Leffakanta

Kristian Toro / 014292040

Helsingin yliopisto
Tietojenkäsittelytieteen laitos
Aineopintojen harjoitustyö: Tietokantasovellus
27.4.2014

Sisällysluettelo

1. Johdanto

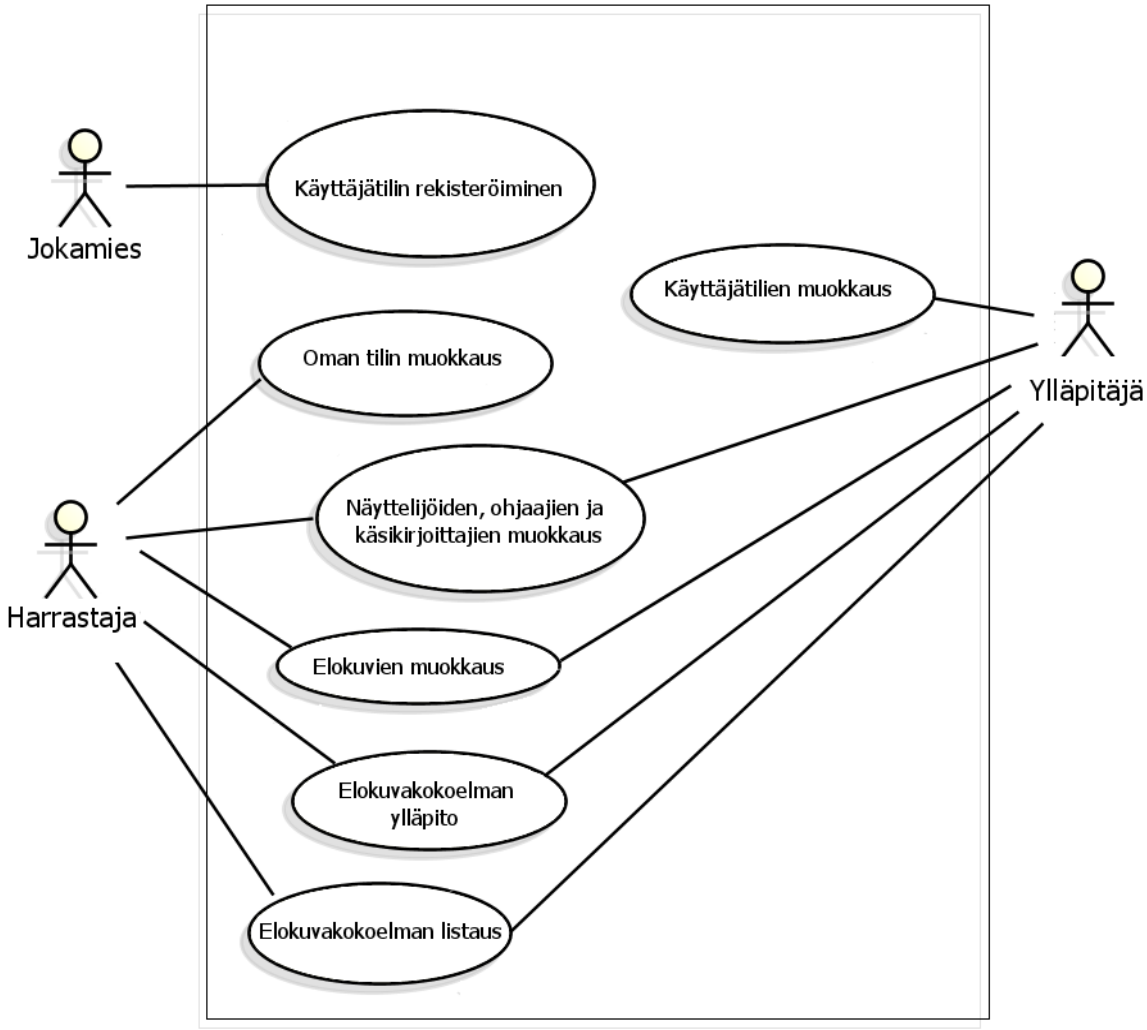
Työn tarkoituksena on luoda tietokanta elokuvaharrastajille, jossa käyttäjä voi lisätä oman elokuvakokoelmansa tiedot järjestelmään ja luoda siten helposti hallittavan listan elokuvista. Järjestelmän yksi päämäärä on helpottaa etenkin isojen kokoelmien järjestelyä.

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

Työhön toteutettava web-käyttöliittymä käyttää apuna CSS-tyylisivuja, jotta jatkossa sivuston ulkoasua on helppo parantaa ja muokata.

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

Harrastajan käyttötapaukset

Näytteiljöiden, ohjaajien ja käsikirjoittajien muokkaus:

Harrastaja voi muokata elokuviin liitettyjen henkilöiden tietoja kuten näyttelijän nimeä, syntymäaikaa tai linkittää näyttelijästä kasvokuvan.

Elokuvien muokkaus:

Harrastaja voi muokata elokuvien perustietoja, esim nimeä, vuotta ja juonta. Lisäksi harrastaja voi liittää elokuvaan elokuvan henkilöstöä, kuten näyttelijöitä ja ohjaajia jne.

Elokuvakokoelman ylläpito:

Harrastaja voi lisätä omaan kokoelmaansa elokuvia ja poistaa niitä sieltä. Harrastaja voi myös merkitä onko kyseinen kappale elokuvasta esimerkiksi DVD vai Blu-ray ja onko elokuva tällä hetkellä saatavilla vai onko se vaikkapa lainassa.

Elokuvakokoelman listaus:

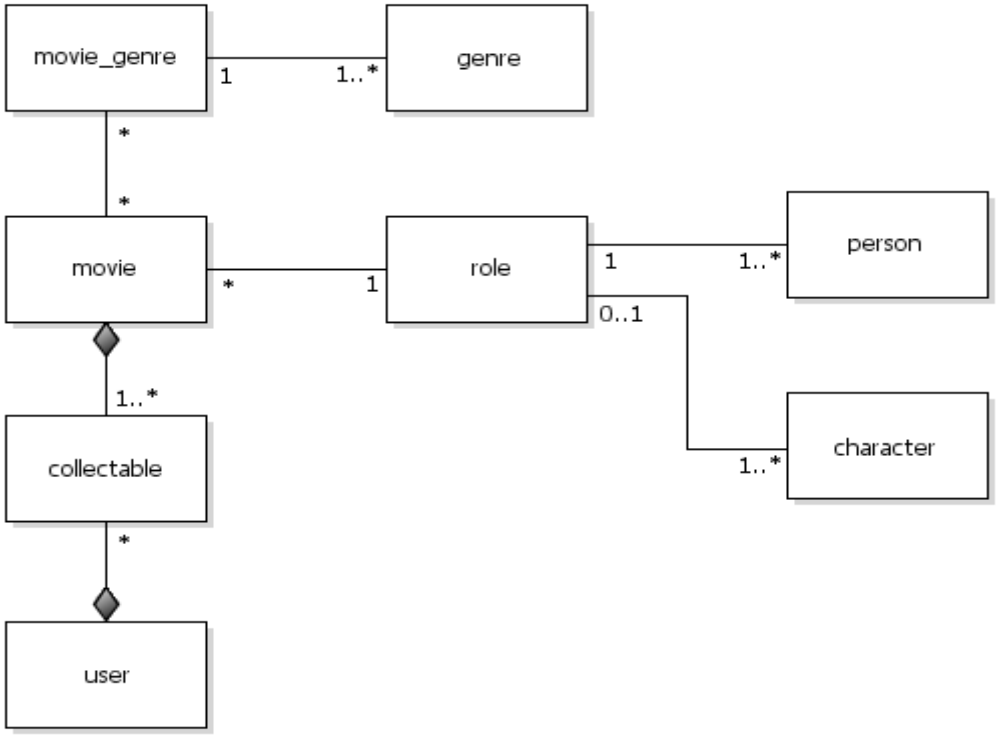
Harrastaja voi listata koko elokuvakokoelmansa sisällön ja valita listalta yksittäisiä elokuvia lähempään tarkasteluun.

Ylläpitäjän käyttötapaukset

Käyttäjätilien muokkaus:

Ylläpitäjä voi tarpeen tullen muokata kaikkia sovellukseen liitettyjä käyttäjätilejä. Ylläpitäjän oikeuksiin kuuluvat käyttäjätunnusten uudelleennimeäminen, uuden salasanan luominen, käyttäjäoikeuksien korottaminen ylläpitäjäksi sekä käyttäjän poistaminen sovelluksesta.

4. Järjestelmän tietosisältö



Käsitekaavio

Tietokohde: movie

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

Movie pitää sisällään yksittäisen elokuvan tiedot.

Tietokohde: user

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

User on yksittäinen sovelluksen käyttäjä.

Tietokohde: collectable

Attribuutti	Arvojoukko	Kuvailu
User	Desimaaliluku	Viiteavain userin käyttäjään
Movie	Desimaaliluku	Viiteavain movien elokuvaan
Format Type	Enumeroitu arvo	Ilmaisee kyseisen elokuvakopion formaatin

Collectable on käyttäjän kokoelman yksi elokuva. Käyttäjällä voi myös olla monta kappaletta samasta elokuvasta, esimerkiksi DVD ja Blu-ray.

Tietokohde: genre

Attribuutti	Arvojoukko	Kuvailu
Genre Name	Merkkijono	Elokuvagenren määrittelynimi

Genre eli elokuvan tyylilaji.

Tietokohde: movie_genre

Attribuutti	Arvojoukko	Kuvailu
Movie	Desimaaliluku	Viiteavain movien elokuvaan
Genre	Desimaaliluku	Viiteavain genren elokuvaluokitteluun

Movie_genre yhdistää tyylilajit elokuvaan. Elokuvalla voi olla monta tyylilajia.

Tietokohde: person

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

Person on elokuvaan liittyvä henkilö, joko näyttelijä, ohjaaja tai käsikirjoittaja.

Tietokohde: character

Attribuutti	Arvojoukko	Kuvailu
Character Name	Merkkijono	Elokuvan roolihahmon nimi

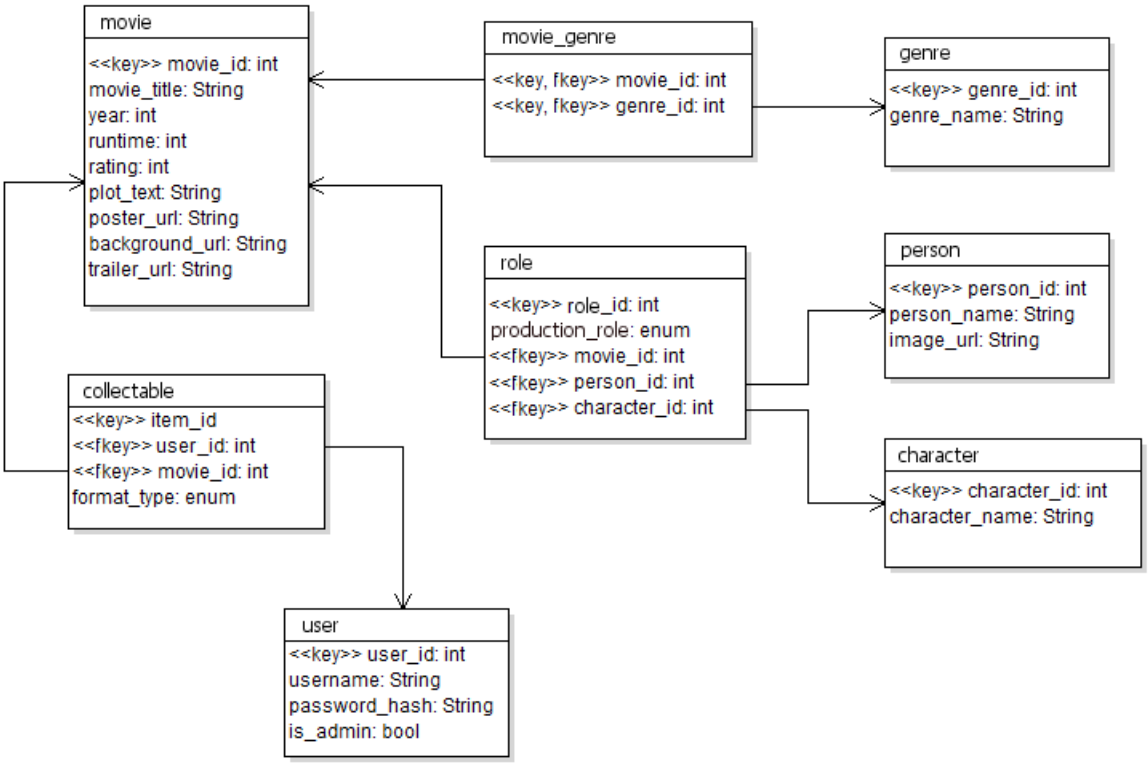
Character on näyttelijän esittämä roolihahmo elokuvassa.

Tietokohde: role

Attribuutti	Arvojoukko	Kuvailu
Production Role	Enumeroitu arvo	Määrittelee onko henkilö näyttelijä, ohjaaja vai käsikirjoittaja
Movie	Desimaaliluku	Viiteavain movien elokuvaan
Person	Desimaaliluku	Viiteavain personin henkilöön
Character	Desimaaliluku	Viiteavain characterin roolihahmoon

Role yhdistää elokuvan ja siihen liittyvät tuotantohenkilöt. Henkilö voi esiintyä elokuvan monessa eri tuotantoroolissa, esimerkiksi yksi henkilö voi olla elokuvan sekä ohjaaja, käsikirjoittaja, että näyttelijä.

5. Relaatiotietokantakaavio

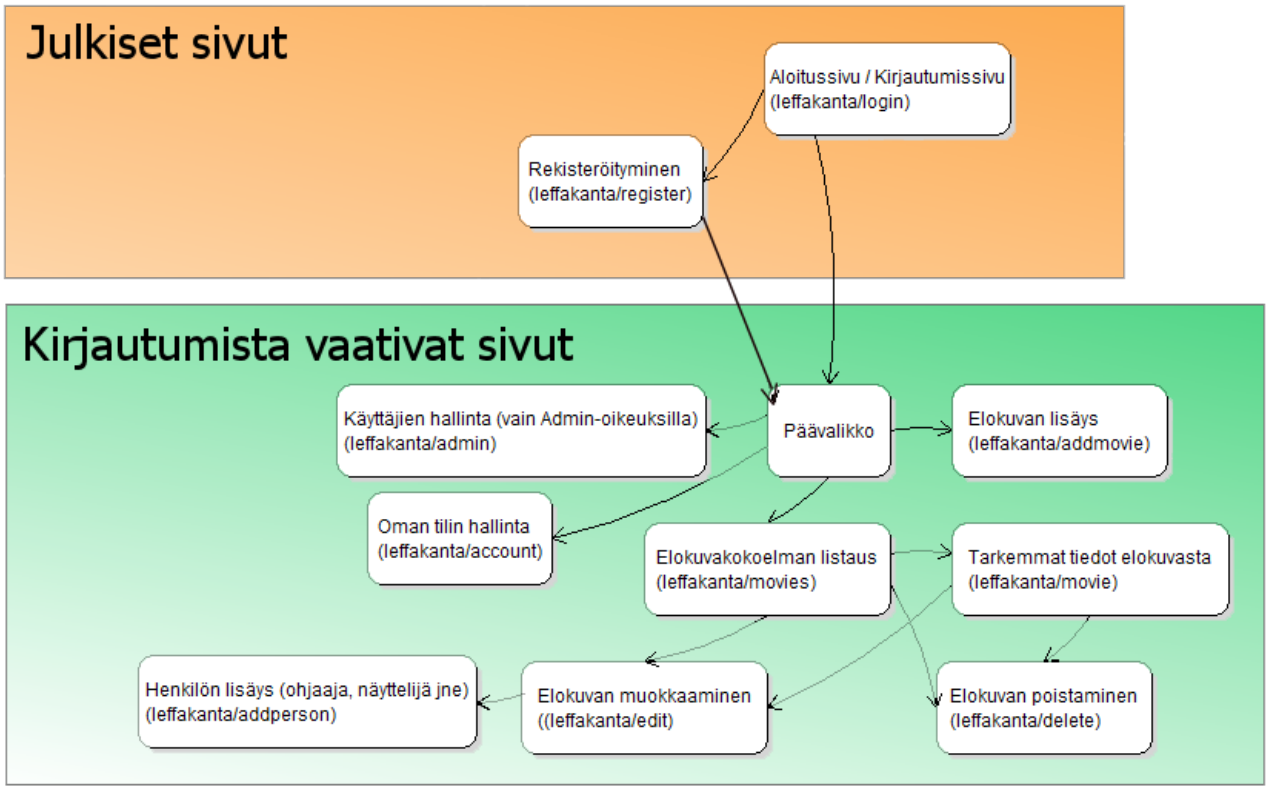


6. Järjestelmän yleisrakenne

Sovulluksessa noudatetaan MVC-mallia. Koska työ hyödyntää raskaasti Spring Frameworkia, sovelluksen koodin osuus on saatu pidettyä tiiviinä ja työ siten hyvin hallittavina.

Työn mallit hoitavat tiedon noutamista tietokannasta ja salasanojen hashien luomisen. Näkymä 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



Sovelluken käyttäminen on suoraviivaista. Mikäli käyttäjällä on järjestelmänvalvojan oikeudet, näkee hän lisää hallintatyökaluja, muuten järjestelmänvalvoja toimii kuten tavallinen käyttäjä.

Sovellus tunnistaa katselulaitteen ja käyttää erillistä muotoilua mikäli käyttäjä käyttää sovellusta puhelimella tai työpöydältä käsin. Sovelluksen toiminnallisuus säilyy laitteesta riippumatta samana.

8. Asennustiedot

Ennen kuin sovellus voidaan asentaa, se tulee 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 webapps-kansioon. Kun tomcat on saanut paketin purettua, tulee vielä konfiguroida tietokannan asetukset tiedostossa */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

Demosovellus voidaan käynnistää 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

Sovelluksella ei ole tunnettuja bugeja, mutta puutteiden osalta katso kohta 11. Jatkokehitysideat.

11. Jatkokehitysideat

Sovelluksessa käytettävät kuvat pitäisi 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 ja henkilöiden tiedot automaattisesti jostain isommasta elokuvatietokannasta pelkän elokuvan nimen ja vuosiluvun perusteella, esim. avoimesta www.themoviedb.org:ista.

Elokuvien ja henkilöiden syöttäminen voisi olla helpompaa jos kannasta haettaisiin kirjoitushetkellä ehdotuksia syötteille, tällöin käyttäjän ei tarvitsisi kirjoittaa koko nimeä jos kyseinen nimi on jo olemassa. Tämän voisi toteuttaa esim. jqueryn autocomplete-skriptillä.

12. Omat kokemukset

Tämä harjoitustyö oli kolmas web-pohjainen tietokantasovellus, jonka olen lähivuosina tehnyt koulutyönä, joten MVC-rakenne sekä tietokantojen käyttö web-sovelluksessa oli jo entuudestaan minulle tuttua. Tutustuin työssä ensimmäistä kertaa kuitenkin PostgreSQL:ään. PostgreSQL käsitteli joitain SQL-pyyntöjä hieman eri syntaksilla kuin

aiemmin käyttämäni MySQL, joten opin PostgreSQL:n käytöstä uusia asioita.
Pyrin työssäni myös syventämään tietämystäni Spring Frameworkista ja opinkin jonkin verran uutta tähän liittyen. Mm. syvensin Springin xml -konfigurointitietämystäni, sekä opin lisää monista Springin kirjastoista.

Haastavinta työssä oli tämän dokumentaation tekeminen ajallaan ja joidenkin Springiin liittyvien toteutustapojen selvittäminen, jätin myös joidenkin toiminteiden toteuttamisen turhan myöhäiseen vaiheeseen ja sen puolesta tulikin kiire. Tämän pohjalta voidaan päätellä, että minun tulisi jatkossa panostaa enemmän aikataulujen laatimiseen, jotta vastaavilta kiireiltä vältyttäisiin jatkossa. Ohjelmointiin liittyvät asiat sekä tietokannan käsittely oli muutamaa poikkeustapausta lukuunottamatta hyvin suoraviivaista.