jamk.fi

EcoReaction ja sosiaalinen media

Käytettyjen teknologioiden uusimmat versiot

Sami Pelkonen

Opinnäytetyö
Joulukuu 2015
Tekniikan ja liikenteen ala
Insinööri (AMK), Mediatekniikan koulutusohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Pelkonen, Sami	Opinnäytetyö, AMK	03.12.2015
	Sivumäärä 62	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
EcoReaction ja sosiaalinen media
Käytettyjen teknologioiden uusimmat versiot

Tutkinto-ohjelma
Mediatekniikan koulutusohjelma

Työn ohjaaja(t)
Pasi Manninen

Toimeksiantaja(t)
Wapice Oy

Tiivistelmä

Opinnäytetyö toteutettiin Wapice Oy:lle. Opinnäytetyön tavoitteena oli toteuttaa sosiaalisen median integraatio toimeksiantajan EcoReaction-energianseurantasovellukseen ja tutustua sovelluksessa käytettyjen teknologioiden uusimpiin versioihin.

Sosiaalisen median integraatio toteutettiin Spring-sovelluskehyksen Social-projektilla. Opinnäytetyössä tutkittiin, kuinka Spring Social -projektilla toteutetaan kirjautuminen sovellukseen sosiaalisen median tunnuksilla, ja miten sillä saadaan tehtyä julkaisuja sosiaaliseen mediaan. Opinnäytetyössä keskityttiin Facebookiin ja Twitteriin.

Opinnäytetyössä pohdittiin, mitä hyötyä sosiaalisen median integraatio tuo sovellukselle, ja mitä etua siitä on asiakkaille. Mietittiin myös tapoja, joilla esittää energiankulutusta mielenkiintoisella tavalla sosiaalisessa mediassa.

Työssä tutkittiin sovelluksessa käytettyjen teknologioiden uusimpia versioita: Java 8, Spring 4 ja Hibernate 5. Uusimpien versioiden tuomia hyötyjä ja uusia ominaisuuksia peilattiin tuotteessa käytettyihin vastaavuuksiin ja tutkittiin, mitä hyötyä nämä tuovat tuotteelle.

Tuloksena opinnäytetyöstä saatiin kattava selvitys, kuinka liittää sosiaalinen media osaksi EcoReaction-sovellusta Spring Social -projektilla ja käytettyjen teknologioiden uusien versioiden tuomista hyödyistä ja uusista ominaisuuksista. Tuloksena saatiin myös prototyyppi sosiaalisen median integraatiosta EcoReaction-sovellukseen.

Sovelluksessa käytetyt teknologiat kannattaa päivittää niiden uusimpiin versioihin. Varsinkin Java 8:n uudistettu Date and Time API tuo helpotusta päivien käsittelyyn ohjelmakoodissa.

Avainsanat (asiasanat)

Sosiaalinen media, Facebook, Twitter, Spring, Java, Hibernate

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Pelkonen, Sami	Bachelor's thesis	03.12.2015
	Number of pages 62	Language of publication: Finnish
		Permission for web publication: x

Title of publication

EcoReaction and social media

Newest versions of technologies in use

Degree programme

Media engineering

Tutor(s)

Pasi Manninen

Assigned by

Wapice Oy

Abstract

The purpose of the thesis was to integrate social media into EcoReaction-product and to get familiarized with the newest versions of the technologies used in it. EcoReaction is web application used to monitor energy consumption. The project was assigned by Wapice Oy.

The social media integration was executed with Spring framework's Social-project. The thesis studies how to execute the signing in to EcoReaction with a social media account and how to post in social media. The thesis focuses on Facebook and Twitter.

The thesis discusses how EcoReaction could benefit from social media integration and what added value would it provide to the customers. Ways of presenting energy consumption in an interesting way in social media were also reflected on.

The thesis studies the newest versions of the technologies used in EcoReaction: Java 8, Spring 4 and Hibernate 5. The benefits and features of the newest versions were reflected upon the similarities in use, and it was studied what EcoReaction could benefit from these.

The product of the thesis was an inclusive research on how to integrate social media into EcoReaction with Spring Social -project, and what benefits and features the newest versions of technologies in use could bring to EcoReaction. Also prototype of social media integration to EcoReaction was achieved with the thesis.

It proved to be worthwhile to update the versions in use, in particular the new Date and Time API of Java 8 which will relieve the handling of dates in code.

Keywords/tags (subjects)

Social media, Facebook, Twitter, Spring, Java, Hibernate

Miscellaneous

SISÄLTÖ

Kuviot

Taulukot

SANASTO

AOP
Aspect Oriented Programming. Ohjelmointimalli, jolla pyritään modulaarisuuteen.

HQL
Hibernate Query Language. Hibernaten kehittämä SQL:n kaltainen tietokantakyselykieli.

IOC
Inversion of Control. Malli, jossa sovellus ei kutsu sovelluskehyksen metodeja, vaan sovelluskehys kutsuu sovelluksessa määriteltyjä toteutuksia.

JavaBean
Java luokka, joka kapseloi monta oliota yhdeksi luokaksi.

JDBC
Java database connectivity. Javan rajapinta tietokantayhteyksiin.

JPA
Java Persistence API. Rajapintamääritelmä relaatiodatan hallinnasta Javassa.

JSP
JavaServer Pages. PHP:n tyyppinen teknologia, joka mahdollistaa dynaamisten verkkosivujen luonnin sekoittamalla Java-koodia HTML:ään.

JVM
Java Virtual Machine. Virtuaalikone, jonka päällä Java-ohjelmat pyörivät.

MVC
Model-View-Controller. Arkkitehtuurimalli, jossa erotetaan ohjelman osat eri kerroksiin.

OAuth
Autentikointiprotokolla, jossa käyttäjä valtuuttaa sovelluksen toimimaan puolestaan ilman tarvetta jakaa salasanaansa.

ORM
Object-relational mapping. Suunnittelumalli objektien kartoitukseen relaatiotietokantaan ja toisinpäin.

OSGi
The Open Services Gateway initiative. Spesifikaatio, jolla määritellään dynaaminen, modularisoitu järjestelmä.

SaaS
Software as a Service. Ohjelmiston jakotapa, jossa sovellus tarjotaan palveluna.

SDK
Software Development Kit. Ohjelmiston työkalupaketti, jolla voidaan tehdä ohjelmistoja ja toteuttaa toimintoja.

SQL
Structured Query Language. Kyselykieli tietokantahakuihin.

Tweet
Twitterissä julkaistava enintään 140 merkin mittainen julkaisu, joka voi sisältää myös kuvan tai videon.

1 TYÖN LÄHTÖKOHDAT

1.1 Tilaaja

Opinnäytetyön tilaajana oli Wapice Oy, joka on teollisuuden teknologiaratkaisuihin erikoistunut yritys. Yrityksen tehtävänä on keskittyä asiakkaiden suorituskyvyn edistämiseen kaikilla toiminnan aloilla hyödyntämällä tietotekniikkaa parhaimmillaan (Yritys n.d.).

1.2 Tehtävä ja tavoitteet

Opinnäytetyön tehtävänä oli toteuttaa sosiaalisen median integraatio EcoReaction-tuotteeseen käyttäen Spring-sovelluskehyksen Social-projektia. Tehtävänä oli myös tutustua EcoReactionissa käytettyjen teknologioiden uusiin versioihin ja selvittää, kuinka tuote päivitettäisiin tukemaan näitä, sekä minkälaista etua ne toisivat.

Tavoitteena oli saada toimiva prototyyppi kaksisuuntaisesta sosiaalisen median integraatiosta ja kirjautumismahdollisuudesta sosiaalisen median tunnuksilla EcoReactioniin. Tavoitteena oli myös saada selvyys EcoReactionin päivittämisestä tukemaan käytettyjen teknologioiden uusimpia versioita.

2 ECOREACTION JA SOSIAALINEN MEDIA

2.1 EcoReaction

EcoReaction on Wapice Oy:n kehittämä verkkopohjainen energiankulutuksenseurantasovellus. EcoReactionia markkinoidaan

energiayhtiölle ja siitä muokataan kunkin yrityksen ilmeen mukainen. Asiakasyritys tarjoaa muokattua EcoReactionia omille asiakkailleen energiankulutuksen seurantaan. EcoReactionin käyttöliittymä (ks. Kuvio 1) on toteutettu HTML5-tekniikoilla ja palvelinpuoli Javalla käyttäen Spring-sovelluskehystä.

Kuvio 1. EcoReaction

2.2 Sosiaalinen media

Käsitteestä sosiaalinen media tulee ensimmäisenä mieleen Facebookin ja Twitterin kaltaiset palvelut, joissa käyttäjät jakavat tietoa itsestään ja kiinnostuksensa kohteista. Sosiaalinen media on kuitenkin paljon enemmän. Wikipedian (Sosiaalinen media n.d.) mukaan Kaplan ja Haenlein (2010) määrittelevät sosiaalisen median seuraavanlaisesti: "Sosiaalinen media on

joukko internet-sovelluksia, joiden ideologinen ja tekninen perusta on Web 2.0:ssa ja jotka mahdollistavat loppukäyttäjien tuottaman sisällön luomisen ja välittämisen".

Web 2.0:lla tarkoitetaan 2000-luvun vaihteessa syntynyttä muutosta internetin sisällössä, joka aiemmin oli hyvin staattista. Internetin käyttäjillä ei ollut keinoja vaikuttaa sisältöön tai tuottaa dataa. Web 2.0:ssa käyttäjät muokkaavat verkon sisältöä ja tuottavat dataa reaaliaikaisesti. (Rouse & Haughn 2015.)

Tämän määrityksen perusteella sosiaalisiksi medioiksi luetaan myös muun muassa Snapchat, YouTube, Wikipedia, kaikenlaiset blogit sekä videopelaajien suosiossa oleva videoiden suoratoistopalvelu Twitch.

Sosiaalisista medioista ylivoimaisesti suosituin on Facebook 1,49 miljardilla kuukausittaisella käyttäjällä. Twitter on listalla vasta seitsemäntenä 316 miljoonalla käyttäjällä. Twitterin ohi menevät muun muassa sellaiset palvelut kuin WhatsApp ja Facebook Messenger, jotka molemmat ovat Facebookin omistuksessa. (Global social networks ranked by number of users 2015 2015.)

2.3 Sosiaalisen median integrointi EcoReactioniin

2.3.1 Sosiaalisen median tuoma lisäarvo sovellukselle

Sosiaalisen median läsnäolo sovelluksessa kuin sovelluksessa on nykyaikana lähestulkoon normi. Käyttäjät ovat tottuneet jakamaan ja julkaisemaan omasta elämästään ja käyttämistään sovelluksista kaikennäköistä tietoa. Tässä valossa on helppo olettaa, että sosiaalisen median integrointi EcoReactioniin on luonnollinen kehitysaskel.

Energiayhtiöt saavat näkyvyyttä, kun käyttäjät jakavat sosiaaliseen mediaansa julkaisuja suoraan sovelluksesta. Tämä tuo energiayhtiöille selvää kilpailuetua. Kuluttajat eivät välttämättä ole edes tietoisia energiayhtiöiden tarjoamista energianseurantapalveluista, jos energiayhtiöt eivät itse mainosta aktiivisesti niitä.

Sosiaalisen median integroinnissa on kolme aspektia, joita tarkastella: kirjautuminen sovellukseen sosiaalisen median tunnuksilla, oman tiedon jakaminen sovelluksesta sosiaaliseen mediaan ja tiedonjako sosiaalisesta mediasta sovellukseen.

2.3.2 Kirjautuminen EcoReactioniin sosiaalisen median tunnuksilla

Yhä useampi sovellus ja palvelu tarjoaa mahdollisuuden rekisteröityä sen käyttäjäksi sosiaalisen median tunnuksilla. Tällä tavoin käyttäjän elämä helpottuu, kun ei tarvitse muistaa jokaiseen palveluun eri tunnuksia. Palvelun näkökulmasta ei välttämättä tarvitse toteuttaa erillistä rekisteröitymis- ja kirjautumismekanismia. Useasti kuitenkin sosiaalisen median tunnuksilla kirjautumista tarjotaan perinteisen kirjautumismenetelmän rinnalla.

EcoReactionin kohdalla kirjautuminen sosiaalisen median tunnuksilla on mutkikkaampi juttu. Sovelluksen kirjautumismenetelmän konfigurointi riippuu pitkälti asiakkaan ympäristöstä. Yleensä käyttäjä kirjautuu ensin asiakkaan ympäristöön ja tämän jälkeen ohjautuu EcoReactioniin. Tällaisessa tapauksessa erillistä kirjautumismekanismia ei tarvitse toteuttaa EcoReactioniin.

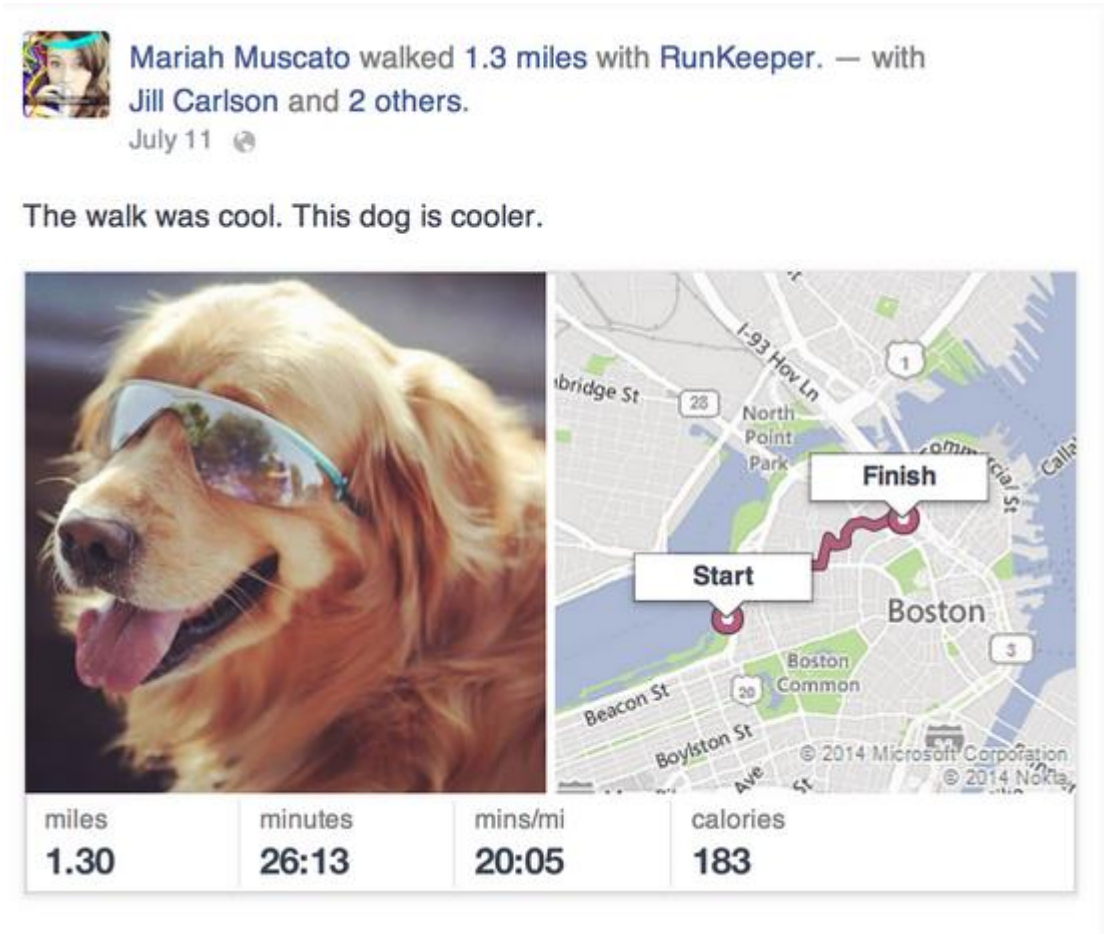
Siinä tapauksessa, että EcoReaction toimii täysin omana palvelunaan irrallaan asiakkaan ympäristöstä, voi sosiaalisen median tunnuksilla kirjautuminen tuoda lisäarvoa sovellukselle. EcoReactionin käyttäjien ei tarvitse erikseen

rekisteröityä palveluun, vaan heillä on yleensä sinne suora pääsy tehtyään sopimuksen energiayhtiön kanssa. Käyttäjä voisi liittää sosiaalisen median tilinsä EcoReactioniin ja jatkossa kirjautua palveluun niillä tunnuksilla.

Tehdäkseen julkaisuja sosiaaliseen mediaan EcoReactionin kautta tulee käyttäjän valtuuttaa EcoReaction tekemään julkaisuja puolestaan. Tämä tarkoittaa käytännössä käyttäjän tunnusten linkittämistä sosiaalisen median tiliin.

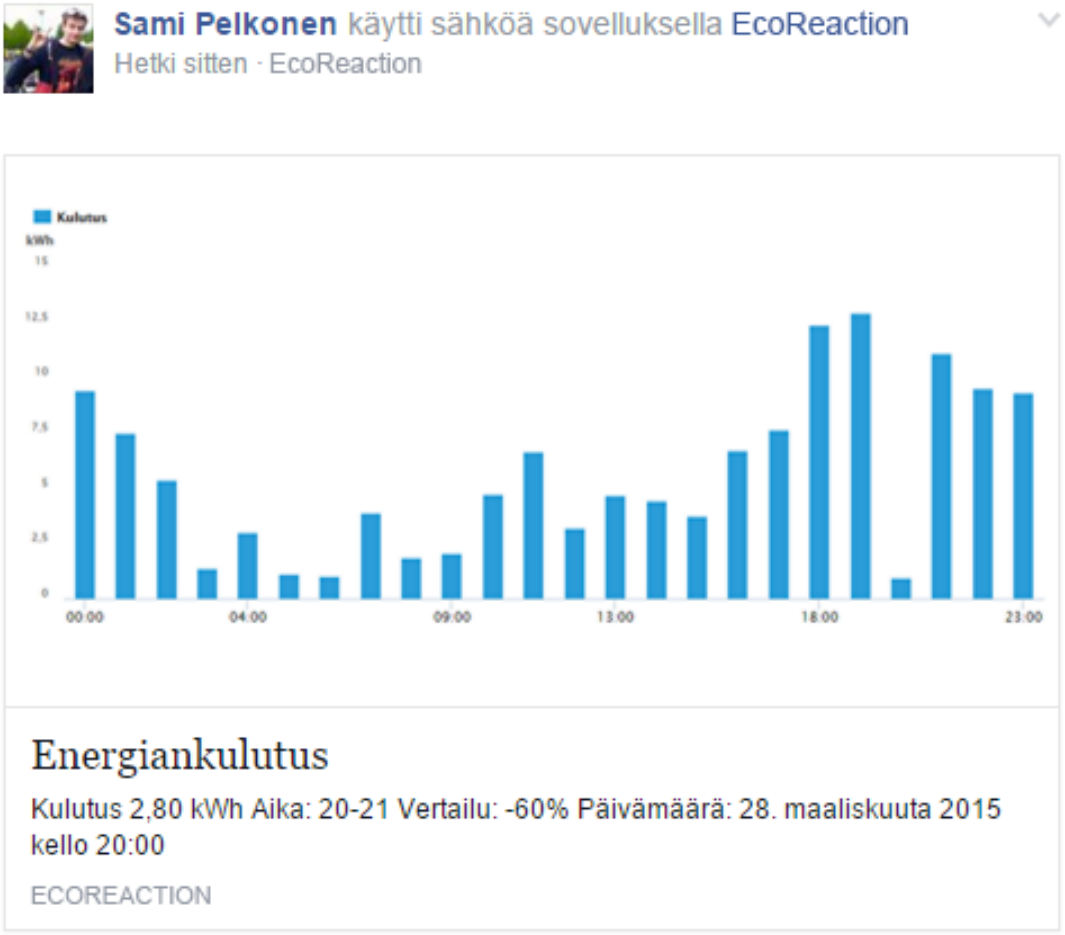
2.3.3 Tiedon jakaminen EcoReactionista sosiaaliseen mediaan

Sovelluksien käyttäjät jakavat sovelluksien keräämiä tietoja itsestään sosiaaliseen mediaan. Yleensä sovellus keskittyy jonkin tietyn asian seuraamiseen ja tietojen keräämiseen. Esimerkiksi Runkeeper seuraa käyttäjän juoksusuorituksia, ja käyttäjä voi jakaa lenkkejään sosiaaliseen mediaan suorituksen tietojen kera. Jaettavia tietoja ovat muun muassa juostu matka, käytetty aika ja kulutetut kalorit. Käyttäjä voi jakaa myös kartan juostusta lenkistä sekä matkallaan ottaman kuvan (ks. Kuvio 2).



Kuvio 2. Runkeeper-suorituksen jako Facebookiin (What to expect when you’re expecting to share n.d.)

EcoReactionista käyttäjä voisi jakaa energiankäyttöönsä liittyviä tietoja. Mahdollisia sovelluksesta jaettavia tietoja on energiankulutus eri aikatasoilla: tunti-, päivä-, viikko-, kuukausi- ja vuositasolla. Energiankulutuksen trendit ja vertailu ulkolämpötilaan voi olla kiinnostavia tietoja jaettavaksi. Jakoon voisi liittää myös kuvion energiankäytöstään. Kuvio voisi olla samanlainen kuin EcoReactionissa esitettävä. (Ks. Kuvio 3.)



Kuvio 3. Facebook-julkaisu energiankulutuksesta

Tärkeä kysymys esitettäväksi lienee, mikä on käyttäjän näkökulmasta mielenkiintoista tietoa jaettavaksi sosiaaliseen mediaan. Normaalin sähkönkulutuksen jakaminen ei saata olla kovin houkuttelevaa tietoa julkaistavaksi. Erilaiset epänormaaliudet ja saavutukset taasen voivat kiinnostaa käyttäjää ja tämän sosiaalista verkostoa.

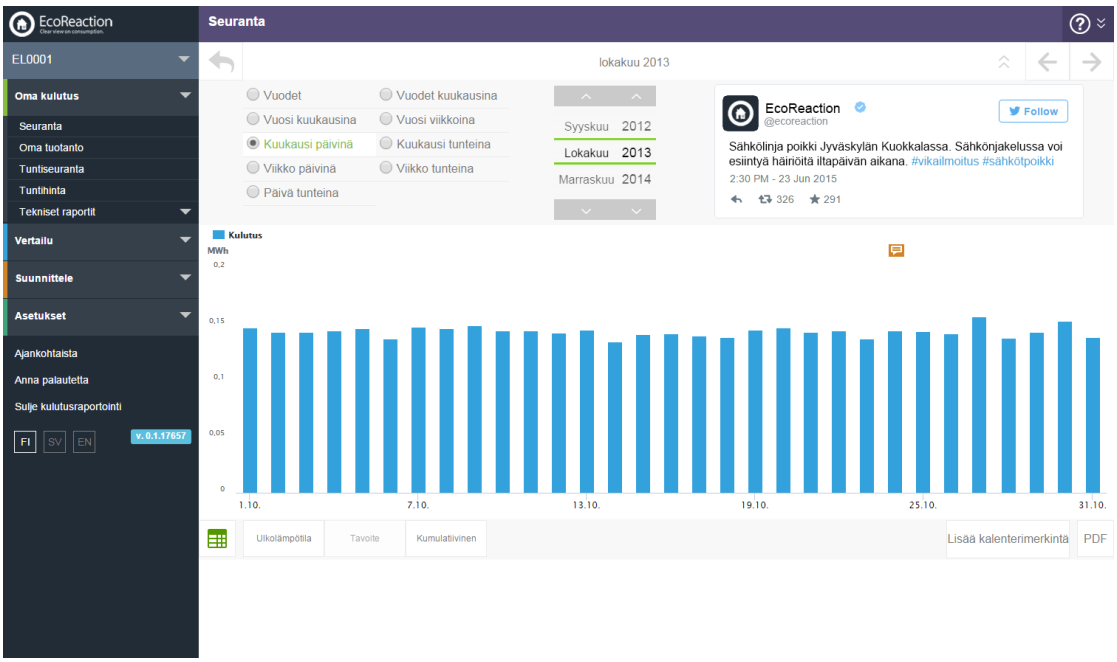
2.3.4 Tiedon jakaminen sosiaalisesta mediasta EcoReactioniin

Monilla verkkosivuilla näkee sivuston Facebook- tai Twitter-syötteen sivun reunassa. Sivuston käyttäjä näkee syötteestä, mitä sivusto on julkaissut sosiaalisessa mediassa. Yleensä tätä käytetään vain mainostamaan sivuston sosiaalisen median näkyvyyttä.

Sosiaalisen median syötettä voidaan kuitenkin käyttää enemmänkin hyödyksi, esimerkiksi tärkeiden tiedotusten levittämiseen. Esimerkiksi sähkökatkoksen aikana sähköyhtiö voisi julkaista tiedotuksen sosiaalisessa mediassa ja tämä julkaisu näkyisi myös EcoReactionissa (ks. Kuvio 4 ja Kuvio 5).

EcoReactionin käyttäjän ei tarvitse olla kirjautunut mihinkään sosiaaliseen mediaan nähdäkseen sosiaalisen median syötteitä sovelluksessa.

Kuvio 4. Esimerkki vikailmoituksesta



Kuvio 5. Esimerkki vikailmoituksesta sovellukseen upotettuna

EcoReactionissa voisi hakea asiakkaan Facebookista tai Twitteristä viimeisimmän julkaisun ja näyttää se sovelluksessa. Vaihtoehtoisesti asiakas voisi määrittää näytille haluamansa julkaisun esimerkiksi Admin-käyttöliittymässä.

2.3.5 Sosiaalisen median integroinnin tuomat riskit

Sosiaalisen median integrointi EcoReactioniin tuo mukanaan monenlaisia riskejä. Integrointi lisää yhden ylläpidettävän ominaisuuden, jossa on omanlaisensa turvallisuusuhkansa ja haasteensa.

Käyttäjän autentikointi sosiaalisen median tunnuksilla altistaa käyttäjän tiedot ulkopuolisille hyökkäyksille. Jos käyttäjän sosiaalisen median tunnukset päätyvät hyökkääjän käsiin, hyökkääjä saa mahdollisesti pääsyn myös EcoReactioniin.

Käyttäjät voivat pilata sähköyhtiön maineen tekemällä sovelluksesta törkyisiä julkaisuja sosiaaliseen mediaan.

Käyttäjä voi vahingossa julkaista arkaluontoista tietoa sosiaaliseen mediaan omasta käyttäytymisestään. Jos käyttäjä esimerkiksi julkaisee päivittäin oman energiankulutuksensa tuntitasolla kaikkien nähtäville, roistot saavat erinomaista dataa käyttäjän päivärytmistä ja tietoa siitä, milloin käyttäjä ei ole kotona.

Epätavallisen suuret energiankulutukset saattavat kiinnittää viranomaistahojen kiinnostuksen. Esimerkiksi marihuanan kasvatus kuluttaa huomattavia määriä sähköä: yhden kilogramman kasvatus kuluttaa 5000 kWh (Hughes 2014).

Sosiaalisen median "pakottaminen" käyttäjän nähtäville tiedotteiden muodossa voi aiheuttaa hämmennystä. Erityisesti riskinä ovat sellaiset käyttäjät, jotka eivät ole diginatiiveja, eivätkä käytä minkäänlaista sosiaalista mediaa. Tällaisilla käyttäjillä voi olla hyvin suuret ennakkoluulot ja pelot tietojensa leviämisestä. He eivät välttämättä ymmärrä, että tiedotteet ovat energiayhtiön julkaisemia, vaan saattavat luulla olevansa itse vastuussa kyseisistä julkaisuista. Olisikin hyvä, että sosiaalisen median syötteet saisi piilotettua sovelluksesta halutessaan.

2.4 Spring Social -projekti

2.4.1 Esittely

Spring Social on Spring-sovelluskehyksen sosiaalisen median projekti. Projektin pääosat ovat Core, Facebook-, Twitter- ja LinkedIn-moduulit. Core sisältää sovelluskehyksen, jolla yhdistetään eri sosiaalisiin median palveluihin ja OAuth-autentikointituen. Facebook-, Twitter- ja LinkedIn-moduulit sisältävät

nimensä mukaisesti tuen kyseisille sosiaalisen median palveluille ja niiden rajapintoihin. Virallisten sosiaalisen median palveluiden moduulien lisäksi Spring-sovelluskehyksen yhteisö on toteuttanut lukuisia moduuleita eri sosiaalisen median palveluihin, kuten Dropboxiin, SoundCloudiin ja Instagramiin. (Walls, Donald & Clarkson 2015b.)

Muita Spring Social -projektin sisältämiä moduuleita on Spring Social Config, joka sisältää Java- ja XML-asetukset Spring Social -projektille, Spring Social Security, joka integroi Spring Securityn Spring Social -projektiin, sekä Spring Social Web, joka hallitsee yhteyksiä verkkosovellusympäristössä. (Mt.)

2.4.2 Spring Social Core

Spring Social Core sisältää Connect-sovelluskehyksen, jolla hallitaan yhdistämistä ja yhteyksiä sosiaalisen median palveluihin.

Lähettämällä POST-pyynnön osoitteeseen /auth/{palveluntarjoaja} saadaan aloitettua kirjautuminen sosiaalisen median palveluun. Palveluntarjoaja on esimerkiksi "facebook" tai "twitter".

Kirjautuminen sosiaalisen median palveluun käynnistää OAuth-tanssin. Spring Social Web -moduuli sisältää ConnectControllerin, joka huolehtii yhdistämisestä palveluntarjoajiin. Se ohjaa käyttäjän palveluntarjoajan sivulle sisäänkirjautumista varten ja palauttaa takaisin sovellukseen autentikoinnin jälkeen. OAuth-tanssin tuloksena sovellus saa palveluntarjoajalta accessTokenin, jolla pystytään tekemään kyselyjä palveluntarjoajan rajapintaan. (Walls, Donald & Clarkson 2015b.)

Spring Social tarjoaa kaksi vaihtoehtoa kirjautumisen toteuttamiseen: SocialAuthenticationFilter tai ProviderSignInController.

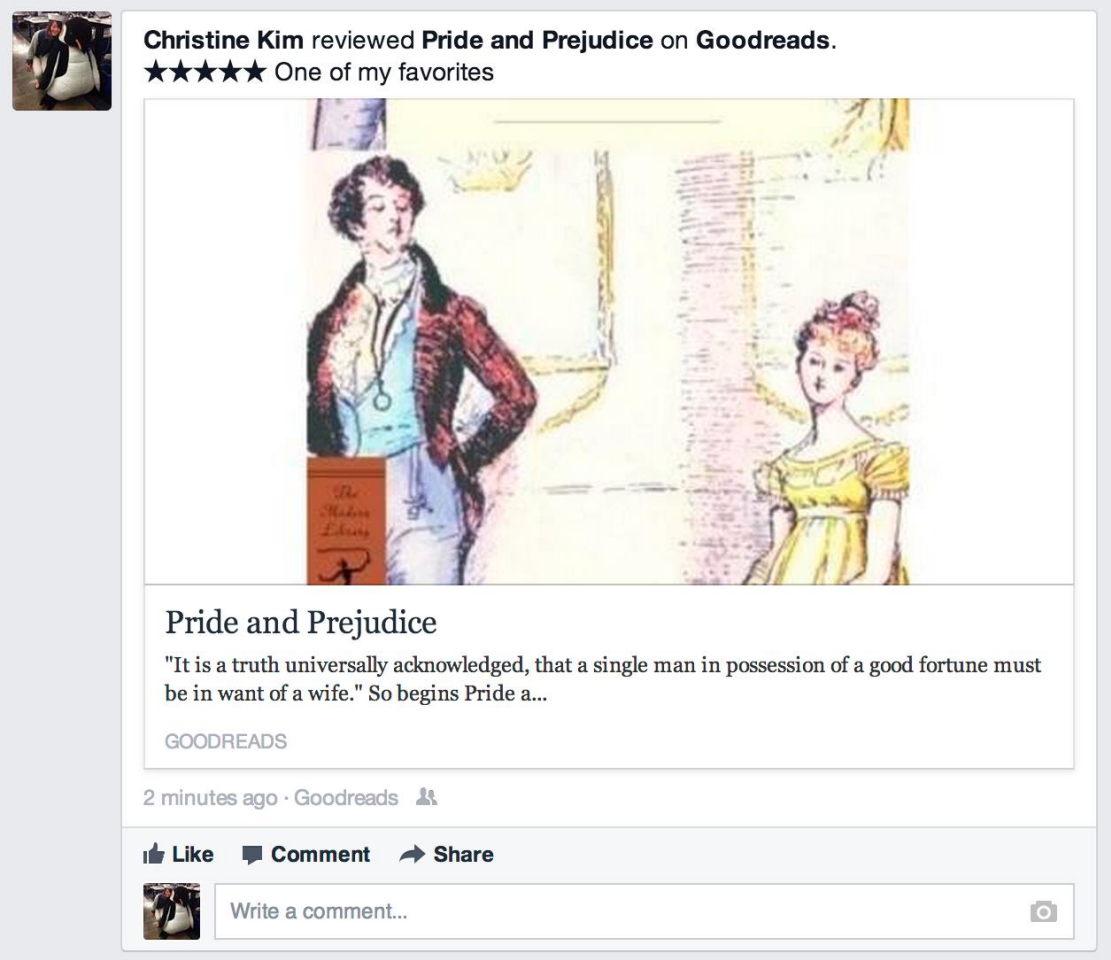
SocialAuthenticationFilteriä suositellaan käytettäväksi, jos sovellus käyttää Spring Securityä, sillä se on Spring Securityn suodatin. SocialAuthenticationFilter tarjoaa lujemman ja luonnollisemman integraation Spring Securityyn. ProviderSignInController toimii minkä tahansa turvallisuusmekanismin kanssa, mukaan lukien Spring Security. (Mt.)

Yhteyden luomisen jälkeen yhteys voidaan säilyttää käyttämällä ConnectionRepository-rajapintaa yhden käyttäjän sovelluksissa tai UsersConnectionRepository-rajapintaa useamman käyttäjän sovelluksissa. Yhteyden voi tallentaa tietokantaan JdbcUsersConnectionRepository-luokan kautta, joka implementoi UsersConnectionRepository-rajapintaa. Tämän tarjoama tietokantataulu on kuvattu liitteessä 1. (Mt.)

2.4.3 Spring Social Facebook

Spring Social Facebook tarjoaa integroinnin Facebookiin ja sen rajapintoihin. Spring Social Facebook piilottaa alleen Facebookin sovelluskehyksen ja tarjoaa omat metodit Facebook-operaatioille. Facebook-integrointi tapahtuu FacebookConnectionFactoryn kautta. (Walls, Donald & Clarkson 2015a.)

Facebook tarjoaa kaksi rajapintaa: Graph API ja Open Graph API. Graph API:lla suoritetaan perustoimintoja, kuten käyttäjän tietojen lukeminen ja aikajanalle julkaiseminen. Open Graph API:lla voidaan luoda tarinoita käyttäjän toiminnasta (ks. Kuvio 6). (Saxena 2013.)



Kuvio 6. Tarina käyttäjän aikajanalla (Submission Process n.d.)

FacebookConnectionFactory konfiguroidaan seuraavanlaisesti:

```
<facebook:config app-id="${facebook.app.id}"
      app-secret="${facebook.app.secret}" />
```

Kyseiset app-id ja app-secret saadaan luomalla sovellus Facebookiin. Nämä ovat yksilölliset joka sovellukselle.

Facebookiin yhdistäessä pitää pyytää käyttäjältä lupa kaikkiin tarvittaviin toimintoihin, kuten julkaisujen tekemiselle aikajanalle. Oletuksena sovellus saa vain pääsyn käyttäjän julkisiin tietoihin. Sovelluksen tulisi pyytää oikeudet vain

niihin toimintoihin, joita se tarvitsee. Näin käyttäjä ei säikähdä, mitä kaikkea tietoja ja toimintoja sovellus saakaan itsestään. On myös suositeltavaa, että oikeuksia pyydetään sitä mukaan, kun niitä tarvitaan. Esimerkiksi lupa julkaista käyttäjän aikajanalle kannattaa pyytää vasta siinä vaiheessa, kun käyttäjä on tekemässä julkaisua sovelluksesta. Joidenkin oikeuksien pyytäminen johtaa siihen, että Facebookin täytyy katselmoida sovellus ennen kuin sen voi julkaista. (Permissions n.d.)

Spring Social Facebook käyttää OAuth2-protokollaa autentikoidessaan käyttäjän. Autentikoinnin jälkeen Spring Social Facebookilla pystytään tekemään kyselyjä Facebookin rajapintaan luomalla FacebookTemplate-objekti, joka ottaa vastaansa vähintään accessTokenin, mutta sille kannattaa myös antaa sovelluksen namespace ja app-id. Namespace ja app-id tarvitaan joidenkin operaatioiden suorittamiseen, kuten Open Graph API:n käyttöön. Yksinkertainen aikajanalle julkaiseminen tapahtuu seuraavanlaisesti:

```
private final Facebook FACEBOOK = new FacebookTemplate(accessToken,
namespace, appid);

FACEBOOK.feedOperations().updateStatus("Status update!");
```

Spring Social Facebookin tarjoama rajapinta Facebookin API:in on jaettu kolmeentoista alirajapintaan. Näistä oleellisimmat on FeedOperations, jolla luetaan ja julkaistaan käyttäjän aikajanalle, UserOperations, jolla haetaan tietoja käyttäjästä ja OpenGraphOperations, jolla hallitaan Facebookin Open Graph API:a tarinoiden tekemiseen. (Walls, Donald & Clarkson 2015a.)

Spring Social Facebookilla voi hakea julkisten sivujen julkaisuja sivun tunnuksella. Sivun tunnus löytyy Facebookista sivun Tietoja-lehden alta. Sivun julkaisujen haku tapahtuu seuraavalla tavalla:

```
PagedList<Post> posts = FACEBOOK.feedOperations().getPosts("sivun
tunnus");
```

Julkisten sivujen julkaisujen haku onnistuu myös ilman sisäänkirjautumista. Tällöin FacebookTemplate-objektille annettava accessToken muodostetaan sovelluksen app-id:stä ja app-secretistä laittamalla |-merkki näiden väliin.

```
String accessToken = appId + "|" + appSecret;
private final Facebook FACEBOOK = new FacebookTemplate(accessToken);
```

2.4.4 Facebook-tarina

Tarina eroaa normaalista julkaisusta siinä, että sille pystytään määrittelemään toiminto ja kohde. Tämän lisäksi tarinassa tulee ilmi, mistä sovelluksesta se on jaettu. Kuviossa 6 toiminto on "reviewed", kohde "Pride and Prejudice" ja sovellus, mistä tarina on jaettu "Goodreads". Tarina sisältää kohteen linkin, otsikon ja kuvauksen. Otsikon ja kuvauksen tilalle voidaan määritellä myös jotain muita tietoja.

Tarina voidaan luoda ohjelmallisesti tai antamalla URL-osoite HTML-sivulle, jonka head-elementtiin on määritelty tarinan vaatimat metatiedot. Kummassakin tapauksessa tarina tarvitsee URL-osoitteen, josta tarinan kohde löytyy. Tarinan julkaisun sisältämää viestiä ei saa täyttää ennalta, vaan käyttäjän on annettava kirjoittaa viesti itse. Tarina voidaan kuitenkin julkaista myös ilman viestiä.

Facebook tukee i18n-internalisaatiota Open Graph -tarinoissa. Tuetut kielet tulee luetella HTML-sivun metatiedoissa. Tarinoiden käännöksiä eri kielille voi tehdä Facebookin hallintapaneelissa.

Jotta sovellus voi julkaista tarinoita käyttäjän aikajanalle, täytyy Facebookin hallintapaneelista laittaa Explicitly Shared päälle. Tarinan metatietoihin täytyy

myös asettaa fb:explicitly_shared arvoon true. Jos Explicilty Shared ei ole päällä eikä asetettu arvoon true, niin tarina tulee luoduksi, mutta sitä ei julkaista käyttäjän aikajanalle, vaan se ilmestyy käyttäjän toimintalokiin. Sovelluksen täytyy myös pyytää käyttäjältä publish_actions-lupa, joka tarvitaan kaikenlaisten julkaisujen tekemiseen.

Spring Social Facebookilla voidaan julkaista tarina seuraavalla tavalla:

```
FACEBOOK.openGraphOperations().publishAction(action, type, link);
```

2.4.5 Facebookin suorittama arviointi sovelluksesta

Facebook vaatii sovelluksen arvioinnin siinä tapauksessa, jos se käyttää mitä tahansa muita oikeuksia kuin public_profile, user_friends tai email. Arvioinnilla Facebook varmistaa, että oikeuksia käytetään oikealla tavalla ja niillä parannetaan sovelluksen käyttäjäkokemusta. Facebook myös varmistaa, että sovellus toimii eri laitteilla moitteettomasti. Facebook arvioi ne alustat, joille sovellus on rekisteröity. EcoReactionin tapauksessa sovellus rekisteröitäisiin verkkosovellukseksi. Muita alustavaihtoehtoja on muun muassa iOS, Facebook canvas ja PlayStation.

Facebookilla on aihealueittain useampia arviointeja. EcoReactionin tapauksessa arvioinnit olisivat Login Review ja Open Graph Review eli sisäänkirjautumisen ja tarinoiden arviointi. Myös Feature Review, ominaisuuksien arviointi, saattaa tulla EcoReactionissa vastaan esimerkiksi, jos tarinaan lisätään mahdollisuus merkitä ystäviä.

Arviointia varten sovelluksella täytyy olla ikoni, pitkä kuvaus ja linkki yksityisyyskäytäntöihin. Käytännössä arviointi tapahtuu niin, että Facebookin hallintapaneelissa sovellus jätetään arvioitavaksi ja annetaan tarvittavat tiedot arvioinnin tekemiseen. Näitä tietoja on muun muassa, mitä oikeuksia sovellus

pyytää käyttäjältään, ohjeet sovelluksen käyttöön ja kuvankaappaukset sovelluksesta ja tarinasta. Jokaisesta pyydetystä oikeudesta tulee olla kuvaus, kuinka sitä hyödynnetään sovelluksessa.

Arviointi vie 3 - 7 työpäivää, jonka jälkeen saa tulokset arvioinnista. Facebook ilmoittaa selkeästi, jos sovelluksessa on jotain korjattavaa. Korjauksien jälkeen voi pyytää uuden arvioinnin.

Jo kertaalleen arvioinnissa hyväksyttyjä oikeuksia ei tarvitse arvioida uudestaan, vaikka sovellusta muutettaisiin. Jos sovellukseen tulee uusia ominaisuuksia, jotka vaativat uusia oikeuksia, niin silloin täytyy pyytää uusi arviointi. Tarinoiden tapauksessa uusi arviointi vaaditaan joka kerta, kun tarinaa muokataan tai luodaan uusi tarina.

2.4.6 Spring Social Twitter

Spring Social Twitter tarjoaa integroinnin Twitteriin ja sen rajapintaan. Spring Social Twitter piilottaa alle Twitterin sovelluskehyksen ja tarjoaa omat metodit Twitter-operaatioille. Twitter-integrointi tapahtuu TwitterConnectionFactoryn kautta. (Walls, Donald & Clarkson 2015c.)

Twitter Connection Factory konfiguroidaan seuraavanlaisesti:

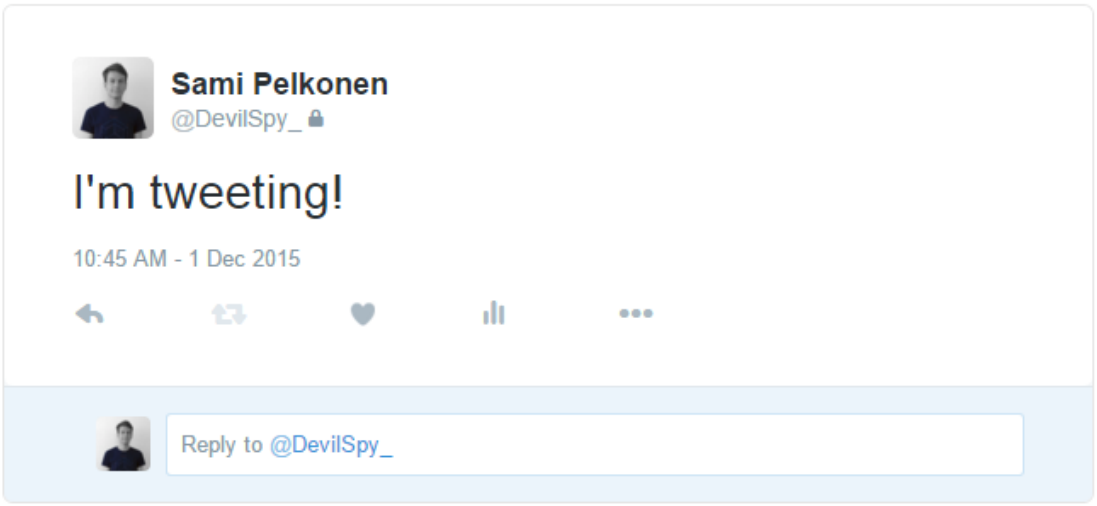
```
<twitter:config app-id="${twitter.consumerKey}"
       app-secret="${twitter.consumerSecret}" />
```

Kyseiset app-id ja app-secret saadaan luomalla sovellus Twitteriin.

Twitterin käyttö on paljon suoraviivaisempaa kuin Facebookin. Pyydettäviä oikeuksia on vain kolme kappaletta: luku, kirjoitus ja viestit. Oikeuksien asettaminen tapahtuu Twitterin hallintapaneelista.

Spring Social Twitter käyttää OAuth1-protokollaa autentikoidessaan käyttäjän. Autentikoinnin tuloksena sovellus saa accessTokenin lisäksi accessTokenSecretin. Autentikoinnin jälkeen Spring Social Twitterillä pystytään tekemään kyselyjä Twitterin rajapintaan luomalla TwitterTemplate-objekti, joka ottaa vastaansa kaikkia toimintoja tehdäkseen sovelluksen consumerKey- ja consumerSecret-arvot, sekä accessTokenin ja accessTokenSecretin. Tweetin (ks. Kuvio 7) tekeminen TwitterTemplatella tapahtuu seuraavanlaisesti:

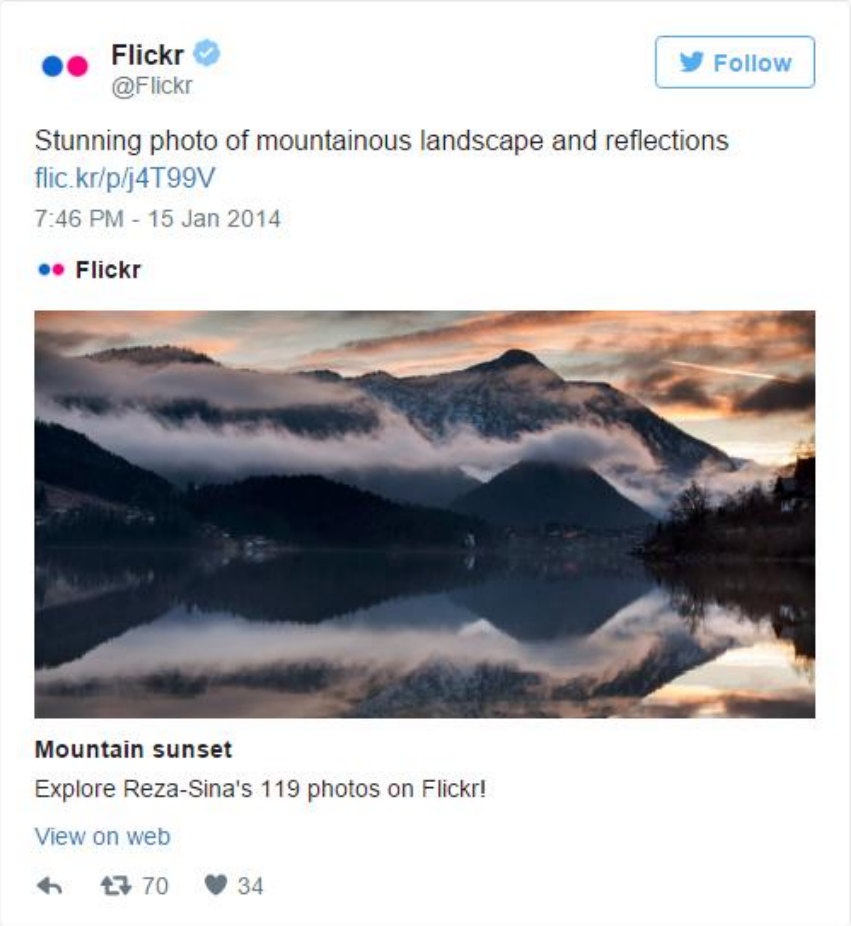
```
private final Twitter TWITTER = new TwitterTemplate(accessToken,
accessTokenSecret, consumerKey, consumerSecret);

TWITTER.timelineOperations().updateStatus("I'm tweeting!");
```



Kuvio 7. Normaali tweet

Spring Social Twitterin tarjoama rajapinta Twitterin API:in on jaettu yhdeksään alirajapintaan. Näistä oleellisimmat on TimelineOperations, jolla luetaan ja julkaistaan tweettejä ja UserOperations, jolla haetaan tietoja käyttäjästä. (Walls, Donald & Clarkson 2015c.)

Twitter tarjoaa paremman käyttäjäkokemuksen luomiseksi "kortit" (ks. Kuvio 8), jotka toimivat hiukan samaan tapaan kuin Facebookin tarinat. Kortin avulla kehittäjä saa paremman hallinnan siitä, minkälainen tweetti luodaan.



Kuvio 8. Twitter kortti (Photo Card n.d.)

Kortti luodaan verkkosivusta sille asetettujen metatietojen perusteella. Twitterin ryömijä osaa lukea Facebookin käyttämiä metatietoja. Tämä vähentää tarvetta toistaa samoja tietoja tarinan ja kortin kanssa. Julkaistakseen kortin käyttäjän täytyy vain tweetata kortin osoite.

Erilaisia korttityyppejä on neljä: Summary Card, Summary Card with Large Image, App Card ja Player Card. Summary Card ja Summary Card with Large Image luovat verkkosivusta kortin, jossa on kuva ja tiivistelmä sivun tekstistä. Summary Cardia voidaan käyttää artikkeleiden tweettaamiseen, kun taas Summary with Large Image soveltuu tämän lisäksi myös näyttävien kuvien tweettaamiseen. App Card soveltuu mobiilisovellusten tweettaamiseen ja Player Card videoiden, äänen ja liikkuvien kuvien tweettaamiseen.

Tweetin lukeminen Spring Social Twitterillä vaatii vain sovelluksen consumer secretin ja keyn TwitterTemplatea luodessa.

```
private final Twitter TWITTER = new TwitterTemplate(CONSUMERKEY,
CONSUMERSECRET);
```

Tweettejä voidaan hakea Twitter-käyttäjän käyttäjänimellä ilman @-merkkiä. GetUserTimeline-metodi hakee 20 viimeisintä tweettiä annetun käyttäjän aikajanalta. Palautuvissa Twitter-objekteissa on muun muassa tweetin id, sisältö, tweettaajan käyttäjänimi ja linkki tämän profiilikuvaa.

```
List<Tweet> tweets =
TWITTER.timelineOperations().getUserTimeline("username");
```

Twitter tarjoaa oEmbed-rajapinnan, jolla pystytään hakemaan verkkosivulle upotettavia tweettejä. Rajapinnasta saa objektin, joka sisältää valmiin HTML-koodin kyseisen tweetin upottamiseen. Rajapinta tukee tweettien hakua id:n tai URL:n perusteella, mutta Spring Social Twitterissä on toteutus vain haulle id:n perusteella. Upotettavan tweetin haku tapahtuu seuraavanlaisesti:

```
TWITTER.timelineOperations().getStatusOEmbed(tweetId);
```

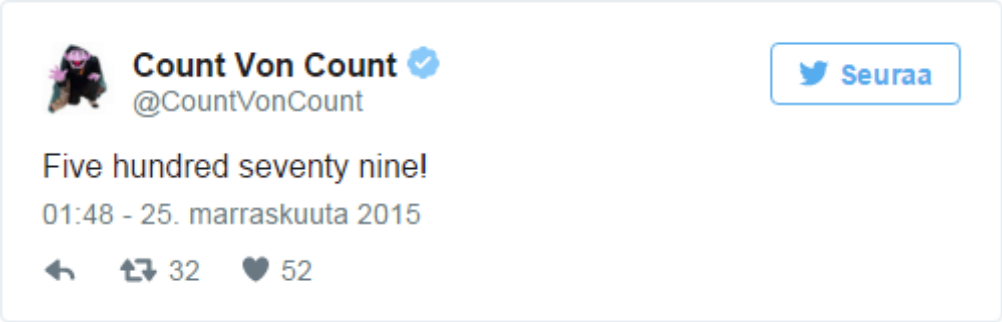
GetStatusOEmbed-metodille voi antaa toisena parametrina OEmbedOptions-objektin, jossa voi määritellä upotettavan tweetin asetuksia. Määriteltäviä asetuksia on muun muassa näytetäänkö tweetin sisältämä media,

maksimileveys, kieli ja näytetäänkö alkuperäinen tweet, jos upotettava tweet on vastaus.

Palautusarvona saadaan seuraavanlainen objekti:


```
{
        extraData: {}
        type: "rich"
        version: "1.0"
        authorName: "Count Von Count"
        authorUrl: "https://twitter.com/CountVonCount"
        providerName: "Twitter"
        providerUrl: "https://twitter.com"
        cacheAge: 3153600000
        height: null
        width: 550
        html: "<blockquote class="twitter-tweet"><p lang="en"
        dir="ltr">Five hundred seventy nine!</p>&mdash; Count Von Count
        (@CountVonCount) <a
        href="https://twitter.com/CountVonCount/status/6693015931127562
        24">November 24, 2015</a></blockquote> <script async
        src="//platform.twitter.com/widgets.js" charset="utf-
        8"></script>"
        url:
        "https://twitter.com/CountVonCount/statuses/669301593112756224"
}
```


Upotettuna tweet näyttää samalta kuin mikä tahansa tweet (ks. Kuvio 9).



Kuvio 9. Upotettu tweet (Count Von Count 2015)

2.4.7 Yhteisön toteuttamat moduulit

Spring Social on toteutettu niin, että siihen on helppo kehittää omia moduuleita sellaisiin sosiaalisen median palveluihin, joihin ei löydy tukea vielä Spring Social -projektin puolesta. Spring Social -projektin dokumentaatiossa on kuvattu askeleet ja suuntaviivat johdonmukaisen moduulin toteuttamiseen.

Yhteisö on toteuttanut moduuleita muun muassa sellaisiin palveluihin kuin Tumblr, Yammer, Foursquare ja Evernote. Yhteisön toteuttamat moduulit on listattu Spring Social -projektin verkkosivuilla ja ladattavissa GitHubista.

2.5 Spring Social -projektin käyttö EcoReactionissa

2.5.1 Yhdistäminen

EcoReactionissa kirjautuminen sosiaalisen median palveluun voidaan käynnistää lisäämällä käyttöliittymään sosiaalisen median nappi, joka lähettää kutsun palvelinpuolen osoitteeseen /auth/{palveluntarjoaja}.

Sosiaaliseen mediaan kirjautumisen jälkeen käyttäjä palautuu palvelinpuolen juuriosoitteeseen. Ohjaus takaisin EcoReactioniin voidaan toteuttaa esimerkiksi luomalla index.jsp-tiedosto, jossa on ohjaus EcoReactioniin:

```
<c:redirect url=”https://www.ecoreaction.com” />
```

Facebook- ja TwitterConnectionFactoreiden peruskonfiguraatio ei ollut riittävä, sillä ongelmia ilmeni, kuinka määritellä riittävät Facebook-oikeudet sovellukselle.

JSP-sovellusta tehdessä oikeudet voitaisiin pyytää käyttäjän yhdistäessä Facebookiin määrittelemällä lomakkeen piilokentässä tarvittavat oikeudet.

Tarkoituksena oli kuitenkin toteuttaa integraatio palvelinpäässä, joten oikeudet pitää määritellä konfiguraatiossa. Alla on esimerkki konfiguraatiosta, jolla pystytään määrittelemään oikeudet. Mukana on myös Twitter-konfiguraatio.

```
<bean id="connectionFactoryLocator"
class="org.springframework.social.security.SocialAuthenticationService
Registry">
       <property name="authenticationServices">
         <list>
          <bean
       class="org.springframework.social.facebook.security.FacebookAut
       henticationService">
             <constructor-arg value="${facebook.app.id}" />
             <constructor-arg value="${facebook.app.secret}" />
             <property name="defaultScope" value="publish_actions" />
           </bean>
           <bean
       class="org.springframework.social.twitter.security.TwitterAuthe
       nticationService">
             <constructor-arg value="${twitter.consumerKey}" />
             <constructor-arg value="${twitter.consumerSecret}" />
           </bean>
         </list>
       </property>
</bean>
```

Kyseinen konfiguraatiossa oleellinen osa on defaultScope, jolla määritellään tarvittavat oikeudet. Tässä tapauksessa pyydetään lupa julkaista käyttäjän aikajanalle publish_actions-oikeudella.

Twitterin oikeudet määritellään Twitterin hallintapaneelista. EcoReaction tarvitsee luku- ja kirjoitusoikeudet eli luvan julkaista käyttäjän Twitter-syötteeseen ja lukea käyttäjän tietoja ja tweettejä.

2.5.2 Julkaisujen tekeminen EcoReactionista sosiaaliseen mediaan

Facebookilla on joitain ennalta määrättyjä tarinoita, kuten kirjan lukemiseen liittyviä tarinoita, esimerkiksi "Read a Book". Open Graph API:lla voidaan

luoda sovelluskohtaisia tarinoita. EcoReactionissa tarinan toiminto voisi olla "use" tai "consume" ja kohde "electricity" tai "energy". Tarinasta luotaisiin verkkosivu, josta löytyy kuvakaappaus energiankulutuksesta, tiedot kulutuksesta ja linkki energianseurantapalvelun tarjoajan sivuille (ks. Kuvio 10).

Kuvio 10. Tarinan verkkosivu

Verkkosivun head-elementin metatietoihin tulee määritellä seuraavanlaiset ominaisuudet:

```
<meta property="fb:app_id" content="1234567890"/>
<meta property="fb:explicitly_shared" content="true"/>
```

```
<meta property="og:locale" content="fi_FI"/>
<meta property="og:locale:alternate" content="en_US"/>
<meta property="og:locale:alternate" content="sv_SE"/>
<meta property="og:url"
content="http://ecoreaction.com/consumption.html"/>
<meta property="og:type" content="ecoreaction:electricity"/>
<meta property="og:title" content="Energiankulutus"/>
<meta property="og:image"
content="http://ecoreaction.com/consumption.png"/>
<meta property="og:image:width" content="1200"/>
<meta property="og:image:height" content="630"/>
<meta property="og:description" content="EcoReactionilla voit seurata
energiankulutustasi."/>
<meta property="ecoreaction:ea_kwh" content="2,80 kWh"/>
<meta property="ecoreaction:ea_time" content="20-21"/>
<meta property="ecoreaction:ea_comparison" content="-60%"/>
<meta property="ecoreaction:ea_date" content="2015-03-
28T20:00:00+02:00"/>
```

Näiden tietojen avulla Facebook osaa luoda oikeanlaisen tarinan. Kyseisissä tiedoissa ea_-alkuiset tiedot on itse määriteltyjä tarinan ominaisuuksia. Pakollisia arvoja näistä on ainoastaan type ja title, mutta kaikki mainitut arvot on hyvä olla, jotta tarinan luominen onnistuisi parhaalla mahdollisella tavalla.

Tarinaa luodessa Facebook "ryömii" annetun verkkosivun sisällön läpi. Nämä metatiedot sisältävät ryömijälle tärkeää tietoa muun muassa tarinaan liitettävästä kuvasta, otsikosta ja kuvauksesta. Ilman metatietoja Facebook joutuu arvailemaan, mitkä verkkosivulta löytyvät tiedot ovat oleellisia tarinan koostamisen kannalta. Ilman metatietoja tarinasta voi tulla täysin erilainen kuin mitä kehittäjä on suunnitellut.

Twitteriin julkaistaessa tweetattaisiin kortti (ks. Kuvio 11), joka muodostetaan samasta verkkosivusta kuin Facebook-tarina. Korttia varten verkkosivun metatietoihin tulee lisätä seuraavanlaiset tiedot:

```
<meta name="twitter:card" content="summary_large_image" />
<meta name="twitter:site" content="@ecoreaction" />
```

Card kertoo, minkälainen kortti on kyseessä, tässä tapauksessa summary_large_image, joka muodostaa kortin, jossa on iso kuva otsikon ja kuvauksen kera. Otsikko ja kuvaus saadaan Facebook-tarinaa varten määritellyistä og:title ja og:description -metatiedoista. Site kertoo, kenen Twitter-käyttäjän tekemä kyseinen kortti on.

Kuvio 11. Twitter kortti energiankulutuksesta

2.5.3 Julkaisujen upottaminen EcoReactioniin

Facebook tarjoaa JavaScript SDK:n julkaisujen upottamiseen verkkosivulle. Sivustolle lisätään koodinpätkä, jonka jälkeen SDK on käytössä. JavaScript SDK tarvitsee julkaisun URL-osoitteen upottaakseen sen verkkosivulle.

Testien mukaan FacebookTemplatella haettu Post-objekti ei sisällä arvoa linkille, vaikka sille löytyykin attribuutti objektista. Linkin voi kuitenkin muodostaa palautuvasta julkaisun id:stä. Id on muotoa sivuId_julkaisuId. Tarvittava URL taas on muotoa https://www.facebook.com/sivuId/posts/julkaisuId. Pienellä parsimisella voidaan siis päätellä tarvittava URL.

Facebookin JavaScript SDK:n lisättyä sivustolle, voidaan julkaisu upottaa seuraavanlaisella elementillä:

```
<div class="fb-post" data-href="{julkaisun-url}"></div>
```

Twitter tarjoaa Twitter for Websites -nimisen työkalun tweettien upottamiseen verkkosivuille. Sivustolle lisätään JavaScript-koodinpätkä, jonka jälkeen työkalu on käytössä. OEmbed-rajapinnan palauttama objekti sisältää html-attribuutin, jonka arvo voidaan lisätä verkkosivulle sellaisenaan:

```
<blockquote class="twitter-tweet"><p lang="en" dir="ltr">Five hundred
seventy nine!</p>&mdash; Count Von Count (@CountVonCount) <a
href="https://twitter.com/CountVonCount/status/669301593112756224">Nov
ember 24, 2015</a></blockquote> <script async
src="//platform.twitter.com/widgets.js" charset="utf-8"></script>
```

Julkaisuja ja tweettejä haettaessa ne on hyvä tallettaa palvelimelle välimuistiin, jottei jokaiselle käyttäjälle tarvitse erikseen hakea palveluntarjoajan rajapinnasta samoja julkaisuja tai tweettejä. Twitterillä pyynnöt on rajattu 180 kappaleeseen 15 minuutissa. Facebook ei ilmoita pyyntöjen maksimimäärää, mutta toteaa rajojen olevan olemassa.

3 KÄYTETTYJEN TEKNOLOGIOIDEN UUSIMMAT VERSIOT

3.1 Esittely

Tässä luvussa tutkitaan ja esitellään EcoReactionissa käytettyjen teknologioiden uusimpia versioita: Java 8, Spring 4 ja Hibernate 5. Uusimpien versioiden tuomia hyötyjä ja uusia ominaisuuksia peilataan tuotteessa käytettyihin vastaavuuksiin ja tutkitaan, mitä hyötyä nämä tuovat tuotteelle.

3.2 Java

Java on luokkapainotteinen, olio-ohjelmointiin keskittynyt ohjelmointikieli. Javan kehitti James Gosling, ja sen ensimmäinen versio julkaistiin 1995. Java-ohjelmat pyörivät Java Virtual Machinen päällä. (Java n.d.)

EcoReactionissa käytetään Javaa palvelinpuolen ratkaisuissa, kuten REST-rajapinnan toteutuksessa.

3.3 Java 8:n uudet ominaisuudet

Java 8 julkaistiin maaliskuun 18. päivä 2014 (Java version history n.d.). Merkittävimpiä uudistuksia Java 8:ssa on tuki funktionaaliselle ohjelmoinnille, Lambda expressionit, Stream API, Optional-luokka, sekä uudistettu Date and Time API (What’s New in JDK 8 2014).

3.3.1 Lambda expressionit

Java 8:n suurin muutos on tuki funktionaaliselle ohjelmoinnille ja Lambda expressioneille. Funktionaalinen ohjelmointi eroaa suuresti "normaalista", imperatiivisesta, ohjelmoinnista.

Imperatiivisessa ohjelmoinnissa luodaan sarja käskyjä, joita tietokone toteuttaa. Yleensä käskyissä käsitellään muuttujan arvoa. Tarpeeksi monen käskyn jälkeen saadaan lopputuloksena haluttu arvo. Erilaiset silmukat ovat tärkeässä osassa imperatiivisessa ohjelmoinnissa, esimerkiksi for-silmukka, joka iteroi listan läpi ja muuttaa muuttujien arvoja. Imperatiivisessa ohjelmoinnissa käskyjen suoritusjärjestys on erittäin tärkeää. Jos järjestys muuttuu, muuttuu myös saatu arvo.

```
for(String s : stringsList){
       s = "This string says " + s;
       System.out.println(s);
}
```

Funktionaalinen ohjelmointi perustuu jäsentyneisiin funktiokutsuihin; funktio kutsuu toista funktiota, joka kutsuu taas seuraavaa funktiota. Jokainen funktio saa arvon edelliseltä funktiolta ja palauttaa arvon itseään kutsuvalle funktiolle. Funktionaalisessa ohjelmoinnissa muuttujan arvoa ei muuteta. Funktiot palauttavat jokaisella kutsukerralla saman arvon. Suoritusjärjestyksellä ei ole vaikutusta lopputulokseen. (Ritter 2015.)

Edellinen for-silmukka muuttuisi funktionaalisessa ohjelmoinnissa muotoon

```
stringsList.stream()
  .map(s -> "This string says " + s).forEach(System.out::println);
```

Funktionaalinen ohjelmointi onkin omiaan käsittelemään kokoelmia, kuten taulukoita ja listoja. EcoReactionissa voitaisiin käyttää Lambda-expressioneita

korvaamaan silmukoita. Koko koodikantaa ei kannata lähteä rakentamaan funktionaalisen ohjelmoinnin keinoin, vaan sieltä kannattaa poimia muutama kätevä keino käyttöön. Tämä ei vaadi paljon uuden opettelua, mutta selkeyttää ja vähentää koodia huomattavasti.

Funktionaalisella ohjelmoinnilla saadaan tehokkaasti ja turvallisesti käyttöön usean prosessorin laskentateho. Funktioita voidaan suorittaa turvallisesti yhtä aikaa eri säikeissä, koska suoritusjärjestyksellä ei ole väliä ja jokaisella suorituskerralla funktio palauttaa saman arvon.

Funktionaalisessa ohjelmoinnissa funktio palauttaa aina saman tuloksen, sillä se on riippuvainen vain sille syötetyistä arvoista. Tällaiset "puhtaat funktiot" voidaan suorittaa missä järjestyksessä tahansa. Jos esimerkiksi

```
f(x) = x*3
g(y) = y+1
```

ja suoritetaan funktiota

```
f(g(1))
```

niin sillä ei ole väliä onko suoritusjärjestys

```
f(g(1))
= f(1+1)
= f(2)
= 2*3
= 6
```

vai

```
f(g(1))
= g(1)*3
= (1+1)*3
= 2*3
= 6
```

Ohjelmoijan ei tarvitse välittää suoritusjärjestyksestä, vaan ohjelman kääntäjä valitsee tehokkaimman järjestyksen.

3.3.2 Optional-luokka

Java 8:n merkittävimpiin muutoksiin kuuluu Optional-luokka. Optional on tehty vähentämään NullPointerException-virheitä. NullPointerException on ehkä yleisin sovelluksen toiminnan pysäyttävä virhe Javassa. Tämä virhe syntyy, kun yritetään käyttää objektia, jonka arvo on null (Preventing NullPointerException n.d.).

Optional on luokka, joka kapseloi nimensä mukaisesti valinnaisen arvon sisäänsä. Optional siis sisältää arvon tai ei, mutta se ei kuitenkaan ole null (ellei sen arvoksi nimenomaan aseteta null). Luokkien attribuutteja voidaan kapseloida Optionalilla seuraavaan tapaan.

```
public class MeteringPoint {
	private Optional<Address> address;

	public Optional<Address> getAddress() {
			return address;
	}
```

Koodissa Optionalia voidaan käyttää seuraavasti:

```
Optional<Address> address = meteringPoint.getAddress();

if(address.isPresent()){
	// do something
}
```

Tämä ei kuitenkaan ole Optionalin optimaalisin käyttötapa, sillä sama asia voitaisiin hoitaa seuraavanlaisesti ilman Optionalia.

```
Address address = meteringPoint.getAddress();
```

```
if(address != null) {
	// do something
}
```

Parhaiten Optionalia käytetään Lambda-expressioneiden kanssa. Perinteisesti addressia haettaessa meteringPoint-objektilta joutuisimme tekemään seuraavanlaiset tarkistukset, jos halutaan välttyä NullPointerExceptioneilta.

```
if (meteringPoint != null) {
	address = meteringPoint.getAddress();

	if(address != null) {
			System.out.println(address);
	}
}
```

Lambda expressioneilla kyseinen koodi muuttuisi muotoon

```
Optional.ofNullable(meteringPoint)
	.map(MeteringPoint::getAddress)
	.ifPresent(x -> System.out.println(x.get()));
```

Kyseisessä esimerkissä ofNullable() on tärkeä, sillä se kertoo, että meteringPoint voi olla null. Jos käytetään metodia of(), niin saadaan NullPointerException meteringPointin ollessa null. Map()-metodilla saadaan meteringPoint-objektin address. IfPresent() tarkistaa, onko Optional<Address> address olemassa. Jos se on olemassa, niin tulostetaan ulos address:n arvo get()-metodilla.

Vaikka Optionalia käytettäisiinkin "normaaliin" tapaan, niin sekin edistää koodia ja vähentää NullPointerExceptioneita, koska ohjelmoija joutuu miettimään Optionalin kanssa käyttötapaukset, joissa isPresent() palauttaa arvon false. Normaalien null-arvojen kanssa käsittely saattaa unohtua, jos ohjelmoija ei huomio, että arvo saattaa palautua null-arvona.

Optionalia on myös kritisoitu. Optional kapseloi jonkun toisen luokan sisäänsä, jolloin käsiteltävien luokkien määrä kaksinkertaistuu ja muistin tarve lisääntyy. Optional voi heittää NullPointerExceptionin luokasta, jota se kapseloi. Optional saattaa myös itse olla null, jolloin se voi aiheuttaa NullPointerExceptionin. Objektien kapselointi hankaloittaa debuggausta ja objektien serialisointia. (Johnson 2014.)

3.3.3 Uudistettu Date and Time API

Ennen Java 8:aa yksi suurimmista Javan murheenkryyneistä on ollut puutteellinen Date and Time API. Päivien ja aikojen käsittely on aiheuttanut kehittäjille päänvaivaa: päivien käsittely ei ole ollut säieturvallista, sillä luokat eivät ole muuttumattomia (immutable) eli objektien tilaa on voitu muuttaa niiden luomisen jälkeen. Java.util.Date on suunniteltu sekavasti: vuodet alkavat vuodesta 1900, kuukaudet 0:sta ja päivät 1:stä. Java 8:ssa on korjattu nämä viat ja kehitetty Date and Time API:a erittäin hyvään suuntaan. (Evans & Warburton 2014.)

Uuden java.time-paketin tärkeimmät luokat ovat Instant, LocalDate, LocalTime, LocalDateTime ja ZonedDateTime. Kyseiset luokat on kuvattu alla olevassa taulukossa (ks. Taulukko 1).

Taulukko 1. Java.Time-paketin tärkeimmät luokat

Luokka	Selite	Esimerkki
Instant	Aikaleima	2015-10-24T14:26:51.895Z
LocalDate	Päivämäärä ilman aikaa	2015-10-24
LocalTime	Aika ilman päivämäärää	17:26:51.878

Jatkuu seuraavalle sivulle.

Jatkoa edelliseltä sivulta.

LocalDateTime	Päivä ja aika	2015-10-24T17:26:51.881
ZonedDateTime	Päivä ja aika aikavyöhykkeen ja vaihesiirron kera	2015-10-24T17:26:51.895+03:00[Europe/Helsinki]

Aiemmin päiviä ja aikoja käsiteltäessä tuli luoda kalenteriobjekti, jonka kautta käsiteltiin päiviä ja aikoja. Tämä on hyvin raskasta ja monirivistä ohjelmointia. Pienimmillään on joutunut tekemään seuraavanlaisesti:

```
Calendar calendar = Calendar.getInstance();
calendar.set(2015, 9, 24);
Date date = calendar.getTime();
```

On tärkeää huomata, että kyseisessä esimerkissä kuukausi numero 9 ei suinkaan ole syyskuu, vaan lokakuu. Uudella Date and Time API:lla kyseinen koodi voidaan esittää seuraavanlaisesti:

```
LocalDate date = LocalDate.of(2015, 10, 24);
```

EcoReactionissa käsitellään hyvin paljon päiviä. Yhden luokan tarkoitus on etsiä kuukauden aloitus- ja lopetuspäivä. Luokan päälogiikan pituus on 14 riviä. Tämä logiikka voidaan toteuttaa kuudella rivillä seuraavanlaisesti:

```
int year = parameteredClass.getYear();
int month = parameteredClass.getMonth();
YearMonth yearMonth = YearMonth.of(year, month);
int lengthOfMonth = yearMonth.lengthOfMonth();
LocalDate monthEndDate = LocalDate.of(year, month, lengthOfMonth);
LocalDate monthStartDate = LocalDate.of(year, month, 1);
```

Java 8:n uudessa Date and Time API:ssa on paljon tällaisia hyödyllisiä toiminnollisuuksia. Onko nyt karkausvuosi voidaan tarkistaa isLeapYear()-metodilla, kahden päivämäärän välistä aikaa voidaan laskea Period-luokan between()-metodilla, kahden päivämäärän suhdetta toisiinsa voidaan tarkistaa

isAfter()- ja isBefore()-metodeilla ja päivämääriä voidaan muokata lisäämällä tai vähentämällä aikayksiköitä minus()- tai plus()-metodeilla. (Paul 2014.)

Aikavyöhykkeitä voidaan käsitellä ZoneId- ja ZonedDateTime-luokkien avulla. ZoneId ottaa parametrina aikavyöhykkeen koko nimen, esimerkiksi Europe/Helsinki tai lyhennyksen, esimerkiksi UTC. Seuraava esimerkki muuttaa tämänhetkisen Suomen ajan UTC-ajaksi:

```
LocalDateTime dateTimeInUTC = ZonedDateTime.of(LocalDateTime.now(),
       ZoneId.of(“Europe/Helsinki")).withZoneSameInstant(ZoneId.of(“UT
       C")).toLocalDateTime();
```

3.4 Spring-sovelluskehys

Spring-sovelluskehys on avoimen lähdekoodin sovelluskehys Java-ohjelmille. Sen ensimmäinen versio julkaistiin vuonna 2004. Spring-sovelluskehys on rakennettu modulaarisesti niin, että siitä voi käyttää niitä osia, mitä tarvitsee. Päämoduulit ovat Core Container, Data Access/Integration, Web, AOP, Instrumentation, Messaging ja Test. Spring-sovelluskehys tarjoaa tuen MVC-mallille ja IoC-prosessille, sekä AOP-ohjelmoinnille. (Johnson, Hoeller, Donald, Sampaleanu, Harrop, Risberg, Arendsen, Davison, Kopylenko, Pollack, Templier, Vervaet, Tung, Hale, Colyer, Lewis, Leau, Fisher, Brannen, Laddad, Poutsma, Beams, Abedrabbo, Clement, Syer, Gierke, Stoyanchev, Webb, Winch, Clozel, Nicoll & Deleuz 2015.)

Inversion of Control (IoC) on malli, jossa sovellus ei kutsu sovelluskehyksen metodeja, vaan sovelluskehys kutsuu sovelluksessa määriteltyjä toteutuksia. Spring-sovelluskehys käyttää Dependency Injectioniksi (DI) kutsuttua IoC-toteutusta.

Dependency Injection on prosessi, jossa määritellään olioiden riippuvuudet ja IoC container injektoi nämä riippuvuudet olioon, kun se luodaan. Olio ei hallitse riippuvuuksien luontia, sijaintia tai niiden kutsumista, vaan IoC container kutsuu niitä tarvittaessa. Tämä on käänteinen sille, että olio hallitsisi tämän kaiken itse. Käytännössä JavaBeanille määritellään sen riippuvuudet ja Spring injektoi nämä riippuvuudet JavaBeaniin, kun se luodaan. (Johnson ym. 2015.)

Alla olevassa esimerkissä on määritelty customerService-JavaBeanin riippuvuudet XML-konfiguraatiossa.

```
<bean id="customerService" class="path.to.class.CustomerServiceImpl">
	<property name="customerDAO" ref="customerDAO" />
	<property name="userService" ref="userService" />
</bean>
```

Java-luokassa riippuvuudet saadaan käyttöön seuraavalla tavalla, esimerkissä käytetään vain customerDAO:a selkeyden vuoksi:

```
public class CustomerServiceImpl implements CustomerService {

	private CustomerDAO customerDAO;

	protected CustomerDAO getCustomerDAO() {
			return customerDAO;
	}

	public void setCustomerDAO(CustomerDAO customerDAO) {
			this.customerDAO = customerDAO;
	}


	public void storeCustomer(Customer customer) {
			getCustomerDAO().storeCustomer(customer);
	}
}
```

3.5 Spring 4 uudet ominaisuudet

Tässä luvussa käydään läpi suurimpia muutoksia, joita on tullut Spring-sovelluskehykseen sitten 3.x-version. Vertailukohtana käytetään Spring 4.2.1.RELEASE -versiota, joka julkaistiin 1.9.2015.

Spring 4:n suurimpiin muutoksiin kuuluu tuki Java 8:lle ja sen uusille ominaisuuksille, muun muassa tuki lambda expressioneille ja uudistetulle Date and Time API:lle. Spring 4:stä on karsittu käytöstä poistettuja paketteja ja metodeja. (Johnson ym. 2015.)

3.5.1 @RestController-annotaatio

Aiemmin rajapintoja tehtäessä on pitänyt käyttää @ResponseBody-annotaatiota jokaisen @RequestMapping-annotaatioilla merkityn metodin kanssa. Spring 4:ssä ei tarvitse merkata jokaista metodia erikseen, vaan koko luokka voidaan merkata @RestController-annotaatiolla, joka yhdistää @ResponseBody- ja @Controller-annotaatiot. (Johnson ym. 2015.)

Esimerkki vanhasta tavasta:

```
@Controller
public class UserController {
	@RequestMapping("/users/{userId}")
	public @ResponseBody User getUserById(
			@PathVariable(value = "userId") final int userIdInt,
			@RequestParam(value = "date") final String
			dateString) throws Exception{
```

Esimerkki uudesta tavasta:

```
@RestController
public Class UserController {
	@RequestMapping("/users/{userId}")
	public User getUserById(
			@PathVariable(value = "userId") final int userIdInt,
```

```
@RequestParam(value = "date") final String
dateString) throws Exception{
```

Tämän käyttö poistaa virhetilanteet, joissa @ResponseBody-annotaatio on unohtunut merkitä rajapintaan. Sen unohtuessa ei välttämättä ole tullut kovin selkeää virheilmoitusta, jolloin on saattanut mennä hyvä tovi selvittää, mistä ongelma johtuu. @RestController-annotaation käyttö myös vähentää ja selkeyttää koodia.

3.5.2 Optionalin käyttö REST-kutsuissa

Java 8:n uutta Optional-luokkaa voidaan käyttää @RequestParam- ja @RequestHeader-annotaatioiden kanssa (Johnson ym. 2015). EcoReactionissa merkataan kukin @RequestParam vaadituksi tai ei vaadituksi. Tällöin jo REST-kutsua tehdessä asiakasohjelma antaa virheilmoituksen vaaditusta parametrista, jos se puuttuu.

Sellaiset parametrit voisi kapseloida Optional-luokkaan, jotka eivät ole vaadittuja. Tällöin ohjelmoija pakotetaan ottamaan huomioon tapaukset, joissa näitä parametreja ei ole annettu. Näin saavutetaan Optional-luokan tuomat hyödyt eli vältytään NullPointerException-virheiltä. Vaadittujen parametrien tapauksessa Optional-luokkaa ei voida edes käyttää, sillä Optional käytännössä merkkaa parametrin valinnaiseksi.

Alla oleva esimerkki palauttaa annetun päivän viikonpäivän. Päivä annetaan muodossa yyyy-MM-dd.

```
@RequestMapping("/daySolver")
public String daySolver(
       @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
       @RequestParam(value = "date", required = false)
                              Optional<LocalDate> dateParam) {

       StringBuilder builder = new StringBuilder();
```

```
        if(dateParam.isPresent()) {
                builder.append("The given day is ");
        } else {
                builder.append("No date given");
        }

        dateParam.ifPresent(x ->
                builder.append(x.getDayOfWeek().toString()));

        return builder.toString();
}
```

3.5.3 @JsonView-annotaatio

Spring 4 tukee Jacksonin @JsonView-annotaatiota, jolla voidaan toteuttaa eritasoisia JSON-tulosteita. Jackson on kirjasto, jonka avulla muodostetaan JSON- tai XML-tulosteita. (Deleuze 2014.)

@JsonView-annotaatiolla voisi EcoReactionissa tehdä esimerkiksi käyttöpaikkoja haettaessa rajapinnat erikseen tiivistelmälle ja kaikkien tietojen palautukselle. Tällä saavutettaisiin kevyemmät ja selkeämmät tulosteet palautuksille. Tarkennettua tulostetta olisi helpompi tulkita, veisi vähemmän kaistaa ja sen parsiminen olisi nopeampaa.

Rajapinta tiivistelmälle:

```
@JsonView(View.Summary.class)
@RequestMapping("/meteringPoints/summary")
public EntityQueryResults<MeteringPoint> getMeteringPointsSummary(
```

Tuloste:

```
{
        meteringPointNumber: "meteringPoint1",
        meteringPointType: "ELECTRICITY"
},
{
        meteringPointNumber: "meteringPoint2",
        meteringPointType: "ELECTRICITY"
}
```

Rajapinta kaikelle:

```
@RequestMapping("/meteringPoints")
public EntityQueryResults<MeteringPoint> getMeteringPoints(
```

Tuloste:

```
{
	created: "2014-01-01T22:00:00.000+0000",
	modified: "2014-01-02T10:15:20.000+0000",
	meteringPointNumber: "meteringPoint1",
	meteringPointType: "ELECTRICITY"
	timeZoneId: "EET",
	.
	.
	.
```

Tämän saavuttaakseen tulee kyseessä olevaan luokkaan merkitä, mitkä attribuutit kuuluvat mihinkin JSON-tulosteeseen. Attribuuttia ei voi merkitä kuuluvaksi kuin yhteen tulosteeseen, mutta perinnöllisyyttä voidaan käyttää hyväkseen; jos attribuutti kuuluu johonkin tulosteeseen, niin se kuuluu myös sen parent-tulosteeseen. (Deleuze 2014.)

Alla olevassa esimerkissä on merkitty MeteringPoint-luokan attribuutteja kuuluvaksi tiivistelmä-näkymään.

```
public class View {
	interface Summary{}
}

public class MeteringPoint {
	@JsonView(View.Summary.class)
	private String meteringPointNumber;

	@JsonView(View.Summary.class)
	private String meteringPointType;

	private Date created;
	private Date modified;
	private String timeZoneId;
```

3.6 Hibernate

3.6.1 Esittely

Hibernate on avoimen lähdekoodin ORM-sovelluskehys, jolla hoidetaan Java-luokkien kartoitus tietokantatauluihin ja Java-tyypeistä SQL-datatyyppeihin. Hibernate tarjoaa toteutuksen JPA-rajapintamäärityksselle ja laajentaa sitä omilla metodeillaan.

Hibernaten avulla haetaan tietokannasta tietoa ja muutetaan se Javan ymmärtämään muotoon ja toisinpäin. Hibernate nopeuttaa sovelluskehitystä, kun dataa ei tarvitse manuaalisesti käsitellä SQL:llä ja JDBC:llä. Tietokantahakuja voi tehdä Hibernatella joko sen omalla kyselykielellä HQL:llä tai Criteria-API:lla. (Hibernate Getting Started Guide n.d.)

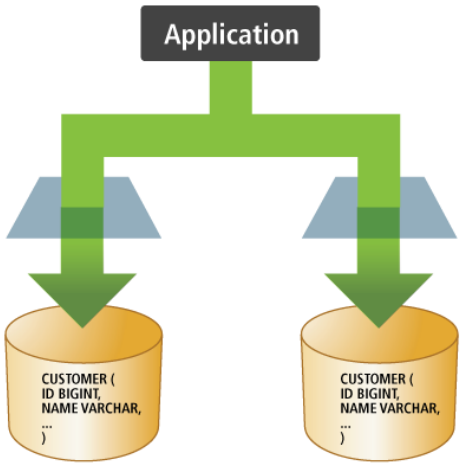
EcoReactionissa käytetään tällä hetkellä Hibernaten 3.x-versiota. Uusin versio Hibernatesta on 5.0.2.Final, joka julkaistiin 30.9.2015. Hibernatesta on siis tullut jo kaksi merkittävää versiopäivitystä. Spring-sovelluskehyksen uusimmassa versiossa on jo tuki Hibernate 5:lle (Johnson ym. 2015).

3.6.2 Hibernate 4

Hibernate 4:n tuomat suurimmat muutokset ovat tuki Multi-Tenancylle, Services API, tuki i18n-viestikoodeille lokissa sekä OSGi-tuki. Näiden lisäksi on tehty koodin siivoamista ja poistettu vanhentuneita osia. (Kapelonis 2012.)

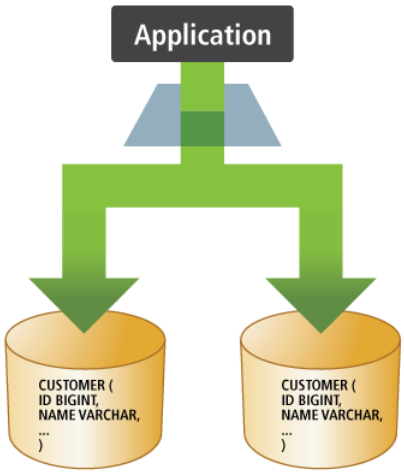
Multi-Tenancy on termi, jolla tarkoitetaan arkkitehtuuria, jossa sama instanssi palvelee useampaa asiakasta (tenant). Tällaista ratkaisua käytetään yleensä SaaS-ratkaisuissa. Multi-Tenancy-arkkitehtuurissa data eristetään niin, että asiakkaat eivät näe toisten dataa. (Hibernate User Guide 2015.)

Hibernate tarjoaa kolme eri tapaa eristää asiakkaiden data. Jokainen tapa vaatii omanlaisen tietokantaskeeman ja JDBC-asetukset. Ensimmäinen tapa on tehdä erilliset tietokannat eri asiakkaille. JDBC-yhteydet osoittaisivat eri tietokantoihin luoden näin jokaiselle asiakkaalle oman tietokantapoolin. Sovelluksessa luotaisiin JDBC Connection pool asiakaskohtaisesti ja käytettävä pooli valittaisiin sisään kirjautuneen käyttäjän tenant identifierin perusteella. Hibernate kutsuu tätä lähestymistapaa nimellä DATABASE (ks. Kuvio 12). (Mt.)



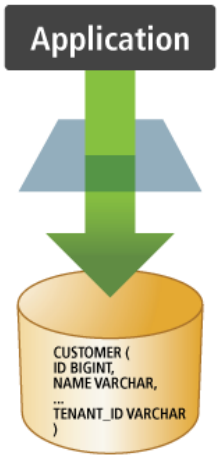
Kuvio 12. DATABASE-lähestymistapa (Hibernate User Guide 2015)

Toinen tapa on käyttää samaa tietokantainstanssia, mutta erillisiä skeemoja. JDBC-yhteyksien hallintaan on kaksi vaihtoehtoa. Ensimmäinen vaihtoehto on käyttää edellisessä tapauksessa käytettäviä erillisiä pooleja. Tämä vaatii sen, että käytettävä skeema pystytään määrittelemään asetuksissa. Toinen vaihtoehto on käyttää samaa poolia, mutta yhteyttä muokattaisiin vastaamaan tenant identifierillä määriteltyä skeemaa ennen yhteyden käyttämistä. Yhteyden muokkaus tapahtuisi esimerkiksi SQL:n set schema -käskyllä. Hibernate kutsuu tätä lähestymistapaa nimellä SCHEMA (ks. Kuvio 13). (Mt.)



Kuvio 13. SCHEMA-lähestymistapa (Hibernate User Guide 2015)

Kolmas vaihtoehto on käyttää samaa tietokantaa, skeemaa ja poolia. Eri asiakkaiden data eriteltäisiin esimerkiksi yksilöllisellä tietokantataulun kolumnin arvon perusteella. Hibernate kutsuu tätä lähestymistapaa nimellä DISCRIMINATOR (ks. Kuvio 14). (Mt.)



Kuvio 14. DISCRIMINATOR-lähestymistapa (Hibernate User Guide 2015)

Multi-Tenancya käyttäessä tietokantayhteyttä avatessa sessiolle tulee määritellä tenant identifier seuraavalla tavalla:

```
Session session = sessionFactory.withOptions()
                  .tenantIdentifier(yourTenantIdentifier)
                  ...
                  .openSession();
```

Käytettävä Multi-Tenancy-tapa tulee määrittää asetuksissa. Oletuksena käytössä ei ole mitään (NONE). Muut vaihtoehdot on lähestymistapojensa mukaan DATABASE, SCHEMA ja DISCRIMINATOR. DISCRIMINATOR ei ole käytössä vielä Hibernate 4:ssä. DATABASE- tai SCHEMA-lähestymistapaa käyttäessä ohjelmoijan tulee tehdä toteutus MultiTenantConnectionProviderille, jotta Hibernate pystyy saamaan asiakaskohtaiset yhteydet. (Mt.)

Multi-Tenancysta on hyötyä EcoReactionille, jos sitä tarjottaisiin SaaS-pohjaisena tuotteena asiakkaalle, jolla on useampia pienempiä energiayhtiöitä asiakkainaan. Tällöin EcoReaction voitaisiin asentaa asiakkaan ympäristöön yhtenä instanssina ja palvella sitä kautta useampaa energiayhtiötä. Sama idea pätee siinä tapauksessa, jos Wapice alkaa jossain vaiheessa tarjota EcoReactionia SaaS-tuotteena.

OSGi-spesifikaatiolla määritellään dynaaminen, modularisoitu järjestelmä. Järjestelmän komponentit voidaan asentaa, aktivoida ja ottaa pois käytöstä järjestelmän ajon aikana. OSGi-sovelluskehys huolehtii komponenttien riippuvuuksista, paketeista ja luokista. (Hibernate 4.2 Developer Guide 2015.)

Hibernaten versiossa 4.3 tuli täysi tuki JPA 2.1:lle. JPA 2.1-spesifikaatiossa on muun muassa tuki tallennetuille proseduureille, josta Hibernate on tehnyt oman versionsa. Tallennettujen proseduurien käsittely ei kuitenkaan ole

Hibernaten vahvinta osa-aluetta, joten niitä lienee parempi käsitellä edelleenkin JDBC:llä EcoReactionissa. (Ebersole 2013.)

3.6.3 Hibernate 5

Hibernate 5:n yksi merkittävimmistä muutoksista on tuki Java 8:n uudelle Date and Time API:lle. Tämä ominaisuus on eristetty omaan riippuvuuteensa: hibernate-java8. Se lisää tuen Java 8:n Date and Time API:n päivä- ja aikatyyppien kartoituksen tietokantatyyppeihin. (Ebersole 2015.)

Muita Hibernate 5:n tuomia muutoksia on muun muassa uusittu bootstrap API, tuki GIS (Geographical Information Systems) datalle eli avaruudelliselle ja maantieteelliselle datalle (Ebersole 2015). Hibernate 5:ssä on myös tuki Multi-Tenancyn tavalle DISCRIMINATOR.

3.6.4 Muutoksia

Hibernaten versiosta 3.6 lähtien Annotations-moduuli on liitetty osaksi Core-moduulia. Täten EcoReactionissa ei tarvitse erikseen määritellä Annotations-moduulia, vaan se on automaattisesti käytössä, kun määritellään pom.xml:ään Hibernate Core. (Hibernate n.d.)

```
<dependency>
	<groupId>org.hibernate</groupId>
	<artifactId>hibernate-core</artifactId>
	<version>5.0.2.Final</version>
<dependency>
```

EcoReactionissa käytetään DAO-luokissa käytöstä poistettua getSession()-metodia. Tämä voidaan korvata Hibernaten SessionFactory-luokan getCurrentSession()-metodilla. SessionFactory tulee injektoida DAO-luokkiin, kuten alla olevassa esimerkissä.

```
private SessionFactory sessionFactory;

protected SessionFactory getSessionFactory() {
	return sessionFactory;
}

public void setSessionFactory(SessionFactory sessionFactory) {
	this.sessionFactory = sessionFactory;
}
```

SessionFactory pitää myös merkitä XML-konfiguraatioon beanin ominaisuuksiin seuraavanlaisesti:

```
<bean id="socialUserDAO" class="path.to.class.HibernateSocialUserDAO">
	<property name="sessionFactory" ref="sessionFactory" />
</bean>
```

4 TULOKSET JA LOPPUPOHDINTA

Opinnäytetyön tuloksena saatiin kattava selvitys, kuinka liittää sosiaalinen media osaksi EcoReaction-sovellusta Spring Social -projektilla, ja miten hyödyntää Facebookin ja Twitterin ominaisuuksia energianseurannan esittämiseen sosiaalisessa mediassa. Opinnäytetyössä pohdittiin, mitä hyötyä sosiaalisen median integraatiosta on sovellukselle, ja mitä lisäarvoa se tuo asiakkaille.

Opinnäytetyön tuloksena saatiin myös katsaus EcoReactionissa käytettävien teknologioiden uusimpiin versioihin. Uusien versioiden tuomia ominaisuuksia ja hyötyjä tutkittiin EcoReactionin kannalta.

Käytettyjen teknologioiden versiopäivitystä ei tehty opinnäytetyön puitteissa, sillä se oli teknisesti hyvin haastavaa ja vaati syvempää tuntemusta EcoReactionista. Versiopäivitetty EcoReaction saatiin opinnäytetyön loppuvaiheessa käyttöön, jolloin sosiaalisen median integraatiota voitiin alkaa

toteuttamaan kunnolla. Tämä tapahtui kuitenkin niin myöhäisessä vaiheessa, ettei sosiaalisen median integraatiota ehditty toteuttamaan täysimittaisesti.

Integraatiosta saatiin toteutettua entity-, DAO-, service- ja controller-luokat, sekä rajapinnat Facebook- ja Twitter-operaatioille. Integraatiosta jäi toteuttamatta testit luokille, tarina- / kortti-sivun generointi julkaisulle ja käyttöliittymän muutokset, joskin käyttöliittymä ei ollutkaan opinnäytetyön raameissa.

Toteutuneella prototyypillä käyttäjä voi linkittää sosiaalisen median tilinsä EcoReactioniin ja tehdä julkaisuja sosiaaliseen mediaan REST-rajapinnan kautta. Myös upotettavien julkaisujen haku onnistuu prototyypillä.

Opinnäytetyöstä on hyötyä, jos ja kun sosiaalisen median integraatiota lähdetään tosissaan toteuttamaan sovellukseen. Opinnäytetyön pohjalta on helpompi suunnitella sosiaalisen median integraation ominaisuuksia ja markkinoida sitä asiakkaille.

Java 8 tuo paljon hyviä uudistuksia, joista merkittävimpänä uusittu Date and Time API. EcoReactionissa käsitellään niin paljon päivämääriä, että tästä on todella suurta hyötyä kehityksen kannalta. Java 8:n myötä voi myös alkaa hyödyntää joitain funktionaalisen ohjelmoinnin etuja ja uusia annotaatioita controller-luokille.

Spring 4 ei tuo mukanaan kovin suuria muutoksia. Tuki Java 8:lle on kuitenkin tarpeeksi merkittävä syy päivittää EcoReaction Spring 4:ään.

Hibernaten uusia versiota tutkittaessa ei tullut vastaan sen suurempia muutoksia tai parannuksia, joita voitaisiin suoraan hyödyntää EcoReactionissa. Hibernaten perustoiminnalisuudet ovat pysyneet

samanlaisina. Uusimpaan versioon päivittäessä voi kuitenkin EcoReactionista löytyä käytöstä poistuneita metodeja.

Spring Social -projektin käyttö osoittautui erittäin työlääksi. Dokumentaatio oli vajavaista ja osittain vanhentunutta. Springin tekemät esimerkkisovellukset käyttivät Spring Socialin vanhoja versioita ja osa niistä oli muutenkin rikkinäisiä. Verkosta ei löytynyt yhtään tutoriaalia, jossa käytettäisiin Spring Social -projektin uusimpia versioita. Aikaa kului paljon hyvän esimerkin löytämiseen ja sen päivittämiseen tukemaan uusimpia versioita.

Springin verkkosivuille ei ollut päivitetty viimeisimpiä versioita moduuleista ja dokumentaatioista, vaan vanhentuneita versioita mainostettiin viimeisimpinä. Oikeasti viimeisimmät versiot löytyivät Maven Centralista ja vaihtamalla dokumentaatioiden URL-poluissa versionumeroita. Tällainen laiminlyönti kehittäjien ja julkaisijan toimesta saa epäilemään, tuetaanko projektia täysillä ja onko siihen kannattavaa panostaa.

Tarinoiden julkaisu Facebookiin vaati paljon selvittelyä. Spring Social Facebookin dokumentaatiossa ei kerrottu siitä paljoa. Facebookin oma dokumentaatio on kattava, mutta sieltä ei löytynyt yhdestä paikasta kaikkea tarvittavaa tietoa, vaan tiedonmuruja oli ripoteltu ympäriinsä.

Päänvaivaa tuotti erityisesti se, että tarinan julkaiseminen ei näyttänyt menevän läpi, vaikka minkäänlaista virheilmoitusta ei tullut ja kaikki tarvittavat tiedot ja oikeudet oli määritelty. Lopulta selvisi, että tarina ilmestyi käyttäjän toimintalokiin, eikä sitä saakaan näkyviin aikajanalle ennen kuin sovellus on arvioitu Facebookin toimesta.

Tarinaa pystyi kuitenkin tarkastelemaan toimintalokin kautta klikkaamalla tarinan aikaleimaa tai julkaisemalla tarina Facebookin Graph API Explorer -rajapintatestaustyökalun kautta, jolloin se ilmestyi aikajanalle.

Kaikesta huolimatta Spring Social Facebookin tarjoama rajapinta on selkeä ja hyvin toteutettu. Se ei kuitenkaan poista tarvetta tutustua myös syvällisesti Facebookin omiin rajapintoihin ja dokumentaatioihin, jos haluaa tehdä jotain perusjulkaisuja erikoisempaa.

Spring Social Twitteriin käyttö oli hyvin sujuvaa sen jälkeen, kun oli tutustunut Spring Social Facebookin käyttöön. Twitterin tarjoamat toiminnot ovat paljon yksinkertaisempia ja selkeämpiä kuin Facebookin. Twitterin kanssa ei törmätty suurempiin ongelmiin.

Opinnäytetyötä tehdessä taidot ja ymmärrys Java EE -sovelluksen tekemisestä Spring-sovelluskehyksellä kasvoivat huimasti. Tutoriaali, jonka pohjalta Spring Social -projektia lähdettiin testailemaan, käytti Spring Social -projektin vanhoja versioita, Java-konfiguraatiota ja Servlettejä. Tutoriaali päivitettiin tukemaan Spring Social -projektin uusimpia versioita ja vastaamaan EcoReactionin toteutustapaa. Tämä sisälsi Java-konfiguraation muuttamisen XML-pohjaiseksi, web.xml-tiedoston luomisen ja ohjelmointikäytänteiden muuttamisen.

Jo nämä toimenpiteet vaativat hurjasti uuden opiskelua, mutta kartuttivat osaamista samassa mittakaavassa. Näiden lisäksi sosiaalisen median rajapintoihin ja kehittäjätyökaluihin tutustuminen, sekä käytettyjen teknologioiden uusimpiin versioihin perehtyminen kehittivät ammattiosaamista uudelle tasolle.

LÄHTEET

Count Von Count. 2015. Count Von Count -käyttäjän Twitter-tili. Viitattu 25.11.2015.
https://twitter.com/CountVonCount

Deleuze, S. 2014. Latest Jackson integration improvements in Spring. Viitattu 10.10.2015.
https://spring.io/blog/2014/12/02/latest-jackson-integration-improvements-in-spring

Ebersole, S. 2013. Callable statement support. Hibernaten verkkosivut. Viitattu 22.10.2015.
http://in.relation.to/2013/04/03/callable-statement-support/

Ebersole, S. 2015. Hibernate ORM 5.0 has gone Final. Hibernaten verkkosivut. Viitattu 22.10.2015.
http://in.relation.to/2015/08/20/hibernate-orm-500-final-release/

Evans, B., Warburton, R. 2014. Java SE 8 Date and Time. Oraclen verkkosivut. Viitattu 18.10.2015.
http://www.oracle.com/technetwork/articles/java/jf14-date-time-2125367.html

Global social networks ranked by number of users 2015. 2015. Statista verkkosivut. Viitattu 17.10.2015.
http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

Hibernate 4.2 Developer Guide. 2015. Hibernaten dokumentaatio. Viitattu 24.10.2015.
http://docs.jboss.org/hibernate/orm/4.2/devguide/en-US/html_single/

Hibernate Getting Started Guide. N.d. Viitattu 4.10.2015.
http://docs.jboss.org/hibernate/orm/4.3/quickstart/en-US/html_single/

Hibernate – Software Components. N.d. Wikipedian verkkosivut. Viitattu 4.10.2015.
https://en.wikipedia.org/wiki/Hibernate_(Java)#Software_components

Hibernate User Guide. 2015. Hibernaten dokumentaatio. Viitattu 22.10.2015.
http://docs.jboss.org/hibernate/orm/5.0/userGuide/en-US/html_single/#d5e3197

Hughes, T. 2014. Power to the pot: Marijuana growers face electric fee. USA Today verkkosivut. Viitattu 17.10.2015. http://www.usatoday.com/story/news/nation/2014/11/10/marijuana-power-consumption/18670007/

Java. N.d. Wikipedian verkkosivut. Viitattu 14.10.2015. https://en.wikipedia.org/wiki/Java_(programming_language)

Java version history. N.d. Wikipedia Java SE 8. Viitattu 14.10.2015. https://en.wikipedia.org/wiki/Java_version_history#Java_SE_8_.28March_18.2C_2014.29

Johnson, H. 2014. Java 8 Optional: What's the Point? Viitattu 12.10.2015. http://huguesjohnson.com/programming/java/java8optional.html

Johnson, Hoeller, Donald, Sampaleanu, Harrop, Risberg, Arendsen, Davi-son, Kopylenko, Pollack, Templier, Vervaet, Tung, Hale, Colyer, Lewis, Leau, Fisher, Brannen, Laddad, Poutsma, Beams, Abedrabbo, Clement, Syer, Gierke, Stoyanchev, Webb, Winch, Clozel, Nicoll & Deleuz. 2015. Spring Framework Reference Documentation. Viitattu 7.10.2015. http://docs.spring.io/spring/docs/4.2.2.BUILD-SNAPSHOT/spring-framework-reference/htmlsingle/

Kapelonis, K. 2012. InfoQ – Jboss Releases Hibernate 4.0. Viitattu 4.10.2015. http://www.infoq.com/news/2012/01/hibernate-4-released/

Paul, J. 2014. Java 8 – 20 Examples of Date and Time API. Javarevisited blogi. Viitattu 24.10.2015. http://javarevisited.blogspot.fi/2015/03/20-examples-of-date-and-time-api-from-Java8.html

Permissions. N.d. Facebook Developers -dokumentaatio. Viitattu 1.11.2015. https://developers.facebook.com/docs/facebook-login/permissions/v2.5#overview

Photo Card. N.d. Twitter Developers -dokumentaatio. Viitattu 22.11.2015. https://dev.twitter.com/cards/types/photo

Preventing NullPointerException N.d. Wikibooks. Viitattu 11.10.2015. https://en.wikibooks.org/wiki/Java_Programming/Preventing_NullPointerException

Ritter, S. 2015. Oracle Massive Open Online Course: Java SE 8 Lambdas and Streams. Oraclen verkkokurssi. Viitattu 13.10.2015

https://apexapps.oracle.com/pls/apex/f?p=44785:141:4768096251904:::P141_PAGE_ID,P141_SECTION_ID:250,1808#prettyPhoto

Rouse, M., Haughn, M. 2015. Web 2.0. WhatIs-verkkosivut. Viitattu 23.10.2015
http://whatis.techtarget.com/definition/Web-20-or-Web-2

Saxena, A. 2013. Is there a difference between Facebook's Open Graph API and its Graph API? Vastaus kysymykseen Stack Overflow -sivustolla. Viitattu 1.11.2015.
http://stackoverflow.com/questions/16639799/is-there-a-difference-between-facebooks-open-graph-api-and-its-graph-api

Sosiaalinen media. N.d. Wikipedian verkkosivut. Viitattu 17.10.2015.
https://fi.wikipedia.org/wiki/Sosiaalinen_media#cite_note-4

Submission Process. N.d. Facebook Developers -dokumentaatio. Viitattu 8.11.2015.
https://developers.facebook.com/docs/opengraph/submission-process

Walls, C., Donald, K. & Clarkson, R. 2015a. Spring Social Facebook Reference. Viitattu 1.11.2015.
http://docs.spring.io/spring-social-facebook/docs/2.0.1.RELEASE/reference/htmlsingle/

Walls, C., Donald, K. & Clarkson, R. 2015b. Spring Social Reference. Viitattu 26.9.2015.
http://docs.spring.io/spring-social/docs/1.1.2.RELEASE/reference/htmlsingle/

Walls, C., Donald, K. & Clarkson, R. 2015c. Spring Social Twitter Reference. Viitattu 15.11.2015.
http://docs.spring.io/spring-social-twitter/docs/1.1.0.RELEASE/reference/htmlsingle/

What’s New in JDK 8. 2014. Oraclen verkkosivut. Viitattu 14.10.2015.
http://www.oracle.com/technetwork/java/javase/8-whats-new-2157071.html

What to expect when you’re expecting to share. N.d. Runkeeper verkkosivut. Viitattu 11.10.2015.
https://support.runkeeper.com/hc/en-us/articles/202222909-What-To-Expect-When-You-re-Expecting-To-Share

Yritys. N.d. Wapice Oy:n verkkosivut. Viitattu 18.9.2015.
https://www.wapice.com/fi/tietoa-meista/tietoa-wapicesta

LIITTEET

Liite 1. JdbcUsersConnectionRepository-luokan tietokantataulu

Field	Type	Null	Key
userId	varchar(255)	NO	PRI
providerId	varchar(255)	NO	PRI
providerUserId	varchar(255)	NO	PRI
rank	int(11)	NO	
displayName	varchar(255)	YES	
profileUrl	varchar(512)	YES	
imageUrl	varchar(512)	YES	
accessToken	varchar(255)	NO	
secret	varchar(255)	YES	
refreshToken	varchar(255)	YES	
expireTime	bigint(20)	YES	