XMLdation Validaattorin Lokijärjestelmä

Jaakko Kantonen

Opinnäytetyö
Toukokuu 2012
Tietotekniikka
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

JAAKKO KANTONEN:
XMLdation Validaattorin Lokijärjestelmä

Opinnäytetyö 45 sivua, josta liitteitä 5 sivua
Toukokuu 2012

Tässä työssä käsitellään Symfony web PHP frameworkilla tehdyn sivuston lokijärjestelmän suunnittelua ja toteuttamista.

Työssä käydään läpi lokijärjestelmän luominen kehitysympäristön pystyttämisestä valmiiseen järjestelmään. Samalla käydään läpi Symfony frameworkin perusteita.

Työ toteutettiin yritykselle korvaamaan vanhaa lokijärjestelmää. Edeltäjäänsä verrattuna uuden järjestelmän pitäisi olla huomattavasti hallittavampi ja kattavampi. Lisäksi oli tärkeää luoda uusi tietokanta lokikirjauksia varten, sekä muodostaa yhteys siihen, ja täten irroitaa loki muusta sivuston datasta.

Tätä työtä seuraamalla lukija kykenee pystyttämään paikallisesti toimivan palvelimen ja kehittämään valmiiseen Symfony-pohjaiseen sivustoon uuden moduulin. Työstä lukija voi myös saada ideoita MVC-pohjaisen sivuston luomiseen.

Lopuksi käydään lokijärjestelmän jatkokehitysmahdollisuuksia läpi.

Asiasanat: symfony, php, web-ohjelmointi, mysql, tietokanta, MVC

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Computer sciences
Software engineering

JAAKKO KANTONEN:
XMLdation Validator Logging System

Bachelor's thesis 45 pages, appendices 5 pages
May 2012

This thesis covers the planning and creation of a logging system for a Symfony web PHP framework –based website.

The thesis consists of the whole creation process from setting up the development environment to a final product. Basics of Symfony web PHP framework are also covered.

The logging system was developed to replace an older logging system. Compared to its predecessor, the new logging system should be noticeably more controllable and extensive. Additionally it was important to create a new database for the log entries, and create a connection to it, thus effectively separating the log from the other data stored on the site.

By following this thesis, the reader can set up a locally working server and develop a new module for a Symfony-based website. The reader may also gain ideas towards developing a MVC-based website.

Further development possibilities for the logging system are investigated in the end part of the thesis.

Key words: symfony, php, web-programming, mysql, database, MVC

SISÄLLYS

ERITYISSANASTO

Linux	Avoimen lähdekoodin käyttöjärjestelmä, josta on tehty useita versioita
Ubuntu	Ilmainen Linux-käyttöjärjestelmä
LAMP	Linux, Apache, MySQL, PHP asennuspaketti
Apache	Avoimen lähdekoodin HTTP palvelinohjelmisto
HTTP	Hypertext Transfer Protocol, selaimien ja palvelinten välinen tiedonsiirtoprotokolla.
Symfony	Avoimen lähdekoodin PHP web framework
PHP	Ohjelmointikieli, jota käytetään usein dynaamisten internet-sivustojen luonnissa
framework	Kokoelma uudelleenkäytettäviä kirjastoja
MVC	Model-view-controller arkkitehtuuri, jota käytetään usein graafisten käyttöliittymien suunnittelussa ja ohjelmoinnissa
Excel	Microsoft Office:n taulukkolaskentaohjelma, tai sen tiedosto
XML	Merkintäkieli, joka on ihmis-luettavassa formaatissa. Käytetään mm. uusissa pankkien välisissä viesteissä
MySQL	Avoimen lähdekoodin relaatiotietokantaohjelmisto
ORM	Object-Relational Mapping, kahden yhteensopimattoman datatyypin käännöstekniikka
applikaatio	Tässä työssä: symfony framework:in sivustokokonaisuus.
NetBeans	Ohjelmointiympäristö, joka tukee mm. PHP-ohjelmointia
Windows	Microsoftin kaupallinen käyttöjärjestelmä
VirtualBox	x86-virtualisointi ohjelmistopaketti
Debian	Ilmainen Linux-käyttöjärjestelmä
Terminal	Ubuntun komentorivikehote
APT	Advanced Package Tool, Debianin ohjelmanhallintatyökalu
Update Manager	APT:n graafinen hallintatyökalu
PEAR	PHP Extension and Application Repository, tarjoaa mm avoimen lähdekoodin PHP kirjastoja
POST	HTTP:n pyyntömethodi
Ubuntu Software Centre	Ubuntun ohjelmisto, jolla voi asentaa lukuisia avoimen lähdekoodin ohjelmia

subversion (SVN)	Versionhallinta
PHPmyAdmin	PHP:lla kirjoitettu ohjelmisto MySQL-tietokannan hallintaa varten
IRC	Internet-relay chat, keskusteluprotokolla. Voi myös viitata ohjelmaan, joka on suunniteltu pelkästään viestimiseen kyseisellä protokollalla
PostgreSQL	Avoimen lähdekoodin tietokantaohjelmisto
Oracle Database	Oraclen tietokantaohjelmisto
Microsoft SQL Server	Microsoftin tietokantaohjelmisto
validointi	Tietojen oikeellisuuden tarkistus (esim. XML-tiedoston rakenteen tarkistus)
simulointi	Pankkikohtaisen XML-vastausviestin luominen ja palauttaminen lähetettävästä XML-maksuaineistosta
OCL	Object Constraint Language, deklaratiivinen ohjelmointikieli
aos2	Suomalainen pankkiviestistandardi
StarUML	Avoimen lähdekoodin ohjelmiston mallinnustyökalu
filtteri	Joko 1. Symfonyn järjestelmän osa, joka ajetaan jokaisen tapahtuman yhteydessä tai 2. Hakusuodatin, jota käytetään tapahtumien suodattamiseen listasta
propel	PHP5 ORM tietokannalle
doctrine	PHP5 ORM tietokannalle
YML	XML:n tapainen merkintäkieli

1 JOHDANTO

Tämän työn tavoitteena on kuvata työtä varten ohjelmoitua lokijärjestelmää XMLdation-nimisen yrityksen XML-validointi sivustolle, sekä sivutaan työssä tarvittavia työkaluja ja niiden asennusta. Suunniteltavan ja toteutettavan lokijärjestelmän tulisi rekisteröidä periaatteessa kaikki suuremmat, käyttäjän aiheuttamat, tapahtumat ja kirjata tiedot niistä uuteen MySQL tietokantaan. Yritys pystyy lokijärjestelmän tietojen avulla seuraamaan asiakkaidensa toimintaa, visualisoimaan sivuston käyttöä, sekä mahdollisissa vikatilanteissa paikantamaan vian aiheuttajan. Järjestelmä rakennetaan jo käytössä olevan sivuston kiinteäksi osaksi Symfony framework:iin. Lokikirjauksista kerätään palaute Symfony:n backend-applikaation uuteen ”Logging”-nimiseen moduuliin, johon vain järjestelmänvalvojilla on pääsy.

2 KEHITYSYMPÄRISTÖN PYSTYTTÄMINEN

Kehitysympäristönä toimii Ubuntu, sekä siihen asennettava NetBeans. Sivuston testaamista varten Ubuntuun tarvitsee asentaa Apache, MySQL ja PHP, sekä konfiguroida ne toimimaan sivuston kanssa paikallisesti.

Sivuston kykenee myös pystyttämään Windows-käyttöjärjestelmään ja käyttämään kehitystyökaluna NetBeans:ia. Ongelmana tässä vaihtoehdossa kuitenkin on, että kaikki ominaisuudet eivät toimi kuten palvelimilla, jotka ovat linux-koneita. Tätä vaihtoehtoa pystyy käyttämään, jos kehittää sivustoon täysin uutta osaa joka ei riipu muista toiminnallisuuksista.

2.1 Virtuaalikone

Ubuntu asennetaan normaalisti suoraan koneen kovalevylle, mutta se on käytännöllisempää asentaa virtuaalikoneeseen (joka asennetaan Windows:iin), koska Windows-käyttöjärjestelmän ohjelmistoja tarvitsee usein muiden työtehtävien yhteydessä. Ubuntun asentaminen virtuaalikoneeseen ei eroa normaalista asennuksesta, mutta virtuaalikone täytyy ensin asentaa ja käynnistää. Työn osalta virtuaalikoneen asennus oli vaihtoehtoinen.

Virtuaalikoneympäristönä käytetään Oracle:n VirtualBox ohjelmistoa, johon pystytetään uusi virtuaalikone. Koneen vähimmäisvaatimuksina on 512Mb RAM-muistia, sekä noin 8Gb kiintolevytilaa. On kuitenkin suositeltavaa asettaa suuremmat arvot, sillä virtuaalikoneessa oleva käyttöjärjestelmä toimii huomattavasti hitaammin kuin fyysisessä koneessa. Käyttöjärjestelmänä tulee toimimaan 32-bittinen Ubuntu, joten vaatimuksena on x86-prosessori. Videokiihdytys saadaan suoraan isäntäjärjestelmältä asentamalla VirtualBox:in ”guest additions”-lisäosa. Tämän saa asennettua suoraan virtuaalikoneen ylävalikosta (Devices->Install Guest Additions...), kun virtuaalikone on päällä.

Työssä käytetyssä virtuaalikoneessa on 768Mb RAM-muistia ja 30Gb kiintolevytilaa. Prosessoreista käytössä on vain yksi ja ”guest additions”-lisäosa on käytössä.

2.2 Ubuntu

Ubuntu on avoimen lähdekoodin Linux-käyttöjärjestelmä, joka pohjautuu Debianiin. Ubuntun asentaminen on helppoa ja sitä päivitetään puolen vuoden välein, mutta päivityksistä julkaistaan kahden vuoden välein pitkäaikaistuellinen versio.

Ubuntun saa ladattua suoraan sen kotisivuilta: http://www.ubuntu.com/download. Tässä työssä käytettiin 11.04 ”Natty Narwhal” desktop-versiota.

2.2.1 Asennus

Ubuntun asennus lähtee käyntiin lataamalla koneeseeen asennusmedia (esimerkiksi USB-tikkua, CD-levyä tai levykuvaa käyttäen) ja käynnistämällä koneen uudelleen käyttäen kyseistä asennusmediaa. Seuraamalla Ubuntun asennusohjeita saadaan käyttöjärjestelmä nopeasti käyttövalmiiksi. Ensimmäisenä asennuksen jälkeen on suositeltavaa päivittää Ubuntu ja sen sovellukset uusimpaan versioon Update Manager:ia apuna käyttäen.

2.2.2 LAMP-asennus ja konfigurointi

Sivustoja kehittäessä on hyvä testata omaa koodia paikallisesti, niin että ei tarvitse jakaa keskeneräisiä tuotoksiaan internetissä. Tätä varten pitää pystyttää palvelin toimimaan paikallisesti kehittäjän koneella. LAMP (Linux, Apache, MySQL, PHP) on yksi yleisimmistä palvelinpaketeista, joten se on täydellinen koodin kehittämiseen ja testaamiseen. Sitä käytetään myös tuotantoympäristöissä.

Ubuntun kehittäjät ovat tehneet LAMP:in asennuksen ja konfiguroinnin hyvin helpoksi. Syöttämällä Ubuntun Terminal-ohjelmaan komennon

```
sudo apt-get install lamp-server^
```

saa ladattua ja käynnistettyä valmiin asennuspaketin. APT-pakettimanageri näyttää komennolla asennettavat ohjelmat, painamalla *enter* hyväksytään asennukset. Tämän jälkeen ohjelma kysyy MySQL-palvelimen root-käyttäjän salasanaa ja salasanan vahvistusta. Salasanan vahvistamisen jälkeen pakettimanageri jatkaa loppujen ohjelmien asentamista. MySQL-palvelimen root-käyttäjän salasana on syytä muistaa, jotta sivuston pystytys ja kehitysvaiheessa pääsee muokkaamaan tietokantaa.

Ensimmäisenä asennuksen jälkeen kannattaa kokeilla toimiiko Apache-palvelin avaamalla selaimella osoite http://localhost/. Osoitteessa kuuluisi onnistuneen asennuksen jälkeen olla testisivu, jossa lukee ”It Works!”. Seuraavaksi kannattaa vielä kokeilla PHP:n asennuksen toimivuus luomalla Ubuntun /var/www/-kansioon ”testing.php”-tiedosto, jossa on seuraava PHP-koodirivi:

```
<?php phpinfo(); ?>
```

Apache-palvelin pitää uudelleenkäynnistää, jotta muutokset tulevat voimaan. Uudelleenkäynnistys tapahtuu syöttämällä seuraava komento terminaaliin:

```
sudo /etc/init.d/apache2 restart
```

Tämän jälkeen voi avata osoitteen http://localhost/testing.php, jossa pitäisi olla PHPinformation-sivulla listattuna PHP:n ominaisuuksia, PHP-asennuksen onnistumisen varmistamiseksi. Tältä sivulta kannattaa tarkistaa seuraavien PHP-lisäosien olemassaolo:

```
php5-openssl
php5-tokenizer
php5-ldap
php5-zlib
php5-curl
php5-sqlite
php5-gd
php5-xmlwriter
php5-bz2
php5-pear
```

```
php5-mysql
php5-mcrypt
php5-devel
php5-xmlreader
php5-gettext
php5-soap
php5-json
php5-ctype
php5-pgsql
php5-zip
php5-dom
php5-xsl
php5-hash
php5-mbstring
php5-pdo
php5-iconv
apache2-mod_php5
```

Kaikkia näitä ei välttämättä löydy, sillä PHP:ta ja lisäosia päivitetään ja integroidaan yhteen jatkuvasti. Puuttuvat osat saattavat löytyä Ubuntun Update Manager-ohjelmasta. Jos jokin lisäosa jää puuttumaan ja sivusto ei toimi odotetulla tavalla, kannattaa tarkistaa virhesanoma ja tämä lisäosalista uudestaan. Osa lisäosista ovat vaadittuja pelkästään työhön liittyvällä sivustolla, mutta niiden asentamisesta ei tule haittaakaan vaikka niitä ei käytä.

Seuraavaksi asennetaan Symfony-framework 1.3.6, joka on helpoin asentaa PEAR:in kautta syöttämällä Terminal:iin seuraavat komennot:

```
pear channel-discover pear.symfony-project.com
pear install symfony/symfony-1.3.6
```

Ensimmäinen rekisteröi symfonyn kanavan asentajaan ja toinen asentaa symfony-frameworkin käyttäen ensimmäisen komennon kanavaa. Tämän jälkeen symfony-framework on asennettuna järjestelmään ja valmis käytettäväksi.

MySQL ja PHP -asetuksia pitää vielä muokata paikallispalvelimen tarpeisiin sopiviksi. Tässä työssä sivusto vaatii suurien tiedostojen ja POST-komentojen sallimisen. Suuret MySQL tietokantakomennot ja paketit pitää myös sallia, jotta sivuston tietokanta saadaan päivitettyä. MySQL-konfiguraatiotiedostoa voi muokata tiedostopolussa ”/etc/my.cnf” ja siinä pitää muokata ”max_allowed_packet”-arvoa seuraavasti:

```
max_allowed_packet = 100M
```

Tämä arvo on hieman ylimitoitettu, mutta tarpeellinen jos halutaan tuoda tuotantotietokanta datoineen paikalliseen kehitykseen. PHP:n voi konfiguroida sallimaan suuret tiedostot ja komennot tiedostopolussa ”/etc/php5/apache2/php.ini” ja muokata sen arvoja seuraavasti:

```
memory_limit = 256M
upload_max_filesize = 10M
post_max_size = 10M
```

2.2.3 Sivuston pystyttäminen

NetBeans asennetaan Ubuntu Software Centre:n kautta. Asennuksen jälkeen avataan NetBeans ja asennetaan vielä PHP ja subversion liitännäiset valitsemalla ”Tools”-välilehdestä ”Plugins”. Tämän jälkeen voidaan yhdistää subversion-palvelimeen ja hakea sieltä kehitettävä sivusto käyttäen NetBeans:in SVN-työkalua.

Vielä ennen sivuston toimimista pitää muodostaa symboliset linkit projektikansion ja paikallispalvelimen välille, jotta sivusto saadaan käyttämään paikallispalvelimen symfony-asennusta. Tämä onnistuu navigoimalla Terminal-ohjelmalla projektikansioon ja ajamalla seuraavat komennot

```
sudo ln -s /usr/share/php/symfony/plugins/sfPropelPlugin/web sfPropelPlugin
sudo ln -s ../plugins/sfFormExtraPlugin/web sfFormExtraPlugin
sudo ln -s /usr/share/php/data/symfony/web/sf/ sf
```

Lisäksi pitää ladata sivustolla käytössä oleva tietokanta vielä MySQL-paikallispalvelimelle. Se onnistuu esimerkiksi PHPmyAdmin-ohjelmaa käyttäen, joka asennetaan LAMP:n mukana, tai suoraan ottamalla yhteyden Ubuntun Terminal:ista MySQL-palvelimeen.

Apache-palvelin pitää laittaa viittaamaan vielä esiasetetusta ”It works!”-testisivusta //localhost/-osoite projektikansioon. Tämä onnistuu muokkaamalla ”httpd.conf”-konfiguraatiotiedostoa polussa ”/etc/apache2/”. Tässä työssä käytetty tiedosto on liitteessä 2. Huomattavimmat kohdat tiedostossa ovat ”Virtualhost”, ”DocumentRoot” ja ”FollowSymLinks”. ”Virtualhost”-elementin attribuutti ”*:80” asettaa kyseisen virtualhost-palvelun saataville osoitteeseen ”//localhost/”. ”DocumentRoot”-kenttä määrittää sivuston juuren, joka tässä tapauksessa asetetaan osoittamaan projektikansioon. "FollowSymLinks"-valinta asettaa tämän sivuston seuraamaan aiemmin asetettuja symbolisia linkkejä.

Apache-palvelimen uudelleen käynnistäminen ajamalla Ubuntun Terminal:issa komento

```
sudo /etc/apache2/ restart
```

viimeistelee asennuksen ja konfiguroinnin. Tämän jälkeen paikallispalvelin on toiminnassa ja Symfony:n paikalliskehitysympäristö pystyssä osoitteessa http://localhost/frontend_dev.php.

3 SYMFONY MVC FRAMEWORK

Symfony on avoimen lähdekoodin ”PHP Web”-ohjelmistojen framework. Sensio Labs suunnitteli sen alunperin omaksi kehitystyökalukseen asiakkaiden nettisivujen luomista varten. Symfony julkaistiin vuonna 2005 MIT Open Source –lisenssillä ja nykyään se on yksi johtavista frameworkeista PHP-kehitykseen.

Sensio Labsin ja suuren yhteisön tukemana Symfony:stä on useita resursseja tarjolla: paljon dokumentaatiota, yhteisön tuki (postituslistat, IRC, jne), ammattilaistuki (konsultointi, opetus, jne). Symfonyä käyttävät useat, suuretkin, yritykset kuten Yahoo!, Dailymotion, Opensky.com, Exercise.com ja XMLdation.

Symfony on kirjoitettu kokonaisuudessaan PHP5:llä. Se on yhteensopiva useimpien tietokantojen kanssa (muunmuossa MySQL, PostgreSQL, Oracle ja Microsoft SQL Server).

3.1 Web-PHP Framework

Symfony on siis ”web PHP framework”, eli kirjasto yhteinäisiä luokkia kirjoitettuna PHP:lla. Se asettaa parhaat toimintatavat, jotka auttavat kehittämään helposti ylläpidettäviä ja turvallisia sivustoja.

Frameworkit, kuten Symfony, sulavoittavat ohjelmakehitystä automatisoimalla useita yleisiä toimintatapoja. Ne myös antavat rakennetta koodille, sekä auttavat kannustamaan kehittäjää luomaan parempaa, luettavampaa ja helpommin huollettavaa koodia. Suurimmaksi osaksi kuitenkin ne helpottavat ohjelmointia paketoimalla monimutkaisia operaatioita yksinkertaisiin komentoihin.

Symfony tarjoaa useita työkaluja ja luokkia jotka ovat suunniteltu monimutkaisten ohjelmistojen luomiseen. Lisäksi se erottaa säännöt (business rules), palvelin logiikan (server logic) ja näkymät (presentation views). Se myös automatisoi useita yleisiä toimintoja, jotta kehittäjät voivat keskittyä ohjelman yksityiskohtiin. Tämä siis tarkoittaa, että uutta ohjelmaa kehitettäessä ei tarvitse niin sanotusti keksiä pyörää uudelleen.

Symfonyn kansiorakenne jakautuu applikaatioihin (/apps/), konfiguraatiotiedostoihin (/config/), liitännäisiin (/plugins/), kirjastotiedostoihin (/lib/), testitapauksiin (/test/) ja sivustokansioon (/web/). Applikaatiot jakautuvat yleensä kahteen /backend/ ja /frontend/ kansioihin. Niissä ovat omat konfiguraatio-, moduuli- ja kirjastokansiot. Moduuleissa ovat aina käsittelijän toiminnot (/actions/) ja näkymän tiedostot (/templates/), lisäksi siellä voi olla esimerkiksi omia konfiguraatio- ja validointitiedostoja. Symfonyn kirjastotiedostoihin (/lib/model/, /lib/form/ ja /lib/filter/) luodaan ORM tietokantaluokat, tietokantaluokkia vastaavat kaaviot ja hakusuodattimet.

3.2 MVC

Symfony perustuu perinteiseen Model-View-Controller (MVC) -rakenteeseen. Model, malli, edustaa tietoa jonka perusteella ohjelma toimii – ohjelman logiikkaa. View, näkymä, renderöi mallin näkyväksi sivuksi käyttäjälle. Controller, käsittelijä, vastaanottaa käyttäjän komennot ja käsittelee mallia ja näkymää vastauksena niihin. Kappaleessa 4.5.1 Omat suodattimet on esitetty MVC:n eri osien koodia.

Malli edustaa sivuston tietoja, useimmiten tietokantaa. Symfonyssä tietokantaa mallintaa Propel tai Doctrine, molemmat ovat ORM käännöstekniikkoja. Tässä työssä on käytössä Propel. Propel luo tietokannan tauluja vastaavat PHP luokat ja tietokantakyselyt niiden tietojen täyttämiseen. Tätä on hyvin helppoa käyttää esimerkiksi käsittelijän puolella.

Käsittelijä ajetaan ennen näkymän renderöintiä ja siinä asetetaan näkymässä käytettäviin muuttujiin tarvittavat tiedot. Tämä mahdollistaa esimerkiksi mobiiliversion tekemisen sivustosta samoilla tiedoilla ja näkymillä pelkästään lisäämällä käsittelijään uuden ehdon. Tiedot haetaan mallista ja asetetaan muuttujiin näkymää varten.

Näkymä renderöi käsittelijässä asetetut muuttujat käyttäjälle nettisivuksi. Useimmiten monimutkaisissa sivustoissa käytetään näkymissä HTML, PHP ja CSS kieliä parhaan lopputuloksen saavuttamiseksi.

MVC-rakenne mahdollistaa erillisten applikaatioiden tekemisen yhdelle symfony sivustolle. Applikaatioilla ovat omat konfiguraatiotiedostot, eivätkä ne jaa tietoa

suoraan keskenään. Ainoat tiedot joita molemmat applikaatiot voivat käyttää ovat kirjastoluokkia projektikansion ”/lib/”-kansiossa. Tämä mahdollistaa ”frontend”-nimisen käyttäjä applikaation ja ”backend”-nimisen järjestelmänvalvoja (admin) applikaation luomisen.

Symfony jakautuu normaalisti kahteen osaan, frontend:iin ja backend:iin. Tämä tehdään pääasiassa siksi, että hallintapuolelle (backendiin) saa asetettua esimerkiksi varmemmat turvatarkastukset ja menetelmät kuin käyttäjäpuolelle. Frontend tarjoaa käyttäjälle sivuston ja backendistä pystyy muokkaamaan esimerkiksi frontendin käyttämien tietokantojen tietoja, eli käytännössä sivustoa suoraan. Tämä jako ei kuitenkaan ole välttämätön, mutta sitä käytetään jotta voidaan hallita koko sivustoa suoraan web-käyttöliittymästä.

4 LOGGING SYSTEM

Korvattava lokijärjestelmä tällä hetkellä tallentaa vain yksinkertaisia tapahtumia, kuten sisäänkirjautumisen ja XML-validointitapahtumien virheet. Nämä ovat tallennukset ovat kovakoodattuna MVC rakenteen sisään, eripuolille ohjelman moduuleja. Tällaista järjestelmää on vaikea ylläpitää ja päivittää, joten parempi vaihtoehto piti luoda.

Lokijärjestelmän tarkoituksena on saada kattavaa tietoa sivuston käytöstä. Pääasiassa tarpeena on seurata asiakkaiden toimintaa ja käyttäytymistä sivustolla. Kattavasta lokijärjestelmästä olisi myös hyötyä mahdollisten vikatilanteiden selvittämisessä.

Lokijärjestelmän pitää olla kattava, tietoturvallinen ja vakaa. Se ei saa häiritä sivuston toimintanopeutta huomattavasti tai häiritä muita sivuston moduuleja. Sen pitää myös olla helposti päivitettävä, ylläpidettävä ja hallittava. Lokitapahtumia pitää pystyä tarkastelemaan ja hallinnoimaan suoraan sivustolta.

4.1 Suunnittelu

Suunnittelu alkaa päämäärästä. Tavoitteena on kirjata käyttäjien yksittäisiä tapahtumia ja toimintoja sivustolta tietokantaan. Näitä tietoja pitää pystyä tarkastelemaan helposti suoraan sivustolta.

Jokaisesta käyttäjän toiminnasta pitää tallentaa seuraavat yleiset tiedot; yksilöintitunnus, aika, käyttäjä, yritys, käyttäjän yritys, tapahtumatyyppi, vakavuustaso, kanava, kuvaus ja kahden eri järjestelmävalvojatason määritykset. Lisäksi pitää tallentaa tapahtumatyypin mukaan lisätietoja. Tapahtumatyypit jakautuvat sivustolla seuraaviin osiin: validointi, simulointi, raportit (sisältävät validointitulokset), sivuston muokkaus ja sivun vaihto. Myös tapahtumia joista ei tallenneta lisätietoja saattaa esiintyä, esimerkiksi sivuston sisäinen virheilmoitus.

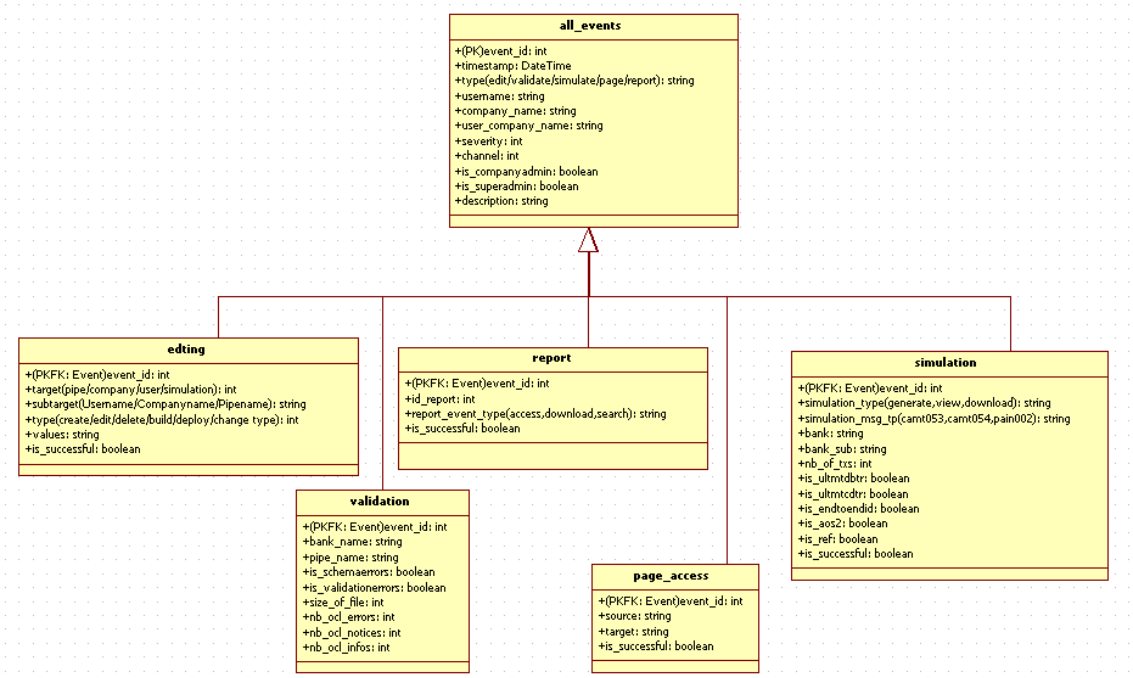
Validoinnista pitää yleisien tietojen lisäksi tallentaa pankin nimi, validointiputki, onko skeemaa vastaan tehdyssä validoinnissa virheitä, onko muita validointivirheitä, validoitavan tiedoston koko, sekä OCL-kielellä löytyneiden virheiden, huomautusten ja infojen lukumäärät. Simuloinnista pitää tallentaa sen tyyppi ja viestin tyyppi, pankin

nimi, pankin simulointiputki, maksutapahtumien lukumäärä, onko ”ultimate creditor”-tietoa käytetty, onko ”ultimate debtor”-tietoa käytetty, onko ”end to end ID”-tietoa käytetty, onko aos2-viestiätyyppiä käytetty, onko referessi-tietoa käytetty ja onnistuiko simulointi. Raporteista pitää yleisten tietojen lisäksi tallentaa raportin yksilöintitieto, raporttitapahtuman tyyppi (lataus, avaus vai haku) ja tapahtuman onnistuminen. Muokkauksista pitää tallentaa kohde, kohteen tunnus, muokkaustyyppi, arvot ja onnistuiko muokkaus. Sivun vaihdosta pitää tallentaa lähde, kohde ja onnistuminen.

Lokitapahtumien tarkastelu toteutetaan luomalla uusi sivu, jossa sivuston järjestelmänvalvojat pystyvät hallinnoimaan ja tarkastelemaan tapahtumia. Lokitapahtumat pitää pystyä myös lataamaan sivustolta suoraan, esimerkiksi Excel-tiedostona.

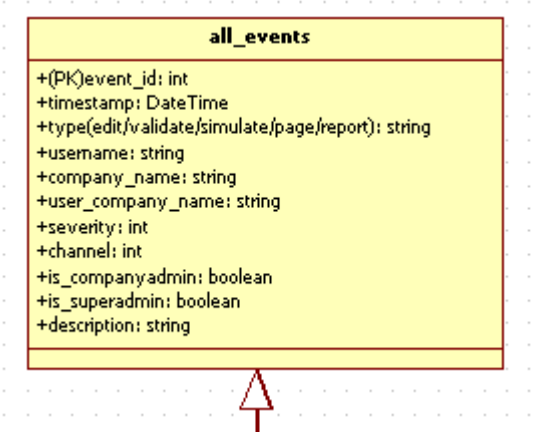
4.1.1 Tietokanta

Tietokanta suunnitellaan StarUML ohjelmaa hyödyntäen (KUVIO 1). Tarvitaan kuusi taulua, yksi johon tallennetaan jokaisen tapahtuman yleiset tiedot, sekä viisi taulua lisätietoja varten (validointi, simulointi, raportit, muokkaus, sivun vaihto). Jokaiseen lisätietotauluun tarvitaan tunniste jolla saadaan yksilöityä ja yhdistettyä yleiset tapahtumatiedot lisätietoihin. Tämä onnistuu helpoiten vierasavaimilla, jotka ovat otettu yleisistä tapahtumista.



KUVIO 1 - StarUML:llä suunniteltu tietokanta

Yleisien tapahtumien tietokantatauluun (KUVIO 2) tarvitaan seuraavat kentät: event_id (tapahtumatunniste), timestamp (aikaleima), type (tyyppi), username (käyttäjänimi), company_name (yrityksen nimi), user_company_name (käyttäjän yrityksen nimi), severity (vakavuus), channel (kanava), is_companyadmin (yritysadmin), is_superadmin (admin) ja description (kuvaus).

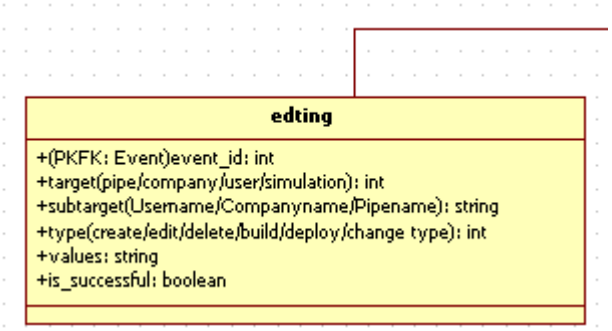


KUVIO 2 – Yleisten tapahtumien tietokantataulu

Tapahtumatunniste yksilöi tietokantatapahtumat. Se asetetaan automaattisesti inkrementoituvaksi kokonaisluvuksi jokaisen tapahtumalisäyksen yhteydessä. Aikaleima asetetaan suoraan tietokannan aikaleima-arvoksi, joten sen arvo otetaan jokaisen tapahtumalisäyksen yhteydessä automaattisesti ja siihen tulee sen hetkinen tietokannan päivämäärä ja aika. Tyyppi-kenttä osoittaa mahdollisen lisätietotaulun ja

tapahtuman tyypin, joka tallennetaan tekstinä. Käyttäjänimi on tapahtuman suorittaneen käyttäjänimi tekstinä. Yrityksen nimi –kenttään tallennetaan yrityksen nimi tekstinä, jonka kautta käyttäjä käyttää palvelua. Käyttäjän yrityksen nimi –kenttään tallennetaan nimensä mukaan yrityksen nimi, jossa käyttäjä on töissä, tekstinä. Vakavuus-kenttään tallennetaan numeroarvo, joka vastaa joko informaatio-, varoitus- tai virhetasoa. Yritysadmin ja admin arvot tallennetaan boolean-arvoina ja ne kertovat onko käyttäjä, joka tapahtuman on tehnyt, yrityksen valvoja tai järjestelmänvalvoja.

Muokkaus-tauluun (KUVIO 3) tarvitaan seuraavat kentät: event_id (yksilöintitieto), target (kohde), subtarget (alikohde), type (tyyppi), values (arvot) ja is_successful (onnistuminen).



KUVIO 3 – Muokkaus-lisätietotaulu

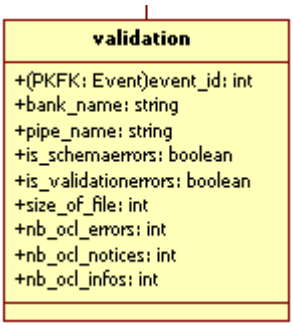
Tapahtumatunniste on vierasavain yleisien tapahtumien taulusta, jotta lisätiedot ja yleiset tiedot saadaan yksilöityä ja yhdistettyä. Kohde-kenttään tallennetaan muokkauksen kohde, eli muokataanko validointiputkea, yrityksen tietoja, käyttäjän tietoja vai simulaatiotapahtumaa. Kohde tallennetaan kokonaislukuna, joka vastaa järjestelmän kohteen yksilöintitietoa. Alikohde-kenttään tallennetaan kohteen tarkempi yksilöintitieto ja nimi tekstinä. Tyyppi-kenttään tallennetaan muokkauksen tyyppi, eli luominen, muokkaaminen, poistaminen ja niin edelleen. Arvot-kenttään tallennetaan muokatut arvot tekstinä. Onnistuminen tallennetaan boolean-arvona sen kenttään.

Sivunvaihto-lisätietotauluun (KUVIO 4) tarvitaan seuraavat kentät: event_id (tapahtumatunniste), source (lähtösivu), target (kohdesivu) ja is_successful (onnistuminen).

KUVIO 4 – Sivunvaihto-lisätietotaulu

Tapahtumatunniste on jälleen vierasavain yleisien tapahtumien taulusta, jotta lisätiedot ja yleiset tiedot saadaan yksilöityä ja yhdistettyä. Lähtösivu on sivu, jolta käyttäjä on tullut ja se esitetään tekstinä. Kohdesivu on sivu, jolle käyttäjä on vaihtamassa ja sekin esitetään tekstinä. Onnistuminen merkataan boolean-arvona jälleen.

Validointi-lisätietotauluun (KUVIO 5) tarvitaan seuraavat kentät: event_id (tapahtumatunniste), bank_name (pankin nimi), pipe_name (putken nimi), is_schemaerrors (skeema-virheet), is_validationerrors (validointivirheet), size_of_file (tiedostokoko), nb_ocl_errors (OCL-virheet), nb_ocl_notices (OCL-huomautukset), nb_ocl_infos (OCL-tiedotukset).



KUVIO 6 – Validointi-lisätietotaulu

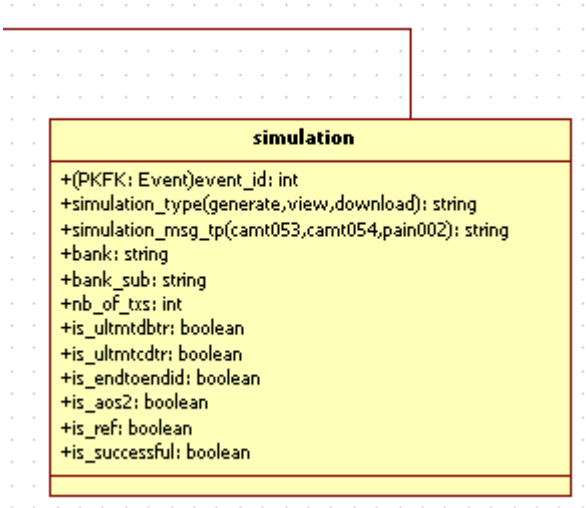
Tapahtumatunniste otetaan yleisien tapahtumatietojen taulusta vierasavaimena jälleen. Pankin nimi -kenttään tallennetaan validointiputken tarjoava pankki tekstinä. Putken nimi –kenttään tallennetaan validointiputken tunniste ja nimi tekstinä. Skeema-virheet –kenttään tallennetaan boolean-arvona löytyikö skeemaa vastaavia virheitä validoinnista vai ei. Validointivirheet-kenttään tallennetaan boolean-arvona löytyikö mitään virheitä validoinnista. Tiedostokoko-kenttään tallennetaan validoitavan tiedoston koko kokonaislukuna. OCL –virheiden, -huomautuksien ja –tiedotuksien lukumäärä tallennetaan kokonaislukuna niitä vastaaviin kenttiin.

Raportit-lisätietotauluun (KUVIO 7) tarvitaan seuraavat kentät: event_id (tapahtumatunniste), id_report (raportin tunniste), report_event_type (raportin tapahtumatyyppi), is_successful (onnistuminen).

KUVIO 7 – Raportit-lisätietotaulu

Tapahtumatunniste on jälleen vierasavain yleisien tapahtumien taulusta. Raportin tunniste -kenttään tallennetaan käytetyn raportin tunniste. Raportin tapahtumatyyppi – kenttään tallennetaan tekstinä tieto mitä raportille tehtiin. Onnistuminen on tässäkin taulussa boolean-arvona.

Simulointi-lisätietotauluun (KUVIO 8) tarvitaan seuraavat kentät: event_id (tapahtumatunniste), simulation_type (simulaatiotyyppi), simulation_msg_tp (simulaatioviestin tyyppi), bank (pankki), bank_sub (pankin simulointiputki), nb_of_txs (tapahtumien määrä), is_ultmtdbtr (ultimate-debtor -tiedon käyttö), is_ultmtcdtr (ultimate-creditor -tiedon käyttö), is_endtoendid (end to end id -tiedon käyttö), is_aos2 (AOS2 -viestityypin käyttö), is_ref (reference id -tiedon käyttö) ja is_successful (onnistuminen).



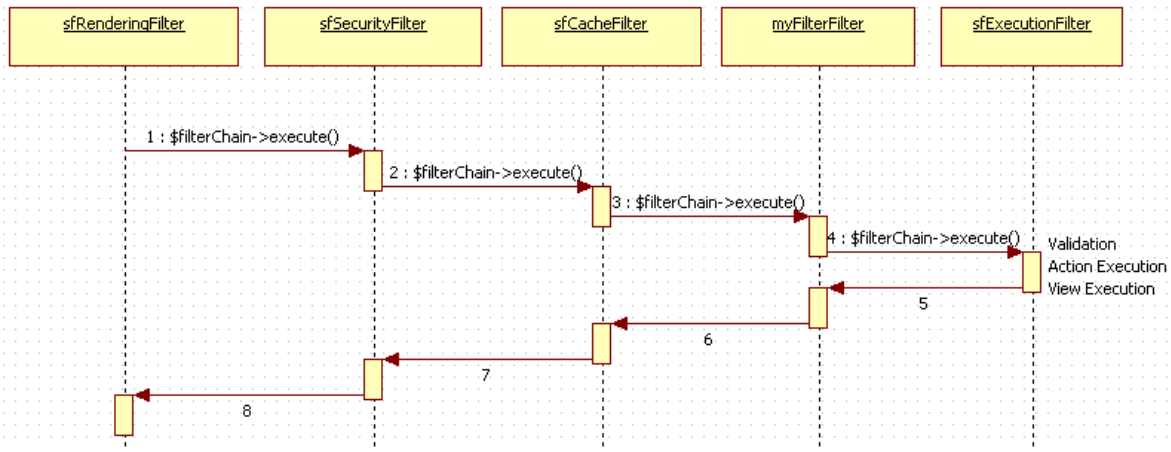
KUVIO 8 – Simulaion-lisätietotaulu

Tapahtumatunnisteena käytetään vierasavainta yleisistä tapahtumatiedoista. Simulaatiotyyppi-kenttään tallennetaan tekstinä käyttäjän simulointitapahtuman käyttö (luominen, tarkastelu ja lataaminen). Simulaatioviestin tyyppi -kenttään tallenneetaan mitä viestityyppiä simulointiin. Pankki-kenttään tallennetaan simuloitu pankki tekstinä. Pankin simulointiputki –kenttään tallennetaan tekstinä käytetty simulointiputki. Tapahtumien määrä -kenttään tallennetaan simuloitavan tiedoston tapahtumien lukumäärä. Eri tietojen ja viestityyppien käyttö –kenttiin tallennetaan boolean arvoina onko kyseistä tietoa tai viestityyppiä käytetty. Onnistuminen-kenttään tallennetaan boolean arvona simuloinnin onnistuminen.

4.1.2 Symfonyn käyttö

Tutkimuksen tuloksena lokijärjestelmälle tarjoutui kaksi eri toteutusvaihtoehtoa. Ensimmäinen vaihtoehto oli tehdä jokaiseen moduuliin erilliset preExecute() ja postExecute() -funktiot, joiden kautta voisi käyttää yleistä kirjastoluokkaa joka kirjaisi tietokantaan tapahtumat. Nämä funktiot ajetaan Symfonyn järjestelmässä aina kyseisen moduulin toiminnan ennen ja jälkeen. Tämä vaihtoehto kuitenkin hajauttaisi koodin useaan paikkaan, eikä sitä pystyisi hallitsemaan hyvin. Toinen vaihtoehto, joka osoittautui paremmaksi, oli luoda uusi filtteri Symfonyn järjestelmään.

Filtterit ovat rakennettu symfonyyn kiinteäksi osaksi järjestelmää. Nämä tekevät muunmuossa seuraavat toiminnot: käyttöoikeuksien tarkistuksen, lomakkeiden oikeellisuuden tarkistuksen (vaaditut kentät täytettynä), toimintojen ajamisen ja näkymän näyttämisen käyttäjälle. Filttereitä voi lisätä tähän ketjuun, joka ajetaan joka toiminnon yhteydessä. Jokainen filtteri (paitsi sfExecutionFilter joka ajaa toiminnon) ajetaan kahteen kertaan ennen käyttäjälle vastauksen palauttamista, kerran ennen näkymän ja toimintojen tekemistä ja kerran näiden jälkeen (KUVIO 9).



KUVIO 9 - Symfonyn filttereiden sekvenssikaavio.

Filttereillä pystyy siis tarkastelemaan käyttäjien toimintojen tiedonvälitystä ja se soveltuu täydellisesti lokijärjestelmän luomiseen. Käyttämän hakemia tietoja pystyy tallentamaan ensimmäisessä filtterien ajossa ja järjestelmän vastausta pystyy seuraamaan toisessa ajossa. Osa lokikirjauksista pitänee kirjata heti kun käyttäjä tekee toiminnon ja osa toimintojen (esimerkiksi XML-tiedoston validoinnin) jälkeen, että saadaan mahdollisimman kattava yhteenveto tapahtumasta.

Ainoana ongelmana filttereiden käytössä on se, että sivustolla on kaksi eri applikaatiota. Backend applikaatio on hallinnollinen puoli, josta pystyy mm. muokkaamaan sivuston asetuksia ja tietokantaa. Frontend applikaatio on käyttöliittymä puoli, josta validointijärjestelmää käytetään. Applikaatiot ovat erillisiä, joten niillä on omat filtterit ja konfiguraatiotiedostot. Molempiin pitää siis luoda oma filtteri vaikka rakenne ja toiminta olisi samantapainen. Tiedot joita applikaatiot käsittelevät ovat kuitenkin eroavia, tämän johdosta kahden eri filtterin luominen ei lisää työmäärää huomattavasti.

4.1.3 Tietokannan apuluokka

Tietokannan käyttöä varten luodaan ”apuluokka” – uusi PHP-luokka, joka tarjoaa methodeja tietokantaan tallennusta varten. Tämä helpottaa huomattavasti hallittavuutta yhtenäistämällä kaikki uuden tietokannan käyttötapahtumat samaan paikkaan. Apuluokka luodaan Symfonyn kirjastoksi, jotta molemmat applikaatiot kykenevät käyttämään tätä.

Apuluokkaan luodaan jokaista tietokannan taulua vastaavat staattiset funktiot, jotka vastaanottavat tallennettavat tiedot ja tallentavat nämä tietokantaan. Lisäksi tarvitaan tietokannan muokkausta varten, sekä tiedon hakemista varten funktiot. Tähän luokkaan luodaan myös tapahtumien tallennuksen hallintaa varten boolean arvot, jossa ’false’ arvolla tietyn tietokantatapahtuman tallennus poistetaan käytöstä ja ’true’ arvolla otetaan käyttöön. Tapahtuman muokkauksia tarvitaan muunmuossa onnistumista kuvaavan boolean arvon muuttamisessa. ”Tapahtuman muokkaus”-funktion tarkempi toimintatapa määrittyy toteutuksessa, jolloin sivuston sisäinen logiikka tarkentuu tapahtumien ajojärjestyksen osalta. Tietokantaan tallennettavat arvot lisätään julkisiin PHP:n ”array”-listoihin. Array:n ”key”-arvo vastaa tietokantaan tallennettavaa tietoa ja ”value”-arvo vastaa koodissa PHP:n ”array_search”-funktio:n etsintäarvona tietokanta arvon etsintään, jota käytetään muunmuossa admin modulen filttereissä ja applikaatioiden filttereissä.

4.1.4 Käyttöliittymä

Useiden muotoiluvaihtoehtojen (LIITE 1) jälkeen parhaaksi ratkaisuksi osoittautui luoda kaksi sivua. Toinen sivu on pääsivu, jolla voi tarkastella tapahtumien yleisiä tietoja listasta ja käsitellä niitä. Toisella sivulla pystyy katsomaan yksittäisiä tapahtumia tarkemmin.

Pääsivu jakautuu neljään osaan; navigaatiopaneeli, haku-filtterit, tapahtumalista ja tapahtumalistan toiminnot. Navigaatiopaneeli on sivuston valmis osa pääsivun ylimpänä, joka otetaan käyttöön lisäämällä pääsivulle Symfonyn PHP-komento:

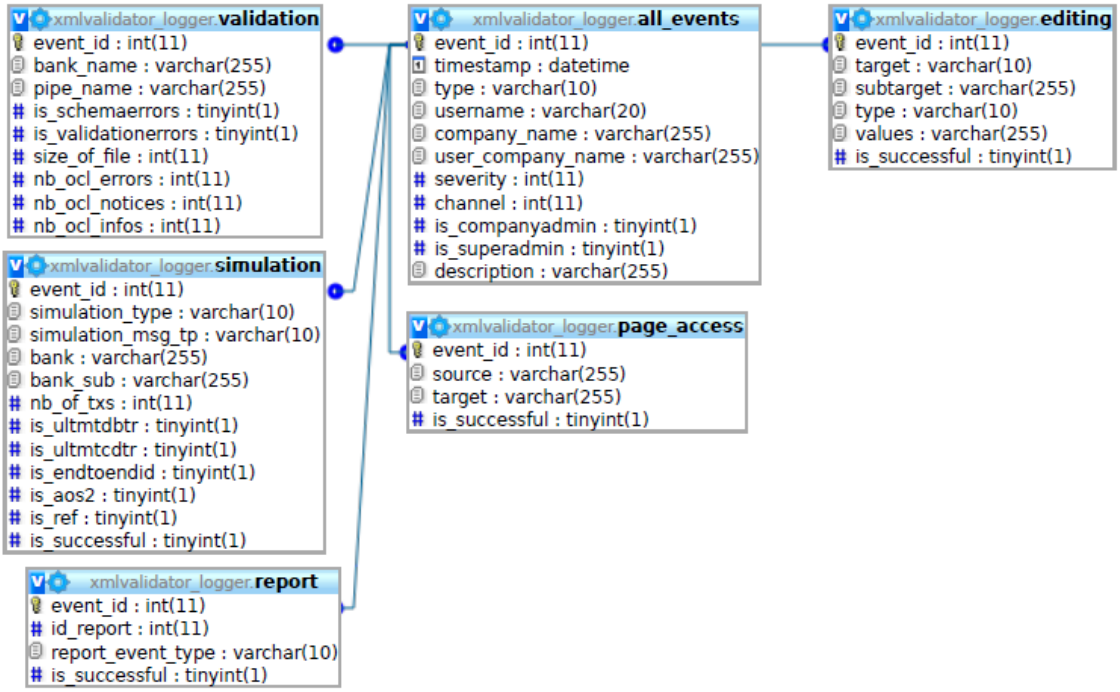
```
<?php include_partial('[sivu]') ?>
```

, jossa *[sivu]* on Symfony:ssä alaviivalla alkava PHP-sivu. Alaviivalla määritellään sivun pienemmät osat - ”partialit”, jotka suunnitellaan käytettäviksi useissa paikoissa. Haku-filtterit luodaan myöskin partialiksi, jossa Symfonyn PHP-komennoilla luodaan hakukaavakkeet. Tapahtumalista on pääsivun tärkein yksittäinen osa, siihen kerätään lokitapahtumien yleiset tiedot esitettäväksi käyttäjälle. Tapahtumalistasta pitää päästä tarkastelemaan tapahtumaa tarkemmin suoraan klikkaamalla. Tapahtumalistan toiminnot sijoitetaan listan alaosaan. Toimintoihin kuuluu tapahtumien poistaminen,

lataaminen ja hakutulosten järjestäminen. Lataaminen toteutetaan PHPExcel-kirjastoa apuna käyttäen, joten tapahtumat ladataan sivustolta Excel-tiedostona.

4.2 Tietokanta

Tietokanta (KUVIO 10) luodaan PHPmyAdmin ohjelmalla. PHPmyAdmin:illa saa kätevästi otettua yhteyden LAMP:n MySQL asennukseen osoitteessa http://localhost/phpmyadmin ja luotua uuden tietokannan, jossa on lokikirjaus tapahtumien vaatimat taulut. Tietokantaan pitää myös lisätä käyttäjäoikeudet, että sivusto pääsee käsittelemään sitä. Käyttäjäoikeuksia tarvitaan kun symfonyyn konfiguroidaan uusi tietokanta ja tälle kannattaa asettaa täydet oikeudet.



KUVIO 10 – Tietokanta myPhpAdmin:issa

Kun tietokanta on pystytetty, se pitää liittää Symfonyyn ja PHP:ssa käytettäväksi. Ennen sen lisäämistä kannattaa tyhjentää Symfonyn cache-kansio ajamalla Ubuntun Terminal-ohjelmalla projektikansiossa komento

symfony cc

Kaikki ”symfony”-alkuiset komennot pitää ajaa sen projektin kansiossa, johon haluaa tehdä symfony muutokset.

Seuraavaksi tietokanta saadaan kiinnitettyä sivustoon Symfonyn automaattisella komennolla

```
./symfony configure:database --name=logger --class=sfPropelDatabase
"mysql:host=localhost;dbname=xmlvalidator_logger" user pass
```

, jossa lipulla ”name” annetaan Symfonyssä käytettävä tietokannan nimi, ”class”-lippu määrittää ORM-luokan (Symfonyssa propel tai doctrine) ja lopuksi annetaan komennolle tietokannan osoite, nimi ja käyttäjätunnus sekä salasana. Tietokantaa käytetään paikallisesti, joten tietokannan osoitteessa localhost on isäntä. Tietokannan osoitteessa ”dbname”-tiedon pitää vastata tietokannan luomisessa käytettyä nimeä ja käyttäjätunnusten tietokannan käyttäjää.

Tämän jälkeen voidaan ajaa Terminal:issa Symfonyn komento

```
symfony propel-build-schema
```

, joka luo tietokannan konfiguraatiotiedoston ”schema.yml”:n. Tässä tiedostossa määritetään tietokantojen yhteysosoitteet, käyttäjätunnukset ja yhteystyypit. Tämän jälkeen ajetaan Symfony:n komento

```
symfony propel:build-all
```

, joka ajaa käytännössä seuraavat yksittäiset komennot ”propel-build-model”, ”propel-build-sql” ja ”propel-insert-sql”. Komento ”propel-build-model” luo tietokantatauluja vastaavan yml-mallin ja tietokantatauluja vastaavat propel-kirjastoluokat. ”propel-build-sql”-komento luo SQL-koodin ”schema.yml”-konfiguraatiotiedoston perusteella. Tällä SQL koodilla pitäisi pystyä luomaan myöhemmin tietokannan uudestaan (esimerkiksi kehitysympäristöstä siirryttäessä tuotantoon). ”propel-insert-sql”-komento työntää SQL koodin tietokantaan. Tämä varmistaa, että tietokanta ja luodut schemat vastaavat toisiaan. Komennolla

symfony propel:build-forms

Symfony luo automaattisesti ”schema.yml”-konfiguraatiotiedoston perusteella yhden luokan taulua kohden, joka tarjoaa lomakkeet taulun käyttöä varten. Lomakkeissa on validaattorit (esimerkiksi jos haluaisi lisätä lomakkeella tietoa tietokantaan, lomake tarkistaa datan oikeellisuuden ettei tekstiä mene numero kenttään ja niin edelleen).

Viimeisenä propel-komentona pitää ajaa

symfony propel:build-filters

komento, joka luo luokat suodattimille etsintätoimintoa varten, samaan tapaan kuin edellämainittu ”build-forms”-komento luo lomakkeet muokkaamista varten. Näiden toimenpiteiden jälkeen on hyvä taas puhdistaa Symfonyn cache-kansio komennolla

symfony cc

jonka jälkeen uusi tietokanta pitäisi olla liitettynä Symfonyn projektiin.

Lisäksi voi puhdistaa scriptillä mahdollisia ”build”-komentojen luomia ylimääräisiä tiedostoja vanhoista tietokannoista ajamalla erillinen ”build-schema.sh”, johon on kerätty muunmuossa valmiita toimenpiteitä puhdistamaan sivustoa. Tämä scripti on sivustokohtainen ja se pitää tarvittaessa luoda itse.

4.3 Tietokannan apuluokka

Tietokannan apuluokassa on yleisinä staattisina muuttujina määriteltynä tietokantaan tallennettavien tapahtumien käyttöönoton määrittäviä boolean arvoja. Staattisina muuttujina on määritelty myös tietokantaan tallennettavat arvot ja niiden selitysten lyhenteet, jotka ovat PHP:n ”array”-muuttujissa ”key-value”-pareina. Muuttujat ovat jaettu tietokannan taulujen kenttien mukaan.

Jokaista tietokannan taulua varten on luotu oma funktio, joka vastaanottaa tietokantaan tallennettavat arvot. Tietokannasta tietojen hakua, sekä tietokanta tapahtuman

muokkausta varten on myös luotu funktiot. Pääkohtana jokaisessa tietokantaan tallentavassa funktiossa ovat tapahtuman, esimerkiksi yleisten tapahtumatietojen (”all_events”-taulun) luominen ja tallentaminen:

```
$newEvent = new AllEvents();
...
$newEvent->save();
```

, jossa uuteen muuttujaan sijoitetaan Symfonyn tietokannan yhdistämisessä luodun kirjaston uusi tietokanta taulu –luokka. Jokaisessa tietokannan taulusta mallinnetussa luokassa on muunmuossa taulua vastaavat arvon asetus funktiot, sekä tallennusfunktio joka luo tietokantayhteyden ja tallentaa tiedot. Yleisien tapahtumatietojen tallennuksen jälkeen haetaan tapahtuman yksilöintitieto ja palautetaan tämä myöhempää käyttöä varten. Tietojen hakua varten tehty funktio vastaanottaa halutun tapahtuman yksilöintitiedon ja palauttaa PHP array:na tapahtuman yleiset tiedot, sekä mahdolliset lisätiedot lisätietotaulusta. Tapahtuman muokkaus funktio vastaanottaa tapahtuman yksilöintitiedon, tapahtumatyypin, muokattavan kentän, uuden arvon ja mahdollisen lisäviestin (käytetään muunmuossa virheen lisäämisessä kuvaus-kenttään).

4.4 Filters

Symfonyn jokaisella applikaatioilla on omat ympäristöt ja ne toimivat periaatteessa toisistaan riippumatta. Tämän vuoksi pitää luoda kaksi eri Symfonyn filtteriä, yksi backend-käyttöliittymään ja toinen frontend-käyttöliittymään.

Molemmat filtterit käyttävät tietokannan apuluokan tarjoamia funktioita tietokantaan tallentamisessa. Osa tiedoista tallennettaessa otetaan suoraan tietokannan apuluokan muuttujalistoista, joka helpottaa osaltaan hallittavuutta. Apuluokassa on myös muuttujat, joilla pystyy ottamaan pois käytöstä filtterit kokonaan, tai poistamaan osan lokiin kirjattavista tapahtumista.

Filtterit luodaan osaksi Symfonyn filtterijärjestelmää lisäämällä uusi luokka halutun applikaation kirjastokansioon. Kirjastoluokka pitää lisätä myös ”filters.yml”-konfiguraatiotiedostoon. Filtterin pitää jatkaa Symfonyn pohjaluokkaa ”sfFilter”. Oman

toiminnallisuuden lisäämiseksi filtteriin pitää lisätä ”execute”-funktio, joka vastaanottaa yhden parametrin. Tämän parametrin funktiota ”execute” pitää kutsua, jotta muut filtterit ajetaan. Muiden filtterien ajon jälkeen sivuston toiminta on tehty ja sen tietoja pystyy tarkastelemaan. Näillä tiedoilla saadaan muunmuossa selvitettyä tapahtuman onnistuminen.

4.4.1 Backend

Backend-käyttöliittymässä ei ole varsinaista validointipalvelua, simulointipalvelua tai raporttien katsomista, joten ainoat tallennukset tulevat tietokannan ”all_events” taulun lisäksi ”editing” ja ”page_access” tauluihin. ”editing” tauluun tallennetaan tarkasti jokaisen validointiputken, käyttäjän, yrityksen ja asetusten muokkaukset. ”page_access” tauluun kirjataan kaikki sivunvaihdot, kuten frontend-applikaatiossakin tullaan tekemään.

Filtterin alussa tarkistetaan onko se käytössä tietokannan apuluokan muuttujista. Toiminta perustuu sivuston sivunvaihdon tunnistamiseen ja pyynnön tietojen tulkintaan. Symfonyssä saa otettua käyttäjän pyynnön ja palvelimen vastauksen sen hetkisestä tapahtumasta filtterissä kutsumalla kontekstiobjektia:

```
$this->getContext()
```

, josta saadaan ”getRequest()” ja ”getResponse()” funktioilla pyyntö ja vastaus objektit. Näistä objekteista pystyy selvittämään sivuston osan, jota käytetään (moduulin) ja parametrit joita sivusto ja käyttäjä lähettävät toisilleen – niiden keskustelun. Parametreista saa tulkittua onko tiedostoja lähetetty ja mitä sivustolla on tehty.

Filtterin alussa tallennetaan muuttujiin yleisiä tietoja kuten käyttäjä, sivu jota haetaan, sivu jolta tultiin, modulen nimi, toiminto (action) joka suoritettiin ja onko sivunvaihto POST-komennolla suoritettu. Tämän jälkeen on käytännössä filtterin runko: switch-case-rakenne jolla vaihdetaan moduulin mukaan toimintoa. Moduulin löydyttyä, tarkistetaan tehty toiminto ja tallennetaan sen perusteella tiedot tietokantaan.

Esimerkiksi jos käyttäjä poistaa yrityksen palvelusta, joutuu hän ensin menemään ”companies”-moduuliin ja siellä käyttämään ”delete”-toimintoa (actionia). Nämä pystyy

tarkastamaan filtterissä, kun toimintoa suoritetaan. Tämä tapahtuma siis tallentaisi ”editing” tauluun ”target”-kenttään ”company”, ”subtarget”-kenttään yrityksen yksilöintitiedon, tyypiksi ”delete” - poistamisen, ”values”-kenttä jäisi tyhjäksi, koska koko yritys poistetaan ja ”is_successful”-kenttä olisi ”true” toiminnon onnistuessa. Ennen ”editing” tauluun tallentamista tallennetaan alussa muuttujiin asetetut yleiset tiedot ”all_events” tauluun (yleisiin tietoihin kuuluu muunmuossa käyttäjänimi ja aikaleima). Apuluokka palauttaa ”all_events” taulun tallennusfunktiolla tapahtuman pääavaimen (ID:n) jota käytetään ”editing” tauluun tallentamisessa vierasavaimena ja pääavaimena.

Päärungon jälkeen on muutamia tarkistuksia mahdollisia muokkauksia varten, sekä tallennus ”page_access” tauluun, joka määritetään tallennettavaksi jollei aiemmin ole asetettu ”$event_id” muuttujaa, joka asetetaan aina kun ”all_events” tauluun tallennetaan. Tässä välissä filterchainia jatketaan eteenpäin, ajetaan käyttäjän pyytämät toiminnot ja luodaan näkymä palautettavaksi käyttäjälle. Toimintojen ajamisen jälkeen saadaan lisätietoa tapahtumista ja tämän perusteella voi olla aiheellista muokata juuri tallennettua tapahtumaa. Muokkaukset painottuvat pääasiassa tapahtuman onnistumisen vaihtamiseen. Alussa kun tapahtuma tallennetaan ennen toiminnon ajamista, tallennetaan toiminta ja sen onnistuminen, vaikka ei tiedetä vielä lopputulosta. Toiminnon suorittamisen jälkeen tarkistetaan onnistuminen ja tarvittaessa muokataan tietokanta tapahtumaa joko onnistuneeksi tai epäonnistuneeksi.

4.4.2 Frontend

Frontend-filtterin rakenne on hyvin samantapainen kuin backend-filtterin. Tässä kuitenkin joutuu tekemään tarkempia tarkistuksia ja muokkauksia sivuston rakenteesta johtuen. Frontendissä tallennetaan tapahtumia jokaiseen tietokannan tauluun. ”page_access” tauluun tallennetaan kaikki sivunvaihdot, paitsi niistä sivunvaihdoista kun tallennetaan muita tietoja, kuten validointi tai simulointi. ”validation” tauluun tallennetaan käyttäjien XML-validoinnit ja niiden tulokset. ”simulation” tauluun tallennetaan käyttäjien simulaattorin käyttö ja jokaisen simulointityypin tulokset. ”report” tauluun tallennetaan käyttäjien raporttien tarkastelu, HTML ja PDF lataus, sekä niiden etsiminen. ”editing” tauluun tallennetaan pelkästään käyttäjien omien tietojen muokkauksia. ”editing” tauluun piti myös lisätä yritysadminin käyttäjien ja yrityksen

tietojen muokkaus, mutta yritysadmin moduuli ei toimi kehitysympäristössä joten sen lokikirjaukset jouduttiin jättämään väliin. Muutamia tapahtumia, kuten tukipyyntö ja erilliset virheet, tallennetaan pelkästään ”all_events” tauluun.

Tämänkin filtterin alussa tallennetaan yleisiä tietoja ”all_events” taulua, sekä myöhempää käyttöä varten. Tämän jälkeen tarkistetaan muutamia mahdollisesti epäonnistuvia tapahtumia. Nämä tapahtumat saattavat tuottaa virheen toimintoa suorittaessa, joten ne pitää tallentaa ennen tapahtuman suoritusta epäonnistuneiksi. Seuraavaksi jatketaan kutsutaan muita filttereitä, luodaan näkymä ja toteutetaan toiminnot. Sen jälkeen tulee tämän filtterin päärunko, joka on toteutettu samalla tavalla kun backend-filtterin päärunko. Switch-case-rakenteella vaihdetaan moduulin mukaan tallennustapahtumaa ja switch-casen sisällä if-ehtolauseilla määritellään tarkemmat tapahtuman tallennustiedot. Tässäkin filtterissä tallennetaan ensin ”all_events” tauluun tapahtuma ja käytetään sen tapahtuman ID-tietoa vierasavaimena toiseen tauluun tallennettaessa pääavainta.

Joitakin tapahtumia muokataan päärungon jälkeen, jos on tarvetta. Näitä tapahtumia ovat esimerkiksi simulaation onnistumisen muokkaus. Tämä on sama toimintatapa kuin backend-filtterissä.

4.5 Backend admin module

Symfonyn automaattinen komento

> symfony propel:generate-admin *[applikaatio]* *[taulu]* --module=*[moduuli]*

, jossa *[applikaatio]* on haluttu symfonyn appikaatio, *[taulu]* tietokantataulun PHP-nimi ”schema.yml”-tiedostosta ja *[moduuli]* moduulin nimi. Komento luo automaattisesti admin-modulen, joka on yhdistetty yhteen tietokannan tauluun. Admin-modulessa on valmiina suodattimet hakua varten, listaobjekti jossa tietokannan rivit näkyvät, sekä tietokanta tapahtuman luominen, muokkaus ja poisto. Listaobjekti käsittelee ”generate-admin” komennossa määritellyn tietokannan taulun tapahtumat automaattisesti ”generator.yml” konfigurointitiedoston määrittämään muotoon (sivumäärä, tapahtumien

määrä per sivu, ja niin edelleen). Konfiguraatiotiedostossa pystyy hallitsemaan myös sivun ulkoasua ja objektien, kuten hakusuodattimet ja toiminnot, näkyvyyttä.

Tähän moduleen lisättiin vielä omat hakusuodattimet (filtterit), jotta pystyy etsimään lisätietotaulujen tietojen perusteella tapahtumia. Tapahtumien tarkastelu muokattiin Symfony-komennolla valmiiksi luodusta ”edit”-sivusta. Excel-lataus toteutettiin linkkinä toimintoon, joka lisää vastausobjektiin excel-tiedoston ladattavaksi. Excel-tiedostoon kerätään valittujen tapahtumien yksilöintitietojen perusteella tietokannasta tiedot, jotka järjestetään sivuittain. Moduuli pitää myös lisätä Symfonyn yleiseen navigaatiosivuun linkiksi, että siihen pääsee käsiksi. Linkki lisättiin backendin ”general”-moduulin välilehtilinkiksi.

4.5.1 Omat suodattimet

Omat suodattimet pitää yhdistää tietokantaan niihin tauluihin, joita Symfonyn ”generate-admin”-komento ei yhdistänyt valmiiksi moduuliin. Nämä ovat siis kaikki lisätietotaulut.

Näkymässä omat suodattimet ovat toteutettu Symfonyn PHP-methodeja käyttäen ja ne ovat asetettu paikalleen HTML:llä ja CSS:llä. Symfonyn methodeilla pystyy muunmuossa luomaan lomakkeen ja kentät siihen. Tässä esimerkiksi luodaan lomake joka kutsuu ”logging” moduulin executeSearchEvents($request) funktiota ja siihen aikaleima-kenttä joka saa tietonsa Symfonyn kalenteri widgetistä:

```
<form action="<?php echo url_for('logging/searchEvents') ?>"
method="post" name="filtersForm">
   <div style="float: left;">
      <div id="search_allevents" style="float: left;">
            <h4>All Events<?php echo tooltip_image('All Events contains
the common filters for all event types. Each event has an unique all events
instance.') ?></h4>
            <?php echo
input_hidden_tag('search_allevents[is_enabled_check]',
$search_allevents['is_enabled_check']); ?>
            <div style="float: left; width: 50%;">
```

```
                    <?php echo label_for('search_allevents[id]',"ID:") ?>
                    <?php echo input_tag('search_allevents[id]',
$search_allevents['id'], 'rich=false') ?>
                </div>
                <div style="float: left; width: 50%;">
                    <div style="float: left;"><?php echo
label_for('search_allevents[timestamp][start]',"Timestamp start:")
?><?php echo tooltip_image('Timestamp format is MM/DD/YYYY.')
?></div>
                    <div style="float: left;"><?php echo
input_date_tag('search_allevents[timestamp][start]', '', array('rich'=>true))
?></div>
                </div>
```

PHP-koodissa suurin osa käytetyistä funktioista tulee valmiina sisäänrakennettuna Symfonyyn. Esimerkiksi kaiuttamalla (echo) ”input_tag($nimi, $vakioarvo, $lisävalinnat)” funktion saa tulostettua HTML muotoisen tekstikentän lomakkeeseen.

”Filter”-napista hakuehdot lähetetään mallille. Jokaisen tietokantataulun jokaista kenttää varten pitää lisätä hakusuodattimet. Hakusuodattimien tiedot määräytyvät tietokantataulujen kenttien mukaan. Osa hakuehdoista haetaan suoraan pudotuslistoiksi tietokannan apuluokasta.

MVC:n käsittelijän puolella omien suodattimien lomakkeen lähettämien hakuehtojen mukaan tehdään kysely mallin (AllEventsPeer) kautta tietokannalle:

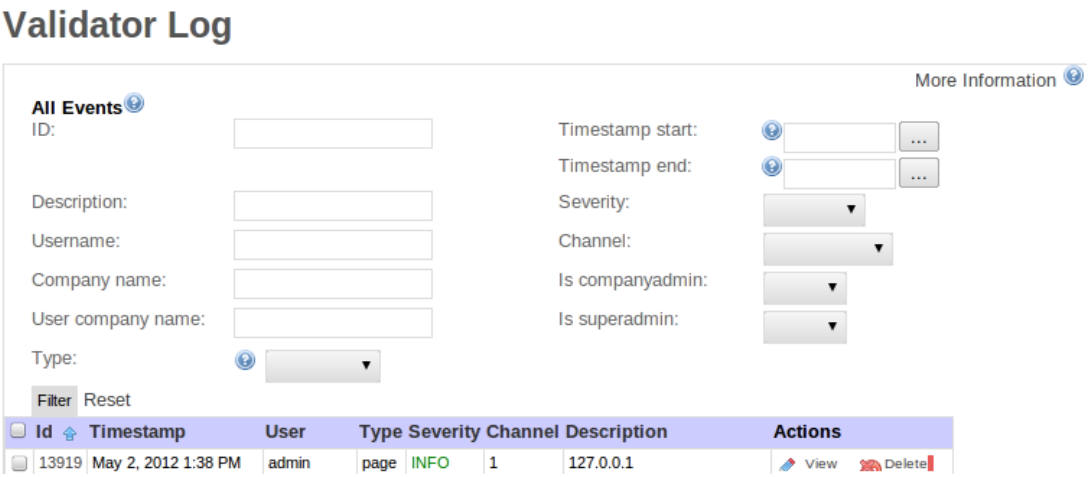
```
$idcriterions = $c->getNewCriterion(AllEventsPeer::EVENT_ID, -1);//get
no events to new criterion
$c2 = new Criteria();
$cton2 = $c2->getNewCriterion(AllEventsPeer::EVENT_ID, '',
Criteria::NOT_EQUAL);//get all events

if($this->search_allevents['id'] != '')
{
```

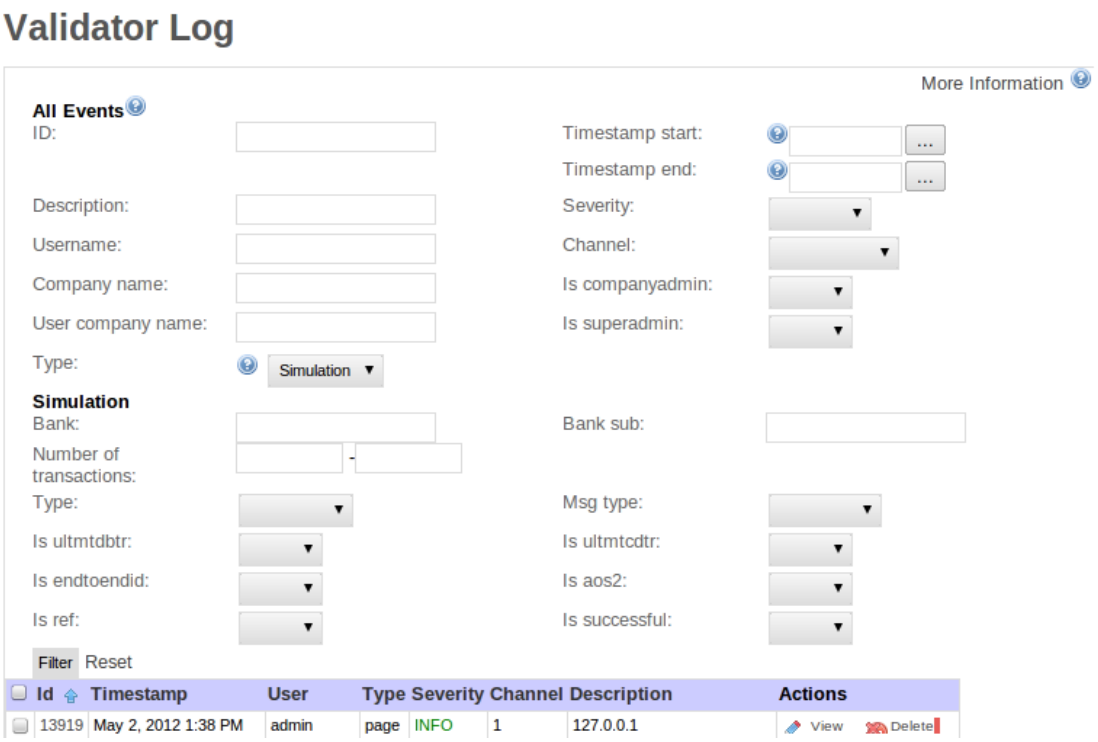
```
$cton3 = $c2->getNewCriterion(AllEventsPeer::EVENT_ID, $this-
>search_allevents['id'], Criteria::LIKE);
        $cton2->addAnd($cton3);
}
```

Vastauksesta otetaan lisätietotaulujen vierasavaimet muistiin. Kun kaikki hakuehdot on kyselty tietokannalta ja vierasavaimet listattu, haetaan yleisistä tiedoista jokainen tapahtuma joka vastaa vierasavaimia.

Omat suodattimet esitetään backend-moduulin pääsivun yläosassa (KUVIO 11). Kysymysmerkeistä tulee lisätietolaatikko kun osoittimen vie merkin päälle. ”More Information” –tekstiä klikkaamalla saa lisätietoa esille hakusuodattimien käytöstä. Alkunäkymässä on esillä pelkästään yleisten tietojen (”all_events” taulun) suodattimet. Tyypin valitsemalla saa sitä vastaavat suodatinkentät esille (KUVIO 12).



KUVIO 11 – Omien filttereiden alkunäkymä



KUVIO 12 – Omat suodattimet jossa simulaatio-tyyppi valittuna

4.5.2 Excel-lataus

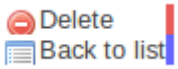
Excel-lataus onnistui helposti sivustolla jo käytetyn PHPExcel-kirjaston avulla. PHPExcel-kirjastolla voi luoda muistiin excel-tiedoston ja muokata koodin kautta tämän tiedoston sisältöä.

Excel-tiedosto luodaan niin, että jokainen lataukseen valittu tapahtuma ensin lisätään ensimmäiselle välilehdelle. Sen jälkeen tapahtumat lajitellaan uusille välilehdille lisätietojen mukaan. Välilehdet nimetään niiden sisältämien tietojen ja tapahtumamäärien mukaan. Jokaisessa välilehdessä on ensimmäisellä rivillä otsikkotiedot.

4.5.3 Tapahtuman tarkastelu

Symfonyn valmiiksi luomasta tapahtuman muokkaus-sivusta tehtiin pelkkä tapahtuman tarkastelu. Periaatteessa koko muokkaus-sivu kirjoitettiin uudestaan, pelkästään ”palaa listaan” ja ”poista tapahtuma” –napit jätettiin (KUVIO 13). Muokkaus-sivulle päästään tapahtuman yksilöintitietoa tai ”View”-nappia klikkaamalla. Ennen sivulle menoa

ajetaan MVC:n mallissa toiminto, jossa haetaan tiedot tietokannasta esitettäväksi tapahtuman yksilöintitiedon ja yleisien tietojen ”type”-kentän perusteella.



KUVIO 13 – Muokkaus-sivun jätetyt napit

Tapahtuma näytetään erillisellä sivulla kahdessa eri taulussaa. Ylempänä (KUVIO 14) on aina tapahtuman yleiset tiedot ja alempana (KUVIO 15) mahdolliset lisätiedot tapahtumasta. Lisäksi näytetään PHP:n ”print_r()”-methodilla tietokannasta haetut tiedot suoraan, jos taulukko jostain syystä ei toimi. Tämä tullaan vielä joko poistamaan tai korvaamaan visualisoinnilla jatkokehityksessä.

event_id	timestamp	type	username	company_name	user_company_name	severity	channel	is_companyadmin	is_superadmin	description
13071	2012-04-10 15:15:34	edit	JaakkoK	Demo Company Oy	XMLdation	1	1	1	1	...0.0.1 Company: Demo Company Oy

KUVIO 14 – Tapahtuman yleiset tiedot esitettynä taulukossa

event_id	target	subtarget	type	values	is_successful
13071	company	1	edit	, 1, Demo Company Oy, 2, , , , , , 1, 1, 10, 0, 0, 0, , 100, 1, 1, 1, 1, 0, <h1>XMLdation.com Oy:n toimitus- ja käyttöehdot 11/2009</h1> <h2>1 Rekisteröityminen </h2> <p>1.1 XMLdatio	1

KUVIO 15 – Tapahtuman lisätiedot esitettynä taulukossa

5 Jatkokehitys

Moduulissa on potentiaalisia jatkokehitysmahdollisuuksia, muunmuossa tapahtumien visualisointi kaavioilla ja yritysadmineille frontend käyttöliittymään oman yrityksen lokinäkymä. Tähän järjestelmään pitää myös lisätä lokikirjaus mahdollisista uusista ominaisuuksista, joita luodaan sivustolle. Uusien tapahtumien luominen on verrattaen helppoa Symfonyn uusiin lokikirjaus-filttereihin.

5.1 Visualisointi

Tietokannasta pystytään luomaan esimerkiksi PHP:lla luodulla pChart-kirjastolla kaavioita. Kaavioilla voisi esimerkiksi esittää sivuston käyttötiheyttä eri tasoilla. Tasoja voisi olla vaikka kaikki käyttäjät ja eri käyttäjäryhmät. Myös eri pankkien validointiputkien käyttötiheyttä voisi visualisoida.

5.2 Frontend käyttöliittymä

Frontend-puolelle voisi tarjota asiakkaille käyttäjä tai käyttäjäryhmäkohtaisia lokisivuja samaan tapaan kuin backend-puolen lokitapahtumien tarkastelu. Tämä luotaisiin uudeksi moduuliksi ja siihen voisi lisätä myös edellisessä kappaleessa käsitellyn visualisoinnin. Frontend-käyttöliittymä auttaisi asiakkaita seuraamaan omaa palvelun tarvettansa.

6 POHDINTA

Työn tavoitteena oli suunnitella ja toteuttaa lokijärjestelmä jo olemassa olevalle sivustolle. Vaatimuksina oli saada toteutettua helposti hallittava ja laajennettava uusi osa sivustolle, joka korvaisi lopulta vanhan lokijärjestelmän.

Sivusto, jolle uusi lokijärjestelmä tulisi, pystytettiin VirtualBox virtuaalikoneelle Ubuntu käyttöjärjestelmään. Järjestelmää suunniteltaessa pidettiin lopputavoite koko ajan mielessä ja edettiin vaiheittain sitä kohti. Vaiheet koostuivat suunnittelusta ja suunnitellun osan toteutuksesta. Vaiheet jakautuivat kutakuinkin tietokantaan, tietokannan ja sivuston yhdistämiseen, tietokantatapahtumien kirjaamiseen ja tapahtumien esittämiseen.

Lopputulos oli hyvä, koska tavoitteet saavutettiin. Järjestelmä jakautuu vain kolmeen kirjastoluokkaan, yhteen Symfonyn backend-applikaation moduuliin, sekä muutamiin YML-konfiguraatiotiedostoihin. Näillä pystyy hallitsemaan koko järjestelmää. Järjestelmää pystyy laajentamaan lisäämällä ehtoja halutun applikaation filter-kirjastoluokkaan samaan tapaan kun siellä jo olevat ehdot.

Järjestelmä on liitetty yrityksen testipalvelimelle käyttöön ja siitä on löytynyt muutamia vajaavaisuuksia jotka ovat jo korjattu tai lisätty. Eräs näistä oli esimerkiksi filter:in kääriminen PHP:n try-catch-rakenteen sisään, joka mahdollistaa lokikirjauksen virhetilanteessa virheilmoituksen kirjaamisen lokijärjestelmään. Tämä try-catch käärittiin vielä toisen try-catch-rakenteen sisään, mahdollisen tietokantayhteyden hajoamisen tuottaman virheen välttämiseksi.

Järjestelmää suunniteltaessa tuli kerrattua tietokannat ja niiden käyttö. Toteutettaessa oppi hyvinkin paljon. Kokemusta tuli erityisesti palvelinympäristöistä, MVC-frameworkeista ja yli kaiken PHP-ohjelmoinnista, jota ei ole tullut tehtyä ennemmin kovinkaan paljoa.

Järjestelmää on tarkoitus testata ja parantaa vielä ennen tuotantoon käyttöönottoa. Järjestelmään pitää myös lisätä lokikirjaukset tällä hetkellä kehitteillä olevista osista, koska nämä tulevat tuotantoon samaan aikaan kun lokijärjestelmä.

LÄHTEET

Official Ubuntu Documentation. ubuntu.com Luettu 11.11.2011. https://help.ubuntu.com/

Home | Ubuntu. ubuntu.com. Luettu 11.11.2011. https://www.ubuntu.com/

Ubuntu Tutorials. ubuntu-tutorials.com Luettu 11.11.2011. http://ubuntu-tutorials.com/2010/06/26/install-virtualbox-guest-additions-on-virtualbox-guests/

Tux Tweaks. Luettu 11.11.2011. http://tuxtweaks.com/2010/04/installing-lamp-on-ubuntu-10-04-lucid-lynx/

XMLdation wiki. xmldation.com. Luettu 11.11.2011. http://wiki.xmldation.com/

PEAR Symfony project. symfony-project.com. Luettu 11.11.2011. http://pear.symfony-project.com/

Writing good MVC code with the Symfony framework. Hugo Hamon. Luettu 11.05.2012. http://hugohamon.com/en/blog/writing-good-mvc-code-with-the-symfony-framework

NetBeans Docs & Support. netbeans.org. Luettu 11.11.2011. http://netbeans.org/kb/docs/php/configure-php-environment-ubuntu.html#specifyDocumentRoot

The symfony framework installation. symfony-project.org. Luettu 14.11.2011. http://www.symfony-project.org/installation/1_3

A Gentle Introduction to Symfony. symfony-project.org. Luettu 28.11.2011. http://www.symfony-project.org/get/pdf/gentle-introduction-1.4-en.pdf

MySQL Documentation. mysql.com. Luettu 19.12.2011. http://dev.mysql.com/doc/

PHP: Documentation. php.net. Luettu 21.12.2012. http://php.net/docs.php

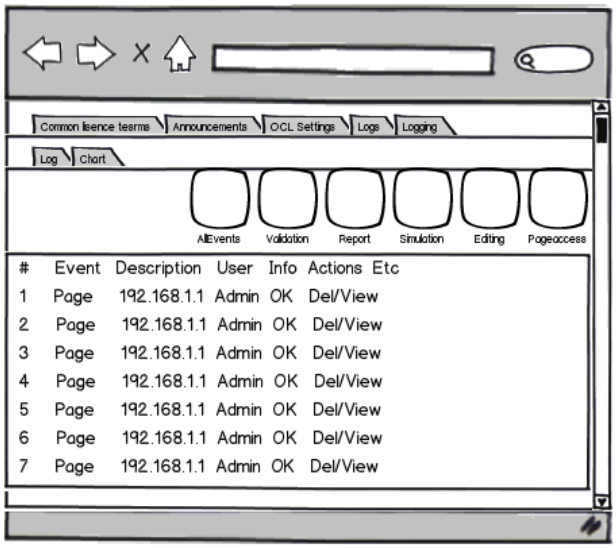
Symfony API. symfony-project.org. Luettu 2.1.2012. http://www.symfony-project.org/api/1_3/

Recently active 'symfony' questions. Stackoverflow. Luettu 9.1.2012. http://stackoverflow.com/questions/tagged/symfony

LIITTEET

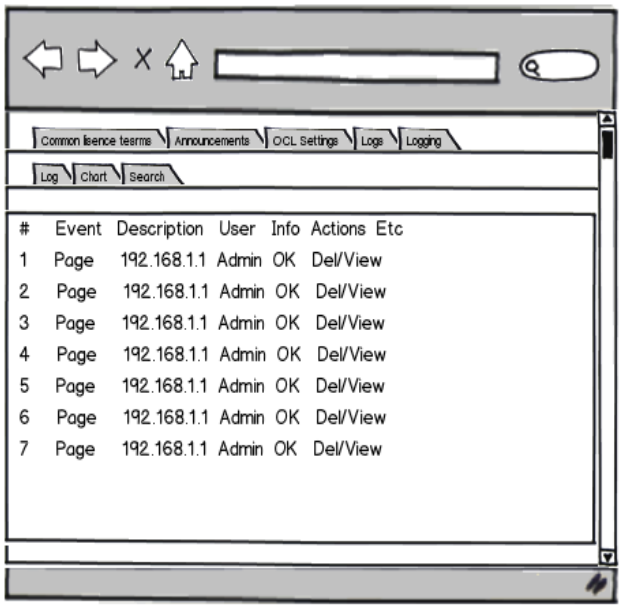
Liite 1. Käyttöliittymän suunnittelupiirrustukset balsamiq Mockups piirroksina. 1 (4)

Pääsivu suunnitelma 1:



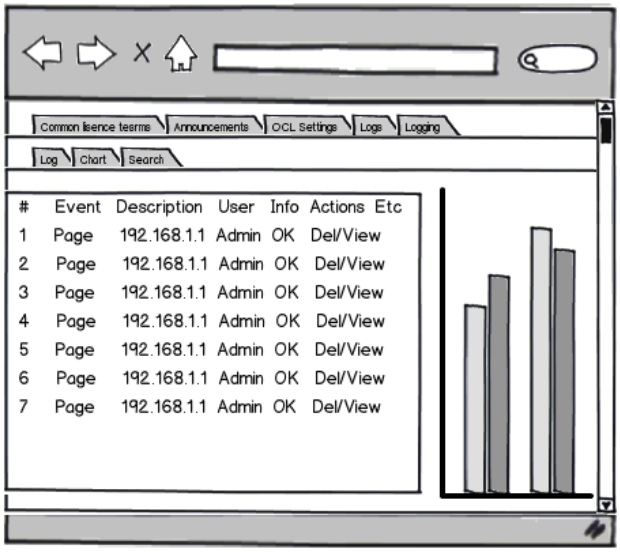
- 2 tabs
  -"Log" contains event listings (current view)
  -"Chart" contains general info in visual form
- 6 different collapsable search filters openned from buttons above the listing

Pääsivu suunnitelma 2:



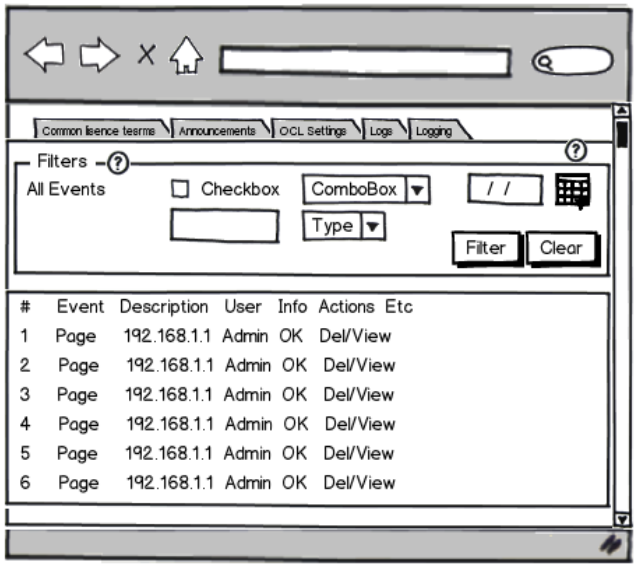
- 3 tabs
  -"Log" contains event listings (current view)
  -"Chart" contains general info in visual form
  -"Search" contains search filters
  -> redirects after search to "Log"
- 6 different collapsable search filters openned from buttons above the listing

Pääsivu suunnitelma 3:



- 2 tabs
  -"Log" contains event listings (current view)
  -"Chart" contains general info in visual form
  -"Search" contains search filters
  -> redirects after search to "Log"
- 6 different collapsable search filters openned from buttons above the listing
- Additional chart with general info on the "Log" page

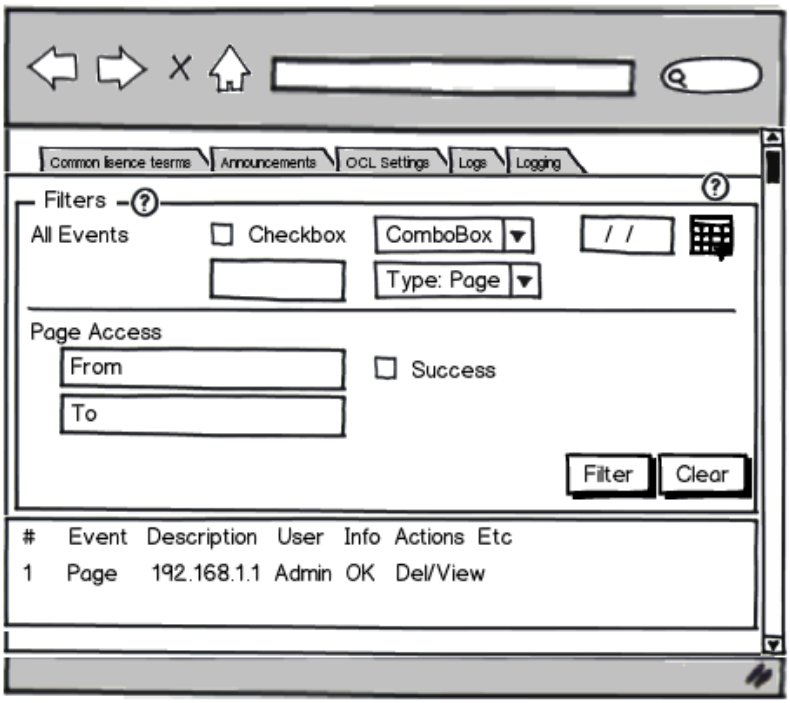
Pääsivu suunnitelma 4 (tyyppiä ei valittu):



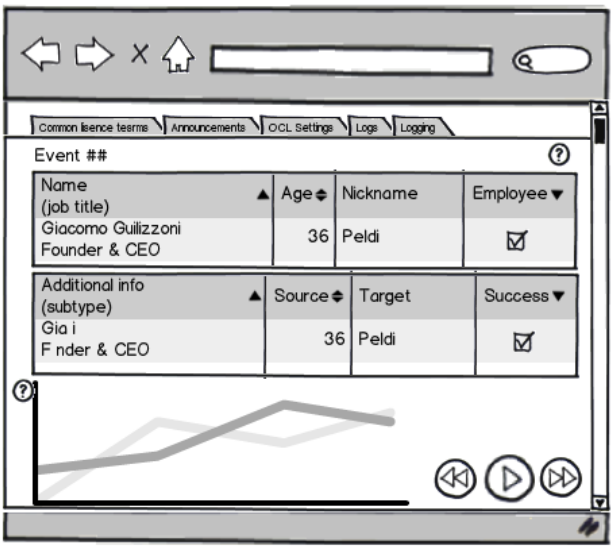
- No tabs
- All Events search filter visible at all times
- Additional filter options become available when Type-field is selected

3(4)

Pääsivu suunnitelma 4 (tyyppi valittu):

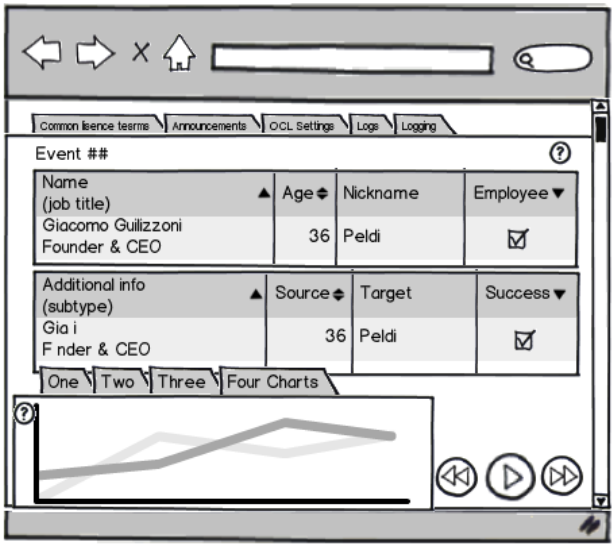


Tapahtuman tarkastelusivu suunnitelma 1:



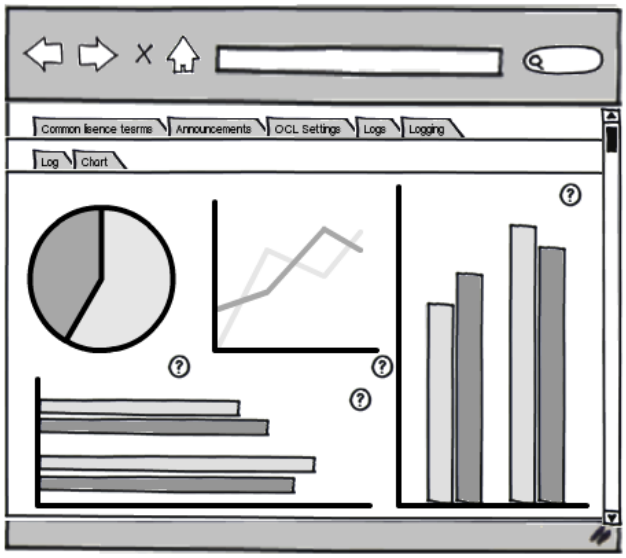
- Two datatables
  -first one containing the general info of the event
  -second contains the type specific information
- One chart to display related data visually

Tapahtuman tarkastelusivu suunnitelma 2:



- Two datatables
  -first one containing the general info of the event
  -second contains the type specific information
- Multiple charts displaying related data visually on tabpages

Kaaviosivun suunnitelma:



- 2 tabs
  -"Log" contains event listings
  -"Chart" contains charts (current)
- Several charts visualizing the site usage

Liite 2. /etc/apache2/httpd.conf-tiedosto

```
<VirtualHost *:80>
        ServerName www.xmldation.com.localhost
        ServerAdmin webmaster@localhost
        DirectoryIndex index.php
        DocumentRoot /home/jaakko/NetBeansProjects/work/trunk/web
        DirectoryIndex index.php
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /home/jaakko/NetBeansProjects/work/trunk/web>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>
        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>
        ErrorLog /var/log/apache2/error.log
        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn
        CustomLog /var/log/apache2/access.log combined
</VirtualHost>
```