Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Tuomas Rinne

Opinnäytetyö

Asynkroninen web-palvelu PHP:lla
Case: illtags.com

Työn ohjaaja
Paula Hietala
Työn teettäjä
Kaisla Innovations Oy, CTO Wesa Aapro
Tampere 12/2009

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma

Tekijä	Tuomas Rinne
Työn nimi	Asynkroninen web-palvelu PHP:lla Case: illtags.com
Sivumäärä	37
Valmistumisaika	12/2009
Työn ohjaaja	Paula Hietala
Työn tilaaja	Kaisla Innovations Oy, CTO Wesa Aapro

Tiivistelmä

Opinnäytetyö käsittelee asynkronisen web-sovelluksen toteuttamista perinteisiä web-tekniikoita hyödyntäen. Tarkemmin tarkastellaan miten sovelluksen data haetaan ja miten sitä siirretään sovelluksen sisällä käyttäen pääasiassa PHP- ja MySQL-tekniikoita.

Kohdesovelluksena on Kaisla Innovations Oy:n Illtags-palvelu, joka on toteutettu kesällä 2009. Tavoitteena on varmistaa, että sovelluksessa käytetyt ratkaisut olivat järkeviä. Lisäksi tavoitteena on samalla pohtia ideoita sovelluksen jatkokehitykseen.

Illtags on sosiaalista mediaa hyödyntävä palvelu, joka antaa käyttäjälle ajankohtaista tietoa tautien tartuntariskeistä tietyillä maantieteellisillä alueilla. Palvelu toimii siten, että käyttäjä lähettää sovellukselle oireita ollessaan kipeänä. Oireita lähetetään palveluun tunnisteina käyttäen hyväksi sosiaalisen median palveluita tai illtags.com-sivustoa. Palvelu kerää lähetetyt oireet ja tekee niistä paikallistettuja infektioriskidiagnooseja. Kaikki käyttäjät näkevät palvelun verkkosivustolta ajankohtaiset diagnoosit ja voivat näiden avulla vältellä paikkoja, joissa on suuri tartuntariski juuri sillä hetkellä.

Asynkronisen web-sovelluksen toteuttaminen perinteisillä www-tekniikoilla on mielenkiintoinen prosessi ja vaatii hieman soveltamista sovelluksen osissa, joita varten perinteisiä www-tekniikoita ei ole alun perin suunniteltu.

Avainsanat: PHP, asynkroninen, Illtags, web, palvelu

Tampere University Of Applied Sciences
Business Information Systems

Writer	Tuomas Rinne
Thesis	Asynchronous Web-Service Using PHP Case: illtags.com
Pages	37
Graduation time	12/2009
Thesis supervisor	Paula Hietala
Co-operating Company	Kaisla Innovations Ltd. CTO Wesa Aapro

Abstract

This thesis concentrates on developing an asynchronous web application utilizing traditional web techniques. The thesis focuses more precisely on data collection and data transfer proportions of the application that are developed by utilizing mainly PHP- and MySQL -techniques.

The target application is a web-application, Illtags from Kaisla Innovations Ltd, which has been developed during the summer of 2009. The aim of the commission is to make sure that the decisions made during the application development were sensible. In addition to that, the aim is also to gather some ideas for future development.

Illtags is a service that utilizes social media to inform the user about current infectious diseases in specific geographical areas. Users send their medical symptoms in forms of tags for the service while they are sick. The tags can be sent via social media services or via illtags.com-website. The service gathers the symptoms that the users have been sending and calculates the infection risk diagnoses based on geographical locations. All the users can see the current diagnoses in the website of the service and, based on those diagnoses, can avoid places that are infectious at the moment.

Developing an asynchronous web application using traditional web techniques is an interesting process and requires a little bit of creativity in such parts of the application that need to do things which traditional web techniques were not originally designed to do.

Keywords: PHP, asynchronous, Illtags, web, service

Sisällysluettelo

Keskeisiä käsitteitä

AJAX

Asynchronous JavaScript And XML eli AJAX pohjautuu JavaScriptin HTTP-pyyntöihin. AJAX käyttää hyväkseen tahdistamattomuutta hakiessaan dataa palvelimelta. Tämä mahdollistaa sen, että käyttäjän ei tarvitse ladata koko sivua uudelleen vaan sivun palasia voidaan ladata palvelimelta niin sanotusti taustalla.

Asynkroninen

Asynkronisen suora suomenkielinen käännös on tahdistamaton. Tässä yhteydessä tahdistamaton tarkoittaa sitä, että toiminto suoritetaan itsenäisesti riippumatta muista toiminnoista. Tällä tavoin saavutetaan se etu, että muiden toimintojen ei tarvitse odottaa yhden toiminnon valmistumista, jotta nämä voivat jatkaa omia toimintojaan. Tarkempi selvitys asynkronisesta toiminnasta löytyy luvusta 4 Asynkroninen web-sovellus.

Blogi

Blogi on Internet-sivu, jota yleensä ylläpitää yksittäinen henkilö. Ylläpitäjä julkaisee blogissaan säännöllisin väliajoin videoita, kuvia, kommentteja, artikkeleita ja muuta, joka yleensä häntä itseään koskettaa, tai johon hänellä on itsellään jotain sanottavaa. Blogia seuraavat käyttäjät voivat jättää kommenttejaan koskien ylläpitäjän julkaisuja tai jotka muulla tavoin liittyvät aiheeseen.

Facebook

Facebook on tämän hetken sosiaalisen verkostoitumisen lippulaiva. Alunperin korkeakouluopiskelijoille suunniteltu Internet-palvelu on nyt kaikkien yli 13-vuotiaiden käytettävissä. Käyttäjät pystyvät lisäämään toisia käyttäjiä ystävikseen, seurustelemaan toisten käyttäjien kanssa eri tavoin, kommentoimaan käyttäjien kuvia, kommentteja ynnä muuta. Facebook on monille käyttäjille yhteydenpitoväline vanhoihin ystäviin, sekä paikka jossa voi viettää aikaa.

JSON

JavaScript Object Notation eli JSON on kevyt formaatti datansiirtoon, jota ihmisen on helppo lukea ja kirjoittaa. Koneiden on myös helppo luoda ja parsia JSON:ia. JSON sisältää avain-arvo-pareja, jotka voivat olla myös sisäkkäisiä.

MySQL

MySQL-lyhenne tulee sanoista *My Structured Query Language*. MySQL on tietokantapalvelin, joka toimii useilla erilaisilla alustoilla. MySQL-tietokanta on avointa lähdekoodia ja sen voi ladata ilmaiseksi ja sitä voi käyttää vapaasti.

PHP

PHP: Hypertext Preprocessor, eli PHP, on komentosarjakieli, joka suoritetaan palvelimella. PHP tukee useita tietokantatyyppejä ja on avoimen lähdekoodin ohjelmisto, jonka voi ladata täysin ilmaiseksi ja sitä voi käyttää vapaasti.

RSS ja Atom syöte

Verkkosyötteet ovat Internet-sivujen sisällön koosteita. RSS ja Atom ovat verkkosyötteiden suosituimpia tekniikoita. Kun käyttäjä tilaa syötteen, hänen syötteenlukijaansa ilmestyy tieto sivuston päivityksistä. Syötteet sisältävät usein pienen osan päivitykseen liittyvästä tekstistä. Esimerkiksi uutissyötteessä on usein uutisen otsikon lisäksi lyhyt ingressin tapainen teksti ja linkki, josta käyttäjä voi siirtyä lukemaan koko uutisen verkkosivulta, mistä hän on syötteen tilannut. Syötteistä on tullut yksi lukuisista Web 2.0:n suosituista ilmiöistä.

Tag

Tag on englanninkielinen sana tunnisteelle. Tunnisteet ovat määritelmiä asioille tai esineille. Esimerkiksi maidon tunnisteita voisivat olla neste, valkoinen, juoma ja terveellinen. Tunnisteiden avulla voidaan helposti hakea koneellisesti esimerkiksi kaikki juomat, jotka ovat valkoisia. Arkikielessä tunnisteesta käytetään sanoja tagi tai tägi.

Twitter

Twitter on erittäin suosittu miniblogi- ja sosiaalisen verkostoitumisen palvelu, jonka avulla käyttäjä voi lähettää maksimissaan 140 merkkisen viestin profiiliinsa. Viestejä kutsutaan twiiteiksi (tweet). Viestit ovat julkisia ja näkyvät käyttäjän profiilissa, vaikka vierailijalla itsellään ei olisikaan Twitter-tiliä. Twitterin käyttäjät voivat ilmoittautua seuraamaan toista käyttäjää, jolloin heille välitetään suoraan tämän käyttäjän kirjoittamat viestit.

XML

Extensible Markup Language, on merkkauskieli, jota suositellaan käytettäväksi kaiken datan siirtämisessä verkossa.

1 Johdanto

Tavallinen web-sovellus toimii niin, että käyttäjä navigoi selaimellaan sivustolle, jonka seurauksena sovellus toteuttaa sille määrättyjä tehtäviä. Jokaisella sivuston sivulla on määrätty toteutettavaksi erilaisia toimintoja käyttäjän saapuessa sivulle. Käyttäjä siis käynnistää sovelluksen toimintoja navigoimalla sivustolla. Kun tarvitaan web-sovellus, joka pystyy toimimaan itsenäisesti ilman käyttäjän toimintaa tulee harvoin mieleen, että sovellus on mahdollista toteuttaa miltei kokonaan käyttäen perinteisiä www-tekniikoita.

Opinnäytetyönäni selvitän, miten voidaan luoda asynkroninen web-palvelu käyttäen pääasiallisesti perinteisiä www-tekniikoita, kuten PHP:ta ja MySQL-relaatiotietokantaa. Tarkempaan selvitykseen otan sen, miten dataa liikutellaan sovelluksen sisällä ja miten data kerätään, sekä miten se varastoidaan ja lähetetään eteenpäin. Sovellus on laaja ja se sisältää paljon erilaisia toiminnallisuuksia, mutta en keskity työssäni varsinaisiin datankäsittelyihin ja sovelluksen toimintaan kooditasolla. Tavoitteena on kertoa tämän opinnäytetyön toimeksiantajan, Kaisla Innovations Oy:n, illtags.com-palvelussa toteutetuista datansiirtotoiminnoista ja varmistua siitä, että valitut käytännön ratkaisut olivat järkeviä ja samalla pohtia mahdollisia ideoita sovelluksen jatkokehitykseen.

Toimeksiantajana toimii Helsingissä vuonna 2008 perustettu, terveydenhuoltoon liittyviä digitaalisia palveluita kehittävä yritys, Kaisla Innovations Oy. Kaisla Innovations Oy:n ensimmäinen tuote on sosiaalisia medioita, kuten Facebookia ja Twitteriä hyödyntävä palvelu Illtags. Olen toiminut ohjelmistokehittäjänä Illtags-palvelussa, johon opinnäytetyöni myöskin on kohdennettu.

Illtags on ensimmäinen palvelu, joka yhdistää sosiaalisen median ja lääketieteen. Se pyrkii täten palvelemaan käyttäjiä antamalla ajankohtaista tietoa tautien tartuntariskeistä tietyillä maantieteellisillä alueilla ja sitä kautta vähentämään tartuntoja. Palvelu toimii niin, että käyttäjä kohdistaa sosiaalisessa mediassa, kuten esimerkiksi Facebookissa, viestin Illtags-palvelulle. Viestissä käyttäjä kertoo omaan terveydentilaansa liittyviä tietoja tunnisteina (tag). Illtags kerää käyttäjien lähettämät tunnisteet ja tekee oireista diagnooseja. Toistaiseksi palvelu diagnosoi ainoastaan flunssaa ja sikainfluenssaa.

Käyttäjää pyydetään myös liittämään itsensä paikkoihin tai sosiaalisiin ryhmiin, joissa hän yleisesti liikkuu. Tämän tiedon avulla Illtags pystyy sijoittamaan oireita ja diagnooseja rajattuihin paikkoihin sekä ryhmiin ja näin ollen varoittamaan saman ryhmään tai paikkaan kuuluvia ihmisiä tartuntariskistä. Kaikki tieto on anonyymiä. Käyttäjät näkevät omiin ryhmiinsä kuuluvien toisten käyttäjien lähettämiä oire-tunnisteita, mutta eivät sitä kautta pysty yhdistämään niitä suoraan tiettyyn käyttäjään.

2 Web 2.0

Web 2.0 -termille on lukuisia määritelmiä, joista kaikki ovat yhtä paljon oikeassa ja väärässä. Kyseiselle termille ei kuitenkaan ole yhtään täysin tarkkaa määritelmää, jonka avulla asian voisi helposti ja selkeästi selittää. Termi tuli julkisuuteen ensimmäisen kerran vuonna 2004, kun O'Reilly Media, John Battelle ja yhtiö nimeltä CMP järjestivät *Web 2.0 Conference* -nimisen tapahtuman. Nykyään termi on hyväksytty yleisesti käyttöön ja sitä käytetään jopa markkinointikeinona, vaikka ei välttämättä oikein ymmärretä mitä se tarkoittaa. (O'Reilly 2008.)

Vuonna 2004 järjestetyn Web 2.0 -konferenssin tarkoitus oli auttaa ihmisiä ymmärtämään Internetin mahdollisuudet ja sen käyttäminen alustana. Vuonna 2005 Tim O'Reilly, O'Reilly Median perustaja, julkaisi artikkelin siitä, mikä on Web 2.0 *(What is Web 2.0?)*. Artikkelissaan O'Reilly väittää, että lopulta Web 2.0 on sitä, että osataan valjastaa verkostoituminen ja käyttäjien älykkyys ja näin ollen luoda sovelluksia, jotka vain paranevat käyttäjien lisääntyessä. Artikkelin julkaisun jälkeen termi Web 2.0 on ollut laajalti hyväksytty ja artikkelin konseptit ovat olleet useiden uusien liikeideoiden ja kehitysstrategioiden perustana. (O'Reilly 2008.)

Web 2.0 -termille on myös teknologisia määritelmiä. Sovelluksen pitää olla rakennettu käyttämällä tiettyjä tekniikoita ja ratkaisuja, jotta teknologisten määritelmien mukaan kyseessä olisi Web 2.0 -sovellus. Sovelluksen HTML-rakenteen ja CSS-tyylitiedostojen on oltava standardien mukaisia. Tämä mahdollistaa sen, että sovellus toimii oikein mahdollisimman monessa eri ympäristössä ja samalla sovelluksen käytettävyys paranee. Web 2.0 -sovelluksessa käytetään apuna JavaScript-kieltä ja sen mahdollistamaa Ajax tekniikkaa, joiden avulla käyttökokemuksesta saadaan rikkaampi, sekä toteutuksesta saadaan toimivampi. JavaScript suoritetaan käyttäjän tietokoneella, jolloin kaikkea toiminnallisuutta ei tarvitse tehdä palvelimella ja näin ollen saadaan myös tasattua tehtäviä palvelimen ja käyttäjän tietokoneen välille. Web 2.0 -sovelluksissa on yleensä myös mahdollisuus tilata syötteitä, kuten RSS tai Atom, joiden avulla käyttäjä saa helposti päivitystiedot sovelluksesta, mikäli hän on sen tilannut. Blogien, keskustelufoorumien ja muiden interaktiivisten elementtien ja sosiaalisen verkostoitumisen välineiden sisällyttäminen ja hyödyntäminen sovelluksessa on myös osa, joka sisältyy Web 2.0:n teknisiin määritelmiin. (Zervaas 2008, 2.)

Kuva 1: Markus Angermeierin vuonna 2005 luoma Web 2.0 -teeman tunnistepilvi, joka itsessään on yksi Web 2.0:n ilmiöistä (Wikipedia: Web 2.0, n.d.)

Vaikka Web 2.0 on yleisesti käytössä kuvailtaessa tietyn tyyppisiä Web-sovelluksia, ei se tarkoita, että kaikki olisivat asiasta yhtä mieltä. Internetin keksijänä pidetty Sir Tim Berners-Lee on johtava Web 2.0 -konseptin kritisoija. Hänen mukaansa Internetin alkuperäinen tarkoitus on ollut ihmisten yhdistäminen, joten Web 2.0:ssa ei ole mitään uutta, eikä kukaan hänen mukaansa oikein tiedä, mitä koko Web 2.0 edes tarkoittaa. Berners-Leen mukaan sosiaalinen verkostoituminen on aina ollut Internetin ydin. (Hosch n.d.)

Riippumatta siitä, onko Web 2.0 olemassa vai ei, sisältävät monet sen määritelmistä asioita, jotka toteutuvat Illtags-palvelussa. Tarkastelunäkökulmallakaan ei ole merkitystä, sillä niin tekniset kuin teoreettisetkin määritelmät sopivat palveluun.

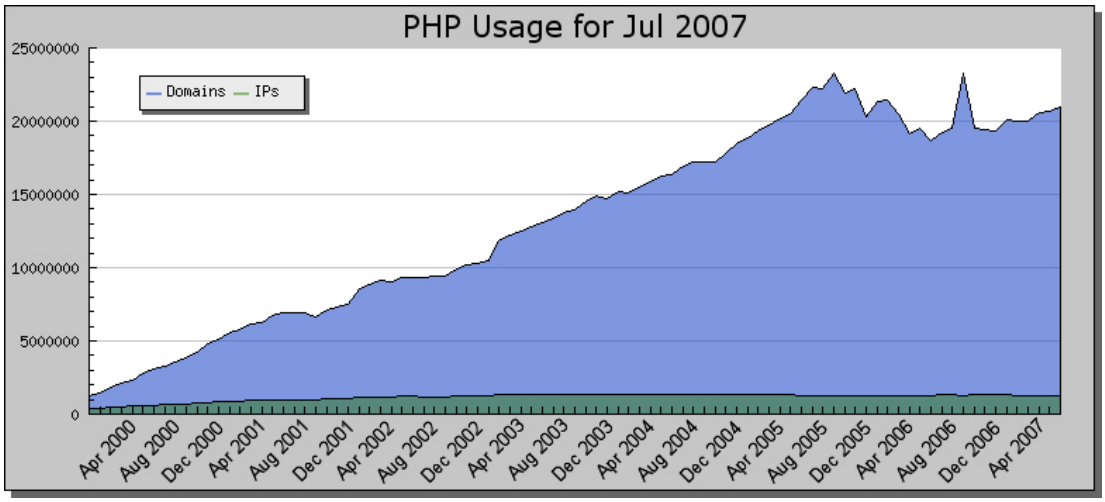
Tarkasteltaessa Illtags-palvelua ja sen internetsivua teknisestä näkökulmasta, voidaan sanoa, että sovellus täyttää teknisten vaatimusten osalta Web 2.0 -määritelmän erittäin hyvin. Illtagsin internetsivu on toteutettu käyttäen standardien mukaista HTML-kieltä ja standardin mukaisia CSS-tyylitiedostoja. Käyttöliittymässä on käytetty runsaasti JavaScriptiä ja Ajax-tekniikkaa, jotta palvelusta tulisi toimivampi ja mukavamman näköinen. Interaktiivisina elementteinä palvelussa on tunnisteiden keräys ja niiden käsittely. Myös tyypillinen Web 2.0 -ilmiö, tunnistepilvet (tagcloud) (Kuva 1), on tärkeä osa illtags.com-sivuston käyttöliittymää. Tärkein Illtags-palvelun osa on kuitenkin tunnisteiden kerääminen käyttäjiltä, ja tämä tapahtuu hyödyntäen sosiaalisia medioita, joista tällä hetkellä käytössä on Facebook ja Twitter.

Kun tarkastellaan palvelua muussa kuin teknisessä mielessä, huomataan myös O'Reillyn artikkelin konseptien sopivan Illtags-palveluun. Käyttäjät ovat Illtagsin toiminnan kannalta välttämättömiä. Illtags-palvelu ei toimi ilman käyttäjiä, koska heidän lähettämänsä oireet ovat se, johon koko palvelu perustuu. Myös vähäinen käyttäjämäärä on ongelma Illtags-palvelussa, koska tällöin ei voida suorittaa palvelun analyysejä tai ne ovat epätarkkoja. Jos käyttäjiä on vähän, myös heidän saamansa hyöty palvelusta on mitätön. Illtagsiin sopii siis suoraan myös O'Reillyn toteamus siitä, että Web 2.0 -sovellus paranee sitä mukaa, mitä enemmän sovelluksella on käyttäjiä.

3 Miksi Illtags valitsi PHP:n?

Internetin ohjelmointiaiheisia keskustelupalstoja lukiessa huomaa pian, että kehittäjillä on monia mielipiteitä eri ohjelmointikielistä. Java-foorumilla kehittäjät luonnollisesti puolustavat Javaa ja PHP-foorumien kehittäjät PHP:ta. Usein henkilökohtaiset mieltymykset johtavat siihen, että kehittäjän mielestä jokin ohjelmointikieli on parempi kuin toinen.

Netcraftin teettämän tutkimuksen mukaan PHP on vuonna 2005 ylittänyt 20 miljoonan domainin rajan (Kuva 2) ja yli miljoonan web-palvelimen rajan (PHP: PHP Usage Stats n.d.). Luvuista voidaan päätellä, että kyseessä on todella suosittu alusta web-ohjelmointiin. PHP:n suosion mukana on syntynyt erittäin suuri web-kehittäjien yhteisö, joka jakaa auliisti tietämystään erilaisilla aiheeseen liittyvillä keskustelufoorumeilla. Tämä yhteisö mahdollistaa ongelmatilanteissa nopean teknisen tuen ja ratkaisun löytyminen voi olla usein huomattavasti nopeampaa, kuin se olisi muiden tekniikoiden kohdalla. (Reiss 2009.)



Kuva 2: PHP:n käyttömäärä sivustoilla ja palvelimilla (PHP: PHP Usage Stats n.d.)

Toimiva ja erittäin laaja funktiokirjasto on edesauttanut PHP:n suosiota web-kehityksessä. Kirjastosta on olemassa myös selkeä dokumentaatio, johon jokaisella kehittäjällä on helppo pääsy. Riittää, että navigoi omalla selaimellaan php.net-sivustolle. PHP:lle on myös saatavilla runsaasti valmiita ohjelmia ja eri kehittäjien luomia koodin osia ja funktioita, jotka on jo testattu eri sovelluksissa ja todettu toimiviksi. Vielä kun kyseiset valmiit koodit ovat vapaasti käytettävissä, on suorastaan tyhmää olla hyödyntämättä näinkin suurta apua uusia palveluita kehitettäessä. Raskas kehitystyö on osittain jo tehty ja siksi pystytään nopeammin kehittämään uusia web-palveluita keskittymällä vain oman palvelun kannalta tärkeisiin seikkoihin. (Reiss 2009).

PHP ei pakota olio-ohjelmointiin. Tämä on samanaikaisesti PHP:n hyvä ja huono puoli. Hyvänä puolena voidaan pitää PHP-ohjelmoinnin oppimisen helppoutta. Koska koodin ei tarvitse olla hyvän ohjelmointitavan mukaisesti muodostettua, on helppo päästä alkuun ja saada nopeasti

ensimmäinen toimiva koodi. Oppimisen helppouden lisäksi tätä ominaisuutta voidaan hyödyntää projekteissa erinäisillä tavoilla. Ketterää kehitystä hyödyntävissä projekteissa, kuten Illtags-palvelussa, tämä antaa mahdollisuuden saada nopeasti toimiva ohjelman osa, jota voidaan myöhemmin tarvittaessa parantaa. (Reiss 2009.) Huonona puolena on koodin ylläpidettävyys ja laajennettavuus, mikäli sovellus vaati paljon monimutkaista koodia. Koodia ei voida myöskään käyttää uudelleen, vaan saman toiminnallisuuden saaminen muualle sovellukseen joudutaan toteuttamaan kopioimalla toiminnallisuuden koodi. Jos sama toiminnallisuus joudutaan kopioimaan useaan paikkaan ja jälkikäteen huomataan, että toiminnallisuutta pitää muuttaa, pitää muutos tehdä jokaiseen kopioituun osaan erikseen. Tällöin olio-ohjelmoinnilla toteutetussa ratkaisussa on huomattava etu, sillä muutoksen voi tehdä yhteen paikkaan ja muutos tulee voimaan jokaisessa paikassa, jossa oliota kutsutaan. Joskus voi olla tapauskohtaisia tilanteita, jossa yksinkertainen ja huonon ohjelmointitavan mukainen lyhyt komentosarja saattaa toimia sovelluksessa paremmin kuin raskas ja oikeaoppisesti muodostettu, olio-ohjelmointia hyödyntävä ratkaisu.

Pienen oppimiskynnyksen ansiosta PHP-ohjelmointia osaavia ihmisiä on erittäin paljon. Tämä myös edesauttaa pienten yritysten, kuten Kaisla Innovations Oy:n, päätöstä käyttää PHP:ta sovelluksissaan. Koska osaajia on niin paljon, ei osaavia ohjelmoijia tarvitse etsiä pitkiä aikoja ja näin tuhlata kallista aikaa, jonka voisi käyttää ennemmin sovelluksen tai palvelun kehittämiseen.

Vuonna 2004 julkaistun PHP:n versio 5:n parannuksista eniten huomiota on kerännyt sen uudelleen rakennettu oliomalli, joka mahdollistaa tehokkaamman olio-ohjelmointi -tyylisen kehityksen (PHP: New Object Model 2009). Aiemmin olio-ohjelmoinnista PHP:n kanssa oli miltei enemmän haittaa kuin hyötyä, sillä oliomalli oli niin huonosti toteutettu (Hudson 2005, 128). PHP:ta on kritisoitu usein sen huonosta laajennettavuudesta, mutta PHP:n versio 5:n johdosta PHP-koodista saadaan tehtyä helpommin laajennettavaa. Tämän ansiosta pystytään rakentamaan laajempia sovelluksia siten, että koodi pysyy ymmärrettävämpänä ja sitä on helpompi hallita.

PHP:n suurin etu muihin ohjelmointikieliin verrattuna on kehittämisen nopeus ja vaivattomuus. PHP:ssa kehittäjä voi kirjoittaa koodin, päivittää sivun ja kaikki muutokset tulevat voimaan. Jos verrataan PHP:ta vaikka Perliin, huomataan molemmissa omat hyvät puolensa. Kutsuttaessa Perl-komentosarjaa ensimmäisen kerran koko komentosarja tallennetaan muistiin. Tämän jälkeen joka kerta, kun samaa komentosarjaa kutsutaan uudelleen, se voidaan ladata muistista. Tämä on suuri etu PHP:hen nähden, sillä muistista ladattu komentosarja on nopeampi kuin joka kerta uudelleen käännetty ja suoritettu PHP-komentosarja. Kehitysvaiheessa Perl kuitenkin on hitaampi, sillä aina kun ohjelmakoodia muutetaan, joudutaan palvelin käynnistämään uudelleen. Vasta tämän jälkeen uudistunut komentosarja voidaan kääntää ja tallentaa uudelleen muistiin. PHP:lla tätä ongelmaa ei ole, vaan ohjelmakoodi käännetään ajon aikana eikä mitään tallenneta muistiin, ja näin kehittäjä voi tehdä muutoksen, päivittää sivun ja voi nähdä tuloksen saman tien. Vaikka Perlin koodi on suoritusvaiheessa nopeampaa, sen kehittäminen ei ole yhtä vaivatonta kuin PHP:lla. (Reiss 2009.)

Vaikka PHP ei tallenna tietoja muistiin ja komentosarja käännetään ja suoritetaan joka kerta, kun sivu ladataan, ei PHP silti ole hidas. Itse asiassa se on yksi nopeimmista tämänhetkisistä komentosarjakielistä ja haastaa nopeudessaan muun muassa Perlin ja ASP:n. PHP:n viimeisissä versioissa on suoritettu paljon optimointia, jotta saavutettaisiin paras mahdollinen nopeus. PHP:hen voidaan myös yhdistää aputyökaluja, jotka tallentavat usein tarvittavia tietoja välimuistiin, kuten esimerkiksi *Memcached.* Tällaisen yhdistelmän avulla PHP:n suorituskyky yleensä vähintäänkin kaksinkertaistuu, mutta joskus saatetaan saavuttaa moninkertaisia suoritusnopeuksia hyödyntämällä tätä tekniikkaa. (Hudson 2005, 4.)

Vaadittaessa merkittävää tehonlisäystä, on syytä harkita käännettyä kieltä, tulkattavan kielen, kuten PHP:n sijaan. Käännetyn kielen lähdekoodi on käännetty valmiiksi konekielelle, jolloin sen suoritusnopeus on ehdottomasti nopeampi kuin tulkattavan kielen, jossa lähdekoodi käännetään ja suoritetaan ajon aikana. Kehittäminen ei ole tällöin yhtä nopeaa, kuin esimerkiksi PHP-kehittäminen, mutta sovellus itsessään toimii tehokkaammin. (Compiled versus interpreted languages n.d.)

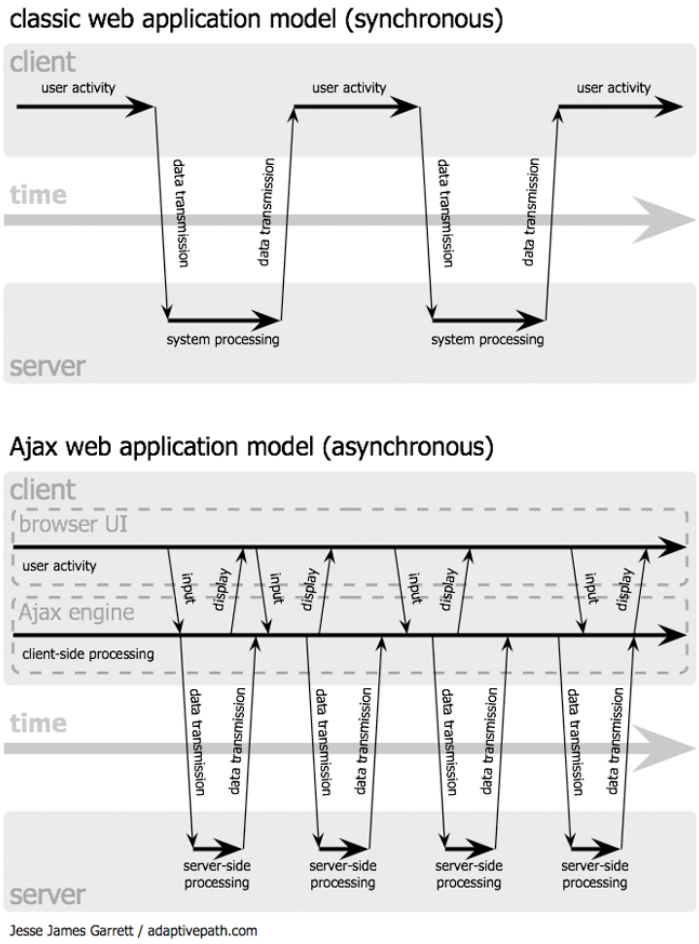
Kaisla Innovations Oy:n Illtags-palveluun valittiin ohjelmointikieleksi PHP yllä esitettyjen seikkojen perusteella. Lisäksi Kaisla Innovations Oy on palkannut ohjelmistokehitysryhmänsä kiinnittäen enemmän huomiota ryhmän yhdessä toimimiseen kuin yksilötason osaamiseen. Hyvin yhdessä toimivan kehitysryhmän jäsenet osasivat jo valmiiksi PHP-ohjelmointia kiitettävästi, mikä myös helpotti huomattavasti päätöksen tekoa. Nopealla kehitysaikataululla ei luonnollisesti ollut varaa alkaa opetella alusta uusia ohjelmointikieliä. Ottaen huomioon yrityksen suhteellisen pienen budjetin, PHP sopii yritykselle mainiosti, sillä tekniikka ja kehittäminen ovat halpoja. Lisäksi PHP on suunniteltu web-sovelluksia varten ja sillä kehittäminen on erittäin nopeaa. Illtags.com on web-sovellus, joten PHP on luonnollinen valinta. Osa tämän hetken suurimmista web-sovelluksista, kuten esimerkiksi Facebook ja Wikipedia, käyttävät PHP:ta. (Gaarsmand n.d.) Tästä voidaan päätellä, että huolellisesti ohjelmoidun PHP-sovelluksen suorituskyky on hyvä ja riittää usein pitkälle.

4 Asynkroninen web-sovellus

Asynkronisen web-sovelluksen toiminnan ymmärtämiseksi, täytyy ensin tietää mitä tarkoittaa asynkroninen. Sanakirjasta katsottu käännös englannin kielen sanalle *asynchronous* on tahdistamaton tai asynkroninen. Tämä ei välttämättä paljon selvennä mitä asynkroninen tarkoittaa puhuttaessa Internet-sovelluksesta. Ajax-tekniikan tutkiminen helpottaa asynkronisen toiminnan ymmärtämistä.

Vuonna 2005 Jesse James Garret kirjoitti artikkelin, josta termi Ajax on lähtenyt liikkeelle. Kyseessä ei ollut uusi tekniikka, vaan yhdistelmä vanhoista perinteisistä web-tekniikoista, joiden avulla verkkosivustojen käyttökokemuksesta saataisiin miellyttävämpää. (Negrino, Smith 2007.) Ajax ohjelman tarkoituksena on poistaa Internet-sivuston käyttöön liittyvä töksähtelevä luonne. Tavallisessa web-sovelluksessa käyttäjä tekee sivustolla jonkin toiminnon ja selain lähettää pyynnön palvelimelle, jossa toiminto suoritetaan käyttäjän odottaessa. Kun toiminto on suoritettu, lähetetään uusi tieto takaisin selaimelle, jolloin sivu vaihtuu ja käyttäjä näkee toiminnon suorituksen tuloksen. Ajaxia hyödyntävässä sovelluksessa pyritään poistamaan käyttäjältä turha odottelu. Käyttäjän tehdessä halutun toiminnon, voidaan palvelimelle lähettää pyyntö, mutta sen sijaan, että käyttäjä joutuisi odottamaan toiminnon valmistumista ja uuden sivun latautumista, voidaan toiminto suorittaa ikään kuin taustalla. Käyttäjä voi työskennellä web-sovelluksessa koko ajan, ja kun jokin toiminto on suoritettu, päivittyvät sen tulokset sivulle ilman uuden sivun lataamista. Tällöin web-sovellus toimii samaan tapaan, kuin mihin on totuttu normaalissa työpöytäsovelluksessa. (Garret 2005.)

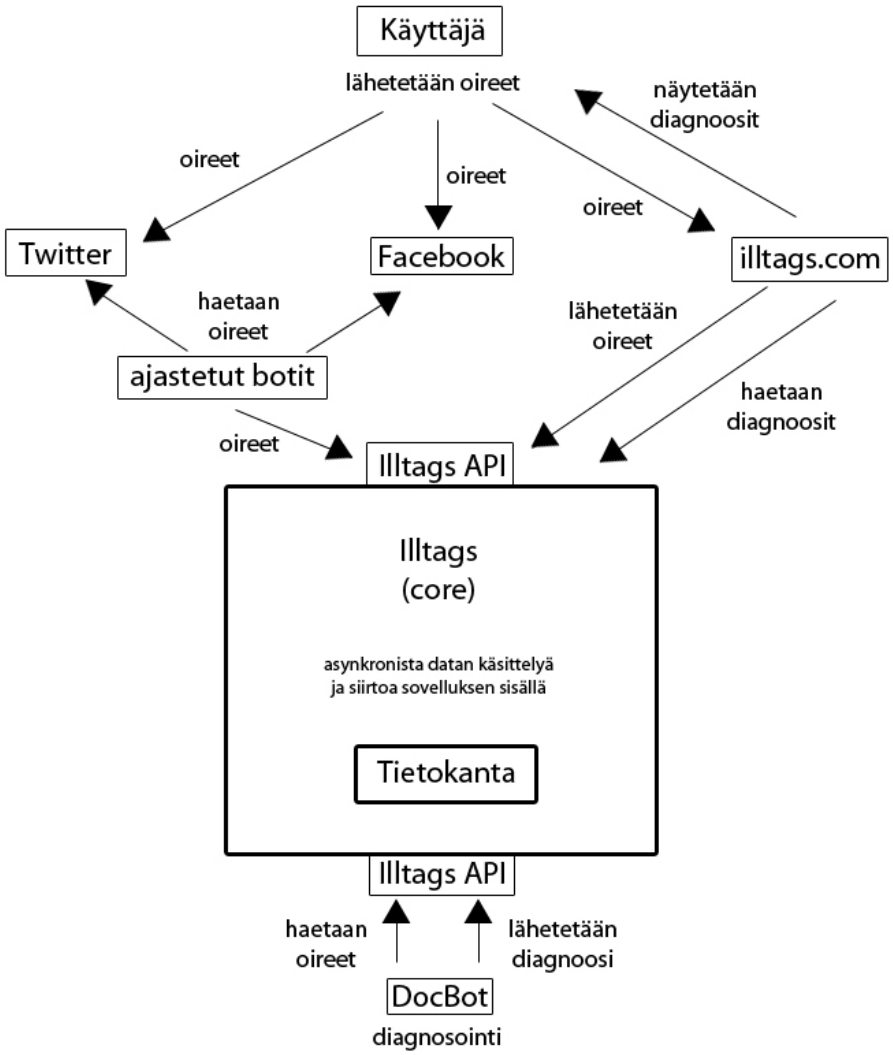
Alla olevien kaavioiden (Kuva 3) avulla voidaan selventää synkronisen ja asynkronisen toiminnan eroja. Ylemmässä kaaviossa on kuvattu synkroniset toiminnat käyttäjän ja palvelimen välillä suhteessa aikaan. Aina kun tietoa siirretään käyttäjän ja palvelimen välillä, aikaa kuvaava nuoli katkeaa ja siirtyy joko palvelimen tai käyttäjän puolelle. Toisin sanoen aina kun käyttäjän selain tekee pyynnön palvelimelle, joutuu se odottamaan kunnes palvelin palauttaa vastauksen. Asynkronisessa toiminnassa taas käyttäjän puolen nuolet eivät katkea, mikä tarkoittaa sitä, että selain ei joudu odottamaan tietoa palvelimelta. Näin käyttäjälle ei aiheudu turhaa odottelua sivun uudelleenlataamisen takia.



Kuva 3: Jesse James Garrettin kuvaus asynkronisesta toiminnasta Ajax-tekniikassa (Garret 2005)

Illtags-palvelussa sovellettiin asynkronista tekniikkaa PHP-ohjelmointiin. Yleensä PHP-sovellus toimii niin, että käyttäjän pyynnön jälkeen suoritetaan haluttu PHP-komentosarja ja käyttäjä odottaa, kunnes palvelin on suorittanut tehtävän ja palauttaa vastauksen, jolloin käyttäjälle näytetään uusi sivu, jossa muutokset mahdollisesti jo näkyvät. Näin ei voida kuitenkaan toimia Illtagsissa, sillä palvelun kasvaessa diagnosoitavien oireiden määrä on valtava ja tarkempien diagnoosien tekeminen voi kestää kauan. Lisäksi samoista oireista pystytään mahdollisesti diagnosoimaan useita eri tauteja, joten diagnooseja on myös analysoitava. Kaiken tämän tekeminen käyttäjän odottaessa ei ole mahdollista. Illtagsissa sovellettiin diagnoosien tekemisessä jonotusta, jossa uuden oireen lähetys lisää sen lähettäneen käyttäjän ja käyttäjään liittyvän kontekstin diagnoosijonoon. Diagnosointi tapahtuu siis jatkuvasti palvelimella, josta tulokset välitetään käyttäjille nähtäväksi niiden valmistuttua.

Seuraavalla sivulla on kaavio sovelluksen asynkronisesta toiminnasta yleisellä tasolla (Kuva 4). Kaavio on pelkistetty, eikä sisällä kaikkea Illtagsin toiminnallisuutta. Nuolet kuvaavat sovelluksen toimintaa siten, että nuolen juuressa oleva sovelluksen osa aloittaa toiminnan. Nuolen vieressä oleva teksti kertoo, minkä tyylistä dataa kyseinen toiminto siirtää tai mitä kyseinen toiminto suorittaa.



Kuva 4: Pelkistetty kuvaus sovelluksen toiminnasta

4.1 Datan kerääminen

Datan kerääminen Internetistä on monimutkainen prosessi, johon hakukoneita valmistavilla yrityksillä on omat keinonsa. Hakukonejätti Google on perustettu vuonna 1998 ja on siitä asti kerännyt dataa Internetistä erittäin menestyksekkäästi. Google on kehittänyt omia tekniikoitaan, joiden avulla dataa voidaan kerätä ja hakuja suorittaa erittäin nopeasti. Esimerkiksi Googlen PageRank-teknologialla voidaan arvioida sivun merkitystä tehtyyn hakusanaan nähden, ja näin palauttaa käyttäjän hakuun parhaiten täsmäävät sivustot järjestyksessä. (Google Yritystiedot 2009.)

Aloittelevan yrityksen tuotteella, Illtagsilla, ei ole resursseja samankaltaiseen massiiviseen tiedonkeräämiseen, vaan tiedonkerääminen on tehtävä jollakin muulla tavalla. Viimevuosien sosiaalisen median suosion kasvu, sekä yrityksen perustajien tekemät henkilökohtaiset havainnot siitä, miten ihmiset käyttävät sosiaalista mediaa, johtivat siihen päätökseen, että Illtagsin pää-

asiallinen datan kerääminen tapahtuu sosiaalisten medioiden kautta. Käyttäessään sosiaalisia yhteisöpalveluita, Kaisla Innovations Oy:n perustajat huomasivat ihmisten kertovan usein heikosta voinnistaan ystävilleen yhteisöpalveluiden välityksellä. Sen sijaan, että ihmiset valittavat huonoa vointiaan vain ystäville, voisivat he samalla jakaa tiedon myös Illtagsille, joka pystyisi hyödyntämään tätä diagnooseissaan.

4.1.1 Twitter

Twitter julkaisi vuonna 2006 API:n, eli ohjelmointirajapinnan, jonka avulla sovelluskehittäjät voivat käyttää Twitterin palveluita kehittämissään sovelluksissa. Aluksi ohjelmointirajapinta tarjosi vain 4 yksinkertaista toiminnallisuutta, mutta myöhemmin toiminnallisuuksien määrä on kasvanut ja rajapinnan yli voi nyt jo tehdä miltei kaikkea, jota Twitterin omalla sivulla pystyy tekemään. (Makice 2009.) Tätä ohjelmointirajapintaa käyttämällä, myös Illtags pystyy keräämään tarvitsemaansa dataa Twitteristä.

Twitteristä ei ole julkisesti saatavilla virallisia käyttötilastoja, mutta kolmannen osapuolen kehittäjät ovat tehneet sovelluksia, joilla saadaan epävirallisia statistiikkoja kerättyä Twitteristä. Yksi tilastoja keräävistä sivustoista on *tweetrush.com*. Kerättyjen tilastojen mukaan Twitteriin saattaa tulla päivittäin yli miljoona uutta viestiä. (Makice 2009.) Twitterin ohjelmointirajapinnassa on myös rajoitettu tehtyjen pyyntöjen määrää, mikä asettaa myös rajoituksia haettavan datan määrälle. Yhdellä haulla pystytään hakemaan maksimissaan 200 tilapäivitystä, ja oletuksena voidaan tehdä 150 pyyntöä yhden tunnin aikana. Tarvittaessa Twitteriltä voidaan anoa lisää pyyntöjä, jolloin määrä kasvaa 20000 pyyntöön tunnissa. (Twitter API Wiki / Rate limiting 2009.) Valtavasta viestimäärästä Illtagsin kannalta oleellista on vain pieni osa: viestit jotka sisältävät sairauteen tai oireisiin viittaavaa tietoa. Ei siis ole järkevää kerätä kaikkea Twitterissä olevaa dataa, vaan vain Illtagsin kannalta olennaiset. Saadakseen kerättyä dataa tasaisesti, on Twitteriin lähtevät pyynnöt ajastettava tehtäviksi tasaisin väliajoin. Tällöin saadaan myös estettyä se, ettei pyyntöjä tehdä liikaa tunnin aikana ja samalla vältytään turhalta ylimääräiseltä liikenteeltä sovelluksessa.

Twitterissä käyttäjä voi kohdentaa viestin toiselle käyttäjälle kirjoittamalla viestiin *@käyttäjänimi.* Tällainen kohdennettu viesti voidaan helposti hakea käyttämällä Twitterin ohjelmointirajapintaa. Jotta viestien kohdentaminen on mahdollista, on myös Illtagsilla oltava oma käyttäjätili Twitterissä, johon käyttäjät voivat kohdentaa viestinsä. Lisäksi pitäisi koneellisesti pystyä päättelemään, mitkä osat viestistä ovat oireita ja mitä oireita ne ovat. Oireet voitaisiin erottaa viestistä, mikäli olisi käytettävissä kunnollinen luonnollisen kielen tulkki, jota Illtags voisi hyödyntää. Näin ei kuitenkaan ole, eikä pienellä aloittavalla yrityksellä ole resursseja alkaa tällaista rakentamaan. Ongelmaan ratkaisu on *hashtagit*, jotka ovat syntyneet Twitter yhteisössä ja ovat yleinen tapa kategorisoida viestejä Twitterissä. Hashtagi on aivan kuin tavallinen tunniste, jolla voidaan kuvailla tiettyä asiaa, mutta sen edessä on #-merkki (hash/pound). (Twitter Support :: What are

hashtags (the ”#” symbol)? 2009.) Illtags hyödyntää tätä merkintätapaa Twitterin kautta lähetettävissä oireissa siten, että jokainen oire merkitään #-merkillä. Käyttäjä voi siis lähettää Illtagsille viestin seuraavalla tavalla: *@illtags #nuha #kuume*. Tällöin luonnollisen kielen tulkkia ei tarvita ja kaikki Illtagsille kohdistetut hashtagit kerätään talteen ja analysoidaan Illtagsissa. Ongelmana tässä on se, ettei vähemmän Twitteriä käyttävät henkilöt välttämättä tiedä hashtageista, eikä niiden käyttö ole tällöin heille luonnollista. Paras mahdollinen tapa kerätä oireita olisi siis tulkita myös normaalia tekstiä.

Tiedon kerääminen Twitteristä ei ole Illtagsissa toteutettu kovinkaan monimutkaisesti. Tätä tehtävää suorittaa yksinkertainen botti, joka toimii Illtags sovelluksen päällä, ikään kuin välikätenä Twitteriin. Tämä botti on ajastettu tekemään Twitteriin pyyntöjä, joiden välityksellä Illtagsille suunnatut viestit haetaan. Kun haku on suoritettu ja mikäli haku löysi uusia viestejä, viestit parsitaan ja siirretään Illtagsin tietokantaan odottamaan jatkokäsittelyä.

Normaali työpöytäsovellus, joka on toteutettu esimerkiksi C-ohjelmointikielellä, voi käynnissä ollessaan hakea Twitteristä uusia viestejä jatkuvasti ajastuksen avulla. Illtags on toteutettu PHP:lla, joka ei ole jatkuvasti käynnissä palvelimella. Tämän vuoksi Twitteriin lähetettävien pyyntöjen ajastus on toteutettava hieman eri tavalla, kuin miten se voidaan toteuttaa työpöytäsovelluksessa. Tästä ratkaisusta lisää luvussa 5.

```
1 public static function getTweets ($latestId, $count) {
2   // Initialize connection
3   $ch = curl_init();
4   // Define URL
5   $curlURL = 'http://twitter.com/statuses/mentions.xml?count=' . $count . '&since_id=' . $latestId;
6   // Define options
7   $options = array(
8                   CURLOPT_URL => $curlURL,
9                   CURLOPT_RETURNTRANSFER => 1,
10                  CURLOPT_USERPWD => TWITTER_USERNAME . ':' . TWITTER_PASSWORD,
11                  CURLOPT_HEADER => true
12                  );
13  curl_setopt_array($ch, $options);
14  // Fetch
15  $result = curl_exec($ch);
16  $headers = mb_substr($result, 0, curl_getinfo($ch, CURLINFO_HEADER_SIZE));
17  $result = mb_substr($result, curl_getinfo($ch, CURLINFO_HEADER_SIZE));
18
19  // Close curl session
20  curl_close($ch);
21  return $result;
22 }
```

Esimerkki 1: Sovelluksen osa, joka hakee uusia viestejä Twitteristä

Twitterin ohjelmointirajapinta on toteutettu RESTful-tekniikkaa käyttäen. Tämä tarkoittaa lyhyesti sanottuna sitä, että pyynnöt tehdään HTTP-pyyntöinä ja pyyntö lähetetään määriteltyyn URL-osoitteeseen, jonka avulla tiedetään mitä tietoa haetaan. Havainnollistetaan asia pilkkomalla edellisen esimerkin (Esimerkki 1) URL-osoite ja muokkaamalla sitä hieman luettavammaksi: http://twitter.com/statuses/mentions.xml?count=200&since_id=195672. Alussa oleva

http://twitter.com, tulee kaikkiin ohjelmointirajapintaa käyttäviin pyyntöihin. Kohta /statuses/ tarkoittaa sitä, että pyydetään tilapäivityksiä (status update), eli Twitter viestejä. Tämän jälkeen tulee halutut vaihtoehtoiset tiedot, jotka tässä tapauksessa ovat mentions.xml, count=200 ja since_id=195672. Ensimmäinen parametri mentions.xml kertoo, että pyydetään XML-muodossa niitä viestejä, jotka on kohdistettu tietylle käyttäjälle, tässä tapauksessa Illtagsille. Twitterin ohjelmointirajapinta tarjoaa mahdollisuuden pyytää nämä myös JSON-, RSS- ja Atom-muotoisina. Tällöin *mentions* sanan jälkeen tulisi haluttu tiedostopääte, esimerkiksi mentions.json. Toinen parametri count=200 määrittelee, kuinka monta viestiä halutaan noutaa. Kaikilla Twitterin viesteillä on oma tunnisteensa. Myöhemmin saapuneissa viesteissä tunniste on suurempi kuin aiemmissa. Viimeinen parametri since_id=195672 määrittelee, että halutaan vain viestit, jotka ovat tulleet sen viestin jälkeen, jonka tunniste on 195672. RESTful-tekniikan mukaan haettaessa tietoa käytetään HTTP GET metodia, jolloin pyynnön URL-osoitteesta tulee esimerkin mukainen. Muita RESTful-tekniikan hyödyntämiä HTTP metodeja ovat POST, PUT ja DELETE, joiden avulla voidaan suorittaa erilaisia toimenpiteitä.

Jotta voitaisiin hyödyntää kaikkia RESTful ohjelmointirajapinnan toimintoja, on suoritettava todentaminen (authentication) yhdistettäessä Twitterin ohjelmointirajapintaan. Tämä on Illtagsissa toteutettu käyttäen PHP:n cURL kirjastoa. cURL funktiot on lisätty PHP:hen versiosta 4.0.2 lähtien ja nämä mahdollistavat kommunikoinnin eri palvelimien välillä useita eri protokollia käyttäen. Lisäksi cURL-kirjasto mahdollistaa myös todentamisen, jossa käytetään käyttäjänimeä ja salasanaa. (PHP: Introduction cURL 2009.) Todentamisella Twitter varmistaa, että rajapintaan tulleen pyynnön tekee sallittu sovellus. Lisäksi todentamisella pystytään päättelemään minkä käyttäjätunnuksen tietoja haetaan. Aiemmassa esimerkissä todentamisen avulla kerrotaan, että halutaan Illtags-käyttäjälle lähetetyt viestit.

Illtagsissa samaa bottia, joka kerää tiedot Twitteristä, käytetään myös varoituksien lähettämiseen, mikäli käyttäjään liittyvissä paikoissa havaitaan suuria tartuntariskejä. Tällöin Illtagsin Twitter-botille lähetetään tieto siitä, että käyttäjälle on lähetettävä henkilökohtainen varoitus. Botti luo tarvittavan viestin ja lähettää sen Illtagsin Twitter tililtä kyseessä olevalle käyttäjälle yksityisviestinä. Jälleen kaikki liikenne Twitteriin tapahtuu saman ohjelmointirajapinnan yli.

4.1.2 Facebook

Oiredatan kerääminen Facebookista eroaa jonkin verran Twitterin oiredatan keräämisestä. Jotta Facebookista saadaan haettua tietoa, on ensin tehtävä Facebook-sovellus, jonka käyttäjä voi asentaa omaan profiiliinsa. Vasta tämän jälkeen sovelluksen asentaneilta käyttäjiltä saadaan kerättyä tietoa. Facebookista ei siis pystytä keräämään ollenkaan julkista tietoa, joka Twitterissä olisi mahdollista. Tästä johtuen myös Illtagsille kohdennetut viestit on rajattu vain niihin käyttäjiin, joilla on Illtags-sovellus asennettuna Facebook-profiiliinsa.

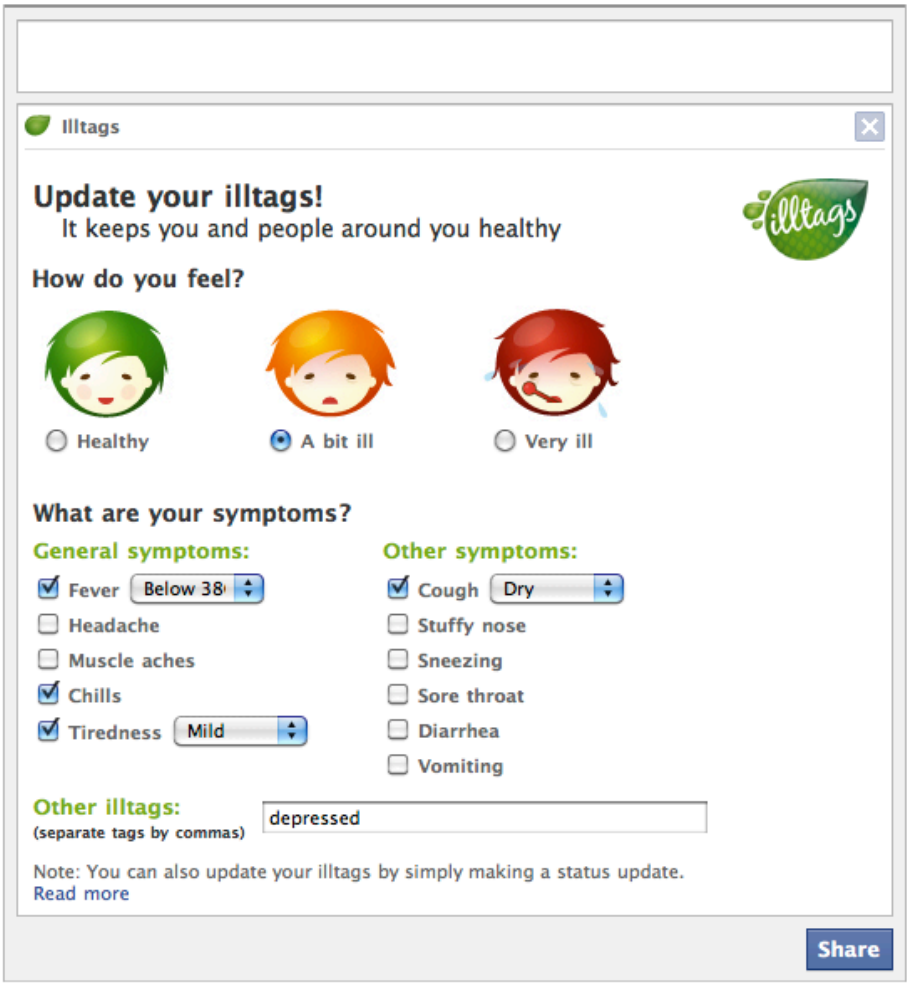
Facebook tarjoaa kehittäjille myös enemmän juuri Facebookia varten kehitettyjä työkaluja. Vaikka Facebook käyttää hyväkseen myös RESTful-ohjelmointirajapintatekniikkaa, on sen käyttäminen mahdollistettu eri tavoilla kuin Twitterissä. Facebook tarjoaa kehittäjille oman kehyksen (framework), jota käyttämällä kehittäjä voi helposti ottaa yhteyden Facebookiin ja suorittaa kyselyitä esimerkiksi toisella Facebookin omalla tekniikalla, FQL:llä (Esimerkki 2). (Facebook Query Language) FQL:n syntaksi on samantapaista kuin SQL, joten kehittäjien ei tarvitse opetella juurikaan mitään uutta. Facebookin tietoja pystytään hakemaan FQL:n avulla tarkemmin rajattuna, kuin Facebook kehyksen tarjoamilla hauilla. (Facebook Developer Wiki.)

```
1  // Initialize Facebook connection
2  require_once(COREROOT . '/facebook/facebook.php');
3  $facebook = new Facebook(FACEBOOK_APIKEY, FACEBOOK_SECRET);
4
5  // Get groups using Facebook Framework and FQL
6  $groups = $facebook->api_client->fql_query("SELECT gid, name, office,venue FROM group
7                  WHERE gid IN (SELECT gid FROM group_member
8                                    WHERE uid={$user['userID']})"
9                                    );
```

Esimerkki 2: Yhdistäminen Facebook kehyksen avulla ja esimerkki FQL kysely

Yllä oleva esimerkki hakee tietyn käyttäjän Facebook-ryhmien tietoja ja liittää ne myöhemmin Illtagsissa käyttäjään. Tällä tavoin voidaan hyödyntää Illtagsissa Facebookin ryhmiä, joihin käyttäjä kuuluu. Lähetetyt oiretunnisteet linkitetään Illtagsissa valittujen paikkojen lisäksi myös Facebook ryhmiin, joilla on maantieteellinen sijainti. Vastaavaa ei voida toteuttaa Twitterillä, sillä Twitter ei sisällä ryhmiä, joita Illtags voisi hyödyntää. Tämä on hyvä lisä Illtagsin omien paikkojen rinnalle ja antaa käyttäjälle mahdollisuuden tarkemmin seurata infektioriskejä ryhmissä, joihin hän kuuluu.

Illtagsin Facebook-sovellus noudattaa hyvin pitkälle totuttuja Facebook sovelluksen piirteitä. Oireet voidaan lähettää kahdella tavalla. Nopeaa ja vaivatonta tapaa suosivat tehokäyttäjät voivat kirjoittaa tilapäivitykseensä ”illtags:”, jonka jälkeen halutut oireet pilkulla erotettuna. Toinen vaihtoehto oireiden lähettämiseen on käyttää Facebook sovelluksen tarjoamaa graafista käyttöliittymää, josta voi valita sopivat oireet ja niiden voimakkuuden (Kuva 5). Lisäksi myös tässä tavassa on jätetty mahdollisuus kirjoittaa vapaasti oireita.



Kuva 5: Illtagsin Facebook sovelluksen graafinen oireiden lähettäminen

Graafisen oireenilmoittamisen lisäksi Illtagsin Facebook-sovellus näyttää ystäville lyhennetyn viestin oireista, ja tehdystä diagnoosista. Tämän jälkeen Facebook sovellus tarjoaa ystäville myös mahdollisuuden lähettää sairaalle ystävälleen sähköisiä kortteja, joilla voi kohottaa sairaan ystävän mieltä. Facebook-sovellus on hyvä tapa ilmoittaa oireita ja samalla seurata ystävien sairastumista, jotta voi itse välttää tartuntoja. Graafinen tapa ilmoittaa oireista on joillekin käyttäjille myös varmasti helpompi tapa ilmoittaa oireita, kuin esimerkiksi Twitterin oma Illtags-syntaksi. Facebook-sovellus on interaktiivisempi ja näin ollen myös varmasti kiinnostavampi käyttäjän kannalta.

Vaikka Facebookin tietojen kerääminen on hieman erilaista kuin Twitterin tietojen kerääminen, toimivat ne Illtagsin näkökulmasta samalla tavalla. Siinä missä Twitterin kohdalla yksinkertainen botti on ajastettu hakemaan tietoa, tekee Facebook-ohjelma tämän osan vastaavasti Facebookista. Ohjelma lähettää saamansa datan Illtagsin tietokantaan odottamaan myöhempää käsittelyä.

4.1.3 illtags.com

Illtags kerää oireita myös omalla internetsivullaan osoitteessa illtags.com ja tarjoaa käyttäjälle samalla lisätietoa infektioriskeistä. Käyttäjä voi kirjautua sivulle omalla Facebook- tai Twitter-tunnuksella. Kirjautuminen on toteutettu Facebookin osalta käyttäen sen omaa Facebook Connect ohjelmointirajapintaa, jonka avulla päästään käsiksi käyttäjän profiilitietoihin ja voidaan hyödyntää niitä kohdeohjelmassa. Illtags hyödyntää Facebookista käyttäjän profiilikuvaa, ryhmiä ja tietysti käyttäjän tunnistetietoja, joiden avulla voidaan yhdistää Facebookin kautta lähetetyt oireet käyttäjään. Twitter tunnuksella kirjautumiseen käytetään taas Twitter OAuth -tekniikkaa, joka hyödyntää avointa OAuth-protokollaa käyttäjän autentikoinnissa. (Twitter API Wiki / Oauth FAQ n.d.) Twitter tunnuksella kirjauduttaessa saadaan Illtagsin tietoon käyttäjän profiilitietoja ja tunnistetiedot, joiden avulla taas Twitteristä lähetetyt oireet voidaan yhdistää käyttäjään.

Illtagsissa käyttäjä ei voi yhdistää Twitter- ja Facebook-tilejään. Tämä on yksi palvelun heikoista puolista, sillä monet sosiaalista mediaa hyödyntävistä ihmisistä käyttävät molempia edellä mainittuja palveluja, ei vaan jompaakumpaa. Nyt kummastakin palvelusta saapuva tieto ei yhdisty samaan käyttäjään, vaan Illtagsissa Twitter tunnuksella tai Facebook tunnuksella kirjautuva käyttäjä on palvelun mukaan eri henkilö vaikka näin ei olisikaan. Tietysti tämä toisi lisää kehitettävää palveluun myös siinä mielessä, että monet käyttäjät päivittävät profiiliaan käyttäen yhtä ohjelmaa, joka lähettää päivityksen kaikkiin käyttäjän valitsemiin palveluihin. Tällöin pitäisi aina tarkistaa, että samaa viestiä ei lasketa kahdeksi, vaikka ne tulevat eri palveluista. Tämäkään tosin ei olisi vaikea tarkistus, eikä tilien yhdistäminenkään vaatisi suuria muutoksia sovelluksessa. Käytettävyyden taso nousisi tällaisenkin pienen korjauksen myötä melko paljon.

Illtags.com-sivuston käyttöliittymässä näkyy käyttäjän lähettämät oireet tunnistepilvenä. Lisäksi käyttäjä voi linkittää itsensä maantieteellisiin paikkoihin, jolloin näkyy muiden käyttäjien lähettämiä oireita, jotka kuuluvat samaan maantieteelliseen ryhmään. Maantieteelliset ryhmät ovat näkyvillä myös kartassa ja niiden infektioasteet on värikoodattu. Punainen tarkoittaa suurta tartuntariskiä, keltainen keskinkertaista tartuntariskiä ja vihreä olematonta tartuntariskiä. Käyttäjä voi myös tutkia paikkojen viikon takaista historiaa, viemällä hiiren kursorin paikan päälle. Tällöin kartalle piirtyy kuvaaja Illtagsin tekemistä diagnooseista sikainfluenssan ja flunssan osalta ja niiden infektioasteesta kuluneen viikon aikana.

Internet-sivu on myös yksi tapa lähettää oireita palveluun. Oire lähetetään perinteisellä lomakkeella syöttämällä oireet pilkulla erotettuna lomakkeen oirekenttään ja painamalla lähetä nappia. Toinen tapa lähettää oireita illtags.com-sivustolla on poimia valmiiksi listatuissa oireista haluamansa ja lähettää lomake, jonka oirekenttään valitut oireet ilmestyvät. Tällä tavoin lähetetyt oireet menevät suoraan Illtagsin tietokantaan odottamaan käsittelyä.

4.2 Datan siirto ja käsittely

Tavallisessa Internet-sovelluksessa käyttäjän lähettämää dataa siirretään yleensä lomakkeiden välityksellä. Tällöin tavallinen HTML-lomake kerää datan ja lähettää sen joko URL-muuttujina (HTTP-GET) tai HTTP-POST metodilla tiedon käsittelevälle komentosarjalle. (HTML form method Attribute n.d.) Käsittelevä komentosarja voi olla toteutettu esimerkiksi PHP:lla, jolloin lomakkeelta tuleva data saadaan kerättyä PHP:n $_GET tai $_POST –muuttujista suoraan ja voidaan tehdä tarvittavat käsittelyt. PHP:n sisällä dataa voi siirtää funktioiden välillä esimerkiksi omien muuttujien avulla. Funktioihin voi antaa muuttujan parametrina ja funktio voi palauttaa jonkin muuttujan suoritettuaan sille määrätyt toiminnot.

Aiemmin mainittu tapa toimii niin pitkään, kunnes pitää ladata uusi PHP-sivu. Tällöin PHP:n toimintatavan mukaan ladataan ja suoritetaan uusi komentosarja, jossa kaikki muuttujat alustetaan uudelleen alkuarvoihinsa, mikäli niitä ei ole tallennettu jonnekin ja haeta sieltä käyttöön. Tällaisia tallennuspaikkoja voi olla käyttäjän selaimeen tallentuvat eväste- ja palvelimelle tallentuvat istuntotiedot. Nämä kaksi ovat yleisiä käyttäjätietojen tallennuspaikkoja. Istuntotietoihin ja evästeisiin voidaan tallentaa käyttäjän tietoja ja esimerkiksi verkkokauppojen ostoskorin tietoja. Tällöin esimerkiksi verkkokaupan uusien sivujen lataaminen ei tyhjennä käyttäjän ostoskoria. (Käytännön PHP-opas n.d.)

Mikäli datalle tarvitaan pidempiaikainen varastointipaikka, tarjoaa PHP mahdollisuuden käyttää useita eri tietokantoja. Yleinen vaihtoehto on MySQL-relaatiotietokanta, jonka kanssa kommunikointiin PHP:ssa on hyvin toimiva laajennus. Tietokantaan voidaan tallentaa erittäin paljon dataa ja se voidaan hakea PHP:n käyttöön aina tarvittaessa. Tietokantahaut ja tiedon käsitteleminen kestävät kuitenkin jonkin verran kauemmin kuin sellaisen tiedon käsitteleminen, jota ei tarvitse hakea tietokannasta. Jotta datan käsitteleminen olisi nopeampaa, kannattaa usein haettavaa dataa tallentaa palvelimelle muistiin. Tähän tarkoitukseen on tarjolla *Memcached*, joka pystyy varastoimaan tietoa, kuten esimerkiksi tietokantakyselyn tuloksia, ja näin nopeuttamaan web-palvelua. (MySQL 5.1 Reference Manual n.d.)

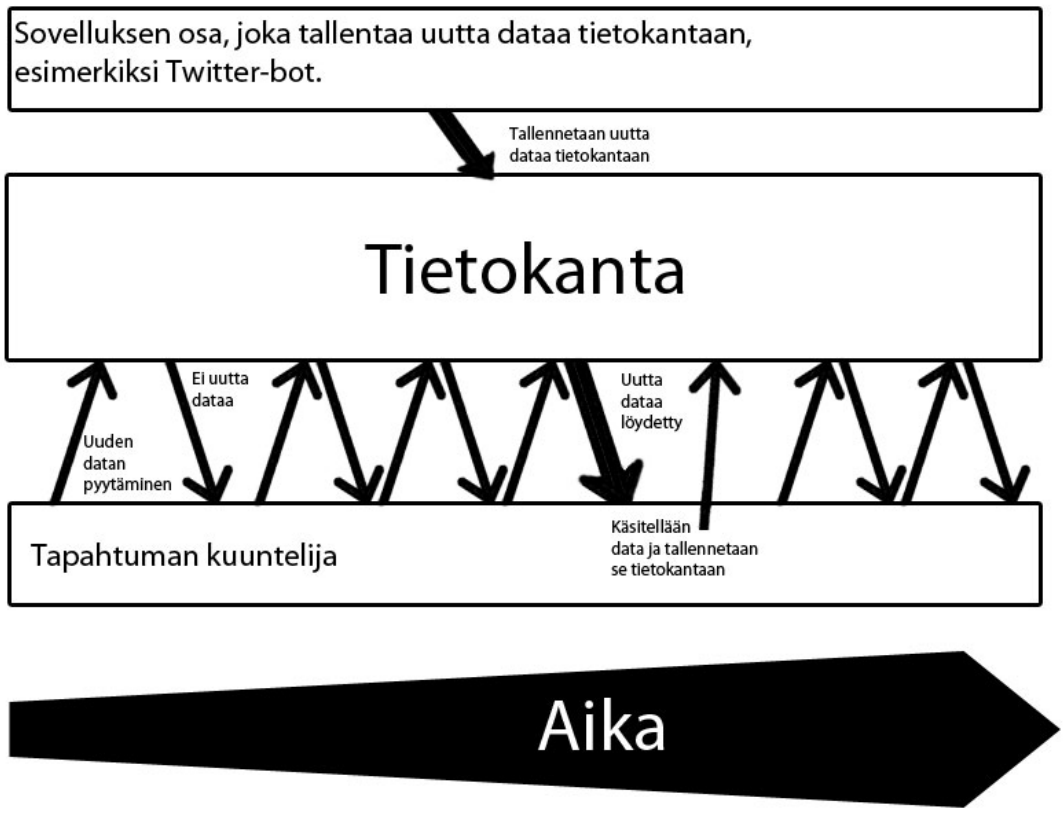
Illtagsissa dataa käsitellään monessa paikassa ja erityyppisesti. Tietoturvan kannalta on tärkeää, että kaikki sisään tuleva data tarkistetaan, ennen kuin se tallennetaan tietokantaan. Tarkistaminen on yksi hyvin pieni osa Illtagsin datan käsittelyä. Tämän lisäksi dataa käsitellään tiedon siirtojen yhteydessä, jolloin data saatetaan muuttaa toiseen muotoon, esimerkiksi normaalista merkkijonosta XML-muotoon, jotta sen käsitteleminen vastaanottavassa sovelluksen osassa olisi mahdollista. Dataa myös analysoidaan ja sen avulla tehdään diagnooseja. Tästä lisää alaluvussa 5.2.2 DocBot. Tietokantaa voidaan käyttää PHP:ssa hyväksi myös datan siirtämisen apuvälineenä. Illtagsissa dataa käyttävät sovelluksen eri osat, jotka on ajastettu toimimaan tietyin väliajoin. PHP:ssa ei ole mahdollista lähettää dataa suoraan ajastetulle komentosarjalle odottamaan käsittelyä. Ajastetun komentosarjan on haettava sille kuuluva data itse silloin kun

komentosarja suoritetaan. Tätä datansiirron välivaihetta varten, jossa data odottaa käsittelyn alkamista, Illtagsissa sovellettiin Flash-kehityksessä käytettyä tapahtuma-mallia (event).

4.2.1 Event-ajattelumalli

Adoben Flashissä käytetään omaa komentosarjakieltä, Actionscriptiä. Miltei kaikki Actionscriptillä toteutetun sovelluksen toiminnasta perustuu tapahtumiin (event), jotka ovat aina joitakin muutoksia, joita sovelluksessa tapahtuu. Tapahtuma voi olla Actionscriptiin sisäänrakennettu tai sovelluksen tekijän itsensä toteuttama. Hyvin yksinkertainen tapahtuma on esimerkiksi hiiren painallus. Jotta tapahtumaa voidaan käsitellä, täytyy tapahtumalla olla kuuntelija (event listener). Kuuntelijan ensimmäinen tehtävä on odottaa ja kuunnella, koska sille lähetetään tapahtuma. Tapahtuman kuuntelija on funktio, joka suorittaa sille määrätyn toiminnon, kun se vastaanottaa sille määritellyn tapahtuman. Tapahtumiin liittyy Actionscriptissä siis aina kaksi osapuolta: tapahtuman lähettäjä ja tapahtuman kuuntelija. (Moock 2007.)

Illtagsin datansiirrossa sovellettiin tällaista ajattelutapaa. PHP:ssa ei suoraan pystytä toteuttamaan edellä mainittua tapahtuma-kuuntelija -ratkaisua, sillä PHP-komentosarja ei ole jatkuvasti käynnissä, toisin kuin Flashissä käytettävä Actionscript. Tämä ongelma pystytään kiertämään siten, että ajatellaan kuuntelijaa ennemminkin kyselijänä, joka tarkistaa onko jotain dataa kohdistettu sille (Kuva 6). Näin pystytään käyttämään tietokantaa hyväksi välivaiheessa, jossa datan lähettävä komentosarja on suorittanut loppuun sille määritetyn tehtävän ja haluaa herättää tapahtuman. Tämän jälkeen kuuntelija, tai tässä tapauksessa kyselijä, tarkistaa onko sille lähetetty dataa käsiteltäväksi. Useissa tapauksissa Illtags-sovelluksessa kyselijälle ei ole käsiteltävää dataa, jolloin se tekee kyselyn hetken kuluttua uudelleen ja jossain vaiheessa saa dataa käsiteltäväksi.



Kuva 6: Esimerkkikaavio tapahtuma-mallista kohdesovelluksessa

Edellä kuvatun kaltaisia kuuntelijoita on Illtags-sovelluksessa useita ja ratkaisu on perusteltu, sillä kuuntelijat ja muut taustalla ajettavat sovelluksen osat tekevät asynkronisesta toiminnasta mahdollisen. Sovelluksen asynkronisen toiminnan perustana ovat säikeet, joita Illtags-sovelluksessa käytetään useita. Muuttunutta dataa voi jossain tilanteessa olla niin paljon, että tietyn käsittelyn suorittava sovelluksen osa ei selviä yksin kaiken tarvittavan datan käsittelystä tarpeeksi nopeasti. Tällöin on pystyttävä toteuttamaan esimerkiksi useampia samanlaisia kuuntelijoita, jotka pystyvät hakemaan muuttunutta dataa ja käsittelemään sitä samalla kun toinen kuuntelija käsittelee vielä edellistä. Tavallisessa toimintamallissa data välitetään suoraan PHP-komentosarjalta toiselle samalla kun toista PHP-komentosarjaa kutsutaan. Tätä ei kuitenkaan pystytä toteuttamaan Illtagsissa, koska käsittelyyn tarvittava data voi tulla monesta eri paikasta. Data kertyy tietokantaan, josta tapahtuman kuuntelija voi hakea sen käsiteltäväksi, kun kaikki tarvittava data on valmiina. Toinen tavallisen toimintamallin estävä asia on käsittelyn priorisointi. Jos kaikki uusi data käsiteltäisiin kerralla, ja käsittelyn päättyessä kutsuttaisiin seuraavan käsittelyn tekevää sovelluksen osaa, ei käsittelyn priorisoinnista olisi juuri hyötyä, koska seuraava osa voisi aloittaa datan käsittelyn kuitenkin vasta ensimmäisen valmistuttua. Kun käytetään kuuntelijoita, jotka käsittelevät dataa pieninä paloina ja siirtävät sitä pikkuhiljaa eteenpäin sovelluksessa, saadaan nopeampaa käsittelyä tarvitseva data liikkeelle samalla kun vielä käsitellään muuta dataa.

Tapahtumien kuuntelijoiden käyttäminen on erikoinen, mutta erittäin toimiva ratkaisu suureen datamäärään varauduttaessa. Tämän ratkaisun avulla sovelluksen osat eivät ole riippuvaisia toisista sovelluksen osista siinä mielessä, että ne pystyvät itsenäisesti hakemaan niille kuuluvan datan. Tällä tavoin on myös helpompi kehittää sovellukseen lisäominaisuuksia, koska jälkeenpäin ei tarvitse alkaa muokkaamaan jo valmiina olevaa koodia, jotta saataisiin se kutsumaan uutta sovelluksen osaa suoritusvaiheessaan. Uusi sovelluksen osa voi kuunnella, koska sille kuuluva data on käsiteltävissä ja suorittaa näin käsittelyn ja tiedon hakemisen itsenäisenä, muista riippumattomana sovelluksen osana.

4.2.2 DocBot

DocBot on Illtagsin lääketieteellisen diagnoosin tekevä botti, joka sai nimensä englannin kielen tohtori (doctor) sanan lyhenteestä *doc.* DocBot on siis erittäin oleellinen osa Illtagsin datankäsittelyä. Kaikki lääketieteelliset oireet kulkeutuvat ajan myötä DocBotille käsittelyyn. Toistaiseksi DocBot käsittelee oireita paikkojen perusteella, eli tehdyt diagnoosit eivät ole henkilökohtaisia. Lisäksi tämän hetken diagnosointi on kohdistettu vain flunssan ja sikainfluenssan tunnistamiseen, jotka ovat tämän työn kirjoitushetkellä erittäin suuren kiinnostuksen kohteena.

Yleiseltä toiminnaltaan DocBot vastaa tavallista sovelluksen kuuntelijaa, joka tarkistaa säännöllisin väliajoin onko diagnosoitavaksi tullut uutta dataa. Se mikä erottaa DocBotin toiminnan normaalista kuuntelijasta on priorisointi, sekä tietysti datan käsittelyvaiheen diagnosointi. Jos uutta dataa on useammasta paikasta, päättelee DocBot priorisoinnin, jolla paikat diagnosoidaan. Priorisointiin vaikuttaa esimerkiksi aika, joka on kulunut viimeisen diagnoosin tekemisestä kyseisen paikan kohdalla. DocBot siis hakee diagnosoitavaksi yhden paikan kerrallaan, merkitsee tietokantaan, että on hakenut datan käsiteltäväksi ja aloittaa diagnosoinnin. Tietokannassa olevan *”käsittelyssä”*-merkinnän tarkoituksena on estää päällekkäisten diagnosointien tekeminen, mikäli käytössä on samanaikaisesti useampia DocBotteja. Samaa käytäntöä käytetään myös muissa sovelluksen osissa, joissa käsittelyä voi tehdä useampi samantyyppinen botti.

Kun DocBot on päätellyt sen, millä paikalla on korkein prioriteettiaste, hakee se paikan datan tietokannasta. Käsiteltävää dataa voi olla yhdessä paikassa hyvinkin paljon ja se tulee DocBotille XML-muotoisena, jota käydään läpi PHP:n SimpleXML laajennuksen avulla. SimpleXML laajennuksen avulla voidaan helposti muuttaa xml-muotoinen data PHP:n olioksi, jota on helppo käsitellä normaalin PHP syntaksin mukaisesti. (PHP: Introduction SimpleXML 2009.) Käsitellessään dataa, DocBot suorittaa tiettyjä algoritmeja, joiden avulla se laskee todennäköisyyksiä siitä, miten suurella todennäköisyydellä tietyssä paikassa on flunssaa tai sikainfluenssaa. Tämä datan käsittelyn osa saattaa kestää pidemmän aikaa, jonka vuoksi diagnosointi ei tapahdu reaaliaikaisesti, vaan diagnoosijonoa puretaan sitä myötä, kun DocBot on saanut diagnooseja valmiiksi. Koska DocBot suoritetaan itsenäisenä sovelluksen osana, voidaan tarvittaessa monistaa DocBotteja, mikäli diagnosointia halutaan nopeuttaa. Sovellukseen voidaan myös haluttaessa

kehittää eri diagnooseja tekeviä DocBotteja samaa monistusperiaatetta hyödyntäen. Tällöin yhden tyyppinen DocBot voi diagnosoida esimerkiksi flunssaa ja toinen vesirokkoa. Koska DocBot toimii itsenäisenä osana, ei muutoksia tarvitse tehdä jo valmiina olevaan sovellukseen. Riittää kun uudelle DocBotille määrätään, minkälaista dataa sen pitää tarkkailla.

Suoritettuaan diagnoosin DocBot lähettää paikan diagnoosin tietokantaan ja merkitsee hakemansa datan käsitellyksi. Tämän jälkeen datan esittävä sovelluksen osa voi hakea diagnooseja ja näyttää tulokset illtags.com-sivustolla. DocBot jatkaa työskentelyä diagnosoimalla seuraavan paikan.

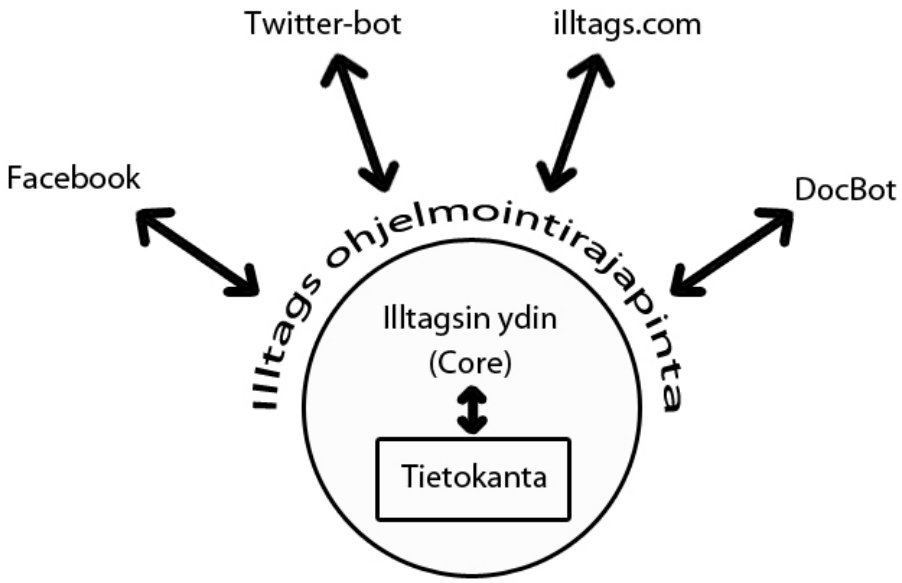
DocBot on sovelluksen osa, jonka toteuttamisessa voitaisiin mahdollisesti kuvitella käytettävän ennemmin esimerkiksi C-ohjelmointikieltä kuin PHP:ta. C-ohjelmointikieli on käännettävä kieli, mikä tarkoittaa sitä, että sen suorittaminen on nopeampaa, koska lähdekoodi on käännetty valmiiksi konekieleksi. (Compiled versus interpreted languages n.d.) Mikäli siis PHP-koodilla toimiva DocBot alkaa viemään liikaa palvelimen resursseista suorittaessaan diagnooseja, voisi C-ohjelmointikielellä toteutettu, hyvin optimoitu DocBot olla parempi vaihtoehto.

4.2.3 Illtags API

Monissa tämän päivän suosituissa verkkopalveluissa on käytössä ohjelmointirajapinta, jonka avulla kehittäjät voivat hyödyntää palvelun toimintoja omissa sovelluksissaan. Tällaisia palveluita on Twitterin ja Facebookin lisäksi muun muassa Yahoolla ja Googlella. Suuri osa viimeaikoina toteutetuista web-palveluista hyödyntävät RESTful ohjelmointirajapintaa. Poikkeuksena on esimerkiksi Googlen palvelut, jotka hyödyntävät SOAP ohjelmointirajapintaa, joka on kehittäjän kannalta RESTful rajapintaa hieman monimutkaisempi. RESTFul ohjelmointirajapinnassa dataa siirretään HTTP-pyynnöillä ja pyydettävä data haetaan URL-osoitteen avulla. Tästä on esimerkki aiemmassa alaluvussa 4.1.1 Twitter. SOAP ohjelmointirajapinta toimii hieman samalla tavalla, mutta URL-osoitteiden sijaan käytetään PHP:n metodikutsun tyylisiä pyyntöjä, esimerkiksi *api->haeKayttaja()*. (Campbell 2006.) RESTful ohjelmointirajapinta palauttaa pyydetyn datan ihmisen luettavassa muodossa, jota on helppo alkaa käsittelemään rajapintaa käyttävässä sovelluksessa. SOAP rajapinnan vastaukset joudutaan käsittelemään monimutkaisemmalla tavalla, koska data ei ole suoraan luettavassa muodossa. (Freitag 2005.) Tietoturvan kannalta pitää huomioida se, että RESTful rajapinnan yli liikkuva data on luettavaa. SOAP on parempi ratkaisu, mikäli rajapinnan yli siirretään arkaluontoista dataa. Illtagsin rajapinnan yli liikkuva data ei ole arkaluontoista, joten yksinkertaisempi ja kevyempi RESTful ohjelmointirajapinta on sopiva valinta sovellukseen.

Illtagsin rakenne on hyvin modulaarinen. Pääasiassa Illtags koostuu useista boteista ja sovelluksen ytimestä. Illtagsin ydin on sovelluksen osa, joka suorittaa kaiken kommunikaation tietokannan kanssa, sekä jakaa siltä pyydettyä dataa. Ytimen ja muiden sovelluksen osien välinen

datansiirto tapahtuu Illtagsin oman ohjelmointirajapinnan kautta, joka on toteutettu RESTful-menetelmällä (Kuva 7). Tämä mahdollistaa sovelluksen helpon laajentamisen, koska uuden sovelluksen osan, esimerkiksi jonkinlaisen DocBotin tai tunnisteita hakevan botin, kehittäjän ei tarvitse ensin opetella, miten Illtagsin monimutkainen tietokanta toimii ja miten sitä kannattaa hyödyntää. Sen sijaan kehittäjä voi lähettää tai hakea yksinkertaisen RESTful ohjelmointirajapinnan avulla tarvitsemansa datan. Illtagsin ydin hoitaa tämän jälkeen tietokantahaut ja palauttaa dataa, joko XML tai JSON muotoisena.



Kuva 7: Esimerkki Illtags sovelluksen ohjelmointirajapinnan käytöstä

4.3 Datan esittäminen

Datan esittäminen on se osa kohdesovelluksesta, joka näkyy käyttäjälle. Se on myös osa sovellusta, joka toimii perinteisen web-sovelluksen tavoin. Tämä osa ei ole Ajax-osiota lukuun ottamatta asynkroninen ja itsenäisesti toimiva, vaan vaatii käyttäjältä komentoja toimintojen suorittamiseksi.

Kun käyttäjä navigoi ja kirjautuu sisään illtags.com sivustolle, hänelle näytetään dataa sekä graafisessa että tekstimuodossa. Graafinen osio sisältää kartan, jossa näkyvät kaikki ne paikat, johon käyttäjän on mahdollista itsensä yhdistää, tai joihin hän jo kuuluu. Kartta on toteutettu hyödyntäen Google Maps -karttapalvelun tarjoamaa ohjelmointirajapintaa, jolla maailmankartan saa sijoitettua omalle sivustolleen. Google Maps käyttää Ajax-tekniikkaa kartassaan, johon pystytään sitomaan Illtagsin omia paikkamerkintöjä käyttäen JavaScriptiä. Paikkamerkit voidaan asettaa leveys- ja korkeuspiiriarvojen mukaan oikeaan paikkaan kartalla, hyödyntäen Google Maps -ohjelmointirajapinnan päällyste-ominaisuutta (overlay). Tämän ominaisuuden avulla kar-

talle voidaan asettaa itse tehtyjä merkkejä (marker), jotka Illtagsissa näyttävät paikan infektioasteen värillään. (Map Overlays - Google Maps API 2009.)

Paikkamerkkien myötä voidaan käyttäjälle esittää tarkempaa dataa paikasta käyttäen hyödyksi Ajax-tekniikkaa. Kun käyttäjä vie hiiren jonkin paikan päälle kartalla, ilmestyy paikan yläpuolelle pieni ikkuna, jossa näkyy kyseisen paikan infektioaste sikainfluenssan ja flunssan osalta viimeisen 7 päivän aikana graafisena kuvaajana. Lisäksi ikkunassa näkyy myös käyttäjien määrä, jotka ovat liittyneet kyseiseen paikkaan. Tämä toiminnallisuus on toteutettu hyödyntäen avointa jQuery JavaScript kirjastoa. JavaScript kirjasto jQuery mahdollistaa tapahtumienkäsittelyn, animaation ja monen muun vuorovaikutteisuuden kehittämisen web-projektissa helposti ja nopeasti (jQuery: The Write Less, Do More, JavaScript Library 2009).

Graafisten kuvaajien piirtoon Internet-sivulla on olemassa useita ratkaisuja. Yksinkertaisten kuvaajien piirtoon voidaan hyödyntää esimerkiksi PHP:n omaa GD kirjastoa, jonka avulla pystytän käsittelemään tai jopa luomaan kuvia. (PHP: Introduction GD 2009.) On myös olemassa lukuisia valmiita kuvaajien piirtoon tarkoitettuja kirjastoja, jotka käyttävät eri tekniikoita. Käytettävän kuvaajan valinta web-sovellukseen on hyvin pitkälle makuasia, kuten kaikki graafiset osat sivustolla. Illtags halusi helppokäyttöisen ja kevyen, mieluiten JavaScriptillä toimivan kuvaajan. Siksi sivustolla päädyttiin käyttämään kuvaajaa nimeltä *Flot.* Flot on avoin jQuery kirjasto, joka piirtää graafisen kuvaajan dynaamisesti käyttäjän koneella sille annetusta datasta (Flot Attractive Javascript plotting for jQuery 2009). Koska kuvaaja piirretään JavaScriptillä, saadaan sen avulla poistettua kuvaajan piirtämisen ja sen datan käsittelyn aiheuttama kuorma palvelimelta.

Illtagsin verkkosivuilla on vielä yksi tapa esittää käyttäjälle dataa ja se on tekstimuotoisten tunnisteiden esittäminen. Tunnisteet kerätään tunnistepilviksi, joita käyttäjälle esitetään kaksi. Toisessa on käyttäjän itsensä lähettämät tunnisteet ja toisessa kaikki ne tunnisteet, joita samoihin paikkoihin linkittyneet käyttäjät ovat lähettäneet.

Kaikki sivustolla näytettävä data haetaan käyttäen Illtagsin ohjelmointirajapintaa. Tunnisteiden kohdalla haetaan tunnisteet käyttäjän ja käyttäjään liittyvien kontekstien tauluista Illtagsin tietokannasta ja paikkojen infektioasteet haetaan Illtagsin diagnoositaulusta, johon DocBot on tallentanut aiemmin tehdyt diagnoosit. Tunnisteet, kartalle tulevat paikat ja niiden infektioasteet haetaan sivuston latauksen yhteydessä normaalin web-sivuston tapaan. Tarkemmat infektiotiedot haetaan Ajax-tekniikalla ajon aikana, kun käyttäjä vie hiiren kursorin paikan päälle, josta haluaa lisätietoja. Tällöin myös piirretään kuvaaja haetun 7 päivän datan perusteella.

Kaikki esitettävä data haetaan Illtags ohjelmointirajapinnan yli, joka mahdollistaa myös datan esittämistavan muuttamisen haluttaessa. Ohjelmointirajapinta mahdollistaa myös datan esittämisen muualla, kuin illtags.com-sivustolla, ja siksi on myös järkevää, että kaikki esitettävissä oleva data on saatavilla ohjelmointirajapinnan kautta.

5 Säikeistys sovelluksessa

Puhuttaessa tietokoneen sovelluksista yleisellä tasolla, voidaan sanoa, että jokainen sovellus käynnistää tietokoneella oman prosessinsa. Tällainen sovelluksen prosessi sisältää säikeen *(thread)*, jossa sovelluksen toimintalogiikka tapahtuu. Kaikki mitä sovelluksen käynnistämisen jälkeen tapahtuu, suoritetaan säikeessä. Joidenkin sovellusten prosesseissa voi olla useampia säikeitä. Tällainen prosessi on monisäikeinen *(multithread)* prosessi. (Tanenbaum 2001, 81.)

Tanenbaumin mukaan monisäikeisissä prosesseissa on muutamia etuja verrattuna yksisäikeisiin prosesseihin. Suurimmat hyödyt ovat tehokkuudessa, sillä säikeet ovat prosessien sisällä ja niiden käynnistäminen ja lopettaminen on huomattavasti nopeampaa kuin kokonaisten prosessien käynnistäminen ja lopettaminen. Useamman säikeen hyödyt tulevat esiin silloin, kun ohjelman pitää tehdä saman aikaisesti useampaa eri tehtävää. Esimerkiksi ladattaessa tiedostoa Internetistä selaimen latausikkuna näyttää käyttäjälle tietoja latauksen kulusta. Yleisiä näytettäviä tietoja on latauksen nopeus, arvio latauksen suoritusajasta ja latauksen edistyminen. Kun ohjelma lataa tiedostoa ja haluaa näyttää sen edistymisen käyttäjälle, kannattaa hyödyntää kahta säiettä, joista toinen suorittaa latauksen ja toinen suorittaa latauksen tilan esittämisen. Yhdellä säikeellä toteutettuna latauksen suorittava säie joutuisi keskeyttämään lataustoimintansa aina, kun käyttäjälle näytettävää tietoa joudutaan päivittämään. (Tanenbaum 2001, 85-90.)

Samanaikaisesti suoritettavat eri prosessit ovat se, mitä Illtags tarvitsee suorittaessaan datan käsittelyä ja hakua. PHP:ssa ei ole suoraan tukea säikeistyksille, mutta monisäikeistystä pystyttäisiin simuloimaan erityyppisillä ratkaisuilla, joista löytyy Internetistä useita erilaisia esimerkkejä. Nämä eivät kuitenkaan vakuuttaneet Illtagsin kehitystiimiä. Vaihtoehtona PHP:lla suoritettavaan säikeistyksen simulointiin on toteuttaa säikeistys käyttäen sitä suoraan tukevaa ohjelmointikieltä, kuten esimerkiksi Java- tai C/C++ -ohjelmointikieltä. Tällöin sovelluksessa voitaisiin toteuttaa toimiva monisäikeistys, joka suorittaa samanaikaisia tehtäviä. Tällainen ratkaisu Illtagsissa voi kuitenkin olla ongelmallinen, sillä kaikki sovelluksen osat suoritettaisiin yhden sovelluksen sisällä. Mikäli sovellusta laajennetaan tai johonkin sovelluksen osaan, esimerkiksi oireiden noutamiseen Twitteristä, on tehtävä muutoksia, pitäisi koko ohjelma sammuttaa muutoksen ottamiseksi käyttöön sovelluksessa.

Illtags-sovelluksessa suoritetaan useita samanaikaisia tehtäviä, mutta jokainen sovelluksen osa suoritetaan itsenäisesti. Vaikka periaatteessa tämä vastaa toiminnaltaan hieman monisäikeistystä, ei se käytännössä ole sitä, vaan jokainen sovelluksen osa on oma prosessinsa palvelimella. PHP:n toimintalogiikan mukaan komentosarja suoritetaan, kun käyttäjä navigoi selaimellaan PHP-komentosarjan sisältävälle sivulle. Illtagsissa taas PHP-komentosarjoja pitää pystyä suorittamaan itsenäisesti ilman, että käyttäjä käynnistää toiminnot. Tätä varten Illtagsin PHP-komentosarjojen käynnistämiseen on toteutettu yksinkertainen *bash*-komentosarja, joka suorite-

taan palvelimella. Tässä komentosarjassa on ikuinen silmukka, joka kutsuu sille määrättyä PHP-komentosarjaa jatkuvasti (Esimerkki 3).

```
1 #!/bin/bash
2 while true; do
3    php -f twitter.php
4 done
```

Esimerkki 3: Bash-komentosarja, joka kutsuu PHP-komentosarjaa

Yllä olevassa esimerkissä on bash-komentosarja, joka käynnistää datan haun Twitteristä. Twitter on rajoittanut sille tehtävien pyyntöjen määrää ja siksi twitterGetTweets.php -komentosarjan sisällä on huolehdittava, että pyyntöjen määrä ei ylity. Tämä on toteutettu yksinkertaisesti PHP:n usleep*()*-funktion avulla, joka keskeyttää komentosarjan suorittamisen sille parametrina annetuksi ajaksi. Esimerkiksi *usleep(5000000)* keskeyttää suorituksen 5 sekunnin ajaksi, koska parametri annetaan mikrosekunteina. Sama suorituksen keskeyttäminen ajaksi x voitaisiin toteuttaa myös bash-komentosarjassa, mutta tällöin ei huomioitaisi PHP-komentosarjan suorittamiseen kuluvaa aikaa, jolloin esimerkiksi Twitteriin voisi lähteä pyyntöjä liian usein.

Huomioon ottaen sovelluksen tarpeet saada useita eri sovelluksen osia ajettua samanaikaisesti ja silti säilyttäen helppo ylläpidettävyys ja laajennettavuus, on tällainen ratkaisu hyvinkin toimiva. Vaikka PHP on nykyään jo suhteellisen tehokas kieli, saattaa se tässä sovelluksessa joutua koville. Mikäli jokin sovelluksen osa alkaa viemään liikaa palvelimen resursseja tai se ei suoriudu tehtävistään riittävän nopeasti, voidaan se jälkikäteen toteuttaa esimerkiksi tehokkaammalla C-ohjelmointikielellä, ilman että joudutaan tekemään muutoksia sovelluksen muihin osiin. Tällainen tilanne saavutetaan kuitenkin ainoastaan silloin, kun sovelluksen käyttäjämäärä kasvaa todella suureksi, ja miltei jokainen lähettäisi oireita samanaikaisesti.

6 Yhteenveto

PHP:n valinta sovelluksen ohjelmointikieleksi mahdollisti sovelluksen nopean kehittämisen edullisesti. Tämä oli tärkeää Kaisla Innovationin kiireisen aikataulun ja pienen budjetin takia. Vaikka PHP suoriutuu sovelluksessa toistaiseksi hyvin, ei se silti tarkoita sitä, etteikö muiden ohjelmointikielten käyttäminen ainakin osassa ohjelmistoa olisi hyödyllistä, varsinkin kun sovelluksen käyttäjämäärät kasvavat ja käsiteltävän datan määrä moninkertaistuu. Sovelluksen rakenteen modulaarisuus mahdollistaa sovelluksen osien päivittämisen helposti jälkikäteen ja tällöin on myös mahdollista muuttaa joitain sovelluksen osia eri ohjelmointikielillä toteutetuiksi.

Sovelluksen erikoisen toimintatavan takia jotkut käytetyistä ratkaisuista ovat hyvinkin paljon normaalista web-sovelluksesta poikkeavia. Näistä ominaisuuksista ainakin monisäikeistyksen korvaavat bash-komentosarjoilla suoritettavat sovelluksen osat, jotka tekevät sovelluksen toiminnasta asynkronisen, ovat tärkeä osa sovelluksen modulaarisuutta. Koska ohjelman taustalla on käynnissä erillisiä, itsenäisiä prosesseja, voidaan tarvittaessa jokin prosesseista pysäyttää päivityksen tai huollon vuoksi, ja silti sovellus toimii muilla käynnissä olevilla prosesseilla normaaliin tapaan. Tämä ratkaisu on erittäin onnistunut tulevaisuuden ylläpitoa silmällä pitäen.

Datan kerääminen sovellukseen tapahtuu pääasiassa Twitterin ja Facebookin kautta. Molemmat palvelut tarjoavat kehittäjille käytettäväksi ohjelmointirajapinnan, jonka välityksellä voidaan hakea dataa palvelusta. Tämän sovelluksen osan toteuttamisessa ei ole muita mahdollisuuksia, kuin rajapintojen tarjoamat vaihtoehdot. Ainoa asia, johon datan keräämisessä pystyy vaikuttamaan on se, millaisella syntaksilla data lähetetään palveluun. Twitterin oma hashtageja hyödyntävä ratkaisu on varmasti Twitteriä enemmän käyttäville hyvinkin tuttu ja yksinkertainen tapa lähettää dataa. Mikäli vähemmän Twitteriä käyttänyt henkilö haluaa syystä tai toisesta käyttää Twitteriä lähettäessään oireita palveluun, on se varmasti hankalampaa, kuin Facebookissa käytettävä syntaksi, jossa ei ole erikoisia merkkejä viestiä lähetettäessä. Myös useamman eri syntaksin käyttäminen oireiden lähettämisessä ei välttämättä ole paras mahdollinen ratkaisu. Tällöin käyttäjä joutuu opettelemaan erilaisia syntakseja, mikäli haluaa joskus ilmoittaa oireita käyttäen eri palvelua kuin ennen. Paras mahdollinen tapa kerätä oireita, olisi käyttää luonnollisen kielen tulkkia, jolloin käyttäjä voisi lähettää oireet haluamassaan muodossa ja palvelu tunnistaisi nämä siitä huolimatta. Toistaiseksi tällaisen ratkaisun toteuttaminen ei ole realistista ottaen huomioon Kaisla Innovationin ja Illtagsin resurssit. Twitter- ja Facebook-tilien pitäminen erillään on myös yksi käytettävyyteen liittyvä ongelma palvelussa. Käyttäjä ei pysty yhdistämään omia Twitter ja Facebook tilejään ja sama käyttäjä voi olla kaksi eri käyttäjää Illtagsin näkökulmasta. Monien kohdalla tämä ei välttämättä ole ongelma, mutta jos käyttäjä lähettää oireita välillä käyttäen Twitteriä ja välillä Facebookia, niin silloin oireet menevät Illtagsin näkökulmasta eri käyttäjille.

Illtagsin oma ohjelmointirajapinta yhdessä, bash-komentosarjojen ja tapahtumankuuntelijoiden kanssa tekevät sovelluksesta erittäin modulaarisen. Sovelluksen sisäinen datansiirto on erikoinen, sillä periaatteessa sovelluksen sisäinenkin liikenne tapahtuu oman ohjelmointirajapinnan läpi. Tämä on kuitenkin perusteltua, sillä tämä mahdollistaa osittain sovelluksen asynkronisen toiminnallisuuden sekä helpottaa ylläpidettävyyttä ja laajennettavuutta huomattavasti, koska sovelluksen osien toimintoja voidaan muuttaa hyvinkin perusteellisesti ja silti dataa liikutellaan samalla tavalla sovelluksen osien välillä.

Sovelluksessa datan siirtoon ja keräämiseen käytetyt ratkaisut ovat toimivia ja mahdollistavat helposti laajennettavan ja ylläpidettävän sovelluksen. Kuitenkin datan keräämisessä voitaisiin vielä miettiä sovelluksen toimintaa käytettävyyden kannalta. Tehokkaampaa tiedon käsittelyä tarvittaessa on sovelluksen raskaita toimintoja suorittavat osat mahdollista muuttaa toimimaan tehokkaampaa ohjelmointikieltä, kuten esimerkiksi C-kieltä käyttäväksi vaikuttamatta sovelluksen muuhun toimintaan.

Sosiaalisen median palvelut ovat tulleet jäädäkseen. Tämän ansiosta on mahdollista kerätä yhä enemmän ihmisten mielipiteitä ja ajatuksia eri asioista. Hyödyntämällä sosiaalista mediaa, voidaan luoda uusia ja entistä mielenkiintoisempia palveluita, joissa käsitellään ja tilastoidaan käyttäjien lähettämää tietoa uudella tavalla. Tietoa kertyy niin paljon, ettei normaalin web-sovelluksen avulla pystytä millään käsittelemään sitä kokonaan. Asynkroninen web-sovellus on tällöin sopiva ratkaisu ja tulevaisuudessa varmasti monet web-palveluista tulevat perustumaan jonkinlaiseen asynkroniseen toimintaan.

Lähteet

Campbell, Ryan 2006. How to Add an API to your Web Service. [online] [viitattu: 14.11.2009] http://particletree.com/features/how-to-add-an-api-to-your-web-service/

Compiled versus interpreted languages. [online] [viitattu: 15.11.2009] http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp?topic=/com.ibm.zos.zappldev/zappldev_85.htm

Facebook Developer Wiki. [online] [viitattu 9.11.2009] http://wiki.developers.facebook.com

Flot Attractive Javascript plotting for jQuery 2009 [online] [viitattu: 15.11.2009] http://code.google.com/p/flot/

Freitag, Pete 2005. REST vs. SOAP Web Services. [online] [viitattu: 14.11.2009] http://www.petefreitag.com/item/431.cfm

Gaarsmand, Regin. The PHP web site all PHP developers ought to know about [online] [viitattu: 3.11.2009] http://www.sourcerally.net/regin/21-The-PHP-web-sites-all-PHP-developers-ought-to-know-about

Garret, Jesse James, 2005. Ajax: A New Approach to Web Applications [online] [viitattu: 3.11.2009] http://www.adaptivepath.com/ideas/essays/archives/000385.php

Google Yritystiedot: Tekniikan yleiskatsaus 2009. [online] [viitattu: 4.10.2009] http://www.google.fi/intl/fi/corporate/tech.html

Hosch, William L. Web 2.0 [online] [viitattu: 31.10.2009] http://www.britannica.com/EBchecked/topic/1192837/Web-20

HTML form method Attribute. [online] [viitattu 10.11.2009] http://www.w3schools.com/tags/att_form_method.asp

Hudson, Paul 2005. PHP in a Nutshell. O'Reilly Inc, Sebastopol, CA.

jQuery: The Write Less, Do More, JavaScript Library. [online] [viitattu: 15.11.2009] http://jquery.com/

Käytännön PHP-opas: Osa 7 – Evästeet ja istunnot [online] [viitattu 11.11.2009] http://www.ohjelmointiputka.net/opas.php?tunnus=phpj7

Makice, Kevin 2009. Twitter API: Up and Running. O'Reilly Inc, Sebastopol, CA.

Map Overlays – Google Maps API 2009. [online] [viitattu: 15.11.2009] http://code.google.com/apis/maps/documentation/overlays.html

Moock, Colin 2007. Essential ActionScript 3.0. O'Reilly Media Inc, Sebastopol, CA.

MySQL 5.1 Reference Manual :: 14.5 Using MySQL with memcached [online] [viitattu: 11.11.2009] http://dev.mysql.com/doc/refman/5.1/en/ha-memcached.html

Negrino, Tom, Smith Dori 2007. JavaScript Tehokas hallinta. Readme.fi. Helsinki.

O'Reilly, Tim 2008. Esipuhe Amy Shuenin teoksessa A Strategy Guide. Business thinking and strategies behing successful Web 2.0 implementations. O'Reilly Media Inc, Sebastopol, CA.

PHP: Introduction cURL 2009. [online] [viitattu: 8.11.2009] http://www.php.net/manual/en/intro.curl.php

PHP: Introduction GD. [online] [viitattu: 15.11.2009] http://fi.php.net/manual/en/intro.image.php

PHP: Introduction SimpleXML 2009. [online] [viitattu: 14.11.2009] http://www.php.net/manual/en/intro.simplexml.php

PHP: New Object Model 2009. [online] [viitattu: 1.9.2009] http://www.php.net/manual/en/migration5.oop.php

PHP: PHP Usage Stats 2009. [online] [viitattu: 31.8.2009] http://www.php.net/usage.php

Reiss, Eric 2009. Why PHP won. [online] [viitattu: 31.8.2009] http://startuplessonslearned.blogspot.com/2009/01/why-php-won.html

Shuen, Amy 2008. Web 2.0: A Strategy Guide. Business thinking and strategies behing successful Web 2.0 implementations. O'Reilly Media Inc, Sebastopol, CA.

Tanenbaum, Andrew S 2001. Modern Operating Systems (2nd edition). Prentice Hall. Englewood Cliffs, NJ.

Twitter API Wiki / OAuth FAQ 2009. [online] [viitattu: 9.11.2009] http://apiwiki.twitter.com/OAuth-FAQ

Twitter API Wiki / Rate limiting 2009. [online] [viitattu 6.11.2009] http://apiwiki.twitter.com/Rate-limiting?SearchFor=limit&sp=1

Twitter Support :: What are hashtags (the "#" symbol)? 2009. [online] [viitattu: 7.11.2009] http://help.twitter.com/forums/10711/entries/49309

Wikipedia: Web 2.0. [online] [viitattu: 8.11]
http://en.wikipedia.org/wiki/Web_2.0

Zervaas, Quentin 2008. Practical Web 2.0 Applications with PHP. Apress. Berkeley.