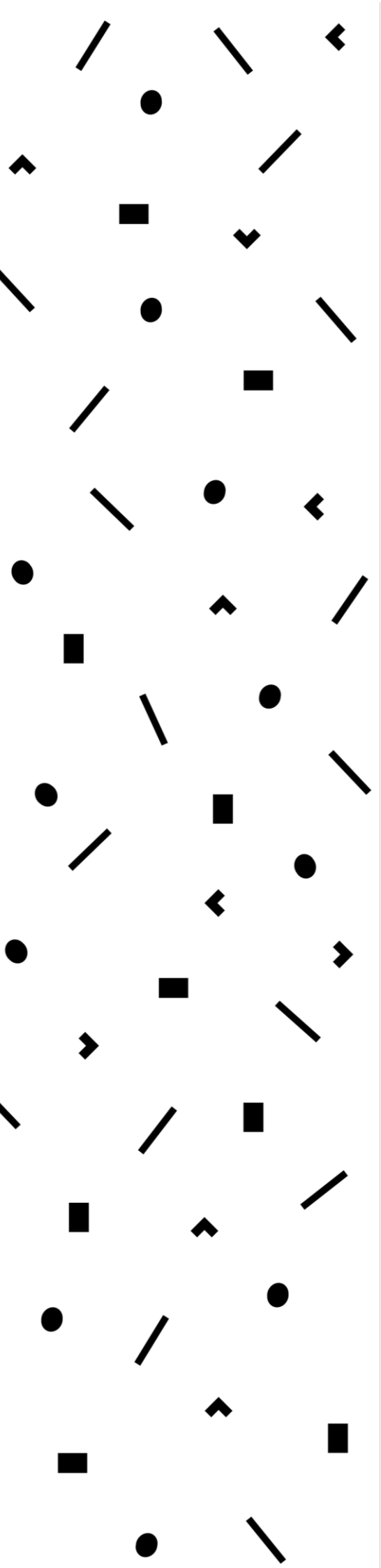


LAMK Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

Randomness-websovellus

Responsiivisuus ja frameworkit web-kehityksessä

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2016
Eeva Nikkilä

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

NIKKILÄ, EEVA:

Randomness-websovellus
Responsiivisuus ja frameworkit web-kehityksessä

Ohjelmistotekniikan opinnäytetyö, 36 sivua

Kevät 2016

TIIVISTELMÄ

Opinnäytetyön aiheena on web-sovelluskehys Symfony. Symfonysta annetaan perustiedot, mitä eri komponentteja se pitää sisällään ja millaisia lisäosia siihen voidaan liittää. Lisäksi vertaillaan, miten Symfony eroaa muusta web-kehityksestä.

Opinnäytetyön käytännön osuutena toteutettiin Randomness-sovellusprototyyppi, jonka tarkoituksena on inspiroida taiteen tekijöitä näyttämällä kuva- tai sana-slideshow'ita, joiden materiaalin käyttäjät tuottavat. Sovelluksen koodista annetaan muutamia esimerkkejä sekä esitellään sovellusta kuvin. Lisäksi kerrotaan hieman projektin sujumisesta ja siitä, miten hyvin se onnistui.

Lisäksi opinnäytetyössä tutkittiin, mitä responsiivisuus tarkoittaa, miten se toteutetaan ja miksi se on tärkeää. Myös web-sovellusten turvallisuutta sivuttiin Randomness-sovelluksen ja Symfonyn osalta. Opinnäytetyössä todettiin Symfonyn olevan hyvä työkalu laajempiin web-sovelluksiin, mutta pienemmille nettisivuille sitä ei välttämättä tarvita.

Asiasanat: Symfony, MVC, sovelluskehys, bundle, Doctrine, responsiivisuus

Lahti University of Applied Sciences
Degree Programme in Information Technology

NIKKILÄ, EEVA:

Symfony-webframework
Randomness web application using Symfony

Bachelor's Thesis in Software Engineering, 36 pages

Spring 2016

ABSTRACT

This thesis deals with a web framework called Symfony. The goal of the thesis was to give basic information about Symfony, for example what components it has and what plug-ins can be added. Lastly, there is some comparison between Symfony and other web developing.

The practical part of this thesis was a Randomness web application prototype, which was developed for Korpimedia Oy. The idea behind Randomness is to give artists inspiration by showing them pictures or words in a slideshow. The users of Randomness are also able to create their own slideshows. The thesis presents a few examples of the code and screenshots from Randomness. Lastly, the thesis tells how the project succeeded.

The thesis also examines what responsive web design means and how it can be implemented. It also considers why responsive web design is important. The importance of information security in web applications is also considered, and there are a few examples of how you can make a site more secure with Symfony.

This thesis found out that Symfony is a good tool to use when developing massive web applications, but for single web pages Symfony is not necessary.

Key words: Symfony, MVC, framework, bundle, Doctrine, responsive web design

SISÄLLYS

1 JOHDANTO

Web-ohjelmointi tarvitsee päivä päivältä kehittyneempiä työkaluja, koska ihmisten tarpeet eri palveluiden saamiseen verkossa kehittyvät. Lisäksi ohjelmoinnista pyritään tekemään tehokkaampaa ja nopeampaa. Tähän tarpeeseen on vastattu erilaisilla ohjelmointikirjastoilla sekä ohjelmistokehyksillä, jotka tarjoavat kehittäjälle valmiita komponentteja web-sovellusten tekemiseen, ja Symfony on yksi niistä.

Symfony on web-ohjelmistokehys, jolla voi tehdä nettisivuja ja -sovelluksia. Se tarjoaa monipuoliset työkalut web-kehittämiseen. Symfony käyttää ohjelmointikielinään PHP:tä, HTML:ää, JavaScriptiä ja CSS:ää.

Ohjelmistokehys on runko, jota voidaan täydentää kehitettävän ohjelmiston vaatimin tavoin. Ohjelmistokehys on siis vaillinaisesti toteutettu ohjelmisto, jossa on aukkoja täydennyksiä varten (Koskimies & Mikkonen 2005). Symfony tarjoaa kehittäjälle kansiorakenteen sekä nettisivupohjan MVC-mallilla toteutettuna, joka erittelee sovelluksen moduuliksi, näkymäksi ja kontrolleriksi. Lisäksi Symfonyyn voi ladata netistä tai tehdä itse bundleja eli paketteja, jotka ovat valmiita ohjelmakomponentteja. Esimerkiksi käyttäjän kirjautumiseen sekä tiedostojen lähettämiseen löytyy omat bundlensa.

Opinnäytetyön käytännön osuuden tarkoituksena oli toteuttaa Randomness-verkkosovellus Korpimedia Oy:lle. Korpimedia Oy on Lahdessa sekä Helsingissä toimivat yritys, joka tekee pääasiassa nettisivuja eri yrityksille (Korpimedia Oy 2015). Sovelluksen tarkoitus on toimia inspiraation lähteenä taiteilijoille, ja siinä voidaan tarkastella kuvia tai tekstiä slideshow’n muodossa. Käyttäjät pystyvät myös luomaan omia slideshow’itaan. Sovellus toteutettiin kesällä 2015 yhden henkilön voimin, ja sen pohjana käytettiin kahta käyttöliittymäkuvaa sekä suullista ohjeistusta siitä, mitä ohjelman tulisi sisältää.

Opinnäytetyö pyrkii tarjoamaan perustiedot Symfonysta kiinnostuneille suomen kielellä. Symfonylla on hyvä dokumentaatio nettisivuillaan, mutta kaikki tieto löytyy pelkästään englanniksi, eikä suomeksi löydy tietoa lähes

ollenkaan. Opinnäytetyössä käydään läpi Symfonyn perusominaisuudet, MVC-mallin mukainen kehitys sekä muutamia kolmansien osapuolten bundleja, joita voi Symfonyn itsensä lisäksi hyödyntää.

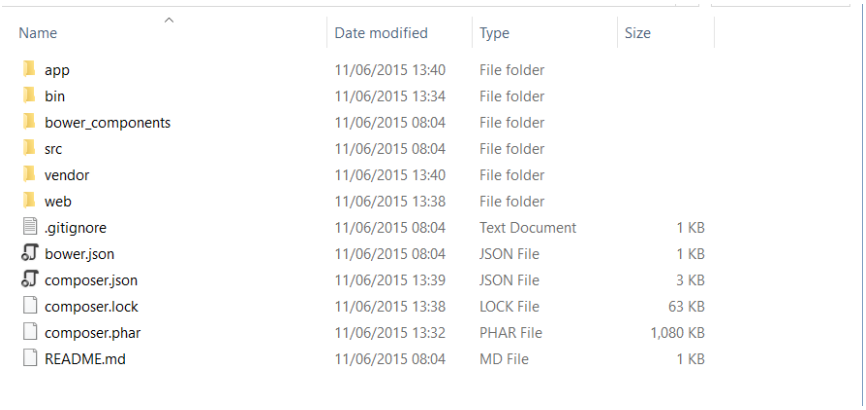
Symfonyn lisäksi opinnäytetyössä käydään läpi, mitä tarkoittaa responsiivisuus, miksi se on tärkeää ja miten se voidaan toteuttaa. Responsiivisuutta tarvitaan, jotta web-sivut ja -sovellukset toimivat eri laitteilla. Lisäksi pohditaan, mitä tarkoittaa tietoturvallisuus web-sovelluksen kannalta ja mitä apuja Symfony tarjoaa sen toteuttamiseen. Lopuksi vertaillaan ”perinteistä” web-kehitystä sekä Symfonya muihin web-kehyksiin.

Opinnäytetyössä kerrotaan yleisesti, millaisia asioita Symfony sisältää, mutta varsinaisia ohjelmointiohjeita se ei tarjoa. Muutamia kuvallisia esimerkkejä esimerkiksi controller- ja view-tiedostoista annetaan, jotta saadaan yleiskuva syntaksista. Symfonyn dokumentaatiosta löytyvät tarkemmat ohjeet Symfony-sovelluksen ohjelmointiin (SensioLabs 2016 b).

2 SYMFONY

2.1 Yleistä Symfonysta

Symfony on web-sovelluskehys, joka sisältää kokoelman erilaisia PHP-komponentteja. Symfony tarjoaa valmiita web-ohjelmiston osia, joiden pohjalta on helppo lähteä kehittämään omaa sovellusta. Symfony on SensioLabsin kehittämä, ja se on kehittänyt myös Symfonyn käyttämät Twigin ja Swift Mailerin (SensioLabs 2016 c).



Name	Date modified	Type	Size
app	11/06/2015 13:40	File folder	
bin	11/06/2015 13:34	File folder	
bower_components	11/06/2015 08:04	File folder	
src	11/06/2015 08:04	File folder	
vendor	11/06/2015 13:40	File folder	
web	11/06/2015 13:38	File folder	
.gitignore	11/06/2015 08:04	Text Document	1 KB
bower.json	11/06/2015 08:04	JSON File	1 KB
composer.json	11/06/2015 13:39	JSON File	3 KB
composer.lock	11/06/2015 13:38	LOCK File	63 KB
composer.phar	11/06/2015 13:32	PHAR File	1,080 KB
README.md	11/06/2015 08:04	MD File	1 KB

KUVIO 1. Symfony-projektin kansiorakenne

Symfony tarjoaa oman palvelimen sekä konsolisovelluksen, jonka avulla voidaan esimerkiksi luoda uusi Symfony-projekti sekä hallinnoida tietokantaa. Myös bundlet eli ohjelmistopaketit voivat käyttää konsolia hyväkseen. Symfony-projekti luodaan komentorivillä, ja komento luo oletusprojektin, jossa on muutama valmis sivu sekä kaikki projektin tarvitsemat tiedostot (kuvio 1). Esimerkki projektin luomisesta on kuviossa 2.

```
# Linux, Mac OS X
$ symfony new my_project_name

# Windows
c:\> cd projects/
c:\projects\> php symfony new my_project_name
```

KUVIO 2. Symfony-projektin luominen Linux-, Mac- ja Windows-ympäristöissä

2.2 Symfonyn asentaminen

Symfonyn asentaminen tapahtuu komentoriviä käyttäen. Kuviossa 3 on ohjeet Symfonyn asentamiseen Linux- tai Mac-ympäristössä (mikäli curl on asennettu). Windowsille asentaminen vaatii muutaman lisäasetuksen, mutta sillekin asentaminen on helppoa. Kun Symfony on asennettu, voidaan sitä käyttää komentorivillä komennolla symfony (Windowsissa php symfony).

```
$ sudo curl -LsS https://symfony.com/installer -o /usr/local/bin/symfony
$ sudo chmod a+x /usr/local/bin/symfony
```

KUVIO 3. Symfonyn asentaminen

Symfony-projektin voi kuitenkin luoda myös ilman sen asentamista. Tämä tapahtuu käyttämällä composer-ohjelmaa, jota käytetään usein myös bundlejen asentamisessa. Mikäli composer on asennettu koneelle, voidaan uusi Symfony-projekti luoda kuviossa 4 näkyvällä komennolla.

```
$ composer create-project symfony/framework-standard-edition my_project_name
```

KUVIO 4. Symfony-projektin luominen Composerilla

Kun Symfony on asennettu, esimerkkiprojektin pääsee näkemään käynnistämällä Symfonyn serverin ja menemällä selaimella osoitteeseen http://localhost:8000/. Symfony siis tarjoaa oman serverinsä, mutta sitä

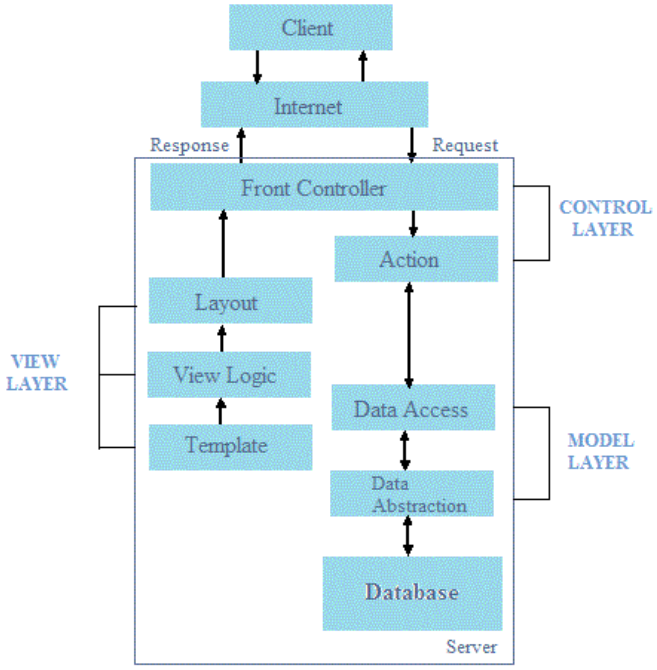
toki voidaan käyttää myös muiden servereiden kanssa. Kuviossa 5 on esimerkki serverin käynnistämisestä Windows-ympäristössä.

```
$ cd my_project_name/
$ php bin/console server:run
```

KUVIO 5. Symfonyn serverin käynnistäminen

2.3 Symfonyn komponentit

Kehitys Symfonyllä tapahtuu lähinnä PHP-kieltä käyttäen. Symfony-projekti perustuu MVC-malliin: erikseen on jaoteltu sivuston template (view) eli sivun ulkonäkö; controller eli sivun toiminnallisuudet; sekä model, joka yhdistää kaikki projektin osat ja hallitsee esimerkiksi tietokantahakuja. Symfonyssa varsinaista modelia ei ole kuin abstraktilla tasolla, vaan controllerit hallitsevat tietokantahaut. Havainnollistava kuva MVC-mallista on kuviossa 6.



KUVIO 6. MVC-malli (Anghel 2010)

Omaa projektia muokatessa kehittäjän ei yleensä ole tarkoitus koskea Symfonyn valmiisiin asetustiedostoihin. Muokattaessa projektia luodaan oma bundle /src-kansioon, jonka sisällä voidaan tarvittaessa kirjoittaa yli /vendor-kansiossa olevien valmiiden bundlejen asetuksia. Lisäksi muokkauksia täytyy tehdä /app-kansiossa sijaitsevaan AppKernel.php-tiedostoon, jossa kaikki bundlet lisätään projektiin. /app-kansiosta löytyy myös valmiiden sivujen reititystiedostot sekä näkymät. Omat reititys- ja näkymätiedostot on kuitenkin suositeltavaa laittaa oman bundlen sisälle. Yleinen nimi bundlelle on AppBundle, jota tässäkin opinnäytetyössä on käytetty.

2.3.1 Controller

Controller-tiedosto sisältää sivuston logiikan. Controllerissa voi olla esimerkiksi tietokantakutsut sekä oikean viewin näyttäminen sivulla. Controller kirjoitetaan PHP:llä, mutta Symfony sisältää joitain omia funktioita. Esimerkiksi tietokantakutsut tehdään Doctrinen avulla. Kuviossa 7 on esimerkki tietokantakutsusta. Ensin kutsutaan Doctrinen manageria, joka hoitaa yhteyden muodostamisen tietokantaan. Sen jälkeen pystytään helposti hakemaan haluttu tietokantataulu ja sieltä tässä tapauksessa ID:n mukaan kaikki käyttäjät. getUsers()-metodi on määritelty Account-entiteettiä luotaessa, ja se palauttaa käyttäjien tiedot.

```
$em = $this->getDoctrine()->getManager();
$account = $em->getRepository('AppBundle:Account')
        ->find($id)->getUsers();
```

KUVIO 7. Tietokantakutsu

Kuviossa 8 controller koskee sivua, joka on osoitteessa /accounts. Se hakee Account-nimisestä tietokantataulusta kaikki löytämänsä tilit. Mikäli asiakkaita löytyi, näytetään template browse.html.twig ja annetaan sille

lisäksi lista kaikista tileistä ’item’-muuttujassa. Tämän jälkeen template huolehtii asiakaslistauksen muotoilusta ja muista sivun ulkonäköseikoista.

```php
<?php
namespace AppBundle\Controller;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
use AppBundle\Entity\Account;

class AccountController extends Controller
{
    /**
     * @Route("/accounts", name="accounts")
     */
    public function showAccountsAction()
    {
        $em = $this->getDoctrine()->getManager();
        $account = $em->getRepository('AppBundle:Account')->findAll();

        if (!$account){$this->redirectToRoute('homepage');}

        return $this->render('AppBundle:Public:browse.html.twig',
                array('item' => $account));
    }

```

KUVIO 8. Esimerkki controller-tiedostosta

Controller-tiedostot sijaitsevat projektin bundle-kansiossa Controller-kansion alla, esimerkiksi /src/AppBundle/Controller. Symfonyssa käytetään CamelCase-nimeämistapaa tiedostojen nimissä, esimerkiksi bundle nimetään tyyliin MyBundle ja controller MyController.

2.3.2 View

Template-tiedostossa määritellään sivun ulkoasu eli view. Templateissa käytetään SensioLabsin kehittämää Twigiä, joka kääntää templatet php-koodiksi (SensioLabs 2016 c). Lisäksi templateihin voidaan helposti sisällyttää muita templateja: näin esimerkiksi yhtenäisten valikoiden ja rakenteiden tekeminen jokaiselle sivulle on helppoa.

Kuviossa 9 on esimerkki kuvion 8 Customer-controllerin templatesta. Lopputulos on esitetty kuviossa 10. Koodin ensimmäisellä rivillä kerrotaan,

että tässä templatessa käytetään layout.html.twig:ä, jossa määritellään sivuston navigaatiopalkki. Sen jälkeen määritellään, mistä Bundlesta käännöstiedostot otetaan. Käännöstiedostoja käsitellään tarkemmin luvussa 2.3.6. Käännösten määrittelyn jälkeen on title-block, jossa käytetään ensimmäistä kertaa käännettyä tekstiä.

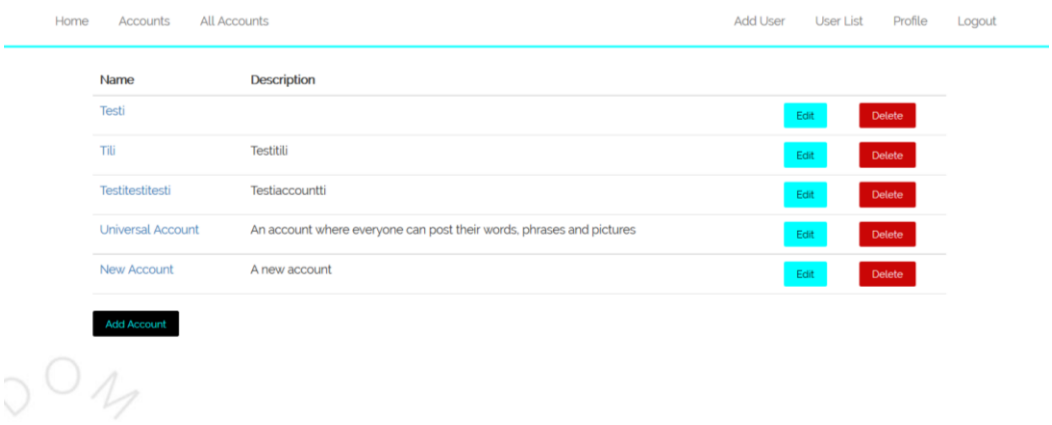
```
1  {% extends 'AppBundle::layout.html.twig' %}
2  {% trans_default_domain 'AppBundle' %}
3  {% block title %}{{ 'layout.allaccounts'|trans }}{% endblock %}
4  {% block body %}
5      <table class="table">
6          <tr>  <th>{{ 'layout.name'|trans }}</th>
7              <th>{{ 'layout.description'|trans }}</th>
8              <th colspan="2"></th></tr>
9                  {% for x in item %}
10             <tr>
11                 <td>
12                     <a href="{{ path('publicaccount',{'id': x.id}) }}">{{ x.name }}</a>
13                 </td>
14                 <td>{{ x.description }} </td>
15                 <td>
16                     <a href="{{ path('accountinfo',{'id': x.id}) }}" class="btn btn-sm btn-reverse">
17                         {{ 'layout.edit'|trans }}</a>
18                 </td>
19                 {% if is_granted('ROLE_ADMIN') %}
20                     <td>
21                         <a href="{{ path('deleteAccount',{'id': x.id}) }}" class="btn btn-sm btn-danger"
22                            onclick="return confirm('{{'layout.confirm'|trans}}');">
23                             {{ 'layout.delete'|trans }}</a>
24                     </td>
25                 {% endif %}
26             </tr>
27         {% endfor %}
28     </table>
29     <a href="{{ path('addaccount') }}" class="btn btn-sm btn-primary">{{ 'layout.addaccounts'|trans }}</a>
30 {% endblock %}
```

KUVIO 9. Esimerkki template-tiedostosta

Body-block rivillä 4 toimii samaan tyyliin kuin HTML:n body-tagit: niiden sisälle laitetaan sivun rakenne. Tämän sisällä on on taulukko, johon tulostetaan Randomness-sovelluksen ”tilit”. Randomnessia käsitellään lisää luvussa 4. Rivillä 9 oleva for-silmukka on Twigin omaa syntaksia. Item on controllerissa aiemmin määrätty muuttuja, joka sisältää kaikki tilit. Silmukan sisällä on taulukon solut, joihin haetaan tilien nimet, joissa on linkit tilin julkiseen sivuun. Tämän jälkeen on nappi tilin muokkaamiseen (accountinfo). Mikäli kirjautuneella käyttäjällä on admin-oikeudet, näytetään hänelle myös delete-nappi. Linkit noudattavat Symfonyn path-syntaksia: reititystiedoissa kerrotaan, mihin controlleriin linkistä pääsee, ja controller hoitaa sivun näyttämisen. Lisäksi linkkiä painettaessa lähetetään

controllerille tieto siitä, mikä valitun tilin id oli. Näin esimerkiksi deleteAccount-controller osaa poistaa oikean tilin.

Tietojen tulostamisen jälkeen for-silmukka suljetaan omalla sulkemistagillaan endfor. Lopussa on vielä linkki uuden tilin lisäämiseen, joka näkyy kaikille käyttäjille. Body-block suljetaan tagilla endblock body.



KUVIO 10. Accounts-sivun ulkonäkö admin-käyttäjälle

Twig-tiedoston pääte on .html.twig. Sen sisälle voidaan kirjoittaa normaalia html-koodia, mutta lisäksi voidaan käyttää Twigin omia php-komentoja. Twigin avulla voidaan tehdä blockeja eli lohkoja, joilla saadaan sivun rakenne selkeämmäksi (esimerkiksi sivupalkille tai navigaatiolle voidaan tehdä oma lohko) ja käyttää lohkoa suoraan muissa template-tiedostoissa. Lisäksi esimerkiksi for-silmukoiden käyttäminen onnistuu suoraan html:n keskellä. Myös controllerit helpottavat tietojen lisäämistä sivuille, kun looginen osuus voidaan tehdä sen puolella ja vain tulostaa lopputulema templatessa. Lisäksi sivuston sisäiset linkit ovat helposti muokattavissa: pathin nimi pysyy samana, vaikka kohteen osoitetta muutettaisiin routing- tai controller-tiedostossa.

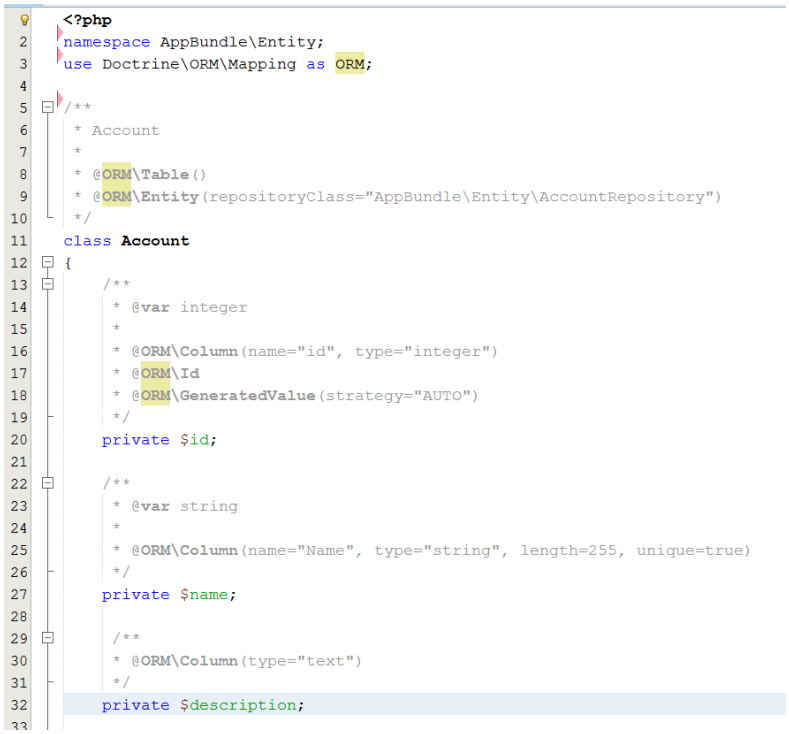
2.3.3 Routing

Routing eli reititys määrittelee sivuston osoitteet. Routing voidaan tehdä controller-tiedostossa, ennen tiettyä controlleria. Kuviossa 8 sivulla 7 voidaan nähdä esimerkki tästä riveillä 10 - 12. Reitti ilmoitetaan esimerkiksi lausekkeella ”@Route("/account", name="account")”, jossa /account on sivun aliosoite sivustolla ja nimi account se, jota sivuun viitattaessa käytetään esimerkiksi templatessa polkua (path) käytettäessä. Tällöin jos halutaan muuttaa sivun käyttäjille näkyvää nimeä, voidaan muuttaa osoitetta, mutta viitattua nimeä ei tarvitse vaihtaa, sillä se ei käyttäjille näy.

2.3.4 Doctrine

Symfonyn ORMina (Object-relational Mapping) käytetään Doctrinea. Sen avulla voidaan helposti luoda sekä hallinnoida tietokantaa Symfony-projektissa. Myös tietojen hakemisen, lisäämisen ja muokkaamisen toteuttaminen web-sovellukseen on Doctrinen avulla helppoa.

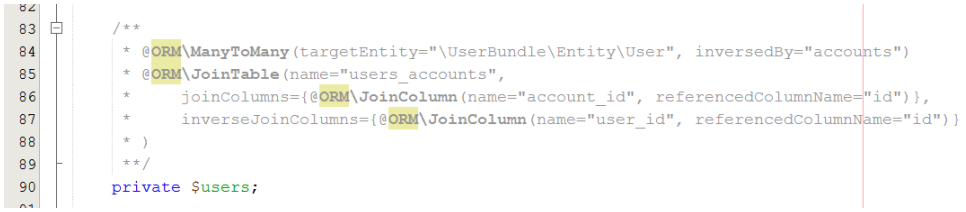
Tietokantaa varten luodaan entity-luokkia, jotka ovat käytännössä tietokantataulujen ominaisuuksien määrittelyjä. Näiden perusteella ORM luo tietokantataulut entiteeteille. Entity-tiedostot sijaitsevat projektissa kansiossa /src/AppBundle/Entity, ja jokaiselle tietokantaoliolle tehdään siis oma luokkatiedosto.



```php
<?php
namespace AppBundle\Entity;
use Doctrine\ORM\Mapping as ORM;

/**
 * Account
 *
 * @ORM\Table()
 * @ORM\Entity(repositoryClass="AppBundle\Entity\AccountRepository")
 */
class Account
{
    /**
     * @var integer
     *
     * @ORM\Column(name="id", type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    private $id;

    /**
     * @var string
     *
     * @ORM\Column(name="Name", type="string", length=255, unique=true)
     */
    private $name;

    /**
     * @ORM\Column(type="text")
     */
    private $description;
```

KUVIO 11. Entity-tiedoston alku, jossa määritellään tietokantataulun soluja

Kuviossa 11 on esimerkki Entity-tiedostosta. Kyseessä on Account-entiteetti, jonka nimi ja id haettiin templateen. Tiedostossa määritellään, mitä tyyppiä mikäkin Accountin ominaisuus on, sekä niille getter- ja setter-metodit. Lisäksi voidaan määritellä muiden tietokannan taulujen suhteet, joista on esimerkki kuviossa 12.



```php
    /**
     * @ORM\ManyToMany(targetEntity="\UserBundle\Entity\User", inversedBy="accounts")
     * @ORM\JoinTable(name="users_accounts",
     *      joinColumns={@ORM\JoinColumn(name="account_id", referencedColumnName="id")},
     *      inverseJoinColumns={@ORM\JoinColumn(name="user_id", referencedColumnName="id")}
     * )
     **/
    private $users;
```

KUVIO 12. Tietokantojen suhteiden määrittelyt

Tietokantataulun soluille voidaan määritellä myös muita ominaisuuksia, esimerkiksi Id generoituu automaattisesti ja on tyyppiä integer. Nimi taas on string, pituus voi olla maksimissaan 255 merkkiä ja sen täytyy olla

uniikki. Näiden alla on tavanomaiset getterit, jotka palauttavat haetun arvon, ja setterit, jotka määrittävät uuden arvon. Id:llä ei setteriä ole, koska se generoituu automaattisesti. Lisäksi voidaan tehdä muita funktioita kuten esimerkiksi ToString. Accountille voidaan määritellä tietyt käyttäjät, jotka pääsevät käsiksi sen muokkaamiseen. Käyttäjät säilötään users-soluun, josta on viittaukset User-luokkaan. Esimerkki users-solun luomisesta ja sen suhdemäärittelyistä on kuviossa 12. Entity-tiedostossa on tehty näiden käyttäjien hakemiselle, lisäämiselle ja poistamiselle omat funktiot, joista esimerkki kuviossa 13.

```
200     /**
201      * Add users
202      *
203      * @param \UserBundle\Entity\User $users
204      * @return Account
205      */
206     public function addUser(\UserBundle\Entity\User $users)
207     {
208         $this->users[] = $users;
209
210         return $this;
211     }
212
213     /**
214      * Remove users
215      *
216      * @param \UserBundle\Entity\User $users
217      */
218     public function removeUser(\UserBundle\Entity\User $users)
219     {
220         $this->users->removeElement($users);
221     }
222
223     /**
224      * Get users
225      *
226      * @return \Doctrine\Common\Collections\Collection
227      */
228     public function getUsers()
229     {
230         return $this->users;
231     }
```

KUVIO 13. Käyttäjän lisääminen, poistaminen ja hakeminen Entity-tiedostossa

2.3.5 Bundlet

Symfonyn perusominaisuuksien lisäksi siihen voi ladata lisää valmiita bundleja eli paketteja. Tällaisia ovat esimerkiksi FOSUserBundle, joka tarjoaa valmiin käyttäjähallinnan rekisteröitymisineen ja kirjautumisineen, sekä Iphp FilestoreBundle, joka toteuttaa tiedostojen lähettämisen palvelimelle. Lisäksi usein tarvitaan bundle muiden kirjastojen, kuten vaikkapa Bootstrapin, liittämiseen Symfony-projekteihin. Näitäkin onneksi löytyy valmiina reilusti. Bundlet ovat kolmansien osapuolten toteuttamia. Niitä voi kehittää myös itse, ja jokainen oma projekti olisi tarkoitus tehdä omaan bundleensa. Lisäksi myös sovelluksen eri osat voidaan eritellä omiin bundleihinsa.

```
1  $ composer require friendsofsymfony/user-bundle "~1.3"
```

KUVIO 14. Esimerkki bundlen asennuksesta

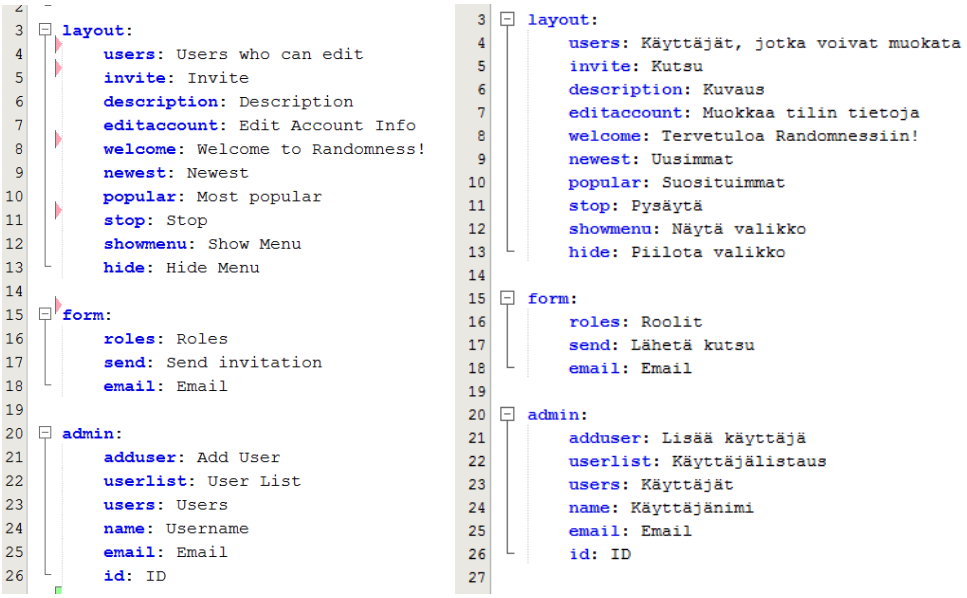
Bundlet asennetaan useimmiten Composeria apuna käyttäen. Tästä on esimerkki kuviossa 14. Composer lataa bundlen Symfonyn vendor-kansioon, mutta lisäksi bundle täytyy vielä aktivoida. Tämä tapahtuu app/AppKernel.php-tiedostossa, jossa on lista kaikista projektissa käytettävistä bundleista. Kuviossa 15 on esimerkki AppKernel-tiedostosta. Myös itse lisätyt bundlet tulee lisätä bundlelistaan. Suurin osa bundleista on jo valmiiksi lisättynä Symfonyyn projektin luomisessa, kuvion 15 esimerkissä vain neljä viimeistä bundlea on lisätty itse. Tiedostosta huomataan, että esimerkiksi Twig ja Doctrine ovat vain bundleja.

```php
<?php
use Symfony\Component\HttpKernel\Kernel;
use Symfony\Component\Config\Loader\LoaderInterface;
class AppKernel extends Kernel
{
    public function registerBundles()
    {
        $bundles = array(
            new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
            new Symfony\Bundle\SecurityBundle\SecurityBundle(),
            new Symfony\Bundle\TwigBundle\TwigBundle(),
            new Symfony\Bundle\MonologBundle\MonologBundle(),
            new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
            new Symfony\Bundle\AsseticBundle\AsseticBundle(),
            new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
            new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
            new Braincrafted\Bundle\BootstrapBundle\BraincraftedBootstrapBundle(),
            new AppBundle\AppBundle(),
            new FOS\UserBundle\FOSUserBundle(),
            new UserBundle\UserBundle(),
            new LoginBundle\LoginBundle(),
            new Iphp\FileStoreBundle\IphpFileStoreBundle(),
        );
        if (in_array($this->getEnvironment(), array('dev', 'test'))) {
            $bundles[] = new Symfony\Bundle\DebugBundle\DebugBundle();
            $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
            $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();
            $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
        }
        return $bundles;
    }
```

KUVIO 15. AppKernel.php-tiedosto

2.3.6 Käännökset

Symfony tekee helpoksi myös sivuston kääntämisen eri kielille. Jokaiselle kielelle voidaan määritellä omissa käännöstiedostoissaan, mitä sivulle tulostetaan. Käännöstiedosto määriytyy automaattisesti käyttäjän tietokoneen ja selaimen oletuskielen mukaan, tosin sen pystyy myös asettamaan koodissa.



```yaml
layout:
    users: Users who can edit
    invite: Invite
    description: Description
    editaccount: Edit Account Info
    welcome: Welcome to Randomness!
    newest: Newest
    popular: Most popular
    stop: Stop
    showmenu: Show Menu
    hide: Hide Menu

form:
    roles: Roles
    send: Send invitation
    email: Email

admin:
    adduser: Add User
    userlist: User List
    users: Users
    name: Username
    email: Email
    id: ID
```

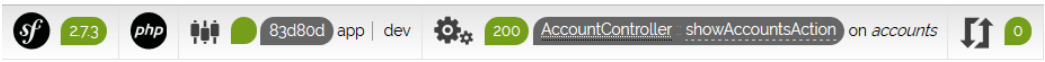
```yaml
layout:
    users: Käyttäjät, jotka voivat muokata
    invite: Kutsu
    description: Kuvaus
    editaccount: Muokkaa tilin tietoja
    welcome: Tervetuloa Randomnessiin!
    newest: Uusimmat
    popular: Suosituimmat
    stop: Pysäytä
    showmenu: Näytä valikko
    hide: Piilota valikko

form:
    roles: Roolit
    send: Lähetä kutsu
    email: Email

admin:
    adduser: Lisää käyttäjä
    userlist: Käyttäjälistaus
    users: Käyttäjät
    name: Käyttäjänimi
    email: Email
    id: ID
```

KUVIO 16. Käännöstiedostot

Kuviossa 16 on esimerkki englannin- ja suomenkielisistä käännöstiedostoista, jotka on kirjoitettu yaml-kielellä. Käännöstiedoston tiedostopääte on kieli.yaml, esimerkiksi AppBundle.en.yaml tai AppBundle.fi.yaml. Käännöstiedostot sijoitetaan bundlen sisällä Resources/translations-kansioon. Sivun 8 kuvion 9 esimerkissä template-tiedostosta näkyy, kuinka käännöksiä käytetään. Trans_default_domain määrittelee bundlen, josta tiedostot haetaan. ’Layout.editaccount’ |trans hakee yaml-tiedostosta kohdan, joka on layoutin alapuolella kohdassa editaccount. Yaml-tiedostossa nämä pitää olla sisennettynä, jotta tiedetään editaccountin kuuluvan juuri layoutin, eikä juuren, alapuolelle.

2.3.7 Kehittäjän työkalut

Symfony-projekti voidaan asettaa kahteen eri tilaan: kehittäjä ja julkinen. Niiden erona on se, että kehittäjän tilassa sivun alalaidassa näkyy Symfonyn oma työkalupalkki, jossa on useita eri toimintoja. Lisäksi virheiden sattuessa julkisessa tilassa näytetään vain sovelluksen oma error-sivu, jonka voi itse tehdä, kun taas kehittäjälle näytetään tarkempaa tietoa virheen luonteesta ja sijainnista.



KUVIO 17. Ensimmäinen puolikas kehittäjän työkalupalkista

Symfonyn työkalupalkissa vasemmassa reunassa (kuvio 17) kerrotaan esimerkiksi Symfonyn versionumero, statuskoodi (joka tässä tapauksessa on 200 eli success) sekä se, missä controllerissa ja sen funktiossa (action) ollaan. Palkin oikeassa reunassa (kuvio 18) näytetään, kuinka kauan sivun lataamiseen meni sekä sivun koko. Lisäksi kerrotaan, kuinka monta käännöstä tehtiin, ja ilmoitetaan, mikäli sanoja on kääntämättä. Tietokantahauista kerrotaan, kuinka monta niitä tehtiin ja miten kauan niissä meni. Lisäksi näytetään kirjautuneen käyttäjän tunnus.

KUVIO 18. Toinen puolikas kehittäjän työkalupalkista

Työkalupalkista päästään näkemään myös lisätietoja kuvakkeita klikkaamalla tai hiirellä osoittamalla. Esimerkiksi kaikki tietokantahaut sekä käännökset saadaan listattua ja virhetilanteissa nähdään, mikä niistä ei onnistunut.

2.3.8 Lomakkeiden käsittely

Lomakkeiden käsittely on usein haastavaa ja tarvitsee paljon erilaisia tarkistuksia, jotta se olisi turvallista. Symfonyssa lomakkeiden luomiseen on tehty oma komponenttinsa, mikä helpottaa niiden käsittelyä.

```php
// src/AppBundle/Controller/DefaultController.php
namespace AppBundle\Controller;

use AppBundle\Entity\Task;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\DateType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;

class DefaultController extends Controller
{
    public function newAction(Request $request)
    {
        // create a task and give it some dummy data for this example
        $task = new Task();
        $task->setTask('Write a blog post');
        $task->setDueDate(new \DateTime('tomorrow'));

        $form = $this->createFormBuilder($task)
            ->add('task', TextType::class)
            ->add('dueDate', DateType::class)
            ->add('save', SubmitType::class, array('label' => 'Create Task'))
            ->getForm();

        return $this->render('default/new.html.twig', array(
            'form' => $form->createView(),
        ));
    }
}
```

KUVIO 19. Lomakkeiden luominen

Kuviossa 19 on esimerkki lomakkeiden luomisesta. Task-entiteetti on aiemmin luotu ja sille on määritelty esimerkiksi setTask-metodi. Tämän jälkeen voidaan luoda lomake, jolla lisätään uusi task. Esimerkin riveillä 20 - 24 määritellään, mitä tietoja lomakkeella kysytään. Lomakkeeseen lisätään task- ja dueDate-input-kentät, joiden tietotyypit ovat teksti ja päivämäärä. Lomakkeeseen lisätään myös submit-nappula nimeltä save, jolle määritellään tekstiksi ”Create Task”. Lopuksi kutsutaan lomaketta. Funktio palauttaa render-funktion, joka näyttää sivun new.html.twig, ja antaa lisätietona kutsun luoda lomake sivulle. Tämän jälkeen lomake voidaan asettaa haluttuun kohtaan templateen, tästä esimerkki kuviossa 20.

```twig
{# app/Resources/views/default/new.html.twig #}
{{ form_start(form) }}
{{ form_widget(form) }}
{{ form_end(form) }}
```

KUVIO 20. Formin renderöinti templatessa

Lomakkeita voi tehdä Symfonyssa myös perinteisellä tavalla, mutta Symfonyn omalla lomakkeidenkäsittelijällä voidaan helposti päättää, mitä lomakkeen keräämälle tiedolle tehdään eli mille controllerille se ohjataan. Myös lomakkeiden validointi on tehty helpoksi.

2.3.9 Turvallisuus

Symfony tarjoaa helpot välineet verkkosovelluksen pääsyn rajoittamiseen, tämänhetkisen käyttäjän hakemiseen sekä muun turvallisuuden määrittelyyn. Perusturvallisuusasetukset löytyvät security.yaml-tiedostosta, joka sijaitsee hakemistossa app/config/. Kuviossa 21 on esimerkki tästä tiedostosta. Tiedostossa voidaan vaihtaa esimerkiksi palomuurin asetuksia.

```yaml
# app/config/security.yml
security:
    providers:
        in_memory:
            memory: ~

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false

        default:
            anonymous: ~
```

```yaml
    access_control:
        # require ROLE_ADMIN for /admin*
        - { path: ^/admin, roles: ROLE_ADMIN }
```

KUVIO 21. Turvallisuusasetustiedosto

Symfonyssa on myös sisäänrakennettuna pääsyn rajoittaminen eli authorisointi. Sovelluksen tietyille sivuille voidaan siis asettaa vaatimuksena, että käyttäjän tulee olla kirjautunut sisään tai hänellä tulee olla tietty rooli (esim. admin). Kuvion 21 alimmaisena on esimerkki siitä, kuinka määritetään, että kaikki verkko-osoitteet, jotka sisältävät polun /admin, ovat nähtävillä vain admin-käyttäjille. Myös tiettyyn controlleriin pääsyä tai templatessa tietyn osion näkemistä voidaan rajoittaa, jolloin käyttäjä kyllä näkee sivun, muttei pysty tekemään tiettyjä rajoitettuja toimintoja tai ei näe esimerkiksi tiettyä linkkiä sivulla. Näin voidaan käyttää samoja sivuja esimerkiksi kirjautuneille käyttäjille sekä admin-käyttäjille, ja vain rajoittaa näkyvyyttä tietyiltä osin. Esimerkit controllerin sekä viewin käytön rajoittamisesta on esitetty kuviossa 22.

```php
// ...
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Security;

/**
 * @Security("has_role('ROLE_ADMIN')")
 */
public function helloAction($name)
{
    // ...
}
```

```twig
{% if is_granted('ROLE_ADMIN') %}
    <a href="...">Delete</a>
{% endif %}
```

KUVIO 22. Pääsyn rajoittaminen controllerissa ja viewissä

Lisäksi Symfonyssa on helppoa tarkistaa, onko käyttäjä kirjautunut sisään ja kuka käyttäjä on kyseessä. Tästä on esimerkki kuviossa 23.

```php
public function helloAction($name)
{
    if (!$this->get('security.authorization_checker')->isGranted('IS_AUTHENTICATED_FULLY')) {
        throw $this->createAccessDeniedException();
    }

    // ...
}
```

KUVIO 23. Käyttäjän sisäänkirjautumisen tarkistaminen

Myös uloskirjautuminen on helppo määrittää Symfonyn avulla. Security.yaml-tiedostoon tarvitsee vain määrittää, mistä osoitteesta uloskirjautuminen tapahtuu, ja ohjata uloskirjautumislinkki kyseiseen osoitteeseen. Kuviossa 24 on esimerkki uloskirjautumisen määrittämisestä.

```yaml
# app/config/security.yml
security:
    # ...

    firewalls:
        secured_area:
            # ...
            logout:
                path:   /logout
                target: /
```

KUVIO 24. Uloskirjautumisen määrittely

Symfony siis tarjoaa hyvät työkalut rekisteröitymisen ja kirjautumisen tekemiseen itse. FOSUserBundle kuitenkin hoitaa sen automaattisesti. Pääsyn rajoittaminen ja käyttäjän tietojen saaminen on kuitenkin tärkeä osuus web-sovellusta, jossa käyttäjätileillä on väliä.

3 RESPONSIIVISUUS JA TIETOTURVALLISUUS

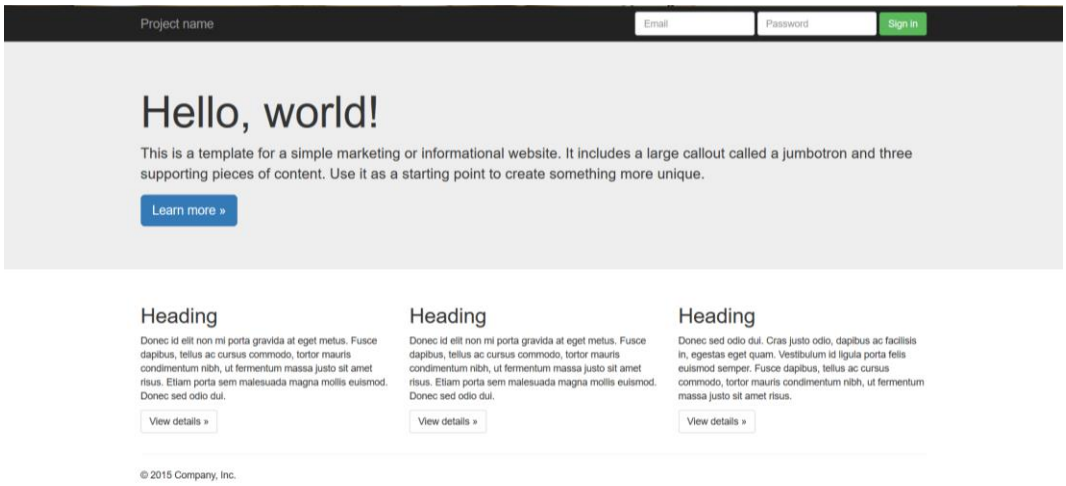
3.1 Responsiivisuuden tärkeys

Responsiivisuus on nykyään tärkeää jokaiselle nettisivulle ja -sovellukselle. Mobiililaitteiden käyttäminen kasvaa koko ajan, ja käyttäjät haluavat päästä käyttämään internetin palveluita missä tahansa, millä laitteella tahansa. Myös datayhteyksien paraneminen lisää mobiililaitteen käyttöä ensisijaisena tietotekniikkalaitteena. Responsiivisten nettisivujen ansiosta ei aina tarvitse tehdä omaa mobiilisovellusta jokaiselle verkkosivustolle tai -palvelulle.

Erityisen tärkeää responsiivisuus oli tämän opinnäytetyön osana toteutetussa Randomness-sovelluksessa, sillä sovellus on tarkoitettu taitelijoiden ja designereiden käytettäväksi, ja heidän työnsä on liikkuvaa, ja designereilla todennäköisesti on modernit mobiililaitteet käytössään. Lisäksi sovelluksen käyttöliittymä oli yksinkertainen ja idealtaan jo valmiiksi mobiililaitteille sopiva.

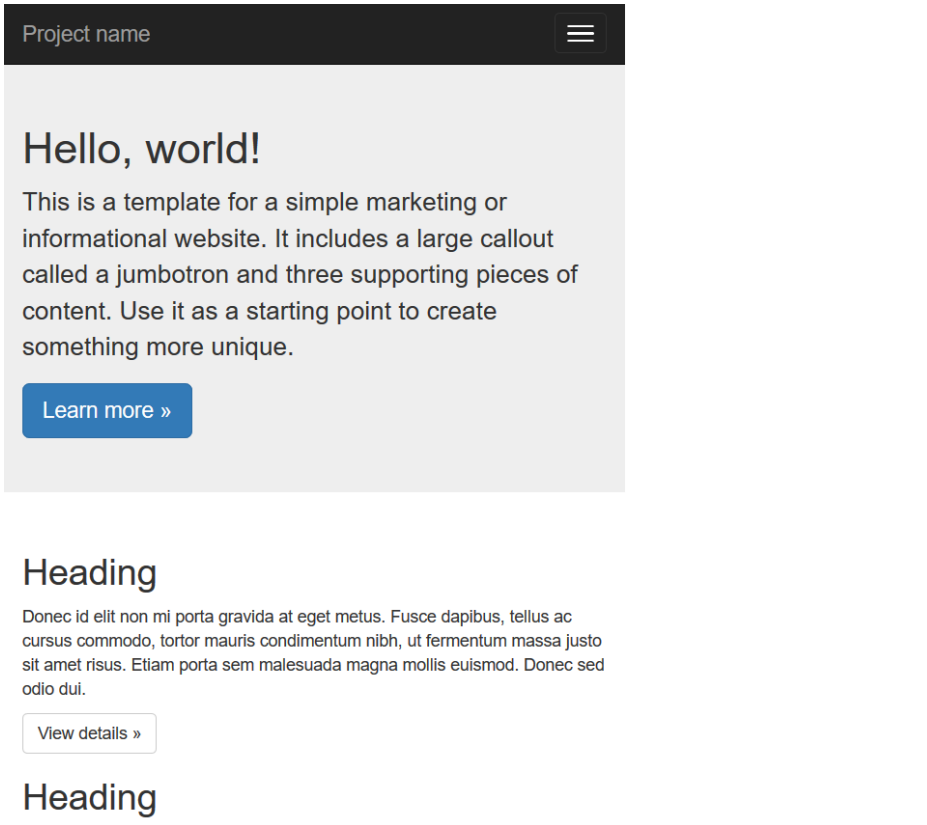
3.1.1 Responsiivisuuden määritelmä

Responsiivisuus tarkoittaa sitä, että nettisivusto toimii ja sitä on mukava käyttää eri laitteilla. Usein se tarkoittaa sitä, että nettisivun asettelu muuttuu, kun vaihdetaan ruudun kokoa. Sivuston sisältö kuitenkin pysyy ainakin suurimmaksi osaksi samana. Esimerkiksi tietokoneen selaimella, tabletilla ja puhelimella responsiivinen nettisivu näyttää erilaiselta. Lisäksi responsiivisuus voi tarkoittaa eri selaimien ja käyttöjärjestelmien välistä toimivuutta sekä käyttöliittymän ohjaustapoja, kuten kosketus- tai puheohjausta.



KUVIO 25. Pöytätietokoneen näkymä (Bootstrap 2016)

Kuvissa 25 ja 26 on esimerkit, millaiselta sama sivu voi näyttää pöytätietokoneella sekä mobiililaitteella.



KUVIO 26. Mobiililaitteen näkymä (Bootstrap 2016)

3.1.2 Responsiivisuuden toteuttaminen

Responsiivisuuden kolme tärkeintä osa-aluetta ovat sivun elementtien ja kuvien skaalautuminen, sekä laitteen tiedon saaminen sivun tietoon, jotta voidaan tarjota paras mahdollinen versio sivusta laitteen vaatimuksien mukaan. Sivun elementtien skaalautuminen toteutetaan usein grid- eli ruudukkosysteemillä, jolloin ruutujen asettelua voidaan muuttaa näytön koon mukaan. Kuvien ja tekstien skaalautuminen toteutetaan useimmiten suhteellisilla kokomerkinnöillä, esimerkiksi pikselileveyden sijaan käytetään prosentuaalista leveyttä. Laitteen tiedot saadaan selville CSS:n mediatiedusteluilla, jotka kertovat esimerkiksi käyttäjän näytön leveyden sekä käytössä olevan selaimen. (Wikipedia 2016.)

Käyttäjän näkökulmasta responsiivisuus toteutetaan useimmiten niin, että esimerkiksi kuvat ja tekstit pienenevät, valikot menevät piiloon ja osa sisällöstä voidaan ottaa kokonaan pois. Vain kaikista tärkeimmät asiat jätetään ja koristuksia karsitaan, esimerkiksi taustakuva, mainokset ja kuvituskuvat saatetaan ottaa pois. Usein valikot ovat "hampurilaisen" takana ja ne saadaan klikkaamalla auki. Lisäksi sormella hipaisemalla voidaan esimerkiksi mennä seuraavalle sivulle. Huomioon pitää ottaa myös hitaan datayhteyden päässä olevat käyttäjät, jotka eivät jaksa odotella raskaiden sivujen latautumista.

Koodaajan näkökulmasta responsiivisuus voidaan toteuttaa CSS:llä. CSS-tiedostoon kirjoitetaan erikseen tyylittelyt kullekin näytön leveydelle. Helppo tapa toteuttaa responsiivisuus on käyttää Bootstrap-kirjastoa, jota käytettiin myös Randomnessia tehdessä. Bootstrapissa on grid-systeemi, jolla sivun asettelu hoidetaan. Grideille voidaan asettaa eri leveydet eri näytönko’oille, jolloin pöytätietokoneen näytöltä katsottuna esimerkiksi kolme kuvaa voidaan laittaa vierekkäin, mutta älypuhelimen näytöllä ne menevät allekkain. Lisäksi Bootstrapilla on helppo toteuttaa responsiiviset valikot. Bootstrapin asentamisesta Symfony-projektiin kerrotaan lisää luvussa 4.3.2.

3.2 Tietoturvallisuus web-sovelluksissa

Toinen tärkeä asia web-kehityksen kannalta tässä opinnäytetyössä sekä web-sovelluksissa yleensä on tietoturvallisuus. Suuri osa web-sovelluksista käyttää tietokantaa sekä lomakkeita, ja ne voivat tehdä sovelluksen haavoittuviksi. Mikäli lomakkeita ei validoida eli tarkisteta, millaista tietoa lomakkeeseen on syötetty, millään tavalla, voi käyttäjä pystyä syöttämään esimerkiksi SQL-komentoja ja päästä näin käsiksi tietokantaan. Tätä kautta käyttäjä saattaisi pystyä muokkaamaan vaikkapa admin-käyttäjän salasanan haluamakseen ja päästä siten kirjautumaan sivustolle ylläpitäjätunnuksilla.

Symfonyssa lomakkeiden validointi on tehty helpoksi Symfonyn lomakkeenkäsittelijän ansiosta. Lomakkeen syötteille voidaan esimerkiksi asettaa tietyt tietotyypit, pituudet tai voiko syötteen jättää tyhjäksi. Kuviossa 27 on yksi esimerkki validoinnista. Siinä task ja dueDate eivät voi olla tyhjiä, ja lisäksi dueDate pitää olla tyyppiä DateTime. Tässä tapauksessa tiedon ei tarvitse myöskään tulla lomakkeen kautta, vaan myös esimerkiksi koodin kautta tapahtuva uuden taskin lisääminen validoidaan samaan tapaan.

```php
// src/AppBundle/Entity/Task.php
namespace AppBundle\Entity;

use Symfony\Component\Validator\Constraints as Assert;

class Task
{
    /**
     * @Assert\NotBlank()
     */
    public $task;

    /**
     * @Assert\NotBlank()
     * @Assert\Type("\DateTime")
     */
    protected $dueDate;
}
```

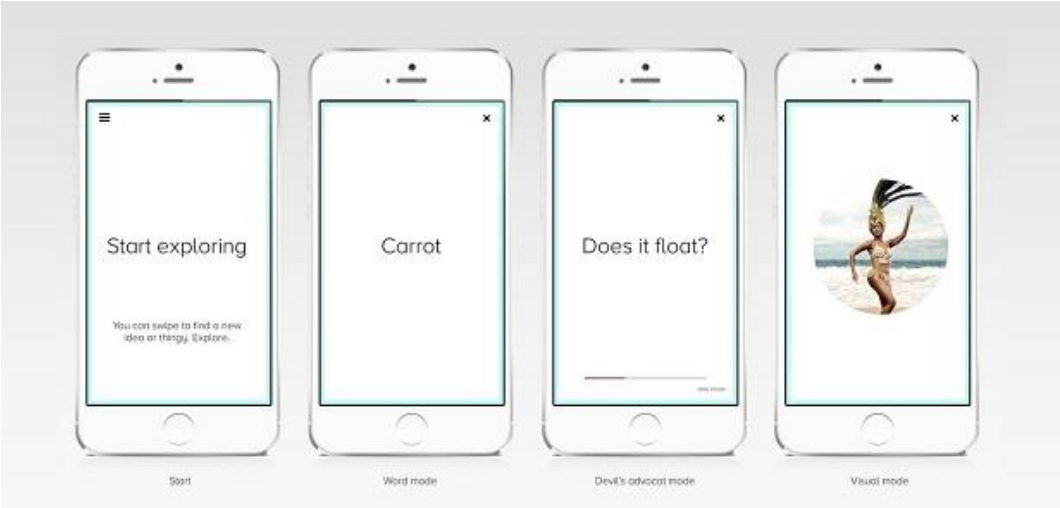
KUVIO 27. Validointi

Lomakkeiden validoinnin lisäksi Symfony tarjoaa turvallisuuteen käyttäjähallinnan. Symfonyn avulla on helppo estää tavalliselta tai kirjautumattomalta käyttäjältä pääsy tiettyihin sovelluksen osa-alueisiin. Esimerkiksi Randomness-sovelluksessa admin-käyttäjä pystyy poistamaan käyttäjiä sekä minkä tahansa käyttäjän luoman ”tilin”, kun taas käyttäjä pystyy poistamaan vain itse luomiaan tilejä. Sen sijaan jopa kirjautumattomat käyttäjät pääsevät selailemaan tilejä ja katselemaan slideshow’ita.

4 RANDOMNESS

4.1 Tavoite

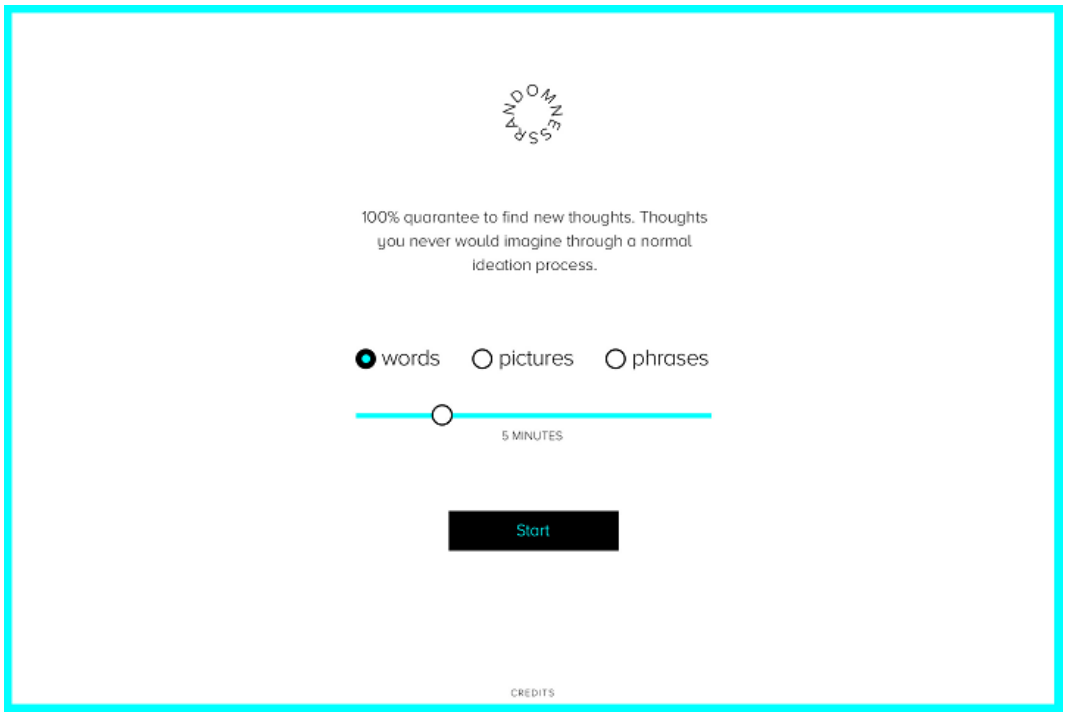
Randomness on Korpimedia Oy:n kehittämä websovellus, jonka tarkoituksena on tarjota inspiraatiota luovilla aloilla työskenteleville. Randomnessissa käyttäjä selaa kuvia, sanoja ja lauseita joko itse tai slideshow'na saadakseen uusia ideoita, joita ei olisi itse keksinyt ajatella. Käyttäjä pystyy valitsemaan esimerkiksi slideshow'n, joka esittää uuden satunnaisen kuvan aina 5 sekunnin jälkeen (kuvio 29). Slideshow'n pystyy pysäyttämään muistiinpanojen tekemisen ajaksi. Näytössä ei saa olla muita häiriötekijöitä selaamisen aikana: siispä käyttöliittymä on valkoinen ja pelkistetty, jotta käyttäjä voi keskittyä kuviin tai teksteihin ilman häiriötekijöitä (kuvio 28).



KUVIO 28. Käyttöliittymäsuunnitelma tilin esittelynäkymästä

Kuvien ja tekstien selaamisen lisäksi rekisteröityneenä käyttäjä voi itse lisätä niitä. Käyttäjät siis tekevät slideshow't toisille käyttäjille. Rekisteröitynyt käyttäjä pystyy tekemään uuden "tilin", johon lisätään kuvat ja tekstit. Lisäksi voidaan antaa muokkausoikeudet haluamilleen muille käyttäjille, jolloin tiimi voi tehdä oman "inspiraatiopankin". Jo olemassa olevien käyttäjien seasta sähköpostin mukaan etsimisen lisäksi sovelluksessa pitää pystyä kutsumaan käyttäjiä rekisteröitymään

sivustolle. Lisäksi on olemassa niin sanottu universaalitili, johon kaikki käyttäjät voivat lisätä kuviaan ja tekstejään. Kuvia ja tekstejä täytyy myös luonnollisesti pystyä muokkaamaan ja poistamaan.



KUVIO 29. Käyttöliittymäsuunnitelma slideshown valintanäkymästä

Randomness toteutettiin viime kesän aikana opiskelijaharjoittelussa Korpimedia Oy:n tarpeisiin. Sovelluksen täytyi olla responsiivinen, jotta sitä pystyisi käyttämään myös kännyköissä. Se toimii kuitenkin vain selaimessa, eikä sille tehty erillistä mobiilisovellusta.

4.2 Toteutus

Randomnessin tekemiseen käytettiin Symfonya. Slideshow’t toteutettiin Javascriptilla/jQuerylla. Erillisiä slideshow-kirjastoja ei käytetty. Tietokannan hallinnointiin käytettiin Doctrinea ja näkymien tekemiseen Twig-templateja. Lisäksi käytettiin Swiftmaileria sähköpostikutsujen lähettämiseen.

Sovelluksessa käytettiin käännöstiedostoja, tosin tällä hetkellä kielenä oli pelkästään englanti. Symfonyn käännöstenkäsittelyn ansiosta sovellus on

kuitenkin helppo kääntää eri kielille. Kaikki sovelluksen tekstit tulevat käännöstiedostosta, ja eri vaihtoehtoja tekemällä sovelluksesta voidaan tehdä globaali.

4.3 Bundlet

Randomnessin toteuttamiseen käytettiin muutamia tunnettuja Symfonyn bundleja. Bundlet nopeuttivat projektin tekemistä, sillä jokaista sivuston osaa ei tarvinnut tehdä alusta asti itse. Bundleiksi valittiin Friends of Symfony UserBundle, Braincrafted Bootstrap Bundle ja Iphp Filestore Bundle. Nämä toteuttivat sovelluksen käyttäjähallinnan, Bootstrapin lisäämisen projektiin sekä kuvien lataamisen palvelimelle käyttäjän toimesta.

4.3.1 Friends Of Symfony UserBundle

FOS UserBundle on Symfonyyn tehty bundle, joka hoitaa käyttäjähallinnan. Sen käyttämiseksi täytyy olla tietokanta, ja se tukee Doctrinea, MongoDB:tä ja Propelia. Sen on kehittänyt knpLabs (GitHub 2016).

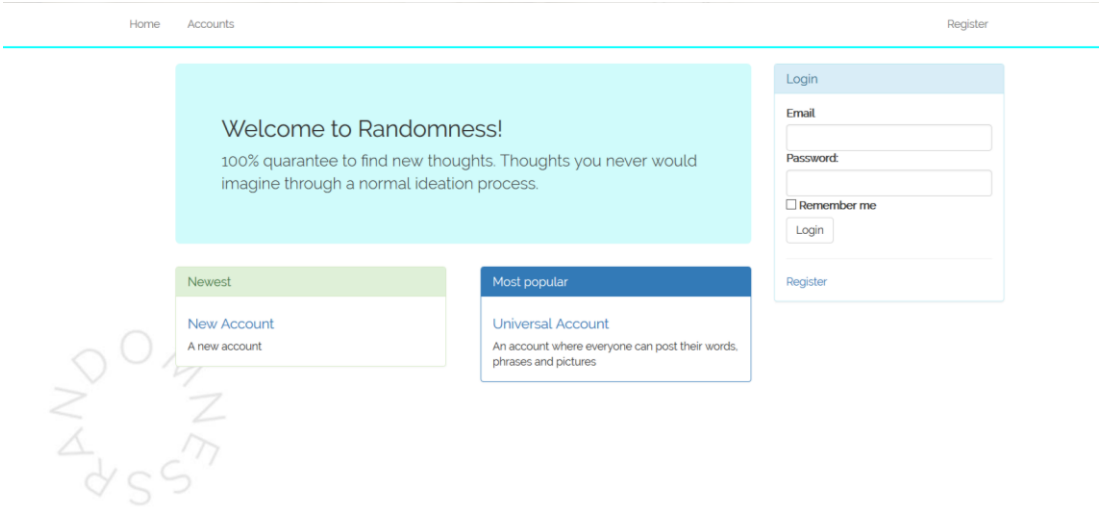
FOSUserBundle tarjoaa muun muassa käyttäjien rekisteröitymisen, kirjautumisen, roolien hallinnan ja salasanojen vaihdon. FOSUserBundlen asentamalla ei siis tarvitse käytännössä huolehtia ollenkaan käyttäjähallinnasta, sillä bundle tekee sen automaattisesti. Bundlessa on omat komentorivikomennot, joilla käyttäjiä voidaan luoda. Lisäksi valmiina on rekisteröitymis- ja kirjautumissivut. Nämä on jopa valmiiksi käännetty kymmenille eri kielille, mukaan lukien suomeksi.

Randomnessissa FOSUserBundlea käytettiin kaikkeen käyttäjähallintaan. Bundlen koodia itsessään ei tarvinnut muokata, ainoastaan tyylejä muutettiin. Lisäksi tehtiin adminille käyttäjälisäyssivu, jossa admin voi tehdä lisää admin-käyttäjiä. Admin pääsee myös tarkastelemaan käyttäjälistaa sekä poistamaan käyttäjiä. Bundlea sovellettiin myös

käyttäjän profiilisivun tekemiseen, jossa käyttäjä voi muokata käyttäjänimeään sekä muuttaa salasanaansa.

4.3.2 Braincrafted Bootstrap Bundle

Mikäli Symfony-projektissaan haluaa käyttää Bootstrapia, on se helpointa lisätä bundlen kautta. Yksi bundle tähän tarkoitukseen on Braincrafted Bootstrap Bundle, joka hoitaa Bootstrapin liittämisen Symfonyyn. Käytännössä muita käyttötarkoituksia bundlella ei ole, mutta ilman sitä Bootstrapin liittäminen olisi huomattavasti hankalampaa.



KUVIO 30. Randomnessin etusivu

Randomnessissa käytettiin Bootstrapia pohjatyylinä. Bootstrapin perustyylejä muokattiin kuitenkin paljon. Eniten hyötyä oli Bootstrapin grid-systeemistä, joka helpottaa sivun asettelua. Kuviossa 30 näkyy, kuinka eri elementit on aseteltu vierekkäin ja allekkain. Gridien avulla sovelluksesta oli helppo tehdä responsiivinen, ja esimerkiksi kuvion 30 sisäänkirjautumislomake siirtyy pienemmällä näytön koolla allekkain muiden elementtien kanssa.

4.3.3 Iphp Filestore Bundle

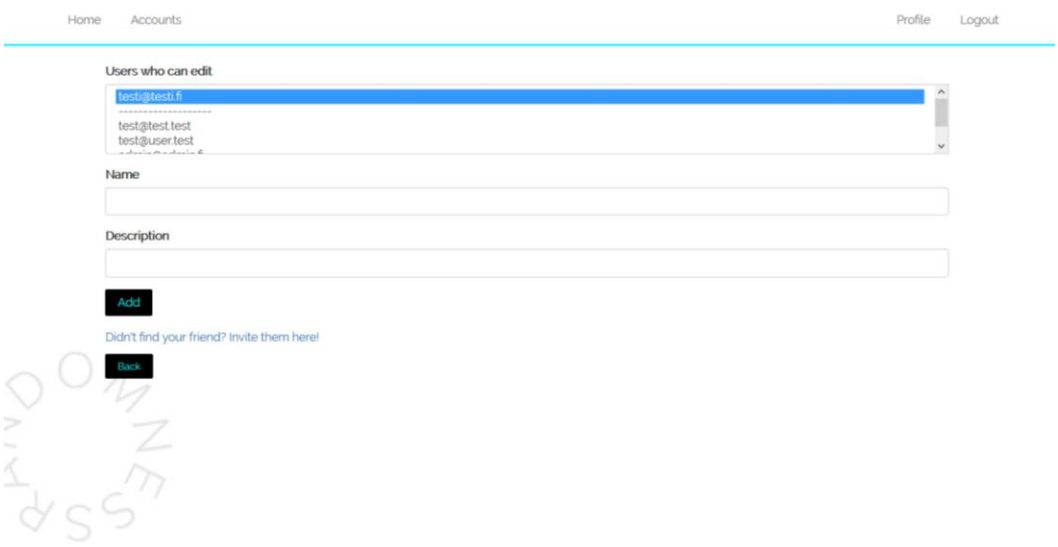
Koska Randomnessissa käytetään kuvia, täytyy niiden varastointiin olla jonkunlainen keino. Projektin alussa yritettiin tehdä kuvien käsittely ilman bundlea, mutta valmista bundlea käyttämällä sen toteuttaminen oli huomattavasti helpompaa. Onneksi tehtävän hoitamiseen löytyi ratkaisu Iphp FilestoreBundlesta, joka toteuttaa kuvien lisäämisen ja hakemisen kokonaan. Bundle lisää tietokantaan kuvan id:n ja osoitteen, ja lähettää tiedoston Symfonyn assets-kansioon, josta kuva on helppo hakea ja näyttää (kuvio 31).

KUVIO 31. Kuva-slideshow

4.4 Lopputulos ja kehittäminen

Lopputulos oli enimmäkseen tyydyttävä. Kaikki alussa määritetyt ominaisuudet saatiin toteutettua, mutta testaus jäi lopulta vähäiseksi. Esimerkiksi sähköpostikutsujen lähettämistä ei pystytty testaamaan luotettavasti. Lisäksi on mahdollista, että käyttäjän toimiessa odottamattomasti esimerkiksi Javascript-slideshown toiminta saattaa häiriintyä. Myöskään turvallisuutta ei ole testattu, joten sovellusta ei vielä

tällaisenaan pysty julkaisemaan. Ennen julkaisua ohjelmaa pitäisi siis testata mahdollisimman monen ihmisen toimesta, sekä rakentaa sovelluksesta turvallisempi. Lisäksi sovellus vaatisi jonkinlaista moderointia, mikäli se julkaistaan suurelle yleisölle.

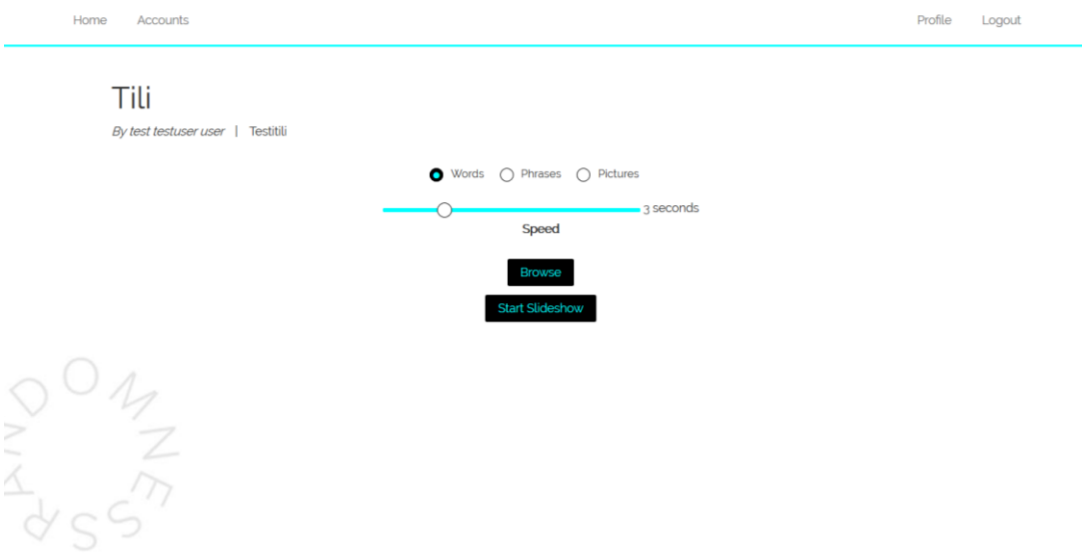


KUVIO 32. Uuden tilin luominen

Jatkokehitystä ajatellen ohjelmaa pitäisi testata ja mahdollisia virheitä korjata. Turvallisuustestaus on ainakin yksi tärkeä tehtävä, mikä pitäisi hoitaa ennen julkaisua. Lisää turvallisuudesta ja sen tärkeydestä kerrotaan luvussa 3.2. Myöskään responsiivisuutta ei päästy kokeilemaan oikealla mobiililaitteella, sillä kehitys tapahtui oman tietokoneen serverillä. Tulevaisuudessa sovelluksen voisi myös kääntää eri kielille, ainakin suomeksi. Lisäksi ulkoasussa olisi varmasti paranneltavaa, ja käyttäjillä olisi hyvä olla jonkinlainen viestiominaisuus. Tällä hetkellä käyttäjät voivat joko suoraan lisätä olemassa olevan käyttäjän tilin muokkaamiseen tai lähettää sähköpostikutsuja niille käyttäjille, jotka eivät ole jo sovelluksen käyttäjiä (kuvio 32). Olisi siis hyvä, jos myös sovelluksen sisällä pystyisi keskustelemaan muiden kanssa.

Kaiken kaikkiaan projekti onnistui olosuhteisiin nähden hyvin. Aikataulu oli laadittu kahdelle kehittäjälle, mutta lopulta sovellus tuli valmiiksi, vaikka toinen jättikin projektin kesken. Lisäksi sovelluksen kehittämistä varten

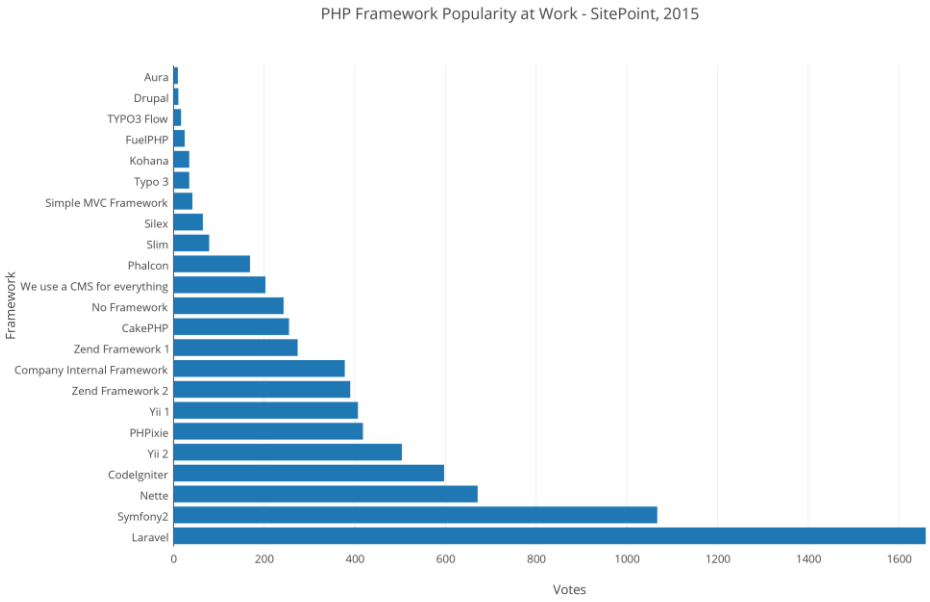
täytyi opetella monia täysin uusia asioita, kuten esimerkiksi JavaScript sekä Symfony, mutta siitäkin huolimatta jäljestä tuli tyydyttävää ja lopputulos vastaa hyvin käyttöliittymäsuunnitelmaa, kuten kuviosta 33 näkyy.



KUVIO 33. Slideshow’n aloitusvalikko

5 SYMFONY VERSUS MUU WEB-KEHITTÄMINEN

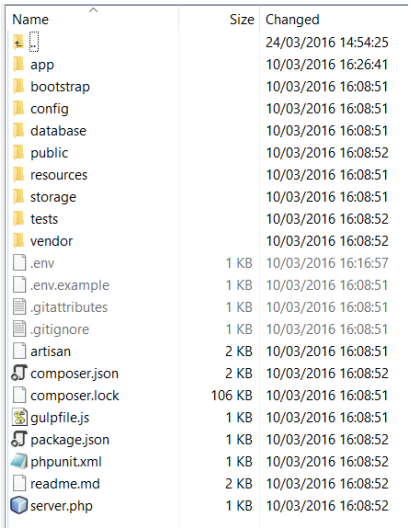
Web-kehysten käyttäminen verkkosovellusten kehittämisessä yleistyy, ja tarjolla on useampia PHP-kehyksiä, kuten Laravel, Zend ja CakePHP (Smith 2014). Näiden käyttäminen helpottaa usein kehittäjän työtä, vaikkakin uuden työkalun opettelussa saattaa mennä aikaa. Kun yhden ohjelmistokehyksen on oppinut, on kuitenkin helppo oppia toinenkin; ainakin Laravel on varsin samankaltainen Symfonyn kanssa, ja se käyttääkin joitain Symfonyn komponentteja (SensioLabs 2016 a). Varsinkin suuren web-sovelluksen, joka tarvitsee tietokantaa, kanssa on suositeltavaa käyttää jotakin ohjelmistokehystä. Lista suosituimmista PHP-sovelluskehyksistä on kuviossa 34, jossa käy ilmi, että Laravel on suosituin ja Symfony toisena (Skvorc 2015).



KUVIO 34. Suosituimmat PHP-kehykset (Skvorc 2015)

Symfonyssa esimerkiksi html-syntaksi eroaa normaalista Twigin takia, samoin kuin Controllereissa käytettävässä PHP:ssa on valmiita funktioita Doctrine-ORM:n ansiosta. Tämä helpottaa pidemmän päälle, kun esimerkiksi PHP-kieltä voidaan kirjoittaa lähes suoraan html-dokumenttiin ilman PHP-tageja, tarvitaan siis vain Twigin tarjoamat tagit kuten for ja if. Uusia tapoja voi olla aluksi hankala opetella, mutta ne nopeuttavat

kehittämistä. Symfonyyn on lisäksi tehty paljon valmiita komponentteja bundlejen muodossa, esimerkiksi käyttäjähallintaa eli kirjautumista ja rekisteröitymistä ei tarvitse tehdä itse, vaan kaiken saa valmiina pakettina. Toisaalta Symfonyn käyttäjiä on huomattavasti vähemmän kuin vaikkapa PHP:n tai Javascriptin, eikä kaikkiin ongelmiin välttämättä löydy valmista vastausta netistä. Symfonyn varsinainen dokumentaatio, vaikka kertookin kaikki tärkeimmät asiat, on loppujen lopuksi suppea, ja ongelmatilanteissa tieto löytyy lähinnä kolmansien osapuolten foorumeilta, mikäli jollakulla on ollut aiemmin sama ongelma.



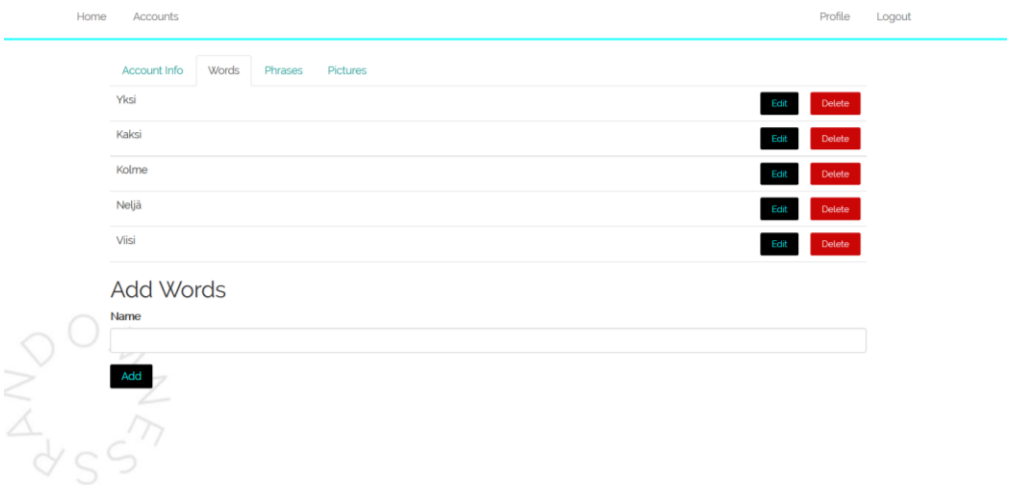
KUVIO 35. Laravelin kansiorakenne

Toinen PHP-ohjelmistokehys Laravel on hyvin samankaltainen Symfonyn kanssa, ja se käyttääkin muutamia Symfonyn komponentteja. Laravelissa on kuitenkin oma kansiorakenteensa, joka eroaa Symfonysta (kuvio 34). Perusperiaatteet ovat samat, kuten controller-tiedostot, viewit ja routet, mutta ne löytyvät eri paikoista ja käyttävät joissain asioissa erilaista syntaksia. Esimerkiksi näkymissä käytetään Twigin sijaan Bladea, joka tekee käytännössä samat asiat kuin Twig, mutta erilaisella syntaksilla. Twigissä kaikki blockit merkitään hakasulkeilla, kun taas Bladessa ne merkitään @-merkillä. Niitä kuitenkin käytetään samaan tapaan, eli esimerkiksi for- ja if-lausekkeita voidaan käyttää suoraan HTML:n lomassa. Tietokantojen luomisessa ja hallinnoimisessa käytetään Doctrinen sijaan Eloquent ORM:a.

Laraveliä kannattaa käyttää, mikäli haluaa ohjelmistokehykseltään helppokäyttöisyyttä sekä nopeaa prototyypitystä. Symfony taas on modulaarinen ja hyvin tuettu. Ohjelmistokehys kannattaa aina valita omien tarpeidensa sekä mieltymystensä mukaan. (Zen of Coding 2016.)

6 YHTEENVETO

Opinnäytetyö käsitteli Symfonya, joka on ohjelmistokehys, jonka tärkein ohjelmointikieli on PHP. Se tarjoaa web-kehittäjälle työkalut sovellusten tekemiseen, kuten esimerkiksi MVC-mallin mukaisen rakenteen, näkymien rakentamisen helposti Twigin avulla, debuggauksen sekä Doctrine-tietokantahallinnan. Lisäksi Symfonyyn on helppo kehittää omia tai lisätä valmiita lisäosia, joita kolmannet osapuolet ovat kehittäneet useita. Esimerkiksi käyttäjähallinnan saa sivuilleen helposti bundlen eli paketin lataamalla.



KUVIO 36. Esimerkki Randomnessin sanojen lisäämisestä

Opinnäytetyön käytännön toteutuksena toimi Randomness-verkkosovellus, joka kehitettiin Korpimedia Oy:n toimeksiannosta. Randomnessin tarkoituksena on antaa inspiraatiota taiteen tekijöille näyttämällä kuva- tai sana-slideshow'ita, joiden materiaalin käyttäjät tuottavat. Kuviossa 35 esimerkki sanojen lisäämisestä slideshow'hun.

Sovellus tuli suurimmaksi osaksi valmiiksi ja toimii hyvänä prototyyppinä, mutta sellaisenaan sitä ei voida julkaista. Kaikki ominaisuudet saatiin kuitenkin toteutettua ja sovellus on toimiva, lähinnä validoinnissa ja

authorisoinnissa saattaa olla aukkoja.

Työn valmistumista hankaloitti ryhmän toisen jäsenen jättäytyminen pois projektista mitään ilmoittamatta, ja vasta projektin loppuvaiheilla todettiin, että sitä on parasta jatkaa kokonaan yksin. Sovelluksen slideshow-ominaisuuteen käytettiin JavaScriptiä, eikä se ollut tuttua, joten sekin piti opetella projektia tehdessä. Myös Symfony oli jokseenkin uusi tekniikka kehittäjälle, joten senkin opetteluun meni aikaa. Projektin tekemiseen käytettiin kaksi kuukautta ja yhteensä noin 240 työtuntia.

Opinnäytetyössä selvitettiin myös, mitä responsiivisuudella tarkoitetaan ja miten se käytännössä toteutetaan. Responsiivisuudella on kasvava merkitys maailmassa, jossa yhä suuremmalla osalla ihmisistä on oma puhelin ja yhä useampi haluaa päästä käsiksi samoihin internet-sivuihin kuin pöytätietokoneellakin. Responsiivisten nettisivujen tekeminen mahdollistaa sen, että nettisivut näyttävät siltä kuin tekijä haluaa erilaisilla laitteilla. Myöskään mobiilisovelluksen tekeminen ei aina ole välttämätöntä, mikäli selainversio toimii hyvin myös kännykällä. Lisäksi mietittiin, mitä turvallinen web-sovellus tarkoittaa ja miksi se on tärkeää.

Lopuksi opinnäytetyössä käsiteltiin sitä, miten Symfonyllä kehittäminen eroaa muusta web-kehittämisestä, sekä vertailtiin Symfonyä hieman Laraveliin, joka on toinen web-sovelluskehys. Symfony helpottaa suurten web-sovellusten kehittämistä, mutta pienemmillä verkkosivuilla, jotka eivät käytä tietokantaa tai käyttäjien rekisteröitymistä, ei Symfonya välttämättä tarvita eikä edes kannata käyttää. Symfony vaatii myös hieman uuden opettelua, ja uudella tavalla esimerkiksi HTML-tiedostojen tekemiseen voi kestää hetken tottua.

Suurin ongelma Symfonyn opettelussa on se, että vaikka Symfonyn dokumentaatio onkin hyvä, ei siitä tahdo löytyä tietoa muista lähteistä. Varsinkaan suomenkielisiltä sivuilta ei löydy edes mainintoja Symfonysta. Opinnäytetyön tarkoituksena oli tarjota perustiedot Symfony-sovelluskehyksestä suomeksi siitä kiinnostuneille, ja tämä myös saavutettiin. Itse ohjelmointiin ei kuitenkaan haluttu antaa tarkkoja ohjeita,

sillä se olisi mennyt enemmän käyttöohjeen kuin opinnäytetyön puolelle. Myös muutamia Symfonyn ominaisuuksia jätettiin käsittelemättä ja keskityttiin tärkeimpiin. Lopulta Symfonyn todettiin olevan hyvä ohjelmistokehys suurempien web-sovellusten tekemiseen.

LÄHTEET

Anghel, O. 2010. Compose a MVC Paradigm for PHP with Symfony [viitattu 26.4.2016]. PHPBuilder. Saatavissa: http://www.phpbuilder.com/columns/Octavia_Anghel012110.php3

Bootstrap. 2016. Getting started – Examples [viitattu 26.4.2016]. Bootstrap. Saatavissa: http://getbootstrap.com/getting-started/#examples

GitHub. 2016. GitHub - FriendsOfSymfony/FOSUserBundle: Provides user management for your Symfony2 Project. Compatible with Doctrine ORM & ODM, and Propel [viitattu 19.3.2016]. GitHub. Saatavissa: https://github.com/FriendsOfSymfony/FOSUserBundle

Korpimedia Oy. 2015. Korpimedia Oy - Kumppanisi liiketoiminnan digitalisointiin [viitattu 21.4.2016]. Saatavissa: http://www.korpimedia.fi/

Koskimies, K. & Mikkonen, T. 2005. Ohjelmistoarkkitehtuurit. Helsinki: Talentum Media Oy

SensioLabs. 2016 a. Projects using Symfony [viitattu 19.3.2016]. SensioLabs. Saatavissa: http://symfony.com/projects

SensioLabs. 2016 b. Symfony, High Performance PHP Framework for Web Development [viitattu 19.3.2016]. SensioLabs. Saatavissa: http://symfony.com/

SensioLabs. 2016 c. Twig – The flexible, fast and secure PHP template engine [viitattu 19.3.2016]. SensioLabs. Saatavissa: http://twig.sensiolabs.org/

SensioLabs. 2016 d. Validation [viitattu 21.4..2016]. SensioLabs. Saatavissa: symfony.com/doc/current/book/validation.html

Skvorc, B. 2015. The Best PHP Framework for 2015: SitePoint Survey Results [viitattu 26.4.]. SitePoint. Saatavissa:

http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/

Smith, G. 2014. 13 PHP Frameworks to Help Build Agile Applications [viitattu 19.3.2016]. Mashable. Saatavissa: http://mashable.com/2014/04/04/php-frameworks-build-applications/#SN1ZgODo_GqE

Wikipedia. 2016. Responsive Web Design [viitattu 26.4.2016]. Wikipedia. Saatavissa: https://en.wikipedia.org/wiki/Responsive_web_design

Zen of Coding. 2016. The Great PHP MVC Framework Showdown Of 2016 – (CakePHP 3 Vs Symfony 2 Vs Laravel 5 Vs Zend 2) [viitattu 26.4.2016]. Zen of Coding. Saatavissa: http://zenofcoding.com/2015/11/16/the-great-php-mvc-framework-showdown-of-2016-cakephp-3-vs-symfony-2-vs-laravel-5-vs-zend-2/