RAMI NOROLAHTI

SELATTAVAN WEB-SISÄLLÖN KÄYTTÄJÄKOHTAINEN SUODATUS

Diplomityö

Tarkastajat: professori Hannu Jaakkola, assistant professor Sami Hyrynsalmi
Tarkastajat ja aihe hyväksytty Talouden- ja rakentamisen tiede-kunta- neuvostossa 24. helmikuuta 2017

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
NOROLAHTI, RAMI: Selattavan web-sisällön käyttäjäkohtainen suodatus
Diplomityö, 43 sivua
Maaliskuu 2017
Pääaine: Ohjelmistotuotanto
Tarkastajat: professori Hannu Jaakkola, assistant professor Sami Hyrynsalmi

Avainsanat: sisällönsuodatus, käyttäjäkohtainen suodatus, web-sisällön suodatus,selainlaajennus,suodatusjärjestelmä

Tämän diplomityön tarkoituksena oli suunnitella ja toteuttaa toimiva selattavan web-sisällön suodatusjärjestelmä ja käydä läpi aiheeseen liittyvää teoriaa sekä kuvata järjestelmän rakenne. Järjestelmän rakenteesta esitetään arkkitehtuuri, käyttöliittymät ja tärkeät prosessit. Työssä pohditaan myös järjestelmän tietoturvaa, julkaisua sekä järjestelmän jatkokehitysmahdollisuuksia.

Järjestelmä muodostuu Google Chrome-selaimeen asennettavasta laajennuksesta (asiakasosa) sekä palvelinosasta. Selainlaajennukseen kirjaudutaan sähköpostiosoitteella ja salasanalla. Uusi käyttäjä voi rekisteröity käyttäjäksi selainlaajennuksen rekisteröintinäkymästä. Selainlaajennus tuo käyttäjälle web-selaimessa näkyvään näkymään hallintakerroksen, joka mahdollistaa suodatettavan sisällön valinnan sekä suodattamisen. Suodatettava sisältö määrittyy käyttäjän määrittelemänä.

Tämän tutkimuksen lopputuloksena on saatu toimiva ja julkaistavissa oleva sisällönsuodatusjärjestelmä.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master’s Degree Programme in Information Technology
NOROLAHTI, RAMI: User spesific browsable web content filtering
Master of Science Thesis, 43 pages
March 2017
Major: Software Engineering
Examiner: Professor Hannu Jaakkola, assistant professor Sami Hyrynsalmi

Keywords: content filtering, user specific filtering, web content filtering, browser extension, filteringsystem

The purpose of this master's thesis was to design and implement fully operating content filtering system for web pages. The purpose was also to write about theory of subject and describe the system's structure. Description of the structure includes architecture, user interfaces and description of important processes. The thesis also includes pondering about system's security, publishing and further development.

The system consists Google Chrome extension (client) and server. User logs in Chrome extension with email address and password. Unregistered new user can fill registration form at extension registeration view. The extension brings a control layer to user. With control layers functions user can select web pages content to be filtered out.

Result of this thesis is a system that could be published.

ALKUSANAT

Tämän työn idea syntyi 2015, lukiessani osakesijoittamiseen keskittyvää keskustelupalstaa ja huomasin että keskustelussa toistuivat samat, asian kannalta epäolennaiset tai turhat viestit. Ensivaiheessa yritin etsiä valmiita ratkaisuja näiden viestien suodattamiseksi. Pitkän etsimisen jälkeen päädyin itse toteuttamaan tarvitsemani ratkaisun. Saatuani valmiiksi alkeellisen proof of consept version jätin ohjelman kehittämisen, muiden kiireiden vuoksi. Syksyllä 2016 miettiessäni itselleni diplomityön aihetta, päädyin herättämään vanhan idean. Tavoitteeksi asetin järjestelmän kehittämisen sellaiselle tasolle, joka mahdollistaisi järjestelmän julkaisun.

Tämän työn tekeminen on ollut minulle erittäin opettavainen kokemus sekä suuri etsintäprosessi. Tämän työn myötä osaamiseni eri aihealueilla on kasvanut runsaasti.

Lopuksi haluan myös kiittää professori Hannu Jaakkolaa sekä professori Sami Hyrynsalmea hyvästä ohjauksesta ja asiantuntevista kommenteista työn loppuun saattamiseksi.

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

CSS	Cascading Style Sheets, web-sivujen tyylityskieli
Tekstinelementti	Web-sivun osa, joka pitää sisällään ihmisluettavaa tekstiä
HTML	Lyhenne sanoista Hypertext Markup Language, avoimesti standardoitu kuvauskieli
Asiakas-palvelin arkkitehtuuri	Arkkitehtuuri, jossa kommunikaatio perustuu asiakkaan ottamaan yhteydenottoon eikä esimerkiksi ennaltamääriteltyyn kaksisuuntaiseen yhteyteen
Selainlaajennus	Web-selaimeen asennettava lisäohjelma
Google Chrome	Googlen ilmainen web-selain
Tiivistealgoritmi	Algoritmi joka muodostaa merkkijonosta tai muusta tiedosta määrämittaisen tiivisteen

1 JOHDANTO

Tämän diplomityön tarkoituksena on toteuttaa Google Chrome-selaimessa käytettävä sisällönsuodatuslaajennus. Sisällönsuodatuksella tarkoitetaan ihmisluettavan tekstin ehdollista suodatusta. Maailmalta löytyy useita ratkaisuja sähköpostin ja mainoksien suodattamiseen sekä myös monia suosittelujärjestelmiä eri tarkoituksiin. Nämä ratkaisut eivät kuitenkaan toimi mallissa, jossa käyttäjä haluaa suodattaa tekstisisältöä valitsemiltaan web- sivuilta oman mieltymyksensä mukaan. Valmiit ratkaisut toimivat yleisellä tasolla.

Tämän diplomityön tavoitteet ovat:
- Esittää aiheeseen liittyvää teoriaa
- Kuvata järjestelmän arkkitehtuuri
- Esittää ja selittää järjestelmän keskeiset prosessit
- Kuvata järjestelmän käyttöliittymät ja näkymät
- Pohtia järjestelmän tietoturvaa
- Kuvata valmiin järjestelmän mahdollinen julkaisuprosessi
- Pohtia järjestelmän mahdollisia jatkokehitysmahdollisuuksia
- Toteuttaa toimiva web- sivujen sisällönsuodatusjärjestelmä

Tässä työssä sisällönsuodatusjärjestelmällä tarkoitetaan järjestelmää joka mahdollistaa web- sivujen tekstin sisällön suodattamisen.

Teoriaosioissa (luvut 2, 3, 4 ja 5) käydään läpi aiheeseen liittyvää teoriaa. Osiossa kuvataan työn aihealueeseen liittyvää teoriaa ja sekä asiaan pohjustavaa omaa pohdintaa ja johtopäätöksiä. Teoriaosion jälkeen lukijalla pitäisi olla tarpeellinen kuva aiheeseen liittyvästä teoriata.

Arkkitehtuuriosiossa (luku 6) käydään läpi järjestelmän arkkitehtuuriratkaisuja ja rakennetta. Arkkitehtuuriosiossa kuvataan järjestelmän toteutuksessa käytetyt tekniikat sekä perustellaan, miksi niihin on päädytty. Osion lukemisen jälkeen lukijalla pitäisi täydellinen kuvat järjestelmän arkkitehtuurista.

Prosessiosiossa (luku 7) käydään läpi järjestelmän tärkeimmät prosessit, sekä kuvataan niiden toiminta yksityiskohtaisesti. Osion on tarkoitus antaa lukijalle täydellinen kuva järjestelmän prosesseista.

Käyttöliittymäosiossa (luku 8) kuvataan järjestelmän käyttöliittymät niiden hahmotelmat sekä niiden viimeistellyt lopputulokset. Osiossa selitetään myös, miten on päädytty mihinkin käyttöliittymä ratkaisuun.

Tietoturvaosiossa (luku 9) käydään läpi mahdollisia haavoittuvuuksia sekä ratkaisuja joilla voidaan pienentää riskiä hyökkäyksille ja uhille. Osiossa on tarkoituksena selittää myös käytössä olevia tietoturvaratkaisuja sekä miten ne vaikuttavat järjestelmän tietoturvaan. Osiossa kuvataan osittain myös ratkaisuja, joita ei tämän työn puitteissa vielä toteuteta, mutta mahdollisen jatkokehityksessä tulevat ajankohtaisiksi.

Julkaisuosiossa (luku 10) kuvataan järjestelmän yksi mahdollinen julkaisutapa. Osiossa kuvataan niin palvelinosan kuin Google Chrome-selainlaajennuksen julkaisu.

Jatkokehitysosiossa (luku 11) pohditaan järjestelmän erilaisia kehitysmahdollisuuksia. Nämä jatkokehitys mahdollisuudet ovat järjestelmään lisättäviä ominaisuuksia, jotka mahdollistavat moni- ja helppokäyttöisemmän apuohjelman.

2 TUTKIMUSMENETELMÄ

Diplomityön tutkimuksellisen laadun varmistamiseksi, tässä osiossa määritellään työssä käytettävä tutkimusmenetelmä sekä kuvataan tutkimuksen oleelliset pohjatiedot.

2.1 Suunnittelutieteellinen tutkimusote

Tämän työn tutkimusmenetelmänä käytetään suunnittelututkimusmenetelmää (englanniksi Design-Science). Design-science-menetelmällä on juuret tekniikassa ja "keinotekoisten tieteissä". Se on pääasiassa ongelmanratkaisumalli. Se pyrkii luomaan keksintöjä jotka määrittelevät ideoita, käytäntöjä, teknisiä ominaisuuksia, ja tuotteita joiden kautta analyysi, suunnittelu, toteutus, hallinta ja tietojärjestelmien käyttö voidaan tehokkaasti toteuttaa (Hevner et al. 2004).

Design-Science menetelmän lopputuloksena on IT-artefakti. IT-artefaktit pyrkivät ratkaisemaan jonkin organisaation ongelman. Artefaktit ovat rakenteellisessa muodossa ja vaihtelevat ohjelmistosta, muodolliseen logiikkaan, eksaktiin matematiikkaan tai epävirralliseen luonnollisen kielen kuvaukseen. Tämä mahdollistaa lopputulokselle monet erityyppiset kvantitatiivista artefaktin arviointimenetelmät. Sopivat arviointimenetelmät pitävät sisällään optimoinnin todistamisen, analyyttisen simuloinnin ja vaihtoehtoisten menetelmien kvantitatiivisen vertaamisen (Hevner et al. 2004).

Tutkimusmenetelmään on määritetty 7 suuntaviivaa (Hevner et al. 2004). Seuraavaksi kerron näistä 7 suuntaviivasta tarkemmin.

Suuntaviiva 1. Lopputuloksena artefakti:
Tutkimusmenetelmän lopputulokseksi pitää valmistua hyödyllinen IT artefakti.

Suuntaviiva 2. Ongelman oleellisuus:
Lopputuloksena syntyvän ratkaisun täytyy olla oikeasti tärkeä ja relevantti ongelman ratkaisun kannalta.

Suuntaviiva 3. Lopputuloksen arviointi:
Lopputuloksen laatu ja tehokkuus täytyy voida osoittaa hyvin suoritetuilla arviointimetodeilla.

Suuntaviiva 4. Tutkimus luo jotain uutta:
Tehokkaan tutkimuksen pitää antaa jotain uutta lopputuloksen osa-alueella.

Suuntaviiva 5. Tutkimuksen täsmällisyys:
Tutkimuksen täytyy tukeutua täsmällisiin metodeihin, sekä toteuttamisessa että arvioinnissa.

Suuntaviiva 6. Lopputulos on hakuprosessin tulos:
Tehokkaimman lopputuloksen löytämiseksi täytyy käyttää kaikkia mahdollisia lähteitä, jotta kaikki vaatimukset voidaan täyttää, kuitenkin pysyen ongelman osa-alueella. Lopputulosta voi täten pitää hakuprosessin tuloksena.

Suuntaviiva 7. Tutkimuksen tuloksien julkaisu:
Lopuksi tutkimuksen tulokset täytyy välittää asian kuuluville henkilöille.

Nämä suuntaviivat määrittävät tälle työlle hyvät raamit ja näitä suuntaviivoja pyrin tässä työssä myös tarvittavalla tasolla noudattamaan.

2.2 Ongelma ja alkutilanne

Tämän tutkimuksen ratkaistavana ongelmana on, miten toteutetaan web:iä selaavalle peruskäyttäjälle mahdollisimman helppokäyttöinen mutta kuitenkin hyvät työkalut tarjoava sisällönsuodatusjärjestelmä.

Järjestelmän toiminta pohjautuu tekstinluokitteluun. Tekstinluokittelun saralla maailmalta löytyy monenlaisia ratkaisuja. Näitä ratkaisuja ovat erilaiset suosittelujärjestelmät, roskapostinsuodatusjärjestelmät, keskustelupalstojen moderointibotit, sisältöä internetistä hakevat botit sekä dokumenttienluokittelu järjestelmät. Kaikki nämä ratkaisut toimivat siis samalla kentällä ja saman ongelman parissa. Tämän työn kohteena oleva järjestelmä eroaa aivan täysin osasta edellä mainittuja järjestelmiä mutta sivuaa myös joidenkin toimintaa osittain. Eniten samankaltaisuutta järjestelmällä on suosittelujärjestelmien sekä moderointibottien kanssa. Samankaltaisuus näiden järjestelmien/ratkaisuiden välillä jää hyvin pieneksi ja näitä eroja pyrin seuraavaksi avaamaan.

2.2.1 Suosittelujärjestelmät

Suosittelujärjestelmät toimivat web-sivuistolla ehdottaen käyttäjälle sisältöä joista hän voisi olla kiinnostunut. Suositusjärjestelmät perustavat ehdotuksensa käyttäjän aiempaan toimintaan. Jotkin järjestelmät pyrkivät luokittelemaan käyttäjiä ja ehdottamaan samassa luokassa oleville henkilöille sisältöä, jota muut luokkaan kuuluvat ovat selanneet (Wikipedia 2016).

2.2.2 Moderointibotit

Moderointibotit toimivat www-palveluissa (keskustelupalstat ja sosiaalinen media) ja piilottavat näkyvistä niihin kirjoitettua selvää vihapuhetta ja häirintää.

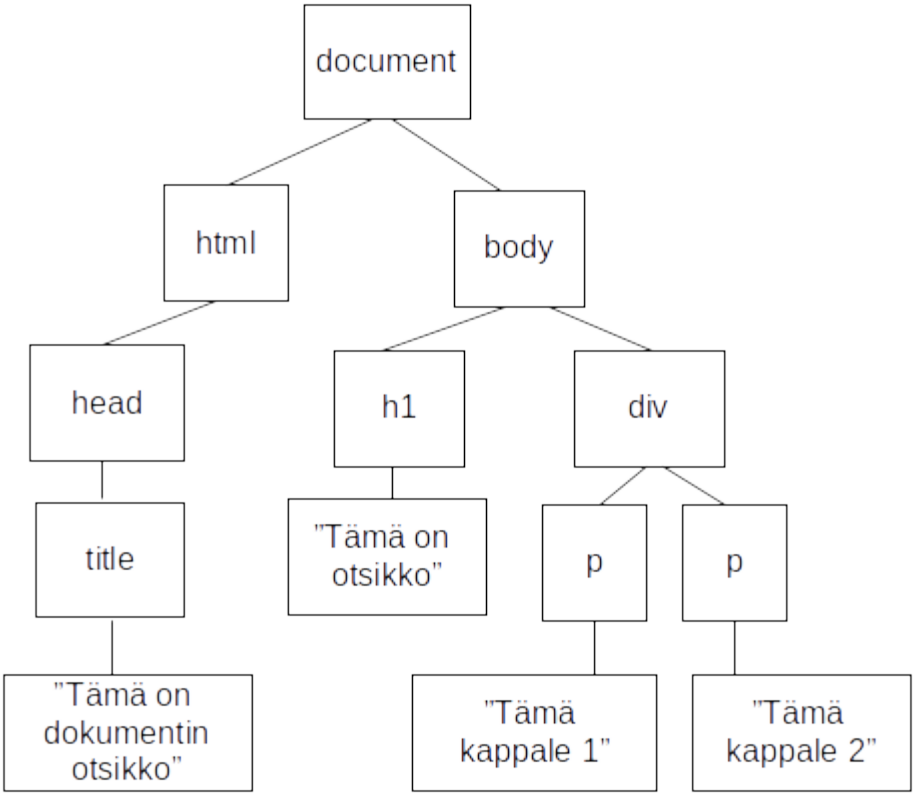
2.2.3 Eroavaisuudet

Yllä mainitut ratkaisut ovat toimivia ja niitä kehitetään monen yrityksen toimesta. Tämän järjestelmän on taas tarkoitus toimia käyttäjästä nähden näitä järjestelmiä korkeammalla tasolla ja lähempänä käyttäjää. Tämän järjestelmän tarkoitus on mahdollistaa käyttäjälle henkilökohtainen suodatusprofiili, joka toimii sivustorajoista huolimatta käyttäen samaa suodatuskaavaa. Tämä mahdollistaa sen että kohdesivustolla ei välttämättä ole minkäänlaisia suodatusmekanismeja muiden käyttäjien julkaisemaan tietoon, mutta tietoa voidaan silti suodattaa käyttäjän omasta toimesta. Tämän ratkaisun on tarkoitus myös mahdollistaa se, että rajattu tieto on kuitenkin käytettävissä helposti, jos käyttäjä kokeekin tiedon tarpeelliseksi.

3 SELATTAVA WEB-SISÄLTÖ

Tämän työn ydinasia on selattavan web-sisällön tulkinta ja muokkaus eli tapauksessa sisällönsuodatus. Web-sisällöllä tarkoitetaan tässä työssä ihmiselle luettavissa olevaa tekstipohjaista ja web-selaimella tarkasteltavaa tietoa. Selattavalla tiedolla tarkoitetaan kaikkea mahdollista rakenteellista muotoon tuotua tekstitietoa. Tämä tieto voi olla niin uutissivustojen, keskustelupalstojen, blogien jne. tekstisisällöt. Koska näissä sivustoissa voi olla paljon tietoa, joka ei lukijaa kiinnosta tai miellytä ja toistavat tiettyä kaavaa, niin voi olla tarpeellista piilottaa käyttäjältä nämä tiedot.

Tämän työn perusajatus on, että web-selain pystyy muodostamaan DOM-puurakenteen kaikista yllä mainituista tiedoista, joista halutaan poimia tai piilottaa osia.



Kuva 1: Yksinkertainen DOM puurakenne

DOM

DOM (Document Object Model/suom. dokumenttioliomalli) on alusta- ja kieliriippumaton liittymä, joka mahdollistaa selainpään skriptien päivittää dokumenttien sisältöä, rakennetta ja tyylejä. (W3C 2016). DOM -liittymä käsittelee HTML-, XHTML- ja XML-dokumentteja puurakenteena. Puurakenteessa jokainen solmu esittää jotain dokumentin osaa (Kuva 1).

4 SUODATETTAVAN TIEDON VALINTAMEKANISMIT DOM-PUUSTA

Kuten edellisessä osiossa todettiin, web-sivut ovat rakenteellisia dokumentteja, joista muodostetaan DOM- puurakenne. Dokumentti voi muodostua monista erityyppisistä elementeistä. Jotta näiden eri elementtien joukosta voitaisiin valita oikeat elementit, täytyy pohtia menetelmiä, joilla valinta voitaisiin toteuttaa. Oikeat elementit ovat ne jotka pitävät sisällään tekstiä, ja joita voidaan käyttäjän toimesta määritellä suodatuksen kohteeksi. Seuraavaksi on tarkoitus kuvata ja arvioida eri menetelmiä, joilla saa mielestäni parhaiten löydettyä oikeat elementit.

4.1 Valintamenetelmät

Oikeiden elementtien valintaan voidaan käyttää monia menetelmiä. Menetelmät voivat olla joko täysin käyttäjälähtöisiä, automaattisia tai näiden väliltä. Käyttäjälähtöisessä valinnassa käyttäjä voi itse valita web-sivulta elementin, johon suodatus kohdistuu. Käyttäjän valittua elementin, asiakasohjelman logiikka merkitsee sivun muut samat ominaisuudet (esim. CSS-luokkamääritelmä ja elementin nimi) omaavat elementit suodatuksen kohteeksi. Käyttäjälähtöinen menetelmä voidaan toteuttaa monella tavalla ja seuraavaksi kuvaan muutamia omaan pohdintaani perustuvia menetelmiä.

Käyttäjäystävällisin valintamenetelmä

Käyttäjäystävällisin ja selkein valintamenetelmä on se, että käyttäjä painaa valintatoiminnon aktivointipainiketta, jonka jälkeen käyttäjä voi valita sivulta elementin, johon suodatus kohdistuu. Teknisesti tämä on hankalaa, koska elementin valinta edellyttää, että sivuston normaalia toimintalogiikkaa muokataan niin, että hiiren painikkeen painamisen havaitseva tapahtumankäsittelijä(t) (eng. Event handler) kaapataan toiminnon ajaksi. Toiminnon suorittamisen jälkeen tapahtuman käsittelijät täytyisi palauttaa ennalleen. Koska web-sivut ovat toteutettu eri tavalla, tapahtumankäsittelijöihin kajoaminen voisi estää sivun normaalin käytön.

Sivustoystävällinen valintamenetelmät

Jotta selattavan web-sivun toimintalogiikkaan ei jouduta kajoamaan liikaa, tekstielementtien valinta voidaan toteuttaa lisäämällä sivulle valintakontrollit eli painikkeet. Va-

lintapainikkeiden lisääminen tuo web-sivun oman rakenteen päälle uuden kerroksen. Painikkeet lisätään ennalta määritetyn logiikan mukaisesti kaikille tekstiä sisältäville elementeille. Tämän tavan haittapuolena voi olla sivunäkymän täyttyminen valintapainikkeilla tapauksissa, joissa elementtejä on käytetty runsaasti. Koska valintamekanismia on tarkoitus oikeassa käytössä käyttää vain muutamia kertoja jokaisella sivustolla, niin haitta ei ole merkittävä.

Automaattiset valintamenetelmät

Automaattiset valintamenetelmät ovat menetelmiä, joilla sivulta etsitään itsenäiset tekstikokonaisuudet sisältävät elementit, ilman että tarvitsematta käyttäjän vuorovaikutusta. Automaattisilla valintamenetelmillä mallissa löydettyjen elementtien ominaisuudet (esim. CSS-luokkamääritelmä ja elementin nimi) tallennetaan sivustokohtaiseen profiiliin.

Käytettävä valintamenetelmä

Diplomityötä tehdessä täytyy huomioida käytettävissä olevat resurssit. Automaattinen mekanismi olisi käyttäjän kannalta helpoin, ja lisäisi käytettävyyttä, mutta toimivan heuristiikan kehittäminen tämän diplomityön puitteissa on pois suljettu. Epätäydellisesti toimiva automaattinen valinta voisi pahimmassa tapauksessa tehdä suodattimesta käyttökelvottoman joillakin web-sivustoilla. Tämän vuoksi päädyin oman määritykseni mukaiseen sivustoystävälliseen mekanismiin.

Jotta valittua mekanismia/mallia voitaisiin käyttää, täytyy tutkia, mitkä HTML-elementtityypit pitävät yleisemmin sisällään itsenäistä tekstisisältöä. Seuraavaksi tarkoitukseni on selvittää edellä mainittu ongelman.

4.2 Tekstielementit

HTML-standardissa on monia elementtityyppejä, jotka voivat sisältää tekstiä, mutta näitä kaikkia ei kannata asettaa käyttäjän valittavaksi käytettävällä menetelmällä, koska se voisi tehdä käyttöliittymästä sekavan. Elementtien määrittämiseksi päätin käydä läpi useita web-sivuja ja kerätä elementtien käytöstä tilastoa. Tarkoitukseni oli tarkastella suomenkielisiä sekä ulkomaisia uutis- ja keskustelusivustoja sekä keskittyä etsimään vain niitä elementtejä, jotka pitävät sisällään selvän itsenäisen tekstikokonaisuuden.

	Elementti	article	h1	h2	div	tr	li	p
Sivu								
1. Uutissivu		1	0	1	0	0	1	0
2. Uutissivu		1	1	0	0	0	0	0
3. Uutissivu		1	0	0	0	0	1	0
4. Uutissivu		1	0	0	0	0	1	0
5. Uutissivu		0	0	0	0	0	0	0
6. Uutissivu		0	0	0	1	0	1	0
7. Uutissivu		1	0	0	0	0	1	0
8. Uutissivu		1	0	0	0	0	0	0
9. Uutissivu		1	0	0	0	0	0	0
10. Uutissivu		1	0	0	0	0	0	0
11. Uutissivu		0	0	0	1	0	0	0
12. Uutissivu		0	0	0	1	0	0	1
13. Uutissivu		1	0	0	0	0	0	0
14. Uutissivu		1	0	0	0	0	0	0
15. Keskustelu		0	0	0	0	1	0	0
16. Keskustelu		0	0	0	1	0	1	0
17. Keskustelu		1	0	0	0	0	1	0
18. Keskustelu		0	0	0	0	0	1	0
19. Keskustelu		0	0	0	0	0	1	0
20. Keskustelu		0	0	0	1	0	0	0
Yhteensä		**11**	**1**	**1**	**5**	**1**	**9**	**1**
% sivuista		**38%**	**3%**	**3%**	**17%**	**3%**	**31%**	**3%**

Kuva 2: Käytetyimmät HTML-elementit jotka pitivät sisällään selvästi itsenäisiä tekstiosiota

Tutkin kahdenkymmenen sivuston HTML-sivurakenteet, ja keräsin elementtityypit, jotka pitivät sisällään selvästi omat itsenäisen tekstiosionsa (Kuva 2). Kuvassa 2 on listattu kaikki tutkittujen web-sivujen elementtityypit, sekä merkitty numero 1 tai 0 kuvaamaan sen ilmentyminen kullakin sivulla. Merkintä ei tarkoita elementtityyppien ilmentymien määrää sivulla, vaan toimii binäärisenä indikaattorina.

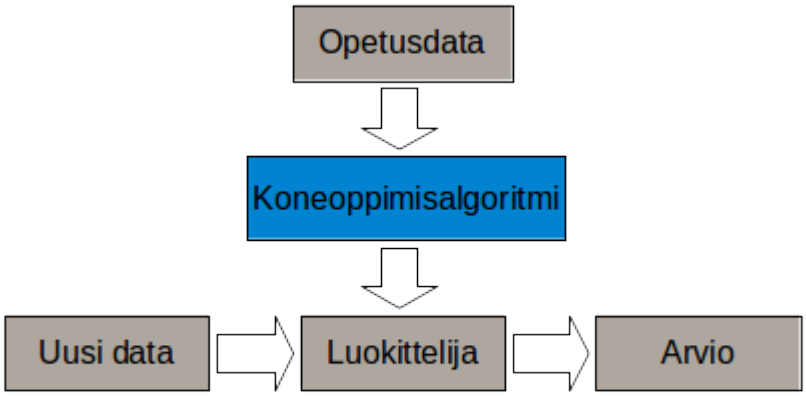
Kerätty tieto viittaa siihen, että usealla sivulla (38%) on käytetty HTML5-standardissa (W3C 2014) määritettyä article-elementtiä, joka määrityksen mukaan ”sisältää itsenäisen tai uudelleen käytettävän osion dokumentista” (W3C 2014). HTML5-standardi pitää siis sisällään halutun kaltaisen elementin, mutta tämän yhden elementtityypin käsittely ei yksin riitä, kuten voidaan kuvasta 2 päätellä. Toinen useasti sivulla ilmenevä elementtityyppi on li-elementti, jota käytetään listoissa ja on myös hyvä elementti käytettäväksi. Kuvan 2 tietojen perusteella voidaan tehdä päätelmä, että käyttäjälle voisi antaa mahdollisuuden valita kaikista taulukon elementtityypeistä. Jos valinnan mahdollisuutta rajattaisiin, voisi järjestelmän käytettävyys kärsiä merkittävästi joillakin web-sivuilla.

5 SISÄLLÖN LUOKITTELU

Sisällön luokittelu on tämän järjestelmän kannalla erittäin tärkeässä asemassa, jotta tietoa voitaisiin suodattaa. Luokkina tässä tapauksessa toimivat ”kiinnostava” ja ”ei kiinnostava”. Koska käsiteltävää tietoa voi olla paljon sekä käyttöajan karttuessa määrä kasvaa, on tärkeää, että se käsitellään optimaalisesti ja väärien negatiivisten sekä positiivisten määrä pysyy hyvissä hyväksyttävissä rajoissa. Väärien negatiivisten ja positiivisten liiallinen määrä tekee suodatusohjelmasta jopa käyttökelvottoman. Koska järjestelmän tarpeiden mukainen luokittelutapa perustuu käyttäjän aloitteesta tapahtuvaan, käyttäjäopasteiseen ja oppivaan luokitteluun, kerron seuraavaksi yleisesti opastetusta tekstinluokittelusta.

5.1 Yleisesti

Opastettu tekstinluokittelu (Kuva 3) perustuu opetusdataan, koneoppimisalgoritmiin (eng. Machine learning algorithm), luokittelijaan, luokiteltavaan dataan (uusi data) ja arvioon. Opetusdatalla tarkoitetaan aiemmin kerättyjä tietoja ja se on valmiiksi luokiteltua. Koneoppimisalgoritmi on algoritmi joka muodostaa mallin opetusdatan perusteella.



Kuva 3: Opastetun tekstinluokittelun malli

Luokittelija luokittelee uuden datan oppimisalgoritmin luoman mallin mukaan ja tekee arvion. Arvion avulla tehdään päätös kuuluuko uusi data määritettyihin luokkiin, ja jos kuuluu niin mihin. Menetelmiä tekstinluokittelun toteuttamiseksi on monia ja niitä voisi syvemmällä tasollakin vertailla, mutta se ei ole tämän työn tavoitteena.

Tämän diplomityön kohteena olevassa sovelluksessa ei ole aiemmin määriteltyä opetusdataa ja luokituksia, vaan oppimisprosessi tapahtuu käyttäjän aktivoimana ja opastetusti (supervised learning). Seuraavaksi kuvaan valitsemaani menetelmää. Menetelmän valitsin monien vaihtoehtojen joukosta todeten, että se on tarpeeksi yksinkertainen, tehokas sekä käytössä monissa samankaltaisissa ratkaisuissa.

5.2 Naiivi Bayes -luokittelijat

Naiivi Bayes -luokittelijat ovat lineaarisia luokittelijoita, jotka ovat toteutukseltaan yksinkertaisia ja tehokkaita. Todennäköisyysmalliin perustuvat naiivi Bayes-luokittelijat (eng. Naive Bayes classifiers) perustuvat Bayesin teoreemaan (Kaava 5-1) ja oletukseen, että käsiteltävän aineiston ominaisuudet ovat toisistaan riippumattomia. Käytännössä riippumattomuuden oletusta rikotaan yleisesti, mutta naiivi Bayes-luokittelijat toimivat silti hyvin. Esimerkiksi pienillä otantamäärillä Bayes-luokittelijat voivat päihittää tehokkaimmat vaihtoehdot. Koska luokittelijat ovat melko karkeita, ne ovat helppoja ja nopeita ottaa käyttöön. Siksi niitä käytetäänkin monilla eri aloilla (Raschka 2014). Bayesin luokittelijat ratkaisevat tekstinluokittelussa ongelman ”Millä todennäköisyydellä dokumentti kuuluu määritettyyn luokkaan”. Bayesin luokittelijoita on useita mutta vain kaksi niistä soveltuvat tekstinluokitteluun.

$$P(h|d)=\frac{(P(d|h)*P(h))}{P(d)}$$

- *P(h|d) on todennäköisyys hypoteesille h datassa d.*
- *P(d|h) on todennäköisyys että että hypoteesi h on totta datassa d.*
- *P(h) on todennäköisyys että hypoteesi h on totta.*
- *P(d) on d:n priori-todennäköisyys.*

Kaava 5-1: Bayesin teoreemaa

Multinomiaalinen malli

Multinomiaalisessa mallissa dokumentissa esiintyvillä toistuvilla sanoilla on merkitystä luokituksen arvioinnissa. Mallissa dokumentin todennäköisyys kuulua tarkasteltavaan luokkaan lasketaan kaavan 5-2 mukaisesti.

$$P(c|d) \propto P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

Kaava 5-2

Kaavassa 5-2 $P(t_k|c)$ on ehdollinen todennäköisyys sanalla t_k esiintyä luokan c dokumentissa. Voidaan myös tulkita että $P(t_k|c)$ mittaa, paljonko on todisteita, että sana t_k kuuluu luokkaan c. Kaavassa 5-2 P(c) on alustava todennäköisyys, että dokumentti kuuluu luokkaan c. Jos dokumentin sanojen todennäköisyyksien avulla ei pystytä selvästi määrittämään mihin luokkaan se kuuluu, valitaan todennäköisin (Manning, Raghavan ja Schütze 2008).

Bernoullin malli

Toisin kuin multinomiaalisessa, Bernoullin mallissa tarkastellaan sanan löytymistä dokumentista eikä sanan toistuvuutta. Edellä mainittu seikka vaikuttaa siihen, että Bernoullin malli tekee virheitä pitkissä dokumenteissa. Esimerkiksi se voi luokitella koko kirjan liittyväksi kiinaan, vaikka kirja sisältäisi vain yhden ilmentymän sanasta kiina (Manning, Raghavan ja Schütze 2008).

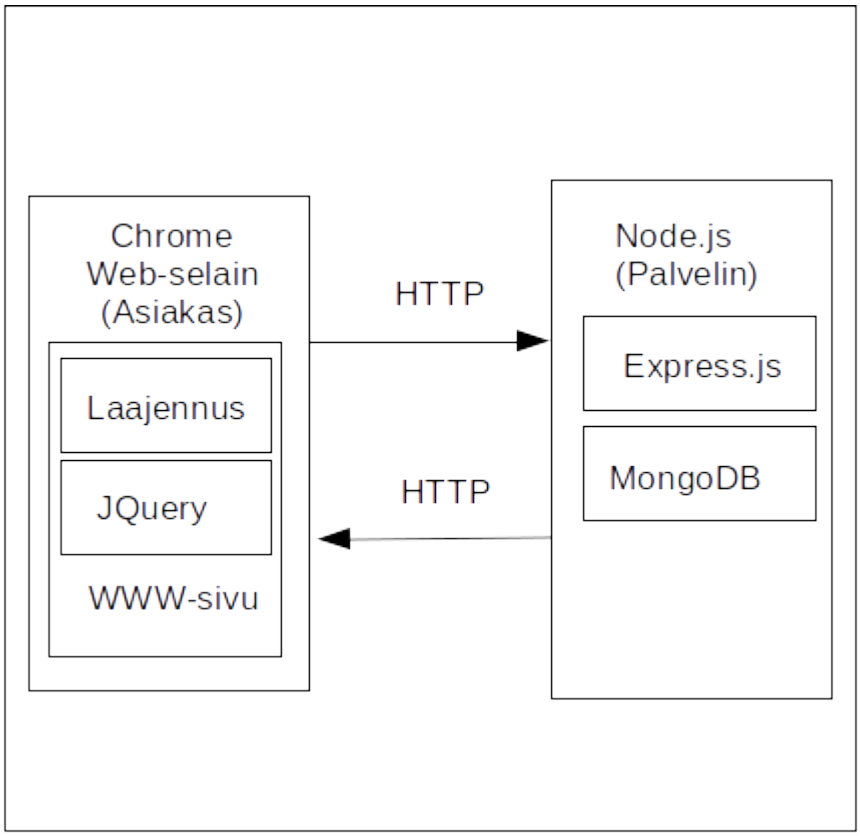
Bag of Words

Bag of words-malli, joka tarkoittaa vapaasti suomennettuna ”säkillinen sanoja”, on yksinkertaistava esitystapa luonnollisten kielien ja käsittelyssä ja tiedon haussa (Wikipedia 2017). Mallissa teksti (kuten lauseet tai dokumentti) esitetään joukkona sanoja, kiinnittämättä huomiota kielioppiin tai edes sanajärjestykseen, mutta kuitenkin huomioiden sanojen toiston. Bag of words mallia käytetään myös esim. konenäkösovelluksissa. Bayes-menetelmää käyttäen sanat luokitellaan kuuluvaksi johonkin ”säkkiin”, tämän työn tapauksessa nämä ovat kiinnostava ja ei kiinnostava.

6 ARKKITEHTUURI

Järjestelmän arkkitehtuuriksi on valittu asiakas-palvelin arkkitehtuuri. Kyseiseen valintaan on päädytty, koska on haluttu pitää asiakaspään toteutuksen mahdollisimman kevyenä sekä mahdollistaa suorituskyvyn helpon kohottamisen palvelimen laskentakapasiteetin lisäämisellä. Malli tekee myös suodatuslogiikan ydintoimintojen optimoinnin helpommaksi, koska asiakasohjelmia ei tarvitsisi tuotantokäytössä päivittää. Valittuihin tekniikkoihin on päädytty jotta asiakas- ja palvelinosien kehitys voidaan tehdä samalla kielellä eli JavaScript:illä.

Arkkitehtuuri muodostuu asiakas- ja palvelinosioista, jotka kommunikoivat keskenään HTTP-protokollan avulla. Molemmat osiot muodostuvat erillisistä osista jotka toteuttavat omaa tehtäväänsä. Järjestelmän arkkitehtuuri on kuvattu yleisellä tasolla kuvassa 4.



Kuva 4: Arkkitehtuuri

Seuraavaksi tarkoitukseni on kuvata järjestelmän eri osiot (asiakas ja palvelin) sekä näiden oleellisimmat osat.

6.1 Palvelinohjelmisto

Palvelinohjelmisto toimii järjestelmän ytimenä suorittaen laskennallisesti vaativimmat tehtävät. Palvelin ohjaa asiakasohjelmaa näyttämään oikean näkymän käyttöprosessin kussakin vaiheessa. Palvelinohjelmiston tehtävänä on tehdä arvio, onko tieto käyttäjää kiinnostavaa sekä välittää arvio asiakkaalle. Palvelimelle on myös tallennettu käyttäjien käyttäjätiedot sekä suodatusprofiilit. Käyttäjistä tallennettavat käyttäjätiedot ovat kirjautumiseen tarvittavat tiedot. Suodatusprofiilit ovat käyttäjäkohtaisia ja mukautuvat käyttäjän käytön mukaisesti. Palvelinohjelmistolle kuuluu myös selaimella käytettävän ylläpitokäyttöliittymän tarjoaminen.

NodeJS

NodeJS on alustariippumaton Javascript-ajoympäristö (Node.js Foundation 2017). NodeJS pohjautuu Chromen V8 Javascript-moottoriin. Se toimii tapahtumapohjaisesti ja asynkronisesti. Tapahtumapohjaisuus ja asynkronisuus tekevät NodeJS:stä kevyen alustan. NodeJS toimii alustana palvelimelle ja sen päälle rakennetaan kerrokset käyttäen useita eri moduuleita. Kaikki tässä työssä käytetyt moduulit ovat avointa lähdekoodia.

ExpressJs

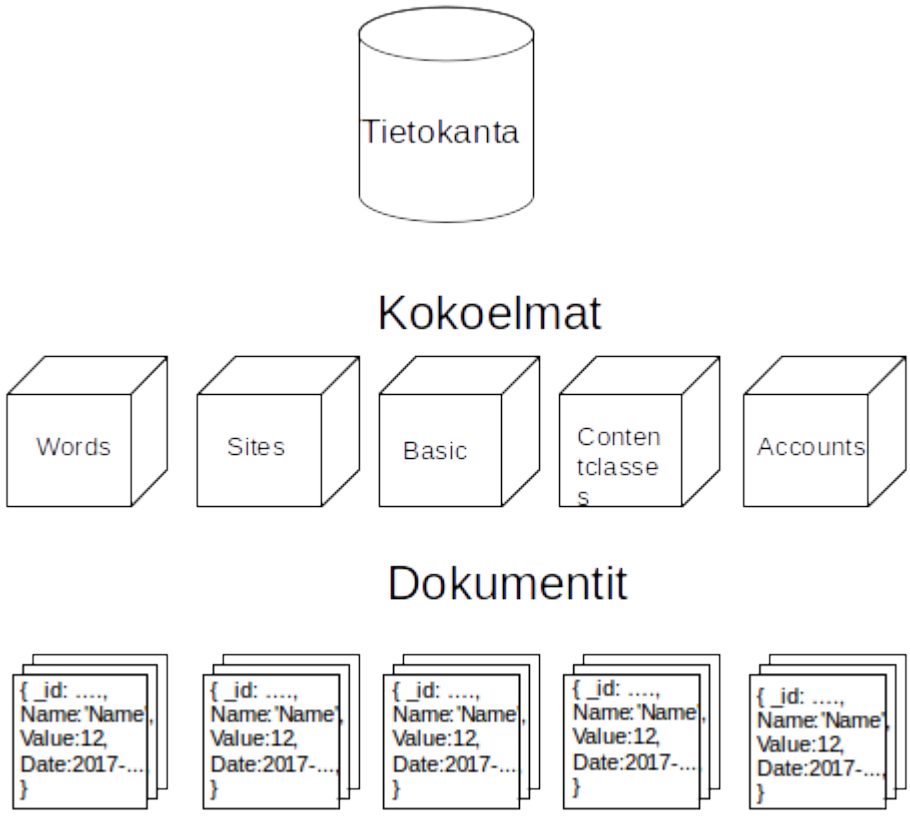
ExpressJs on avoimeen lähdekoodin pohjautuva web-sovelluskehitysalusta NodeJs ajoympäristöön. ExpressJs tarjoaa NodeJS-ympäristöön HTTP-palvelimen perustoiminnot. ExpressJs:n tarjoamia ja tässä toteutuksessa hyödynnettyjä HTTP-palvelimen toimintoja ovat:

- HTTP-yhteyden hallinta
- GET-pyyntöjen vastaanotto
- GET-pyynnön lähettämien kyselymerkkijonojen (eng. Query string) jäsentäminen (eng. parse)
- POST-pyyntöjen vastaanotto
- POST-pyynnön lähettämien tietojen jäsentäminen

Listatut toiminnot mahdollista yhteyden asiakkaan ja palvelimen välillä.

MongoDB

MongoDB toimii järjestelmän tietovarastona. MongoDB on ilmainen ja avoimeen lähdekoodiin pohjautuva sekä alustariippumaton dokumentti -suuntautunut tietokantaohjelma. MongoDB-tietokanta (Kuva 5) muodostuu tietokannoista ja niiden sisältämistä kokoelmista (eng. Collections). Kokoelmat toimivat tallennustilana tietueille eli BSON-dokumenteille. BSON-dokumentit ovat binäärisiä kuvauksia JSON-dokumenteista. Järjestelmässä tiedot tallennetaan JSON dokumentteina MongoDB-tietokantaan.



Kuva 5: Tietokannan rakenne

Tietokanta muodostuu seuraavista kokoelmista:

- Accounts
- Words
- Contentclasses
- Basic

- Sites

Accounts-kokoelma toimii tallennuspaikkana käyttäjien käyttäjätilitiedoille. Kokoelmaan tallennettavat käyttäjätiedot ovat seuraavat:

- Tietokannan muodostama yksilöivä tunniste (_id)
- Sähköposti (email)
- Salasanan tiiviste (pass)
- Rekisteröitymispäivä (date)
- Onko käyttäjä aktiivinen (active)
- Viimeinen käyttöaika (lastUsed)

Words-kokoelma toimii käyttäjäkohtaisen sanaston tallennuspaikkana. Words-kokoelmaan tallennettavien dokumenttien tiedot ovat seuraavat:

- Tietokannan muodostama yksilöivä tunniste (_id)
- Sana pienaakkosmuodossa (word)
- Sanan ilmentyminen positiivissa yhteyksissä (pos)
- Sanan ilmentyminen positiivissa yhteyksissä (neg)
- Käyttäjä, jonka sanastoon sana kuuluu (user)

Contentclasses-kokoelmaan tallennetaan käyttäjäkohtaiset sivustojen valittujen tekstisisältöelementtien tunnisteet (css-luokka). Kokoelman dokumenttien tiedot ovat seuraavat:

- Tietokannan muodostama yksilöivä tunniste (_id)
- Sisältöelementin tunniste(contentClass)
- Sivusto (site)
- Käyttäjä (user)

Basic-kokoelma pitää sisällään käyttäjäkohtaisia yksittäisiä tietoja joita ei ole tarve tallettaa omiin kokoelmiinsa. Basic-kokoelman dokumenttien tiedot ovat seuraavassa muodossa:

- Tietokannan muodostama yksilöivä tunniste (_id)
- Tietueen nimi (name)
- Käyttäjä (user)
- Tietueen arvo (value)

Sites-kokoelmaan tallennetaan käyttäjä ja sivustokohtaisesti tieto siitä, onko suodatusohjelma aktiivinen kyseisessä sivustonäkymässä. Sites-kokoelman dokumenttien tiedot ovat seuraavat:

- Tietokannan muodostama yksilöivä tunniste (_id)
- Sivusto (site)
- Käyttäjä (user)

AngularJS

AngularJS on Googlen ylläpitämä avoimen lähdekoodin JavaScript-ohjelmistokehys. AngularJS:ää käytetään käyttäjien hallintaliittymän eli järjestelmän ylläpitokäyttöliittymän toteuttamisessa. AngularJS mahdollistaa helposti yhden sivun sovelluksien (eng. Single page applications) nopean toteuttamisen. Koska hallintaliittymän on tarkoitus pitää sisällään hyvin vähän toimintoja, niin toteutus yhden sivun sovelluksena on järkevää.

6.2 *Asiakasohjelmisto*

Asiakasohjelma toimii Chrome-selaimen sisällä selainlaajennuksena ja tarjoaa yksinkertaisen käyttöliittymän suodatustoiminnoille. Asiakkaan tehtävänä on tarjota käyttäjälle käyttöliittymä, kerätä tekstisisältöä ja mahdollistaa elementtityyppien valinta web-sivulta.

Chrome-laajennus

Chrome-laajennus on Googlen Chrome -selaimessa toimiva lisäohjelma. Laajennuksia voidaan myydä, jakaa ja ostaa Googlen Chrome Web Storen kautta. Google Chrome laajennukset eivät toimi mobiili Chromella.

Selainlaajennus kytketään osittain selaimen näkyvissä olevaan sivurakenteeseen. Kytkennällä mahdollistetaan DOM-puun lukeminen, muokkaus, sekä sisällön hallintatoimintojen tuominen sivulle. Laajennus muodostuu kolmesta osasta: tausta-skripti (background), ponnahdusikkuna (popup) ja sisältö-skripti (content). Laajennuksen osilla jokaisella on oma tarkalleen määritetty tehtävänsä.

Tausta-skripti toimii laajennuksessa keskeisimmässä asemassa. Tausta-skriptin tehtävänä on pitää yllä tietoja laajennuksen tilasta sekä toimia viestinvälittäjän palvelimen ja muiden osien välillä. Tausta-skriptin ylläpitämiä tietoja käyttäjän ohjauspaneelissa valitsemat asetukset. Tausta-skripti myös välittää kirjautumis-, rekisteröinti- ja uloskirjausviestit palvelimelle.

Ponnahdusikkuna toimii käyttöliittymänä laajennukselle ja pitää sisällään ohjauspaneelin. Ponnahdusikkuna välittää käyttäjän valinnat tausta-skriptille.

Sisältö-skripti upotetaan web-sivulle ja se tuo sivun rakenteeseen sisällönhallintapainikkeet. Sisältö-skripti viestii myös suoraan palvelimen kanssa lähettämällä ja pyytämällä sisällönhallintaan liittyviä tietoja.

Osien välinen viestintä perustuu niin sanottuihin ”pitkäikäisin yhteyksiin” (eng. Long-lived connections). Tässä ratkaisussa yhteys muodostetaan niin, että tausta-skripti luo onConnect tapahtumakuuntelijan (eng. Event listener). Tapahtumankuuntelija vastaanottaa muiden osien lähettämät viestit. Sisältö ja ponnahdusikkuna osat muodostavat yhteyden tausta-skriptin tarjoamaan yhteysliittymään.

Kuva 6: Laajennuksen sisäinen viestintä

Vietin kulkua eri osien välillä (Kuva 6) voidaan kuvata esim. tapauksella jossa käyttäjä kirjautuu järjestelmään:

1. Käyttäjä syöttää kirjautumistiedot ponnahdusikkunan kirjautumisnäkymään.
2. Ponnahdusikkuna lähettää viestin ”doLogin” (kirjaudu) tausta-skriptille.
3. Tausta-skripti vastaanottaa ”doLogin” viestin ja lähettää kirjautumistiedot eteenpäin palvelimelle.
4. Palvelin palauttaa tiedon kirjautumisen onnistumisesta tausta-skriptille.
5. Tausta-skripti lähettää viestin kirjautumisen onnistumisesta ponnahdusikkunalle.
6. Ponnahdusikkunaan vaihdetaan näkyminä kirjautumisen tilan mukaisesti.

JQuery

JQuery on alusta- ja selainriippumaton Javascript- kirjasto. Kirjasto on tarkoitettu yksinkertaistamaan HTML:n asiakaspuolen skriptausta. JQuery on ilmainen, avointalähdekoodia ja käyttää MIT-lisenssiä (Wikipedia 2017).

JQueryn syntaksi on suunniteltu siten että dokumenttien läpikäynti, DOM-elementtien valinta, animaatioiden luonti, tapahtumien käsittely ja ajax-sovellusten kehittäminen oli-

si helppoa. JQuery mahdollistaa kehittäjiä luomaan laajennuksia kirjaton päälle. JQueryn modulaarinen lähestymistapa mahdollistaa myös tehokkaiden dynaamisten web-sivujen ja web-sovellusten luomisen. (Wikipedia 2017)

JQueryn ydintoiminnoista DOM-elementtien valinta sekä manipulointi mahdollistavat asiakasohjelman puolella suodatuslaajennuksen toimintapainikkeiden esille tuomisen sekä tekstisisällön suodatuksen. JQueryn tarjoamaa AJAX-rajapintaa käytetään asiakkaan ja palvelimen välisessä viestinnässä.

7 PROSESSIT

Tässä osiosta kuvataan järjestelmän prosessit. Prosessit ovat tämän järjestelmän keskeisiä toimintoja. Prosessit muodostuvat järjestelmän osien välisen kommunikaation seurauksena. Prosessit kuvataan ensin sanallisesti ja tämän jälkeen kuvallisesti, jos monimutkaisuus sen vaatii.

Rekisteröitymisprosessi

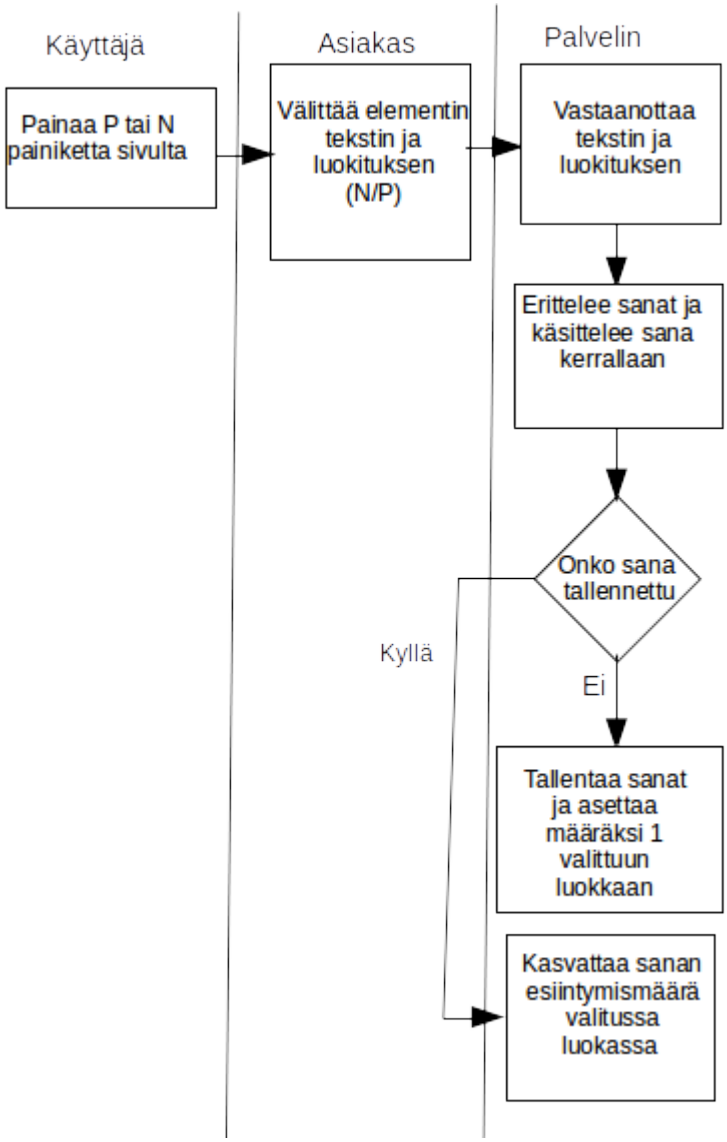
Rekisteröitymisprosessissa käyttäjä rekisteröityy järjestelmän käyttäjäksi. Käyttäjä painaa selaimen osoiterivin vieressä olevaa asiakasohjelman painiketta. Seuraavaksi käyttäjä painaa ponnahdusikkunassa olevaa ”Rekisteröidy” painiketta. Asiakasohjelma kysyy käyttäjältä sähköpostiosoitteen, salasanan sekä pyytää käyttäjää toistamaan salansana. Tietojen syöttämisen jälkeen käyttäjä painaa ”Rekisteröidy” painiketta. Asiakasohjelma lähettää käyttäjän syöttämät tiedot palvelimelle. Palvelin vastaanottaa asiakasohjelman lähettämät tiedot. Vastaanotettuaan tiedot palvelin tarkastaa löytyykö sähköpostiosoitetta vastaava käyttäjätili jo järjestelmästä. Jos käyttäjätiliä ei löydy niin palvelin luo uuden käyttäjätilin ja tallentaa sen tietokantaan. Palvelin palauttaa asiakasohjelmalle tiedon rekisteröinnin onnistumisesta/epäonnistumisesta.

Kirjautumisprosessi

Kirjautumisprosessi käynnistyy käyttäjän aloitteesta, kun käyttäjä painaa selaimen osoiterivin vieressä olevaa asiakasohjelman painiketta. Asiakasohjelma kysyy käyttäjältä sähköpostiosoitteen ja salasanan. Käyttäjän painettua ”Kirjaudu” painiketta asiakasohjelma lähettää kirjautumistiedot palvelimelle. Palvelin vastaanottaa kirjautumistiedot ja tarkastaa löytyykö tietokannasta käyttäjätiliä, jonka tiedot täsmäävät kirjautumistietojen kanssa. Käyttäjätietojen löytyessä palvelin alustaa käyttäjälle istunnon ja palauttaa asiakasohjelmalle tiedon kirjautumisen onnistumisesta. Jos kirjautumistiedot eivät vastaa olemassa olevaa käyttäjätiliä, palvelin lähettää asiakasohjelmalle tiedon kirjautumisen epäonnistumisesta. Asiakasohjelma ilmoittaa käyttäjälle epäonnistuneesta kirjautumisesta tiedotusikkunalla.

Sisällönluokitteluprosessi

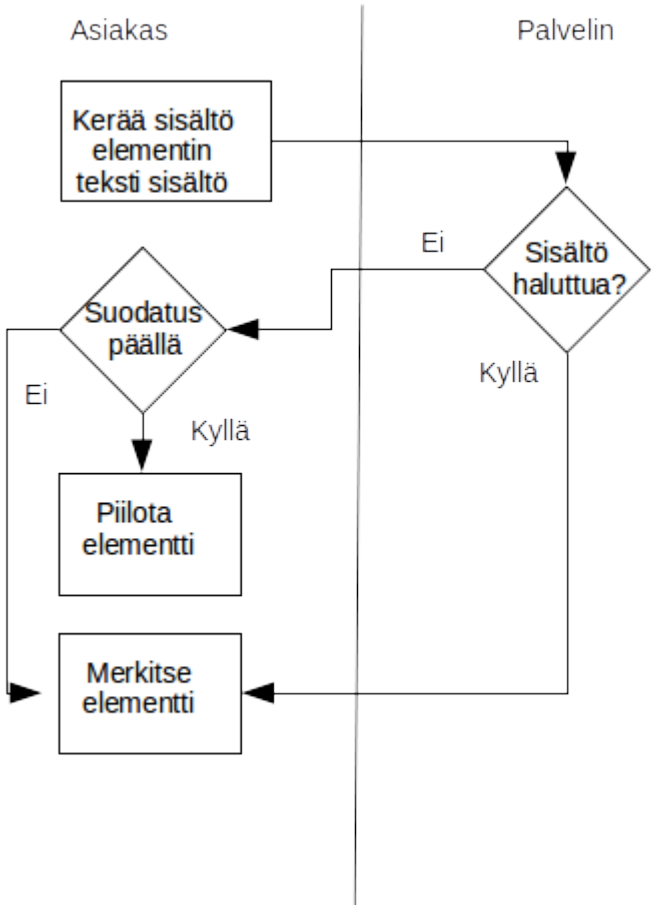
Sisällönluokitteluprosessilla (Kuva 7) tarkoitetaan tässä tapauksessa prosessia, jolla jokin sivulla olevan tekstielementin sisältö määritetään käyttäjän toimesta joko kiinnostavaksi tai ei kiinnostavaksi. Prosessi käynnistyy kun käyttäjä painaa tekstielementissä olevaa P tai N painiketta. Asiakasohjelma vastaanottaa käyttäjän palautteen. Asiakasohjelma lähettää tekstin sekä luokituksen palvelimelle. Palvelin vastaanottaa luokitustiedon sekä tekstin. Palvelin erittelee tekstistä sanat ja muuttaa kirjaimet pienkirjaimisiksi (eng. lower case). Tämän jälkeen palvelin tarkastaa yksi kerrallaan, onko sana tallennettu. Jos sanaa ei ole tallennettu, palvelin tallentaa sen ja asettaa sen kuuluvaksi valittuun luokkaan. Jos sana on tallennettu, palvelin kasvattaa sen todennäköisyyttä kuulua valittuun luokkaan.



Kuva 7: Sisällönluokitteluprosessi

Suodatusprosessi

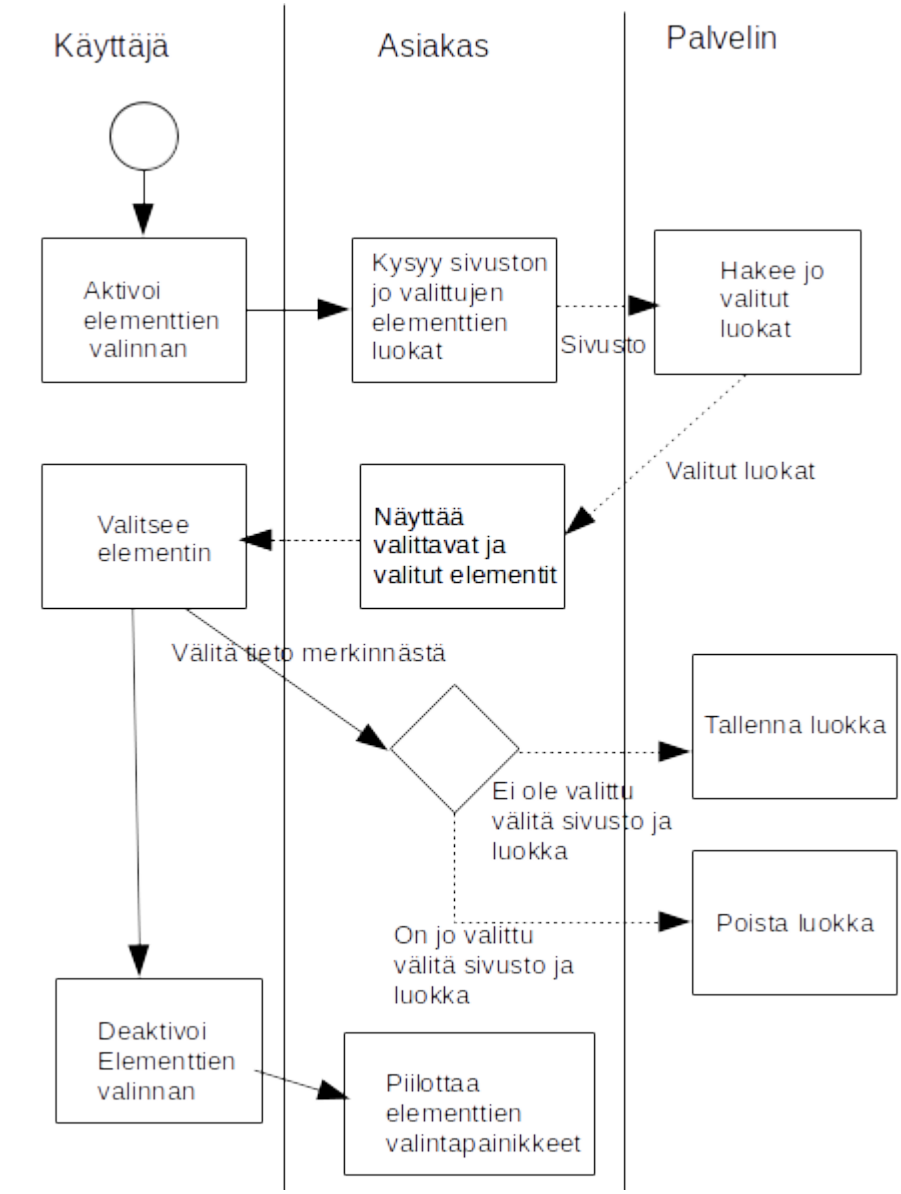
Seuraavaksi pyrin kuvaamaan yksityiskohtaisesti suodatusprosessin. Suodatusprosessi (Kuva 8) käynnistyy asiakasohjelman (selainlaajennus) kerätessä sivulta tekstisisältöelementit. Asiakasohjelma välittää tekstisisältöelementtien tekstisisällöt yksi kerrallaan palvelimelle. Palvelin erittelee tekstistä sanat ja muuttaa pienkirjainmuotoon. Palvelin tekee päätöksen sisällön kiinnostavuudesta. Kiinnostavuus määritetään Naivii Bayesin-menetelmällä (kts. 5.2). Tämä tapahtuu vertailemalla lauseiden sanojen positiivisten yhteistodennäköisyyksien ja negatiivisten yhteistodennäköisyyksien suuruutta. Palvelin palauttaa päätöksen asiakkaalle. Jos suodatus on sivustokohtaisesti määritetty päällä olevaksi, asiakas merkitsee testin sisältävän elementin piilotetuksi tai merkitsee (korostaa) elementin.



Kuva 8: Suodatusprosessi

Sisällönvalintaprosessi

Sisällönvalintaprosessi (Kuva 9) on käyttäjän aktivoima prosessi, jolla web-sivun sisältöelementeistä valitaan ne elementit, joihin halutaan suodatus kohdistaa. Prosessin aktivoimiseksi käyttäjä painaa ohjauspaneelin ”Näytä sisällön määritys” painiketta. Käyttäjän toiminta saa asiakasohjelman lähettämään kyselyn palvelimelle sivuston aiemmin valituista elementtien luokista. Palvelin suorittaa haun tietokannasta käyttäen hakukriteerinä käyttäjään tunnistetta sekä sivustonnimeä. Palvelin palauttaa asiakkaalle listan aiemmin määritetyistä luokista tai tyhjän listan. Saatuaan listan asiakas näyttää web-sivulla niiden elementtien valintapainikkeet vihreinä, joiden luokka löytyy listasta. Valitsemattomien elementtien valintapainikkeet näytetään punaisina.



Kuva 9: Sisällönvalintaprosessi

Käyttäjän painaessa vihreää valintapainiketta asiakasohjelma merkitsee valintapainikkeen punaiseksi ja välittää palvelimelle tiedon valitun elementin luokan poistamiseksi listalta. Käyttäjän painaessa punaista valintapainiketta asiakas merkitsee painikkeen vihreäksi ja välittää palvelimelle tiedon valitun elementin luokan lisäämiseksi listalle. Käyttäjän valittua haluamansa elementit käyttäjä painaa ”Näytä sisällön määritys” vaihtopainiketta uudelleen. Painikkeen painamisen jälkeen asiakasohjelma piilottaa elementtien valintapainikkeet.

Kirjautuminen ylläpitäjän käyttöliittymään

Prosessi käynnistyy ylläpitäjän siirtyessä selaimella ylläpitäjän käyttöliittymään. Ylläpitäjälle tuodaan näkyviin kentät käyttäjätunnuksen ja salasanan syöttämiseksi. Ylläpitäjän painettua kirjautumispainiketta selain lähettää kirjautumistiedot palvelimelle. Palvelin tarkastaa lähetettyjen ylläpitäjäkirjautumistietojen oikeellisuuden. Kirjautumistietojen ollessa oikeat palvelin välittää tiedon selaimelle ja selaimessa vaihdetaan käyttäjien hakunäkymä. Kirjautumistietojen ollessa väärät kirjautujalle näytetään ilmoitus.

8 KÄYTTÖLIITTYMÄ

Koska suodatusohjelmisto suunnitellaan ja toteutetaan oikeaan käyttöön, sekä asiakasohjelman käyttäjäryhmäksi on tarkoitettu internetsivuja selaavat peruskäyttäjät, täytyy käyttöliittymän suunnitteluun kiinnittää erityisesti huomiota. Ohjelmiston käyttäjäryhmän ollessa erittäin heterogeeninen, käyttöliittymän täytyy olla yksinkertainen, helposti omaksuttavissa ja helppo käyttää. Järjestelmän käyttöliittyminä toimivat asiakasohjelman käyttöliittymä sekä ylläpitäjänkäyttöliittymä.

Tässä osiossa on tarkoitus kuvata ohjelmiston eri käyttöliittymät, esitellä niiden hahmotelmat (alkutilanne/proto) sekä iteratiivisen suunnitteluprosessin lopputulos eli lopullinen käyttöliittymä. Tarkoituksena on myös avata siirtymäprosesseja eri näkymien välillä.

8.1 Asiakasohjelma

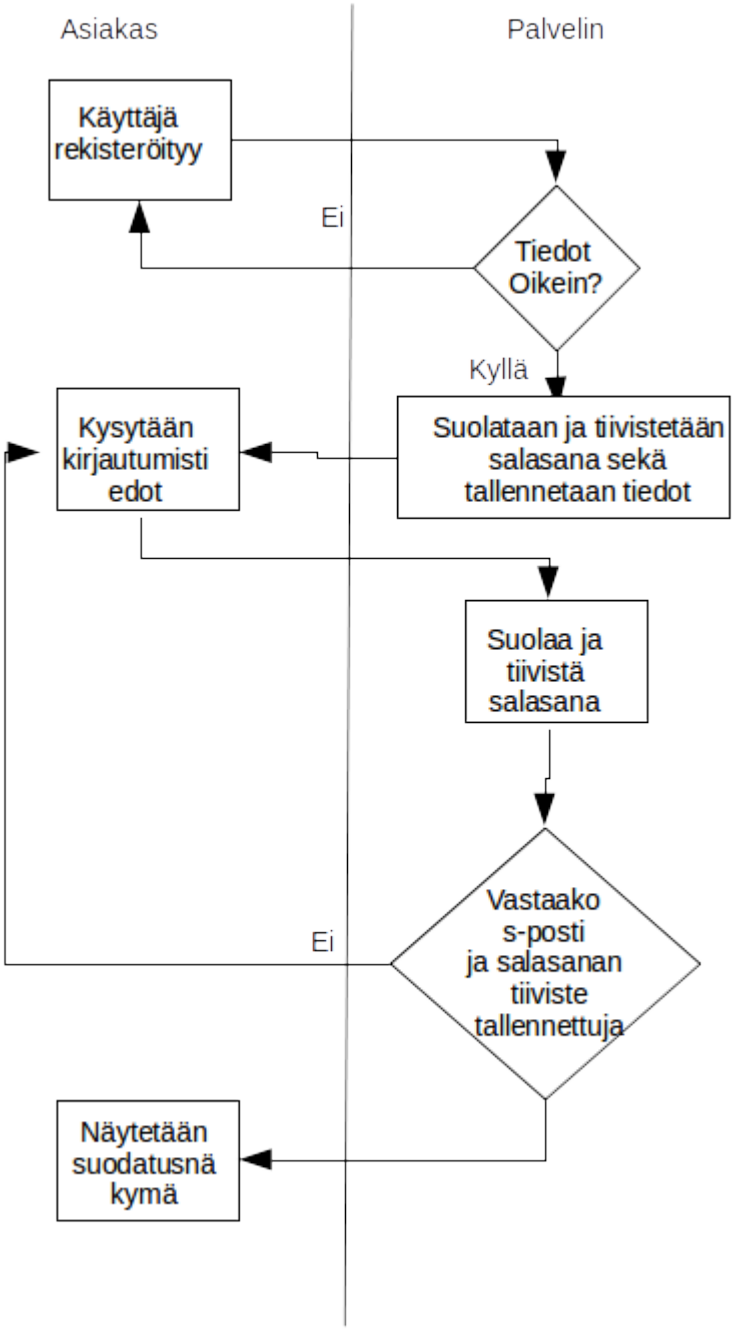
Asiakasohjelman käyttöliittymän suunnittelu on vaativin osuus käyttöliittymien suunnitteluprosessissa. Asiakasohjelman käyttöliittymä muodostuu kolmesta erillisestä osasta. Ensimmäisenä osana on selaimen osoiterivin viereen tuotava selainlaajennuksen näyttöpainike. Toisena käyttöliittymän osana on ohjauspaneeli (ponnahdusikkuna) joka aktivoidaan ensimmäisen osan painikkeesta. Kolmantena on sivun ”päälle” tuova sisällönhallintakerros. Asiakasohjelman eri käyttöliittymäosissa panostettiin visuaalisen selkeyteen valitsemalla sopivat värit komponenttityypeille, jotta ne erottuisivat toisistaan.

Osoiterivin näyttöpainike

Osoiterivin näyttöpainike on asiakasohjelman selaimen osoiteriville tuoma painike, jonka tehtävänä on tuoda näkyviin ohjauspaneeli sekä toimia myös suodatusohjelman tilan indikaattorina. Käyttäjän ollessa kirjautuneena suodatusohjelmaan painikkeessa näytetään sisäänkirjautunutta indikoitava kuvake, kun taas uloskirjautumisen jälkeen tai ennen sisäänkirjautumista painikkeen indikaattorina on uloskirjautumista symboloiva kuvake.

Ohjauspaneeli

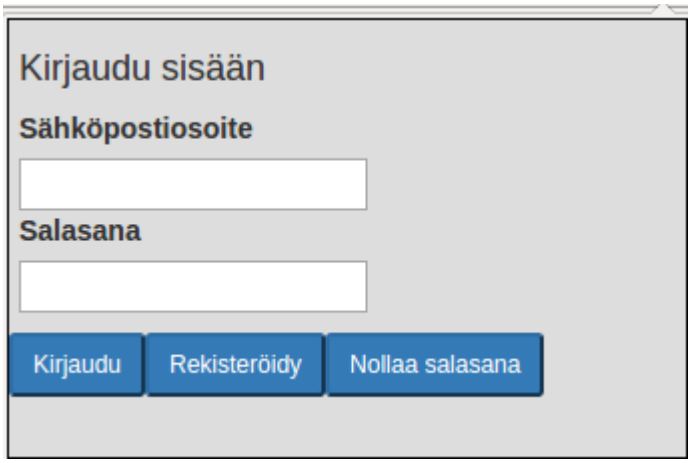
Ohjauspaneelilla on kolme näkymää. Näkymät vaihtuvat käyttäjän valintojen perusteella. Ohjauspaneelin näkymiä ovat rekisteröitymisnäkymä (Kuvat 13 ja 14), kirjautumisnäkymä (Kuvat 11 ja 12) sekä suodatuksen ohjausnäkymä (Kuvat 15 ja 16). Näkymien vaihtumisprosessi käyttäjän toimesta on kuvattu kuvassa 10.



Kuva 10: Ohjauspaneelin näkymien vaihtuminen

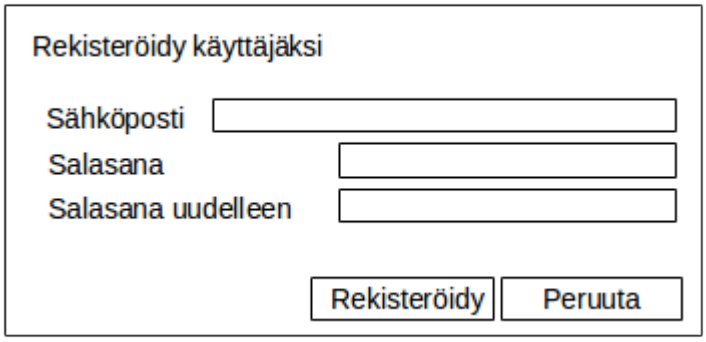
Ensimmäisenä näkymänä kirjautumattomalle käyttäjälle tulee aina kirjautumisnäkymä (Kuvat 11 ja 12). Kirjautumisnäkymän suunnittelussa on pyritty näkymän yksinkertaisuuteen ja selkeyteen. Näkymän painikkeet järjestetään vasemmalta oikealle painikkeen todennäköisemmän käyttötaajuuden mukaan. Edellä mainittu menetelmä typistyy siihen, että ”Kirjaudu”-painike sijaitsee äärivasemmalla, sitä seuraa toiseksi käytetyin näkymän painike ”Rekisteröidy”-painike ja viimeisenä ”Salasanan palautus” -painike. Näkymän painikkeet ovat itsensä selittäviä. ”Kirjaudu”-painikkeella käyttäjä kirjautuu järjestelmään syöttämillään sähköpostiosoitteella ja salasanalla. Kirjautumisnäkymästä on alle liitetty kuvat alkuvaiheen hahmotelmasta sekä toteutuneesta lopullisesta näkymästä.

Kuva 11: Kirjautumisnäkymä (hahmotelma)

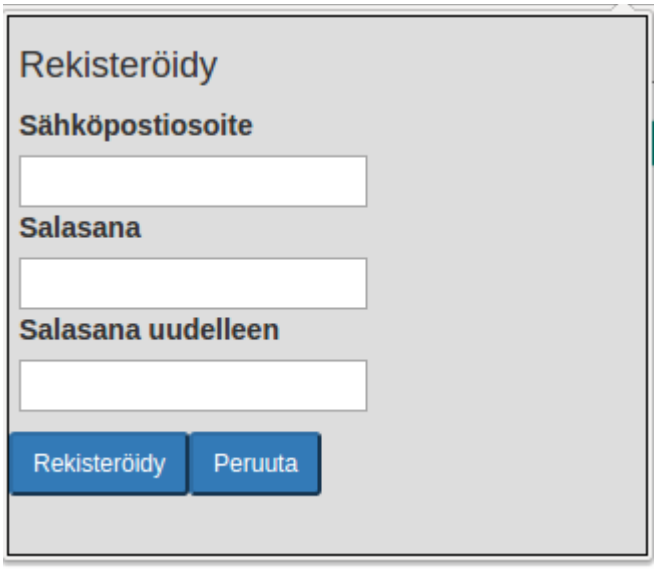


Kuva 12: Kirjautumisnäkymä (lopullinen)

Rekisteröitymisnäkymä (Kuvat 13 ja 14) on suunniteltu myös aiemmin mainittujen käytettävyyden ja selkeyden periaatteiden mukaisesti. Näkymä on rakenteeltaan hyvin itsensä selittävä. Rekisteröitymisnäkymässä käyttäjä syöttää sähköpostiosoitteensa, haluamansa salasanan ja rekisteröityy ”Rekisteröidy”-painikkeella .



Kuva 13: Rekisteröintinäkymä (hahmotelma)



Kuva 14: Rekisteröitymisnäkymä (lopullinen)

Suodatuksen ohjausnäkymässä (Kuvat 15 ja 16) ei poikettu valituista selkeyden ja käytettävyyden periaatteista. Suodatuksen ohjausnäkymä muutti muotoaan useampaan otteeseen käyttöliittymän kehitysprosessin aikana. Lopulliseen näkymään toteutuivat painikkeet, joiden toiminnan kuvaan yksityiskohtaisemmin seuraavaksi.

Käyttöliittymässä ensimmäisenä ylhäältä alaspäin katsottaessa näkyy valintalaatikko ”Käytä tällä sivulla suodatusta” määrittää käytetäänkö sisällönsuodatuslaajennusta näkyvällä web-sivulla.

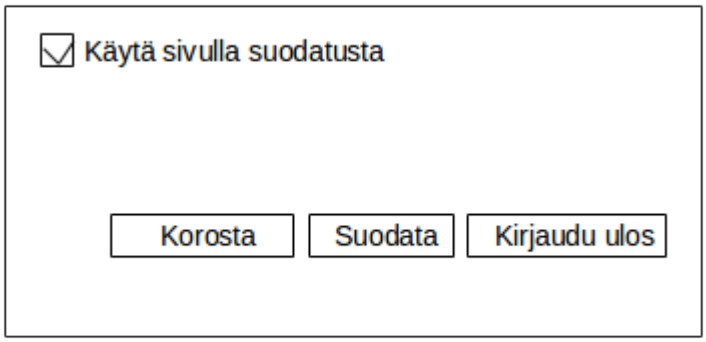
Seuraavana kontrollina on painike ”Näytä sisällön määritys”, tätä painiketta ei ollut vielä hahmotteluvaiheessa, ja tarve ilmeni vasta järjestelmää toteuttaessa. Painikkeella tuo-

daan web-sivulla olevaan sisällönhallintakerrokseen valintapainikkeet, joilla tekstielementit voidaan lisätä suodatusprosessin tarkastelulistalle.

Kolmantena painikkeena on ”Suodata”-painike joka lisää sisällönhallintakerrokseen luokittelu painikkeet sekä määrittää ”selvästi” negatiivisen sisällön piilotettavaksi.

Neljäs painike ”Korosta”, toimii muuten ”Suodata”-painikkeen tavoin, mutta määrittää että sisältöä ei piiloteta vaan se korostetaan. Viimeinen painike ”Kirjaudu ulos” on itsensä selittävä, eli painikkeen avulla käyttäjä kirjautuu ulos asiakasohjelmasta.

Painikkeet ”Näytä sisällön määritys”, ”Suoda” ja ”Korosta” toimivat niin sanottuina vaihtopainikkeina (eng. toggle button) ja näistä kaksi viimeistä eivät luonnollisesti voi olla päällä samanaikaisesti.



Kuva 15: Suodatuksen ohjausnäkymä(hahmotelma)

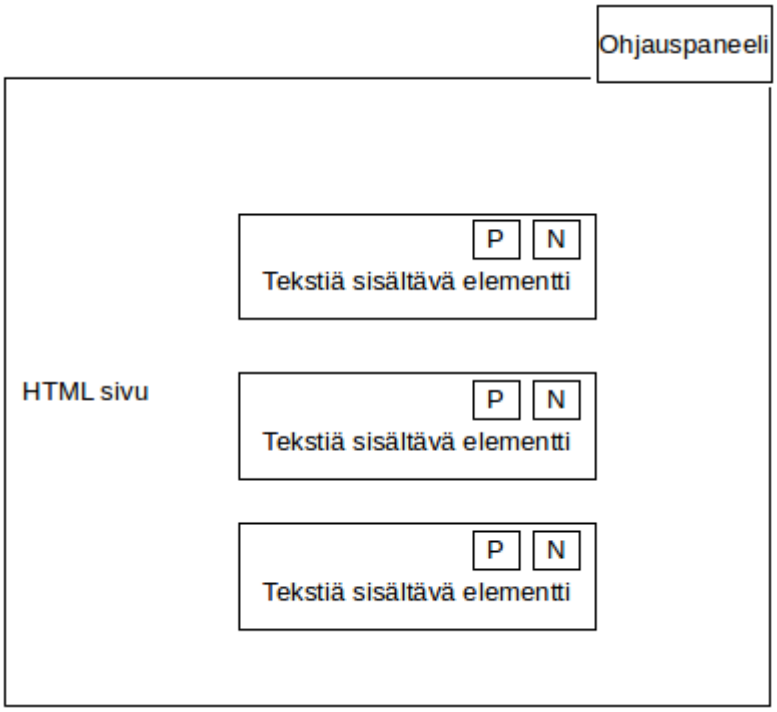
Kuva 16: Suodatuksen ohjausnäkymä(lopullinen)

Sisällönhallintakerros

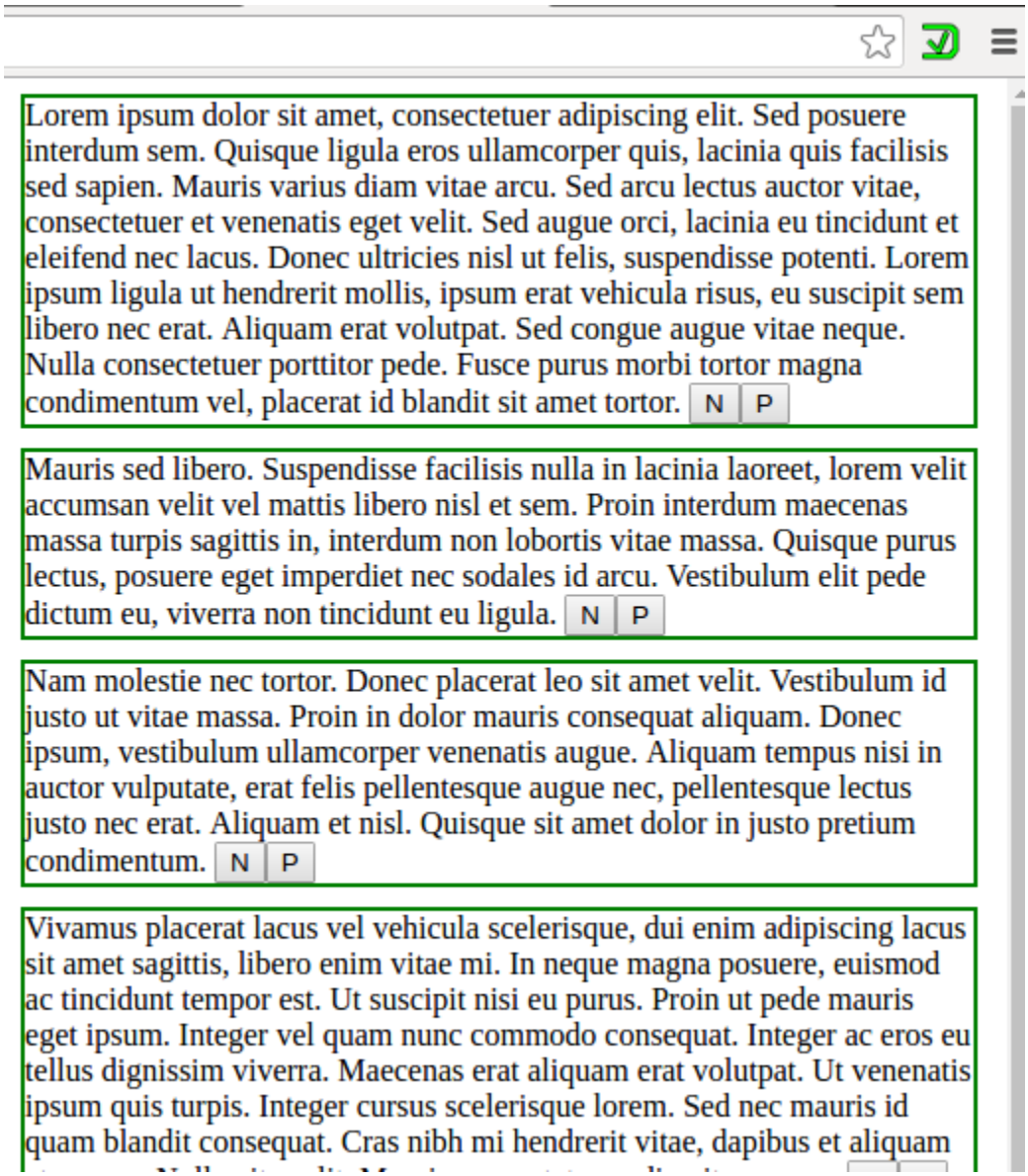
Sisällönhallintakerros (kuvat 17 ja 18) on osa asiakasohjelman käyttöliittymää. Sisällönhallintakerros on upotettu web-sivun DOM-rakenteeseen. Koska tämä käyttöliittymä näkyy osana selattavaa web-sivua, sen täytyy erottua alkuperäisestä sivusta, mutta se ei saa tarpeettomasti sotkea sitä. Erottumiseen käytetään vahvoja visuaalisia menetelmiä. Visuaaliset menetelmät tai toisin sanoen tehosteet ovat vahvojen värien käyttö (punainen,vihreä) sekä alueiden rajaviivojen katkoviivoittaminen. Punainen ja vihreä ovat hyviä indikaattoreita mutta, jotkut ihmiset eivät voi näitä kahta erottaa värisokeuden vuok-

si. Tämän vuoksi käytössä on myös katkoviivoittaminen. Sisällönhallintakerroksen komponentteja ovat N- ja P-painikkeet, sisällönvalintapainikkeet sekä tekstielementtien rajausviivat. Tekstielementtien N- ja P-painikkeilla määritetään sisällön kiinnostavuus (positiivinen tai negatiivinen).

Sisällönhallintakerros on myös jaettu kahteen erilliseen kerrokseen. Nämä kerrokset ovat sisällönvalinta ja sisällönsuodatus. Kerrokset ovat erillisiä siinä mielessä että ne voivat olla näkyvissä joko samanaikaisesti tai eri aikaan.



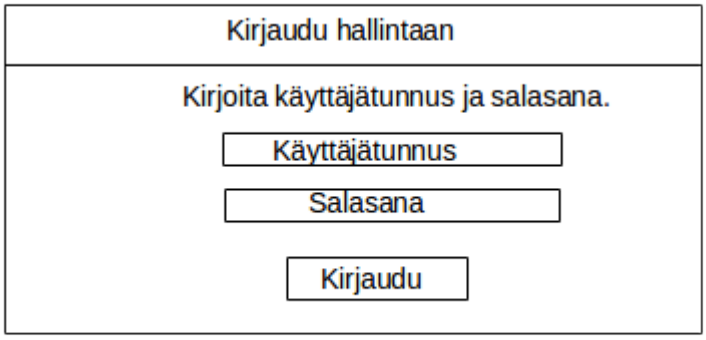
Kuva 17: Selaimen sivunäkymä kun hallintapainikkeet on lisätty (hahmotelma)



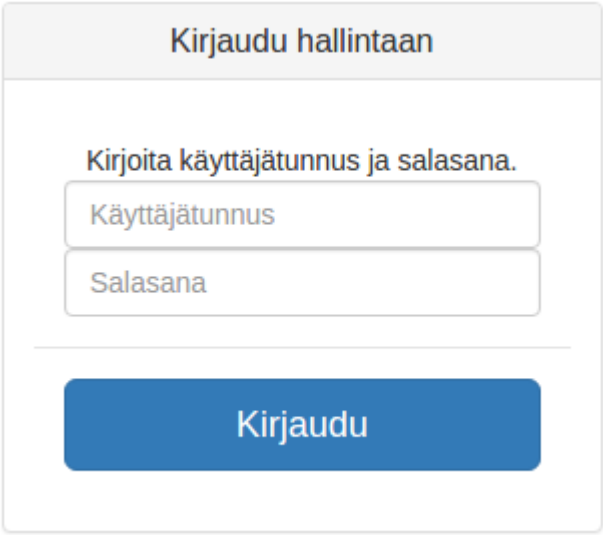
Kuva 18: Selaimen sivunäkymä kun hallintapainikkeet on lisätty (toteutunut)

8.2 Ylläpitäjän käyttöliittymä

Ylläpitäjän käyttöliittymä toimii käyttäjienhallintaliittymänä. Ylläpitäjän käyttöliittymä on yksinkertainen ja suunniteltu vain ylläpitäjän käyttöön. Päästäkseen selaamaan käyttäjätietoja pääkäyttäjän on ensin kirjauduttava kirjautumisnäkymässä (Kuvat 19 ja 20). Ylläpidon kirjautumisnäkymästä on tehty mahdollisimman yksinkertainen.



Kuva 19: Hallintaliittymään kirjautuminen (hahmotelma)

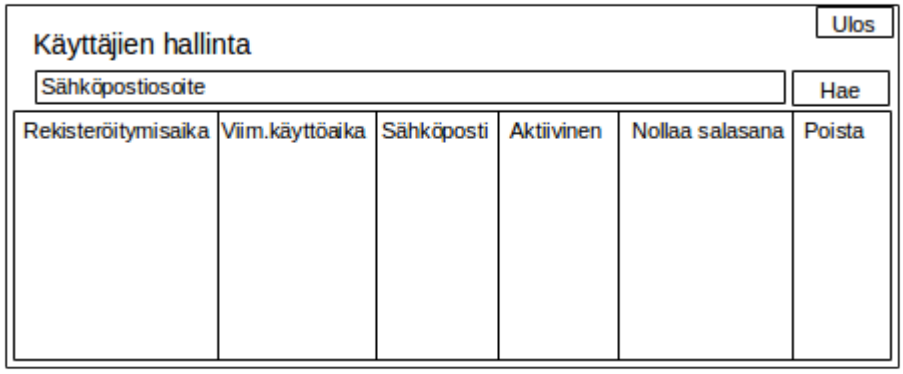


Kuva 20: Hallintaliittymään kirjautuminen (toteutunut)

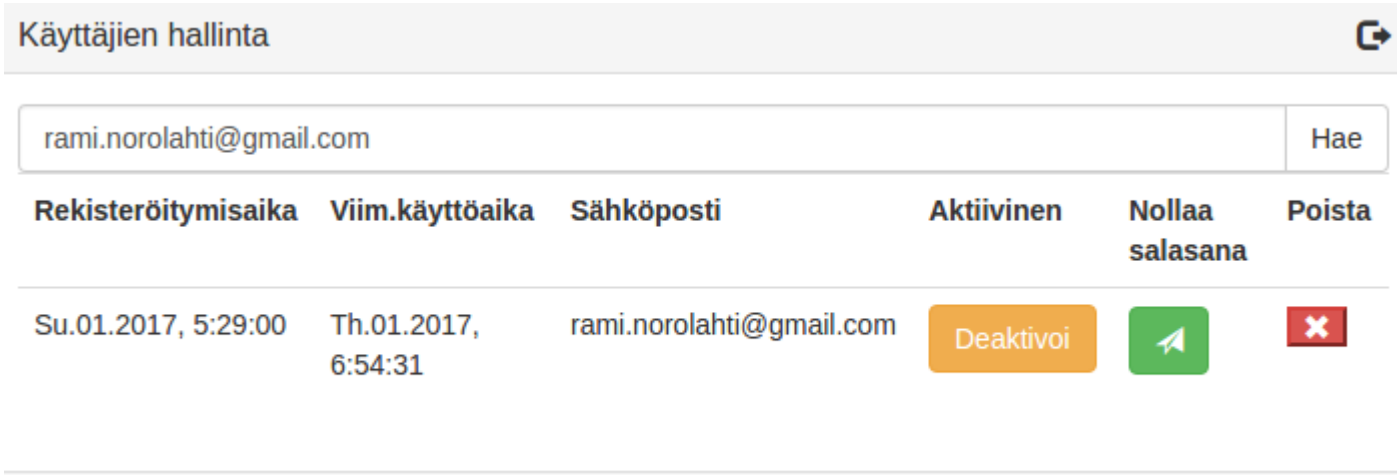
Käyttäjien ylläpito

Käyttäjien ylläpitonäkymästä (Kuvat 21 ja 22) on tehty mahdollisimman yksinkertainen ja selkeä.
Käyttäjien ylläpito- näkymä pitää sisällään seuraavat toiminnot:

- Käyttäjien haku sähköpostiosoitteella
- Salasanan nollaus
- Käyttäjän poistaminen
- Käyttäjän aktivointi
- Uloskirjautuminen



Kuva 21: Käyttäjien ylläpito (hahmotelma)



Kuva 22: Käyttäjien ylläpito (toteutunut)

9 TIETOTURVA

Koska järjestelmä on suunniteltu ja tarkoitettu monen käyttäjän järjestelmäksi, sekä mahdollisesti jossain vaiheessa Google Web Storesta ladattavaksi, tietoturvan pohtiminen sekä turvallisuusajattelun on oltava mukana alusta alkaen. Tässä osiossa on tarkoitus käydä läpi järjestelmän tietoturvaa sekä järjestelmään liittyviä tietoturvauhkia.

9.1 Käyttäjätiedot

Jotta järjestelmää voi käyttää, täytyy käyttäjän rekisteröityä käyttäjäksi. Rekisteröinti edellyttää sähköpostiosoitteen ja salasanan syöttämistä. Rekisteröityneen käyttäjän täytyy aina kirjautua järjestelmään käyttäen sähköpostiosoitetta ja salasanaa voidakseen käyttää sitä. Palvelinpäässä käyttäjien salasanoihin lisätään ns. suola ja yhdistelmästä muodostetaan tiiviste käyttäen SHA-256 algoritmia. Algoritmilla muodostettu tiiviste tallennetaan MongoDB-tietokantaan. Käyttäjän kirjautuessa kirjautumisnäytöllä annettu salasana viedään yllä mainitun prosessin läpi, saatua tiivistettä sekä sähköpostiosoitetta verrataan tietokannassa oleviin tietoihin. ”Suolaamalla” ja tiivistämällä salasana, mahdollistetaan ettei käyttäjän salasanaa voida suoraan palauttaa tietokannasta.

Tapauksissa, joissa käyttäjä on unohtanut salasanansa, voidaan se palauttaa käyttäen kirjautumisnäkymässä olevaa ”Palauta salasana” painiketta. Salasanan palautuksessa käyttäjän sähköpostiin lähetetään linkki, joka ohjaa salasanan palautussivulle. Salasanan palautussivulla käyttäjä voi asettaa uuden salasanan.

9.2 XSS hyökkäys

Koska asiakasohjelma on upotettu osittain www-sivun rakenteeseen, voidaan siihen myös kohdistaa Cross site scripting-hyökkäyksiä (XSS). XSS-hyökkäyksessä kolmasosapuoli voi yrittää manipuloida asiakasohjelmaa Javascriptia käyttäen. Pahimmassa tapauksessa hyökkääjä voi varastaa käyttäjän sähköpostiosoitteen sekä selkokielisen salasanan.

9.3 Chrome Content Security Policy

Jotta edellisessä osiossa mainittu XSS-hyökkäys sekä monet muut haavoittuvuudet voitaisiin estää, on Google Chrome:en toteutettu sisällön turvallisuuspolitiikka (CSP).

Yleisellä tasolla CSP toimii musta- ja valkolistamekanismina resursseille, joita laajennus lataa tai suorittaa. Järkevän turvallisuuspolitiikan määritteleminen laajennukselle laittaa kehittäjän miettimään, mitä resursseja laajennus tarvitsee. Turvallisuuspolitiikkaan määritetyt resurssit ovat ainoita resursseja, joihin selain antaa pääsyn.
CSP määritykset tehdään laajennuksen manifest.json-tiedostoon (Google 2017). CSP määrittää vahvasti, mihin resursseihin tausta-skriptillä on oikeus. Sisältö-skripteihin CSP:llä ei yleisesti ottaen ole vaikutusta.

9.4 Asiakas–palvelin yhteyden turvaaminen

Asiakkaan ja palvelimen välisessä yhteydessä käytetään HTTP-protokollaa. HTTP-protokolla on salaamaton selkokielinen protokolla. Yhteysväli voidaan kuitenkin salata vaihtamalla HTTP-protokolla HTTPS-protokollaan (kts. HTTP-yhteys käyttäen TLS:ää). Tämän diplomityön puitteissa ei kuitenkaan toteuteta yhteysvälin salauksen käyttöönottoa. TLS-salauksen käyttöönotto ei silti vaadi suuria muutoksia asiakas- tai palvelinohjelmistoihin.

Asiakasohjelman HTTPS toteutus

Asiakasohjelman muutokset kohdistuisivat ainoastaan määritettyihin palvelimen URL-osoitteisiin, joista korvataan protokolla HTTP-protokollalla HTTPS. URL-muutokset toteutettaisiin tausta-skriptin url-muuttujaan sekä manifest.json tiedoston permissions-osioon. Kevyet asiakasohjelman muutokset johtuvat siitä, että Chrome-selain toteuttaa itsessään HTTPS-protokollan ja suorittaa selainlaajennuksien tietoyhteydet tämän toiminnon kautta.

Palvelinohjelmiston HTTPS toteutus

NodeJs-ympäristössä HTTPS-yhteyden toteutus on monimutkaisempi kuin asiakaspuolella. Jotta NodeJS voisi kommunikoida HTTPS:n avulla, täytyy ottaa käyttöön tls-moduuli sekä luoda yksityinen avain ja hankkia varmenne. Suojatun yhteyden (HTTPS) salausparametrit (yksityinen avain ja varmenne) määritetään ohjelmakoodissa palvelimen käynnisty- osiossa.

HTTP yhteys käyttäen TLS:ää

HTTP yhteys käyttäen TLS:ää eli HTTPS on HTTP-protokollan ja TLS-protokollan yhdistelmä, jossa HTTP-protokollan sanomat salataan ja lähetetään käyttäen TLS-proto-

kollaa. HTTPS-protokollassa TLS toimii siis omana lisäkerroksen ja ei muuta HTTP-protokollan toimintaa (IETF 2000)

TLS protokolla

TLS-protokollan pääasiallinen tehtävä on turvata yksityisyys ja tiedon eheys kahden keskenään viestivän sovelluksen välillä. TLS käyttää symmetristä salausta (kuten DES, RC4, SCH jne.) yhteyden salaamiseen. Jokaiselle yhteydelle luodaan myös yksilölliset avaimet. Jokainen yhteys on myös luotettava, koska jokaisesta viestistä on luotu tiiviste algoritmilla (SHA, MD5 jne.)(IETF 2006).

10 JULKAISU

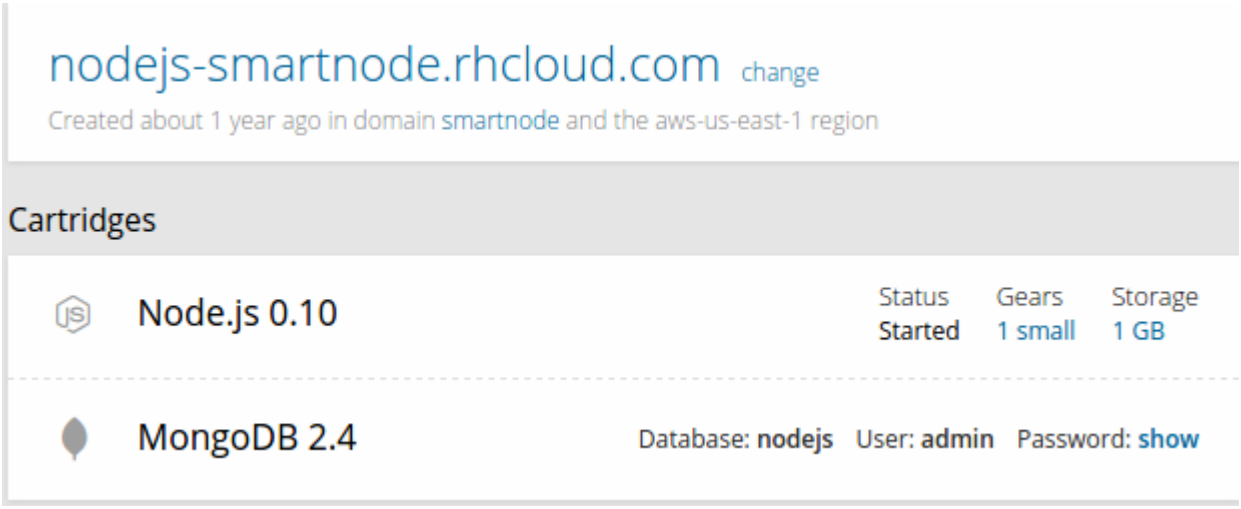
Tämän työn tavoitteisiin ei kuulu järjestelmän julkaisu, mutta jos se julkaistaisiin, kuvailen tässä osiossa millainen julkaisuprosessi olisi. Julkaisuprosessi on jaettu kahdeksi osaksi, ensimmäisessä osassa kuvataan palvelinympäristön julkaisua ja toisessa osassa käydään läpi asiakasohjelman (Chrome selainlaajennus) julkaisua.

10.1 Palvelinympäristö

Järjestelmän suunnittelu- ja toteutusvaiheissa palvelinympäristönä on toiminut Red Hat:in OpenShift Online-pilviympäristö (Openshift 2017), joka on mahdollistanut palvelinosion helpon päivittämisen ja ylläpidon, automatisoimalla ympäristön hallinnan ja skaalaamisen. OpenShift on suunniteltu varsinkin Startup-käyttöön. Se toimii PaaS-periaatteella (Platform as a Service).

OpenShift online-arkkitehtuuri

Openshift Online:n arkkitehtuuri muodostuu palvelinalustasta, jonka päällä toimivat sovelluksen (eng. Applications). Sovellukset muodostuvat kaseteista (Kuva 23) (eng. Cartridges). Kasetit ovat tässä tapauksessa NodeJs-ympäristö sekä MongoDB-tietokanta. Jokainen kasetti voi käyttää useampaa Gear:ia (vapaasti suomennettuna ratas). Gear:it ovat ohjelmakoodille tarkoitettuja suojattuja ympäristöjä. Jokaiselle gear:ille on varattu tietty määrä CPU-, muisti-, levyresursseja sekä verkkokaistaa.



Kuva 23: Kehitysympäristö OpenShift online ympäristössä

Julkaisu

OpenShift ympäristön ollessa helposti skaalattavissa toimisi se myös hyvin tuotantokäytössä. Koska ympäristö on ollut käytössä suunnittelu- ja toteutusvaiheissa, on siirtyminen tuotantokäyttöön helppoa. Tuotantoympäristön rakentaminen tapahtuisi luomalla uusi ympäristö OpenShift Online-pilviympäristöön ja kopioimalla sinne ohjelmakoodit kehitysympäristöstä.

10.2 Chrome selainlaajennus

Koska järjestelmän asiakasohjelma on toteutettu Google Chrome laajennuksena, on sen viralliselle julkaisulle hyvin vähän vaihtoehtoja. Virallisena ja ”de facto” julkaisuväylänä toimii Googlen Chrome Web Store. Toinen mahdollinen väylä on laajennuksen jakaminen omalta web-sivulta. Chrome Web Store tekee laajennuksen jakamisen suurelle yleisölle helpommaksi, sekä tuo käyttäjille jonkinlaisen turvatakuun ja luottamuksen laajennusta kohtaan. Chrome Web Store mahdollistaa myös monia maksujärjestelmiä ja rahoitusmalleja,joista kuvaan seuraavaksi muutamia.

Rahoitusmallit

Ensimmäisenä rahoitusmallina on ”Freemium” eli malli joka mahdollistaa käyttäjille kokeiluversion lataamisen. ”Freemium”-mallissa käyttäjille voidaan ehdottaa lisäominaisuuksia maksua vastaan tai rajoittaa laajennuksen ilmainen käyttö tietyksi aikajaksoksi.

Toisena Web Storen mahdollistamana rahoitusmallina on kertamaksu. Kertamaksu-mallissa käyttäjä tekee kertalaatuisen maksusuoritteen ja saa peruuttamattoman käyttöoikeuden laajennukseen.

Web Store tarjoaa myös mahdollisuuden määräaikaiseen tilaukseen. Määräaikaisessa tilausmallissa käyttäjä maksaa vain määräaikaisesta käyttöjaksosta.

Google Web Store mahdollistaa myös selainlaajennusten ilmaisen jakamisen jolloin käyttäjät eivät maksa laajennuksesta mitään. Ilmaiseen mallin voidaan lisätä myös mainoksia.

Tämän työn tuloksena tehdyn järjestelmän kaltaiselle tuotteelle parhaana rahoitusmallina voidaan pitää määräaikaista tilausta. Määräaikainen tilaus mahdollistaa joustavimman rahoituksen siten, että käyttäjämäärän kasvaessa voidaan kattaa palvelinkapasiteetin lisääminen korottamalla määräaikaisten tilausten maksuja.

Julkaisu

Selainlaajennuksen julkaiseminen Chrome Web Store:en on erittäin suoraviivainen prosessi. Julkaisu tapahtuu seuraavissa vaiheissa (Google 2017):

- Selainlaajennuksen paketointi zip-tiedostomuotoon.
- Google kehittäjätilin luominen.
- Sovellus lataaminen Googlen palvelimelle.
- Rahoitusmallin valinta
- Sovelluksen tietojen lisääminen (kuvaus, kategoria, kuvakaappaukset jne.).
- Sovellustunnisteen (eng. App ID) hankkiminen.
- OAuth avaimen hankkiminen.
- Kehittäjämaksun maksaminen.
- Sovelluksen julkaisu.

11 JATKOKEHITYS

Järjestelmän perustoiminnot on toteutettu toimiviksi ja helppokäyttöisiksi, mutta jatkokehitykseen on vain ”taivas rajana”. Toimintoja voidaan lisätä melkein rajattomasti sekä käytettävyyttäkin voidaan parantaa. Tässä osiossa on tarkoitus pohtia mahdollisia lisätoimintoja sekä käytettävyyden kehitysmenetelmiä.

11.1 Linkkien kohteen sisällön arviointi

Tällä hetkellä suodatuslaajennus toimii vain käyttäjän selaaman ja näkemän web-sivun sisällä. Tätä voidaan helposti muuttaa niin, että epähalutusta sisällöstä varoitetaan jo ennen sivulta toiselle siirtymistä. Tällä ennalta varoittamisella tarkoitetaan mahdollisuutta näyttää näkyvällä web-sivulla olevissa linkeissä tieto linkin kohdesivun sisällön epähaluttavuudesta. Ominaisuus olisi erittäin hyödyllinen esimerkiksi uutissivustoilla, joissa ”raflaavilla” otsikoilla pyritään houkuttelemaan käyttäjiä lukemaan uutisia. Tämä ennaltaehkäisevä ”varoitusjärjestelmä” voisi toimia seuraavalla tavalla:

1. Käyttäjä siirtyy selaimella uutissivustolle.
2. Selainlaajennuksen ollessa aktiivinen selattavalla sivulla se kerää sivun DOM-puurakenteesta linkkielementit.
3. Selainlaajennus lähettää linkkielementeissä olevat URL-osoitteet palvelimelle yksi kerrallaan.
4. Palvelin hakee URL-osoitteita vastaavat sivut kunkin osoitteen lähdepalvelimelta.
5. Palvelin muuttaa jokaisen hakemansa HTML-sivun DOM-puurakenteeksi.
6. Palvelin etsii DOM-puurakenteesta tekstisisältöelementit jotka kuuluvat käyttäjän suodatettavaksi valitsemiin CSS-luokkiin.
7. Palvelin tekee tekstielementtien tekstisisällölle luokittelun.
8. Palvelin palauttaa selainlaajennukselle linkkielementtikohtaisen sisältöluokituksen.
9. Selainlaajennus merkitsee linkit palvelimelta saadun luokituksen mukaisesti
10. Epähaluttuun sisältöön osoittavat linkit merkitään tai piilotetaan riippuen käyttäjän valinnasta (suodata tai korosta)

Tämä ennakkovaroitusjärjestelmä voisi toimia niin, että linkkien kohdesivujen linkit käydään läpi automaattisesti. Ongelmaksi edellä mainitussa tavassa muodostuisi se että mikä on sopiva ”etsintäsyvyys” eli montako linkkiä seurataan.

11.2 Dynaamisesti päivittyvän sisällön suodatus

Järjestelmä on suunniteltu niin, että käyttäjä joko aktivoi suodatuksen sivulla käydessään tai se aktivoituu automaattisesti käyttäjän aiempien määritysten perusteella web-sivun latauduttua. Tämä mahdollistaa myös tilanteita, joissa suodatus voi jättää tekstiä suodattamatta. Tällaisia tilanteita tapahtuu, jos web- sivun sisältöä päivitetään dynaamisesti. Esimerkkinä voidaan pitää sosiaalisen median palvelun viestien ilmestymistä muiden käyttäjien toimesta. Tähän ongelmaan siis järjestelmä ei tällä hetkellä pysty vastaamaan. Jotta suodatusjärjestelmään voitaisiin käyttää myös sosiaalisen median viestien suodatuksessa, täytyy siihen lisätä toiminto DOM- puurakenteen muutosten tarkkailuun. Ongelmaksi muodostuu suodatusjärjestelmän tekemien omien DOM- puurakenteen muutosten huomiotta jättäminen. Järjestelmän omat DOM- puurakenteen muutokset voivat aiheuttaa ylimääräisien kyselyiden lähettämistä palvelimelle. Edellä mainittu ongelma on syy sille, miksi suodatusjärjestelmää ei suunniteltu vielä dynaamisesti päivittyvän sisällön suodatukseen.

12 YHTEENVETO

Työssä käytiin läpi selattavan web-sisällön käyttäjäkohtaisen suodatusjärjestelmän suunnittelua sekä järjestelmän toteuttaminen.

Työssä käytiin myös läpi aiheeseen liittyvää teoriaa sekä omaa pohdintaa. Teoriasta kuvattiin vain työn kannalta olennaiset asiat. Nämä asiat olivat alkutilanne, käytetty tutkimusmenetelmä, selattavan web-sisällön määritys, tekstinluokittelu sekä tiedon valintamekanismit. Alkutilanteen ongelmana tiivistyy lauseeseen ”Miten suodattaa tekstisisältöä luettavalta web-sivulta”. Jotta ongelma saatiin ratkaistua, täytyi etsiä sopivat menetelmät tekstitiedon poimimiseen web-sivulta (oikeiden elementtien löytäminen) sekä sopiva tekstinluokittelumenetelmä. Oman pohdinnan ja tutkimuksen kautta löysin sopivan ratkaisun tekstin poimintaan. Poiminnassa käytettiin menetelmää, jossa käyttäjä valitsee itse web-sivulta tekstielementit selainlaajennuksen ehdottamien joukosta. Tekstinluokittelu menetelmistä päädyin multinomiaaliseen Bayes-malliin. Menetelmään päädyttiin, koska se on toimiva, tehokas ja helppo toteuttaa. Tutkimusmenetelmän suuntaviivat toimivat hyvinä raameina tämän työn toteuttamisessa.

Työssä kuvattiin järjestelmän arkkitehtuuri mahdollisimman tarkasti, mutta kuitenkin keskittymättä selittämään erilaisten kirjastojen yksityiskohtaisia hienouksia. Työssä päädyttiin asiakas-palvelin arkkitehtuuriin. Asiakas- ja palvelinosien toteutuksessa päädyttiin laajasti käytettyihin avoimen lähdekoodin kirjastoihin.

Työssä kuvattiin myös järjestelmän keskeiset prosessit. Keskeiset prosessit ovat järjestelmän tärkeitä tapahtumaketjuja. Keskeisiä prosesseja ovat:

- Rekisteröitymisprosessi
- Kirjautumisprosessi
- Sisällönluokitteluprosessi
- Suodatusprosessi
- Sisällönvalintaprosessi
- Kirjautuminen ylläpitäjän käyttöliittymään

Työssä kuvattiin järjestelmän käyttöliittymät ja näkymät sekä kehitysprosessin alkuvaiheen hahmotelmat aina viimeisteltyyn lopulliseen tuotokseen. Käyttöliittymä suunnitte-

lu oli työläimpiä vaiheita. Työlääksi vaiheen teki erityisesti se että järjestelmän käyttäjäkuntana voisivat toimia kaikki internetin peruskäyttäjät.

Koska järjestelmän mahdollinen käyttäjäryhmä on laaja, täytyi myös tietoturvaan kiinnittää erityisesti huomiota. Tämän vuoksi pohdittiin myös järjestelmän tietoturvaa sekä siihen liittyviä kysymyksiä ja ongelmia. Mahdollisiksi ongelmiksi havaittiin XSS-hyökkäyksen mahdollisuus, sekä järjestelmässä käytettävä salaamaton HTTP-yhteys. Näihin ongelmiin löydettiin ratkaisut Chromen omasta tietoturvapolitiikasta sekä HTTPS-yhteydestä.

Työssä käytiin läpi myös järjestelmän mahdollinen julkaiseminen. Järjestelmän julkaisu ei ole ollut tämän työn tavoite joten julkaisu käytiin läpi hyvin yleisellä tasolla. Julkaisua pohdittiin sekä palvelimen että asiakasohjelman osalta.

Tutkimusmenetelmäosiossa määritettyjen suuntaviivojen mukaisesti lopputuloksena on saatu toimiva ja ongelman ratkaiseva järjestelmä, jonka toteutuksessa ja tutkimisessa on käytetty tehokkaimpia ratkaisuja.

LÄHTEET

Google(2017), Publish in Chrome Web Store. Saatavilla https://developer.chrome.com/webstore/publish. Viitattu 23.2.2017

Hevner Alan R., March Salvatore T., Park Jinsoo, Ram Sudha (2004),DESIGN SCIENCE IN INFORMATIONSYSTEMS RESEARCH. MIS Quarterly Vol. 28 No. 1

IETF(2006),RFC4346. Saatavilla https://tools.ietf.org/html/rfc4346. Viitattu 7.1.2017

Internet Engineering Task Force (IETF)(2000),RFC 2818. Saatavilla https://tools.ietf.org/html/rfc2818. Viitattu 6.1.2017

Node.js Foundation(2017), NodeJS . Saatavilla https://nodejs.org/en/about/. Viitattu 3.1.2017

Manning Christopher D., Raghavan Prabhakar ja Schütze Hinrich (2008), Introduction to Information Retrieval,13. Saatavilla:http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf. Viitattu 12.2.2017

Openshift(2017), Openshift. Saatavilla https://www.openshift.com/features/. Viitattu 22.2.2017

Raschka Sebastian (2014), Naive Bayes and Text Classification. Saatavilla https://arxiv.org/pdf/1410.5329.pdf. Viitattu 2.2.2017

W3C(2014), HTML5 A vocabulary and associated APIs for HTML and XHTML. Saatavilla https://www.w3.org/TR/html5/sections.html#the-article-element. Viitattu 15.01.2017

W3C(2016), DOM. Saatavilla https://www.w3.org/DOM/#ressources. Viitattu 13.1.2017

Wikipedia(2016), Suosittelujärjestelmä.Saatavissa https://fi.wikipedia.org/wiki/Suosittelujärjestelmä. Viitattu 11.1.2017

Wikipedia(2017),Bag of words model. Saatavilla https://en.wikipedia.org/wiki/Bag-of-words_model. Viitattu 13.2.2017