MODULAARINEN WEB-SOVELLUSKEHYS

Joonas Loppi

Opinnäytetyö
Kesäkuu 2012
Tietotekniikka
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotekniikka

JOONAS LOPPI:
Modulaarinen Web-sovelluskehys

Opinnäytetyö 48 sivua, josta liitteitä 1 sivua
Kesäkuu 2012

Tässä työssä suunniteltiin ja toteutettiin modulaarinen Web-sovelluskehys sekä testimoduuleita järjestelmän koekäyttämiseksi. Web-sovelluskehys mahdollistaa WWW-selainpohjaisten sovellusten luomisen. Modulaarisuus helpottaa järjestelmän laajennettavuutta erilaisten asiakkaiden tarpeisiin. Asiakkaan tarvitsemat ominaisuudet asennetaan järjestelmään omina moduuleinaan.

Tässä raportissa käydään läpi toteutetun sovelluskehyksen arkkitehtuuria, modulaarisuutta, käyttöliittymää ja kuvia esimerkkijärjestelmästä.

Asiasanat: moduuli, modulaarisuus, web, sovelluskehys, web-sovelluskehys, php

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
ICT Engineering
Software Engineering

JOONAS LOPPI:
Modular web-application framework

Bachelor's thesis 48 pages, appendices 1 pages
June 2012

This thesis documents the design and implementation of a modular Web-application framework and a few example modules to test the finished framework. Web-application framework enables the implementation of browser-based applications. Modularity eases the extensibility of the system for varying customer requirements. The requirements set by a specific customer are fulfilled by installing modules that satisfy the requirements set by the customer.

In this thesis we will walk through the architecture, modularity, user interface and pictures of an example system running with the finished framework.

Key words: module, modularity, web, web-application, application framework, php

SISÄLLYS

LYHENTEET JA TERMIT

PHP	Web-ympäristöissä käytetty avoimen lähdekoodin ohjelmointikieli.
SQL	Structured Query Language – IBM:n kehittämä rakenteellinen kyselykieli, jolla haetaan tietokannoista tietoa.
HTTP	HyperText Transfer Protocol – tiedonsiirtoprotokolla, jolla haetaan Web-sivuja ja muita resursseja Web-palvelimilta.
URL	Uniform Resource Locator – merkkijono, jolla voidaan universaalisti paikallistaa tietty resurssi.
MVC	Model-View-Controller – sunnittelumalli, jolla saadaan eroteltua käyttöliittymä ja tietokantalogiikka sovelluskoodista.
MySQL	Avoimen lähdekoodin tietokantaohjelmisto.
AJAX	Asynchronous Javascript and XML – tekniikka, jolla saadaan WWW-sivulla haettua tietoa palvelimelta lataamatta sivua kokonaan uudestaan.
ORM	Object-Relational mapping – tapa mallintaa relaatiotietokannan tietueet olio-ohjelmointipohjaisissa kielissä olioiksi.
Pub/Sub	Publish/Subscribe – ohjelmointimalli, joka perustuu tapahtumien julkaisijoihin ja kuuntelijoihin nimetyillä viestikanavilla.
AD-domain	Microsoft Windows-toimialueen käyttäjätietokanta ja hakemistopalvelu. Yrityksissä hyödynnetään muun muassa eri tietojärjestelmiin kirjautumiseen yhdellä käyttäjätunnuksella.

1 JOHDANTO

Tämän opinnäytetyön tavoitteena oli toteuttaa modulaarinen Web-sovelluskehys ja muutama esimerkkimoduuli, millä sovelluskehyksen modulaarisuutta saadaan testattua. Web-sovelluskehys on sovelluskehys, jonka päälle itse sovellus toteuttamalla saadaan halutunlainen verkkopalvelu. Sovelluskehyksen idea on toteuttaa osa sovelluksen usein tarvitsemista toiminnoista uudelleenkäytettäviksi komponenteiksi itse sovelluskehykseen sovelluksen sijaan, jolloin itse konkreettisen sovelluksen toteuttaminen helpottuu.

Sovelluskehyksen kantava ajatus on modulaarisuus eli se, että järjestelmän pystyy räätälöimään asiakkaan tarpeisiin sopivaksi lisäämällä moduuleita, jotka toteuttavat asiakkaan tarvitsemat toiminnot.

Työn idea lähti halusta testata omien ohjelmointitaitojen rajoja sekä akateemisesta mielenkiinnosta toteutustapaan - siihen, kuinka tällainen järjestelmä ohjelmoitaisiin.

Järjestelmällä ei ole vielä virallista nimeä. Se on kuitenkin kulkenut työnimellä Adepto7, jolla tässä raportissa ohjelmistoon viitataan.

2 PROJEKTI

2.1 Motivaatio

Motivaationi projektiin on se, että uskon järjestelmällä, joka integroi kaiken asiakkaan tarvitseman yhteen järjestelmään, ja tekee sen hyvin, olevan suurta kysyntää.

2.2 Kilpailijat

Maailma on pullollaan ilmaisia Web-pohjaisia ohjelmistoja, jotka ovat erinomaisia omilla yksittäisillä sovellusalueillansa. WordPress on ohjelmisto, joka on hyvin tunnettu blogipiireissä. Drupal on ohjelmisto, jolla pääasiallisesti hallitaan verkkosivujen sisältöä. MediaWiki on ohjelmisto, jolla pystyy ylläpitämään wikisivustoa. osCommerce on ohjelmisto, millä voi ylläpitää verkkokauppaa.

Miten menestyä tekemällä Web-ohjelmisto, jonka alalla on kova kilpailu ja taso, jopa ilmaisohjelmistoissa?

Tämän projektin päämääränä on tarjota yksittäinen tuote, jonka avulla asiakas pystyy moduuleita asentamalla muokkaamaan järjestelmän vastaamaan omia käyttötarpeitaan. Edullisia/ilmaisia järjestelmiä, jotka integroivat kaiken asiakkaan tarvitseman yhteen järjestelmään, ja hoitavat sen hyvin, on harvassa tai ei ollenkaan.

Esimerkiksi jos asiakas haluaa pyörittää verkkokauppaa, wikiä ja asiakkuudenhallintajärjestelmää, ovat ne usein eri tuotteensa. Näin pahimmassa tapauksessa asiakkaan käyttäjillä on eri kirjautumistunnukset näihin eri järjestelmiin, ja aina tiettyä tietoa etsittäessä täytyy tietää mistä järjestelmästä se löytyy. Olen useita kertoja kuullut työelämässä olevilta tuttaviltani valitusta siitä, kuinka turhauttavaa on kirjautua ja vaihdella eri järjestelmien välillä.

Uskon vakaasti siihen, että yksi järjestelmä, joka hoitaa kaiken, on selkeä kilpailuetu verrattuna siihen, että asiakas joutuu ylläpitämään useita eri järjestelmiä tarpeisiinsa.

3 ARKKITEHTUURI

Tässä kappaleessa käydään läpi Adepto7-järjestelmän arkkitehtuuria ohjelmointikielen, tietokannan, ytimen ja tärkeimpien tekniikoiden osalta. Modulaarisuus on yksi tärkeimpiä osia tämän työn arkkitehtuurissa, mutta se on laajuutensa vuoksi omana kappaleenaan.

3.1 Ohjelmointikieli

Ohjelmointikielenä on PHP. Hyviä ohjelmointikielivaihtohtoja on paljon, mutta päädyin PHP:hen sen takia, että minulla on sen kanssa pitkä historia ja sitä kautta vahva osaaminen. PHP on Web-ympäristöissä myös erittäin laajassa käytössä oleva kieli, joten sitä kautta on myös Adepto7:n mahdollisten asiakkaiden käytettävissä paljon palvelinympäristöjä, missä on mahdollista järjestelmää suorittaa.

Minimivaatimus PHP:n versiolle on 5.3, koska siinä versiossa PHP:hen tuotiin nimiavaruus-konsepti, jota tämän projektin moduuliajattelu käyttää vahvasti hyväkseen. Nimiavaruudet on tapa jakaa ohjelmiston komponentteja omiksi loogisiksi kokonaisuuksikseen ja varmistaa se, ettei kaksi samannimistä elementtiä sekoitu toisiinsa.

3.2 Tietokanta

Tietokantana on aluksi MySQL, mutta kaikki tietokantakyselyt kulkevat abstraktiokerroksen kautta, joka mahdollistaa käytännössä minkä tahansa SQL-pohjaisen tietokannan käyttämisen yksinkertaisen tietokanta-adapterin toteuttamalla.

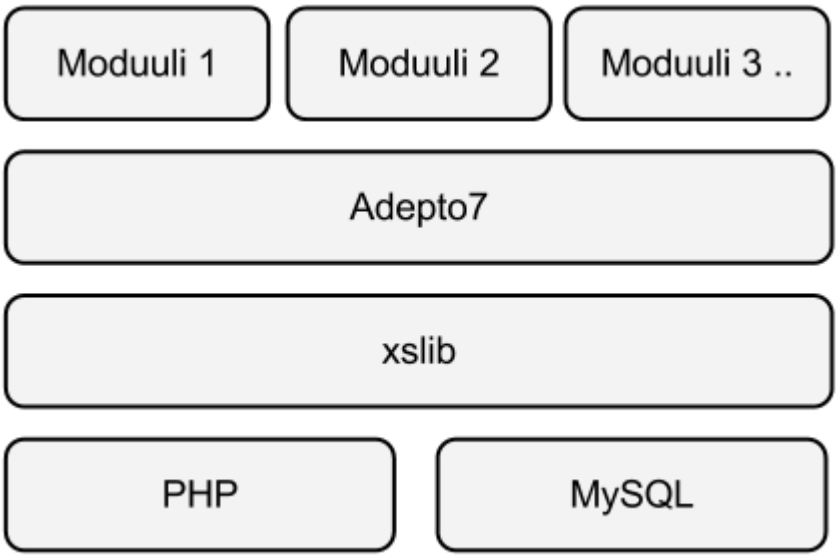
Tulevaisuudessa on tarkoitus nostaa abstraktiotasoa vielä ylöspäin siten, että järjestelmä ei vaadi alta enää SQL-tietokantaa, jolloin voimme vaihtaa alle esim. NoSQL-tietokannan. NoSQL on suosiotaan kasvattava suuntaus tietokantaohjelmistoissa, jonka ominaispiirteisiin kuuluu se, ettei kyselykielenä ole SQL, tieto ei välttämättä ole jäsennelty kuten relaatiotietokannassa, eikä tietomallille löydy välttämättä lukkoon lyötyjä tietomäärityksiä. NoSQL-tietokannat voivat joissain tapauksissa suoriutua paremmin joko tiedon nopeasta hakemisesta ja kirjoittamisesta, tai tiedon valtavasta

määrästä ja hajautettavuudesta, kuin perinteiset relaatiotietokannat. (Wikipedia - NoSQL)

3.3 Ydin, Adepto7

Järjestelmän ydin, Adepto7, tarjoaa moduuleille ajoympäristön sekä rajapinnat, joilla moduuli saa tuotua toiminnallisuutensa käyttäjien nähtäviksi. Moduulit kirjoitetaan Adepto7:n rajapintoja vasten, eli ne vaativat Adepto7:n olemassaolon toimiakseen.

Adepto7 pyörii xslib:n päällä, joka on PHP:llä kirjoitettu luokkakirjasto yleisempien ohjelmointiongelmien ratkaisemiseksi. xslib käyttää vahvasti hyväkseen PHP:n tarjoamia valmiita rajapintoja muun muassa tietokannan kanssa keskustelemiseen ja ulkoisten järjestelmien kanssa keskustelemiseen TCP/IP –soketeilla.



KUVIO 1. Kaavio arkkitehtuurin eri kerroksista

3.4 Lazy loading -mekanismi

PHP on dynaaminen kieli. Mikäli ohjelmakoodissa halutaan suorittaa koodia tai viitata koodiin, joka sijaitsee toisessa kooditiedostossa, täytyy siihen viitata include-käskyllä, jolloin PHP-tulkki käy lukemassa ja parsimassa viitatun kooditiedoston tiedostojärjestelmästä. Mitä enemmän include-käskyjä koodi sisältää, sitä hitaampaa ohjelmakoodista tulee, sillä kooditiedostojen lataaminen ja parsiminen tiedostojärjestelmästä on suhteellisen hidasta. Parsiminen tarkoittaa sitä, että PHP-tulkki muuntaa vapaasti kirjoitetun PHP-koodin tiukasti määritellyiksi PHP-tulkin operaatiokoodeiksi. Tämä on samankaltainen, mutta eri asia kuin kääntäminen prosessorin suoritettavaksi koodiksi.

xslib:n käytössä on lazy loading-mekanismi, joka on toteutettu PHP:n __autoload() –rajapinnalla. Käytännössä tämä merkitsee sitä, ettei include-käskyjä tarvitse käyttää itse ohjelmakoodissa. Kun ohjelmakoodista viitataan luokkaan, jota ei ole vielä ladattu muistiin, PHP pyytää __autoload() –rajapintaa suorittamaan include-käskyn, jolla luokka saadaan ladattua muistiin. Tällä mekanismilla päästään eroon siitä, että luokan tarvitsee ladata kaikki mahdolliset kooditiedostot muistiin, mitä se omaa koodia suorittaessaan saattaisi tarvita. Koska ohjelmakoodin vuo voi erota huomattavastikin suorituksesta toiseen, eivät ohjelmakoodin viittaukset muihin ohjelmatiedostoihin ole aina vakiot, jolloin tällä mekanismilla voidaan säästää huomattavastikin suoritinaikaa.

3.5 Välimuisti

Tietokannasta tietojen hakeminen on yksi Web-ohjelmiston eniten aikaa kuluttavista operaatioista. Kaikki tietokantaoperaatioihin saatava helpotus nopeuttaa Web-ohjelmiston suorituskykyä, ja sitä kautta sivulatausten nopeutta.

Välimuistina toimii memcached. memcached on avain-arvopareihin perustuva avoimen lähdekoodin hajautettu olioiden välimuistipalvelin. (memcached)

Jos haluamme tallettaa välimuistiin tietokannasta löytyvän käyttäjälistan, voimme keksiä sille kuvitteellisen välimuistiavaimen ”userlist”. Käyttäjälistaa kaivatessamme voimme kysyä välimuistilta, löytyykö siltä avaimelle ”userlist” tietoja. Jos löytyy, ei käyttäjälistaa tarvitse hakea tietokannasta. Jos ei löydy, haetaan käyttäjälista tietokannasta ja tallennetaan tulosjoukko ”userlist” –avaimen alle välimuistiin. Ohjelmakoodissa käyttäjiä muokattaessa täytyy kuitenkin muistaa tyhjätä välimuistista ”userlist” –avain, ettei välimuistista haeta seuraavalla sivulatauksella vanhentuneita tietoja.

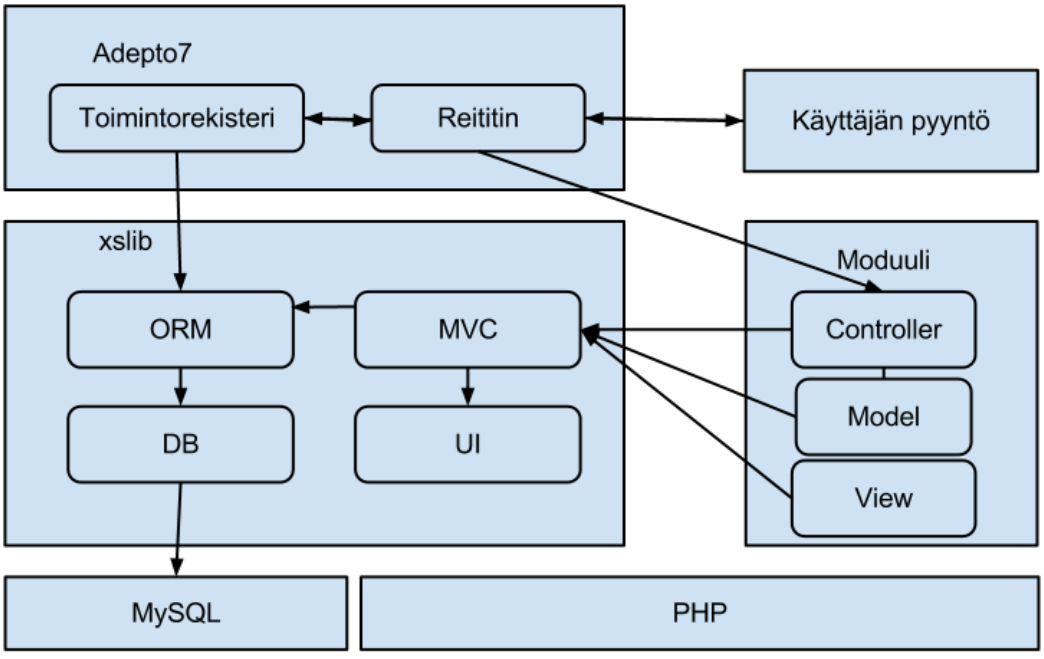
Välimuistia voi käyttää tietokantatulosten lisäksi muidenkin laskentaintensiivisten tietojen välimuistitukseen.

3.6 Pyyntö-vastaus -prosessi

Web-pohjaisissa järjestelmissä käyttäjä siirtyy tiettyyn URL-osoitteeseen selaimellansa. Selain muodostaa tästä URL-osoitteesta HTTP-pyynnön, lähettää sen palvelimelle, palvelin prosessoi pyynnön, tuottaa siihen sopivan vastauksen ja lähettää sen asiakkaalle HTTP-vastauksena. Käyttäjän selain näyttää asiakkaalle tämän etäjärjestelmän tuottaman vastauksen.

Kuvion 2 kaaviossa käymme läpi, kuinka Adepto7 vastaa käyttäjän pyyntöön. Kaavion vuo kulkee seuraavasti:

1) Käyttäjän selain lähettää pyynnön reitittimelle (”Käyttäjän pyyntö”).
2) Reititin selvittää käyttäjän pyyntöä vastaavan objektin, ja hakee sille toimintorekisteristä kontrollerin, joka osaa vastata objektia vastaavaan pyyntöön.
3) Reititin antaa käyttäjän pyynnön kohdassa 2 löydetylle moduulin kontrollerille.
4) Kontrolleri tuottaa MVC-mallilla objektia vastaavan näkymän ja lähettää sen vastauksena käyttäjän selaimelle.



KUVIO 2. Pyyntö-vastaus –prosessi

Kuvitteellisen asiakkaan Adepto7-järjestelmän osoite on http://esimerkki.com/. Asiakkaan järjestelmän käyttäjä kirjoittaa selaimeensa URL-osoitteen

http://esimerkki.com/asiakas/tamk, joka viittaa esimerkki.com Adepto7-järjestelmän asiakasrekisterin asiakkaaseen, jonka nimi on TAMK.

Käyttäjän selain lähettää HTTP-pyynnön Adepto7:n reititin-komponentille. Reititin-komponentin tehtävä on paikallistaa pyydetty objekti Adepto7:n tietorakenteista ja löytää sille toiminto, joka osaa piirtää tätä objektia vastaavan WWW-sivun. Kun objekti on löytynyt, tietää reititin myös objektin tyypin. Objektin tyypillä löydetään toimintorekisteristä tieto moduulista ja sen kontrollerista, joka osaa vastata pyyntöön.

Reititin välittää pyynnön asiakasrekisteri-moduulin kontrollerille, joka hakee tietokannasta tarvittavat tiedot ja tuottaa käyttäjälle näytettävän WWW-sivun.

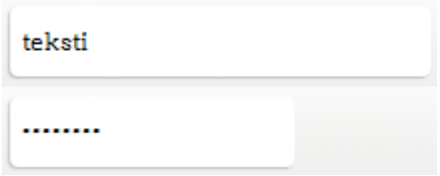
3.7 xslib

xslib on PHP:llä itse kirjoittamani laajahko luokkakirjasto, joka on suunniteltu uudelleenkäytettäväksi erilaisissa projekteissa. xslib tarjoaa ohjelmalogiikkaa yleisten ohjelmointiongelmien ratkaisemiseksi. Muutamia esimerkkejä xslib:n tarjoamista komponenteista ovat käyttöliittymäkomponentit, tiedon haku tietokannasta, verkkoliikenne sekä ulkoisten järjestelmien rajapintojen hyödyntäminen.

3.7.1 xslib:n käyttöliittymäkomponentit

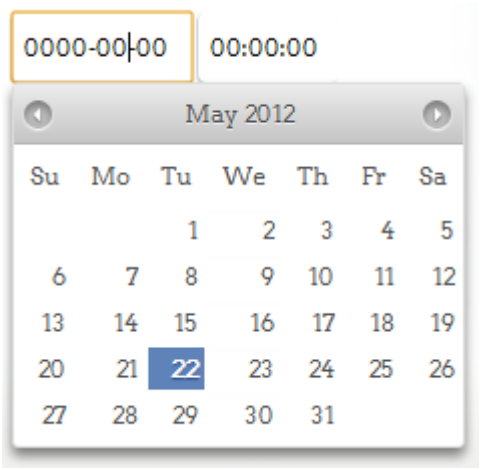
xslib tarjoaa sovelluskehittäjän käytettäväksi käyttöliittymäkomponentteja muun muassa syötteiden käsittelyyn.

Teksti- ja salasanakentillä syötetään tavallista tekstiä. Salasana –kenttä on muuten tekstikenttää vastaava, mutta salasanakentän sisältöä ei esitetä selkokielisenä.



KUVA 1. Teksti- ja salasanakentät

Ajanhetken syöttökenttä kysyy käyttäjältä erikseen päivämäärän sekä kellonajan. xslib tarjoaa myös pelkän päivämäärä -kentän.

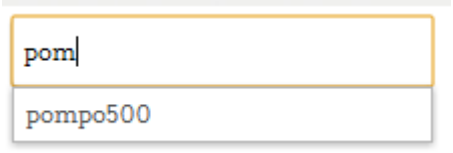


KUVA 2. Ajanhetken syöttökenttä

Automaattitäydennys-kenttä on ollut yksi haastavimmista kentistä toteuttaa, sillä se hakee palvelimelta reaaliajassa ehdotuksia käyttäjän syötteelle. Automaattitäydennys –kentälle määritellään koodissa tiedot tietorakenteesta, mistä kenttä hakee käyttäjän syötteelle ehdotuksia.

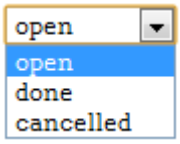
Käyttäjä pystyy valitsemaan kentän alapuolelle avautuvista ehdotuksista oikean joko hiirellä klikkaamalla tai siirtymällä ehdotuksen kohdalle näppäimistön nuolinäppäimillä.

Käyttäjän valittua jonkun ehdotuksista, tallentuu valitun tietueen tunnistetieto kentän piilotettuihin tietoihin. Tunnistetieto yksilöi yksiselitteisesti valitun tietueen, sillä esim. kontaktirekisteristä nimellä haettaessa kontaktin nimi ei välttämättä ole yksilöivä tieto. Lomakkeen lähetettäessä lähetetään vain valitun tietueen tunnistietieto.



KUVA 3. Automaattitäydennys –syöttökenttä

Pudotusvalikko-syötekentälle määritellään ennalta rajatut vastausvaihtoehdot, mistä käyttäjä saa valita tietyn vaihtoehdon.

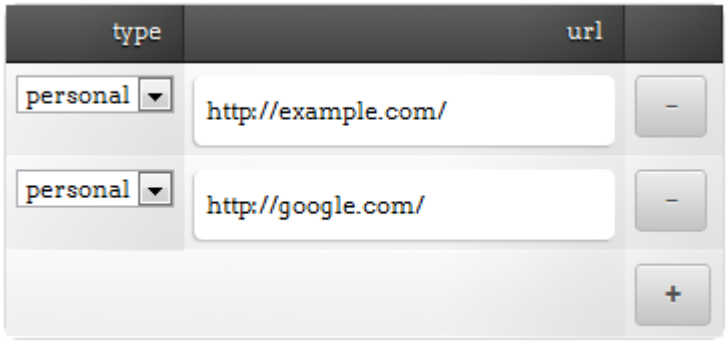


KUVA 4. Pudotusvalikko –syöttökenttä

Painonappeja käytetään esimerkiksi kun halutaan poistaa tietue tietokannasta tai avata tietty tietue.

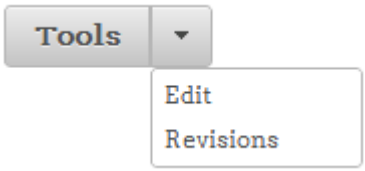
KUVA 5. Painonappi

Muokattavassa taulukossa voi olla 0-n riviä. Yhdellä rivillä on 1-n solua ja viimeisenä rivin poisto -nappi. Yhdellä solulla on aina jokin käyttöliittymäkomponentti. Muokattava taulukko oli teknisesti haastavin toteuttaa, sillä uuta riviä lisättäessä rivin HTML-sisältö haetaan AJAXilla palvelimelta, ja palvelimen tulee tällöin tietää millaiset käyttöliittymäkomponentit riville tulee lisätä.



KUVA 6. Muokattava taulukko

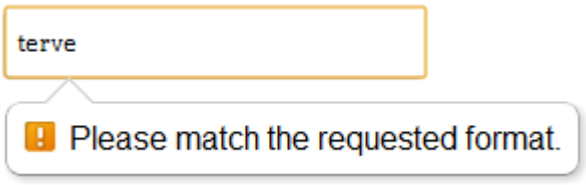
Painonappi/pudotusvalikko –yhdistelmä sisältää tavallisen painonapin lisäksi alaspäin osoittavan nuolisymbolin, jota klikkaamalla saa esille lisätoimintoja.



KUVA 7. Painonappi/pudotusvalikko –yhdistelmä

3.7.2 Käyttöliittymäkomponenttien syötteiden validointi

Useimmille syötekentille pystyy määrittelemään syötteen muodon, mitä kentän syötteen tulee noudattaa. Mikäli käyttäjä on lähettämässä lomaketta, jonka tulisi tallentaa jotain tietoa, pystytään lomakkeessa olevat mahdolliset virheet kertomaan käyttäjälle ennen tietojen lähetystä palvelimelle.



KUVA 8. Sähköpostiosoitteen syöttökenttä. Syöte ei ollut kelvollinen sähköpostiosoite.

Syötteen muoto määritellään käyttöliittymäkomponentille säännöllinen lauseke –muodossa (engl. regular expression). Säännölliset lausekkeet ovat olleet käytössä tietotekniikassa jo 1960-luvulta asti, ja niillä voidaan muun muassa varmistaa merkkijonon vastaavan haluttua muotoa.

Esimerkki säännöllisestä lausekkeesta: [a-z]{3}\d+

Ylläoleva säännöllinen lauseke hyväksyy merkkijonon, jossa on täsmälleen kolme peräkkäistä kirjainta väliltä a-z, jonka jälkeen on 1 tai useampi numero. Lauseke hyväksyy syötteen ”cat123”, mutta ei hyväksy syötettä ”ab9” tai ”abc”.

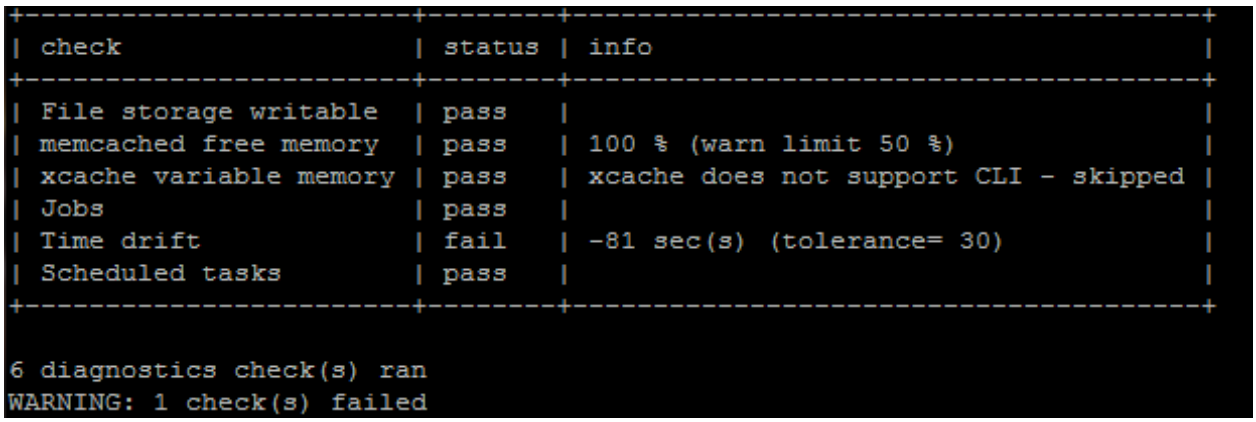
3.8 Toimintavarmuus

Toimintavarmuus varmistetaan diagnostiikka-raporteilla, jotka tarkkailevat palvelun tilaa tekemällä erilaisia diagnostiikkatestejä. Esimerkiksi järjestelmän keskustellessa ulkopuolisten järjestelmien kanssa käytetään usein sellaisia salaustekniikoita, jotka vaativat molempien järjestelmien kellon olevan samassa ajassa, tietyllä toleranssilla. Näin ollen yksi oleellinen diagnostiikka-tarkistus on kysyä omalta palvelimelta mitä kello on omasta mielestämme. Tätä omaa käsitystä ajasta verrataan ulkopuolisen palvelimen kellonaikaan, jonka tiedämme olevan oikeassa. Mikäli nämä kaksi käsitystä kellonajoista eroaa yli tietyn toleranssin, teemme tästä hälytyksen.

Toinen tärkeä asia on taustaprosessien monitorointi. Taustaprosesseille on määritelty aikaleima, jolloin taustaprosessin pitäisi lähteä käyntiin. Mikäli taustaprosessi ei ole tietyn toleranssin sisään tästä suunnitellusta ajanhetkestä lähtenyt käyntiin, tehdään tästäkin hälytys.
Myös mikäli taustaprosessin ajamisessa tapahtuu tarpeeksi monta virhettä peräkkäin, kytketään taustaprosessi pois ajettavien listalta ja tehdään tästäkin hälytys.

Näitä useita diagnostiikkatestejä tarkastelemalla saamme siis pääteltyä, onko palvelun taso kunnossa vai epäkunnossa. Siinä tilanteessa kun palvelu menee alas, ei palvelu itse tästä pystyisi varmuudella lähettämään hälytystä eteenpäin, joten ulkopuolinen palvelu on säädetty kyselemään 15 minuutin välein diagnostiikka-rajapinnan kautta yksinkertaisen kysymyksen: ”Oletko kunnossa?”. Ylläpitäjälle lähtee hälytys tämän ulkopuolisen monitorointipalvelun kautta, mikäli yksikin seuraavista ehdoista toteutuu:

a) Yksikin diagnostiikka-tarkistus epäonnistuu.
b) Palvelu ei ole siinä kunnossa että pystyy edes raporttia ajamaan.
c) Ei ole edes tavoitettavissa verkon kautta.



KUVA 9. Diagnostiikkatestit komentoriviltä ajettuna

Kuva 9 esittää, kuinka ”Time drift” –diagnostiikkatesti epäonnistuu, koska sille asetettu toleranssi (30 sekuntia) ylittyy.

3.9 Turvallisuus

Tässä kappaleessa käydään läpi käyttäjän turvallisuuteen vaikuttavat tekijät: yhteyden suojaus, arkaluontoisten tietojen salaus sekä sisäänkirjautuminen.

3.9.1 Yhteyden suojaus

Yhteyden suojaus ohjelmistossa on otettu huomioon mahdollistamalla tuki HTTPS-protokollalle niissä tapauksissa, missä HTTPS täytyy ottaa sovellusta toteutettaessa erityisesti huomioon. HTTPS suojaa käyttäjä <-> palvelin –yhteyden siten, että sitä ei pysty ulkopuolinen salakuuntelemaan. HTTPS varmistaa myös käyttäjälle sen, että palvelimen identiteetti on se, mitä palvelin väittää olevansa - eikä esimerkiksi hyökkääjän palvelin.

3.9.2 Arkaluontoisten tietojen salaus

Käyttäjien turvallisuuteen vaikuttaa myös se, ettei heidän arkaluontoisia tietojansa (salasana, luottokorttitiedot ym.) varastoida salaamattomana. Siinä harmillisessa tilanteessa, että joku hyökkääjä saa murtauduttua palveluun, saadaan tietomurron vakavuutta minimoitua mikäli arkaluontoiset tiedot on saatu salattua siten, että salauksen purku on joko mahdotonta tai edes vaikeaa.

Yleinen käytäntö salasanojen suojaamiseen on yksisuuntainen hash-algoritmi. Hash-algoritmi laskee salasanasta tiivisteen, jota tietämällä ei ole mahdollista päätellä alkuperäistä salasanaa. Esimerkiksi sanan ”testi” MD5-tiiviste on 9627df7a4a5b849f67fce863e82adc71.

Hyvä käytäntö hash-algoritmin tuoman suojan lisäksi on lisätä ennen hash-algoritmin käyttöä salasanaan vielä ns. salt, joka estää rainbow table –hyökkäyksen käyttämisen. Rainbow table –hyökkäys on sellainen, missä hyökkääjä etsii tiivistettä tietokannasta, johon on etukäteen laskettu tunnettuja sanastoja käyttämällä valmiiksi suuri määrä tiivisteitä. Esim. suomen kielen sanastoa käyttämällä voidaan rakentaa tietokanta, joka sisältää tiedon siitä, että sana ”testi” antaa tiivisteen 9627df7a4a5b849f67fce863e82adc71. Tästä tietokannasta tiivisteellä etsittäessä siis saadaan alkuperäinen salasana, ”testi”, selville. Tämä on ns. rainbow table-hyökkäyksen perusperiaate. Saltin käyttö ennen hash-algoritmia lisää salasanan perään jonkin kiinteän sanan, saltin, joka on esimerkiksi ”kl3m45”. Näin ollen hyökkääjän tietokannasta täytyisi nyt löytyä etukäteen laskettu tiiviste sanalle ”testikl3m45”, joka on selvästikin epätodennäköistä, sillä sitä ei todennäköisesti löydy mistään sanastosta eikä millään laskennallisella [sanastosta löytyvä sana][numerosarja] yhdistelmällä tai vastaavalla.

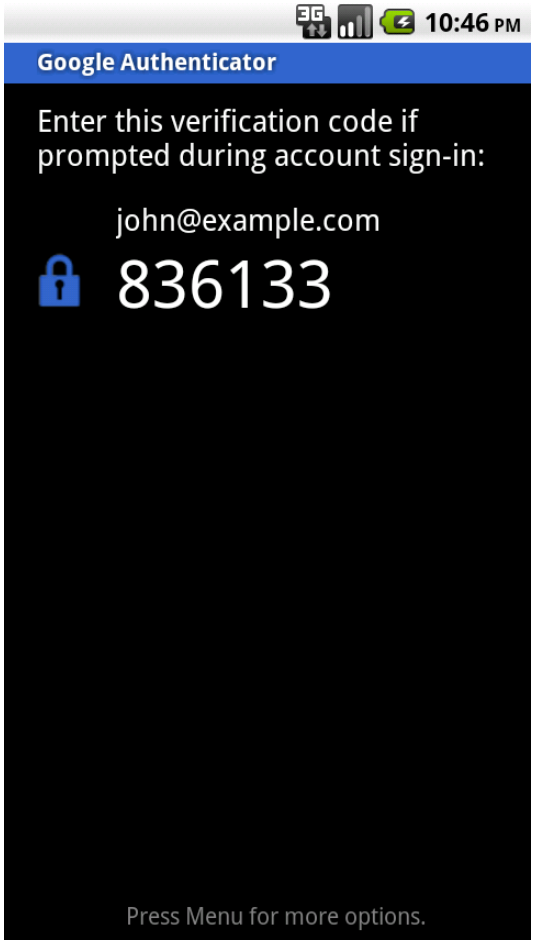
Luottokorttitietojen salaamiseen täytyy käyttää kaksisuuntaista salausalgoritmia, sillä tiedot täytyy saada purettua takaisin salatusta muodosta, toisin kuin salasanan tapauksessa. Kaksisuuntaisessa salauksessa haasteena on salausavaimen varastointi turvallisesti siten, että hyökkääjän päästessä järjestelmään ei hyökkääjä pääsisi käsiksi myös salausavaimeen, sillä salausavaimen varastamalla hyökkääjä saa purettua salatut tiedot.

3.9.3 Sisäänkirjautuminen

Sisäänkirjautumisessa kysytään käyttäjätunnusta ja salasanaa. Mikäli Adepto7:n on asennettu LDAP-kirjautumismoduuli, pääsee järjestelmään kirjautumaan myös yrityksen AD-domain –tunnuksilla. Käyttäjän on myös mahdollista kirjautua muiden ulkopuolisten palveluiden tunnuksilla, esim. Facebookin, Twitterin tai Googlen, mikäli Adepto7:n on asennettu tällainen ulkopuolisen järjestelmän kirjautumisen toteuttava moduuli.

Yllä mainittuja autentikointitapoja kutsutaan ensimmäisen tekijän tunnistautumiseksi. Tämä riittää kirjaamaan käyttäjän sisään, mikäli käyttäjä ei ole asettanut tilinsä asetuksista toisen tekijän tunnistusta päälle. Toisen tekijän tunnistus tarkoittaa sitä, että käyttäjä haluaa kasvattaa tietoturvaansa vaatimalla sisäänkirjautumisen yhteydessä vahvemman todistuksen siitä, että sisäänkirjautumassa olevalla taholla on oikeus käyttäjän tiliin. Tavallisesti toisen tekijän tunnistus toteutetaan käyttäjän hallussa olevalla laitteella (erikoislaite tai matkapuhelin), joka laskee kertakäyttöisiä koodeja ajan funktiona muuttumattomasta salausavaimesta. Näin ollen mahdollisen hyökkääjän on saatava haltuunsa käyttäjän tunnus-salasana –parin lisäksi myös käyttäjän toisen tekijän tunnistautumislaite, mikä tekee hyökkäyksestä huomattavasti vaikeampaa.

Adepto7:ssä toisen tekijän tunnistus edellyttää ensimmäisen tekijän tunnistautumisen lisäksi kertakäyttöistä koodia, jonka käyttäjä lukee omasta puhelimestaan TOTP-yhteensopivalla ohjelmistolla (IETF - TOTP). Ohjelmistoja on saatavilla ainakin Android-, iPhone-, Windows- ja Blackberry –puhelimille.



KUVA 10. Toisen tekijän tunnistautuminen Android-puhelimessa

Kuva 10 on Google Authenticator –sovelluksesta, joka toteuttaa TOTP-standardin mukaisen toisen tekijän tunnistautumisen.

KUVA 11. Salausavaimen siirto puhelimeen viivakoodin avulla

Adepto7 antaa kuvan 11 mukaisen kaksiulotteisen viivakoodin, minkä avulla käyttäjä saa puhelimen kameran avulla siirrettyä salausavaimen puhelimeensa ilman sen manuaalista syöttämistä. Yllä mainituista sovelluksista ainakin Google Authenticator tukee tätä siirtotapaa.

4 ADEPTO7-MODUULI

Tässä kappaleessa käydään läpi, mitä modulaarisuus on ja miten se on toteutettu Adepto7:ssä.

4.1 Moduulin yleinen määritelmä

Moduuli on ohjelmiston osa, joka toteuttaa yhden asiakokonaisuuden itsenäisesti. Moduuli on käytännössä sama kuin lisäosa. Esimerkkinä Web-selaimet, joihin pystyy lisäämään ominaisuuksia asentamalla niihin lisäosia.

4.2 Modulaarisuus Adepto7:ssä

Aiemmin mainittu palvelun räätälöitävyys asiakkaan tarpeisiin mahdollistetaan modulaarisuudella. Adepto7 on runko, jonka päälle pystytään rakentamaan moduuleita, jotka tuovat järjestelmään uusia ominaisuuksia.

Yksi esimerkki on kommentointimoduuli, joka sallii palvelun käyttäjien kommentoivan erilaisia objekteja. Objektit ovat abstrakti esitys konkreettisista tietotyypeistä (esim. asiakas, lasku, käyttäjä tai blogikirjoitus), mitä järjestelmä pitää sisällään. Moduulien käsitellessä konkreettisia tietotyyppejä abstraktilla tasolla mahdollistuu se, että moduuli pystyy käsittelemään asiaa, josta se ei oikeasti ymmärtäisi mitään. Esim. kommentointimoduuli voi sallia kommentin liittämisen laskuun, vaikka kommentointimoduuli ei tiedä mikä lasku olisi tai miten sitä muilta osin käsiteltäisiin. Tämä mahdollistaa myös sen, että eri kehittäjien moduulit pystyvät tietyissä tilanteissa tekemään yhteistyötä toistensa tietotyypeillä.

Modulaarisuuden saavuttamisessa on omat haasteensa. Esim. rajapintojen suunnittelu järkevästi siten, että järjestelmä tarjoaa ohjelmistokehittäjille tarpeeksi hyvät rajapinnat erilaisiin tilanteisiin, joihin moduulin tulisi reagoida. Haastetta on myös kaiken tämän suunnittelussa siten, etteivät rajapinnat paisu liian monimutkaisiksi ymmärtää.

4.3 Adepto7-moduulin tietorakenteet

Liitteessä 1 on UML-luokkakaavio, joka kuvaa Adepto7-moduulin keskeisten tietorakenteiden suhteita. Kaaviossa olevat luokat eivät sisällä juurikaan ohjelmalogiikkaa, sillä luokat ovat lähinnä tietotyyppejä, joita käytetään hyväksi MVC-ohjelmointimallin kontrollereissa. Tässä kappaleessa käydään läpi kaaviossa mainitut keskeisimmät tietorakenteet.

4.3.1 Moduuli

Moduulilla on nimi, koodi, versio ja tieto siitä, onko moduulin ajo sallittu. Moduulin koodi on Java-tyylinen pakettinimi, esimerkiksi ”fi.xs.customer” voisi viitata xs.fi organisaation tekemään asiakasrekisteri –moduuliin. Koodia käytetään ohjelmakoodissa viittaamaan moduuliin yksiselitteisesti.

4.3.2 Pageletit

Moduulilla on 0..* pageletiä. Pageletit ovat näkymiä, joita yhteen kokoamalla saadaan koostettua lopullinen käyttäjälle näytettävä sivu. Esimerkiksi yksi sivu voi koostua laskutus -moduulin laskun näyttäminen –pageletistä sekä kommentointi -moduulin kommenttien näyttäminen –pageletistä. Yhden sivun pilkkominen pienemmiksi asiakokonaisuuksisksi, pageleteiksi, mahdollistaa sivun sisällä olevan yksittäisen asian päivittämisen ilman, että koko sivua täytyy ladata uudestaan. Esimerkiksi jos joku käyttäjä kommentoi tiettyä laskua, niin uuden kommentin näyttämiseksi lasku-sivulla vaaditaan vain kommentti-pageletin päivittäminen. Pageleteillä voi olla parametreja, jotka voivat vaikuttaa pageletin piirtämiseen. Esim. Google Maps –moduulin kartta-pageletillä on parametri ”Address” –joka määrää sen, minkä alueen kartta piirretään.

4.3.3 Tietomalli

Moduulilla on 0..* tietomallia (engl. model). Tietomalli määrittelee moduulin Adepto7:lle tarjoamat tietotyypit abstraktilla tasolla siten, että Adepto7 osaa käsitellä niitä siltä osin kuin on välttämätöntä. Tietomalli määrittelee muun muassa sen, mitkä sivut osaavat suorittaa tietomallin tietueelle jonkun toiminnon. Esimerkiksi kun käyttäjä haluaa katsella laskua, on tietomallilla tieto siitä, mikä sivu hoitaa laskun näyttämisen.

Jos laskua halutaan muokata, on tietomallilla tieto siitä, mikä sivu hoitaa laskun muokkaamisen. Sivu on kokoelma pageletejä. Yhdellä sivulla voi olla eri moduulien pageletejä.

4.3.4 Parametri

Moduulilla on 0..* parametria. Parametrit ovat asetuksia, joita käyttäjä pystyy muokkaamaan käyttöliittymän kautta. Parametrilla on tyyppi, joka voi olla esimerkiksi merkkijono, kokonaisluku, ajanhetki tai olioviite. Parametri voi olla myös listallinen mitä tahansa edellä mainittuja tyyppejä. Esimerkiksi järjestelmän ydin –moduulilla on parametri ”sitename”, joka määrittelee sivuston nimen.

4.3.5 Tiedosto

Tiedosto on Adepto7-järjestelmässä mikä tahansa tiedosto, joka syntyy käyttäjän toimien myötä. Adepto7:n ohjelmakoodit eivät ole tässä kappaleessa mainittuja tiedostoja. Tiedostolla voi olla monta sijaintia – eli vaikka tiedosto olisi tallennettuna kahdessa paikassa tiedostojärjestelmää tai kahdessa eri tiedostojärjestelmässä, lasketaan tiedosto silti yhdeksi. Jokaisella sijainnilla on tieto siitä, missä tiedostojärjestelmässä tiedosto sijaitsee. Tiedostojärjestelmä voi olla paikallinen tai etäjärjestelmä.

Moduulilla on oletus ”file storage policy”, joka määrittelee miten ja mihin moduulin tiedostot tallennetaan. Esimerkiksi järjestelmään voidaan määritellä, että oletuksena kaikki dokumentit –moduulin tiedostot tallennetaan paikallisesti, mutta kaikki kuvagalleria –moduulin tiedostot tallennetaan nopean tallennuksen vuoksi vain tilapäisesti paikalliseksi, mutta tallennuksen jälkeen käynnistetään taustatehtävä siirtämään paikallisesti tallennettu tiedosto johonkin etäjärjestelmään.

4.3.6 Pub/Sub –järjestelmän subscriptionit

Moduulilla on 0..* subscriptionia, jotka kertovat mistä Pub/Sub –järjestelmän aiheiden viesteistä moduulin eri kontrollerit ovat kiinnostuneita. Jos moduuli haluaa tarjota ominaisuutta, joka osaa jakaa jonkun sivun linkin Facebookiin, voi moduulilla olla subscription aiheelle ”com.adepto7.share.offer”. Tällä subscriptionilla on tiedossa, mikä

moduulin kontrolleri ja action osaa vastata tämän aiheen kysymykseen. Kun aiheelle lähetään viesti, kuljettaa Adepto7:n Pub/Sub –alijärjestelmä tiedon tästä oikeille kontrollereille.

4.3.7 Ajastettu tehtävä

Moduulilla on 0..* ajastettua tehtävää. Ajastetuilla tehtävillä pystytään määrittelemään toistuvia tehtäviä, joita suoritetaan tietyin väliajoin. Yksi esimerkki ajastetuista tehtävistä on Google Calendar –moduuli, joka synkronoi tunnin välein käyttäjien mahdollisista Google Calendar –kalentereista tapahtumat Adepto7:n kalenteriin.

4.3.8 Moduulin muut tietorakenteet

Moduulin muut tietorakenteet määrittyvät moduulin itsensä tarpeiden mukaan. Nämä ovat käytännössä SQL-tauluja, joissa jokainen taulun rivi mallintuu ohjelmointikielen olioksi ORM-kerroksen kautta.

Otetaan esimerkiksi kuvitteellinen blogimoduuli, jonka tietorakenteet voisivat olla:

- Blogit
- Blogikirjoitukset.

Näistä muodostetaan SQL-taulut. Blogit-tietorakenteessa on 0..* suhde blogikirjoitukseen, eli yhdessä blogissa voi olla nollasta äärettömään määrä blogikirjoituksia.

Blogiohjelmistoille tyypillinen blogikirjoitusten kommentointi ei vaadi omaa tietorakennettansa tässä modulaarisessa ohjelmistossa, sillä blogikirjoitusta näyttäessä moduuli lähettää Pub/Sub –järjestelmään signaalin ”jos joku moduuli osaa hoitaa kommentoinnin tälle blogikirjoitukselle, niin näytä kommentointilomake tässä”.
Myöskään moduulin ei tarvitse itse toteuttaa käyttäjärekisteriä, sillä blogimoduuli voi käyttää käyttäjärekisteri-moduulia hyväkseen.

4.4 Ytimen moduulille tarjoamat palvelut

Jotta moduulin toiminta olisi tehokasta ja sen ohjelmointi mielekästä, ei jokaisen moduulin tarvitse eikä kannata toteuttaa Web-sovelluksissa usein tarvittavia ominaisuuksia aina itse, vaan Adepto7 tarjoaa moduulien käyttöön usein käytettäviä rajapintoja. Näitä ovat muun muassa parametrijärjestelmä, käyttäjien hallinta, käyttäjäoikeudet, ajastetut tehtävät ja diagnostiikka-ajot.

4.4.1 Pub/Sub järjestelmä

Jotta ohjelmistosta saadaan modulaarinen, tulee moduulien pystyä keskustelemaan keskenään tietämättä toisistaan. Mikäli moduuliin ohjelmoitaisiin suora viittaus toiseen moduuliin, on moduuli riippuvainen toisen moduulin läsnäolosta, ja se ei silloin ole kovin modulaarista koska modulaarisuuden idea on toteuttaa yksi asiakokonaisuus itsenäisesti. Pub/Sub on hyvä suunnittelumalli tähän tarpeeseen.

Teemme esimerkkinä moduulin, jonka tarkoituksena on sallia tietyn sivun jakaminen ”Jaa linkki” -nappia painamalla johonkin palveluun. Sen sijaan, että ohjelmoisimme tähän moduuliin sisäänrakennetun tuen Twitterille ja Facebookille, voimme tehdä tästä jako-moduulista sellaisen, että se kysyy kysymyksen ”osaako joku moduuli jakaa tämän objektin linkin johonkin Web-palveluun?”. Kaikki moduulit, jotka vastaavat ”kyllä” tähän kysymykseen, listataan tämän napin pudotusvalikkoon ikoneineen. Näin jätämme mahdollisuuden auki kolmansille osapuolille rakentaa moduuleita, jotka pystyvät integroitumaan tähän linkkien jako-ominaisuuteen.

```
46          $providers = array();
47
48          foreach ($this->kernel->publish('com.adepto7.share.offer') as $provider)
49          {
50              $providers[] = $provider->definition;
51          }
```

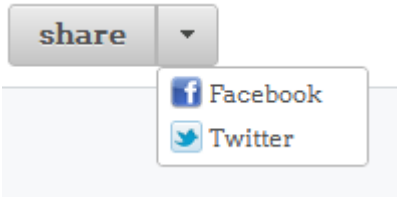
KUVA 12. Rivillä 48 julkaistaan (”publish”) signaali aiheeseen (”topic”) ”com.adepto7.share.offer”.

Kuvan 12 koodi pyytää metodilla ”publish” kanavan ”com.adepto7.share.offer” kuuntelijoilta listan moduuleista, jotka haluavat tarjota linkkien jako -ominaisuuden.

```
16      /*!
17          @topic com.adepto7.share.offer
18      */
19      public function getDefinition($request, $response)
20      {
21          $response->definition = array(
22              'name' => 'Facebook',
23              'code' => 'facebook',
24              'icon' => 'app/fi.xs.facebook.share/icons/facebook.png'
25          );
26      }
```

KUVA 13. Facebookin linkkejä jakavan moduulin vastaanottaja kuvan 12 kysymykselle

Kuvan 13 rivin 19 funktio on vastaanottaja kanavan ”com.adepto7.share.offer” viesteille. Funktio palauttaa kuvauksen itsestään (nimi, koodi ja ikoni), millä ylemmän tason jako-nappi osaa lisätä sen valikkoon ja välittää napinpainalluksen takaisin tälle luokalle, mikäli Facebook-jako pudotusvalikosta valitaan.



KUVA 14. Lopputulos

Kuvan 14 rivit Facebook ja Twitter ovat omia moduleita, jotka ovat ilmoittautuneet kykeneväksi jakamaan tämän sivun linkin.

4.4.2 Parametrointijärjestelmä

Parametrit ovat moduulikohtaisia asetuksia, joilla ohjataan ohjelmalogiikan toimintaa.

Parametrillä on tietoina:

- Moduuli, joka omistaa parametrin
- Koodi, jolla ohjelmistokoodissa viitataan parametriin
- Tietotyyppi
- Arvo
- Lippu, joka kertoo päivitetäänkö parametrin arvojen muutokset reaaliajassa selaimeen

Tietotyyppejä on:

- Kokonaisluku
- Liukuluku
- Totuusarvo
- Päivämäärä
- Päivämäärä + kellonaika
- Olioviite
- Tekstirivi
- Monirivinen teksti

Parametri pystyy myös määrittelemään, mikäli parametrin arvoja pystyy olemaan monta kappaletta, esim. listallinen tekstirivejä.

Adepto7 tarjoaa moduulien käyttöön ohjelmistorajapinnan, millä näitä parametreja voi hakea. Järjestelmän hallintapaneeli mahdollistaa käyttäjän muokata parametrien arvoja ilman, että moduulin kehittäjän tarvitsee itse toteuttaa parametrin muokkauksen käyttöliittymää.

```
14          $serverUrl = \fi\xs\system\Setting::getString('fi.xs.system', 'dx.time_drift.server_url');
15
16          $httpRequest = new \xs\Io\Net\Http\Request($serverUrl);
17
18          $httpRequest->connect();
19
20          $ourTime = \xs\Time\DateTime::fromNow();
21
22          $httpResponse = $httpRequest->get();
23
24          $jsonTime = json_decode($httpResponse->getResponseBody());
25
26          $theirTime = \xs\Time\DateTime::fromTextInUtc($jsonTime->datetime);
27
28          $difference = $ourTime->difference($theirTime);
29          $tolerance = \fi\xs\system\Setting::getInteger('fi.xs.system', 'dx.time_drift.tolerance');
30
31          if (abs($difference) > $tolerance) {
32              $response->check_status = 'fail';
33          }
34
35          $response->check_info = sprintf('%d sec(s) (tolerance= %d)', $difference, $tolerance);
```

KUVA 15. Koodiesimerkki parametrijärjestelmän käyttämisestä

Kuvan 15 rivillä 14 haetaan ulkopuolisen järjestelmän URL-osoite (tietotyyppi tekstirivi), josta haetaan oikea kellonaika ja rivillä 29 haetan toleranssi (tietotyyppi kokonaisluku). Ylläoleva koodi varmistaa, että paikallisen palvelimen kello on tietyn toleranssin sisällä oikeasta ajasta.

4.4.3 Objektin tunnisteet ulkopuolisessa järjestelmässä

Tärkeä nykypäivän asettama vaatimus järjestelmälle on se, että se osaa keskustella verkon yli muiden tietojärjestelmien kanssa.

Adepto7 tarjoaa moduulien käyttöön rajapinnan, jolla jokaiselle objektille pystytään tallettamaan ja hakemaan ulkopuolisten järjestelmien tunnisteita.

Esim. jos teemme kalenteri-moduulin, jonka tarvitsee synkronoida kalenteritapahtumat käyttäjän Google Calendar -kalenterista, täytyy meidän Googlen järjestelmän kanssa keskustellessa tietää, millä tunnisteella Googlen järjestelmät viittaavat samaan kalenteritapahtumaan. Esim. meidän järjestelmässä kalenteritapahtuman tunniste on 594, kun sama tapahtuma Googlen järjestelmässä on tunnisteella ”tn008qcigmdjdqrs8hp52pjeog”. Näin ollen meidän kalenteritapahtuma-olion alla on järjestelmän ”com.google.calendar” alle tallennettu tunniste ”tn008qcigmdjdqrs8hp52pjeog”. Yhdellä oliolla voi olla talletettuna monen eri järjestelmän ulkopuoliset tunnisteet.

4.4.4 Tiedostojen palvelu

Adepto7 tarjoaa moduuleille rajapinnan tiedostojen varastointiin ja palveluun. Tiedostojen palvelussa ei ole yhden palvelimen ympäristössä mitään ihmeellistä, mutta hajautettuun järjestelmään siirryttäessä tiedostojen hallinta monimutkaistuu. Otetaan esimerkkinä hajautettu järjestelmä, joka koostuu kolmesta sovelluspalvelimesta. Järjestelmän käyttäjä tallettaa järjestelmään liitetiedoston, ja talletuspyyntö kulkee sovelluspalvelimen numero kaksi kautta. Mikäli sovelluspalvelin tallentaa tiedoston omaan levyjärjestelmäänsä, eivät muut sovelluspalvelimet näe tiedostoa. Tiedostot tarvitsee siis tallentaa levyjärjestelmässä sellaiseen paikkaan, mistä kaikki palvelimet näkevät samat tiedostot.

Toinen esimerkkitapaus on sellainen, missä haluamme tallentaa tiedostot johonkin etäjärjestelmään, esimerkiksi FTP:n yli. FTP on protokolla, jolla pystyy siirtämään tiedostoja järjestelmästä toiseen.

Mikäli moduulit joutuisivat itse huolehtimaan näistä asioista tallettaessaan ja käyttäessään tiedostoja, tulisi siitä moduulien sovelluskehittäjille hyvin vaikeaa, koska he joutuisivat itse huolehtimaan erilaisten asiakkaiden tiedostojen varastointitarpeista. Adepto7:n tehtävä on siirtää monimutkaisuus Adepto7:n ytimeen siten, ettei sovelluskehittäjän itse tarvitse painia näiden ongelmien kanssa.

Sovelluskehittäjille on tarjolla tiedostojen käsittelyyn rajapinta, joka sisäisesti päättelee moduulin file storage policyn kautta, mihin sijaintiin tiedosto tulee tallentaa. Sijainti määrittelee moduulin, joka hoitaa itse tiedon tallennuksen ja hakemisen. Näin ollen asiakas voi asentaa itselleen Amazon S3 storage -moduulin, ja määritellä että kaikki kuvagalleria –moduuliin talletetut tiedostot tallennetaan Amazon S3 järjestelmään. Amazon S3 on Web-pohjainen tiedostojen varastointipalvelu. Moduuli kysyy asennusvaiheessa käyttäjältä käyttäjän Amazon S3 käyttäjätunnukset. Nyt valokuva-albumiin valokuvaa lisättäessä Adepto7 tallettaa valokuvat Amazon S3 järjestelmään ilman, että kuvagalleria-moduulin sovelluskehittäjän täytyy edes tietää, mikä on Amazon S3.

5 KÄYTTÖLIITTYMÄ

Käyttöliittymän teknisen puolen tavoite on mahdollistaa lähes minkälaisen tahansa sisällön näyttäminen, koska järjestelmän päämäärä on mukautua erilaisiin käyttötarkoituksiin. Tämän vuoksi käyttöliittymä on toteutettu ruudukolla, joka on osoittautunut hyvin joustavaksi ratkaisuksi.

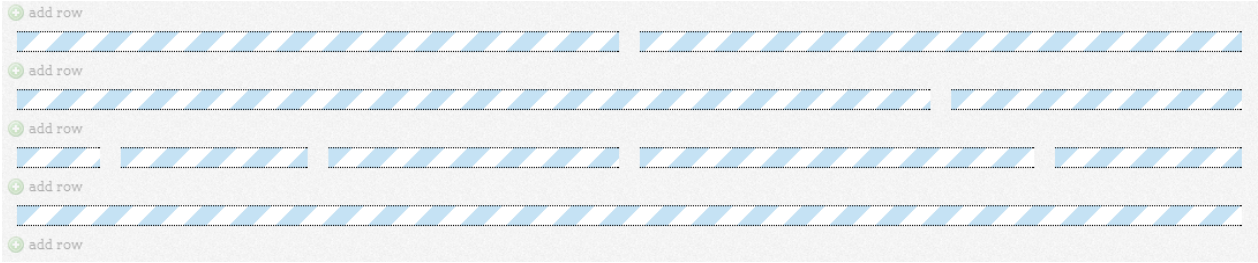
5.1 Käyttöliittymän muokattavuus

Moduulin toiminnot tuovat järjestelmään omia oletussivujansa, joihin on lähinnä esimerkin kannalta upotettu moduulin eri toimintoja nähtäville. Kaikki järjestelmän sivut ovat kuitenkin käyttäjän muokattavissa, koska eri moduulien tuomia näkymiä on pystyttävä integroimaan myös toisten moduulien tuottamiin näkymiin. Esimerkiksi laskutus -moduulin laskun katselu -näkymään on käyttäjän itse pystyttävä liittämään kommentointimoduulin kommentointilomake, mikäli käyttäjä haluaa tehdä laskusta kommentoitavan.

5.2 Ruudukko

Käyttöliittymän rakenne on toteutettu virtuaalisella ruudukolla. Ruudukko koostuu riveistä ja soluista. Yhden rivin leveys on aina vakio, mutta rivi voi koostua eri kokoisista soluista. Käyttäjä voi itse määrittää rivin solujen määrät ja leveydet sekä halutessaan lisätä uusia rivejä mihin väliin tahansa. Kaikki käyttöliittymässä näkyvät elementit sijoittuvat ruudukon soluihin.

5.2.1 Ruudukon rakenne visualisoituna



KUVA 16. Ruudukon rakenne visuaalisena muokkausnäkymässä

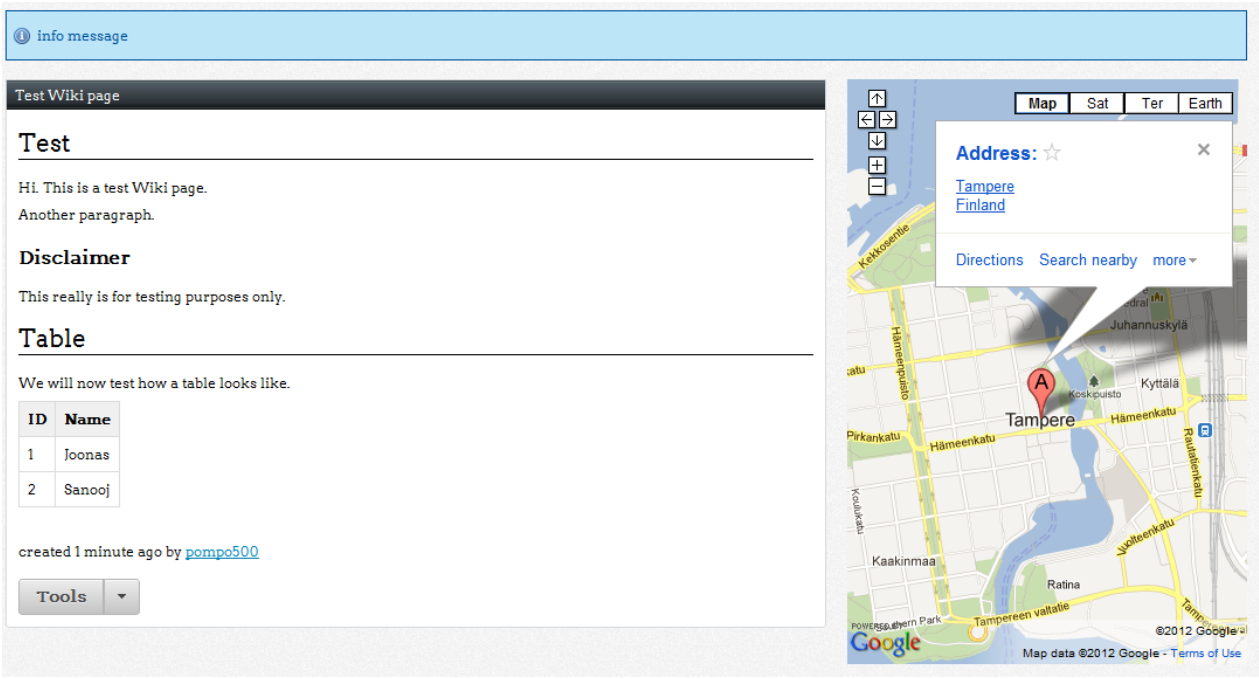
Ylläolevassa kuvassa on neljä riviä. Rivin solut erottaa sinivalkoisesta karamelliväristä. Rivien solujen leveyksien summa on aina 12 yksikköä.

Rivi	Solujen koot	Summa
1	6 + 6	12
2	9 + 3	12
3	1 + 2 + 3 + 4 + 2	12
4	12	12

KUVIO 3. Taulukko kuvan 16 rivien solujen leveyksistä

Kuvan 17 esimerkkitoiminto koostuu ruudukosta, jossa on kaksi riviä. Ylimmällä rivillä on yksi 12 yksikköä leveä solu, johon on pistetty huomautusviesti.

Toisella rivillä on kaksi solua. Ensimmäinen solu on 9 yksikköä leveä ja siihen on pistetty wikisivu. Toinen solu on 3 yksikköä ja siihen on pistetty Tampereen kartta.

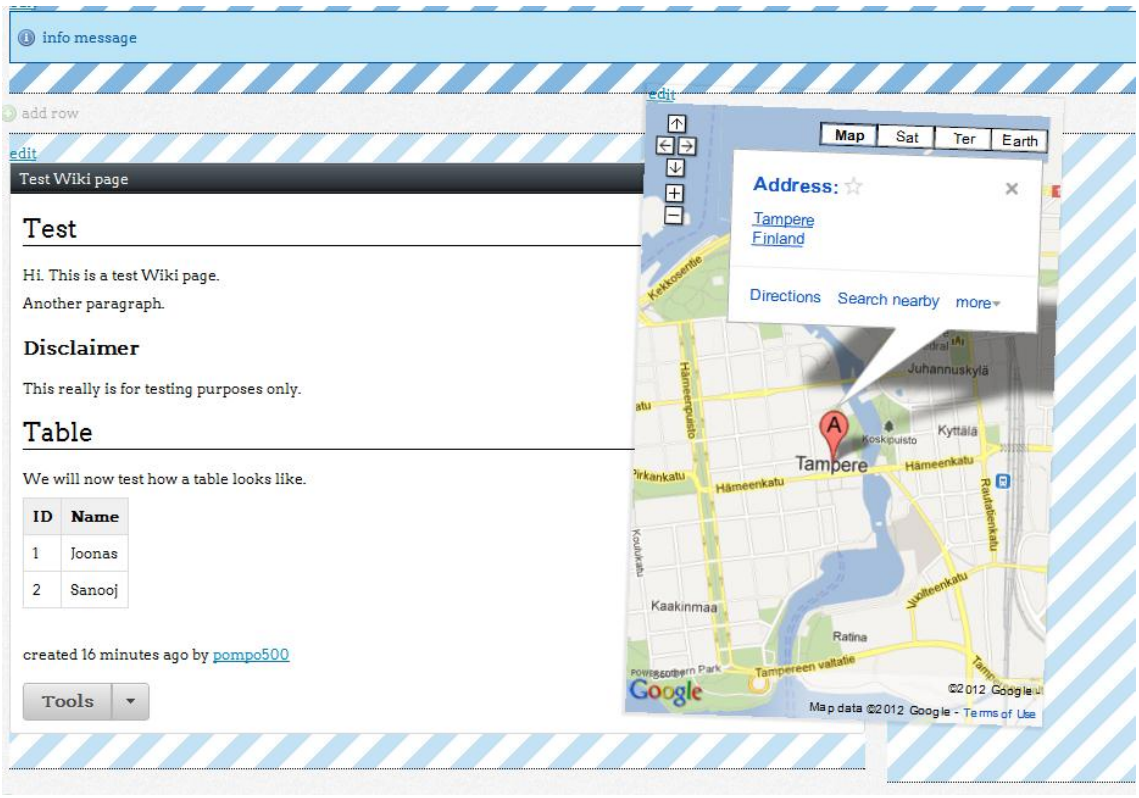


KUVA 17. Esimerkkitoiminto

Kuvassa 18 käyttäjä on siirtämässä elementtiä toisen rivin toisesta solusta ensimmäisen rivin ainoaan soluun.

Siirrettäessä elementtiä solusta toiseen korostuu hiiren alla oleva solu tummemmalla korostusvärillä symboloidakseen sitä, että vapautettaessa hiiren nappi, tipahtaa elementti hiiren kursorin alla olevan soluun.

Samassa solussa voi olla useampi elementti. Viimeisimpänä soluun tipautettu elementti menee järjestyksessä viimeiseksi.



KUVA 18. Elementin siirtäminen

5.3 Käyttöliittymän elementit

Käyttöliittymä koostuu kahdesta pääelementistä: sisältöalueesta sekä yläpalkista.

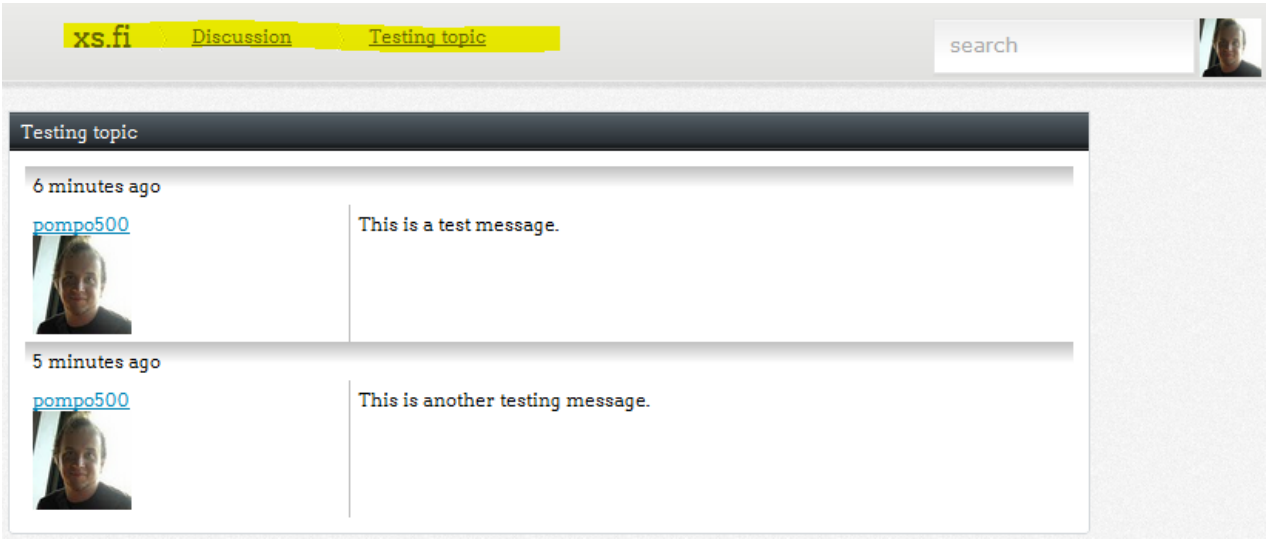
5.3.1 Sisältöalue

Sisältöalue on ”tyhjä taulu”, jonka sisällön jokainen järjestelmän sivu määrittelee itse. Sivut ovat käyttäjän muokattavissa. Sisältöalueen sisältö asetellaan aiemmin kuvailtuun ruudukkoon.

5.3.2 Yläpalkin elementit

Yläpalkissa on elementteinä sijainti, hakupalkki, käyttäjän kuva, ilmoitukset sekä toimintopalkki.

Sijainti-elementti (kuvassa 19 korostettuna) kertoo senhetkisen loogisen sijainnin, mitä näkymää käyttäjä on katsomassa. Tässä esimerkissä olemme sivuston xs.fi keskustelupalstalla katselemassa keskusteluaihetta ”Testing topic”. Sijainti-elementissä eroteltuna olevat hierarkkiset tasot ovat linkkejä, mitä painamalla käyttäjä pääsee helposti navigoimaan sivuhierarkiassa ylöspäin. ”Discussion” –linkkiä klikkaamalla käyttäjä pääsee takaisin selaamaan keskustelupalstan muita keskusteluaiheita.



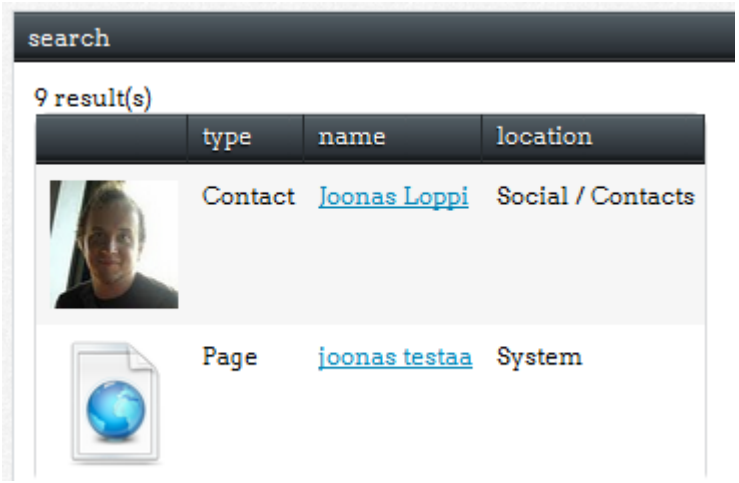
KUVA 19. Sijainti

Haku –elementti hakee kaikista Adepto7:n tietorakenteista hakutuloksia. Hakukenttä laajenee leveämmäksi käyttäjän aktivoidessa sen.

Haku toimii AJAXilla, eli se osaa hakea ehdotuksia hakutuloksista käyttäjän vasta kirjoittaessa hakusanaa. Valitsemalla hakutuloksen käyttäjä siirtyy kunkin tietotyypin (kuvan esimerkissä tietotyyppejä ovat ”Customer” ja ”Contact”) omaan näkymään.

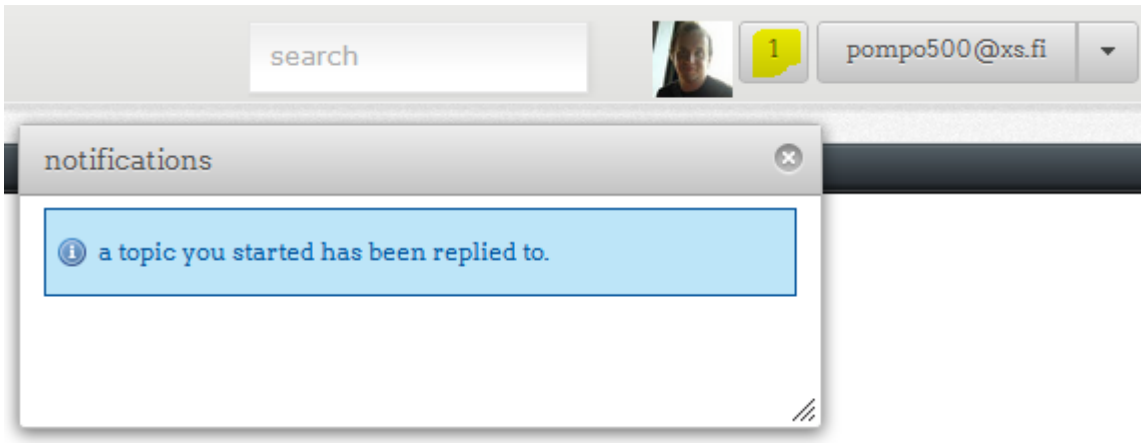
KUVA 20. Haku

Mikäli käyttäjä painaa enter –näppäintä hakukentässä valitsematta mitään ehdotuksista, siirtyy käyttäjä hakutulossivulle, mikä näyttää enemmän tietoja löytyneistä hakutuloksista.



KUVA 21. Hakutuloslistaus –sivu

Ilmoitukset –elementissä on lukemattomien ilmoitusten lukumäärä. Sitä klikkaamalla aukaa dialog-ikkuna, missä näytetään lukemattomat ilmoitukset. Ilmoitusta klikkaamalla ilmoitus kuittaantuu luetuksi, eikä sitä enää näytetä lukemattomien ilmoitusten joukossa.

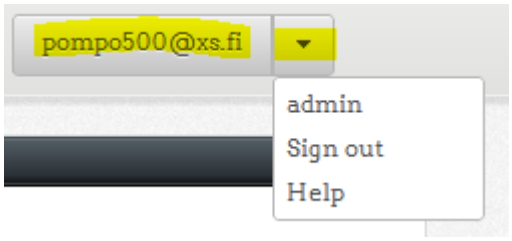


KUVA 22. Ilmoitukset

Työn alla on vielä kuvan 22 kaltaisten ilmoitusten linkittäminen näkymään, missä ilmoitukseen liittyvä kohde sijaitsee.

Toimintopalkissa näytettävää sähköpostiosoitetta klikkaamalla käyttäjä pääsee omalle profiilisivulleen. Mikäli käyttäjä ei ole sisäänkirjautunut, on sähköpostiosoitteen tilalla ”Sign in” –linkki, mistä käyttäjä pääsee sisäänkirjautumisnäkymään.

Toimintopalkin alanuoli–ikonia painamalla saa auki alavalikon mistä pääsee: lukemaan ohjeita, kirjautumaan ulos tai ylläpitosivulle oikeuksien niin salliessa.



KUVA 23. Toimintopalkki

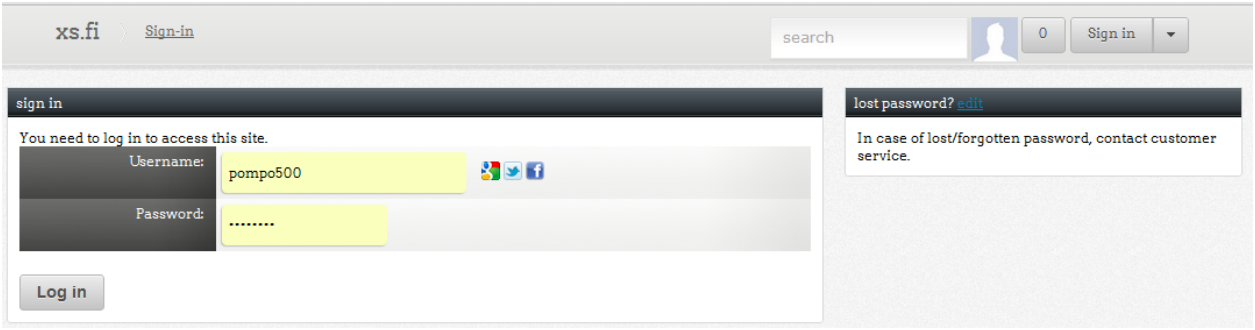
6 KUVIA ESIMERKKIJÄRJESTELMÄSTÄ

Tässä kappaleessa esitellään Adepto7:llä toteutettua esimerkkijärjestelmää kuvien avulla. Esimerkkijärjestelmä on omaan käyttööni tehty projektinhallintajärjestelmä, jossa on asiakasrekisteri, tuntikirjanpito ja integraatio Mercurial-pohjaiseen versionhallintajärjestelmään.

6.1 Sisäänkirjautuminen

Sisäänkirjautumissivulla kysytään käyttäjätunnusta ja salasanaa. Mikäli Adepto7:n on asennettu LDAP-kirjautumismoduuli, pääsee järjestelmään kirjautumaan myös yrityksen AD-domain –tunnuksilla.

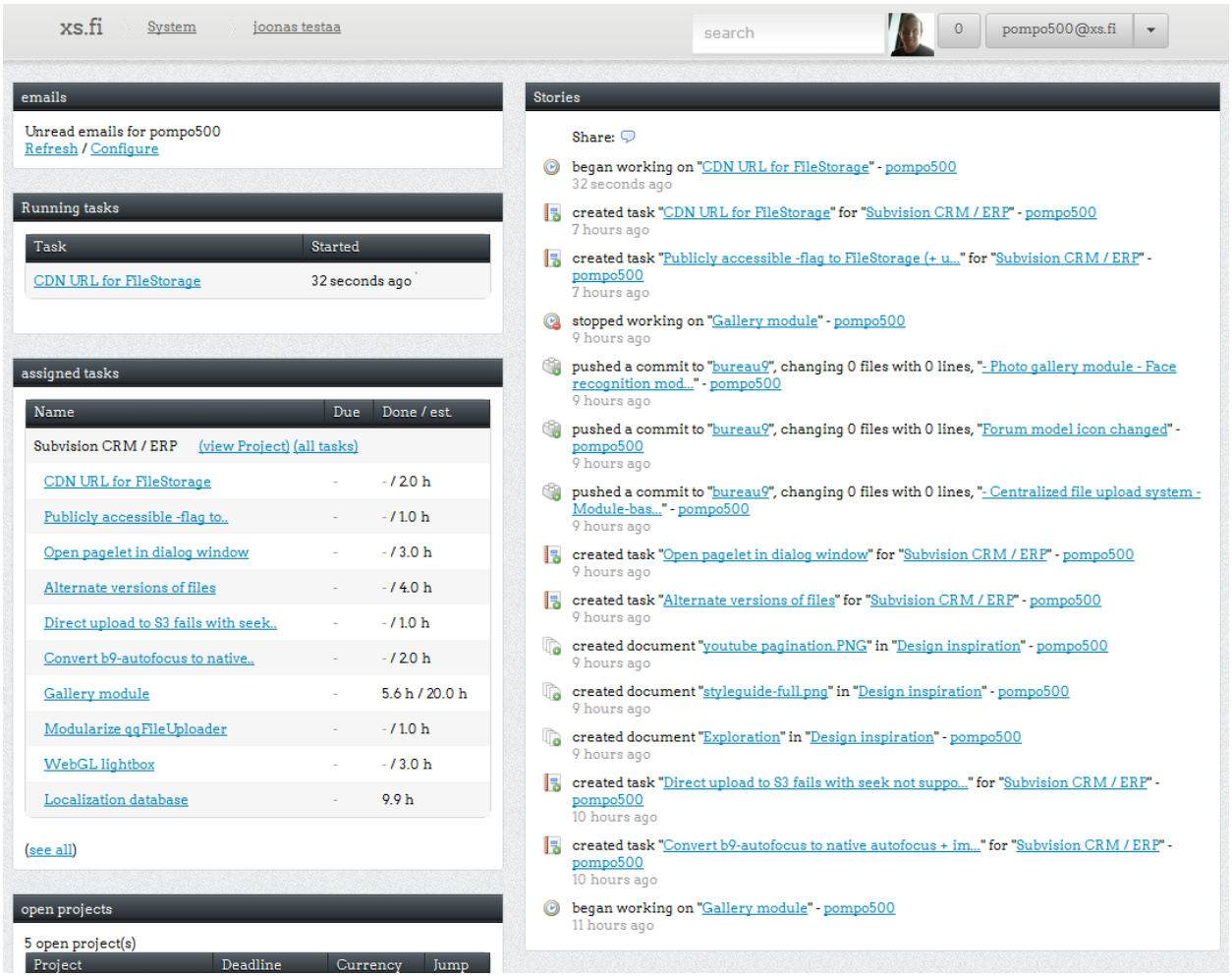
Kuvan 24 järjestelmässä on asennettuna kolme ulkopuolisen järjestelmän tunnistautumismoduulia, joista kukin lisää oman ikoninsa käyttäjätunnus-kentän vieressä oleviin ulkopuolisten palveluiden ikoneihin. Ikonia klikkaamalla käyttäjä pääsee kirjautumaan sisään esim. Google-, Facebook- tai Twitter –tilinsä avulla.



KUVA 24. Sisäänkirjautuminen

6.2 Kojelautanäkymä

Käyttäjä pystyy halutessaan rakentamaan Adepto7-järjestelmään kojelautanäkymän (engl. dashboard). Kojelautanäkymä on pääpiirteinen katsaus järjestelmässä olevaan tietosisältöön. Esimerkiksi auton kojelauta kertoo auton kuljettajalle tärkeimmät tiedot auton tilasta. Samoin tietojärjestelmän kojelautanäkymä koostaa käyttäjälle oleellisimmat tiedot tietojärjestelmästä.



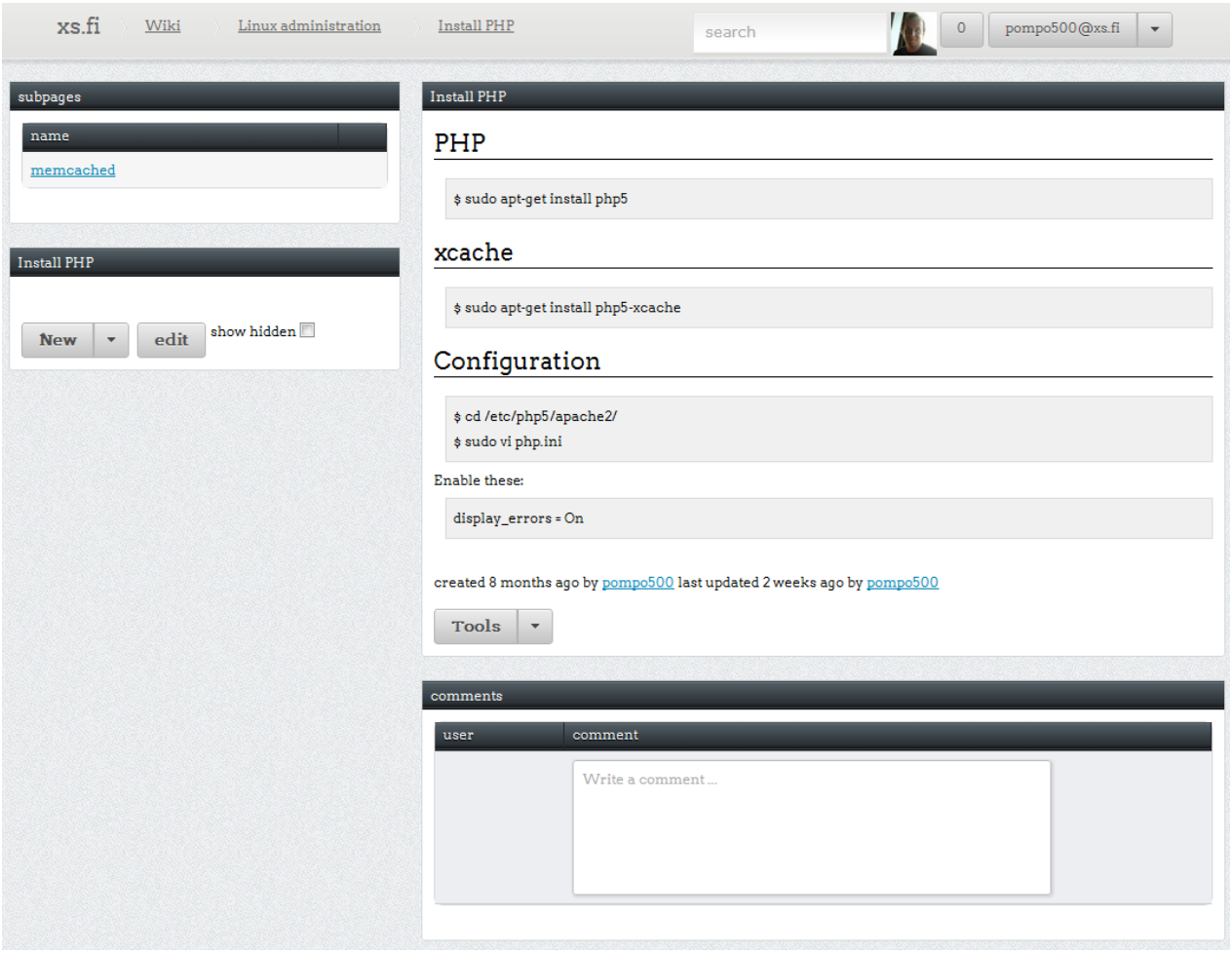
KUVA 25. Kojelautanäkymä

Kuvan 25 kojelautanäkymä on suunniteltu projektinhallintaan. Vasemmalla näkyy omissa laatikoissaan lukemattomat sähköpostit, tuntikirjauksen alla olevat tehtävät, käyttäjälle osoitetut tehtävät sekä avoimet projektit. Oikealla näkyy uutisvirta viimeisimmistä käyttäjien tapahtumista.

6.3 Wikisivu

Wikisivu on sivu, jota lähtökohtaisesti kaikki kirjautuneet käyttäjät pystyvät muokkaamaan. Käytin sanaa lähtökohtaisesti, koska sivun muokkaamisoikeudet ovat kuitenkin rajattavissa tarpeen mukaan, mutta wiki-ohjelmistojen filosofia on kuitenkin sallia avoin tietojen muokkaaminen. Käytännössä yrityskäytössä wikit yleensä sallivat yrityksen omille työntekijöille vapaan muokkaamisen.

Väärinkäytösten vaikutuksen minimoimiseksi wikisivun pystyy palauttamaan mihin tahansa edelliseen versioon, sillä wiki-järjestelmä pitää täydellistä muutoshistoriaa sivun muokkauksista, käyttäjistä ja päivämääristä.

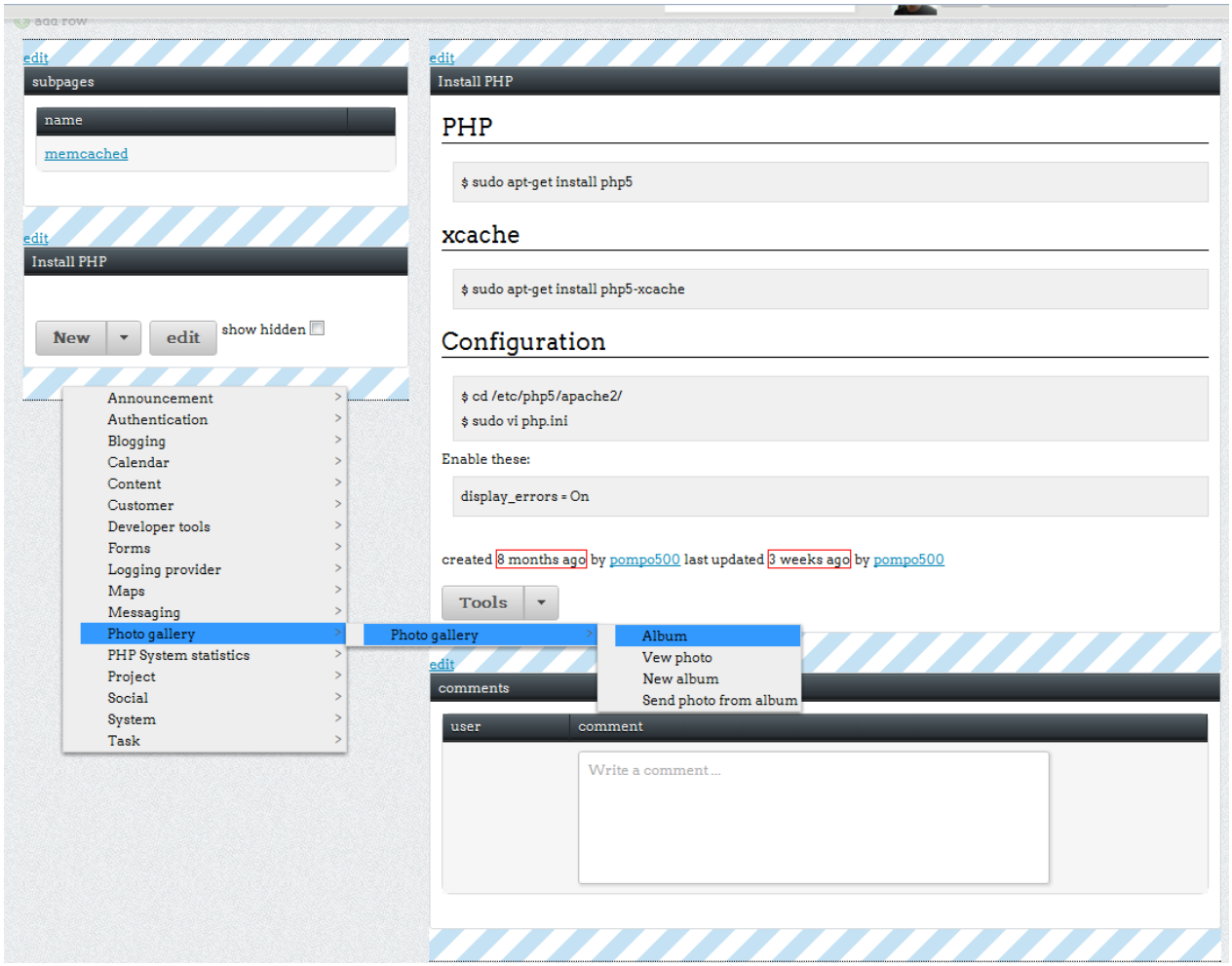


KUVA 26. Wikisivu

Mikäli Adepto7-järjestelmään on asennettu kommentointimoduuli, pystytään wikisivulle sisällyttämään kommentointilomake, joka sallii wikisivujen kommentoinnin. Näin on tehty kuvan 26 järjestelmässä.

6.4 Sivun muokkaaminen

Adepto7:n kaikki sivut ovat käyttäjän muokattavissa. Kuvassa 27 muokataan kuvan 26 wikisivu-näkymän sivua. Käyttäjä on painanut hiiren oikealla napilla vasemmasta karamellivärisestä solusta, jolloin siihen on auennut popup-valikko, mistä soluun saa lisättyä uuden pageletin. Pageletit ovat tällä hetkellä järjestetty moduulin kategoria -> moduulin nimi -> pageletin nimi –hierarkiaan. Käyttäjä on lisäämässä valokuva-albumia wikisivun katselunäkymään. Muutokset heijastuisivat kaikille sivuille, joilla katsellaan wikisivua.



KUVA 27. Sivun muokkaaminen

7 PROJEKTIN LAAJUUS

Tämän kappaleen tarkoituksena on selventää projektiin käytettyä työmäärää, verrata lähdekoodin määrää laajudeltaan PHP-pohjaisiin kilpailijoihin ja kertoa toteutetuista moduuleista.

7.1 Työmäärä

Tunteja projektille olen kirjannut 673 h. Tunteja on mahdollisesti kertynyt enemmänkin, sillä en ole ollut aina hirveän tarkka tuntien kirjaamisesta, sekä tuntien kirjauksen olen suorittanut Adepto7:n ”tehtävä”-moduulilla, joka luonnollisesti oli käyttökunnossa vasta jonkun aikaa Adepto7-projektin aloittamisen jälkeen.

Projektin alkutaival sijoittuu joulukuulle 2010, ensimmäinen muutosmerkintä versionhallintaan on tehty 9.12.2010. Projektin kesto on 17 kuukautta.

7.2 Lähdekoodin määrä, Adepto7 + moduulit

Kieli	**Rivimäärä**	**Tiedostomäärä**
PHP	22 282	265
JavaScript	6 362	38
CSS	2 732	36
HTML	4 257	166

KUVIO 4. Lähdekoodin määrä, Adepto7 + moduulit. Lähdekoodimäärät sisältävät ohjelmakoodin lisäksi tyhjät rivit sekä kommenttirivit. Tilastot laskettu 27.5.2012.

7.3 Lähdekoodin määrä, xslib

Kieli	**Rivimäärä**	**Tiedostomäärä**
PHP	23 817	198

KUVIO 5. Lähdekoodin määrä, xslib. Lähdekoodimäärät sisältävät ohjelmakoodin lisäksi tyhjät rivit sekä kommenttirivit. Tilastot laskettu 27.5.2012.

7.4 PHP-lähdekoodin määrä verrattuna kilpailijoihin

Projekti	Lähdekoodin määrä, PHP
Adepto7	22 000
Adepto7 + xslib	46 000
Drupal	900 000 (Ohloh - Drupal)
WordPress	170 000 (Ohloh - WordPress)

KUVIO 6. Lähdekoodin määrä verrattuna kilpailijoihin

Lähdekoodimäärät sisältävät ohjelmakoodin lisäksi tyhjät rivit sekä kommenttirivit. Tilastot haettu 27.5.2012.

xslib on sisällytetty laskuihin, koska se on kiinteä osa Adepto7:ää, ja se on itse kirjoittamani luokkakirjasto pääasiassa Adepto7:ää varten.

WordPress on blogiohjelmisto, johon myös pystyy lisäämään toimintoja asentamalla siihen lisäosia – se on myös tietyllä tapaa modulaarinen. Adepto7:n ja WordPressin modulaarisuuden toteutustavoissa on kuitenkin huomattavia eroja, ja WordPress on pääasiallisesti blogiohjelmisto.

WordPressillä on PHP-lähdekoodia neljä kertaa enemmän kuin Adepto7:ssä, mutta noin 16 aktiivista kehittäjää. WordPressin aktiivisiksi kehittäjiksi laskin Ohloh-analyysisivun (Ohloh – WordPress - Contributors) käyttäjät, joilla on versionhallinnassa yli 100 muutosmerkintää.

Drupal on sisällönhallintaohjelmisto, jonka toteutustapa on myös modulaarinen. Drupalin modulaarisuus on myös toteutettu huomattavasti eroavalla tavalla kuin Adepto7:n. Muun muassa jokainen Drupal-järjestelmään asennettu moduuli suoritetaan jokaisella sivulatauksella riippumatta siitä, tarvitaanko moduulia sivun tuottamisessa. Adepto7:ssä on käytössä lazy loading -mekanismi, joka lataa vain ja ainoastaan sivun tuottamiseen vaaditut resurssit.

7.5 Toteutetut moduulit

Moduuleita on toteutettu 87 - reilusti enemmän, kuin opinnäytetyön tavoitteena oli. Ylimääräiset moduulit olen ohjelmoinut omasta mielenkiinnostani. Kuviossa 7 on listattuna muutamia mielenkiintoisia moduuleita.

Moduulin nimi	**Kuvaus**
Google authentication	Järjestelmään kirjautuminen Google-tunnuksilla.
Wiki	Wiki-sivujen ylläpito.
Customer	Asiakasrekisteri. Asiakkaiden ja heidän yhteyshenkilöiden hallinta.
Google Docs	Google Docs –dokumenttien liittäminen. Google Docs sisältää muun muassa tekstinkäsittely- ja taulukkolaskenta -dokumentteja.
Amazon S3 file storage	Tiedostojen tallentaminen ja palveleminen Amazon S3 -tiedostonvarastointijärjestelmästä.
Invoicing	Laskujen luonti, seuranta ja lähetys PDF-muodossa.
Photo gallery	Kuva-albumien luonti. Tukee kuvien lisäyksen yhteydessä henkilön tunnistamista kasvojen avulla, mikäli järjestelmästä löytyy ominaisuuden toteuttava moduuli.
face.com face recognition	Henkilön kasvojen tunnistus face.com rajapinnan avulla. Tekniikka osaa eritellä valokuvista kohdat, missä esiintyy ihmisen kasvot. Jokaisen ihmisen kasvo täytyy ensimmäisen kerran kouluttaa face.com:ille. Tämän jälkeen face.com tunnistaa tulevaisuudessa lisättävät kuvat, joissa esiintyy tunnistetun henkilön kasvot.
Business process	Erilaisten prosessien luominen erilaisiin tilanteisiin. Asiakas pystyy itse määrittelemään esim. luotaville laskuille prosessin, missä lasku kulkee seuraavien vaiheiden läpi: luotu, vahvistettu, lähetetty ja maksu vastaanotettu. Prosessin eri vaiheille pystyy määrittelemään myös useampia siirtymiä. Esimerkiksi lasku voi lähetetty –vaiheen jälkeen siirtyä joko maksu vastaanotettu- tai maksuhäiriö -tilaan.

KUVIO 7. Toteutettuja moduuleita

8 YHTEENVETO

Projektin tavoite oli toteuttaa PHP-ohjelmointikielellä modulaarinen Web-sovelluskehys sekä muutama moduuli, jolla sovelluskehyksen modulaarisuutta pystytään testaamaan.

Sovelluskehys, Adepto7, valmistui tavoitteita vastaavaan tilaan - siitä syntyi joustava, teknisesti edistynyt ja nopea. 17 kuukauden kehitystyön tuloksena syntyi sovelluskehyksen lisäksi 87 moduulia erilaisiin tarpeisiin, vaihdellen laskutuksesta valokuvien kasvojentunnistukseen. Projekti ylitti asetetut tavoitteet ja olen itse käyttänyt Adepto7:ää projektinhallinta- ja tuntikirjaus -työkaluna.

Työn toteutus oli mielenkiintoinen, ja kasvatti selvästi omia ohjelmointitaitoja.

Adepto7:llä on kaupallista potentiaalia, mikäli sitä kehitetään eteenpäin sekä ohjelmoidaan lisää moduuleita, jotka toteuttavat myyntikelpoisia ominaisuuksia. Matkaa markkinointikelpoiseksi tuotteeksi kuitenkin on paljon, muun muassa helppokäyttöisyys kaipaa hiomista ja yrityskäyttö vaatii tukea kunnolliselle varmuuskopioinnille, tietojen tuonti/vienti –toiminnoille sekä ulkoisiin järjestelmiin integroitumiselle.

Tulevaisuuden haaveissa on miettiä Adepto7:n kaupallistamista. Paljon pohtimista on lisensointimallissa – siinä, miten käyttäjämäärät saadaan muutettua kassavirraksi. Haaveena on kuitenkin saada Adepto7 mahdollisimman suureen levitykseen, joten ideaalinen ratkaisu olisi, jos jollain tavalla keksisi järkevän lisensointimallin, jolla saisi ohjelmistosta ilmaisversion harrastekäyttäjille ja pienille organisaatioille, mutta suuret yrityskäyttäjät saisi maksullisen lisenssin taakse. Tässä on paljon miettimistä.

Toinen tulevaisuuden haave on toteuttaa moduulikauppa, johon kolmannen osapuolen sovelluskehittäjät saavat ladattua moduuleitansa levitettäviksi. Adepto7-järjestelmien ylläpitäjille moduulin asennus tulisi olla niin helppoa kuin moduulikaupasta ”Asenna moduuli” –kaltaisen nappulan klikkaaminen, vahvistuskysymys ja moduulin mahdollisten asetusten kysyminen.

LÄHTEET

Ohloh - WordPress [online] [viitattu 27.5.2012]
http://www.ohloh.net/p/WordPress/analyses/latest

Ohloh – WordPress – Contributors [online] [viitattu 27.5.2012]
http://www.ohloh.net/p/WordPress/contributors

Ohloh - Drupal [online] [viitattu 27.5.2012]
http://www.ohloh.net/p/drupal/analyses/latest

Wikipedia – NoSQL [online] [viitattu 31.5.2012]
http://en.wikipedia.org/w/index.php?title=NoSQL&oldid=494967756

memcached [online] [viitattu 2.6.2012]
http://memcached.org/

IETF – TOTP [online] [viitattu 3.6.2012]
https://tools.ietf.org/html/rfc6238

LIITTEET

Liite 1. Moduulin UML-luokkakaavio