Aineopintojen harjoitustyö: Tietokantasovellus
Twitter-kloonin dokumentaatio

Mika Viinamäki

25. huhtikuuta 2013

Sisältö

1 Johdanto

1.1 Järjestelmän tarkoitus

Järjestelmän tarkoitus on olla yksinkertainen kopio Twitterin tarjoamasta mikroblogista. Käytännössä käyttäjät voivat lähettää enintään 140 merkin pituisia viestejä omalle sivulleen ja halutessaan lisätä niihin "hashtageja"lisäämällä jonkin viestin sanan eteen `#`-merkin tai viittauksia muihin palvelun käyttäjiin lisäämällä käyttäjätunnuksen eteen `@`-merkin.

Käyttäjä pystyy myös seuraamaan muita käyttäjiä, jolloin niiden viestit näkyvät omalla sivulla. Myös viestit, joissa käyttäjää on viitattu, näkyvät samalla sivulla – tämä toiminnallisuus eroaa jonkin verran Twitterin vastaavasta.

1.2 Toimintaympäristö

Järjestelmä on toteutettu Pythonilla Linux-ympäristössä käyttäen apuna Flask-sovelluskehystä, ja vaatii siis toimiakseen Pythonia tukevan ympäristön. Tietokantanaan se pystyy käyttämään ainakin SQLite:a ja PostgreSQL:aa, todennäköisesti muutkin järjestelmän käyttämän SQLAlchemy-kirjaston tukemat tietokannat toimivat.

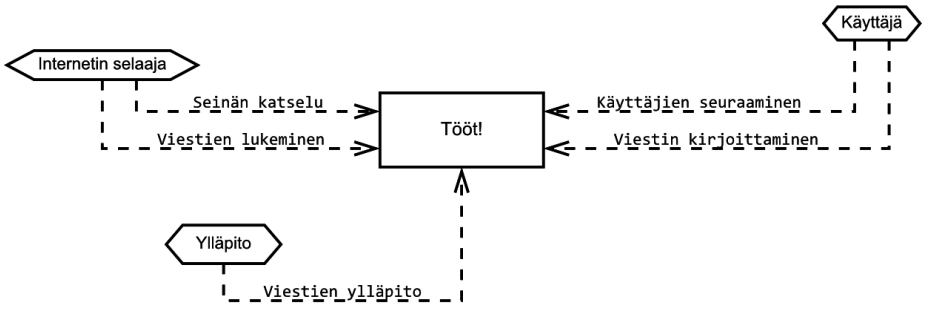
Järjestelmä on suunniteltu toimimaan Heroku-palvelussa suoraan. Testiasennus löytyy osoitteesta `http://tsoha-toot.herokuapp.com`. Testaamista varten voi rekisteröityä palveluun, sähköpostiosoitteiden ei tarvitse olla oikeita.

1.3 Rajaukset

Järjestelmään ei varmaankaan tulla toteuttamaan mitään ylläpitotoimintoja.

2 Yleiskuva järjestelmästä

2.1 Sidosryhmäkaavio



2.2 Käyttäjäryhmät

Internetin selaaja Satunnainen, sivulle jostain syystä eksynyt internetin käyttäjä, joka on kiinnostunut käyttäjien kirjoittamista viesteistä.

Käyttäjä Palveluun rekisteröitynyt käyttäjä, joka haluaa lähettää palveluun viestejään.

Ylläpito Palvelun toiminnasta huolehtivat henkilöt.

3 Käyttötapaukset

3.1 Internetin selaaja

3.1.1 Käyttäjän viestien katseleminen

Kuka tahansa voi katsella jonkun tietyn käyttäjän lähettämiä viestejä.

3.1.2 Käyttäjän seinän katseleminen

Kuka tahansa pystyy katselemaan kenen tahansa käyttäjän seinää, jossa näkyy käyttäjän seuraamien käyttäjien lähettämät viestit ja viestit, jossa kyseinen käyttäjä on mainittu.

3.1.3 Tietyn hashtagin sisältävien viestien katseleminen

Kuka tahansa pystyy katselemaan viestejä, joissa jokin tietty hashtag on mainittu.

3.2 Käyttäjä

3.2.1 Viestin lähettäminen

Käyttäjä pystyy lähettämään maksimissaan 140-merkkisen viestin palveluun. Viestissä voi olla hashtageja, linkkejä tai viittauksia muihin palvelun käyttäjiin.

3.2.2 Käyttäjän seuraaminen ja seuraamisen poistaminen

Käyttäjä pystyy asettamaan jonkun toisen käyttäjän seuratuksi ja poistamaan seurauksen.

3.2.3 Oman viestin poistaminen

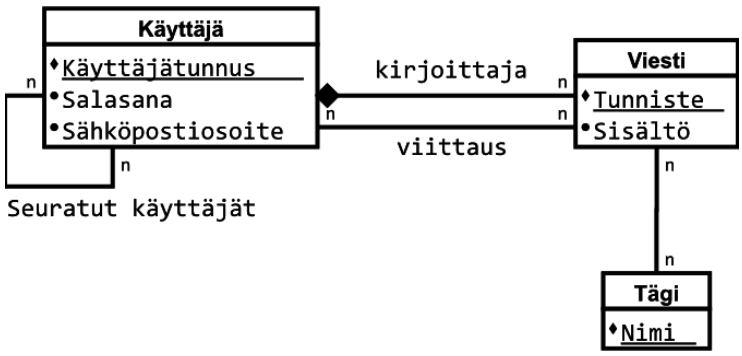
Käyttäjä voi poistaa lähettämänsä viestin palvelusta.

3.3 Ylläpitäjä

Ylläpitäjältä ei varmaankaan toteuteta mitään käyttötapauksia.

4 Järjestelmän tietosisältö

4.1 Tietosisältökaavio



4.2 Tietokohteiden kuvaukset

4.2.1 Käyttäjä

Palveluun rekisteröinyt käyttäjä. Yksi käyttäjä voi olla lähettänyt palveluun useita viestejä. Lisäksi käyttäjään voi liittyä useita muita käyttäjiä, joita hän seuraa.

Attribuutti Käyttäjätunnus

Arvojoukko Pienistä kirjaimista koostuva 6-20 merkkiä pitkä merkkijono.

Kuvailu Käyttäjän palvelussa käyttämä uniikki käyttäjätunnus.

Attribuutti Salasana

Arvojoukko Salasanasta muodostettu bcrypt-hash.

Kuvailu Käyttäjän kirjautumiseen käyttämä salasana.

Attribuutti Sähköpostiosoite

Arvojoukko Validi sähköpostiosoite merkkijonona.

Kuvailu Käyttäjän sähköpostiosoite.

4.2.2 Viesti

Käyttäjän palveluun lähettämä viesti. Viesti kuuluu aina jollekin käyttäjälle. Lisäksi viestiin voi liittyä erilaisia tägejä. Lisäksi viestissä voi olla viittaus useaan käyttäjään. Tägit ja viittaukset voidaan päätellä viestin sisällöstä - ne on

merkitty sisältöön laittamalla sanan eteen tägin tapauksessa `#`-merkin ja viittauksen tapauksessa `@`-merkin.

Attribuutti Tunniste

Arvojoukko Uniikki numero.

Kuvailu Viestin numeromuotoinen yksilöivä tunniste.

Attribuutti Sisältö

Arvojoukko Merkkijono, enintään 140 merkkiä.

Kuvailu Viestin sisältö.

4.2.3 Tägi

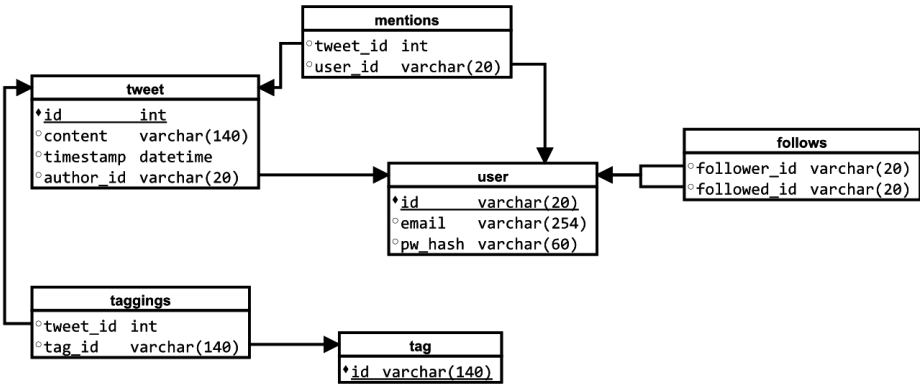
Viestiin liittyvä tägi. Sama tägi voi liittyä useaan viestiin, ja yhteen viestiin voi liittyä useita tageja.

Attribuutti Nimi

Arvojoukko Uniikki, pienistä kirjaimista koostuva, enintään 140 merkkiä pitkä merkkijono.

Kuvailu Tägin nimi.

4.3 Relaatiotietokantakaavio



Ohjelma käyttää SQLAlchemy-kirjaston deklaratiivista tietokannan määrittelyä, jonka avulla pystytään generoimaan SQL-lauseet monille eri tietokantaohjelmistoille – siispä SQL-lauseita ei ole dokumentissa mukana tai liitteenä.

5 Järjestelmän yleisrakenne

5.1 Projektin tiedostot

`docs/` Kansio, jossa tähän dokumenttiin liittyvät tiedostot.

`docs/docs.pdf` Tämä dokumentti.

`app/` Kansio, joka sisältää kaiken ohjelmakoodin.

`app/initialize_db.py` Python-skripti, jolla saa luotua tietokannan.

`app/static/` Kansio, joka sisältää kaikki web-sovelluksen staattiset tiedostot - esimerkiksi CSS-tyylitiedostot.

`app/static/css/bootstrap.min.css` Bootstrap-kirjasto, jota käytetään sivun ulkoasun pohjana.

`app/static/css/own.css` Omat CSS-tyylit.

`app/templates/` Kansio, jossa ovat HTML-koodin generoinnissa käytettävät Jinja2-templatet.

`app/main.py` Sovelluksen "päätiedosto". Sisältää mm. SQL- ja muut asetukset sekä kaikki näkymät.

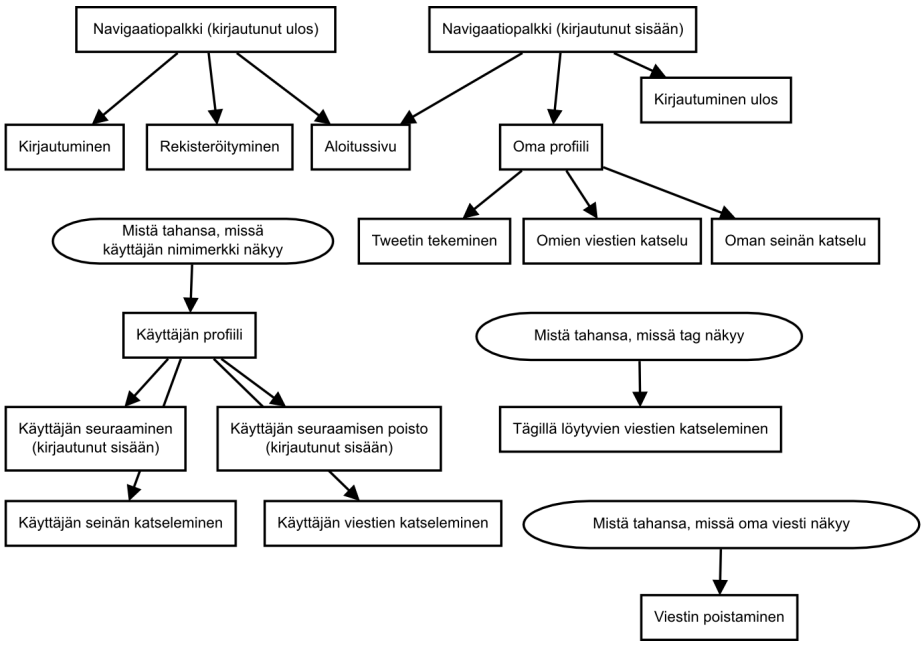
`app/models.py` Tietokantamallit sisältävä tiedosto.

`app/forms.py` Lomakkeet sisältävä tiedosto.

`Procfile` Sovelluksen ajamiseen tarvittavat tiedot Herokua varten.

`requirements.txt` Sovelluksen tarvitsemat kirjastot.

5.2 Käyttöliittymäkaavio



6 Asennustiedot

Ohje olettaa, että käytössä on Linux, Python 2.7 ja virtualenv.

6.1 Paikallisen version ajaminen

Ensin kloonataan git-repositorio GitHubista paikalliselle koneelle ja siirrytään kyseiseen repositorioon:

`$ git clone git://github.com/Kauhsa/tsoha-toot.git` (tai vaikkapa oman forkin git-osoite)
`$ cd tsoha-toot`

Luodaan uusi virtualenv - ei pakollista, mutta riippuvuuksien hoitaminen on muuten vaikeaa ainakin ilman root-oikeuksia.

```
$ mkvirtualenv --python=python2 toot
```

Asennetaan ohjelmiston tarvitsemat riippuvuudet.

```
$ pip install -r requirements.txt
```

Luodaan tietokanta, ohjelmisto käyttää oletuksena SQLiteä jos muuta ei määritellä:

```
$ python app/initialize_db.py
```

Ajetaan kehityspalvelin:

```
$ python app/main.py
```

Nyt palvelimen pitäisi olla päällä, ja sovellukseen pääsee käsiksi selaimella käyttämällä osoitetta `http://127.0.0.1:5000/`.

6.2 Herokuun siirtäminen

Oletetaan, että Heroku-tunnus on luotu, siihen on kirjauduttu ja Heroku Toolbelt on asennettu.

Luodaan uusi Heroku-applikaatio.

```
$ heroku create
```

Lisätään projektiin tietokanta:

```
$ heroku addons:add heroku-postgresql:dev
```

Komento antaa jonkun ympäristömuuttujan nimen, jossa on PostgreSQL-yhteyden tiedot, esim. `HEROKU_POSTGRESQL_PURPLE_URL`. Asetetaan se pää-tietokannaksi seuraavasti:

```
$ heroku pg:promote HEROKU_POSTGRESQL_PURPLE_URL
```

Viedään projekti Herokuun:

```
$ git push heroku master
```

Luodaan tietokanta palvelimelle:

```
$ heroku run python2 app/initialize_db.py
```

Avataan ohjelma:

```
$ heroku open
```

Web-sovelluksen pitäisi avautua selaimessa.