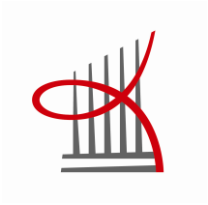


# XMLDATION VALIDAATTORIN TEAMS-MODUULI

Jukka Mäkiharju

Opinnäytetyö
Kesäkuu 2014
Tietotekniikka
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

**TIIVISTELMÄ**

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

JUKKA MÄKIHARJU:
XMLdation Validaattorin Teams-moduuli

Opinnäytetyö 48 sivua, joista liitteitä 8 sivua
Kesäkuu 2014

---

Tässä työssä käsitellään moduulin lisäämistä Symfony-frameworkilla tehtyyn sivustoon.

Työssä käydään läpi kehitysympäristön pystytys, moduulin suunnittelu ja sen toteutus. Lopputuloksena on toimiva sivu tietokantoineen, käyttöliittymineen ja taustalla olevine toimintoineen. Näiden ohella tutustutaan Symfony-frameworkin perusteisiin käymällä läpi Symfonyssa käytetty PHP-framework ja MVC-malli.

Työssä tehty moduuli lisättiin yrityksen olemassa olevaan ympäristöön. Uusi moduuli lisää tiimi-tason yrityksen johtajan ja normaalin käyttäjän väliin.

Työn perusteella lukija osaa pystyttää paikalliseen ympäristöön palvelimen ja tietokannan. Lukijalle tulee selväksi Symfonyn asentaminen, moduulin luominen ja tietokannan käsittely ORM-mallin mukaisesti.

Lopuksi pohditaan työn lopputulosta sekä frameworkin etuja ja haittoja.

---

Asiasanat: symfony, php, mysql, tietokanta, MVC

**ABSTRACT**

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Computer sciences
Software engineering

JUKKA MÄKIHARJU:
XMLdation Validator Teams Module

Bachelor's thesis 48 pages, appendices 8 pages
June 2014

---

This thesis is about adding a module to a website made with Symfony framework.

The thesis covers the setting up the development environment, designing the module and implementing the module. The result will be a fully functioning webpage with database connections, user interface and underlying functions. Additionally the basics of Symfony framework will be covered by looking at the PHP framework and MVC model used in Symfony.

The module was added to an existing environment used by the company. The new module adds a team level between the company administrator and a normal user.

By following this thesis the reader is able to set up a server and a database to local environment. The reader will also know how to install Symfony, create a module and manage a database using the ORM model.

Finally the result of the thesis and the pros and cons of using a framework are looked at.

---

Key words: symfony, php, mysql, database, MVC

**SISÄLLYS**

**ERITYISSANASTO**

| | |
|---|---|
| Linux | Avoimen lähdekoodin käyttöjärjestelmä |
| Apache | Web-palvelin |
| Ubuntu | Linuxin ilmainen jakeluversio |
| VirtualBox | Pääasiassa käyttöjärjestelmien virtualisoimiseen käytetty ohjelma |
| LAMP | Asennuspaketti, johon kuuluu Linux, Apache, MySQL ja PHP |
| NetBeans | Ohjelmointiympäristö, joka on suosittu Java-ohjelmoinnissa |
| HTTP | Hypertext Transfer Protocol, tiedonsiirtoprotokolla selaimen ja palvelimen välillä |
| PHP | Ohjelmointikieli, jota käytetään erityisesti web-ohjelmoinnissa |
| framework | Kokoelma valmiita ohjelman osia |
| Symfony | Avoimen lähdekoodin PHP framework |
| subversion | Versionhallintaohjelma (SVN) |
| MVC | Model-view-controller -malli, jolla suunnitellaan ja toteutetaan graafisia käyttöliittymiä |
| XSS | Cross-site scripting mahdollistaa haitallisen tiedon syöttämisen web-sivulle |
| ORM | Object-relational mapping mallintaa olio-mallisesti muuten keskenään yhteensopimatonta dataa. |
| XML | Rakenteellinen kuvauskieli, jolla myös useat pankkien maksuaineistot on tehty |
| validointi | XML-aineistojen sisällön tarkistaminen tehtyjen sääntöjen perusteella |
| simulointi | XML-viestin palauttaminen lähetetystä maksuaineistosta |
| propel | ORM-mallinnus tietokannalle PHP-kielellä |
| doctrine | ORM-mallinnus tietokannalle PHP-kielellä |

# 1 JOHDANTO

Tämä työ kuvaa uuden moduulin lisäämistä XMLdation-nimisen yrityksen Symfony-frameworkilla toteutettuun järjestelmään. Moduulin avulla toteutetaan toiminnallisuus tiimien hallintaan. Tiimien avulla yrityksen johto ja tiimien vetäjät voivat hallita käyttäjien tietoja, kuten käyttäjille annettuja oikeuksia.

Työstä kertominen aloitetaan käymällä läpi kehitysympäristön pystyttäminen. Luvussa 2 käsitellään virtuaalikoneen, käyttöjärjestelmän, käytettävien ohjelmien ja palvelimen asentaminen sekä palvelimen ja tietokannan konfigurointi.

Luvussa 3 käydään läpi Symfony PHP-framework ja Model-View-Controller -malli. Luvussa selvitetään mikä PHP-framework on ja miten se liittyy Symfonylla tehtävään työhön. Lisäksi selvitetään miten Model-View-Controller -mallia on sovellettu Symfony-ympäristössä.

Luvussa 4 käsitellään palvelun nykyinen tilanne eli selvitetään käytettävissä olevat palvelut ja käyttäjähierarkia.

Luvussa 5 käydään läpi työn varsinainen aihe eli tiimi-tason suunnittelu ja toteutus. Suunnittelussa käsitellään tietokannan ja käyttöliittymän suunnittelu. Tämän jälkeen käsitellään tietokannan, toimintojen ja käyttöliittymän toteutus.

Lopuksi luvussa 6 pohditaan miten työ onnistui ja lisäksi mitä etuja ja haittoja frameworkin käyttämisessä voi olla.

# 2 KEHITYSYMPÄRISTÖN PYSTYTTÄMINEN

Kehitysympäristö koostuu käyttöjärjestelmästä ja siihen asennetuista ohjelmista ja kirjastoista. Tässä työssä kehitysympäristöön kuuluu virtuaalikoneeseen asennettu Ubuntu-käyttöjärjestelmä, johon on asennettu ohjelmointiympäristö, palvelin, tarvittavat kirjastot ja tietokanta.

## 2.1 Virtuaalikone

Windows-käyttöjärjestelmään asennetaan Oraclen VirtualBox-ohjelmisto, johon tehdään uusi virtuaalikone. Kehitysympäristö asennetaan virtuaalikoneeseen, koska monet työssä käytetyt ohjelmat ovat Windows-ohjelmia, jolloin myös Windows täytyy olla käytössä. Virtuaalikonetta käytettäessä on hyvä huomioida, että tietokoneen resurssit jaetaan fyysisen koneen ja käytetyn virtuaalikoneen välillä.

Virtuaalikoneen valikosta luodaan uusi kone (Machine->New), jolle annetaan haluttu nimi ja valitaan käyttöjärjestelmän tyyppi ja versio. Tässä tapauksessa valittiin Linux ja Ubuntu 32-bit. Järjestelmän muistiksi asetettiin noin puolet fyysisen järjestelmän muistista eli 4 Gt ja tallennustilaksi 60 Gt. Videokiihdytys täytyy asentaa virtuaalikoneen asetuksista (Devices->”Install Guest Additions..”), jotta saadaan virtuaalikoneen käyttöjärjestelmä näkymään koko näytöllä.

## 2.2 Ubuntu

Ubuntun on Canonicalin ylläpitämä avoimen lähdekoodin käyttöjärjestelmä. Käytössä on 32-bittinen Ubuntu 12.04 LTS. Ubuntu asennetaan käyttöjärjestelmille yleisellä tavalla käyttämällä CD-levyä tai USB-muistitikkua. Asennus on erittäin suoraviivaista ja ohjattua eikä käytännössä eroa Windows-käyttöjärjestelmän asentamisesta. Asennuksen ja tietokoneen uudelleenkäynnistämisen jälkeen käyttöjärjestelmä on valmiina kehitysympäristön muiden osien asentamiseen.

## 2.3 NetBeans

Ohjelmointiympäristönä käytetään Ubuntuun asennettua Netbeans IDE:n versiota 7.2. NetBeansin asennuspaketti haettiin NetBeansin sivustolta, koska Ubuntun pakettivarastosta löytyi vanhempi versio. Netbeans on tunnettu Java-kielen kehitysympäristönä, mutta kehitystyö tehdään PHP-kielellä. NetBeans on käytössä lähinnä siihen valmiina löytyvien ja hyvin toimivien lisäosien ja testaustyökalujen takia.

## 2.4 LAMP:n asennus ja konfigurointi

Kehitystyö tehdään paikallisesti, joten kehitysympäristöön täytyy asentaa palvelin. Palvelin on osa LAMP-pakettia. LAMP-lyhenne kuvaa Linux-ympäristössä erittäin yleistä web-palvelinteknologian kokoonpanoa, johon kuuluu Linux-käyttöjärjestelmä, Apache -http-palvelin, MySQL-tietokanta ja PHP-skriptikieli.

LAMP:in asennus on varsin yksinkertaista esimerkiksi Ubuntu Suomen sivuilta löytyvillä ohjeilla. Sivusto löytyy osoitteesta http://wiki.ubuntu-fi.org/LAMP_Asennus. Tarvittavat ohjelmat asentuvat kirjoittamalla komentoriville komento:

```
sudo aptitude install apache2 php5 apache2.2-common libapache2-mod-
auth-mysql php5-mysql mysql-server
```

LAMP:in asennuksen jälkeen puuttuu muutama tarvittava tiedosto, jotka täytyy asentaa manuaalisesti. Puuttuvat tiedostot ovat:

```
php5-ldap
php5-curl
php5-sqlite
php-pear
php5-pgsql
php5-xsl
```

MySQL-palvelimen asennuksen yhteydessä ohjelma kysyy root-salasanaa, jota ei tässä vaiheessa tarvitse antaa, mutta tietoturvasyistä se kannattaa asettaa jo tässä vaiheessa.

Salasana on myös syytä muistaa tai sivustoa ei pääse pystyttämään eikä tietokantaa pääse muokkaamaan. Apachen toimivuuden voi testata kirjoittamalla selaimen osoiteriville osoite ”http://localhost”. Jos Apache on asentunut oikein, avautuu selaimessa palvelimen oletus web-sivu, jonka otsikkona on teksti: ”It works!”. Oletuksena Ubuntussa Apache etsii web-sivut hakemistopolusta ”/var/www/”. PHP:n toimivuutta voi testata tallentamalla ”/var/www/”-kansioon esimerkiksi ”test.php”-tiedosto, jossa on jokin php-koodi, kuten

```
<?php echo ”Hello world!”;  ?>
```

Kun selaimen osoitteeksi kirjoittaa ”http://localhost/test.php”, selaimen ikkunaan pitäisi tulostua teksti ”Hello world!”.

Seuraavaksi konfiguroidaan Apache näyttämään sivut vain paikallisesti kehityskäytön takia. Apachen asetuksia muokataan tekstinkäsittelyohjelmalla antamalla päätteeseen komento:

```
sudo gedit /etc/apache2/ports.conf
```

Apache ”kuuntelee” oletuksena HTTP:n porttia 80. Apachen kuuntelema portti 80 vaihdetaan osoittamaan internetistä localhostiin eli ”ports.conf”-tiedostossa oleva ”Listen 80” rivi korvataan rivillä ”Listen 127.0.0.1:80”. Näin kehityskäyttö ei vaadi yhteyttä varsinaiseen käyttöympäristöön internetissä. Työn lopputulos siirretään lopulta manuaalisesti käytettäväksi palvelussa.

Lopuksi Apache ja MySQL käynnistetään uudelleen muutosten käyttöönottamiseksi antamalla päätteessä komennot:

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

## 2.5 Symfonyn asentaminen

Seuraavaksi asennetaan Symfony-framework. Käytetty Symfonyn versio on 1.3.6. Saatavilla oli myös Symfonyn 2.0 -versio, mutta sen tiedostorakenne ja tietokannan hallinta eroavat merkittävästi vanhemmasta frameworkista, joten sitä ei voi käyttää samassa ympäristössä. Asennus onnistuu helpoiten käyttämällä PEAR-pakettienhallintaohjelmaa. PEAR asennetaan antamalla komentoriville komento:

```
pear channel-discover pear.symfony.com
```

Tämä komento rekisteröi Symfonyn kanavan käytettäväksi asentajassa. Symfony-framework voidaan tämän jälkeen asentaa käyttämällä PEAR:in kanavaa:

```
pear install symfony/symfony-1.3.6
```

Tämän jälkeen Symfony-framework on asennettu ja kaikki siihen sisältyvät toiminnot ovat käytettävissä.

Tietokannasta tullaan hakemaan tietoa suurilla tietokantakomennoilla, joten MySQL:n asetuksia täytyy muokata sallimaan isot tiedostot. Asetustiedosto sijaitsee linuxin tiedostopolussa: ”/etc/mysql/my.cnf”. Asetustiedostosta muokataan paketin maksimi koko arvoksi 100 megatavua: ”max_allowed_packet = 100”. Ylimitoitetulta vaikuttava arvo on kuitenkin perusteltu, jotta tuotantoympäristön tietokanta saadaan tuotua paikalliseen kehitysympäristöön.

PHP:n pitää pystyä hallitsemaan suuret tiedostot ja POST-komennot, joten myös PHP:n asetuksia muokataan tiedostopolussa: ”/etc/php5/apache2/php.ini”. Asetustiedoston arvoja muokattiin seuraavilla arvoilla:

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

## 2.6 Sivuston pystyttäminen

Ubuntuun asennettuun NetBeans-ohjelmointiympäristöön asennetaan tuki PHP-kielelle ja subversion-liitännäinen NetBeansin ”Tools”-valikon ”Plugins”-välilehdeltä. Asennuksen jälkeen subversion konfiguroidaan ottamaan yhteys käytettyyn subversion-palvelimeen ja käyttämällä NetBeansin SVN-työkalua haetaan uusin versio kehitettävästä sivustosta paikalliseen ympäristöön.

Jotta sivusto toimii oikein, täytyy muodostaa symboliset linkit projektikansion ja paikallispalvelimen välille. Ilman linkkejä sivusto ei osaa käyttää paikallispalvelimen Symfony-ympäristöä, jolloin Symfonyn tietokanta-luokat eivät löydä käytettyä tietokantaa ja monet sivustolla käytetyt liitännäiset eivät toimi. Linkit luodaan antamalla järjestelmänvalvojana 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
```

Käytetty tuotantotietokanta siirretään MySQL:n paikalliseen käyttöön komentoriviltä komennolla:

```
mysql -u root -p local_databasename < database_file.sql
```

Aluksi käytetyn tietokannan tuominen kaatui virheeseen, jonka perusteella tietokantapaketit olivat suurempia kuin mitä MySQL:n asetukset sallivat. Siksi edellä mainitut MySQL:n asetustiedoston muutokset ovat tarpeellisia.

Tässä vaiheessa Apache-palvelin viittaa ”http://localhost/”-osoitteeseen. Palvelin täytyy asettaa viittaamaan Symfonyn projektikansioon. Tämä tehdään muokkaamalla tiedostoa hakemistopolussa ”/etc/apache2/httpd.conf”. Apachen httpd-asetustiedosto on liitteessä 1. Merkittävimpinä tiedostossa ovat asetukset otsikoilla ”VirtualHost”, ”DocumentRoot” ja ”FollowSymLinks”.

VirtualHost:in attribuutti ”*:80” asettaa virtualhost-palvelun saataville localhost:issa eli osoitteessa ”http://localhost/”. Attribuutin ”*”-merkki hyväksyy kaikki IP-osoitteet, joten IP-osoitteen muuttuminen lähiverkossa ei aiheuta ongelmia. DocumentRoot kertoo Apache-palvelimelle mistä käytetyt tiedostot löytyvät ja FollowSymLinks kertoo palvelimelle, että sen tulisi seurata edellä asetettuja symbolisia linkkejä. Asetusten tekemisen jälkeen Apache täytyy käynnistää uudelleen komennolla:

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

Tämän jälkeen asennus ja konfigurointi on valmis. Paikallispalvelin on käytettävissä ja selain löytää Symfonyn paikallisen kehitysympäristön osoitteesta ”http://localhost/frontend_dev.php”.

# 3 SYMFONY FRAMEWORK

Symfony on Sensio Labsin kehittämä avoimen lähdekoodin framework, jota käytetään PHP-kehitykseen. Symfonyä käyttävät muun muassa Yahoo!, Dailymotion, phpBB ja Drupal. Symfony on kirjoitettu PHP5:llä ja tällä hetkellä se on yksi suosituimmista frameworkeista web-kehitykseen. Symfonyn vahvuutena kehitystyötä ajatellen on kattava ja toimiva yhteisön tuki. Symfony on lisäksi yhteensopiva muun muassa MySQL-, PostgreSQL- ja SQLite-tietokantojen kanssa. Symfony on julkaistu Open Source MIT -lisenssillä, joka ei rajoita käyttöä ja sallii sekä avoimen lähdekoodin, että yksinoikeudella tehtyjen ohjelmistojen kehittämisen.

## 3.1 Symfony PHP framework

Symfonyn ”web framework” on kokoelma luokkia, jotka on kirjoitettu PHP:lla.
Frameworkin tarkoituksena on parantaa ohjelmiston kehitystyötä siten, että ohjelmistokehittäjät voivat keskittyä oman ohjelmiston toimintojen tekemiseen sen sijaan, että he käyttäisivät aikaa ohjelmiston taustalla olevan rakenteen kehittämiseen. Olemassa oleva ja testattu framework parantaa ohjelmiston laatua, luotettavuutta ja virheensietokykyä. Frameworkiin saatavat päivitykset voivat tuoda uusia ominaisuuksia ja parannuksia suorituskykyyn ilman frameworkin käyttäjien kehitystyötä.

Käytännössä Symfonyn framework näkyy kehittäjille siten, että valmiita metodeja voidaan käyttää monimutkaistenkin toimintojen suorittamiseen. Tämän lisäksi valmiista luokista voidaan periyttää omia luokkia, joita voidaan muokata tarpeen mukaan. Tämä ei riko frameworkin rakennetta ja antaa kehittäjille mahdollisuuden toteuttaa monipuolisia toimintoja käyttämällä Symfony-frameworkin omia toimintoja kehitystyön pohjana.

Symfonyn projektikansio on jaettu kansioihin niiden toimintojen ja käyttötarkoitusten mukaan. Kansiorakenne on jaettu applikaatioihin (apps), konfiguraatiotiedostoihin (config), liitännäisiin (plugins), kirjastotiedostoihin (lib), testitapauskansioihin (test) ja verkkosivustokansioihin (web). Tärkeimpinä kansioina tämän työn suhteen voidaan pitää muun muassa lib-kansion alla olevia model (malli)- ja form (lomake) -kansioita, jotka sisältävät vastaavasti malli- ja lomake-luokat. Malli-luokissa on tietokannasta

muodostetut luokat ja metodit tietokantataulujen käsittelyyn sisältäen muun muassa get- ja set-metodit, joilla tietokantataulun arvot haetaan ja asetetaan. Lomake-luokissa on web-lomakkeissa käytettävissä olevia syötekenttiä, kuten käyttäjän nimi tai validointiputken kuvaus, joilla tieto voidaan tallentaa suoraan tietokantaan kunhan lomakkeeseen on tehty tallennustoiminto.

Applikaatiot on jaettu backend- ja frontend-kansioihin. Molemmille applikaatioille on omat konfiguraatio-, kirjasto-, moduuli (modules)- ja näkymäkansiot (templates). Applikaatioilla voi olla myös muita kansioita, kuten lokalisaatiokansio kielikäännöksiä varten.

### 3.2 Symfony MVC

Symfony perustuu Model-View-Controller (MVC) -rakenteeseen, jossa ohjelman toiminnot on jaettu kolmeen osaan. MVC-rakenne on usein käytössä ohjelmistoissa, joissa on käyttöliittymä, kuten web-ympäristössä. Malli (model) koostuu tiedosta, jota järjestelmä käsittelee. Näkymä (view) esittää tiedon käyttäjälle. Käsittelijä (controller) ottaa vastaan käyttäjän komennot ja välittää ne joko malli- tai näkymä-kerrokselle.

Malli koostuu Symfonyssä pääasiassa tietokannasta. Symfonyssä on käytössä Object-relational mapping (ORM) -tekniikalla toteutetut Doctrine ja Propel. ORM mahdollistaa saman tiedon käyttämisen keskenään yhteensopimattomissa olio-pohjaisissa järjestelmissä. Tässä työssä käytetään Propel:ia. Propel luo tietokantatauluista PHP-luokat ja mahdollistaa tietokannan käsittelyn olio-ohjelmoinnin keinoin käyttämällä luokista luotuja olioita ja niiden metodeja, joten perinteisiä SQL-kyselyitä ei tarvitse rakentaa.

Näkymä esittää tiedon käyttäjälle. Näkymä on se, mitä käyttäjälle esitetään web-selaimessa, joten se on rakennettu HTML- ja PHP-kielellä ja muotoiltu usein CSS-tyylisivukielellä. Näkymän muokkaamiseen käytetään usein myös Javascript-kieltä.

Käsittelijä asettaa näkymässä tarvittuihin muuttujiin arvot ja se ajetaan aina ennen näkymän esittämistä. Oletuksena käsittelijä on tyhjä luokka, mutta sen alle voidaan luoda julkisia metodeja, jotka käsitellään web-selaimessa omina web-sivuina jos

metodeille on tehty näkymä tiedon esittämistä varten. Yleisesti käsittelijän luokka kuitenkin toimii kuten mikä tahansa olio-pohjaisen ohjelmointikielen luokka metodeineen ja attribuutteineen.

Symfonyssä on mahdollista luoda samalle sivustolle eri applikaatioita, jotka ovat itsenäisiä eivätkä jaa keskenään mitään tietoa. Applikaatioilla on ainoastaan yhteiset tiedostot projektikansion lib-kirjastokansiossa. Symfonyssä sivusto on usein jaettu frontend- ja backend-applikaatioihin. Erillisiä applikaatioita ei välttämättä tarvita, mutta web-ympäristössä kannattaa erottaa käyttäjän (frontend) ja hallinnan (backend) ympäristöt toisistaan. Tällöin backend:ille saa asetettua tarkemmat turvatarkastukset ja sillä saa hallittua web-käyttöliittymästä koko sivuston tietoja. Esimerkiksi frontend-käyttäjien tietoja voi muokata backend-käyttöliittymästä suoraan ja frontend tarjoaa asiakkaille näkyvän sivuston.

# 4 PALVELUN NYKYINEN TILANNE

Nykyisessä tilanteessa ”Company admin”-käyttäjä on ylin käyttäjä ja ainoa, joka pystyy hallitsemaan ”Normal”-käyttäjiä. Käyttäjillä voi olla käyttöoikeus validointiputkiin ja simulointiputkiin. Käyttäjien putkioikeudet voidaan säätää eri tasoilla. Käyttäjille voidaan valita putkioikeudet periytyväksi yhtiö-tasolta tai käyttäjille voidaan valita yksittäiset putkioikeudet käyttäjäkohtaisesti.

## 4.1 Validointiputkioikeudet

XML-aineistojen validointi on käytetyin ominaisuus palvelussa. Maksuaineisto validoidaan ensin XML-skeeman perusteella. Tämän lisäksi jokaiselle validointiputkelle on pankki- tai yrityskohtaisesti tehdyt säännöt, joiden perusteella käyttäjän validointipalveluun syöttämä aineisto tarkastetaan. Tehtyjen sääntöjen perusteella XML-aineisto voidaan joko hyväksyä tai hylätä. Validoinnista näytetään käyttäjälle raportti, jossa näytetään sääntöjen perusteella löytyneet virheet.

## 4.2 Simulointiputkioikeudet

Simulointiputkiin kuuluvat maksupalautteet ja tiliraportit. Maksupalautesimulaatiolla (”status feedback”) voidaan simuloida viestiä, joka palautetaan maksutiedostosta maksutilanteessa. Viesti voi sisältää palautteen virheellisestä tai hyväksytystä tilanteesta. Palautteena voi olla esimerkiksi hyväksytty tekninen validointi tai tieto siitä, että ensimmäinen tilisiirto on hylätty virheellisen tilinumeron takia.

Tiliraportteihin (”account reporting”) kuuluvat tapahtumaraportit (”camt.054”) ja tiliotteiden simuloinnit (”camt.053”). Tapahtumaraportti voi sisältää esimerkiksi viiteluettelon, ennakkotiedot tai erittelyt. Tiliotteiden simuloinnilla voidaan esittää tiliote visuaalisessa muodossa.

Työssä viitataan tästä eteenpäin vain validointi- ja simulointiputkioikeuksien käsittelyyn asian yksinkertaistamiseksi. Simulointiputkien oikeuksien käsittely tapahtuu samalla tavalla kuin validointiputkioikeuksien käsittely.

### 4.3 Käyttäjähierarkia

Järjestelmässä on käytössä eri käyttäjätasoja. Peruskäyttäjät (”Normal”) ovat käyttäjiä, joilla ei ole mitään omien tai muiden oikeuksien hallintaan liittyviä toiminnallisuuksia. Näitä käyttäjiä hallitsevat yrityksen johtajat (”Company admin”), jotka voivat muokata, luoda ja poistaa käyttäjiä sekä hallinnoida käyttäjien oikeuksia. Yrityksen johtajat eivät voi kuitenkaan hallita toistensa oikeuksia. Yritysten johtajien oikeudet säädetään XMLdation:in toimesta sopimuskohtaisesti.

# 5 TIIMI-TASO

Työn tarkoituksena on lisätä moduuli, jolla saadaan uusi hallintataso olemassa olevien käyttäjätasojen väliin. Tästä voi olla hyötyä esimerkiksi yrityksille, joilla on yksiköitä eri maissa. Tällöin tiimien hallinnoijat voivat säätää käyttäjien oikeuksia maakohtaisesti.

Työssä toteutettava tiimi-taso lisää hallintatason ”Company admin”- ja ”Normal”-käyttäjien välille. Lisättävä käyttäjätaso on ”Team admin” -taso, jolla on hallintaoikeus oman yrityksen sisällä omaan tiimiin kuuluviin käyttäjiin. Yrityksessä ei ole pakko käyttää tiimejä käyttäjäoikeuksien hallintaan. Tiimit asetetaan yrityskohtaisesti päälle XMLdation:in puolesta. Yritys-tasolta säädetään tiimien ja ”Team admin”-käyttäjien maksimimäärä.

Jos yrityksellä on tiimi-oikeudet asetettuna aktiiviseksi, jokaiselle tiimille valitaan erikseen validointi- ja simulointiputkioikeudet. Tiimiin kuuluvalle käyttäjälle valitaan näistä putkista joko tiimikohtaiset tai käyttäjäkohtaiset oikeudet.

Tiimi-tason hallintaoikeuden haltija (”Team admin”) voi säätää oman tiiminsä käyttäjien putkien käyttöoikeuksia. Hallintaoikeuksiin kuuluu myös käyttäjien lisääminen ja poistaminen sekä käyttäjien siirtäminen eri tiimien välillä jos ”Team admin” -käyttäjällä on hallintaoikeus useaan tiimiin. Lisäksi ”Team admin” -käyttäjä voi muokata tiimiensä käyttäjien tietoja, kuten käyttäjänimeä tai salasanaa.

## 5.1 Suunnittelu

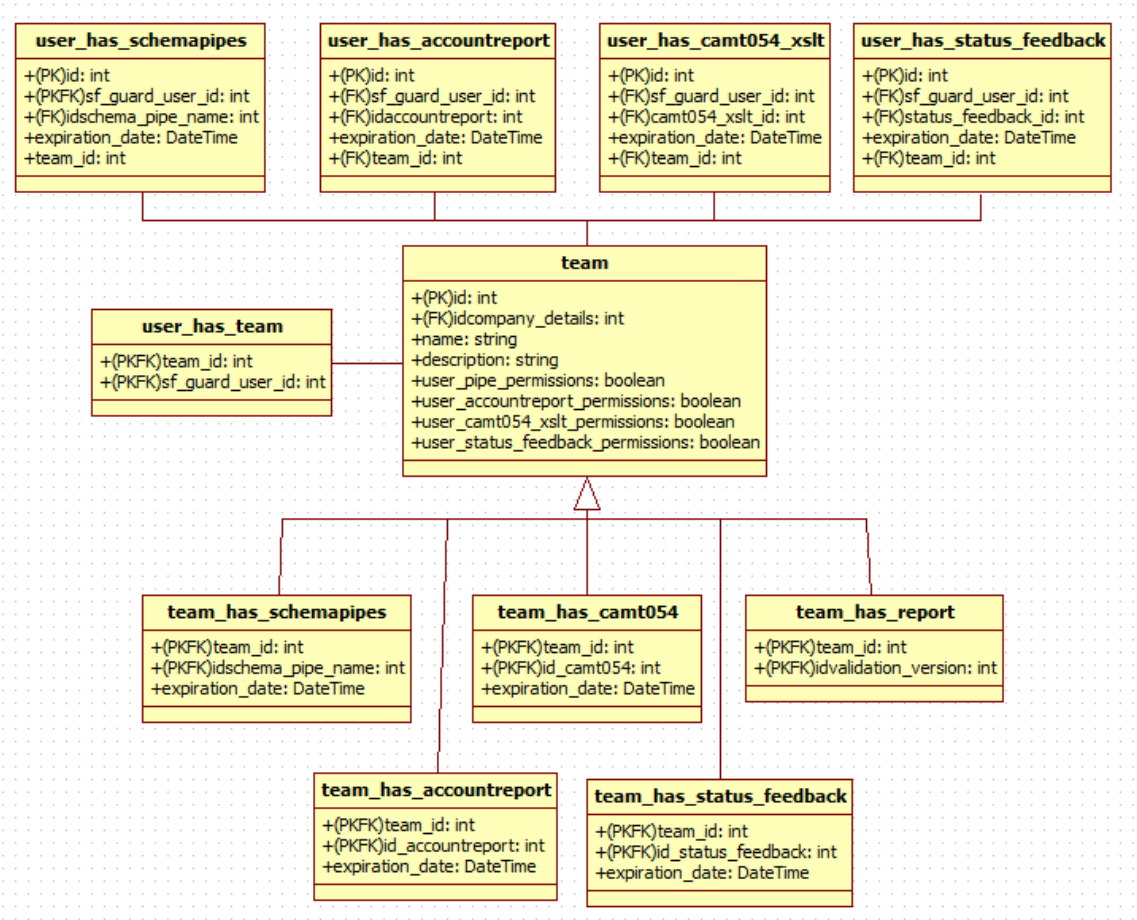
Suunnitteluun kuuluu tietokannan ja käyttöliittymän suunnittelu. Tietokantaan pitää pystyä tallentamaan luodut tiimit. Yritysten, käyttäjien ja käyttöoikeuksien suhteet luotuihin tiimeihin pitää myös tallentaa. Sivuston käyttöliittymän kautta pitää pystyä hallinnoimaan tiimejä, tiimien jäseniä ja tiimien jäsenten oikeuksia.

Tiimistä pitää pystyä tallentamaan ainakin tiimin nimi ja tiimiin kuuluvat käyttäjät. Tiimin tiedoissa näkyy kaikki yrityksen tasolta periytyvät oikeudet, joita voidaan jakaa edelleen tiimin jäsenille joko valitsemalla kaikki oikeudet yrityksen tasolta tai

valitsemalla ne käyttäjille yksitellen. Sekä käyttäjille periytyvät, että yksittäin valittavat oikeudet täytyy pystyä tallentamaan.

### 5.1.1 Tietokanta

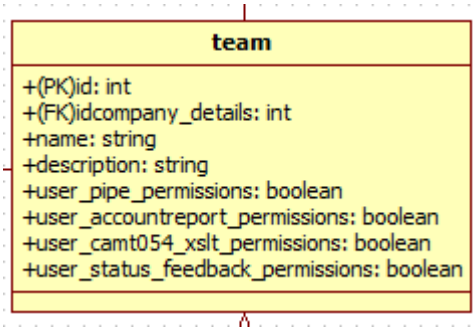
Tietokanta suunnitellaan StarUML-ohjelmalla (KUVIO 1). Tietokantaan tarvitaan seitsemän uutta taulua. Näiden lisäksi käyttäjäkohtaisia oikeuksia tallentaviin tauluihin lisätään viittaus tiimi-tauluun. Tiimi-taulun lisäksi tarvitaan viisi linkkitaulua tiimille kuuluvien oikeuksien tallentamiseen. Linkitys tiimin ja käyttöoikeusosa-alueen välillä tehdään vierasavaimilla.




KUVIO 1 - Suunniteltu tietokanta

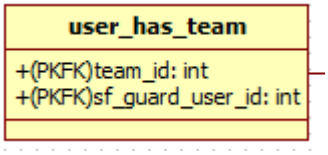
Tiimi-tauluun (KUVIO 2) tallennetaan yrityksen tunniste, tiimin nimi, vapaa kuvaus ja tieto siitä määritelläänkö käyttäjäoikeudet yksitellen tiimin jäsenille. Tarvitut kentät ovat idcompany_details (yrityksen tunniste), name (tiimin nimi), description (tiimin kuvaus), user_pipe_permissions (käyttäjäkohtaiset putkioikeudet),

user_accountreport_permissions (käyttäjäkohtaiset tilioteoikeudet), user_camt054_xslt_permissions (käyttäjäkohtaiset tapahtumaraportit) ja user_status_feedback_permissions (käyttäjäkohtaiset tilanneraporttioikeudet). Yrityksen tunniste -kenttä on vierasavain yritys-tauluun.



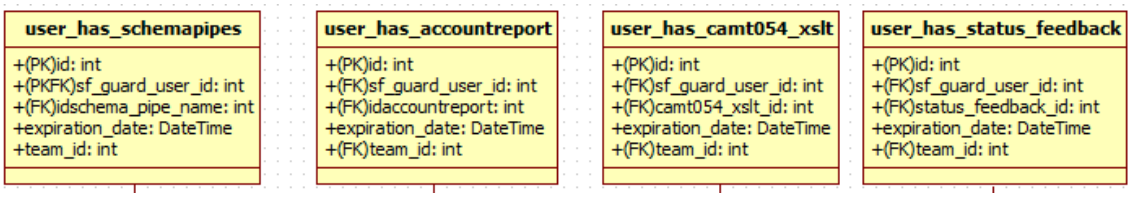

KUVIO 2 - Tiimi tietokantataulu

Käyttäjän kuuluminen tiimiin tallennetaan linkkitauluun (KUVIO 3), joka sisältää kentät sf_guard_user_id (käyttäjän tunniste) ja team_id (tiimin tunniste).



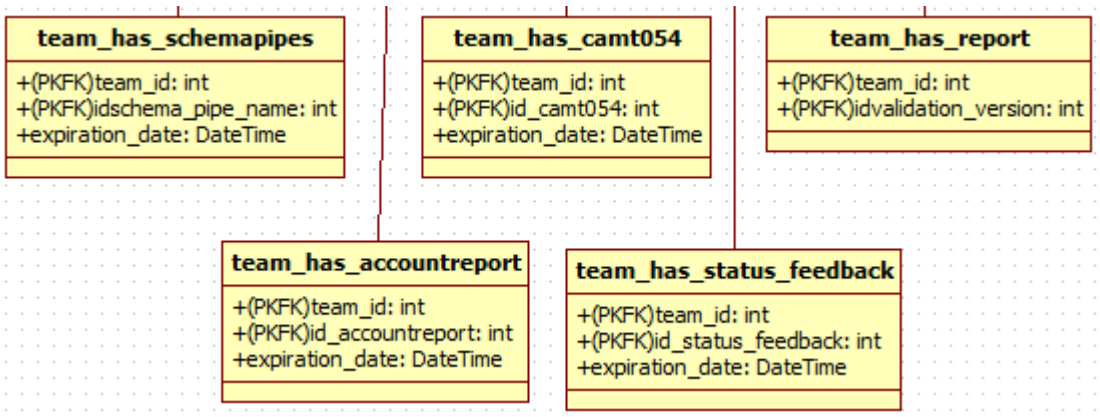

KUVIO 3 - Käyttäjän kuuluminen tiimiin

Käyttäjäkohtaiset linkkitaulut (KUVIO 4) sisältävät kentät sf_guard_user_id (käyttäjän tunniste), validointi- tai simulointiputken tunniste, expiration_date (käyttöoikeuden päättymispäivä) ja team_id (tiimin tunniste).




KUVIO 4 - Käyttäjäkohtaiset linkkitaulut

Tiimin oikeuksien linkkitaulut (KUVIO 5) sisältävät kentät team_id (tiimin tunniste), validointi- tai simulointiputken tunniste ja expiration_date (käyttöoikeuden päättymispäivä).



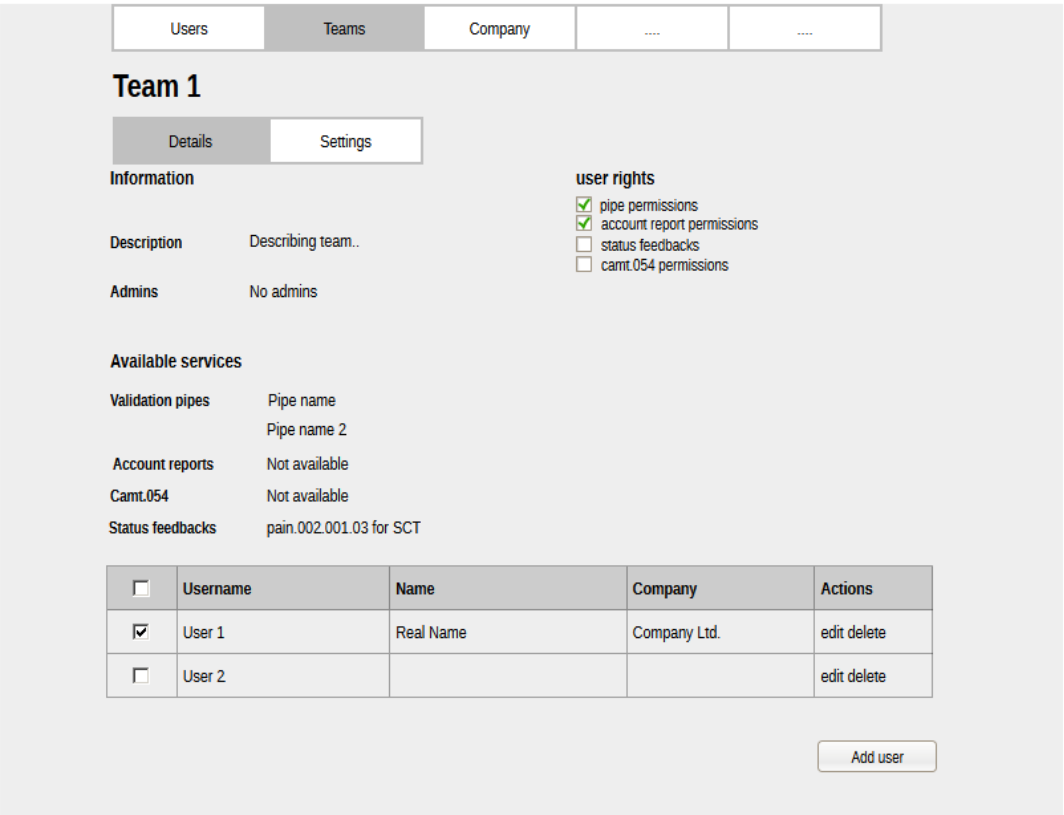

KUVIO 5 - Tiimikohtaiset linkkitaulut

### 5.1.2 Käyttöliittymä

Tiimien hallintaa varten tarvitaan käyttöliittymä. Käyttöliittymässä on näkymä, joka on erilainen riippuen siitä onko käyttäjä ”Company admin”- vai ”Team admin” -käyttäjä. Käyttäjillä, jotka ovat ”Normal”-tyyppisiä ei ole pääsyä tähän käyttöliittymään. Kun valitaan valikosta ”Teams”, avautuu näkymä, jossa listataan kaikki yrityksen tiimit tai näytetään ainoan tiimin tiedot (KUVIO 6). Tiimilista näytetään jos ”Company admin” -käyttäjän yrityksellä on yksikin tiimi tai jos ”Team admin” -käyttäjällä on hallintaoikeus useampaan kuin yhteen tiimiin.

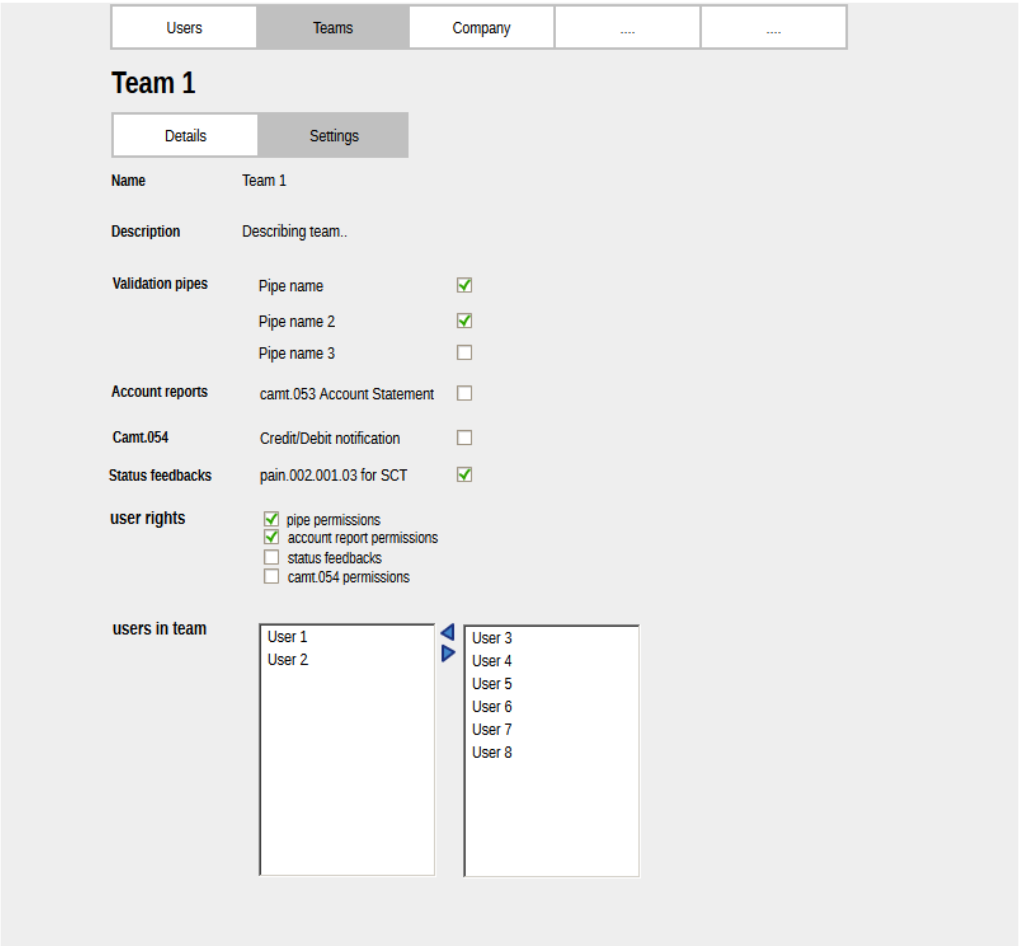

KUVIO 6 - Tiimilistasivun suunnitelma

Tiimin kuvaussivu (KUVIO 7) avautuu suoraan jos ”Team admin” -käyttäjällä on hallintaoikeus vain yhteen tiimiin. Tiimin kuvaussivulle pääsee valitsemalla listasta (KUVIO 6) tiimin kohdalta ”edit”. Kuvaussivulla näytetään yleistä tietoa tiimistä, tiimille valitut palvelut eli validointi- tai simulointiputket, sekä tiimiin kuuluvat käyttäjät. Käyttäjien tietoja voidaan muokata käyttäjälistauksen kautta.




KUVIO 7 - Tiimin kuvaussivun suunnitelma

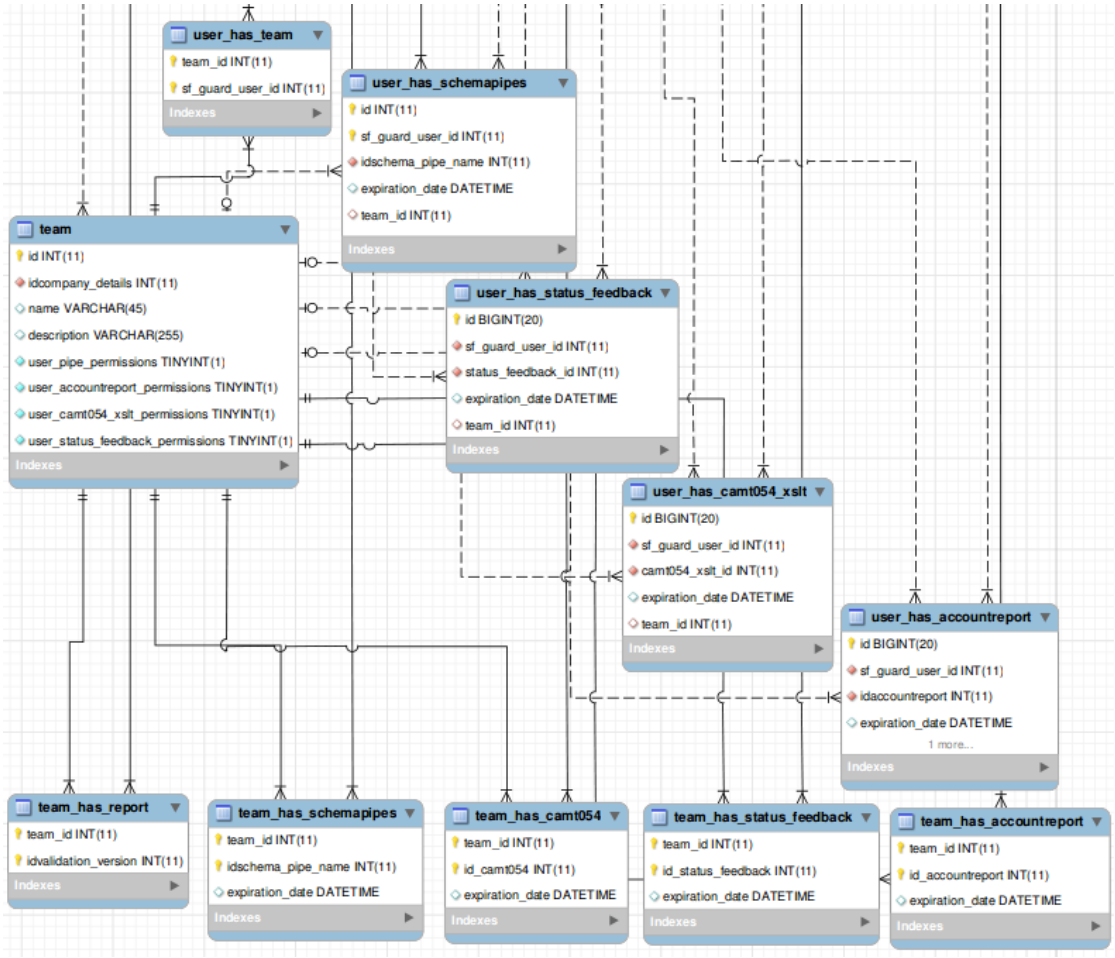
Tiimin muokkaussivulla (KUVIO 8) näytetään tiimin nimi ja kuvaus sekä tiimille valittavissa olevat oikeudet palveluihin. Sivulta voidaan myös asettaa perivätkö käyttäjät palveluoikeudet tiimin tasolta vai määrätäänkö oikeudet käyttäjille yksitellen. Muokkaussivulta voidaan myös valita tiimiin kuuluvat käyttäjät.




KUVIO 8 - Tiimin muokkaussivun suunnitelma

## 5.2 Tietokanta

Tietokantataulut luotiin suunnitelman perusteella MySQL Workbench -ohjelmalla (KUVIO 9). Tietokantamalli on vain graafinen esitys tietokannasta ennen asetusten siirtämistä tietokantaan. Tietokantamalli täytyy synkronoida MySQL-tietokannan kanssa.




KUVIO 9 - Tietokanta MySQL Workbenchin näkymässä

Valikosta valitaan ”Database/Synchronize Model”. Tältä sivulta valitaan yhteys eli kehitysympäristössä localhost. Yhteyden luomisessa tarvitaan edellä luodun tietokannan käyttäjänimi ja salasana. Tämän jälkeen valitaan MySQL Workbench:in mallitiedosto, joka halutaan synkronoida tietokannan kanssa.

Seuraavaksi näytetään olemassa olevaan tietokantaan tehtävät muutokset ja lopulta synkronoidaan skeema tietokannan kanssa eli tässä tapauksessa luodaan uudet taulut (”team”, ”user_has_team”- ja ”team_has_”-taulut) ja muokataan olemassa olevia tauluja (”user_has_”-taulut).

Tietokanta on nyt päivitetty, mutta Symfony ei ole vielä tietoinen muutoksista. Antamalla komentorivillä komento

```
php symfony propel:build-schema
```

päivitetään projektin config-kansiossa oleva schema.yml-tiedosto tietokannan rakenteen pohjalta. Liitteessä 2 on team-taulua kuvaava osio schema.yml-tiedostosta. Uutena tauluna ”Team”-taulusta luodaan annetulla komennolla kansioon ”/lib/model/om” suoraan tietokannasta generoidut kantaluokat ”BaseTeam” ja ”BaseTeamPeer”. Näitä ei ole tarkoitettu muokattavaksi. Ne generoidaan aina uudelleen kun tietokantaan tehdään muutoksia, jolloin sinne tehdyt muutokset ylikirjoittuvat. Peer-luokat sisältävät staattisia metodeja, joilla tietoja saa haettua vastaavista tauluista.

Muokattavaksi tarkoitetut luokat ”Team” ja ”TeamPeer” generoidaan kansioon ”/lib/model” kun annetaan komento:

```
php symfony propel:build-model
```

Tässä vaiheessa tietokantaa voi käsitellä ORM:n mukaisesti PHP-koodilla.

Sovellusta hallitaan web-selaimella, joten tietojen muokkaamiseen tarvitaan HTML-lomakkeita. Lomakkeiden tietojen syöttämiseen tarkoitetut kentät voidaan luoda käsin, mutta se vie paljon aikaa. Tämän lisäksi muutokset täytyisi päivittää käsin joka kerta kun tietokantaan tehdään muutoksia, jolloin myös model (malli) muuttuu. Viimeinen tarvittava komento

```
php symfony propel:build-forms
```

generoi jokaisesta tietokantataulusta luokan ja sinne validaattorin ja tietojen syöttämiseen käytetyn widgetin jokaiselle taulun sarakkeelle. Prosessi huomioi myös taulujen väliset relaatiot.

### 5.3 Moduulin luominen

Moduulin eli käytännössä uuden sivun lisääminen sivustolle tehdään Symfonyssa navigoimalla projektikansioon ja antamalla komentoriville komento:

```
php symfony generate:module frontend teams
```

Komennolla luodaan frontend-applikaatioon teams-sivu. Komento generoi kansiot ja niihin oletustiedostot:

| | |
|---|---|
| hakemisto | /apps/frontend/modules/teams/actions |
| tiedosto | /apps/frontend/modules/teams/actions/actions.class.php |
| hakemisto | /apps/frontend/modules/teams/templates |
| tiedosto | /apps/frontend/modules/teams/templates/indexSuccess.php |
| tiedosto | /test/functional/frontend/teamsActionsTest.php |

Symfony luo jokaiselle uudelle moduulille oletus ”index”-toiminnon (KUVIO 10), joka koostuu toiminto-metodista ”executeIndex” ja näkymätiedostosta ”indexSuccess.php”, joka on oletuksena tyhjä.

```php
<?php
    class teamsActions extends sfActions
    {
        public function executeIndex(sfWebRequest $request)
        {
            $this->forward('default', 'module');
        }
    }
>?
```

KUVIO 10 - Käsittelijä-luokka actions.class.php-tiedostossa

Symfonyssä voidaan nimetä metodit käyttämällä ”execute”-etuliitettä, jonka jälkeen annetaan halutun sivun nimi. Jos näkymäkansiossa tiedosto on nimetty siten, että ensin on metodin nimi pienellä alkukirjaimella ja tämän jälkeen on ”Success”, ohjaa Symfony automaattisesti käyttäjän tälle sivulle. Ohjaamalla selain osoitteeseen

http://localhost/frontend_dev.php/teams/index

avautuu edellisen esimerkin tyhjä sivu.

Työn toteutus aloitettiin tekemällä ”list”-metodi ja sille ”listSuccess.php”-tiedosto tiimilistaussivua varten. Metodin koodi on liitteessä 3. Metodille välitetään oletuksena parametrina web-pyyntö, joka sisältää HTTP-parametrit:

public function executeList( sfWebRequest $request ){}

Parametria ”request” ei kuitenkaan tarvita tässä metodissa. Seuraavassa esimerkissä ”company_id”-muuttujalle haetaan Symfonyn metodeilla ensin käyttäjäobjekti, jolla saadaan käyttöön käyttäjän hallintaan käytetyt työkalut. Tämän jälkeen kirjautuneen käyttäjän kautta saadaan haettua käyttäjän tunnisteeseen linkitetyn yrityksen tunniste. Yrityksen tunniste haetaan tietokannasta Propel:in metodilla ”getIdCompanyDetails”:

```
$company_id =
$this->getUser()->getGuardUser()->getIdCompanyDetails();
```

Criteria-luokkaa käyttämällä voidaan rakentaa tietokannan kyselyitä, jolloin tietoja ei haeta suoraan perinteisillä SQL-kyselyillä. Ensin luodaan uusi instanssi Criteria-luokasta:

```
$criteria = new Criteria();
```

Tämän jälkeen tarkistetaan onko käyttäjällä yrityksen johtajan eli ”company_admin”-oikeudet ja tämän perusteella Criteria-instanssille annetaan hakuehdot. Hakuehtoja lisätään käyttämällä ”add”-metodia, jolle annetaan parametreina halutut ehdot. Ensimmäisenä parametrina on taulussa oleva arvo ja toisena parametrina arvo, johon sitä verrataan. Esimerkissä

```
$criteria->add(TeamPeer::IDCOMPANY_DETAILS, $company_id);
```

”Team”-taulussa olevan ”idcompany_details”-kentän arvon pitää olla sama kuin edellä haettu käyttäjän yrityksen tunniste. Jos käyttäjä on tiimin johtaja eli hänellä on ”team_admin”-oikeudet, täytyy edellisen hakuehdon lisäksi tarkistaa, että tiimin johtaja kuuluu haettaviin tiimeihin:

```
$criteria->addAnd(
        UserHasTeamPeer::SF_GUARD_USER_ID,
        $this->getUser()->getGuardUser()->getId());
```

Tämän lisäksi hakuun yhdistetään ”käyttäjän tiimi” -taulusta tulokset, joissa käyttäjän tiimin tunniste vastaa taulussa olevan tiimin tunnistetta. Tällä saadaan haettua tiimin nimi, joka näytetään tiimilistassa:

```
$criteria->addJoin(TeamPeer::ID, UserHasTeamPeer::TEAM_ID);
```

Hakutulokset järjestetään tiimin tunnisteiden mukaan nousevaan järjestykseen:

```
$criteria->addAscendingOrderByColumn(TeamPeer::ID);
```

Symfonyssä on valmiina ”pager”-toiminto, jolla hakutuloksia saadaan näytettyä kerralla haluttu määrä. Tarvittaessa tulokset jakautuvat hakutulosten määrän mukaan usealle sivulle. Parametreina tälle ”sivutus”-toiminnolle annetaan tietokantataulu, jonka hakutulokset näytetään ja kerralla näytettävien tulosten määrä:

```
$pager = new sfPropelPager('team', $page_limit);
```

Sivutus-toiminnolle asetetaan edellä rakennettu, hakutulokset sisältävä Criteria-instanssi ja mahdollisesti HTTP-parametrina välittyvä viimeksi tarkastellun sivun numero, joka ”tallennetaan” istuntokohtaisesti. Sivutus alustetaan ja lopuksi muuttuja asetetaan näkyväksi näkymän puolella, jossa sen tulokset näytetään:

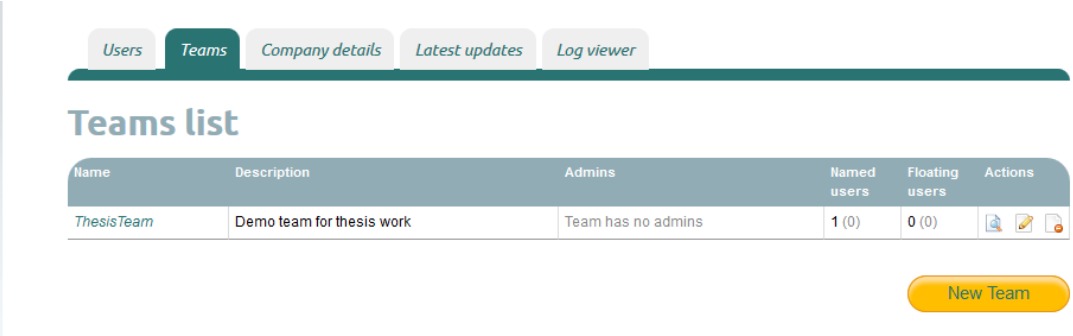
```
$pager->setCriteria($criteria);
$pager->setPage($this->getRequestParameter('page', 1));
$pager->init();
$this->pager = $pager;
```

Teams-moduulin toiminnallisuuden takia tarkistetaan edellä rakennettujen kriteerien perusteella tiimien lukumäärä ja kirjautuneen käyttäjän oikeudet. Jos käyttäjä on yrityksen johtaja, näytetään tiimilistaussivu joka kerta. Jos käyttäjä on tiimin johtaja ja hänellä on hallintaoikeus useampaan tiimiin, näytetään tiimilistaussivu. Muussa tapauksessa siirrytään suoraan ainoan tiimin muokkaussivulle:

```
$team_id = $team->getId();
if (TeamPeer::doCount($criteria) == 1 &&
!checkPermission::hasPermission('company_admin')) {
$this->redirect('teams/edit?team_id=' . $team_id);
}
```

## 5.4 Käyttöliittymä

Työn käyttöliittymä mallintaa Symfonyn MVC-mallin ”view”:iä (näkymää). Käyttöliittymä tehdään käyttämällä näkymä-tiedostoja. Näkymien muotoilu tehdään HTML-kielellä, mutta melkein kaikissa näkymätiedostoissa käytetään HTML-kielen seassa PHP-kieltä, jolla haetaan käsittelijän toiminto-luokassa asetetut arvot näytettäväksi näkymässä sekä tehdään ehto- ja silmukkarakenteet. Edellisen esimerkin toiminto-metodille tehty näkymä (KUVIO 11) on tiedostossa ”listSuccess.php”, jonka koodi on liitteessä 4.



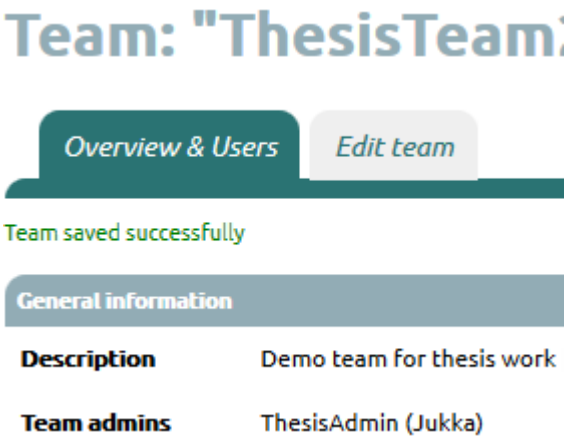

KUVIO 11 - Tiimien listaussivu

Sivulla näytetään aina otsikko ”Team list”:

```
<h1><?php echo __('Teams list') ?></h1>
```

Otsikko tulostetaan PHP-koodilla, koska sivustolla käytetään kielikäännöksiä. Teksti tulostetaan PHP:n echo-komennolla, jonka muoto on ”__(’[tulostettava teksti]’)”. Näin tulostettu teksti näkyy automaattisesti käännöstiedostossa, jolloin sille voidaan kirjoittaa käännös.

Sivulla näytetään uutta tiimiä tallennettaessa flash-viesti onnistuneesta tai virheellisestä tallennuksesta (KUVIO 12). Flash-viestit ovat yhden HTTP-pyynnön ajaksi käyttäjäistuntoon tallennettavia viestejä tai muita tietoja.




KUVIO 12 - Flash-viesti onnistuneesta tallennuksesta

Viestit asetetaan käsittelijän toiminto-luokan metodissa. Seuraava koodi asettaa kirjautuneen käyttäjän istuntoon ”success”-tunnisteelle viestin ”Team saved successfully”:

```
$this->getUser()->setFlash('success', 'Team saved successfully');
```

Näkymässä viesti haetaan käyttäjän istunnosta ja näytetään komennolla:

```
<?php echo $sf_user->getFlash('success') ?>
```

Template- eli näkymätiedostoissa voidaan käyttää osasivuja eli ”partial”-sivuja, joilla sivun rakenne voidaan jakaa erillisiin tiedostoihin. Tämä mahdollistaa esimerkiksi ehdolliset näkymien toteuttamiset, jolloin sivulle voidaan liittää haluttuja osia tiettyjen ehtojen toteutuessa. Esimerkiksi yrityksen johtajalle ja tiimin johtajalle liitetään lista-sivun runkoon erilaiset tiimi- ja käyttäjälistat. Osasivujen käyttö voi myös selkeyttää koodia oikein suunniteltuna. Osasivut liitetään komennolla:

```
include_partial(’[sivu]’, [lisäparametrit])
```

Komennolle annetaan parametreina liitettävän osasivun nimi ja vaihtoehtoiset lisäparametrit. Osasivut ovat muuten samanlaisia kuin muutkin näkymän tiedostot, mutta tiedostot nimetään liittämällä niiden eteen alaviiva (”_”). Yrityksen johtajan tiimilista-osasivu liitetään pääsivuun komennolla:

```
<?php include_partial('companyAdminList', array('pager' => $pager)) ?>
```

Osasivulle välitetään toiminto-luokan ”executeList”-metodissa määritelty pager-instanssi.

Tiimien listaussivulla on uuden tiimin tekemistä varten nappi, jonka komento on:

```
<?php echo button_to(__('New team'), 'teams/new', array('class' =>
'theme_button')) ?>
```

Metodi “button_to()” on esimerkki Symfonyn metodeista, joilla voidaan toteuttaa HTML-kielen toimintoja hieman selkeämmällä syntaksilla käyttämällä PHP-kieltä. Edellä mainittu nappi ohjaa käyttäjän ”executeNew”-metodin kautta uudelleenohjattuna ”executeEdit”-metodiin ja sen näkymään ”editSuccess.php”, joka on tiimin muokkaussivu. Muokkaussivu on kokonaisuudessaan liitteessä 7.

Tiimin muokkaussivulla käytetään Symfonyn lomake-toimintoa, joka luodaan komennolla:

```
$this->form = new TeamForm(
null,
array('sf_guard_user' => $this->getUser()->getGuardUser(),
'idcompany_details'=>$idcompany_details,)
);
```

Lomaketta luotaessa sille voidaan antaa tarvittaessa parametreja. Tässä tapauksessa annetaan ensimmäisenä parametrina lomakkeen instanssi, joka on uudella lomakkeella “null” ja toisena parametrina taulukko, jossa on käyttäjän instanssi ja yrityksen tunniste.

TeamForm-lomaketta käytetään tietojen syöttämiseen ja tallentamiseen tietokantaan. Tiimin muokkauslomakkeen luokka on liitteessä 5. Lomakeluokka sisältää oletuksena vain ”configure”-metodin:

```
class TeamForm extends BaseTeamForm{
        public function configure()    { }
}
```

Configure-metodin sisällä esitellään ja alustetaan kaikki ”widgetit”, jotka otetaan käyttöön kun lomakeluokasta luodaan uusi instanssi. Widget on tietokantataulun kentän tyypin mukainen syötetoiminto tiedolle.

Tässä tapauksessa tiimille kuuluvat raporttioikeudet (”team_has_report_list”) asetetaan pois käytöstä tai muuten uuden tiimin tallennus epäonnistuu, koska relaatiota ei vielä ole tiimin ja validointiraportin välillä:

```
unset($this['team_has_report_list']);
```

Yrityksen tunnisteesta tehdään ”sfWidgetFormInputHidden”-widget eli sen syötekenttä piilotetaan lomakkeella näkyvistä, mutta tieto siirtyy silti tallennettaessa tietokantaan ennalta määrätyllä arvolla:

```
$this->widgetSchema['idcompany_details'] =
        new sfWidgetFormInputHidden();
```

Yrityksen tunnisteelle määrätään oletusarvoksi lomakkeelle parametrina annettu tieto käyttämällä ”setDefault”-toimintoa. Lomakkeelle välitetyt tiedot saadaan haettua istunnosta ”getOption”-komennolla:

```
$this->setDefault('idcompany_details',
        $this->getOption('idcompany_details'));
```

Yrityksen tunnisteelle asetetaan lomakkeen validaattori, joka tarkistaa syötetyn arvon joko Symfonyn omalla tai itse tehdyllä validaattorilla. Tässä tapauksessa validaattori on Symfonyn validaattori, joka tarkistaa, että syötetty arvo on kokonaisluku:

```
$this->validatorSchema['idcompany_details'] = new sfValidatorInteger();
```

Tiimin nimelle asetetaan tekstin syötekenttä ”sfWidgetFormInputText”-widgetillä. Sille asetetaan validaattoriksi kustomivalidaattori ”sfValidatorStringXss”, joka on muokattu Symfonyn tekstivalidaattorista siten, että se tarkistaa ettei teksti sisällä cross-site scripting:issä (XSS) käytettyjä skriptien merkkejä:

```
$this->widgetSchema['name'] = new sfWidgetFormInputText();
$this->validatorSchema['name'] = new sfValidatorStringXss();
```

Tiimin kuvaukselle on myös tekstin syöttöalue ja validaattorina ”sfValidatorStringXss”. Validaattorille annetaan parametrina ”’required’=>false”, joka määrää ettei kyseisen kentän arvoa tarkisteta jos se on tyhjä. Merkkijonovalidaattori antaisi muuten virheen tyhjästä kentästä. Widget ja validaattori luodaan komennoilla:

```
$this->widgetSchema['description'] = new sfWidgetFormInputText();
$this->validatorSchema['description'] =
        new sfValidatorStringXss(array('required'=>false));
```

Käyttäjälistaa varten rakennetaan uusi Criteria-instanssi, jolle annetaan ehdoksi, että haettavien käyttäjien täytyy kuulua yrityksen johtajan yritykseen:

```
$criteria = new Criteria();
$criteria->add(sfGuardUserPeer::IDCOMPANY_DETAILS,
        $this->getOption('idcompany_details'));
```

Käyttäjälistaksi valitaan ”sfWidgetFormPropelChoiceMany”, jolla listasta voi valita kerralla monta käyttäjää. Listalle annetaan parametreina ”model”, edellä tehty kriteria ja sivulla näkyvä otsikko (”’label”’). Model-parametri kertoo mistä tietokannan taulusta tiedot haetaan:

```
$this->widgetSchema['user_has_team_list'] = new
sfWidgetFormPropelChoiceMany(array('model' => 'SfGuardUser',
                'criteria' => $criteria,
                'label' => 'Users in team'));
```

Käyttäjälistalle asetetaan lisäasetus ”sfWidgetFormSelectDoubleList”, joka renderöi käyttäjälistasta kaksi listaa, joiden perusteella valitaan käyttäjät:

```
$this->widgetSchema['user_has_team_list']->setOption('renderer_class',
'sfWidgetFormSelectDoubleList');
```

Lomakkeessa tarkistetaan liittyykö se uuteen vai olemassa olevaan tiimiin ja sen perusteella määrätään käytetyt kentät. Uudessa lomakkeessa käytetään kenttiä ”id”, ”idcompany_details”, ”name”, ”description” ja ”user_has_team_list”. Tiimi pitää tallentaa tietokantaan ennenkuin voidaan käyttää relaatioita taulujen välillä ja niiden widget:ejä.

Tallennetussa tiimissä otetaan käyttöön loput kentät, jotka liittyvät validointi- ja simulointiputkioikeuksiin. Käyttäjäkohtaisesti valittavat validointiputkioikeudet valitaan HTML-valintaruudulla (”checkbox”), jolle määrätään validaattoriksi boolean-validaattori (”0” tai ”1”). Muille käyttäjäoikeuksille on käytössä samanlaiset widgetit:

```
$this->widgetSchema['user_pipe_permissions'] =
        new sfWidgetFormInputCheckbox();
$this->validatorSchema['user_pipe_permissions'] =
        new sfValidatorBoolean();
```

Validointiputkioikeuksien valintaa varten tulostetaan lista. Listaa varten kootaan taulukot validointiputkien tunnisteista ja objekteista:

```
$schemapipes = CompanyDetailsPeer::retrieveByPK(
$this->getOption('idcompany_details'))->getAvailablePipes();
$schemapipe_ids = array();
$schemapipe_arr = array();
foreach($schemapipes as $object)
{
    $schemapipe_ids[] = $object->getIdschemaPipeName();
    $schemapipe_arr[$object->getIdschemaPipeName()] = $object;
}
$company = CompanyDetailsPeer::retrieveByPK(
        $this->getOption('idcompany_details'));
```

Lista rakennetaan projektin widget-kansioon tehdyssä kustomiwidgetissä ”sfWidgetFormPipesCheckboxList”, jolle välitetään parametreina edellä kootut taulukot:

```
$this->widgetSchema['team_has_schemapipes_list'] =
new sfWidgetFormPipesCheckboxList(array('type' => 'team', 'team' =>
$this->getObject(), 'choices'=> $schemapipe_arr, 'groups' => $company-
>getAvailabePipeGroups()));
```

Widgetille asetetaan sivulla näkyvä kentän nimi ja validaattori, jolla tarkistetaan, että listassa on vain sallittujen putkien tunnisteet:

```
$this->widgetSchema['team_has_schemapipes_list']->setLabel('Team ac-
tive pipes');
$this->validatorSchema['team_has_schemapipes_list'] = new sfValida-
torChoiceMany(array('required'=>false,'choices'=>$schemapipe_ids));
```

ExecuteSave-metodissa tallennetaan lomakkeelle syötetyt ja siitä valitut tiedot. Metodin koodi on liitteessä 6. Ensin tarkastetaan, että tallennusmetodiin saavutaan lomakkeelta HTML:n POST-metodin kautta eikä käyttämällä selaimen osoiteriviä. Tällä pyritään estämään vihamieliset toiminnot:

```
if($request->isMethod('post'))
```

Tiimin tunniste saadaan HTTP-parametrista ”team[id]” ja tällä haetaan tietokannasta tiimi-objekti:

```
$team_id = $request->getParameter('team[id]');
$team = TeamPeer::retrieveByPK($team_id);
```

Tallennusta varten haetaan uudestaan tiimin lomake, jolle tällä kertaa välitetään tiimi-objekti, jotta tiedot tallentuvat muokatulle tai uudelle tiimille:

```
$this->form = new TeamForm($team, array(
        'sf_guard_user' => $this->getUser()->getGuardUser(), 'id-
        company_details'=>$idcompany_details));
```

Tiimin tiedot syötetään käyttämällä HTTP:n POST-metodia. Tällöin bind-metodi sitoo lomakkeen syötettyihin tietoihin ja ajaa validointimekanismin:

```
$this->form->bind($request->getParameter($this->form->getName()));
```

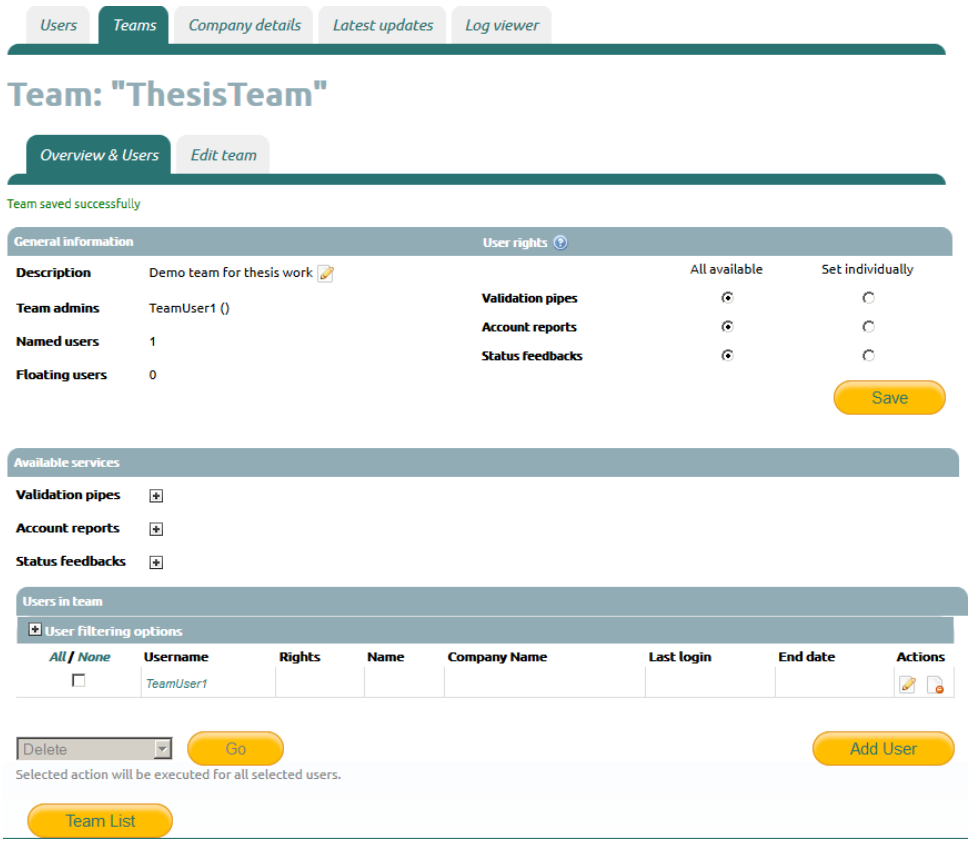
Lomake tarkastetaan ennen tallentamista. Komento ”$this->form->isValid()” tarkistaa lomakkeen sille tehdyillä validaattoreilla ja palauttaa ”true” tai ”false” riippuen siitä ovatko syötetyt tiedot hyväksyttyjä. Jos esimerkiksi jokin vaadittu tieto puuttuu, tiimin muokkaussivu jätetään näkyviin, virheelliseen kenttään annetaan lomake-luokassa määritelty vaihtoehtoinen virheilmoitus ja käyttäjälle näytetään virheilmoitus käyttämällä flash-viestiä:

```
$this->getUser()->setFlash('failure', 'Team was not saved. Form invalid.');
$this->forward('teams', 'edit');
```

Onnistuneen lomakkeen validoinnin jälkeen tarkistetaan ettei yrityksen tiimien määrä ylitä maksimimäärää. Jos tallennus sallitaan, lomakkeen tiedot tallennetaan tietokantaan komennolla:

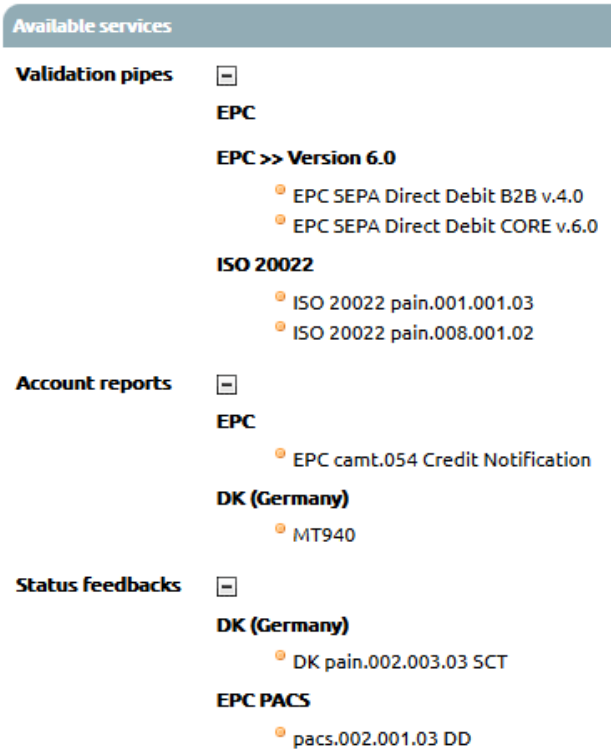
```
$this->form->save();
```

Tiimille on käytettävissä kuvaussivu (KUVIO 13), jolla näytetään tietoja tiimistä. Sivu on jaettu neljään osioon.



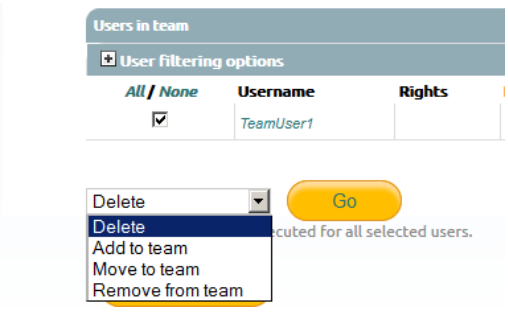

KUVIO 13 - Tiimin kuvaussivu

Yleiskuvaus (”General information”) sisältää tiimin kuvauksen, tiimin johtajat ja tiimiin jäsenten käyttäjätyyppien lukumäärät. Käyttäjäoikeudet-osio (”User rights”) sisältää valinnan käyttäjä- ja tiimikohtaisista oikeuksista. Tiimin käyttöoikeudet (”Available services”) listaa tiimille valitut validointi- ja simulointiputkioikeudet. Listat ovat oletuksena piilotetut tilan säästämiseksi ja ne voidaan avata käyttöoikeustyypin vieressä olevasta plus-napista (KUVIO 14).




KUVIO 14 - Tiimin käyttöoikeudet avattuna

Tiimin käyttäjälistaus (”Users in team”) listaa tiimin käyttäjät ja niiden tiedot. Käyttäjiä voidaan muokata ja poistaa listan ”Actions”-osiosta ja lisätä sivun ”Add user”-toiminnon kautta. Usealle käyttäjälle voidaan suorittaa toimenpiteitä joukkotoiminnon avulla (KUVIO 15). Listalta valituille käyttäjille valitaan suoritettava toiminto alasvetovalikosta. Käyttäjiä voidaan poistaa, lisätä toiseen tiimiin, siirtää toiseen tiimiin ja poistaa nykyisestä tiimistä.




KUVIO 15 - Käyttäjien joukkotoimintovalikko

# 6 POHDINTA

Työn tarkoituksena oli lisätä olemassa olevaan ja asiakkaiden käyttämään ympäristöön moduuli, jolla toteutetaan erillinen hallintataso yrityksen johtajan hallintatason ja normaalikäyttäjien välille.

Työssä onnistuttiin tavoitteiden mukaisesti ja toteutettu toiminnallisuus on tällä hetkellä asiakkaiden käytössä. Asiakkaalla on käytössä tiimi-toiminto eri maissa olevien käyttäjien tietojen ja käyttöoikeuksien hallintaan. Joitain toiminnallisuuksia tai ominaisuuksia on jo muokattu käyttökokemusten myötä. Esimerkiksi tiimin johtajan ja yrityksen johtajan näkymiä on yhtenäistetty, mutta suurempia muutoksia toiminnallisuuteen ei ole tehty ja niitä tuskin tarvitaan.

Työtä tehtäessä relaatiotietokantojen opinnoista oli huomattavasti hyötyä ja tietokantataulujen välisten suhteiden hahmottaminen edisti tietokantaosaamista. Olio-ohjelmoinnin periaatteita hyödynnettiin onnistuneesti PHP-kielellä ja web-ohjelmointi PHP:lla vaikuttaa nyt luontevalta. MVC-mallin rakenne selkiytyi Symfonyn luokkarakenteeseen tutustumisen myötä.

Merkittävänä aihepiirinä työssä oli juurikin Symfony-frameworkin käyttäminen. Framework on usein vaikeasti opittava ja ajan myötä se voi muuttua erittäin monimutkaiseksi varsinkin jos frameworkin omia toimintoja muokataan tai peräti ohitetaan omilla mukautetuilla toiminnoilla. Tätä pitäisi välttää viimeiseen asti, mutta käytännössä usein mennään siitä yli missä aita on matalin ja ongelmat korjataan myöhemmin. Ongelma ei siis liity omien luokkien periyttämiseen olemassa olevista luokista vaan valmiiden konfiguraatiotiedostojen, luokkien tai metodien muokkaamiseen.

Framework tuo silti kiistämättömiä etuja ohjelmistokehitykseen poistamalla ”ylimääräistä” kehitystyötä. Lisäksi framework on välttämätön monimutkaisen tuotantoympäristön kehityksen ja ylläpidon kannalta.

**LÄHTEET**

LAMP. Luettu 18.2.2014. http://linux.fi/wiki/LAMP

LAMP l. Linux, Apache, MySQL ja PHP asennus. Luettu 19.2.2014. http://wiki.ubuntu-fi.org/LAMP_Asennus

What is a Software Framework? And why should you like 'em? Luettu 4.3.2014. http://info.cimetrix.com/blog/bid/22339/What-is-a-Software-Framework-And-why-should-you-like-em

Symfony. What is Symfony. Luettu 5.3.2014. http://symfony.com/about

Chapter 8 - Inside The Model Layer. Why Use an ORM and an Abstraction Layer? Luettu 4.3.2014. http://symfony.com/legacy/doc/book/1_0/en/08-Inside-the-Model-Layer

Day 3: The Data Model. Luettu 4.3.2014. http://symfony.com/legacy/doc/jobeet/1_2/en/03?orm=Propel

Chapter 4 - Propel Integration. Luettu 4.3.2014. http://symfony.com/legacy/doc/forms/1_1/en/04-Propel-Integration

Chapter 4 - The Basics Of Page Creation. Luettu 10.3.2014. http://symfony.com/legacy/doc/book/1_2/en/04-The-Basics-of-Page-Creation

Symfony2 and HTTP Fundamentals. Luettu 10.3.2014. http://symfony.com/doc/current/book/http_fundamentals.html

**LIITTEET**

Liite 1. /etc/apache2/httpd.conf -tiedosto

```
<VirtualHost *:80>
        ServerName validator.localhost
        ServerAlias 192.168.1.84
        ServerAdmin webmaster@localhost
        DirectoryIndex index.php
        DocumentRoot /var/www/development/validator/current/web
        DirectoryIndex index.php
        <Directory />
                Options FollowSymLinks
                AllowOverride All
        </Directory>
        <Directory /var/www/development/validator/trunk>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride All
                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>
```

Liite 2. team-osio schema.yml-tiedostosta

```
team:
    _attributes: { phpName: Team }
    id: { phpName: Id, type: INTEGER, size: '11', primaryKey: true,
autoIncrement: true, required: true }
    idcompany_details: { phpName: IdcompanyDetails, type: INTEGER,
size: '11', required: true, foreignTable: company_details, foreignRef-
erence: idcompany_details, onDelete: CASCADE, onUpdate: CASCADE }
    name: { phpName: Name, type: VARCHAR, size: '45', required: false
}
    description: { phpName: Description, type: VARCHAR, size: '255',
required: false }
    user_pipe_permissions: { phpName: UserPipePermissions, type: TI-
NYINT, size: '1', required: true, defaultValue: '0' }
    user_accountreport_permissions: { phpName: UserAccountreportPer-
missions, type: TINYINT, size: '1', required: true, defaultValue: '0'
}
    user_camt054_xslt_permissions: { phpName: Us-
erCamt054XsltPermissions, type: TINYINT, size: '1', required: true,
defaultValue: '0' }
    user_status_feedback_permissions: { phpName: UserStatusFeedback-
Permissions, type: TINYINT, size: '1', required: true, defaultValue:
'0' }
    _indexes: { team_FK_1: [idcompany_details] }
    _uniques: { idcompany_details: [idcompany_details, name] }
```

Liite 3. action.class.php-tiedostossa oleva executeList-metodi.

```
/*
 * @param sfWebRequest $request
 */
public function executeList( sfWebRequest $request )
{
    $company_id = $this->getUser()->getGuardUser()->getIdCompanyDetails();
    $criteria = new Criteria();
    $page_limit = 10;
    if( checkPermission::hasPermission('company_admin'))
    {
        $criteria->add(TeamPeer::IDCOMPANY_DETAILS, $company_id);
    }
    else if(checkPermission::hasPermission('team_admin') )
    {
        $criteria->add(TeamPeer::IDCOMPANY_DETAILS, $company_id);
        $criteria->addAnd(UserHasTeamPeer::SF_GUARD_USER_ID, $this->getUser()->getGuardUser()->getId());
        $criteria->addJoin(TeamPeer::ID, UserHasTeamPeer::TEAM_ID);
    }
    $criteria->addAscendingOrderByColumn(TeamPeer::ID);
    $pager = new sfPropelPager('team', $page_limit);
    $pager->setCriteria($criteria);
    $pager->setPage($this->getRequestParameter('page', 1));
    $pager->init();
    $this->pager = $pager;

    // Check for redirect
    $team = TeamPeer::doSelectOne($criteria);
    if( is_object($team ))
    {
        $team_id = $team->getId();
        if (TeamPeer::doCount($criteria) == 1 && !checkPermission::hasPermission('company_admin')) {
            $this->redirect('teams/edit?team_id=' . $team_id);
        }
    }
}
```

Liite 4. listSuccess.php tiedostossa oleva näkymän toteutus.

```
<?php if(checkPermission::hasPermission('company_admin') || $sf_user->getGuardUser()->getIsSuperAdmin()): ?>
    <?php include_partial('company/mainMenu') ?>
<?php endif ?>
<div id="sf_admin_container" style="width: 100%;" >
    <div id="report_container" style="width: 100%;" >
        <div id="report_content">
            <h1><?php echo __('Teams list') ?></h1>
            <?php if ($sf_user->hasFlash('success')): ?>
                <div class="success"><?php echo $sf_user->getFlash('success') ?></div></br>
            <?php endif ?>
            <?php if ($sf_user->hasFlash('failure')): ?>
                <div class="failure"><?php echo $sf_user->getFlash('failure') ?></div></br>
            <?php endif ?>

            <?php if (checkPermission::hasPermission('company_admin')): ?>
                <?php include_partial('companyAdminList', array('pager' => $pager)) ?>
                <div style="float: right; padding-top: 20px;">
                    <?php echo button_to(__('New team'), 'teams/new', array('class' => 'theme_button')) ?>
                </div>
            <?php elseif (checkPermission::hasPermission('team_admin')): ?>
                <?php include_partial('teamAdminList', array('pager' => $pager)) ?>
            <?php endif ?>
        </div>
    </div>
</div>
```

Liite 5. Team-taulun muokkauslomake ”TeamForm.class.php” 1(2)

```
class TeamForm extends BaseTeamForm
{
    public function configure()
    {
        unset($this['team_has_report_list']);
        $this->widgetSchema['idcompany_details'] = new sfWidgetFormInputHidden();
        $this->setDefault('idcompany_details', $this->getOption('idcompany_details'));
        $this->validatorSchema['idcompany_details'] = new sfValidatorInteger();

        $this->widgetSchema['name'] = new sfWidgetFormInputText();
        $this->validatorSchema['name'] = new sfValidatorStringXss();
        $this->widgetSchema['description'] = new sfWidgetFormInputText();
        $this->validatorSchema['description'] = new sfValidatorStringXss(array('required'=>false));

        $criteria = new Criteria();
        $criteria->add(sfGuardUserPeer::IDCOMPANY_DETAILS, $this->getOption('idcompany_details'));
        $this->widgetSchema['user_has_team_list'] = new sfWidgetFormPropelChoiceMany(array(
            'model' => 'SfGuardUser',
            'criteria' => $criteria,
            'label' => 'Users in team')
            );
        $this->widgetSchema['user_has_team_list']->setOption('renderer_class', 'sfWidgetFormSelectDoubleList');

        if($this->isNew())
        {
            $this->useFields(array('id','idcompany_details','name','description','user_has_team_list'));
        }else
        {
            $this->widgetSchema['user_pipe_permissions'] = new sfWidgetFormInputCheckbox();
            $this->validatorSchema['user_pipe_permissions'] = new sfValidatorBoolean();
            $this->widgetSchema['user_accountreport_permissions'] = new sfWidgetFormInputCheckbox();
            $this->validatorSchema['user_accountreport_permissions'] = new sfValidatorBoolean();
            $this->widgetSchema['user_camt054_xslt_permissions'] = new sfWidgetFormInputCheckbox();
            $this->validatorSchema['user_camt054_xslt_permissions'] = new sfValidatorBoolean();
            $this->widgetSchema['user_status_feedback_permissions'] = new sfWidgetFormInputCheckbox();
            $this->validatorSchema['user_status_feedback_permissions'] = new sfValidatorBoolean();

            $status_feedbacks = CompanyDetailsPeer::retrieveByPK(
                                $this->getOption('idcompany_details'))->getAvailableStatusFeedbacks();
            $status_feedback_ids = array();
            $status_feedbacks_arr = array();
            foreach($status_feedbacks as $feedback)
            {
              $status_feedback_ids[] = $feedback->getId();
              $status_feedbacks_arr[$feedback->getId()] = $feedback;
            }
            $this->widgetSchema['team_has_status_feedback_list'] =
                new sfWidgetFormStatusFeedbackCheckboxList(array(
                    'choices' => $status_feedbacks_arr,
                    'label' => 'Team Status Feedbacks',
                    'type' => 'team',
                    'team' => $this->getObject())
                    );
            $this->validatorSchema['team_has_status_feedback_list'] =
                new sfValidatorChoiceMany(array(
                    'required'=>false,
                    'choices'=>$status_feedback_ids)
                    );
            $camt054_xslts = CompanyDetailsPeer::retrieveByPK(
                                $this->getOption('idcompany_details'))->getAvailableCamt054Xslts();
            $camt054_xslt_ids = array();
            $camt054_xslt_arr = array();
```

```
            foreach($camt054_xslts as $object)
            {
              $camt054_xslt_ids[] = $object->getId();
              $camt054_xslt_arr[$object->getId()] = $object;
            }
            $this->widgetSchema['team_has_camt054_list'] =
                new sfWidgetFormCamt054XsltCheckboxList(array(
                    'choices' => $camt054_xslt_arr,
                    'label' => 'Team Camt.054',
                    'type' => 'team',
                    'team' => $this->getObject())
                    );
            $this->validatorSchema['team_has_camt054_list'] =
                new sfValidatorChoiceMany(array(
                    'required'=>false,
                    'choices'=>$camt054_xslt_ids)
                    );

            $accountreports = CompanyDetailsPeer::retrieveByPK(
                        $this->getOption('idcompany_details'))->getAvailableAccountreports();
            $accountreport_ids = array();
            $accountreport_arr = array();
            foreach($accountreports as $object)
            {
              $accountreport_ids[] = $object->getId();
              $accountreport_arr[$object->getId()] = $object;
            }
            $this->widgetSchema['team_has_accountreport_list'] =
                new sfWidgetFormAccountReportCheckboxList(array(
                    'choices'=>$accountreport_arr,
                    'label' =>'Team Account Reports',
                    'type' => 'team',
                    'team' => $this->getObject())
                    );
            $this->validatorSchema['team_has_accountreport_list'] =
                new sfValidatorChoiceMany(array(
                    'required'=>false,
                    'choices'=>$accountreport_ids)
                    );

            $schemapipes = CompanyDetailsPeer::retrieveByPK(
                        $this->getOption('idcompany_details'))->getAvailablePipes();
            $schemapipe_ids = array();
            $schemapipe_arr = array();
            foreach($schemapipes as $object)
            {
              $schemapipe_ids[] = $object->getIdschemaPipeName();
              $schemapipe_arr[$object->getIdschemaPipeName()] = $object;
            }

            $company = CompanyDetailsPeer::retrieveByPK($this->getOption('idcompany_details'));
            $this->widgetSchema['team_has_schemapipes_list'] =
                new sfWidgetFormPipesCheckboxList(array(
                    'type' => 'team',
                    'team' => $this->getObject(),
                    'choices'=> $schemapipe_arr,
                    'groups' => $company->getAvailabePipeGroups())
                    );
            $this->widgetSchema['team_has_schemapipes_list']->setLabel('Team active pipes');
            $this->validatorSchema['team_has_schemapipes_list'] =
                new sfValidatorChoiceMany(array(
                    'required'=>false,
                    'choices'=>$schemapipe_ids)
                    );
        }
    }
}
```

Liite 6. action.class.php-tiedostossa oleva executeSave-metodi.

```
public function executeSave(sfWebRequest $request)
{
    $idcompany_details = $this->getUser()->getGuardUser()->getIdCompanyDetails();
    $criteria = new Criteria();
    $criteria->add(TeamPeer::IDCOMPANY_DETAILS, $idcompany_details);
    $criteria->addJoin(TeamPeer::ID, UserHasTeamPeer::TEAM_ID);
    $this->team_count = TeamPeer::doCount($criteria);

    if($request->isMethod('post'))
    {
        $idcompany_details = $this->getUser()->getGuardUser()->getIdCompanyDetails();
        $team_id = $request->getParameter('team[id]');
        $team = TeamPeer::retrieveByPK($team_id);
        $this->team = $team;
        $this->form = new TeamForm($team, array(
                                    'sf_guard_user' => $this->getUser()->getGuardUser(),
                                    'idcompany_details'=>$idcompany_details));

        if( checkPermission::hasPermission('company_admin') == false || $request->getParameter('edit_description', false))
        {   // Use only fields that team admin can see and edit
            $this->form->useFields(array(
                                    'id',
                                    'idcompany_details',
                                    'description',
                                    'user_pipe_permissions',
                                    'user_accountreport_permissions',
                                    'user_camt054_xslt_permissions',
                                    'user_status_feedback_permissions'));
        }

        $this->form->bind($request->getParameter($this->form->getName()));

        if( $this->form->isValid())
        {
            $company = CompanyDetailsPeer::retrieveByPK($idcompany_details);
            if($this->form->isNew() && $company->getTeamLimit() <= count($company->getTeams()))
            {
                $this->getUser()->setFlash('failure', 'Team was not saved. Team limit is full.');
                $this->forward('teams', 'list');
            }
            $this->form->save();
            $this->getUser()->setFlash('success', 'Team saved successfully');
            $team_id = $this->form->getObject()->getId();
            if(checkPermission::hasPermission('company_admin',$this->getUser()->getGuardUser()->getId()))
            {
                $this->redirect('teams/view?team_id='.$team_id);
            }else
            {
                $this->redirect('teams/edit?team_id='.$team_id);
            }
        }else
        {
            $this->getUser()->setFlash('failure', 'Team was not saved. Form invalid.');
            $this->forward('teams', 'edit');
        }
    }
    else
    {
        $this->getUser()->setFlash('failure', 'Team was not saved');
        $this->forward('teams', 'list');
    }
}
```

Liite 7. Tiimin muokkaussivu.