SOSIAALISEN MEDIAN SELAINPOHJAISET KOONTISOVELLUKSET

Tekniikat, haasteet ja suunniteltumallit

Joonas Toivanen

16.12.2011

Itä-Suomen yliopisto
Tietojenkäsittelytiede
Pro Gradu -tutkielma

Tiivistelmä

Koontisovellukset ja erityisesti sosiaalinen media ovat kuluneen vuosikymmenen aikana nousseet yhdeksi Internetin suosituimmista palvelumuodoista. Tutkielmassa selvitetään selainpohjaisten verkkosovellusten toteuttamiseen tarkoitettujen teknologioiden soveltuvuutta sosiaalisen median koontisovellusten toteuttamiseen. Lisäksi käsitellään tämänkaltaisten sovellusten kehittämiseen liittyviä yhteiskunnallisia ja teknologisia haasteita.

Arkkitehtuuriselta kannalta koontisovelluksia suunniteltaessa on tärkeää ymmärtää palvelukeskeisen arkkitehtuurin rooli. Osa teknologiaan liittyvistä kysymyksistä, kuten palvelinpuolen toteutuskieli ja välitysformaatti ovat tapauskohtaisia päätöksiä, joihin ei ole olemassa yleispätevää valintaa. Toisaalta taas esimerkiksi asiakaspuolen kielivaihtoehdot rajoittuvat pääsääntöisesti JavaScriptiin erikoistapauksia lukuun ottamatta. Tiedonvälitysformaatin valinnassa JSON tarjoaa tällä hetkellä XML-formaattia parempaa suorituskykyä, mutta välitysmuodon valinta on ennen kaikkea riippuvainen tietolähteen rajapinnan tarjoamista mahdollisuuksista.

Teknologisten kysymysten lisäksi on huomioitava erityisesti tietolähteiden lisensointiehdot, tietoturva sekä käyttäjien yksityisyys, sillä sosiaalisen median palvelut ja sitä kautta niitä hyödyntävät koontisovellukset saattavat päästä käsiksi varsin henkilökohtaiseen tietoon käyttäjistään.

Löydöksien pohjalta suunnittelin Feednic-nimisen sosiaalisen median koontisovelluksen varmentaakseni teknologioiden käytännön soveltuvuuden koontisovellusten toteuttamiseen.

Tutkimus on luonteeltaan pääosin kvalitatiivinen, mutta sisältää myös muutamia kvantitatiivisia elementtejä teknologiavertailujen osalta.

ACM-luokat (ACM Computing Classification System, 1998 version): C.2.2, D.2.1, K.4.2

Avainsanat: sosiaalinen media, koontisovellukset, web 2.0, tiedon aggregointi, web-ohjelmointi

SISÄLLYSLUETTELO

1 JOHDANTO

Vuosituhannen vaihteen *dot net -buumin* ja *IT-kuplan* jälkeisen *Web 2.0* -aikakauden suosituimmaksi palvelutyypiksi noussut sosiaalinen media ja erityisesti sosiaaliset verkostot ovat lyhyessä ajassa muokanneet käyttäjien verkkokäyttäytymistä niin käytettyjen palveluiden että niihin sijoitetun ajankäytön kannalta.

Yksistään Facebookin 800 miljoonaa käyttäjää kuluttavat aikaa palveluun 700 miljardin minuutin edestä kuukausittain - toisin sanoen 36 miljoonan henkilötyövuoden verran joka vuosi[1] [2]. Yhdysvaltalainen uutistoimisto CBS kutsui vuoden 2011 Egyptin vallankumousta nimellä *Egyptin sosiaalisten verkostojen vallankumous*, viitaten sosiaalisten verkostojen ratkaisevaan rooliin tiedonvälityksessä niin vallankumouksellisten kesken kuin myös maasta ulospäin[3]. eMarketer on arvioinut pelkän alan mainosliiketoiminnan arvon nousevan vuonna 2011 yli 4 miljardiin USA:n dollariin vuosittaisen kasvuvauhdin ollessa yli 100 %. Sosiaalinen media on ensimmäisenä aktiviteettina WWW:n historiassa ohittanut aikuisviihteen netin suosituimpana ajankäyttömuotona. Näiden käytösmallien muutosten valossa sosiaalisen median yhteiskunnallinen vaikutus lienee kiistämätön.

Uudet palvelut tuovat kuitenkin mukanaan uusia haasteita ja jopa varsinaisia vaaroja. Yhtenäisen verkkoidentiteetin puute, yksityisyyden ongelmat ja suurten tietomäärien hallinnan haastavuus ovat osa verkkokansalaisen arkipäivää. Toisaalta sosiaalinen media avaa aivan uusia mahdollisuuksia erityisesti *sosiaalisen älykkyyden* ja kohdennetun markkinoinnin segmenteissä. Vaikka sosiaalisen median ongelmista on uutisoitu viime aikoina varsin aktiivisesti median toimesta, kokee yli miljardi käyttäjää kuitenkin Sosiaalisen median tarjoaman *koetun hyödyn* tarpeeksi suureksi jatkaakseen sen käyttöä.

Sosiaalinen media on ollut suosittu tutkimuskohde lähivuosien aikana johtuen epäilemättä sen suosion nopeasta kasvusta, ajankohtaisuudesta ja tavallaan jopa trendikkäästä asemasta. Tutkimukset ovat kuitenkin suurilta osin keskittyneet sosiaalisen mediaan

---

[1] http://www.facebook.com/press/info.php?statistics
[2] http://www.digitalbuzzblog.com/facebook-statistics-stats-facts-2011/
[3] http://www.cbsnews.com/stories/2011/02/12/eveningnews/main20031662.shtml

ilmiönä jättäen teknologisen näkökulmat vähemmälle huomiolle. Tämä on toisaalta ymmärrettävää, sillä palveluiden toteutukseen käytetyt teknologiat ovat pääosin yhteneväisiä muiden verkkopalveluiden toteutusvälineiden kanssa[4].

Koontisovellukset ovat huomattavasti sosiaalista mediaa vähemmän tutkittu ohjelmistotuotannon haara[5]. Ne ovat ilmiönä sosiaalista mediaa uudempi ja myös loppukäyttäjien tietoisuus niiden olemassaolosta on heikompaa. Koska suurin osa koontisovelluksista on toteutettu käyttäen palvelinpohjaista arkkitehtuuria, käsittelee myös suurin osa aiheeseen liittyvästä kirjallisuudesta tätä toteutusmallia jättäen päätelaitepohjaisen suoritusarkkitehtuurin vähemmälle huomiolle.

Tutkielman tavoitteena on tutkia erilaisia menetelmiä ja teknologioita, joiden avulla voidaan rakentaa sosiaalisten medioiden tuottamaa informaatiovirtaa hyödyntäviä koontisovelluksia. Vaikka Internetin informaatiovirtojen koontiin soveltuvia standardeja ja sovelluksia on ollut saatavilla aina graafisen Internetin alkuajoista lähtien, eivät sosiaalisen median suuret toimijat ole kuitenkaan juurikaan tarjonneet tukeaan näille tekniikoilla, vaan luottaneet itse kehittämiinsä alustoihin ja rajapintamäärittelyihin. Ehdotuksia yhtenäisen standardin luomiseksi on esitetty mm. Googlen *OpenSocial*-määrityksen muodossa, mutta tähän mennessä suurin osa verkostoista on pysytellyt eristyneinä saarekkaina.

Työn tärkeimpiä osa-alueita ovat tekniseen toteutukseen liittyvien menetelmien, välineiden ja haasteiden lisäksi palveluiden ongelmat yksityisyyden ja tietoturvan tasolla. Lisäksi tutkielman kirjallisuuskatsauksessa luodaan yleiskatsaus sosiaaliseen mediaan ja koontisovelluksiin niin yhteiskunnalliselta ilmiönä että tekniikan kannalta.

Salminen et al. (2010) toteaa nykyisten koontisovellusten luomiseen soveltuvien metodien ja työvälineiden olevan toistaiseksi keskeneräisiä ja huonosti tuettuja. Sovellusten luonteesta johtuen yleispätevien työkalujen toteuttaminen on myös hankalaa ja historiallisesti syistä johtuen suurin osa työkaluista on tarkoitettu *palvelinpohjaisten (Server-*

[4] ACM-artikkelitietokanta sisältää 17 510 tieteellistä paperia hakusanalla *social media*
[5] ACM-artikkelitietokanta sisältää 647 tieteellistä paperia hakusanalla *mashup*

side) sovellusten toteuttamiseen *päätelaitteella suoritettavien (client-side)* toteutuksien sijaan.

Tulosten konkretisoinnin ja käsiteltyjen teknologioiden soveltuvuuden toteamiseksi käytännössä suunnittelin sosiaalisen median koontipalvelu Feednic:in, joka kerää informaatiota sosiaalisen median palveluista. Palvelun tarkoituksena on tarjota referenssitoteutus sosiaalisen median koontisovelluksien toteuttamiseen ja vastata aiheeseen liittyviin kysymyksiin parhaista toteutusteknologioista ja malleista.

RIA -sovellusten[6] toteutuksessa yleisesti käytettävät kolmansien osapuolten alustat kuten *Adobe Flash*, *Microsoft Silverlight* ja *JavaFX* on pääosin rajattu tutkimusalueen ulkopuolelle. Tutkielma keskittyy toteutusteknologioihin, jotka ovat hyödynnettävissä nykyaikaisissa web-selaimissa suoraan ilman erillisiä lisäosia.

[6] RIA (Rich Internet Application). Selainpohjainen, työpöytäsovelluksen piirteitä omaava web-sovellus jonka toteutuksessa käytetään usein kolmansien osapuolien selainlaajennuksia.

2 KOONTISOVELLUKSET JA SOSIAALINEN MEDIA

Tämä kappale käsittelee koontisovelluksia ja sosiaalista mediaa yleisellä tasolla pyrkien selittämään niihin liittyviä tärkeimpiä käsitteitä sekä niiden tarjoamia mahdollisuuksia. Lisäksi tutustutaan lyhyesti muutamaan sosiaalisen median palveluun sekä niitä hyödyntävään koontisovellukseen. Lopuksi keskustellaan sosiaalisen median koontisovellusten tarjoamista hyödyistä liiketoiminnan näkökulmasta.

Käytän tässä tutkielmassa termisuomennosta *koontisovellus* kuvaamaan alkuperäistermiä *mashup/hybrid-application*. Termille ei tällä hetkellä ole olemassa virallista suomenkielistä vastinetta. Sosiaalisesta mediasta ja -verkostoista puhuttaessa on huomattavaa, että sosiaaliset verkostot ovat sosiaalisen median osajoukko, eli yksi sosiaalisen median ilmenemistavoista. Näin ollen sosiaalista mediaa koskevat toteamukset koskevat pääsääntöisesti myös sosiaalisia verkostoja, mutta sosiaalisiin verkkoihin liittyvät toteamukset eivät välttämättä pidä paikkaansa puhuttaessa sosiaalisesta mediasta yleisellä tasolla.

2.1 Sosiaalinen media ja sosiaaliset verkostot

Kaplan et al. (2009) mukaan sosiaalisen median palvelut voidaan jakaa kuuteen kategoriaan *sisällön rikkauden* ja *mukanaolon aktiivisuuden* perusteella taulukon 1. mukaisesti. Tärkein kriteeri sosiaalista mediaa määriteltäessä on palvelun tyypistä riippumatta *käyttäjien luoma sisältö.*

Taulukko 1: Sosiaalisen median palvelujen kategoriat (Kaplan et al., 2009)

		Sisällön rikkaus		
		Matala	**Keskitaso**	**Korkea**
Mukanaolon aktiivisuus	**Matala**	Blogit	Sosiaalisen verkottumispalvelu (esim. Facebook)	Virtuaalimaailman (esim. Second Life)
	Korkea	Yhteistyöprojektit (esim. Wikipedia)	Sisältöön perustuvat yhteisöt (esim. YouTube)	Virtuaaliset pelimaailmat (esim. World of Warcraft)

Boyd et al. (2008) määrittelee *sosiaalisen verkoston* (*SNS, Social Networking Site*) palveluksi, jossa käyttäjät luovat itsestään (ainakin jossain määrin) julkisen profiilin, muodostavat kontaktiverkoston ja pystyvät selaamaan muiden kontaktiverkostossaan olevien käyttäjien muodostamaa sisältöä ja kommunikoimaan heidän kanssaan. Nykyiset yhteisöpalvelut tarjoavat kuitenkin lähes poikkeuksetta näiden lisäksi muita sosiaalisen median palveluita, kuten kuvagallerioita, blogeja ja keskustelumahdollisuuksia. Tunnetuin esimerkki sosiaalisesta verkostosta on Facebook.

Vaikka sosiaaliset verkostot ovat tulleet laajemman yleisön tietoisuuteen vasta hiljattain, eivät ne kuitenkaan ole uusi keksintö. Ensimmäinen miljoona käyttäjää kerännyt SNS-palvelu *SixDegrees.com* perustettiin jo vuonna 1997 – lähes 10 vuotta ennen Facebookia[7]. Nykyään sosiaalinen media on Internetin suosituin käyttömuoto, ja esimerkiksi Facebook on joillakin maantieteellisillä alueilla saavuttanut suosituimman Internet-sivuston position arvon[8]. Taulukossa 2 on listattu tämän hetken suosituimmat SNS-palvelut kävijämäärien mukaisesti (Alexa). Seuraavaksi esittelen lyhyesti muutamia tämän tutkielman kannalta oleellisia sosiaalisen median palveluita, jotka tarjoavat tietojaan koontisovellusten käyttöön erilaisten rajapintojen kautta.

[7] Facebook perustettiin vuonna 2004, mutta alunperin palvelu oli avoin ainoastaan Harvardin yliopiston opiskelijoille (rekisteröityminen vaatii sähköpostiosoitteen, joka kuului yliopiston osoiteavaruuteen). Julkisesti avoimeksi palveluksi se muuttui vuonna 2006 (Boyd et al., 2008)

[8] Alexa site statistics for Facebook http://www.alexa.com/siteinfo/facebook.com

Taulukko 2: Suosituimmat yhteisöpalvelut (Alexa, 2011)

Palvelu	Kävijöitä / kk	Perustamisvuosi	Kuvaus
Facebook	700,000,000	2004	Yhteisöpalvelu
Twitter	200,000,000	2006	Mikroblogi
LinkedIn	100,000,000	2003	Verkostoitumispalvelu
MySpace	80,500,000	2003	Yhteisöpalvelu (musiikki)
Ning	60,000,000[9]	2005	Yhteisöpalvelujen luomiseen tarkoitettu alusta
Google+	32,000,000	2011	Yhteisöpalvelu
Tagged	25,000,000	2004	Yhteisöpalvelu
Orkut	15,500,500	2004	Yhteisöpalvelu (Google)
Hi5	11,500,000	2003	Yhteisöpalvelu
MyYearBook	7,450,000		Yhteisöpalvelu / mikroblogi

2.1.1 Facebook

Facebook on vuonna 2004 avattu yhteisöpalvelu, jossa käyttäjät voivat luoda profiili-, yhteisö- ja tapahtumasivuja, joiden välityksellä kommunikoida toisilleen. Perustoimintojen lisäksi sivusto mahdollistaa myös reaaliaikaiset keskustelut, kuvagalleriat ja kolmansien osapuolten sovellukset.

Facebook on sosiaalisen median toimijoista suurin ja tunnetuin. Sen 750 miljoonaa rekisteröityä käyttäjää käyttävät kuukausittain 700 miljardia minuuttia sisällön tuottamiseen, selaamiseen ja jakamiseen (käyttäjät tuottavat keskimäärin 90 sisältöä kuukausittain ja jakavat niitä edelleen yhteensä 30 miljardia kappaletta)[10]. Vaikka Facebookin mainosmyynnin yksinään ennustetaan nousevan yli neljään miljardiin dollariin vuoden

[9] Alustan päälle toimivia verkkosivuja on n. 90 000. Kävijöiden määrä koostuu näiden sivustojen yhteenlasketusta liikenteestä. http://www.forbes.com/sites/tomiogeron/2011/06/15/with-revenue-up-400-ning-adds-paid-access-service/

[10] http://www.facebook.com/press/info.php?statistics

2011 aikana ja yhtiön kokonaisarvoksi on analysoitu jopa 50 miljardia, ei se kuitenkaan ole julkinen pörssiyhtiö[11] [12].

2.1.2 LinkedIn

LinkedIn on vuonna 2003 julkiseen käyttöön lanseerattu *verkostoitumisväline*, joka muista valituista medioista poiketen on selkeästi keskittynyt hyödyntämään sosiaalisen median mahdollisuuksia ammatillisessa viestinnässä vapaa-ajan sijaan. Sen käyttäjät voivat luoda itsestään profiilisivuja, joihin voidaan sisällyttää perustietojen lisäksi käyttäjän CV:n, suosituksia yhteistyökumppaneilta sekä kiinnostuksen kohteita. Käyttäjät voivat verkostoitua toisten käyttäjien kanssa ja käyttää palvelua mielenkiintoisten työpaikkojaan tai potentiaalisten työnhakijoiden etsimiseen.

Palvelulla on maailmanlaajuisesti 120 miljoonaa henkilökäyttäjää ja 2 miljoonaa yrityskäyttäjää, jotka suorittivat vuonna 2010 lähes 2 miljardia hakuoperaatiota[13]. LinkedIn on listautunut New Yorkin pörssiin ja se on arvioiden mukaan n. 2 miljardin dollarin arvoinen[14].

2.1.3 Google+

Internet-jätti Googlen kesällä 2011 julkaisema yhteisöpalvelu Google+ muistuttaa toiminnoiltaan suurissa määrin Facebookia: käyttäjät voivat luoda profiili-, yhteisö- ja tapahtumasivuja, joiden välityksellä he voivat kommunikoida toisilleen. Google+:ssa kontaktit lajitellaan *piireihin (Google+ Circles)*, jotka edustavat käyttäjän sidosryhmiä (työkaverit, ystävät), ja joita hallinnoimalla käyttäjä voi kohdistaa viestinsä tietylle vastaanottajaryhmälle. Vastaavasti käyttäjä voi seurata ainoastaan haluamaansa käyttäjäryhmän uutisvirtaa.

---

[11] http://www.emarketer.com/blog/index.php/quick-stat/
[12] http://online.wsj.com/article/SB10001424052748704436004576297310274876624.html
[13] http://press.linkedin.com/about
[14] http://www.bloomberg.com/news/2010-07-27/linkedin-valued-at-more-than-2-billion-after-investmentby -tiger-global.html

Palvelulla on maailmanlaajuisesti yli 25 miljoonaa käyttäjää[15]. Palvelun uutuudesta johtuen tilastoja palvelun muista demografeista ei ole saatavilla.

2.1.4 Twitter

Twitter eroaa toimintalogiikaltaan huomattavasti muista esitellyistä medioista. Viestintä tapahtuu *twiittaamalla*, eli kirjoittamalla 140 merkin mittaisia lyhytviestejä. Viestit ovat näkyvissä käyttäjän profiilisivuilla ja ilmestyvät automaattisesti toisten käyttäjien uutisvirtaan, mikäli he ovat tilanneet lähettäjän kanavan. Twitteriä kutsutaan mikroblogi -palveluksi, koska sinne välitetyt viestit ovat pituudeltaan lyhyitä ja päivitystahti on usein nopea. Twitteriä pidetään myös usein palveluna, jossa uutiset leviävät virallisia kanavia nopeammin[16].

Twitterillä on 200 miljoonaa rekisteröityä käyttäjää, jotka tuottavat vastaavaan määrän viestejä päivittäin[17]. Yrityksen arvoksi on arvioitu 7.8 miljardia dollaria ja se on mahdollisesti listautumassa julkiseksi pörssiyhtiöksi vuonna 2013[18].

2.2 Koontisovellukset

Salminen et al. (2010) määrittele koontisovelluksen ohjelmaksi, joka yhdistelemällä heterogeenista informaatiosisältöä useista eri lähteistä muodostaa uuden käyttäjäkokemuksen. Vaikka tieto on usein peräisin erillisistä www-lähteistä, voidaan lähteinä käyttää esimerkiksi yrityksen oman intranet-verkon sisältöä tai mobiililaitteiden tuottamaa paikkatietoa. Koontisovellukset sisältävät usein vaihtoehtoisen käyttöliittymän, joka mahdollistaa esimerkiksi tietojen suodatukseen tai visualisointiin liittyviä toimintoja. Määrittelyn kannalta oleellista on kuitenkin se, että informaatio, oli se sitten tekstiä, kuvaa tai paikkatietoa, tulee useammasta lähteestä.

---

[15] http://mashable.com/2011/08/02/google-plus-25-million-visitors/
[16] http://twitter.com/about
[17] http://www.bbc.co.uk/news/business-12889048
[18] http://www.sfgate.com/cgi-bin/article.cgi?f=/g/a/2011/03/04/businessinsider-twitter-valued-at-78-billion-in-private-market-auction-2011-3.DTL

Koontisovellukset voidaan kategorisoida viiteen ryhmään seuraavanlaisesti (Wong et al., 2008):

- Aggregointisovellukset
- Vaihtoehtoiset käyttöliittymät
- Personalisointi
- Keskitetty tietonäkymä
- Reaaliaikainen seuranta

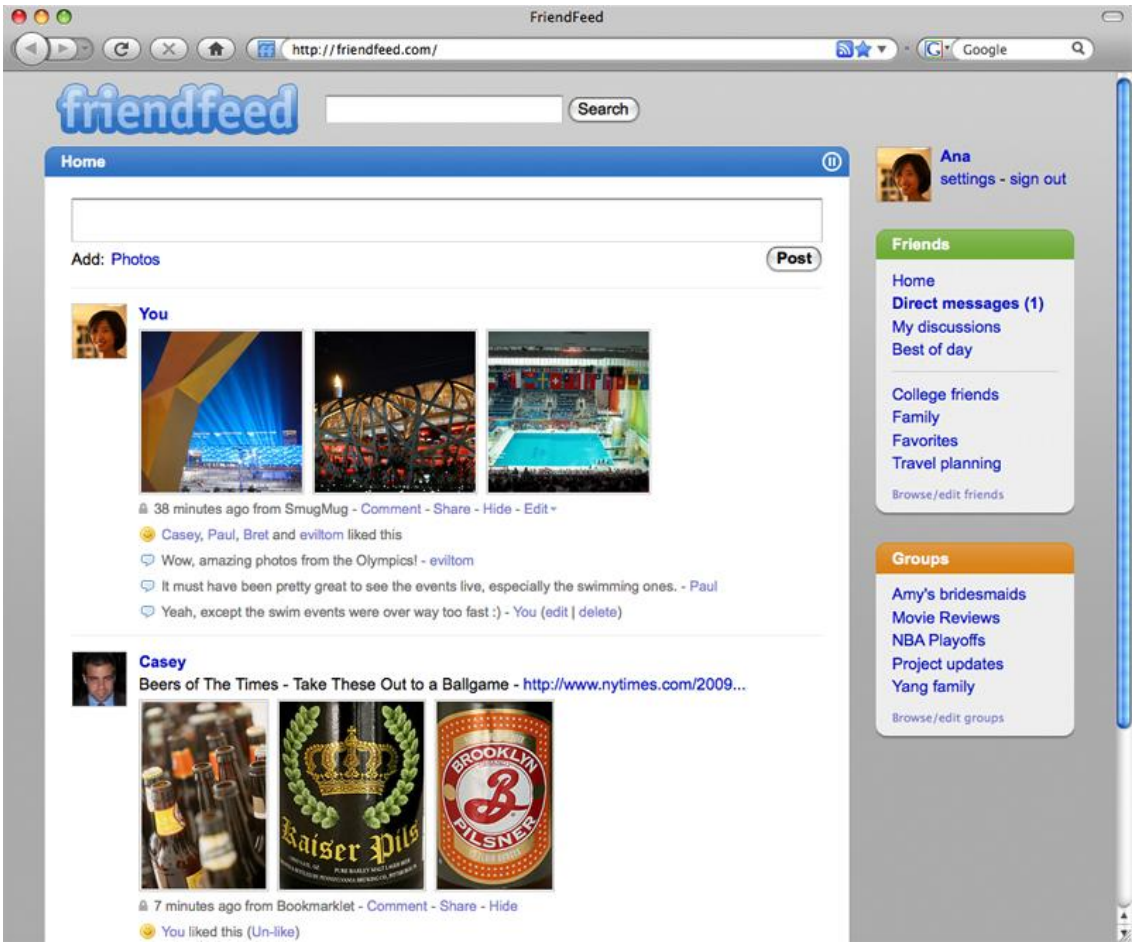
On huomattavaa, etteivät ryhmät ole toisensa pois sulkevia tai yksiselitteisiä, ja koontisovellukset sisältävät usein elementtejä useammasta eri ryhmästä. Koontisovellukset ovat ilmiönä melko uusi, mistä johtuen niiden toteuttamiseen käytetyt tekniikat ja menetelmät eivät ole vielä vakiinnuttaneet asemaansa.

Koontisovellusten suuresta määrästä johtuen niiden käsittely kattavasti tämän tutkielman yhteydessä on mahdotonta. Seuraavaksi esittelen kolme toisistaan poikkeavaa sosiaalista mediaa hyödyntävää koontisovellusta: *FriendFeed:in, BBC SoundIndex:in* ja *RSS-lukijan*. Lisäksi luodaan katsaus koontisovellusten luomiseen tarkoitettuun *Yahoo! Pipes* -palveluun ja palveluiden yhdistämiseen pyrkivään Googlen OpenSocial-alustaan.

2.2.1 FriendFeed

Vuonna 2007 perustettu FriendFeed on reaaliaikainen sosiaalisen median koontisovellus, joka nykymuodossaan tukee 58 sosiaalisen median palvelua. Sen näkökulma informaation koontiin on käänteinen verrattuna moniin muihin palveluihin: se keskittää käyttäjän oman informaation yhdeksi syötteeksi (FriendFeed:iksi), jota seuraamalla käyttäjän kontaktit saavat käyttäjän luoman informaation kosteena käymättä alkuperäisissä palveluissa[19].

[19] http://friendfeed.com/about/help



Kuva 1: FriendFeed-uutisvirta[20]

Monipuolisesta tuesta huolimatta sovelluksen tuki varsinaisille SNS-palveluille on kuitenkin heikko: tunnetuimmista palveluista tuettuna on ainoastaan Facebook[21]. Lisäksi tilapäivitykset Facebookista päivittyvät FriendSream:iin ainoastaan, mikäli päivityksen tekijällä on myös FriendStream-käyttäjäprofiili, jonka hän on liittänyt omaan Facebook-tiliinsä[22]. Kuva 1 havainnollistaa, miltä käyttäjän pääsivu voisi näyttää.

[20] (c) 2011, FriendFeed, http://friendfeed.com/about/
[21] Facebook osti palvelun itselleen Elokussa 2009
[22] http://www.pcworld.com/article/169515/3_social_media_aggregators_that_bring_it_all_together.html

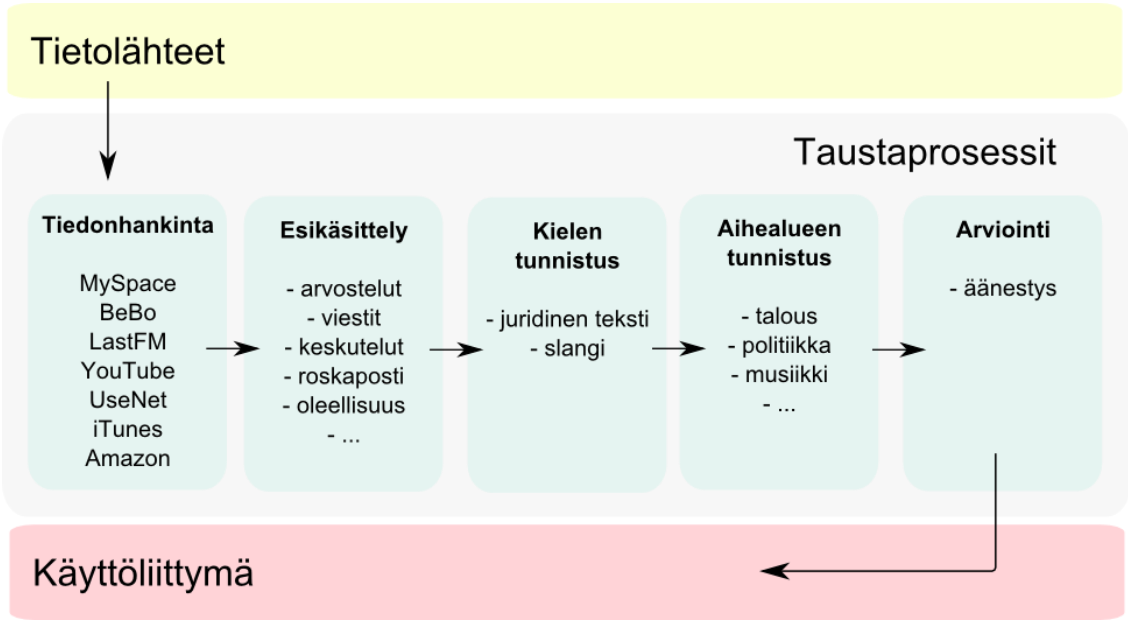
2.2.2 BBC SoundIndex

BBC SoundIndex (Gruhl et al., 2010) oli BBC:n ja IBM:n yhteistyönä syntynyt, vuonna 2008 toiminut reaaliaikainen musiikin suosiota mittaavaa järjestelmä, joka perustui ns. *joukkoälykkyyden (crowd wisdom)* hyödyntämiseen. Sen lähtöajatukseni oli, etteivät numeeriset myyntitilastot yksinään kerro koko totuutta jonkin tietyn musiikin todellisesta suosiosta. Perinteisen myyntilastojen lisäksi järjestelmä analysoi sosiaalisissa medioissa käytävien keskustelujen ja kommenttien sisältöjä, joiden perusteella järjestelmä muodosti käyttäjäkohtaisia listoja määriteltyjen parametrien, kuten musiikkityyli ja kuuntelijoiden ikä, mukaisesti.

SoundIndex:in tiedonhankintakomponentit suorittavat yli 40 miljoonaa tietoalkion hakua päivittäin valituista lähteistä, joita olivat Bebo, Google Groups, iTunes, MySpace ja YouTube. Näistä lähteistä hankitun jäsentelemättömän tiedon (kommentit, tykkäykset, jakamiset) tueksi järjestelmä hankkii numeerista informaatiota levymyynnistä sekä Amazonin että iTunesin musiikkikaupoista. Jäsentelemättömän tiedon suuri määrä tekee järjestelmästä varsin ainutlaatuisen koontisovellusten joukossa, sillä tarvittava informaatio joudutaan suodattamaan pääosin puhekielisten tekstialkioiden joukosta, joka vaatii varsin kehittyneitä kontekstianalyysin menetelmiä. Järjestelmän toteutuksessa kohdatuista ongelmista keskustelen lisää myöhemmin kohdassa *Koontisovellusten ongelmat*.

Arkkitehtuurisesti SoundIndex on jaettu viiteen peräkkäiseen komponenttiin (kuva 2), joista jokainen jalostaa sosiaalisesta mediasta kerättyä raakainformaatiota eteenpäin, kunnes se on käyttöliittymän hyödynnettävässä muodossa. Prosessin viimeiseen vaiheeseen, eli tiedon visualisoimiseen loppukäyttäjälle on kiinnitetty runsaasti huomiota, koska järjestelmän tuottaman informaation loppukäyttäjät ovat liiketoimintatietojärjestelmien yleisestä kohderyhmästä poiketen ns. peruskäyttäjiä, joille todennäköisesti ei ole tuntemusta *Business intelligence*[23] -järjestelmien käytöstä. Palvelun beta-vaihe sulkeutui vuoden 2008 lopussa, jonka jälkeen siitä ei ole ollut saatavilla julkiseen käyttöön tarkoitettua versiota.

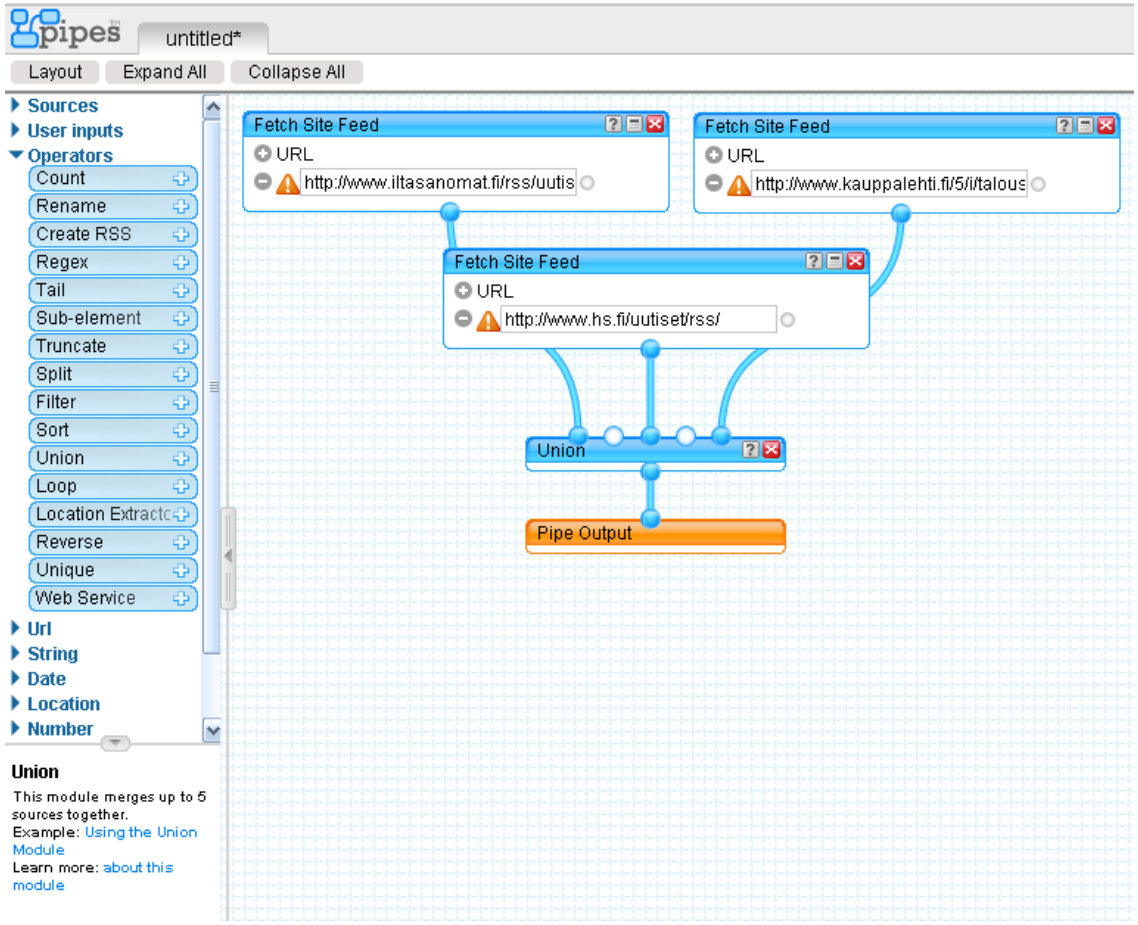
[23] Liiketoimintatiedon hallinta



Kuva 2: BBC SoundIndex -järjestelmän komponenttiarkkitehtuuri (Gruhl et al., 2010)

2.2.3 Yahoo! Pipes

Yahoo! Pipes on vuonna 2007 julkaistu WWW-pohjainen aggregointipalvelu, jonka avulla käyttäjät voivat luoda räätälöityjä syötteitä valitsemistaan tietolähteistä. Syötteet luodaan yhdistelemällä erilaisia komponentteja toisiinsa JavaScript-pohjaista graafista käyttöliittymää hyödyntäen. Tietolähdekomponenttien (esimerkiksi RSS-syötteet, verkkosivut, Flickr) tarjoamaa raakainformaatiota voidaan manipuloida, tai nimenmukaisesti putkittaa, erilaisten sääntöjen avulla. Esimerkiksi vain tietyn avainsanan sisältämät viestit voidaan suodattaa loppusyötteeseen. Kuvassa 3 on havainnollistettu yksinkertainen putki, joka noutaa uutiset RSS-syötteinä kolmesta kotimaisesta uutispalvelusta (Helsingin Sanomat, Kauppalehti ja Iltasanomat) ja yhdistää ne sitten yhdeksi RSS-syötteeksi.



Kuva 3: Yahoo Pipes:in käyttöliittymä[24]

Yahoo! Pipes on ennemmin alusta erilaisten verkkopalvelujen tietolähteiden muodostamiseen kuin loppukäyttäjälle suunnattu palvelu. Mikäli yhteisöpalvelut tarjoaisivat uutisvirtojaan RSS-syötteinä tai Pipes puolestaan ymmärtäisi näiden tarjoamia rajapintoja, tarjoaisi se valmiin ja erittäin käyttökelpoisen ratkaisun sosiaalisen median koontisovellusten toteuttamiseen. Nykyisellään se soveltuu kuitenkin huonosti tiedon keräämiseen rekisteröitymistä ja tunnistautumista vaativista palveluista, kuten sosiaalisista verkoista[25].

Muita Yahoo! Pipes:in kaltaisia palveluita ovat IBM Mashup Center, Microsoft Popfly ja lakkautettu Google Mashup Editor.

[24] http://pipes.yahoo.com
[25] http://pipes.yahoo.com/pipes/docs?doc=overview

2.2.4 OpenSocial

OpenSocial-määritys on julkaistu Google Inc.:n toimesta loppuvuodesta 2007 ja sen kehityksestä vastaan nykyisin OpenSocial Foundation. Alustan tarkoitus on tarjota kokoelma rajapintoja, joita hyödyntämällä kehittäjät voivat toteuttaa sovelluksia hyödyntäen useiden eri sosiaalisten medioiden palveluja ja tietovarastoja. Näinollen kehittäjät voivat toteuttaa sovelluksia, jotka toimivat useissa eri sosiaalisissa medioissa ilman ohjelmakoodin muokkausta (Häsel, 2011).

OpenSocial täyttää spesifikaationsa perusteella tämän tutkielman käytännön osuuden tarpeet erinomaisesti, mutta valitettavasti valittujen palveluiden sille tarjoama tuki on puutteellinen. Facebookin käyttämä *Facebook Platform* tukee ainoastaan Facebookia, ja markkinajohtajana sillä tuskin on aikeita lähitulevaisuudessa siirtyä tukemaan OpenSocial-standardia sen oman alustan menestyksen vuoksi (Häsel, 2011). Googlen oma Google+ - palvelu ei tällä hetkellä tarjoa tukea standardille, kuten ei myöskään Twitter tai LinkedIn.

OpenSocial on kuitenkin mielenkiintoinen tämän tutkielman kannalta, sillä se perustuu pääosin standardeihin teknologioihin (*OAuth* ja *RESTful API*), joista monet ovat oleellinen osa luotaessa sosiaalisen media palveluja tukevia sovelluksia. OpenSocial saattaa tulevaisuudessa tarjota varteenotettavan sovelluskehyksen markkinaosuuksien muuttuessa tai markkinajohtajien sisällyttäessä sen tarjoamat rajapinnat omiin järjestelmiinsä.

2.2.5 RSS-lukijat

RSS (Rich Site Summary tai *Really Simple Syndication)* on toistuvasti päivittyvän verkkosisällön päivitysten välittämiseen tarkoitettu formaatti. Perinteisesti RSS-syötteitä ovat hyödyntäneet RSS-lukijat, joiden avulla käyttäjät pystyvät seuraamaan keskitetysti haluamiaan WWW-tietolähteiden päivityksiä ilman tarvetta vierailla itse palvelussa. Näitä palveluja ovat esimerkiksi uutispalvelut, blogit ja *podcastit*, mutta teknisesti palvelun sisällön luonteella ei ole merkitystä. Erillisten RSS-lukijoiden lisäksi lähes kaikki

nykyaikaiset selaimet tukevat RSS-syötteitä[26]. Listaus 1 kuvaa kaksi uutispäivitystä sisältävää RSS-syötettä.

Teknisesti RSS perustuu XML 1.0 -määritykseen. RSS-syötteet omaavat hierarkkisen rakenteen ja rakentuvat <rss>-elementin alle. Koska syötteen rakenne on määritelty yksiselitteisesti spesifikaatiossa, on syötteiden aggregointi yhdistetyksi syötevirraksi melko triviaalia

```
<rss version="2.0">
    <channel>
    <title >RSS-esimerkkiuutiset </title>
    <link> http://www.example.org </link>
    <description> Uutisia Rss-muodossa </description>
    <language> fi-fi </language>
        <item>
            <title>Rss-esimerkkiuutiset avattu!</title>
            <link> http://www.example.org/?newsId=1 </link>
            <description> Example.org avaa esimerkkiuutiset-palvelun.</description>
            <pubDate>Mon, 03 Oct 2011 09:39:21 GMT</pubDate>
        </item>
        <item>
            <title>Rss-esimerkkiuutiset suljettu!</title>
            <link> http://www.example.org/?newsId=1 </link>
            <description> Example.org sulkee esimerkkiuutiset-palvelun vähäisen käyttäjämäärän johdosta.</description>
            <pubDate>Fri, 07 Oct 2011 09:39:21 GMT</pubDate>
        </item>
    </channel>
</rss>
```

Listaus 1: RSS -uutissyöte

[26] http://www.rssboard.org/rss-specification

NewsFeed RSS oli vuonna 2009 julkaistu Facebook-sovellus, joka mahdollisti RSS-syötteen muodostamisen käyttäjän omasta uutisvirrasta[27]. Facebook esti sovelluksen käytön alle viikossa vedoten tietoturvaseikkoihin ja yksityisyyden vaarantumiseen[28].

Sosiaalisen median tietovirtojen aggregointiin tarkoitetun palvelun toteuttaminen muodostamalla RSS-syötteitä palveluiden sisällöstä olisi ideaali ratkaisu, koska RSS-syötteet ovat laajalti tuettuja ja niiden seuraamiseen tarvittavat sovellukset laajasti saatavilla. Facebookin yksityisyyspolitiikka kuitenkin estää tämän ratkaisun käytön (käytännön mukaisesti yksityisyysasetusten suojaaman informaation tarjoaminen yleisesti saataville on kielletty). Esimerkiksi luomalla julkisen RSS -syötteen omasta Facebook-seinästään myös käyttäjän kontaktien luomat päivitykset näkyvät syötteessä kaikille syötteen tilaajille, vaikka Facebookin omat suojausasetukset estäisivätkin tiedon näkymisen itse Facebookin sisällä.

Yhteisöpalvelujen tarjoama tuki RSS-tekniikalle on muutamaa poikkeusta lukuun ottamatta melko vähäistä, joten näitä hyödyntävien koontisovellusten toteuttamiseen tekniikka soveltuu huonosti.

2.3 Sosiaalisen median koontisovellukset liiketoiminnan tukena

Liiketoiminnan hallintajärjestelmiä (*Business Intelligence*) käytetään hyödyllisen tiedon louhimiseksi ja jalostamiseksi informaatioarvoltaan köyhemmästä tietojoukosta, kuten yrityksen omasta *tietovarastosta.* Social Intelligence -järjestelmät puolestaan käyttävät tietovarastonaan esimerkiksi sosiaalisesta mediasta saatua informaatiota. Aikaisemmin esitelty BBC SoundIndex, joka mittaa populaarimusiikin trendejä lähes reaaliaikaisesti, on hyvä esimerkki siitä, kuinka sosiaalista älykkyyttä voidaan hyödyntää luomalla uudenlainen lähestymistapa johonkin jo olemassa olevaan ongelmaan. Sosiaalisen älykkyyden muuttaminen liiketoiminnan kannalta hyödylliseksi tiedoksi on kuitenkin haasteellista, kuten myöhemmin tulemme huomaamaan. (Gruhl, 2010.)

[27] http://www.readwriteweb.com/archives/five_things_you_can_do_with_this_new_facebook_rss.php
[28] http://www.readwriteweb.com/archives/facebook_shuts_down_rss_feed_app.php

Ongelmista huolimatta sosiaalisen älykkyyden hyödyntäminen liiketoiminnassa on kuitenkin usein kannattavaa, sillä suurilta käyttäjämääriltä saatu tieto voi olla sekä tarkempaa että nopeammin saatavilla verrattuna asiantuntijoilta peräisin olevaan. Lisäksi sosiaalisen median palvelut sisältävät huomattavan määrän tietoa, jota ei ole laisinkaan saatavilla muista lähteistä. Näin ollen sosiaalisen älykkyyden hyödyntäminen voi joissain tilanteissa olla ainoa tapa kerätä liiketoiminnan prosessien kehittämiseen tarvittavaa informaatiota. (Gruhl, 2010.)

3 KEHITTÄMINEN, TEKNOLOGIAT JA ONGELMAT

Tässä kappaleessa läpikäydään koontisovellusten toteutukseen liittyviä periaatteita, käsitteitä ja teknologioita. Lisäksi käsittelen näitä ilmiöitä koskevia yhteiskunnallisia ja teknisiä ongelmia.

3.1 Suuntaviivoja koontisovelluksen luomiseksi

Salminen et al. (2010) esittelee seuraavat kaksitoista kirjoittajien kokemuksiin perustuvaa suuntaviivaa koontisovelluksen luomiseksi.

Suunnittele sovellukset ohjelmistokehityksen periaatteiden mukaisesti. Luotaessa koontisovelluksia tulisi hyödyntää yleisiä, jo olemassa olevia suunniteperiaatteita. Käyttöliittymä-, toimintalogiikka- sekä tietokerros tulisi erottaa toisistaan ja toteutuksen tulisi pohjautua vakaisiin kirjastoihin ja rajapintoihin.

Suunnittele monimuotoiselle alustakirjolle. Erilaiset selain-, käyttöjärjestelmä- ja päätelaitekombinaatiot muodostavat heterogeenisen joukon alustaja, joilla palvelua saatetaan käyttää. Erityisesti mobiililaitteet asettavat haasteita sekä teknisen toteutuksen että käytettävyyden kannalta.

Yksi koko ei sovi kaikille. Mobiili-, työpöytä- ja palvelinkäyttöön tarkoitetut järjestelmät vaativat erilaisia suunnitteluperiaatteita. Näin ollen ohjelmasta voi joutua kirjoittamaan useita eri versioita kunkin ympäristön tarpeiden mukaisesti.

Kiinnitä huomiota turvallisuuteen. Suunnittelussa ja toteutuksessa tulisi hyödyntää verkon yleisiä turvallisuusmekanismeja, kuten SSL/TLS suojausta ja *saman alkuperän käytäntöä*. Ulkoisista lähteistä noudettujen tietoelementtien sisältö tulisi käsitellä joko selaimen tai jonkin yleisesti tuetun ohjelmistokehyksen metodeilla haitallisen koodin suorituksen ehkäisemiseksi.

Testaa huolellisesti. Huolellinen testaus sisältää myös sovellustyypille ominaisten erityispiirteiden testauksen, kuten testauksen hitaalla tai epävakaalla verkkoyhteydellä.

Kirjastojen ja selaimien tarjoamia apuvälineitä tulisi hyödyntää ja testit tulisi tehdä usealla eri kokoonpanolla (muuttujina esimerkiksi selain, käyttöjärjestelmä ja päätelaite).

Harkitse laadun arviointiin kehitettyjen ohjelmistokehysten käyttöä. Ohjelmiston rajapintoja, komponenttien tarjoamaa informaatiota, dokumentaatiota ja käyttöliittymä voidaan arvioida laadullisesti erilaisten järjestelmien avulla. Näiden järjestelmien hyödyntäminen mahdollistaa laadullisen kokonaiskuvan saamista valmiista tuotteesta.

Ole visuaalisesti vakuuttava ja kiinnitä huomiota käytettävyyteen. Informaatio tulisi yhdistää innovatiivisesti ja visuaalisesti näyttävällä tavalla, sillä loppukäyttäjät näkevät järjestelmästä yleensä ainoastaan käyttöliittymäkerroksen. Vanhahtava ja epäinnovatiivinen visuaalinen ilme voi helpottaa karkottaa käyttäjän, kun taas puolestaan kekseliäs ja näyttävä toteutus voi helpottaa monimutkaisten kokonaisuuksien hahmottamista.

Rakenna vakaalle alustalle. Tietolähteiden valinnassa tulisi suosia vakiintuneita, vakaita ja hyvin dokumentoituja verkkopalveluja. Suosi yhteisön ylläpitämiä tietolähteitä vähentääksesi riippuvuutta informaation toimittajaa kohtaan.

Ennakoi muutokset. Sovelluksen sitomista liian tiukasti johonkin tiettyyn palveluun tai välitysformaattiin tulisi välttää, kuten myös *manuaalista tiedon louhintaa (scraping)* HTML-sivuilta. Kerroksiin ja komponentteihin perustuvan arkkitehtuurin joustavuutta kannattaa hyödyntää.

Lähetä tiedoksiantoja ja tarjoa varasuunnitelma. Sovelluksen tulisi sisältää automaattinen tiedoksiantomekanismi, joka informoi kehittäjää sovellusvirheen tapahtuessa. Hyödyntämällä toissasijaisia palveluja voidaan ohjelmiston toimivuus varmistaa ongelmien ilmetessä. Myös ohjelmointiympäristön tarjoamia virheentarkistusmekanismeja tulisi hyödyntää.

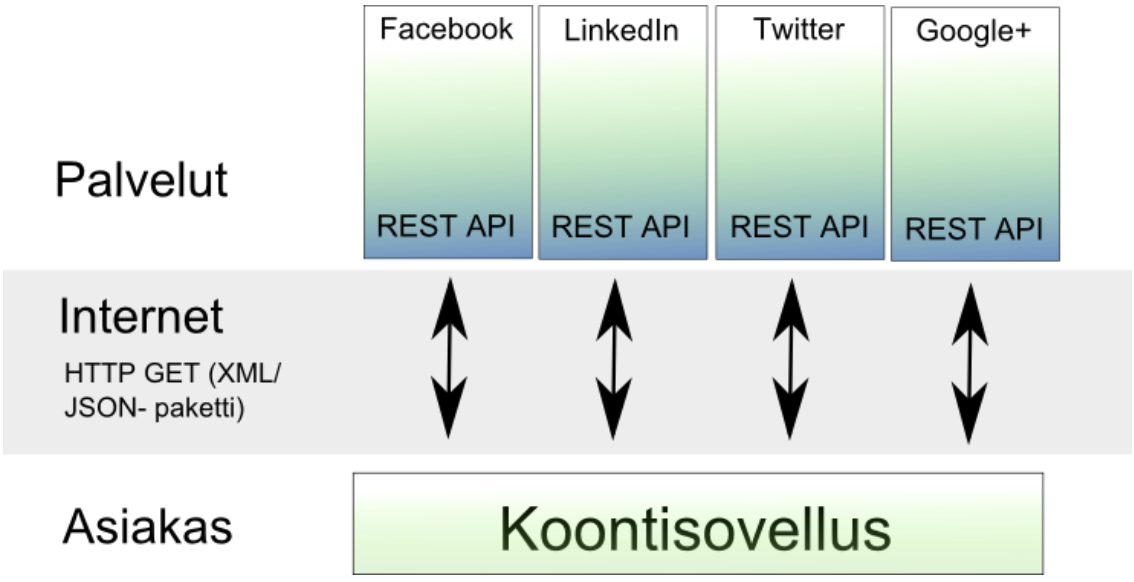
Noudata standardeja ja suosituksia. Suosi vakiintuneita tietolähteitä, rajapintoja ja avoimia, yleisesti käytössä olevia standardeja, koska standardoidussa muodossa oleva informaatio tulee todennäköisesti olevaan samassa muodossa myös tulevaisuudessa. Suositusten, ohjenuorien ja käytäntöjen seuraaminen toteutuksessa lisää ohjelmiston laatua, turvallisuutta ja suorituskykyä.

Kiinnitä huomiota lakiteknisiin seikkoihin. Suunnittelun yhteydessä tulisi kiinnittää huomiota palveluiden lisensointi- ja käyttöehtoihin. Immateriaalioikeuksien selvittäminen erityisesti manuaalisella tiedonlouhinnalla (scraping) saatuun tietoon on tärkeää.

3.2 Koontisovellusten arkkitehtuuri

Koontisovellukset ovat yleensä luonteeltaan hajautettuja järjestelmiä, joissa sovellus noutaa informaatiota useista eri lähteistä. Palvelukeskeinen arkkitehtuuri on hajautettujen järjestelmien suunnittelumalli, jossa järjestelmän komponentit toimivat itsenäisinä kokonaisuuksina, jotka kommunikoivat keskenään käyttäen standardoituja rajapintoja. (Bianco, 2007.)

Palvelukeskeisen arkkitehtuurin soveltaminen koontisovelluksen kehittämisessä ulkopuolisia tietolähteitä hyödynnettäessä on yleensä välttämätöntä, sillä sovelluksen kehittäjällä itsellään on harvoin mahdollisuutta vaikuttaa tietolähteen tapaan kommunikoida ulospäin. Sovelluksen sisäisen arkkitehtuurin ei kuitenkaan tarvitse noudattaa palvelukeskeistä suunnittelumallia. Kuva 4 kuvaa tiedonkulkua palvelukeskeisessä arkkitehtuurissa koontisovelluksen ja tietolähteiden välillä. Kaksi yleisimmin käytettyä palvelukeskeistä arkkitehtuuria toteuttavaa suunnittelumallia ovat REST ja SOAP.



Kuva 4: Palvelukeskeinen arkkitehtuuri (SOA)

REST-arkkitehtuuri on pohjimmiltaan *asiakas-palvelin -malli*, jossa osapuolet sopivat keskenään pyyntöjen ja vastauksien rakenteen vähentäen metatiedon tarvetta ja siten myös kaistan kulutusta. Sitä käytetään yleensä yhdessä HTML-protokollan kanssa hyödyntäen standardinmukaisia HTML-komentoja[29]. HTML-protokollan hyödyntäminen mahdollistaa kommunikoinnin ilman kiinteitä TCP/IP-yhteyksiä yhteyden muodostuessa ainoastaan kyselyä suoritettaessa tai siihen vastattaessa. Arkkitehtuuria hyödynnettäessä järjestelmän osat voivat olla toisistaan täysin erillisiä ja komponenttien rakenne voi muuttua, kunhan kyselyiden ja vastauksien rakenne pysyy muuttumattomana. Jokaista resurssia varten on määritelty uniikki URI-identifioija. (Christensen, 2009.)

Koontisovelluksien kannalta REST-arkkitehtuuri on olennainen, koska:

- Monet sosiaalisen median palveluiden tarjoamat rajapinnat ovat REST-yhteensopivia.
- Hyvä yhteensopivuus yleisten ohjelmointikielien, ohjelmistokehyksien ja tekniikoiden kanssa.
- Käytettäessä standardia HTML-porttia (yleensä 80 tai 8080) asiakkaan ei tarvitse avata eksklusiivista porttia sovellukselle palomuurista.
- Kiinteiden TCP/IP-yhteyksien puuttuminen ja viestien sisällön määritteleminen etukäteen keventää verkkokuormitusta

Listaus 2 noutaa Twitter-tilin *exampleAccount* 10 viimeisintä päivitystä. Se on toteutettu JavaScript-kielellä hyödyntäen Prototype-kirjastoa. Twitter hyödyntää REST-arkkitehtuuria, joten voimme suorittaa kyselyn käyttäen HTML:n GET-komentoa. Vastauksena onnistuneeseen kyselyyn palvelu palauttaa JSON-elementin, joka sisältää palvelun lähettämän resurssin sisältävän vastauksen. Huomaa, kuinka kyselyn parametrit ovat osa kutsuttavaa URI-identifioijaa.

[29] HEAD, POST, PUT, GET ja DELETE

```
new Ajax.Request('
/http://twitter.com/status/user_timeline/
exampleAccount.json?count=10&callback=JSONPcallbackFu
nction, {
                    method:'get',
  onSuccess: function(transport){
     var json = transport.responseText.evalJSON();
   }
});
```

Listaus 2: JSON-kysely REST-arkkitehtuuria hyödyntävään Twitter-palveluun

Web Services -teknologiaa hyödynnettäessä palveluiden rajapinnat määritellään käyttäen WDSL-kuvauskieltä ja kommunikointi tapahtuu käyttäen SOAP-protokollaa. SOAP on pääasiassa Microsoftin kehittämä protokolla, mutta koska se on ilmainen standardi, on sen hyödyntämiä kirjastoja saatavilla myös muille, kuin .NET-alustaan kuuluville kielille. SOAP-viestit perustuvat XML-kieleen. (Bianco, 2007.)

SOAP:in hyödyt ovat pääosin samankaltaiset kuin aikaisemmin listatut REST-protokollan vahvuudet. Yhteisöpalvelujen tarjoama tuki sille on kuitenkin lähes olematonta. Mulligan et al. (2009) selventää osaltaan sen vähäistä tukea palveluiden keskuudessa. Tutkimuksen mukaan SOAP:in perustuvien järjestelmien latenssi ja verkkokuormitus voivat joissakin tilanteissa olla jopa lähes kolminkertaiset verrattuna REST-protokollaa hyödyntävään toteutukseen.

3.3 Informaation noutaminen: rajapinnat ja kerääjät

Informaation noutamiseen käytetyt tekniikat voidaan jakaa karkeasti kahteen kategoriaan: *rajapintoihin ja kerääjiin* (*crawler/scraper*) (Gruhl et al., 2010).

Rajapinnat (API) ovat verkkopalveluiden määrittelemiä kutsu/vastaus-yhdistelmiä, joiden avulla ne tarjoavat palveluita kolmansien osapuolten sovelluksille. Viestintä perustuu usein joko SOAP- tai REST-määritykseen ja tiedon siirrossa hyödynnetään laajalti sekä JSON- että XML-rakenteita. Rajapinnat ovat oleellinen osa SOA-arkkitehtuuriin perustuvien järjestelmän toteutuksessa.

Salminen et al. (2010) mukaan kerääjiä voidaan hyödyntää raakatiedon louhimiseen silloin, kun ohjelmistorajapintaa tiedon noutamiseen ei ole saatavilla. Tällöin ohjelmisto kerää ja jäsentelee tietoa manuaalisesti HTML-dokumentin sisällöstä. Menetelmä on kuitenkin yleensä varsin hidas ja virhealtis, koska lähdetietoa ei ole rakenteistettu, vaan ohjelmisto joutuu päättelemään tiedon sijainnin dokumentissa sääntöjen perusteella.

Monien sosiaalisen median palveluiden tapaan Facebook, Twitter, LinkedIn ja Google+ tarjoavat varsin kattavan rajapinnan, joten kerääjien käyttäminen näitä palveluita hyödynnettäessä ei ole järkevää[30]. Lähtökohtaisesti kerääjien käyttö on lähes poikkeuksetta hitaampaa, virhealttiimpaa ja vaikeammin toteutettavaa kuin rajapintojen hyödyntäminen, joten niiden käyttöä tulisi välttää mahdollisuuksien mukaisesti.

3.4 Toteutuskielet, ohjelmistokehykset ja kirjastot

Käytettävä alustavalinta luo lähtökohdan koontisovelluksen toteuttamiseen käytettyjen ohjelmointikielien valinnalle. Web-kehityksessä hyödynnetyt ohjelmointikielet voidaan jakaa palvelimella suoritettaviin ja käyttäjän päätelaitteella suoritettaviin kieliin.

3.4.1 Palvelinkielet

Palvelinpuolen ohjelmointikieliä käytetään HTML-sivujen luomiseen dynaamisesti. Käytetyimpiä kieliä ja alustoja ovat *JAVA (JavaServerPages)*, *PHP*, *Ruby (Ruby on Rails)* ja *ASP.NET*. Muita kategoriaan kuuluvia ohjelmointikieliä ovat *ColdFusion*, *Python* ja *Perl*. Palvelinpuolen kieltä valittaessa tulee varmistaa itse palvelinohjelmiston tuki kyseiselle kielelle. Kaavio 1 kuvaa kielien markkinaosuutta vuonna 2011.

Ruby on puhtaasti oliopohjainen yleiskäyttöinen kieli, jonka kehityksessä on pyritty yhdistämään oliopohjaisten kielien runsaat ominaisuudet komentosarjakielien yksinkertaisuuteen[31]. Koska Ruby itsessään on yleiskäyttöinen ohjelmointikieli, käytetään sen yhteydessä *Ruby on Rails (Ror)* -ohjelmistokehystä verkkopalveluja toteutettaessa. Käyttöliittymien suunnittelussa RoR hyödyntää laaja-alaisesti Open Source -pohjaisia

[30] Google+:san tarjoaman API:n palvelut ovat vielä toistaiseksi varsin rajoittuneet johtuen rajapinnan tämänhetkisestä beta-vaiheesta.
[31] Ruby specification (final draft)
http://www.ipa.go.jp/osc/english/ruby/Ruby_final_draft_enu_20100825.pdf

JavaScript-kirjastoja kuten jQuery. Tiedonvälityksen osalta RoR on siirtynyt SOAP-protokollasta REST-protokollaan[32]. Sekä Ruby että Ruby on Rails ovat saatavilla ilmaiseksi Open Source -lisenssien alaisuudessa[33].

PHP (PHP: Hypertext Preprocessor) on ylivoimaisesti eniten käytetty palvelinpuolen ohjelmointikieli[34]. Toisin kuin monet uudemman ohjelmointikielet, se on alun perin suunniteltu toteuttamaan proseduraalista paradigmaa oliopohjaisen sijaan. Kieli on kuitenkin myöhemmin päivitetty sisältämään olio-ohjelmoinnin vaatimat rakenteet. PHP:lle on kehitetty useita ohjelmistokehyksiä kuten Yii ja Codergniter yksinkertaistamaan ja nopeuttamaan verkkosovellusten luomista, koska PHP tai sen luokkakirjastot eivät itsessään tarjoa suoria metodeja esimerkiksi REST-kutsujen tekemiseen[35]. PHP on saatavilla Open Source lisenssin alaisuudessa[36].

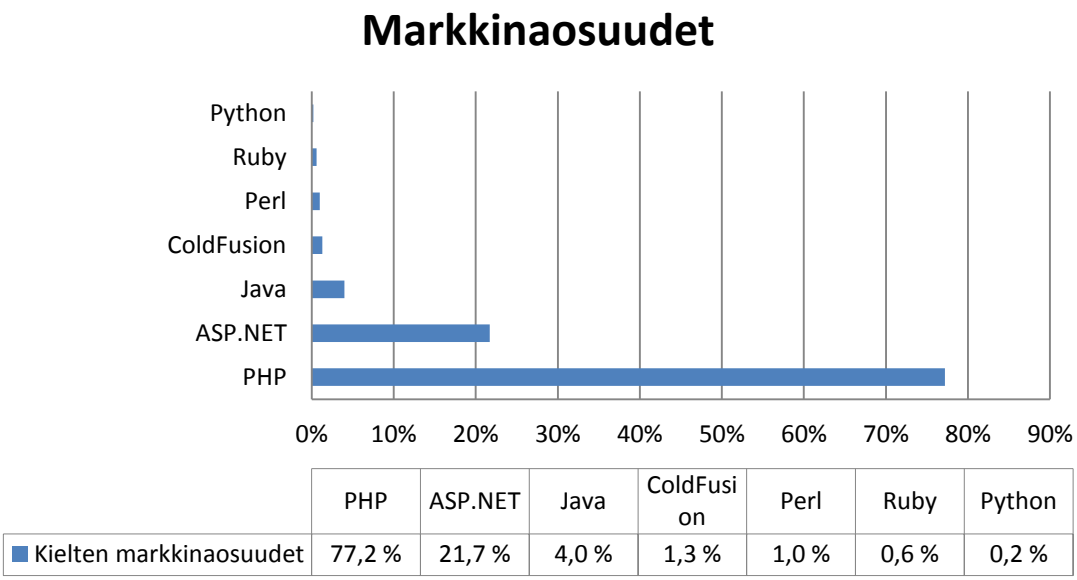
Termi **Java** käsittää itse ohjelmointikielen lisäksi myös siihen liittyvän virtuaalikoneen ja luokkakirjaston. Toisin kuin monet kilpailijansa, Java on varsin yleiskäyttöinen kieli, ja suurin osa sitä käyttävistä sovelluksista ei ole verkkosovelluksia. Javaa voidaan käyttää sekä päätelaitepuolen (*Java Applet*) että palvelinpuolen (*JSP, Java EE*) toteutuksissa, joskin jälkimmäinen käyttötapa on näistä yleisempi. Näistä esimerkiksi Java EE tarjoaa varsin laajan tuen sekä SOAP- että REST-määrittelyille. Java on saatavilla Open Source -lisenssin alaisuudessa[37].

ASP.NET on Microsoftin .NET sovelluskehykseen perustuva määritys verkkosivustojen ja -palvelujen kehitykseen. Toisin kuin muut esitellyt teknologiat, ASP. NET ei ole sidottu yhteen ohjelmointikieleen, vaan arkkitehtuuri tukee kaikkia .NET määritykseen kuuluvia kieliä (mm. *C#, C++, VB.NET).* ASP.NET nojaa tiedonvälityksessä vahvasti SOAP-määrittelyyn, joka on myös pääosin Microsoftin kehittämä teknologia. Määritys

---

[32] Ruby on Rails virallinen sivusto http://rubyonrails.org/
[33] Ruby License/Gnu General Public license v2 (Ruby) ja MIT License (Ruby on Rails)
[34] http://w3techs.com/technologies/overview/programming_language/all
[35] PHP Reference Manual http://www.php.net/manual/en/index.php
[36] PHP License http://www.php.net/license/
[37] GNU General Public License http://www.gnu.org/licenses/gpl-3.0.html

on lisensoitu suljetun lisenssin alaisuuteen, josta johtuen sen käyttöjärjestelmä- ja palvelinohjelmistotuki ovat varsin rajoitetut[38] [39].



	PHP	ASP.NET	Java	ColdFusion	Perl	Ruby	Python
Kielten markkinaosuudet	77,2 %	21,7 %	4,0 %	1,3 %	1,0 %	0,6 %	0,2 %

Kaavio 1: Yleisimpien palvelinpuolen ohjelmointikielten hyödyntäminen nettisivuilla (W3Techs, 2011)[40]

Palvelinpuolen kielten suorituskykyä on tutkittu jonkin verran. Trent et al. (2008) osoittavat kielten suorituskyvyn olevan melko yhtenäisiä, ja niinpä valinnassa voi painottaa muita muuttujia, kuten toteuttajien kokemusta, käytettyä alustaa, hintaa ja yhteisön tarjoamaa tukea.

3.4.2 Komentosarjakielet, AJAX ja kirjastot

Koska palvelinpuolen kielet tuottavat lopputuotteenaan pääsääntöisesti HTML-standardin mukaista sisältöä, ei toteutuskielellä ole käyttäjän selaimen kannalta merkitystä. Asiakkaan selaimessa suoritettavan komentosarjakoodin osalta tilanne on toinen, sillä pystyäkseen suorittamaan komentosarjakoodia, on selaimen implisiittisesti tuettava sitä. Tästä johtuen JavaScript-kielestä on tullut web-suunnittelun de facto -standardi, koska pääsääntöisesti kaikki selaimet tukevat sitä. Hallitseva valta-asema rajoittaa tehokkaasti muiden kielien yleistymistä, koska uudella kielellä toteutetut palvelut eivät

[38] MONO on Open Source -projekti, joka pyrkii toteuttamaan ASP.NET määritykseen avoimen lähdekoodin alaisuudessa. Se ei ole kuitenkaan täysin yhteensopiva alkuperäisen määrityksen kanssa.
[39] ASP.NET Developer Center http://msdn.microsoft.com/fi-fi/asp.net/
[40] Usage of server-side programming languages for websites. Sivustot voivat käyttää useita kieliä rinnakkain. http://w3techs.com/technologies/overview/programming_language/all

toimi käyttäjien selaimissa, ennen kuin selaintuottajat toteuttavat määrityksen osana selaintaan ja käyttäjät päivittävät selaimensa uuteen versioon. JavaScriptin lisäksi on olemassa kuitenkin useita selainkohtaisia kieliä kuten VBScript (Internet Explorer) ja XUL (Mozilla Firefox).

JavaScriptiä käytetään verkkopalveluissa pääasiassa dynaamisten sivustojen sekä erityisesti niiden käyttöliittymien toteutuksessa. Kieli tukee sekä olio-, imperatiivista- että funktionaalista ohjelmointiparadigmaa. Se kehitettiin alunperin[41] osaksi Netscape Navigator -selainta ja se standardoitiin loppuvuodesta 1996. JavaScriptiä voidaan käyttää myös palvelinpuolen ohjelmointikielenä, joskaan tämä ei ole kovin yleistä[42].

JavaScriptiä käytettäessä hyödynnetään usein kolmansien osapuolten tarjoamia ohjelmistokehyksiä ja kirjastoja. Käytetyimpiä kirjastoja ovat mm. JQuery, MooTools, Prototype ja Dojo. Käytetyn kirjaston valintaan vaikuttavat:

- Toivottu toiminnallisuus
- Palvelimen konfiguraatio
- Selainyhteensopivuus
- Kirjaston koko

Eri kirjastot tarjoavat erilaisia toiminnallisuuksia, jotka usein vaikuttavat järjestelmän lopullisiin selainvaatimuksiin. Kirjastojen valintaan vaikuttavat myös käytettävä palvelinkonfiguraatio, koska Ajaxia hyödyntävät sovellukset kommunikoivat usein palvelimen kanssa. Jotkin kirjastot, kuten *ASP.NET AJAX*, tukevat vain ennaltamääriteltyjä palvelinpuolen ohjelmointikieliä. Myös kirjastojen kokoerot ovat suuria, joten kehitettäessä sovelluksia kehittyville markkinoille, voi kirjaston kuluttama siirtokaista olla ratkaiseva tekijä kirjastoa valittaessa. Kirjastot ovat saatavilla pääsääntöisesti jonkin Open Source -lisenssin alaisuudessa. Yhdessä XML-kuvauskielen ja muutaman muun teknologian kanssa JavaScript muodostaa *Ajax*-määrityksen.

---

[41] Kieli tunnettiin nimillä *Mocha* ja *LiveScript* ennen sen muuttamista nykymuotoonsa. Nimestään huolimatta kielellä ei ole mitään tekmistä Java-ohjelmointikielen kanssa.

[42] ECMA-262 ECMAScript Language Specification 5.1 edition http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262%20edition%205.1,%20June%202011.pdf

Ajax (Asynchronous JavaScript and XML) on kokoelma web-teknologioita, joiden avulla voidaan luoda erilaisia dynaamisia www-palveluja. Termiä ei ole virallisesti määritelty, mutta yleensä siihen katsotaan kuuluvat ainakin seuraavat teknologiat[43]:

- *XHTML ja CSS*
- *DOM (Document Object Model)*
- *XML ja XSLT*
- *XMLHttpRequest*
- *JavaScript*

XHTML ja CSS määrittelevät sivuston rakenteen ja ulkonäön DOM-mallin mahdollistaessa rakenteen ja sisällön muuttamisen dynaamisesti. XMLHttpRequest huolehtii informaation siirtämisestä palvelimen ja käyttäjän selaimen välillä käyttäen XML- ja XSLT-formaatteja. JavaScriptiä käytetään yhdistämään teknologiat toisiinsa.

3.5 Informaation välitys: XML ja JSON

Tiedonvälityksessä palveluiden välillä käytetään yleensä XML:ää tai jotain muuta rakenteista formaattia, kuten JSON. Monet suosituimmat koontipalveluiden kehityksessä hyödynnettävät ohjelmistokehykset kuten Prototype ja jQuery tukevat natiivisti näiden formaattien jäsentämistä. Formaatteja yhdistää myös niiden esitysmuoto, joka joitakin erityistapauksia kuten *Binary XML* lukuun ottamatta on ihmiselle luettavaa puhdasta ASCII-tekstiä. Formaattien yhtenäisyys on hyödyllistä koontisovelluksia toteutettaessa, koska tällöin eri lähteistä peräisin olevan tiedon jäsentely voidaan toteuttaa tehokkaasti ja turvallisesti käyttämällä jo olemassa olevia työkaluja. (Salminen et al., 2010.)

Listaukset 3 ja 4 kuvaavat XML- ja JSON-notaatioiden rakenteen. Aikaisemmin esitelty RSS perustuu rakenteeltaan XML:ään, joten listaukset 1 (RSS) ja 2 (XML) ovat melko samankaltaisia. Sama informaatiosisältö esitettynä JSON-notaatiolla kuluttaa esimerkkisisällössä 28 % vähemmän tiedonsiirtokapasiteettiä verrattuna XML-notaatioon. Tämän lisäksi JSON määrittelee suoraan alkioiden tyypin (esimerkiksi

[43] Ajax: A New Approach to Web Applications http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

merkkijono, kokonaisluku), kun taas saman tiedon lisääminen XML-formaattiin joudutaan tekemään manuaalisesti, joka kuluttaa lisää siirtokapasiteettiä[44] [45].

```
{
“messageOrigin”:
{
    "serviceName": "Social Media Service",
    "serviceID" : "0001",
},
    "notification":
    [
       {
         "type"  : "friendRequest",
         "description"  : "A friend request from Mary Mayor",
         "url"  : "www.example.org/platform.php?request=123445678",
       },
       {
         "type"  : "taggedPhoto",
         "description"  : "Baby Doe has tagged a photo of you",
         "url"  : "www.example.org/platform.php?request=87654321",
       },
    ]
 }
```

Listaus 3: JSON-notaatio

[44] http://www.json.org/
[45] http://www.xml.org/

```
<notationExample>
<messageOrigin>
    <serviceName> Social Media Service </serviceName>
    <serviceID> 0001 </serviceID>
</messageOrigin>
<notifications>
  <notification>
    <type>friendRequest</type>
     <description> A friend request from Mary Mayor</description>
     <url>www.example.org/platform.php?request=123445678</url>
</notification>
<notification>
<type>taggedPhoto</type>
<description>Baby Doe has tagged a photo of you</description>
<url>www.example.org/platform.php?request=87654321</url>
</notification>
<notifications>
</notationExample>
```

Listaus 4: XML-notaatio

XML ja JSON ovat tietorakenteita, joiden sisällä informaatiota välitetään palvelulta toiselle. Määrittelyt ovat riippumattomia välityskerroksen tekniikasta ja arkkitehtuurista. Web-palveluita toteutettaessa niitä käytetään usein osana REST (Representational state transfer)-määritystä.

3.6 Tiedonlouhinnan ja koontisovellusten ongelmat

Tiedonlouhintaan ja koontisovelluksiin liittyvät ongelmat voidaan jakaa kahteen pääryhmään, teknisiin ja yhteiskunnallisiin. Tekniset haasteet liittyvät itse varsinaiseen toteutukseen yhteiskunnallisten keskittyessä enemmänkin eettisiin ja lakiteknisiin näkökulmiin.

Tässä kappaleessa esiteltyjen yleisluontoisten ongelmien lisäksi läpikäyn luvussa 4 esimerkkisovelluksen suunnittelussa ilmenneitä ongelmia, kuten rajapintakutsujen- ja teknologiaresurssien rajallisuutta. Nämä ongelmat johtuvat pääosin esimerkkisovellukseen valituista tekniikoista ja tietolähteistä, eivätkä ne ole yhtä yleispäteviä kuin tässä kappaleessa läpikäytävät asiat. Näiden rajoitteiden olemassaolo on kuitenkin hyvä tiedostaa ennen koontisovelluksen kehittämisen aloittamista.

3.6.1 Metodien ja ympäristön rajoitteet

Metodien ja ympäristön rajoitteilla tarkoitetaan koontisovellusten tekniseen toteutukseen liittyvien teknologioiden ongelmia, joiden huomiointi on tärkeää toimivan ja luotettavan lopputuloksen aikaansaamiseksi. Gruhl et al. (2010) esittää tutkimuksessaan BBC SoundIndex -järjestelmän kehittämiseen liittyneiksi suurimmiksi haasteiksi:

- Tietolähteiden haavoittuvuus. Kerättäessä tietoa reaaliajassa verkkoviiveet, kävijämäärien vaihtelut ja mahdolliset palveluihin kohdistuvat hyökkäykset tekevät tiedonsaannin epävarmaksi.
- Hajautetun järjestelmän epäluotettavuus. Edellä mainitut verkkoinfrastruktuurin ongelmat saattavat hidastaa tai pahimmassa tapauksessa estää järjestelmäkomponenttien kommunikointia.
- Informaation ajankohtaisuuden ja oleellisuuden todentaminen. Sisältöanalyysin soveltamista hankaloittaa sosiaalisessa mediassa paljon käytetty epämuodollinen kirjoitustyyli.
- Tiedonlouhinnan algoritmien soveltaminen määrällisesti suureen, monimuotoiseen sekä osittain strukturoimattomaan tietomäärään.

Hyödyntämällä useiden eri tietolähteiden päällekkäistä luonnetta voidaan haavoittuvuuteen ja epäluotettavuuteen liittyviä riskejä pienentää. Erimerkiksi karttapalveluita hyödyntävä sovellus voi noutaa karttatietoa Google Maps:in lisäksi Microsoftin Bing- tai Yahoo! Maps-palveluista (Salminen et al., 2010). Usean tietolähteen implementointi asiakasohjelmistoon kuitenkin kasvattaa kehitykseen kuluvaa aikaa ja ohjelmiston kokoa. Näin ollen myös ohjelmiston virhealttius kasvaa.

Edellä mainittujen lisäksi Zang et al. (2008) toteaa koontisovellusten toteutuksen kärsivän erityisesti seuraavilla osa-alueilla:

- Ohjelmistorajapinnan toimivuus: yleisimmin ongelmat autentikoinnissa sekä suorituskyvyssä
- Dokumentaatio: rajapintakuvausten, esimerkkien ja tutoriaalien puute
- Ohjelmoinnin yksityiskohdat: verkkopohjaisten koontisovellusten luomisen haastavuus verrattuna perinteiseen ohjelmistokehitykseen.

Lisäksi suuri määrä erilaisia selaimia ja käytettyjen tietolähteiden tarjoaman tiedon muotoilun/rajapinnan muutokset aiheuttavat ongelmia toteutustasolla erityisesti yhteensopivuuden kannalta.

3.6.2 Saman alkuperän käytäntö

XMLHttpRequest on rajapinta, joka mahdollistaa sekä normaalien- että salattujen HTTP-yhteyksien muodostamisen suoraan komentosarjakoodin suorittavan päätelaitteen ja palvelimen välille. Tietoturvasyistä johtuen rajapinnan määritys pakottaa toteutuksen noudattamaan s*aman alkuperän käytäntöä*. Tämä tarkoittaa sitä, että suoritettava koodi on noudettava samalta palvelimelta, kuin miltä se noutaa tietoa. Palvelimen nimen varmentaminen koostuu kolmesta eri osa-alueesta, joiden kaikkien tulee täyttyä: domain-nimestä, käytetystä protokollasta ja portista. Tämän käytännön tarkoitus on ehkäistä ns. *Cross site scripting* (XSS) -haavoittuvuuksia, joiden avulla hyökkääjä voi päästä käsiksi käyttäjän tietoihin[46] [47].

Koontisovelluksissa, joissa suoritettava ohjelmakoodi ja sisältö tulevat eri lähteistä, on ristikkäisnouto välttämätöntä palvelun toiminnallisuuden kannalta. Niitä kehitettäessä ongelma kierretään usein hyödyntämällä sovelluspalvelimella asennettavaa välityspalvelinta, jolloin informaatiovirta loppukäyttäjälle tulee yksinomaan sivuston ohjelmakoodin lähettäneeltä palvelimelta. Tämä lähestymistapa ei kuitenkaan sovellu palvelui-

[46] http://httpd.apache.org/info/css-security/
[47] http://www.w3.org/TR/XMLHttpRequest/

hin, jotka suoritetaan pääosin käyttäjän omalla päätelaitteella. Ongelman kiertämiseksi tarjolla on seuraavat vaihtoehdot (Salminen et al., 2010):

- *JSONP*
- *XMLHttpRequest Level 2*
- *Cross-Origin Resource Sharing*
- *Liitännäisten käyttö*

Poikkeuksen saman alkuperän käytännössä tekee HTML-dokumentin <*body*>-elementin sisään sijoitettu <script>-elementti, jota käytetään JavaScript-koodin noutamiseen ulkopuoliselta palvelimelta. Tätä hyödyntäen voidaan informaatiota (JSON-elementtejä) noutaa kolmannen osapuolen palvelimelta (Salminen et al., 2010). Menetelmää kutsutaan nimellä *JSONP (JSON with padding)*. Teknisesti JSONP on helposti toteutettavissa ja laajalti tuettu, mutta se tukee kuitenkin vain HTTP-standardin *GET-metodia*, joka rajoittaa sen käyttökelpoisuutta mahdollistaen ainoastaan informaation noutamisen[48].

Kakkostason määritys *XMLHttpRequest*-rajapinnasta mahdollistaa kommunikoinnin eri lähteistä olevan komentosarjakoodin ja tietolähteiden välillä. Tekniikan ongelmana on kuitenkin toistaiseksi huono tuettavuus, ja suosituista selaimista sitä tukevat ainoastaan Firefox (versio 3.5), Google Chrome ja Safari (versio 4). Esimerkiksi Internet Explorer -selaimen yksikään versio ei vielä tue tätä toiminnallisuutta[49].

Cross-Origin Resource Sharing -spesifikaatio mahdollistaa XMLHttpRequest-rajapinnan käytön selaimen saamien vastausten *header*-tietoja muokkaamalla. Menetelmä on kuitenkin jopa XMLHttpRequest Level 2 -määritystä huonommin tuettu, mikä rajaa sen käytännöllisyyttä[50].

HTML-standardiin perustuvia rajoitteita voidaan kiertää myös käyttämällä erilaisia kolmannen osapuolen **liitännäisiä** välityspalvelimina, koska HTML-standardin rajoit-

---

[48] http://www.json-p.org/
[49] http://www.w3.org/TR/XMLHttpRequest2/
[50] http://www.w3.org/TR/cors/

teet eivät koske näitä. Liitännäiset vaativat ohjelman asentamista selaimeen ja/tai päätelaitteelle. Esimerkkejä näistä ovat esimerkiksi *Adobe Flash ja Microsoft Silverlight*. Liitännäisten käyttöä tulisi kuitenkin harkita tarkasti yhteensopivuuden kannalta. Esimerkiksi laajalti tuettua Flash-formaattia ei tueta laisinkaan Apple Inc.:n valmistamissa tuotteissa.

3.6.3 Tunnistautuminen ja authorisointi

Tämän tutkimuksen kannalta oleellisimmat käyttäjän tunnistukseen liittyvät teknologiat ovat *OpenID* ja *OAuth*. OpenID mahdollistaa saman käyttäjätunnus-salasana -yhdistelmän käyttämisen kaikissa sitä tukevissa palveluissa, kun taas OAuth antaa käyttäjälle mahdollisuuden authorisoida palvelun pääsemään käsiksi sivuston tietoihin toisen sivuston kautta (Recordon et al., 2006 ja Hammer-Lahav et al., 2011).

Tunnistautumisen ja authorisoinnin lisäksi myös tarve keskitetylle verkkoidentiteetille on kasvanut sitä mukaa, kun käytettävien web-palveluiden määrä käyttäjää kohden kasvaa. Kyseessä eivät ole ainoastaan erilaisten käyttäjätunnus-salasana -identifioijien määrän aiheuttamat ongelmat, vaan tarve palveluiden kesken koherentille, yhteiselle profiilille. Tapiador et al. julkaisi vuonna 2008 ehdotuksen arkkitehtuurista, joka mahdollistaisi tällaisen laajennetun, sosiaalisen median tarpeisiin soveltuvan keskitetyn identiteetin. Samoihin aikoihin julkaistu OpenID 2.0 -spesifikaatio on pääosin toteuttanut Tapiadorin esittämän ajatuksen.

OpenID-pakka koostuu pienistä, sekä itsenäisesti että keskenään yhteentoimivista avoimista määrityksistä, jotka toimivat standardin HTTP-protokollan päällä. OpenID ei ole ainoastaan tunnistautumisjärjestelmä, vaan alusta, joka mahdollistaa profiilitiedon jakamisen alustaa käyttävien järjestelmien välillä sekä käyttäjien välisten viestien välittämisen. Järjestelmä on luonteeltaan hajautettu ja vapaasti hyödynnettävissä.

OAuth 2.0 -spesifikaatio[51] (Hammer-Lahav et al., 2011) listaa viisi käyttäjätunnus- salasana -yhdistelmän kolmannelle osapuolelle luovuttamiseen liittyvää ongelmaa autentikoinnin toteutustapana:

- Kolmansien osapuolen ohjelmistojen tulee tallentaa kirjautumistiedot myöhempää käyttöä varten, usein puhtaassa tekstimuodossa.
- Palvelimet joutuvat tukemaan salasanapohjaista autentikointia huolimatta sen mahdollisesti aiheuttamista tietoturvaongelmista.
- Kolmansien osapuolten ohjelmistot saavat ylimitoitetut oikeudet kohdejärjestelmän tietoihin antamatta resurssien omistajalle mahdollisuutta rajata niiden käyttöoikeuksia niin ajallisesti kuin määrällisesti.
- Resurssien omistaja ei pysty peruuttamaan kolmannen osapuolen ohjelmiston käyttöoikeutta muutoin kuin vaihtamalla salasanansa. Salasanan vaihtumisen myötä myös muiden resurssia käyttävien ohjelmien pääsy resurssiin estyy.
- Kolmansien osapuolen ohjelmiston tietoturvan vaarantuessa myös kohderesurssit sisältävän järjestelmän tietoturva vaarantuu.

OAuth 2.0 -protokolla mahdollistaa asiakasohjelman käyttöoikeuksien eriyttämisen resurssin omistajan, yleensä loppukäyttäjän, käyttöoikeuksista. Sen sijaan, että asiakasohjelma käyttäisi resurssin omistajan tunnuksia, sille myönnetään *käyttöoikeus (access token)* resurssiin. Käyttöoikeuden parametrit asetetaan asiakasohjelman tarpeen mukaisesti, jolloin asiakasohjelma pääsee käsiksi vain sille osoitettuun tietoon ja vain käyttöoikeuden voimassaoloajan mukaisesti.

3.6.4 Anonymiteetti

Ortmann et al. (2011) tutkimus osoittaa sosiaalisten medioiden ylläpitäjien tietävän käyttäjistään paljon enemmän kuin uskomme. Sosiaalisten medioiden ansaintamallit perustuvat pääosin laajan profiilitietovaraston hyödyntämiseen, joko mainosten välityksen tai tiedon jälleenmyynnin avulla. Liiketoiminnan näkökulmasta käyttäjän yksityisyyttä voikin pitää yritykselle kuluna, jota karsimalla voidaan kasvattaa yrityksen tuot-

[51] OAuth 2.0 -spesifikaatioon liittyvät tiedot perustuvat draft-vaiheessa olevaan luonnokseen. Spesifikaation sisältö voi muuttua ennen sen lopullista hyväksyntään.

tavuutta. Odotetusti yhteisöpalvelut käyttävätkin kaikki saamansa mahdollisuudet tiedon keräämiseen.

Käyttäjän ei usein tarvitse luovuttaa tietoa itse, vaan informaatiota voidaan kerätä käyttäjän toimintaa ja kontekstia seuraamalla tiedonlouhinnan menetelmillä. Huomattavaa on myös, ettei käyttäjän itsensä tarvitse luovuttaa tietoa itsestään, vaan hänen verkostonsa saattaa tehdä sen hänen puolestaan. Tästä esimerkkinä Ortmann mainitsee sovelluksen, joka pyytää suosittelemaan itseään muille aihealueesta (kuten tuote) mahdollisesti kiinnostuneille jäsenille.

Edellisten menetelmien lisäksi sosiaalisen median palvelut käyttävät myös yleisesti Internetissä käytettyjä tiedonkeruumenetelmiä, joiden avulla saadaan selville esimerkiksi käyttäjän sijainti (*IP-osoite*), käytetty selain, sekä sivuston osoite, josta käyttäjä on palveluun tullut ja jonne palvelun jälkeen navigoinut. Näitä tietoja yhdistelemällä voidaan jälleen päätellä käyttäjästä lisää. Mikäli esimerkiksi kaksi käyttäjää kirjautuu palveluun toistuvasti samasta IP-osoitteesta, voidaan heidän päätellä mahdollisesti olevan läheisessä kanssakäymisessä myös reaalimaailmassa.

Sen lisäksi, että yhteisöpalvelut tietävät tarkasti tekemisesi palvelun sisällä, osa niistä seuraa käyttäjän toimintaa niiden ulkopuolella. Facebookin *tykkään*-painikkeen sivuilleen integroineita palveluita on Internetissä jo yli 2 miljoonaa (Ortamann et al., 2011). Käyttäjän navigoidessa toiminteen sisältävälle sivustolle saa Facebook tiedon käyttäjän vierailusta sivustolle varustettuna edellisessä kappaleessa mainituilla perustiedoilla. Käyttäjän ei tarvitse välttämällä olla kirjautuneena palveluun, vaan hänet voidaan tunnistaa esimerkiksi IP-osoitteen tai selaimen *evästeiden* avulla[52].

Yksityisyydestä huolestuneiden Internet-käyttäjien tulisi olla tietoisia, että heistä voidaan kerätä suuret määrät tietoa, vaikka he eivät olisikaan rekisteröityneet palveluun, puhumattakaan palvelun käyttäjistä. Lisäksi on tärkeää huomioida, että palveluiden yksityisyysasetukset vaikuttavat ainoastaan käyttäjäntietojen näkymiseen muille käyttäjille, eivät alustan ylläpitäjälle. Yksityisyyteen liittyvät ongelmat eivät kuitenkaan ole

[52] Myös Twitter, Google+ ja LinkedId käyttävät vastaavanlaisia verkkosivuihin upottetuja toiminteita. Ortmann et. al ei kuitenkaan tutkimuksessaan tutkinut näiden toimintaa tietoturvan näkökulmasta.

sidoksissa pelkästään sosiaaliseen mediaan, vaan esimerkiksi Google on kerännyt huomattavia määriä tietoa Internetin käyttäjistä jo ennen Google+-palvelun lanseerausta (Conti, 2006).

Koontisovelluksien käytön kannalta nämä seikat ovat oleellisia, koska koontisovellus voi kerätä tietoa useasta eri lähteestä samanaikaisesti luoden superprofiilin käyttäjästä pohjautuen kaikkien palveluiden yhteisiin tietoihin, vaikka palvelut eivät näitä kaikkia tietoja luovuttaisikaan rajapintojensa avulla. Lisäksi käyttäjä ei voi lähtökohtaisesti olla varma siitä, mitä kaikkea koontisovellus hänen tiedoillaan tekee.

3.6.5 Alustojen ja aineiston käyttöehdot

Salminen et. al. (2010) nostaa julkaisussaan esille myös immateriaalioikeuksiin liittyvät ongelmat. Koska koontisovellukset koostavat tietoa useista eri lähteistä, täytyy toteutuksessa ottaa huomioon kaikkien näiden palvelujen käyttö- ja lisensointiehdot.

Aikaisemmin mainittu Facebook-sovellus RSS NewsFeed on esimerkki siitä, kuinka alustojen omistajat valvovat käyttöehtojen noudattamista. Sovelluksen kehittämisen määrittelyvaiheessa on tärkeää tutustua alustojen käyttöehtoihin, koska pahimmassa tapauksessa alustan käyttöehtoja rikkovan sovellukset kehitykseen panostetut resurssit voivat mennä kokonaan hukkaan.

3.7 Koontisovellusten tulevaisuus: Mobiilisovellukset, HTML5 ja CSS3

HTML-kieli soveltuu lähtökohtaisesti huonosti dynaamisten web-sovellusten toteuttamiseen, koska se on kehitetty alun perin staattisen sisällön esittämistä silmälläpitäen. Lisäksi suurin osa nykyisistä koontisovellusten kehittämiseen tarkoitetuista ohjelmistokehyksistä on tarkoitettu pääasiassa palvelinpuolella suoritettavien sovellusten kehittämiseen, tehden niistä huonosti soveltuvia asiakaspohjaista suorittamista suosivien järjestelmiin toteuttamiseen. (Salminen et al., 2010.)

HTML-standardin laajimmin tuettu versio 4.0 julkaistiin jo vuonna 1997, mistä johtuen sen tarjoama tuki web 2.0-sovelluksille on varsin heikko. Standardin kehittäjät ovat

kuitenkin huomioineet tarpeiden muutokset uudessa 5.0-versiossa. Uusi määrittely tarjoaa monia, erityisesti koontisovellusten kehittämistä helpottavia ja tehostavia tekniikoita. Vaikka useat selaimet tukevat osaa 5.0-määrittelyn uusista ominaisuuksista, on standardointiprosessi kuitenkin vielä kesken, eikä tekniikka näin ollen ole vielä kypsä tuotantokäyttöä ajatellen. Artikkelissa *Mashup development with HTML5* (Aghaee et al., 2010) keskustellaan laajasti määrittelyn tarjoamista mahdollisuuksista. Näistä tärkeimmät selityksinään on listattu taulukkoon 3.

GeoLocation API mahdollistaa paikkatietoja hyväksikäyttävien koontisovellusten toteuttamisen. Vaikka palveluiden on nykyäänkin mahdollista vastaanottaa tietoa käyttäjän sijainnista IP-osoitteen perusteella, on siihen perustuva paikannus kuitenkin epätarkkaa ja virhealtista. GeoLocation API mahdollistaa paikkatiedon keräämisen reaaliajassa useasta eri lähteestä (IP-osoite, GPS, WiFi, GSM-verkko, puhelimen tunniste). Erityisen käyttökelpoinen rajapinta on mobiilisovelluksia kehitettäessä.

Mobiilisovelluksien tarpeisiin vastaavat osaltaan myös *CSS3:n Media Queries* -määrittely sekä *Offline Caching* -rajapinta. Media Queries mahdollistaa helpon ulkoasumäärittelyn erilaisia näyttötarkkuuksia silmälläpitäen. Näin koontisovelluksesta saadaan nopeasti kehitettyä useille eri alustoille optimoitu versio ilman sen muokkaamista alustariippuvaiseksi *hybridisovellukseksi*[53]. Offline Caching taas mahdollistaa informaation varastoinnin offline-käyttöä varten. Näin ollen sovellus voidaan kehittää siten, että se käyttää aktiivista nettiyhteyttä ainoastaan informaatiota noudettaessa. Suurin osa mobiiliselaimista tukee HTML5-määrittelyjä laajasti ja esimerkiksi maailman suurin älypuhelinvalmistaja Apple Inc.[54] on kampanjoinut näyttävästi standardin puolesta jättäen tuen esimerkiksi kilpailevalle Flash-teknologialle kokonaan pois tuotteistaan[55].

WebSocket-rajapinta mahdollistaa HTML-standardin GET- ja POST-komentoihin perustuvien passiivisten yhteyksien korvaamisen aktiivisilla yhteyksillä, joita voidaan muodostaa nykyisestä poiketen myös käyttäjien välille perinteisen asiakas-palvelin -arkkitehtuurin sijaan. Nopeuden ja tehokkuuden lisäksi aktiivinen yhteys vähentää ky-

---

[53] *Hybridisovellukseksi* kutsutaan sovellusta, joka on kehitetty käyttäen mobiililaitteen natiivia sovellusympäristöä, mutta joka kuitenkin hyväksikäyttää Internetpalveluiden tarjoamia rajapintoja.
[54] http://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=6522
[55] http://www.apple.com/html5/

selyjen määrää informaatiota tarjoavan palvelun ja koontisovelluksen välillä. Turhien rajapintakutsujen määrä vähenee, kun rajapinta voi itse aktiivisen yhteyden kautta ilmoittaa uuden informaation saapumisesta.

Client-side -arkkitehtuuriin perustuvia koontisovelluksia kehitettäessä *WebWorker ja Web Storage* -rajapinnat avaavat uusia mahdollisuuksia arkkitehtuurisuunnittelussa. WebWorker -rajapinnan avulla tiedon noutaminen, integrointi ja prosessointi voidaan eriyttää käyttöliittymästä omaksi taustaprosessikseen. Web Storage taas mahdollistaa informaation tallentamisen päätelaitteen massamuistiin. Nämä ominaisuudet tuovat www-pohjaiset ohjelmistot lähemmäksi perinteisiä työpöytäohjelmistoja ja vahvistavat selaimien asemaa järjestelmäriippumattomana sovellusalustana. Saman alkuperän käytännöstä johtuvat ohjelmat on HTML5-spesifikaatiossa ratkaistu postMessage- ja Cross-site XHR -rajapintojen avulla. Näitä tekniikoita hyödyntämällä voidaan luoda puhtaita asiakaspuolen sovelluksia, jolloin kehitykseen kuluva aika voi lyhentyä huomattavasti, koska jo pelkästään tarvittavien ohjelmointikielien määrä vähenee, koska palvelinohjelmistoa ei tarvitse enää toteuttaa.

Taulukko 3: HTML5-määrityksen koontisovellusten kannalta olennaiset uudet ominaisuudet (Aughaee et al., 2010)

HTML5-ominaisuus	Selite
GeoLocation API	Sijaintitietoa hyväksikäyttävien sovellusten mahdollistaminen. Yhdistää eri lähteistä (mm. GPS, IP-osoite, WiFi) saatavan paikkatiedon ja tarjoaa sen sovellusten käytettäväksi
WebSocket API	Tehokkaampien yhteistyö- (jaettu käyttöliittymä) ja server-side -sovellusten mahdollistaminen. Mahdollistaa kiinteiden yhteyksien muodostamisen palveluiden/käyttäjien välille (vrt. POST/GET-metodien käyttö)
postMessage API	Mahdollistaa suoran kommunikoinnin UI-komponenttien *(widgets)* välillä.
WebWorker API	Mahdollistaa tiedon noutamisen, integroinnin ja prosessoinnin delegoinnin taustaprosessille client-side applikaatioita kehitettäessä.
Cross-site XHR	Ratkaisee saman alkuperän käytännön muodostamat ongelmat mahdollistamalla informaation noutamisen eri tietolähteistä.
Offline Caching API	Mahdollistaa sovellusten käytön myös offline-tilassa tilanteissa, joissa palvelun saatavuus on heikentynyt.
CSS3 Media Queries	Helpottaa skaalautuvien käyttöliittymien toteutusta eri laitteille (tietokoneet, mobiililaitteet, tabletit).

4 SUUNNITTELUESIMERKKI: FEEDNIC

Selvittääkseni teknologioiden soveltuvuutta ja niihin liittyviä löydöksiä suunnittelin sovellusesimerkkinä sosiaalisen median aggregointipalvelu nimeltä Feednic. Palvelu mahdollistaa käyttäjäkohtaisten, mukautettujen informaatiovirtojen muodostamisen valikoiduista sosiaalisen median palveluista. Suunnitelman mukaan palvelu sisältää tuen neljälle sosiaalisen median palvelulle: Facebookille, Twitterille, LinkedIn:ille sekä Google+:lle. Palvelu on prototyyppi-vaiheessa, eikä se ole julkisesti saatavilla.

4.1 Tavoitteet

Palvelun ensisijainen tavoite on tutkia esiteltyjen teknologioiden käyttökelpoisuutta koontisovelluksia toteutettaessa. Toiminnallisuuden näkökulmasta järjestelmä kokoaa valittujen tietolähteiden tietovirran keskitetysti sovellukseen helpottaen tietovirran seurantaa. Tämän perusidean lisäksi määrittelin järjestelmälle toissijaisia tavoitteita tukemaan teknisiä valintoja. Toissijaiset tavoitteet ovat:

- Laskenta- ja verkkokuormituksen mahdollisimman suuri keskittäminen loppukäyttäjän päätelaitteelle (client-side -arkkitehtuurin hyödyntäminen)
- Modulaarinen rakenne ohjelmiston ylläpidon ja laajentamisen helpottamiseksi
- Nykyaikaisten verkkoteknologioiden laaja-alainen hyödyntäminen
- Käyttöliittymän minimalistisuus ja helppokäyttöisyys

4.2 Tietolähteet

Esimerkkilähteiksi valitsin luvussa 2 esitellyt SNS-palvelut: Facebook, Twitter, LinkedIn ja Google+. Valintaan vaikuttaneita syitä olivat palveluiden homogeenisyys ja henkilökohtainen kokemus palveluiden käyttämisestä. Facebook ja Google+ ovat puhtaita SNS-palveluita, LinkedIn verkostoitumisväline ja Twitter mikroblogi.

Kaikki valitut palvelut täyttävät Boyd et al. (2008) määritelmän sosiaalisen verkoston palvelusta: Käyttäjät luovat itsestään profiilin, muodostavat kontaktiverkoston ja pystyvät selaamaan muiden tuottamaa sisältöä. Palvelut ovat perusmuodossaan käyttäjille

maksuttomia, vaikka jotkin Facebookin alustaa hyödyntävät ohjelmat ja LinkedIn:in lisäominaisuudet ovat saatavilla ainoastaan maksullisessa versiossa.

Valituista medioista kaikki tukevat tiedonvälitystä käyttäen JSON-elementtejä. Tietojen saapuminen asiakasohjelmistolle standardoidussa muodossa tuo palvelun kehityksen kannalta etuja sekä kehityssyklin pituudessa yksinkertaistaen järjestelmän rakennetta, että asiakasohjelmiston keveydessä vähentäen suorituskapasiteetin tarvetta.

Kehitysprosessia helpottavat myös palveluiden tarjoamat laajat JavaScript-pohjaiset kehitysympäristöt.

4.3 Haasteet ja ongelmat

Luvussa 3 käsiteltyjen yleisluontoisten haasteiden lisäksi toteutuksiin liittyy usein sovelluskohtaisia ongelmia, jotka ovat sidoksissa teknologia-, toteutus- ja tietolähdevalintoihin. Tässä osiossa käsittelen näistä Feednic:in kannalta oleellisimpia: rajapintakutsujen sekä verkko- ja laskentaresurssien rajallisuutta ja saman alkuperän käytäntöä.

4.3.1 Rajapintakutsujen rajallisuus

Feednic käyttää tietolähteenään kolmansien osapuolten tietovarastoja pääosin näiden tarjoamien ohjelmistorajapintojen kautta. Monet palvelut rajoittavat näiden rajapintakutsujen määrää estääkseen niiden väärinkäyttöä (esimerkiksi *DoS-hyökkäykset*) ja rajoittaakseen kolmansien osapuolten, tässä tapauksessa Feednicin, tuottamaa laskenta- ja verkkokuormaa. (Salminen et al., 2010.)

Twitterin REST API -kutsut rajoittuvat 150 kutsuun tunnissa authorisoimatonta IP-osoitetta kohden. Käytettäessä OAuth-protokollaa kutsujen määrä tunnissa on rajattu 350. Rajan ylittyessä palvelu vastaa HTTP 401 -virheellä[56].

LinkedIn rajoittaa kutsujen määrää neljän eri luokan mukaisesti: käyttäjän omistaman-, käyttäjän verkon-, ryhmä- sekä työ- ja yritystiedon mukaan. Lisäksi se asettaa omat

[56] https://dev.twitter.com/docs/rate-limiting

rajansa sovellus- ja käyttäjäkohtaisesti. Kutsujen määrä haetun tietotyypin mukaisesti sijoittuu 10 - 1000 (käyttäjäkohtainen) ja 5 000 - 500 000 (ohjelmakohtainen) päivittäisen operaation välille. Rajan ylittyessä palvelu vastaa HTTP 403 -virheellä[57].

Google+ API on toistaiseksi kehitysvaiheessa, mistä johtuen kutsujen määrä on rajoitettu tuhanteen päivässä. Kutsujen määrän kuitenkin nousee rajapinnan virallisen julkaisuversion ilmestyessä. Rajan ylittyessä palvelu vastaa HTTP 400 -virheellä[58].

Facebook ei ole virallisesti julkistanut rajapintaansa kohdistuvien kutsujen maksimimääriä. Joidenkin lähteiden mukaan kutsujen määrää olisi kuitenkin rajattu 600 kutsuun 600 sekunnin aikana *pääsyvaltuutusta* (*access token)* kohden sekä 100 miljoonaan kutsuun ohjelmaa kohden[59]. Nämä rajoitukset vaikuttavat suoraan Feednic:in arkkitehtuuriin, toteutukseen ja käytännön toimintaan seuraavasti:

Google+-integraation toteuttaminen on mahdollista, mutta tuotantokäyttöön 1000 päivittäistä kutsua ohjelmaa kohden on liian vähän. Integraatio voidaan implementoida ja ottaa myöhemmin käyttöön rajapinnan tuotantoversion ilmestyessä ja päivittäisen kutsurajan noustessa riittävälle tasolle.

Koska Twitter rajoittaa myös auktorisoidut kutsut 150 hakuun tunnissa IP-osoitetta kohden, on järjestelmän toimittava käyttäjän päätelaitteella, jolloin käyttäjien tekemän kutsut tulevat Feednic:in palvelimen sijaan heidän omista IP-osoitteistansa.

LinkedIn:in ohjelmakohtainen rajoitus aiheuttaa ongelmia suuremmilla käyttäjämäärillä. Teknisesti rajoitusten määrä vaikuttaa päivitystahtiin sekä käyttäjien maksimimäärään.

---

[57] https://developer.linkedin.com/documents/throttle-limits
[58] https://developers.google.com/+/api/
[59] http://www.quora.com/Whats-the-Facebook-Open-Graph-API-rate-limit

4.3.2 Verkko- ja laskentaresurssien rajallisuus

Feednic:illä ei ole omaa palvelinta, vaan se on sijoitettu ulkoisen hosting-yrityksen pilvipalveluun. Vaikka verkko- ja laskentaresursseja ei varsinaisesti ole palveluntarjoajan puolesta rajoitettu, on palvelu tarkoitettu perinteisten sivustojen ylläpitämiseen.

Rajalliset resurssit pakottavat järjestelmän toteutuksen noudattamaan mahdollisimman paljon asiakaspohjaista suoritusta, jossa tiedon prosessoinnista suuri osa suoritetaan hyödyntäen käyttäjän oman päätelaitteen laskentakapasiteettia. Myös aikaisemmin mainitut ongelmat rajapintakutsujen rajoittamista IP-osoitetta kohden tukevat hajautettua ratkaisua.

4.3.3 Saman alkuperän käytäntö

Koska Feednic:in ajonaikainen suoritus perustuu päätelaitteella suoritettavaan koodiin, joka ladataan sen omalta palvelimelta (www.feednic.com), ja joka kutsuu rajapintoja eri yhteisöpalveluista (esim. www.facebook.com), estää saman alkuperän käytäntö järjestelmän toteuttamisen XMLHttpRequest-rajapinnan avulla. Yleensä koontisovellukset kiertävät tämän ongelman käyttämällä hyväkseen sovelluspalvelimelle asennettua välityspalvelinta, mutta koska pyrin Feednic:in toteutuksessa minimoimaan palvelinkuorman, tämä mahdollisuus on poissuljettu. (Salminen et al., 2010.)

4.4 Teknologiat

Tässä osiossa läpikäyn Feednic:in teknologiavalinnat sekä perustelut niiden soveltuvuudesta kuvatunkaltaisen järjestelmän toteuttamiseen. Tarkastelun kohteina ovat tiedonvälitysformaatit, toteutuskielet ja niihin liittyvät kirjastot.

4.4.1 Tiedonvälitys: XML vai JSON?

XML ja JSON ovat yleisimmät verkkopalveluissa ja käytetyt tiedonvälitysformaatit. Luvussa 2 saimme viitteitä JSON-formaatin tehokkuudesta tilankäytössä verrattuna XML:llään ja todentaakseni tämän olettamuksen paikkansapitävyyden todellisissa olosuhteissa kokeilin asiaa empiirisesti.

Kokeessa hyödynnetään Twitter-palvelun ohjelmistorajapinnan tarjoamaa mahdollisuutta saada REST-kyselyiden vastaukset sekä XML- että JSON-formaatissa. Informaatiosisällöltään vastaukset ovat yhtenäisiä ainoastaan formaatin rakenteen muuttuessa. Koetta varten noudin Twitterin 10 seuratuimman tilin 10 uusinta päivitystä molemmissa formaateissa (Taulukko 4).

Taulukko 4: Twitter-haut XML- ja JSON-muodossa

Tile	Seuraajia	XML	JSON	Kokoero
ladygaga	16,015,121	26,799	18,864	42 %
justinbieber	14,589,087	8,882	6,415	38 %
katyperry	12,162,044	28,181	20,070	40 %
KimKardashian	11,365,931	27,093	19,244	41 %
BarackObama	11,160,656	25,304	17,968	41 %
britneyspears	11,128,408	27,514	19,536	41 %
shakira	10,458,657	28,155	20,040	40%
rihanna	9,667,216	28,765	20,737	39%
taylorswift13	9,218,480	18,733	13,095	43 %
aplusk	8,393,725	29,219	21,130	38 %
Yhteensä				40 %

Pienimmillään ero formaattien käyttämässä siirtokaistassa oli 38 % ja suurimmillaan 43 % keskiarvon ollessa 40 %. Näin ollen tieto noudetaan JSON-formaatissa kohdejärjestelmän tämän salliessa[60].

4.4.2 Tietojen noutaminen

Tiedon noutaminen tietolähteistä toteutetaan hyödyntäen sivustojen tarjoamia JavaScript-kirjastoja (LinkedIn) ja Prototype:n tarjoamia valmiita Ajax-kutsufunktioita (Facebook ja Twitter). Koodiesimerkit (Listaus 5) noutavat käyttäjän verkoston uusimmat päivitykset. Google+:n rajapinta on vielä kehitysvaiheessa, joten sitä ei ole sisällytetty esimerkkeihin. Toimiakseen esimerkit vaativat, että ohjelma on valtuutettu aikaisemmin pääsemään käsiksi näihin tietoihin.

[60] Kaikki tutkielmaan valitut tietolähteet tukevat JSON-formaattia

LinkedIn:

```
IN.API.NetworkUpdates()
  .params({"count":1,"type":"SHAR"})
  .result(function(result) {
       $("#stream").html(JSON.stringify(result));
  } )
```

Facebook:

```
New     Ajax.Request('/https://graph.facebook.com/me/home?access_token=
AAAAAAITEghMBAH1pj0iVN1WcHZBzpXVtk9asqXEza3yOR4f6ZCDoy26nDDbZAjQ7fyf
gkMLAkUF1SJYDzimF8lhqs3wi9hJqJRVY0lQzYT2ZB97CZCGaG, {
method:'get',
  onSuccess: function(transport){
     var json = transport.responseText.evalJSON();
   }
});
```

Twitter:

```
New Ajax.Request('/ https://api.twitter.com/1/statuses/public_timeline.
json?count=3&include_entities=true, {
method:'get',
  onSuccess: function(transport){
     var json = transport.responseText.evalJSON();
   }
});
```

Listaus 5: Uusimpien päivitysten noutaminen LinkedIn:stä, Facebook:ista ja Twitter:istä

Esimerkkikyselyiden palauttamia vastauksia en ole liittänyt tutkielmaan niiden suuren tiedostokoon takia. Kaikki vastaukset ovat JSON-muodossa (Listaus 3) ja ne täsmäsivät palveluiden tarjoamien web-syötevirtojen kanssa. Alustojen rajapinnat ovat pääosin melko vakaita, joten en suorittanut pidempiaikaista testausta todentaakseni niiden luotettavuutta[61].

[61] Facebook ja Twitter tarjoavat reaaliaikaisen seurannan rajapintojen tilasta
https://dev.twitter.com/status
http://developers.facebook.com/live_status/

4.4.3 Toteutuskielet ja käyttöliittymä

Palvelinpuolen ohjelmointikieleksi valitsin PHP:n. Valintaan vaikuttaneita tekijöitä olivat:

- Aikaisempi ohjelmointikokemus kielellä
- Saatavilla ilmaiseksi
- Laajalti tuettu

Feednic:in käyttöliittymä perustuu dynaamiseen HTML eli DHTML:än. Sivustolla ei ole kiinteästi määriteltyä rakennetta, vaan sisältöelementit muodostetaan suorituksen aikaisesti käyttäjän toimintojen perusteella. Tämän toiminnallisuuden saavuttamiseksi ilman kolmansien osapuolten liitännäisiä hyödynnän JavaScript:iä asiakaspuolen ohjelmointikielenä. Ratkaisua tukee myös se, että lähestulkoon kaikki asiakaspuolen suoritukseen vahvasti nojautuvat koontisovellukset on toteutettu sen avulla (Salminen et al., 2010).

JavaScript-kielen tueksi valitsin Prototypen ja siihen liittyvän grafiikkakirjasto Script.aculo.uk:sen. Valintaan vaikutti kirjaston laaja tuki XML- ja JSON-elementtien käsittelylle ja DOM:ille. Itse sivuston ja sen elementtien määrittelyssä käytin HTML4-kieltä yhdistettynä CSS-tyylimäärittelyihin. Teknologiat ovat Internet-sivujen kehittämisen de facto -standardeja, joten niiden hyväksikäyttäminen on käytännössä pakollista. Taulukko 5 listaa käytetyt teknologiat.

Taulukko 5: Feednic:in toteutuksessa käytetyt teknologiat

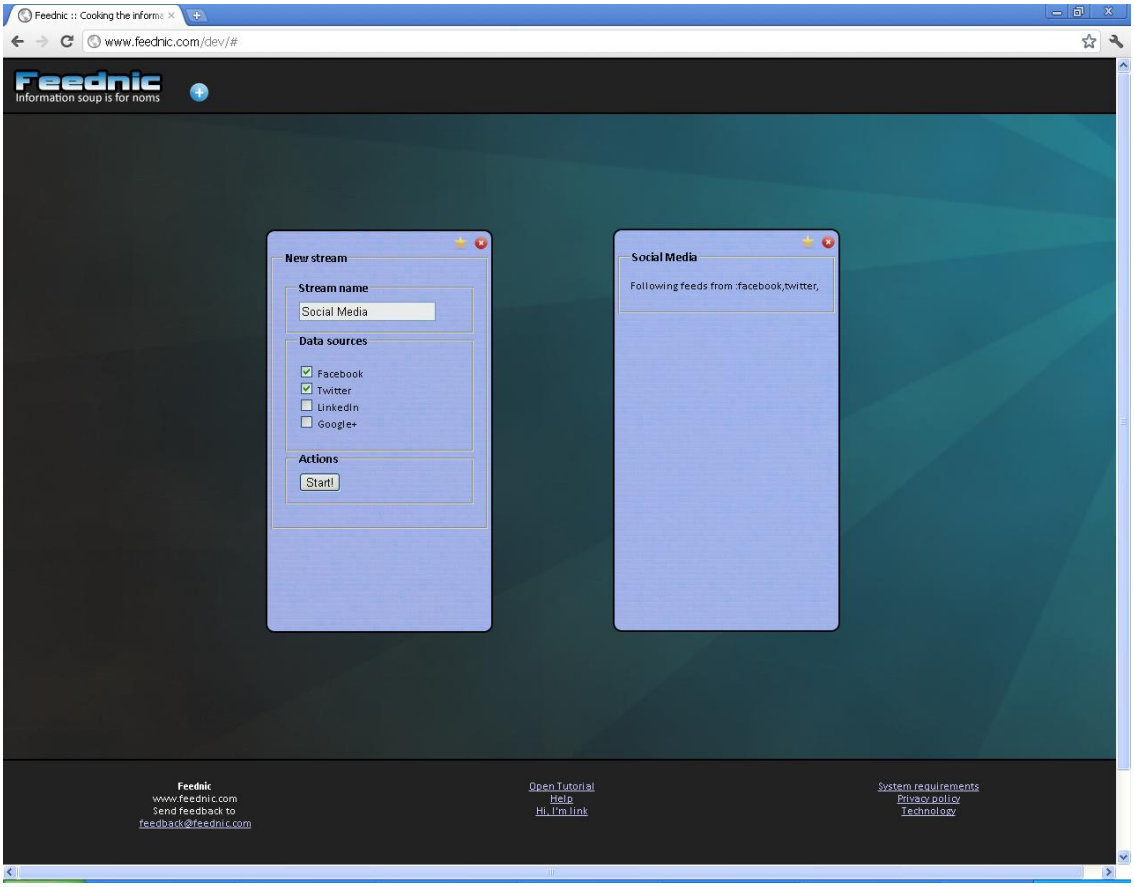
Teknologia	Selite
HTML	Elementteihin perustuva rakenteinen merkkauskieli hypertekstin kuvaamiseen. Käytetyin verkkosivujen toteutuksessa hyödynnetty teknologia[62].
CSS	Verkkosivujen ulkonäön kuvaukseen käytetty määrittely, jonka avulla muodostetaan sivun hyödyntämä tyylisäännöstö[63].
PHP	Dynaamisten verkkosivustojen kehittämiseen käytettävä ns. palvelinpuolen ohjelmointikieli, jonka ohjelmakoodi suoritetaan ajonaikaisesti[64].
JavaScript	Erityisesti WWW-ympäristössä käytettävä komentosarjakieli, joka mahdollistaa dynaamisten toiminnallisuuksien lisäämisen verkkosivulle staattisen HTML-sisällön lisäksi[65].
Prototype	Ohjelmistokehys JavaScript-kielelle, joka lisää tuen luokkamäärittelyille ja yksinkertaistaa DOM-elementtien muokkaamista[66].
Script.aculo.us	Protypen päälle rakennettu JavaScript-kirjasto, joka mahdollistaa erilaisten tehosteiden käytön sivuston elementeissä[67].

Käyttöliittymän rakentamista dynaamiseksi puoltavat:

- Suorituskapasiteetin delegoiminen palvelimelta loppukäyttäjälle
- Käyttöliittymän joustavuus
- Käyttöliittymän nopea vaste käyttäjän toimiin, kun yhteyttä palvelimelle ei tarvita. Vähentää myös tiedonsiirto kapasiteetin tarvetta.
- Visuaalinen näyttävyys (Salminen et al., 2010)

[62] HTML 4.01 Specification - http://www.w3.org/TR/1999/REC-html401-19991224/
[63] Cascading Style Sheets 2.11 Specifications - http://www.w3.org/TR/CSS2/
[64] PHP Manual, Preface - http://fi2.php.net/manual/en/preface.php
[65] ECMAScript Language Specification-http://www.ecma-international.org/publications/standards/Ecma-262.htm
[66] Prototype API-ducumentation - http://api.prototypejs.org/
[67] Script.aculo.us API Documentation - http://madrobby.github.com/scriptaculous/

Kääntöpuolena ratkaisu lisää laskentatehon tarvetta ja nostaa selainvaatimuksia. Kuva 5 esittää Feednic:in käyttöliittymää Google Chrome -selaimessa. Ylin osio toimii toimintovalikkona, josta voidaan lisätä uusia tai tallennettuja syötteitä. Sen alapuolella on itse työtila, jossa syötevirrat sijaitsevat. Kaikkein alimmaisena sijaitsee navigointipalkki, jossa on palvelun sisäisiä linkkejä informaatiosivuille.



Kuva 5: Feednic:in käyttöliittymä (prototyyppi)

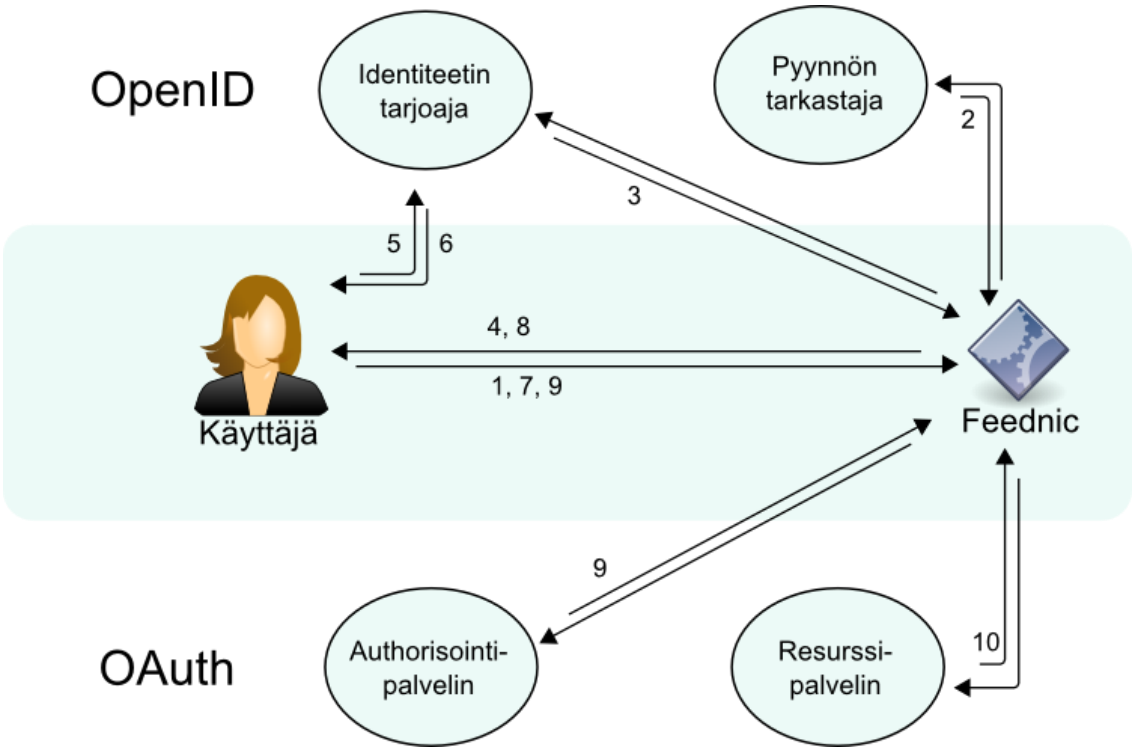
4.5 Tietoturva

Koska Feednic yhdistää tietoja useasta sosiaalisen median palvelusta, on sen toteutuksessa pyritty turvaamaan käyttäjän yksityisyys. Teknisten turvatekijöiden lisäksi suunnitelmassa on noudatettu seuraavia periaatteita:

- Ohjelma suoritetaan käyttäjän päätelaitteella, jolloin käyttäjän henkilökohtainen sisältö liikkuu ainoastaan sosiaalisen median ja käyttäjän koneen välillä.
- Käyttäjän tunnistetietoja (käyttäjänimi/salasana) käytettäviin palveluihin ei tallenneta järjestelmän palvelimille.
- Palveluista haettuja tietoja ei tallenneta järjestelmään

Feednic:in authorisointi perustuu OAuth-protokollaan. Sen lisäksi, että sen hyödyntäminen lisää toteutuksen turvallisuutta, on sen käyttäminen edellytyksenä koko järjestelmän toteuttamiselle. Esimerkiksi elokuusta 2010 lähtien kaikkien kolmannen osapuolen Twitter-sovellusten tulee toteuttaa authorisointi käyttäen OAuth-protokollaa (Hannah, 2010). Twitterin lisäksi myös Facebook ja LinkedIn hyödyntävät protokollaa tehden siitä luonnollisen valinnan käytettäväksi authorisointimenetelmäksi myös Feednic:in toteutuksessa. Myös Google tukee protokollaa monissa jo lanseeratuissa rajapinnoissaan, joten sen sisällyttäminen osaksi tulevaa Google+-rajapintaa on melko todennäköistä. Tunnistautuminen tapahtuu käyttäen OpenID-protokollaa.

OAuth ja OpenID ovat laajalti tuettuja erityisesti sosiaalisen median palveluiden piirissä, mikä tekee niiden hyödyntämiseen järkeväksi valinnaksi. Lisäksi niiden käyttäminen lisää tietoturvan tasoa niin loppukäyttäjän kuin järjestelmämme kannalta, koska palvelun ei itse tarvitse tallentaa tunnuksia prosessin missään vaiheessa.



Kuva 2: Tietoturvaprotokollien tietovuokaavio (Hammer-Lahav E., 2011 & Recordon D., 2009)

Kuva 2 havainnollistaa tietovuon käyttäjän, sovelluksen sekä OpenID- ja OAuth-protokollien välillä. Kuvaa tulkittaessa on huomattava OpenID:n ja OAuth:in olevan erillisiä ja toisistaan riippumattomia protokollia. OpenID-tunnistuksen vaiheet ovat seuraavat:

1) Kirjautumisprosessi käynnistyy käyttäjän syöttäessä URL-osoitteen tunnistautumista vaativaan palveluun
2) Palvelu noutaa käyttäjäkohtaisen *Yadis-dokumentin*, joka sisältää tiedon palveluista joihin käyttäjän OpenID-tunniste on liitetty.
3) Palvelu luo *jaetun salaisuuden henkilöllisyyden tarjoajan* (esimerkiksi Google) kanssa.
4) Palvelu ohjaa käyttäjän selaimen henkilöllisyyden tarjoajan kirjautumissivulle.
5) Käyttäjä syöttää OpenID-kirjautumistiedot henkilöllisyyden tarjoajan kirjautumissivulle
6) Henkilöllisyyden tarjoaja ohjaa käyttäjän selaimen takaisin palvelun sivulle. Uudelleenohjausosoitteen mukana kulkee myös kohdassa 3 muodostettuun jaettuun salaisuuteen perustuva kryptografinen todiste siitä, että käyttäjä on kirjautunut onnistuneesti palveluun.

7) Käyttäjä on nyt kirjautunut sisään tunnistautumista vaatineeseen palveluun.

Käytettäessä OAuth-protokollaa palvelun authorisointiin ovat tietovuon vaiheet seuraavat (Kuva 2):

8) Palvelu (esim. Feednic) pyytää *resurssin (*esim. Facebookin profiili) *omistajalta eli käyttäjältä* lupaa käyttää resurssia käyttäjän puolesta
9) Käyttäjä syöttää kirjautumistietonsa kirjautumissivulle. Kirjautumissivu sijaitsee usein resurssin omistajan palvelimella, estäen palveluntarjoajaa saamasta alkuperäisiä kirjautumistietoja Palvelu välittää tiedon onnistuneesta kirjautumisesta *authorisointipalvelimelle*
10) Authorisointipalvelin välittää palvelulle todisteen käyttöoikeuden (access token) myöntämisestä. Palvelu välittää todisteen käyttöoikeudesta *resurssipalvelimelle,* jonka perusteella palvelin luovuttaa informaatiota palvelulle.

Feednic:in toteutukseen valituista järjestelmistä Google+ ja Facebook (toissijaisena tunnistusmuotoja) tukevat OpenID-tunnistusta. Facebook, Twitter ja LinkedIn tarjoavat mahdollisuuden hyödyntää tilitietoja käyttäjien kirjautuessa kehittäjän omaan palveluun (*Facebook Connect*, *Sign in with Twitter* ja *Sign in with LinkedIn*)[68] [69] [70].

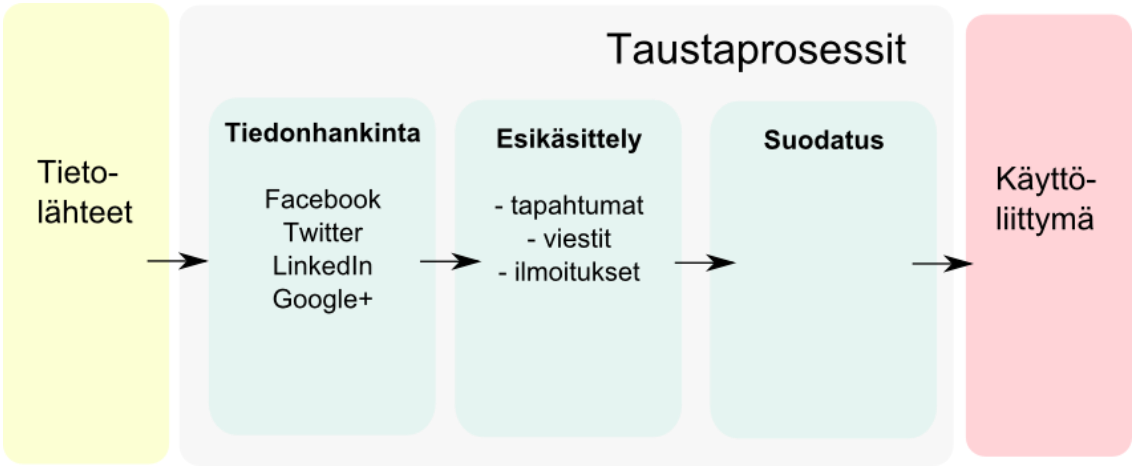
Tunnistautumis- ja authorisointiprosessit ovat monivaiheisia, mutta niiden toteuttaminen on tässä tapauksessa kuitenkin melko triviaalia. Tämä johtuu siitä, että valittujen palvelujen rajapinnat tarjoavat tunnistautumispalvelut valmiiksi paketoituina helppokäyttöisinä korkean tason kokonaisuuksina.

4.6 Järjestelmän rakenne

Feednic:in rakenteen voi kuvailla komponenttien muodostamana putkena, jonka alkupäästä raakainformaatio tulee sisään ja jonka toinen pää muodostaa loppukäyttäjän havainnoiman näkymän tiedon jalostuessa jokaisen komponentin kohdalla. Rakenne on

[68] https://dev.twitter.com/docs/auth/sign-in-with-twitter
[69] http://developer.linkedin.com/documents/sign-linkedin
[70] http://www.facebook.com/help/?page=730

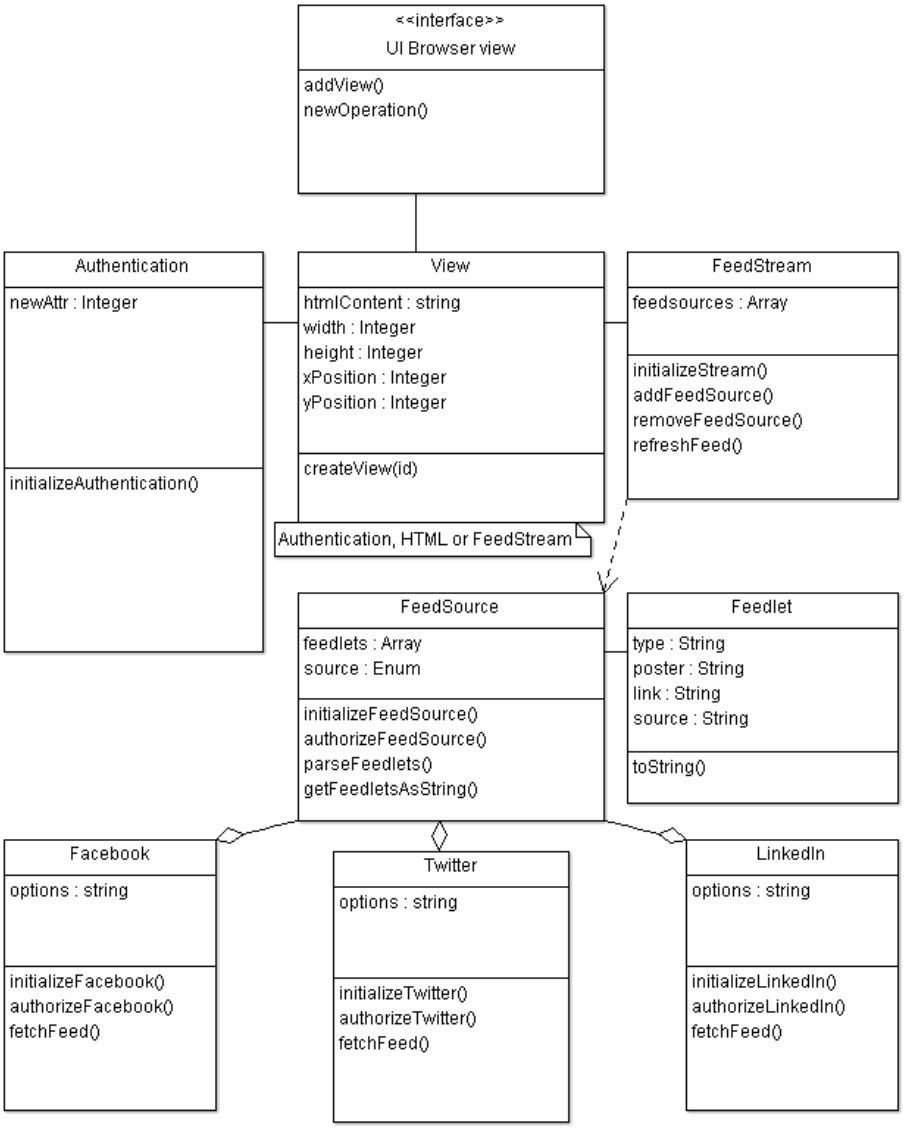
samantyyppinen kuin luvussa 2 esitellyssä BBC SoundIndex -järjestelmässä (kuva 1), joskin komponenttien määrä on pienempi johtuen Feednic:in tiedonprosessoinnin pienemmästä tarpeesta[71] (kuva 6).



Kuva 6: Feednic:in tiedonjalostusprosessi kuvattuna komponenteittain

Järjestelmän arkkitehtuurinen luokkajako on kuvattu kuvassa 7. Tietolähdekohtaiset tiedonhakijat aktivoituvat tietyin väliajoin noutamaan uutta informaatiota palveluista. Informaatio normalisoidaan yhtenäiseen muotoon (*Feedlet*) ja informaatioalkiot päivitetään käyttöliittymään.

[71] BBC SoundIndex:in käyttämä raakadata on pääosin rakenteistamatonta ja siihen joudutaan soveltamaan useita sisältöanalyysin ja tiedonlouhinnan menetelmiä, ennen kuin se on jalostettu lopulliseen muotoonsa. Feednic puolestaan noutaa tiedon valmiiksi rakenteisina JSON-elementteinä.



Kuva 7: Feednic:in arkkitehtuurikaavio

5 TULOSTEN ANALYSOINTI

Tämä luku käsittelee esiteltyjen teknologioiden soveltuvuutta sekä mallien ja ohjenuorien toteutumista esimerkkisovelluksen osalta. Tämän jälkeen läpikäydään toteutukseen liittyviä tuloksia ja niiden hyödyntämistä koontisovellusten toteutuksessa. Lopuksi keskustellaan toteutuksen yhteensopivuudesta laite- ja ohjelmistotasolla sekä järjestelmän kehittämismahdollisuuksista.

Wong et al. (2008) koontisovellusluokituksen mukaan Feednic on perusluonteeltaan aggregointisovellus. Kuten Wong et al. kuitenkin toteaa, koontisovellukset ovat harvoin kategorisoitavissa yksiselitteisesti tiettyyn ryhmään. Näin on myös Feednic:in osalta, ja järjestelmä sisältääkin elementtejä, jotka kuuluvat reaaliaikaisen seurannan ja keskitetyn tietonäkymän segmentteihin puhtaan aggregoinnin lisäksi.

5.1 Salminen et al. esittämien ohjenuorien toteutuminen Feednicissä

Kappaleessa *Suuntaviivoja koontisovelluksen luomiseksi* referoin koontisovelluksen luomiseen tarkoitettuja suuntaviivoja perustuen Salminen et al. (2010) tutkimustuloksiin. Tämän osion tarkoituksena on peilata näiden periaatteiden toteutumista esimerkkisovellus Feednic:issä ja pohtia niihin liittyviä seikkoja.

Suunnittele sovellukset ohjelmistokehityksen periaatteiden mukaisesti. Feednic:in arkkitehtuuri perustuu verkkopalveluiden toteutuksessa laajasti käytettyyn MVC-malliin, jossa käyttöliittymä-, toimintalogiikka- ja tietokantakerrokset on toteutettu toisistaan erillisinä komponentteina. Toteutuksessa on hyödynnetty laajalti käytettyä ja vakaata Prototype-kirjastoa ja sen Script.aculo.us-laajennusta.

Suunnittele monimuotoiselle alustakirjolle. Järjestelmän suunnittelussa ei ole erityisesti huomioitu muiden kuin PC-laitteiden asettamia erityisvaatimuksia. Käytetyt teknologiat on kuitenkin valikoitu sitten, etteivät ne itsessään rajoita käyttöympäristöä. Tarkemmat kuvaukset järjestelmän vaatimuksista ja yhteensopivuustesteistä löytyvät myöhemmin tästä luvusta. Suurimmat yhteensopivuusongelmat muodostuvat selaimien puutteellisista tai epäyhteensopivista CSS3-standardin toteutuksista.

Yksi koko ei sovi kaikille. Järjestelmästä ei ole kirjoitettu erillisiä versioita erilaisia päätelaitteita silmälläpitäen palvelun esimerkkiluonteesta johtuen. Mikäli järjestelmä lanseerattaisiin tuotantokäyttöön, olisi siitä järkevää tehdä hybridisovellukset tunnetuimmille älypuhelinalustoille kuten Android ja iOS.

Kiinnitä huomiota turvallisuuteen. Palvelun tietolähteistä noutama informaatio liikkuu verkossa salaamattomana. Suunnitteluperiaatteiden ja yksityisyyden kannalta olisi turvallisempaa välittää tieto salatun SSL/TLS-yhteyden yli. On kuitenkin huomionarvoista, etteivät Facebook tai LinkedIn oletusarvoisesti käytä salattua yhteyttä[72]. Sen sijaan Twitter ja Google+ pakottavat käyttäjän selaimen salattuun tilaan. JSON-elementtien jäsentäminen tapahtuu joko hyödyntäen palveluiden omia rajapintoja tai niiden puuttuessa Prototypen omaa JSON-jäsentelijää. Näin pyritään välttämään manuaalisen jäsentämisen mahdollisesti aiheuttamia tietoturvaongelmia.

Testaa huolellisesti. Koska järjestelmä on vasta prototyyppi-asteella, ei kattavaa testausta ole suoritettu.

Harkitse laadun arviointiin kehitettyjen ohjelmistokehysten käyttöä. Rajapintojen, komponenttien, dokumentaation tai käyttöliittymän laatua ei ole arvioitu minkään olemassa olevan menetelmän avulla. Jonkinasteisen analyysin tekeminen olisi kuitenkin suotavaa, mikäli järjestelmä lanseerattaisiin julkiseen käyttöön. Erityisesti käyttöliittymän toimivuudesta ja käyttömukavuudesta olisi tärkeää saada tietoa sen muokattavasta luonteesta johtuen.

Ole visuaalisesti vakuuttava ja kiinnitä huomiota käytettävyyteen. Feednic:in käyttöliittymä itsessään on valtavirrasta poikkeava ja mahdollisesti mielenkiintoa herättävä. Käyttöliittymän sisältämä informaatio esitetään kuitenkin visuaalisesti ainoastaan tekstimuodossa, vaikka rajapinnat tarjoavatkin hyvät mahdollisuudet lisätä esimerkiksi aiheeseen liittyvää kuvamateriaalia tekstin joukkoon.

---

[72] Molemmat palvelut voidaan pakottaa käyttämään SSL/TLS- salausta muuttamalla osoiterivin URL muodosta *HTTP://* muotoon *HTTPS://.*

Rakenna vakaalle alustalle. Tutkielman tietolähteeksi valitut mediat ovat paikkansa vakiinnuttaneita ja laajasti tunnettuja. Niiden tarjoamat rajapinnat ovat vakaita, kohtalaisen helppokäyttöisiä sekä hyvin dokumentoituja. Todellisia koontisovelluksia tuotettaessa tilanne ei kuitenkaan aina ole yhtä suotuisa, ja kehittäjän kannattaakin kiinnittää huomiota edellä mainittuihin seikkoihin tietolähteitä valittaessa.

Salminen et al. (2010) mukaan tulisi suosia tietolähteitä, jotka ovat yhteisön ylläpitämiä. Tässä mielessä sosiaaliset mediat ovat ihanteellisia koontisovellusten kannalta, sillä käytännöllisesti katsoen kaikki niiden sisältämä informaatio on käyttäjien itsensä tuottamaa sivuston tarjotessa alustan informaation esittämiselle.

Ennakoi muutokset. Koska koontisovellusta ei tulisi sitoa liian tiukasti mihinkään tiettyyn palveluun tai tiedonvälitysformaattiin, on Feednic toteutettu komponenttipohjaisesti siten, ettei kokonaisuuden toiminta häiriinny, vaikka jokin käytetyistä palveluista lopettaisi toimintansa tai rajoittaisi tarjoamansa rajapinnan käyttöä. Vikasietoisuuden lisäksi tämä mahdollistaa järjestelmän helpon päivitettävyyden uusien tietolähteiden osalta. Kaikki valitut palvelut tarjoavat rajapinnan informaation noutamiseen, joten tiedon manuaalista louhintaa ei tarvita. Myös tämä lisää palvelun vikasietoisuutta.

Lähetä tiedoksiantoja ja tarjoa varasuunnitelma. Feednicissä ei ole toteutettu virhetilanteista kehittäjää tiedottavaa järjestelmää, kuten ei myöskään toissijaisia tietolähteitä käyttävää varmistusta. Lähtökohtaisesti toissijaisten tietolähteiden käyttö on tässä tapauksessa mahdotonta, sillä käyttäjät harvemmin lisäävät samaa informaatiota useaan palveluun (profiilitietoja lukuun ottamatta). Toisaalta taas yhden palvelun toimintahäiriö ei lamauta koko järjestelmää, vaan ainoastaan jättää kyseisen palvelun viestit välittämättä käyttäjälle.

Noudata standardeja ja suosituksia. Palvelun kaikki tietolähteet tarjoavat tiedon vakiintuneessa JSON-muodossa. Koska korvaavaa, yleisesti käytössä olevaa teknisesti parempaa välitysmuotoa ei ole saataville, voidaan olettaa välitysmuodon pysyvän samana vielä tulevaisuudessakin.

Kiinnitä huomiota lakiteknisiin seikkoihin. Valittujen medioiden lisensointiehdoissa ei alustavien tutkimusten mukaan ole Feednic:in kaltaisen sovelluksen kehittämistä estä-

viä ehtoja. Suunniteltaessa järjestelmää todelliseen tuotantokäyttöön tulisi ehtoihin kuitenkin paneutua huomattavasti tarkemmin[73] [74] [75] [76].

5.2 Tiedonvälitysformaatin valinta ja tietojen noutaminen

Tehdyn kokeen perusteella XML-formaatti kuluttaa 40 % enemmän siirtokapasiteettiä verrattuna JSON-formaattiin (Taulukko 3). Tuloksia tulkittaessa tulee huomioida Twitterin luonne mikroblogi-palveluna. Koska viestit ovat lyhyitä, on suurin osa siirretystä tiedosta metatietoa varsinaisen informaatiosisällön sijaan. Näin ollen siirtomuotojen erot todennäköisesti pienevät informaatiosisällön suhteellisen määrän kasvaessa metadatan osuuden vastaavasti laskiessa kokonaistiedon osuudesta. SNS-palveluiden informaatioalkiot ovat kuitenkin yleensä sisällöltään lyhyitä, joten tulosten voidaan olettaa olevan suuntaa antavia myös muiden palveluiden osalta.

Vaikka tiedostokoot ovat pieniä[77], tulee huomioida kokoerojen kumulatiivinen vaikutus. Yhteensä testiin valituilla 10 tilillä on yli 110 miljoonaa seuraajaa, joten kokonaiskaistan kulutus on varsin mittavaa.

Palveluiden rajapinnat tarjoavat testien ja palveluiden itsensä tarjoamien tilastojen perusteella luotettavan tavan tietojen noutamiseen. Tämä tulos oli odotettavissa, sillä palvelut pyrkivät houkuttelemaan mahdollisimman suuren kehittäjä- ja sitä myöten käyttäjäjoukon palvelun pariin, joten panostaminen toimivaan, luotettavaan ja helppokäyttöiseen infrastruktuuriin on luonnollinen vaihtoehto.

5.3 Selainvaatimukset ja yhteensopivuus

HTML-määritykseen kuulumattomien tekniikoiden käyttäminen järjestelmän toteutuksessa asettaa vaatimuksia loppukäyttäjän järjestelmän kokoonpanolle erityisesti WWW-

[73] http://developers.facebook.com/policy/
[74] http://developer.linkedin.com/documents/linkedin-apis-terms-use
[75] https://dev.twitter.com/terms/api-terms
[76] https://developers.google.com/+/terms
[77] XML avg. 24,87 kt ja JSON avg. 17,71 kt

selaimen JavaScript-tuen osalta. Toteutuksessa käytetty Prototype-kirjasto asettaa järjestelmän toiminnalle seuraavanlaiset selainvaatimukset[78]:

- Microsoft Internet Explorer for Windows: versio 6.0
- Mozilla Firefox: versio 1.5
- Apple Safari: versio 2.0.4
- Opera: versio 9.25
- Chrome: versio 1.0

Järjestelmän visuaalisen ilmeen määrittelyssä on käytetty joitakin CSS3-standardiin kuuluvia ominaisuuksia, kuten elementtien pyöristettyjä kulmia. Koska määritelmän standardointivaihe on kuitenkin vielä kesken, on erityisesti vanhempien selainversioiden tuki näille ominaisuuksille melko puutteellista. Ominaisuuksien tuen puuttuminen selaimesta ei kuitenkaan estä järjestelmän käyttöä, vaan ainoastaan sivuston visuaalinen ilme esitetään yksinkertaistettuna. Käytettyjä CSS3-määrityksiä tukevat ja täyden käyttökokemuksen antavat selaimet ovat[79]:

- Microsoft Internet Explorer for Windows: versio 9.0
- Mozilla Firefox: versio 1.0 (Prototype nostaa vaatimuksen versioon 1.5)
- Apple Safari: versio 3.0
- Opera: versio 10.5
- Chrome: versio 3.0

Selainvaatimuksissa on listattu vain yleisimmät työpöytäselaimet. Monet mobiiliselaimet perustuvat pc-selainten kanssa samoihin selainytimiin, joten suurin osa mobiiliselaimista on myös yhteensopivia järjestelmän kanssa. Näiden yhteensopivuutta on testattu taulukon 6 mukaisesti. Vaikka selaimien tuki käytetyille tekniikoille on yleisesti hyvä, rajoittaa erityisesti näytön pieni koko ja hiiren puute järjestelmän täysipainoista käyttöä.

---

[78] http://api.prototypejs.org/
[79] http://www.css3.info/preview/rounded-border/

Taulukko 6: Feednic:in yhteensopivuus mobiiliselaimissa

Selain	Päätelaite	Näytön tarkkuus	Toimivuus
Nokia Browser 7.3.1.26 (Symbian)	Nokia C5	240x320 (QVGA)	Toimivuus on teknisesti kunnossa lukuun ottamatta CSS3-muotoiluja. Näytön pieni koko, tarkkuus ja kosketusohjauksen puute tekevät järjestelmän käytöstä kuitenkin epäkäytännöllistä.
Android Browser (2.3.5)	Samsung Galaxy S 2	480x800 (WVGA)	Toimivuus on teknisesti kunnossa. Näytön tarkkuus tekee kokemuksesta epäkäytännöllisen.

Näiden vaatimusten valossa pidän teknologiavalintoja pääosin onnistuneina pois lukien CSS3-määrityksen hyödyntäminen. Sen käyttäminen käyttöliittymän ulkoasun määrittelyssä nostaa järjestelmän vaatimuksia huomattavasti teknologian uutuudesta johtuen. Näin ollen en suosittele määrityksen hyödyntämistä koontisovellusten toteutuksessa.

5.4 Järjestelmän kehittämismahdollisuudet

Modulaarisesta rakenteesta johtuen uusien tietolähteiden ja suodattimien lisääminen on toteutettavissa kohtalaisen helposti. Mikäli järjestelmää halutaan kehittää esimerkiksi yleisluontoisemmaksi *informaakeskittimeksi (Information Hub)*, voidaan siihen lisätä tuki RSS-syötteille ja erilaisille sähköpostiprotokollille. Yksi syötevirta voisi näin ollen tarjota uutisia (RSS-syötteitä tukevat uutissivustot), toinen ilmoittaa uusista viesteistä (sosiaalinen media ja sähköposti) ja kolmas koostaa ystävien tilapäivityksiä omalta lähialueelta (sosiaalinen media ja Google Maps).

Kuten aikaisemmin totesin, järjestelmä ei esitä tietoa kovinkaan intuitiivisesti. Pieniä ja kohtalaisen yksinkertaisesti lisättäviä visuaalisia parannuksia olisi esimerkiksi käyttäjien profiilikuvien noutaminen viestien yhteyteen.

Modulaarinen rakenne helpottaa myös laitekohtaisten versioiden kehittämistä, koska joissakin tapauksissa ainoastaan käyttöliittymäkerrokset joudutaan uudelleenkirjoittamaan. Mobiililaitteita (Android, iOS ja Mobile Windows) silmälläpitäen järjestelmästä tulisi kehittää joko erilliset hybridisovellukset jokaiselle alustalle tai vaihtoehtoisesti HTML5-pohjainen yleispätevä ratkaisu.

6 YHTEENVETO

Sosiaalisen median palveluiden tarjoamat rajapinnat ja niiden sisältämä käyttäjälähtöinen sisältö tarjoavat lähes rajattomat mahdollisuudet erilaisten koontisovellusten ja palvelukonseptien kehittämiselle, erityisesti muihin Internetin palveluihin yhdistettynä. Vaikka koontisovellukset ovat verkkosovellusten osajoukko, tulee niitä kehitettäessä ottaa jo suunnitteluvaiheessa huomioon muutamia niihin liittyviä erityispiirteitä.

Toteutusteknologioita valittaessa on tärkeää tiedostaa myös ei-teknisten muuttujien roolin merkitys onnistuneen lopputuloksen kannalta. Koska palvelinpuolen ohjelmointikielien tehokkuus ja ominaisuudet ovat monissa tapauksissa hyvin yhteneväisiä, nousevat kustannuskysymykset ja saatavilla olevien henkilöresurssien tietotaito tärkeiksi tekijöiksi. Vaikka alustat ovat pääsääntöisesti saatavilla ilmaiseksi Open Source -lisenssien alaisuudessa, on esimerkiksi maksullisen ASP.NET-alustan hyödyntäminen järkevää, mikäli saatavilla on jo valmiiksi on Microsoftin .NET-osaamista. Laajasti tuettujen ja vakiintuneiden alustojen kuten PHP ja Java käyttämistä puoltaa suuren ja aktiivisen kehittäjäyhteisön tarjoama laaja tuki.

Toisin kuin palvelinpuolella, valittaessa asiakaspuolen komentosarjakieltä tuotantokäyttöön vaihtoehdot rajoittuvat JavaScriptiin. Muitakin komentosarjakieliä voidaan hyödyntää esimerkiksi yrityksen sisäisiä järjestelmiä kehitettäessä selainpohjan ollessa yhtenäinen, mutta lähtökohtaisesti en löydä erikoistapauksia lukuun ottamatta syitä valikoida näitä JavaScriptin tilalle. On kuitenkin tosiasia, että JavaScript on ylivoimaisesti suosituin ja tuetuin komentosarjakieli verkkopalvelujen kehitykseen. Tätä puoltaa myös kielelle saatavilla olevien laadukkaiden kirjastojen ja alustojen määrä. Sovelluksen ulkonäköä määriteltäessä kannattaa edelleen tukeutua CSS2-määritykseen uudemman kolmannen version sijaan, sillä selainten tuki uudelle versiolle on vielä varsin puutteellista.

Arkkitehtuurivalintoja tehtäessä on muistettava verkkopohjaisten koontisovellusten olevan luonteeltaan Internet-pohjaisia hajautettuja järjestelmiä, joten näihin yleisesti liittyvät ongelmat kuten tiedonsiirron epävarmuus ja hitaus vaikuttavat myös koontisovelluksiin. Palvelukeskeiseen arkkitehtuurin perustuva REST-määritys näyttelee erityi-

sen tärkeää roolia tiedonvälityksessä sosiaalisen median palveluiden rajapintojen sille tarjoaman laajan tuen ansiosta.

Koska koontisovellukset ovat usein täysin riippuvaisia tietolähteistään on suunnittelussa kiinnitettävä myös huomiota tietolähteiden lisenssiehtoihin. Sovelluksen laadulla ei ole paljon väliä käytettävän tietolähteen päättäessä peruuttaa sen pääsyn tietoihin. Yksityisyyden ja tietoturvan roolia on vaikea olla korostamatta sovelluksissa, jotka käsittelevät käyttäjien henkikohtaisia tietoja. Teknologiat kuten OpenID ja OAuth ovat tehokkaita ja tuotantokäytössä toimiviksi havaittuja tietoturvaprotokollia, joiden käyttö on paitsi suositeltavaa myös joskus väittämätöntä palveluiden rajapinnoista johtuen. Suunnittelussa kannattaa myös huomioida palveluiden rajapintojen tarjoamien kutsujen rajallinen määrä.

Erityisesti suosituimmat sosiaalisen median palvelut tarjoavat kehittyneitä, tehokkaita ja hyvin dokumentoituja rajapintoja koontisovelluksien hyödynnettäväksi, mikä helpottaa huomattavasti sovellusten rakentamista. Kuten ohjelmistoprojekteissa yleensäkin, maksaa huolellisesti tehty tilanne- ja teknologiakartoitus itsensä takaisin myös sosiaalisen median koontisovelluksia kehitettäessä.

VIITTEET

Aghaee S., Pautasso C.: *Mashup Development with HTML5.* Mashups'10 (Joulukuu 2010)

BiancoP., Kotermanski R., Merson P.: *Evaluating a Service-Oriented Architecture.* (Syyskuu 2007)

Boyd D., Ellison N.: *Social Network Sites: Definition, History and Scholarship.* Journal of Computer-Mediated Communication (2008) 13: s. 210-230

Christensen J.: *Using RESTful Web-Services and Cloud Computing to Create Next Generation Mobile Applications.* OOPSLA 2009 (Lokakuu 2009)

Conti, G.: *Googling Considered Harmful*. Proceeding of NSPW 2006 (Syyskuu 2006) s. 67-76

Gruhl D., Nagarajan M., Pieper J, Robson C., Sheth A.: *Multimodal social intelligence in a real-time dashboard system*. The VLDB Journal (2010) 19: s. 825-848

Hammer-Lahav E., Recordon D., Hardt D.: *The OAuth 2.0 Authorization Protocol (draft-ietf-oauth-v2-21).* Network Working Group (Syyskuu 2011).

Hannah, A.: *A Primer to the OAuth Protocol*. Linux Journal #206 (Heinäkuu 2011)

Häsel M.: *OpenSocial, An Enabler for Social Applications on the Web.* Communications of the ACM (Tammikuu 2011) vol. 54, no. 1: s. 139-144

Kaplan M., Haenlein M.: *Users of the world, unite! The challenges and opportunities of Social Media.* Business Horizons (2010) vol 53, no. 1: s. 59-68

McCafferty D.: *Brave, New Social World.* Communications of the ACM (Heinäkuu 2011) vol. 54, no.7: s. 139-144

Mulligan G., Gračanin D.: *A Comparison of Soap and Rest implementations of a service based interaction independence middleware framework.* Proceedings of the 2009 Winter Simulation Conference (2009) s. 1423-1432

Ortmann S., Langendörfer P.: *The Impact of social networks on user privacy : What Social Networks Really Learn about their Users!*. Webist 2011 (2011) s. 21-26

Recordon D., Reed D.: OpenID 2.0: *A Platform for User-Centric Identity Management.* DIM'06 (Marraskuu 2009)

Salminen A., Nyrhinen F., Taivalsaari A.: *Developing Client-Side Mashups: Experiences, Guidelines and the Road Ahead.* MindTrek 2010 (Lokakuu 2010)

Tapiodor A., Fumero A., Salvachúe J.: *Extended Identity for Social Networks.* BlogTalk 2008/2009, LNCS 6045 (2010) s. 162-168

Trent S., Tatsubori M., Suzumura T., Tozawa A., Onodera T.: *Performance Comparison of PHP and JSP as Server-Side Scripting Languages.* Middleware 2008, LNCS 5346 (2008) s.164-182

Zang N., Rosson M., Nasser V.,: *Mashups: who? what? why?.* CHI EA '08 (Huhtikuu 2008)