LAUREA
AMMATTIKORKEAKOULU
Yhdessä enemmän

Verkkosivuston sisällön kategorisointi ja video-keskustelupalvelun integrointiselvitys – Case Seniori365.fi-verkkopalvelu

Hänninen, Henri

2016 Laurea

Laurea-ammattikorkeakoulu

Verkkosivuston sisällön kategorisointi ja videokeskustelupalvelun integrointiselvitys – Case Seniori365.fi-verkkopalvelu

Henri Hänninen
Tietojenkäsittely
Opinnäytetyö
Kesäkuu, 2016

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittely

Hänninen, Henri

Verkkosivuston sisällön kategorisointi ja videokeskustelupalvelun integrointiselvitys – Case Seniori365.fi-verkkopalvelu

Vuosi 2016 Sivumäärä 44

Tämän toiminnallisen opinnäytetyön tavoitteena on toteuttaa jatkokehityssuunnitelma Seniori365.fi-verkkopalvelun jatkokehittäjille. Jatkokehityssuunnitelma sisältää kaksi erilaista kokonaisuutta. Suunnitelman ensimmäinen osio sisältää ohjeistuksen kuntakohtaisen kategorisointitoiminnallisuuden toteuttamiseksi. Jatkokehityssuunnitelman toinen osio esittelee teknologiat, joita on mahdollista käyttää videokeskustelutoiminnallisuuden toteuttamiseksi. Teknologioiden esittelyn lisäksi opinnäytetyössä esitellään kolme valmista videokeskustelutoteutusta sekä sopivin tapa videokeskustelutoiminnallisuuden lisäämiseksi Seniori365.fi-verkkopalveluun.

Ohjeistus kuntakohtaisen kategorisointitoiminnallisuuden toteuttamiseksi tehtiin mahdollisimman yksityiskohtaiseksi, jotta toiminnallisuuden toteuttaminen olisi jatkokehittäjille heidän teknisestä taustastaan riippumatta mahdollisimman helppoa ja suoraviivaista. Opinnäytetyössä on myös osittaisia esimerkki-implementaatioita, joita on mahdollista hyödyntää toiminnallisuuden toteuttamisessa.

Opinnäytetyössä esiteltiin eri teknologiavaihtoehdot, joita on mahdollista käyttää videokeskustelutoiminnallisuuden toteuttamiseen. Eri teknologiat esiteltiin riittävällä tarkkuudella, mutta opinnäytetyössä keskityttiin yksityiskohtaisesti potentiaalisimpaan teknologiaan. Opinnäytetyössä esiteltiin lisäksi kolme erilaista suosittua selainpohjaista videokeskustelutoteutusta ja verrattiin niiden sopivuutta Seniori365.fi-verkkopalveluun. Vaatimusmäärittelyn ja valmistoteutuksien vertailujen perusteella opinnäytetyössä suositeltiin Seniori365.fi-verkkopalvelulle sopivinta tapaa videokeskustelutoiminnallisuuden toteuttamiseksi.

Asiasanat: seniori365, seniori, verkkopalvelu, drupal, sisällönhallintajärjestelmä, kategorisointi, videokeskustelupalvelu, webrtc

Laurea University of Applied Sciences **Abstract**
Bachelor's Degree Programme in Business Information Technology

Hänninen, Henri

Categorizing the Website Content and Integration Study of Video Chat Service – Case Seniori365.fi

Year 2016 Pages 44

The purpose of this thesis was to create development plan for the web developers of Seniori365.fi–website. The thesis is designed mainly for the web developers of Seniori365 web service, who understand the technologies and terminology used in the development of a dynamic website. The main points and conclusions of thesis are written as clearly as possible, so that also the non-technical readers can understand the purpose of this thesis.

The development plan is consisted of two very different sections. The first section of the development plan provides a detailed guidance for implementing a content categorizing feature on the website. The categorizing feature is used to filter content based on the municipality. The first section also contains an example implementation, which can be used as a basis for the actual production environment implementation.

The second section of the development plan focuses on different technologies, which can be used to build and implement a video chat feature on Seniori365 website. The purpose of the second section is to benchmark different technologies and provide the best solution which can be used to implement the video chat service on Seniori365 website. The thesis does not contain an implementation plan for the video chat service.

Keywords: seniori365, senior, web service, drupal, content management system, categorizing content, video chat service, webrtc

Sisällys

1 Johdanto

Opinnäytetyön tavoitteena on luoda kaksiosainen tekninen jatkokehityssuunnitelma Seniori365.fi –verkkopalvelun jatkokehittäjille. Suunnitelman ensimmäinen osa sisältää yksityiskohtaisen ohjeistuksen kuntakohtaisen sisällön kategorisoimiseksi Drupal-sisällönhallintajärjestelmällä. Suunnitelman jälkimmäinen osio käsittelee eri teknologiavaihtoehtoja, joita on mahdollista käyttää interaktiivisen videokeskustelupalvelun toteuttamiseen. Osion tarkoituksena on myös suositella Seniori365-sivustolle soveltuvinta tapaa videokeskustelupalvelun integroimiseen.

Seniori365.fi –verkkopalvelu on nykyisellään espoolaisille senioreille suunnattu verkkopalvelu, josta löytyy monialaista senioreille kohdistettua tietoa, hyvinvointipalveluita ja –tuotteita sekä viihdettä. Asiakkaan tavoitteena on laajentaa sivustoa eri kuntiin, joten sivustolle on kehitettävä ominaisuus joka mahdollistaa sisällön luonnin ja esittämisen kuntakohtaisesti.

Opinnäytetyön ensimmäisen osion tavoitteena on luoda yksityiskohtainen määrittely tarvittavista muutoksista, jotta Seniori365-palvelu on mahdollista laajentaa myös muiden kuntien käyttöön. Suunnitelmasta käy ilmi tietorakenteisiin ja ulkoasuun tehtävät muutokset. Suunnitelmassa tuodaan esiin erilaisia ratkaisuehdotuksia sekä osittaisia implementointeja ratkaisua vaativiin ongelmakohtiin ja asiakkaan haluamiin muutoksiin. Suunnitelmassa on mukana myös valmiit ratkaisuehdotukset lähdekoodeineen, jotka kuitenkin vaativat tulevilta tekijöiltä kunnollista testaamista sekä implementaatioita tuotantojärjestelmään.

Asiakkaan tavoitteena on implementoida tulevaisuudessa videokeskustelutoiminnallisuus sivustolle. Tämän opinnäytetyön jälkimmäisessä osassa käydään läpi eri teknologiat, joita on mahdollista hyödyntää selainpohjaisen videokeskustelutoteutuksen kehittämisessä. Lisäksi esitellään valmiita kolmannen osapuolen toteuttamia videokeskustelusovellutuksia. Opinnäytetyön jälkimmäisen osan tavoitteena ei ole tarjota valmista ratkaisua tai toteutussuunnitelmaa. Jälkimmäisen osan tavoitteena on suositella potentiaalisinta toteutustapaa asiakkaan toiveiden, eri tekniikkavaihtoehtojen mahdollisuuksien ja rajoitusten perusteella.

2 Opinnäytetyön tausta

Seniori365.fi–verkkopalvelu oli alunperin osa InnoEspoo-hanketta, jonka tarkoituksena oli luoda hyvinvointiin liittyviä palveluinnovaatioita. InnoEspoo-hanke oli Espoon alueen oppilaitosten sekä Espoon kaupungin yhteishanke, jonka rahoittajana toimi Euroopan sosiaalirahasto.

Seniori365.fi–palvelun suunnittelu alkoi alunperin 2013 syksyllä. Palvelun tekninen kehittäminen alkoi kesäkuussa 2014. Palvelun teknisestä kehittämisestä vastaavat Laurea-ammattikorkeakoulun opiskelijat. Palvelu oli alunperin suunniteltu lähinnä Espoon alueen senioreiden käyttöön, joten palveluun ei ole toteutettu kategorisointia, joka mahdollistaisi sisällön luomisen tai suodattamisen kuntakohtaisesti.

Opinnäytetyön toimeksiantajalle on kuitenkin syntynyt tarve muuttaa palvelu tukemaan kuntakohtaista sisältöä. Kuntakohtaisen sisällön tuen toteuttaminen mahdollistaisi palvelun käyttämisen myös muiden kuntien alueella. Ilman kuntakohtaisen sisällön tukea palvelun laajentaminen muihin kuntiin on haastavaa, sillä huomattava osa palvelun sisällöstä on täällä hetkellä hyvin Espoo-keskeistä, eikä se siten suoraan palvele muiden kuntien alueella asuvia käyttäjiä.

Asiakkaan keskeisimpänä tavoitteena on luoda palveluun tuki usealle eri kunnalle. Palveluun pitäisi pystyä jatkossa lisäämään kuntakohtaista sisältöä sekä käyttäjillä pitää olla saatavilla helppokäyttöinen tapa suodattaa sisältöä kuntakohtaisesti. Tässä opinnäytetyössä keskitytään ohjeistamaan yksityiskohtaisesti miten sivustolle voidaan lisätä tuki usealle kunnalle. Lisäksi opinnäytetyössä käydään läpi muutamia videopalveluita ja niiden integroimisen helppoutta sekä soveltuvuutta Seniori365-sivustolle.

2.1 Toimeksiantaja

Seniori365.fi-palvelun ylläpito ja kehitys siirtyi Laurea Ammattikorkeakoululle InnoEspoo-hankkeen päätyttyä. Palvelun kehityksestä on vastuussa Laurean aluepalvelupäällikkö, joka toimii myös tämän opinnäytetyön toimeksiantajana.

2.2 Opinnäytetyön kohderyhmä

Opinnäytetyö toimii teknisenä suunnitelmana Seniori365.fi –palvelun jatkokehittäjille. Opinnäytetyötä kirjoitettaessa on oletettu kohderyhmän olevan tietoinen verkkosivustojen suunnittelun ja toteutuksen periaatteista sekä sanastosta. Opinnäytetyö on kuitenkin pyritty kirjoittamaan mahdollisimman selkeästi, jotta myös vähemmän tekniikkaorientoituneet lukijat pystyvät sisäistämään suunnitelmien ydinkohdat.

3 Käsitteet

Tässä kappaleessa esitellään opinnäytetyössä käytettyä sanastoa ja täsmennetään termien merkitystä.

Drupal	Seniori365.fi–palvelun kehittämisessä käytetty sisällönhallintajärjestelmä.
Sisältötyyppi	Tapa kategorisoida erilaisia sisältöjä. (eng. content type)
Pohja	Tiedosto, joka sisältää ulkoasun generoimiseen tarkoitettua HTML -lähdekoodia (eng. template)
Teema	Sivuston ulkoasu (eng. theme). Seniori365 –sivustolla on käytössä kustomoitu teema, joka alunperin pohjautuu suosittuun ilmaiseen valmisteemaan.

Rajapintafunktio	Funktio, jota käytetään Drupalin ytimen toiminnallisuuksien muokkaamiseen (eng. hook). Yleensä rajapintafunktio on eräänlainen takaisinkutsufunktio (eng. callback), jolle syötetään objektiviittauksia parametreina (eng. pass-by-reference). Rajapintafunktion parametreja muokataan, jonka jälkeen ydin suorittaa toiminnallisuuden käyttäen muokattuja arvoja. Rajapintafunktiot on aina nimetty seuraavasti {teeman tai moduulin nimi}_{api funktio} (Travis, B. 112-116).
	
	Esimerkki :
	`// Forces Drupal to use custom page template: my-page.tpl.php`
	`function seniori365_preprocess_page(&$variables){`
	`  $variables['theme_hook_suggestions'] = array('my_page');`
	`}`

Suodatuskriteeri	Views –moduulin asetus, jolla rajataan tietokantahakua kenttäkohtaisesti. Käytännössä suodatinkriteerin lisääminen muodostaa WHERE lausekkeen generoitavaan SQL –kyselyyn. (eng. Filter criteria)

4 Seniori365.fi –verkkopalvelu

Seniori365.fi –palvelu koostuu useasta eri alaosiosta. Tässä osiossa käydään läpi Seniori365.fi –palvelun keskeisimmät alaosiot ja niiden toiminnallisuudet. Kyseiset alaosiot ovat kaikki nähtävillä myös sivuston päävalikossa.

Etusivu tarjoaa tietoa itse palvelusta ja sen käyttötarkoituksesta. Etusivulta myös löytyy palstoja, johon nostetaan näkyviin uusimpia palveluita, tuotteita sekä ajankohtaisia tapahtumia. Etusivulta löytyy myös linkitykset sosiaalisiin medioihin sekä pikalinkkejä sivuston muihin osioihin.

Palvelut ja tuotteet alasivut ovat toiminnallisuuksiltaan lähes identtisiä. Niiden tehtävänä on tarjota käyttäjälle käyttöliittymä verkkopalveluun lisättyjen hyvinvointipalvelujen ja -tuotteiden hakemiseen sekä selaamiseen. Palvelut-osio ja Tuotteet–osio ovat sivuston tärkeimmät osa-alueet.

Yleistietoa-osiosta löytyy nimensä mukaisesti tietoa ja ohjeita senioreille, riippumatta heidän sen hetkisestä asuinkunnastaan tai asuinmaastaan. Osion keskeisin tarkoitus on koota yhteen paikkaan senioreiden hyvinvointiin liittyvää tietoa useista eri lähteistä. Osion sisältöä on koottu haettu muun muassa Kelan ja erilaisten järjestöjen sivuilta. Artikkelien yhteydessä on aina myös mainінta alkuperäisestä lähteestä.

Aktiviteetit-osio sisältää monenlaista viihteeseen ja vapaa-aikaan liittyvää sisältöä, kuten videoita, linkkejä verkkolehtiin ja pelisivustoille, keskustelufoorumi, ruokaohjeita sekä erilaisia muisteluun ja tarinan kerrontaan liittyvää sisältöä. Aktiviteetit-osion sisältö on myös luonteeltaan sellaista, joka soveltuu kaikille sivuston käyttäjille eikä tarvetta kuntakohtaiselle lokalisoinnille ole.

Omaishoito-osiosta löytyy omaishoitajille suunnattua yleistä tietoa sekä kuntakohtaista tietoa. Yleistä tietoa ovat muun muassa tiedot omaishoitajuudesta sekä omaishoitajahaastattelut. Paikallinen sisältö taas koostuu alueellisten omaishoitajajärjestöjen esittelystä, kuntakohtaisten omaishoitosäädösten esittelystä, ohjeistuksesta omaishoitotuen hakemiseen sekä tärkeistä yhteystiedoista.

Vapaaehtoistyö-osiosta löytyy laaja-alaista tieto vapaaehtoistyöhön liittyen. Osiossa käydään yleisesti läpi vapaaehtoistyötä sekä siihen liittyviä oikeuksia ja velvollisuuksia. Osion alisivustoilta löytyy myös tietoa erilaisista vanhustyöhön liittyvistä vapaaehtoisjärjestöistä sekä kokemuksia vapaaehtoistyötä tekeviltä henkilöiltä.

5 Palvelun nykyinen tekninen toteutus

Seniori365.fi –palvelu on toteutettu Drupal-sisällönhallintajärjestelmällä. Drupal-sisällönhallintajärjestelmää käytetään erityisesti hallinnollisten organisaatioiden sekä isojen yritysten verkkopalveluissa. Drupal on kolmanneksi suosituin julkaisujärjestelmä ja sen suosio on ollut hienoisessa kasvussa (W3Techs, 2016).

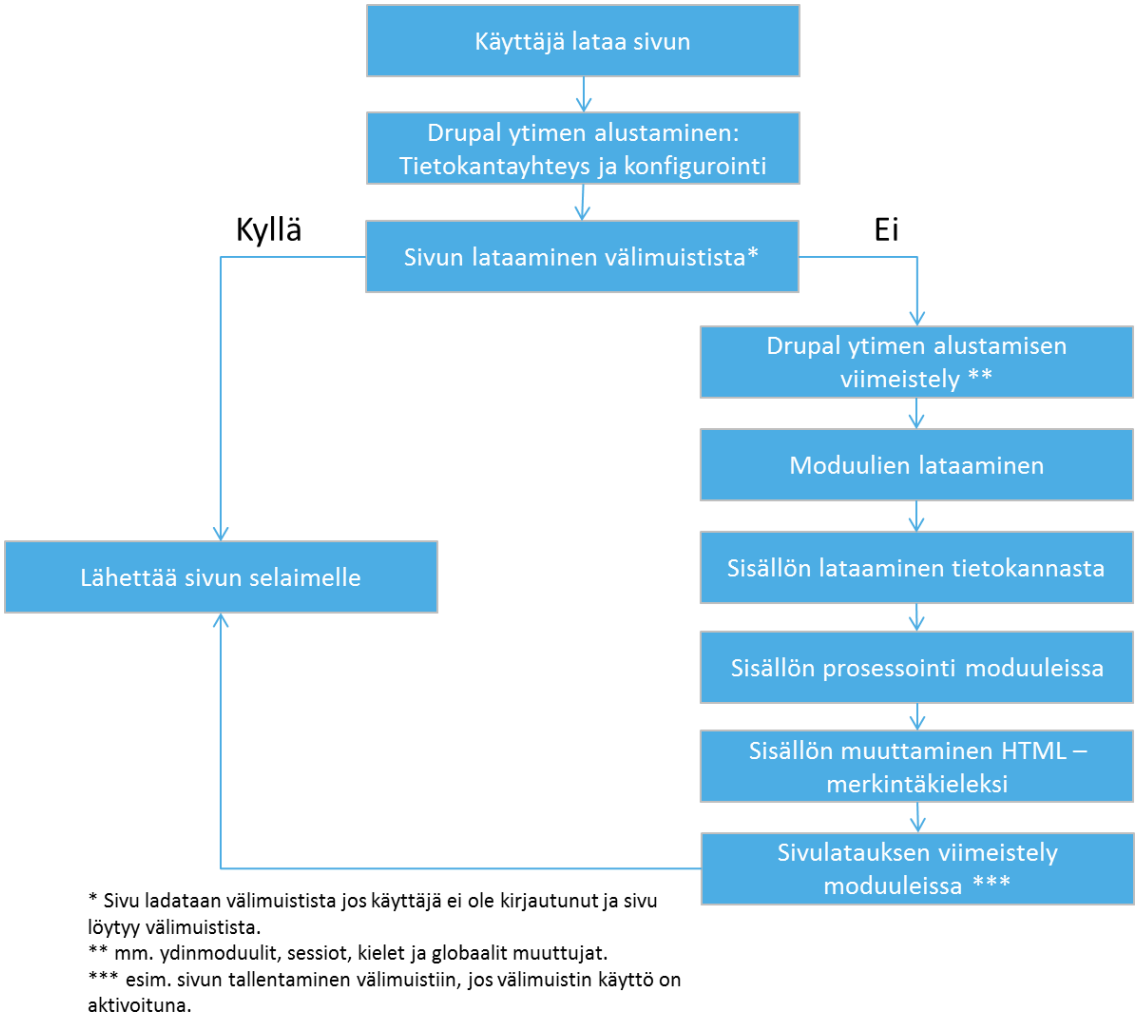
Sivustolla käytetty teema on suurelta osin kustomoitu vaikka sen ydin perustuukin suosittuun valmispohjaan. Kustomoitu ratkaisu on toimiva, sillä se mahdollistaa ulkoasun vapaan jatkokehittämisen ilman riippuvuutta kolmannesta osapuolesta. Seniori365.fi –palvelussa on pyritty huomiomaan esteettömyys, helppokäyttöisyys sekä selainyhteensopivuus mahdollisimman kattavasti. Sivuston ulkoasusta on tehty responsiivinen vaikka kehityksessä eniten huomiota onkin kiinnitetty työpöytänäkymään.

Seniori365.fi –verkkopalvelu tukee muun muassa vanhoja Internet Explorer 8 selaimia, sillä tärkeillä yhteistyökumppaneilla sekä potentiaalisilla ja nykyisillä käyttäjillä on vielä käytössä Windows XP käyttöjärjestelmä, johon ei ole mahdollista saada uudempaa Microsoftin selainta. IE8 tuki aiheuttaa joitakin rajoitteita sivuston ulkoasun suunnittelulle, sillä kaikkia uudempia tekniikoita ei voida hyödyntää. Esimerkiksi monet hyvinkin yksinkertaiset asiat, kuten liukuvärit tai elementtien kulmien pyöristämiset eivät toimi IE8:ssa.

Drupal–järjestelmän toimintalogiikka on proseduraalinen (Wilkins, J. 9). Proseduraalinen ohjelmointi on hyvin prosessikeskeistä. Proseduraalisessa ohjelmoinnissa toiminnallisuudet tapahtuvat proseduureissa, Drupalin tapauksessa funktioissa. Funktiot on myös usein jaoteltu loogisiin, toisistaan mahdollisimman riippumattomiin kokonaisuuksiin eli moduuleihin. (Techopedia, 2016). Drupalin toimintalogiikka on koostettu useista funktioista, joita järjestelmä kutsuu vuorollaan. Kun funktio on lopettanut toiminnallisuutensa suorittamisen, Drupal kutsuu seuraavaa funktioita. Drupal-järjestelmä hyödyntää osin myös objektiorientoitunutta ohjelmointityyliä, mutta suurin osa ytimen toiminnallisuudesta on proseduraalista.

Drupal-sisällönhallintajärjestelmässä esitys- ja logiikkakerros on erotettu toisistaan. Esityskerroksen ja logiikkakerroksen eriyttämisellä mahdollistetaan logiikan ja visuaalisuuden riippumattomuus toisistaan (Wilkins, J. 63). Eriyttämisestä on monia erilaisia hyötyjä, kuten mahdollisuus ulkoasun uudistamiseen ilman että ohjelmiston ydinlogiikkaan tarvitsee tehdä muutoksia, esitettävän datan muokkaaminen ilman HTML-pohjiin tehtäviä muutoksia sekä kehitysprosessien jakaminen osiin ja delegointi eri henkilöille. Esitys– ja logiikkakerroksen täydellistä riippumattomuutta toisistaan ei kuitenkaan ole mahdollista saavuttaa, sillä logiikan ja esittämisen raja on käyttöliittymäkomponentteja kehitettäessä häilyvä (Travis, B. 181).

Drupal on hyvin monipuolinen, mutta samalla myös monimutkainen sisällönhallintajärjestelmä. Drupal-kehittäjän on hyvä sisäistää Drupalin HTTP –kyselyn prosessointilogiikka, ainakin pääpiirteittäin. Tämä helpottaa järjestelmän kehittämistä ja ylläpitoa sekä auttaa hahmottamaan missä vaiheessa eri rajapintafunktioita kutsutaan. Oheisessa prosessikaaviossa on esitelty Drupalin HTTP -kyselyn prosessointilogiikka, joka alkaa aina verkkopalvelimen juurikansiossa sijaitsevasta index.php –tiedostosta (Travis, B. 31).



Kuva 1: Kyselyn prosessointi Drupal -järjestelmässä

Logiikkakerroksen muodostavat Drupalin ydin sekä moduulit. Moduulit ovat olennainen osa Drupal-järjestelmää, sillä Drupalin ydintoiminnallisuudetkin on jaettu modulaarisiin kokonaisuuksiin, ydinmoduuleihin. Moduulit kommunikoivat Drupal-ytimen kanssa rajapintafunktioilla. Nämä rajapintafunktiot ovat tavallisia PHP-funktioita, jotka yleensä prosessoivat funktion parametreina annettua dataa, kuten taulukon arvoja. Rajapintafunktiot eivät yleensä palauta mitään arvoa, vaan toiminnallisuuteen tehtävät muutokset tapahtuvat annettuja parametreja muokkaamalla.

Esityskerroksesta vastaa Drupal-sisällönhallintajärjestelmissä teemat. Teemat koostuvat HTML-pohjista, PHP-koodista, CSS-tyylimäärittelyistä sekä selaimissa ajettavasta Javascript-ohjelmointikielestä. Esityskerrosta käytetään pääasiassa muuttamaan logiikkakerroksen generoima data visuaaliseen, selainten tukemaan muotoon eli HTML-merkintäkieleksi. Esityskerros pystyy myös prosessoimaan dataa rajapintafunktioiden avulla lähes samalla tavalla kuin mo-

duulitkin, mutta esimerkiksi tietokantaan vaikuttavat rajapintafunktiot on sallittu vain moduuleissa.

6 Kuntakohtaisen sisällön tuen toteuttaminen

Palvelun sisällön kuntakohtainen räätälöinti edellyttää järjestelmältä tukea kategorisoida sisältöjä kuntakohtaisesti. Tällä hetkellä palveluita ja tuotteita lukuun ottamatta sisältöä ei voi rajata koskemaan vain tiettyä kuntaa. Järjestelmään on siis ensimmäisenä lisättävä tuki sisältöjen kategorisoinnille kunnittain. Tämä on helpointa toteuttaa lisäämällä haluttuihin sisältötyyppeihin uusi kategoria, joka määrittää kunnan tai kunnat joihin sisältö liittyy. Kategoriavalinta on toteutettava mahdollisimman yksinkertaisesti ja helppokäyttöisesti. Helppokäyttöisyyttä suunniteltaessa on huomioitava sivuston ylläpitäjät, sisällön tekijät sekä varsinaiset loppukäyttäjät.

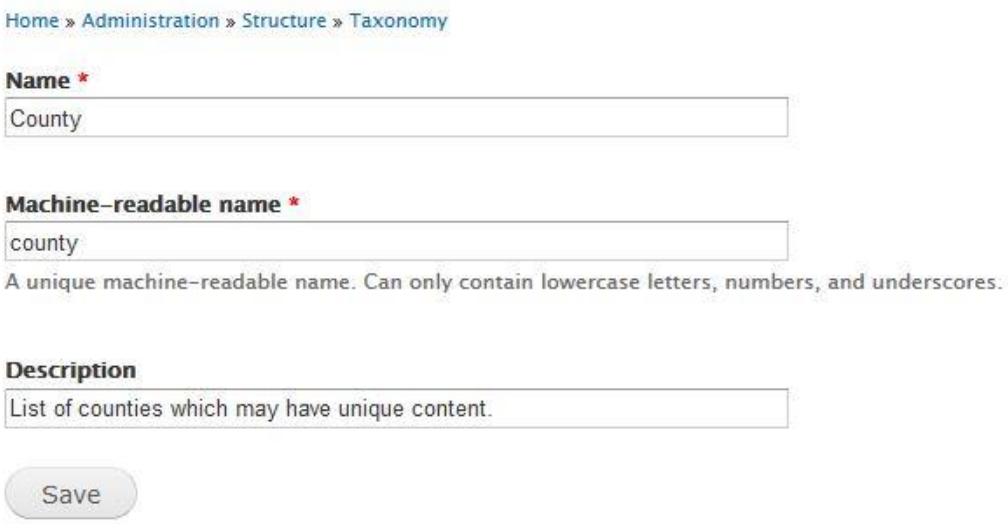
Sisältöjen luominen ja sisällön liittäminen haluttuihin kuntiin on tehtävä mahdollisimman helpoksi, jotta kuka tahansa pystyy tarvittaessa lisäämään sivustolle sisältöä. Sisällöntuottajien on ymmärrettävä, mitä heidän on tehtävä pystyäkseen lisäämään sisältöä.

Valikkoratkaisu jonka avulla loppukäyttäjä määrittää minkä kunnan sisältöä haluaa katsella, on tehtävä intuitiiviseksi käyttää. Käytettävyyttä voi parantaa myös tallentamalla käyttäjän kuntakohtaisen valinnan muistiin, esimerkiksi keksiin tai palvelimen sessioon. Käyttäjän vaihtaessa sivua kuntakohtainen valinta säilyy muistissa ja samaa asetusta käytetään kaikilla osioilla, jotka hyödyntävät kuntakohtaista sisällön jaottelua.

6.1 Kategorian lisääminen

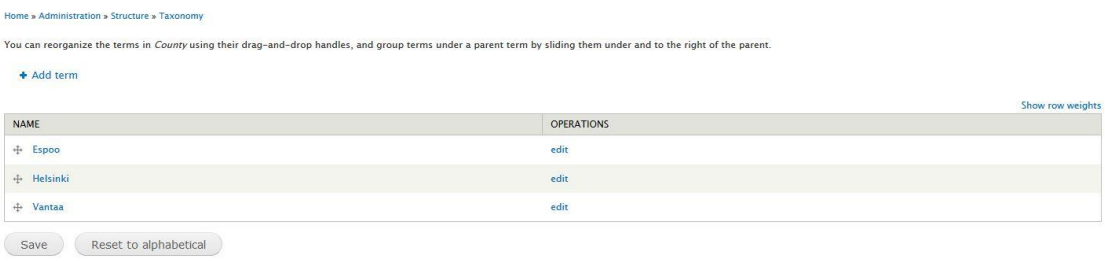
Kuntakohtaisen kategorisoinnin toteuttamisessa ensimmäinen vaihe on uuden kategorian luominen. Luotavaa kategoriaa käytetään määrittämään sisällön kuntarajausta. Tässä opinnäytetyössä kategorian nimeksi on asetettu *County*. Nimi voi tietysti olla mitä tahansa, mutta yleensä ohjelmoinnissa termien englanninkielinen käyttö on parempi vaihtoehto kuin muunkielisten termien käyttö.

Kategoria lisätään Drupalin hallintapaneelin kautta Structure -> Taxonomy -> Add vocabulary valikosta.



Kuva 2: County -nimisen kategorian luominen

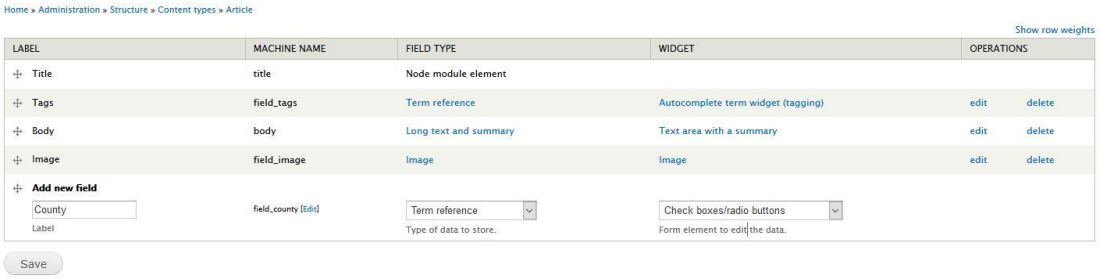
Kategorian lisäämisen jälkeen siihen liitetään kuntien nimet, joihin sisältöä on pystyttävä liittämään.



Kuva 3: County -kategoria johon on lisätty valittaviksi kunniksi Espoo, Vantaa ja Helsinki

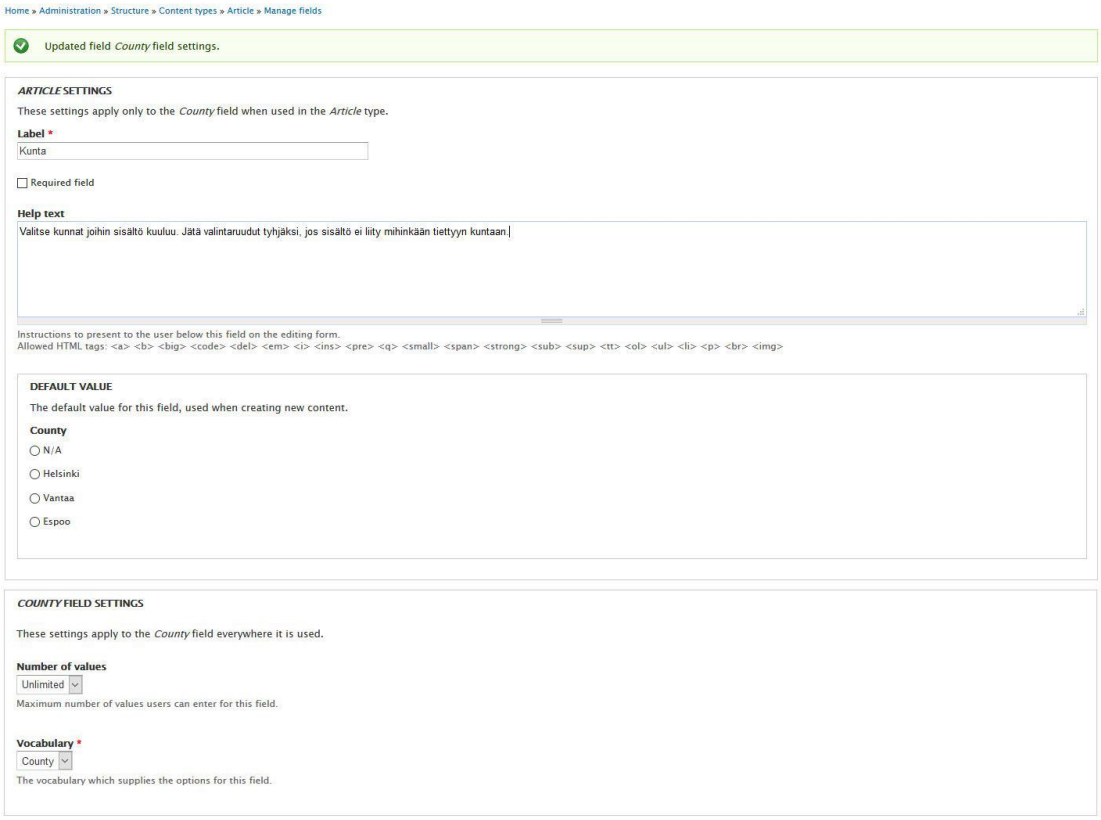
6.2 Kuntavalinnan liittäminen sisältötyyppeihin

Kuntavalinnat on seuraavaksi liitettävä haluttuihin sisältötyyppeihin. Sisältötyyppejä pääsee editoimaan Structure -> Content type -> manage fields valikosta.



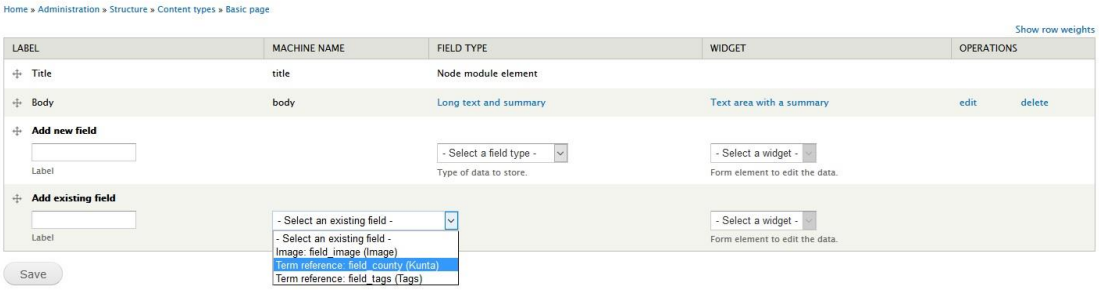
Kuva 4: Kuntavalintakentän luominen

Kuntavalintakentän luomisen jälkeen se on konfiguroitava halutusti. Sisällönsyöttäjien työn helpottamiseksi Label ja Help text –kenttiin on laitettava kuvaavat tekstit, jotta sisällönsyöttäjä tietää mitä sisältöä kyseisiin kenttiin on laitettava. Mikäli sisältöjä pitää pystyä liittämään useisiin kuntiin, on kentän Number of values –arvo asetettava suuremmaksi kuin yksi. Vaihtoehtoisesti valinnan arvoksi voidaan asettaa rajoittamaton. Kun kentän asetukset on tallennettu, sisältöä voidaan luoda kuntakohtaisesti.



Kuva 5: Kuntavalintakentän konfigurointi

Kun kuntavalintakenttä on kerran luotu, se on helppo liittää muihin tarvittaviin sisältötyyppeihin. Olemassa olevan kentän liittäminen sisältötyyppiin tehdään samasta näkymästä kuin uuden kentän lisääminen. Kun kenttä liitetään sisältötyyppiin, se on konfiguroitava halutusti.



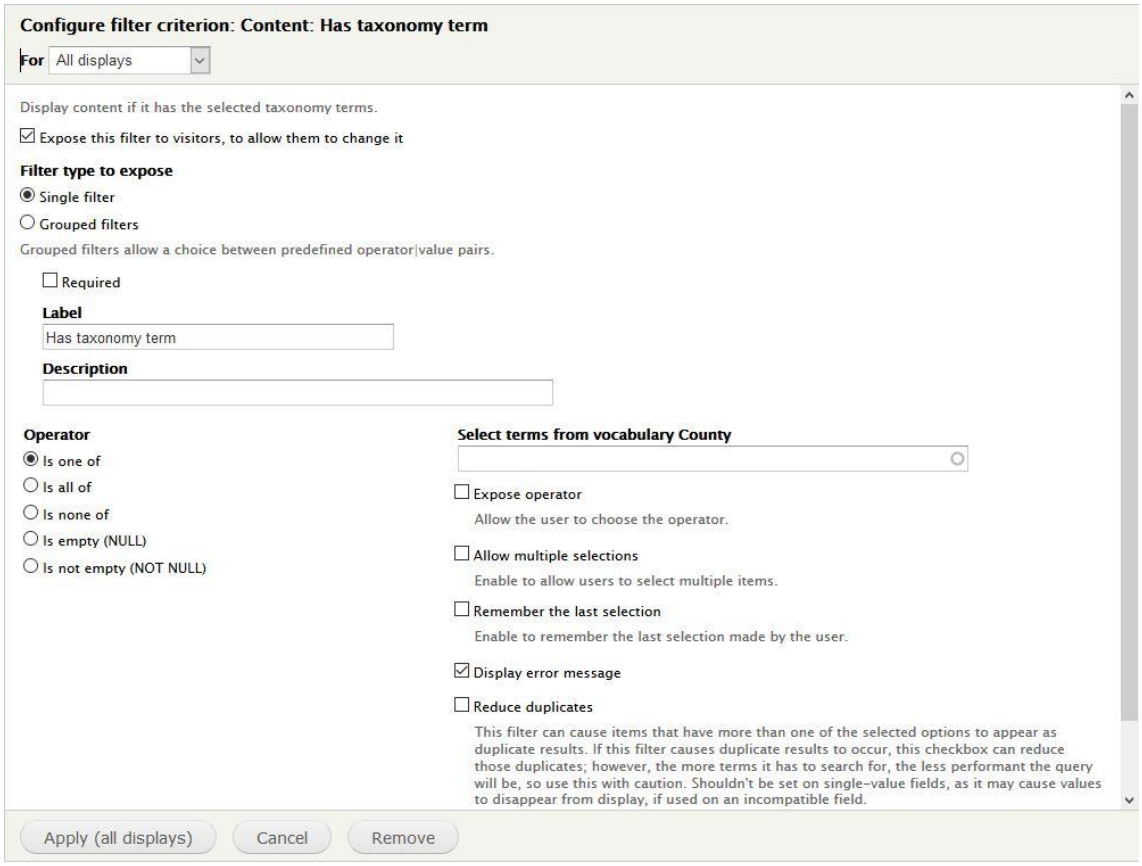
Kuva 6: Olemassa olevan kentän lisääminen sisältötyyppiin

6.3 Sisällön suodattaminen kuntavalinnan perusteella

Seniori365 -sivustolla on käytössä Views-moduuli, joka on Drupalin kaikista suosituin moduuli (Drupal, 2016). Drupal ytimessä ei ole helppoa tapaa monimutkaisten tietokantakyselyiden toteuttamiseen, eikä hallintapaneelissa ole minkäänlaista graafista käyttöliittymää kyselyiden luomista varten. Views on kyseistä käyttötapausta varten luotu moduuli. Views mahdollistaa monimutkaisten kyselyiden luomisen graafista käyttöliittymää hyödyntäen. Kyselyiden muokkaaminen ja kyselyiden tulosten esittäminen on täysin konfiguroitavissa, joko käyttöliittymää hyödyntäen tai viimekädessä Drupalin rajapintafunktioita hyödyntäen.

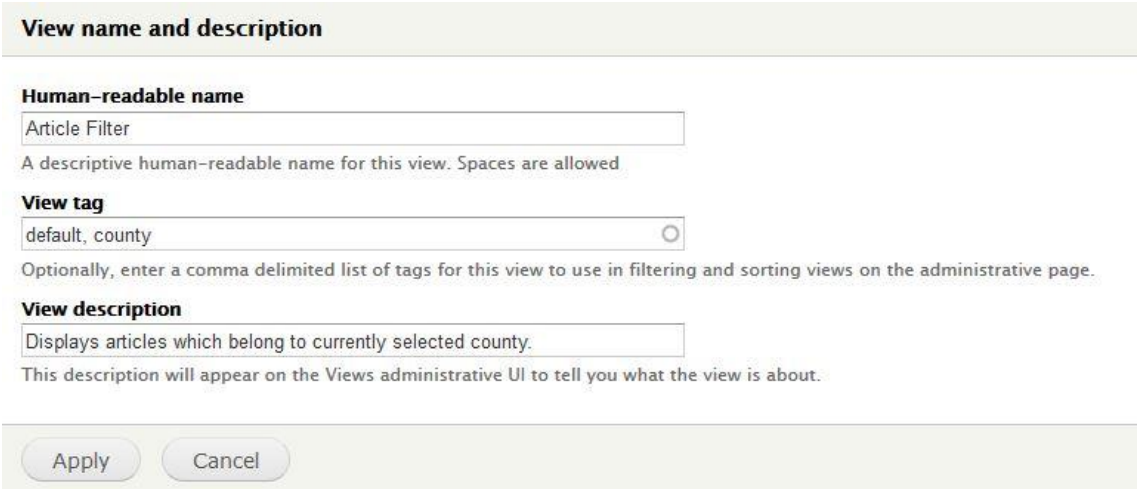
Kuntavalinnan suodattaminen toteutetaan Views-moduulia hyödyntäen. Seniori365-palvelussa on hyödynnetty runsaasti Views-moduulia eri toiminnallisuuksien toteuttamiseen. Esimerkiksi koko Linkit-sivu on yksi Views-moduulilla tehty näkymä, johon haetaan kaikki Linkki-nimisen sisältötyypin sisällöt. Sisältöjen suodattaminen kuntavalinnan mukaan tapahtuu lisäämällä olemassa olevaan Views-näkymään uusi suodatuskriteeri kuntavalintakentälle. Suodatuskriteerin voi lisätä Views-näkymän editointitilassa. Views-näkymän editointitilaan pääsee Structure -> Views -> edit valikosta.

Suodatuskriteeri lisätään editointinäkymän Filter criteria -> Add kohdasta. Oikea suodatusvaihtoehto on kategorian mukaan suodattaminen eli *Content: Has taxonomy term*. Kun suodatuskriteeri lisätään Views-näkymään, se on konfiguroitava toimimaan halutusti. Lisätyn suodatuskriteerin asetuksista on sallittava suodattimen muokattavuus loppukäyttäjälle, muuten Views-näkymä ei salli suodatuskriteerin muuttamista dynaamisesti.
Kun suodattimen muokkaaminen on sallittu suodatuskriteerin asetuksista, käytettävä suodatus asetetaan ”tid” - nimisellä GET-parametrilla. Mikäli kyseistä parametria ei ole linkissä määritelty, sisältöä ei suodateta kuntakohtaisesti.



Kuva 7: Kuntavalinnan suodatuskriteerin konfigurointi

Kun näkymä on konfiguroitu ja asetukset tallennettu, sitä voi testata editointinäkymän alareunassa olevalla esikatselunäkymällä. Koska Views-moduulin oletusulkoasu suodattimille on yksinkertainen eikä se sellaisenaan sovellu hyvin Seniori365-sivustolle, on kuntavalinnan käyttöliittymä tehtävä itse. Ennen käyttöliittymän toteuttamista on kuitenkin muutettava yksi Views-näkymän asetus. Views-editointinäkymän oikeassa ylälaidassa on edit view name/description-nappi, josta pääsee muokkaaman näkymän nimeä, tageja ja kuvausta.



Kuva 8: Tagin lisääminen Views-näkymään

On erittäin tärkeää että kaikkiin Views-näkymiin, jotka käyttävät samaa kuntavalinnan käyttöliittymää lisätään yhteinen uniikki tagi, esimerkiksi county. Tämä tagi on tärkeä, sillä se lisää Views näkymiin tuen saman ulkoasupohjan hyödyntämiseen useassa eri näkymässä. Oletuksena Views -pohjat noudattavat kaavaa views-{view/views-laajennus}--{views-näkymän nimi}.tpl.php, joten niitä ei ole mahdollista käyttää useassa eri Views-näkymässä, sillä Views-näkymän nimi on aina uniikki.

Kun Views-näkymään lisää tagin, on mahdollista käyttää kustomoitua pohjaa, jossa Views-näkymän uniikki nimi on korvattu tagilla. Suodatuskriteerien toiminnallisuuden käyttöliittymäosuus hyödyntää Views-moduulin sisäänrakennettua exposed-form nimistä laajennusta, joten county-tagilla olevan Views-näkymän ulkoasupohjan nimeksi tulee siten views-exposed-form--county.tpl.php.

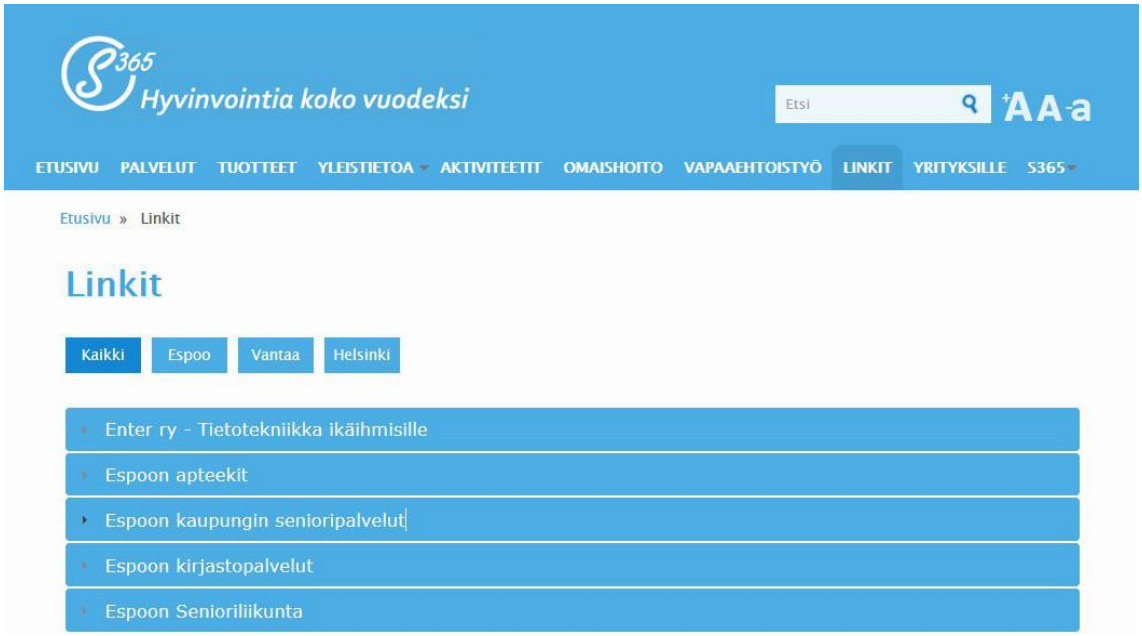
Taulukko 1: Käyttäjän muokattavissa olevien suodatuskriteerien lomakepohjan nimeäminen

Kuvaus	Pohjan tiedostonimi
Ilman tagia	views-exposed-form--{views name}.tpl.php
Tagilla	views-exposed-form--{tag}.tpl.php
County -tagilla	views-exposed-form--county.tpl.php

6.4 Kuntavalinnan käyttöliittymä

Valikkoratkaisu jonka avulla loppukäyttäjä määrittää minkä kunnan sisältöä haluaa katsella, on tehtävä intuitiiviseksi käyttää. Valikkoratkaisun toteutuksessa on huomioitava sivuston tämän hetkinen tyyli ja toteutus, sillä yhtenäinen käyttökokemus on yksi tärkeimmistä verkkosivuston käytettävyyteen vaikuttavista tekijöistä (Nielsen, J. 1999). Sivustolla on käytetty

linkki- ja korostusvärinä sinistä väriä HTML värikoodi #4cade4:ää. Samainen väri toimii myös sivuston yleisenä tunnusvärinä. Onkin luontevaa että toteutettava valikkoratkaisu käyttää kyseistä väriä linkkien indikoimiseen.



Kuva 9: Kuvakaappaus valikkoratkaisusta Linkit-sivulla

Yllä oleva ruutukaappaus ulkoasusta noudattaa värimaailmansa osalta sivuston yleistä ilmettä. Elementit ovat isoja ja ne eivät ole kiinni toisissaan, jotta virhepainallusten määrä pysyisi mahdollisimman pienenä. Tämä on erityisen tärkeää ottaen huomioon sivuston kohderyhmän. Aktiivinen linkki on merkattu selkeästi tummemmalla sinisellä kuin oletuslinkkiväri. Sininen on myös sävyltään tummempi kuin päävalikon aktiivisuutta indikoiva sininen, sillä päävalikon väri ei erotu tarpeeksi selkeästi, jos elementit ovat irrallaan toisistaan.

Linkkielementit ovat myös muodoltaan napin näköisiä, jotta käyttäjän on mahdollisimman helppoa huomata, että elementit ovat klikattavia. Visuaalisia indikaatioita klikattavuudesta voi myös halutessaan parantaa pyöristämällä nappien kulmia hiukan sekä tekemällä aktiivisen linkin tai kaikkien linkkien tekstistä alleviivattua (Loranger, H. 2015).

6.5 Kuntavalinnan implementointi nykyisiin alaosioihin

Kuntakohtaista sisältöä olisi tulossa melkein kaikille sivuston osa-aluille. Useimmat osa-alueet on mahdollista muuttaa helposti tukemaan kuntakohtaista sisältöä, mutta osa sivuston osa-alueista vaatii nykyisen toiminnallisuuden refaktoroimista eli uudelleen kirjoittamista. Tässä osiossa esitellään Seniori365.fi-palveluun tarvittavat muutokset alaosiokohtaisesti.

6.5.1 Linkit –osio

Linkit-osio on yksi selkeimmistä osa-alueista, missä kuntakohtaisen sisällön käyttämisellä voidaan parantaa loppukäyttäjän käyttökokemusta rajaamalla näytettävää sisältöä kuntakohtaistesti. Linkit-osiossa on nykyään runsaasti sekä yleistietoon liittyviä linkkejä sekä vain Espoon alueeseen liittyviä linkkejä. Onkin järkevää että käyttäjälle annetaan mahdollisuus rajata näytettävien linkkien määrää vain häntä kiinnostaviin kuntiin.

Linkit-osioon tarvitaan käyttöliittymä, jonka avulla käyttäjän on mahdollista suodattaa linkkejä kuntakohtaisesti. Käyttöliittymän lisäksi on myös toteutettava palvelinpuolelle toiminnallisuus, joka mahdollistaa sisällön suodattamisen. Käyttöliittymän sekä palvelinpuolen toteutus löytyy seuraavasta osiosta.

Edeltävissä kappaleissa esiteltiin tapa toteuttaa sisällön kategorisointi sekä tiedon haku kuntakohtaisesti. Lisäksi edeltävässä kappaleessa esiteltiin kuntakohtaisen sisällön suodattamiseen mahdollisesti käytettävä käyttöliittymä. Kyseiset ominaisuudet ovat suoraan implementoitavissa Linkit-osioon.

6.5.2 Yleistietoa –osio

Yleistietoa–osioon on tehtävä samat muutokset kuin Linkit–osioon. Käytännössä Yleistietoa-osion kuntakohtaisuuden toteutuksessa voidaan käyttää samaa palvelinpuolen toteutusta kuin Linkit-osion toteutuksessa. Myös samaa käyttöliittymää voidaan käyttää Yleistietoa-osiossa, mutta Yleistietoa-osioon kannattaa harkita myös toisenlaista suodatusratkaisua. Tällä hetkellä Yleistietoa-alaosiossa on käytössä sivupalkki, jossa näkyy kaikki kategoriat sekä sisällön lukumäärä per kategoria. Kyseinen sivupalkki on alanavigaatio, jota käytetään artikkeleiden suodattamiseen artikkelikategorioiden mukaan. Koska sivupalkkia käytetään artikkelisivulla suodattamaan sisältö, olisi loogista että myös muut sisällön kategorioimiseen käytetyt vaihtoehdot olisivat sivupalkissa. Kuntakohtainen sisällön suodatus voitaisiin lisätä esimerkiksi pudotusvalikkona artikkelikategorioiden alapuolelle.

Kuva 10: Ehdotus Yleistietoa-osion kuntavalinnan käyttöliittymäksi

Yleistietoa-alaosion navigaationa toimivassa sivupalkissa näkyy kategorian nimen perässä kyseiseen kategoriaan liitetyn sisällön määrä. Käyttökokemuksen ja informatiivisuuden parantamiseksi olisikin loogista että kategorian perässä näytettävä lukumäärä päivittyisi valitun kunnan mukaan. Sisällön lukumäärän laskuri hyödyntää tällä hetkellä Taxonomy Menu-moduulia eikä sitä ole mahdollista hyödyntää sellaisenaan, jos näytettävään sisällön lukumäärään vaikuttaa myös toinen kategoria. Tämä tarkoittaa, että sivupalkin valikkototeutuksen sisällön lukumäärän laskentaan käytettävä toiminnallisuus on korvattava kustomoidulla ratkaisulla. Toiminnallisuuden toteuttaminen vaatii kustomoidun moduulin kehittämisen, sillä kaikkia tarvittavia rajapintafunktioita ei pysty käyttämään suoraan kustomoidusta teemasta. Tämän opinnäytetyön liitteestä 2 löytyy lähes valmis lähdekoodi kyseistä toiminnallisuutta varten.

6.5.3 Palvelut ja tuotteet

Palvelut ja tuotteet osioihin ei ole tarpeellista tehdä ulkoasumuutoksia, sillä kyseisten osioiden sisältöä on jo mahdollista suodattaa kuntakohtaisesti. Kyseisten osioiden kuntatietoa on tosin käyttäjien syöttämää eikä siten välttämättä yhdenmuotoista. Palveluiden ja tuotteiden kuntatieto olisikin hyvä yhtenäistää, jotta sisällön haettavuus olisi mahdollisimman luotettavaa. Olemassa olevien kuntatietojen päivittäminen on helpointa ja nopeinta tehdä kustomoiduilla tietokantakyselyillä.

Tuotteiden ja palveluiden kuntatieto on tallennettuna drup_field_data_field_postitoimipaikka sekä drup_field_revision_field_postitoimipaikka nimisiin tauluun. Molempien taulujen tietueet on päivitettävä siten, että paikkakuntien kirjoitusasu on yhtenäinen. Alla olevat esimerkkitie-

tokantakyselyt olettavat, että yhtenäinen kirjoitusasu tarkoittaa paikkakunnan suomenkielistä nimeä isolla alkukirjaimella.

```
UPDATE drup_field_data_field_postitoimipaikka
SET `field_postitoimipaikka_value` = CONCAT(
     UPPER(SUBSTR(`field_postitoimipaikka_value`,1,1)),
     LOWER(SUBSTR(`field_postitoimipaikka_value`,2))
 );
```

Yllä oleva tietokantakysely muuntaa field_postitoimipaikka_value -sarakkeen merkkijonon pieniksi kirjaimiksi, joka alkaa aina yhdellä isolla kirjaimella.

```
SELECT * FROM drup_field_data_field_postitoimipaikka WHERE
`field_postitoimipaikka_value` NOT IN ('Helsinki', 'Espoo', 'Vantaa',
'Kauniainen');
```

Yllä oleva kysely hakee tietokannasta kaikki tietueet, joiden arvona EI ole Helsinki, Espoo, Vantaa tai Kauniainen. Koska suurin osa olemassa olevista arvoista on jokin edellä mainituista paikkakunnista, saadaan kyselyn tulosta rajattu merkittävästi pienemmäksi. Rajauksen jälkeen on käytävä manuaalisesti kaikki tulokset läpi, jotta voidaan varmistua tulosten yhdenmukaisuudesta ja oikeellisuudesta. On hyvä huomioida, että yllä mainitut muutokset on tehtävä *drup_field_data_field_postitoimipaikka* sekä *drup_field_revision_field_postitoimipaikka* nimisiin tauluihin.

Olemassa olevien tietueiden päivittämisen lisäksi on muutettava rekisteröintilomakkeen paikkakuntatiedon tekstikenttä pudotusvalikkoon, josta löytyvät kaikki suomen kunnat. Näin käyttäjien rekisteröityessä voidaan olla varmoja, että kuntatietojen kirjoitusasut ovat yhteneväisiä, sillä käyttäjä on tehnyt valinnan ennaltamääritetyistä vaihtoehdoista. Paikkakunnan kirjoitusasu on tietysti myös validoitava ennen rekisteröintilomakkeen tallentamista.

6.5.4 Aktiviteetit

Aktiviteetit-osioon ei ole tarpeellista luoda toiminnallisuutta, joka mahdollistaisi sisällön suodattamisen kuntakohtaisesti. Aktiviteetit-osion sisältö on jo tälläkin hetkellä yleistä, eikä se koske vain tiettyä kuntaa.

6.5.5 Vapaaehtoistyö ja omaishoito

Tällä hetkellä vapaaehtoistyö- ja omaishoito-osion ulkoasut ja toiminnallisuudet ovat hyvin samankaltaisia. Molempien osioiden sisältö koostuu hyvin yleisluonteisista sivuista, mutta molemmista osioista löytyy myös kuntakohtaista sisältöä. Molempien osioiden kuntakohtaisen sisällön luominen kannattaa ainakin alkuvaiheessa tehdä lisäämällä uusia alisivuja osioon. Mikäli kuntakohtaisen sisällön määrä kasvaa suureksi, on syytä harkita suodatusmenetelmän käyttöönottoa. Vapaaehtoistyö ja Omaishoito-osiohin voi soveltaa Linkit-osioon suunniteltua suodatusratkaisua.

Vapaaehtoistyö ja Omaishoito-osioissa voisi myös harkita ankkurilinkkien käyttöä palvelinpuolella tehtävän sisällön suodattamisen sijaan. Esimerkiksi Vapaaehtoistyö -osiosta löytyvä Vapaaehtoisverkosto-alisivu sisältää runsaasti tietoa eri verkostoista, mutta ne eivät ole suodatettavissa palvelinpuolella eikä niiden muuttaminen omiksi sisältötyypeikseen ole järkevä ratkaisu. Kyseiset verkostot voisi ryhmitellä kunnittain ja jokaisen ryhmän html attribuutteihin voisi lisätä uniikin tunnisteen, jolloin sitä pystyy käyttämään ankkurilinkkinä. Ankkurilinkkien ulkoasuksi sopisi Linkit–osion valikkoratkaisu, josta on poistettu ”Kaikki” niminen linkki.

7 Interaktiivinen videokeskustelutoiminnallisuus

Asiakkaan toiveena on lisätä Seniori365.fi-palveluun interaktiivinen videokeskustelutoiminnallisuus. Interaktiivisen videokeskustelutoiminnallisuuden tarkoituksena olisi helpottaa kotona asuvia senioreita kommunikoimaan muiden ikäihmisten kanssa. Kyseisen ominaisuuden olisi siis oltava hieman Skypen kaltainen, helppokäyttöinen usean henkilön keskustelutoteutus videokameratuella. Toiminnallisuuden tulisi olla käytettävissä selaimesta, eikä sen pitäisi vaatia videokameran ja mikin koneeseen liittämisen lisäksi muuta konfigurointia tai käyttäjätilin luomista. Videokeskustelutoteutuksen käyttöliittymä on oltava selkeä ja suomenkielinen. Käyttöliittymän kustomointimahdollisuus on tärkeää, sillä videototeutus olisi selkeintä upottaa iframena verkkosivuston alasivulle.

Reaaliaikaisen, usean käyttäjän videokeskustelutoiminnallisuuden toteuttaminen alusta alkaen vaatii paljon aikaa. Videokeskustelutoiminnallisuuden toteuttaminen edellyttää myös laaja-alaista osaamista verkko-protokollien toiminnasta, palvelimien konfiguroimisesta sekä itse videotoiminnallisuuden kehittämisestä. Videoiden reaaliaikainen lähettäminen ja vastaanottaminen vaativat myös usein runsaasti palvelinresursseja, sillä hyvälaatuiset videot ovat usein suuria mikä edellyttää palvelimelta prosessointikapasiteettia sekä paljon verkkokaistaa. Lisäksi laajan selainyhteensopivuuden saavuttaminen on yhtä teknologiaa hyödyttämällä mah-

dotonta, sillä reaaliaikaiseen videokommunikointiin ei ole vielä olemassa valmiita standardeja vaan eri selainversiot tukevat eri teknologioita.

7.1 Mediasisällön välittämiseen käytettävät teknologiavaihtoehdot

Seuraavissa alaosioissa esitellään kolme yleisintä teknologiaa, joita käytetään reaaliaikaisen median välittämiseen verkkosivujen käyttäjille.

7.1.1 Flash

Reaaliaikaisen median siirtämiseen on perinteisesti käytetty Flash-teknologiaa, mutta lukuisten tietoturvaongelmien, epävakauden, suorituskykyongelmien sekä mobiililaitteiden tuen puuttumisen (Winokur, D. 2011) vuoksi Flashin käyttö on vähentynyt merkittävästi (W3Techs, 2016b). Flash–lisäosan kehittäminen alkoi vuonna 1997 yhdysvaltalaisen Macromedian toimesta (Wikipedia, 2016). Vuonna 2005 Adobe osti Macromedian ja jatkoi Flash-lisäosan jatkokehittämistä (Adobe. 2005).

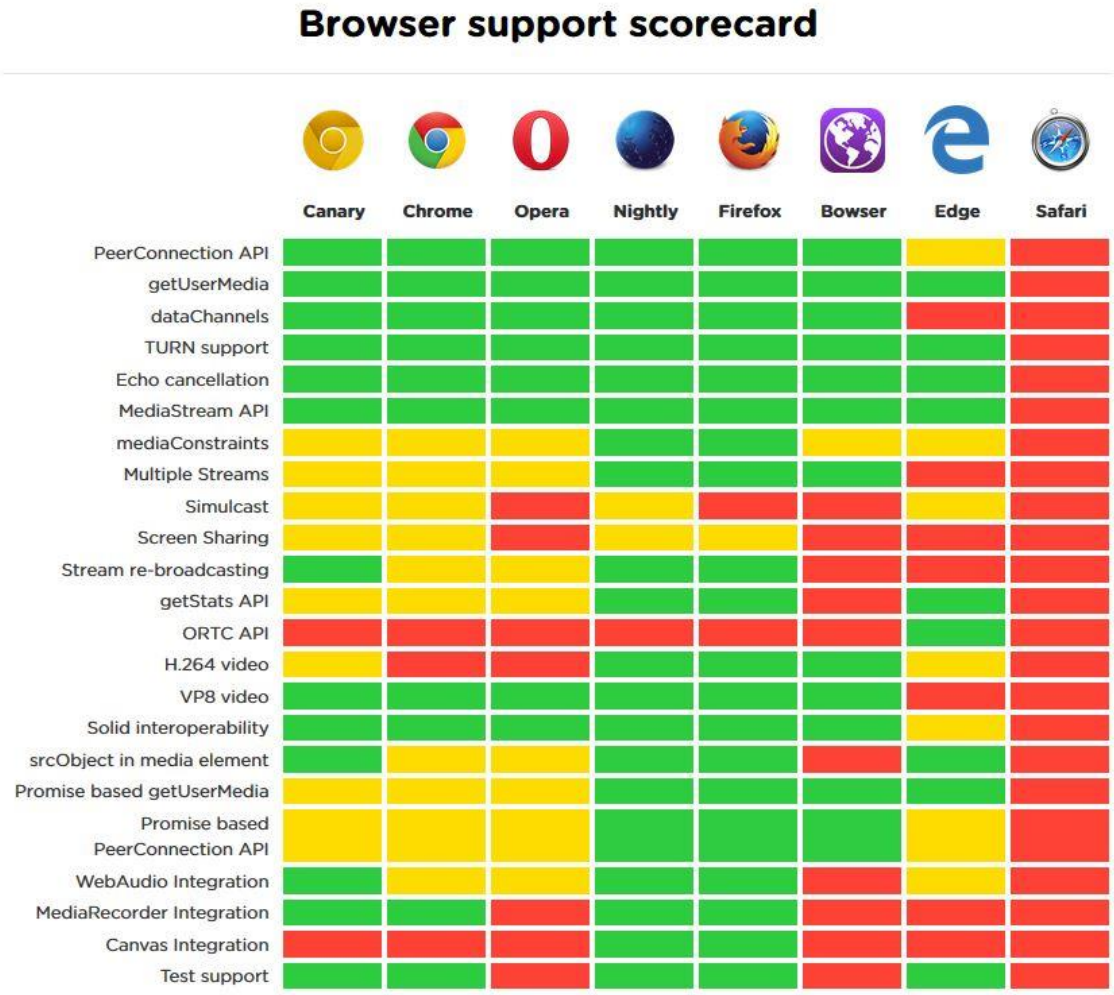
Viime vuosikymmenellä Flash oli ainut teknologia, joka mahdollisti reaaliaikaisen ja monimutkaisen median käyttämisen verkkosivuilla. Tämän seurauksena suuri osa olemassa olevista videopalveluista hyödyntää Flash-teknologiaa, kuten avoimen lähdekoodin Red5Chat (Red5Chat, 2013). HTML5 teknologian ja mobiililaitteiden yleistymisen seurauksena Flash-teknologian käyttö on pudonnut merkittävästi. Flash-teknologian päälle ei siten kannata rakentaa uutta videokeskustelupalvelua. Flash-lisäosa on nykyään myös oletuksen estetty Mozilla Firefox ja Google Chrome selaimissa ja sen tuki loppunee lopullisesti lähitulevaisuudessa (Smedberg, B. 2015).

7.1.2 Silverlight

Silverlight on Microsoftin valmistama selainlisäosa, joka on saatavilla Firefox, Safari ja Internet Explorer selaimille. Silverlight-lisäosa on saatavilla MacOS ja Windows–käyttöjärjestelmille. (Microsoft. 2016). Silverlight tarjoaa käytännössä samat ominaisuudet kuin Flash, minkä vuoksi sen käyttö on erittäin vähäistä (W3Techs, 2016b) ja se on tekniikkana käytännössä kuollut. Esimerkiksi Microsoftin valmistama moderni Edge-selain ei tue Silverlight-lisäosaa. Lisäksi Silverlight tuki poistuu Firefox-selaimista vuoden 2016 lopussa (Smedberg. 2015).

7.1.3 WebRTC

WebRTC (Web Real-Time Communications) standardin tarkoitus on luoda yhtenäinen ja standardoitu protokolla median, kuten videoiden, tiedostojen ja audion, jakamiseen selaimelta selaimelle tai selaimelta palvelimelle. (W3C, 2016). Toisin sanoen WebRTC-standardin tarkoituksena on määritellä selaimissa käytettävä median jakamiseen tarkoitettu vertaisverkkoprotokolla. Useimmat modernit selaimet tukevat vielä keskeneräistä WebRTC-standardia.



Kuva 11: WebRTC-standardin selaintuki 17.4.2016 (IsWebRTCreadyyet, 2016)

WebRTC on uusi teknologia ja sen selainyhteensopivuus on vielä keskeneräinen. WebRTC tukea ei myöskään löydy kaikista selaimista, kuten Microsoftin Internet Explorer tai Applen Safari selaimista. WebRTC tuki on kuitenkin mahdollista lisätä myös Internet Explorer ja MacOS Safari selaimiin Temasys-yrityksen valmistamalla ilmaisella selainlisäosalla (Currier, N. 2016).

7.1.4 Teknologiavaihtoehtojen selaintuki

Taulukko 2: Videokeskustelupalvelun teknologiavaihtoehtojen selaintuki

	Flash	WebRTC	Silverlight
Chrome	Kyllä	Kyllä	Ei
Firefox	Kyllä	Kyllä	Kyllä[4]
Safari	Kyllä	Kyllä[1]	Kyllä
Edge	Kyllä	Kyllä[2]	Ei
Internet Explorer	Kyllä	Kyllä[1]	Kyllä
Opera	Kyllä	Kyllä	Ei
Android (natiiviselain)	Ei	Kyllä[3]	Ei
Android (chrome)	Ei	Kyllä	Ei
iOS (safari)	Ei	Ei	Ei
iOS (chrome)	Ei	Ei	Ei

[1] Vaatii kolmannen osapuolen valmistaman selainlisäosan asentamisen.

[2] Ei tue vertaisverkkopohjaista median jakamista.

[3] Käyttöjärjestelmäversio vähintään 5.0. Selainydinversio vähintään Chromium M37.

[4] Tuki lakkaa vuoden 2016 lopussa. Tuettuna Windows- ja MacOS-käyttöjärjestelmissä.

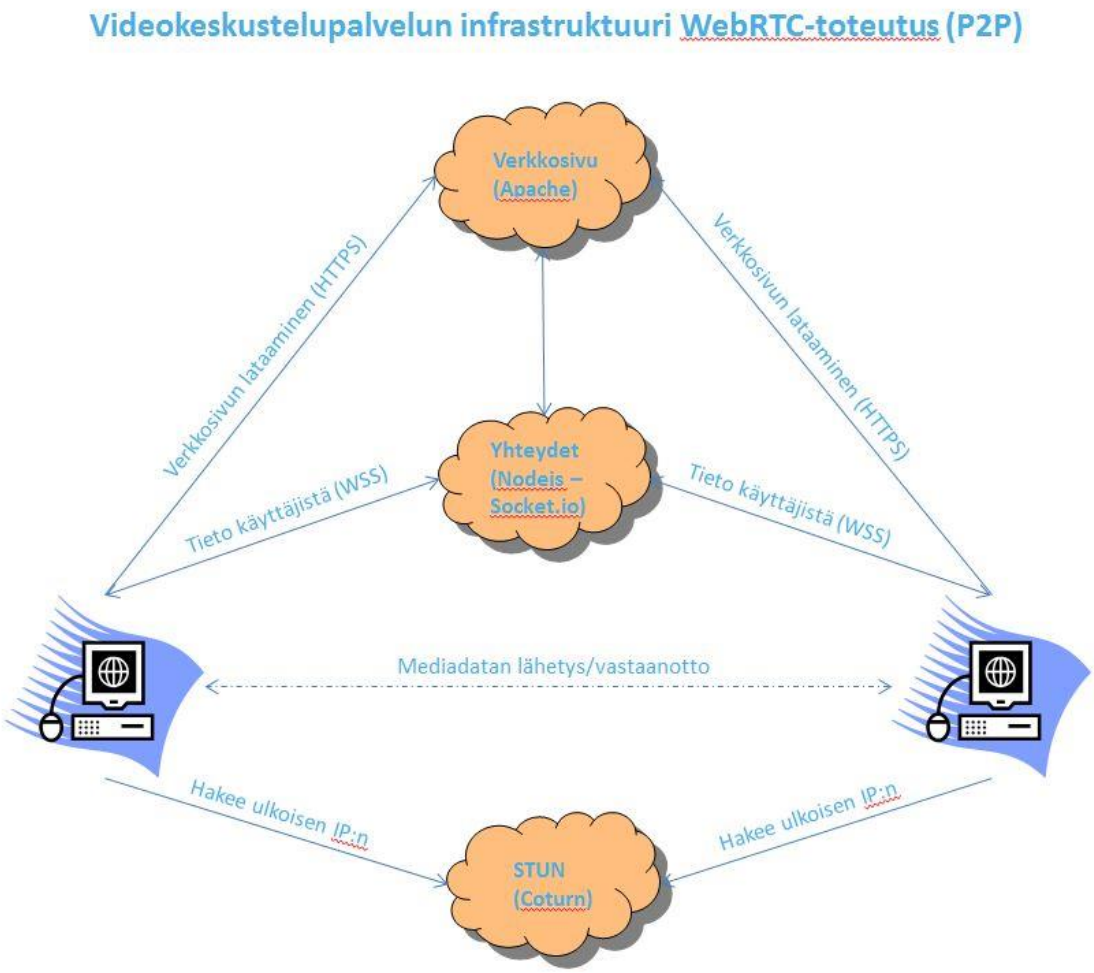
7.2 Videokeskustelupalvelun tekninen rakenne yleisesti

Tässä osiossa käydään läpi Flash- ja WebRTC-pohjaisten videokeskustelupalveluiden tekninen toiminta yleisellä tasolla, sillä Flash ja WebRTC ovat ainoat järkevät teknologiavaihtoehdot selainpohjaisen videokeskustelupalvelun toteuttamiseen.

7.2.1 WebRTC

WebRTC-standardi mahdollistaa käyttäjän videokameran ja mikrofonin hallinnoinnin selaimessa Javascript-rajapintoja hyödyntäen. WebRTC-pohjaisen palvelun käyttöliittymäosuuden toteuttaminen ei eroa normaalista verkkosivun toteuttamisesta, sillä se hyödyntää normaaleja HTML-, CSS- ja Javascript-tekniikoita. WebRTC on siten hyvä valinta modernin videokeskustelupalvelun toteuttamiseen. WebRTC-rajapintojen etuna on riippumattomuus kolmannen osapuolen valmistamista selainlaajennoksista. WebRTC-standardi mahdollistaa mediadatan, kuten videoiden, jakamisen suoraan käyttäjien kesken. Tämä pienentää palvelimiin kohdistuvaa kuormaa merkittävästi, sillä vertaisverkossa jaettua videodataa ei tarvitse lähettää ollenkaan palvelimelle.

Vertaisverkkoperiaatteella toteutettu videokeskustelupalvelu kärsii kuitenkin kapasiteettirajoitteista, joihin ei pysty juuri vaikuttamaan. Vertaisverkkona toimiva videopalvelu vaatii käyttäjiltä paljon enemmän lähetyskaistaa, sillä jokaisen käyttäjän on jaettava oma videonsa kaikille videokeskusteluun osallistuville käyttäjille erikseen. Kuluttajaliittymien lähetysnopeus on usein reilusti pienempi kuin latausnopeus ja yhden megabitin lähetysnopeus on useissa liittymissä yleinen, mikä asettaa merkittävän rajoitteen yhtäaikaisten käyttäjien määrälle. Tämän vuoksi vertaisverkkoperiaatteella toimivat videopalvelut rajaavat maksimiosallistujamäärän usein 8-12 käyttäjään.



Kuva 12: Vertaisverkkoperiaatteella toimiva WebRTC-pohjainen videokeskustelupalvelu

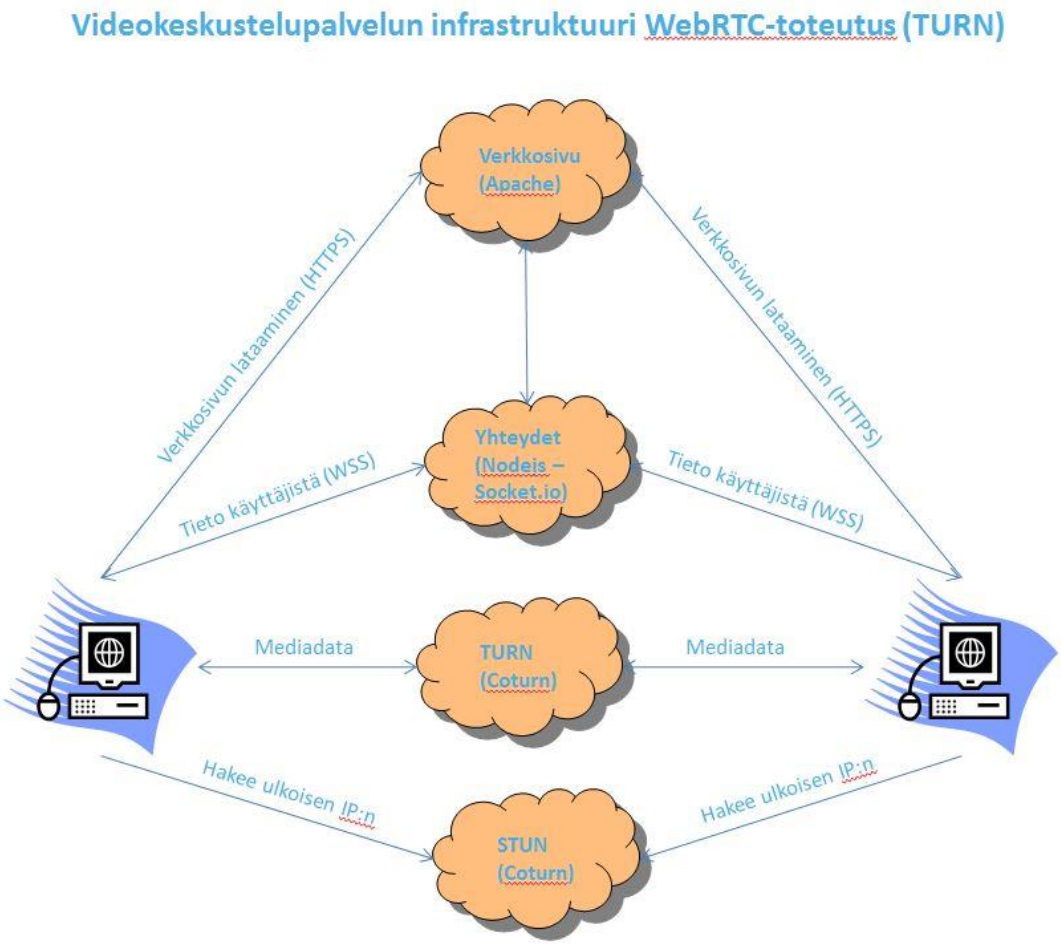
Vertaisverkkopohjainen WebRTC-toteutus on yksinkertaisin tapa toteuttaa videokeskustelupalvelu, sillä kyseissä tavassa ei tarvitse juuri huolehtia palvelinratkaisun tehokkuudesta tai kapasiteetin riittävyydestä koska videoiden lähetys tapahtuu käyttäjiltä käyttäjille. Videopalvelun ydintoiminnallisuudeksi jääkin käyttäjien yhdistäminen toisiinsa sekä mahdollisen tekstipohjaisen viestintäominaisuuden toteuttaminen.

Vertaisverkkopohjainen WebRTC-videokeskustelupalvelutoteutus koostuu kolmesta eri palvelimesta: verkkopalvelimesta, signalointipalvelimesta sekä STUN-palvelimesta. Vaikka palvelussa tarvittavien eri osasien määrä vaikuttaa ensi alkuun suurelta, on palvelun toteuttaminen kokonaisuudessaan hyvinkin suoraviivaista. Verkkopalvelinta, esimerkiksi Apachea, käytetään varsinaisen videopalvelun lähdekoodin jakamiseen käyttäjille.

WebRTC:ssä signaloinnilla tarkoitetaan käyttäjien yhdistämistä toisiinsa sekä viestien välittämistä käyttäjältä toiselle. WebRTC-standardi ei määrittele signalointiin käytettävää protokollaa tai tekniikkaa, joten se on jokaisen palvelun itsensä päätettävissä. Signalointipalvelin on järkevintä toteuttaa, joko HTTP- tai Websocket-pohjaisena toteutuksena. HTTP-protokollaa hyödyntävä signalointitoteutus ei vaadi erillistä signalointipalvelinta. Signaaliviestit ovat kuitenkin luonteeltaan kaksisuuntaisia eikä HTTP-protokolla tue kaksisuuntaista viestintää, vaan asiakkaan on kysyttävä palvelimelta toistuvasti tietoa mahdollisista uusista viesteistä (eng. polling). Websocket-protokolla on HTTP-protokollaa parempi tapa kaksisuuntaisten viestien välittämiseen, sillä Websocket-protokolla on suunniteltu kaksisuuntaiseen pitkäkestoiseen viestintään. Websocket-protokolla muodostaa pitkäkestoisen TCP yhteyden palvelimen ja asiakkaan välille. (IETF. 2011).

STUN–protokollaa tukevaa palvelinta käytetään selvittämään NAT:in takana sijaitsevan laitteen ulkoinen IP-osoite ja portti. (IETF. 2008). Pienimuotoiseen ja ei-kaupalliseen käyttöön löytyy myös ilmaisia ja avoimia STUN-palvelimia eri toimijoiden, kuten Googlen ylläpitäminä (zziuni. 2012). Vaihtoehtoisesti on mahdollista asentaa ja konfiguroida ilmainen avoimen lähdekoodin palvelintoteutus, kuten Coturn.

WebRTC-protokolla mahdollistaa myös TURN-protokollan hyödyntämisen mediadatan välittämiseen käyttäjiltä muille käyttäjille varsinaisen vertaisverkkotoiminnallisuuden sijaan. Vertaisverkkotoiminnallisuutta ei ole aina mahdollista käyttää, sillä palomuuri tai välityspalvelin voi estää vertaisverkon toiminnan. Tällöin TURN-protokollaa voidaan käyttää datan välittämiseen käyttäjien välillä. TURN-palvelimella on julkinen osoite, johon käyttäjät voivat ottaa yhteyttä, vaikka he olisivatkin palomuurin tai välityspalvelimen takana. TURN-palvelimet kuluttavat runsaasti kaistaa ja prosessointitehoa, sillä kaikkien osallistujien kuvaama videodata on uudelleenohjattava palvelimen kautta. (Dutton, S. 2013)



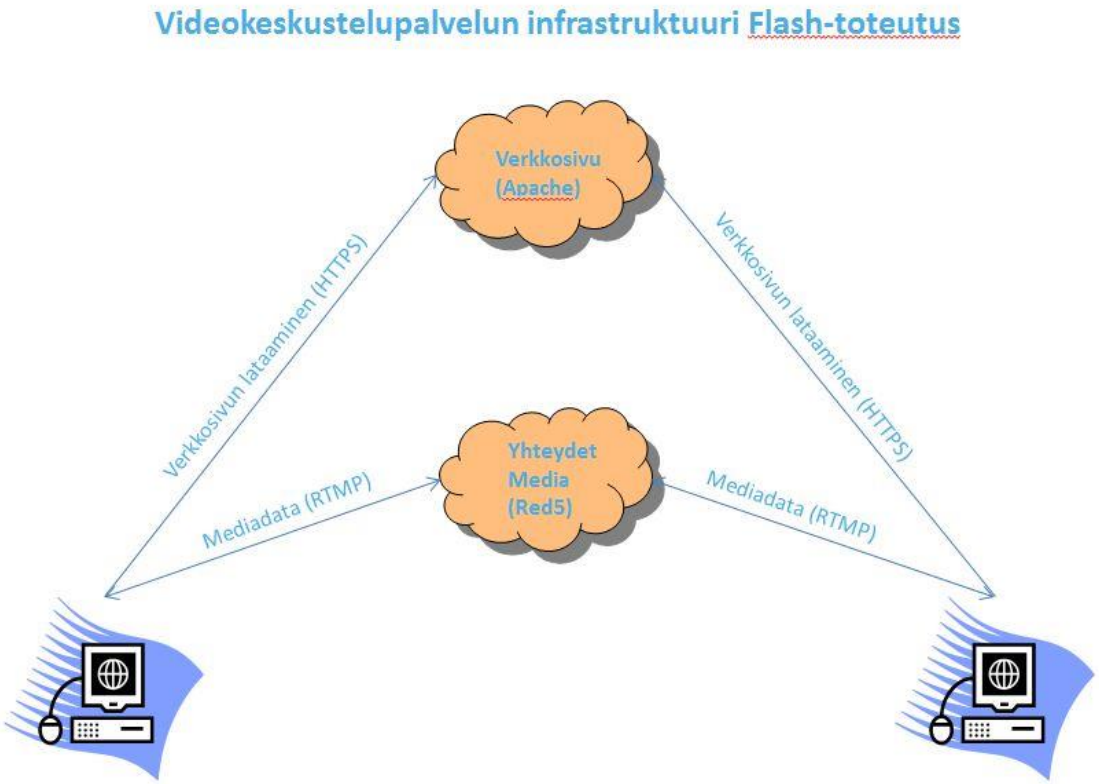
Kuva 13: TURN-palvelinta hyödyntävä WebRTC-pohjainen videokeskustelupalvelu

WebRTC-standardiin pohjautuvan videokeskustelupalvelun on myös pakko käyttää salattuja yhteyksiä, kuten HTTPS- ja WSS-protokollia suojaamattomien sijaan. Tämä johtuu Googlen tietoturvapolitiikasta, sillä Google on estänyt WebRTC-rajapinnan käyttämisen salaamattomilla yhteyksillä suositussa Chrome-selaimessa. (Dutton, S. 2015)

7.2.2 Flash

Flash-sovellukset toteutetaan ActionScript-ohjelmointikielellä, joka pohjautuu ECMAScript-standardiin. Vaikka ActionScript on implementaatio ECMAScript-standardista, se ei ole skriptikieli, vaan käännettävä oliopohjainen ohjelmointikieli. Flash-sovellusten kehitys on siten hitaampaa kuin HTML ja Javascript-pohjaisen ohjelmiston toteuttaminen. Flash-käyttöliittymän toteuttaminen on myös hitaampaa ja vähemmän intuitiivisempaa kuin HTML-pohjaisen käyttöliittymän toteuttaminen.

Flash-pohjaisen videokeskustelupalvelun tekninen rakenne on hiukan yksinkertaisempi kuin WebRTC-teknologiaan pohjautuva palvelu, sillä Flash-pohjainen toteutus vaatii vain yhden Flashia tukevan palvelimen. Flash suunniteltiin alun perin interaktiivisen median jakamiseen internetissä, minkä vuoksi videoiden lähettämiseen käyttäjältä toiselle löytyy valmiita yhden toteutuksen ratkaisuja, kuten avoimen lähdekoodin Red5 palvelin. Flash-pohjaisten videokeskutelupalveluiden suurin haaste onkin käyttöliittymän kustomointi ja muokkaaminen sekä selainyhteensopivuus. Flash-teknologia ei ole tuettuna mobiiliselaimissa sekä se on oletuksena poissa käytössä osassa työpöytäselaimia.



Kuva 14: Flash-pohjaisen videokeskustelupalvelun rakenne

7.3 Valmiit videokeskustelutoteutukset

Upotettavia sekä muita valmiita videokeskustelutoteutuksia löytyy useita erilaisia. Useimmat selainkäyttöön tarkoitetut videokeskustelutoteutukset on tehty WebRTC- tai Flash-tekniikalla. Upotettavat Flash-pohjaiset toteutukset ovat maksullisia, sillä Flashillä tehdyt toteutukset vaativat aina dedikoidun mediapalvelmien videoiden välittämiseen käyttäjiltä toisille. Useat Flash-pohjaiset valmistoteutukset ovat myös käyttöliittymältään vanhentuneita eikä niiden kustomointi ole yhtä helppoa kuin WebRTC-pohjaisen toteutuksen käyttöliittymän kustomointi. Flash-pohjaiset palvelut ovat siten huono vaihtoehto videokeskustelutoteutukselle.

Uudet videokeskustelupalvelut pohjautuvat käytännössä aina WebRTC-standardin mukaisiin teknologioihin. WebRTC mahdollistaa palvelun toteuttamisen vertaisverkkoteknologialla, minkä vuoksi palvelinkapasiteetin tarve on pieni verrattuna Flash-pohjaisiin toteutuksiin. Monet WebRTC-pohjaiset videokeskustelupalvelut tarjoavatkin ilmaista vaihtoehtoa maksullisten ohessa. Ilmaisissa vaihtoehdoissa on aina rajattu maksimikäyttäjämäärä, eikä ilmaisiin palveluihin sisälly TURN-palvelinta. Palvelusta riippuen kuukausi- tai kertamaksulla saa ostettua TURN-palvelimen tuen, kasvatettua maksimikäyttäjämäärää sekä kustomoitua videopalvelun käyttöliittymää.

7.3.1 Appear.in

Appear.in on WebRTC-rajapintaan pohjautuva videokeskustelutoteutus. Appear.in toteutuksen käyttöliittymä on selkeä ja helppokäyttöinen. Appear.in toimii vertaisverkkotekniikalla, eikä siinä ole mahdollisuutta hyödyntää TURN-palvelimia. Appear.in tukee enintään kahdeksaa yhtäaikaista käyttäjää ja palvelun voi upottaa iframe:na mille tahansa SSL-sertifioidulle sivustolle. Appear.in palvelun käyttö on ilmaista. (Appear.in. 2016)

Appear.in on kiinnostava ja moderni palvelu, joka hyödyntää uusia WebRTC-rajapintoja. Palvelu on erittäin yksinkertainen niin ulkoasullisesti, kuin ominaisuuksiltaan. Appear.in on kuitenkin liiankin yksinkertainen. Sen suurin heikkous onkin lokalisoinnin ja kustomisointimahdollisuuksien puuttuminen. Palvelun kielenä on englanti, eikä sitä ole mahdollista kääntää suomeksi. Palvelun käyttöliittymää ei voi myöskään kustomisoida, mikä tekee palvelun käyttöliittymän sovittamisen Seniori365.fi –sivustolle haasteelliseksi.

7.3.2 Tokbox

Tokbox on WebRTC-rajapintoihin pohjautuva palvelu, joka tarjoaa yhtenäistetyn rajapintaimplementaation erilaisille WebRTC yhteensopiville laitteille ja selaimille. Tokboxin ja Appear.in palveluiden suurin ero integraation helppous ja kustomoitavuus. Appear.in on erittäin helppo integroida. Se ei vaadi kuin upotuskoodin liittämisen verkkosivustolle. Tokbox tarjoaa paljon matalamman tason tuotetta, WebRTC-rajapintaimplementaatiota, joka mahdollistaa täysin kustomoidun palvelun kehittämisen. Tokbox ei tarjoa valmista videokeskustelupalvelua vaan työkalut sen kehittämiseen. Tokbox tarjoaa myös käyttövalmiita TURN palvelimia. (Tokbox. 2016)

Tokbox tarjoaa kehitysrajapinnat palvelin- sekä asiakasympäristöille. Tokboxin asiakasympäristörajapinnat tukevat WebRTC yhteensopivia selaimia sekä Android- ja iOS-käyttöjärjestelmiä. Tokbox on kuitenkin maksullinen palvelu, jonka hinta ovat vähintään 50€/kk. Hinta on suuri kun huomioi Seniori365-palvelussa hyödynnettävissä olevien ominai-

suuksien määrän (TURN-palvelin sekä Javascript/PHP kehityskirjastot). Tokbox soveltuukin paremmin sivustoille ja sovelluksille, joiden ydintoiminnallisuutena on tarjota videokeskustelupalvelua.

7.3.3 Apache Openmeetings

Apache OpenMeetings on avoimen lähdekoodin videokeskustelupalvelu. Apache OpenMeetings on rakennettu avoimen lähdekoodin Red5 palvelimen päälle. Red5 on kirjoitettu Javalla ja se käyttää Flashia videoiden nauhoittamiseen ja jakamiseen. Apache OpenMeetings on suosittu avoimen lähdekoodin videokeskustelutoteutus. OpenMeetings tukee monipuolista kustomointia sekä lokalisointia, joten kustomoidun käyttöliittymän luominen on mahdollista. OpenMeetings tukee vain Flashia, joten sitä ei voi käyttää mobiililaitteilla. (Apache OpenMeetings. 2016)

Red5-palvelimen valmistanut Infrared5 -yritys on kuitenkin suunnitellut WebRTC-tuen lisäämistä uuteen Red5 Pro–palveluunsa, joka hyödyntää Red5-palvelinta. (Red5. 2016). Mikäli myös avoimen lähdekoodin Red5-palvelin saa tulevaisuudessa WebRTC-tuen, tulee OpenMeetings-alustasta yksi potentiaalisimmista ilmaisista videokeskustelupalvelun kehitysalustoista. WebRTC- sekä Flash-tuki samassa toteutuksessa mahdollistaisi laajan selaintuen. Ainoastaan iOS käyttöjärjestelmä ja vanhat Android-selaimet eivät tukisi OpenMeetingsiä.

Apache OpenMeetings vaatii myös palvelimelta enemmän suorituskapasiteettia, sillä kaikki videot kulkevat palvelimen kautta. OpenMeetingsin vähimmäisvaatimuksena on yksi giga muistia ja yhden ytimen suoritin. Jos Apache OpenMeetingis:iä on tarkoitus pyörittää samalla palvelimella kuin varsinaista verkkosivua, on palvelimen suoritustehoa todennäköisesti nostettava vähintään vähimmäisvaatimusten verran. Apache OpenMeetings tarvitsee myös paljon kaistaa, mikäli videoiden laatu halutaan pitää korkeana. Apache OpenMeetings tarjoaa laskurin jota voi käyttää tarvittavan kaistan arviointiin. Esimerkiksi puolen tunnin kestoinen 10 yhtäaikaisen käyttäjän videokeskustelusessio kuluttaa kaistaa yli 80 gigabittiä, jos kaikkien osallistujien videoresoluutiona on 480x360 (360p). (Apache OpenMeetings, 2016b)

7.4 Suositeltu implementaatio

Erilaisia videokeskustelutoteutuksia on olemassa runsaasti, mutta niissä kaikissa on omat puutteensa. Parhaiten Seniori365–palveluun soveltuu toteutus, joka on helppo implementoida ja ylläpitää. Suuren painoarvon saa myös palvelun hankintahinta. Ilmainen Apache OpenMeetings on yksi parhaista mahdollisista vaihtoehdoista. Se on laajasti kustomoitavissa ja siihen löytyy runsaasti ohjeita palvelun omilta foorumeilta sekä muilta sovelluskehitykseen erikoistuneilta sivustoilta. OpenMeetingsiin pohjautuvan toteutuksen suurin haaste on palvelinpuolen suorituskykyvaatimukset. OpenMeetings pakottaisi nostamaan palvelimen tehoja, mikä tar-

koittaa palvelimen kuukausimaksun nousua. OpenMeetings myös pohjautuu Flashiin, mikä on huonoa videokeskustelupalvelun tulevaisuutta ajatellen. Flash-teknologia tulee poistumaan selaimista tulevaisuudessa kun WebRTC-standardin mukaiset rajapintaimplementaatiot yleistyvät.

Seniori365-palveluun soveltuu parhaiten ratkaisu, joka ei vaadi merkittävästi lisää palvelinresursseja. Tämän vuoksi suositeltava tapa on kehittää oma, WebRTC vertaisverkkoteknologiaan pohjautuva videokeskustelupalvelu. Vertaisverkkototeutusta hyödyntämällä voidaan ulkoistaa palvelimiin normaalisti kohdistuvaa kapasiteettitarvetta palvelun käyttäjille. Vertaisverkkototeutuksen suurimmat puutteet ovat vielä osittain puutteellinen selaintuki sekä maksimikäyttäjien määrän pienuus. Vertaisverkkopohjaisen toteutuksen hyvät puolet kuitenkin peittoavat huonot puolet: täydellinen kustomoitavuus, helppo toteutettavuus sekä toteutuksen käytön ilmaisuus.

Kustomoitu videokeskustelutoteutus kannattaa kehittää täysin irrallisena osionaan varsinaisesta Seniori365-verkkosivustosta. Videokeskustelupalvelua ei pysty toteuttamaan käytössä olevalla sisällönhallintajärjestelmällä, joten on loogista tehdä siitä täysin irrallinen komponentti, joka liitetään Seniori365-sivustolle esimerkiksi lataamalla videokeskustelutoteutus iframe-elementillä.

8 Yhteenveto

Opinnäytetyön tarkoituksena oli luoda yksityiskohtainen ohjeistus sisällön kuntakohtaiseen kategorisointiin sekä esitellä sopivin tapa videokeskustelutoiminnallisuuden toteuttamiseen. Ensimmäisessä tavoitteessa eli kuntakohtaisen kategorisoinnin toteuttamissuunnitelman luomisessa onnistuttiin. Kuntakategorian luominen on erittäin yksityiskohtaisesti selitetty, jotta Seniori365.fi-palvelun jatkokehittäjät pystyvät implementoimaan kategorisointitoiminnallisuuden mahdollisimman helposti, vaikka heillä ei itsellään olisikaan aikaisempaa kokemusta Drupal-sisällönhallintajärjestelmästä.

Opinnäytetyön videokeskustelutoiminnallisuutta käsittelevässä osiossa esiteltiin kattavasti kaikki mahdolliset teknologiat, joita on mahdollista käyttää halutun toiminnallisuuden toteuttamiseen sekä esiteltiin kolme erilaista videokeskustelutoteutusta. Opinnäytetyössä esitellyistä teknologiavaihtoehdoista keskityttiin eniten moderneimpaan teknologiavaihtoehtoon, WebRTC-standardin mukaisiin rajapintoihin. Vanhentuneiden sekä käytöstä poistuvien teknologioiden esittely oli huomattavasti suppeampaa.

Opinnäytetyössä esiteltiin Appear.in, Apache OpenMeetings sekä Tokbox videokeskustelutoteutukset. Kyseiset toteutukset ovat suosittuja sekä keskenään hyvin erilaisia kokonaisuuksia, joiden integrointi Seniori365.fi-palveluun osoittautui vaatimuksista johtuen vaikeaksi. Opinnäytetyössä päädyttiinkin suosittelemaan kustomoidun WebRTC-rajapintoihin pohjautuvan videokeskustelutoiminnallisuuden toteuttamista. Opinnäytetyössä ei toteutettu varsinaista videokeskustelutoiminnallisuuden toteutussuunnitelmaa, joten sen toteuttaminen jää mahdolliseksi jatkokehityshankkeeksi.

Lähteet

Kirjalliset lähteet

Travis, B. 2011. Pro Drupal 7 for Windows Developers. Yhdysvallat: Apress.

Wilkins, J. 2010. Drupal 7 Module Development. Yhdysvallat: Packt Publishing.

Sähköiset lähteet

Adobe. 2005. Adobe to acquire Macromedia. Viitattu 23.4.2016. https://www.adobe.com/aboutadobe/invrelations/adobeandmacromedia.html

Apache OpenMeetings. 25.3.2016. Open-Source Web-Conferencing. Viitattu 8.5.2016. http://openmeetings.apache.org/

Apache OpenMeetings. 25.3.2016b. Network bandwidth for OpenMeetings. Viitattu 8.5.2016. http://openmeetings.apache.org/NetworkCalculator.html

Appear.in. 2016. Appear.in. Viitattu 8.5.2016. https://appear.in/

Currier, N. 20.4.2016. WebRTC Plugins. Viitattu 24.4.2016. http://confluence.temasys.com.sg/display/TWPP

Drupal. Project usage overview. Viitattu 19.3.2016. https://www.drupal.org/project/usage

Dutton, S. 2015. Chrome 47 WebRTC: media recording, secure origins & proxy handling. Viitattu 24.4.2016. https://developers.google.com/web/updates/2015/10/chrome-47-webrtc

Dutton, S. 4.11.2013. WebRTC in the real world: STUN, TURN and signaling. Viitattu 1.5.2016. http://www.html5rocks.com/en/tutorials/webrtc/infrastructure/

IETF (Internet Engineering Task Force). 2011. The WebSocket Protocol. Viitattu 1.5.2016. https://tools.ietf.org/html/rfc6455

IETF (Internet Engineering Task Force). 2008. Session Traversal Utilities for NAT (STUN). Viitattu 1.5.2016. https://tools.ietf.org/html/rfc5389

IsWebRTCreadyyet. 2016. Viitattu 17.4.2016 http://iswebrtcreadyyet.com/

Loranger, H. 8.3.2015. Beyond Blue Links: Making Clickable Elements Recognizable. Viitattu 20.3.2016. https://www.nngroup.com/articles/clickable-elements/

Meyer, K. 12.7.2015. The Characteristics of Minimalism in Web Design. Viitattu 20.3.2016. https://www.nngroup.com/articles/characteristics-minimalism/

Microsoft. 2016. Get Microsoft Silverlight. Viitattu 23.4.2016. https://www.microsoft.com/getsilverlight/Get-Started/Install/Default.aspx

Nielsen, J. 22.8.1999. Do Interface Standards Stifle Design Creativity? Viitattu 20.3.2016. https://www.nngroup.com/articles/do-interface-standards-stifle-design-creativity/

Red5. 4.11.2015. What Infrared5 Has Quietly Been Working On: Red5 Pro - Going Back to Our Roots. Viitattu 8.5.2016. http://red5.org/#!/news

Red5Chat. 2013. Viitattu 17.4.2016. http://www.red5chat.com/

Smedberg, B. 8.10.2015. NPAPI Plugins in Firefox. Viitattu 17.4.2016.
https://blog.mozilla.org/futurereleases/2015/10/08/npapi-plugins-in-firefox/

Techopedia. 2016. Procedural Language .Viitattu 10.4.2016.
https://www.techopedia.com/definition/8982/procedural-language

Tokbox. 2016. Build Communication into your Web and Mobile Apps. Viitattu 8.5.2016.
https://tokbox.com/

W3C. 28.1.2016. WebRTC 1.0: Real-time Communication Between Browsers. Working Draft 28 January 2016. Viitattu 17.4.2016. https://www.w3.org/TR/webrtc/

W3Techs. 2016. Historical yearly trends in the usage of content management systems for websites. Viitattu 19.3.2016.
http://w3techs.com/technologies/history_overview/content_management/all/y

W3Techs. 2016b. Historical yearly trends in the usage of client-side programming languages for websites. Viitattu 23.4.2016.
http://w3techs.com/technologies/history_overview/client_side_language/all/y

Wikipedia. 2016. Macromedia. Viitattu 23.4.2016. https://en.wikipedia.org/wiki/Macromedia

Winokur, D. 9.11.2011. Flash to Focus on PC Browsing and Mobile Apps; Adobe to More Aggressively Contribute to HTML5. Viitattu 23.4.2016.
http://blogs.adobe.com/conversations/2011/11/flash-focus.html

zziuni. 18.9.2012. stuns. Viitattu 1.5.2016. https://gist.github.com/zziuni/3741933

Kuvat

Taulukot

Liitteet

Liite 1 Esimerkkilähdekoodi kustomoituun Views -pohjaan

Oheinen lähdekoodi ei ole selaintestattua, eikä sitä tule käyttää sellaisenaan. Muun muassa tyylit on hyvä siirtää ensin tyylitiedostoon. Pohjatiedoston tulee sijaita teeman juuressa ja se on nimettävä seuraavasti:

- views-exposed-form--county.tpl.php

```
<div class="city-wrapper">
    <input class="hidden-city" type="hidden" name="tid" value="All" />

    <div class="city-filter">
        <?php foreach($form['tid']['#options'] as $key => $value): ?>
            <a href="?tid=<?php echo $key; ?>"
                class="city <?php echo $key == $form['tid']['#value'] ? 'se-
lected' : ''; ?>">
                <?php echo $key == 'All' ? 'Kaikki' : $value; ?>
            </a>
        <?php endforeach; ?>
    </div>

    <style>
        .city-filter{
          clear: both;
          content: "";
          display: table;
          margin-bottom: 10px;
        }

        .city{
          display: inline-block;
          float: left;
          padding: 5px 0px;
          background-color: #4cade4;
          color: #fff;
          margin: 0px 5px 5px 0px;
          min-width: 70px;
          text-align: center;
          cursor: pointer;
          -webkit-transition: all 0.3s ease;
          transition: all 0.3s ease;
        }

        .city:first-child{
            padding-left: 0px;
            margin-left: 0px;
        }

        .city.selected, .city:hover{
          background-color: #1487d4;
          color: #fff;
          text-decoration: none;
        }
    </style>
</div>
```

Liite 2 Kuntakohtaisen sisällön määrän laskentaan käytettävä moduuli

Kustomoitu moduuli, jota käytetään laskemaan Yleistietoa osion kuntakohtaisen sisällön määrä. Moduuli vaatii toimiakseen Taxonomy Menu –moduulin. Moduulin nimenä on ”oppari”, mutta se kannattaa muuttaa paremmin sen toiminnallisuutta kuvaavaksi. Vaihtoehtoisesti moduulin toimintalogiikka voidaan lisätä toiseen, jo olemassa olevaan kustomoituun moduuliin. Tällöin .info –tiedostoa ei tarvita.

```
Tiedosto: /drupal/path/sites/all/modules/oppari/oppari.info

core          = "7.x"
dependencies[] = taxonomy
dependencies[] = menu
dependencies[] = taxonomy_menu
description   = "Custom module which modifies the information vocabu-
lary node count based on the currently selected county."
name          = "County node count"
package       = Seniori365
files[] = oppari.module
```

Tiedosto: /drupal/path/sites/all/modules/oppari/oppari.module

```
<?php


/**
 * Get the node count by tids. Only nodes which have reference to all
of the given tids are included.
 *
 * @param tids - The array of tids which node count should be re-
turned.
 * @return The node count as int. Returns -1 if given parameter is not
non-empty array.
 */
function oppari_get_node_count_by_tids($tids){

    // Check that there is atleast one tid in provided array
    if(empty($tids) || !is_array($tids)){
        return -1;
    }

    // Create the query which will retrieve the nodes whit the first
id
    $query = db_select('taxonomy_index', 't')->condition('t.tid',
$tids[0]);

    $count = count($tids);
    if($count > 1){

        // Include (only) the nodes which have reference to all given
tids
        for($i = 1; $i < $count; $i++){
            $query->leftJoin('taxonomy_index', 't' . $i, 't.nid = t' .
$i . '.nid');
            $query->condition('t' . $i . '.tid', $tids[$i]);
        }
    }

    // Make the query as a count query
    $query->addExpression('COUNT(*)', 'count');
    $results = $query->distinct()->execute();

    // Return the count
    $result = $results->fetchObject();
    return $result->count;
}


/**
 * Get the count of nodes per term in given vocabulary.
 *
 * @param vid - The id of vocabulary which node count is updated.
 * @param county - The county term id.
 */
function oppari_get_vocabulary_node_count_per_county($vid, $county){

    // Get all terms in given vocabulary
    $query = db_select('taxonomy_term_data', 't')->fields('t', ar-
ray('tid'))->condition('t.vid', $vid);
    $tids = $query->execute();
```

```
    // Get the node count per term
    $data = array();
    foreach($tids->fetchCol() as $tid){

        // Create the count query
        $query = db_select('taxonomy_menu', 'tm')->fields('tm', ar-
ray('mlid'))->condition('tm.tid', $tid);

        $results = $query->execute();
        $result = $results->fetchObject();

        if(!empty($result->mlid)){
            $data[$result->mlid] = op-
pari_get_node_count_by_tids(array($tid, $county));
        }
    }

    return $data;
}



/**
 * Implements hook_preprocess_menu_link().
 * This is only called for menu named as "navigation".
 *
 * Replaces the total node count in taxonomy menu,
 * if current request url contains "county" GET -parameter.
 */
function oppari_preprocess_menu_link(&$variables){

    // Get reference to the array element
    $element = &$variables['element'];

    // Process only if menu is correct one
    if($element['#original_link']['menu_name'] !== 'navigation'){
        return;
    }

    // If current request has county -parameter set, use the county
specific entry count
    $county = isset($_GET['tid']) ? $_GET['tid'] : null;
    if(!empty($county)){

        // Get the previously cached node count
        $data = variable_get('information_node_count_by_term_' .
$county, array());

        // Mlid is added only added in taxonomy menus
        $mlid = isset($variables['element']['#original_link']['mlid'])
? $variables['element']['#original_link']['mlid'] : '';

        // Modify the title so that it contains the county specific
count instead of total count
        if(!empty($mlid) && in_array($mlid, array_keys($data))){
            $old = $element['#title'];
            $parts = explode(' (', $old);

            // Skip the title modification if there is no total count
```

```
            if(count($parts) > 1){
                unset($parts[count($parts) - 1]);

                $new = implode('(', $parts);
                $new .= ' (' . $data[$mlid] . ')';

                $element['#title'] = $new;
            }
        }
    }
}


/**
 * Used to update the county specific node count (articles in infor-
mation section).
 */
function oppari_update_county_node_count(){

    // Set these to correct ids
    $county_id = 2;
    $information_id = 1;

    // Set the id of county vocabulary
    $query = db_select('taxonomy_term_data', 't')->fields('t', ar-
ray('tid'))->condition('t.vid', $county_id);
    $tids = $query->execute();

    // Update the node count of each county term
    foreach($tids->fetchCol() as $tid){
        $key = 'information_node_count_by_term_' . $tid;
        $data = op-
pari_get_vocabulary_node_count_per_county($information_id, $tid);

        // Cache the result
        variable_set($key, $data);
    }
}


/**
 * Implements hook_taxonomy_vocabulary_delete().
 */
function oppari_taxonomy_vocabulary_delete($vocabulary) {
    oppari_update_county_node_count();
}

/**
 * Implements hook_taxonomy_term_insert($term).
 */
function oppari_taxonomy_term_insert($term) {
    oppari_update_county_node_count();
}

/**
 * Implements hook_taxonomy_term_update().
 */
function oppari_taxonomy_term_update($term) {
    oppari_update_county_node_count();
}
```

```
/**
 * Implements hook_taxonomy_term_delete().
 */
function oppari_taxonomy_term_delete($term) {
    oppari_update_county_node_count();
}

/**
 * Implements hook_node_insert().
 */
function oppari_node_insert($node) {
    oppari_update_county_node_count();
}

/**
 * Implements hook_node_update().
 */
function oppari_node_update($node) {
    oppari_update_county_node_count();
}

/**
 * Implements hook_node_presave().
 */
function oppari_node_presave($node) {
    oppari_update_county_node_count();
}

/**
 * Implements hook_node_delete().
 */
function oppari_node_delete($node) {
    oppari_update_county_node_count();
}

/**
 * Implements hook_taxonomy_menu_insert().
 */
function oppari_taxonomy_menu_insert(&$item) {
    oppari_update_county_node_count();
}

/**
 * Implements hook_taxonomy_menu_update().
 */
function oppari_taxonomy_menu_update(&$item) {
    oppari_update_county_node_count();
}
```