TODOAPP

Dokumentaatio v.0.5

1. Johdanto

TodoApp on monipuolinen, kollaboratiivinen tehtävälista. Sovelluksen avulla käyttäjä pystyy hallitsemaan helposti työn ja vapaa-ajan tehtäviään yksin ja yhdessä muiden käyttäjien kanssa.

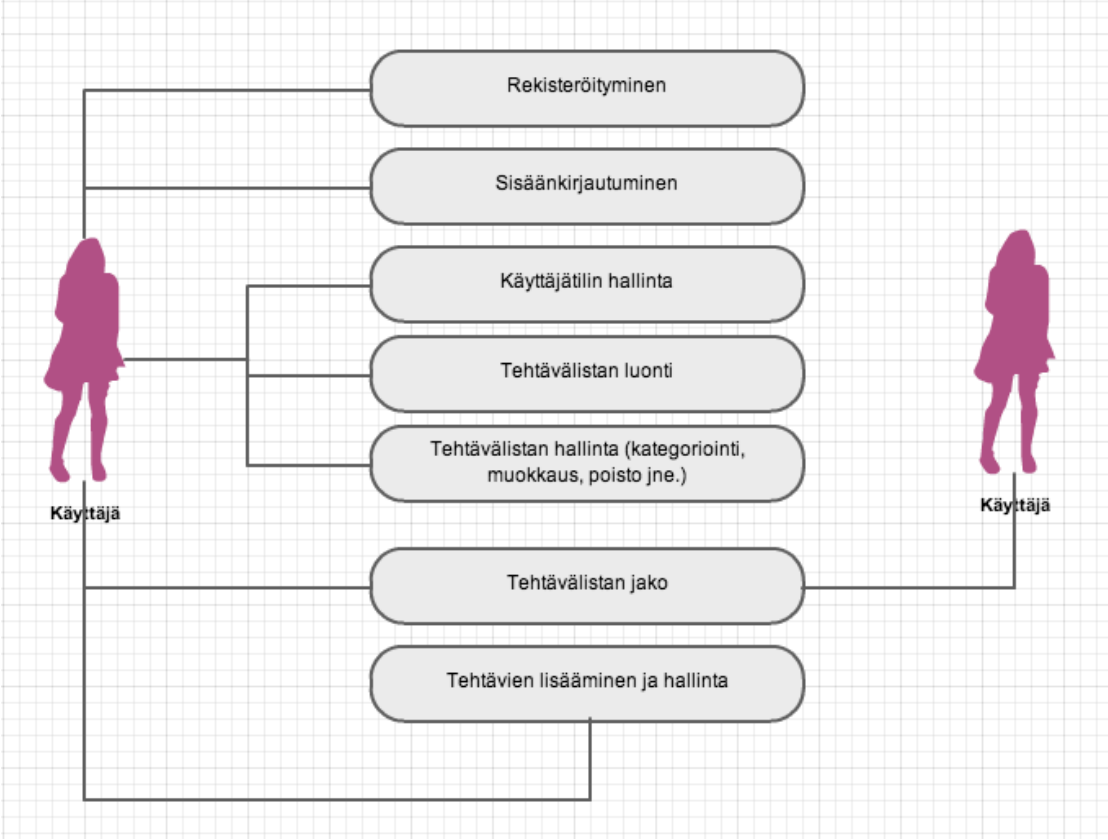
Sovellus toteutetaan käyttäen node.js-alustaa [http://nodejs.org/], joka on kevyt, tapahtumavetoinen palvelinpään ohjelmointiin tarkoitettu avoimen lähdekoodin projekti. Pysyväisdatan tallennukseen käytetään MySQL-tietokantaa.

Sovellusta voidaan ajaa tietojenkäsittelytieteen laitoksen koneissa, joihin on asennettu node.js tai vaihtoehtoisesti Heroku:n [http://heroku.com/] pilvipalvelussa.

Sovellus tukee ensimmäisessä vaiheessa ainoastaan Google Chrome –selainta versiosta 23 alkaen. Selaimen tulee sallia JavaScriptin suorittaminen, sillä käyttöliittymä on dynaaminen ja tosiaikainen.

2. Yleiskuva järjestelmästä

Järjestelmässä on vain yhden kaltaisia käyttäjiä. Sovelluksen käyttö edellyttää rekisteröitymistä ja sisäänkirjautumista.



3. Käyttötapaukset

3.1. Käyttäjät

3.1.1. Rekisteröinti

Voidakseen käyttää sovellusta, käyttäjän tulee rekisteröityä. Rekisteröityminen tapahtuu täyttämällä sähköpostiosoite ja salasana rekisteröitymislomakkeeseen.

3.1.2. Sisäänkirjautuminen

Käyttäjä voi kirjautua sovellukseen syöttämällä käyttäjätunnuksensa ja salasanansa sisäänkirjautumislomakkeeseen.

3.1.3. Uloskirjautuminen

Sisäänkirjautunut käyttäjä voi lopettaa sovelluksen käytön kirjautumalla ulos.

3.1.4. Käyttäjätietojen muokkaus

Käyttäjä voi vaihtaa salasanaansa ja kirjautumiseen käytettävää sähköpostiosoitetta.

3.2. Tehtävälista

3.2.1. Tehtävälistan luonti

Sisäänkirjautunut käyttäjä voi luoda uuden tehtävälistan.

3.2.2. Tehtävälistan muokkaus

Sisäänkirjautunut käyttäjä voi muokata luomansa tehtävälistan tietoja (kategoriat ym.).

3.2.3. Tehtävälistan poisto

Sisäänkirjautunut käyttäjä voi poistaa luomansa tehtävälistan ja kaiken sen sisältämän tiedon.

3.2.4. Tehtävälistan jako

Sisäänkirjautunut käyttäjä voi jakaa luomansa tehtävälistan valitsemilleen käyttäjille. Lisätyillä käyttäjillä on oikeutta lisätä, poistaa ja muokata tehtävälistaan kuuluvia tehtäviä, mutta heillä ei ole mahdollisuutta muokata itse tehtävälistaan esimerkiksi lisäämällä muita käyttäjiä tai vaihtamalla sen nimeä.

3.3. Tehtävät

3.3.1. Tehtävän lisääminen

Sisäänkirjautunut käyttäjä voi lisätä uusia tehtäviä hallitsemiinsa tehtävälistoihin.

3.3.2. Tehtävän muokkaaminen

Sisäänkirjautunut käyttäjä voi muokata hallitsemiensa tehtävälistojen tehtäviä.

3.3.3. Tehtävän poistaminen

Sisäänkirjautunut käyttäjä voi poistaa tehtäviä hallitsemistaan tehtävälistoista.

3.4. Tagit

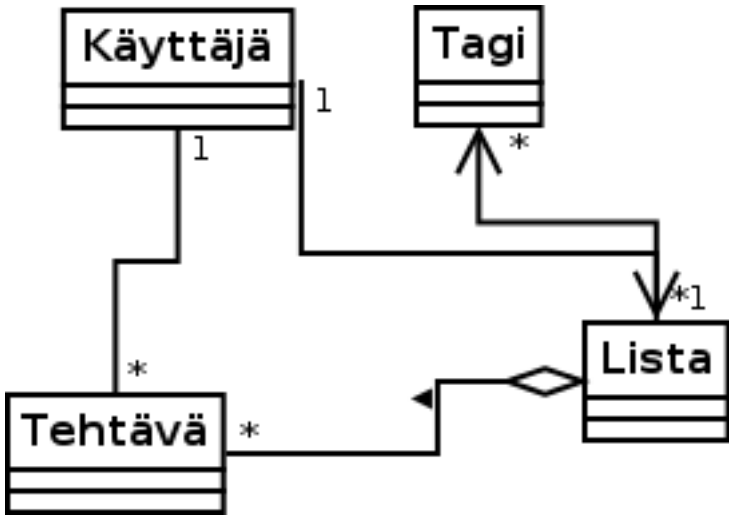
3.4.1. Tagin lisääminen

Sisäänkirjautunut käyttäjä voi lisätä useita tageja kullekin tehtävälistalle.

3.4.2. Tagin poistaminen

Sisäänkirjautunut käyttäjä voi poistaa tagin luomistaan tehtävälistoista.

4. Järjestelmän tietosisältö



Tietokohde: Käyttäjä

Attribuutti	Arvojoukko	Kuvailu
Tunnus.	Kokonaisluku.	Uniikki tunnus, joka identifioi käyttäjän.
Sähköposti	Merkkijono, max. 80 merkkiä.	Käyttäjän sähköpostiosoite, joka toimii käyttäjätunnuksena.
Salasana	Merkkijono, max 255 merkkiä.	Käyttäjän salasanasta muodostettu turvallinen tiiviste.
Etunimi	Merkkijono, max. 100 merkkiä	Käyttäjän etunimi.
Sukunimi	Merkkijono, max. 100 merkkiä	Käyttäjän sukunimi.

Käyttäjä on järjestelmään rekisteröitynyt käyttäjä. Käyttäjätunnuksena käytetään käyttäjän sähköpostiosoitetta. Samalla sähköpostiosoitteella voi

rekisteröidä vain yhden käyttäjän. Käyttäjän tulee olla kirjautunut sisään luodakseen listoja, tehtäviä ja kategorioita.

Tietokohde: Lista

Attribuutti	**Arvojoukko**	**Kuvailu**
Tunnus.	Kokonaisluku.	Uniikki tunnus, joka identifioi listan.
Nimi	Merkkijono, max. 255 merkkiä.	Listalle annettava nimi.
Luomishetki	Aikaleima.	Serverin aikaleima ajalta, jolloin tehtävä on luotu.
Muokkaushetki	Aikaleima.	Serverin aikaleima ajalta, jolloin tehtävää on muokattu.

Lista on tehtävälistan perusyksikkö. Kullakin listalla on yksi kategoria.

Tietokohde: Tagi

Attribuutti	**Arvojoukko**	**Kuvailu**
Tunnus.	Kokonaisluku.	Uniikki tunnus, joka identifioi kategorian.
Luoja	Kokonaisluku.	Kategorian luoneen käyttäjän tunnus.
Nimi	Merkkijono, max 255 merkkiä.	Kategorialle annettava nimi.

Tagit määrittelevät tehtävälistan luokittelun. Tageja voi olla useita kussakin tehtävälistassa.

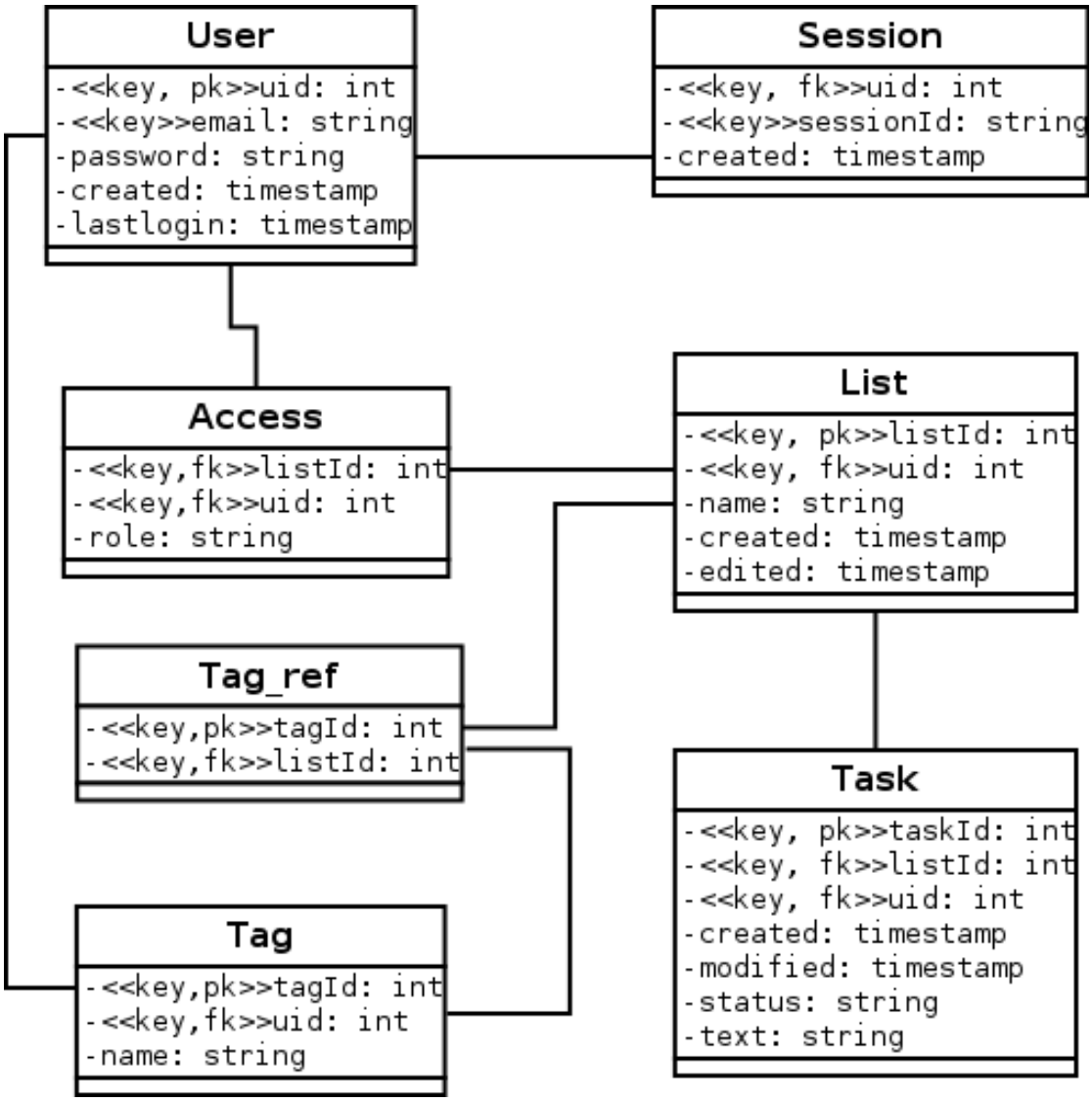
Tietokohde: Tehtävä

Attribuutti	**Arvojoukko**	**Kuvailu**
Tunnus	Kokonaisluku.	Uniikki tunnus, joka identifioi tehtävän.
Lista	Kokonaisluku.	Viittaus listan tunnukseen, johon tehtävä kuuluu.
Luoja	Kokonaisluku.	Viittaus käyttäjään, joka on luonut tehtävän alunperin.
Muokkaaja	Kokonaisluku.	Viittaus käyttäjään, joka on muokannut tehtävää viimeksi.
Luomishetki	Aikaleima.	Serverin aikaleima ajalta, jolloin tehtävä on luotu.
Muokkaushetki	Aikaleima.	Serverin aikaleima ajalta, jolloin tehtävää on muokattu.
Tila	Merkkijono, max 50	Tehtävän tila, ”kesken”

	merkkiä.	tai ”valmis”.
Teksti	Merkkijono, max 255 merkkiä.	Tehtävän tekstiosio, jossa määritellään mitä tehtävä koskee.

Tehtävä kuuluvat aina johonkin listaan ja niillä on jokin selite. Tehtävällä on useita eri tiloja; tehtävä voi olla kesken tai valmis.

5. Relaatiotietokantakaavio



Relaatiotietokantakaavio poikkeaa käsitemallista siten, että siihen on lisätty pari aputaulua [Access, Session ja Tag_ref]. Näistä Session taulussa seurataan sisäänkirjautuneiden käyttäjien sessioita. Access taulussa puolestaan mallinnetaan käyttäjien pääsynhallintaa eri listoihin. Tag_ref-taulu huolehtii kunkin käyttäjän tagien parittamisesta eri listoihin.

6. Järjestelmän yleisrakenne

Järjestelmä noudattaa MVC-arkkitehtuurimallia. Serveripuolella mallit sijaitsevat models-kansiossa, kontrollerit routes-kansiossa ja näkymät views-kansiossa. Järjestelmän asetukset löytyvät config-kansiosta – osa asetuksista annetaan ympäristömuuttujissa (tarkemmin myöh.). Public-kansiossa sijaitsee kuvat, tyylitiedostot sekä javascript-tiedostot ja –templatet. Css-tiedostot generoidaan less-kansiossa olevien less-tyyilitiedostojen pohjalta.

Serveripuolelta tarjoillaan ainoastaan kaksi näkymää, sivu sisäänkirjautumista ja rekisteröitymistä varten sekä erillinen sivu sisäänkirjautuneille käyttäjille. Varsinaiset näkymät rakennetaan public-kansion javascript-tiedostojen ja –templatejen avulla.

Kansio node_modules sisältää node.js:n käyttämät kirjastot.

```
├──── config
├──── less
│   └──── includes
├──── model
├──── node_modules
├──── public
│   ├──── css
│   ├──── img
│   ├──── js
│   │   ├──── controllers
│   │   └──── lib
│   └──── templates
├──── routes
└──── views
```

7. Järjestelmän komponentit

app.js
Ohjelman käynnistystiedosto. Alustaa ohjelmiston konfiguraatiot ja yhteyden MySQL-tietokantaan sekä käynnistää serverin oletusarvoisesti porttiin ***8080*** tai ympäristömuuttujana ***PORT*** välitettyyn porttiin.

package.json
Sisältää sovelluksen riippuvuudet. Riippuvuudet haetaan koneelle ajamalla ***npm install***.

config/development.js
config/production.js
Kehitysy- ja tuotantoympäristön konfiguraatiot. Tällä hetkellä sisältää ainoastaan tietokantayhteyden asetukset.

views/login.html
Sisäänkirjatumis- ja rekisteröitymissivu. Palvelin tarjoilee sivun autentikoimattomalle käyttäjälle. Tarjoaa lomakkeet olemassa olevan käyttäjän kirjautumiselle sekä uuden käyttäjän rekisteröitymiselle. Lomakkeet generoidaan ***public***-kansion javascript kutsuilla ja templateilla.

views/app.html
Sovelluksen pääsivu. Palvelin tarjoilee sivun autentikoiduille käyttäjille. Sisältää pohjan sovelluksen käyttöliittymälle, joka luodaan ***public***-kansion javascript kutsuilla ja templateilla.

routes/index.js
Käsittelee sovelluksen http-reitityksen (kontrollerit). Ottaa vastaan POST ja GET pyynnöt ja välittää ne eteenpäin oikeille objekteille, jotka käsittelevät varsinaisen logiikan.

routes/session.js
Sisäänkirjautumisen ja rekisteröinnin hallinta. Tarjoaa toiminnallisuudet uuden käyttäjän rekisteröitymiselle, olemassa olevan sisäänkirjautumiselle sekä session hallinnan.

routes/list.js
Listojen hallinta. Tarjoaa CRUD-operaatiot listoille.

routes/error.js
Yleinen palvelinpuolen virheidenhallinta.

model/lists.js
Listojen tietokantamalli. Suorittaa SQL-kyselyt tietokantaan.

model/sessions.js
Sessioiden tietokantamalli. Suorittaa SQL-kyselyt sessio-tauluun.

model/tasks.js
Ei toteutettu vielä

model/users.js
Käyttäjien tietokantamalli. Luo käyttäjät tietokantaa ja validoi sisäänkirjatumiset.

less/*
Less-tyylitiedostot pilkottuina loogisiin osiin.

public/css/main.css
Less-tiedostoista generoitu päätyylitiedosto.

public/js/login-bootstrap.js

Sisäänkirjatumisivun alustus require.js-tiedosto. Käynnistää kaiken muun js-toiminnallisuuden, kuten lomakkeiden lähettämisen ja validoinnin sisäänkirjatumissivulla.

public/js/app-bootstrap.js
Pääohjelman alustus require.js-tiedosto. Käynnistää kaiken muun js-toiminnallisuuden, kuten lomakkeiden lähettämisen ja validoinnin ohjelman pääsivulla.

public/controllers/*
Kukin kontrolleri edustaa require.js-moduulia, jolla on oma erityinen tehtävänsä käyttöliittymässä.

public/controllers/add-list.js
Listan lisääminen käyttöliittymässä.

public/controllers/left-panel.js
Käyttöliittymän vasemman paneelin generointi.

public/controllers/list-view.js
Listojen listaaminen käyttöliittymässä.

public/controllers/login.js
Sisäänkirjautumislomakkeen hallinta.

public/controllers/middle-panel.js
Pääsovelluksen keskimmäinen paneeli.

public/controllers/notification.js
Sovelluksen lähettämät notifikaatiot.

public/controllers/register.js
Rekistreöitymislomakkeen hallinta.

public/controllers/right-panel.js
Käyttöliittymän oikean paneelin generointi.

public/controllers/toolbar.js
WIP

Javascript-kirjastot

public/js/lib/bacon-bjq.js
Bacon bjq. Jquery-toteutus bacon-kirjastosta.

public/js/lib/bacon.js
Bacon.js. Reaktiivisen käyttöliittymän koodaamiseen tarkoitettu kirjasto. Käytetään eri lomakkeiden lähettämisessä.

public/js/lib/handlebars.js
Handlebars.js. Javascript templtetointiin tarkoitettu kirjasto.

public/js/lib/hbs.js
Hbs.js. Reuquire.js-liima handlebars.js-kirjastoa varten.

public/js/lib/hbs/i18nprecompile.js
Hbs.js:n riippuvuus.

public/js/lib/hbs/json2.js
Hbs.js:n riippuvuus.

public/js/lib/hbs/underscore.js
Hbs.js:n riippuvuus.

public/js/lib/helpers.js
Kustomi apufuktioita sovelluksen käyttöön.

public/js/lib/jquery-1.10.2.min.js
jQuery-kirjaston minifoitu tuotantoversio 1.10.2.

public/js/lib/lodash.js
Lodash.js. Funktionaalisen javascript-ohjelmoinnin kirjasto. Ei käytössä vielä

public/js/lib/moment.js
Moment.js. Paranneltu ajanhallinta javascriptiin. Ei käytössä vielä.

public/js/lib/require.js
Require.js. Käytetään käyttöliittymän javascriptin modularisointiin ja parempaan riippuvuuksien hallintaan.

public/js/lib/text.js
Handlebars.js:n riippuvuus.

8. Käyttöliittymä

Käyttöliittymä on toteutettu single page app –logiikkaa noudattaen. Varsinaisia perusnäkymiä on vain kaksi.

Sisäänkirjatuminen (login.html):
Näytetään kaikille käyttäjille, joilla ei ole avointa sessiota. Tästä ei pääse eteenpäin ellei kirjaudu sisään.

Sovellus (app.html):
Näytetään kaikille sisäänkirjautuneille käyttäjille. Jaettu neljään palstaan:

Toolbar
Logo ja uloskirjatumispainike.

LeftPanel
Tagien listaus ja sisällön suodattaminen niiden perusteella. Uusien listojen ja tagien lisääminen. Uuden listan lisääminen luo MiddlePaneliin listaelementin.
MiddlePanel
Tehtävälistojen listaus. Kullakin tehtävällä on erikseen edit & share –painikkeet, joiden avulla listaelementtiä voi muokata tai jakaa toisille käyttäjille. Näiden painikkeiden painaminen avaa modaali-ikkunaan lomakkeen, jolla voi suorittaa haluamiaan toimenpiteitä. Listaelementtiä painamalla RightPanel populoidaan listaan kuuluvilla tehtävillä

RightPanel
Kunkin tehtävälistan tehtävien listaus. Tehtäviä voi lisätä paneelin lomakkeella, muokata tuplaklikkaamalla, merkitä tehdyksi checkboxista ja poistaa painamalla punaista ruksia.

9. Asennustiedot

Sovellus on asennettu ajettavaksi Herokussa. Tämänhetkinen url on dome.herokuapp.com.

Sovelluksen lähdekoodin saa kloonattu GitHubista komennolla
git clone https://github.com/JaakkoL/todoapp.git.

Sovelluksen saa asennettua muuhun ympäristöön muokkaamalla config-kansiossa olevaa production.js-tiedostoa. Sovellukselle tulee antaa pääsy MySQL-tietokantaan antamalla tunnukset tiedostossa seuraavaan tyyliin:

```
var config = {
 db : {
  host     : 'example.com',
  user     : 'username,
  password : 'password',
  database : 'databsename'
 }
}
```

Lisäksi sovellus tarvitsee node (http://nodejs.org/) ja npm (npmjs.org) ohjelmat. Npm tulee tätä nykyä tuoreen node.js-asennuksen mukana.

Ohjelman riippuvuudet sijaitsevat ***package.json***-tiedostossa ja ne saa asennettua ajamalla ohjelman juuressa komento:
npm install.

10. Käynnistys- /käyttöohje

Sovelluksen saa käynnistettyä, kun riippuvuudet on ladattu ja tietokanta konfiguroitu. Käynnistyskomento sovelluksen juuressa on:
NODE_ENV=<konfiguraationympäristö> node app.js

Ohjelma pyörii oletusarvoisesti portissa ***8080***, mutta portin voi syöttää myös ympäristömuuttujassa seuraavalla tavalla:
PORT=XX NODE_ENV=<konfiguraationympäristö> node app.js

11. Testaus, tunnetut bugit ja putteet & jatkokehitysideat

Sovelluksen testaus on ollut minimaalista. Aikapaineista johtuen, luovuin tietoisesti testiskriptien kirjoittaimisesta. Kerkesin vain hieman aloittaa node.js sovelluksen testaamisen harjoitelua. Käyttöön olisi tullut mocha-niminen testiframework. Pari alustavaa testi löytyy kansiosta test.

Osa toiminnallisuudesta jäi puuttumaan lopullisesta tuotteesta:

1) **Listojen suodatus tagien perusteella.** Kuhunkin listaan voi lisätä vapaavalintaisen määrän tageja ja tallentuvat tietokantaan oikein. Tageilla ei kuitenkaan tehdä vielä mitään. Tageja ei myöskään vielä listata missään.

2) **Käyttäjätietojen muokkaus.** Rekisteröitynyt käyttäjä ei voi muokata omia tietojaan.

Tunnetut bugit:

1) Pieniä visuaalisia bugeja siellä täällä.

Jatkokehitysideoita:

1) Autocomplete tagien syöttöön. Tämä on suhteellisen yksinkertainen toteuttaa yhdellä tietokantakyselyllä. Käyttöliittymä tälle on jo valmiina.
2) Autocomlete jakamiselle. Muiden käyttäjien automaattinen listaaminen syötettäessä sähköpostiosoitetta.
3) Testauksen automatisointi.
4) Puuttuvien toiminnallisuuksien toteuttamien.
5) Käyttöliittymän viilaaminen.

12. Omat kokemukset

Tietokantasovelluksen teko oli erittäin mielenkiintoista ja antoisaa, joskin paikoin hyvin työlästä valitusta toteutustavasta johtuen. Koska minulla oli pohjatietämystä aiheesta jo melkoisesti, en halunnut valita helppoa tietä, vaan päätin keskittyä itselleni hieman vieraampiin ympäristöihin.

Lopputulemana sain sen vaikutelman, että node.js sopii erittäin hyvin web-sovellusten tekoon. Tekeminen on selkeää ja dokumentaatiota sekä ohjeita aihepiiristä löytyy melkoisesti.

Käyttöliittymän tekeminen puhtaasti javascriptilla käyttäen javascript-templetointia mahdollistaa paremman käytettävyyden tuomisen sovellukseen. Se saattaa aiheuttaa haasteita sovelluksen rakenteen ylläpitämiseen, mutta on voitettavissa hyvällä suunnittelulla ja refaktoroinnilla.

13. Liitteet

- Lähdekoodi
- Tietokannan määrittelyn sql-lauseet (create-tables.sql, drop-tables.sql)

Liitteet löytyvät sovelluksen doc-kansiosta