CHANNELS

Kanava yhteisölliseen viestintään

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2010
Marko Korhonen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

KORHONEN, MARKO:

Channels
Kanava yhteisölliseen viestintään

Ohjelmistotekniikan opinnäytetyö, 46 sivua

Kevät 2010

TIIVISTELMÄ

Tämä opinnäytetyö käsittelee yhteisölliseen viestintään suunnatun Channels-sovelluksen suunnittelua ja toteutusta. Ensiksi opinnäytetyössä käsitellään sosiaalisen median taustoja ja sitä kautta lähtökohtia sovelluksen toiminnalle. Channels-sovelluksena on tarkoitettu helpottamaan sisällön tuottamista ja läpinäkyvää viestintää erikokoisissa organisaatioissa hyödyntäen monista sosiaalisista verkkopalveluista tuttuja toimintatapoja ja työkaluja.

Sovellusta on kehitetty Datafisher Oy:n omien ja sen asiakkaiden tarpeita ajatellen. Sovelluksen loppukäyttäjät voivat olla käyttäjäorganisaation sisäisiä, ulkoisia tai molempia. Channels on vaihtoehto erilaisille organiaatioissa käytössä oleville intranet- ja extranetsovelluksille, joiden ongelmana ovat yleensä sisällön tuottamisen vaikeus tai käyttämistä hankaloittavat käyttäjäoikeudet. Channels voi olla myös organisaation yhteisöpalvelu olemassa olevien palveluiden ohessa. Channels-sovelluksessa käytettävyys ja käyttäjää miellyttävä käyttöliittymä ovat sovelluksen tärkeimpiä osa-aluita ja näihin seikkoihin on myös kiinnitetty enemmän huomiota toteutuksessa. Toteutuksessa hyödynnettiin PHP-pohjaista Zend Framework-ohjelmakirjastoa ja kehityksen avuksi otettiin useita suunnittelumalleja, jotka ohjasivat varsinaista kehitystyötä ja antavat vankan pohjan jatkokehitykselle.

Sovellus saavutti tämän opinäytetyön aikana suurimman osan tavoitteista, jotka sille alussa määriteltiin. Osa sovelluksen toiminnallisuuksista muuttui työn aikana sekä joitain ominaisuuksia jäi pois. Jatkuvan ideoinnin ja palautteen perusteella sovellukseen tuli myös paljon uusia ominaisuuksia. Ajan puutteen takia paras mahdollinen testausympäristö jäi toteuttamatta, kun sovelluksen perusominaisuuksien toteuttaminen oli etusijalla. Tämä seikka on kuitenkin syytä huomioida jatkossa tarkemmin. Channels-sovelluksen kehitys jatkuu edelleen, ja tarkoitus on saada otettua se käyttöön Datafisher Oy:ssä sisäisesti sekä useissa eri asiakasorganisaatioissa.

Avainsanat: kanavat, sisällöt, sosiaalinen media, web 2.0, php, sovelluskehys, sovellus, ohjelmointi

Lahti University of Applied Sciences
Degree Programme in Information Technology

KORHONEN, MARKO: Channels – a platform for community communication

Bachelor's Thesis in Information Technology, 46 pages

Spring 2010

ABSTRACT

The aim of this thesis was to design and implement an application called Channels which emphasizes communication and content creation on the community level. Community-based collaboration in content creation and sharing is shaping the tools and services on the web. The Channels application is strongly influenced by concepts such as social media and Web 2.0.

This application was designed to serve both Datafisher's own and its customers' needs. Channels provide an alternative to existing intranet and extranet applications used widely by companies today. It can also co-exist with these systems and act as a community platform for more informal content and communication. The basic idea of the application is to let users create and share content easily. Content is published into user defined channels. Channels create a context around the content published in them.

The application was developed with PHP-based Zend Framework, which provides the MVC model approach and enables rapid software development. Many design patterns were also used as guidelines for programming. Leaning towards frameworks and design patterns is beneficial for successful software development.

Most of the initial goals had been met at the time of writing this thesis. Some of the features were discarded and some were modified. Testing was one aspect which was disregarded in favor of other aspects and will need more attention in the future. Development of this application is still active and the goal is to get Channels to production use by Datafisher and by its customers in the near future.

Key words: channels, content, social media, web 2.0, php, framework, application, programming

SISÄLLYS

SANASTO JA LYHENTEET

Active Directory - Käyttäjätietokanta

AJAX - Asynchronous JavaScript And XML

ASP – Application Service Provider, palveluntarjoja

Atom - XML-kieleen pohjaava standardi verkkosyötteille ja tiedon syndikoinnille

Backend – Taustajärjestelmä

Best practices – Parhaat käytänteet

Blogi – ns. verkkopäiväkirja

Cache – Välimuistitus

Design patterns – Suunnittelumallit

Facebook – Suosittu sosiaalinen verkkopalvelu

Frontend – Edustajärjestelmä (esim. selaimessa ajettava JavaScipt)

HTTP - HyperText Transfer Protocol

IDE – Integrated Development Environment

Layout – Yleinen sivupohja

LDAP - Lightweight Directory Access Protocol

Lucene - Hakukoneindeksi

MVC – Model-View-Controller

OpenID – Osoitepohjainen käyttäjätunnistus

PHP – rekursiivinen akronyymi: Hypertext Preprocessor

RSS – XML-pohjainen verkkosyötemuoto

SaaS – Software as a Service, sovellus palveluna

Tagi – Asiasana, hakusana

TDD – Test Driven Development

Template – Sivukohtainen näkymä

URL rewriting – Palvelimen päässä tapahtuva osoitteiden uudelleenkirjoitus

1 JOHDANTO

Internetissä on useita erilaisia palveluita ja työkaluja, joilla käyttäjät voivat luoda erimuotoista sisältöä verkkoon. Samalla nämä palvelut luovat käyttäjien välisiä verkostoja erilaisten kiinnostuksen kohteiden mukaan sekä tarjoavat käyttäjille työkaluja keskinäiseen vuorovaikutukseen. Nämä ns. sosiaaliset verkkopalvelut ovat tehneet tuloa viime aikoina myös yrityksiin ja organisaatioihin. On kuitenkin ollut epäselvää, miten yritykset ja organisaatiot voisivat sosiaalisia verkkopalveluita hyödyntää.

Viime vuosien aikana internet on muuttunut pelkästään asioita esittävästä mediasta vuorovaikutteisemmaksi. Ensin alkoi tulla käyttöön työkaluja, joilla käyttäjät pystyivät itse luomaan sisältöä verkkoon. Sen jälkeen vuorovaikutus mahdollistui yhden käyttäjän tuottaman sisällön ja muiden käyttäjien välillä. Tämä ilmeni esimerkiksi kommentointi- ja arvostelutyökalujen sekä sosiaalisten kirjanmerkkipalveluiden tulemiseksi kiinteäksi osaksi verkossa tuotettua sisältöä.

Monet organisaatioissa käytössä olevat sovellukset ovat luonteeltaan muodollisia ja eivät tarjoa yhteisöllistä ulottuvuutta. Samalla monimutkaiset käyttäjäoikeudet hankaloittavat sisällön tuottamista ja saavutettavuutta. Channels-sovellus on suunnattu tähän tarpeeseen. Channels voi olla joko vaihtoehto intranet- ja extranetsovelluksille tai sitten se voi olla organisaation yhteisöllinen verkkopalvelu näiden muiden sovelluksien rinnalla. Channels-sovelluksen tarkoituksena on helppo ja mutkaton sisällön tuottaminen ja jakaminen. Samalla organisaatiossa toimivien ihmisten viestinnästä tulee läpinäkyvämpää ja saavutettavampaa.

Kun tarkastellaan perinteisiä sisältöjä, kuten uutisia tai TV-ohjelmia, niin ne ovat luonteeltaan jatkuvaa tietovirtaa, joita on kuitenkin mahdollista suodattaa erilaisten määreiden avulla tai tarkastella erilaisista näkökulmista. Uutisissa suodatusta ovat esimerkiksi uutiskategoriat, kuten urheilu tai ulkomaan uutiset. TV-kanavan valinta taas vaikuttaa suoraan siihen, minkälaista ohjelmaa käyttäjä näkee televisiosta.

Internetissä julkaistavat sisällöt myös liittyvät johonkin tietovirtaan, jota tarkastellaan valitusta näkökulmasta. Internetin luonne on mahdollistanut lukemattoman määrän erilaisia tapoja katsella sisältöä. Esimerkiksi blogi kuvastaa kirjoittajansa näkökulmaa, jota voi yleensä suodattaa vielä aiheiden tai ajankohdan mukaan. RSS-virrat antavat lukevalle osapuolelle päätäntävallan sisällön esitystavasta. Twitter on tuonut uutena tapana mikrosyötteet, joissa käyttäjät voivat luoda ja jakaa helposti lyhyitä viestejä muille. Channels lainaa monia hyviä ajatuksia monista eri verkkopalveluista ja yhdistää ne käytettäväksi kokonaisuudeksi, jota erilaiset organisaatiot voivat hyödyntää omien yhteisöllisten sisältökanavien luomiseksi.

Tämän työn tavoitteena on määritellä ja toteuttaa helppo ja joustava sovellus (Channels) yhteisölliseen sisällöntuotantoon ja viestintään sekä pohtia sovelluksen mahdollisia ansaintamalleja. Channels antaa käyttäjälle mahdollisuuden tuottaa omaa materiaalia sekä linkittää olemassaolevien verkkopalveluiden sisältöjä, kuten karttoja ja videoita. Näin voidaan hyödyntää myös muissa verkkopalveluissa jo tuotettua sisältöä, ilman että sitä tarvitaan luoda uudestaan.

Sisällöt tuotetaan aihealueittain kanaviin, joiden avulla sisältöjen seuraaminen on helppoa. Työkalut ja niiden käyttötavat ovat itsessään tuttuja käyttäjille monista muista yhteyksistä. Esim. kommentit, tagit (asiasanat), arvosteleminen ja sosiaalisten kirjanmerkkipalveluiden käyttäminen on mahdollista. Kanavat voivat myös toimia pieninä yhteisöinä, joissa useampi käyttäjä tuottaa sisältöä samaan kanavaan.

Tarkoitus ei ollut keksiä uutta innovaatiota vaan jotain, joka yhdistää olemassaolevia tapoja ja työkaluja käyttäjiä miellyttävällä tavalla. Tarkoitus ei myöskään ole kilpailla olemassa olevien kuvapankkipalveluiden, blogisovellusten tai sosiaalisten verkostosovellusten kanssa.

Työ on rajattu seuraavasti:

- Sovelluksen ajatusmaailmaa esitellään yleisellä tasolla.
- Työssä pohditaan sovelluksen ansaintamalleja.
- Teknisestä toteutuksesta esitellään valittuja malleja sekä teknologisia ratkaisuja yleisellä tasolla.
- Sovelluksen palvelinarkkitehtuuri esitellään.
- Työssä ei kuvata yksityiskohtaisesti yksittäisiä teknisiä ratkaisuja.
- Työssä ei avata ohjelmakoodin näkökulmasta toiminnallisuuksia.

Sovelluksesta on tarkoitus tuottaa kaupalliseen käyttöön sopiva versio Datafisher Oy:n käyttöön. Datafisher on oppimisen ja oppimista tukevan teknologian kärkiasiantuntijoita Suomessa. Datafisher on vuonna 1998 Savonlinnassa perustettu yritys, joka toimii Vallilassa, Helsingissä, ja siinä työskentelee tällä hetkellä 9 henkilöä.

Yrityksen arvioitu liikevaihto vuonna 2009 on n. 0,8 miljoonaa euroa. Yrityksen liiketoiminta on kannattavaa ja luottoluokitus on AAA. Datafisherin asiakkaita ovat mm. StoraEnso, Finnair, Microsoft Suomi, Suomen Posti, Coloplast AS, YLE, Opetushallitus, Ulkoasianministeriö, Raja- ja merivartiosto sekä Helsingin yliopisto. (Datafisher, 2009.)

2 LÄHTÖKOHDAT JA TAUSTA

2.1 Käsitteet sovelluksen taustalla

Channels on tämän hetken teknologisten trendien mukainen verkkosovellus. Siinä on monia muista verkkopalveluista tuttuja ja hyväksi havaittuja ominaisuuksia. Web 2.0 ja sosiaalinen media liittyvät käsitteellisesti sovellukseen sekä sen toimintatapojen että teknologisten ratkaisujensa puolesta.

2.2 Web 2.0

Tim O'Reillyn keksimä käsite "Web 2.0" on viime vuosien trenditermi, jolla on pyritty kuvaamaan hieman kaikkea, mitä internetissä on tapahtunut 2000-luvulla ja mikä on ollut jollain tasolla uutta. Internet ei kuitenkaan ole muuttunut tai päivittynyt millään tavalla, joka antaisi aihetta antaa sille uuden versionumeron. Lähinnä on kyse siitä, että kasvaneet tiedonsiirtonopeudet ja tallennuskapasiteetti ovat mahdollistaneet uudenlaisia tapoja käyttäjille tuottaa sisältöä ja jakaa sitä toisille.

Yleisimmät käsitykset Web 2.0:sta tuovat ihmisten mieliin käsitteitä kuten blogit, wikit, sosiaalinen media ja jakaminen. Lopulta jokainen saa käsittää tai selittää termin omalla tavallaan, ja tässä työssä Web 2.0 tarkoittaa juurikin käyttäjälähtöistä sisällöntuotantoa ja vaivatonta jakamista. Loppu on vain teknisiä yksityiskohtia, kuinka tämä kaikki tapahtuu.

Mikko Tirronen kuvaa kirjassaan "Web 2.0" termiä ja sen ehkäpä harhaanjohtavaa nimeä seuraavasti:

> *"Omien tuotosten helppo jakaminen ja palautteen saaminen houkuttelevat yhä useampia kerääntymään yhteen verkossa. Innostuneet jäsenet muodostavat nopeasti aktiivisen sosiaalisen verkon, joka houkuttelee yhä vain uusia ihmisiä. Web 2.0 – termiä kritisoivat toteavat, että näin asioiden pitikin edetä. Verkon piti yhdistää ihmisiä tavalla, jota ei ole ennen nähty. Heidän mukaansa ei ole mielekästä nimetä verkkoa uudestaan*

> *vain siksi, että se on alkanut muodostua juuri sellaiseksi kokonaisuudeksi, joksi sitä alkujaan kaavailtiinkin. Web 2.0 on vain Web 1.0:n luonnollista kehittymistä, ei mitään suurempaa tai pienempää.”*

(Tirronen 2008, 1.)

2.3 Sosiaalinen media

Sosiaalinen media sisältää paljon metatason ideoita sekä teknologiaa. Idean tasolla sosiaalisuus kertoo siitä, että kyseessä on ihmisten välisestä toiminnasta (tiedostojen jakaminen, mielipiteiden kertominen). Media taas viittaa sisältöön digitaalisessa julkaisukanavassa, eli kyseessä on käyttäjien itsensä tuottamaa sisältöä toisille käyttäjille sekä itselleen.

KUVIO 1. Sosiaalisen median kenttä (Cavazza, 2008)

Kun tarkastellaan teknologiaa sosiaalisen median taustalla, niin tullaan väistämättä tutuiksi blogien, wikien, ajaxin ja muiden uusien sovellusten ja

tekniikoiden kanssa. Fred Cavazzan tekemässä kuviossa (KUVIO 1) on esitelty sosiaalisen media eri palveluita jaoteltuna näkökulman mukaan, kuten jakaminen (Share) tai sosiaaliset verkostot (Social Networks).

Yhteisöllisyys on sosiaalisen median ydin. Esimerkiksi joku yhteinen kiinnostuksen kohde voi luoda verkossa ympärilleen yhteisön, joka ylittää maantieteelliset tai muut perinteiset rajat. Tämänlainen yhteisö kuluttaa ja tuottaa sisältöä liittyen kiinnostuksen kohteeseensa. Jokin aihe siis yhdistää ihmiset virtuaaliseksi yhteisöksi ja aktivoi heidät tuottamaan sisältöä toisilleen. Näin nämä ihmiset muuttuvat samalla passiivisista kuluttajista (consumer) tuottaja-kuluttajiksi (prosumer).

2.4 Teknologisten ratkaisujen valinta

2.4.1 Valintakriteerit

Kaikille sovelluksen toteutukseen valituille teknologioille yhteinen kriteeri oli ilmaisuus sekä avoimuus. Ns. avoimen lähdekoodin (open source) ohjelmistot täyttävät nämä molemmat ehdot. Toinen tärkeä kriteeri oli se, että valituista teknologioista löytyy paljon laadukasta dokumentaatiota sekä paljon käyttäjälähtöistä materiaalia. Tällaista materiaalia ovat esimerkiksi tutoriaalit ja esimerkkiohjelmakoodit. Hyvä dokumentaatio ja siihen liittyvä muu materiaali nopeuttavat omalta osaltaan sovelluskehitystä. On myös helppoa ottaa yhteyttä muihin kehittäjiin esimerkiksi keskustelupalstojen kautta, kun kysymyksiä tai ongelmia tulee vastaan. Moni ongelma on jo luultavasti ratkaistu, joten on turhaa tuhlata siihen itse aikaa.

Tärkeää oli myös, että valittujen ohjelmistojen lisenssit on tarpeeksi joustavia etenkin sovelluksen kaupallista käyttöä ajatellen. Lisenssit ovat yleensä muodostettu niin, että ei-kaupallisessa tai käyttäjille ilmaisessa palvelussa kyseinen ohjelmisto on ilmainen, kun taas kaupallisessa tai yritysten sisäisessä käytössä lisenssi voi olla maksullinen.

2.4.2 Taustajärjestelmät

Taustajärjestelmät (backend) ovat palvelimen päässä toimivia sovelluksen osia. PHP-ohjelmointikieli sekä tietokantapalvelin ovat esimerkiksi palvelimella toimivia taustajärjestelmän osia.

Ohjelmointikieleksi valittiin siis PHP ja toteutusta varten valittiin Zend Framework-ohjelmakirjasto. Valinnan pohjana toimi osaltaan oma seminaarityö, joka selvitti muutamien tunnettujen PHP-pohjaisten ohjelmakirjastojen eroja. Tietokannaksi valittiin MySQL siitä syystä, että PHP-kielellä itsellään ja sillä toteutetuilla kirjastoilla on pitkäaikainen tuki MySQL-tietokannalle.

Sovelluksella on myös muita vaatimuksia taustajärjestelmältä, kuten videoiden kääntäminen erilaisista video formaateista Flash-videoiksi sekä Flash-videon streamaus webkamerasta palvelimelle. Sovellus tarvitsee myös ajastettuja tehtäviä erinäisiin toimintoihin. Ajastetuille tehtäville löytyy tuki niin Windows- kuin Unix-ympäristöistä. Ajastettujen tehtävien avulla voidaan toteuttaa esimerkiksi ylläpidollisia toimia järjestelmässä. Näitä voivat olla esimerkiksi erilaiset puhdistustoiminnot sekä tietokannan ja hakuindeksin optimointi.

2.4.3 Edustaosa

Sovelluksen edustaosana (frontend) toimii selain. Selain luo esityksen käyttöliittymästä ja sen toiminnasta palvelimelta tulleen vastauksen perusteella. Sovelluksen toimintaa edustaosassa voidaan myös ohjelmoida JavaScript-kielellä. Monista eri JavaScript-kirjastoista valittiin jQuery sen monipuolisten ominaisuuksien sekä helppokäyttöisyyden vuoksi.

JavaScript mahdollistaa myös AJAX-teknologian käytön. Sen avulla voidaan tehdä osittaisia kutsuja palvelimelle ilman, että koko sivua ladataan uusiksi selaimeen. Tämän avulla voidaan toteuttaa vuorovaikutteisempia käyttöliittymiä. Channels-sovelluksessa käytetään AJAX-teknologiaa yhdessä Taconite-tekniikan kanssa. Siinä palvelimelta palautetaan XML-dokumentti käyttäjän toimintojen

ja/tai syötteen perusteella. XML-dokumentti kertoo, miten käyttöliittymää muokataan dynaamisesti. Esimerkiksi käyttäjän toimi voi saada aikaan muutoksia sivun sisällöissä, dialogin tai virheilmoituksen ilman sivun uudelleen lataamista.

2.5 Vertailu muihin palveluihin

Channels on yhteisöllinen palvelu, jonka avulla käyttäjät voivat luoda ja jakaa sisältöä keskenään. Jos sitä vertaa esim. Facebook-yhteisöpalveluun, niin siitä löytää helposti monia yhtäläisyyksiä. Esimerkiksi molemmissa palveluissa on tärkeää, että käyttäjä voi luoda ja jakaa sisältöä helposti valitsemalleen kohderyhmälle.

Eroavaisuus Facebookin tai MySpacen kaltaisiin palveluihin on siinä, että Channels-palvelussa kanavat ovat toiminnan keskipiste, jonka ympärille sisällöt ja toiminta keskittyvät. Facebookissa ja MySpacessa taas käyttäjä on kaiken ydin.

2.6 Kohderyhmät ja ansaintamallit

Pääasiallisesti sovellus on suunniteltu erilaisten yritysten tai organisaatioiden käytettäväksi sisäisesti. Tällaisessa tapauksessa sovellus on kaupallinen tuote. Toinen vaihtoehto on luoda avoin palvelu kaikille käyttäjille, jolloin ansainta tapahtuu esim. mainonnan avulla.

Sosiaalisten verkkopalveluiden hyödyistä yrityksille ja organisaatioille on vielä vähän tutkimuksellista tietoa. Ainoa mitä voidaan todeta tässä vaiheessa, on se että vahvasti sosiaalisissa mediossa olevat brändit ovat myös kehittäneet eniten tuottoa. (ENGAGEMENTdb, 2009)

2.6.1 Yritykset ja organisaatiot

Channels voi toimia yrityksessa tai organisaatiossa esimerkiksi yleisenä viestintävälineenä, jolla on yhteisöllinen painotus. Toisaalta Channels voi toimia

henkilöstölle suunnattujen ohjemateriaalien jakelukanavana. Sisältö itsessään määrittelee suureksi osaksi sovelluksen käyttötarkoituksen.

Asiakas (yritys tai organisaatio) voi ostaa Channels-sovelluksen käyttöönsä palveluna. Tällaisessa tapauksessa ansaintamallina on kaupallinen palvelulisenssi. Lisenssin hinta voi määräytyä esimerkiksi käyttäjämäärän mukaan aikaan sidottuna tai kertahankintana, johon liittyy ylläpito.

Paras vaihtoehto ylläpidon puolesta olisi myydä sovellusta ASP-palveluna (Application Service Provider), jossa sovellusta tarjotaan verkon yli palveluna asiakkaalle. Silloin sovellus on täysin tarjoajan (provider) hallinnassa ja näin ollen sovelluksen ylläpito helpottuu. Sovellusten tarjoamisesta verkon yli käytetään yleisesti myös termiä SaaS (Software as a Service, sovellus palveluna).

Kaupallinen käyttö vaikuttaa osaan toiminnallisuuksista. Esimerkiksi Google Maps –rajapinnan käyttö yrityksen sisäisessä käytössä tai käyttäjille maksullisessa palvelussa vaatii kaupallisen lisenssin. Tämän kaltaiset kulut on huomioitava lisenssin hinnoittelussa.

Yrityksen tai organisaation kannattaa hankkia Channels -sovellus, jotta asiakkaan työntekijät voivat helposti ja tuottavasti tuottaa ja jakaa sisältöä joko keskenään ja/tai omien asiakkaidensa kanssa. Sisältö voi liittyä työtehtäviin ja muuten työyhteisöä koskeviin asioihin. Kaiken tämän on tarkoitus helpottaa viestintää työntekijöiden kesken sekä yrityksen ja työntekijöiden kesken. Yhteisöllisyys ja vapaa tiedon jakaminen ja sen läpinäkyvyys hyödyttävät kaikkia osapuolia ja parantavat näin työn tuloksia sekä motivaatiota.

2.6.2 Avoin palvelu

Sovellus voidaan toteuttaa myös kaikille avoimena palveluna (vrt. Facebook, Twitter tms.). Tässä tapauksessa riskinä on se, että kannattavaa ansaintamallia on vaikea keksiä. Yleisin ansaintamalli avoimissa palveluissa on mainonta. Onnistuneen mainonnan toteuttaminen vaatii kuitenkin oman suunnittelunsa ja sen

toteuttamiseen on tarjolla monia eri palveluita. Toinen yleinen ansaintakeino on ns. premium-palvelut, joita ostamalla käyttäjä saa lisää ominaisuuksia palveluunsa.

Avoimen ja käyttäjälle ilmaisen palvelun etuna on, että esimerkiksi Google Maps-rajapinnan käyttö on ilmaista. Sama ehto koskee useita palveluita internetissä, joten tämän vaihtoehdon etuna on hyödynnettävien palveluiden melkein rajaton määrä.

2.6.3 Markkinatilanne ja kilpailu

Sosiaaliset verkkosovellukset ovat jo tulleet yrityksiin ja organisaatioihin blogien, Wikien ja vastaavien sovelluksien myötä. Monet yritykset miettivät tällä hetkellä, miten ne voisivat hyödyntää sosiaalisia verkkosovelluksia. Myös monet olemassa olevat sovellukset ovat lisänneet ominaisuuksiinsa sosiaalisia ulottuvaisuuksia.

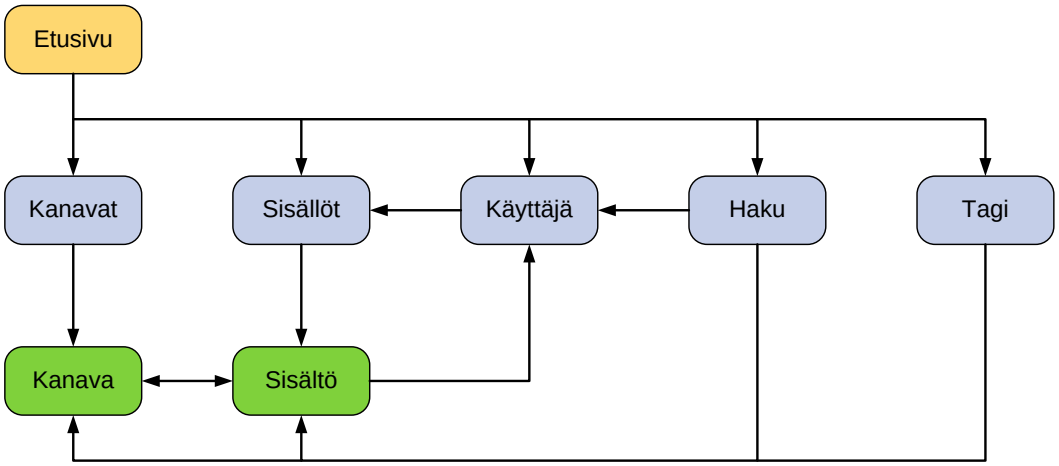
Sovellus ei suoraan pyri kilpailemaan olemassaolevien tuotteiden tai palveluiden kanssa. Tarkoitus on tarjota vaihtoehtoinen tapa hyödyntää monista sosiaalisista verkkopalveluista tuttuja työkaluja ja palveluita yhdestä palvelusta käsin. Näin ollen Channels myös hyötyy muiden olemassaolevien palveluiden olemassaolosta.

3 MÄÄRITTELY

3.1 Konsepti

Sovelluksen tarkoituksena on antaa käyttäjälle mahdollisuus luoda erimuotoisia sisältöjä helposti. Toiminta-ajatus on yksinkertainen. Sisällöt luodaan kanaviin, jotka määrittelevät aihealueen sisällöille. Käyttäjille tarjotaan useita eri tapoja selata ja löytää sisältöä.

Käyttäjät voivat myös luoda halutessaan yhteisön kanavan ympärille. Yhteisöllisyys toteutuu vähintään niin, että muut käyttäjät liittyvät kanavan jäseniksi ja/tai kommentoivat sisältöjä. Yhteisöllisyys voi laajeta käyttäjien saadessa laajempia oikeuksia kanavaan, jolloin he itse alkavat luomaan sisältöä kyseiseen kanavaan.

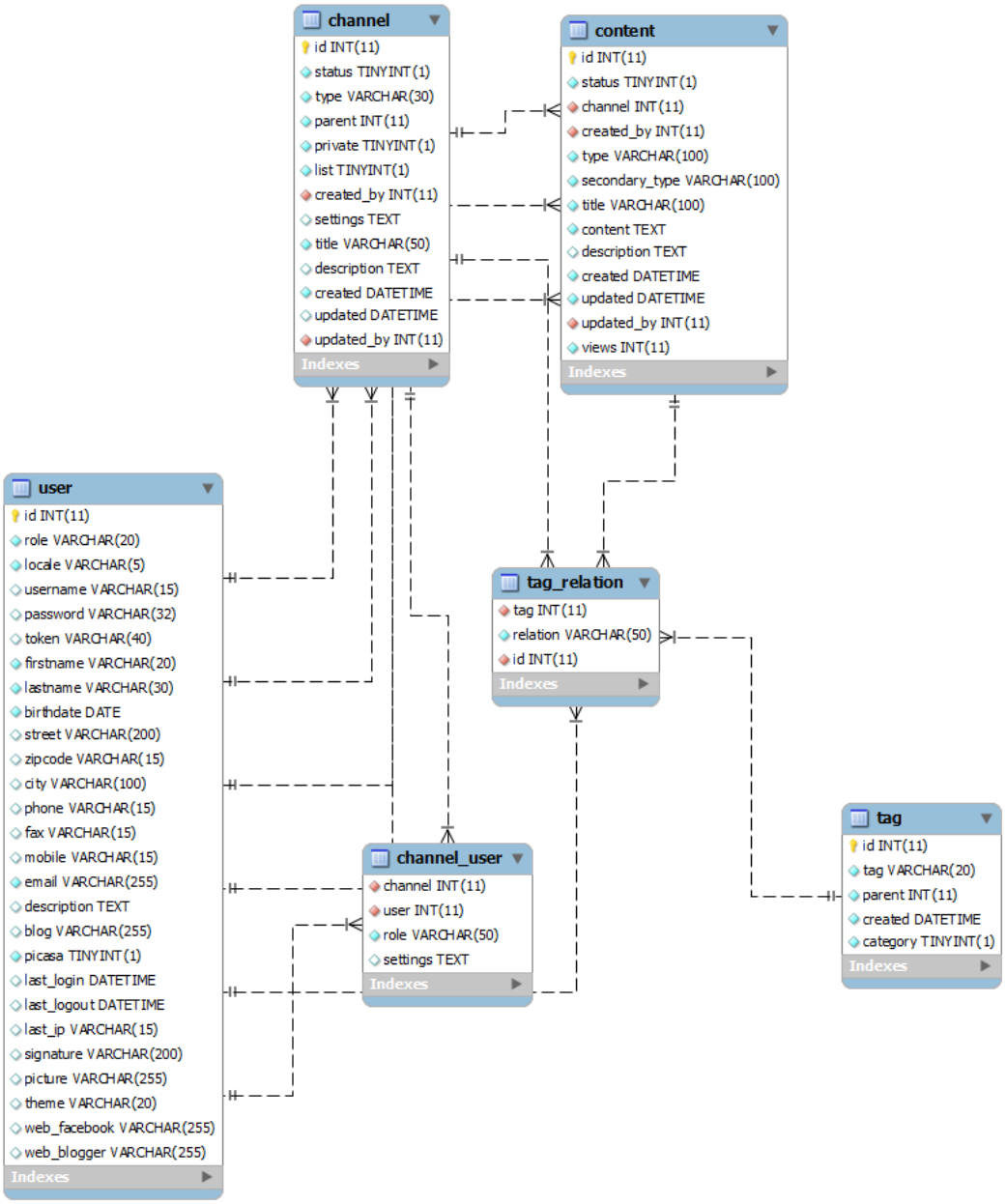


KUVIO 2. Sovelluksen konseptuaalinen rakenne

Ylläolevassa kuviossa (KUVIO 2) nähdään, miten kanavat ja sisällöt linkittyvät toisiinsa. Samalla kuviossa nähdään, mitä eri reittejä luotuun sisältöön on. Esimerkiksi yksi sisältö löytyy kanavasta, sen lisänneen käyttäjän profiilin kautta,

siihen liitettyjen tagien kautta tai hakukoneella. Kuviossa ei esitetä liitännäisiä, kuten kommentteja ja arvosteluja.

Alla on esiteltynä palvelun tietokannan relaatiomalli (KUVIO 3). Se kuvaa keskeisten tietueiden suhdetta toisiinsa. Mallissa ei ole kuitenkaan kaikkia sovelluksen tauluja.



KUVIO 3. Tietokannan relaatiomalli

Kuviossa nähdään, miten sisällöt ja kanavat linkittyvät toisiinsa sekä miten käyttäjä linkittää sisältöihin tekijänä sekä kanaviin omistajana tai jäsenenä. Liitännäistiedosta on esimerkkinä tagit, jotka liitetään sisältöihin ja kanaviin liitostaulun kautta.

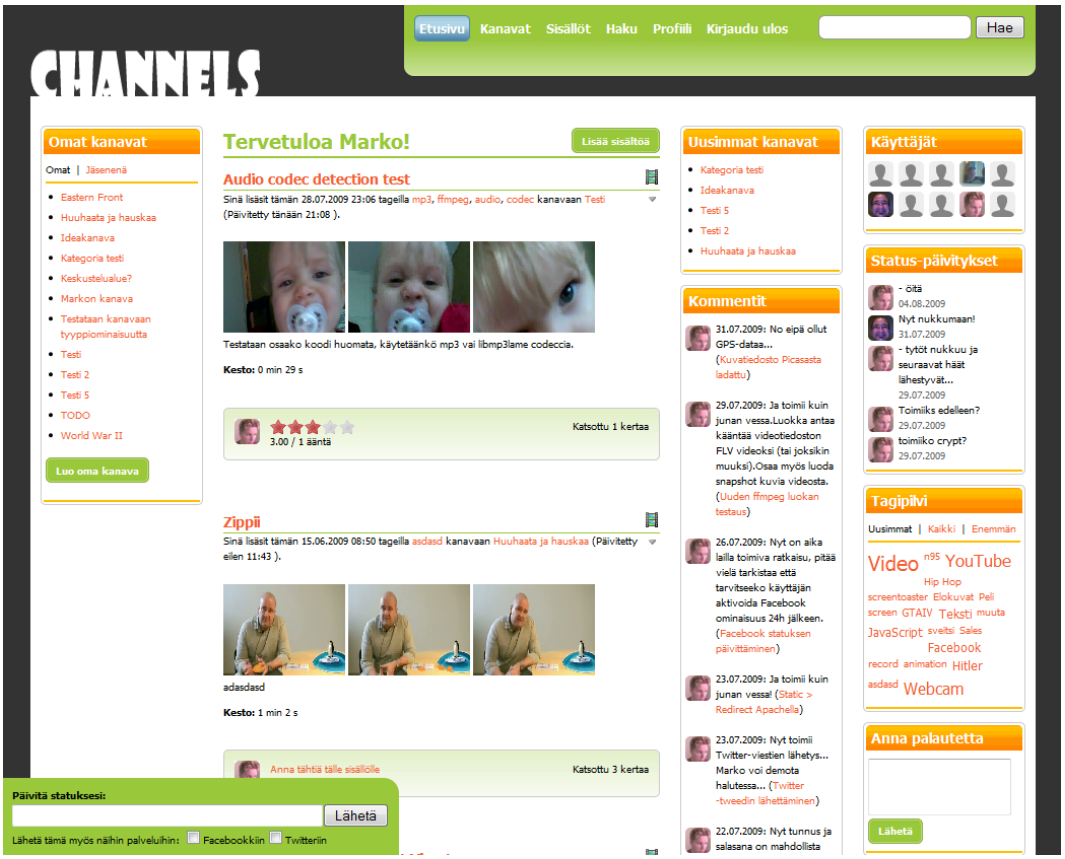
3.1.1 Käyttöliittymä

Käyttöliittymän visuaalisen ilmeen on oltava selkeä ja yksinkertainen. Käyttöliittymän on ohjattava käyttäjää löytämään etsimänsä sisältö tai toiminnallisuudet helposti. Navigoinnissa käyttäjää helpottavat kanaville ja sisällöille annetut asiasanat (tagit) sekä mahdollisuus suodattaa sisältöjä listauksissa erilaisten suodattimien avulla. Suodattimina toimivat erilaiset laskurit, kuten kommenttien lukumäärä. Sisältöjä voi myös suodattaa sisältötyyppien mukaan.

Sovelluksen käyttöliittymämalli pohjautuu perinteisiin palstamaisiin rakenteisiin. Alla listataan käyttöliittymän yleisimmät ominaisuudet ja toiminnallisuudet.

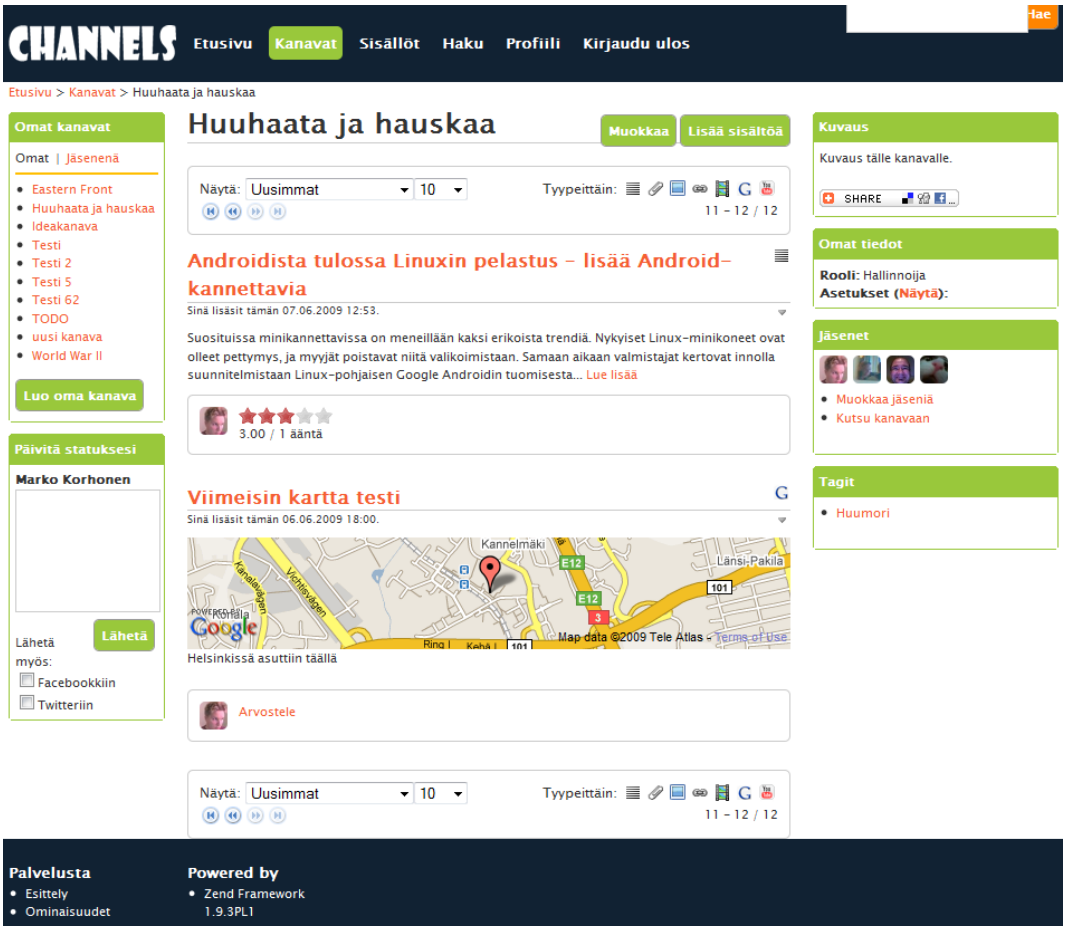
Käyttöliittymän pääelementit ovat seuraavat:

- yläosio (header)
    - logo
    - päänavigaatio
    - haku
    - murupolku
- sisältöalue (2 - 4 palstaa, body/content)
    - vasen palsta, käyttäjän omat kanavat
    - keskimmäinen palsta pääasialliselle sisällölle
    - oikeat 1 - 2 palstaa, eri nostoja sivun sisältöön liittyen
- alaosio (footer)
    - linkit
    - ohjeet
    - yhteystiedot
    - tekijänoikeudet yms.



KUVIO 4. Palvelun etusivu

Yllä olevassa kuvakaappauksessa (KUVIO 4) esitellään sovellukselle muodostettu etusivu. Käyttöliittymässä päänavigaatio on sijoitettu yhdessä haun kanssa oikeaan yläkulmaan. Varsinainen sisältöalue on jaettu neljään palstaan. Ensimmäisessä palstassa on jokaisella sivulla toistuvat elementit, esim. omat kanavat. Toinen palsta toimii pääsisältöalueena, ja siinä esitellään uusimpia sisältöjä. Kolmas ja neljäs palsta tuovat esiin erilaisia nostoja, kuten kommentteja, käyttäjien viestejä ja tagipilven. Tärkeää on myös, että käyttöliittymästä on myös mahdollista tehdä erilaisia visuaalisia versioita eli teemoja esimerkiksi eri asiakkaille (KUVIO 5).



KUVIO 5. Grid960-teema

Myös sisällön lisääminen käyttäjälle on tehtävä niin helpoksi kuin mahdollista. Käyttöliittymän pitää tukea visuaalisesti käyttäjää, kun hän lisää sisältöä. Esimerkiksi karttaa, linkkejä tai YouTube-videota lisätessä käyttäjälle näytetään jo esikatselussa kyseinen materiaali ennen tallennusta.

3.1.2 Kanavat

Kanavat ovat sovelluksen oletusluokittelu sisällöille. Kaikki sisällöt kuuluvat johonkin kanavaan. Kanavien asiasanat taas toimivat kanavia luokittelevina kategorioina. Kanavia voi luoda myös toisten kanavien alle, jolloin voidaan luoda osa-aihealueita. Koska kanavat ovat tietovirtoja, ne voidaan myös viedä RSS-verkkosyötteinä erilaisiin lukijasovelluksiin. Käyttäjä voi siis seurata kanavaa valitsemansa syötelukijan avulla ilman, että hänen tarvitsee käydä sovelluksessa.

Kanavan tyyppi määrittää sen, miten kanavan sisältöjä oletuksena katsellaan. Oletustyyppi listaa sisältöjä blogimaisesti eli uusin sisältö ensin. Book-kanava taas toimii itsessään hierarkisena kirjana sovelluksen sisällä. Tiedostoalue-kanava luo yksinkertaisen tiedostolistan ja salliikin vain tiedostojen lisäämisen kanavaan.

Kanavalla on aina omistaja eli käyttäjä, joka on kanavan luonut. Toisten käyttäjien luomiin kanaviin on myös mahdollista liittyä, jolloin käyttäjä tulee jäseneksi kyseiseen kanavaan. Jäsenyys tarkoittaa kyseisen käyttäjän kiinnostusta kanavan aihealueeseen. Omistaja voi myös antaa kanavan jäsenille rooleja, joiden avulla jäsenet voivat saada lisäoikeuksia kyseisen kanavan ja sen sisältöjen suhteen. Kanavan voi myös määrittää ns. vapaaksi kanavaksi, jolloin kaikilla kirjautuneilla käyttäjillä on oikeus lisätä sisältöä kyseiseen kanavaan.

Kanavan jäsenellä voi olla joku seuraavista rooleista: Jäsen (member) voi lukea ja kommentoida sisältöjä sekä tilata sähköpostiin viestin, jos kanavaan tulee uusia sisältöjä tai kommentteja. Sisällöntuottaja (contributor) voi tuottaa uutta sisältöä kanavaan ja voi muokata omia sisältöjään. Editori (editor) saa tuottaa uutta sisältöä kanavaan ja hän voi muokata omia ja muiden sisältöjä. Hallinnoijalla (samat oikeudet kuin omistajalla) on kaikki oikeudet kanavan sisältöihin, mutta hän voi myös hallinnoida kanavan asetuksia ja käyttäjiä. Allaolevassa taulukossa (TAULUKKO 1) esitetään perusoikeudet eri rooleille kanavissa.

TAULUKKO 1. Kanavan ns. Access Control List eli pääsylista

Oikeus	**Jäsen**	**Sisällöntuottaja**	**Editori**	**Hallinnoija***
Kanava				
Luku	x	x	x	x
Muokkaus				x
Käyttäjät				x
Kanavan sisällöt				
Luku	x	x	x	x
Lisäys		x	x	x
Omien muokkaus		x	x	x
Muokkaus/poisto			x	x
Kommentointi	x	x	x	x

*) Myös omistaja

3.1.3 Sisällöt

Käyttäjä voi luoda kanavaan erityyppisiä sisältöjä. Luotuja sisältöjä on voitava tarkastella muutenkin kuin vain kanavien kautta. Sovellukseen on mahdollista tuottaa myös uusia sisältötyyppejä tarpeen mukaan. Sovelluksen sisäiset rajapinnat mahdollistavat uusien moduulien määrittelemät uudet sisältötyypit. Näin järjestelmän toiminnallisuutta voidaan laajentaa uusien moduulien avulla.

Kaikki sisältötyypit sisältävät yleisiä ominaisuuksia, kuten otsikon, kuvaustekstin ja asiasanat. Sisältötyyppi taas määrittää sen, miten kyseinen sisältö lisätään kanavaa sekä miten sisältö esitetään käyttäjälle. Sisällöt on jaettu muutamaan päätyyppiin niiden luonteen mukaan. Käyttäjälle nämä tyypit esitellään käyttöliittymässä toiminnan kautta: kirjoita (tekstit), linkitä (linkit), lataa (tiedostot) ja näytä kartalla (kartat).

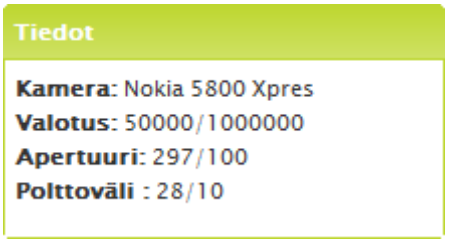
Tekstityyppinen sisältö voi olla lyhyt juttu, artikkeli tai sivu Book-muotoisessa kanavassa. Sisällön muokkaukseen löytyy editorista enemmän työkaluja kuin muista sisältötyypeistä.

Linkki on perinteinen linkki johonkin muuhun verkkosivustoon. Sovellus pitää kirjaa, kuinka monta kertaa linkki on avattu. Linkitetyn sivuston voi myös avata ilman että poistuu sovelluksesta. Tällöin sivun ylälaitaan jää palkki, jonka kautta käyttäjä voi navigoida takaisin muualle sovellukseen. Channels osaa käsitellä erikseen joidenkin verkkopalveluiden linkit, kuten Youtube, Spotify tai Slideshare. Niiden linkeille luodaan suoraan oikeanlainen esitystapa linkin takana olevaan materiaaliin. Tämä on toteutettu käyttämällä näiden palveluiden omia rajapintoja.

Kun käyttäjä on luomassa linkkiä, sovellus yrittää esitäyttää linkin luomislomakkeen kentät hakemalla kyseisen sivun HTML-datan ja etsimällä siitä otsikon ja mahdollisen kuvaustekstin. Samanlainen tekniikka on käytössä esimerkiksi Facebookin linkin jakamistoiminnossa.

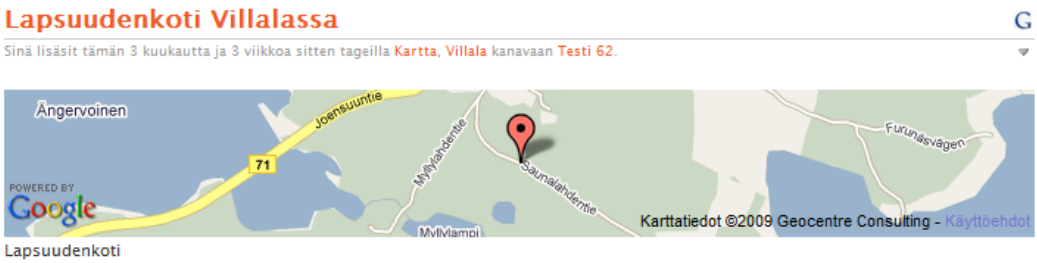
Tiedosto voi olla melkein mikä tahansa tiedostomuoto. Käyttäjä lataa haluamansa tiedoston ja antaa sille perustiedot. Sovellus osaa automaattisesti erottaa tietyt tiedostot, kuten videot, äänet ja kuvat. Silloin sovellus antaa tiedostolle toisen määreen, jonka mukaan se esitetään. Videot esitetään erillisen videosoittimen avulla ja kuvatiedostot voidaan näyttää erikokoisina kuvina. Myös äänitiedostoille luodaan soitin.

Kuvatiedosto voi myös sisältää GPS-dataa, jolloin kuvan yhteydessä voidaan näyttää myös kartta. Paikkatieto saadaan koordinaattitietona, joka voidaan sitten viedä Google Maps-rajapinnalle kartan muodostusta varten. Kuvasta voidaan myös näyttää kameran tietoja, jos niitä on saatavilla. Kuvien metadata perustuu standardiin kuvien metadatamalliin EXIF:iin (Exchangeable image file format). Yleisiä kameratietoja ovat esimerkiksi kameran merkki, malli sekä valotus- ja polttovälin arvot. (KUVIO 6).



KUVIO 6. Esimerkki valokuvan EXIF-datasta

Tässä vaiheessa sovelluksessa on mahdollista lisätä kartta määrittelemällä osoite. Haettu osoite merkitään karttaan ja kuvausteksti lisätään karttaan (KUVIO 7). Tällä hetkellä ainoa tuettu karttasovellus on Google Maps. Huomioitavaa Google Mapsin käytössä on se, että se on maksullinen kaupallisessa käytössä.



KUVIO 7. Google Maps -kartta

Materiaalipaketti on zip- tai rar-pakattu tiedosto, joka sisältää materiaalin. Materiaali voi olla esimerkiksi staattinen websivusto tai joku muu staattinen sivupaketti. Datafisher käyttää materiaalipaketti muotoa jakaakseen Adobe Captivatella tehtyjä Flash-pohjaisia opasteanimaatioita. Paketti koostuu yleensä yhdestä HTML-tiedostosta, yhdestä JavaScript-tiedostosta ja varsinaisesta opasteanimaatiotiedostosta, joka siis on Flash-muodossa. Yleensä opasteanimaatiot ovat ruutunauhoituksia tietokoneohjelmista, joihin on lisätty erilaisia ohjeita.

Videot ovat videopalveluista linkitettyjä ja sovellukseen upotettuja videoita. Tällä hetkellä on mahdollista linkittää YouTube-videoita joko käyttämällä videon omaa linkkiä tai tekemällä hakuja YouTubeen. Muita tuettuja videopalveluota ovat Viddler ja Vimeo. Videopalveluita käytetään rajapintojen avulla. YouTuben käyttämiseen löytyy valmis rajapinta Zend Frameworkista. Vimeo taas käyttää helppoa ja monipuolista rajapintaa, jossa on mahdollista tehdä kyselyitä Vimeoon ja saada vastaukset esimerkiksi XML-, JSON- tai PHP-muodossa.

Käyttäjä voi myös käyttää webkameraansa ottamaan videokuvaa, joka tallennetaan Flash-videotiedostona. Tätä varten on oma Flash-komponentti, joka mahdollistaa käyttäjän webkameran kuvan lähettämisen palvelimelle. Lähetys tapahtuu RTMP-protokollan kautta (Real Time Messaging Protocol). Palvelimella oleva streaming palvelin kuuntelee tiettyä porttia tietyssä osoitteessa, joka on annettu RTMP-protokollan käyttöön. Kun käyttäjä aloittaa nauhoituksen, Flash-komponentti lähettää kameran kuvaa palvelimelle tähän kyseiseen porttiin ja osoitteeseen. Palvelin tallentaa kameran kuvan Flash video-muodossa suoraan palvelimelle, joten se on suoraan Channels-sovelluksen käytettävissä.

3.1.4 Sisällön suodattaminen ja hakeminen

Sisältöjä voi suodattaa ja järjestellä seuraavilla eri tavoilla:

- kanavan mukaan
- sisältötyypin mukaan
- käyttäjän mukaan
- kommenttimäärän mukaan
- katselumäärän mukaan
- arvosanan mukaan
- tagin/tagien mukaan.

Sisältöjä voidaan suodattaa myös erilaisilla yhdistelmillä, esimerkiksi kaikki tietyn tyyppiset sisällöt tietyltä käyttäjältä järjestettynä arvosanan mukaan. Toinen esimerkki on suodattaa sisällöistä kaikki tiedostot, joilla löytyy joku tietty tagi.

Sisältöjä voidaan myös hakea hakukoneen avulla. Hakukoneen hauissa voidaan käyttää loogisia operaattoreita sekä jokerimerkkiä (wildcard) ja osoittaa haun tiettyyn tietoon kuten otsikkoon. Hakukonetta varten luodaan Lucene-hakuindeksi Zend_Search-luokan avulla.

Kaikissa sisältöä listaavissa näkymissä pyritään antamaan samanlainen näkymä sisällöistä, jotta eri listausnäkymät olisivat visuaalisesti linjassa toisiinsa nähden. Esimerkiksi hakukoneen hakutulos, kanavan perusnäkymä sekä yleinen sisältölistaus esittävät sisältölistaukset samannäköisinä.

3.1.5 Sisältöjen jakaminen

Sisältöjä voi jakaa muihin palveluihin käyttämällä niiden linkitysmahdollisuuksia. Tätä varten löytyy monia valmiita ratkaisuja, joista valittiin AddThis-palvelu. AddThis on JavaScipt-ohjelmakoodi, joka luo automaattisesti jakamistoiminnon, jolla kyseisen sivun voi jakaa yli sataan eri palveluun. Jakamistoiminto on mahdollista ottaa pois, jos sovellusta käytetään suljetussa ympäristössä. Toinen

esimerkki jakamisesta on mahdollisuus julkaista sisältö suoraan myös Facebookissa omalla seinällään. Tämän edellytyksenä on, että käyttäjä linkittää Facebook-tilinsä Channels-sovelluksen kanssa. Avoimessa palvelussa on myös mahdollista käyttää ns. linkinlyhennyspalveluita, joiden tuloksena saa lyhyen linkin, jonka voi jakaa muille. Tämä lyhyt linkki sitten ohjautuu varsinaiseen osoitteeseen Channels-sovelluksessa.

3.1.6 Sisältöjen luominen Atom-protokollan avulla

Käyttäjä voi myös lähettää sisältöjä matkapuhelimestaan hyödyntäen Atom-protokollaa. Aluksi tämä mahdollisuus luodaan Nokian älypuhelimiin saatavilla olevan Share Online-sovelluksen avulla. Share Online-sovellukseen voidaan lisätä palveluita, jotka määrittelevät itsensä ja mahdollistavat esimerkiksi kuvien ja videoiden lähettämisen matkapuhelimesta palveluun. Nokian puhelimet sisältävät nykyään muutaman palvelun valmiiksi asennettuina puhelimeen. Esimerkiksi Nokian oma Ovi-palvelu, kuvapalvelu Flickr ja Vox-blogipalvelu löytyvät monista Nokian puhelimista valmiiksi asennettuina ja valmiina käyttämään Atom-protokollaa.

3.2 Käytetyt ohjelmistot

3.2.1 Zend Framework

Zend Framework on nimensä mukaisesti Zend-nimisen yhtiön vetämä avoimen lähdekoodin hanke. Käytännössä Zend Framework on suuri luokkakirjasto, josta löytyy eri tarkoituksiin suunnattuja luokkia. Tämä tarkoittaa esimerkiksi sitä, että luokista voi käyttää vain omaan tarpeeseen sopivan luokan jo olemassa olevan sovelluksen osana. Ja on tietenkin mahdollista toteuttaa oma sovellus alusta loppuun hyödyntäen esimerkiksi Zend Frameworkin MVC-toteutusta ja tarpeen mukaan muita luokkia. Tällä hetkellä yleisesti saatavilla oleva versio Zend Framework-kirjastosta on versio 1.10.2.

Muita vastaavia luokkakirjastoja tai sovelluskehyksiä on olemassa PHP:lle useita. Esimerkiksi PHP:lle löytyy seuraavat ohjelmistot: Symfony, CakePHP, CodeIgniter. Jokainen aiemmin mainituista kuitenkin ohjaa kehittäjän käyttämään tietynlaista kokonaispakettia ja osa on edelleen PHP 4-version ohjelmistoja. Zend Framework valittiin, koska se on PHP 5-version ohjelmakirjasto ja se ei pakota kehittäjää minkäänlaiseen tietynlaiseen ratkaisuun.

Zend Frameworkkia hyödyntävien tahojen kannalta on helpottavaa se, että Zend Frameworkin kehitys kulkee tarkkana prosessina Zendin kautta. Tämä tarkoittaa sitä, että vaikka kuka tahansa voi ottaa osaa kehitystyöhön, niin Zendillä on oma tiimi ohjelmakoodin analysoimiseen ja laadun valvontaan. Näin ollen julkaistavat luokat ovat hyvin testattuja, hyvin dokumentoituja sekä toteuttavat projektin hyväksymät mallit ja säännöt ohjelmakoodin suhteen.

Vaikka Zend on kaupallinen yritys, Zend Framework on ilmainen avoimen lähdekoodin ohjelmisto, jota voi käyttää mihin tahansa tarkoitukseen. Zend tarjoaa kaupallisina tuotteina kehitystyökaluja ja palvelintuotteita tukemaan PHP-kehitystä. Zendin maksavat asiakkaat voivat vaikuttaa Zend Frameworkin kehitettäviin komponentteihin. Tämä tapahtuu yleensä, kun jotain haluttua toiminnallisuutta ei vielä ole tai halutun toiminnallisuuden toteutus on vielä kesken ja sen valmistumista halutaan nopeuttaa johtuen asiakkaan omien projektien vaatimuksista.

Zend-yhtiö on solminut yhteistyö- ja partneruussopimukset Microsoftin ja IBM:n kanssa vuonna 2006, joten PHP on saanut tukea myös alan isoilta vaikuttajilta. PHP on myös osoittanut olevansa oiva valinta myös suurten palveluiden luomiseen. Esimerkiksi Facebook on toteutettu PHP-kielellä.

3.2.2 jQuery

jQuery on yksi suosituimmista JavaScript-kirjastoista. Se on ilmainen ja laajalti käytössä erilaisissa ilmaisissa ja kaupallisissa sovelluksissa. jQueryä käyttävät esim. Google, Dell, Digg, NBC, Mozilla (Firefox-selaimen tekijä) ja Drupal

(jQuery, 2009). Microsoft on myös lisännyt jQueryn Visual Studio-kehitysympäristöönsä.

```
$("p.neat").addClass("ohmy").show("slow");
```

KUVIO 8. Esimerkki jQueryn komentosyntaksista

jQueryn perusidea on muuttaa JavaScriptin kirjoittaminen helpommaksi ja ymmärrettävämmäksi. Samalla se pyrkii piilottamaan kehittäjältä eri selainten väliset erot JavaScriptin tulkitsemisessa, jotta kehittäjä voi keskittyä toiminnallisuuksiin. Yllä olevassa esimerkissä (KUVIO 8) etsitään ensin elementti P, jonka class-attribuutti on arvoltaan ”neat”. jQuery mahdollistaa ketjutuksen komentojen antamisessa, ja esimerkissä nähdään, miten elementille annetaan ketjuna kaksi komentoa. Ensimmäinen komento antaa elementille uuden class-määreen ”ohmy” ja toinen komento kertoo, että elementti pitää tuoda näkyviin hitaasti.

JavaScriptin kohdalla on oleellista mainita myös AJAX-teknologia. Sen avulla voidaan lähettää palvelimelle kutsuja ja sitten vastausten perusteella voidaan muokata osaa sivusta ilman, että se tarvitsee ladata uudestaan. Tämä mahdollistaa monipuolisemman käyttöliittymän suunnittelun. jQuery sisältää monia hyödyllisiä AJAX-toimintoja, joita Channels-sovelluksessa käytetään.

3.2.3 MySQL-tietokanta

MySQL-tietokantapalvelin on nykyään Oraclen omistama relaatiotietokanta. Se on ilmainen, ja se on suosituin ns. Open Source -tietokanta. MySQL-tietokannan valintaa puoltaa myös PHP:n pitkäaikainen tuki MySQL:lle. Tällä hetkellä MySQL:n tulevaisuus on hieman epäselvä ja moni pelkää kaupallisen toimijan tekevän MySQL:stä vähintään osittain suljettua koodia.

Aiemmin MySQL-tietokanta ei ollut ammattilaisten suosiossa puutteellisten ominaisuuksien kuten transaktioiden sekä aitojen vierasavaimien puuttuessa. Se on kuitenkin kaventanut eroa kaupallisiin kilpailijoihinsa viime vuosina ja parantanut ohjelmistoaan karsimalla edellä mainittuja puutteitaan. Esimerkiksi Yahoo!, Google, Nokia ja YouTube käyttävät MySQL-tietokantaa.

Toteutuksessa käytetään Zend Frameworkin Zend_Db-luokkaa, joten tietokannan käyttämiseen ei tarvita muita erillisiä ohjelmistoja. Zend Framework tukee myös muita tietokantoja, ja yhteisen rajapinnan avulla vaihto toiseen tietokantaan on mahdollista ja suhteellisen helppoa. Luokka tuottaa taustalla MySQL:nkin käyttämän SQL-kyselykielen (Structured Query Language) lisäys-, muutos- ja hakukomentoja ja vie ne käytettävän tietokannan sovitinluokalle, joka tekee varsinaiset kutsut tietokantaan.

3.2.4 Videoiden käsittely

Videoiden kääntämistä varten palvelimelle asennettiin ffmpeg-ohjelmisto. Kyseinen ohjelma mahdollistaa käyttäjien omien videoiden kääntämisen Flash-videoksi, jotta käyttäjän videot on helppo näyttää sovelluksessa. Esimerkiksi käyttäjä voi ladata matkapuhelimellaan kuvaamansa videon sovellukseen MP4-formaatissa ja video käännetään sitten Flash-videoksi. Flash-videoiden käyttö on perusteltua, koska Adobe Flash Player on asennettu yli 98 %:iin internet-yhteydessä olevista koneista (Adobe, 2009).

Web-kamerasta oman videon tallentaminen vaati Flash -palvelimen, joka voi vastaanottaa kamerasta lähetettävää videodataa sekä ohjelman selaimen päähän, joka hoitaa kameran ohjauksen ja videodatan syöttämisen palvelimelle. Muutamien hakujen jälkeen päätettiin hankkia sovellus ostamalla Intialainen Quickcam Pro -ohjelmisto. Quickcam Pro on Flash-pohjainen (frontend) ohjelma, joka ohjaa kameraa ja lähettää sen dataa palvelimelle käyttäen RTMP-protokollaa.

Palvelinpäähän (backend) asennettiin tarkoitusta varten ilmainen Red5-ohjelmisto. Red5 on Javalla toteutettu palvelinohjelmisto, joka pystyy vastaanottamaan ja suoratoistamaan (Streaming) Flash-videota.

Lähitulevaisuudessa uusi HTML5-standardi ja sen mukana tuleva video-tagi voivat korvata Flash-videon. Silloin selain ei tarvitse Flashin tapaista erillistä ohjelmistoa videon näyttämiseen. Esimerkiksi YouTube-palvelusta on jo olemassa testiversio, joka näyttää videot HTML5-tuen avulla. Nykyselaimista esimerkiksi Mozillan Firefoxilla ja Googlen Chrome -selaimella on jo nyt mahdollista kokeilla HTML5-videotagia. HTML5-standardin käyttämästä videoformaatista ei kuitenkaan olla vielä yksimielisiä. Tällä hetkellä formaattina toimii avoimen standardin Ogg-formaatti, mutta sillä on paljon kaupallisia vastustajia, kuten esimerkiksi Nokia ja Apple.

3.3 Suunnittelumallit

Teknistä toteutusta ohjaa siihen valitut suunnittelumallit (Design patterns). Suunnittelumallit ovat abstraktin tason ratkaisuja yleisimpiin suunnitteluongelmiin. Monesti eri tilanteisiin löytyy useampi malli. On kuitenkin tärkeää valita joku malli, johon sitten sitoutuu sovelluksen elinkaaren ajaksi. Välillä voidaan puhua yksinkertaisesti vain hyvistä käytänteistä (Best practices).

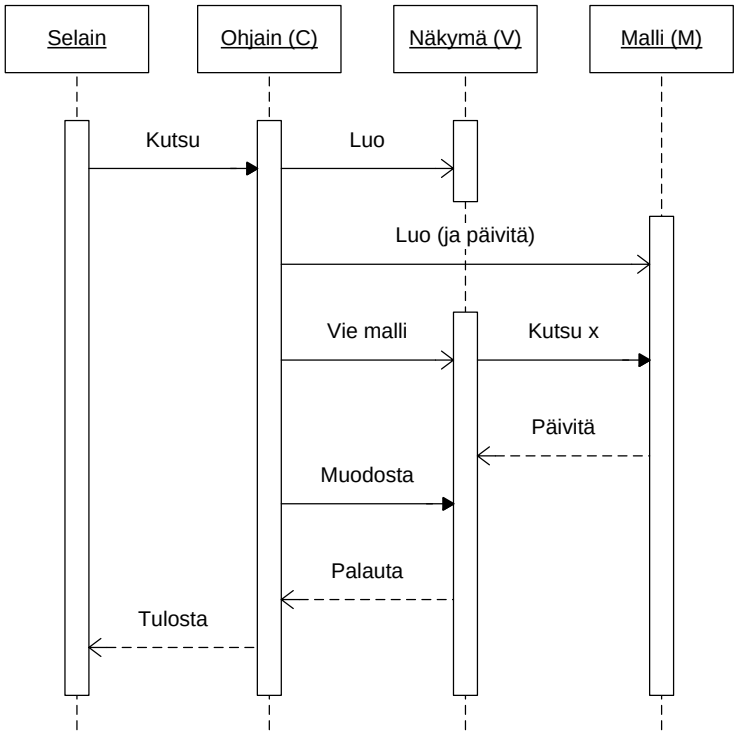
Monesti suunnittelumalli antaa käytännössä hyväksi todetun ehdotuksen tietyn asian toteuttamiseksi. Tällaisen mallin käyttämisestä on hyötyä myös jatkossa. Samaa ajattelua on helppo ottaa käyttöön muissa vastaavissa tilanteissa sekä mallin myötä syntyvä sanasto helpottaa työskentelyä useamman kehittäjän kesken. Suunnittelumallit myös mahdollistavat yllepidettävämmän, muutossietoisemman ja helpommin laajennettavan sovelluksen.

3.3.1 MVC-malli

MVC-arkkitehtuuri (tulee sanoista Model-View-Controller eli malli–näkymä–ohjain) on ohjelmistoarkkitehtuurityyli, jonka tarkoituksena on erottaa käyttäjän

syötteiden käsittely (ohjain), käyttöliittymä (näkymä) ja business-logiikka eli tiedon käsittely (malli). Tämä mahdollistaa, että jokaista osa-aluetta on helpompi kehittää, testata ja ylläpitää itsenäisesti.

MVC-malli on monesti käytössä verkkosovelluksissa, joissa ohjain saa selaimella käyttäjän GET- tai POST-syötteen ja siirtää ne käsiteltäväksi oikeille malleille. Sitten prosessoitu data tuottaa käyttäjälle HTML-muotoisen näkymän (KUVIO 9).



KUVIO 9. MVC-mallin sekvenssikaavio

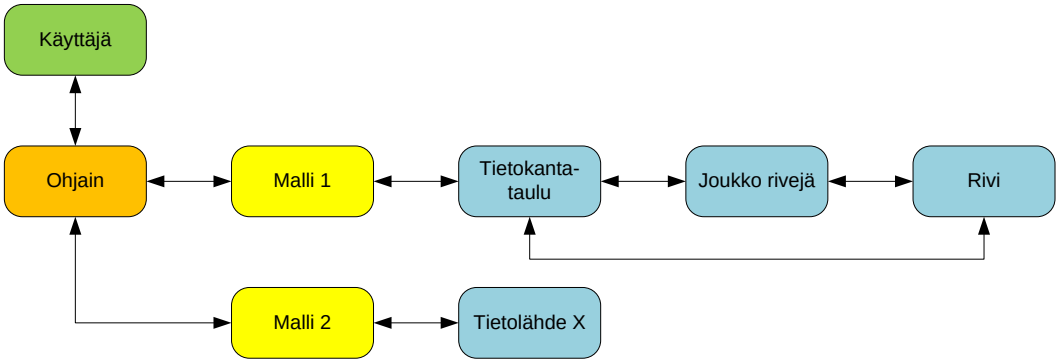
MVC-malliin liittyy useita hieman toisistaan poikkeavia tulkintoja tai sääntöjä, jotka määrittelevät, miten ohjain, malli ja näkymä toimivat suhteessa toisiinsa. Yksi yksinkertainen yleistys mallista on seuraavanlainen. Ohjain välittää käyttäjän toiminnot ja syötteet mallille käsiteltäväksi ja antaa mahdollisesti näkymälle dataa. Malli ei tiedä mitään ohjaimesta tai näkymästä. Sen vastuulla ovat vain sen

oma business-logiikka ja tietovarasto. Näkymä toteuttaa esityksen mallilta saamansa datan perusteella. Näkymä voi myös käyttää mallia suoraan lukutilassa.

Zend Framework-kirjaston avulla on mahdollista toteuttaa MVC-mallin mukainen toteutus. Siinä käyttäjän syötteet välitetään oikealle ohjaimelle sen mukaan, minkälaisen osoitteen eli kyselyn käyttäjä avaa. Osoitteet on linkitetty erilaisten syntaksien mukaan moduuleihin ja niiden ohjaimiin. Esimerkiksi osoite ”/content/3/edit” ohjaa käyttäjän sisältö -moduulin ohjaimeen, joka hoitaa sisältöjen muokkaamisen. Ohjain sitten komentaa mallia, joka on vastuussa sisällöistä ja toteuttaa vain niihin liittyviä toimintoja.

3.3.1.1 Fat models, thin controllers

Fat models, thin controllers -mallissa ajatuksena on, että ohjain (Controller) sisältää vähän ohjelmakoodia ja malli (Model) taas enemmän. Ohjaimen tehtävänä on vain välittää käyttäjän syötteet oikeille malleille, jotka hoitavat tiedon käsittelyn. Sitten malli palauttaa käsitellyn tiedon, jonka ohjain sitten siirtää näkymälle (View). Hyödyt ovat selvät. Ohjelmakoodi muuttuu selkeämmäksi ja helpommin ylläpidettäväksi. Esimerkiksi jos useampi ohjain käyttää samaa mallia, niin ylläpito tarvitsee tehdä vain yhteen malliin usean ohjaimen sijasta.

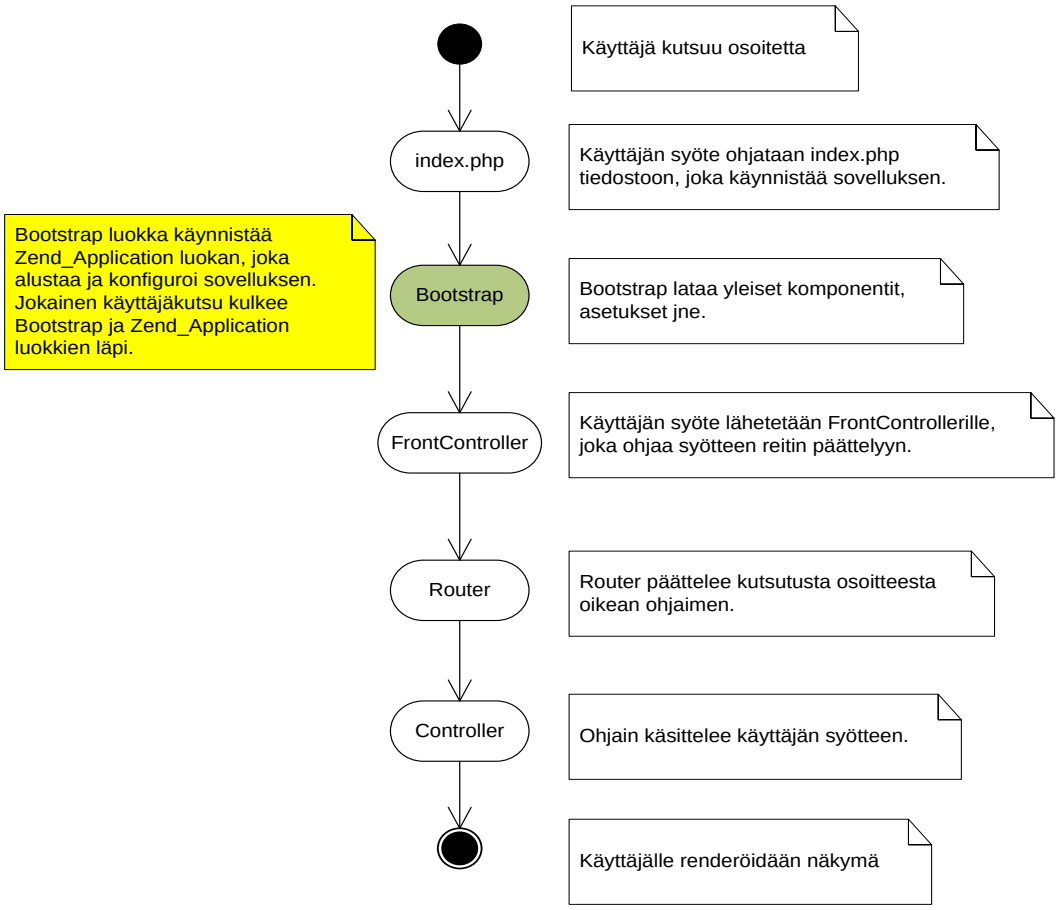


KUVIO 10. Ohjain ja malli

Kuten kuviossa 10 (KUVIO 10) nähdään, ohjain komentaa mallia ja malli käsittelee siihen liitettyjä tietolähteitä. Esimerkiksi jos käyttäjä haluaa lisätä videon, niin ensin hän lähettää video-tiedoston ohjaimelle. Ohjain käskee mallin käsitellä tiedostolatauksen. Malli huomaa, että kyseessä on videotiedosto, joten se toteuttaa videotiedostoihin liittyviä toimintoja. Lopulta kun malli on toteuttanut kaikki tarvittavat tehtävät, se komentaa tietolähteen luomaan uuden rivin tietokantaan uudelle videotiedostolle.

3.3.2 Bootstrapping

Bootstrapping-mallissa kaikki sovellukseen kohdistuvat käyttäjän kutsut ja syötteet ohjataan yhteen tiedostoon, joka sitten asettaa ympäristön, lataa tarvittavat resurssit ja konfiguraatiot sekä huolehtii mahdollisista istunnoista ja välimuistituksista (KUVIO 11).



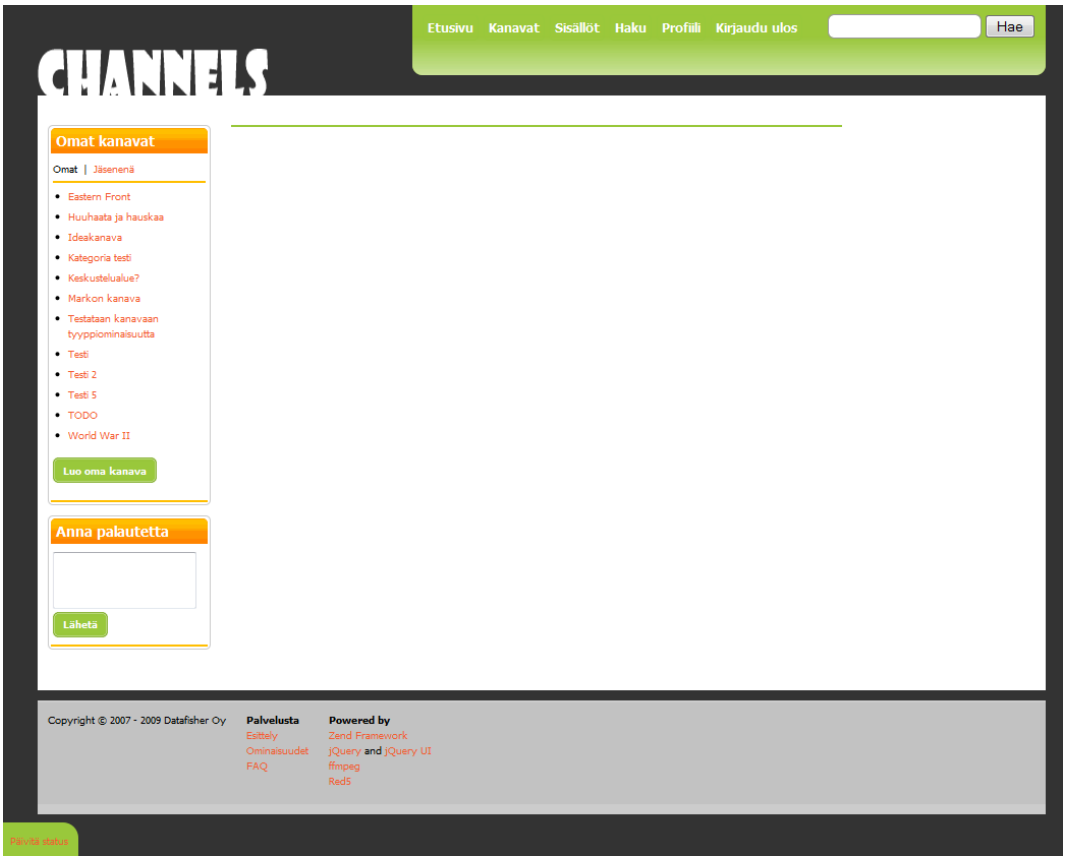
KUVIO 11. Bootstrapping-mallin aktiviteettikaavio

Oletuksena käyttäjän kutsuma osoite toimii perustana sille, mikä ohjain saa käyttäjän syötteen käsiteltäväksi. Palvelimilla on mahdollista toteuttaa osoitteen uudelleenkirjoitustekniikkaa (URL rewriting). Silloin voidaan luoda sääntö, että kaikki kutsutut osoitteet ohjataan haluttuun osoitteeseen niin, että kutsuttu osoite annetaan uudelle osoitteelle parametrinä. Channels-sovelluksessa ja Zend Frameworkin oletustoiminnallisuudessa uudelleenkirjoitus tehdään seuraavasti:

Zend Frameworkissa Bootstrapping-mallin toteuttaa Zend_Application-luokka yhdessä Zend_Controller_Front-luokan kanssa, jolle käyttäjän kutsut ja syötteet ohjataan. FrontController on eräänlainen edustaohjain, joka ohjaa käyttäjän syötteen oikealle ohjaimelle käsiteltäväksi. Tämä tapahtuu kutsutun osoitteen mukaisesti. Järjestelmään on muodostettu sääntöjä sen suhteen, minkälaiset osoitteet ohjataan millekin ohjaimelle. Näitä sääntöjä kutsutaan reiteiksi (routes), ja niiden päättelystä vastaa Zend_Controller_Router-luokka.

3.3.3 Two Step View

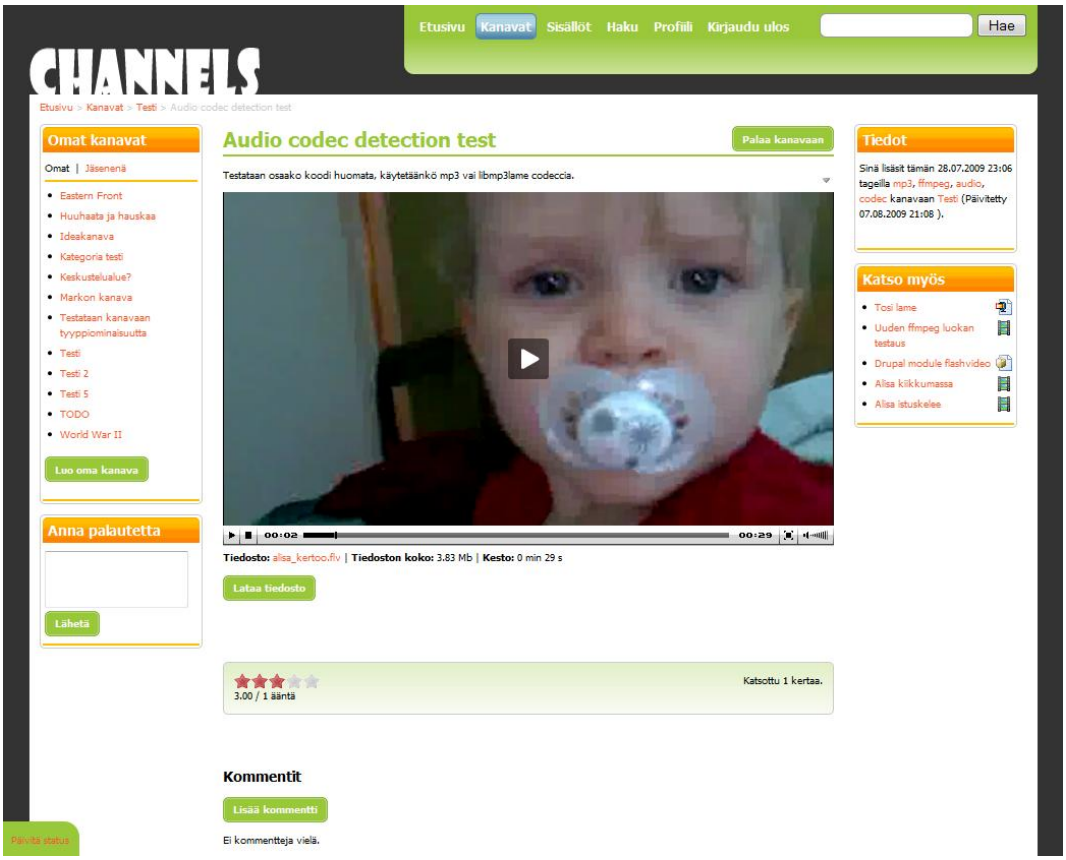
Sovelluksen ulkoasu muodostetaan kahdessa tasossa: yleinen ulkoasu (layout) ja sivukohtainen näkymä (template). Tämän mallin ansioista voidaan tehdä muutama yleinen ulkoasumalli, joita sitten käytetään kaikilla sivuilla vaikka varsinainen sisältö muuttuu. Zend Framework toteuttaa tämän mallin Zend_Layout- ja Zend_View-komponenttien avulla.



KUVIO 12. Yleinen ulkoasu

Mallissa on myös mahdollista, että sivukohtainen näkymä voi vielä muokata yleistä ulkoasua. Yllä olevassa kuviossa (KUVIO 12) on esitelty pelkkä yleinen ulkoasu. Siinä määritellään ylä- ja alaosio sekä yleiset komponentit vasempaan palstaan.

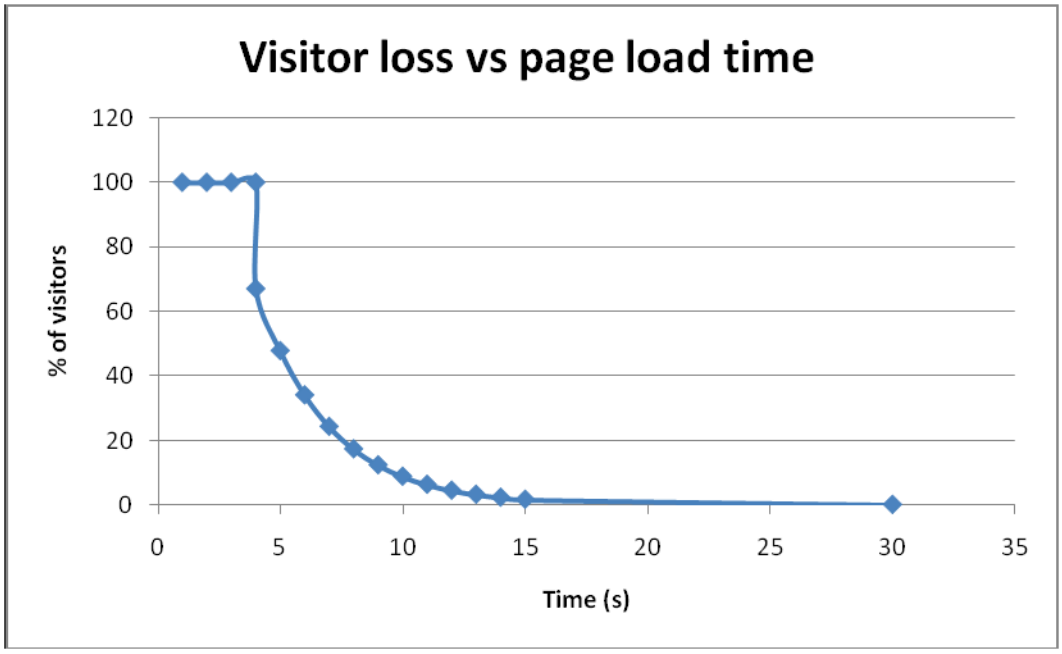
Seuraavassa kuviossa (KUVIO 13) täytetään yleinen ulkoasu sivukohtaisella sisällöllä. Huomaa, että sivupohja osaa syöttää sisältöä sekä keskimmäiseen että oikeaan palstaan. Sovellukseen on toteutettu sisäinen moduuleille yhteinen rajapinta, jonka avulla moduulit voivat määritellä erilaisia lohkoja mihin tahansa sivuun. Lohkot näkyvät kuviossa laatikoina, joiden otsikot ovat oransseissa palkeissa. Lohkot tarjoavat yleensä erilaisia nostoja pääsisältöön liittyvästä tiedosta. Esimerkiksi alla olevassa kuviossa nostetaan yhteen lohkoon muut sisällöt, joilta löytyy samoja tageja kun avatulta sisällöltä.



KUVIO 13. Sivukohtainen näkymä

3.4 Suorituskyky

Verkkopalveluiden suunnittelussa sivuston suorituskyky ja etenkin sivunlatausaika ovat kriittisessä osassa. Yleensä on helppo tehdä ohjelmakoodi, joka toteuttaa halutun toiminnon, mutta toteutus muuttuu selkeästi haastellisemmaksi, jos halutaan, että ohjelmakoodi ei vie liikaa resursseja ja näin ollen hidasta sivunlatausta. Tutkitusti ihmiset hylkäävät koko palvelun käyttämisen, jos sivut eivät lataudu tarpeeksi nopeasti kuten alla olevassa kuviossa (KUVIO 14) ilmenee. Kuviossa käyttäjien prosentuaalinen määrä vähenee kun sivunlatausaika kasvaa.



KUVIO 14. Kävijöiden häviäminen suhteessa sivunlatausaikaan (Pear Analytics, 2009)

Akamain tilaama tutkimus vuodelta 2009 kertoo, että nykyään kriittinen aika sivunlatauksessa on 2 sekuntia. Tutkimuksessa testatun sivuston kävijät siis halusivat, että sivut latautuivat maksimissaan 2 sekunnissa. Vertailuksi mainittakoon Akamain vuonna 2006 tilaama samanlainen tutkimus, jossa kävijöiden mielestä sivujen piti latautua alle 4 sekunnissa. Internetin jatkuva nopeuden kasvu siis vaikuttaa myös käyttäjien odotuksiin vastaavasti. Akamain uudempi tutkimus on saatavilla osoitteesta http://www.akamai.com/2seconds rekisteröitymistä vastaan.

3.4.1 Välimuistitus

Yleisin tapa nopeuttaa sivunlatausta ja vähentää resurssien käyttöä on välimuistitus (cache). Siinä joku toiminto ajetaan ja sen lopputulos tallennetaan välimuistiin, joka voi olla esim. tiedostossa tai tietokannassa. Välimuistiin tallennetulle tiedolle annetaan aika, jonka se on voimassa. Kun tämä aika umpeutuu, kyseinen toiminto välimuistitetaan uudestaan.

Esimerkkinä voidaan käyttää jotain isoa ja resursseja vaativaa tietokantahakua. Jos jokaisen käyttäjän sivulatauksella kyseinen tietokantahaku ajetaan uudestaan, niin se kuormittaa tietokantaa sekä palvelua muutenkin. Jos kyseinen haku ajetaan vain kerran ja se välimuistitetaan, niin suurin osa käyttäjien sivulatauksista saa välimuistista luetun tiedon sen sijaan, että kyseinen haku tehtäisiin uudestaan tietokantaan.

Channels-sovelluksessa välimuistitus toteutettiin kaikille yleisille tietuille niin, että välimuistiversio on voimassa niin kauan, kunnes tietuetta päivitetään. Esimerkiksi kanavat, sisällöt ja käyttäjät ovat välimuistitettu, jotta niiden lataus olisi nopeaa ja vähän resursseja vaativaa. Välimuistitus tukee myös tageja, joiden avulla voidaan luoda logiikkaa cachen tyhjentämiselle. Esimerkiksi tagi ”channel” tuhoaa kaikki välimuistitetut tiedot, joilta kyseinen tagi löytyy. Näin voidaan poistaa useita välimuistitiedostoja, kun jotain tietoa päivitetään.

3.4.2 Yksittäisten tiedostolatauksien määrän vähentäminen

Jokainen sivu verkkopalvelussa sisältää itse sivun (yleensä HTML-dokumentti), sekä siihen liitettyjä muita tiedostoja kuten tyylitiedostoja, JavaScript-tiedostoja ja kuvia. Näin ollen sivun mukana voidaan ladata useita kymmeniä yksittäisiä tiedostoja. Loppukäyttäjän sivunlatausajasta n. 80 % kuluu tähän latausoperaatioon käyttäjän selaimessa (Yahoo!, 2009).

Sovelluksessa päätettiin käyttää kahta perustekniikkaa nopeuttaakseen sivulatausta. Kuvien latausaikaa voi vähentää toteuttamalla CSS Sprites-tekniikalla. Siinä yksi kuvatiedosto sisältöö useita eri kuvia, joita käytetään sivustolla. Sitten CSS-tyylimäärittelyssä määritetään X- ja Y-koordinaattien avulla haluttu kuva sprite-kuvan sisältä (ks. esimerkki KUVIO 15). CSS-määreiden avulla originaalia kuvaa siirretään ja rajataan, jotta saadaan tuloksena näkyville vain haluttu osa.

KUVIO 15. Esimerkki CSS Sprites-tekniikan toiminnasta

Toinen perustekniikka on yhdistää JavaScript- ja CSS-tiedostot yhdeksi tiedostoksi. Tässä hyödynnettiin Minify PHP -sovellusta, joka osaa yhdistää ja välimuistittaa halutut JavaScript- ja CSS-tiedostot. Minify pystyy myös pienentämään tiedostojen kokoa poistamalla tyhjää tilaa tiedostoista. Haasteena tässä tekniikassa on se, että kaikilla sivuilla ei välttämättä käytetä kaikkia samoja JavaScript- tai CSS-tiedostoja. Näin ollen mahdollinen yhteen pakattu tiedosto sisältäisi kaikki järjestelmässä käytetyt tiedostot kaikilla sivuilla, vaikka niitä ei tarvittaisikaan.

3.5 Mahdolliset integraatiot

Kun luodaan sovelluksia yrityksien käyttöön, on otettava huomioon, että yrityksellä voi olla jo olemassa sovelluksia tai järjestelmiä, joihin tarvitsee tehdä integraatioita. Yleisimpinä esimerkkeinä tälläisistä integraatioista ovat keskitetyt käyttäjätietokannat kuten Active Directory tai LDAP. Zend Framework tarjoaa Zend_Auth-luokan LDAP-adapterin avulla mahdollisuuden autentikoida käyttäjät myös Active Directory -palvelimelta tai LDAP-palvelimelta.

Jos sovellusta ajetaan avoimena palveluna, niin silloin käyttäjä-integraatio voidaan toteuttaa niin, että uusi käyttäjä voi käyttää jonkun muun palvelun tunnuksia kirjautuakseen sisään. Esim. Facebookilla on oma Connect-rajapinta, jota käyttämällä muut sivustot voivat antaa käyttäjien kirjautua heidän Facebook-

tunnuksillaan. Toinen yleinen autentikointimalli on käyttää yleistä OpenID-autentikointia. OpenID:ssä mikä tahansa sivusto voi olla käyttäjätietokantana, jota vasten autentikaatio tehdään. Esim. Google, Microsoft ja Yahoo! tukevat myös OpenID:tä. Googlella on myös oma Friend Connect-palvelu.

Yhteistä näissä palveluissa on se, että käyttäjä autentikoidaan alkuperäisessä palvelussa uuden palvelun sijasta. Kun autentikaatio onnistuu, niin käyttäjä ohjetaan uuteen palveluun, joka sitten voi tallentaa käyttäjän tietoja ja näin yhdistää käyttäjän tilit. Hyöty on siinä, että käyttäjä voi käyttää yhtä tunnusta useissa tämän kaltaista tekniikkaa tukevissa palveluissa. OpenId on myös mahdollista toteuttaa omaan palveluun, jolloin se toimii alkuperäisenä käyttäjätietokantana.

4 TOTEUTUS

4.1 Pohjaratkaisu

Koska valittu ohjelmakirjasto Zend Framework mahdollistaa useamman eri tavan sovelluksen toteuttamiseen, piti aluksi päättää sovelluksen toteutusmalli. Tarkoitus oli käyttää mahdollisimman paljon Zend Framework-kirjaston esittelemiä toteutusmalleja ja samalla välttää turhaa oman ohjelmakoodin tuottamista.

Epäselvissä tapauksissa kannatti hyödyntää aktiivista kehittäjäyhteisöä esimerkiksi keskustelemalla foorumeilla vastaan tulleista ongelmista ja erilaisista toteutusmalleista. Pohjaratkaisuksi valittiin MVC-malliin pohjautuvan modulaarisen sovelluksen toteuttava Zend_Application-luokka yhdessä Zend_Controller_Front-luokan kanssa nopean sovelluskehityksen takaamiseksi.

Zend_Application toteuttaa bootstrapping ajattelun sekä luo valmiin pohjan modulaarisen sovelluksen tekemiselle yhdessä Zend_Controller_Front-luokan kanssa. Zend_Application hoitaa sovelluksen käyttämät resurssit kuten tietokannat, kielikäännökset, konfiguraatiotiedostot, hakemisto- ja moduulirakenteen.

4.2 Moduulit

Sovellus koostuu moduuleista, jotka muodostavat varsinaiset toiminnallisuudet ja näkymät sovellukselle. Moduulit ovat osakokonaisuuksia, jotka toteuttavat jonkun selkeän oman kokonaisuuden sovelluksessa. Suurin osa toteutetuista moduuleista on pakollisia sovelluksen toiminnan kannalta. Sovellus on kuitenkin suunniteltu niin, että lisämoduulit voidaan asentaa tai poistaa vapaasti niin, että sovellus jatkaa toimintaansa.

Käyttäjä-moduuli on perusmoduuli, joka tarjoaa kaikki toiminnallisuudet käyttäjään liittyen. Käyttäjä-moduuli hoitaa mm. käyttäjien kirjautumisen ja uloskirjautumisen. Käyttäjä voi moduulin avulla ylläpitää profiilitietojaan sekä päivittää omaa profiilikuvaansa. Hallinnoija voi tämän moduulin avulla tehdä käyttäjiin liittyviä hallinnollisia toimia, kuten luoda uusia käyttäjiä, poistaa käyttäjien tilejä käytöstä jne.

Kanava-moduuli antaa käyttäjälle mahdollisuuden luoda kanavan, hallita sen käyttäjiä ja luoda siihen sisältöä. Se listaa olemassa olevat kanavat sekä antaa näkymät kanavien ja niiden sisältöjen esittämiselle. Moduuli mahdollistaa myös kanavien RSS-syötteet. Koska kanava on tekijänsä ja mahdollisesti muiden käyttäjien yhteinen toiminta-alue, moduuli sisältää myös kanavakohtaiset käyttöoikeudet ja niiden määrittelemisen.

Sisältö-moduuli on laajin moduuleista ja se sisältää työkalut kaikkien erityyppisten sisältöjen luomiseen, esittämiseen, muokkaamiseen ja poistamiseen. Jokainen erityyppinen sisältö tarvitsee omat lomakkeensa, käsittelijänsä ja esitysmuodot. Sisältö-moduuli sallii myös muiden moduulien määritellä uusia sisältötyyppejä. Sisältötyypit määritellään kaksitasoisesti niin, että esimerkiksi YouTube-videon linkki on alatyyppi päätyypille linkki.

Sisältö-moduuli sisältää valmiina seuraavat sisältötyypit:

- teksti
    - juttu, artikkeli tai esimerkiksi uutinen
- linkki
    - mikä tahansa linkki
    - YouTube-video
    - Vimeo-video
    - Viddler-video
    - SlideShare-esitys
- tiedosto
    - mikä tahansa tiedosto
    - kuvatiedosto

    - videotiedosto
    - materiaalipaketti
- kartta
    - Google Maps -kartta
- Web-kamera nauhoitus.

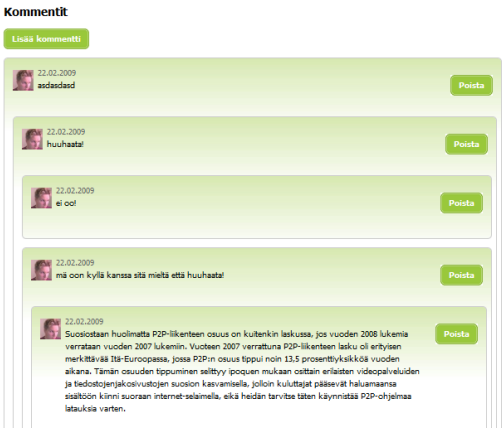
Tagi-moduulin toteuttama kohteiden taggaaminen on lisätty Channels-sovelluksessa kanaviin ja sisältöihin. Kanavissa tagit toimivat kategorioina ja sisällöissä asiasanoina. Tagipilvi (KUVIO 16) on ehkä yksi tutuimmista esimerkeistä, miten tageja hyödynnetään. Siinä esitetään tageja, joita on eniten käytetty, ja tagia klikkaamalla annetaan käyttäjälle lista kohteista, joille kyseinen tagi on annettu.

KUVIO 16. Tagipilvi

Jatkossa tageja voisi hyödyntää ns. Faceted-haussa, jossa käyttäjä valitsee tagin, joka antaa tietyn hakutuloksen. Sen jälkeen käyttäjä voi valita hakutuloksesta muodostuneen uuden tagi-joukon sisältä toisen tagin, joka rajaa edelleen samaa hakutulosta.

Kommentti-moduuli lisää kommentoinnin kiinteäksi osaksi Channels-sovelluksen sisältöjä. Kirjautuneet käyttäjät voivat kommenttiominaisuuden avulla antaa palautetta sisällöstä tai sitten kommentit voivat toimia keskusteluketjuna sisältöön liittyen. Kommentit voidaan esittää joko ajan mukaan tai sitten hierarkisesti

(KUVIO 17), jolloin myös kommentteihin voidaan lisätä kommentteja. Uusimmat kommentit listataan palvelun etusivulla, josta pääsee siirtymään kommenttien lukemiseen. Uusimmat kommentit liittyvät yleensä myös käyttäjille mielenkiintoisimpiin sisältöihin. Kommentti-moduuli on rakennettu niin, että kommentoinnin voi lisätä ohjelmallisesti mihin tahansa kohteeseen.



KUVIO 17. Esimerkki hierarkisesta kommentti-listasta

Arvostelu-moduulin avulla erilaisille kohteille järjestelmässä voidaan toteuttaa käyttäjille nopea palautteen antaminen. Palaute annetaan tähdillä, joka voidaan ajatella arvosanana asteikolla 1 - 5 (KUVIO 18). Käyttäjä voi antaa vain yhden arvostelun yhtä sisältöä kohden. Arvostelutoiminto on rakennettu teknisesti niin, että sen voi lisätä mihin tahansa kohteeseen. Esimerkiksi arvosteluominaisuus voitaisiin lisätä käyttäjään, jolloin muut käyttäjät voisivat antaa arvosanan esim. käyttäjän toiminnasta palvelussa.

KUVIO 18. Arvosteluominaisuus

Haku-moduuli lisää hakukoneen Channels-sovellukseen. Sovelluksen hakumoottorina toimii Zend_Search_Lucene, joka käyttää Java maailmassakin tuttua Lucene hakuindeksiä. Siinä halutut kohteet lisätään dokumentteina indeksiin, jossa kohteen tiedot paloitellaan hakua varten. Itse indeksi on tiedostopohjainen datavarasto.

Hakukone mahdollistaa erilaisten hakujen käytön, jossa voidaan käyttää esimerkiksi loogisia operaatiota, kuten AND-, OR- ja NOT-operaatioita. Hakutulosten järjestys määräytyy oletuksena haetun tai haettujen hakutermien esiintymystiheyden mukaan.

Mobiili-moduuli lisää sovellukseen mobiili-liittymän, joka toteutettiin Atom-protokollan avulla. Se mahdollistaa kuva-, video- ja tekstisisältöjen lähettämisen Channels-sovellukseen protokollaa tukevalla matkapuhelimella. Atom on XML-pohjainen protokolla, jonka avulla käyttäjä voi esimerkiksi lähettää kuvan ja siihen liittyvää kuvatekstiä haluamaansa kanavaan sovelluksessa.

KUVIO 19. Nokia 5800 Xpress Music ja Share Online -sovellus

Atom-protokollasta on muutama versio, joiden toiminta eroaa toisistaan selkeästi. Esimerkiksi uusin versio mahdollistaa sisältöjen selailun kun aiempi versio

mahdollisti ainoastaan sisältöjen lähettämisen. Protokollan käyttämisessä on edelleen monia ratkaistavia asioita, joita protokolla itse ei ratkaise. Tällä hetkellä mobiili-liittymää on testattu vain Atom versiolla 0.3 ja Nokian Share Online -sovelluksella (KUVIO 19). Dokumentoinnin puute on ollut osasyynä siihen, miksi versiota 1.0 ei ole vielä saatu käyttöön.

Channels-sovellukseen on luotu useita toimintaa laajentavia moduuleja. Esimerkiksi Facebook-moduuli mahdollistaa käyttäjän Facebook-tilin linkittämisen Channels-sovellukseen. Tämän jälkeen käyttäjä voi päivittää statuksensa Facebookkiin Channels-sovelluksesta käsin. Käyttäjä pystyy myös selaamaan omaa uutisvirtaansa suoraan Channels-sovelluksessa. Datafisher Oy:n käyttöön tuotetaan myös moduuli, joka integroi olemassa olevan projektihallinnoinnin osaksi Channels-sovellusta.

4.3 Riskien hallinta

Sovelluskehityksessä sopivat työkalut ovat ratkaisevassa osassa, kun halutaan luoda nopeasti toimivia ratkaisuja sekä halutaan hallita riskejä, joita siihen kuuluu. Työhön kuuluu olennaisena osana myös riskien hallinta. Esimerkiksi kehittäjän tietokoneen hajoaminen ja ohjelmakoodin katoaminen pitää ottaa huomioon.

Versionhallinnan avulla pidetään kirjaa ohjelmakoodiin kohdistuneista muutoksista sekä sen avulla voidaan palata taaksepäin, jos uusi ohjelmakoodi rikkoo sovelluksen toiminnan. Siksi ohjelmakoodi säilytetään versionhallinnassa, jotta muutokset on seurattavissa sekä asennusten päivitys helpottuu. Versionhallintaohjelmistona toimii Subversion (SVN), jota käytetään Windows-työasemalta käyttämällä joko TortoiseSVN-ohjelmaa tai ohjelmointiympäristönä toimivan Aptana Studion kautta.

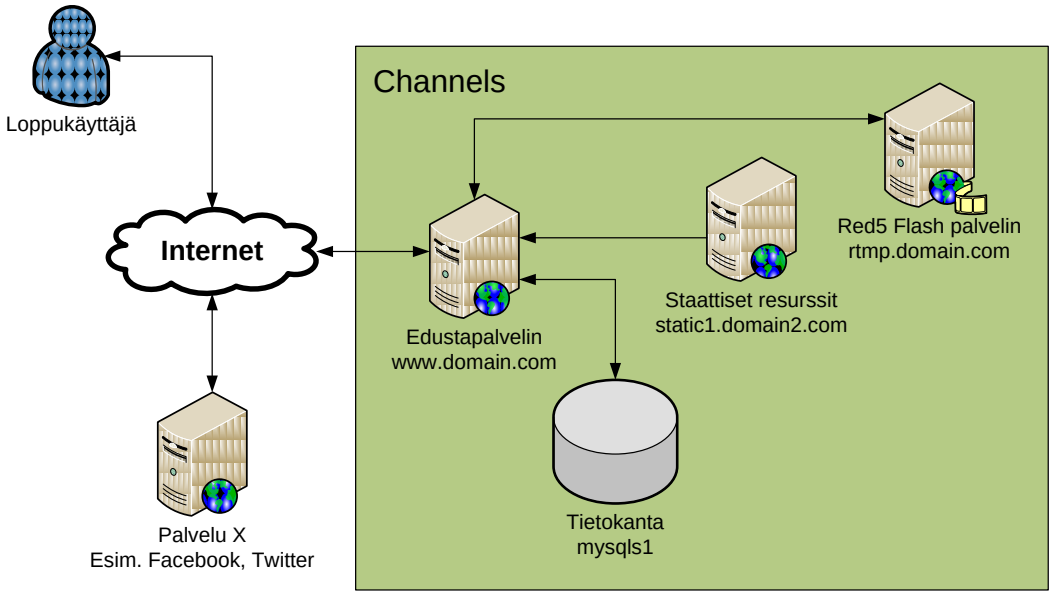
Tietokannan ja käyttäjien tiedostojen varmuuskopiointi on kriittisen tärkeää tuotantopalvelimella toimivan palvelun osalta. Tietojen palautukseen kannattaa olla olemassa valmis suunnitelma, jonka avulla voidaan nopeuttaa palvelun palauttamista toimintaan ja samalla minimoida uuden tiedon häviäminen.

4.4 Palvelinympäristö

Channels-sovelluksen kehitysversio on rakennettu alla kuvatun Linux-palvelimen (LAMP) päälle ja sen tuotantoympäristön pitää olla pääpiirteittäin samanlainen. Palvelinympäristö voidaan toteuttaa muunkin Linux-distribuution päälle (esim. CentOs tai Ubuntu) tai Microsoft Windows -palvelimelle. Näissä tapauksissa kuitenkin voi tulla haasteita valittujen lisäosien kanssa, kuten ffmpeg-ohjelmiston tai Red5-palvelimen kanssa.

- Fedora Core 9 -linux käyttöjärjestelmä
- Apache v2.2.9 -HTTP palvelin
- PHP v5.2.9 -tulkki (+ lukuisia lisäosia)
- MySQL v5.0.77 -tietokantapalvelin
- ffmpeg -videoiden konvertointityökalu
- Red5 -Flash palvelin Flash-videon streamaukseen
- crontab -ajastetut ohjelma-ajot.

Palvelun arkkitehtuurikaaviossa (KUVIO 20) esitellään perusmalli palvelun toteutuksesta arkkitehtuuritasolla. Edustapalvelin sisältää itse sovelluksen, se siis käsittelee käyttäjän kutsut ja syötteet. Staattiset resurssit, kuten JavaScript-tiedostot ja jotkut kuvat, on sijoitettu omalle palvelimelleen oman domain-osoitteen alle. Tietokanta ja Flash Server ovat myös omina palveluinaan. Kehitysvaiheessa kaikki nämä voidaan luoda saman fyysisen palvelimen sisään, mutta tuotantokäytössä ja etenkin käyttäjämäärän kasvaessa arkkitehtuuri auttaa skaalaamaan palvelua tarpeen mukaan.

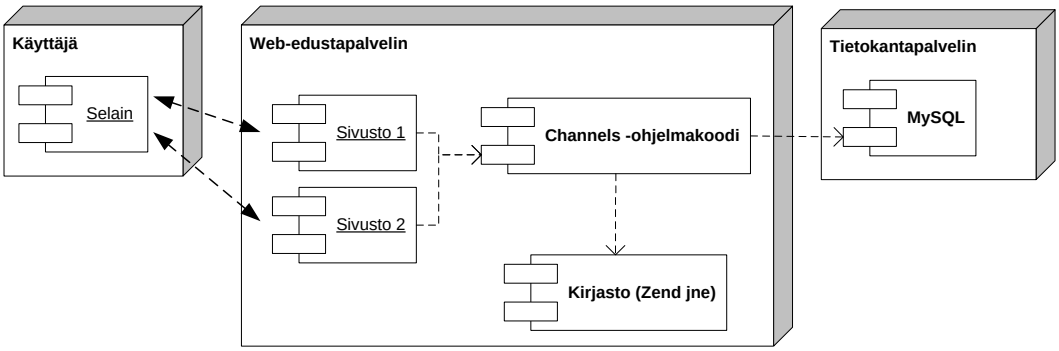


KUVIO 20. Palvelinarkkitehtuuri

Esimerkiksi edustapalvelimia voidaan lisätä arkkitehtuuriin lisää, jos käyttäjien ja edustapalvelimien välille lisätään kuormanjakaja (load balancer). Samoin tietokantapalvelimia ja staattisia resurssipalvelimia voidaan lisätä arkkitehtuuriin tarvittaessa.

4.5 Asentaminen ja konfigurointi

Sovellus koostuu kolmesta pääosasta (KUVIO 21): kirjasto, sovellus sekä sivusto-instanssin konfiguraatiot ja sivustokohtaiset tiedostot. Kirjasto on kokoelma eri ohjelmakirjastoja, joita sovellus hyödyntää. Esimerkiksi Zend Framework löytyy kirjastosta. Sovellus on itse Channels-sovelluksen ohjelmakoodi. Siellä on kaikki moduulit, kontrollerit, mallit ja näkymät. Sivusto-instanssin hakemisto sisältää yhden Channels-sovellusta hyödyntävän sivuston konfiguraatiotiedostot sekä kaikki muuttuvat tiedostot. Tähän hakemistoon tulee esimerkiksi sivuston oma välimuistitus, käyttäjien lataamat tiedostot sekä hakuindeksi.



KUVIO 21. Esimerkki palvelinasennuksesta

Asennus on suunniteltu niin, että kirjaston ja sovelluksen tarvitsee olla vain yhdessä paikassa palvelimella, joita sitten eri sivusto-instassit käyttävät jaetusti. Tämä helpottaa ohjelmakoodin ylläpitoa. Tämä on myös oletustilanne, jos sovellusta tarjotaan palveluna asiakkaille sen sijaan, että kaikki sovelluksen elementit asennettaisiin asiakkaan palvelimelle.

5 JOHTOPÄÄTÖKSET

5.1 Tavoitteiden toteutuminen

Työn tavoitteena oli suunnitella ja toteuttaa yhteisölliseen viestintään suunnattu Channels-sovellus. Sovellus saatiin toteutetuksi noudattaen valittuja suunnittelumalleja ja käyttäen valittuja teknologioita tehokkaasti. Kehitysvaiheen aikana lähinnä yksityiskohtaiset toteutukseen liittyvät tekniset asiat muuttuivat. Sovelluksesta tuli kuitenkin pääpiirteittäin suunnitellun mukainen. Kanavien ja sisältöjen osalta sovellus on määrittelyn mukainen.

Haasteita ilmeni testauksen järjestelmällisessä toteuttamisessa. Tästä seurasi monia virheitä ohjelmassa, jotka johtuivat jostain muusta sovelluksen osasta. Kehitystä kuitenkin auttoi säännöllinen ohjelmakoodin versionhallinta sekä Zend Frameworkin valinta toteutuksessa. Zend Framework päivittyi kehityksen aikana myös paljon, joka osaltaan ratkaisi tiettyjä ongelmia ja toisaalta loi välillä uusia ongelmia. Ansaintamallien toimivuudesta ei ole vielä tässä vaiheessa näyttöä. Niiden arviointi tarvitsee useamman asiakastoteutuksen, jotta voidaan todeta toimivatko ne vai eivät.

Jos sovelluksen kehitys alkaisi uudestaan, joustavan ja järjestelmällisen testaustavan määrittelyyn kannattaisi panostaa enemmän. Se nopeuttaisi ja auttaisi kehitystä poistamalla paljon turhaa työtä ja sitä kautta virheitä sovelluksessa. Myös työtavat ja käytettävät kehitysvälineet tulisi arvioitua aiempaa tarkemmin. Esimerkiksi Test Driven Development (TDD) voisi olla hyvä lähtökohta testaamiselle. Siinä kehitys tapahtuu testaus edellä niin, että ensin tehdään testitapaus ja sitten vasta toteutetaan testin läpäisevä varsinainen ohjelmakoodi.

5.2 Tulevaisuuden näkymät

Jatkossa sovellusta on tarkoitus kehittää edelleen. Etenkin kanavien erilaiseen esittämiseen tullaan panostamaan, kun aiemmin on panostettu sisältöjen

esittämiseen. Aluksi kannattaa kehittää olemassa olevia moduuleja ja jalostaa niiden ominaisuuksia vakaiksi kokonaisuuksiksi ja sitten vasta keskittyä uusien ominaisuuksien toteuttamiseen.

Todennäköisiä kehityskohteita on entistä parempi mobiilikäytettävyys sekä erilaiset työkalut monitasoisempien sisältöjen luomiseksi. Esimerkiksi tarjoituksena on toteuttaa työkalu, jolla voidaan luoda dynaamisesti materiaalipaketteja. Työkalulla voisi hallinnoida materiaalipaketin tiedostoja sekä sen dataa, joka yleensä on XML-muodossa. Myös käyttäjien välinen kommunikointi viestien ja reaaliaikaisen keskustelun merkeissä on suunnitteilla. Tämä mahdollistaisi entistä joustavamman viestinnän itse kanavien ja niiden sisältöjen ympärillä.

Tarkoitus on myös kokeilla Channels-sovellusta julkisena ja kaikille avoimena palveluna. Silloin olisi ainutlaatuinen mahdollisuus saada laajempaa käyttäjäpalautetta. Avoimesta palvelusta saatava palaute antaisi paljon tietoa siitä, mikä toimii ja mikä tarvitsee kehittämistä. Avoimessa palvelussa voitaisiin myös testata siihen sopivia ansaintamenetelmiä. Uuden palvelun testaaminen tarvitsee uusia käyttäjiä ja uusien käyttäjien saaminen olisi tehtävä mahdollisimman helpoksi. Tässä kannattaa tutkia esimerkiksi Facebook Connect -rajapinnan hyötykäyttöä. Silloin olemassa olevat Facebook-käyttäjät voisivat liittyä käyttäjäksi Channels-palveluun napin painahduksella. Muita vastaavia tekniikoita ovat OpenID ja Googlen Friend Connect.

Sovelluksen kehittäminen toi esiin paljoa uutta tietoa ohjelmistoarkkitehtuurista nykypäivänä. Myös moduulaaristen sovellusten kehitys sekä erilaisten rajapintojen hyödyntäminen tuli tutuksi. Sovelluksen luonne ja ominaisuudet omalta osaltaan ohjasivat tutustumaan moniin aiemmin tuntemattomiin, mutta hyödyllisiin ja muualla hyväksi havaittuihin ratkaisuihin.

Seuraava askel on toteuttaa Channels-sovelluksesta sisäiseen käyttöön suunnattu versio Datafisherille. Samalla sovellusta laajennetaan projektienhallintamoduulien

avulla. Myös Datafisherin kotisivut toteutetaan muunnellulla Channels-sovelluksella, joka soveltuu julkisen materiaalin esittämiseen.

LÄHTEET

Adobe. 2009 Methodology for Adobe plug-in technology study [viitattu 12.1.2009]. Saatavissa: http://www.adobe.com/products/player_census/methodology/

Akamai Technologies, Inc. 2010. eCommerce Web Site Performance Today [viitattu 20.1.2010]. Saatavissa: http://www.akamai.com/2seconds/

ENGAGEMENTdb. 2009. The world's most valuable brands. Who's most engaged? [viitattu 19.1.2010]. Saatavissa: http://www.engagementdb.com/downloads/ENGAGEMENTdb_Report_2009.pdf

Frédéric Cavazza. 2008. Social Media Landscape [viitattu 30.7.2009]. Saatavissa: http://www.fredcavazza.net/2008/06/09/social-media-landscape/

Haikala, I. & Märijärvi, J. 2006. Ohjelmistotuotanto. 11. painos. Jyväskylä: Gummerus.

jQuery. 2009. John Resig and the jQuery team [viitattu 22.5.2009]. Saatavissa: http://jquery.com

Pear Analytics. 2009. How Webpage Load Time Is Related to Visitor Loss [viitattu 20.1.2010]. Saatavissa: http://www.pearanalytics.com/wp-content/uploads/2009/08/How-Load-time-relates-to-visitor-loss.pdf

Tirronen, M. 2008. Web 2.0 – Verkon numerologia. Vaajakoski: Gummerus.

Wikipedia. 2007. MVC -arkkitehtuuri [viitattu 21.5.2009]. Saatavissa: http://fi.wikipedia.org/wiki/MVC-arkkitehtuuri

Yahoo!. 2009. Best Practices for Speeding Up Your Web Site [viitattu 12.8.2009]. Saatavissa: http://developer.yahoo.com/performance/rules.html