Leffakanta

Kristian Toro / 014292040

Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Aineopintojen harjoitustyö: Tietokantasovellus
23.3.2014

Sisällysluettelo

1. Johdanto

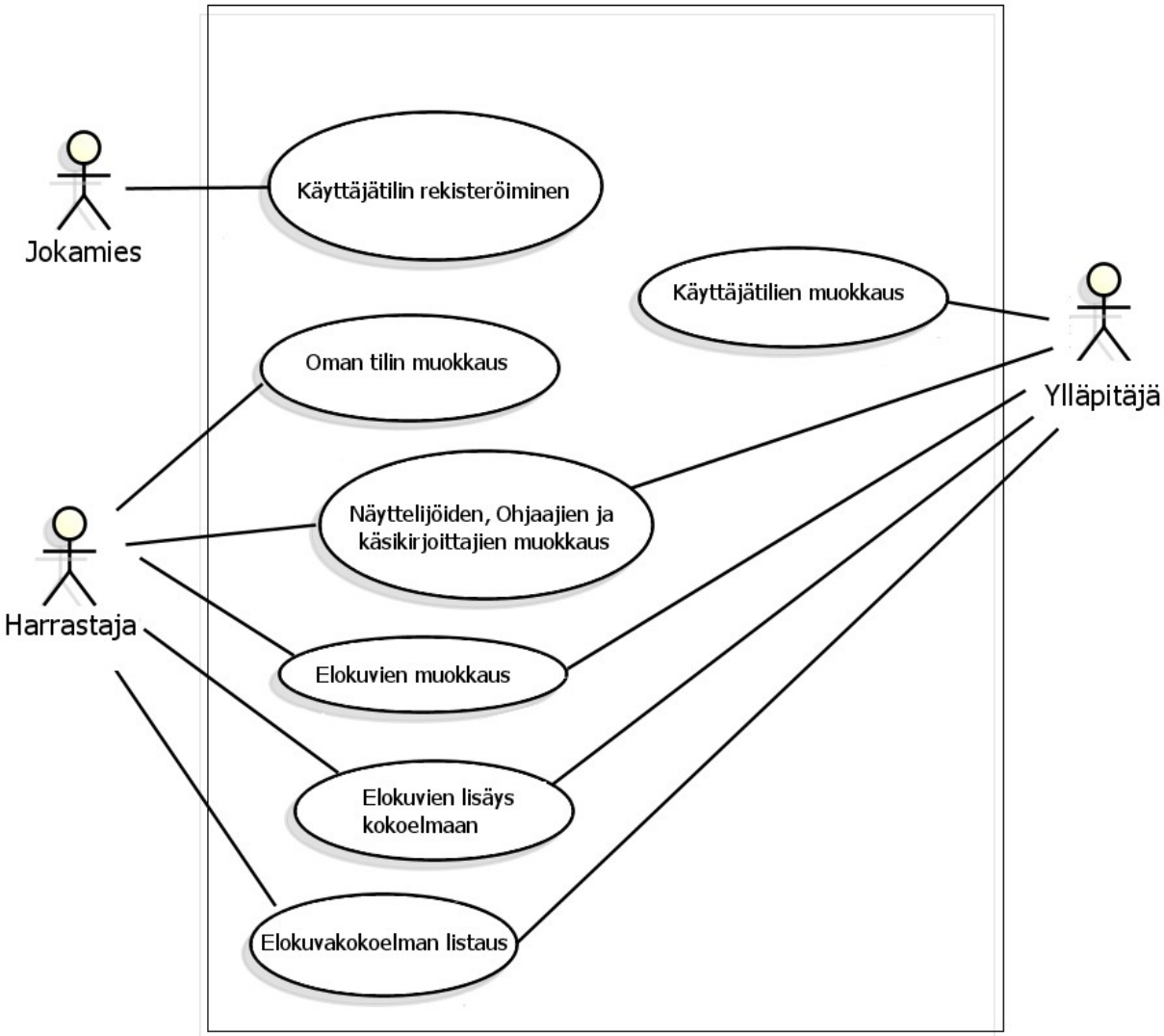
Työn tarkoituksena on luoda tietokanta elokuvaharrastajille, jossa käyttäjä voi lisätä oman elokuvakokoelmansa tiedot järjestelmään ja luoda siten helposti hallittavan listan elokuvista. Järjestelmän yksi päämäärä on helpottaa etenkin isojen kokoelmien järjestelyä.

Työn ohjelmointi toteutetaan Javalla käyttäen apuna Spring -frameworkkia. Tietokanta hoidetaan PostgreSQL -tietokannalla ja molempia, sekä ohjelmaa että tietokantaa ajetaan helsingin yliopiston users -palvelimelta.

Työhön toteutettava web-käyttöliittymä käyttää apuna CSS-tyylisivuja ja Javascriptiä paremman käyttökokemuksen saavuttamiseksi.

2. Yleiskuvaus

Käyttötapauskaavio



Käyttäjäryhmät

Jokamies

Jokamiehellä tarkoitetaan ketä tahansa, joka kykenee käyttämään leffakantaa internetin välityksellä.

Harrastaja

Harrastaja on aina rekisteröitynyt käyttäjä jonka elokuvakokelmia leffakannalla järjestellään.

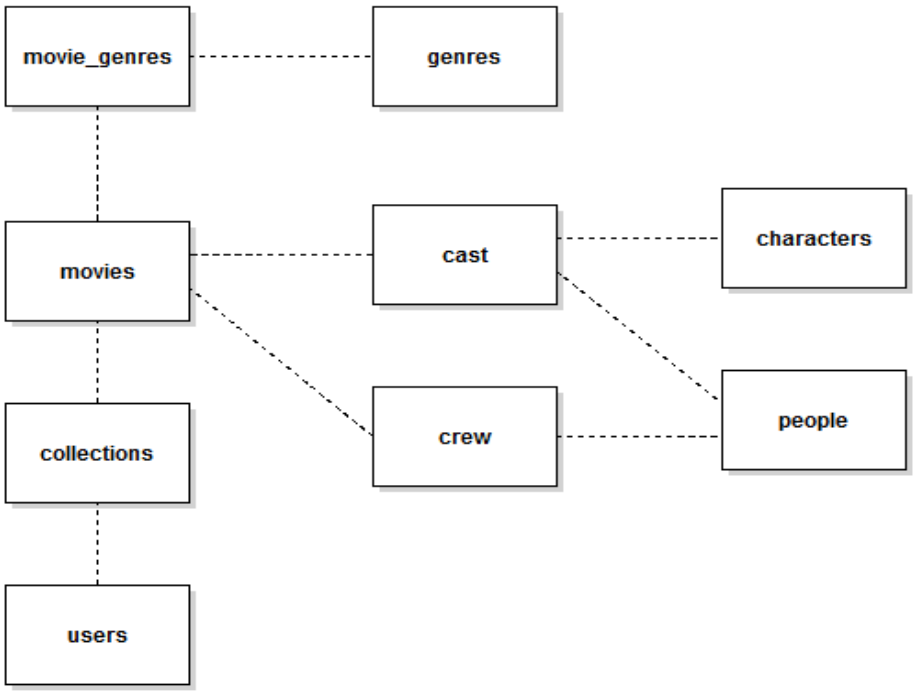
Ylläpitäjä

Ylläpitäjällä tarkoitetaan sellaista harrastajaa jolla on lisäksi oikeus hallita käyttäjätilejä.

3. Käyttötapaukset

Täytetään myöhemmin..

4. Järjestelmän tietosisältö



Käsitekaavio

Tietokohde: movies

Attribuutti	Arvojoukko	Kuvailu
Title	Merkkijono	Elokuvan nimi
Year	Desimaaliluku	Elokuvan julkaisuvuosi
Runtime	Desimaaliluku	Elokuvan kesto minuuteissa
Rating	Desimaaliluku	Elokuvan keskimääräinen arvosana
Plot	Merkkijono	Elokuvan juoni tiivistettynä
Poster	Merkkijono	Linkki elokuvajulisteen kuvaan
Background	Merkkijono	Linkki elokuvan fan-art taustakuvaan
Trailer	Merkkijono	Linkki elukuvan youtube-traileriin

Tietokohde: users

Attribuutti	Arvojoukko	Kuvailu
Username	Merkkijono	Käyttäjän tunnus sovelluksessa
Password Hash	Merkkijono	Salasanan hash-arvo salasanan tunnistamiseksi
Password Salt	Merkkijono	Salasanan suolausarvo edellisen hashin kanssa käytettäväksi
Is Admin	Totuusarvo	Ilmaisee onko kyseinen käyttäjä ylläpitäjä

Tietokohde: collections

Attribuutti	Arvojoukko	Kuvailu
Owner	Desimaaliluku	Viiteavain usersin käyttäjään
Movie	Desimaaliluku	Viiteavain moviesin elokuvaan
Format Type	Enumeroitu arvo	Ilmaisee kyseisen elokuvakopion formaatin
Availibility	Enumeroitu arvo	Ilmaisee onko teos saatavilla, lainassa vai kadonnut

Tietokohde: genres

Attribuutti	Arvojoukko	Kuvailu
Genre Name	Merkkijono	Elokuvagenren määrittelynimi

Tietokohde: movies_genres

Attribuutti	Arvojoukko	Kuvailu
Movie	Desimaaliluku	Viiteavain moviesin elokuvaan
Genre	Desimaaliluku	Viiteavain genresin elokuvaluokitteluun

Tietokohde: people

Attribuutti	Arvojoukko	Kuvailu
Person Name	Merkkijono	Henkilön (näyttelijä, ohjaaja jne) koko nimi
Image	Merkkijono	Linkki henkilön kuvaan

Tietokohde: characters

Attribuutti	Arvojoukko	Kuvailu
Character Name	Merkkijono	Elokuvan roolihahmon nimi

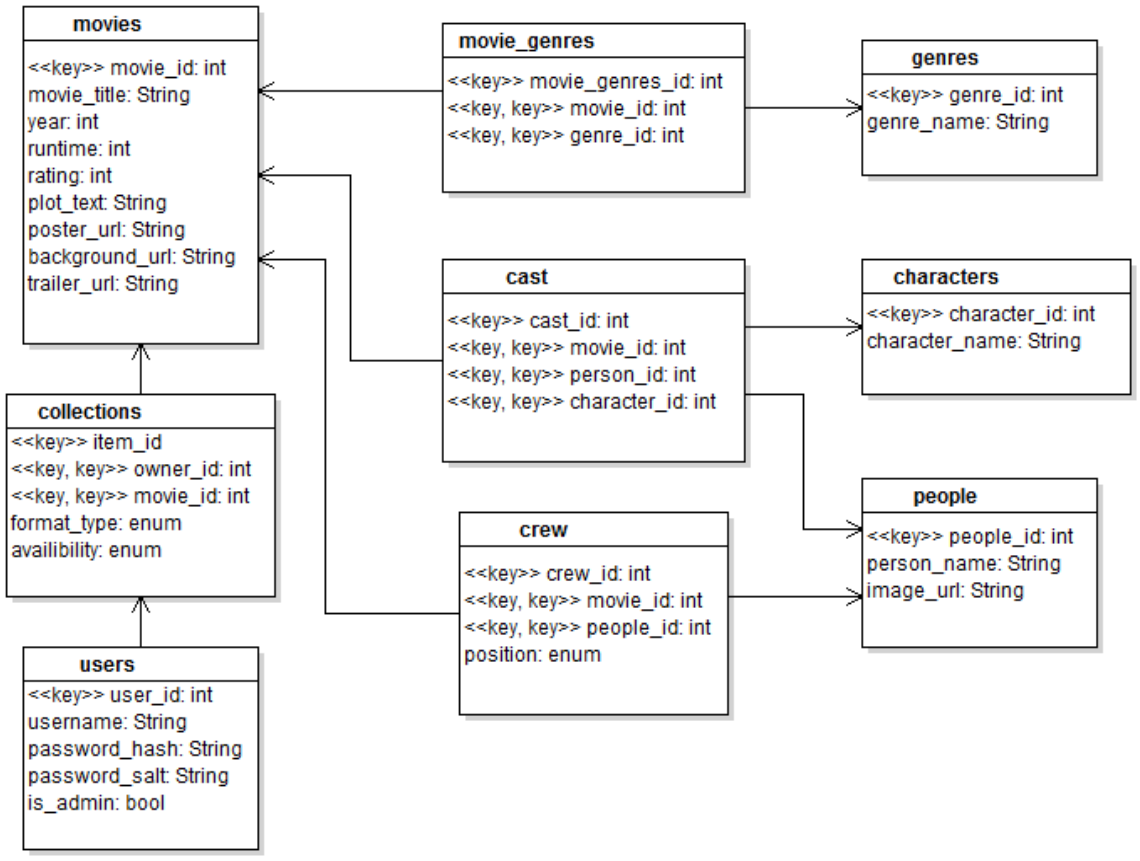
Tietokohde: cast

Attribuutti	Arvojoukko	Kuvailu
Movie	Desimaaliluku	Viiteavain moviesin elokuvaan
Person	Desimaaliluku	Viiteavain people:n henkilöön
Character	Desimaaliluku	Viiteavain charactersin roolihahmoon

Tietokohde: crew

Attribuutti	Arvojoukko	Kuvailu
Movie	Desimaaliluku	Viiteavain moviesin elokuvaan
People	Desimaaliluku	Viiteavain people:n henkilöön
Position	Enumeroitu arvo	Määrittelee onko henkilö elokuvan tuottaja, ohjaaja vaiko käsikirjoittaja

5. Relaatiotietokantakaavio



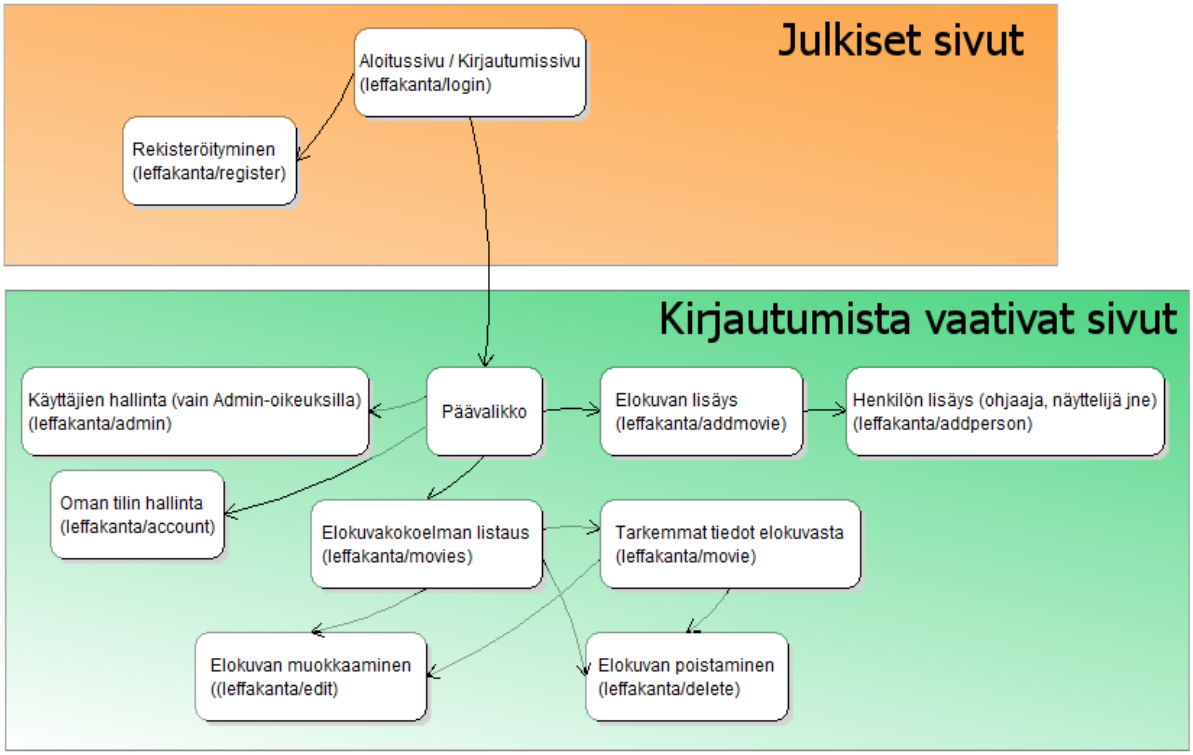
Tietokannan relaatiot näkyvät kaaviosta. Turhat indeksointiavaimet poistunevat vielä myöhemmin.

6. Järjestelmän yleisrakenne

Sovulluksessa noudatetaan tarkkaa MVC-mallia. Koska työ hyödyntää raskaasti Spring -Framework:ia, sovelluksen koodin osuus on saatu pidettyä tiiviinä ja johdonmukaisena ja työ siten hyvin hallittavissa. MVC-mallin malli (Model), näkymä (View) ja kontrolleri (Controller) toteutuvat työssä täysimääräisinä.

Työn malli hoitaa ainoastaan tiedon noutamista tietokannasta. Näkymän taas generoi käyttöliittymän ajonaikaisesti modulaarisesti rakennetuista jsp-tiedostoista, mikä myös helpottaa ylläpidettävyyttä. Html:ään viittaavia asioita ei hoideta näkymän ulkopuolella, poislukien pakolliset url-viittaukset itse tietokannassa kuvien ja elokuvatrailerin web-osoitteille. Näkymän ulkoasu myös tyylitellään CSS-tyylisivun sääntöjen avulla, joten elementtien sijoittelu on hyvin suoraviivaista. Kontrolleri lopuksi niputtaa näkymän saamat syötteet ja hoitaa näkymän ja mallin välistä tiedonsiirtoa, pitäen huolta, että sovelluksen kukin osa hoitaa vain sen, mikä sille kuuluu.

7. Käyttöliittymä ja järjestelmän komponentit



8. Asennustiedot

Ennen kuin sovellus voidaan asentaa, se tulee ensin kääntää. Projektikansiossa on tarvittavat tiedostot projektin kääntämiseksi maven-ympäristössä. Kääntäminen onnistuu kätevästi esim netbeansilla kunhan siihen on asennettuna ajantasainen Maven-laajennus. Käännettyä ja pakattua war -pakettia voi ajaa esim paikallisesti Jettyn kautta (Jetty haetaan automaattisesti mavenin debug-buildissa) tai käyttämällä jotain web-palvelinta, esim Apachen Tomcattia.

Tomcatille paketin asennus onnistuu siirtämällä leffakanta.war -tiedosto tomcatin webappsin alle, kun tomcat on saanut paketin purettua, tulee vielä konfiguroida tietokannan asetukset tiedostoon */tomcat/webapps/leffakanta/WEB-INF/classes/jdbc.properties*. Tiedostosta tulee muuttaa *jdbc.username* ja *jdbc.password* vastaamaan PostgreSQL:n käyttäjänimeä ja salasanaa. Tämän jälkeen Tomcat tulee käynnistää uudelleen ja sovellusta voi alkaa käyttämään.

9. Käynnistys- / käyttöohje

Sovellus käynnistetään osoitteesta http://t-kristiat.users.cs.helsinki.fi/leffakanta. Sovellusta voi käyttää joko luomalla omat tunnukset (linkki rekisteröintiin löytyy aloitussivulta) tai käyttämällä valmiita tunnuksia:

(tavallinen käyttäjä)
tunnus: ohdake
salasana: test

(järjestelmänvalvoja)
tunnus: mikkelinmies
salasana: test

10. Tunnetut bugit ja puutteet

Lomakkeiden sisällön oikeellisuuden tarkistaminen ja suodattaminen on vielä puutteellista. Syötteiden tarkistamista ja virheilmoituksia tullaan paikkaamaan projektin edetessä.

11. Jatkokehitysideat

Sovelluksessa käytettävät kuvat pitäisi oikeasti tallentaa aina omalle palvelimelle nykyisten suorien linkkien sijaan. Samaten kuville voisi luoda mahdollisuuden ladata ne sovellukseen myös paikallisesta tiedostosta nykyisten linkkien sijaan.

Olisi myös hyödyllistä, mikäli sovellus osaisi hakea elokuvien tiedot (mukaanlukien tuotantohenkilöt) automaattisesti jostain isommasta elokuvatietokannasta pelkän elokuvan nimen ja vuoden perusteella, esim avoimesta www.themoviedb.org:ista.

12. Omat kokemukset

Työ oli kolmas samantyyppinen työ jonka olen koulutyönä tehnyt joten MVC-rakenne ja tietokantojen käyttö web-sovelluksessa oli jo entuudestaan minulle tuttua. Tutustuin työssä ensimmäistä kertaa kuitenkin PostgreSQL:ään ja koska se käsitteli jotain SQL-pyyntöjä hieman eri syntaksilla kuin aiemmin käyttämäni MySQL, opin PostgreSQL:n käytöstä hieman uutta. Koitin työssä myös syventää tietämystäni spring frameworkista, ja opinkin jonkin verran uutta tähän liittyen, mm syvensin springin xml -konfigurointitietämystäni sekä opin lisää springin kirjastoista. Vaikka harjoitustyön pääasiat olivat minulle ennestään tuttua, kurssi auttoi minua syventämään osaamistani edellämainituilla sektoreilla. Haastavinta työssä oli tämän dokumentaation tekeminen ajallaan ja joidenkin springin toteutustapojen selvittäminen. Muut ohjelmoimintiin liittyvät asiat, sekä tietokannan käsittely oli sen sijaan aika suoraviivaista.

13. Liitteet

???