuille. JavaScriptillä voidaan sivuille suorittaa erilaisia tehtäviä, kuten luoda matemaattisia laskutoimituksia datalle, lomakkeen tietojen tarkistus ja tapahtumien(events) käsittely esim. käyttäjän klikatessa nappia. JavaScript nimestään riippumatta ei ole sama asia kuin Java-ohjelmointikieli, vaikkakin JavaScriptissä on vaikutteita Javasta. JavaScript on luokiteltu yhdeksi kolmesta ohjelmointikielestä, jotka web-ohjelmoijan tulee osata. (Duckett 2010, 480-482.)

Käsittelen JavaScriptin osalta DOM:ia, koska se on suuressa osassa käytännön osuudessa. DOM eli Document Object Model on osa JavaScriptin dynaamista puolta. DOM:issa sivu esitetään käyttämällä *document*-objektia, linkit esitetään *links*-objektilla, lomakeet esitellään *forms*-objektilla jne. DOM:illa voidaan siis hakea ja asettaa web-sivun ominaisuuksia, kuten taustavärin, kutsua metodeita, jotka suorittavat toimintoja, kuten lisäämällä uuden rivin sivulle ja reagoida tapahtumiin, kuten käyttäjän napin painallukseen. DOM:ia voidaan kuvitella rajapintana selaimen ja ohjelmointikielen välillä. Tätä voidaan verrata kaukosäätimeen, joka toimii rajapintana käyttäjän ja TV:n välissä. Käyttäjä tietää, että painamalla 1-nappia kaukosäätimestä vaihtaa kanavan 1-kanavalle tai, että äänenvoimakkuuden nappi lisää tai vähentää äänenvoimakkuutta. Samalla tavalla DOM on eräänlainen kaukosäädin, joka mahdollistaa JavaScriptin toimivuuden selaimessa ja itse sivustolla. (Duckett 2010, 483-484.)

```
<!DOCTYPE html>
<html>
<body>

<p id="intro">Hello World!</p>

<script>

txt=document.getElementById("intro").innerHTML;

document.write(txt);
</script>

</body>
</html>
```

Hello World!

Hello World!

Kuvio 3. Yksinkertainen esimerkki DOM:in käytöstä, joka tulostaa tekstin Hello World. (W3Schools 2014)

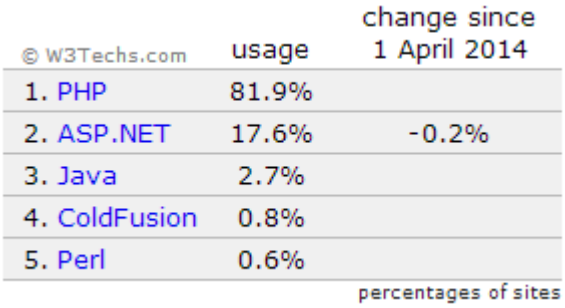
Kuviossa 3 nähdään yksinkertainen esimerkki DOM:in käytöstä. Aluksi HTML-koodiin luodaan paragrafi, jonka id:ksi annetaan intro ja arvoksi ”Hello World!”. Sen jälkeen luodaan txt-niminen muuttuja, johon haetaan paragrafin arvo eli innerHTML ”Hello World!” elementin id:n perusteella, joka oli tässä tapauksessa intro. Lopuksi txt-muuttuja tulostetaan document.writellä näytölle.

2.5 PHP

PHP on ensisijaisesti työväline dynaamisten web-dokumenttien luomiseen. PHP on html-dokumenttien sisään upotettava web-palvelimella tulkattava ohjelmointikieli. Tulkattavia ohjelmointikieliä kutsutaan yleisesti skriptikieliksi. PHP:n syntaksi on lainattu osaksi C-kielestä ja lisäksi mukana on piirteitä Java, Perl- ja C++-kielistä muutamine täysin omine PHP-lisäyksineen. PHP:n avulla voidaan käyttää mm. monia tietokantoja, kuten MySQL, PostgreSQL ja Oracle. PHP on avoin ohjelmisto ja sen lähdekoodi on vapaasti saatavilla. (Rantala 2005, 9.)

PHP:n etuna on se, että se on suunniteltu nimenomaan web-sovellusten rakentamiseen. PHP:n etuja ovat erityisen matala aloituskynnys, tehokkuus ja monipuolisuus, vapaasti käytettävissä kaikille käyttäjille, runsas käyttäjäkunta sekä olioohjelmoinnin uudistetut mekanismit. PHP on heikosti tyypitetty kieli eli

muuttujien tyyppiä ei tarvitse määritellä, vaan tyyppi määräytyy sen mukaan, millaista dataa muuttujaan sijoitetaan. PHP:n syntaksi on monipuolinen ja se toimii erityisen hyvin Apachen moduulina. PHP:tä tarjoavia web-hotelleja on tarjolla runsaasti ja dokumentaatiota ja apua löytyy runsaasti verkosta. (Rantala 2005, 10.)



© W3Techs.com	usage	change since 1 April 2014
1. PHP	81.9%	
2. ASP.NET	17.6%	-0.2%
3. Java	2.7%	
4. ColdFusion	0.8%	
5. Perl	0.6%	

percentages of sites

Kuvio 4. Käytetyimmät palvelinpuolen ohjelmointikielet (w3Techs 2014.)

Kuviosta 4 nähdään, että PHP on tällä hetkellä ylivoimaisesti käytetyin palvelinkieli. Tämä johtuu siitä, että PHP on täysin avoin ohjelmointikieli, jolla on erittäin runsas käyttäjäkunta ja se on saatavilla ilmaiseksi kenen tahansa käyttöön.

2.6 MySQL

SQL eli Structured Query Language on kieli, jota käytetään relaatiotietokantojen kyselyissä, päivityksessä ja hallinnassa. Sen avulla voidaan hakea, lajitella ja suodattaa tietokannasta poimittavia tietoja. SQL-kieli toimii itsenäisenä toteutusympäristössä mutta usein nämä lauseet kirjoitetaan osaksi ohjelmointikielellä kirjoitettua sovellusta, joka käyttää tietovarastona relaatiotietokantaa. (Päijät-Hämeen koulutuskeskus SQL-kieli, 2000.)

MySQL-tietokanta on yksi suosituimpia web-palveluiden tietokantoja. Se luetaankin neljän suurimman tietokannan mukaan. Suomalainen Monty Widenius ohjelmoi SQL-pohjaista tietokannan hallintajärjestelmää, jonka ensimmäiset versiot valmistuivat 1990-luvun puolivälissä nimellä MySQL. Nyttemmin

MySQL on ruotsalais-suomalaisen MySQL.com-yrityksen hallinnassa ja perusversion saa ladata ilmaiseksi yrityksen kotisivuilta. (Hovi 2012, 4.)

MySQL myytiin Sun Microsystemsille vuonna 2008 ja vuotta myöhemmin Oracle osti Sun Microsystemsin ja samalla MySQL:n. Oraclen ostettua MySQL:n, on raportoitu, että Oracle olisi pidättäytynyt virheiden korjaamisista ja huhujen mukaan Oracle olisi ajamassa MySQL:ää vähitellen alas. Monty Widenius onkin Oracle-kauppojen jälkeen luonut uuden MariaDB:n, joka perustuu MySQL 5.1-versioon. Huhtikuussa 2013 Wikipedia ilmoitti vaihtavansa englanninkielisen ja saksankielisen Wikipedian tietokannat MariaDB:hen. Google on myös ilmoittanut vaihtavansa tulevaisuudessa MariaDB:hen.

3 MUISTIINPANOSOVELLUSTEN NYKYTILANNE JA KEHITYS

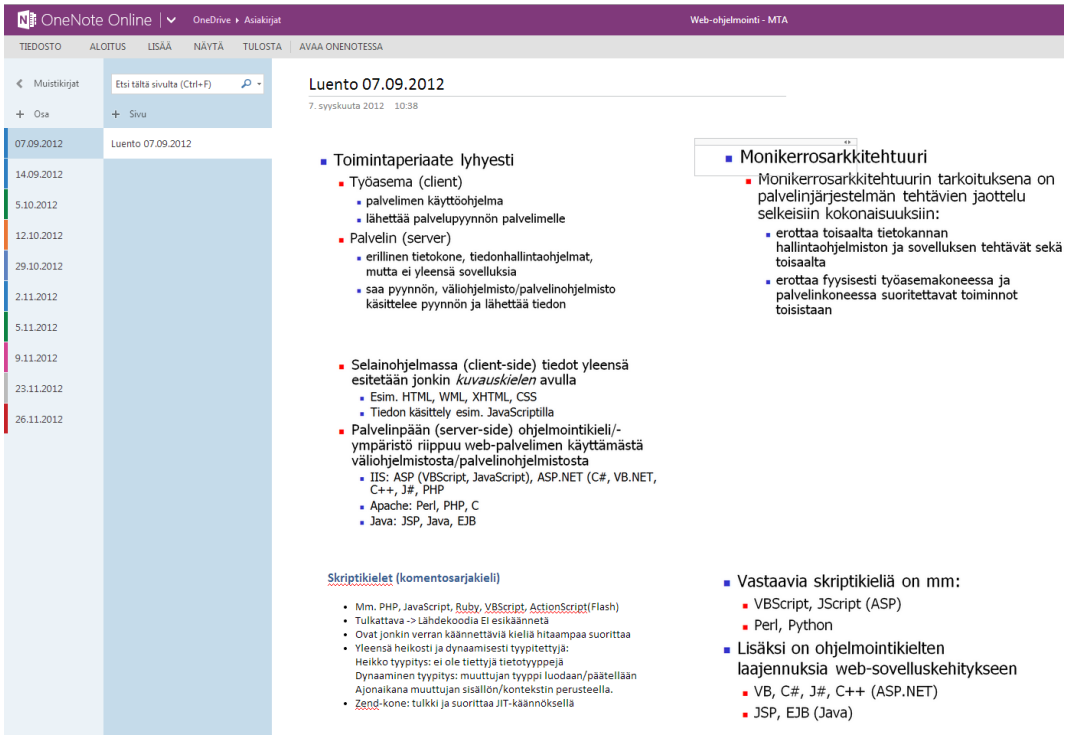
Ennen vanhaan muistiinpanojen tekoon tietokoneella oli mahdollisuus vain muistion kautta, kunnes Microsoft toi markkinoille Word-ohjelman, joka mullisti tekstinkäsittelyn ja muistiinpanojen luonnin. Mutta nykyään Word ei ole enää ns. muistiinpanosovellus, vaan sen käyttö perustuu ensisijaisesti pitkien esseiden yms. tekoon. Muistiinpanosovelluksella halutaankin saada tehtyä nopeasti ja vaivattovasti esim. kauppalista tai mieleen tullut idea talteen. Nykyään onkin jo olemassa paljon erilaisia muistiinpanosovelluksia juuri tähän tarkoitukseen.

3.1 Microsoft OneNote

OneNote on Microsoftin tekemä muistiinpanosovellus, joka toimii tietokoneella natiivisovelluksena, Office 365-pilvisovelluksena tai mobiilisovelluksena. Nykyinen versio on OneNote 2013.

OneNotella pystytään luomaan omia muistikirjoja, joihin tehdään omat muistiinpanot. Ohjelmaan voidaan tekstin lisäksi liittää esim. kuvia, videoita ja laskukaavoja. Paikallisen tallennuksen lisäksi OneNotessa on myös pilvitallennus. Tällöin OneNoteen kirjaudutaan omalla Windows Live-tilillä, jonka jälkeen muistiinpanot voidaan synkronoida OneDrive-pilvipalveluun tai SharePointiin.

Mobiiliversiossa on hieman rajoitetut ominaisuuden mutta tekstin kirjoitus ja kuvien lisääminen onnistuu myös mobiilisti. Selainversiota voidaan käyttää OneDriven kautta, jolloin se on OneNote Web App-nimellä. Selainversiossa voidaan lisätä myös mm. taulukoita ja Clipart-kuvia. (Microsoft 2014.)



Kuvio 5. OneNote Web App-esimerkki omista muistiinpanoista.

Kuviossa 5 näkyy muistikirjan luentomuistiinpanot. Vasemmalla palkissa näkyvät myös muut muistiinpanot samasta muistikirjasta. Ylhäällä tekstiruudussa on otsikkopalkki ja alapuolella itse tekstikenttä.

OneNote on mielestäni todella hyvä muistiinpanoohjelma, jossa on ominaisuuksia enemmän kuin tarpeeksi. Ohjelma on todella helppo käyttää ja kuvien ja yms. lisääminen on nopeaa ja vaivatonta. Käyttöliittymä on selkeä ja muistuttaa yläpalkiltaan muita Office-ohjelmia. Varsinkin web-sovelluksen ulkoasu on mielestäni yksinkertaisen loistava. Mielestäni hieno ominaisuus on OneNotessa se, että voit piirtää mihin tahansa muistiinpanossa ja alleviivata esim. kuvakaappauksista tiettyjä kohtia. Miinusta tuo kirjautumisen tarve joka kerta, jos haluat tallentaa tai hakea muistiinpanot OneDriveltä. Tämän lisäksi muistiinpanojen tuonti ja synkronointi kestää jopa luvattoman kauan. Tämä seikka

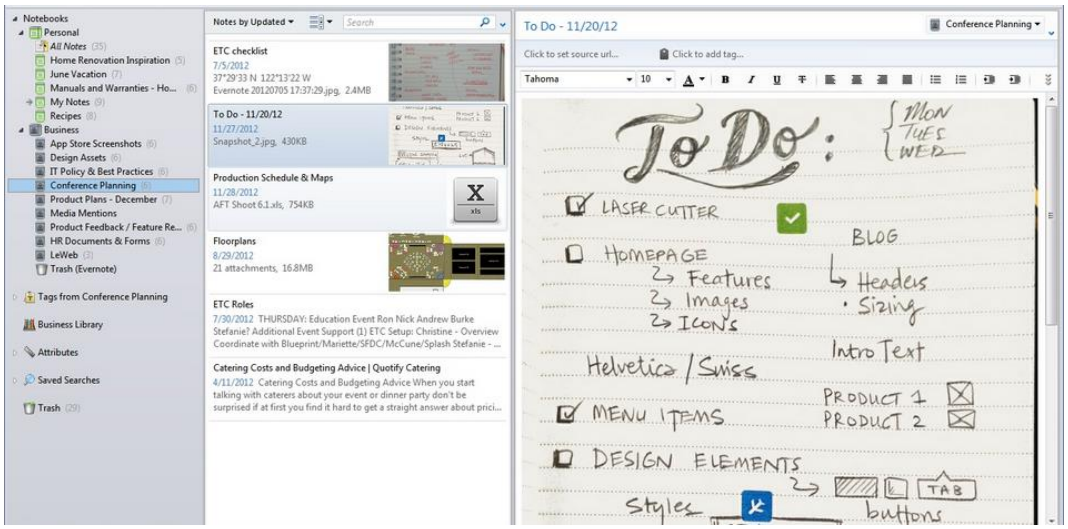
juuri hieman hidastaa sitä, jos haluaa nopeasti luoda muistiinpanon mutta toisaalta, jos ei käytä pilvipalvelua hyväkseen, niin tällöin muistiinpanon luonti onnistuu kyllä nopeasti paikallisesti tallennettuna. Käyttäjien harmiksi työpöytäohjelma on myös maksullinen.

Opinnäytetyössäni olen ottanut OneNotesta ideoita varsinkin tekstin tallennuksen kannalta. OneNotessa ei tarvita painaa tallennuspainiketta vaan ohjelma tallentaa automaattisesti jokaisen tekstin lisäyksen tai yms. toiminnon jälkeen. Olen tätä ideaa toteuttanut omassa ohjelmassani.

3.2 Evernote

Evernote on Evernote Corporationin tekemä muistiinpanosovellus, joka toimii OneNoten tapaan tietokoneella natiivisovelluksena, selainsovelluksena tai mobiilisovelluksena.

Evernotella voidaan kirjoittaa tekstiä, lisätä kuvia ja myös luoda suoraan ääni- ja videomuistiinpanoja. Evernotesta löytyy myös tekstintunnistus kuvia varten. Mobiilisovelluksella voidaan tehdä OneNotea enemmän eli mobiililla voidaan liittää äänitteitä muistiinpanoihin ja myös käyttää Google Mapsia. Mobiiliversiosta löytyy myös puheentunnistus. Selainversiosta löytyvät yleisimmät ominaisuudet mm. lihavointi, kursivointi, yliviivaus ja vaakaviiva. Chromeen ja Firefoxiin on ladattavissa Web Clipper-lisäosa, jolla voidaan lisätä verkkosivuja muistiinpanoiksi. Evernoten käyttö edellyttää rekisteröitymisen, jonka jälkeen ohjelma käyttö on ilmaista ”Free-tilillä” mutta jolla on rajoittuneet käyttäjäoikeudet ja 60 MB muistitilaa kuukaudessa. Premium-tilillä saadaan suurempi tallennustila eli jopa 1 GB kuukaudessa, offline-käytön sekä ryhmän keskeisten muistikirjojen luonnin. (Evernote 2014.)



Kuvio 6. Evernoten käyttöliittymä, johon liitetty erillinen kuva paperille tehdystä muistiinpanosta. (Evernote 2014.)

Kuviossa 6 nähdään Evernoten käyttöliittymä. Vasemmalla näkyvät omat muistikirjat ja niiden muistiinpanot. Oikealla näkyy itselisätty kuva, johon voi myös kirjoittaa päälle.

Evernoten ehdoton vahvuus on sen ilmaisessa käytössä ja hyvässä käytettävyydessä. Vuonna 2011 lopulla sillä olikin jo 20 miljoonaa käyttäjää ympäri maailman. Evernoten käyttöliittymä on mielestäni todella selkeä ja nopea käyttää. Evernote täyttääkin pikamuistiinpanon teon kriteerit, koska varsinkin mobiililaitteella uuden muistiinpanon luonti on nopeaa. Näissä seikoissa Evernote yllättäen viekin voiton OneNotelta. Lisäosien avulla Evernote nostaa itsensä OneNoten tasolle ja ohikin. Lisäosina on ladattavissa mm. käsinkirjoitukseen ja piirtämiseen tarkoitettu lisäosa, ihmisten tietojen ja kuvien lisäykseen tarkoitettu lisäosa ja Androidille ladattava lisäosa, jolla voidaan luoda pikalinkkejä aloitusnäytölle. (Evernote 2014.)

Muistiinpanosovellukset ovat menneet viime vuosina eteenpäin ja muistiinpanosovelluksien vaatimuksiin on tullut varsinkin muistiinpanojen nopea luonti sekä ryhmän yhteisten muistiinpanojen luonti ja jakaminen. Omassa ohjelmassani tulenkin toteuttamaan ainakin tekstin pikatallennuksen, julkisen artikkelin luonti-mahdollisuuden sekä selkeän käyttöliittymän.

4 MUISTIINPANOSOVELLUKSEN TOTEUTUS

Tein opiskelukaverini kanssa kesällä 2013 ensimmäisen prototyypin muistiinpanosovelluksesta. Prototyypissä oli toteutettu itse tekstin tallennus ja tekstin hakutoiminnot. Tekstin tallennuksessa kuitenkin havaittiin useita virheitä ja esimerkiksi tekstiä jäi joskus puuttumaan tallennuksen jälkeen. Itse ulkoasu jäi myös melko vajaaksi ja tässä opinnäytetyöversiossa se otetaankin paremmin huomioon. Sovelluksen koodi oli myös melko varmistamatonta ja sen huomasi prototyypin valmistuessa etenkin monena erilaisena virheenä sovelluksessa. Kokemukset Ajaxin käytöstä prototyypissä olivat kuitenkin äärimmäisen positiiviset ja se antoi kipinän tehdä aiheesta opinnäytetyö, joka on huolellisesti tehty muistiinpanosovellus ja ulkoasultaan ja käytettävyydeltään mahdollisimman yksinkertainen käyttää.

Ohjelman teossa päärooli saa Ajax-tekniikka, jota käytetään yhdessä HTML5, CSS3:n, JavaScriptin, MySQL:n ja PHP:n kanssa. Minua kiinnosti erityisesti Ajax-tekniikan käyttö, koska sitä oikein käyttämällä saadaan luotua erittäin dynaamisia sivustoja ja sen hallinnasta tulee olemaan työelämässä suurta hyötyä. HTML5:n ja CSS3:n avulla saan luotua ulkoasun ohjelmalle. JavaScriptin osalta tulen käyttämään varsinkin DOM-rakennetta, jolla sovelluksen toiminta Ajax:in kanssa saadaan toimimaan. PHP:n ja MySQL:n kautta saadaan luotua tietokantayhteys ja samalla tekstin tallennus.

Ohjelma luodaan fnote.sadcompany.net-nimiselle domainille ja se myös esitellään tätä kautta esityksissä. Ohjelma kirjoitetaan Notepad++-ohjelmaa käyttäen, tiedostoja hallitaan cPanelin File Managerin kautta ja tietokantoja hallitaan PHPMyAdminin kautta. Toteutuksen alussa kerron ulkoasun luomisesta, jonka jälkeen kerron itse toiminnallisuuden teosta sekä tietokantaratkaisusta.

4.1 Sovelluksen ensimmäinen versio

Selitän aluksi tarkemmin sovelluksessa käytettävät toiminnot. Käyttäjä pystyy heti sovellukseen tullessaan hakemaan muistiinpanojaan hakupalkin avulla. Haku-funktion avulla haetut hakutulokset tulostetaan tekstikenttään. Hakutulokset ovat linkkejä, joita painamalla siirrytään itse muistiinpanon kirjoittamiseen. Kun on

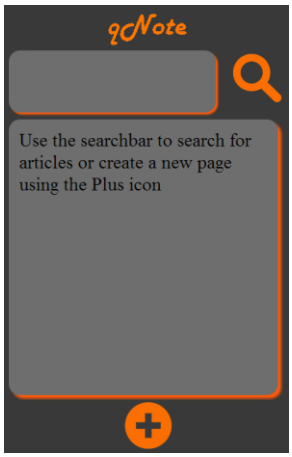
siirrytty linkin kautta muistiinpanoon, niin tällöin käyttäjä voi muuttaa otsikkoa tai tekstiä oman mielensä mukaan. Kun käyttäjä painaa haku-painiketta, niin käyttäjä voi jälleen hakea otsikkopalkin kautta omia muistiinpanojaan.

Aloitin ohjelman toteutuksen ja pääasiassa keskityin alussa pelkästään ulkoasun luomiseen. Ulkoasun oli tarkoitus olla samalla yksinkertainen ja käyttäjälle miellyttävän näköinen eli tehtävä ei ollut helppo. Aluksi ajattelin, että palkkien sekä nappien määrä saisi olla maksimissaan kaksi, koska tarkoitukseni oli saada ulkoasu mahdollisimman yksinkertaiseksi.

4.1.1 Ulkoasun luonti

Aloitin lisäämällä logon yläreunaan, jossa se pysyy kokoajan kiinteänä. Sen jälkeen lisäsin hakupalkki-divin, joka myös tulisi toimimaan otsikkokenttänä tekstiä kirjoittaessa. Otsikko/Hakupalkin jälkeen lisäsin itse tekstikenttä-divin, johon muistiinpanon teksti kirjoitetaan ja hakutulokset tulostetaan. Seuraavaksi sijoitin kaksi painiketta, jotka olivat haku-painike sekä uusi sivu-painike. Haku-painikkeen laitoin hakupalkin viereen oikealle puolelle, jotta haku olisi aina tavoiteltavissa helposti ja uusi sivu-painikkeen laitoin sivun alareunaan. Haku-painikkeen tehtävä on käynnistää hakutoiminto, jonka jälkeen voidaan hakea omaa artikkelia hakupalkkia käyttäen. Uusi sivu-painikkeella luodaan käyttäjälle uusi artikkeli. Painikkeille hain omat kuvakkeet, jotka havainnoivat hyvin niiden tarkoitustapaa.

Lopuksi valitsin ensimmäisen vedoksen värimaailman katsomalla esimerkkejä sivustoista, jotka olivat saaneet erityismainintoja hyvästä värimaailmastaan. Loppuvalinnaksi tuli tummanharmaa taustavärille ja palkkien taustaväriksi vaaleanharmaa. Nappien väriksi tuli oranssi. Lisäsin myös oranssin varjostuman palkeille sekä pyöristykset ulkonäön vuoksi. Tässä vaiheessa ajattelin, että ulkoasu oli riittävän yksinkertainen ja selkeä, jotta voin aloittaa jo toiminnallisuuden teon.



Kuvio 7. Ulkoasun ensimmäinen versio.

Kuviosta 7 huomataan, että ulkoasu on äärimmäisen yksinkertainen. Kaksi nappia oli alunperin suunnitelma, joka mielestäni onnistui hyvin tässä kohtaa. Ylempi palkki on hakupalkki ja alempi iso kenttä on tekstikenttä. Suurennuslasi-ikonilla haetaan tekstiä ja Plus-merkillä luodaan uusi sivu.

4.1.2 Toiminnallisuuden teko – Ajax

Toiminnallisuuden teko alkoi määrittelemällä, millä tapahtumalla eli eventillä funktion suorittaminen alkaa sovelluksessa. Koska halusin saada sovelluksesta mahdollisimman helppokäyttöisen ja yksinkertaisen, haku toimisi niin, että kun kirjoitetaan hakupalkkiin yksikin kirjain, niin sovellus automaattisesti hakee tietokannasta artikkeleita ja tulostaa tulokset ruudulle. Erillisen napin käyttäminen tässä kohtaa ei tullut kuuloonkaan eli päätin käyttää onkeydown-toimintoa jokaisella toimintapalkissa eli jokaisella näppäimistön lyönnillä suoritetaan funktio. Näin vältetään yksi napinpainallus vähemmän. Tekstin tallennus tulisi toimia samalla tavalla eli jokaisen näppäimistön painalluksen jälkeen tallennetaan teksti.

Periaate tallennuksen ja tekstin haun välillä oli, että hakutilanteessa moodi on 1 ja tallennusvaiheessa moodi on 2. Tällöin hakutulokset voidaan tulostaa tekstikenttään, kun moodi on haku eli 1 ja tallennusta ei tällöin tapahdu otsikkopalkissa eikä tekstipalkissa. Kun muistiinpano haetaan hakutuloksista, niin

moodi on tallennus eli 2 ja tällöin muistiinpanoon voidaan lisätä tai poistaa otsikkoon tai tekstiin tekstiä. Eri moodit siis mahdollistavat samojen palkkien käytön haussa että tekstin tallentamisessa. Kun sovellukseen siirrytään, niin moodi on aina aluksi haku eli 1.

Toiminnallisuuden tekeminen jatkui JavaScript-koodauksella, joiden avulla itse Ajax-kutsut tehdään PHP:lle. Koodin vähentämiseksi keksin käyttää Ajax:in XMLHttpRequestin luontia kiinteinä funktioina, jolloin sitä voidaan käyttää pelkällä funktion nimellä itse Ajax-yhteyden luonnissa. Loin XMLHttpRequest-funktiot tekstin tallennukselle, otsikon hakemiselle tietokannasta, tekstin hakemiselle tietokannasta ja hakutulosten tulostamiselle. XMLHttpRequestilla siis luodaan itse Ajax-kutsu, jolla myös voidaan määrittää, mihin vastausteksti eli responsetext halutaan sijoittaa jos halutaan.

```
function headerAjax()
{
        var xmlhttp=new XMLHttpRequest()
        xmlhttp.onreadystatechange=function(){
        if (xmlhttp.readyState==4){
        if (xmlhttp.status==200){
        document.getElementById("header").innerHTML=xmlhttp.responseText;
        }
        else{
        alert("AJAX request didnt succeed. Please refresh the webpage.");
        }
    }
}
        return xmlhttp;
}
```

Kuvio 8. Esimerkki XMLHttpRequest-funktion luonnista, joka palauttaa xmlhttp-muuttujan muille funktioille käytettäväksi. Tällä funktiota käytetään otsikon hakemissa tietokannasta.

Kuvion 8 esimerkissä nähdään headerAjax-funktio. Aina kun jossain JavaScript-funktiossa kutsutaan tätä XMLHttpRequest-funktiota, niin funktio sijoittaa palvelimen vastauksen eli responseTextin header-nimiseen diviin. Tämä funktio otetaan käyttöön muissa funktioissa seuraavasti: ”xmlhttp = headerAjax();”. Tällä samalla tavalla haetaan kaikki muutkin XMLHttpRequest-funktiot.

Seuraavaksi tein itse Ajax-funktiot tekstin tallennukselle, otsikon hakemiselle tietokannasta, tekstin hakemiselle tietokannasta, uuden sivun luonnille ja hakutulosten tulostamiselle. Kaikki funktiot suoritetaan aina onkeydown-toiminnolla, joten haku ja tekstin tallennus tapahtuu automaattisesti joka napin painalluksen jälkeen. Näin käyttäjän ei tarvitse koskaan painaa erikseen tallennus-nappia tallentaakseen muistiinpanonsa.

```
function search()
{
var searchname = encodeURIComponent(document.getElementById("search").value);
var xmlhttp = searchAjax();

xmlhttp.open("POST", "search.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("searchname="+searchinput);
}
```

Kuvio 9. Esimerkki Ajax-kutsusta PHP:lle, jolla haetaan hakutulokset tietokannasta.

Esimerkkikuviossa 9 on haku-funktio, jolla haetaan käyttäjän antamalla hakusyötteellä tietokannasta. Searchname-muuttujaan haetaan HTML-sivulta DOM:in avulla hakupalkin merkkijono. Sen jälkeen muuttuja se lähetetään POST-metodilla search.php:lle suoritettavaksi. Käytän POST:ia kaikissa Ajax-kutsuissa, koska se on paljon turvallisempi kuin GET-metodi vaikkakin GET-metodi saattaisi olla hieman nopeampi tälläisissä pienen datan lähetyksissä mutta turvallisuus on itselleni tässä tapauksessa tärkeämpää. Hakutulokset tulostetaan aina tekstikenttään.

JavaScript-koodissa vältin globaalien muuttujien käyttöä mutta lopulta jouduin käyttämään yhtä globaalia muuttujaa nimeltä tempid. Tähän muuttujaan sijoitetaan tietokannasta haettu id jokaiselle luodulle tai haetulle artikkelille. Tämän muuttujan avulla pystytään paikantamaan haluttu artikkeli tietokannasta. Tempid-muuttujaa käytetään siis jokaisessa sovelluksen toiminnossa, jotka olivat muistiinpanon hakeminen, muistiinpanon tulostaminen näytölle, muistiinpanon tekstin tallentaminen, uuden muistiinpanon luonti sekä muistiinpanon poistaminen.

```
var xmlhttp=new XMLHttpRequest()
        xmlhttp.onreadystatechange=function(){
        if (xmlhttp.readyState==4){
        if (xmlhttp.status==200){
        tempid=xmlhttp.responseText;
        getNewPage();
        }
        else{
        alert("AJAX request didn't complete. Please refresh the website.");
        }
    }
}

xmlhttp.open("POST", "addnewpage.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send();
```

Kuvio 10. Esimerkki uuden sivun luonti - funktiosta, joka käyttää hyväkseen globaalia muuttujaa tempid.

Esimerkkikuviossa 10 nähdään, että tempid-muuttujaan sijoitetaan addnewpage-php:ltä tullut vastaus, jota tässä tapauksessa käytetään uuden sivun luontiin, vastauksen sijoittamisen jälkeen suorittamalla getNewPage-funktio. GetNewPage-funktiossa haetaan addnewpage.php:llä luotu sivu tempid-muuttujaan sijoitetun id:n perusteella tietokannasta. Sivu sen jälkeen tulostetaan otsikkopalkkiin sekä tekstikenttään.

Kun sain luotua kaikki JavaScript-funktiot, niin aloin suunnitella PHP-koodeja, joilla saisin haettua tietokannasta halutut artikkelin sekä tallentaa kirjoitetut tekstit tietokantaan.

4.1.3 Toiminnallisuuden teko – Tietokanta

Ennen kuin lähdin tekemään itse PHP-puolta, loin ensin itse tietokannan, jonne luodut artikkelit tullaan lisäämään.

Tietokannan nimeksi tuli fNote, jonne tässä vaiheessa tuli vain yksi taulu, jonka nimi on articles. Riviä tauluun tuli neljä, jotka ovat id, header, text ja time. Id-rivi yksilöi jokaisen artikkelin, jolloin ristiriitoja muiden muistiinpanojen kanssa ei tule ja riville tuli tietotyyppi int(255), jonka pituus varmasti riittää tässä vaiheessa kapasiteetiksi. Id:llä toimii primary key:nä ja jolla on myös auto increment eli aina

kun uusi muistiinpano luodaan, niin id nousee yhdellä. Header-rivi sisältää muistiinpanon otsikon ja tietotyypiksi otin varchar (100), jonka 100 merkin pituus varmasti riittää otsikkotasolla. Text-rivi sisältää muistiinpanon tekstin ja tietotyypiksi valitsin text, joka ottaa 65535 merkkiä vastaan ja varmasti riittää ainakin näin aluksi. Time-rivi sisältää muistiinpanon luontipäivämäärän ja tietotyyppi on date, koska en halunnut kellonaikaa ainakaan vielä sisällyttää ohjelmaan.

1	**id**	int(255)		Ei	*None*	AUTO_INCREMENT
2	**header**	varchar(100)	latin1_swedish_ci	Kyllä	*NULL*	
3	**text**	text	latin1_swedish_ci	Kyllä	*NULL*	
4	**time**	date		Kyllä	*NULL*	

Kuvio 11. Ensimmäisen version articles-taulu

Kuviosta 11 nähdään articles-taulun rakenne. Ensimmäisen version aikana tämän taulun rakenne oli mielestäni tarpeeksi kattava ohjelman tekoon. Header- ja text-riveille annoin vielä kielikoodiksi latin1_swedish_ci, jolla saan tallennettua tietokantaan myös ä:n, ö:n ja å:n.

Loin muutaman rivin valmiiksi tietokantaan suoraan, jotta voin testata hakupalkin toimivuutta jo ennen uuden sivun luontia.

4.1.4 Toiminnallisuuden teko – PHP

Aloitin PHP-koodin luonnin hakutoiminnon PHP-tiedostolla. Jokaisen PHP-tiedoston alussa luodaan muuttujat ja niihin sijoitetaan JavaScriptiltä tulleet arvot ja tässä tapauksessa luodaan $search, johon sijoitetaan POST:illa lähetetty arvo eli $search = $_POST[”searchname”];.

Seuraavaksi luodaan yhteys tietokantaan mysqli_connect-komennolla. Yhteyden luonnin jälkeen lisäsin if-lauseen, joka tulostaa virheilmoituksen, jos yhteys ei onnistunut. Tarkistuksen jälkeen tein $sql-nimisen muuttujan, johon sijoitin itse SQL-kyselyn, jolla haetaan käyttäjän hakutekstin mukaan muistiinpanoja.

```
if($search == "*")
{
$sql = "SELECT * FROM articles ORDER BY time DESC";
}

else
{
$sql = "SELECT * FROM articles WHERE header LIKE '%$search%' ORDER BY time DESC LIMIT 0,5";
}
```

Kuvio 12. Hakutoiminnon SQL-kysely

Kuviossa 12 nähdään, kuinka hakukysely toimii. Ihan aluksi mahdollistin kaikkien muistiinpanojen näyttämisen pelkällä * - merkillä. Kun etsitään hakutekstin mukaan, niin haetaan otsikon perusteella haluttua muistiinpanoa LIKE '%$search%'-kyselyllä. Lopuksi järjestetään muistiinpanot ajan mukaan laskevasti sekä rajoitetaan hakutulosten määrä viiteen.

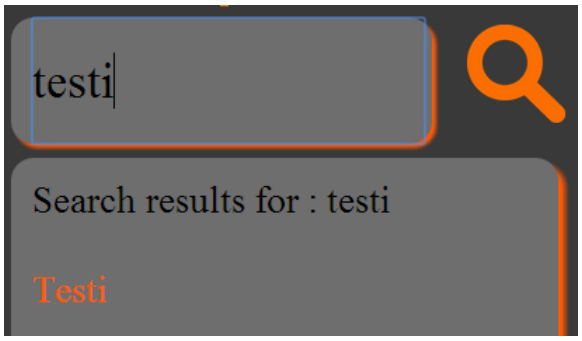
Tietokantayhteyden sekä SQL-lauseen avulla luotiin varsinainen kysely mysqli_query-komennolla ja jälleen lisättiin if-lauseella tarkistus, onnistuiko kysely. Kyselystä laskettiin rivien määrä mysqli_num_rows-komennolla. Jos kyselyllä ei löydetä yhtääkään riviä tietokannasta, niin tulostetaan print-komennolla, että yhtäkään hakutulosta ei löytynyt ja suljetaan SQL-kysely sekä tietokantayhteys. Jos löydetään yksi tai enemmän rivejä tietokannasta, niin tällöin haetaan while-lauseella jokainen rivi joka löytyi tietokannasta käyttäen mysqli_fetch_array-komentoa. Tietokantakyselylle luodaan myöskin mahdollisia virhetilanteita varten virheilmoitukset.

```
$rows = mysqli_num_rows($tj);
    if($rows == 0)
    {
    print("<br/>No search results");
    mysqli_free_result($tj);
    }
    else if($rows > 0)
    {
        while($form = mysqli_fetch_array($tj, MYSQLI_ASSOC))
        {
            print("<br/><a href='#' id='" . $form["id"] . "'
            onclick='getPage(this.id)' tabindex =\"2\">".
            $form["header"] . " " . $form["time"] . "</a><br/>");
        }
        mysqli_free_result($tj);
    }
```

Kuvio 13. Rivien etsiminen tietokannasta ja sen liittäminen <a>-elementtiin

Kuviossa 13 näkyy itse while-lauseen rakenne. Haen rivitiedot <a>-elementtiin, jonka id:ksi sijoitetaan tietokannasta saatu id-rivi ja tekstiksi haetaan rivin otsikko sekä päivämäärä. Onclick-eventin getPage-funktiota käytetään kyseisen muistiinpanon hakemiseen tietokannasta. Kun rivin id sijoitetaan <a>- elementin id:ksi, niin sitä voidaan helposti kutsua this.id-komennolla getPage(this.id). Tabindexin avulla voidaan tabulaattorilla selata hakutuloksia läpi.

Eri hakutulokset siis tulostetaan tekstikenttään linkkeinä ja kun hakutuloslinkkiä klikkaa, niin siirrytään itse muistiinpanoon, jolloin muistiinpanon otsikko ja teksti haetaan näytölle. Tässä vaiheessa pystyin jo testaamaan, toimiiko haku.



Kuvio 14. Esimerkki hakutilanteesta

Kuten kuviosta 14 nähdään, haku toimii moitteetta ja tulostaa hakutuloksen perusteella halutun linkin. Haku löytää tässä vaiheessa jo muistiinpanoja, koska niitä lisättiin tietokannan luonnin aikana suoraan tietokannasta.

Kun hakutoiminto oli testattu, seuraavaksi tehtävänä oli saada tulostettua muistiinpano hakukentästä niille kuuluviin kenttiin.

```
//Part of getPage(). This retrieves page header
function getPageHeader(str)
{
var xmlhttp = headerAjax();

xmlhttp.open("POST", "getHeader.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("iid="+str);
}

//Part of getPage(). This retrieves page text
function getPageText(str)
{
var xmlhttp = MainAjax();

xmlhttp.open("POST", "getText.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("iid="+str);
}
```

Kuvio 15. GetPage-funktion kautta käynnistetään otsikon haku sekä tekstin haku tietokannasta eri funktioilla.

Otsikon sekä tekstin tulostukseen käytetään kuviossa 15 näkyviä funktioita, jotka suoritetaan hakutuloslinkkiin luodun getPage-funktion kautta. Nämä funktiot lähettävät PHP:lle klikatun <a>-elementin id:n, joka haettiin tietokannasta hakutilanteessa kyseiselle muistiinpanolle. Funktiot suorittavat sekä otsikon että tekstin tulostuksen niille kuuluviin kenttiin.

Loin kaksi uutta PHP-tiedostoa otsikon ja tekstin hakua varten. Id:n avulla etsitään tietokannasta haluttu muistiinpano ja tulostetaan sen tiedot. PHP-tiedostojen koodien periaate on sama kuin aiemmin luodulla hakutoiminnon PHP:llä. Poikkeuksena on SQL-lause, johon lisätään ”WHERE id = ’$id’”, missä $id on hakutuloslinkin id. While-lauseessa tulostetaan vain tietokannasta otsikko ja teksti. Muuten koodin syntaksi pysyi samanlaisena.

Kuvio 16. Muistiinpanon tulostus, kun hakutuloslinkistä on painettu.

Kuviossa 16 on otsikkopalkkiin sekä tekstipalkkiin tulostettu Testi-niminen muistiinpano tietokannasta. Muistiinpanojen tulostus onnistuu hienosti. Muistiinpano haettiin linkkiä painamalla, jolloin PHP haki id:n avulla kyseisen muistiinpanon tietokannasta. Tässä kohtaa moodi vaihdetaan tallennusmoodiin.

Kun muistiinpanon tulostus onnistui, aloin tehdä itse sekä otsikon että tekstin tallennusta.

```
function headerSave()
{
var headerinput = encodeURIComponent(document.getElementById("header").innerHTML);
var iid = tempid;
var xmlhttp = basicAjax();

xmlhttp.open("POST", "saveHeader.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("header="+headerinput+"&iid="+iid);
}

function textSave()
{
var textinput = encodeURIComponent(document.getElementById("textinput").innerHTML);
var iid = tempid;
var xmlhttp = basicAjax();

xmlhttp.open("POST", "saveText.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("text="+textinput+"&iid="+iid);
}
```

Kuvio 17. Funktiot otsikon ja tekstin tallennukselle

Välttääkseni turhaa tallennusta, loin erikseen otsikolle ja tekstille omat tallennusfunktionsa. Tällöin ei tarvitse molempia kenttiä tallentaa, jos kirjoittaa otsikkoon tai tekstiin lisäystä. Kuviossa 17 headerSave-funktiolla tallennetaan

otsikkopalkin sisältö ja textSave-funktiolla tekstipalkin sisältö. Funktioissa id:n lisäksi PHP:lle lähetetään myös otsikko- tai teksti-muuttuja.

Otsikon ja tekstin tallennus PHP:ssä käytetään SQL UPDATE-komentoa. Se määritellään seuraavasti ja esimerkkinä otsikon tallennus: “UPDATE articles SET header=’$header’ WHERE id=’$id’”, jossa $header on otsikkopalkista haettu tieto. Muuta määrittelyä ei tarvita tallennuksessa.

Kun sain tehtyä tallennustiedostot valmiiksi, aloin testata tallennusta. Testaus alkoi huonosti, sillä aluksi kaikki otsikko- tai tekstipalkkiin kirjoitettu teksti katosi joka napin painalluksella. Kävin kaikki funktiot sekä PHP-tiedostot huolella läpi mutta mitään virhettä en löytänyt. Lopuksi kuitenkin huomasin puuttua itse tallennuksen XMLHttpRequest-funktioon. Siinä määriteltiin, että xmlhttp.responseText sijoitettiin joko otsikko- tai tekstipalkkiin aina tallentaessa. Mutta tämä tarkoitti sitä, että jokaisella napin painalluksella tapahtui ns. ”Ajax-päivitys” sivulla, jolloin haettiin uudelleen tietokannasta muistiinpanon otsikko tai teksti ja tämä johti tekstin totaaliseen katoamiseen. Vielä enemmän syvennettyäni asiaan huomasin, että tässä kohtaa responseTextin hakeminen oli täysin turhaa. Otsikko tai teksti haettiin aina turhaan tietokannasta joka tallennuksen jälkeen. Korjasin tämän asian pelkistämällä tallennus-XMLHttpRequest-funktion siten, että responseTextiä ei sijoiteta mihinkään tallennuksen aikana. Tällöin sivu ei päivity turhaan vaan tallennus suoritetaan täysin taustalla. Korjauksen jälkeen tallennus alkoi toimia kuten pitääkin.

Lopuksi lisäsin sovellukseen uuden sivun luonti-toiminnon. Kun uusi sivu luodaan, otsikoksi tulee tulla New Page ja tekstikenttä on tyhjänä. Samalla moodi vaihtuu tallennusmoodiksi.

Uuden sivun luonti-PHP:ssä otsikoksi laitetaan suoraan ”New Page” ja nyt ajaksi otetaan PHP:n date-komennolla luontipäivämäärä. Sivun luonnissa käytetään SQL INSERT-lausetta ja jonka syntaksi menee näin: INSERT INTO articles VALUES (NULL, ’$header’, ’$text’, ’$time’). Id:n arvolla on NULL, koska id:llä on tietokannassa auto increment. Seuraavaksi keksin käyttää PHP-komentoa mysqli_insert_id(yhteyden nimi). Tällä komennolla saan haettua juuri luodun

muistiinpanon id:n ja sitä käyttämällä pystyn samalla hakea ja tulostaa sen tietokannasta JavaScriptin tempidn käyttöön.

Kun sivun luonti PHP-tiedosto on suoritettu, funktio hakee juuri luodun muistiinpanon tietokannasta.

Kuvio 18. Uuden sivun luonti

Uuden sivun luonti tulostaa näytölle kuvion 18 mukaisen tuloksen. Nyt uudelle muistiinpanolle voi kirjoittaa oman otsikon ja tekstin.

4.1.5 Ensimmäisen version testaus ja tulosten tulkinta

Kun kaikki toiminnot olivat tehtynä, suoritin vielä perusteellisen testauksen ohjelmalle. Perustoiminta oli hyvä mutta mitä enemmän syventyi ohjelmaan sitä enemmän alkoi löytymään virheitä.

Ensimmäinen isompi vika, joka löytyi, oli tekstin tallennuksessa. Viimeinen tallennettu kirjain jäi miltei joka kerta pois. Pitkän aikaa etsittyäni löysin vihdoin tuon ihmeellisen vian. Vian aiheutti onkeydown-event tallennuksessa. Kun vaihdoin tuon eventin onkeyupksi, vika poistui. Tämän täytyy johtua siitä, että funktio keritään suorittaa ennen kuin näppäimistö kerkeää lisätä kirjaimen selaimelle. Tällöin viimeinen kirjain jää tallentamatta. Tuntuu hullulta ajatukselta

mutta vika korjaantui, kun vaihdoin eventin onkeyup:ksi eli tallennus tehdään näppäimen tullessa takaisin ylös. Tämä korjataan täysin seuraavassa versiossa.

Toinen iso vika ilmaantui, kun painettiin hakutuloslinkkiä. Tällöin haettuun muistiinpanoon tallentui hakutulosten listanäkymä. Asiaa tutkittuani päättelin, että tämä johtui siitä, että onclick-eventillä suoritettiin tekstikentässä sekä hakutuloksen haku että tekstin tallennus. Kun linkkiä painetaan, Ajax on välillä niin nopea, että se muistiinpanon haun jälkeen tallentaa välittömästi onclick-eventillä haettuun muistiinpanoon hakutuloslistan. Tämä vika on johtuu siitä, että tekstikenttä ja hakutuloskenttä ovat samassa kentässä. Tämä vika tullaan korjaamaan seuraavassa versiossa.

Ensimmäinen versio oli mielestäni kokonaisuudessaan tyydyttävä, vaikkakin varsinkin tallennuksessa oli vielä paljon parannettavaa ja ohjelmasta löytyi paljon korjattavia virheitä, jotka tulee korjata seuraavaan versioon. Ohjelmasta puuttuu myös vielä täysin poisto-mahdollisuus eli seuraavaan versioon riittää vielä koodin osalta tekemistä. Ajattelin myös lisätä kirjautumisen seuraavaan versioon.

Lopullinen ulkoasu ei myöskään pitkän päälle enää miellyttänyt. Värit olivat tylsät ja jotenkin koko sisällön asettelu ontui. Varsinkin tekstikentän ja hakutuloskentän yhdistäminen on aiheuttanut vain harmia. Päätin seuraavaan versioon päivittää ulkoasun miltei kokonaan uudenlaiseksi.

4.2 Sovelluksen toinen versio

Toiseen versioon päätin uusia kokonaan sovelluksen ulkoasun, korjata tallennukseen ja muihin toimintoihin liittyviä suuria bugeja, lisätä kirjautumisen ja muistiinpanon poistotoiminnon. Ensimmäisen version toiminnallinen puoli onnistui mielestäni hyvin ja näihin ominaisuuksiin tehdään vain pieniä korjauksia tässä versiossa.

4.2.1 Ulkoasun uudistaminen

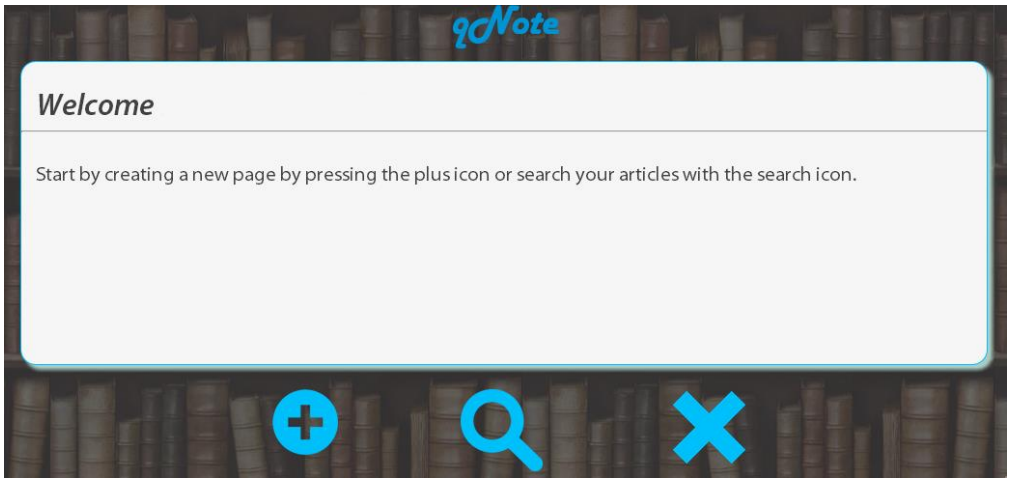
Ensimmäisen version ulkoasu oli pettymys niin visuaalisesti kuin toiminnalliselta kannalta. Aloitinkin ulkoasun luonnin melkein puhtaalta pöydältä. Suunnittelun

alussa käänsin katseeni ensin opinnäytetyössäni mainittuihin sovelluksiin eli OneNoteen sekä Evernoteen ja niiden visuaaliseen toteutukseen. Molemmissa ohjelmissa huomasin yhden samanlaisen piirteen eli otsikon ja tekstin ryhmittelyn. Ajatus oli siis, että otsikkopalkki sekä tekstipalkki ovat ikäänkuin samaa palkkia. Tätä ryhmittelyä päätin myös käyttää omassa sovelluksessani.

Ulkoasun uudistuksen aloitin kuitenkin tekemällä uuden taustan. Mikään taustaväri ei oikein lopuksi kelvannut ja päädyinkin lopulta käyttämään taustakuvaa. Taustakuvaksi valitsin vanhan kirjahyllyn kuvan, joka toistuu taustalla. Tämä mielestäni lisäsi hyvällä tavalla visuaalisuutta sovellukselle. Seuraavaksi päätin värimaailmaksi kaikille elementeille vaaleansinisen sekä valkoisen, jotka olivat mielestäni paljon neutraalimpia värejä kuin edellä käytetyt harmaa sekä oranssi.

Seuraavaksi tein itse otsikko- ja tekstipalkkiratkaisun. Kehitin tämän mallin muiden muistiinpanosovellusten innoittamina. Ensin loin itse pääpalkin, jonka sisään tulevat otsikko- ja tekstipalkki. Ylimpänä on itse otsikkopalkki ja alempana suurempi tekstipalkki. Palkin erotellaan <hr>-elementin eli viivan avulla toisistaan. Lisäsin myös palkeille taustavärin, kun palkin päälle tullaan hiiren kanssa tai kun palkilla on fokus.

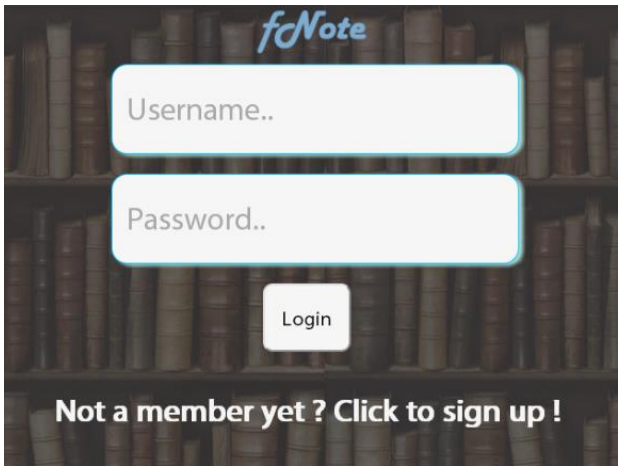
Siirsin painikkeet sovelluksen alareunaan ja ne asetettiin riviin. Koska poistotoiminto tullaan lisäämään tässä versiossa, niin painikkeiden määrä nousi kolmeen. Poistopainikkeen kuvakkeeksi valitsin X-muotoisen kuvan, joka myöskin tuo esiin sen käyttötarkoituksen. Koko sisällölle lisään vielä läpinäkyvän tumman taustan, jolloin se erottuu paremmin taustasta.



Kuvio 19. Uudistettu ulkoasu

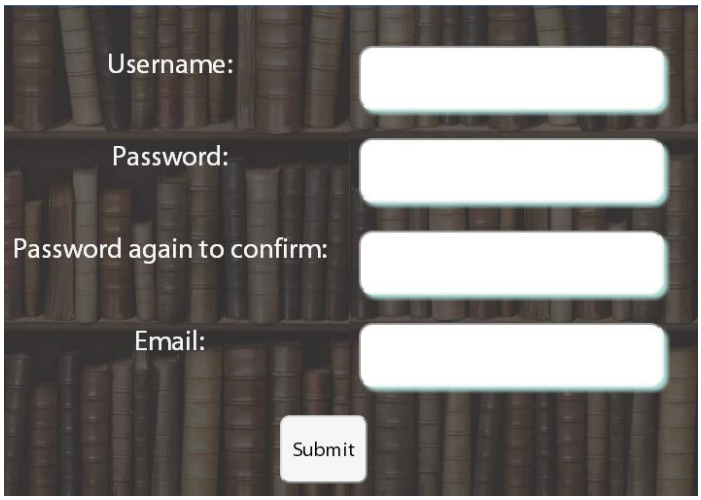
Kuviossa 19 näkyy sovelluksen täysin uudistettu ulkoasu. Olin todella tyytyväinen tähän uuteen ulkoasuun, joka oli paljon miellyttävämpi silmälle kuin ensimmäisen version ulkoasu. Uudistettu otsikko- ja tekstipalkki, uusi tausta ja värimaailma sekä painikkeiden niputtaminen yhteen toivat paljon kaivattua selkeyttä sovellukselle, mitä jäin kaipaamaan ensimmäisessä versiossa.

Lopuksi tein kirjautumiselle erillisen ikkunan, jonka kautta siirrytään itse sovellukseen. Tein käyttäjätunnukselle sekä salasanalle omat palkit ja kirjautumispainikkeen. Näiden alle tein tekstipainikkeen, josta avautuu liittymislomake, jonka täyttämällä ja lähettämällä saadaan luotua oma käyttäjätunnus sovellukseen.



Kuvio 20. Kirjautumisikkuna

Kuten kuviosta 20 nähdään, kirjautumisruudussa saadaan heti kirjoitettua tunnus sekä salasana eli sovellukseen kirjautuminen on nopeaa. Alareunassa näkyy tekstipainike, josta itse liittymislomake ilmestyy ruudulle.



Kuvio 21. Liittymislomakkeen kentät

Kuviossa 21 näkyy liittymislomakkeen kentät, jotka ovat käyttäjätunnus, salasana, salasana uudestaan ja sähköposti. Alimpana on lähetys-painike, jolla vahvistus lähetetään käyttäjän sähköpostiin ja lisää tunnuksen tietokantaan.

Uudistettu ulkoasu oli mielestäni erittäin onnistunut ja se loi pohjan sovelluksen toiminnallisuuden parantamiseen. Seuraavaksi aloin keskittymään sovelluksen toiminnallisuuden virheiden korjaamiseen.

4.2.2 Toiminnallisuuden bugien korjaus ja poistotoiminnon lisäys

Ensimmäisen version testaus toi ilmi ainakin pari suurempaa vikaa, jotka tuli korjata toiseen versioon ja aloitinkin ensin helpoimmasta päästä eli onkeydownin muuttamisen onkeyupksi. Korjaus oli nopea tehdä, koska minun tarvitsi vain vaihtaa kaikki onclick-eventit onkeyupksi. Tämä pieni korjaus kuitenkin korjasi tallennusongelman kokonaan, jossa viimeinen kirjain tai useampi jäi tallentamatta.

Kun tämä ensimmäinen tallennusbugi oli saatu korjattua, siirryin toiseen tallennusbugiin. Hakutulosten ja muistiinpanon tekstin tulostaminen samaan palkkiin ei toiminut kuten piti, koska hakutulokset tallennettiin joskus muistiinpanoon, joka oli valittu hakutuloksista. Huomasin sen johtuvan siitä, että samassa tekstipalkissa oli yhtäaikaa onclick-event sekä hakutuloksen hakemiselle että muistiinpanon tekstin tallentamiselle. Tähän piti keksiä uusi ratkaisu. Parhaaksi ratkaisuksi jäi lopulta erillinen hakukenttä, joka ilmestyy silloin, kun käyttäjä hakee hakupalkilla muistiinpanoa. Tämä yksinkertainen ratkaisu korjasi monta ongelmaa niin koodissa että itse sovelluksessa. Kun hakupalkki ja tekstipalkki ovat erillään, niin haku- ja tallennus onclick-eventtejä ei enää suoriteta yhtä aikaa, jolloin hakutulokset eivät enää tallennu juuri haettuun muistiinpanoon. Samalla sain poistettua koodista eri moodit, koska nyt hakupalkki ja muistiinpanon otsikko- ja tekstipalkit olivat kiinteitä, jolloin eri moodeja ei enää tarvita.

Ensimmäisestä versiosta oli vielä jäänyt puuttumaan kokonaan muistiinpanon poistomahdollisuus, joten lisäsin sen toiseen versioon. Poistotoiminnon lisääminen oli nopeaa, koska sen lisäämiseen ei tarvittu mitään uutta kooditietoa. Poistotoiminto toimii niin, että käyttäjä siirtyy poistettavaan muistiinpanoon ja sen jälkeen painaa poisto-nappia. Ajax-kutsut ovat samanlaisia kuten muissakin toiminnoissa eli tietokantaan lähetetään tempid-muuttuja, jonka avulla poistetaan haluttu muistiinpano. Poistamisen PHP-tiedostossa MySQL-lause menee: "DELETE FROM articles WHERE id= '$id'".

Lopuksi korjasin tekstinkäsittelyyn liittyviä pieniä bugeja. Poistin mm. mahdollisuuden tehdä uusi rivi otsikkopalkissa, koska sovelluksen muotoilu alkoi kärsiä, jos otsikkopalkilla oli monta riviä.

4.2.3 Kirjautumisen lisääminen sovellukseen

Useamman käyttäjän mahdollisuus omassa muistiinpanosovelluksessani oli ollut määränpääni aivan sovelluksen suunnittelun alusta saakka. Olen opinnoissani ja muissa omissa projekteissani saanut jo hyvin kokemusta kirjautumisrutiinista, joten kirjautumisominaisuus ei ollut ihan vieras. Kirjautumisen lisääminen sovellukseen ei ollut pieni juttu vaan miltei jokaiseen JavaScript-, PHP- ja MySQL-koodiin piti tehdä muutoksia. Myös uusi taulu jouduttiin tekemään tietokantaan. Tarkoituksena oli, että käyttäjä luo kirjautumislomakkeen kautta omat tunnuksensa, jonka jälkeen käyttäjä voi tehdä omia muistiinpanoja sovellukseen.

Aloitin kirjautumisen teon uuden taulun luomisella tietokantaan. Taulun nimeksi tuli users, jolle lisään kaksi saraketta: username ja password. Username-sarakkeeseen tallennetaan käyttäjätunnus ja password-sarakkeeseen salasana. Username-sarakkeelle annan tyypin varchar(50), koska käyttäjät laittavat varmasti kirjaimia ja numeroita käyttäjätunnukseensa ja mielestäni pituus on riittävä. Salasana-sarakkeen tyypiksi annan varchar(255), jonka pituus riittää myös mahdollisesti myöhemmin lisättävään salaukseen.

#	Nimi	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	I
1	**username**	varchar(50)	latin1_swedish_ci		Ei		
2	**password**	varchar(255)	latin1_swedish_ci		Ei	*None*	

Kuvio 22. Users-taulun rakenne

Kuviosta 22 nähdään users-taulun tietokantarakenne. Sarakkeille annetaan myös latin1_swedish_ci-kielikoodi, jotta käyttäjät voivat laittaa ö:n, ä:n ja å:n tunnuksiinsa. Nämä kaksi kenttää riittävät hyvin kirjautumisrutiinin luontiin. Loin itselleni testausta varten käyttäjätunnuksen Aku.

Samalla lisään myös uuden author sarakkeen articles-tauluun, jolla saan yksilöityä muistiinpanot eri käyttäjille.

	#	Nimi	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo	Lisätiedot
☐	1	id	int(255)			Ei	None	AUTO_INCREMENT
☐	2	header	varchar(100)	latin1_swedish_ci		Kyllä	NULL	
☐	3	text	text	latin1_swedish_ci		Kyllä	NULL	
☐	4	time	date			Kyllä	NULL	
☐	5	author	varchar(50)	latin1_swedish_ci		Ei	None	

Kuvio 23. Articles-taulun uusi rakenne

Uusi rakenne näkyy kuviossa 23, jossa authorille annettiin sama tyyppi kuin käyttäjätunnukselle eli varchar(50). Annoin jo luoduille muistiinpanoille authoriksi oman käyttäjätunnukseni testausta varten.

Seuraavaksi oli vuorossa itse kirjautuminen sovellukseen. Ulkoasuosiossa olin jo luonut aloitussivun, jonka kautta siirrytään itse sovellukseen ja tämän nimi on index.php ja itse sovelluksen sivun nimi on mainpage.php. Kun käyttäjä on syöttänyt tunnuksensa ja painaa login-painiketta, niin JavaScriptin ja Ajax:in kautta lähetetään tunnus ja salasana autentikointiPHP:lle.

AutentikointiPHP:ssä tietokannasta valitaan käyttäjätunnus ja salasana ja verrataan selaimelta tulleita tunnuksia. Jos tunnukset ovat väärin, sovellus ilmoittaa asiasta eikä käyttäjä pääse sovellukseen. Jos tunnus on oikein, tällöin käynnistetään käyttäjän sessio session_start-komennolla ja haetaan käyttäjätunnus tietokannasta PHP Session-muuttujaan nimeltä username ja ohjataan käyttäjä itse sovellukseen. PHP Session-muuttujaa tarvitaan estämässä käyttäjän pääsyä kirjautumisen ohi URL:in kautta.

```php
$rows = mysqli_num_rows($tj);
if($rows == 0)
{
        print("0");
        mysqli_free_result($tj);
        mysqli_close($connection);
}
else if($rows == 1)
{
        session_start();
        while($form = mysqli_fetch_array($tj, MYSQLI_ASSOC))
        {
                $uid = $form["username"];
                $_SESSION["username"] = $uid;
        }
        mysqli_free_result($tj);
        mysqli_close($connection);
        // redirection to mainpage
        header("Location:mainpage.php");
}
```

Kuvio 24. Käyttäjätunnuksen liittäminen PHP Session-muuttujaan ja uudelleenohjaus itse sovellukseen.

Kuviossa 24 nähdään kirjautumisen pääprosessi. Uudelleenohjaus tapahtuu käyttämällä header()-komentoa, jolla siirrytään mainpage.php:hen. Ennen header-komentoa ei saa olla mitään HTML-koodia tulostettuna PHP:llä, koska tällöin header-komento ei toimi.

Kun PHP-tiedosto oli valmis, niin tein eston itse HTML-koodiin, ettei itse sovellukseen päästä käsiksi ilman kirjautumista. Tämä hoituu asettamalla PHP-koodi ennen mitään HTML-koodia. PHP-koodin alussa käynnistään taas sessio, jolla päästään myös käsiksi PHP Session-muuttujaan. Tässä kohtaa otetaan käyttöön kirjautumisvaiheessa luotu PHP Session-muuttuja nimeltä username. Muuttujalla tarkistetaan onko käyttäjä kirjautunut. Jos käyttäjä on syöttänyt tunnukset oikein, tällöin tunnus on lisätty Session-muuttujaan ja PHP löytää sen ja päästää käyttäjän sovellukseen. Jos käyttäjä ei ole kirjautunut, tällöin PHP katkaisee yhteyden. Käytin yhteyden katkaisemiseen die-komentoa.

```php
<?php
session_start();

if((!isset($_SESSION["username"])) && (empty($_SESSION["username"])))
{
    die("<title>Authentication</title>You have not logged in.");
}
?>
```

Kuvio 25. Käyttäjä estetään, jos käyttäjätunnusta ei ole lisätty kirjautumisvaiheessa PHP Session-muuttujaan.

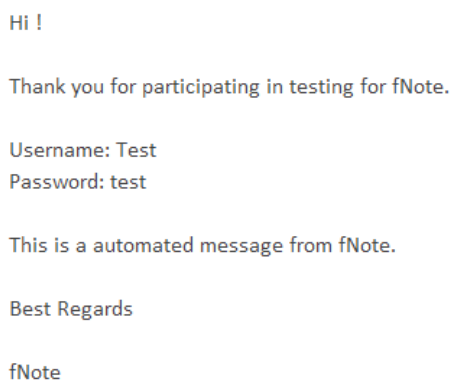
Kuvion 25 koodista nähdään, että jos Session-muuttuja usernamea ei ole asetettu tai se on tyhjä, niin tällöin katkaistaan yhteys die-komennolla ja tulostetaan virheteksti käyttäjälle.

Tässä vaiheessa jouduin miettimään, kuinka muistiinpanot saadaan haettua tietokannasta käyttäjän perusteella. Päätin, että muistiinpanot haetaan käyttäjän tunnuksen mukaan, mikä tarkoitti sitä, että käyttäjän tunnus pitäisi saada johonkin talteen JavaScriptin käyttöön. Ensiksi tein väliaikaisen div:in, johon tulostin Session-muuttujan, jossa käyttäjäntunnus oli tallessa. Tämän jälkeen JavaScriptillä haettiin divin sisältö, jonka jälkeen se voitiin lähettää PHP-tiedostolle. JavaScript-toimintofunktioihin lisättiin vain ”var author= document.getElementById(”hiddendiv”).innerHTML;” PHP-tiedostoissa lisättiin MySQL-kyselyihin lisäehto ”WHERE author=’JavaScriptiltä tullut käyttäjätieto’”.

Kuitenkin menetelmää testattuani huomasin väliaikaisen divin olevan äärimmäisen paha tietoturva-aukko, koska divin arvo saadaan helposti muutettua eri menetelmillä. Tällöin jos toisen käyttäjän tunnus vaihdetaan diviin, päästään käsiksi sen käyttäjän muistiinpanoihin ilman eri kirjautumista hakupalkin kautta. Lopulliseksi ratkaisuksi keksin käyttää hyväkseni PHP Session-muuttujaa kaikissa PHP-tiedostoissa. Tällöin käyttäjätunnus voidaan lähettää PHP:ltä PHP:lle eikä JavaScriptiä tarvita ollenkaan tunnuksen lähetykseen, joka parantaa paljon tietoturvaa. Tämä myös vähentää JavaScriptistä koodin määrää, joten ratkaisu oli monelta kannalta hyvä. PHP-tiedostojen alkuun lisätään vain session_start-komento, jotta saadaan siirrettyä session kirjautumisvaiheessa luotu muuttuja

käyttöön PHP-tiedostoissa. Käyttäjän muistiinpanoja paikannetaan PHP:ssä muuttujalla ”$author = $_SESSION["username"];” ja MySQL-kyselyllä ”SELECT * FROM articles WHERE id='$id' AND author = '$author';”

Lopulta kun kaikki muut toiminnot oli saatu tehtyä, tein vielä kirjautumislomakkeen toiminnallisuuden. Kun käyttäjä oli täyttänyt kaikki kentät ja painanut lähetä-painiketta, niin ohjelma lisää tunnukset tietokantaan ja lähettää sähköpostilla varmistusviestin käyttäjälle. PHP:ssa tunnukset lisätään INSERT-komennolla. Lisäyksen syntaksi on "INSERT INTO users VALUES ('$user','$password')". Tämän jälkeen käyttäjä voi heti käyttää luotuja tunnuksiaan.


Hi !

Thank you for participating in testing for fNote.

Username: Test
Password: test

This is a automated message from fNote.

Best Regards

fNote

Kuvio 26. Varmistusviesti sovellukselta lomakkeen lähetyksen jälkeen

Sovellus lähettää PHP mail-ominaisuudella käyttäjälle kuvion 26 mukaisen varmistusviestin, jossa näkyvät luotu tunnus ja salasana.

4.2.4 Toisen version testaus ja tulosten tulkinta

Kun kaikki toisen version korjaukset ja lisäykset oli tehty, oli taas lopputarkastuksen aika. Sovellus alkoi vihdoin näyttää ja tuntua oikealta sovellukselta. Muistiinpanon hakeminen hakupalkilla, tekstin tallennus ja muistiinpanon poistaminen toimivat moitteetta ja tämä oli iso helpotus, koska nämä ovat ohjelman ydinkohtia, joiden pitää toimia 100-prosenttisesti.

Kuitenkaan virheettömästä versiosta ei ollut kyse, koska tekstinkäsittelyssä ja varsinkin tietoturvassa oli vielä paljon virheitä. Tekstinkäsittelyssä jotkin

erikoismerkit katkaisevat välillä tekstin siitä kohtaa, minne erikoismerkki on laitettu. Ohjelmaan pystytään kirjautua SQL-injektion avulla ja PHP- ja MySQL-koodia voidaan syöttää tekstikenttiin, joten tietoturvasta ei ole vielä tietoakaan ja näitä tietoturva-aukkoja tullaan korjaamaan kolmanteen versioon.

Loppujen lopuksi vaikka ulkoasu oli jo hyvä, niin löysin yhden pahan epäkohdan nimittäin painikepalkin sijottelun. Kun muistiinpano on oikein pitkä, painikepalkki eksyy kauas sivuston alareunaan. Nyt joudutaan aina rullaamaan sivun alareunaan, että saadaan haettua, lisättyä tai poistettua muistiinpanoja. Hakupalkin esiintuominen napin kautta alkoi myös tuntua oudolta idealta, koska hakupalkkiahan käytetään miltei kokoajan ja se myös tuo samalla lisää ylimääräisiä napin painalluksia. Päätinkin, että se saakin olla aina esillä. Ainakin nämä kohdat tulen muuttamaan kolmanteen versioon.

Kokonaisuutena olin jo erittäin tyytyväinen toisen version tulokseen. Sovellus toimii jo pääpiirteittäin hyvin. Kuitenkin jäin kaipaamaan joitain ominaisuuksia sovelluksesta kuten julkisen rajapinnan, jolla käyttäjät voivat asettaa muistiinpanonsa halutessaan julkisiksi kaikille käyttäjille. Liittyminen sovellukseen tapahtuu myös ehkä turhan suoraviivaisesti ja sen kautta voidaankin myös lähettää haittakoodia tietokantaan ilman vaikeuksia, joten tähänkin tulen tekemään paremman keinon.

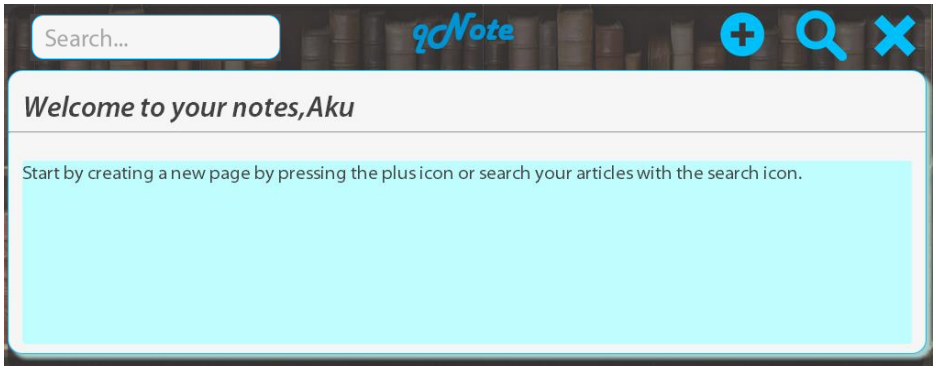
4.3 Kolmas versio

Kolmanteen versioon tärkeimpänä parannuksena päätin lisätä tietoturvaa runsaasti ja tarkoitus olikin täyttää mahdollisimman monta tietoturva-aukkoa. Sovellukseen rekisteröityminen tehtäisiin vahvistusviestin kautta, joka lähetetään käyttäjän sähköpostiin, jonka on tarkoitus ainakin vähentää hakkereiden haittaohjelmien mahdollisuutta tehdä tunnus sovellukseen. Ulkoasuun tehdään vielä parannuksia ja sovellukseen lisätään myös julkinen rajapinta muistiinpanoille.

4.3.1 Ulkoasun muuttaminen

Jälleen aivan aluksi paransin sovelluksen ulkoasun asettelua käyttäjäystävällisemmäksi. Ajatuksena oli siirtää painikkeet sovelluksen yläreunaan ja asettaa hakupalkki aina näkyväksi sivun yläreunaan.

Aloitin siirtämällä painikkeet sivun oikeaan yläreunaan hieman pienempinä kuin ne olivat alunperin. Nyt ne ovat aina näkyvissä, vaikka sovelluksessa olisikin pitkä muistiinpano. Seuraavaksi lisäsin hakupalkin sivun vasempaan yläreunaan, joka on näkyvissä kokoajan.

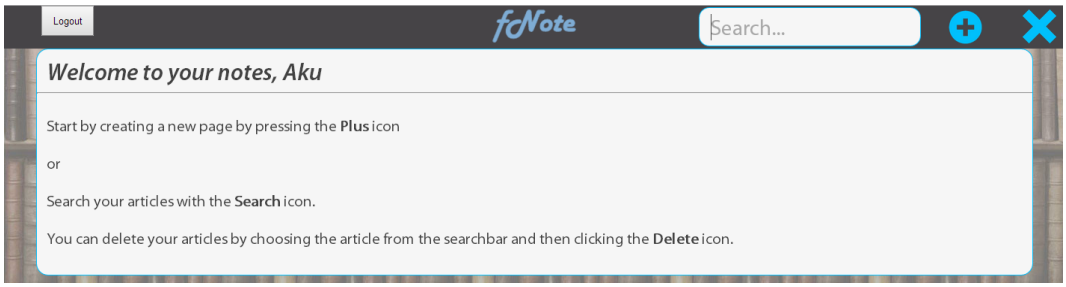


Kuvio 27. Painikkeiden ja hakupalkin sijoittaminen uudelleen.

Kuviossa 27 nähdään painikkeiden ja hakupalkin uudet sijoituspaikat. Tämä sijoittelu oli mielestäni erittäin onnistunut ja mielessäni kävi jo, että tämä on melko lopullinen asettelu näille elementeille. Mutta vaikkakin elementtien sijoittelu olikin miltei täydellinen, niin vanha ongelma oli edelleen olemassa. Ongelmana oli se, kun muistiinpano oli pitkä ja rullattiin sivuston alareunaan minne muistiinpano loppui, niin painikkeet ja hakupalkki olivat tällä kertaa sivuston yläosassa ja jäivät edelleen piiloon.

Ratkaisuksi tein kiinteän yläpalkin, joka liikkuu mukana, kun sivua rullataan alaspäin ja takaisin ylös. Tämä onnistui käyttämällä CSS-ominaisuutta nimeltä position, jolla annettiin arvo fixed. Tämä tarkoittaa sitä, että elementti on aina näkyvillä näytöllä annettujen koordinaattien mukaan, vaikka sivustolla rullataan ylös, alas tai sivuttain. Laitoin yläpalkin taustaväriksi harmaanmustan. Lisäsin painikkeet ja hakupalkin yläpalkin sisälle. Poistin myös samalla haku-painikkeen,

koska sitä ei enää tarvitse käyttää. Muistin tässä vaiheessa myös, että sovelluksesta uloskirjautuminen tarvitsi myös painikkeen ja lisäsinkin sen samalla vasempaan yläreunaan. Siirsin lopuksi vielä hakupalkin oikeaan yläreunaan painikkeiden vasemmalle puolelle ja vaalensin myös samalla hieman taustakuvaa, etteivät taustan yksityiskohdat kiinnitä liikaa huomiota.



Kuvio 28. Lopullinen ulkoasu uudella yläpalkkiratkaisulla

Yläpalkin luomisen jälkeen ulkoasu oli erittäin toimiva ja selkeä. Kuviossa 28 nähdään eri elementtien sijoittelu yläpalkissa. Pienenä loppusilauksena tein vielä uloskirjautumispainikkeen vasemmalle puolelle näkymättömän divin, johon tulee sivun lataamista esittävä GIF. Kuvassa pyörii sininen neliö ympyrää. Tämä GIF näkyy aina silloin, kun sivu suorittaa jotain JavaScript-funktiota. Tämän tarkoitus on osoittaa käyttäjälle, että sovelluksen suorittaa toimintoja taustalla esim. muistiinpanoa haettaessa. LatausGIF luodaan yksinkertaisesti tekemällä funktio loadingGif(), joka sijoittaa näkymättömään diviin GIF:in. Divin innerHTML:lle annetaan funktiossa kuvan paikka palvelimella eli "img src='images/circle.gif'"; Kun GIF halutaan näyttää sivustolla, niin juuri luotu loadingGif()-funktio sijoitetaan vain suoritettavan funktion alkuun, jolloin GIF alkaa pyöriä. Kun GIF halutaan lopettaa, niin luodaan erillinen funktio stopLoadingGif(), jossa divin innerHTML tyhjätään, jolloin GIF poistuu näytöltä.

4.3.2 Tietoturvan parantaminen

Tavoitteena oli parantaa tietoturvaa niin, että sovellus olisi ainakin tavallisimmilta tietoturvahyökkäyksiltä turvassa kuten SQL-injektioilta.

Keskityin ensimmäisenä SQL-injektion estämiseen. Ongelmana kaikissa tekstikentissä oli se, että niihin voitiin syöttää HTML, PHP tai MySQL-kieliä ja

tällöin sovelluksen sisältö joko muuttui tai muistiinpanoja voitiin manipuloida. Ensimmäisenä ratkaisuna lisäsin jokaiseen PHP:hen strip_tags-nimisen komennon. Tämä karsii pois kaikki tagit, joita löytyy strip_tags-komennolle annettavassa muuttujassa mutta sille voidaan syöttää myös poikkeukset, jotka pääsetään läpi. Syntaksi menee näin: ”strip_tags($muuttuja, ’<poikkeustagit>’)”. Tällä tavalla PHP:lle ei mene läpi esim. ”<?php echo ”Hello World !” ?>”. Tämä metodi estää jo useita SQL-injektiotapauksia mutta ei vielä riitä täydelliseen suojaan.

Strip_tags-metodin avuksi tehtiinkin JavaScriptilla encodeURIComponent()-komento jokaiselle muuttujalle joka lähetetään PHP:lle. Tämä komento muuttaa jokaisen erikoismerkin ja korvaa ne JavaScriptin omilla ”encode-entiteeteillä”, jotka eivät kuitenkaan ole sama asia kuin HTML-entiteetit. Syntaksi menee näin: ”encodeURIComponent(”muuttuja tai elementti, joka halutaan koodata”);” Tällöin erikoismerkit eivät mene PHP:lle asti normaaleina merkkeinä ja PHP ei täten osaa käsitellä niitä. Tällöin esimerkiksi selaimelta tullut koodi jää PHP:ssä suorittamatta.

Kolmanneksi ratkaisuksi tein jokaiselle PHP:lle vielä erikseen str_replace-metodilla erikoismerkkien korvauksen. Str_replace-metodin avulla voidaan korvata haluttu merkki annetusta muuttujasta vaikka esimerkiksi saman merkin HTML-entiteetillä. Syntaksi menee näin: ”str_replace(”merkki, joka halutaan korvata”, korvaava merkki”, muuttuja tai teksti, josta halutaan merkki korvata)”. Nyt voidaan vielä yksitellen korvata halutut merkit pois PHP:lle tulevasta tekstistä. Itse korvasin ainakin seuraavat merkit, jotka ovat haittakoodin kannalta tärkeitä: ”< > / ? : @ & = + $ #”.

Lopuksi tein jokaisen PHP-tiedoston alkuun varmistukset, ettei näihin päästä URL:in kautta suoraan käsiksi. PHP:n alkuun lisätään yksinkertaisesti sama varmistus kuin HTML:llään tehty Session-varmistus eli tarkistetaan onko käyttäjä kirjautunut sisään.

4.3.3 Rekisteröitymisen uudistaminen

Käyttäjän rekisteröityminen toisessa versiossa oli tietoturvallisesti katsottuna iso riski, koska tunnus lisättiin heti tietokantaan ilman mitään tarkistuksia, että onko tunnuksessa tai salasanassa haittakoodeja ja toisessa versiossa ei myöskään tehty varmistusta, että onko kyseessä aito ihminen vai botti. Nämä kohdat oli tarkoitus korjata ja samalla kehittää oma ratkaisu käyttäjän varmistukselle.

Haittakoodien tarkistuksen aloitin JavaScripistä, jossa rajattiin tunnuksen ja salasanan merkit pieniin ja isoihin kirjaimiin sekä numeroihin ja samalla erikoismerkkien laittaminen tunnuksiin on estetty. Tämä tehtiin käyttämällä RegExp()-komentoa, jolla voidaan rajata sallittavat merkit. Oma rajaukseni menee näin: RegExp(”[^a-zA-Z0-9]”). Komennon ^-merkki tarkoittaa sitä, että komento etsii jokaisen kirjaimen joka ei ole sulkujen sisällä ja jos tälläinen löytyy, ehto ei täyty. Tätä komentoa voidaan käyttää tarkistuksessa tekemällä ensin muuttuja esim. ”var reg” ja antamalla sille arvo ”new RegExp(”[^a-zA-Z0-9]”)”, jonka jälkeen muuttujaa kutsutaan tässä tapauksessa reg.test()-komennolla, jonka sulkuihin sijoitetaan tarkistettava elementti.

```
var reg = new RegExp("[^a-zA-Z0-9]");

if (reg.test(document.frmSignup.un.value))
{
alert("Only letters and numbers are allowed");
document.frmSignup.un.select();
return;
}
```

Kuvio 29. RegExp-komennolla tarkistetaan formin usernamen arvo

Kuviossa 29 näkyy oma ratkaisuni usernamen tarkistuksesta. Username eli ”un” tarkistetaan reg.test()-komennolla ja jos käyttäjä laittoi muita merkkejä kuin kirjaimia tai numeroita, ohjelma ilmoittaa asiasta ja keskeyttää toiminnon.

JavaScript-tarkistus ei kuitenkaan ole tarpeeksi vahva, koska JavaScript on melko helppo ohittaa ja kytkeä pois. Tämän vuoksi rekisteröitymis-PHP:lle laitan vielä erikseen PHP:n oman rajauskomennon, joka on preg_match(). Syntaksi on sama kuin itse RegExp-komennossakin eli ”preg_match(’^[a-zA-Z0-9]’)”. Nyt

tunnukselle ja salasanalle on JavaScriptissä ja PHP:ssä tarkistukset, jotka estävät melko tehokkaasti rekisteröitymisen lähetyksen kautta tulevat haittakoodit.

Seuraavaksi pohdittavana oli käyttäjän varmistus. Varmistuksen pääajatuksena oli, että varmistetaan käyttäjän olevan oikea ihminen eikä botti. Ongelmana oli, että botti ja miksei myös ihminenkin voi helposti pommittaa tietokannan täyteen turhia tunnuksia ja sitä kautta vaikuttaa sovelluksen toimintaan. Päätinkin tehdä sovellukseen sähköpostivarmistuksen, jonka avulla käyttäjä voi varmentaa tunnuksensa sovellukseen. Käyttäjä saa sähköpostissa linkin, johon on liitettynä uniikki avain, jota käytetään verifikoinnissa. Käyttäjä siirtyy saamansa linkin kautta sovellukseen, jossa suoritetaan verifikointi ja tunnusten liittäminen sovellukseen, jos avain oli oikein. Käyttäjällä on 24 tuntia aikaa käyttää avain, jonka jälkeen avain vanhentuu ja tunnukset täytyy luoda uudelleen.

Varmistuksen tekoon päätin luoda uuden taulun tietokantaan, jonne käyttäjän tunnukset sijoitettaisiin väliaikaisesti, kunnes käyttäjä on sähköpostin kautta varmentanut tunnuksensa. Taulun nimeksi tuli verification, jonne tullaan lisäämään 4 riviä, jotka ovat varmistuskoodi ”token”, varmistuskoodin aika ”time”, tunnus ”username” ja salasana ”password”. Token-riville tulee PHP:llä luotu uniikki avain, jolla todennetaan käyttäjän oikeellisuus ja sille annoin varchar(255)-tyypin, joka riittää hyvin koodin turvaamiseen. Time-riville asetetaan PHP:ltä tullut serverin kellonaika sekunteina ja sille annetaan sen tarvitsema tyyppi int(10). Username ja Password-riveille tulevat käyttäjän antama tunnus ja salasana ja ovat samaa tyyppiä kuin users-taulussakin.

#	Nimi	Tyyppi	Aakkosjärjestys	Attribuutit	Tyhjä	Oletusarvo
1	**token**	varchar(255)	latin1_swedish_ci		Ei	*None*
2	**username**	varchar(50)	latin1_swedish_ci		Ei	*None*
3	**time**	int(10)		UNSIGNED	Ei	*None*
4	**password**	varchar(255)	latin1_swedish_ci		Ei	*None*

Kuvio 30. Verification-taulun rakenne

Kuviossa 30 nähdään varmennus-taulun rakenne. On tärkeää, että usernamen-rivin ja password-rivin tyypit ja tyypin pituus ovat samat kuin users-taulussa, koska tällöin tunnus tai salasana saattaa muuttua ja eikä tällöin toimi oikein.

Seuraavaksi siirrytään itse PHP-puolelle. Lomakkeen painalluksen jälkeen liittymis-PHP:ssä luodaan uniikki avain eli token, joka liitetään muuttujana osaksi linkkiä, joka lähetetään sähköpostilla käyttäjälle. Linkin URL menee näin: ”http://fnote.sadcompany.net/verify.php?token=”.$combine”, jossa lähetetään GET-metodilla token-muuttuja varmennus-PHP:lle. Avaimen tekoon voidaan käyttää paljon erilaisia salausmenetelmiä ja niillä voidaan myös luoda uniikki merkkijono. Itse käytän kolmea eri salausmenetelmää avaimen luontiin mm. md5-salausta. Nämä kolme salausmenetelmän merkkijonoa liitetään toisiinsa, joista syntyy itse luotu uniikki avain. Keksin oman ratkaisun, kuinka tätä uniikkia avainta käytetään varmentamisessa. Ratkaisussani vain tietty osa avaimesta tarkistetaan ja tällöin hakkerilla on ainakin hieman pähkäiltävää mikä osa avaimesta tarkistetaan. Ainoastaan tämä tarkistettava osa lisätään tietokantaan.

Salasana myös salataan samalla, kun se lähetetään verification-taululle. Salasanan salauksessa paras keino olisi ollut käyttää PHP:n uutta password_hash()-komentoa, joka on melko hyvin suojattu mutta komento tarvitsee PHP-versio 5.5:n ja käyttämäni palvelin on vielä 5.3, joten en voinut vielä opinnäytetyötäni tehdessäni sitä käyttää. Sen sijaan jouduin käyttämään yleistä salausmenetelmää, joka on nykyään todella helppo murtaa mutta tulen vaihtamaan password_hash-salauksen heti kun palveluntarjoajani päivittää PHP-version uudempaan. Tarkistettava osa token-muuttujasta lisätään verification-tauluun käyttäjän tunnuksen ja salasanan kanssa ja lisäksi tallennetaan palvelimen sen hetkinen aika

sekunteina, jotta voidaan määrittää varmennus-PHP:ssä, onko käyttäjä varmentanut tilinsä 24 tunnin sisällä.

Lopuksi luodaan varmennus-PHP, johon käyttäjä siirtyy sähköpostiin tulleen varmennuslinkin kautta. Aivan PHP:n alussa tarkistetaan, onko token-muuttujalla arvo ja onko muuttujassa ei-sallittuja merkkejä käyttämällä preg_match()-komentoa. Tässä kohtaa pystytään estämään muuttuja, jos muuttujassa on haittakoodia. Jos muuttujan pituus tai arvo on eri kuin säännöt sanovat, tällöin ilmoitetaan käyttäjälle, että muuttujassa on vieraita merkkejä, muuttujasta puuttuu merkkejä tai pituus on väärä. Mielessäni kävi, että tällaisissä tapauksissa laitettaisiin käyttäjän IP-osoite ja muuttujan arvo talteen jonkinlaiselle error-taululle, jonka avulla pystyttäisiin estämään IP-osoite, jos hakkerointia löytyi muuttujasta mutta tämä jää vielä kehitysideaksi tässä kohtaa. Jos muuttuja oli sääntöjen mukainen, silloin muuttujan arvo otetaan vastaan PHP:lle käytettäväksi.

```
if (isset($_GET["token"]) && preg_match('/^[0-9A-F]{80}$/i', $_GET["token"]))
{
    $combine = $_GET["token"];
}
else {
    die("Implemented text found or token value has been changed. Valid token not provided.");
}
```

Kuvio 31. Varmennus-PHP:n URL:in tarkistus

Kuten nähdään kuviossa 31, jos muuttuja on sääntöjen mukainen se lisätään PHP-muuttujan arvoksi mutta muutoin varmennus lopetetaan die-komennolla ja ilmoitetaan siitä käyttäjälle.

Tämän jälkeen luodaan SQL-kysely, jolla haetaan token-muuttujan avulla tiedot verification-taulusta. Jos yhtään riviä ei löydy tietokannasta, PHP:n suoritus keskeytyy ja käyttäjälle ilmoitetaan asiasta. Jos yksi rivi löytyy, tällöin tarkistetaan vielä, onko token jo vanhetunut. Tämä tehdään luomalla ensiksi muuttuja nimeltä expire, jolle annetaan arvoksi ”86400”, joka on 24 tuntia sekunteina. Sen jälkeen if-lauseessa vähennetään nykyinen palvelimen aika tietokannaan tallennetusta ajasta. Jos tulos on suurempi kuin 24 tuntia eli 86400 sekuntia, tällöin token on vanhentunut ja siitä ilmoitetaan käyttäjälle.

```
$expire = 86400;
$user = "";
$password = "";
$time = "";
        while($form = mysqli_fetch_array($tj, MYSQLI_ASSOC))
        {
                $user = $form["username"];
                $password = $form["password"];
                $time = $form["time"];
        }

        if ($_SERVER["REQUEST_TIME"] - $time > $expire)
        {
                $sql4 = "DELETE FROM verification WHERE token = '$token'";
                if(!mysqli_query($connection,$sql4))
                {
                die("Valid token not provided");
                }
                die("Token has expired.");
```

Kuvio 32. Tokenin vanhentumisajan tarkistus

Kuviossa 32 nähdään if-lauseen rakenne tarkistuksessa. Jos palvelimen nykyinen aika miinus tietokannasta saatu aika on suurempi kuin 86400 sekuntia eli 24 tuntia, tällöin token on vanhentunut. Vanhentunut token poistetaan samalla verification-taulusta.

Jos token ei ole vanhentunut, käyttäjä on varmentanut onnistuneesti tunnuksensa sovellukseen ja tunnukset lisätään itse users-taulun ja käyttäjä pystyy nyt käyttämään tunnuksiaan välittömästi. Käytetty token poistetaan samalla verification-taulusta. Nyt uudistettu sovellukseen rekisteröityminen oli valmis ja ryhdyin tekemään loppusilauksena julkista rajapintaa muistiinpanoille.

4.3.4 Julkisen rajapinnan lisääminen muistiinpanoihin

Päätin lisätä julkisen rajapinnan, jotta käyttäjät voivat jakaa omia muistiinpanojaan ja sitä kautta omia ajatuksiaan ja ideoita muiden käyttäjien kanssa.

Rajapinnan luominen alkoi lisäämällä articles-tauluun yksi lisärivi nimeltä public. Tämä rivi määrittelee onko muistiinpano kaikille näkyvissä vai ei ja siihen tullaan syöttämään true tai false-arvot. Tyypiksi riville tuli boolean. Tämän rivin avulla voidaan hakutilanteessa tarkistaa, mitä muistiinpanon ovat julkisia ja samalla voidaan näyttää ne hakutuloksissa eri sarakkeessa. Samalla tein myös ulkoasuun pienen muutoksen ja lisäsin public-nimisen checkboxin sovelluksen alareunaan.

Ensiksi loin muistiinpanon asettamisen julkiseksi. JavaScriptissä lähetetään PHP:lle väliaikainen id, jolla tunnistetaan muistiinpano ja public-checkboxin arvo ja käyttäjän tiedot menevät suoraan Session-muuttujan kautta PHP:lle. Tämän jälkeen PHP:llä päivitetään articles-tauluun joko true tai false riippuen mikä checkboxin arvo on. Tämän jälkeen muistiinpanon pitäisi olla näkyvissä muille käyttäjille hakutoiminnossa mutta ensiksi hakutoimintoa täytyy myös päivittää.

Lopuksi päivitetään itse hakutoiminnon PHP-tiedosto. Haku-PHP:ssä joudutaan nyt kaksi erilaista SQL-kyselyä, että saadaan hakutulokset eriteltyä omiin ja julkisiin muistiinpanoihin. Uusi SQL-kysely menee niin, että tarkistetaan articles-taulusta kaikki rivit, joissa public-rivi on true. Tällä saadaan tulostettua julkiset rivit tietokannasta.

```
$sql2 = "SELECT * FROM articles WHERE header LIKE '%$search%'
 AND public = '$cbvalue' ORDER BY time DESC LIMIT 0,5";
```

Kuvio 33. SQL-kysely julkisten muistiinpanojen hakuun

SQL-kysely on melko samanlainen kuin pääkyselykin mutta nyt haetaan public-rivin tiedot eikä author-riviä. Kuviossa 33 näkyy myös ”ORDER BY time DESC LIMIT 0,5”-erittely, jolla laitetaan muistiinpanot aikajärjestykseen uusin ensimmäiseksi ja rajoitetaan hakutulosten määrä maksimissaan viiteen. Tämän jälkeen luodaan julkisille muistiinpanoille oma sarake hakutuloksiin ja tulostetaan tulokset samalla tavalla kuin omatkin muistiinpanot mutta sillä erotuksella, että sarakkeen muistiinpanot ovat kaikki julkisia. Nyt hakutuloksissa näkyvät kaikki omat muistiinpanot sekä julkiset muistiinpanot. JavaScriptiin lisäsin vielä uuden funktion, joka hakee muistiinpanoa haettaessa public-checkboxin arvon tietokannasta.

Kuvio 34. Omien ja julkisten muistiinpanojen hakutulokset

Kuviossa 34 nähdään eri sarakkeet omille muistiinpanoille ja julkisille muistiinpanoille. Testi-muistiinpano ei ole julkinen ja täten ei näy julkisissa hakutuloksissa.

4.3.5 Kolmannen version testaus ja tulosten tulkinta

Kolmatta versiota testasin erityisen paljon ja yritin luoda mahdollisimman paljon eri virhetilanteita. Näitä aluksi löytyikin melko paljon mutta tarkemmin asiaa tutkittuani huomasin virheiden olevan onneksi vain merkkien puuttumisia ja virhelyöntejä itse koodissa. Kaikki uudet lisätyt ominaisuudet tuntuivat toimivat sulavasti ja haittakoodikaan ei tunnu menevät läpi tekstikentistä, mikä oli alunperin tarkoituskin. Rekisteröityminen onnistui virheittä ja julkinen rajapinta toimii nopeasti. Ulkoasukin tuntuu toimivan kaikissa yleisimmissä selaimissa moitteetta.

Olin erittäin tyytyväinen sovelluksen kolmanteen versioon ja alkoikin tuntua siltä, että sovellus on pikkuhiljaa valmis. Sovelluksessa oli kaikki tässä vaiheessa haluamani ominaisuudet ja ulkoasukin alkoi jo kelpaamaan itsellenikin. Kuitenkin joudun todeta tässä vaiheessa, että jonkin verran korjattavaa on vielä mutta enää ei onneksi puhuta kovin suurista korjauksista. Yksi korjauksista tulee tähtäämään

itse tekstin tallennukseen, jossa jokaisen painalluksen jälkeen teksti lähetetään aina PHP:lle tallennettavaksi, joka lisää kuormitusta aikalailla palvelinpuolelle sekä JavaScriptin puolelle. Ainakin tähän ongelmaan ja pieniin virheisiin tulen puuttumaan sovelluksen neljännessä versiossa.

4.4 Neljäs versio

Ohjelma kokonaisuutena alkoi olla valmiina jo kolmannen version lopussa. Neljännen version oli tarkoitus lähinnä olla pienten bugien ja testauksessa ilmentyneiden puutteiden korjausta mutta testauksessa myös löytyi yksi isompi ”puute”, joka tarvitsi uuden ratkaisun. Ulkoasuun ei tässä versiossa enää tarvittu muutoksia.

4.4.1 Tekstin tallennuksen kuormituksen vähentäminen selain- ja palvelinpuolella

Testailtuani perinteisin tavoin ohjelmaa, niin aloin myös käyttää testauksessa Fiddler-nimistä ohjelmaa, jolla pystytään kartoittaa selainpuolen kaikkien JavaScript- ja PHP-tiedostojen liikenne palvelimelle. Fiddlerillä huomasin erään seikan, joka oli ”unohtunut” kokonaan ohjelmaa tehdessä. Kyseinen seikka oli selain- ja palvelinpuolen kuormitus tekstin tallennuksessa. Fiddlerillä huomasin, että jokaisella näppäimistön painalluksella lähetettiin PHP:llä teksti tallennettavaksi palvelimelle, joka luonnollisesti kuormittaa palvelinpuolta melkoisesti. Tämä seikka olisi pitänyt suoraan sanottuna olla ilmiselvää jo alusta alkaen mutta vasta tässä testausvaiheessa asia kirkastui itselleni kunnolla, kun näin konkreettisesti kuinka paljon PHP-kutsuja palvelimelle lähetettiin. Samalla myös selainpuoli kuormittuu, koska JavaScriptiä ajetaan saman verran kuin PHP-kutsujakin.

Ongelman ratkaisun keksin melko nopeasti mutta toteutusta en aluksi meinannut saada tehtyä. Yksinkertainen ratkaisuajatus oli se, että jos JavaScriptille tulee tietyllä aikavälillä saman funktion kutsu monta kertaa, niin JavaScript aloittaa funktion alusta ja sitten vasta kun aikaraja täyttyy niin suoritetaan funktio loppuun asti. Yritin itse kehittää ajastimen, joka nollaisi aina funktion suorituksen monella

painalluksella mutta vain tyydyttävin tuloksin. Sain funktion toimimaan loppujen lopuksi mutta testauksessa huomasin, että jos laitetaan vain yksi kirjain, niin ei funktiota suoriteta ollenkaan. Takerruin tähän ongelmaan jo liiankin pitkäksi aikaa ja lopulta päädyinkin etsimään Googlesta ratkaisua ajastimen käyttöön. Löysinkin ajastimen esimerkin www.lucadentella.it-sivustolta, joka oli juuri sellainen kuin tarvitsinkin. Ajastimen avulla sain vähennettyä funktioiden määrää huomattavasti.

```
var delay = (function () {
    var timer = 0;
    return function (callback, ms) {
        clearTimeout(timer);
        timer = setTimeout(callback, ms);
    };
})();

function headerSave()
{
delay(function(){
var headerinput = encodeURIComponent(document.getElementById("header").innerHTML);
var iid = tempid;
var xmlhttp = basicAjax();

xmlhttp.open("POST", "saveHeader.php", true);
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xmlhttp.send("header="+headerinput+"&iid="+iid);
},500);
}
}
```

Kuvio 35. Ajastimen rakenne ja käyttö tallennuksessa

Kuviossa 35 nähdään ajastimen toimintaperiaate. Ensin luodaan muuttuja nimeltä delay, joka toimii ajastimen kutsumuuttujana. Muuttujan sisälle tehdään funktio, jonne lisätään muuttuja nimeltä timer ja alustetaan se arvolla 0. Joka kerta, kun delay-muuttujaa kutsutaan, niin funktio tekee return function-toiminnon ja nollaa itsensä ja hakee määritellyn aikavälin funktiosta, joka ajastetaan käyttämällä delay-muuttujaa. Delay-muuttujan rakenne on sama kuin setTimeout-funktiollakin. Kuviossa esimerkkinä näkyy headerSave-funktio eli otsikon tallennus Tämän funktion koodi lisätään delay-muuttujan sisälle ja loppuun annetaan aikaväli, jolloin headerSave-funktio halutaan suorittaa. Nyt jokaisella näppäimistön painalluksella funktio nollaa ajastimen, jos painallus tulee nopeampaa kuin annettu aikaväli esimerkiksi kuviossa 35 näkyvä 500 millisekuntia eli 0,5 sekuntia.

Ajastin osoittautui erittäin toimivaksi ja se vähensikin suoritettavien funktioiden määrää ainakin 90 %. Samalla tekstin tallennus muuttui entistäkin vakaammaksi ja varmemmaksi, koska JavaScriptillä ja PHP:llä on enemmän aikaa suorittaa tallennusfunktioita.

4.4.2 Painikkeiden tuplapainallusten korjaaminen

Kun annoin ohjelman testattavaksi vanhemmilleni, yksi ongelma pomppasi heti esiin. Vanhemmillani on vieläkin se tapa, että kaikkia painikkeita ja kuvakkeita pitää painaa vähintään kaksi kertaa perätysten. Tämä aiheutti ohjelmassa sen, että uusi sivu luotiin kaksi kertaa ja muistiinpanoa yritettiin poistaa kaksi kertaa tuplapainallusten takia.

Onnekseni olin juuri löytänyt ohjelmaani ajastintekniikan, joka korjasi tämän ongelman yhtä nopeasti kun se oli ilmentynytkin. Käytin siis samaa ajastinta nappien painalluksissa kuin tekstin tallentamisessa eli kun tuplaklikataan painiketta, niin ensimmäistä tai tietyllä aikavälillä tapahtuvia painalluksia ei noteerata. Nyt jos painikkeita tuplaklikataan, uusia sivuja ei luoda useita tai poisteta samaa muistiinpanoa kahta kertaa. Tämän ongelman korjaaminen siis tapahtui ajastimen ansiosta erittäin vaivattomasti.

5 LOPPUSANAT

Tavoitteenani oli tutkia, miten saadaan luotua muistiinpano-ohjelma käyttäen Ajax-tekniikkaa. Toteutin käytännön osuuden käyttämällä uudenaikaisimpia tekniikoita Ajax:in lisäksi, jotka olivat HTML5, CSS3, JavaScript, MySQL ja PHP 5. Ennen toteutuksen tekoa minulla oli oppimispohjana kesällä 2013 tehty prototyyppi, josta sain hyvät alkukohdat opinnäytetyöhöni. Opinnäytetyön eri versioita tehdessä kuitenkin opin paljon syvemmin uusista eri koodaustavoista sekä kehityin ongelmanratkonnassa.

Opinnäytetyössäni tehty muistiinpano-ohjelman toteutus sujui kokoajan kehittyvään tahtiin. Kehityksen kulku meni alusta asti sujuvasti eteenpäin mutta eri versioiden välillä tuli paljon monenlaisia ongelmia vastaan, joiden ratkomiseen

kului välillä ehkä hieman liikaakin aikaa mutta ongelmista tuli opittua paljon. Toteutuksen teossa tuli käytettyä luonnollisesti paljon aikaa, koska tein ohjelman miltei puhtaalta pöydältä ja ilman kielikirjastoja tai valmiita sapluunoita. Eniten aikaa toteutuksessa vei JavaScript- ja PHP-funktioiden teko.

Opinnäytetyössäni valmistunut muistiinpano-ohjelma oli mielestäni hyvin onnistunut ja täytti kaikki asettamani tavoitteet ja vähän enemmänkin. Tekstin tallennus ja muistiinpanojen Ajax:in avulla toimi lopussa moitteettomasti. Tietoturvallisesti katsottuna ohjelma on korjattu kaikista yleisimmistä tietoturvaongelmista mutta tietoturva ei ole vielä läheskään täydellinen. Ohjelman ominaisuudet toimivat täysin kaikilla yleisimmillä selaimilla eli Chromella, Internet Explorerilla, Firefoxilla, Operalla ja Safarilla. Kuitenkaan selainversiot, jotka eivät tue uusimpia HTML5-tekniikoita kuten IE 8, eivät toimi kunnolla ohjelmassa mutta ohjelman tarkoitus ei ollutkaan olla yhteensopiva vanhoille tekniikoille vaan ohjelman käyttö vaatii uusien selainversioiden käytön.

Vaikka olin ohjelman lopputulokseen tyytyväinen, ilmoille jäi vielä paljon erilaisia kehitysideoita. Yksi tärkeimmistä kehitysideoista oli tekstin tallennuksen varmuuskopiointi esim. internet-yhteyden katkeamisessa kesken kirjoituksen. Tähän teinkin jo alustavan ratkaisun käyttämällä HTML5:ssa esiteltyä Local Storage-teknikkaa. Ratkaisussa voidaan tallentaa dataa evästeiden sijasta selaimen omaan tallennustilaan, joka on myös turvallisempi kuin evästeet. Idea oli, että teksti tallennetaan Local Storageen, kun internet-yhteys katkeaa. Tämä idea kaatui siihen, että Local Storage ei ole vielä hyvin tuettu selaimissa ja tallennettu data voidaan tyhjentää turhan helposti esim. evästeiden mukana, kun selain suljetaan. Tallennustilan määräkin oli turhan pieni, joten tämä ominaisuus jäi vielä toteuttamatta. Toisena tärkeänä kehitysideana oli muistiinpanon siirtäminen Word-tiedostoksi ja Word-tiedoston tuominen ohjelmaan. Tällöin saataisiin muutettua ajatuksia nopeasti Word-tiedostoksi tai tallennettua tärkeitä Word-tiedostoja suoraan ohjelmaan, joihin pääsisi käsiksi mistä tahansa selaimesta. Käyttäjänhallinta jäi vielä myös tekemättä, jossa käyttäjä voisi muokata omia yhteystietojaan. Viimeisenä tärkeänä kehitysideana on alisivujen luonti

mahdollisuus muistiinpanoille. Tällöin voitaisiin luoda esim. reseptit-muistiinpano, jolle annettaisiin eri reseptejä alisivuina.

Ajax-tekniikan tulevaisuus näyttää valoisalta ja myös onkin sitä jo. Suuret palveluntarjoajat, kuten Google ovatkin ottaneet jo Ajaxia laajalti käyttöön ja muut tulevat perässä. Ajaxilla onkin oikein käytettynä hyvät mahdollisuudet päästä lähelle työpöytäsovellusta ja sen toiminnallisuutta. Mutta pitää kuitenkin muistaa, että web-sovellusten ja eri tekniikoiden tulevaisuutta on vaikea ennustaa, koska uusia tekniikoita kehitellään jatkuvasti ja samalla sovellusten ominaisuudet muuttuvat mutta ainakin tällä hetkellä Ajax on jo yleisesti käytetty tekniikka ja selainten jatkuvasti parantuva tuki pitää Ajaxin lähivuosina varmasti pinnalla.

LÄHTEET

Can I Use 2014. Can I Use. WWW-dokumentti. http://caniuse.com/. Päivitetty 23.06.2014. Luettu 20.05.2014.

Duckett, Jon 2010. E-kirja. HTML, XHTML, CSS and JavaScript.

Evernote 2014. Evernote. WWW-dokumentti. https://evernote.com/intl/fi/. Päivityksestä ei tietoa. Luettu 21.05.2014.

Findmemyip 2013. Findmyip. WWW-dokumentti. http://fmbip.com/litmus/. Päivityksestä ei tietoa. Luettu 20.05.2014.

Ford, Jerry Lee 2008. E-kirja. Ajax Programming for the Absolute Beginner.

Hovi, Ari 2012. SQL-opas. Jyväskylä, Docendo.

Lucadentella 2013. Lucadentella. WWW-dokumentti. http://www.lucadentella.it/en/2013/06/17/javascript-aggiungere-un-ritardo-allevento-keyup/. Päivitetty 17.06.2013. Luettu 01.05.2014.

Microsoft 2014. Microsoft OneNote. WWW-dokumentti. http://office.microsoft.com/fi-fi/onenote/. Päivityksestä ei tietoa. Luettu 21.05.2014.

Ohjelmointiputka 2008. Opasarkisto Ajax. WWW-dokumentti. http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=ajax. Päivitetty 2008. Luettu 23.05.2014.

W3Schools 2014. W3Schools. WWW-dokumentti. http://www.w3schools.com/html/html5_intro.asp. Päivitetty 27.06.2014. Luettu 05.05.2014.

W3Techs 2014. W3Techs Survey. WWW-dokumentti. http://w3techs.com/. Päivitetty 01.04.2014. Luettu 20.05.2014.