TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Tutkintotyö

Mika Haulo

DVD-KIRJASTOJÄRJESTELMÄ RUBY ON RAILS -TOTEUTUKSELLA

Työn ohjaaja Jari Mikkolainen
Työn teettäjä Ionific Oy, valvojana CTO Otto Chrons
Tampere 2006

TAMPEREEN AMMATTIKORKEAKOULU
Ohjelmistotekniikka
Tuotesuunnittelu
Haulo, Mika
Dvd-kirjastojärjestelmä Ruby on Rails -toteutuksella
Tutkintotyö 42 sivua + 2 liitesivua
Työn ohjaaja Jari Mikkolainen
Työn teettäjä Ionific Oy, valvojana CTO Otto Chrons
Helmikuu 2006
Hakusanat web-sovellus, kirjastojärjestelmä, äänestys, Ruby on Rails

TIIVISTELMÄ

Tässä työssä esiteltävä dvd-kirjastojärjestelmä on Ionific Oy:n sisäiseen käyttöön tarkoitettu dvd-kirjaston www-pohjainen hallintasovellus. Sovelluksen käyttöympäristönä on Linux-pohjainen palvelin, ja sitä käytetään web-käyttöliittymän kautta.

Sovelluksen avulla pyritään helpottamaan yrityksen dvd-valikoiman hallintaa ja seurantaa sekä uusien elokuvien äänestystä. Aikaisemmin käytössä ei ole ollut mitään yhtenäistä järjestelmää, mutta elokuva- ja työntekijämäärän lisääntyessä dvd-kirjastojärjestelmän tarve on noussut oleelliseksi asiaksi.

Dvd-kirjastojärjestelmä on toteutettu Ruby on Rails -järjestelmällä, joka on melko uusi kehitysympäristö. Ruby on Rails sopii erityisesti web-sovellusten tekemiseen, joten se oli mielenkiintoinen vaihtoehto toteutustavaksi. Työn toinen tarkoitus onkin tutustua tähän järjestelmään käytännön hankkeen kautta.

Syksyn 2005 aikana kirjastojärjestelmästä valmistunut testiversio on idealtaan toimiva, mutta toteutukseltaan liian hatara tuotantokäyttöön. Hataran toteutuksen syitä olivat kehitysympäristön erilaisuus muihin tekniikoihin verrattuna ja laajat ominaisuudet, joiden sisäistäminen vaati luultua enemmän aikaa ja opettelua.

Tuotantokäyttöön tulevan version kehitys alkaa vuoden 2006 alussa. Vaikka sovellus on tarkoitus kirjoittaa alusta asti uudestaan, on olemassa olevan koodin käyttäminen suurelta osin mahdollista. Koodissa on kuitenkin paljon sellaisia osia, jotka on syytä uudistaa entistä paremmin Rails-ohjelmistokehyksen ideologiaan sopivaksi. Lisäksi sovelluksen käyttöliittymän suunnittelu on tarkoitus siirtää käytettävyyteen ja käyttöliittymäsuunnitteluun erikoistuneille työntekijöille.

TAMPERE POLYTECHNIC
Software Engineering
Product developement
Haulo, Mika
Dvd-library software built on Ruby on Rails -technology
Engineering Thesis 42 pages, 2 appendices
Thesis Supervisor Jari Mikkolainen
Commissioning Company Ionific Oy, Supervisor: Otto Chrons (CTO)
February 2006
Keywords web application, library, voting, Ruby on Rails

ABSTRACT

Ionific Oy provides a possibility to borrow movies from the company's dvd collection as a benefit for all its employees. The collection currently consists of about 500 individual titles and new movies are purchased bi-monthly. The Dvd Library System, which is introduced in this study, is a web-based software designed for Ionific Oy's internal use. Its purpose is to make maintaining the company's movie collection easier. The software allows users to browse the entire movie collection, borrow and reserve available movies, and to vote for movies to be acquired. The Dvd Library System is built using Ruby on Rails technology, which is a new and interesting framework especially for developing web-based software. This study is also a brief research of using Ruby on Rails as a development environment for web applications. The first version of the Dvd Library System was written during fall 2005. The development of an improved version for production use will begin in February 2006.

Sisällysluettelo

Lyhenteet ja termit

CVS	Concurrent Versions System. Versionhallintaohjelmisto.
CSS	Cascading Style Sheets. Www-sivuilla käytettävät tyylimäärittelyt.
ER-kaavio, ER-malli	Tietokannan rakenteen kuvaava relaatiomalli (engl. entity relation model).
GNU	GNU's Not Unix. Richard Stallmanin vuonna 1984 aloittama vapaiden ohjelmistojen projekti.
GPL	GNU General Public Licence. Vapaiden ja avoimen lähdekoodin ohjelmistojen yleisesti käyttämä ohjelmistolisenssi.
Kolmas normaalimuoto (3NF)	Tietokannan normaalimuoto, joka sallii ei-triviaalit, toiminnalliset riippuvuudet vain avainattribuuteilta.
HTML, XHTML	(Extensible) Hypertext Markup Language. Www-sivujen toteutukseen käytettävä kuvauskieli.
Java-appletti	Www-sivun yhteydessä suoritettava pieni Java-kielinen sovellus.
JavaScript	Www-selaimessa tulkittava skriptikieli, joka lisää www-sivujen dynaamisuutta ja toiminnallisuutta. JavaScript ei ole sama asia kuin Java.
Malli	MVC-arkkitehtuurin osa, joka määrittelee ja kuvaa sovelluksen tietosisällön.
Mix-in-luokka	Olio-ohjelmoinnissa esiintyvä, abstraktia luokkaa muistuttava käsite, jonka avulla voidaan laajentaa olemassa olevien luokkien toiminnallisuutta.
Monisäikeisyys	Tekniikka, jossa prosessori suorittaa useaa eri tehtävää samanaikaisesti (engl. multithreading).
MVC-malli	Model-View-Controller. Ohjelmistoarkkitehtuuri, jossa sovelluksen toimintalogiikka, tietosisältö ja näkymä erotetaan erillisiksi kokonaisuuksiksi.
Näkymä	MVC-arkkitehtuurin osa, joka määrittelee sovelluksen käyttöliittymän.
Object-SQL mapping	Tekniikka, jolla yhdistetään relaatiotietokannan taulut oliopohjaisen sovelluksen luokkiin.
Ohjain	MVC-arkkitehtuurin osa, joka määrittelee sovelluksen toiminnallisen osuuden.
Ohjelmistokehys	Apujärjestelmä ohjelmistojen kehittämiseen (engl. framework).

Ohjelmointiparadigma	Tapa kuvata ohjelman rakennetta ja toimintaa.
Olio-ohjelmointi	Ohjelmistojen helppoon ylläpitoon ja koodin uudelleenkäytettävyyteen tähtäävä ohjelmointiparadigma.
PHP	PHP: Hypertext Preprocessor. Yleinen www-sivujen dynaamisen sisällön toteutukseen käytettävä skriptikieli.
Proseduraalinen ohjelmointi	Ohjelmointiparadigma, jossa ohjelma ositetaan yksinkertaisiin, peräkkäisillä komennoilla suoritettaviin kokonaisuuksiin.
Protokolla	Kommunikointitapa
SMTP	Sähköpostipalvelimien käyttämä protokolla.
SSH	Secure Shell. SSL-salattu yhteysprotokolla.
SSL	Secure Sockets Layer. Turvallinen tiedonsalausjärjestelmä.
SQL	Structured Query Language. Relaatiotietokannan kyselykieli.
Tietokannan normalisointi	Tietokannan rakenteen mallinnustapa, jolla pyritään tekemään tietojen tallentamisesta eheää ja tehokasta. Normalisoinnin taso on nimeltään normaalimuoto.
UNIX	Ken Thompsonin ja Dennis Richien 1960-luvun lopulla kehittämä käyttöjärjestelmä. Nykyään UNIX-nimitystä käytetään yleisnimenä kaikille UNIX-johdannaisille ja niiden kaltaisille käyttöjärjestelmille.
URL	Uniform Resource Locator. Internet-osoite.

1 JOHDANTO

Tämä tutkintotyö esittelee Ionific Oy:n sisäiseen käyttöön tarkoitetun dvd-kirjaston hallintaan käytettävän web-sovelluksen. Sovelluksen tarkoitus on helpottaa yrityksen dvd-kirjaston hallintaa. Elokuvia on tällä hetkellä noin 500 ja kokoelma kasvaa jatkuvasti työntekijöiden valitsemilla elokuvilla. Tarkoitus on, että dvd-kirjastojärjestelmän avulla saadaan selville kaikki olennainen tieto sekä yrityksen jo omistamista elokuvista että suunnitelluista hankinnoista.

Ionific Oy:ssä on ollut jo kauan elokuvakirjaston hallintasovelluksen tarve. Sekä työntekijämäärän että elokuvavalikoiman alati kasvaessa on keskitetyn tietolähteen tarve elokuvien lainauksista ja muista tiedoista tullut yhä tärkeämmäksi. Suunnitelmissa ollut hanke on vain aina jäänyt asiakasprojektien varjoon.

Dvd-kirjastojärjestelmän avulla työntekijät voivat merkitä elokuvan lainatuksi ja palautetuksi, lisätä elokuvia toivelistalle ja äänestää seuraavia hankintoja, tehdä varauksia lainavuoroista sekä tilata järjestelmältä erinäisiä ilmoituksia sähköpostilla. Järjestelmä pitää kirjaa elokuvista, niiden lainaustilanteesta ja hankintalistalla olevien elokuvien äänestystilanteesta.

Tämän työn toteutustavaksi on valittu Ruby-skriptikieli ja Rails-ohjelmistokehys. Soveltuvia tekniikoita olisi ollut useita, mutta nimellä Ruby on Rails tunnettu verraten uusi ja suosiotaan lisäävä www-pohjaisten sovellusten toteutustapa nousi kiinnostavimmaksi vaihtoehdoksi. Ruby on Rails -teknologian kehutaan säästävän aikaa ja vähentävän virheitä sovelluksen kehityksessä. Tämän työn toinen tarkoitus onkin tutustua Ruby on Railsin käyttöön ja arvioida sen soveltuvuutta dvd-kirjastojärjestelmän kaltaisten sovellusten rakentamiseen.

Työn tavoitteena on toteuttaa ensimmäinen versio dvd-kirjastojärjestelmästä ja arvioida samalla Ruby on Rails -teknologian käyttöä sovellusalustana käytännön kautta. Koska Ruby sekä Rails ovat molemmat varsin laajoja kokonaisuuksia, ei tarkoitus ole perehtyä niihin kattavasti vaan lähinnä yleisimpien asioiden osalta.

2 TYÖN TEKEMINEN

Tässä luvussa esitellään lyhyesti dvd-kirjastojärjestelmän tekemiseen ja ajamiseen käytetyt ohjelmistot ja teknologiat sekä kerrotaan kehitysprosessin vaiheista.

2.1 Työn vaiheistus

Työn tekeminen alkoi kevällä 2005, ja sitä on tehty pääosin Ionific Oy:n tilojen ulkopuolella omalla ajalla. Ohjelmisto on kirjoitettu ssh-etäyhteyden avulla suoraan sille palvelinkoneelle, jolla sitä on tarkoitus käyttää. Nykytekniikan ja nopeiden verkkoyhteyksien ansiosta ei tekopaikalla ole ollut lainkaan merkitystä.

Tutkintotyön kahta aihepiiriä, tutustumista Ruby on Rails -tekniikkaan ja dvd-kirjastojärjestelmäsovellusta, on tehty samanaikaisesti. Ruby-kielen ja Rails-sovelluskehyksen opiskelu on siis tapahtunut käytännön kautta tekemällä sovellusta. Ruby-kielen tai Rails-ohjelmistokehyksen dokumentaatioon ei ole tutustuttu ennalta kattavasti, vaan niitä on käytetty tarpeen mukaan työn edetessä.

2.2 Kehitys- ja käyttöympäristöjen teknologiat

Työssä käytetyt ohjelmistot ja teknologiat perustuvat avoimeen lähdekoodiin, ja ne ovat maksutta kaikkien käytettävissä. Kaikki käytetyt ohjelmistot ovat saatavilla GPL-lisenssin ehtojen mukaisesti.

2.2.1 Ruby on Rails

Ruby on Rails on dvd-kirjastojärjestelmän tekemiseen käytettävä tekniikka ja toinen tässä tutkintotyössä käsiteltävistä aiheista. Ruby on Rails -järjestelmästä kerrotaan tarkemmin luvussa 5.

2.2.2 WEBrick

WEBrick on Rails-ohjelmistokehyksen oma palvelinohjelma, joka on tarkoitettu nimenomaan Rails-sovellusten alustaksi. WEBrickin etu on integraatio ohjelmistokehykseen, ja sen ansiosta se on helppo otettavaksi käyttöön Rails-

sovelluksissa. WEBrickin haittapuolia ovat rajoittuneisuus vain Rails-sovelluksiin ja hitaus yleiskäyttöisiin www-palvelimiin verrattuna.

2.2.3 Debian GNU/Linux

Linux on suomalaisen Linus Torvaldsin alulle panema UNIX-tyylinen käyttöjärjestelmä. Tarkempi nimitys GNU/Linux tarkoittaa, että käyttöjärjestelmä koostuu Linux-ytimestä (engl. kernel) ja GNU-projektin sovelluksista.

Debian on eräs Linux-jakelupaketti eli distribuutio. Debian on tunnettu vakaudestaan ja luotettavuudestaan, ja se on suosittu erityisesti kokeneiden Linux-käyttäjien keskuudessa.

Dvd-kirjastojärjestelmä asennetaan palvelimelle, jonka käyttöjärjestelmä on tämän työn kirjoitushetkellä Debian Linux versio 3.1 ("Sarge").

2.2.4 MySQL

MySQL on erityisesti web-ympäristöissä laajalti käytetty tietokantapalvelinohjelmisto. Se sopii hyvin pieniin ja keskisuuriin sovelluksiin, mutta sitä käytetään menestyksekkäästi myös suuriksi luokiteltavissa ympäristöissä (esimerkkinä mainittakoon Internet-tietosanakirja Wikipedia, http://www.wikipedia.org). MySQL on saatavilla kahdella eri lisenssillä: vapaa ja ilmainen GPL sekä kaupallinen lisenssi, jota voidaan käyttää GPL-ei-yhteensopivissa käyttökohteissa.

MySQL valittiin dvd-kirjastojärjestelmän käyttämäksi tietokantapalvelimeksi sen yleisyyden takia ja myös siksi, että se oli valmiiksi asennettuna käytetylle palvelinkoneelle.

2.2.5 SSH

Koska dvd-kirjastojärjestelmää on kehitetty suoraan lopullisella palvelimella, on kaikki työskentely tapahtunut ottamalla etäyhteys palvelimelle SSH-ohjelmalla. SSH-yhteyden käyttö on myös mahdollistanut työn tekemisen miltä tahansa Internetiin kytketyltä koneelta.

2.2.6 GNU Emacs

Emacs on UNIX-järjestelmissä tunnettu monipuolinen tekstieditori. Tyypillisesti Emacsia käytetään tekstipohjaisena (vaikka graafinenkin toteutus on olemassa), minkä vuoksi se sopikin mainiosti SSH-yhteyden kanssa työskentelyyn. Dvd-kirjastojärjestelmän lähdekoodi on kirjoitettu kokonaisuudessaan Emacs-editorilla.

2.2.7 Cron

Cron on UNIX-pohjaisissa järjestelmissä käytetty työkalu ajastettujen tehtävien suorittamiseksi. Sitä käytetään tyypillisesti järjestelmän säännöllisten huoltorutiinien ajamiseen. Myös tavalliset käyttäjät voivat lisätä tehtäviä Cronin suoritettavaksi.

2.2.8 CVS

Dvd-kirjastojärjestelmän lähdekoodin versionhallintaan on käytetty CVS-ohjelmistoa. CVS (Concurrent Versions System) on avoimen lähdekoodin projekteissa yleisin versionhallintatyökalu ja sisältyy lähes kaikkiin Linux-jakeluihin. CVS valittiin versionhallintatyökaluksi sen yleisyyden takia ja myös siksi, että se oli valmiiksi asennettuna kaikilla tässä työssä käytetyillä työasemilla ja palvelimilla.

3 OLIO-OHJELMOINNIN PERUSKÄSITTEITÄ

Oliopohjaisen ohjelmoinnin tavoitteena on ohjelmistokomponenttien ylläpitämisen ja uudelleen käytettävyyden helpottaminen. Näiden lisäksi olioparadigman suosiota lisää havainnollinen ja selkeä tapa kuvata ohjelmiston rakennetta. Lähes jokainen esine tai asia voidaan mieltää olioksi, joten olioajattelu vastaa melko hyvin ihmisen tapaa hahmottaa maailmaa. Olioparadigma on myös yleiskäyttöinen, ja se sopii käytännössä kaikenlaisiin ohjelmistoihin. /3/

Oliolla tarkoitetaan ohjelman toteutuksessa olevaa tietojen ja palvelujen kokonaisuutta /2/. Oliolla ei ole tarkkaa määritelmää, sillä lähes mikä tahansa ohjelman osa voi olla olio. Hieman paremmin olion luonnetta kuvaakin sen englanninkielinen nimi object (suomeksi esine tai kohde). Kaikkia olioita yhdistävä piirre on kuitenkin olion vastuu sen omasta tietosisällöstä ja tiedon käsittelyyn tarvittavista palveluista /2/. Teknisesti sanottuna olio on ohjelman rakenteen perusyksikkö. Se ei ole täysin toiminnallinen, kuten aliohjelma eikä toisaalta tietueen kaltainen puhdas tietorakenne /3/. Olio yhdistää nämä molemmat ominaisuudet ja luo niistä suojatun kokonaisuuden kapseloinniksi kutsutun tekniikan avulla /3/.

Ohjelmakoodissa oliot luodaan niiden ominaisuudet määrittelevästä luokasta. Olio on siis luokan ilmentymä eli instanssi ja luokka puolestaan olion tyyppi. Jokainen olio on vähintään yhden luokan jäsen. Luokassa määritellään olion toiminta ja ominaisuudet eli palvelut ja tietosisältö. Tiedot eli attribuutit kuvailevat olion luonnetta, palvelut eli metodit puolestaan kertovat, miten olion tietoja käytetään ja hyödynnetään. /2/

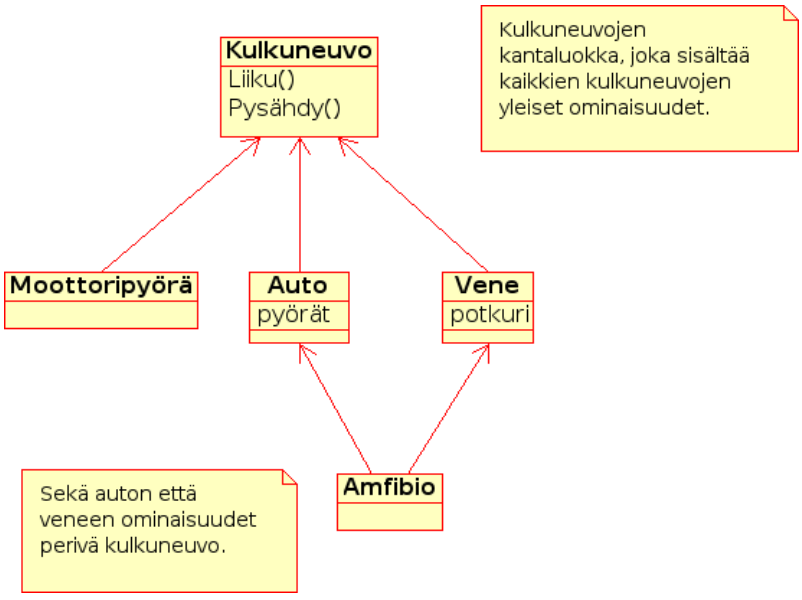
Tärkeä olioajattelun periaate on tiedon piilottaminen. Tämä tarkoittaa sitä, että olio pystyy käsittelemään vain omia tietojaan eikä pääse käsiksi toisen olion tietoihin. Olion tarjoamia palveluita ja tietoja käsitellään olion julkisten rajapintojen kautta, jotka on oliota käytettäessä tunnettava olion käyttötarkoituksen ohella. /2/

Olioiden uudelleenkäytettävyys toteutetaan luokkien periytymisen avulla. Periytymisellä tarkoitetaan luokan omien ominaisuuksien jatkamista jonkin toisen

luokan ominaisuuksilla. Luokka, josta ominaisuudet periytetään, on nimeltään yliluokka tai kantaluokka. Perivä luokka on puolestaan aliluokka. Periytymistä käytetään yleensä silloin, kun eri luokkien yhteisiä ominaisuuksia voidaan kirjoittaa kantaluokaksi ja erot luokkien välillä voidaan toteuttaa aliluokissa. Tällöin yhteisiä ominaisuuksia ei tarvitse kirjoittaa jokaiseen aliluokkaan erikseen, ja niiden ylläpito helpottuu. /2/

Luokka voi periä ominaisuuksia joko yhdestä tai useammasta kantaluokasta. Kun ominaisuuksia peritään vähintään kahdesta luokasta, puhutaan moniperiytymisestä.

Käytännön esimerkkejä olioista voisivat olla kulkuneuvo, auto, vene ja amfibio. Näistä kulkuneuvo on kantaolio. Sen ominaisuuksia (jotka siis pätevät myös kulkuneuvo-luokasta periytetyille luokille) voivat olla esimerkiksi väri ja paino ja palveluita esimerkiksi liikkuminen ja pysähtyminen. Auto ja vene voidaan molemmat periyttää kulkuneuvosta mutta niiden erot tulevat julki niiden omien ominaisuuksien perusteella: autossa on pyörät, veneessä potkuri. Amfibio puolestaan perii sekä auton että veneen ominaisuudet, joten sillä voi liikkua sekä maalla että vedessä. Tämä esimerkki on havainnollistettu kuvassa 1.



Kuva 1 Esimerkki olioista ja periytymisestä

4 WEB-SOVELLUKSEN PERIAATTEITA

Web-sovelluksen erottaa selvimmin tavallisesta sovelluksesta sen käyttöympäristö. Web-sovellus ei ole täysin itsenäinen kokonaisuus, sillä se vaatii toimiakseen toisista sovelluksista koostuvan ympäristön. Tähän ympäristöön kuuluvat tyypillisesti www-palvelinsovellus, tietokantamoottori ja www-selain.

Web-sovelluksen suurin etu on riippumattomuus käyttöympäristöstä käyttäjän näkökulmasta. Koska sovellusta käytetään www-selaimen avulla, ei sovelluksesta tarvitse tehdä erikseen versiota jokaiselle käytettävälle käyttöjärjestelmäalustalle tai prosessoriarkkitehtuurille. Myös ylläpito ja päivitykset helpottuvat, kun toimenpiteitä tehdään keskitetysti palvelimille eikä mahdollisesti jopa tuhansille työasemille.

Web-pohjaisten sovellusten heikkouksia ovat muun muassa tavallisia sovelluksia huonompi suorituskyky ja kyky kommunikoida palomuurin yli. Molemmat heikkouksista liittyvät http-protokollaan, jota web-sovelluksissa tyypillisesti käytetään tiedonvälitykseen palvelimen ja selaimen välillä. Ruuhkainen verkko saattaa aiheuttaa pitkiä viiveitä tiedonkulussa ja pahimmillaan estää sovelluksen käytön kokonaan. Palomuurin ohittaminen on puolestaan mahdollista siksi, että web-sovelluksen ja tavallisen www-käytön verkkoliikennettä on vaikea ellei mahdoton erottaa toisistaan. Näin ollen web-sovellus pystyy kommunikoimaan ulkoverkon kanssa, vaikka tällaista ei esimerkiksi yrityksissä haluttaisi sallia.

Web-sovellus voidaan toteuttaa monella eri tavalla. Yksinkertaisimmillaan selaimelle palautetaan html-muotoinen sivu ilman selaimessa suoritettavia osia. Tällainen sovellus toimii varmimmin eri ympäristöissä, mutta toiminnallisuudesta saatetaan joutua tinkimään. Mikäli halutaan lisätä dynaamisuutta ja interaktiivisuutta, voidaan käyttää lisäksi selaimessa suoritettavia komponentteja, kuten JavaScript-koodia, Java-appletteja tai Macromedian Flash -tekniikkaa. Uutena interaktiivisuutta lisäävänä tekniikkana AJAX (Asynchronous Javascript And XML, asynkronisen JavaScriptin ja XML:n yhdistäminen) on kasvavassa suosiossa.

5 RUBY ON RAILS

Tämä luku esittelee dvd-kirjastojärjestelmän toteutustekniikoina olevien Ruby-skriptikielen ja Rails-ohjelmistokehyksen toimintaperiaatteet pääpiirteittäin. Lisäksi esitellään tapa sovellusrungon luomiseksi Rails-projektille.

5.1 Ruby

Ruby on Yukihiro Matsumoton 1990-luvun puolivälissä kehittämä oliopohjainen skriptikieli. Vaikutteita Ruby on saanut monesta muusta kielestä kuten Python, Lisp, Dylan, Ada, CLU ja Eiffel. /6/

Ruby on luonteeltaan vahvasti oliopohjainen, sillä jopa perustyypit, kuten numerot, ovat olioita. Vastaavasti myös jokainen funktio on jonkin luokan metodi. Muita Rubyn tärkeitä ominaisuuksia ovat poikkeuskäsittely, iteraattorit, säännölliset lausekkeet (engl. regular expressions), operaattoreiden ylikuormitettavuus, roskien keruu (vrt. Java), riippumattomuus käyttöalustasta ja monisäikeistys kaikissa käyttöympäristöissä. Moniperintää Ruby ei tue, mutta vastaava ominaisuus saavutetaan moduulien liittämisellä luokkiin ja niin sanottuja mix-in-luokkia käyttämällä. /6/

Ruby sallii luokkien ja olioiden ominaisuuksien ajon aikaisen muokkaamisen. On esimerkiksi mahdollista lisätä luokkaan tai sen instassiin uusia metodeja ajon aikana, jolloin kahdella saman luokan eri oliolla voi olla eri ominaisuuksia ja ne voivat käyttäytyä eri tavalla. /5/

Ruby-kielen tuki monisäikeisyydelle on täysin riippumaton sovelluksen käyttöjärjestelmäalustasta. Ruby-kielellä on siis mahdollista luoda säikeitä käyttävä sovellus, vaikka käyttöjärjestelmä ei säikeitä tukisikaan (esimerkiksi MS-DOS). /5/

Ruby tukee useita ohjelmointiparadigmoja. Vaikka kieli sinänsä on täysin oliopohjainen, on esimerkiksi proseduraalinen ohjelmointimalli mahdollista määrittelemällä metodit ja muuttujat luokkien ulkopuolelle, jolloin niistä tulee osa niin

sanotua root-oliota (ylimmän tason kantaluokka). Proseduraalinen ohjelmointi Ruby-kielellä onkin siis vain näennäisesti proseduraalista. /6/

Ruby-kielestä on olemassa kolme erilaista toteutusta: virallinen Ruby-tulkki, joka on saatavilla monelle eri käyttöjärjestelmäalustalle, Java-pohjainen JRuby ja roolipelien tekemiseen käytettävä RPG Maker XP -Windows-sovellus. Tässä työssä paneudutaan vain viralliseen toteutukseen. Virallinen Ruby-julkaisu sisältää myös IRB-komentorivitulkin, jolla voidaan nopeasti ja kätevästi testata ohjelmakoodia. /6/

Ruby-kieltä levitetään kahdella eri lisenssillä: GPL:llä (GNU General Public License) tai Rubyn omalla lisenssillä, joka on nähtävissä www-osoitteessa http://www.ruby-lang.org/en/LICENSE.txt. /13/

Vertailutaulukoita muihin korkean tason ohjelmointi- ja skriptikieliin on nähtävissä osoitteissa www.approximity.com/ruby/Comparison_rb_st_m_java.html ja www.smallscript.com/Language%20Comparison%20Chart.asp. Tiivistelmä on esitetty taulukossa 1. Lisätietoa Ruby-kielestä saa sen kotisivuilta osoitteesta http://www.ruby-lang.org.

Taulukko 1 Vertailu Rubyn, C++:n ja Javan välillä

	Ruby	C++	Java
Tyypitys	dynaaminen	staattinen	staattinen
Ajon aikainen pääsy metodeihin/luokkiin/muuttujiin	kyllä	ei	kyllä
Periytyminen	mix-in, moduulit	moniperiytyminen	yksittäisperiytyminen
Näkyvyysmääreet	kyllä	kyllä	kyllä
Mallit (templates)	ei tarvetta	kyllä	ei
Roskien keruu	kyllä	ei	kyllä

Parhaimmillaan Ruby lienee www-pohjaisissa sovelluksissa, mutta se soveltuu myös muunlaisten ohjelmistojen tekemiseen. Ruby on monella tapaa mielenkiintoinen ja persoonallinen vaihtoehto ohjelmointikieleksi, mutta sen oppimiskynnys saattaa olla muita kieliä korkeampi. Kielen sisäistämisen jälkeen sen ideologia ja yksinkertainen syntaksi ovat suureksi avuksi.

5.2 Rails

Rails on Ruby-kielellä kirjoitettu MVC-arkkitehtuuria (Model, View, Controller) noudattava ohjelmistokehys (engl. framework), jonka kantava ajatus on vähentää kirjoitettavan lähdekoodin määrää ja näin minimoida ohjelmointivirheiden mahdollisuus.

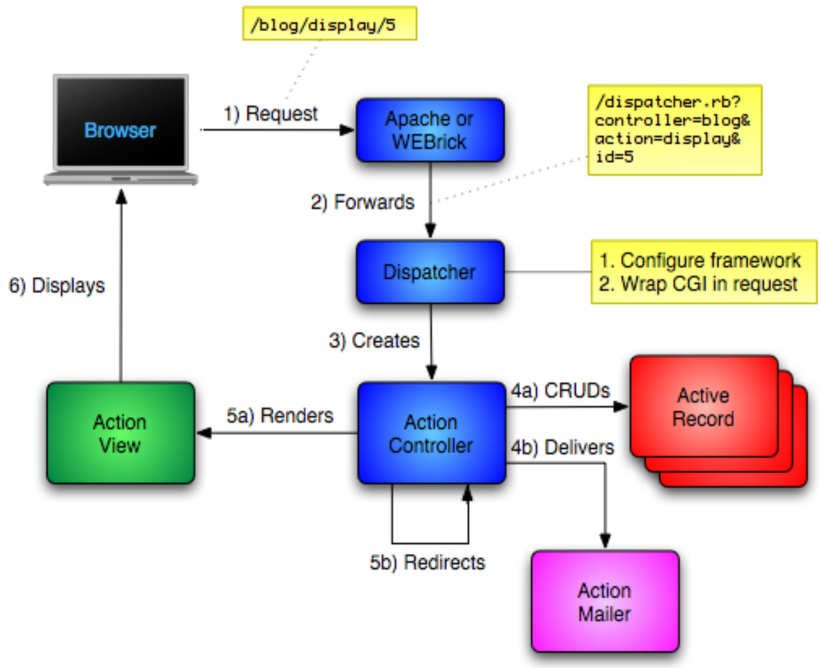
Rails automatisoi monia www-pohjaisiin sovelluksiin liittyviä rutiineja. Esimerkiksi sovelluksen malli (Model) rakentuu automaattisesti, eikä sovelluksessa tarvitse erikseen kertoa tietokannan rakennetta. Rails-periaatteen mukaan rakenteen kertominen malliluokassa olisi turhaa toistoa, koska se on tehty jo tietokantaa luotaessa. Rails osaa tutkia tietokantaa, ja tämän ansiosta hyödyntävissä sovelluksissakaan ei SQL-kielen tuntemusta välttämättä vaadita. Yksinkertaisuutensa ja pitkälle viedyn automaationsa ansiosta Railsin kehutaan vähentävän tarvittavan työmäärän jopa kymmenesosaan perinteisiin tekniikoihin verrattuna. /4/

MVC-arkkitehtuurin mukaisesti Rails-sovelluksen käyttöliittymän määrittää View-komponentti eli näkymä. Näkymät luodaan tekemällä halutulle sivulle rhtml-päätteinen mallinne, joka sisältää tavallisen HTML- tai XHTML-koodin lisäksi myös Ruby-kielen omia rakenteita, joiden avulla luodaan sivun dynaaminen sisältö. /7/

Näkymässä käytetyt tietorakenteet sekä sovelluksen ohjelmalogiikka määritellään sovelluksen Controller- eli ohjainosuudessa. MVC-mallin mukainen toiminnallisuuden eriyttäminen näkymästä helpottaa selvästi sovelluksen muokkaamista ja jatkokehitystä.

Sovelluksen malli (Model) puolestaan käsittelee sovelluksen tietosisältöä ja tietokannan rakennetta. Rails-ohjelmistokehys lukee sovellukselle annetun tietokannan rakenteen ja tulkitsee sen automaattisesti. Sovelluksen kehittäjältä ei vaadita tietokannan luomisen lisäksi muita toimia. Malliluokkaan voidaan määritellä sovelluksessa käytettävät tietokokonaisuudet, kuten lainassa tai toivelistalla olevat elokuvat. Näitä tietovarastoja sitten käsitellään edellä mainitussa ohjainluokassa.

Rails-ohjelmistokehyksen toiminta on havainnollistettu kuvassa 2.



Kuva 2 Rails-ohjelmistokehyksen toimintaperiaate /1/

Rails-ohjelmistokehys koostuu kolmesta pienemmästä kehyksestä: Active Record, Action Pack ja Action Mailer. /4/

Active Record yhdistää tietokannan taulut niitä vastaaviin luokkiin ja luo käyttäjälle näkyvän yhtenäisen konseptin käsiteltävästä järjestelmästä. Active Record on niin sanotun Object-SQL mapping -tekniikan eräs toteutus. Muita vastaavia toteutuksia ovat Java-kielen JDO (Java Data Objects) ja Perl-kielen DBI-luokka (Class::DBI). /10/

Action Pack käsittelee käyttäjältä (selaimelta) vastaanotetut pyynnöt. Pyyntö jaetaan kahteen osaan, joista sovelluksen ohjainlogiikka toteuttaa pyynnön toiminnallisuuden ja näkymämallien avulla tulos esitetään takaisin käyttäjälle. /11/

Action Mailer on vastuussa erilaisten viestien lähettämisestä sekä lähtevän ja tulevan sähköpostin käsittelystä. Sähköpostin lähettämiseen voidaan käyttää joko erillistä SMTP-palvelinta tai paikallista sendmail-ohjelmaa. /12/

Tarkempia tietoja Rails-sovelluskehyksestä voi lukea www-osoitteesta http://www.rubyonrails.com, jossa on myös saatavilla kattava dokumentaatio Ruby on Rails -järjestelmän käytöstä.

5.3 Yksinkertaisen Rails-sovelluksen luominen

Rails-sovelluksen tekeminen aloitetaan tyypillisesti luomalla sovellukselle tietokantarakenne. Rails-käytännön mukaisesti jokaisen tietokannan taulun pääavaimena tulee käyttää id-nimistä kokonaislukukenttää. Erillisen id-kentän käyttö on käytännöllistä ja suositeltavaa myös yleisesti tietokantojen rakenteita suunniteltaessa.

Rails-sovellusta tehtäessä käytetään ohjelmistokehyksen mukana tulevia komentorivityökaluja. Sovelluksen runko luodaan komennolla "rails Projekti", jossa "Projekti" on toteutettavan sovelluksen nimi. Komento luo työhakemistoon kuvan 3 mukaisen hakemistorakenteen, joka sisältää kaikki sovelluksessa tarvittavat tiedostot. Kuvaukset hakemistojen sisällöistä ja merkityksistä on esitetty taulukossa 2. Merkittävin luoduista hakemistoista on app-hakemisto, joka sisältää sovelluksen lähdekoodit. Ohjain-, malli- ja näkymäluokat ovat omissa alihakemistoissaan. Ainoa sovelluksessa tarvittava asetustiedosto on config-hakemistossa sijaitseva database.yml. Tähän tiedostoon määritellään tarvittavat tiedot tietokannan käsittelyyn: tietokannan tyyppi (jokin Railsin tukemista tietokantamoottoreista), sovelluksen tietokannan nimi, tietokantapalvelimen osoite sekä käyttäjätunnus ja salasana.

Rails-ohjelmistokehyksen toinen merkittävä hakemisto on public-hakemisto. Sinne tallennetaan muun muassa sovelluksen käyttöliittymässä esitettävät kuvat, CSS-tyylitiedostot sekä muut julkisesti esitettävät tiedostot. Näille tiedostoille tarvitaan erillinen hakemisto, sillä toisin kuten tavallisten www-sivujen tapauksessa sovelluksen näkymän URL-osoite ei vastaa suoraan sovelluksen hakemistorakennetta.

Kuva 3 Rails-sovelluksen hakemistorakenne

Taulukko 2 Rails-sovelluksen hakemistorakenne

Hakemisto	Merkitys ja sisältö
app	Sovelluksen lähdekoodit
app/controllers	Ohjainluokat
app/helpers	Ns. helper-tiedostot, joita käytetään toistuvan koodin käytön tehostamiseen.
app/models	Malliluokat
app/views	Näkymät
config	Asetustiedostot
config/environments	Kehitys-, testaus- ja tuotantokäyttöympäristöjen asetuksia.
doc	Sovelluksen dokumentointi ja ohjeet.
log	Sovelluksen logitiedostot.
public	Julkiset tiedostot, kuten kuvat ja css-tyylitiedostot.
public/javascripts	Erillisten JavaScript-tiedostojen sijoituspaikka.
script	Erinäisiä sovellusrungon skriptejä.
test	Sovelluksen testiluokat ja muut testaukseen liittyvät tiedostot.
test/fixtures	Testiluokkia
test/functional	Testiluokkia
test/unit	Testidataa

Sovelluksessa tarvittavat MVC-mallin mukaiset malli- ja ohjainluokat luodaan script-hakemistossa sijaitsevalla Ruby-kielisellä generate-skriptillä. Projektihakemistosta ajettuina komennot ovat muotoa "ruby script/generate model Malli" ja "ruby script/generate controller ohjain". Jotta ohjelmistokehys toimisi oikein, on nimeämiskäytäntöjä noudatettava tarkasti. Sovelluksen mallin nimen tulee olla yksikössä ja alkaa isolla kirjaimella. Vastaavasti ohjaimen nimen tulee alkaa pienellä kirjaimella. Käytännössä sovellus pitää toteuttaa englanninkielisenä, jotta järjestelmä osaa automaattisesti yhdistää mallien ja ohjainten nimet oikeisiin tietokannan tauluihin. Ohjain- ja malliluokkia voi lisätä sovellukseen koska tahansa tarpeen mukaan.

Rails-sovelluksen jokaiselle metodille voidaan määrittää näkymä luomalla views-hakemistoon metodin nimeä vastaava rhtml-päätteinen tiedosto. Tiedosto sisältää normaalin HTML- tai XHTML-koodin lisäksi Ruby-kieltä, jolla luodaan näkymän dynaaminen sisältö. Ruby-kielinen osuus kirjoitetaan <% %>- tai <%= %> -merkintöjen väliin. Näistä jälkimmäinen eroaa ensin mainitusta siten, että se liittää osion sisällä suoritettujen komentojen tulosteet osaksi näkymää.

Näkymän kirjoittamiseen tarvittavaa koodia voi vähentää käyttämällä niin sanottua helper-tiedostoa. Siihen kirjoitetaan näkymässä usein toistuva koodi funktioksi, jota voidaan kutsua näkymästä eri parametreilla. Ominaisuus on hyödyllinen esimerkiksi erilaisten listojen tulostuksissa, joissa listojen ulkoasu on sama mutta sisältö toisistaan poikkeava. Tällaisessa tapauksessa listan muotoileva koodi kirjoitettaisiin helper-tiedostoon funktioksi, ja näkymässä funktion parametriksi annettaisiin listan sisältö taulukkona (joka puolestaan olisi määritelty malliluokassa).

Vaihtoehtoinen tapa helper-tiedoston käytölle on render_partial-menetelmä. Sen avulla liitetään näkymään erillinen mallitiedosto, joka määrittelee toistuvan osion ulkoasun. Toistuvan osion (X)HTML-koodi kirjoitetaan erilliseen tiedostoon, jonka nimi alkaa alaviivalla (_). Varsinaisessa näkymässä osio liitetään render_partial-metodilla, jolle annetaan parametreina liitettävän osion nimi ja tulostettava tietosisältö samaan tapaan kuin helper-tiedostoa käytettäessä.

Rails-pohjaisen web-sovelluksen tekeminen vaatii myös www- ja tietokantapalvelinohjelmistot. Railsin tukemia www-palvelimia ovat Apache, lighttpd, ja sovelluskehyksen mukana toimitettava WEBrick. Tietokantamoottoriksi soveltuvat MySQL, PostgreSQL, SQLite, SQL Server, DB2 ja Oracle.

6 DVD-KIRJASTOJÄRJESTELMÄ

Ionific Oy tarjoaa työntekijöilleen mahdollisuuden lainata elokuvia yrityksen dvd-elokuvakirjastosta. Elokuvavalikoimaa on kerätty vuodesta 1999 lähtien, ja nykyään elokuvia on noin 500. Tässä luvussa kerrotaan elokuvien lainaukseen, äänestykseen ja hankintaan liittyvistä käytännöistä sekä esitellään elokuvakokoelman hallinnan helpottamiseksi kehitetty web-sovellus.

6.1 Lainaus- ja äänestyskäytäntö

Ionificin elokuvavalikoimaan tilataan uusia dvd-elokuvia kahden viikon välein 100 euron budjetilla. Jokaisella työntekijällä on mahdollisuus vaikuttaa hankittaviin elokuviin äänestämällä kolmea haluamaansa elokuvaa siten, että ensimmäinen saa neljä pistettä, toinen kolme pistettä ja kolmas kaksi pistettä. Pisteet kerääntyvät elokuvalle jokaisella äänestyskierroksella, kunnes elokuvalla on tarpeeksi pisteitä ja se päätyy tilattavaksi. Kalliiden elokuvien (hinta yli 30 euroa) pistemäärää vähennetään suhteellisesti elokuvan hintaan nähden, jotta ne eivät veisi kohtuuttoman isoa osaa budjetista liian usein./9/

Elokuvat hankitaan ennalta valituista kaupoista (DVDBoxOffice, DiscShop.fi, NetAnttila tai YesAsia.com). Lisättäessä elokuvaa toivelistalle on jokin näistä kaupoista valittava ostopaikaksi. Elokuvan voi lisätä toivelistalle aikaisintaan kolme kuukautta ennen sen dvd-julkaisupäivää. Mikäli julkaisematon elokuva päätyy hankintalistalle keräämiensä pisteiden ansiosta, jäävät sen pisteet normaalisti voimaan, mutta elokuva ei kuitenkaan päädy tilaukseen. /9/

Uusia eli alle kuukauden hyllyssä olleita elokuvia saa pitää lainassa vain päivän kerrallaan. Mikäli uudella elokuvalla on monta halukasta lainaajaa, on etuoikeus lainaamiseen elokuvaa äänestäneillä. Vanhojen elokuvien laina-aika on yksi viikko. Viikonloppu luetaan yhdeksi päiväksi. /9/

6.2 Vanha järjestelmä

Ionific Oy:llä ei ole ollut yhtenäistä dvd-kirjaston hallintajärjestelmää, vaan elokuvalistaa on voinut selailla Intervocativen DVDProfilerin (http://www.intervocative.com) avulla ja äänestys on hoidettu taulukkolaskentaohjelman ja sähköpostin avulla. Lainauksia ei ole myöskään merkitty mihinkään, vaan elokuvan on saanut vapaasti hakea hyllystä. Menetelmä on toiminut ennen, mutta työntekijöiden ja elokuvien määrän kasvaessa on syntynyt entistä paremman hallintajärjestelmän tarve.

6.3 Uusi järjestelmä

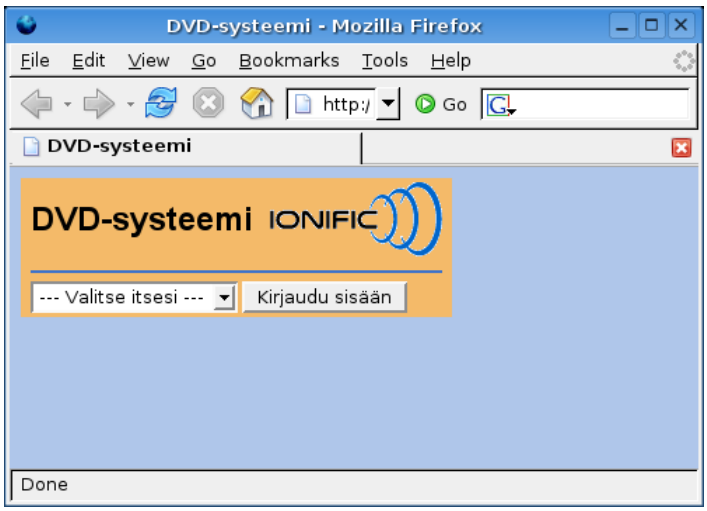
Kantavia ajatuksia uuden dvd-kirjastojärjestelmän suunnittelussa ovat yksinkertaisuus ja helppokäyttöisyys. Koska järjestelmän tarkoitus on helpottaa ja selkeyttää elokuvien lainaamista ja hankintaa, ei uuden järjestelmän käyttöönotto saa aiheuttaa ylimääräistä vaivaa lainaamisessa tai äänestämisessä. Eritoten on tärkeää, ettei lainojen merkintää järjestelmään koeta epämiellyttäväksi tai helposti unohtuvaksi asiaksi.

Ehkä suurin yksittäinen tekijä, joka vaikuttaa sovelluksen käytön helppouteen ja miellyttävyyteen, on sovelluksen käyttöliittymä. Käyttöliittymä on ainoa välikappale sovelluksen käyttäjän ja toiminnallisuuden välillä, ja sen laatu saattaa vaikuttaa sovelluksen käyttökelpoisuuteen jopa teknisiä ominaisuuksia enemmän.

Dvd-kirjastojärjestelmässä pääpaino on näyttävyyden sijaan yksinkertaisessa ja nopeassa käyttöliittymässä. Kaikki oleelliset toiminnot sijoitetaan yhdelle sivulle, ja lähes jokainen toiminto suoritetaan yhtä painiketta painamalla. Tällä tavoin pyritään minimoimaan sovelluksen käyttöön tarvittava aika ja näyttämään kaikki käytössä olevat ominaisuudet yhdellä silmäyksellä.

Dvd-kirjastojärjestelmään kirjaudutaan valitsemalla kirjautusmisnäkymän pudotusvalikosta oma käyttäjänimi. Käyttäjätunnuksen valinta on järkevintä toteuttaa pudotusvalikon avulla yksinkertaisuuden ja pienen tilantarpeen vuoksi. Valikon viemä tila ei ole myöskään missään suhteessa käyttäjätunnusten määrään, joten sivun

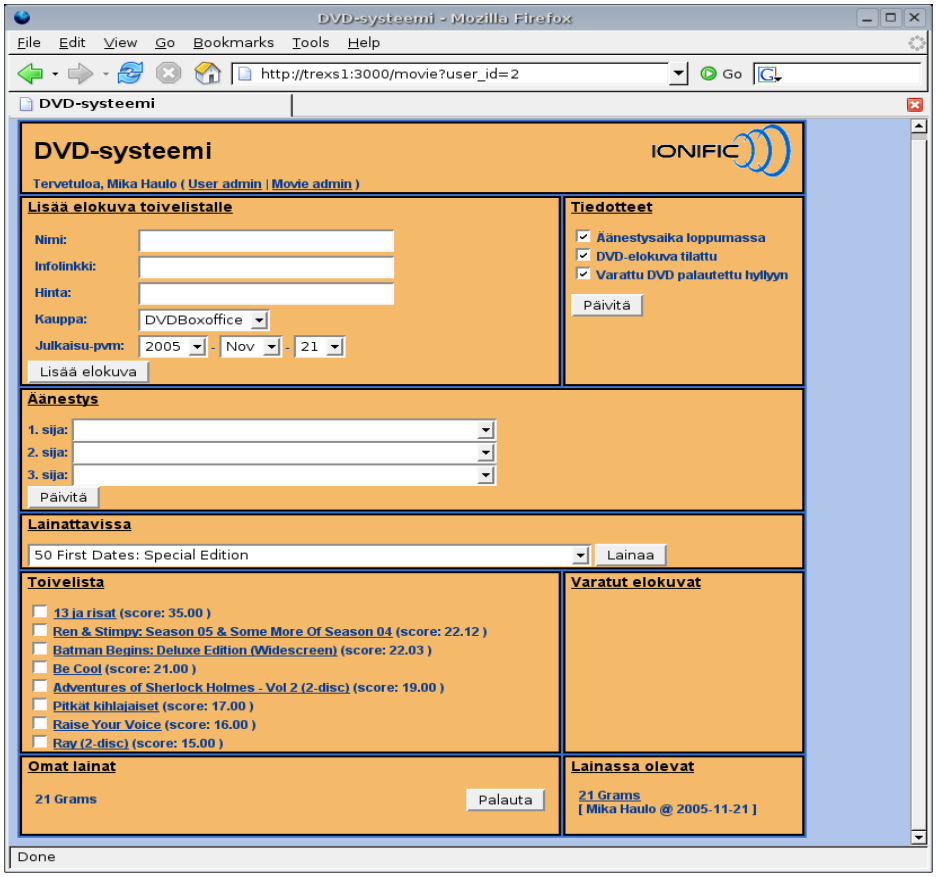
asetelma ei muutu käyttäjätunnusten määrän kasvaessa. Salasanapohjaiselle kirjautumiselle ei toistaiseksi ole tarvetta. Dvd-kirjastojärjestelmän kirjautumisruutu on esitetty kuvassa 4.



Kuva 4 Dvd-kirjastojärjestelmän kirjautumisruutu

Sovelluksen päänäkymä jakautuu otsikkopalkin lisäksi useaan eri osaan. Päänäkymän yläosassa olevilla osioilla lisätään elokuva toivelistalle ja valitaan käyttäjäkohtaiset tiedotteet. Näiden alla on äänestysosio, jolla äänestetään toivelistalla olevia elokuvia hankittaviksi. Äänestyslomakkeen alla on pudotusvalikko lainattavissa olevista elokuvista. Tämän osion avulla merkitään elokuvat lainatuiksi. Viimeiset osiot listaavat toivelistalla olevat, varatut ja lainatut elokuvat. Järjestelmän päänäkymä on esitetty kuvassa 5.

Elokuvan lisäysosiossa elokuvalle syötettäviä tietoja ovat elokuvan nimi, www-linkki lisätietoihin, ostopaikka ja elokuvan hinta. Näistä linkki lisätietoihin on ainoa valinnainen tieto ja muut pakollisia. Tämän lomakkeen kautta kulkeutuvat kaikki elokuvat lopulliseen valikoimaan. Järjestelmä ei tällä hetkellä tarkista millään tavalla, onko elokuva jo syötettynä toivelistalle, sillä sovellus ei mitenkään voi tunnistaa esimerkiksi kirjoitusvirhettä elokuvan nimessä tai muuta eroavaa kirjoitusasua.



Kuva 5 Dvd-kirjastojärjestelmän päänäkymä

Tiedotelista näyttää käyttäjän valitsemat tiedotteet rastitettuina. Tällä hetkellä tarvittavia tiedotteita on kolme: äänestysajan loppumisesta muistuttava ilmoitus (lähetetään joka toinen viikko), ilmoitus dvd-elokuvan tilaamisesta ja varausjonossa ensimmäisenä olevalle henkilölle näkyvä ilmoitus varatun elokuvan palautuksesta hyllyyn. Halutut tiedotteet rastitetaan listasta, ja muutokset päivitetään päivitä-painikkeella. Myöhässä olevan elokuvan palautuskehotusta ei voi kytkeä pois päältä.

Äänestyslomakkeella käyttäjä voi antaa äänestyspisteitä haluamilleen elokuville seuraavasti: ensimmäiselle sijalle valittu elokuva saa neljä pistettä, toiseksi valittu kolme pistettä ja kolmanneksi valittu kaksi pistettä. Äänestettyjä elokuvia voi vaihtaa vapaasti äänestyskierroksen aikana, sillä järjestelmä laskee uudet pisteet aina, kun käyttäjä päivittää antamansa äänet. Uuden äänestyskierroksen alkaessa äänet nollataan. Pisteytysjärjestelmä on kumulatiivinen, eli edellisen kierroksen lopussa voimassa olleet äänestyspisteet jäävät voimaan myös tulevilla kierroksilla.

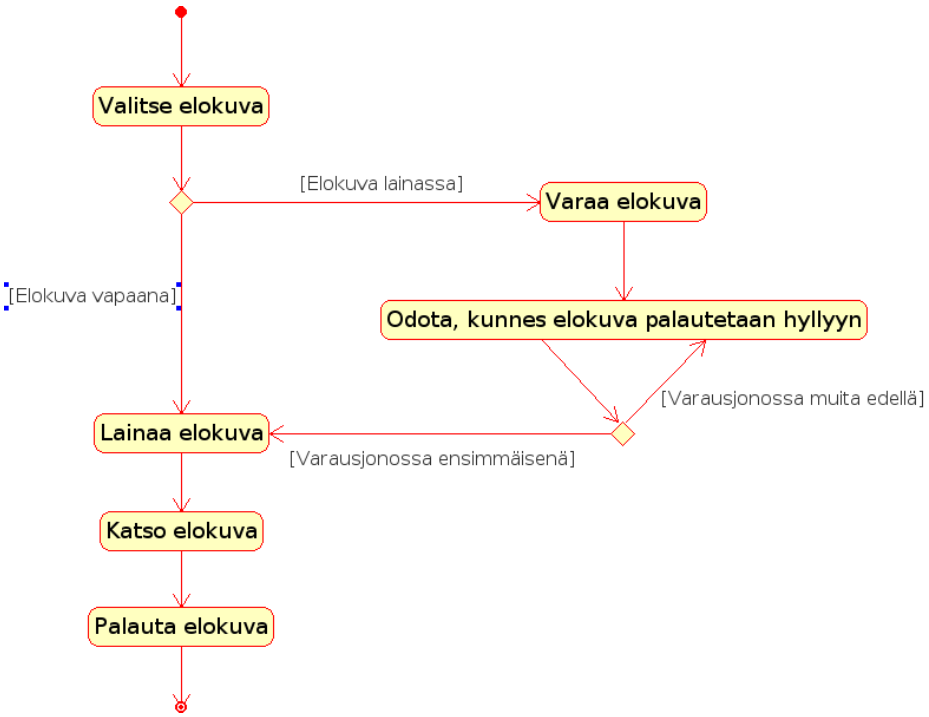
Lainattavien elokuvien listalla ovat kaikki hyllyssä olevat vapaasti lainattavat elokuvat. Elokuvan saa lainata vain silloin, kun sen voi merkitä itselleen lainatuksi ennen hyllystä noutamista. Vastaavasti elokuvaa palautettaessa dvd-levy viedään ensin hyllyyn, ja vasta sen jälkeen merkitään kirjastojärjestelmään palautetuksi. Elokuvat merkitään lainatuksi valitsemalla haluttu elokuva pudotusvalikosta ja painamalla lainaa-painiketta. Pudotusvalikko on käyttäjätunnuslistan tapaan järkevin tapa listata lainattavissa olevat elokuvat. Menettely elokuvaa lainatessa ja palautettaessa on havainnollistettu kuvassa 6.

Toivelista järjestää toivotut elokuvat niiden saamien äänien perusteella. Lista ottaa myös huomioon kalliiden elokuvien pistevähennyksen suhteessa elokuvan hintaan.

Varatut elokuvat esitetään varauslistassa. Kun lainassa ollut elokuva merkitään palautetuksi, ilmestyy varauslistalle palautetun elokuvan kohdalle lainaa-panike sille käyttäjälle, joka on ensimmäisenä varausjonossa. Varauksessa olevat elokuvat eivät näy palautuksen jälkeen lainattavien elokuvien listalla.

Dvd-kirjastojärjestelmä näyttää erikseen kaikki lainatut elokuvat sekä käyttäjäkohtaiset lainatut elokuvat. Käyttäjän itsensä lainaamien elokuvien listassa on palauta-painike, jota painamalla elokuva merkitään palautetuksi hyllyyn. Kaikkien lainojen listassa puolestaan on varaa-painike, jolla varataan elokuvan lainausvuoro. Niiden elokuvien kohdalla, jotka ovat käyttäjällä lainassa, ei tätä painiketta ole. Kaikkien lainojen listalla kerrotaan myös lainaajan nimi ja lainauspäivä.

Ylläpito-oikeudet omistaville käyttäjille esitetään linkit myös manuaaliseen käyttäjien, elokuvien ja tiedotteiden hallintaan. Nämä osuudet on toteutettu Rails-sovelluskehyksen scaffold-ominaisuudella. Näitä osioita ei ole kuitenkaan tarkoitettu normaalikäyttöön, vaan pikemminkin erilaisten asiavirheiden korjaamiseen tarvittaessa.



Kuva 6 Menettely elokuvaa lainatessa, palautettaessa ja varattaessa

Äänestyskierroksen päätyttyä dvd-kirjastojärjestelmä valitsee äänestystulosten perusteella tilattavat elokuvat siten, ettei niiden yhteenlaskettu hinta ylitä ennalta määrättyä budjettia (tällä hetkellä 100 euroa). Elokuva ei myöskään päädy tilaukseen, mikäli sen dvd-julkaisua ei ole saatavilla tilaushetkellä.

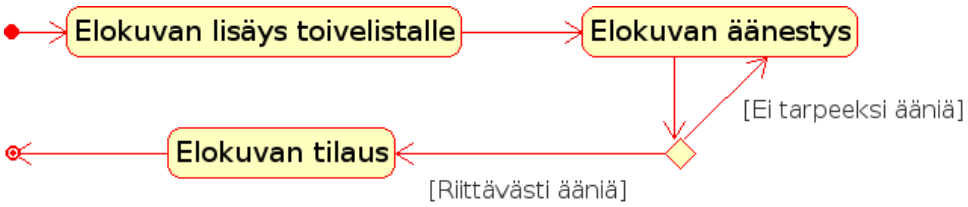
Tilatut elokuvat näkyvät dvd-kirjastojärjestelmässä uusien elokuvien hyväksymislistalla. Elokuvalähetyksen saavuttua täytyy järjestelmälle tiedottaa, että elokuvat ovat lainattavissa. Toimenpide on yksinkertainen ja vaatii vain yhden painikkeen painalluksen, mikäli kaikki tilatut elokuvat ovat saapuneet. Havainnollistus elokuvan päätymisestä dvd-kirjastoon on esitetty kuvassa 7. Hyväksymistoiminnon käyttöliittymä on esitetty kuvassa 8.

6.4 Tekninen toteutus

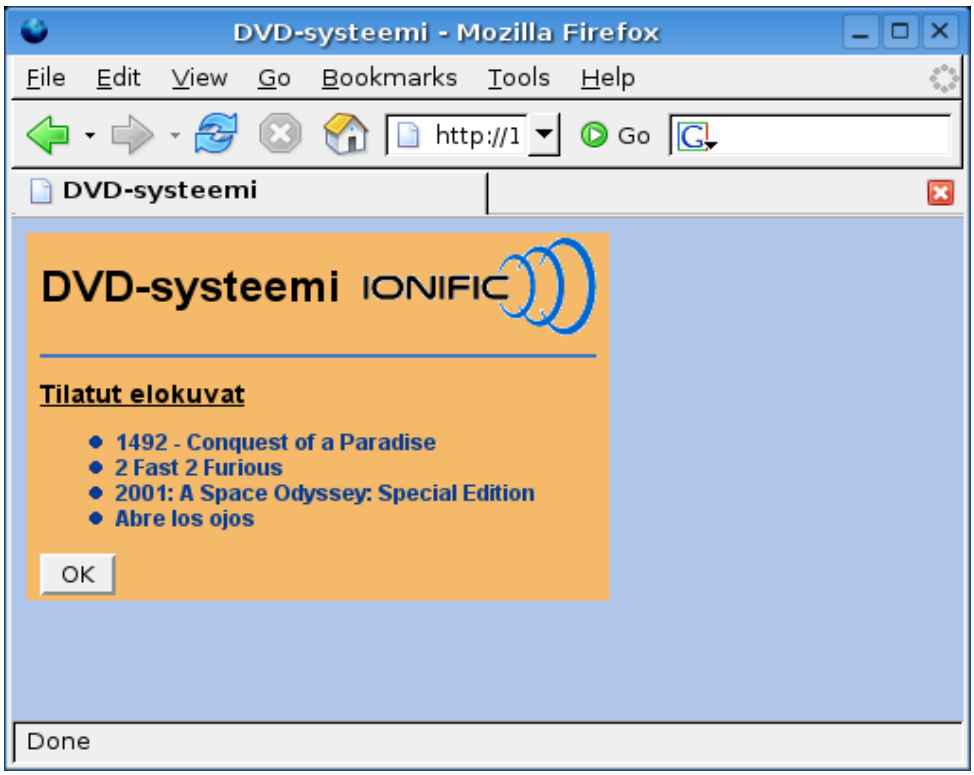
Tässä luvussa kerrotaan dvd-kirjastojärjestelmän teknisestä toteutuksesta. Järjestelmän keskeisiä ohjainluokkia ovat MovieController ja AddMoviesController, ja ne on selostettu ensimmäisinä. Ohjainluokkien jälkeen esitellään keskeiset malliluokat

Movie, Loan ja MonitoredMovie ja lopuksi sovelluksen pääsivun muotoileva näkymäluokka. Kaikkia luokkia ei käydä tarkasti läpi, sillä osa luokista on olemassa vain ohjelmistokehyksen vaatimuksesta eivätkä ne sisällä toiminnallisuutta. Nämä luokat on lueteltu kohdassa "6.4.7 Muut luokat".

Dvd-kirjastojärjestelmän luokkakaavio on esitetty liitteessä 1. Kaaviossa näkyvät dvd-kirjastojärjestelmän omien luokkien lisäksi kantaluokat ActionController::Base, ApplicationController ja ActiveRecord::Base.



Kuva 7 Elokuva päätyminen dvd-kirjastoon



Kuva 8 Uusien elokuvien lisääminen

6.4.1 MovieController

MovieController-luokka on dvd-kirjastojärjestelmän toiminnan ydin. Suurin osa järjestelmän toiminnallisuudesta on koottu tähän luokkaan,a jokainen käytettävissä oleva toiminto kutsuu yhtä tämän luokan metodeista. MovieController-luokan metodeissa hyödynnetään Movie-luokan (sovelluksen mallin) metodeja tietosäilöjen määrittämiseen. Movie-luokka esimerkiksi kertoo ohjainluokalle, mitkä elokuvat ovat lainattavissa ja mitkä lainattuja tarkastelemalla elokuvan status-määreen arvoa.

Login-metodin ainoa tehtävä on koota lista käyttäjistä. Listaa käytetään järjestelmän kirjautumisnäkymässä.

Add_movie-metodia käytetään elokuvan lisäämiseen toivelistalle, ja sen koodi on esitetty kuvassa 9. Metodissa luodaan Movie-luokan olio ja tallennetaan lisäyslomakkeelta saadut tiedot siihen. Mikäli toimenpide onnistuu, ohjataan käyttäjä takaisin etusivulle. Muussa tapauksessa näytetään virheilmoitus lisäyksen epäonnistumisesta.

```
def add_movie
    item = Movie.new
    item.attributes = @params["new_movie"]

    if item.save
      redirect_to(:controller => "movie", :action => "index", :user_id =>
          @params["user_id"])
    else
      render_text "Couldn't add new item"
    end
  end
```

Kuva 9 Elokuvan toivelistalle lisäävä metodi.

Return_movie-metodi merkitsee lainatun elokuvan palautetuksi. Metodin alussa tutkitaan, onko palautettavalla elokuvalla varaajia. Mikäli varaajia on, merkitään elokuvan tilaksi varattu, mutta muussa tapauksessa elokuva merkitään vapaasti lainattavaksi. Jos elokuvan tilan muuttaminen onnistuu, poistetaan lainamerkintä tietokannan loans-taulusta. Toimenpiteen epäonnistuessa näytetään virheilmoitus. Mikäli lainamerkinnän poisto onnistuu, lähetetään varausjonossa ensimmäisenä olevalle käyttäjälle tiedote elokuvan lainausmahdollisuudesta, jos kyseinen käyttäjä on tilannut palautuksesta kertovan tiedotteen. Toimenpiteen epäonnistuessa näytetään

virheilmoitus.

Borrow_movie-metodi merkitsee elokuvan lainatuksi. Parametrina saadun elokuvan tunnistenumeron perusteella tietokannasta etsitään oikea elokuva, ja sen tila merkitään lainatuksi. Lisäksi loans-tauluun lisätään uusi lainamerkintä. Jos toimenpiteet onnistuvat, ohjataan käyttäjä etusivulle. Muussa tapauksessa näytetään virheilmoitus. Borrow_movie-metodin koodi on esitetty kuvassa 10.

```
def borrow_movie
    item = Movie.find(@params["id"])

    item.status = 3
    item.attributes = @params["item"]
    item.save

    loan = Loan.new
    loan.user_id = @params["user_id"]
    loan.movie_id = @params["id"]
    loan.borrowed_at = Time.new

    if item.save && loan.save
      redirect_to(:controller => "movie", :action => "index", :user_id =>
       @params["user_id"])

    else
      render_text "Couldn't update item"
    end

  end
```

Kuva 10 Lainamerkinnän luova borrow_movie-metodi

Borrow_reserved_movie on muuten samanlainen kuin borrow_movie, mutta siinä tuhotaan lisäksi monitored_movies-taulusta elokuvasta tehty varausmerkintä.

Edit_votes-metodilla määritetään käyttäjän äänestämät elokuvat. Tarvittaessa vanhalta äänestettävältä elokuvalta vähennetään äänestyspisteet ennen äänien antamista uudelle elokuvalle. Katkelma metodista on esitetty kuvassa 11.

Update_notifications-metodissa päivitetään tilattavissa olevien tiedotteiden tila (päällä tai pois päältä) valintalomakkeelta saatavien arvojen perusteella.

```
if @params["vote2_old"] != "0"
        item2_old = Movie.find(@params["vote2_old"])

        if item2_old.price > 30
            item2_old.score = item2_old.score * item2_old.price / 25
        end

        item2_old.score = item2_old.score - 3

        if item2_old.price > 30
            item2_old.score = item2_old.score * 25 / item2_old.price
        end

        if item2_old.score < 0
            item2_old.score = 0
        end

        item2_old.save
    end

    if @params["vote2"] != "0"
        item2 = Movie.find(@params["vote2"])

        if item2.price > 30
            item2.score = item2.score * item2.price / 25
        end

        item2.score = item2.score + 3

        if item2.price > 30
            item2.score = item2.score * 25 / item2.price
        end

        item2.save
        user.vote2 = item2.id
    else
        user.vote2 = 0
    end
```

Kuva 11 Katkelma äänestyspisteet päivittävästä edit_votes-metodista

Reserve_movie-metodilla merkitään käyttäjän valitsema elokuva varatuksi luomalla tietokannan monitored_movies-tauluun uusi merkintä. Lisäksi elokuvan tila movies-taulussa merkitään varatuksi.

Monitor_movie-metodilla tilataan ilmoitus valikoimaan lisätystä elokuvasta tai poistetaan ilmoituksen tilaus. Tieto tilauksesta merkitään samaan monitored_movies-tauluun kuin elokuvan varauksissa. Elokuvan varaus ja ilmoituksen tilaus erotetaan toisistaan tietokannan monitored_movies-taulun type-attribuutin eri arvolla.

6.4.2 AddMoviesController

AddMoviesController-luokkaa käytetään tilattujen elokuvien siirtämiseksi elokuvakirjaston valikoimaan. Toteutus on erittäin yksinkertainen, sillä elokuvan siirto tilattujen elokuvien listalta valikoimaan tapahtuu muuttamalla elokuvaolion status-attribuutin arvoa ja asettamalla date-attribuutin arvoksi nykyinen aika.

6.4.3 Movie

Movie-luokka on malliluokka, jonka metodit palauttavat listat lainatuista, lainaamattomista, toivelistalla olevista ja tilatuista elokuvista. Luokan koodi on esitetty kokonaisuudessaan kuvassa 12. Lisäksi Movie-luokassa asetetaan validates_presence_of-lauseen avulla ehdot attribuuteille, jotka ovat pakollisia uutta elokuvaa lisättäessä.

```
class Movie < ActiveRecord::Base

  validates_presence_of :title
  validates_presence_of :price
  validates_presence_of :shop

  def self.find_not_borrowed
    find_all("status = 2")
  end

  def self.find_borrowed
    find_all("status = 3 OR status = 4")
  end

  def self.find_wished
    find_all("status = 0 order by score desc")
  end

  def self.find_ordered
    find_all("status = 1 order by title")
  end

end
```

Kuva 12 Movie-luokka kokonaisuudessaan

6.4.4 Loan

Loan-luokassa on kaksi metodia, joista toinen palauttaa listan vain käyttäjän omista lainoista ja toinen listan kaikista kirjastojärjestelmään merkityistä lainoista. Listat esitetään dvd-kirjastojärjestelmän päänäkymässä.

6.4.5 MonitoredMovie

MonitoredMovie-luokka sisältää elokuvien varaus- ja tarkkailutiedon. Luokan metodien avulla löydetään elokuvat, joilla on varauksia ja jokaisen varatun elokuvan varausjonon kärjessä oleva lainaaja.

6.4.6 Index.rhtml

Sovelluksen päänäkymä muotoillaan views-hakemistossa olevassa index.rhtml-tiedostossa. Tiedoston sisältö on pääosin tavallista HTML-koodia lukuun ottamatta dynaamisen sisällön luovia Ruby-osioita. Kuvassa 13 on osa index.rhtml-tiedoston koodista, joka kokoaa listan lainattavissa olevista elokuvista.

```
<form action="http://localhost:3000/movie/borrow_movie" method="post">
        <input type="hidden" name="user_id" value="<%= @lainaaja.id  %>">
        <select name="id">
        <% @not_borrowed.each do |@item| %>
                <option value="<%= @item.id %>"><%= @item.title %></option>
        <% end %>
        </select>
        <input type="submit" value="Lainaa">
</form>

<select name="month">
        <% (1..Time.now.month-1).each do |monthnum|  %>
          <% t = Time.mktime(Time.now.year, monthnum, Time.now.day, 0, 0, 0) %>
          <option value="<%= monthnum  %>" ><%= t.strftime("%b")  %></option>
        <% end %>

        <option value="<%= Time.now.month  %>" selected>
          <%= Time.now.strftime("%b")  %>
        </option>

        <% (Time.now.month+1..12).each do |monthnum|  %>
          <% t = Time.mktime(Time.now.year, monthnum, Time.now.day, 0, 0, 0) %>
          <option value="<%= monthnum  %>" >
            <%= t.strftime("%b")  %>
          </option>
        <% end %>
</select>
```

Kuva 13 Lainattavien elokuvien listan muodostava osa päänäkymän koodista

6.4.7 Muut luokat

Dvd-kirjastojärjestelmässä on myös luokkia, joista ei ole toteutusta, mutta jotka ovat tarpeellisia sovelluskehyksen toiminnan kannalta. Näitä luokkia on kuusi, ja ne ovat AdminMovie, AdminNote, AdminNotification, AdminUser, Notification ja SelectedNotification.

Dvd-kirjastojärjestelmän hallintaliittymä ei ollut keskeisin asia tässä, joten se on toteutettu yksinkertaisesti Rails-ohjelmistokehyksen scaffold-ominaisuuden avulla. Scaffold luo automaattisesti käyttöliittymän, jonka avulla voi käsitellä halutun tietokannan taulun sisältöä. Luokat AdminMovieController, AdminNotificationController, AdminSelectedNotificationController ja AdminUserController on toteutettu scaffold-ominaisuutta käyttäen.

6.4.8 Ajastetut toiminnot

Dvd-kirjastojärjestelmä seuraa jatkuvasti lainassa olevia elokuvia ja äänestyskierroksen kulkua. Säännöllisesti suoritettavat rutiinit on toteutettu hyödyntäen UNIX-järjestelmien cron-ohjelmaa. Tarkistus myöhässä olevista elokuvista, huomautus äänestysajan loppumisesta ja äänestyskierroksen päättäminen ovat Ruby-kielisiä skriptejä, joissa ei ole käytetty Rails-ohjelmistokehystä. Cron suorittaa myöhässä olevien elokuvien tarkistuksen päivittäin ja äänestyskierroksen rutiinit kahden viikon välein.

Esimerkkinä suoritettavasta skriptistä on kuvassa 14 esitetty myöhässä olevan elokuvan palautuspyyntö.

```
require "mysql"
require "time"
require 'net/smtp'

begin
    db = Mysql.real_connect("server", "username", "password", "database")
    date_now = Time.new
    result = db.query("SELECT l.user_id, l.movie_id, l.borrowed_at, u.email
                       FROM loans l, users u WHERE l.user_id = u.id")

    result.each do |row|
        borrowed_at = Time.parse(db.escape_string(row[2]))
        res = db.query("SELECT date FROM movies WHERE id = " + row[1])
        movie_date = ""

        res.each do |mdate|
            movie_date = Time.parse(db.escape_string(mdate[0]))
        end

        if date_now - movie_date > 2592000
            expiration_date = borrowed_at + 432000
        else
            expiration_date = borrowed_at + 86400
            if date_now.wday == 5
                expiration_date = expiration_date + 172800
            end
        end

        if expiration_date < date_now
            msg = [ "Subject: Testail\n", "\n",
                  "Lainaamasi elokuva on myöhässä\n" ]
            Net::SMTP.start('localhost') do |smtp|
                smtp.sendmail( msg,  'dvdsysteemi@ionific.com', row[3] )
            end
        end
    end

rescue MysqlError => e
    print "Error code: ", e.errno, "\n"
    print "Error message: ", e.error, "\n"
ensure
    db.close
end
```

Kuva 14 Cron-työkalun päivittäin ajama skripti, joka lähettää palautuspyynnön myöhässä oleville lainoille.

6.5 Palvelinohjelmisto

Dvd-kirjastojärjestelmä hyödyntää Rails-ohjelmistokehyksen omaa WEBrick-palvelinohjelmaa. Myös huomattavasti yleisemmän www-palvelinohjelmisto Apachen käyttäminen Rails-sovelluksissa on mahdollista, mutta tässä tapauksessa WEBrick on helpompi ottaa käyttöön, sillä Apachen käyttäminen olisi vaatinut ylimääräistä konfigurointia. Apachen etu WEBrickiin nähden on nopeus. Mikäli sovelluksen suorituskyky on tärkeä kriteeri, on ehkä syytä käyttää aikaa Apachen konfiguroimiseksi sovellusta varten. Dvd-kirjastojärjestelmän palvelimelle aiheuttama kuormitus ei ole niin suuri, että Apachen käyttäminen olisi perusteltua.

6.6 Tietokanta

Dvd-kirjastojärjestelmä käyttää SQL-pohjaista tietokantaa sovelluksen tietosisällön tallentamiseen. Järjestelmän tietokanta koostuu kuudesta taulusta. Tässä kappaleessa esitellään lyhyesti taulujen sisältö ja käyttötarkoitus.

Tietokannan normalisointi on pääasiassa kolmannen normaalimuodon (3NF) mukainen. Hieman poikkeuksellisesti jokaisessa taulussa on erikseen id-niminen pääavain, vaikka olisi mahdollista käyttää jotain toista taulussa olevaa kenttää pääavaimena. Rails-ohjelmistokehys vaatii toimiakseen luotettavasti nimenomaan id-nimisen pääavaimen, mutta sen käyttöä voi selkeyden vuoksi suositella yleisesti muissakin tietokantasovelluksissa. Dvd-kirjastojärjestelmän tietokannan ER-malli on esitetty liitteessä 2.

Selkeä poikkeus kolmannesta normaalimuodosta on users-tauluun sijoitetut elokuvien äänestystiedot (vote1, vote2 ja vote3). Normaalisti äänestystiedot sijoitettaisiin omaan tauluunsa, ja ER-mallissa ne näkyisivät users-taulun moniarvoisena ominaisuutena. Koska äänestysattribuutteja on vain kolme, niiden laittaminen users-taulun attribuuteiksi osoittautui helpoimmaksi vaihtoehdoksi ohjelmistokehyksen toiminnan opiskelun ohessa.

Movies-taulu sisältää elokuvien tiedot. Elokuvista tallennetaan nimikkeen lisäksi

Internet-linkki elokuvasta kertovalle sivulle, pistetilanne toivelistalla olevien elokuvien äänestystä varten, päivämäärä useita eri käyttötarkoituksia varten, elokuvan tilan kertova tieto (esimerkiksi onko elokuva lainassa vai ei), ostopaikka ja elokuvan hinta ostopaikassa. Movies-taulun tarkka rakenne on kuvattu taulukossa 3.

Taulukko 3 Tietokannan movies-taulu

Attribuutti	Tietotyyppi	Kuvaus
id	integer	Taulun pääavain.
title	varchar	Elokuvan nimi.
infolink	varchar	Linkki elokuvasta kertovalle sivulle.
score	integer	Elokuvan saamat äänestyspisteet.
status	integer	Elokuvat tila (esim. hyllyssä tai lainassa).
date	date	Päivämäärä (useita käyttötarkoituksia).
price	float	Elokuvan hinta ostohetkellä.
shop	varchar	Kauppa, josta elokuva ostetaan.

Users-taulu listaa järjestelmän käyttäjät eli Ionific Oy:n työntekijät. Käyttäjätietoja ovat nimi, käyttöoikeustaso, sähköpostiosoite ja käyttäjän äänestämien elokuvien tunnistenumerot. Mikäli käytössä olisi salasanapohjainen järjestelmään kirjautuminen, myös salasanat tallennettaisiin salatussa muodossa tähän tauluun. Jatkokehitysideana on lukea käyttäjätiedot suoraan Ionificin tietojärjestelmien keskitetystä käyttäjätunnustietokannasta. Users-taulun tarkka rakenne on kuvattu taulukossa 4.

Taulukko 4 Tietokannan users-taulu

Attribuutti	Tietotyyppi	Kuvaus
id	integer	Taulun pääavain.
name	varchar	Käyttäjän nimi.
vote1	integer	Elokuva, jolle annetaan 4 äänestyspistettä.
vote2	integer	Elokuva, jolle annetaan 3 äänestyspistettä.
vote3	integer	Elokuva, jolle annetaan 2 äänestyspistettä.
admin	tinyint	Käyttäjän käyttöoikeustaso.
email	varchar	Käyttäjän sähköpostiosoite.

Loans-taulu on kirjanpito lainatuista elokuvista. Siihen tallennetaan lainaajan ja lainatun elokuvan tunnistenumerot ja lainauspäivämäärä. Lainauspäivän perusteella

saadaan selville esimerkiksi se päivä, jolloin elokuva pitäisi palauttaa. Tauluun lisätään rivi aina silloin, kun käyttäjä lainaa elokuvan, ja vastaava rivi poistetaan, kun elokuva palautetaan. Tietokannan ER-mallissa loans-taulu on relaatio users- ja movies-taulujen välillä. Loans-taulun tarkka rakenne on kuvattu taulukossa 5.

Taulukko 5 Tietokannan loans-taulu

Attribuutti	Tietotyyppi	Kuvaus
id	integer	Taulun pääavain.
user_id	integer	Viiteavain käyttäjän tunnistenumeroon.
movie_id	integer	Viiteavain elokuvan tunnistenumeroon.
borrowed_at	date	Elokuvan lainauspäivämäärä.

Tieto käyttäjien varaamista elokuvista ja tilatut tiedotteet uusien elokuvien lisäämisestä hyllyyn tallennetaan monitored_movies-tauluun. Tauluun tallennetaan käyttäjän ja elokuvan tunnistenumerot ja tieto siitä, onko kyse elokuvan varauksesta vai ilmoituksen tilaamisesta. ER-mallissa monitores_movies-taulu on relaatio users- ja movies-taulujen välillä. Monitored_movies-taulun tarkka rakenne on kuvattu taulukossa 6.

Taulukko 6 Tietokannan monitored_movies -taulu

Attribuutti	Tietotyyppi	Kuvaus
id	integer	Taulun pääavain.
user_id	integer	Viiteavain käyttäjän tunnistenumeroon.
movie_id	integer	Viiteavain elokuvan tunnistenumeroon.
type	tinyint	Varauksen tyyppi.

Notifications- ja selected_notifications-taulut liittyvät järjestelmän tarjoamiin tiedotteisiin. Notifications-taulu sisältää tiedotteen nimen, ja selected_notifications-taulu puolestaan tiedon siitä, ketkä käyttäjät ovat tilanneet mitäkin tiedotteita. Tauluun tallennetaan käyttäjän ja tiedotteen tunnistenumerot ja tiedotteen tyyppi. Notes- ja notifications-taulujen tarkat rakenteet on kuvattu taulukoissa 7 ja 8.

Taulukko 7 Tietokannan notifications-taulu

Attribuutti	Tietotyyppi	Kuvaus
id	integer	Taulun pääavain.
description	varchar	Tiedotteen nimi.

Taulukko 8 Tietokannan selected_notifications-taulu

Attribuutti	Tietotyyppi	Kuvaus
id	integer	Taulun pääavain.
user_id	integer	Viiteavain käyttäjän tunnistenumeroon.
notification_type	integer	Tiedotteen tyyppi (viiteavain notifications-tauluun).
notification_value	integer	Tieto siitä, onko tiedote tilattu vai ei.

6.7 Jatkokehitysajatuksia

Vaikka dvd-kirjastojärjestelmä on perustoiminnaltaan ja ominaisuuksiltaan riittävä dvd-kirjaston hallintaan, siinä on vielä paljon parantamisen varaa ja tilaa lisäominaisuuksille. Tässä luvussa esitellään joitakin järjestelmän kehityksen aikana mieleen tulleita ideoita toteutettavaksi.

Tällä hetkellä järjestelmään kirjaudutaan valitsemalla oma nimi kirjautumisnäkymän listasta. Tietoturvan kannalta ratkaisu on riittävä tämän kaltaisessa sovelluksessa, mutta nimien lisääminen ja poistaminen erikseen dvd-kirjastojärjestelmään teettää ylimääräistä työtä. Käytettävyydeltään parempi vaihtoehto olisikin käyttäjätietojen hakeminen suoraan niitä hallinnoivalta palvelimelta. Tällöin tunnus tulisi voimaan samalla, kun uudelle työntekijälle luodaan käyttäjätunnus Ionificin tietoverkkoon.

Järjestelmässä on myös selkeä tarve paremmille käyttäjien ja tiedotteiden hallintaliittymille. Rails-ohjelmistokehyksen scaffold-ominaisuudella saa tehtyä välttävän hallintaliittymän, mutta paremman käytettävyyden, yhtenäisen tyylin ja paremman toiminnan takaamiseksi on käsin tehty yhtenäinen hallintaliittymä tarvittaville ominaisuuksille parempi vaihtoehto.

Koska dvd-kirjastojärjestelmää tehtäessä on Ruby-kielen ja Rails-ohjelmistokehyksen käyttö opeteltu ilman perusteellista kieleen tutustumista, on ohjelmakoodiin jäänyt varmasti paljon kohtia, jotka oikeaoppisesti Rails-ohjelmistokehystä hyödyntämällä olisivat paljon yksinkertaisemmin toteutettuja. Muutenkin projektin kokeiluluontoisuudesta johtuen koodissa on paljon siistittävää ja parannettavaa.

Ionific Oy:ssä on myös käytettävyyden ja käyttöliittymäsuunnittelun osaajia. Dvd-kirjastojärjestelmän käyttömukavuutta voisi lisätä huomattavasti antamalla käyttöliittymän suunnittelu heidän tehtäväkseen. MVC-arkkitehtuurin ansiosta käyttöliittymän suunnittelu ja muokkaus eivät häiritse teknisen toteutuksen muokkaamista.

7 YHTEENVETO

Tämä työ on käsitellyt Ruby on Rails -tekniikan avulla toteutettua web-pohjaista dvd-kirjastojärjestelmäsovellusta. Sovellus on tarkoitettu Ionific Oy:n sisäiseen käyttöön yrityksen dvd-kirjaston hallinnan helpottamiseksi. Työ on koostunut kahdesta rinnakkaisesta osuudesta, joista toinen on ollut itse sovelluksen suunnittelu, ja toinen toteutustekniikkaan tutustuminen.

Ionific Oy:n dvd-kirjastojärjestelmän toteutus valittiin tämän työn aiheeksi, koska yrityksessä on ollut jo kauan dvd-valikoiman hallintajärjestelmän tarve ja koska se soveltuu hyvin web-pohjaiseksi sovellukseksi. Monen käyttökelpoisen tekniikan joukosta toteutustavaksi valittiin Ruby on Rails, joka on suhteellisen uusi ja mielenkiintoinen tulokas web-sovellusten toteutustekniikoiden joukossa.

Ruby on Rails erottuu muista www-sovelluksien kehittämiseen tarkoitetuista tekniikoista persoonallisella tyylillään ja kattavalla ohjelmistokehyksellään. Järjestelmän kantavia ajatuksia ovat kirjoitettavan koodin määrän minimointi ja turhan toiston välttäminen ja näillä tavoin ohjelmointivirheiden mahdollisuuden pienentäminen. Ohjelmistokehittäjän huomio pyritään kohdistamaan ohjelmiston keskeisiin osiin, ja ohjelmistokehys puolestaan huolehtii monesta perustason rutiinista.

Ruby on Rails ei ole tekniikkana ainoastaan uusi vaan myös selkeästi erilainen kilpaileviin tekniikoihin verrattuna, ja se lupaa suuria parannuksia sovelluskehitykseen. Rails-ohjelmistokehyksen tekijöiden mukaan sovelluksen tekemiseen kuluva aika ja kirjoitettavan koodin määrä vähenevät noin kymmenesosaan perinteisiin tekniikoihin verrattuna Rails-ohjelmistokehyksen ansiosta.

Ensivaikutelma Ruby on Rails -järjestelmästä on hieman sekava. Ohjelmistokehyksessä on paljon ominaisuuksia, ja niiden opetteluun kuluu paljon aikaa. Rails-sovelluksen voi toki tehdä myös hyödyntämättä kaikkia sovelluskehyksen ominaisuuksia, mutta tällöin sovellus ei täytä täysin Rails-ajattelun periaatteita eikä sovelluksen lähdekoodi pysy niin yksinkertaisena ja selkeänä kuin se voisi olla.

Huolellinen tutustuminen ohjelmistokehyksen dokumentaatioon ja esimerkkeihin on lähes välttämätöntä. Järjestelmää tuntematta ja ohjeita kehityksen aikana seuraamalla on oman sovelluksen kokonaisuutta varsin vaikea pitää eheänä ja selkeänä. Rails onkin tehokas ohjelmistokehys vain oikein käytettynä ja hyödynnettynä. Osaamattomissa käsissä lopputulos on korkeintaan keskinkertainen.

Vaikka Ruby on Rails vaatii perehtymistä ja aikaa, on sen perusajatus ja ideologia hyvä. Rails-ohjelmistokehyksen kerrosten yhteistoiminta on kiitettävän sulavaa, ja ohjelmistorungon perusrutiinien siirtäminen pois ohjelmistokehittäjän vastuulta on erittäin hyödyllinen ominaisuus.

Ruby on Rails on lupaava ja ennen kaikkea tutustumisen arvoinen kokonaisuus. Lyhyen kokemuksen perusteella ei järjestelmästä kannata vetää kovin vankkoja johtopäätöksiä, mutta jo yksinkertaisen sovelluksen avulla voi tehdä suuntaa antavia havaintoja. Lupailtua ajansäästöä ei ehkä ensimmäisten projektien aikana saavuteta järjestelmän opetteluun kuluvan ajan takia, mutta mitä enemmän järjestelmää opettelee, sitä enemmän sitä oppii arvostamaan ja hyödyntämään.

Dvd-kirjastojärjestelmästä on tämän työn yhteydessä kirjoitettu ensimmäinen testiversio, joka on ominaisuuksiltaan kattava mutta toimintavarmuudeltaan riittämätön tuotantokäyttöön. Sovellus on tehty kokeiluluontoisesti Ruby-kielen ja Rails-ohjelmistokehyksen ominaisuuksia kokeillen ja havainnoiden ja on siksi jäänyt osittain epäyhtenäiseksi ja jättänyt paljon parantamisen varaa.

Ennen dvd-kirjastojärjestelmän lopullista käyttöönottoa sovellus kirjoitetaan alusta asti uudestaan. Tällä tavoin päästään eroon ylimääräisestä ja hatarasta koodista, ja samalla voidaan korjata suunnittelulähtöisiä epäkohtia. Varsinkin luokkajakoa on syytä tarkentaa. Olemassa olevaa koodia on mahdollista käyttää työn pohjana.

Ensimmäinen tuotantokäyttöön tehtävä versio tulee sisältää vain dvd-kirjastojärjestelmän tärkeimmät toiminnot. Järjestelmän jatkokehitys alkaa kuitenkin heti käyttöönoton jälkeen, ja ominaisuuksia voidaan lisätä ja kehittää tarpeen mukaan.

LÄHTEET

Painetut lähteet:

1 Haikala Ilkka, Märijärvi Jukka, Ohjelmistotuotanto, 8. painos. Satku. Helsinki 2002. 430 s.

2 Hietanen, Päivi, C++ ja olio-ohjelmointi, 6. painos. Teknolit. Porvoo 2000. 830 s.

3 Koskimies, Kai, Oliokirja, 1. painos, Satku - Kauppakaari. Helsinki 2000. 422 s.

Sähköiset lähteet:

4 Ruby on Rails. [www-sivu]. [viitattu 28.9.2005] Saatavissa: http://www.rubyonrails.com

5 Ruby Home Page. [www-sivu]. [viitattu 28.9.2005] Saatavissa: http://www.ruby-lang.org

6 Ruby programming language - Wikipedia, the free encyclopedia. [www-sivu]. [viitattu 28.9.2005] Saatavissa: http://en.wikipedia.org/wiki/Ruby_programming_language

7 Ruby on Rails - Wikipedia, the free encyclopedia. [www-sivu]. [viitattu 28.9.2005] Saatavissa: http://en.wikipedia.org/wiki/Ruby_on_Rails

8 Pöyry, Pekka, Relaatiotietokannan suunnittelu. Kurssimateriaali. Tampereen ammattikorkeakoulu. Sähköosasto. Tampere 2005.

9 Botnia Hightechin Intranet: Dvd-äänestys- ja lainausohjeet. [www-sivu]. [viitattu 30.9.2005].

10 Active Record - Object-relation mapping put on rails. [www-sivu]. [viitattu 30.9.2005] Saatavissa: http://ar.rubyonrails.com

11 Action Pack - On rails from request to response. [www-sivu]. [viitattu 20.9.2005] Saatavissa: http://ap.rubyonrails.com

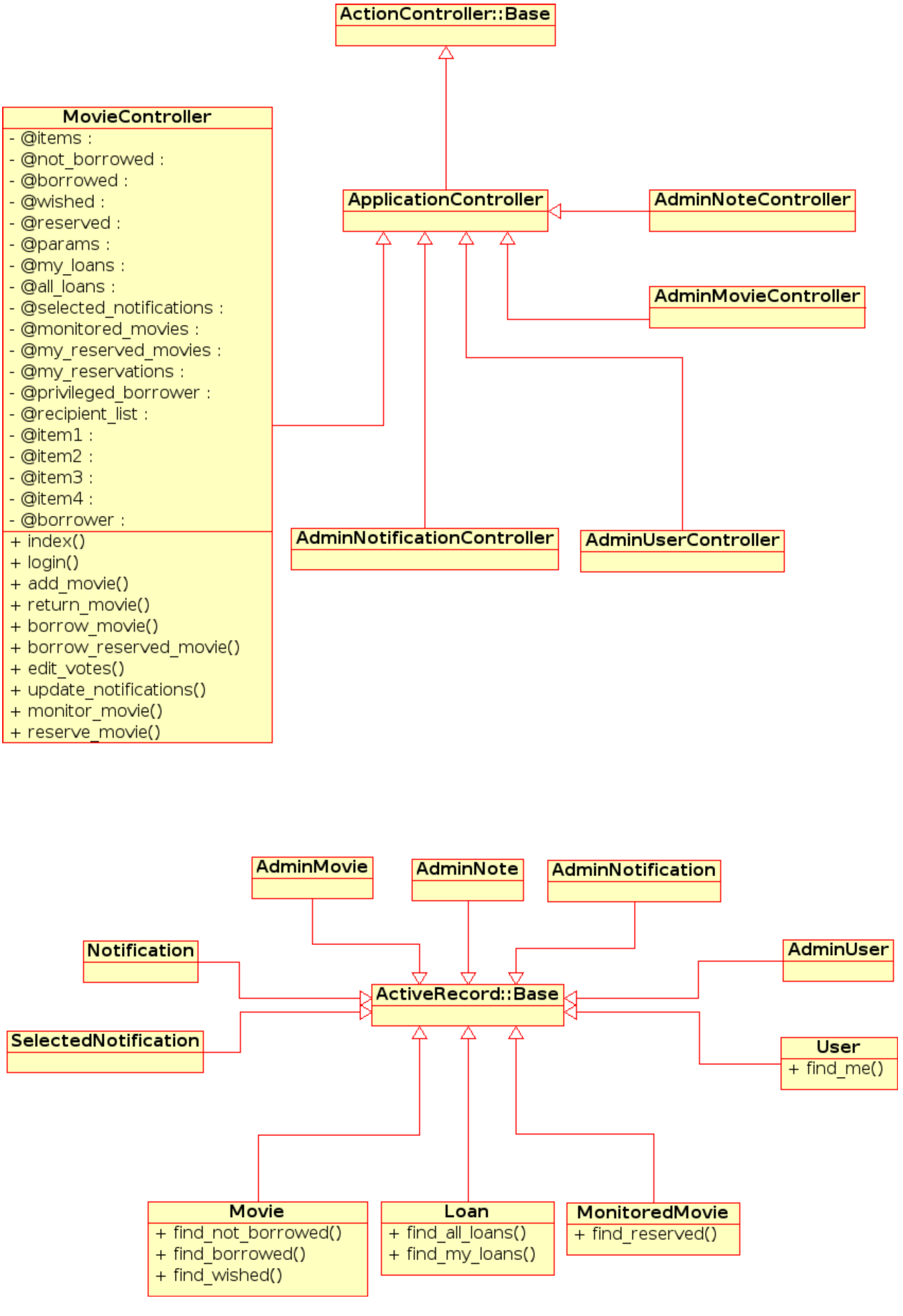
12 Action Mailer - Easy email delivery and testing. [www-sivu]. [viitattu 30.9.2005] Saatavissa: http://am.rubyonrails.com

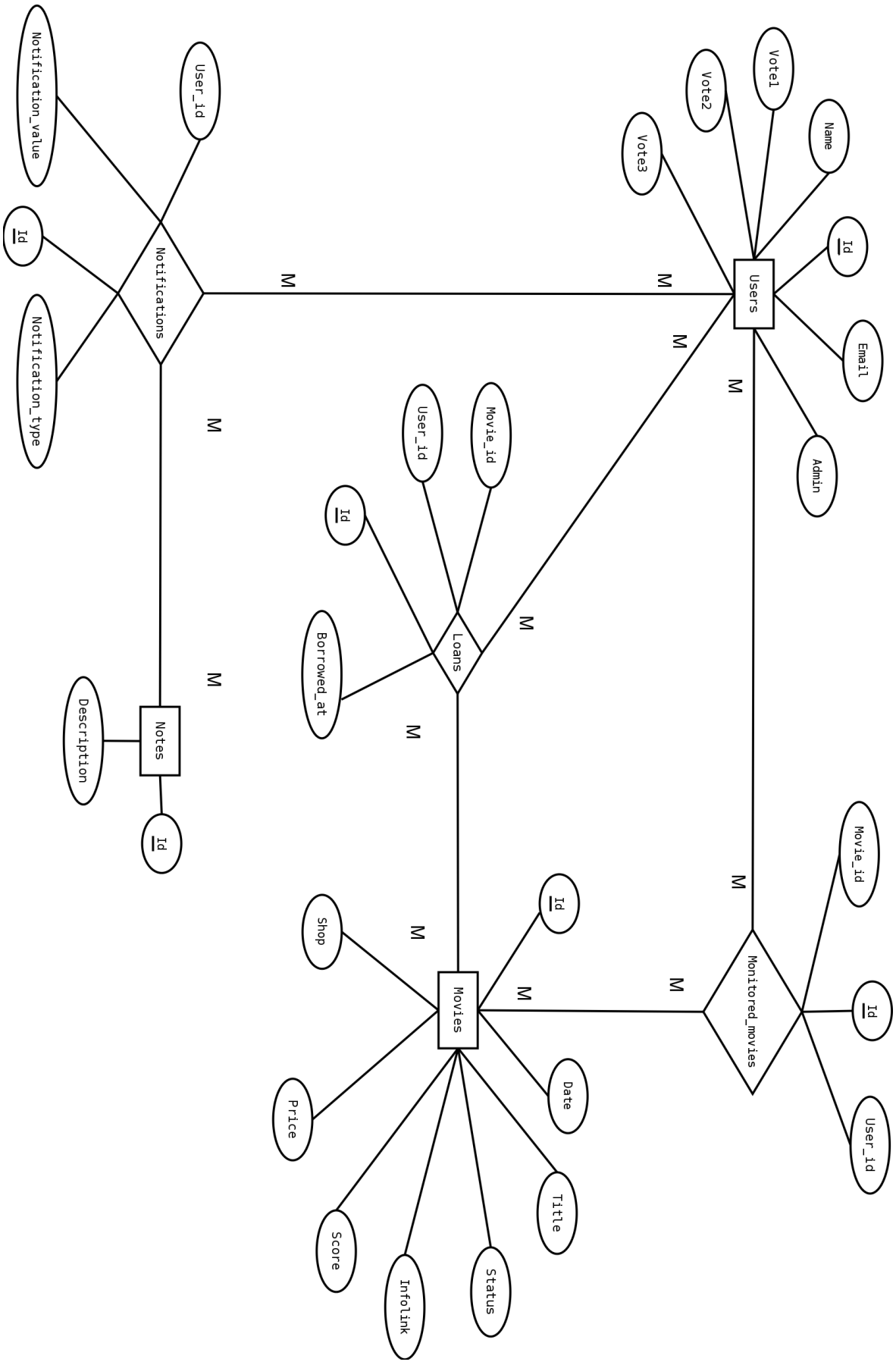
13 Ruby-kielen lisenssi. [www-sivu]. [viitattu 28.9.2005] Saatavissa: http://www.ruby-lang.org/en/LICENSE.txt

14 Ohjelmointikielten vertailutaulukko. [www-sivu]. [viitattu 28.9.2005] Saatavissa: www.approximity.com/ruby/Comparison_rb_st_m_java.html

15 Web application - Wikipedia, the free encyclopedia. [www-sivu]. [viitattu 2.10.2005] Saatavissa: http://en.wikipedia.org/wiki/Web_application

16 Web service - Wikipedia, the free encyclopedia. [www-sivu]. [viitattu 2.10.2005] Saatavissa: http://en.wikipedia.org/wiki/Web_service


ActionController::Base
MovieController
- @items :
- @not_borrowed :
- @borrowed :
- @wished :
- @reserved :
- @params :
- @my_loans :
- @all_loans :
- @selected_notifications :
- @monitored_movies :
- @my_reserved_movies :
- @my_reservations :
- @privileged_borrower :
- @recipient_list :
- @item1 :
- @item2 :
- @item3 :
- @item4 :
- @borrower :
+ index()
+ login()
+ add_movie()
+ return_movie()
+ borrow_movie()
+ borrow_reserved_movie()
+ edit_votes()
+ update_notifications()
+ monitor_movie()
+ reserve_movie()
ApplicationController
AdminNoteController
AdminMovieController
AdminNotificationController
AdminUserController
AdminMovie
AdminNote
AdminNotification
Notification
AdminUser
ActiveRecord::Base
SelectedNotification
User
+ find_me()
Movie
+ find_not_borrowed()
+ find_borrowed()
+ find_wished()
Loan
+ find_all_loans()
+ find_my_loans()
MonitoredMovie
+ find_reserved()


Notification_value
User_id
Id
Notifications
Notification_type
Vote1
Vote2
Vote3
Name
Id
Email
Users
Admin
M
User_id
Movie_id
Id
Loans
Borrowed_at
Description
Notes
Id
Movie_id
Id
Monitored_movies
User_id
Shop
Id
Movies
Date
Price
Title
Score
Infolink
Status