OULUN AMMATTIKORKEAKOULU

Topi Valtanen

Hankintaprosessin digitalisointi

Käyttäjälähtöinen web-portaali

OULUN AMMATTIKORKEAKOULU

Hankintaprosessin digitalisointi

Käyttäjälähtöinen web-portaali

Topi Valtanen
Opinnäytetyö
Kevät 2020
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä(t): Topi Valtanen
Opinnäytetyön nimi: Hankintaprosessin digitalisointi – Käyttäjälähtöinen web-portaali
Työn ohjaaja: Teppo Räisänen
Työn valmistumislukukausi ja -vuosi: Kevät 2020 Sivumäärä: 50 + 1

Opinnäytetyön aiheena oli toteuttaa web-sovellus Oulun Ammattikorkeakoululle hankintaprosessin tueksi. Tavoitteena oli yhtenäistää hankintaprosessia siten, että jatkossa kaikki hankintapyynnöt tapahtuisivat yhdessä paikassa samojen käytäntöjen mukaisesti.

Sovellus toteutettiin PHP:llä käyttäen Symfony 5 -sovelluskehystä. Käyttöliittymä suunniteltiin mukautumaan erityyppisille laitteille käyttäen Bootstrap 4 -kehystä. Näiden lisäksi sovelluksen toteutukseen käytettiin perinteisiä web-ohjelmoinnin työvälineitä kuten HTML, CSS, JavaScript, AJAX, JSON sekä JQuery.

Teoriaosuudessa tutustutaan Symfonyyn ja joihinkin sen keskeisiin perusperiaatteisiin. Tämän lisäksi kerrotaan lyhyesti muista käytetyistä tekniikoista.

Työn tuloksena syntyi käyttövalmis sovellus, jota ei kuitenkaan ole vielä otettu käyttöön. Sovelluksen käyttöönotto tapahtuu kesän lopussa, ennen syyslukukauden alkua, tai pian sen jälkeen.

Asiasanat: web-sovellus, PHP, MVC, Symfony

ABSTRACT

Oulu University of Applied Sciences
Degree programme in Business Information Systems

Author(s): Topi Valtanen
Title of thesis: Digitalization of Acquisition Process – User-oriented web-portal
Supervisor(s): Teppo Räisänen
Term and year when the thesis was submitted: Spring 2020 Number of pages: 50 + 1

The subject of the thesis was to create a web application for Oulu University of Applied Sciences to support purchasing process. The aim was to unify the purchasing process so that in future all purchase requests would take place in same place, following the same practices.

The application was implemented in PHP using the Symfony 5 application framework. The interface was designed to adapt to different kind of devices using Bootstrap 4 framework. In addition to these, several classic web programming languages and techniques were used to create the application, such as HTML, CSS, JavaScript, AJAX, JSON and Jquery.

The theoretical part introduces the Symfony and some of its key principles. In addition, a brief description of the other techniques is provided.

The work resulted in a ready-to-use application, which, however, has not yet been taken into use. The application will be deployed at the end of the summer, before the beginning of autumn semester, or shortly thereafter.

Keywords: web application, PHP, MVC, Symfony

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheena oli websovelluksen tekeminen Oulun Ammattikorkeakoululle hankintaprosessin tueksi. Aikaisemmin hankintoihin on liittynyt kirjavia käytäntöjä, esimerkiksi jonkin laitteen hankinta on voinut lähteä alulle sähköpostitse, puhelimella tai keskustelun pohjalta. Tavoitteena oli yhtenäistää hankintaprosessia siten, että jatkossa kaikki hankinnat tapahtuisivat yhdessä paikassa samojen käytäntöjen mukaisesti. Sovellus tuli toteuttaa mukautuvasti siten, että se on käytettävissä niin pöytäkoneilla, tableteilla kuin älypuhelimillakin.

Sovellus toteutettiin PHP:llä käyttäen komponenttipohjaista MVC-sovelluskehystä Symfony, jonka uusi viitosversio oli julkaistu hieman ennen työn aloittamista. Tietokantaa käytetään Doctrine ORM työvälineen avulla ja sivujen pohjat on tehty käyttäen Twigiä. Käyttöliittymän mukautuvuus eri laitteille toteutettiin käyttäen Bootstrap 4 -kehystä. Näiden lisäksi käytettiin hieman JavaScriptiä ja AJAX:ia parantamaan käytettävyyttä.

Luvussa kolme tutustutaan Symfonyyn, Doctrineen ja Twigiin. Ensin kerrotaan hieman Symfonysta ja MVC-arkkitehtuurista yleisellä tasolla, sen jälkeen mennään pintaa syvemmälle, menemättä kuitenkaan liikaa yksityiskohtiin. Lopuksi tutustutaan vielä Doctrineen ja Twigiin. Luvun tarkoituksena on antaa lukijalle jonkinlainen käsitys Symfonyn toimintaperiaatteista, sekä sen kanssa usein käytetyistä Doctrinesta että Twigistä.

Luku neljä sisältää lyhyet kuvaukset muista sovelluksen tekoon käytetyistä kielistä ja tekniikoista. Muista käytetyistä tekniikoista tärkeimmät ovat PHP, HTML, CSS sekä Bootstrap. Sovelluksessa hyödynnetään myös Javascriptiä ja Ajaxia, mutta niiden rooli sovelluksessa on pieni ja sovellus toimii myös ilman niitä.

Luvussa viisi tutustutaan toteutettuun sovellukseen sekä tärkeimpiin valittuihin ratkaisuihin. Luku jakaantuu siten, että ensin katsotaan kuinka sovellus on rakentunut MVC-mallin mukaisesti. Sen jälkeen tutustutaan toimintaperiaatteeseen, Symfonyn Workflow-komponenttiin ja kuinka sovelluksen käyttämä prosessi on mallinnettu sen avulla. Tämän jälkeen paneudutaan käyttäjiin ja käyttöoikeuksiin, katsotaan joitakin esimerkkejä käyttöliittymästä sekä sen mukautuvuudesta eri kokoisille näytöille. Lopuksi tutustutaan sähköpostiviestien lähettämiseen asynkronisesti käyttäen Symfonyn Mailer ja Messenger komponentteja.

2 VAATIMUKSET

Tässä luvussa kerrotaan tärkeimmät sovellukselle yleisesti asetetut vaatimukset, jotka perustuvat toimeksiantajan toimittamaan prosessikaavioon (Liite 1) ja ensimmäisessä keskustelussa sovittuihin asioihin. Sovellus päätettiin toteuttaa ketterästi ja vaatimukset tarkentuivat työn edetessä.

- Toteutuskieleksi valittiin PHP, koska toimeksiantajan on tarkoitus jatkokehittää sovellusta ja toimeksiantajalta löytyy siihen omaa osaamista.
- Sovelluksen tulee toimia responsiivisesti. Responsiivisuudella tarkoitetaan käyttöliittymän mukautumista erilaisille päätelaitteille siten, että se on käyttökelpoinen esimerkiksi älypuhelimilla, tableteilla ja pöytätietokoneilla.
- Käyttäjä voi tehdä hankintapyynnön, joka kuuluu joko ennalta luotuun kategoriaan tai käyttäjän syöttämään vapaavalintaiseen kategoriaan.
- Hankintapyynnön tiedot kuten laitteet, liittymät ja lisenssit syötetään vapaamuotoiseen tekstikenttään.
- Hankinnan tiedot kuten hanke-, projekti- ja investointinumerot, lisätiedot ja perustelut syötetään vapaamuotoiseen tekstikenttään.
- Hankintapyyntöön on mahdollista lisätä liitetiedostoja.
- Valmis hankintapyyntö lähetetään lähimmän esimiehen hyväksyttäväksi.
- Esimies voi joko hyväksyä, hylätä tai palauttaa hankintapyynnön täydennettäväksi.
- Esimiehen hyväksyttyä hankintapyynnön, IT-palvelut voi tehdä joko myönteisen tai kielteisen hankintapäätöksen.
- Hankintapyyntöön voi liittää keskustelua, johon voi osallistua käyttäjä, esimies sekä IT-palvelut.
- Sekä myönteisestä että kielteisestä hankintapäätöksestä lähetetään ilmoitus hankintapyynnön tekijälle sähköpostitse.
- Loppuun käsitellyt hankintapyynnöt arkistoidaan eikä niitä voi enää muokata.

3 SYMFONY 5

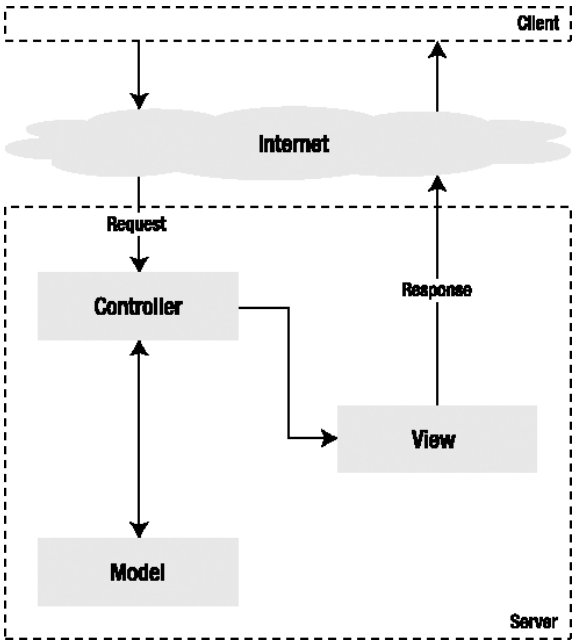
Symfony on avoimen lähdekoodin PHP websovelluskehys. Se koostuu uudelleenkäytettävistä PHP komponenteista, joita voi käyttää yhdessä sovelluskehyksen kanssa tai irrallaan siitä. Symfony on hyvin joustava, se on kaiken tarjoava sovelluskehys (englanniksi full stack framework), josta voit käyttää vain niitä osia, joita tarvitset, ja se taipuu niin pieniin mikrosivustoihin kuin suuriin yrityssovelluksiinkin. (Tutorials Point 2020b, viitattu 11.6.2020.)

3.1 Taustaa

Ensimmäinen versio Symfonystä julkaistiin vuonna 2005. Fabien Potencier, ranskalaisen Sension toimitusjohtaja ja Symfonyn tekijä, rakensi Symfonyn alun perin Sension sisäisiä projekteja varten. Käytettyään Symfonyä onnistuneesti muutamissa projekteissa, Fabien päätti julkaista sen avoimen lähdekoodin lisenssillä. (Zaninotto & Potencier 2007a.) Symfony käyttää sallivaa MIT lisenssiä (Symfony 2020k, viitattu 11.6.2020).

3.2 Arkkitehtuuri

Symfony perustuu MVC-arkkitehtuuriin. MVC tulee sanoista Model-View-Controller eli Malli-Näkymä-Käsittelijä. Se on suunnittelumalli, joka erottaa sovelluslogiikan ja esitystavan toisistaan ylläpidettävyyden parantamiseksi. Siinä sovellus jaetaan nimensä mukaisesti kolmeen eri tasoon: malliin, näkymään ja käsittelijään. Malli edustaa sovelluksen käsittelemiä tietoja – sen sovelluslogiikkaa, näkymää huolehtii mallin esittämisestä websivuna käyttäjän kanssa vuorovaikutukselle sopivalla tavalla ja käsittelijä vastaa käyttäjän toimintoihin sekä tekee muutospyyntöjä malliin tai näkymään tarpeen mukaan. Kuvio 1 havainnollistaa MVC-mallia. (Zaninotto & Potencier 2007b, viitattu 21.4.2020.)



KUVIO 1: MVC malli (Zaninotto & Potencier 2007b, Viitattu 21.4.2020)

3.3 Lyhyt katsaus ytimeen ja keskeisiin osiin

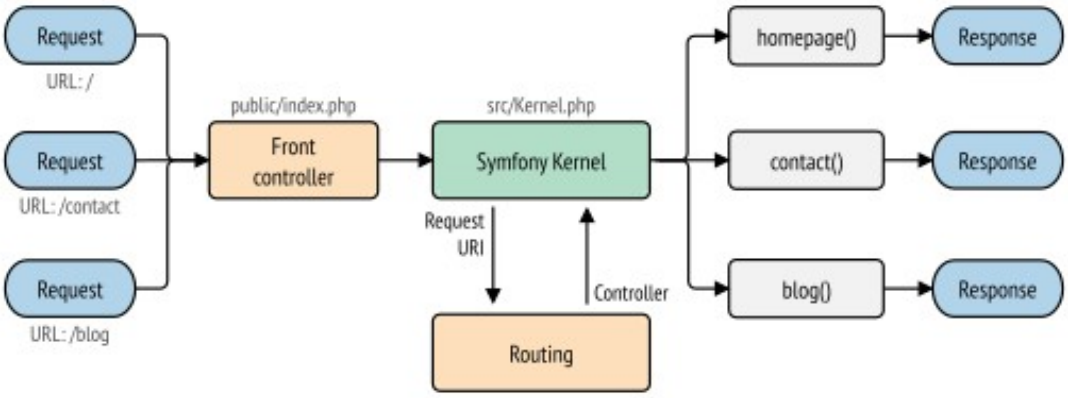
Tässä alaluvussa tutustutaan lyhyesti Symfonyn ytimeen ja joihinkin sen keskeisiin osiin sekä tekniikoihin. Tämä ei ole missään nimessä kattava kuvaus ominaisuuksista, mutta se auttaa ymmärtämään Symfonyn toimintaa.

3.3.1 Front Controller

Symfony käyttää niin kutsuttua Front Controller -suunnittelumallia. Front Controller on se osa sovellusta, jonka kautta kaikki pyynnöt kulkevat. Symfonyssä sen roolissa toimii public/ hakemistossa oleva index.php tiedosto, ja se on Symfony sovelluksessa jokaiselle pyynnölle ensimmäisenä suoritettava PHP-tiedosto. (Symfony 2020t, viitattu 11.7.2020.)

Front Controllerin päätarkoitus on luoda ilmentymä Kernel-objektista, joka on Symfonyn ydin, saada se käsittelemään pyyntö ja palauttamaan vastaus selaimelle. Koska kaikki pyynnöt kulkevat sen läpi, siellä voidaan tehdä globaaleja alustustoimenpiteitä ennen kernelin asettamista tai koristella se lisäominaisuuksilla, kuten autoloader-määrityksillä, HTTP tason välimuistilla tai ottamalla käyttöön debug-komponentin. (Symfony 2020t, viitattu 11.7.2020.)

Kun Front Controller saa pyynnön, se alustaa kernelin, ja antaa sille Request objektin käsiteltäväksi. Symfony pyytää reitittäjää (router) tutkimaan pyyntöä. Reitittäjä kohdistaa tulevan URL-osoitteen tiettyyn reittiin (route) ja palauttaa tiedot reitistä, mukaan lukien käsittelijän, jonka käsiteltäväksi se kuuluu. Tämän jälkeen suoritetaan oikea käsittelijä, joka sitten luo ja palauttaa sopivan Response-objektin. Kuvio 2 havainnollistaa pyynnön käsittelyä. (Symfony 2020m, viitattu 11.7.2020.)



KUVIO 2: Pyynnön käsittely (Symfony 2020m, viitattu 11.7.2020)

3.3.2 Kernel

Kernel on Symfonyn ydin. Sen vastuulla on kaikki sovelluksen käyttämät bundlet ja tarjota niille sovelluksen määritykset. Se luo Service Containerin ennen pyyntöjen käsittelyä. Symfonyn Kernel on abstrakti luokka ja se sisältää muutaman metodin, jotka on toteutettava, ja ne ovat: registerBundles, configureRouters sekä configureContainer. Symfony tarjoaa oletustoteutuksen src/Kernel.php, joka on mukautettavissa omiin tarpeisiin. Tarpeen mukaan sovelluksessa voi olla myös useampia kerneleitä. (Symfony 2020t, viitattu 11.7.2020.)

3.3.3 Reititys

Reitityksen tehtävä on kohdistaa sivupyynnöt oikeille käsittelijöille. Se tarjoaa myös muita hyödyllisiä ominaisuuksia kuten hakukoneoptimoinnille, eli Search Engine Optimization (SEO), ystävälliset URL-osoitteet. Reitit ykset voidaan määritellä Symfonyssä useammalla eri tavalla. Symfony kuitenkin suosittelee käyttämään niin kutsuttuja annotaatioita, koska on kätevää pitää reitti ja käsittelijä samassa paikassa. Kuviossa 3 on esimerkki reitityksen määrittämisestä käyttäen annotaatioita. (Symfony 2020g, viitattu 12.7.2020.)

```php
// src/Controller/BlogController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class BlogController extends AbstractController
{
    /**
     * @Route("/blog", name="blog_list")
     */
    public function list()
    {
        // ...
    }
}
```

KUVIO 3: Esimerkki reitityksen määrittämisestä käyttäen annotaatioita (Symfony 2020g, viitattu 12.7.2020)

3.3.4 Service Container ja autowiring

Service Container -objekti on Symfonyn tapa keskittää sovelluksessa käytettävien palveluiden muodostaminen yhteen paikkaan ja helpottaa niiden käyttöä. Sen käyttöön liittyy paljon automatiikkaa – sen avulla Symfony osaa tarjota automaattisesti oikean palvelun käyttämällä tyyppivihjeitä, jolloin Symfony tunnistaa palvelun luokan nimestä tai rajapinnasta, muodostaa objektin ja injektoi sen käytettäväksi. Symfonyn termein tätä automatiikkaa kutsutaan nimellä Autowiring. (Symfony 2020j, viitattu 12.7.2020.)

Autowiring toimii siten, että se etsii palvelun, jonka id sopii täsmälleen annettuun tyyppivihjeeseen. Ellei sellaista löydy, heitetään poikkeus. Symfony ei yritä arvailla vaan se pyrkii olemaan niin selkeä ja ennakoitava kuin mahdollista. (Symfony 2020c, viitattu 12.7.2020.)

Käytettäessä abstrahointeja, kuten rajapintoja, autowiring ei pysty määrittämään käytettäviä palveluita automaattisesti, koska palvelun id ei sovi täsmällisesti tyyppivihjeeseen. Tällöin käytettävät luokat tulee kertoa sille. Tämä tapahtuu config/services.yaml konfiguraatiotiedostossa lisäämällä palvelulle alias, jonka avulla autowiring osaa käyttää oikeaa luokkaa. Samalla periaatteella

voidaan erotella myös useat samaa rajapintaa käyttävät palvelut toisistaan. (Symfony 2020c, viitattu 12.7.2020.)

3.3.5 Ympäristöt ja asetukset

Symfony tukee useita eri ympäristöjä, joilla jokaisella on omat asetuksensa. Tästä on etua, koska sovelluksen on hyödyllistä mukautua käytön mukaan. Esimerkiksi sovellusta kehitettäessä on hyödyllistä saada käyttöön debug-työvälineet ja tarkemmat lokitiedostot. Tuotannossa tärkeää on suoritusnopeuden optimointi ja lokitiedostoihin halutaan vain virheet. Oletuksena Symfony tarjoaa kolme ympäristöä, jotka ovat: kehitys (dev), tuotanto (prod) ja testaus (test), ja niitä voi tarvittaessa tehdä lisää. Esimerkki tällaisesta voisi olla niin kutsuttu staging-ympäristö, jossa sovellusta voi testata tuotantoympäristöä vastaavassa ympäristössä ennen muutosten siirtämistä varsinaiseen tuotantoympäristöön. (Symfony 2020a, viitattu 14.7.2020.)

Asetustiedostojen määrittelyyn Symfony tarjoaa kolme eri vaihtoehtoa, joista voi valita vapaasti, ja ne ovat: YAML, XML ja PHP. Asetustiedostot sijaitsevat kansiossa config/packages/ ja ympäristökohtaiset asetukset oman nimisissä alakansioissaan. Esimerkiksi kehitysympäristölle yksilölliset asetukset löytyvät kansiosta config/packages/dev/. Asetukset ladataan tiedostoista siten, että ensin ladataan kaikille ympäristöille yhteiset asetukset ja sen jälkeen ladataan ympäristökohtaiset asetukset. Ympäristökohtaiset asetukset voivat korvata osan yhteisistä asetuksista. Yleensä vain pieni osa asetuksista sijaitsee ympäristökohtaisissa asetuksissa. (Symfony 2020a, viitattu 14.7.2020.)

Näiden asetustiedostojen lisäksi, osa asetuksista, kuten tietokantayhteydet määritellään ympäristömuuttujilla. Sen sijaan, että käytettäisiin suoraan palvelimen omia ympäristömuuttujia, Symfony tarjoaa mahdollisuuden ympäristömuuttujien asettamiseen sovelluksen juuressa sijaitsevilla .env-tiedostoilla. Env-tiedostot noudattavat samaa mallia kuin muutkin asetukset, ja on mahdollista määritellä kaikille yhteiset asetukset sekä ympäristökohtaiset asetukset erikseen omissa tiedostoissaan. Tiedosto .env sisältää kaikille yhteiset asetukset ja .env.<ympäristö> sisältää ympäristökohtaiset asetukset. Esimerkiksi tiedosto .env.test sisältää testiympäristölle yksilöllisiä asetuksia. Näiden lisäksi on mahdollista määritellä konekohtaisia .env.local ja .env.<ympäristö>.local tiedostoja, joilla voidaan korvata osa asetuksista, pois lukien testiympäristö, jossa toiminnan tulee olla aina sama, riippumatta missä se suoritetaan. Huomioitavaa on myös se, että aidot ympäristö-

muuttujat voittavat aina .env-tiedostoissa määritellyt arvot, eikä niitä voi niissä ylikirjoittaa. (Symfony 2020a, viitattu 14.7.2020.)

Käytettävän ympäristön valinta tapahtuu sovelluksen juuressa sijaitsevassa .env tai .env.local -tiedostossa määrittelemällä ympäristömuuttuja ”APP_ENV” vastaamaan käytettävän ympäristön nimeä. Esimerkiksi rivi ”APP_ENV=prod” saa sovelluksen käyttämään tuotantoympäristön asetuksia. (Symfony 2020a, viitattu 14.7.2020.)

3.4 Komponentit ja Bundlet

Symfony komponentit ovat uudelleenkäytettäviä PHP-kirjastoja, jotka toteuttavat usein tarvittavia ominaisuuksia. Niitä voi käyttää omissa sovelluksissa myös ilman Symfony-kehystä. (Symfony 2020l, viitattu 6.7.2020.)

Bundlet ovat verrattavissa liitännäisiin. Symfonyn ydinominaisuudet on toteutettu niitä käyttäen ja lisää ominaisuuksia voi saada käyttämällä kolmansien osapuolten Bundleja. Bundlet ovat komponentteja laajempi kokonaisuus, joilla on oma kansiorakenteensa, ja ne voivat sisältää pelkän PHP-koodin lisäksi vaikkapa Twig-templaatteja, tyylitiedostoja, kuvia ja konfiguraatiotiedostoja. Ennen Symfony 4.0 versiota, suositeltava tapa oli järjestää sovellus Bundleihin, mutta nykyisin niitä suositellaan käyttämään vain sellaiselle koodille ja ominaisuuksille, jotka on tarkoitus jakaa useiden eri sovellusten kesken. (Symfony 2020n, viitattu 6.7.2020.)

PHP-pakettien asentaminen ja hallinta tapahtuu käyttäen Composeria (Symfony 2020e, viitattu 6.7.2020). Composer on työväline riippuvuuksien hallintaan PHP:llä. Se asentaa PHP-paketit projektikohtaisesti vendor kansioon (getcomposer.org 2020). Symfony Flex on Composer liitännäinen, joka yksinkertaistaa pakettien asentamista ja poistamista Symfony sovelluksissa. (Symfony 2020e, viitattu 6.7.2020) Symfonyn versiosta 4 alkaen, Flex on kuulunut oletuksena uusiin sovelluksiin, joskaan sen käyttö ei ole pakollista (Symfony 2020u, viitattu 6.7.2020).

3.5 Doctrine ORM

Symfony itsessään on tietokantariippumaton ja tietokantojen käsittelyn voi toteuttaa parhaaksi katsomallaan tavalla. Symfony tarjoaa kuitenkin tietokantojen käsittelyn helpottamiseksi kolmannen osapuolen ratkaisun, Doctrinen, laajennuksen muodossa.

Doctrine on niin kutsuttu ORM-työväline (Doctrine 2020b, viitattu 28.5.2020). ORM (Object-Relational Mapping) on joukko ohjelmointitekniikoita, joilla pyritään saamaan yhteensopimattomat järjestelmät toimimaan yhteistyössä, kommunikoimaan ja vaihtamaan tietoa keskenään. Samalla niiden tarkoitus on helpottaa ohjelmoijan työtä (Kouraklis 2019, viitattu 25.4.2020.)

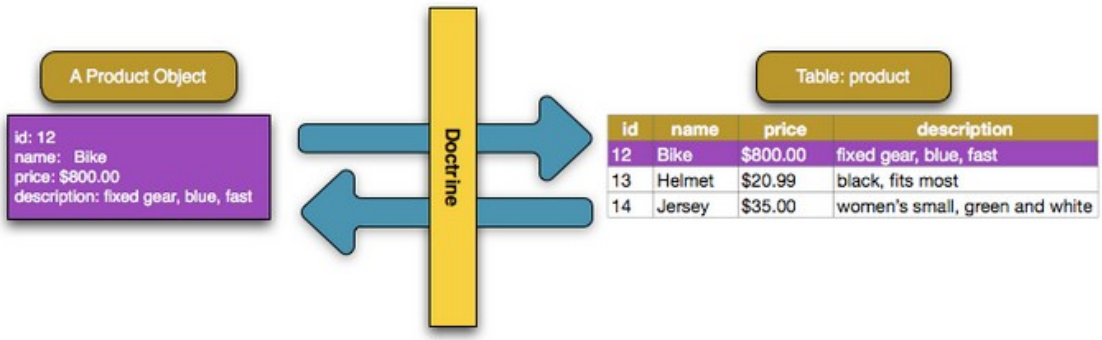
3.5.1 ORM lyhyesti

Tietokannat on suunniteltu tallentamaan ja tarjoamaan pääsy erityyppisiin tietoihin pysyvällä tavalla. Tieto tallennetaan tietokantoihin, joissa se säilyy sitä käyttävän sovelluksen suorituksen päättymisen jälkeenkin. Sovelluskehittäjät, jotka noudattavat olio-ohjelmoinnin periaatteita, ajattelevat tiedon esitystapaa sovelluksissa hyvin eri tavalla. Suurin osa tietokannoista kykenee hallitsemaan vain yksinkertaisia tietoja kuten kokonaislukuja, merkkejä, merkkijonoja, tavuja ja päivämääriä. Olio-ohjelmoinnin maailmassa kehittäjät näkevät maailman olioina, joilla on attribuutteja ja ominaisuuksia, mutta ne tarjoavat myös keinot manipuloida olioiden käyttäytymistä sellaisilla käsitteillä kuten polymorfismi ja periytyminen. Oliot, vaikka ovatkin yleisiä moderneissa ohjelmointikielissä, ovat monimutkaisia toteuttaa eivätkä ne ole yhteensopivia sen kanssa, kuinka tietokannat käsittelevät tietoja. Sen lisäksi, monimutkaisemmat rakenteet eivät ole helposti sovitettavissa tietokantojen tallennustapoihin. (Kouraklis 2019, viitattu 25.4.2020.)

ORM -ohjelmistokehykset tarjoavat olioperustaisen ohjelmakoodin ja tietokantaoperaatioiden välille kerroksen, joka huolehtii olioiden sovittamisesta tietokannan ymmärtämään muotoon. Nämä työvälineet luovat virtuaalisen oliotietokannan, joka mäppäytyy klassisiin tietokantarakenteisiin, on sovelluskehittäjille ymmärrettävä ja toimii olio-ohjelmointiympäristössä odotetunlaisesti. (Kouraklis 2019, viitattu 25.4.2020.)

3.5.2 Entity objektit

Doctrinessa säilytettävät tiedot sijaitsevat niin kutsutuissa Entity-objekteissa. Entityt ovat PHP-objekteja, joilla on yksilöivä tunniste. Entiteettiluokkien ei tarvitse periä mitään kantaluokkaa tai rajapintaa. (Doctrine 2020b, viitattu 28.5.2020.) Doctrine tietää entiteettiluokista vain sen perusteella, mitä sille kerrotaan metatiedoissa ja kuinka entiteetti tulee tallentaa tietokantaan. Tämän metatiedon määrittämiseen Doctrine tarjoaa useita eri vaihtoehtoja, kuten annotaatiot, XML, YAML ja PHP koodi. (Doctrine 2020a, viitattu 28.5.2020). Kuvio 4 esittää Doctrinen toimintaa.



KUVIO 4: Doctrine muuttaa tietokannassa olevat taulu-tyyppiset tiedot objekteiksi ja päinvastoin (Symfony 2020b, viitattu 12.6.2020)

3.6 Twig

Twig on nykyaikainen templaattimoottori PHP ohjelmointikielelle. Sen nykyinen kehittäjä on Symfonyn tekijä Fabien Potencier ja se käyttää sallivaa BSD-lisenssiä. (Symfony 2020s, viitattu 28.5.2020). Syntaksiltaan Twig muistuttaa djangon templaattikieltä ja jinjaa (Wikipedia 2020a, viitattu 28.5.2020). Symfonyn versiosta 5 lähtien puhdasta PHP:tä ei enää tueta templaateissa, sen sijaan templaatteihin suositellaan käyttämään Twigiä (Symfony 2020d, viitattu 28.5.2020). Fabien päätyi valitsemaan Symfonyn templaattikieleksi Twigin etsiessään PHP:lle templaattikieltä, koska hänen mielestään PHP ei ollut enää riittävän hyvä siihen tarkoitukseen ja siltä puuttui useita moderneilta templaattikieliltä vaadittavia ominaisuuksia (Potencier 2009, viitattu 12.6.2020). Kuviossa 5 näytetään esimerkki Twig-templaatista.

```
<!DOCTYPE html>
<html>
    <head>
        <title>My Webpage</title>
    </head>
    <body>
        <ul id="navigation">
        {% for item in navigation %}
            <li><a href="{{ item.href }}">{{ item.caption }}</a></li>
        {% endfor %}
        </ul>

        <h1>My Webpage</h1>
        {{ a_variable }}
    </body>
</html>
```

KUVIO 5: Esimerkki Twig templaatista (Symfony 2020r, viitattu 28.5.2020)

4 MUUT KÄYTETYT KIELET JA TEKNIIKAT

Sovelluksen tekoon käytettiin Symfony-sovelluskehyksen lisäksi useita kieliä ja tekniikoita. Taulukossa 1 kerrotaan lyhyesti tärkeimmistä.

TAULUKKO 1. Muita sovelluksen toteutukseen käytettyjä kieliä ja tekniikoita

Nimi	**Kuvaus**
PHP	PHP (PHP: Hypertext Preprocessor) on laajasti käytetty avoimen lähdekoodin yleiskäyttöinen skriptikieli, joka on erityisen soveltuva web-kehitykseen ja jota voidaan upottaa HTML:n sisään (PHP Group, viitattu 13.12.2019).
HTML	HTML (HyperText Markup Language) on sivunkuvauskieli. Sitä käytetään websivun rakenteen sekä sisällön määrittelyyn. (MDN 2020b, viitattu 20.7.2020.)
CSS	CSS on kieli web-sivujen esitysasun kuvaamiseen mukaan lukien värit, asettelu ja kirjasinlajit. Se mahdollistaa esityksen mukauttamisen erityyppisille laitteille, kuten suurille näytöille, pienille näytöille tai tulostimille. (W3C, viitattu 13.12.2019.)
Javascript	JavaScript on kevyt, tulkattava, oliokieli ensimmäisen asteen funktioilla, ja se on parhaiten tunnettu web-sivujen skriptikielenä, mutta sitä käytetään myös monissa muissa ympäristöissä. Se on prototyyppipohjainen, usean paradigman dynaaminen kieli, joka tukee oliopohjaista, imperatiivista sekä funktionaalista ohjelmointityyliä. (MDN, viitattu 13.12.2019.)
Ajax	Ajax (Asynchronous JavaScript and XML) ei ole teknologia itsessään, vaan se on vuonna 2005 Jesse James Garretin keksimä termi, joka kuvaa uudenlaisen lähestymistavan käyttää yhdessä jo olemassa olevia teknologioita, mukaan lukien HTML, XHTML, CSS, Javascript, DOM, XML, XSLT, ja kaikkein tärkeimpänä XMLHttpRequest -objektia. Kun näitä teknologioita käytetään yhdessä Ajax mallin mukaisesti, websovellukset kykenevät tekemään nopeita, inkrementaalisia päivityksiä käyttöliittymään lataamatta

	koko sivua uudelleen. (MDN 2020a, viitattu 16.4.2020.)
JSON	JSON (JavaScript Object Notation) on kevyt tiedonvälitysformaatti. Se on ihmiselle helppolukuinen ja helposti kirjoitettava. Se on koneille helppo tulkattava ja generoitava. Se perustuu osaan JavaScript-kielen standardia vuodelta 1999. JSON on täysin kieliriippumaton tekstiformaatti, mutta käyttää käytäntöjä jotka ovat tuttuja C-kielen tyyppisistä kieliä käyttäville ohjelmoijille. Nämä ominaisuudet tekevät JSON:ista ideaalisen tiedonvälityskielen. (JSON 2020, viitattu 16.4.2020)
Bootstrap	Bootstrap on alun perin Twitterin kehittämä CSS-sovelluskehys responsiiviseen ”mobile first” front-end web kehitykseen. Se sisältää CSS- ja valinnaisesti javascript-pohjaisia suunnittelumalleja typografialle, lomakkeille, painikkeille, navigoinnille ja muille käyttöliittymäkomponenteille. (Wikipedia 2020b, viitattu 10.6.2020.) Ennen julkaisuaan avoimena lähdekoodina vuonna 2011, se tunnettiin nimellä Twitter Blueprint. Nykyään sitä ylläpitää pieni joukko kehittäjiä GitHubissa. (Bootstrap 2020, viitattu 10.6.2020.)
jQuery	jQuery on vapaa avoimen lähdekoodin JavaScript-kirjasto, joka yksinkertaistaa HTML-dokumenttien käsittelyä ja muokkaamista sekä Ajaxin käyttöä JavaScriptillä (Wikipedia 2020c, viitattu 5.7.2020).

5 SOVELLUS

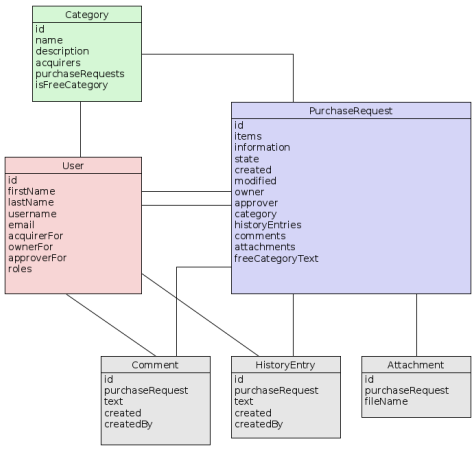
Tämän osion tarkoitus on antaa lukijalle yleiskuva toteutetusta sovelluksesta. Osiossa kerrotaan myös joistakin toteutukseen käytetyistä komponenteista.

5.1 Sovelluksen rakenne

Sovellus on toteutettu käyttäen Symfony 5 sovelluskehystä ja perustuu MVC-arkkitehtuuriin. Tässä alaluvussa esitellään sovelluksen osat lyhyesti MVC-mallin mukaisesti.

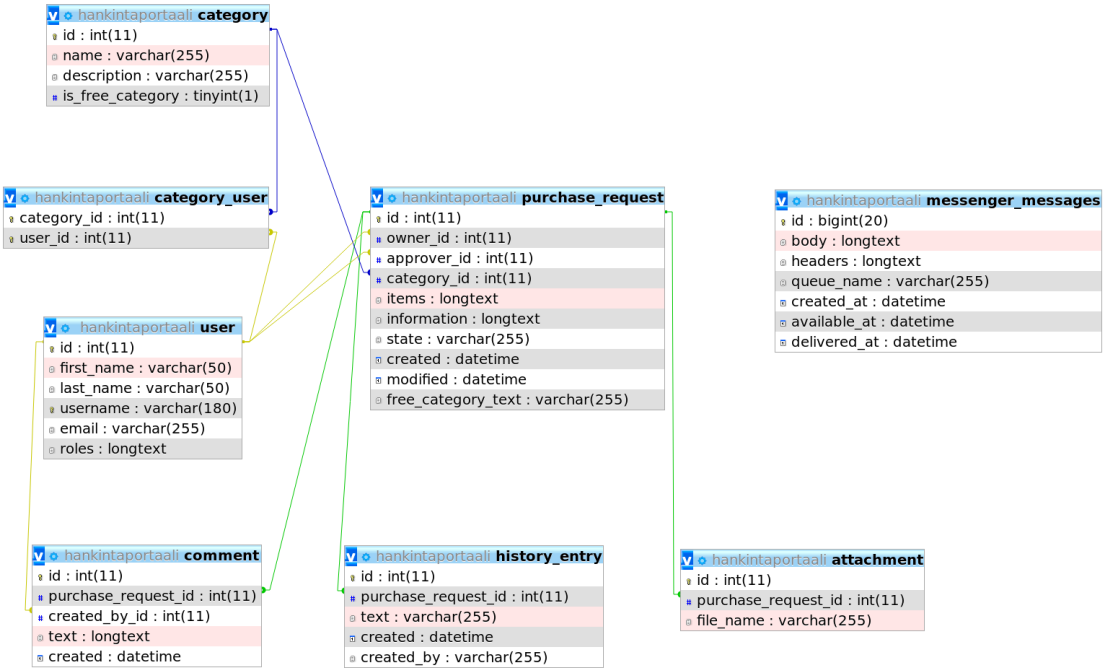
5.1.1 Malli

Sovelluksen malli sisältää kaiken kaikkiaan kuusi entiteettiä, jotka ovat PurchaseRequest, User, Category, Comment, HistoryEntry ja Attachment. Niistä kolme on itsenäisiä ja toiset kolme esiintyvät vain liitettynä muihin entiteetteihin (Kuvio 6). Entiteettien lisäksi malliin kuuluu kaikki muut, jotka eivät kuulu näkymään tai käsittelijään. Näitä ovat esimerkiksi niin kutsutut Repository-luokat, jotka sisältävät Entity-objektien tietokantahakuihin liittyvää logiikkaa, erilaiset palvelu-luokat, jotka huolehtivat esimerkiksi tarvittavien sähköpostiviestien lähettämisestä, esimiestiedon hakemisesta rajapinnan kautta, liitetiedostojen käsittelystä, tunnistautumisesta ldap -hakemistoon ja niin edelleen. Osan luokista voisi ajatella kuuluvan erilliseen palvelukerrokseen, mutta sellaista erottelua ei ole tehty.



KUVIO 6: Entiteetit

Sovellus käyttää tietokantaa Doctrinen avulla. Tietokannan luonti tapahtui määrittelemällä ensin Entity-luokat ja antamalla tämän jälkeen komentoriviltä käsky doctrinelle luoda tietokanta, taulut ja tarvittavat rakenteet. Entity-luokat ovat tavallisia PHP-luokkia, jotka on “koristeltu” Doctrinea varten niin että se ymmärtää niiden kuuluvan tietokantaan. Entity-luokkien luomisen apuna toimi Symfonyn maker bundle, joka sisältää konsolikomentoja mm. entity-luokkien luomiseen ja kenttien lisäämiseen. Alla olevassa kuviossa on esitetty vertailun vuoksi Doctrinen luoma tietokantarakenne phpMyAdmin-työvälineen suunnittelunäkymässä. Kuviossa näkyy myös Messenger-komponentin luoma taulu. Messenger-komponentista kerrotaan myöhemmin.



KUVIO 7: Doctrinen luoma tietokantarakenne phpMyAdminin suunnittelunäkymässä

5.1.2 Näkymä

Näkymä on käyttäjälle näkyvä osa sovelluksesta. Se on toteutettu käyttäen Twigiä, joka tuottaa selaimelle näytettävät sivut HTML-kielellä. Javascriptiä ja Ajaxia käytetään parantamaan käytettävyyttä esimerkiksi näytettäessä lomakkeet Bootstrapin modaalisina ikkunoina käyttöliittymän päällä, mutta ne eivät ole ehdoton vaatimus ja sovellus toimii myös ilman niitä.

Sovelluksella on yksi pohjatemplaatti ”base.html.twig”, joka toteuttaa yleisilmeen ja rakenteen, ja jonka muut templaatit sitten perivät. Näin samaa HTML-koodia ei tarvitse kopioida moneen eri paikkaan ja sen ylläpitäminen on helpompaa. Taulukossa 2 on listattu sovelluksen käyttämät Twig-templaatit, pois lukien fragmentit, jotka sisältävät painikkeita, lomakkeita ja Ajaxia käyttäen päivitettäviä sivun osia.

TAULUKKO 2. Sovelluksen käyttämät Twig-templaatit ja niiden tarkoitukset

Kansio	**Tiedosto**	**Tarkoitus**
templates/emails/		
	email.html.twig	Sähköpostiviestit
templates/home/		
	index.html.twig	Etusivu
	quickhelp.html.twig	Pikaohje
templates/purchase_request/		
	add_attachment.html.twig	Liitetiedoston lisääminen
	add_comment.html.twig	Kommentin lisääminen
	edit.html.twig	Muokkaus
	list.html.twig	Listasivu
	new.html.twig	Uusi hankintapyyntö
	show.html.twig	Hankintapyynön tarkastelu
templates/security/		
	login.html.twig	Kirjautumislomake
templates/base.html.twig/		Pohjamalli

5.1.3 Käsittelijä

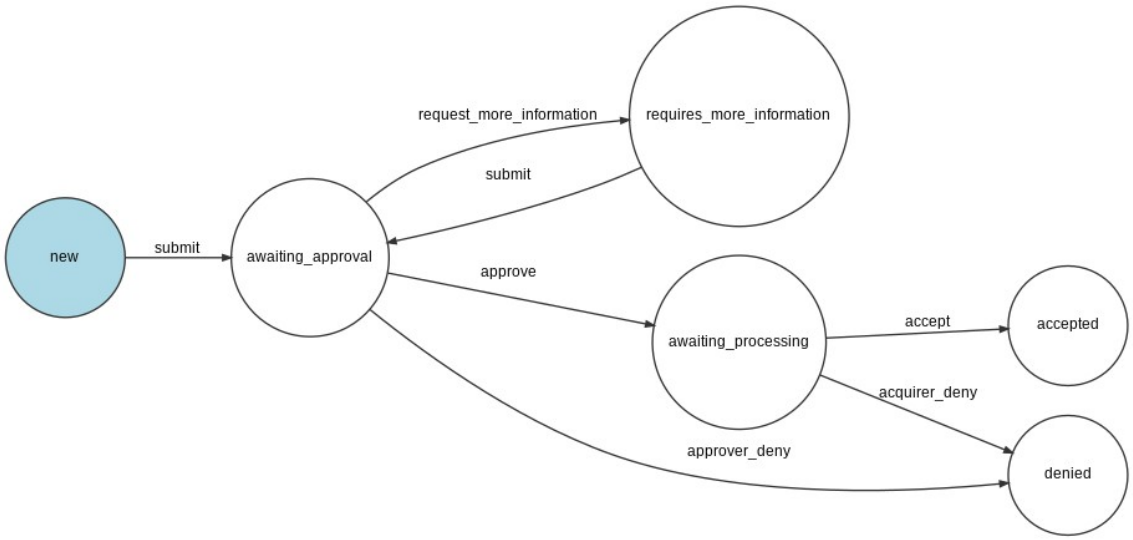
Käsittelijätasolla on kolme eri käsittelijää, joita ovat CategoryController, HomeController ja PurchaseRequestController. Sovelluksen toiminnan kannalta ylivoimaisesti tärkein on PurchaseRequestController, joka sisältää varsinaiset sovelluksen tarvitsemat toiminnot. HomeController vastaa vain sovelluksen etusivusta ja pikaohjeista, eikä vaadi kirjautumista. CategoryController toteuttaa vain pienen Ajax -rajapinnan, jota hyödynnetään lomakkeilla näyttämään valitun kategorian kuvaus.

5.2 Toimintaperiaate

Sovelluksen toiminnan kannalta keskeisessä roolissa toimii Symfonyn Workflow-komponentti. Se huolehtii hankintapyyntöjen etenemisestä halutun prosessin mukaisesti.

Workflow-komponentti tarjoaa oliopohjaisen tavan määritellä prosessi tai elinkaari, jonka läpi olio kulkee. Jokaista prosenssin askelta tai vaihetta kutsutaan paikaksi. Näiden lisäksi määritellään siirtymät, jotka kuvaavat toimintoja päästä paikasta toiseen. (Symfony 2020q, viitattu 11.6.2020) Tilakoneen ja työnkulun ero on siinä, että tilakone voi olla kerrallaan vain yhdessä paikassa, kun taas työnkulku voi olla useammassa paikassa samanaikaisesti (Symfony 2020v, viitattu 8.6.2020).

Sovelluksessa käytetään tilakonemallia, joten hankintapyyntö voi olla kerrallaan vain yhdessä tilassa. Kuviossa 8 näet sovelluksen käyttämän tilakoneen tilat ja siirtymät. Toimintaperiaate on yksinkertainen. Käyttäjä luo hankintapyynnön ja lähettää sen eteenpäin hyväksyttäväksi. Esimies joko hylkää hankintapyynnön, hyväksyy sen tai lähettää sen takaisin täydennettäväksi. Kun esimies on hyväksynyt hankintapyynnön, IT-palvelut tekee lopullisen hankintapäätöksen.



KUVIO 8: Tilakone

Esimerkki hankintapyynnön mahdollisesta reitistä:

1. Käyttäjä luo uuden hankintapyynnön, täyttää tarvittavat tiedot ja lähettää sen hyväksyttäväksi esimiehelle.
2. Tarkistaessaan hankintapyyntöä esimies huomaa siinä joitakin puutteita ja palauttaa sen täydennettäväksi. Halutessaan esimies voi liittää hankintapyyntöön kommentin. Pyyntö voi perustua myös “käytäväkeskusteluun”, joten kommenttia ei vaadita.
3. Käyttäjä lisää tarvittavat tiedot tai liitetiedoston ja lähettää sen uudelleen hyväksyttäväksi.
4. Esimies tarkastaa hankintapyynnön ja hyväksyy sen.
5. IT-palvelut havaitsee vastaavan laitteen olevan jo varastossa ja sopii hankintapyynnön tekijän kanssa hankittavan laitteen korvaamisesta sillä. Hankintaa ei tarvitse tehdä, joten IT-palvelut sulkee hankintapyynnön tarpeettomana ja toimittaa varastossa olevan laitteen.

Tilakoneen määrityksissä käytetään metatietoja, joiden ansiosta lähes kaikki tilakoneeseen liittyvä voidaan pitää yhdessä paikassa ja sen muokkaamisesta on pyritty saamaan mahdollisimman vaivatonta. Metatietoina on asetettu tilakoneen paikkojen, eli tilojen, tekstit, siirtymiin liittyvien toimintopainikkeiden tekstit, historiatietoihin liittyvät tekstit, sekä siirtymään mahdollisesti liittyvät sähköpostiviestit. Näiden lisäksi tilakoneen tilat on määritelty käyttäjäroolien mukaisiin jonoihin (Taulukko 3), joiden perusteella hankintapyynnöt esitetään eri osioissa. Tilakone määritellään packages/workflow.yaml -tiedostossa. Kuvioissa 9-11 näytetään esimerkit tilakoneen tilojen ja siirtymien määrityksestä sekä ”jonon” määritys tilakoneen yleisissä metatiedoissa.

TAULUKKO 3. Käyttäjäroolien mukaiset jonot ja niihin kuuluvat tilat

Nimi	**Selite**	**Jonoon kuuluvat tilat**
user_queue	Käyttäjän ”vaatii toimenpiteitä”	’requires_more_information’, ’new’, null
user_sent	Käyttäjän ”lähetetyt”	’awaiting_approval’, ’awaiting_processing’
approver_queue	Esimiehen ”vaatii toimenpiteitä”	’awaiting_approval’
approver_sent	Esimiehen ”lähetetyt”	’awaiting_processing’
acquirer_queue	IT-palveluiden ”vaatii toimenpiteitä”	’awaiting_processing’
closed	Käsitelty loppuun	’accepted’, ’denied’

```
    requires_more_information:
        metadata:
            text: state.requires_more_information
```

KUVIO 9: Tilan määritys

```
    request_more_information:
        guard: "is_granted('ROLE_APPROVER')"
        from: awaiting_approval
        to: requires_more_information
        metadata:
            action_text: transition.request_more_information
            history_text: transition.history.request_more_information
            email:
                subject: transition.email.subject.request_more_information
                template: 'emails/email.html.twig'
```

KUVIO 10: Siirtymän määritys

```
user_sent: ['awaiting_approval', 'awaiting_processing']
```

KUVIO 11: Jonon määritys

5.3 Käyttäjät ja käyttöoikeudet

Symfony tarjoaa useita käyttövalmiita ratkaisuja kirjautumiseen ja käyttäjien hallintaan, jotka sopivat useimpiin tarpeisiin, ja jos ne eivät sellaisenaan riitä, niin Symfony tarjoaa monipuoliset mahdollisuudet räätälöityihin toteutuksiin. Toteutetussa sovelluksessa käytetään räätälöityä menetelmää, jossa käyttäjät tallennetaan tietokantaan, mutta tunnistus tapahtuu Ldap-hakemistoa vastaan.

5.3.1 Käyttäjä

Käyttäjää varten luotiin Entity-luokka User säilyttämään sovelluksen tarvitsemia tietoja tietokannassa. Samaa luokkaa käytetään sovelluksen sisäisiin tietoihin sekä käyttöoikeuksien hallintaan. Kirjautumista varten luokka toteuttaa rajapinnan UserInterface, joka sisältää kirjautumiseen vaaditut ominaisuudet ja metodit. Käytännössä niitä kaikkia ei kuitenkaan tarvita, koska todentaminen tapahtuu Ldap-palvelinta vastaan eikä salasanoja tarvitse tallentaa.

5.3.2 User Provider

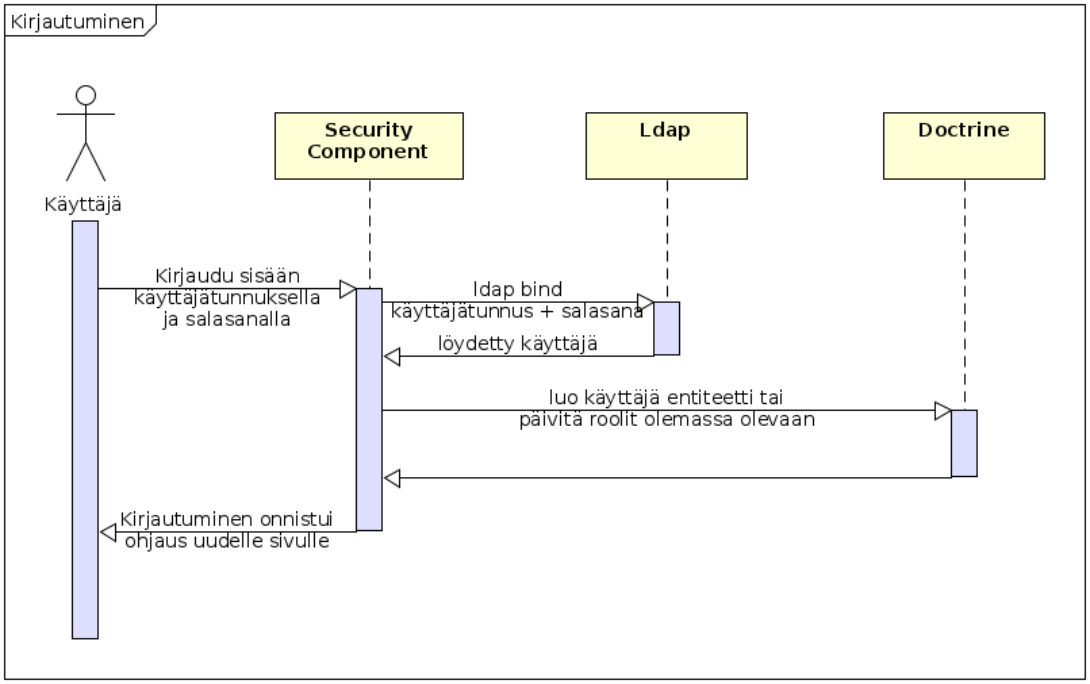
User Provider -luokilla on kaksi tarkoitusta: Ladata käyttäjä uudelleen istunnosta sekä ladata käyttäjä jonkin ominaisuuden käyttöä varten. Symfony tarjoaa valmiina useita eri vaihtoehtoja kuten: Entity User Provider (lataa käyttäjät tietokannasta), LDAP User Provider (lataa käyttäjät LDAP palvelimelta), Memory User Provider (lataa käyttäjät konfiguraatiotiedostosta) sekä Chain User Provider (yhdistää kahden tai useamman providerin yhdeksi). (Symfony 2020i, viitattu 7.7.2020.) Toteutetussa sovelluksessa käytetään Entity User Provider -luokkaa.

5.3.3 Todentaminen (Authentication)

Symfonyssä käyttäjien todentamisesta vastaa niin kutsuttu palomuuri (Firewall). Se määrittelee kuinka käyttäjät voivat tunnistautua – esimerkiksi käyttäen kirjautumislomaketta tai tokenia. Sovelluksessa voi olla useampia palomuureja, mutta vain yksi niistä voi olla kerrallaan aktiivisena si-

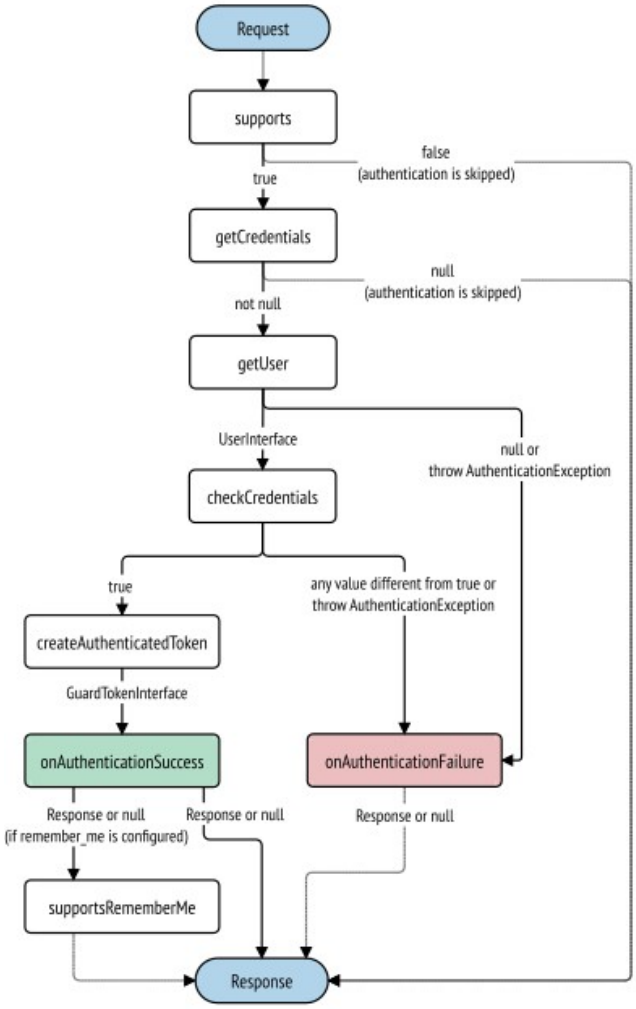
vupyynnölle. Jokainen palomuuri voi sisältää useamman tavan tunnistautua. Jos käyttäjä ei ole tunnistautunut, käytetään anonyymi tilaa. (Symfony 2020h, viitattu 8.7.2020.)

Toteutetussa sovelluksessa on yksi varsinainen palomuuri ”main”, joka vastaa kaikkiin sivupyyntöihin ja kohdistaa ne tarvittaessa todentajalle. Kirjautumislomake esitetään, kun käyttäjä valitsee kirjaudu sisään tai yrittää avata tunnistautumista edellyttävän sivun. Kuviossa 12 on yksinkertaistettu sekvenssikaavio sovellukseen kirjautumisesta.



KUVIO 12: Sekvenssikaavio kirjautumisesta

Sovelluksessa käyttäjän tunnistaminen on toteutettu käyttäen Symfonyn Guard Authenticator tyyppistä todentajaa, joka antaa täyden hallinnan koko todentamisprosessiin. Kuviossa 13 näkyy Guard Authenticator -todentajan metodit (Symfony 2020o, viitattu 8.7.2020).



KUVIO 13: Guard Authenticator metodit

Tunnistautuminen LDAP -hakemistoa vastaan tapahtuu jo todentajan metodissa getUser, joten metodi checkCredentials palauttaa aina arvon tosi. Käytännössä tämä tapahtuu kutsumalla sitä varta vasten luodun luokan LdapAuth metodia authenticate, joka palauttaa onnistuessaan käyttäjän tiedot DTO:ssa (Data Transfer Object). Sen jälkeen DTO ohjataan UserService-luokalle joka luo tarvittaessa uuden käyttäjän tai päivittää olemassa olevan käyttäjän roolit sen mukaan onko hänet asetettu jonkin hankintakategorian hankkijan rooliin tai onko järjestelmässä hankintapyyntöjä, joissa käyttäjä on hyväksyjän roolissa.

5.3.4 Valtuutus (Authorization)

Valtuutuksessa on kaksi puolta: käyttäjälle kirjautumisen yhteydessä asetettavat roolit ja resurssille, esimerkiksi käsittelijä, asetettavat vaatimukset. Valtuutuksen tehtävä on päättää, onko käyttäjällä oikeus kyseiseen resurssiin vai ei. (Symfony 2020h, viitattu 8.7.2020.)

Pääsyn esto on mahdollista toteuttaa kahdella tavalla. Ensimmäinen tapa on määritellä security.yaml -konfiguraatiotiedostoon kirjautumista vaativat URL-osoitteet säännöllisillä lausekkeilla (regular expression). Toinen tapa on tehdä se PHP:llä käsittelijöissä tai muussa koodissa. (Symfony 2020h, viitattu 8.7.2020.) Sovelluksen ylläpito-osio on suojattu käyttäen ensimmäistä menetelmää (Kuvio 14) ja muut sivut toista menetelmää käyttäen. Käsittelijässä on suojattu sivut yleisellä tasolla kirjaantuneen käyttäjän roolien mukaisesti @IsGranted annotaatiolla (Kuvio 15). Sen lisäksi osa sivuista on suojattu tarkemmalla tasolla PurchaseRequestVoter -luokassa, joka osaa ottaa huomioon hankintapyynnön kulloisenkin tilan ja käyttäjien roolin suhteessa hankintapyyntöön. Esimerkki hankintapyynnön poistamiseen liittyvästä tarkistuksesta kuvioissa 16 ja 17.

```
access_control:
    - { path: ^/admin, roles: [ROLE_ADMIN, ROLE_SUPER_ADMIN] }
```

KUVIO 14: Admin osion pääsynvalvonta

```
/**
 * @Route("/purchase_request")
 * @IsGranted("ROLE_USER")
 */
class PurchaseRequestController extends AbstractController
```

KUVIO 15: Käsittelijän pääsynvalvonta

```
$this->denyAccessUnlessGranted('delete', $purchaseRequest);
```

KUVIO 16: Poisto-oikeuden tarkistaminen käsittelijässä

```
private function canDelete($purchaseRequest, $user)
{
    // User can delete only his own requests and only when it's on his own queue
    if ($purchaseRequest->getOwner() === $user &&
        in_array($purchaseRequest->getState(), $this->mds->getMetadata('user_queue'), true))
        return true;
    }

    return false;
}
```

KUVIO 17: canDelete metodi PurchaseRequestVoter -luokassa

5.3.5 Käyttäjäroolit

Tavallisia käyttäjärooleja on kolme: asiakas (peruskäyttäjä), esimies (hyväksyjä) ja IT-palvelut (hankkija). Näiden lisäksi ylläpitäjille on kaksi roolia, joiden ero on se että ns. “superylläpitäjällä” on oikeus hallita ylläpitäjiä. Ylläpitäjien tehtävä on hallita hankintakategorioita ja hankkijan rooleissa toimivia käyttäjiä. Esimiehen rooli päivittyy käyttäjälle automaattisesti kirjautumisen yhteydessä, mikäli tietokannassa on hankintapyyntöjä, joissa hän on esimiehenä. Samalla tavoin päivittyy myös hankkijan rooli sen mukaan, onko hänet asetettu hankkijaksi jollekin kategorialle. Alla olevassa taulukossa on esitetty sovelluksen käyttämät roolit ja niiden tarkoitukset.

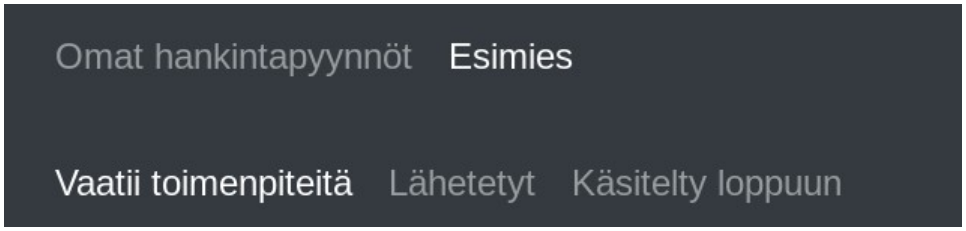
TAULUKKO 4. Käyttäjäroolit ja tarkoitukset

Rooli	**Sovelluksen sisäinen rooli**	**Tarkoitus**
Asiakas (peruskäyttäjä)	ROLE_USER	Hankintapyyntöjen tekeminen
Esimies (hyväksyjä)	ROLE_APPROVER	Hankintapyyntöjen tarkastus, hyväksyminen, hylkääminen ja palauttaminen täydennettäväksi
IT-palvelut (hankkija)	ROLE_ACQUIRER	Hankintapäätöksen tekeminen
Ylläpitäjä	ROLE_ADMIN	Hankintakategorioiden sekä IT-palveluiden (hankkija) roolissa toimivien käyttäjien hallinta
Superylläpitäjä	ROLE_SUPER_ADMIN	Ylläpitäjien hallinta sekä ylläpitäjän tehtävät

Hankintapyyntöön esimiestieto haetaan erillisen rajapinnan, Application Programming Interface (API), kautta hankintapyynnön luonnin yhteydessä. Jokaiselle käyttäjälle taataan löytyvän esimies. Rajapinnan toiminta lyhyesti: Sovellus tekee HTTP-pyynnön määriteltyyn osoitteeseen antaen käyttäjätunnuksen parametrina, johon rajapinta vastaa palauttaen esimiehen tiedot puolipistein eroteltuna merkkijonona. Rajapinnan käyttön helpottamiseksi sovellukseen luotiin ManagerApiClient -luokka, joka huolehtii HTTP-pyynnöstä ja vastauksen käsittelemisestä. Rajapinta kuuluu johonkin OAMK:n järjestelmään, mutta valitettavasti tarkempia tietoja siitä ei ole esittää.

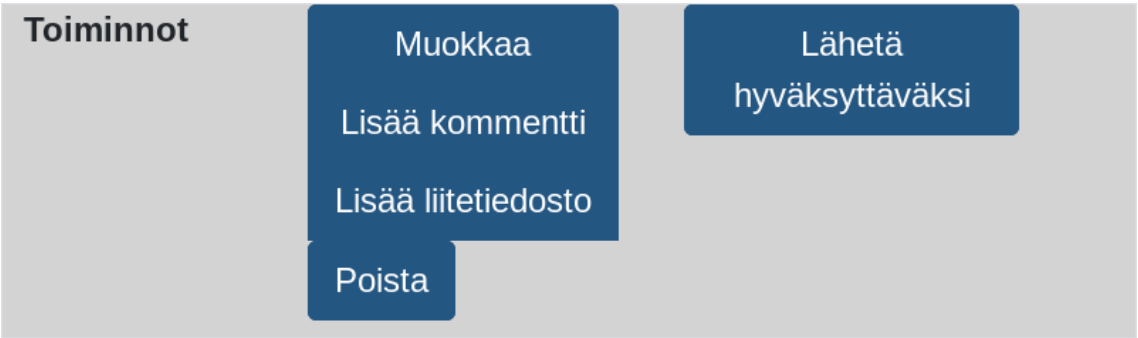
5.4 Käyttöliittymän osiot

Sovellus on jaettu käyttäjäroolien mukaan kolmeen eri osioon, joissa jokaisessa on omat aliosionsa. Aliosiot ovat kaikille rooleille pääosin samanlaisia, mutta niissä näkyy eri vaiheissa olevat hankintapyynnöt (Kuvio 18). ”Vaatii toimenpiteitä” sisältää hankintapyyntöjä, jotka vaativat käyttäjältä jotain toimenpiteitä. ”Lähetetyt” aliosio sisältää eteenpäin lähetetyt hankintapyynnöt, joita ei ole vielä käsitelty loppuun. Nimensä mukaisesti ”Käsitelty loppuun” sisältää loppuun käsitellyt hankintapyynnöt. Näiden lisäksi sovelluksessa on ylläpitäjille oma osionsa. Valittavissa olevat osiot riippuvat kirjautuneen käyttäjän rooleista.



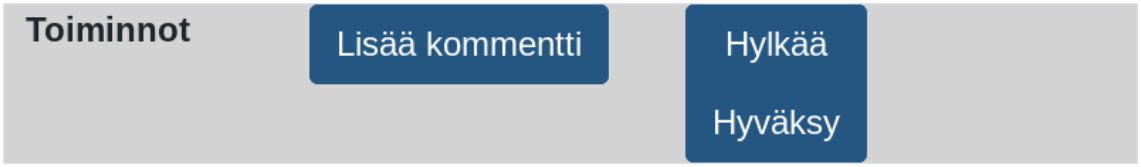
KUVIO 18: Esimerkki osion valinnasta esimiehen näkymässä

Hankintapyyntöihin kohdistettavissa olevat toiminnot määräytyvät dynaamisesti hankintapyynnön tilan ja kirjautuneen käyttäjän roolien mukaan. Alla olevissa kuvioissa on esitetty muutamia esimerkkejä käyttäjän, esimiehen sekä IT-palveluiden mahdollisista toiminnoista. Vasemmalla puolella on yleiset hankintapyyntöön kohdistuvat toiminnot ja oikealla sen tilojen etenemiseen liittyvät toiminnot.



KUVIO 19: Käyttäjän toimintoja

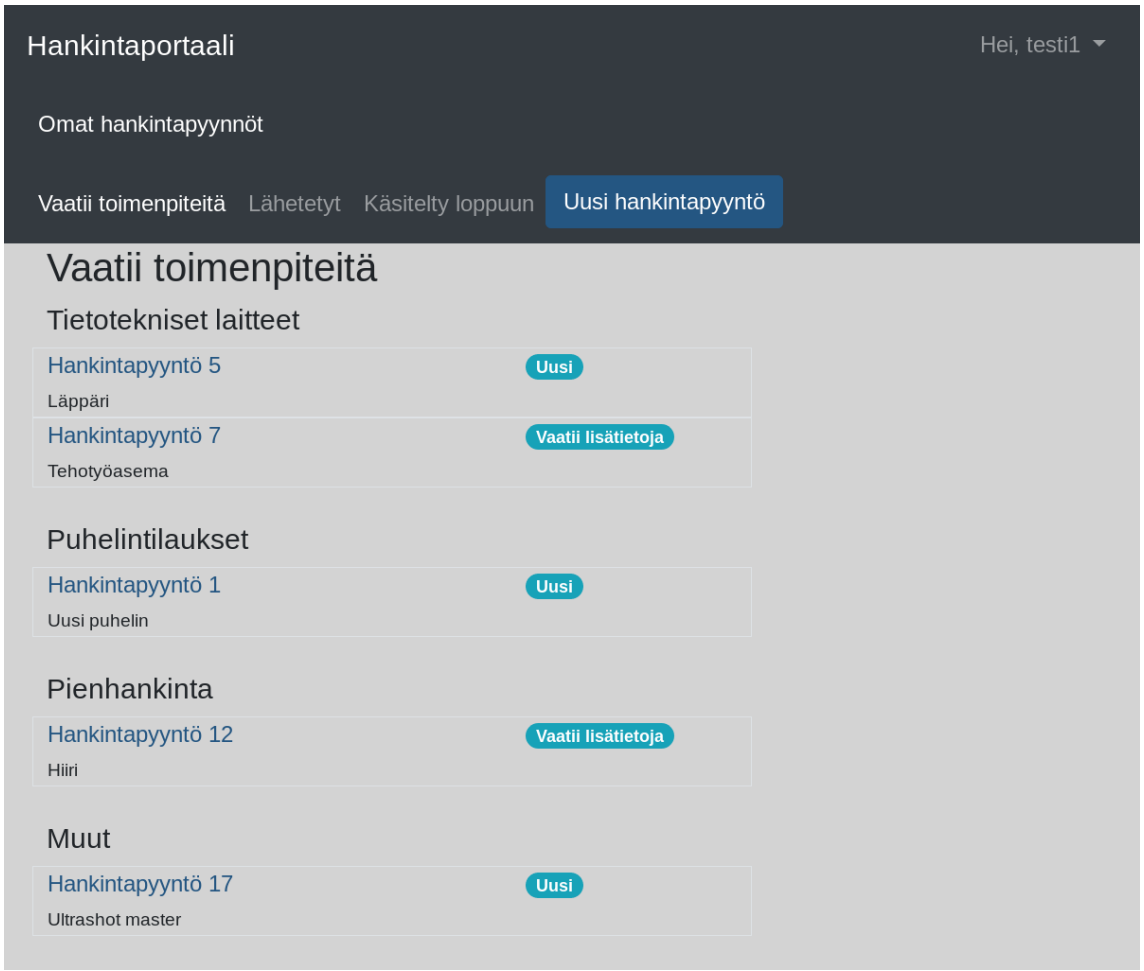
KUVIO 20: Esimiehen toimintoja



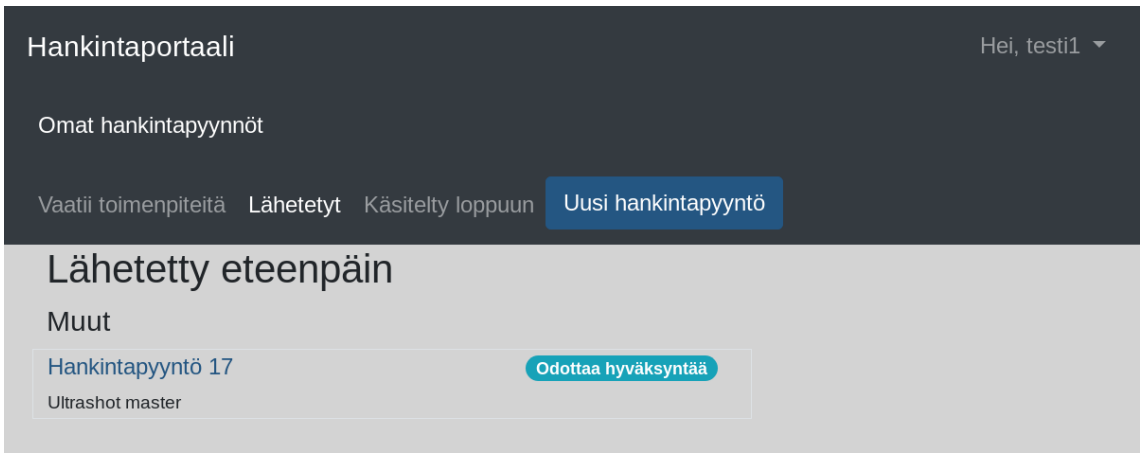
KUVIO 21: IT-palveluiden toimintoja

5.4.1 Omat hankintapyynnöt

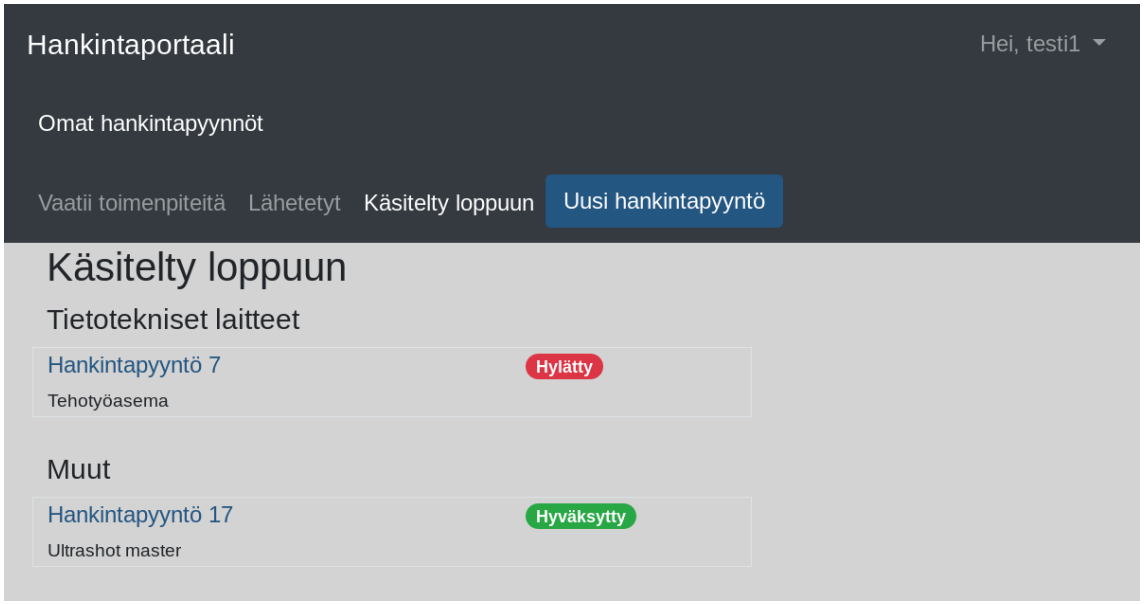
Kaikki kirjautuneet käyttäjät pääsevät käyttämään tätä osiota. Tässä osiossa tapahtuu hankintapyyntöjen luonti ja niiden lähettäminen esimiehen hyväksyttäväksi. Täällä voi seurata omien hankintapyyntöjen etenemistä ja viestiä esimiehen sekä it-palveluiden kanssa hankintapyyntöihin liitettävien kommenttien muodossa. Kuvioissa 22-24 on esitetty omien hankintapyyntöjen listanäkymä hankintapyyntöjen eri vaiheissa.



KUVIO 22: Vaatii toimenpiteitä

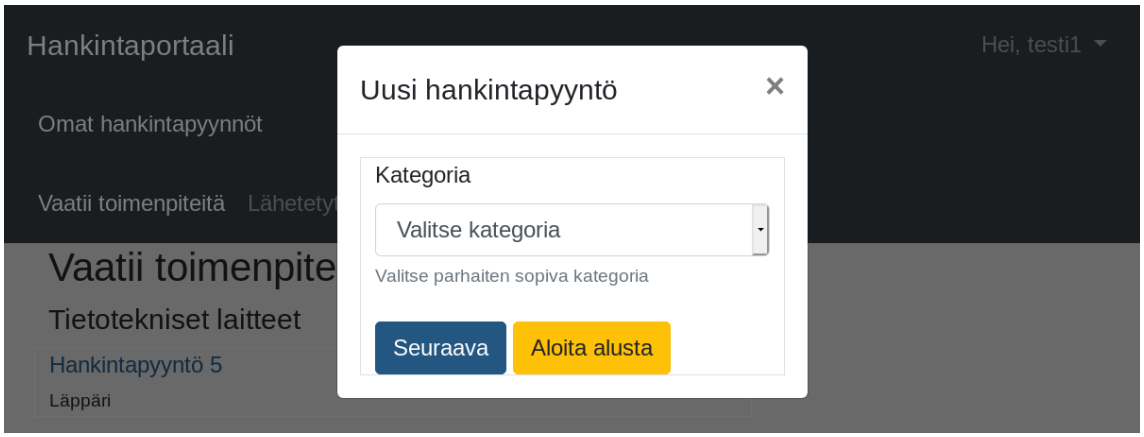


KUVIO 23: Lähetetyt

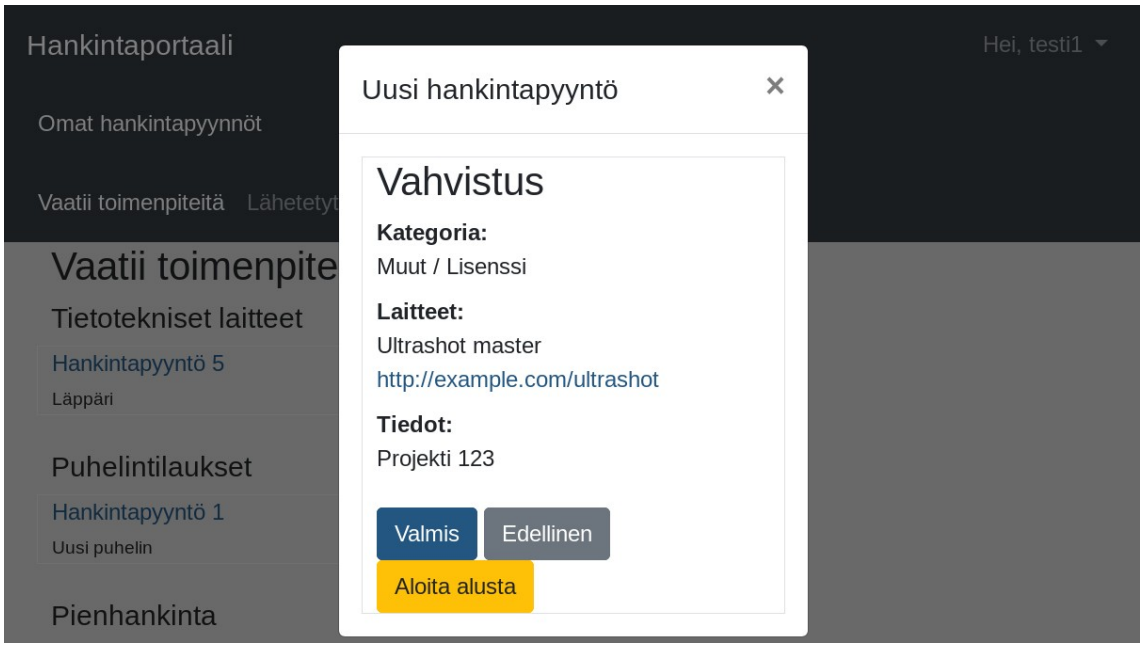


KUVIO 24: Käsitelty loppuun

Hankintapyynnön luonti tapahtuu vaiheistetun lomakkeen avulla. Sen ansiosta sovellusta on vaivatonta käyttää esimerkiksi mobiililaitteella. Vaiheistettu lomake toteutettiin kolmannen osapuolen CraueFormFlowBundle-laajennusta käyttäen. Alla olevissa kuviossa esitetään hankintapyynnön luonnin ensimmäinen ja viimeinen vaihe.



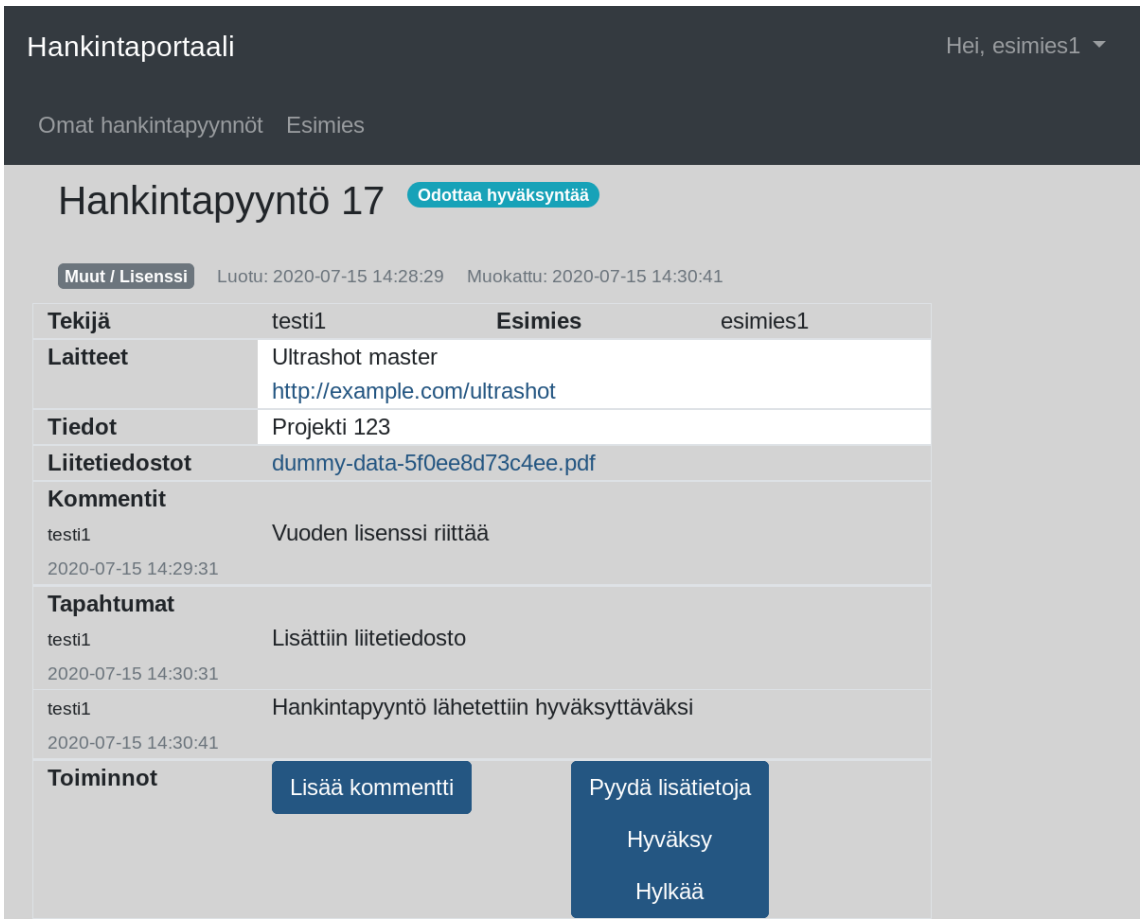
KUVIO 25: Hankintapyynnön luonti vaiheistetulla lomakkeella – ensimmäinen vaihe



KUVIO 26: Hankintapyynnön luonti vaiheistetulla lomakkeella - viimeinen vaihe

5.4.2 Esimies

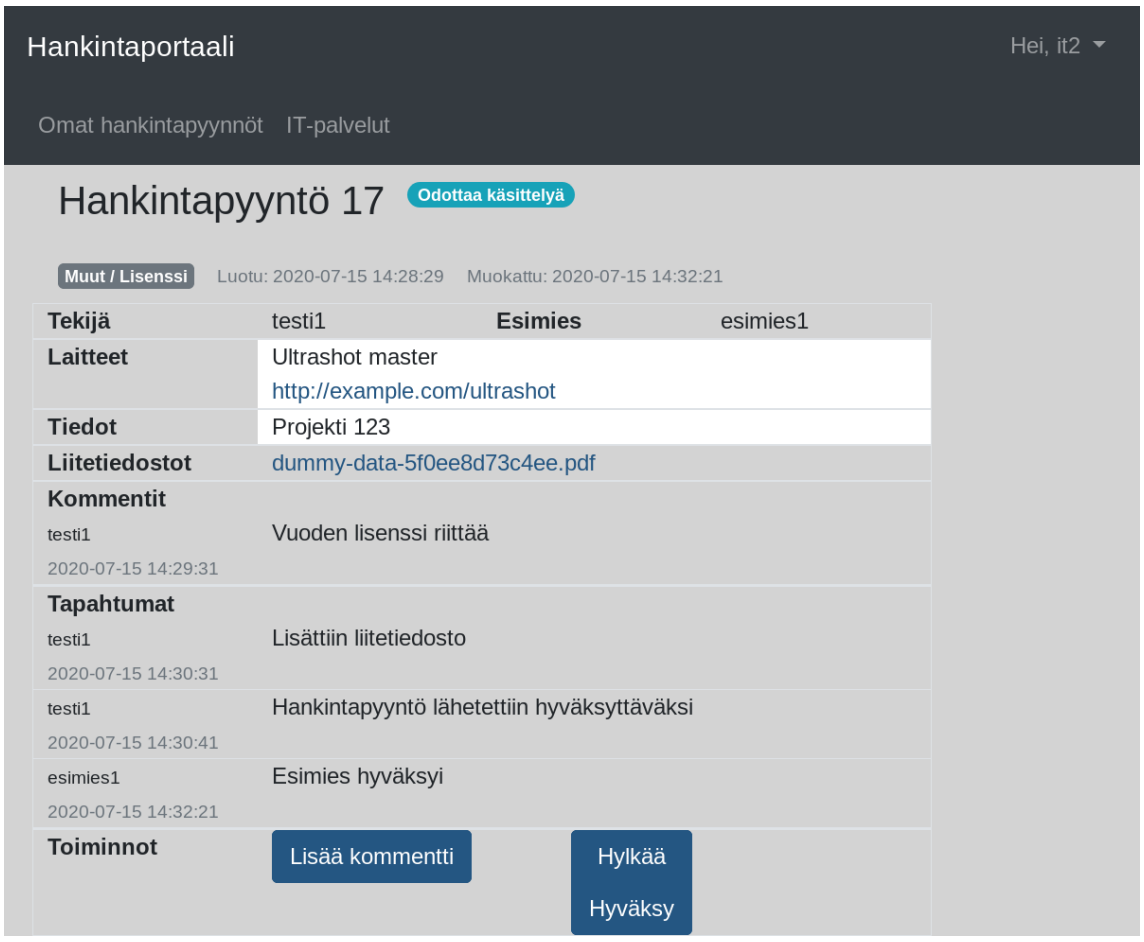
Esimiesosiossa voi tarkastella, hyväksyä, hylätä tai palauttaa täydennettäväksi alaisten lähettämiä hankintapyyntöjä, sekä liittää niihin kommentteja esimerkiksi lisätietojen pyytämistä varten tai ohjeistaakseen IT-palveluita. Kun esimies pyytää lisätietoja, siirtyy hankintapyyntö tekijän käsiteltäväksi ja hänelle lähetetään automaattisesti sähköpostiviesti. Seuraavassa kuviossa on esimerkki esimiehen näkymästä tarkastellessa alaisen hyväksyttäväksi lähettämää hankintapyyntöä.



KUVIO 27: esimiehen näkymä tarkastellessa alaisen lähettämää hankintapyyntöä

5.4.3 IT-palvelut

Tässä osiossa voi tarkastella, hyväksyä tai hylätä esimiehen aiemmin hyväksymiä hankintapyyntöjä. Täällä voi myös osallistua keskusteluun IT-palveluiden ominaisuudessa kommenttien välityksellä. IT-palvelut tekee lopullisen hankintapäätöksen esimerkiksi varastotilanteen mukaan. Hankintaa ei ole välttämättä tarpeellista tehdä, jos varastosta löytyy korvaava tuote. Kuviossa 28 näkyy esimerkki IT-palveluiden näkymästä tarkastellessa käsiteltävänä olevaa hankintapyyntöä.

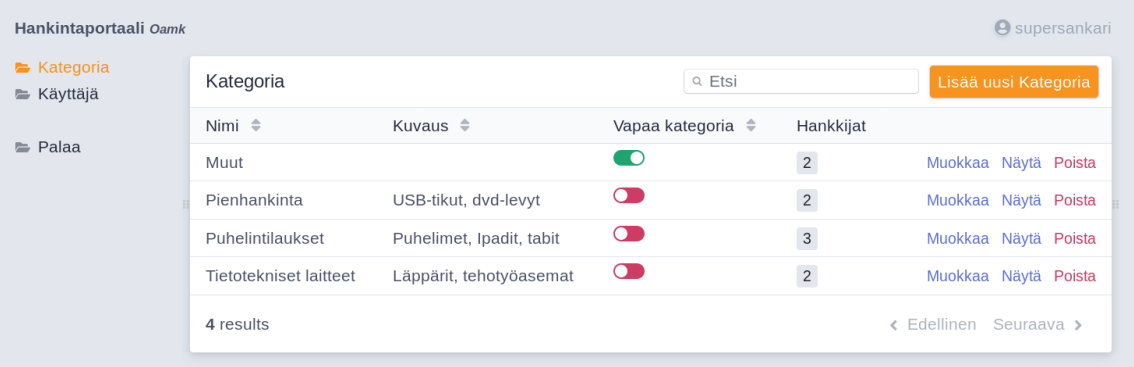


KUVIO 28: IT-palveluiden näkymä tarkastellessa käsiteltävänä olevaa hankintapyyntöä

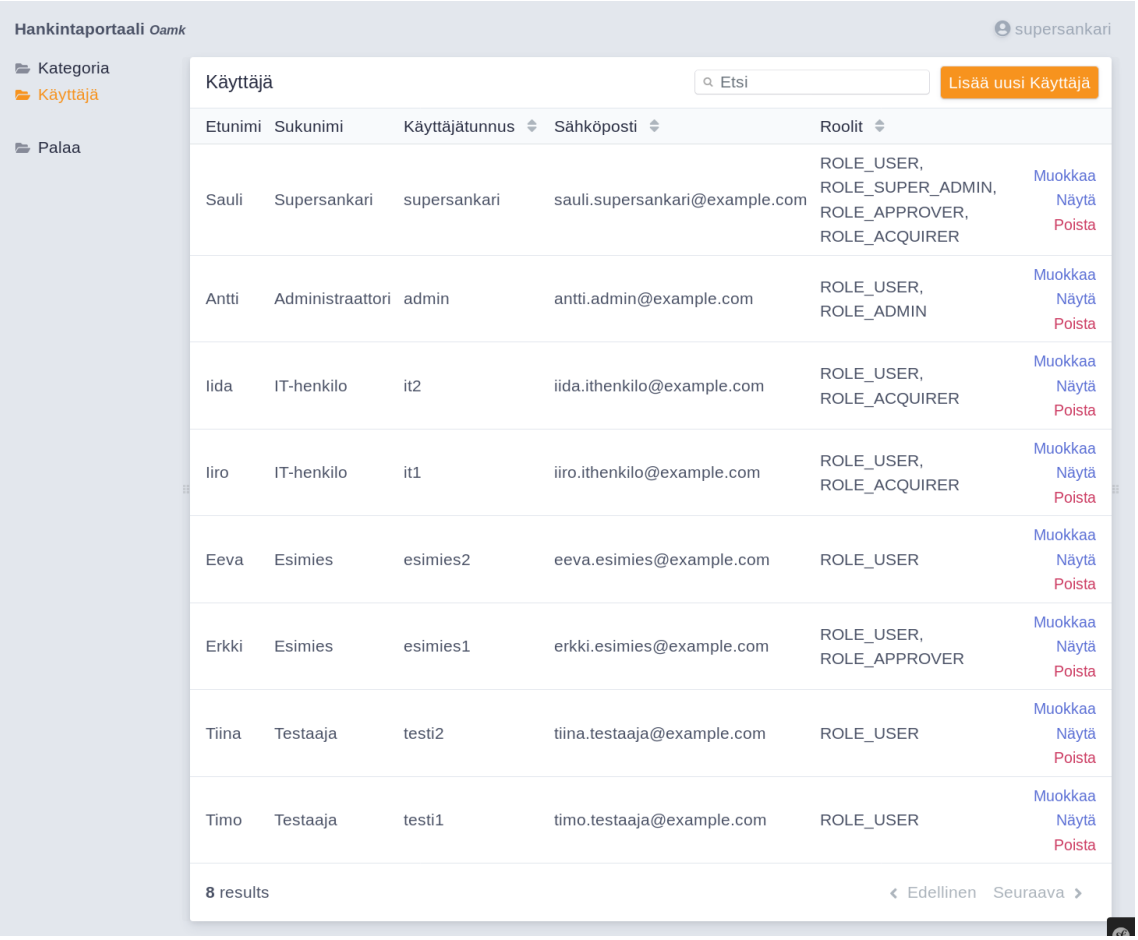
Riippumatta siitä, hyväksyykö vai hylkääkö IT-palvelut hankintapyynnön, lähtee asiakkaalle siitä sähköpostiviesti automaattisesti.

5.4.4 Ylläpito

Ylläpito-osiossa hallinnoidaan sovelluksen taustalla olevia tietoja kuten hankintakategorioita, hankkijoita sekä ylläpitäjiä. Ylläpito-osio on toteutettu käyttäen Symfonyn EasyAdminBundle -laajennusta. Kuvioissa 29 ja 30 näkyy kategorioiden ja käyttäjien hallinta ylläpito-osiossa.



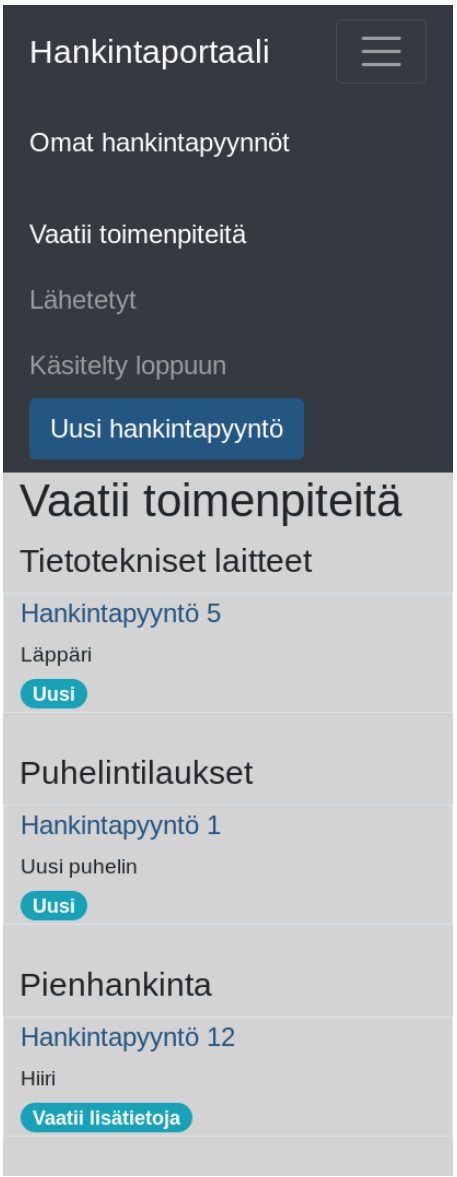
KUVIO 29: Kategorioiden hallinta



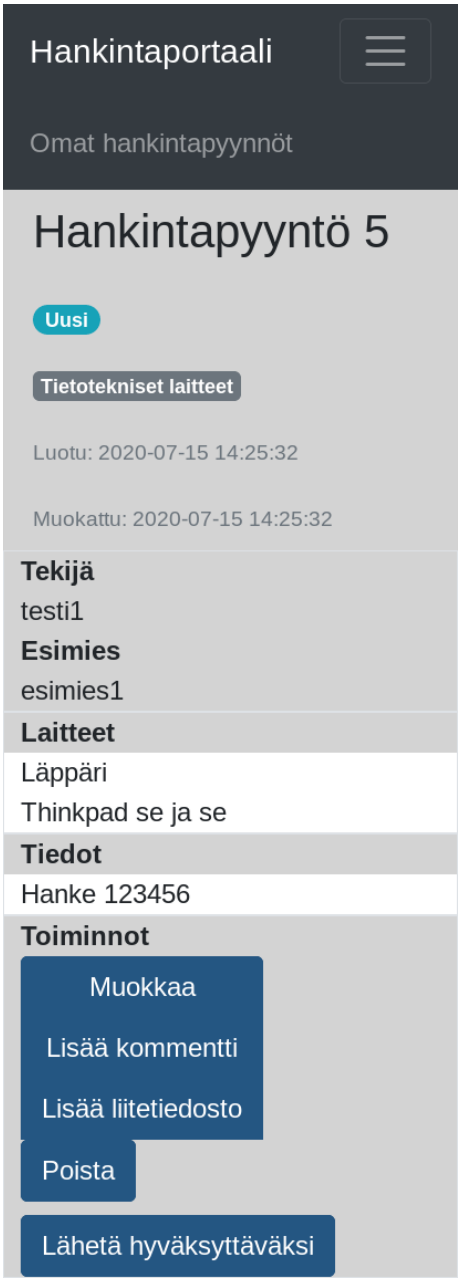
KUVIO 30: Käyttäjien hallinta

5.5 Responsiivisuus

Yhtenä tärkeänä vaatimuksena oli sovelluksen käytettävyys myös älypuhelimilla. Käyttöliittymä suunniteltiin mukautumaan eri kokoisille näytöille käyttäen Bootstrap -kirjastoa siten, että suuremmalla näytöllä aiemmin vierekkäin olleet elementit esitetään pienemmällä näytöllä allekkain. Kuvioissa 31 ja 32 on esimerkit käyttöliittymän mukautumisesta pienemmälle näytölle.



KUVIO 32: Käyttöliittymän mukautuminen esimerkki 1



KUVIO 31: Käyttöliittymän mukautuminen esimerkki 2

5.6 Sähköpostiviestit

Sovellus lähettää sähköpostiviestin, kun hankintapyyntö on käsitelty loppuun tai kun esimies pyytää lisätietoja hankintapyyntöön. Sähköpostiviestien lähettäminen on toteutettu käyttäen Symfonyn komponentteja Mailer ja Messenger.

5.6.1 Mailer

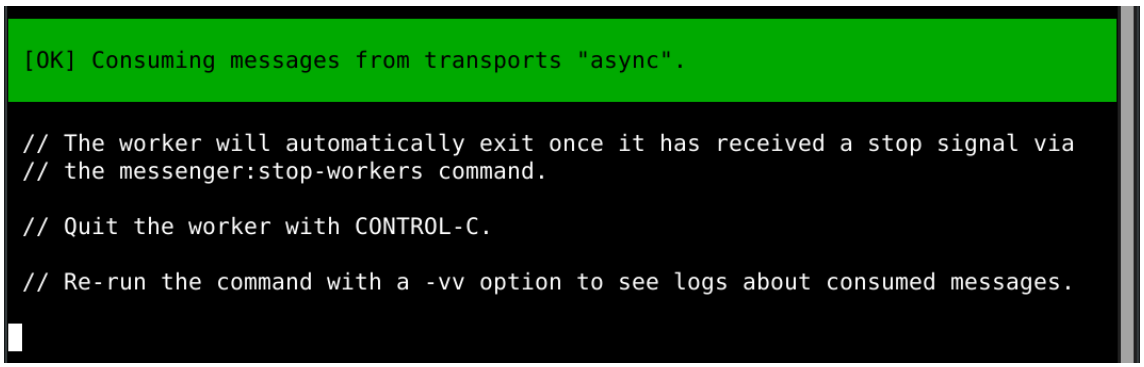
Symfonyn Mailer on sähköpostiviestien lähettämiseen tarkoitettu komponentti. Oletuksena se tukee vain SMTP:tä, mutta erillisen komponentin avulla sille on saatavissa tuki myös useisiin suosittuihin palveluihin kuten Amazon SES, Gmail, MailChimp, Mailgun, Postmark ja Sendgrid. Se tukee myös useita muita hyödyllisiä ominaisuuksia kuten korkeaa saatavuutta käyttäen ”failover” tekniikkaa, kuorman tasausta käyttäen ”round-robin” tekniikkaa, ”TLS Peer Verificationia” sekä asynkronista viestin lähettämistä. (Symfony 2020p, viitattu 10.7.2020.)

5.6.2 Messenger

Messenger tarjoaa viestiväylän, jonka avulla on mahdollista lähettää ja käsitellä viestejä heti sovelluksessa tai ne voi lähettää myös jonkin kuljetustavan (transport) kuten jonon kautta myöhemmin käsiteltäväksi. Oletuksena viestit käsitellään heti lähettämisen jälkeen. Jos viesti halutaan lähettää asynkronisesti, niin sille täytyy määritellä kuljetustapa, joka kykenee lähettämään viestejä esimerkiksi jonojärjestelmään ja sitten lukemaan niitä sieltä erillisen työprosessin avulla. Symfony tukee useita vaihtoehtoisia kuljetustapoja ja niitä voi tehdä lisää myös itse. (Symfony 2020f, viitattu 10.7.2020.)

5.6.3 Toteutus

Sähköpostien lähettäminen tapahtuu asynkronisesti käyttäen erillistä prosessia, joten se ei hidasta sovellusta tarpeettomasti, mikäli viestien lähettämisessä esiintyy ongelmia (Kuvio 33). Koska sovellus käyttää jo muutenkin Doctrinea tietokannan hallintaan, käytetään sitä myös viestien kuljetustapana. Kuviossa 34 on esimerkki lähetetystä sähköpostiviestistä Mailtrap-palvelussa. Myös sähköpostiviesteille on Twig-templaatit, joten niiden muokkaaminen on vaivatonta (Kuvio 35).



KUVIO 33: Sähköpostit asynkronisesti lähettävä työprosessi komentoriviltä

KUVIO 34: Esimerkki sähköpostista mailtrap -palvelussa

```
1  <h1>{{ email.subject }}</h1>
2
3  <p>
4      <a href="{{ target_url }}">{% trans %}email.anchor.open.text{% endtrans %}</a>
5  </p>
```

KUVIO 35: Sähköpostin Twig-templaatti

Sisäisesti viestien lähetys tapahtuu automaattisesti tapahtumien (event) avulla. Viestien lähettämistä varten sovellukseen tehtiin PurchaseRequestMailer.php -luokka, joka toteuttaa rajapinnan EventSubscriberInterface. Luokassa on käsittelijä tilakoneen tapahtumalle workflow.purchase_request.completed, joka suoritetaan aina tilasiirtymän jälkeen. Tapahtumakäsittelijässä luetaan siirtymän metatiedoista, onko siirtymälle määritelty sähköpostia, ja lähetetään sähköposti tarvittaessa.

5.7 Kieli

Sovelluksen tekstejä ei ole kirjoitettu suoraan sellaisenaan, vaan sovellus käyttää teksteihin Symfonyn käännöskomponenttia. Tekstit on pääosin kirjoitettu niiden tarkoituksen mukaan ja varsinainen teksti on käännöstiedostossa. Esimerkiksi ”button.save” käännetään ”Tallenna”. Tämä helpottaa sovelluksen mukauttamista ja ylläpitoa sekä mahdollistaa kansainvälistämisen helposti. Alla olevassa kuviossa on esimerkki käännöstiedostosta.

```
history.add_attachment: 'Lisättiin liitetiedosto'
label.comment: 'Kommentti'
label.attachment: Liitetiedosto
label.category: 'Kategoria'
label.free_category: 'Vapaa kategoria'
label.items: 'Laitteet'
label.information: 'Tiedot'
nav.main.section.user: 'Omat hankintapyynnöt'
nav.main.section.approver: 'Esimies'
nav.main.section.acquirer: 'IT-palvelut'
nav.user.queue: 'Vaatii toimenpiteitä'
nav.user.sent: 'Lähetetyt'
nav.user.closed: 'Käsitelty loppuun'
```

KUVIO 36: Osa käännöstiedostoa

6 POHDINTA

Opinnäytetyön aiheena oli toteuttaa web-sovellus hankintaprosessin tueksi Oulun Ammattikorkeakoululle. Tavoitteena oli yhtenäistää hankintaprosessia siten, että jatkossa kaikki hankinnat tapahtuisivat sen avulla. Sovelluksen tuli olla käytettävissä niin pöytäkoneilla, tableteilla kuin älypuhelimillakin. Sovellus toivottiin toteutettavaksi käyttäen PHP-ohjelmointikieltä, mutta muuten työvälineiden valinta oli vapaa. Minulla ei ollut aikaisempaa kokemusta PHP:stä muuten kuin opintojaksolta, jossa PHP-koodi liitettiin HTML-koodin sekaan, mutta en pitänyt sitä ongelmana.

Aiheen sain hyvissä ajoin syyslukukauden alussa 2019, mutta varsinaisen työn aloitin vasta vuoden vaihteessa. Aihe kypsyi mielessä syksyn aikana muita opintojaksoja suorittaessa. Koska PHP oli minulle uutta, halusin käyttää työssä jotain sovelluskehystä, joka antaisi sovellukselle hyvän rakenteen ja ohjaisi käyttämään hyviä käytäntöjä PHP:n kanssa. Syksyn aikana tutustuin sovelluskehystarjontaan. Valintani päätyi Symfonyyn joululoman aikana, jolloin testasin sen soveltuvuutta tekemällä yksinkertaisen Workflow -komponenttia hyödyntävän sovelluksen. Muita mainitsemisen arvoisia harkitsemiani vaihtoehtoja Symfonylle oli Laravel sekä CodeIgniter. Kaikki olisivat soveltuneet sovelluksen tekoon, mutta Symfony herätti erityisesti mielenkiintoni. Sen lisäksi Symfonyllä oli näistä paras tuki.

Opinnäytetyön teossa suurimmat haasteet liittyivät teoriaosioon, joka viivästyi useilla kuukausilla. Tavoitteena alun perin oli saada teoriaosuus ainakin puoliksi valmiiksi maaliskuun loppuun mennessä, mutta käytännössä sain sen aloitettua kunnolla vasta huhtikuussa. Teoriaosuudessa oli vaikeuksia fokuksen puuttumisen vuoksi. Oman haasteensa toi myös Symfony 5:n uutuus, jonka vuoksi hyviä kirjallisia lähteitä oli hankala löytää. Käytännössä Symfonyn osalta oli tukeuduttava pääosin sen omaan dokumentaatioon. Kesäkuun lopussa päätin muuttaa raporttia Symfony keskeisemmäksi, ja olla välittämättä Symfonyn dokumentaatioon kerääntyvistä viitteistä, minkä jälkeen työ etenikin heti paremmin.

Itse sovellus eteni kohtuullisen hyvää vauhtia heti tammikuusta alkaen ilman suurempia vaikeuksia. Sovelluksen toteuttamiseen liittyvät hankaluudet johtuivat suurelta osin tapaamisten järjestämisiin liittyvistä epäonnistumisista, jonka vuoksi välillä joutui työskentelemään turhankin pitkiä aikoja ilman sovittuja tavoitteita. Pääosin se oli omaa syytä ja sen olisi voinut välttää sopimalla tapaamiset aikaisemmin. Sairastumiselle taas ei voi mitään. Maaliskuussa sovellus oli kuitenkin tär-

keimmiltä osiltaan valmis. Tämän jälkeen sovellukseen tehtiin pääasiassa pieniä parannuksia ja siistimistä.

Sovelluksen tekoon liittyi paljon uutta opittavaa niin PHP:n kuin Symfonynkin osalta, mutta uutena asiana tuli myös responsiivisuus ja Bootstrap. Responsiivisuus oli käsitteenä tuttu, mutta en ollut koskaan tutustunut aiheeseen yhtään syvemmin ja Bootstrapin tiesin vain nimeltä. HTML ja CSS oli ennestään tuttuja, mutta ne ovat kehittyneet paljon siitä, kun niitä viimeksi käytin. Myös Javascriptin, Jqueryn ja Ajax:in osalta oli opittavaa, vaikka niiden käyttö jäikin vähäiseksi. Ajax -toiminnallisuuden lisäsin sovellukseen lopuksi parantamaan käytettävyyttä. Ajax toteutus on tehty yksinkertaisesti ja sen voisi tehdä paremmin, mutta se vaatisi sovellukseen isompia muutoksia. Sellaiseen ei kuitenkaan ollut aikaa.

Lopputuloksena syntyi sovellus, joka tekee sen, mitä sen on tarkoituskin tehdä. Sovelluksessa on kuitenkin mahdollisesti joitakin puutteita, jotka selviävät vasta käyttöönoton jälkeen. Ulkoasuun ei panostettu, pääpaino oli saada sivujen rakenne toimivaksi ja sivut mukautumaan eri kokoisille näytöille. Senkin osalta on mahdollista löytyä puutteita, koska sovellusta ei ole testattu kaikilla mahdollisilla selaimilla tai laitteilla, vaikkakin Bootstrap-kirjastolla on hyvä tuki useimmille. Sovelluksen suunnittelussa pyrittiin huomioimaan jatkokehitettävyys. Esimerkiksi tilakoneeseen voi yksinkertaisimmillaan lisätä uusia vaiheita koskematta lainkaan ohjelmakoodiin, ulkoasun muokkaaminen on helppoa Twig-templaattien ansiosta ja tekstien muuttaminen tapahtuu käännöstiedostossa.

Ensimmäisten kehitettävien asioiden joukossa tietysti on sovelluksen kehittäminen kattamaan kaikki hankinnat pelkkien IT-hankintojen sijaan. Myöhemmin sovellusta voisi jatkokehittää kattamaan koko hankintaprosessin. Nykyisellään sovellus kattaa vain prosessin alkupään hankintatarpeen synnystä hankintapäätöksen tekoon. Näiden lisäksi mieleen nousi joitakin pieniä parannuksia kuten se, että tällä hetkellä mikään ei periaatteessa estä henkilöä toimimasta itselleen hyväksyjänä tai hankkijana, paitsi ettei rajapinnasta sellaista esimiestietoa varmastikaan tule. Tilakoneeseen voisi myös lisätä hankkijalle uuden toiminnon, jolla voi pyytää lisätietoa käyttäjältä samalla tavoin kuin esimieskin voi. Tästä olisi se etu, että käyttäjä voisi saada siitä sähköpostiviestin, eikä joutuisi seuraamaan omien hankintapyyntöjen etenemistä niin aktiivisesti.

Omasta mielestäni aihe oli hyvä ja tarjosi sopivasti haasteita. Opittavia asioita oli paljon, mutta minulla oli aiempaa kokemusta joistakin sovelluskehyksistä, ja se loivensi muuten ehkä hieman jyrkkää oppimiskäyrää.

LÄHTEET

Bootstrap. 2020. About. Viitattu 10.6.2020, https://getbootstrap.com/docs/4.5/about/overview/.

Doctrine. 2020a. Basic Mapping. Viitattu 28.5.2020, https://www.doctrine-project.org/projects/doctrine-orm/en/current/reference/basic-mapping.html.

Doctrine. 2020b. Getting Started with Doctrine. Viitattu 28.5.2020, https://www.doctrine-project.org/projects/doctrine-orm/en/current/tutorials/getting-started.html.

getcomposer.org. 2020. Introduction. Viitattu 6.7.2020, https://getcomposer.org/doc/00-intro.md

JSON. 2020. Introducing JSON. Viitattu 16.4.2020, https://www.json.org/json-en.html.

Kouraklis, J. 2019. Introducing Delphi ORM: Object Relational Mapping Using TMS Aurelius. Viitattu 25.4.2020, https://learning.oreilly.com/library/view/introducing-delphi-orm/9781484250136/html/481234_1_En_1_Chapter.xhtml

MDN 2019. What is JavaScript? Viitattu 13.12.2019, https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.

MDN 2020a. Ajax. Viitattu 16.4.2020, https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX.

MDN 2020b. So what is HTML? Viitattu 20.7.2020, https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

PHP Group 2019. What is PHP? Viitattu 13.12.2019, https://www.php.net/manual/en/intro-whatis.php.

Potencier, F. 2009. Templating Engines in PHP. Viitattu 12.6.2020, http://fabien.potencier.org/templating-engines-in-php.html.

Symfony. 2020a. Configuring Symfony. Viitattu 14.7.2020, https://symfony.com/doc/current/configuration.html.

Symfony. 2020b. Databases and the Doctrine ORM. Viitattu 12.6.2020, https://symfony.com/doc/current/doctrine.html.

Symfony. 2020c. Defining Services Dependencies Automatically (Autowiring). Viitattu 12.7.2020, https://symfony.com/doc/current/service_container/autowiring.html.

Symfony. 2020d. How to Use PHP instead of Twig for Templates. Viitattu 28.5.2020, https://symfony.com/doc/current/templating/PHP.html.

Symfony. 2020e. Installing & Setting up the Symfony Framework. Viitattu 6.7.2020, https://symfony.com/doc/master/setup.html

Symfony. 2020f. Messenger: Sync & Queued Message Handling, viitattu 10.7.2020. https://symfony.com/doc/current/messenger.html

Symfony. 2020g. Routing. Viitattu 12.7.2020, https://symfony.com/doc/current/routing.html

Symfony. 2020h. Security. Viitattu 8.7.2020, https://symfony.com/doc/current/security.html.

Symfony. 2020i. Security User Providers. Viitattu 7.7.2020, https://symfony.com/doc/current/security/user_provider.html.

Symfony. 2020j. Service Container. Viitattu 12.7.2020, https://symfony.com/doc/current/service_container.html

Symfony. 2020k. Symfony Code License, Viitattu 11.6.2020, https://symfony.com/doc/current/contributing/code/license.html.

Symfony. 2020l. Symfony Components. Viitattu 6.7.2020, https://symfony.com/components.

Symfony. 2020m. Symfony versus Flat PHP. Viitattu 11.7.2020, https://symfony.com/doc/current/introduction/from_flat_php_to_symfony.html.

Symfony. 2020n. The Bundle System. Viitattu 6.7.2020, https://symfony.com/doc/current/bundles.html

Symfony. 2020o. The Guard Authenticator Methods. Viitattu 8.7.2020, https://symfony.com/doc/current/security/guard_authentication.html.

Symfony. 2020p. The Mailer Component, viitattu 10.7.2020. https://symfony.com/doc/current/components/mailer.html.

Symfony. 2020q. The Workflow Component. Viitattu 8.6.2020, https://symfony.com/doc/current/components/workflow.html.

Symfony. 2020r. Twig for Template Designers. Viitattu 28.5.2020, https://twig.symfony.com/doc/3.x/templates.html.

Symfony. 2020s. Twig | The flexible, fast, and secure template engine for PHP. Viitattu 28.5.2020, https://twig.symfony.com/.

Symfony. 2020t. Understanding how the Front Controller, Kernel and Environments Work together. Viitattu 11.7.2020, https://symfony.com/doc/current/configuration/front_controllers_and_kernel.html.

Symfony. 2020u. Upgrading Existing Applications to Symfony Flex. Viitattu 6.7.2020, https://symfony.com/doc/current/setup/flex.html

Symfony. 2020v. Workflows and State Machines. Viitattu 8.6.2020, https://symfony.com/doc/current/workflow/workflow-and-state-machine.html.

Tutorials Point. 2020a. JavaScript -- Overview. Viitattu 10.6.2020, https://www.tutorialspoint.com/javascript/javascript_overview.htm.

Tutorials Point. 2020b. Symfony -- Introduction. Viitattu 11.6.2020, https://www.tutorialspoint.com/symfony/symfony_introduction.htm.

Wikipedia. 2020a. Twig (template engine). Viitattu 28.5.2020, https://en.wikipedia.org/wiki/Twig_(template_engine).

Wikipedia. 2020b. Viitattu 10.6.2020, https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework).

Wikipedia. 2020c. Viitattu 5.7.2020, https://en.wikipedia.org/wiki/JQuery.

Zaninotto, F. & Potencier, F. 2007a. The Definitive Guide to symfony. Who Made Symfony and Why? Viitattu 21.4.2020, https://learning.oreilly.com/library/view/the-definitive-guide/9781590597866/ch01.html.

Zaninotto, F. & Potencier, F. 2007b. The Definitive Guide to symfony. Viitattu 21.4.2020, https://learning.oreilly.com/library/view/the-definitive-guide/9781590597866/ch02.html.

TOIMEKSIANTAJALTA SAATU PROSESSIKAAVIO LIITE 1