Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Internet-tekniikka

2012

Hasenen Ahmad

DEMOS-PLAN-SOVELLUKSEN OHJELMISTON TESTAUS JA KOTOISTUS

– PARTERRE-projekti

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Internet-tekniikka

2012 | 24

Ohjaajat:

Ins. Olli Ojala

TkL Juha Nikkanen

Hasenen Ahmad

DEMOS-PLAN-SOVELLUKSEN OHJELMISTON TESTAUS JA KOTOISTUS – PARTERRE-projekti

Parterre-projekti on Eu:n rahoittama hanke, joka kehittää verkkopalveluja. Tässä opinnäytetyössä testattiin ja kotoistettiin saksalaisyrityksen kehittämää DEMOS-Plan for PARTERRE-sovellusta. DEMOS-Plan-sovelluksen tavoitteena on edistää aluesuunnittelun hallinnointia kaikille toimivaltaisille viranomaisille keräämällä virallisia ja epävirallisia kommentteja ja huomautuksia suunnitelman luonnoksesta verkossa.

DEMOS-Plan-sovelluksen PHP-ohjelmakoodi oli salattua, joten testausta jouduttiin toteuttamaan tutkivalla testausmenetelmällä. Sovelluksen tietoturva-aukkoja testattiin myös tietokantahyökkäyksillä. Kaikki havaitut virheet välitettiin suoraan sivuston kehittäjälle.

DEMOS-Plan-sovellus asennettiin LAMP-ympäristöön virtuaalipalvelimelle Turun ammattikorkeakoulun tiloihin ja käännettiin englannin ja saksan kielestä suomen kielelle, jotta sovellus palvelisi Paraisten kaupungin tarpeita aluesuunnittelussa.

Sovelluksesta ei koskaan saatu uutta päivitettyä versiota, jossa ohjelmavirheet olisi korjattu. Kotouttamisen yhteydessä ilmaantui ongelmia, jotka liittyvät ohjelmakoodin salaukseen. Tehtiin johtopäätös, ettei DEMOS-Plan-sovellus ole ainakaan vielä sovelias käytettäväksi pilottikohteena olleen Paraisten kaupungin kaupunkisuunnittelussa. Sovelluksen toimivia osia käytettiin hyväksi pienimuotoisessa pilottikokouksessa asiakirjojen esittämiseen.

Paraisten kaupungin käyttöön löytyi muita www-sovellusvaihtoehtoja.

ASIASANAT:

ohjelmistotestaus, kotoistus, lokalisointi, tutkiva testaus, virtuaalipalvelin

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Internet Technology

2012 | 24

Instructors:

Olli Ojala, B. Eng.

Juha Nikkanen, Lic. Tech., Principal Lecturer

Hasenen Ahmad

DEMOS-PLAN APPLICATION SOFTWARE TESTING AND LOCALIZATION – PARTERRE PROJECT

The Parterre project is an EU-funded project which is developing online services. In this thesis, the DEMOS-Plan for PARTERRE application of a German developer was tested and localized. The aim of DEMOS-Plan application is to boost a shared management of the spatial planning process among all decision-making authorities by collection of formal and informal comments and observations to the plan draft on the web.

The PHP programming code of DEMOS-Plan application was encrypted so testing was performed using the Exploratory testing method. In addition, the security vulnerabilities of the application were tested by attacking the database. All errors found were forwarded directly to the site developer.

The DEMOS-Plan application was installed on a LAMP environment virtual machine on the premises of Turku University of Applied Sciences. The application was localized from English and German to the Finnish language so that this application would serve the spatial planning needs of the city of Pargas, Finland.

We never received a new updated version of the application where the software bugs were fixed. During the localization, there were problems detected relating to the source code encryption. It was concluded that the DEMOS-Plan application was not yet suitable to be used in the modification operation of the town planning. The working parts of the application were used for a presentation of the documents at a small-scale pilot meeting.

There were other web applications that serve the needs of the city of Pargas.

KEYWORDS:

Software testing, localization, exploratory testing, virtual machine

SISÄLTÖ

KUVAT

TAULUKOT

KÄYTETYT LYHENTEET JA SANASTOT

Apache	Apache on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma (Lähde: http://fi.wikipedia.org/wiki/Apache_(palvelinohjelma)
CGI	Standardoitu verkkopalvelin tekniikka, jonka avulla käyttäjä välittää ja saa takaisin dataa palvelimella suoritettavalle ohjelmalle. Lyhenne tulee sanoista Common Gateway Interface. (Lähde: http://searchsoa.techtarget.com/definition/common-gateway-interface)
LAMP	Kokoelma avoimen lähdekoodin ohjelmia, jotka yhdessä muodostavat WWW-palvelimen, jonka avulla voidaan suorittaa dynaamisia verkkosivuja. LAMP lyhenne tulee sanoista: Linux, Apache, MySQL, PHP (Lähdeviite: http://fi.wikipedia.org/wiki/LAMP)
Linux	Unix-pohjainen avoimeen lähdekoodiin perustuva käyttöjärjestelmä (Lähde: http://fi.wikipedia.org/wiki/Linux)
MySQL	WWW-sovelluksissa käytettävä ilmainen tietokanta (Lähde: http://users.jyu.fi/~kolli/ITK215_05/php/?sivu=tietokanta)
PHP	Ohjelmointikieli, jota käytetään erityisesti Web-palvelinympäristössä . Lyhenne sanoista Hypertext Preprocessor (Lähde: http://fi.wikipedia.org/wiki/PHP)
SSL	Standardoitu salausprotokolla, jolla suojataan verkkosovellusten tietoliikennettä verkkojen yli. (Lähde: http://info.ssl.com/article.aspx?id=10241)

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on testata ja kotoistaa saksalaisyrityksen kehittämää DEMOS-Plan for PARTERRE-sovellusta. PARTERRE-projekti on EU:n rahoittama hanke [1].

DEMOS-Plan sovelluksen tavoitteena on edistää aluesuunnittelun hallinnoinnin kaikille toimivaltaisille viranomaisille keräämällä virallisia ja epävirallisia kommentteja ja huomautuksia suunnitelman luonnoksesta verkossa. Tämä tapa säästää arvokasta aikaa ja kustannuksia. Maankäyttöasiakirjat ja -kartat, sekä asianmukaiset tiedot käynnissä olevasta projektista, kuten esimerkiksi yhteyshenkilöt, määräajat, suunnitelmat ja tapaamiset laitetaan verkkoon yleisesti näkyville. Näin päättäjät, kansalaiset ja sidosryhmät voivat keskustella koko projektista. [2]

Sivustoon on tarkoitus laittaa Paraisten kaupungissa tehdyn asemakaavan muutosehdotuksen materiaalit. DEMOS-Plan-sovellus on tarkoitus ottaa käyttöön Paraisten kaupungin kaupunkisuunnittelun välineenä.

DEMOS-Plan-sovelluksen PHP-ohjelmakoodi on salattua, joten se jää työn ulkopuolelle. Sen takia testausta joudutaan toteuttamaan tutkivalla testaus menetelmällä, joka on mustan laatikon (black box) testaustekniikkaa, sillä tarkoitetaan menetelmää, jossa testaajan ei tarvitse hallita kyseistä ohjelmointikieltä eikä tarvitse tietoa sovelluksen toiminnallisesta tai sisäisestä rakenteesta.

DEMOS-Plan-sovelluksen tietoturva-aukkoja testataan tietokanta hyökkäyksillä. Kaikki havaitut virheet välitetään suoraan sivuston kehittäjälle.

DEMOS-Plan-sovellus asennettaan LAMP-ympäristöön virtuaalipalvelimelle Turun ammattikorkeakoulun tiloihin ja käännettään saksan kielestä suomen kielelle.

PARTERRE-projektiin liittyviä opinnäytetöitä on tehty aikaisemmin. Web 2.0:n käytöstä kaupunkisuunnittelussa [3] tehtiin tutkimustyötä jo olemassa olevista

tekniikoista Web 2.0:n liittämisestä karttapohjaisiin sovelluksiin ja suunniteltiin sekä luotiin testitapauksia Demos-Planille. UML-työkalujen hyödyntäminen eTM-ohjelman arvioinnissa ja testauksessa [4] tarkasteltiin UML-kaavioiden roolia eTM-ohjelmiston arvioinnissa ja käsiteltiin erilaisia UML-mallinnuksen työvälineitä sekä niiden hyödyntämistä tietokantapohjaisten sovellusten testauksessa.

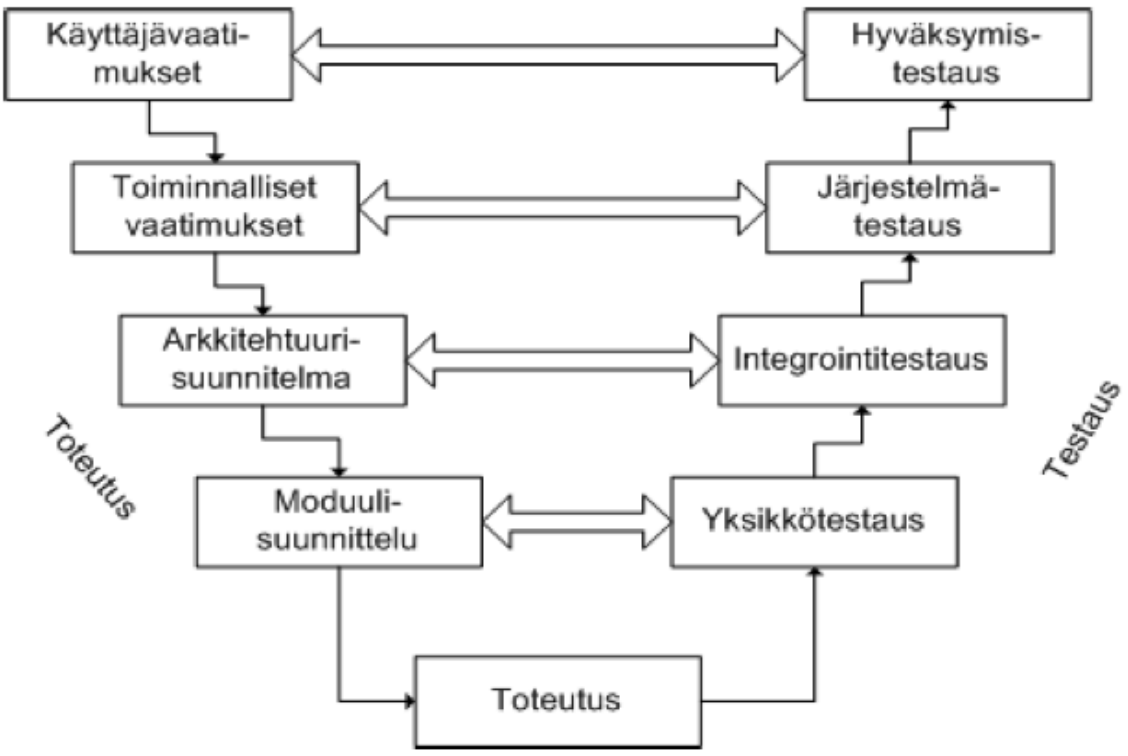
Työ tehtiin Paraisten kaupungille Turun ammattikorkeakoulu toimeksiannosta. Kun sovellus on kotoistettu, testattu ja todettu toimivaksi, sitä on tarkoitus käyttää Paraisten kaupungin kaupunkisuunnittelun tueksi.

Opinnäytetyön seuraavissa luvuissa käydään läpi ohjelmistotestauksen ja lyhyesti kotoistukseen liittyvää teoriaa.

2 OHJELMISTON TESTAAMINEN

Ohjelmistotestauksessa on monta eri testausmenetelmää, joiden tavoitteena on löytää mahdollisimman paljon virheitä ohjelmasta tai palvelusta. Ohjelmiston testaus sisältää ohjelman suorittamista erilaisissa ympäristöissä ja tilanteissa. Havaitut virheet raportoidaan ohjelman kehittäjälle. Tämä tarkoittaa sitä, ettei ohjelman kehittäjän tarvitse testata ohjelmaa, vaan se voi olla muu organisaatio.

Ohjelmiston testaaminen on prosessi, joka kertoo asiakkaalle testattavan ohjelman tai palvelun laadukkuudesta. Samalla asiakkaalle selviää, mitä käyttöönoton riskejä ohjelmistossa voisi olla. Ohjelmistotestauksella voidaan osoittaa, että ohjelma, palvelu tai tuote täyttää tietyt vaatimukset, ohjelmisto toimii oletetusti ja se voidaan ottaa käyttöön halutulla tavalla. Kuvassa 1 näkyy yksi perinteisistä testausmalleista. [5] (Kuva 1.)



Kuva 1. Testauksen V-malli. [6]

Ohjelmistotestaus jakautuu yleisesti kahteen eri luokkaan, toiminnalliseen mustalaatikkotestaukseen ja rakenteelliseen lasilaatikkotestaukseen [7].

2.1 Lasilaatikkotestaus

Lasilaatikkotestaus on menetelmä, jossa testataan sovelluksen sisäistä toimintaa ja koodin rakennetta. Tässä tapauksessa testaajan on hallittava testattavan sovelluksen ohjelmointikielen hyvin, jotta voidaan tunnistaa kaikki sovelluksen läpi kulkevat polut. [7]

2.1.1 Yksikkötestaus

Yksikkötestaus on ensimmäinen todellinen testivaihe, jossa testataan kehittäjän yksittäisen moduulin kaikki uudet ja muuttuneet polut. Tähän kuuluu syötteiden ja ulostulojen, haarojen, silmukoiden, aliohjelma syötteiden, toiminnan toteutuksien ja diagnostiikan läpikäyntiä. Näin varmistetaan, että koodi toimii halutulla tavalla. Tässä vaiheessa löytyvät virheet voivat olla pieniä, mutta jos ne jäävät huomaamatta, ne voivat olla hyvin vahingollisia myöhemmissä testausvaiheissa. [8]

2.1.2 Lause-, haara- ja polkutestaus

Lause-, haara- ja polkutestauksessa tutkitaan sovelluksen lähdekoodia.

Lausetestauksessa (Statement testing) jokainen lause lähdekoodissa suoritetaan ja testataan ainakin yhden kerran.

Haaratestauksessa (Branch testing) kaikki lähdekoodissa olevat haarat testataan vähintään yhden kerran.

Polkutestauksessa (Path testing) kukin polku lähdekoodissa testataan vähintään yhden kerran. [9]

2.2 Mustalaatikkotestaus

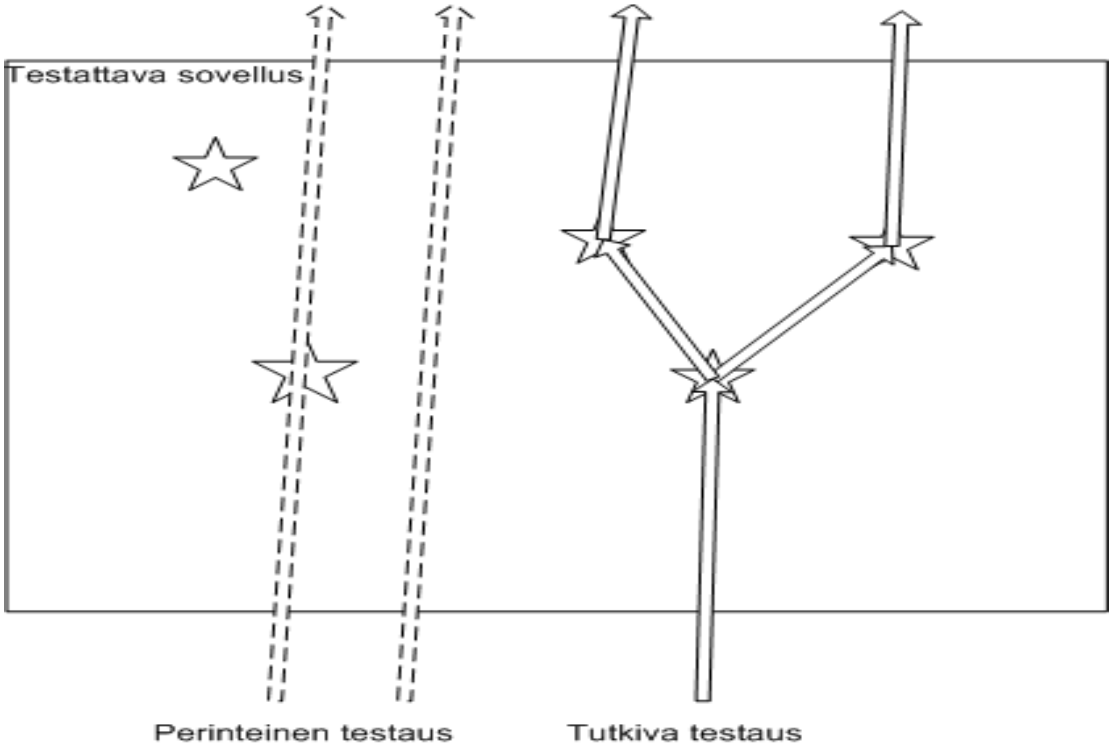
Mustalaatikkotestauksessa sovellusta pidetään mustana laatikkona nimensä mukaan. Testaajalla ei tarvitse olla erityistä tietämystä sovelluksen sisäisestä toiminnasta, rakenteesta tai ohjelmointikielestä. Testaajan tarvitsee tietää ainoastaan, mitä sovelluksen on tarkoitus tehdä, mutta ei tarvitse tietää, miten se tapahtuu. Testaajan tulee kuitenkin tietää kullekin syötteelle oikean ulostulon. [7] (Kuva 2.)

Kuva 2. Mustalaatikkotestaus. [10]

2.2.1 Tutkiva testaus

Tutkivassa testauksessa, joka perustuu mustanlaatikon testaustekniikkaan, ei suunnitella testitapauksia etukäteen, vaan testejä luodaan aikaisempien testitulosten perusteella. Tämä säästää arvokasta aikaa ja rahaa ohjelmiston tilaajalle. Tutkivassa testauksessa pyritään aina miettimään, mikä on kullakin hetkellä paras seuraavaksi suoritettava testi. Se perustuu testaajan tekemiin havaintoihin testattavasta järjestelmässä ja testausten ohjaamista niiden perusteella. Testien tehokkuus ja niiden onnistumiseen vaikuttavat testaajan taidot ja kyvyt. Tutkivassa testauksessa testien muuttuminen nimenomaan mahdollistaa paremmin virheiden löytymistä. Ei siis ole oleellista, että testit voidaan toistaa samanlaisina. Tutkivassa testauksessa on hyvä kirjoittaa, miten tiettyjä virheitä on löydetty, jotta samoja virheitä voidaan helposti tarvittaessa löytää uudestaan. Taas ennalta määrättyjen testien ongelmana on se, että ellei testi löydä ensimmäisellä ajolla virheitä, se ei todennäköisesti löydä myöhemminkään virheitä,

kuten kuvassa 3 näkyy. Testitapauksiin perustuvassa testauksessa kirjataan nekin testit, jotka eivät löydä virheitä. Sen sijaan tutkivassa testauksessa kirjataan yleensä ainoastaan virheisiin johtaneet testit [6]. (Kuva 3.)



Kuva 3. Perinteisen ja tutkivan testauksen ero. [6]

2.2.2 Järjestelmätestaus

Järjestelmätestausta suoritetaan täysin integroidussa järjestelmässä. Tämä on ensimmäinen kerta, kun järjestelmän osat testataan yhtenä yksikkönä. Tämän tarkoituksena on arvioida järjestelmän toimivuutta asetetuilla vaatimuksilla. Järjestelmätestauksen tarkoituksena on havaita mahdolliset ristiriidat itse ohjelmistoyksiköiden tai laitteiston välillä. [8]

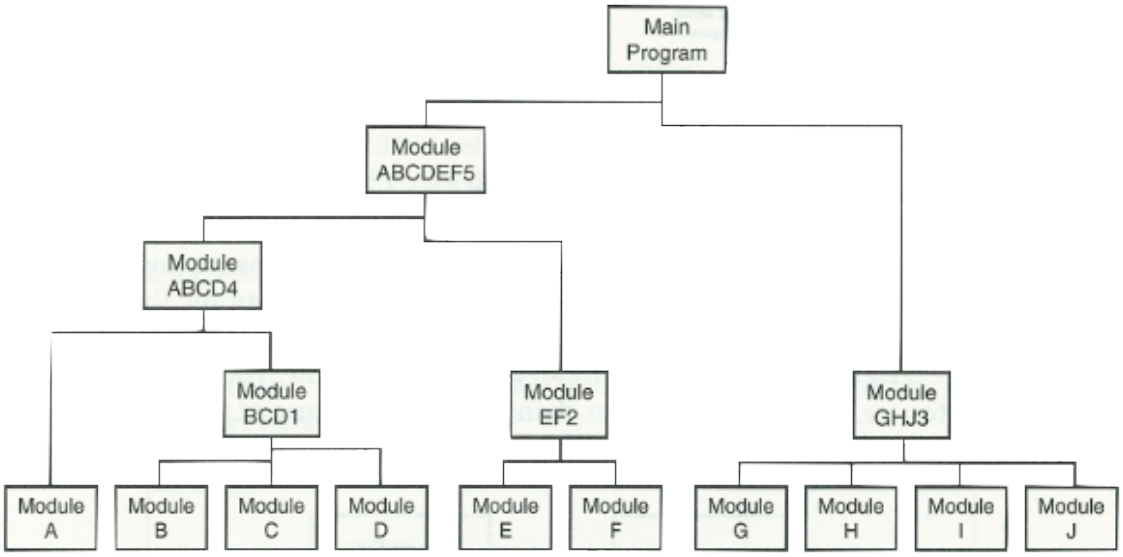
2.2.3 Hyväksymistestaus

Hyväksymistestaus on ohjelmiston tilaajalle ja kehittäjälle tarkoitettu viimeistelytestaus, jolla varmistetaan, että ohjelmisto voidaan ottaa käyttöön suunniteltuun

tarkoitukseen sille asetetuilla vaatimuksilla. Tällä tavalla karsitaan mahdolliset käytännön yllätykset ennen käyttöönottoa. Hyväksymistestaus on myös palaute ja yhteenveto ohjelmiston tilaajalle ja kehittäjälle ohjelmiston laadusta. Testausta voidaan suorittaa tilaajan työympäristössä ja tilaajan laitteistolla. Hyväksymistestaus on tärkeä osa ohjelmistotestauksen ketjua. [8]

2.3 Integraationtestaus

Integraatiotestaus voidaan aloittaa heti, kun saadaan moduulit valmiiksi. Integrointitestaus suoritetaan integroimalla joukko moduuleja yhteen testattavaksi. Tarkoituksena on löytää moduulien välistä sopimattomuutta tai niistä aiheutuvia virheitä. Tämä nouseva testaustoiminto jatkuu integroimalla lisää moduulijoukkoja yhteen siihen saakka kunnes koko ohjelmisto tai suuri osa siitä integroidaan yhteen ja testataan kerralla, kuten kuvassa 4 näkyy. Kun kaikki ohjelmiston osat on integroitu yhteen voidaan aloittaa seuraava testausvaihe, joka on järjestelmän testaus. Integraatiotestausta voi olla sekä toiminnallisissa että rakenteellisissa tekniikoissa. [11] (Kuva 4.)



Kuva 4. Yksittäisiä koodin pätkiä on rakennettu yhteen ja testattu erikseen. Sitten ne on integroitu ja testattu uudelleen. [11]

2.4 Regressiotestaus

Regressiotestauksen tarkoituksena on löytää uusia virheitä taikka regressioita. Regressiot ilmenevät silloin, kun ohjelman toimiva osa lakkautetaan toiminnasta tarkoituksenmukaisesti. Mutta tyypillisesti regressio ilmenee tahattomasti ohjelmistoa muutettaessa, kun muuttuneet osat kohtaavat vanhoja osia. Regressiotestausta suoritetaan uudella kokoonpanolla, ajamalla aiemmin ajettuja testejä uudelleen uusien virheiden löytämiseksi. Regressiotestaus voi oi olla sekä toiminnallisissa että rakenteellisissa tekniikoissa. [5, 11]

2.5 Käytetyt testausmenetelmät

DEMOS-Plan-sovelluksen PHP-ohjelmakoodi oli salattua, joten käytettävissä olivat ainoastaan toiminnalliset testaustekniikat, jotka perustuvat mustalaatikkotestaukseen.

DEMOS-Plan-sovellusta päätettiin testata tutkivalla testausmenetelmällä, koska se oli sopivin vaihtoehto työn rajaus huomioon ottaen. Testauksen ja virheiden korjaamisen jälkeen voidaan suorittaa sovellukselle lopullinen hyväksymistestaus asiakkaan laitteistolla asiakkaan luona, jotta saadaan lopullista varmuutta sovelluksen toimivuudesta asiakkaan asettamilla vaatimuksilla. Samalla saadaan lopullinen arvio ohjelmiston laadusta.

Koodin kehittäjällä ovat käytettävissä sekä rakenteelliset että toiminnalliset testausmenetelmät.

Ohjelmakoodin salauksesta huolimatta osa tietokannasta oli näkyvissä, mutta siihen ei perehdytty tässä opinnäytetyössä aihealueen rajauksen johdosta, koska lähdekoodia ei oltu tarkoitettu loppukäyttäjälle nähtäväksi. Käyttöoikeussopimuksessa rajattiin tietokannan rakenteen ja sisällön tarkastelu tämän työn ulkopuolelle. Jos siihen olisi perehdytty ja tietokantaa olisi testattu, siinä olisi voinut käyttää rakenteellista lasilaatikkotestaustekniikkaa.

3 KOTOISTUS

Kotoistus tarkoittaa ohjelman tai palvelun kääntämistä ja sovittamista uuteen ympäristöön ja kulttuuriin. Tämä toiminto vaatii ohjelman tai palvelun kääntämistä kyseiselle kielelle. Kotoistus ei ole pelkästään sanojen kääntämistä. Kotoistuksessa kulttuurin ja ympäristö lisäksi on otettava huomioon muun muassa miten tietyt asiat merkitään eri kielillä, kuten päiväys ja kellonaika, valuutta, mittayksiköt, nimet ja tittelit [12].

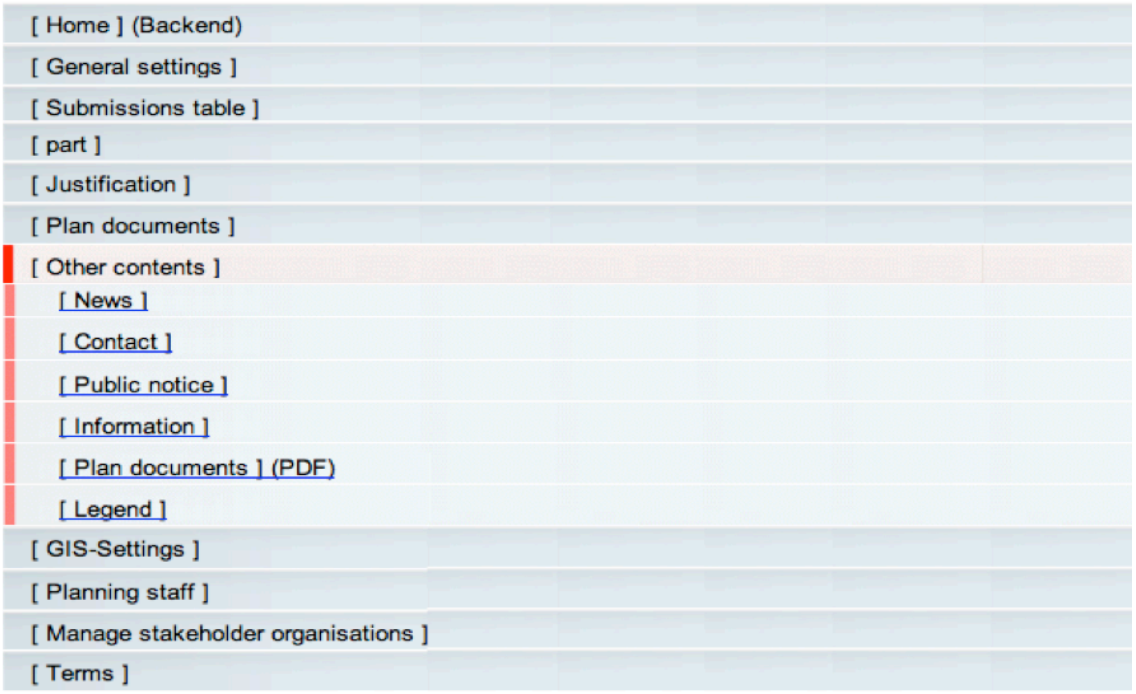
3.1 Kotoistuksen alku

Pöytäkoneet otettiin käyttöön vuonna 1980. Tietotekniikka alkoi hitaasti löytää tiensä käyttäjille, joilla ei ollut välttämättä taustaa tietokoneista, ohjelmoinnista taikka suunnittelusta. 1980-luvun alussa nähtiin ensimmäinen kansainvälistäminen, kun tietokoneohjelmistoja sekä -laitteistoa tuottava yhdysvaltalainen yritys Sun Microsystems aloitti toimintansa Euroopassa 1983 ja laajensi toiminnan Aasiaan ja Eurooppaan vuonna 1986. Microsoft oli aloittanut kansainvälisen toiminnan aikaisemmin, kun se oli avannut ensimmäisen ulkomaan myyntikonttorinsa Tokiossa marraskuussa 1978 ja aloitti laajenemisen Eurooppaan vuotta myöhemmin. Kun tietokoneiden käyttö ei enää ollut ainoastaan yrityksissä ja akatemioissa, silloin vaadittiin muutosta tuotteiden ominaisuuksissa ja toiminnassa, jotta ne sopisivat paremmin normaalin käyttäjän työpöydällä. Tässä tilanteessa ei riittänyt pelkästään, että kehitettiin uusia ohjelmia, jotta työ voitiin tehdä tehokkaammin. Ohjelman piti olla paikallisella kielellä, ja se oli myös sopeutettava paikallisiin standardeihin, tapoihin ja kulttuuriin. [13]

3.2 DEMOS-Plan-sovelluksen kotoistaminen

DEMOS-Plan-sovellus koostuu kahdesta eri osasta, itse sivustosta ja sivuston hallintaosiosta. Tämä johtuu siitä että, PHP-ohjelmakoodia ei päästy muokkaa-

maan. Sovelluksen kehittäjän oli tehtävä tietynlainen muokkaustyökalua sovellukselle. Sivuston hallintaosiossa voidaan muun muassa muokata, kääntää ja lisätä uutta materiaalia. Sivuston hallintaosio oli välttämätön kotouttamisprosessissa.



Kuva 5. Sivuston hallintaosion navigointipalkki.

Kuvassa 5 näkyy mitä valikkoja sivuston hallintaosion sivuosan navigointipalkissa on. Home-valikossa, joka on myös sivuston hallintaosion etusivu kerrotaan seuraavasti mitä kunkin valikon alla on. (Kuva 5.)

General settings-valikossa määritellään alkavaa projektia ja siihen liittyviä tietoja.

Submission table-valikossa tapahtuu ehdotusten muokkaaminen ja niistä lopullisen version lisääminen.

Part-valikossa nimetään suunnitelman, suunnitelmaan liittyvien asiakirjojen lisääminen ja niiden muokkaaminen manuaalisesti.

Justification-valikossa voidaan lisätä ja muokata projektin perusteluun liittyviä asiakirjoja.

Plan documents-valikossa lisätään muiden asiakirjojen, kuten asiantuntijoiden ja suunnitteluvaiheen asiakirjoja. Voidaan myös määritellä voivatko osallistujat kommentoida lisättyä asiakirjaa.

Other contents-valikossa lisätään ja muokataan kuvassa 8 olevan etusivun sivu- ja yläosan navigointipalkissa näkyvien valikkojen sisältöä News-, Contents-, Public notice-, Information-, Plan documents- ja legend-valikkojen kautta. (Kuva 8.)

GIS-Settings-valikossa muokataan kaikki karttaan liittyvät asetukset sekä esitetään karttamateriaalia ja suunnitelman visuaaliseen esitykseen liittyvää aineistoa.

Planning stuff-valikossa säilytetään suunnitteluviranomaisten henkilötietoja, jotta heille voidaan lisätä sivuston hallintaosion käyttöoikeuksia.

Manage stakeholders organisations-valikossa hallitaan järjestelmässä olevien sidosryhmäorganisaatioiden tietoja.

Terms-valikossa käännetään sivustossa ja sivuston hallintaosiossa näkyviä termejä muille kielille.

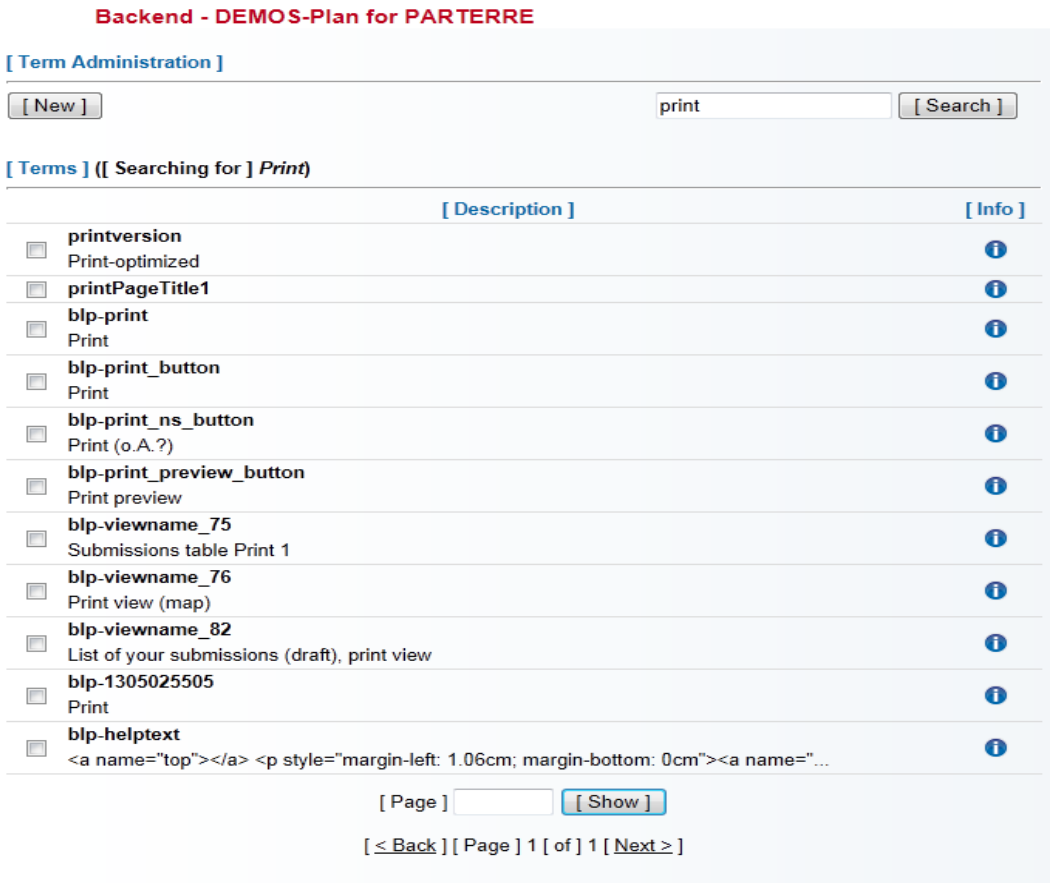
3.2.1 Käyttöliittymän kielenkäännös

Varsinainen kääntäminen onnistuu Terms-valikosta, jossa jokainen sovelluksessa käytetty termi on mainittu ja sitä voidaan kääntää. Kääntäminen onnistuu myös termien perässä olevasta ( )-kynä symbolista. Kynä symboli näkyi ainoastaan silloin, kun kirjauduttiin järjestelmään Translator-käyttäjällä, joka oli luotu termien kääntämisen vuoksi. Translator-oikeudet olivat meidän käytössä. Kun sivuston hallintaosion muut pääkäyttäjät kirjautuvat järjestelmään tunnuksillaan he eivät näe kynä symbolia. He pystyvät muokkaamaan termejä ainoastaan Terms-valikosta.

Termien kääntäminen oli hankalaa Terms-valikosta, koska samoja termejä käytettiin monella eri sivuston sivuilla eri tunnistenimellä, kuten kuvassa 6 näkyy.

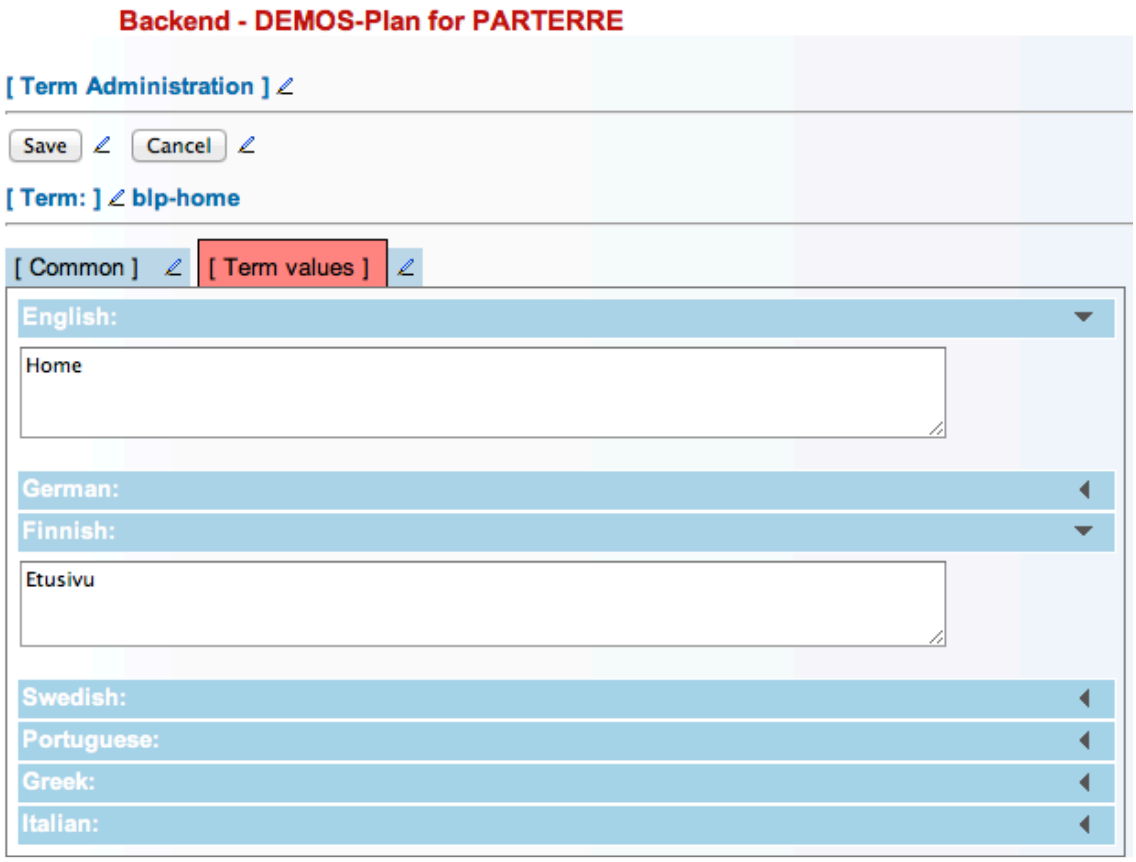
Termejä muokattaessa Terms-valikon kautta ei tiedetty mikä tietty termi muuttui sivustolla. Täten oli järkevämpää käyttää Translator-käyttäjän kynä symbolia kääntämiseen, koska siinä tiedettiin varmasti mitä termiä oltiin muokkaamassa. (Kuva 6.)

Osa samannimisistä termeistä oli sidottu yhteen. Tämä tarkoitti sitä, että muokkaamalla yhtä termiä niistä, muuttuivat kaikki samannimiset termit samalla. Esimerkiksi, jos muutettiin sivustolla olevaa Page-nimistä termiä Sivu-nimiseksi, niin kaikki muutkin sivustolla olevat Page-nimiset termit muuttuivat Sivu-nimiseksi. Tämä oli hyödyllinen ja kääntämistä helpottava ja nopeuttava käytäntö. Mutta suurin osa sivustolla olevista samoista termistä eivät olleet sidottu toisiinsa ja niitä jouduttiin muokkaamaan yksitellen. Joitain termejä ei pystytty muokkaamaan kummallakaan tavalla. Niitä ei ollut Terms-valikon termien hallintaosiossa. Se johtui varmaankin sovelluksen käyttäjän huolimattomuusvirheestä.



Kuva 6. Terms-valikon sanahaku.

Valitsemalla termiä kuvan 6 Terms-valikon termien hallintaosiosta tai Translator-käyttäjän termien perässä olevasta kynä-symbolista aukeaa kummassakin tapauksessa kuvan 7 valikko, jossa on esimerkkinä sivuston etusivun Home-välilehti. Kaikissa kuvan 7 kielivalikoissa oletusasetuksena on englannin kieli. Käytössämme oleva sovellus oli asetettu hakemaan termejä kuvan 7 Finnish-valikosta. Muuttamalla valikossa ollutta Home-sanaa etusivuksi ja tallentamalla toimistoa, muuttui sivustolla ollut Home-välilehti Etusivuksi-välilehdeksi, (Kuva 6, 7)



Kuva 7. Terms-valikon termien hallintatyökalu.

3.2.2 Kotoistukseen liittyvät ongelmat

DEMOS-Plan-sovellukseen on tehtävä tiettyjä muutoksia, jotta se voidaan ottaa Suomessa käyttöön. Seuraavaksi mainitaan muutamia ongelmatapauksia, joita havaittiin sovellusta kotouttaessa ja ehdotetaan niihin liittyviä ratkaisuja.

Suomessa viranomaiskäytössä kielivalinnat merkitään tekstillä toisin kuin DEMOS-Plan-sovelluksen sivustolla, jossa ne oli merkitty maankohtaisilla lipuilla, kuten kuvassa 8 näkyy. Esimerkiksi Suomen lippu viittaa maahan, jonka nimi on Suomi eikä suomen kieleen. Sitä ei ole suositeltavaa käyttää, koska tietyissä maissa, kuten esimerkiksi Suomessa, puhutaan suomen kielen lisäksi ruotsia ja saamea. Ruotsin lipun käyttäminen suomenkielisissä viranomaispalveluissa ei ole suositeltavaa, koska se viittaa kielen sijasta toiseen valtioon [14]. Kielivalintojen muuttaminen teksteiksi onnistuu ainoastaan järjestelmän toimittajan ohjelmoijien toimesta PHP-ohjelmakoodin salauksen vuoksi. (Kuva 8.)

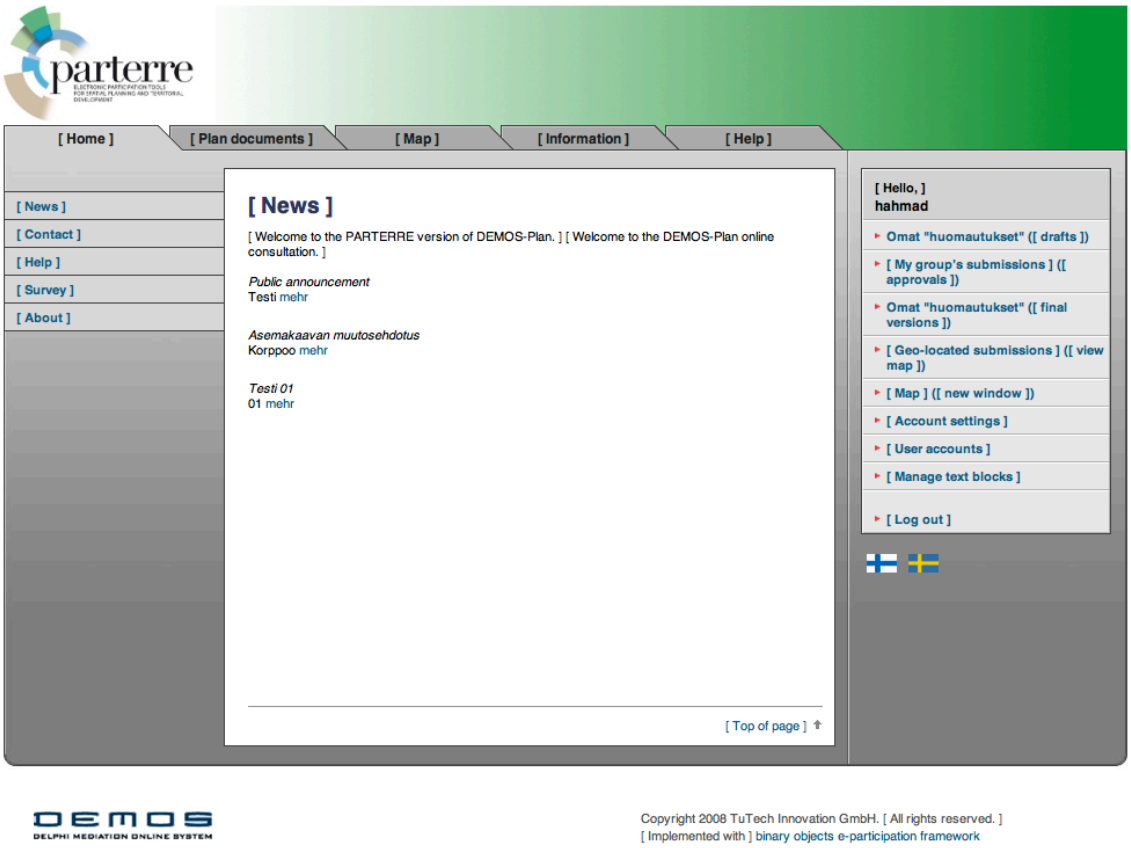
Suomen- ja ruotsinkieliselle sivuostoille oli tehty kummallekin oma erillinen palvelin. Kummassakin palvelimessa oli oma tietokantansa. Tästä syystä palvelimet eivät keskustelleet keskenään. Tämä tarkoittaa sitä, että suomenkieliselle sivustolle kirjoitetut kommentit eivät näkyneet ruotsinkielisessä sivustossa taikka toisinpäin. Tämä ei tietenkään sovellu suomalaiseen menetelmään, jossa kummankin kielen kommentit pitäisi näkyä yhtäaikaisesti kummallakin sivustolla.

DEMOS-Plan-sovellus on tehty saksalaisen kaavoitusmenetelmän mukaan, jossa on tietynlainen organisaatiotaso taikka kansalaisjärjestö välissä, joka kerää kansalaisilta kaavoitukseen liittyviä kommentteja ja huomautuksia lähettääkseen niitä eteenpäin julkaistaviksi. Siinä siis ei kysytä yksittäisiltä kansalaisilta mielipiteitä, vaan mielipiteitä on koottu yhteen. Kun taas Suomessa tällaista käytäntöä on hyvin harvoin käytössä.

Paraisten kaupungin vaakuna tulee olla sivuston etusivulla PARTERRE-projektin logon lisäksi. Tämä käytäntö kuuluu kotoistukseen. Uuden logon tai vaakunan voi lisätä ainoastaan sovelluksen kehittäjä.

4 TYÖN TOTEUTUS

Kuvassa 8 näkyy lähtötilanne, josta aloitettiin sovelluksen testaus ja kotoistus.



Kuva 8. Demos-Plan-sovelluksen sivuston etusivu, kotoistamaton.

4.1 Tuotekehitysympäristö

Työ toteutettiin LAMP-ympäristössä virtuaalipalvelimelle. Tähän palvelinympäristöön tutustuttiin jo aikaisemmin ohjatun sähköisen kaupunkikokouksen eTM-projektin [15] yhteydessä ennen tämän työn aloittamista. LAMP-ympäristö koostuu neljästä (Linux, Apache, MySQL ja PHP) avoimeen lähdekoodiin perustuvista ohjelmistosta, jotka yhdessä muodostavat WWW-palvelimen, jossa voidaan suorittaa dynaamisia verkkosivuja [16].

4.1.1 Linux

Linux on avoimeen lähdekoodiin perustuva käyttöjärjestelmä, jolla tietokoneen eri ohjelmia voidaan suorittaa. Samanaikaisesti käyttäjä voi hallita tietokoneen laitteistoa, jotta saadaan suoritettua halutut toiminnot [17]. Käytimme virtuaalipalvelimessa Linux Ubuntu 10.04.4 LTS –jakeluversiota.

4.1.2 Apache

Apache on avoimeen lähdekoodiin perustuva http-palvelin, joka tarjoaa täyden valikoiman verkkopalvelin ominaisuuksia, kuten CGI, SSL ja virtuaalisia verkkotunnuksi (domaineja). Apache oli alun perin suunniteltu Unix-ympäristöihin, mutta jälkeenpäin julkaistu muille käyttöjärjestelmille [18]. Käytimme virtuaalipalvelimessa Apache 2.2.14 Ubuntu-jakeluversiota.

4.1.3 MySQL

MySQL on avoimeen lähdekoodiin perustuva relaatiotietokannan hallintajärjestelmä. Se on erittäin nopea, luotettava ja joustava tietokantojen hallintajärjestelmä. Se tarjoaa erittäin korkealaatuisen suorituskyvyn ja se on monitasoinen ja monen käyttäjän relaatiotietokannan hallintajärjestelmä [19]. Käytimme virtuaalipalvelimessa MySQL 5.1.62-0ubuntu0.10.04.1 Ubuntu-jakeluversiota.

4.1.4 PHP

PHP on laajasti käytetty avoimeen lähdekoodiin perustuva yleiskäyttöinen ohjelmointikieli, joka sopii erityisesti verkkokehitykseen ja voidaan upottaa HTML-kieleen. [20]

PHP:tä jouduttiin päivittämää taaksepäin jakeluversiosta 5.3 aikaisempaan PHP 5.2.10-2ubuntu6 with Suhosin-Patch 0.9.7 –jakeluversioon yhteensopimattomuusongelmien vuoksi.

4.1.5 IonCube

IonCube on kaupallinen menetelmä, jolla PHP 4 ja 5 lähdekoodia salataan luvattomilta muutoksilta ja varkaudelta, koska itse ohjelmakoodia ei voi patentoida. Laillisesti näkyvästä koodipätkästä voi ottaa mallia. Sovelluksen ohjelmakoodi oli salattu ionCube-ohjelmistolla.

4.2 DEMOS-Plan-sovelluksen testaus

Testattava ohjelmisto on tarkoitus ottaa käyttöön asemakaavoja muutettaessa. Sivustossa ilmoitetaan muutettavista alueista. Lähistöllä asuvat asukkaat, joita muutokset koskevat, voivat kommentoida muutoksia sivustolla. Samalla asukkaat voivat merkitä sivustolla olevaan karttaan, mitä aluetta tarkalleen huomautukset koskevat.

Testaus suoritettiin tutkivalla testausmenetelmällä, jossa ei etukäteen suunniteltu testaustapauksia, vaan testejä luotiin aikaisempien testitulosten perusteella. Ohjelmakoodiin ei perehdytty, eikä perehtyminen ollut edes mahdollista ohjelmakoodin salauksen vuoksi. Salaus oli toteutettu kaupallisella ionCube-ohjelmistolla. Sovelluksen rakennetta ei tiedetty, mutta osattiin käyttää sovellusta ja tiedettiin, miltä sen kuuluisi näyttää. Yritettiin lisätä, muokata ja poistaa asiakirjoja ja tietoa sivuston hallintaosiosta. Jos toiminnot onnistuivat, tarkistettiin, näkyykö suoritettu toiminto oletetulla tavalla ja halutussa paikassa. Testien perusteella löydetyt virheet välitettiin sivuston kehittäjälle viikoittaisten tapaamisten yhteydessä.

Testaus aloitettiin käyttämällä sekä sivustoa että sivuston hallintaosiota normaalisti sille tarkoitetulla tavalla. Sivustolla ilmenneestä virhetapauksesta merkittiin sijainti ja se, miten virhetapaus oli havaittu. Tarvittaessa virhetapauksesta otettiin kuvankaappaus selkeyttämään tapausta.

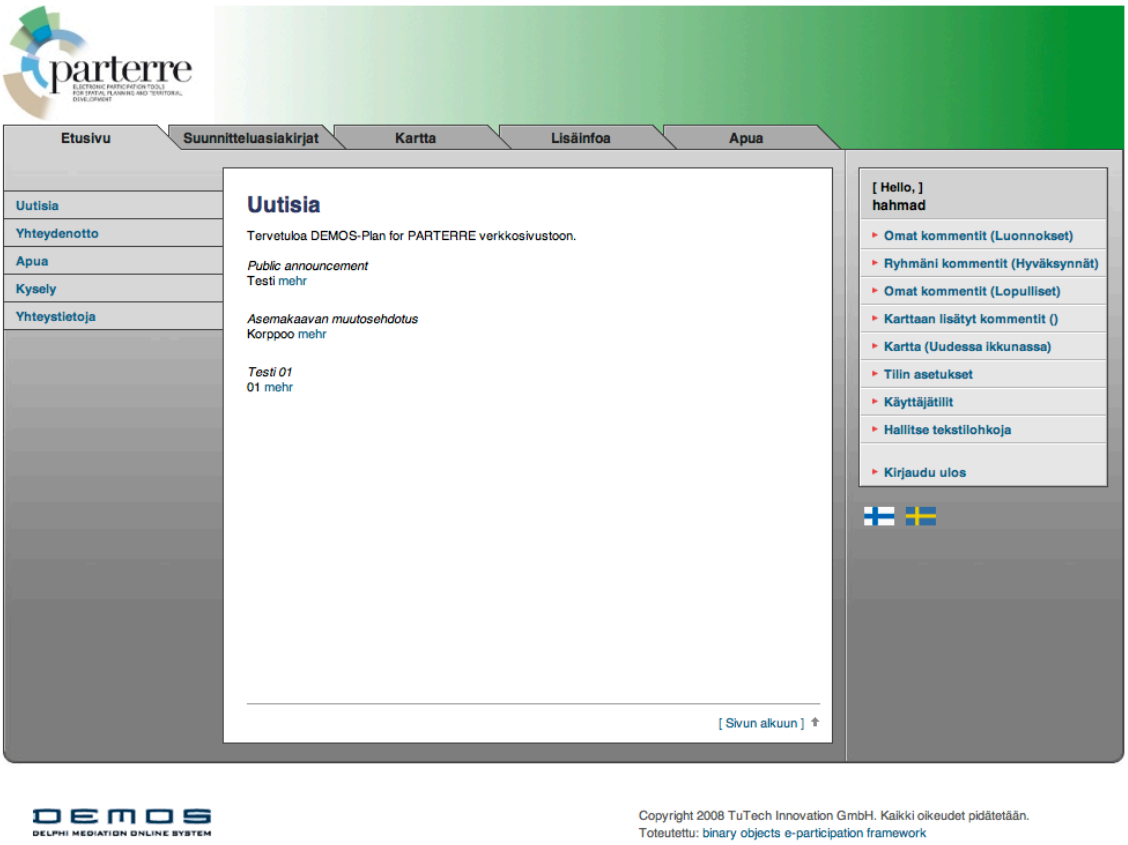
Taulukossa 1 on listattu liitteissä ilmoitettuja virhetapauksia, jotka ovat pieni osa kaikista sivustolla olevista virhetapauksista. Virhetapaukset lähetettiin sivuston kehittäjälle korjattaviksi. Taulukkoon on merkitty, ovatko virheet olleet kriittisiä ohjelmiston käyttöönoton kannalta ja onko niitä korjattu.

Taulukko 1. Liitteinä olevien virhetapauksien kriittisyys ja se, onko virheet korjattu.

	Kiriittinen		Korjattu	
Löydetty virhe	**Kyllä**	**Ei**	**Kyllä**	**Ei**
Liite 1	X			X
Liite 2	X			X
Liite 3	X			X
Liite 4	X			X
Liite 5		X		X
Liite 6	X			X
Liite 7		X		X
Liite 8	X			X
Liite 9	X			X
Liite 10		X		X
Liite 11	X			X
Liite 12		X		X

Myös Aapo Mannelin etsi virheitä Demos-Plan-sovelluksesta opinnäytetyössään [3] Turun ammattikorkeakoulun tietotekniikan opiskelijoiden testauskurssin osallistujien avustuksella. Niistä tehtiin testitapauksia, joissa on ilmoitettu testausvaiheet ja onko niissä saatu testitulos hyväksytty vai hylätty. Suurin osa testitapauksista oli hyväksyttyjä.

Kuvassa 9 näkyy kotoistettu Demos-Plan-sovelluksen etusivu.



Kuva 9. Demos-Plan-sovelluksen sivuston etusivu, kotoistettu.

4.3 DEMOS-Plan-sovelluksen tietoturva

DEMOS-Plan-sovelluksen tietoturvan haavoittuvuutta testattiin SQL-injektiohyökkäyksillä ja Cross Site Scripting-komentosarjalla. Suoritettujen testien perusteella voidaan todeta, että sovellus ei ollut haavoittuvainen edellä mainituille hyökkäyksille.

4.3.1 SQL-injektio

SQL-injektio on tietokantapohjaisissa sovelluksissa esiintyvä hyökkäystekniikka, jossa hyökkääjä luo tai muuttaa olemassa olevia SQL-komentoja paljas-

taakseen salattuja tietoja [21]. Tämä tapahtuu syöttämällä sovelluksen vapaamuotoisiin tekstikenttiin haitallista scriptiä.

4.3.2 Cross Site Scripting

Cross Site Scripting (XSS) ovat injektiotyyppisiä hyökkäyksiä, joissa luotettavan verkkosivuston vapaamuotoiseen tekstikenttään syötetään haitallista koodia, jota yritetään lähettää toisille sivuston käyttäjille. Hyökkäystä voidaan suorittaa kohdejärjestelmässä syöttämällä Javascriptiä järjestelmään kirjautuneen käyttäjän oikeustasolla ja houkuttelemalla toisia käyttäjiä avaamaan esimerkiksi hyökkäyksessä muotoiltua linkkiä. [22]

4.4 Ohjelmiston pilotointi

DEMOS-Plan-sovellusta pilotoitiin Salossa asunnon kerrospiirrosten kommentoinnissa ja Paraisten kaupungin kyläkaavoitus- ja maankäyttösuunnittelussa.

4.5 Lounaispaikka

Lounaispaikka on avoimen paikkatiedon portaali, joka tarjoaa mahdollisuuden paikkatietoaineistojen katseluun [23]. Lounaispaikka voisi olla parempi vaihtoehto Paraisten kaupungin käyttöön, koska DEMOS-Plan-sovelluksessa oli huomattavasti virheitä.

5 YHTEENVETO

Virtuaaliympäristöön asennettua DEMOS-Plan-sovellusta ryhdyttiin testaamaan ja kotoistamaan, jotta se palvelisi Paraisten kaupungin tarpeita aluesuunnittelussa. Sovellusta päästiin testaamaan hyvin ja löydettiin jonkin verran ohjelmavirheitä. Termien kääntäminen onnistui helposti käännöstyökalulla. Mutta PHP-ohjelmakoodin salauksen vuoksi muita kotouttamiseen liittyviä tärkeitä asioita ei päästy tekemään, kuten kielivalinnan muuttamista tekstivalinnaksi, suomen- ja ruotsinkieliselle sivuostoille yhteistä palvelinta ja Paraisten kaupungin vaakunan lisäämistä sivustolle.

DEMOS-Plan-sovelluksen tietoturvan haavoittuvuutta testattiin SQL-injektiohyökkäyksillä ja Cross Site Scripting-komentosarjalla. Todettiin, ettei sovellus ollut haavoittuvainen kyseisille hyökkäyksille.

Varsinaisien testitulosten perusteella työssä todettiin, ettei DEMOS-Plan-sovellus ollutkaan paras vaihtoehto Paraisten kaupungin käyttöön. Tämä johtui hyvin hitaasta sovelluksen kehittäjän sovelluksen päivitysnopeudesta. Sovellus oli kerran päivitetty ennen kuin sen tutkiminen aloitettiin. Tämän jälkeen ei missään vaiheessa saatu uutta päivitysversioita sovelluksesta, vaikka virhetapauksia välitettiin sovelluksen kehittäjälle, eikä korjattu ainuttakaan virhettä. Kehittäjälle välitetyt virhetapaukset ovat liitteinä.

Vaikka virheet korjattaisiin, sovellus ei edelleenkään sovellu käytettäväksi kotouttamisongelmien takia. Tuotteen tuki oli hyvin heikkoa, ja tuote vaatii vielä kehittämistä, jotta se soveltuisi Suomen olosuhteisiin. Suomesta löytyy muita vaihtoehtoja, kuten Lounaispaikka [23] Paraisten kaupungin tarkoitukseen.

Vaikka suuri osa suoritetuista testitapauksista onnistui ja sovellus toimi oikein, sovelluksessa oli vielä huomattavasti virheitä. Tehtiin johtopäätös, että sovellus on vielä kehitysvaiheessa ja se ei ole haluttuun tarkoitukseen soveltuva. Suuresta virhemäärästä johtuen todettiin, ettei tutkivalla testausmenetelmällä lähdetä etsimään kaikkia virheitä, vaan käytetään hyväksi sovelluksen toimivia osioi-

ta. Virheistä huolimatta tuotetta päästiin käyttämään rajoitetusti pilottikokouksessa Salossa. Tosin sen hyöty rajoittui ainoastaan Hakastaron PDF-suunnitteluasiakirjojen esittämiseen, koska niitä ei päästy kommentoimaan verkossa, sen sijaan niitä kommentoitiin paperilla.

Sovellusta piti käyttää Paraisten kaupungin asemakaavoja muutettaessa, mutta ideasta luovuttiin, koska korjattua järjestelmää ei koskaan saatu.

LÄHTEET

[1] Parterre – technology for participatory territorial planning, [www-dokumentti]. Saatavilla: http://parterre.ning.com/ (Luettu: 18.05.2012)

[2] Parterre Project, [www-dokumentti]. Saatavilla: http://www.parterre-project.eu/?demos_plan=1 (Luettu: 18.05.2012)

[3] Mannelin, Aapo, Web 2.0:n käyttö kaupunkisuunnittelussa –PARTERRE-projekti, Opinnäyte-työ, Turun ammattikorkeakoulu, 2011 (Luettu: 19.04.2012)

[4] Hurme, Jarkko, The benefits of using UML-modeling tools in evaluation and testing of eTM software, Opinnäytetyö, Turun ammattikorkeakoulu, 2011 (Luettu: 21.04.2012)

[5] Software testing, [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/Software_testing (Luettu: 29.03.2012)

[6] Tutkiva testaus hyväksymistestauksen menetelmänä, [www-dokumentti]. Saatavilla: http://www.soberit.hut.fi/T-76.5650/Spring_2004/Papers/E.Halme_76650_final.pdf (Luettu: 30.03.2012)

[7] Myers, Badgett, Sandler, The Art of Software Testing, John Wiley & Sons, Inc., Hoboken, New Jersey, 2011 (Luettu: 30.04.2012)

[8] Loveland, Shannon, Miller, Geoffrey, Software Testing Techniques: Finding the Defects That Matters, Charles River Media, Hingham, MA, US, 2004 (Luettu: 20.05.2012)

[9] Unit Testing – Best Practices & Techniques, [www-dokumentti]. Saatavilla: http://www.softwaretestingstuff.com/2010/09/unit-testing-best-practices-techniques.html (Luettu: 06.04.2012)

[10] Black-box testing, [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/Black-box_testing (Luettu: 06.04.2012)

[11] Ron Patton, Software Testing, 2nd Edition, Sams, Indiana, 2006 (Luettu: 11.05.2012)

[12] Kansainvälistäminen ja lokalisointi, [www-dokumentti]. Saatavilla: http://fi.wikipedia.org/wiki/Internationalisointi_ja_lokalisointi (Luettu: 02.04.2012)

[13] The evolution of the localization, [www-dokumentti]. Saatavilla: http://isg.urv.es/library/papers/Esselink_Evolution.pdf (Luettu: 03.04.2012)

[14] JHS 129, Julkishallinnon verkkopalvelun suunnittelun ja toteuttamisen periaatteet, 2005 (Luettu: 02.05.2012)

[15] Parterre Project, [www-dokumentti]. Saatavilla: http://www.parterre-project.eu/?electronic_town_meeting=1 (Luettu: 16.04.2012)

[16] LAMP (software bundle), [www-dokumentti]. Saatavilla: http://en.wikipedia.org/wiki/LAMP_(software_bundle) (Luettu: 05.05.2012)

[17] What Is Linux: An Overview of the Linux Operating System, [www-dokumentti]. Saatavilla: https://www.linux.com/learn/resource-center/376-linux-is-everywhere-an-overview-of-the-linux-operating-system (Luettu: 05.05.2012)

[18] What Is Apache Web Server?, [www-dokumentti]. Saatavilla: http://compnetworking.about.com/cs/webservers/g/bldef_apache.htm (Luettu: 05.05.2012)

[19] What is MySQL, [www-dokumetti]. Saatavilla: http://www.roseindia.net/mysql/mysql5/what-is-mysql.shtml (Luettu: 05.05.2012)

[20] PHP: What is PHP? – Manual, [www-dokumentti]. Saatavilla: http://php.net/manual/en/intro-whatis.php (Luettu: 05.05.2012)

[21] PHP:SQL Injection – Manual, [www-dokumentti]. Saatavilla: http://php.net/manual/en/security.database.sql-injection.php (Luettu: 25.04.2012)

[22] Luusua, Antti, Tietoturvallisuus PHP- ja MySQL-pohjaisessa www-sovelluksessa, Opinnäytetyö, Rovaniemen ammattikorkeakoulu, 2011 (Luettu: 28.04.2012)

[23] Lounaispaikka, Avoimen paikkatiedon portaali, [www-dokumentti]. Saatavilla: http://www.lounaispaikka.fi/ (Luettu: 24.05.2012)

LIITE

Testauksessa havaitut virheet

Tässä on osa virheistä joita havaittiin sovellusta testattaessa. Virheet on välitetty sivusto kehittäjälle.

Liite 1

Hallintaosiossa Other contact / Contact –valikosta ei voida lisätä mitään tietoja. Lisää ja poista painikkeet puuttuvat ainakin.

Liite 2

Perustelut pitäisi näkyä sivustolla perustelut –otsikon alla mutta siitä tekstiosiosta näkyy vaan pieni osa.

Liite 3

Ilman “http://” linkistä tulee sisäinen linkki ja se ei toimi silloin. Lisäksi linkkien nimet ei näy hallintaosiossa ja niitä ei voi muokata. Ainoastaan poistaminen on mahdollista. Kuvassa on kaksi linkkiä lisätty, mutta niiden otsikko ei näy. Katso alla oleva kuva.

Liite 4

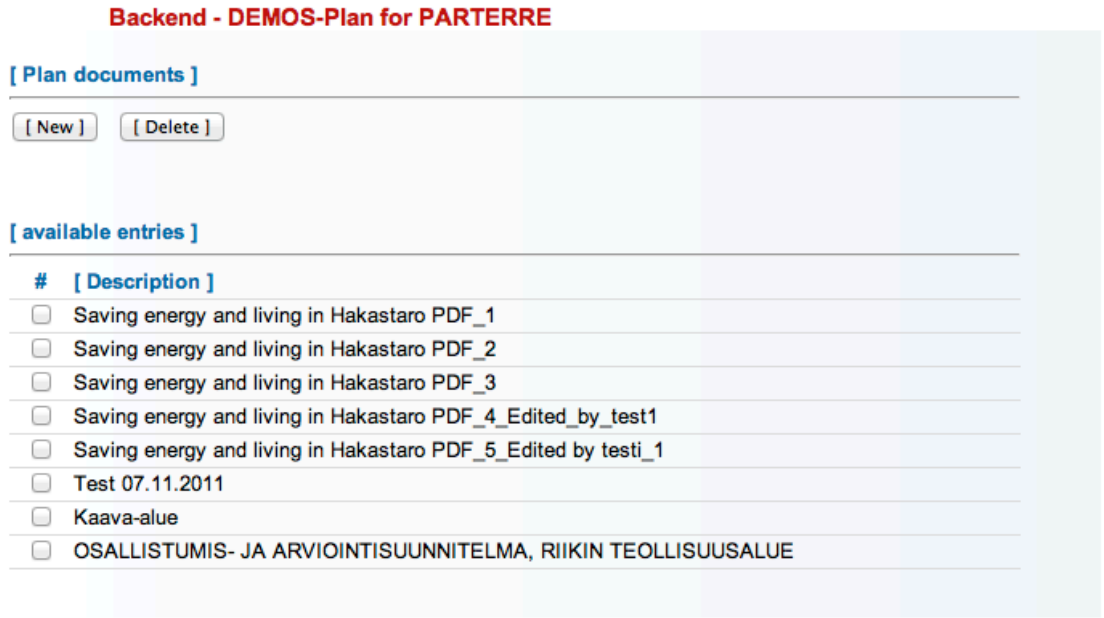
Other contents / Ledge ei voida lisätä mitään tietoja. Lisää ja poista painikkeet puuttuvat ainakin. Katso alla oleva kuva.

Liite 5

“Other Documents/ [Plan documents] (PDF)” on tyhjä. Lisää ja poista painikkeet puuttuvat ainakin. Katso alla oleva kuva.

Mitä eroa on “Other Documents / [Plan documents] (PDF)” ja “Plan Documents” välillä, joka on pääotsikkona navigointi palkissa”?

Siellä PDF dokumentit on näkyvissä. Katso alla oleva kuva.



Liite 6

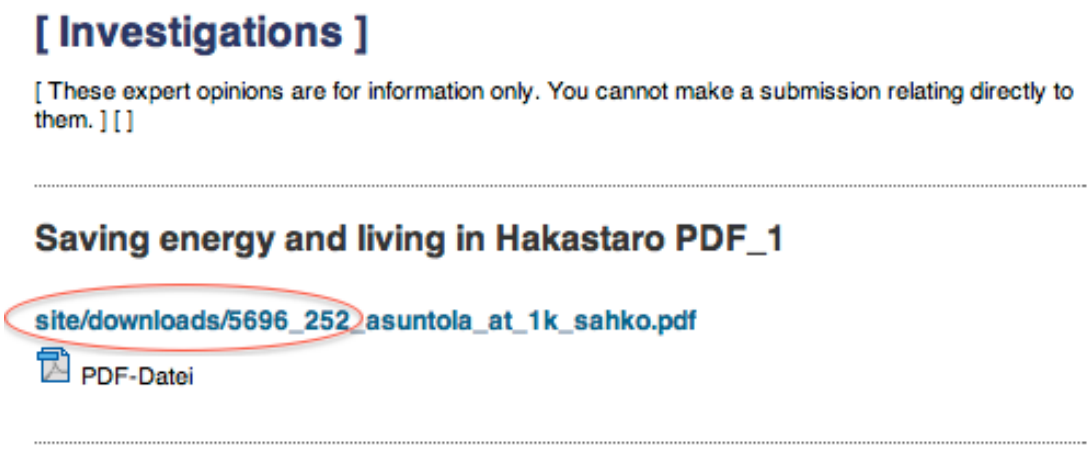
Ei voida ladata PDF dokumenttia klikkaamalla ympäröityä linkkiä Hallintaosion Plan Dokuments navigointipalkista. Katso alla oleva kuva.

Likkiä ei voi ladata myöskään sivustosta, Plan documents-investigation välilehden alta. Katso alla oleva kuva.

⇨ The webpage cannot be found, HTTP 404

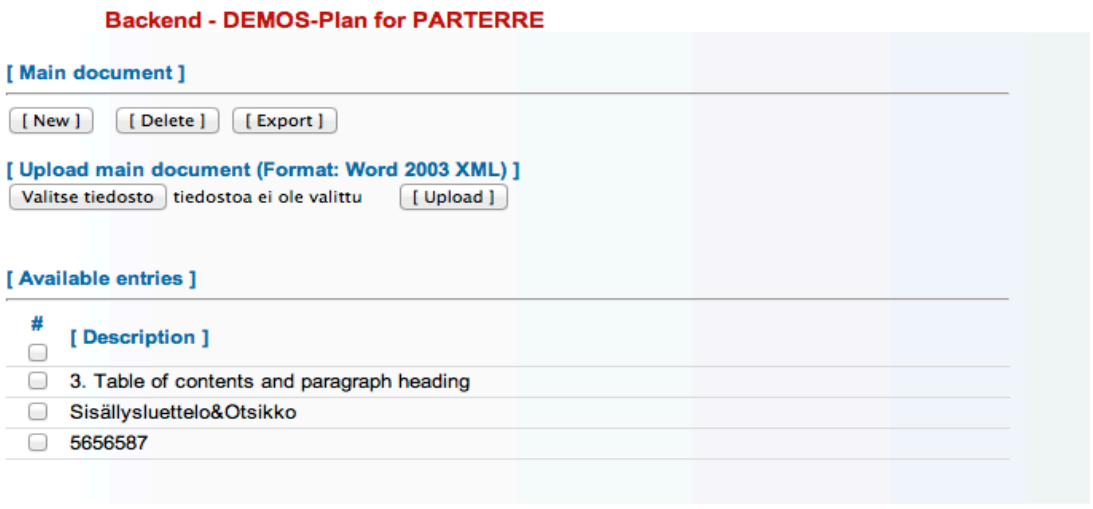
Liite 7

Plan Documents välilehden alla Investigations sivulla PDF dokumenttien nimet ovat monimutkaiset. Nimen eteen tulee itsestään site/download ja numerosarja. Katso alla oleva kuva.



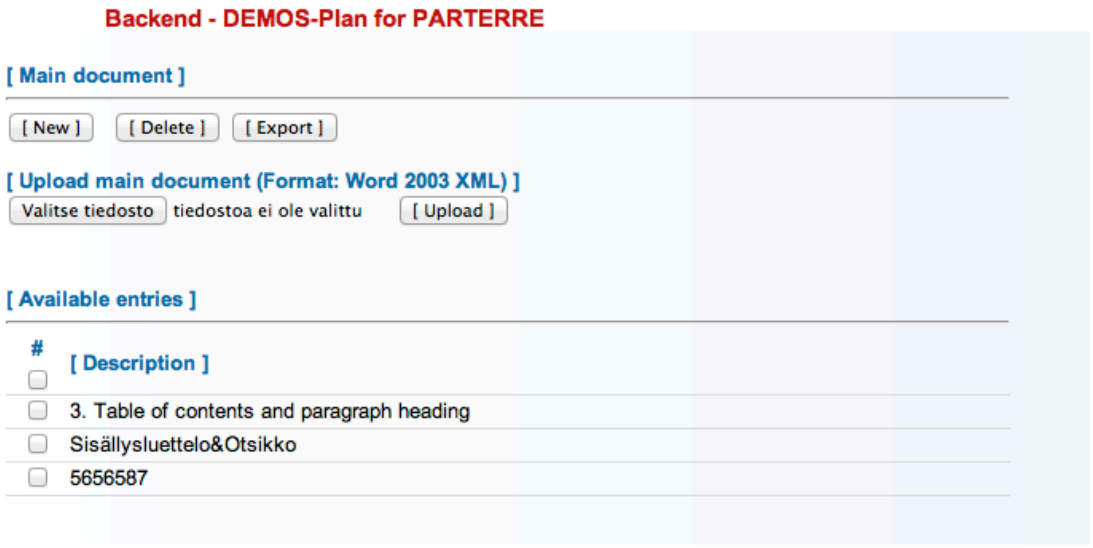
Liite 8

Hallintaosiosta Part – valikosta lisätyt dokumentit näkyy sivustolla Plan Documents –välilehden alla, mutta dokumentin tekstiosiosta näkyy vaan pieni osa.



Liite 9

Tarvitsemme lisätietoa siitä miten hallintaosion Part –valikosta päätiedosto on lisättävissä. Katso alla oleva kuva.



Liite 10

Is it possible to change navigation titles in main menu: our first pilot doesn't include so much spatial planning, but we would like to use (and practice) with this and comment floor plans of housing project.

Liite 11

Our users disappear from "planning staff" -list, it might be related to changing affiliation and/or role.

Liite 12

Is it possible to run two copies of Demos here locally (so we can separately develop/change page for two different pilots 1. with floor plan, 2. with spatial plan in archipelago)

We tried to copy server, but it might be impossible for us to change information about database connection (files seems to be encoded with ioncube)