

Helsingin yliopisto, tietojenkäsittelytieteen laitos

Aineopintojen harjoitustyö: Tietokantasovellus

Gallery-site

<http://gallery.tuhoojabotti.com>

27.4.2014

014245291 Ville Lahdenvuo
ville.lahdenvuo@cs.helsinki.fi
<http://tuhoojabotti.com>

1. Sisällys

- [1. Sisällys](#)
- [2. Johdanto](#)
- [3. Yleiskuva järjestelmästä](#)
 - [3.1. Käyttötapauskaavio](#)
 - [3.2. Käyttäjäryhmät](#)
 - [3.2.1. Jokamies](#)
 - [3.2.1. Kirjautunut](#)
 - [3.2.1. Ylläpitäjä](#)
 - [3.3. Käyttötapauskuvaukset](#)
 - [3.3.1. Kuvien tuominen](#)
 - [3.3.2. Kuvien katseleminen](#)
 - [3.3.3. Kuvien arvosteaminen](#)
 - [3.3.4. Kuvien hallinta](#)
- [4. Käyttöliittymä](#)
 - [4.1. Alustava sivukartta](#)
 - [4.2. Kirjautuminen](#)
- [5. Tietosisältö](#)
 - [5.1. Alustava suunnitelma](#)
 - [5.2. Tietokohteet](#)
 - [5.2.1. Kuva](#)
 - [5.2.2. Tägi](#)
 - [5.2.3. Käyttäjä](#)
 - [5.2.4. Arvostelu](#)
 - [5.3. Relaatiotietokantakaavio](#)
- [6. Järjestelmän yleisrakenne](#)
 - [6.2. Palvelin](#)
 - [6.3. Frontend](#)
- [7. Käyttöliittymä ja järjestelmän komponentit](#)
- [8. Asennustiedot](#)
- [9. Käyttöohje](#)
- [10. Testaus, bugit ja puutteet & jatkokehitys](#)
- [11. Omat kokemukset](#)

2. Johdanto

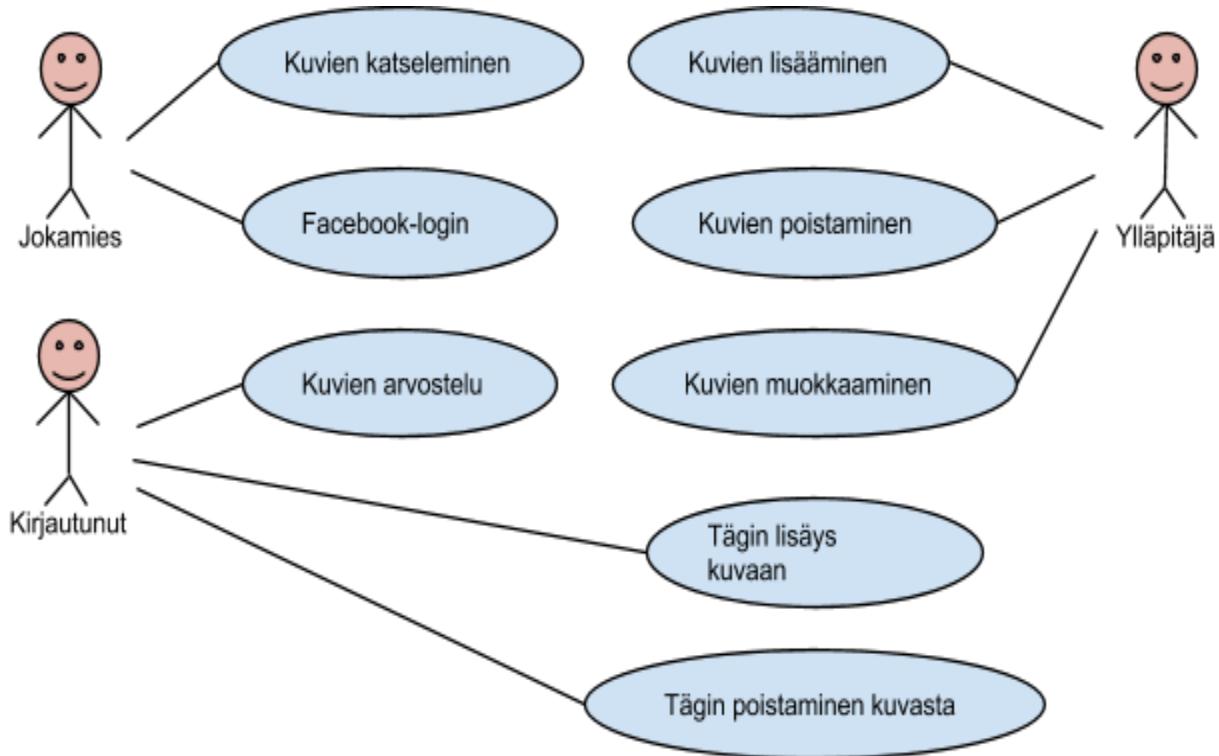
Tämä on dokumentaatio tietokantasovelluksen harjoitustyöstäni. Työn ensisijaisena tavoitteena on saada aikaan toimiva järjestelmä oman valokuvausharrastukseni tueksi. Tällä hetkellä kotisivuni tai portfolioni on aika suppea, sillä en saa töitäni esille hyvin. Olen yritynnyt ratkaista ongelmaa ulkoisilla palveluilla tai erilaisilla itse asennettavilla järjestelmillä, mutta mikään ei ole vastannut vaatimuksiani tähän mennessä. Tarkoitus on saada helposti ja näyttävästi valokuviani esille, mutta myös turvallisesti jakaa niitä ihmisiille.

Sovellusta on tarkoitus ajaa Kapsi Internet-käyttäjät Ry:n palvelimilla ja toteutuskieleksi on valittu JavaScript. Tietokannaksi tulee MySQL, sillä sellainen minulle on jo asennettu. Tietokannan vaihtaminen jälkeenpäin lienee melko helppoa, sillä sovelluksen rakenne on hyvin modulaarinen. Sovellus vaatii, että käyttäjän selain tukee JavaScriptiä.

Järjestelmä rakentuu erillisistä komponenteista, jotka toimivat yhdessä. Palvelin tarjoilee tarvittavat rajapinnat RESTful-periaatteiden mukaan ja frontend huolehtii sisällön näyttämisestä käyttäjälle. Toteutuksessa alustana on palvelinpuolella Node.js ja frontend tulee käyttämään AngularJS-sovelluskehystä.

3. Yleiskuva järjestelmästä

3.1. Käyttötapauskaavio



3.2. Käyttäjäryhmät

3.2.1. Jokamies

Jokainen voi katsella kuvia, selata tägejä ja käyttää hakua.

3.2.1. Kirjautunut

Kirjautunut käyttäjä voi arvostella kuvia sekä hallita tägejä.

3.2.1. Ylläpitäjä

Ylläpitäjällä on oikeus lisätä, poistaa ja muokata kuvia.

3.3. Käyttötapauskuvaukset

3.3.1. Kuvien tuominen

Kuvan lisääminen tapahtuu painamalla "Add Picture" ja syöttämällä kuvan tiedot lomakkeeseen.

3.3.2. Kuvien katseleminen

Aika itsestään selvyys. Kuvia pitää voida katsella selaimella ja kuvia pitää voida hakea tägien/nimen perusteella.

3.3.3. Kuvien arvosteaminen

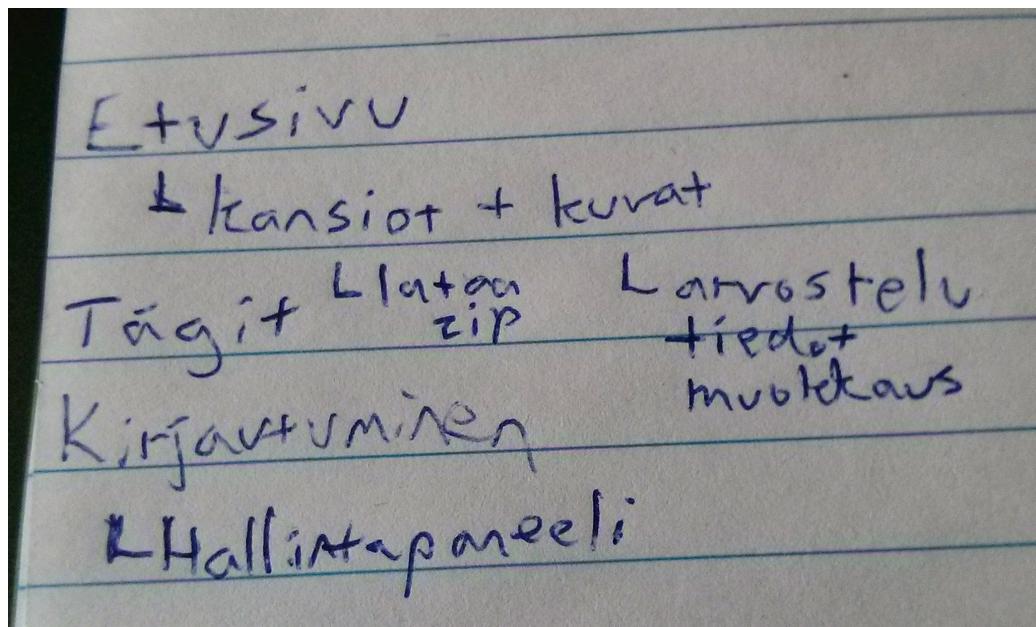
Kirjautunut käyttäjä voi arvostella kuvia asteikolla 1-5. Tämä tapahtuu yksittäisen kuvan sivulla tähtiavostelulla.

3.3.4. Kuvien hallinta

Kaikkia kuvien tietoja voi muokata käyttöliittymästä. Tietoihin kuuluu nimi, tägit ja kuvaus.

4. Käyttöliittymä

4.1. Alustava sivukartta



4.2. Kirjautuminen

Sivusto käyttää kirjautumiseen Facebookin rajapintaa. Idea on, että sivustoon ei tarvitse kirjautua ollenkaan, mutta jos haluaa arvostella kuvaaa, niin on kirjauduttava. Lisäksi kirjautumalla voi tallentaa oikeuden suojaattuun kansioon.

```
✖ ▶ POST https://secure.tuhoojabotti.com/gallery/user/verify 401 (Unauthorized) angular.js:8165
  show modal
    fb logged in!
      logged in!
        api verified! Object {id: "1247782261", name: "Ville Lahdenvuo", admin: 1}
          login cancelled!
✖ ▶ POST https://secure.tuhoojabotti.com/gallery/user/verify 401 (Unauthorized) angular.js:8165
  show modal
    login cancelled!
```

Kuten kuvasta nähdään, kun käyttäjä yrittää tehdä jotain, mikä vaatii autentikointia, palvelin vastaa tietenkin 401-virhekoodilla. Sovellus huomaa tämän ja näyttää kirjautumisnappulan. Kun käyttäjä on kirjautunut, Facebookin rajapinta antaa access tokenin, joka liitetään osaksi jokaista tulevaa HTTP-pyyntöä. Sen takia kun sovellus yrittää kutsua uudelleen, se onnistuu. Mikäli kutsu tehdään toimivalla access tokenilla, mutta käyttäjää ei ole tietokannassa, käyttäjä lisätään tietokantaan.

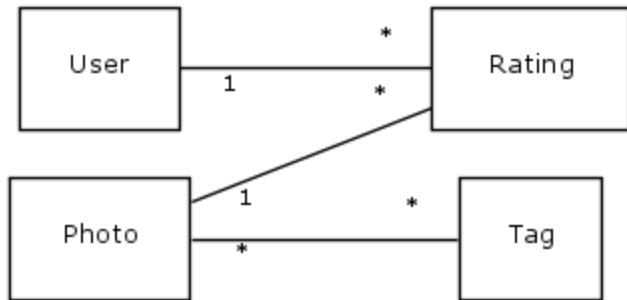
user/verify API on tarkoitettu vain kirjautumisen testaamiseen, eikä se jää lopulliseen versioon. Sovellukseen ei tule erillistä sisäänkirjautumistoimintoa.

Verifiointi on helppo yhdistää järjestelmään:

```
6   get('/users', 'users#index');
7   post('/user/verify', verify, 'users#log');
8   get('/user/:id', 'users#show');
```

5. Tietosisältö

5.1. Alustava suunnitelma



5.2. Tietokohdeet

5.2.1. Kuva

Attribuutti	Arvojoukko	Kuvailu
Nimi	Merkkijono, max. 120 merkkiä	Kuvan nimi
Polku	Merkkijono, max. 120 merkkiä	Kuvan polku
Kuvaus	Merkkijono, max. 1000 merkkiä	Kuvan kuvaus

5.2.2. Tägi

Attribuutti	Arvojoukko	Kuvailu
Nimi	Merkkijono, max. 15 merkkiä	Tägin nimi

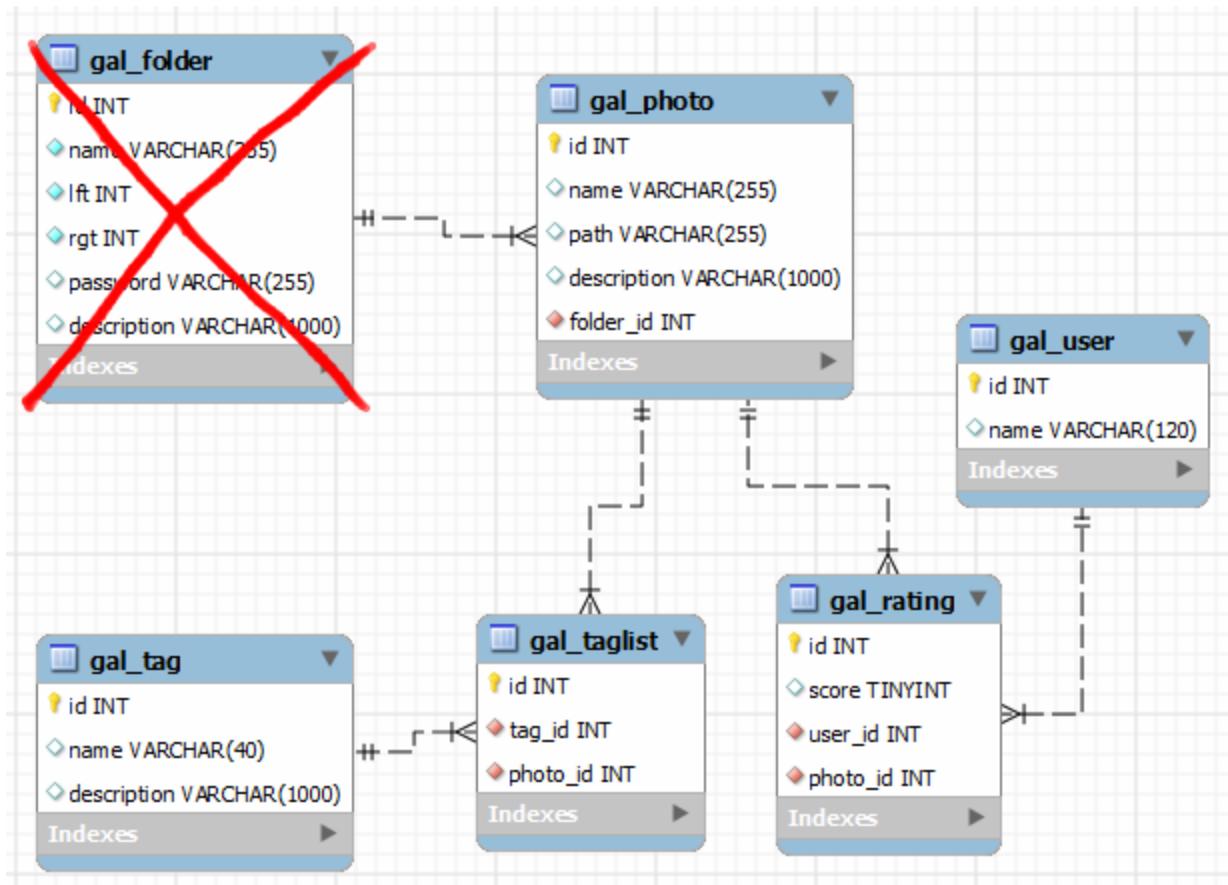
5.2.3. Käyttäjä

Attribuutti	Arvojoukko	Kuvailu
Admin	Boolean	Onko admin
Nimi	Merkkijono, max. 60 merkkiä	Käyttäjän oikea nimi

5.2.4. Arvostelu

Attribuutti	Arvojoukko	Kuvailu
Pisteet	Kokonaisluku 1-5	Tähtiavostelu

5.3. Relaatiotietokantakaavio



6. Järjestelmän yleisrakenne

Sovellus käyttää tavallaan MVC-mallia. Sovelluksen data, logiikka ja näkymät on eroteltu järkevästi toisistaan.

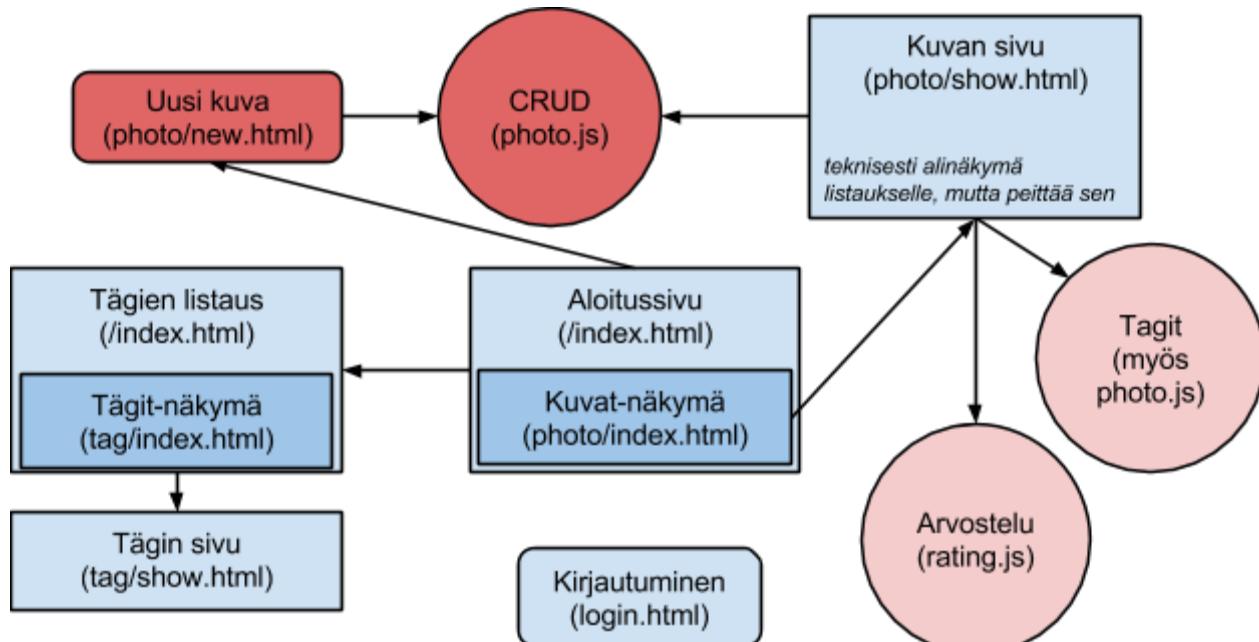
6.2. Palvelin

Palvelin löytyy kansiosta "api" ja se on vain REST-rajapinta. Mallit löytyvät "models" ja kontrollerit "controllers" kansioista. Reititys hoidetaan "routes"-tiedostolla. Apukirjastoja on kaksi, "activerecord"-moduuli on tietokanta abstraktio ja "access-control"-moduuli toimii autentikaatio middlewarena. "index.js"-tiedosto kasaa kaikki moduulit yhdeksi toimivaksi kokonaisuudeksi. Yläkansiossa oleva "app.js"-tiedosto vain hoitaa riippuvuuksien lataamisen ja injektoimisen. Sovelluksen asetukset tulee ".env"-tiedostoon.

6.3. Frontend

Frontend puoli löytyy kansiosta "www". Kaikki pyynnöt ohjataan ".htaccess"-tiedostolla "index.html"-sivulle ja reititys tapahtuu javascriptillä ja HTML5 history-apilla. Näkymät on sijoitettu "views"-kansioon ja sovelluksen osia hoitavat javascript-tiedostot ovat "js"-kansiossa ja ne on nimetty kuvaavasti. Tiedostoja voisi myös jakaa pienempiin komponentteihin, mutta sitten tulisi vielä tärkeämäksi saada mukaan tiedostojen minifointi ynnä muuta. Sovelluksen asetukset (apin url, html5-mode) ovat "main.js"-tiedostossa.

7. Käyttöliittymä ja järjestelmän komponentit



Legend: Punaisella kirjautumista vaativat operaatiot, tummemmalla vain adminille. Pyöreät reunat tarkoittaa modaalia.

Sovelluksen rakenne on vähän monimutkainen, koska näyttöjä ei ole montaa, vaan kaikki tapahtuu javascriptillä ja ajaxilla. Esimerkiksi arvostelu annetaan vain painamalla tähtiä kuvan sivulla.

8. Asennustiedot

Sovelluksen asennus on helppoa. Asennetaan Node.js ja MySQL. Kloonataan projektin tiedostot ja ajetaan npm install -komento, joka asentaa projektin riippuvuudet.

Projektin asetukset ovat .env-nimisessä tiedostossa, johon tulee kaikki asetukset tietokannasta HTTP-porttiin.

Tarvittavat asetukset ovat

- MYSQL_SERVER (serverin osoite)
- MYSQL_USER (käyttäjätunnus)
- MYSQL_PASSWORD (salasana)
- MYSQL_DATABASE (tietokanta)
- MYSQL_TABLE_PREFIX (taululle annettu prefiks)
- PORT (portti, jota REST API kuuntelee)

www-kansion sisältö kopioidaan jonkekin missä se palvelaan käyttäjälle esimerkiksi Apachella.

Tämän jälkeen vain suoritetaan “forever app.js”, joka käynnistää palvelimen ja pitää sen päällä, jos se kaatuu.

9. Käyttöohje

Aava projekti osoitteesta <http://gallery.tuhoojabotti.com/>.

Jos sinun tarvitsee kirjautua sisään käytä seuraavaa tunnusta:

- Tsoha Testerman: tsoha_jimbbkg_testerman@tfbnw.net - hassupassu

Sovelluksen pitäisi olla helppokäytöinen ja intuitiivinen.

10. Testaus, bugit ja puutteet & jatkokehitys

Sovelluksen testaus on toteutettu käsin. Jokainen ominaisuus on testattu samalla, kun sitä on kehitetty. Palvelimen yksinkertaisuuden vuoksi suurin osa testaamisesta on

rajoittunut frontend-puolelle, jossa tekemistä on riittänyt. Olen myös pyrkinyt testaamaan koko sovelluksen kaikkien toimintojen toimimisen, jos olen tehnyt suurempia muutoksia tai refaktorointeja.

Puutteiksi jäi kansioiden toteuttaminen, kuvien kommentointi (ei ollut varma alussa) ja tarkemmat kuvien tiedot (exif). Lisäksi itse kuvien käsittely on sivuutettu kokonaan, eikä edes thumbnailleja generoida. Kuvien lisääminenkin on aika tuskallista.

Jatkossa kuville voisi antaa slugin, joka toimisi kuvan osoitteessa id:n sijaan. Myös joitain kontrollereita voisi optimoida omilla SQL-kyselyillään, kuten arvostelujen keskiarvon laskemisen.

11. Omat kokemukset

Helppoa oli itse koodaaminen, vaikeaa oli sopivan ajan löytäminen koodaamiselle ja uusien juttujen opetteleminen (AngularJS lähinnä).

Opin aika paljon AngularJS:stä ja kehityin modulaarisemman koodin kirjoittamisessa.

Kaiken kaikkiaan olen sitä mieltä, että käytettyyn työmäärään nähden lopputulos on todella näyttävä, kiitos modernien sovelluskehysten ja tuttujen työkalujen (Node.js).