# Suunitteludokumentti

MovieLib

Jaakko Lukkari
jaakko.lukkari@gmail.com
Ohjelmoinninharjoitustyö
24.4.2010
Helsingin Yliopisto Avoin Yliopisto

# Sisällysluettelo

# Johdanto

Tämä dokumentti käsittelee ohjelmoinnninharjoitustyö kurssille tuotettavan MovieLib sovelluksen vaatimusmäärittelyä ja arkkitehtuurisuunitelmaa. Dokumentti sisältää yleiskuvauksen sovelluksesta ja sen toiminnasta, tietosisällön kuvauksen, kuvauksen sovelluksen toiminnan kannalta merkittävimmistä luokista ja tiedot käytettävistä tekniikoista ja standardeista.

## *Käsitteet*

Alle oleva taulukko sisältää selitykset tässä dokumentissa esiintyvistä käsitteistä.

| | |
|---|---|
| VLC | VLC media player on avoin ja alustariippumaton mediasoitin |
| MVC | Sovelluksen suunnittelussa käytetty suunnittelumalli. |
| TMDb | Yhteisövoimin ylläpidetty elokuvatietokanta. Palvelusta on mahdollista hakea kaikki tiedot liittyen elokuvaan ja sen näytteöijöihin |
| SQLite | Pieni, suoraan sovellukseen linkitettävä relaatiokanta. Käytetään monissa eri sovelluksissa. Arvioiden mukaan maailman käytetyin tietokanta. |
| DAO | DAO eli Data Access Object. Palauttaa olioina pyydettyjä tietokannassa olevia tietoja. |

# Yleiskuvaus

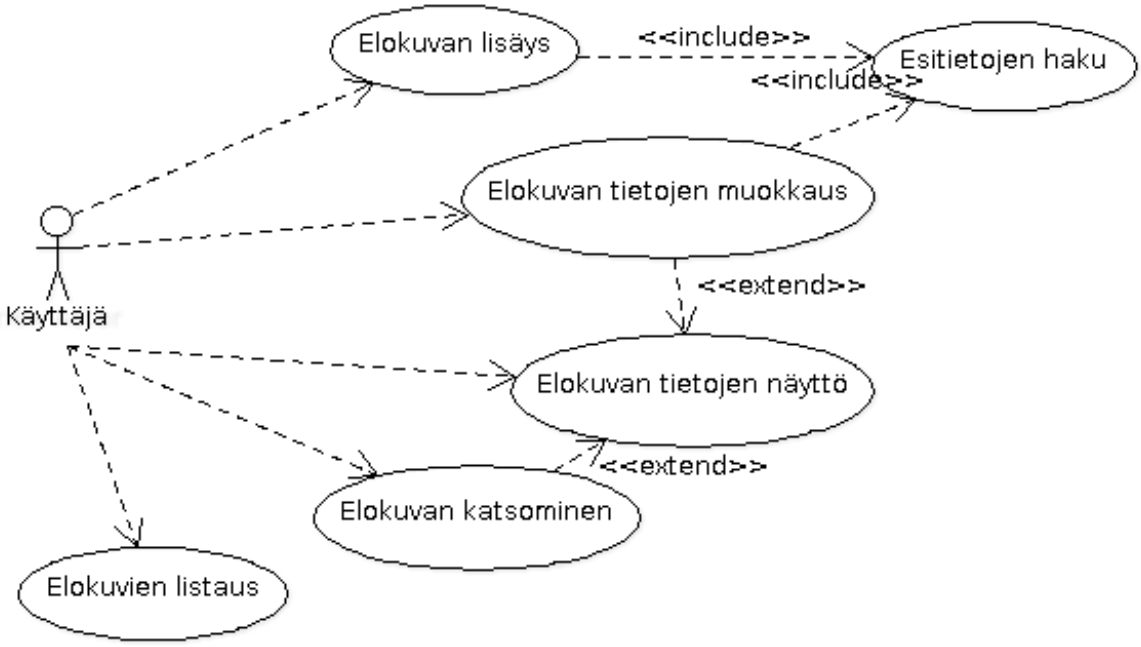
MovieLib on elokuvien arkistointiin ja arkistoitujen elokuvien katsomiseen tarkoitettu sovellus. Sovellus säilöö käyttäjän lisäämän elokuvan tiedot tietokantaan ja kopioi videotiedoston, tekstitykset ja kansikuvan omaan tietovarastoonsa. Käyttäjä pystyy selaamaan ja katsomaan arkistossa olevia elokuvia kokoruutuun aukeavalla käyttöliittymällä. Elokuvien lisäys, poisto ja muokkaus suoritetaan erillisen käyttöliittymän kautta. Sovelluksen jatkokehitykseen varaudutaan toteuttamalla järjestelmä mahdollisimman modulaarisesti ja toteuttumalla sekä malli että näkymä rajapintojen avulla.

# Vaatimukset

Tässä osiossa käsitellään MovieLib -sovelluksen vaatimusmäärittelyä.

## *Käyttötapaukset*

- **Esitietojen haku:** Elokuvan tietojen haku TMDb -palvelusta.
- **Elokuvan lisäys:** Käyttäjä lisää elokuvan graafisen käyttöliittymän kautta. Tapaus sisältää myös elokuvan esitietojen haun.
- **Elokuvan tietojen muokkaus:** Käyttäjä muokkaa järjestelmässä jo olevan elokuvan tietoja. Tapaus sisältää myös elokuvan esitietojen haun ja laajentaa elokuvan tietojen näyttöä.
- **Elokuvan tietojen näyttö:** Käyttäjä tarkastelee järjestelmässä olevan elokuvan tietoja.
- **Elokuvan katsominen:** Käyttäjä katsoo järjestelmässä olevan elokuvan. Tapaus laajentaa elokuvan tietojen näyttö tapausta.
- **Elokuvien listaus:** Käyttäjä listaa järjestelmässä olevat elokuvat.




*Kuva 1: Käyttötapauskaavio*

## *Toiminnalliset vaatimukset*

Toiminnallisiin vaatimuksiin on listattu vaatimukset sovelluksen toiminnoilta.

1. **Elokuvien lisäys** – Sovelluksen elokuvakirjastoon pystyy lisäämään elokuvia. Elokuvien lisääminen tapahtuu yksinkertaisen vaiheittaisen wizard -tyylisen käyttöliittymän kautta. Elokuvien tiedotoja ja kuvia sovellus osaa itse ehdoittaa hakemalla ne TMDb -palvelusta elokuvan nimen perusteella.
2. **Elokuvien listaus –** Sovelluksen elokuvakirjastossa olevat elokuvat pystyy listaamaan, joko nimen, näyttelijän tai muun tuotantohenkilön, ohjaajan, kategorian tai tuotantovuoden mukaisesti. Muokkauskäyttöliittymässä tiedot listataat ainoastaan aakkosjärjestyksessä.
3. **Elokuvan tietojen näyttö –** Sovelluksen elokuvakirjastossa olevan elokuvan tietoja pystyy tarkastelemaan niin videoiden toistoon kuin muokkaamiseen tarkoitetut käyttöliittymän kautta.
4. **Elokuvan tietojen muokkaus** - Sovelluksen elokuvakirjastossa olevan elokuvan tietoja pystyy muokkaamaan muokkaamiseen tarkoitetut käyttöliittymän kautta.
5. **Elokuvan poisto –** Sovelluksen elokuvakirjastossa olevan elokuvan pystyy poistamaan muokkauskäyttöliittymän kautta.
6. **Elokuvien katsominen –** Sovelluksen elokuvakirjastossa olevan elokuvan pystyy katsomaan elokuvien katsomiseen tarkoitetut käyttöliittymän kautta. Käyttäjä pystyy valitsemaan elokuvalle haluamansa tekstit tai katsomaan mahdollisen trailer -videon ennen elokuvan katsomista.

## *Ei-toiminnalliset vaatimukset*

Ei-toiminnallisiin vaatimuksiin on listattu sovelluksen toimintojen ulkopuolelle jäävät vaatimukset.

1. **Suoritusympäristö** - Sovelluksen suoritysympäristönä toimii Java 6 j a Windows tai Linux

käyttöjärjestelmä. Sovellus vaatii toimiakseen vlc -kirjastot.

2. **Suorituskyky** - Sovellukseen tulee pystyä arkistoimaan ainakin 500 elokuvaa ilman, että sillä on näkyviä vaikutuksia sovelluksen suorituskykyyn tai muuhun toimintaan.
3. **Käytettävyys** - Sovelluksen käyttökynnys olla pieni ja normaalin käyttäminen onnistuu ilman sovelluksen ulkopuolista ohjeistusta. Käyttöliittymissä pyritään yksinkertaisuuteen.
4. **Ulkoasu** - Sovellus on ulkoasultaa pelkistetty mutta tyylikäs.

# Pääkomponentit

Tässä osiossa tarkastellaan järjestelmän pääkomponetteja ja niitä edustavia luokkia. Sovellus koostuu kolmesta pääkomponentista, joita ovat malli, näkymä ja ohjain. Sovelluksen käynnistyessä asetuksissa määritetyt tai parametreina annetut malli ja näkymä ladataan sovellukseen tehdyn luokkalataimen avulla. Tämän jälkeen malli ja näkymä annetaan ohjaimelle, joka asetaa näkymälle tarvittavat kuuntelijat. Arkkitehtuurin ja pääkomponenttien osalta noudatetaan MVC -suunittelumallia.

## *Malli*

Malli on tietorakenneluokka ja toimii sovelluksen tietolähteenä. Mallin on yhteydessä tietokantaan DAO -luokan kautta. Malli sisältää myös hieman normaalista MVC -arkkitehtuurista poiketen sovelluslogiikkaan liittyvää toiminnallisuutta. Tämä siksi, että sovelluksen tietosisältö ja sen hallinta on helposti vaihdettavissa toiseen mallin rajapintaa toteuttavaan luokkaan. Sovelluksen yleisluonteen takia mallin ei myöskään tarvitse informoida ohjainta mallissa tapahtuneista muutoksista.

- **MovieModel** – Rajapinta tietolähteitä varten.
- **MovieModelImpl –** MovieModel -rajapinnan toteuttava luokka, joka hakee ja säilöö elokuvan tiedot SQLite -tietokantaan ja siirtää elokuvaan liittyvät media -tiedostot niille varattuihin kansioihin.

### MovieModel -rakenne

```
/**
 * Rajapinta, jonka toteuttavia luokkia voidaan käyttää sovelluksen
 * tietolähteinä.
 *
 */
public interface MovieModel {

	/**
	 * Elokuvatietojen kenttiä edustava enumeraali
	 */
	public enum Field {
		ID, TITLE, DESCRIPTION, SLOGAN, DIRECTOR, TRAILER_URL, YEAR, RATING, ADDED, GENRE,
		CAST
	}

	/**
	 * Kansi ja taustakuvan kokoja edustava enumeraali
	 */
	public enum ImageSize {
		SMALL, MEDIUM, LARGE
	}

	/**
```

```
     * @return Palauttaa tietolähteen nimen
     */
    public String getModelName();

    /**
     * Poistaa elokuvan tietovarastosta.
     *
     * @param movie
     *            poistettava elokuva
     * @return totuusarvo siitä onnistuiko poisto
     */
    public boolean deleteMovie(Movie movie);

    /**
     * Palauttaa id:tä vastaavan elokuvan tiedot
     *
     * @param id
     *            elokuvan id
     * @return Movie -luokan ilmentymä täytettynä elokuvan tiedoilla.
     */
    public Movie getMovie(int id);

    /**
     * Elokuvien haku.
     *
     * @param searchField
     *            Kenttä, jonka perusteella haku tapahtuu
     * @param search
     *            Haettava sisältö
     * @param limit
     *            Vastauksien raja
     * @param page
     *            Monenneltako sivulta vastaukset halutaan
     * @param orderBy
     *            Kenttä, jonka perusteella vastaukset järjestetään
     * @param desc
     *            Järjestetäänkö laskevassa järjestyksessä
     * @param includeCast
     *            Liitetäänkö vastauksiin myös näyttelijät ja muu tuotantoryhmä
     * @param includeGenres
     *            Liitetäänkö vastauksiin myös genret
     * @param includeSubtitles
     *            Liitetäänkö vastauksiin myös tekstitykset
     * @return QueryResult instanssi, joka sisältää tiedot käytetyistä
     *         hakuehdoista ja haun vastauksen.
     */
    public QueryResult<Movie> getMovies(Field searchField, String search,
                int limit, int page, Field orderBy, boolean desc,
                boolean includeCast, boolean includeGenres, boolean includeSubtitles);

    /**
     * Lisää elokuvan tietovarastoon.
     *
     * @param movie
     *            Elokuvan tiedot movie -oliossa.
     * @param cover
     *            Elokuvan kansikuva
     * @param backdrop
     *            Elokuvan taustakuva
     * @return Lisätyn elokuvan id.
     */
    public int saveMovie(Movie movie, BufferedImage cover,
                BufferedImage backdrop);

    /**
```

```
     * Palauttaa elokuvan taustakuvan halutussa koossa
     *
     * @param movie
     *            halutun elokuvan tiedot
     * @param size
     *            koko
     * @return BufferedImage elokuvan taustakuvasta
     */
    public BufferedImage getBackdropForMovie(Movie movie, ImageSize size);

    /**
     * Palauttaa elokuvan kannen halutussa koossa
     *
     * @param movie
     *            halutun elokuvan tiedot
     * @param size
     *            koko
     * @return BufferedImage elokuvan kannesta
     */
    public BufferedImage getCoverForMovie(Movie movie, ImageSize size);

    /**
     * @return onko tietolähteen elokuvia salittu muokattavan
     */
    public boolean allowEdit();

    /**
     * @return onko tietolähteeseen salittu elokuvien lisääminen
     */
    public boolean allowImport();

    /**
     * @return onko tietolähteestä sallittu elokuvien poistaminen
     */
    public boolean allowDelete();

    /**
     * @return kaikki tietolähteessä olevat näyttelijät ja muut tuotantoryhmän
     *         jäsenet
     */
    public ArrayList<String> getAllCast();

    /**
     * @return kaikki tietolähteessä olevat ohjaajat
     */
    public ArrayList<String> getAllDirectors();

    /**
     * @return kaikki tietolähteessä olevat genret
     */
    public ArrayList<String> getAllGenres();

    /**
     * @return kaikki tietolähteessä olevat tuotantovuodet
     */
    public ArrayList<String> getAllYears();

}
```

## *Näkymä*

Näkymä edustaa sovelluksen käyttöliittymiä. Mallin tavoin näkymänä voi toimia mikä tahansa MovieView -rajanpinnan toteuttava luokka. Sovelluksen määrittelyn mukaisesti toteutettavia näkymiä ovat elokuvien katsomiseen tarkoitettu näkymä ja elokuvien muokkaamisen tarkoitettu

näkymä.

- **MovieView –** Rajapinta käyttöliittymiä varten
- **MovieViewImpl –** MovieView rajanpinnan totetuttava luokka, joka on elokuvien katsomiseen tarkoitettu käyttöliittymä.
- **MovieEditorViewImpl** - MovieView rajanpinna totetuttava luokka, joka on elokuvien muokkaamiseen ja lisäämiseen tarkoitettu käyttöliittymä.

### MovieView -rakenne

```
public interface MovieView {

        // KUUNTELIJOIDEN ASETUS
        /**
         * Asettaa kuuntelijan joka tarkkailee sovelluksen sulkupyyntöjä
         *
         * @param listener
         */
        public void setCloseListener(CloseListener listener);

        /**
         * Asettaa kuuntelijan joka tarkkailee elokuviin liittyviä tapahtumia
         *
         * @param listener
         */
        public void setMovieListener(MovieListener listener);

        /**
         * Asettaa kuuntelijan joka tarkkailee näyttelijöiden/tuotantotiimin
         * tapahtumia
         *
         * @param listener
         */
        public void setCastListener(GroupListener listener);

        /**
         * Asettaa kuuntelijan joka tarkkailee ohjaajien tapahtumia
         *
         * @param listener
         */
        public void setDirectorListener(GroupListener listener);

        /**
         * Asettaa kuuntelijan joka tarkkailee vuosien
         *
         * @param listener
         */
        public void setYearListener(GroupListener listener);

        /**
         * Asettaa kuuntelijan joka tarkkailee genrejen tapahtumia
         *
         * @param listener
         */
        public void setGenreListener(GroupListener listener);

        // NÄYTÄ

        /**
         * Pyytää käyttöliittymää näyttämääm parametrinä annetun listan
         * näyttelijöistä ja muista tuotannon jäsenistä
         *
         * @param cast
         *            lista näyttelijöistä tai vst.
         */
        public void showCast(ArrayList<String> cast);

        /**
         * Pyytää käyttöliittymää näyttämääm parametrinä annetun listan genreistä
         *
```

```
     * @param genres
     *            lista genrejä
     */
    public void showGenres(ArrayList<String> genres);

    /**
     * Pyytää käyttöliittymää näyttämääm parametrinä annetun listan ohjaajia
     *
     * @param directors
     *            lista ohjaajia
     */
    public void showDirectors(ArrayList<String> directors);

    /**
     * Pyytää käyttöliittymää näyttämääm parametrinä annetun listan
     * tuotantovuosista
     *
     * @param years
     *            lista tuotantovuosia
     */
    public void showYears(ArrayList<String> years);

    /**
     * Pyytää käyttöliittymää näyttämääm parametrinä annetun listan elokuvista
     *
     * @param movies
     *            lista elokuvia
     */
    public void showMovies(ArrayList<Movie> movies);

    /**
     * Pyytää käyttöliittymää näyttämääm parametrinä annetun elokuvan teidot
     *
     * @param movie
     *            elokuva
     */
    public void showMovie(Movie movie);

    /**
     * Pyytää käyttöliittymää näyttämään esikatselmuksen elokuvista joissa
     * näyttelijä tai vst. on
     *
     * @param movies
     *            ote ryhmään kuuluvista elokuvista
     */
    public void showCastPreview(ArrayList<Movie> movies);

    /**
     * Pyytää käyttöliittymää näyttämään esikatselmuksen elokuvista jotka
     * kuuluvat genreen
     *
     * @param movies
     *            ote ryhmään kuuluvista elokuvista
     */
    public void showGenrePreview(ArrayList<Movie> movies);

    /**
     * Pyytää käyttöliittymää näyttämään esikatselmuksen elokuvista jotka on
     * ohjannut parametrinä annettu ohjaaja
     *
     * @param movies
     *            ote ryhmään kuuluvista elokuvista
     */
    public void showDirectorPreview(ArrayList<Movie> movies);

    /**
     * Pyytää käyttöliittymää näyttämään esikatselmuksen elokuvista jotka on
     * tuotettu kyseisenä
     *
     * @param movies
     *            ote ryhmään kuuluvista elokuvista
     */
    public void showYearPreview(ArrayList<Movie> movies);

    // Käynnistyts

    /**
```

```
         * Kertoo käyttöliittymälle, että kaikki muu on valmista
         */
        public void init();

        // VIRHEET

        /**
         * Näyttää virheilmoitukset
         *
         * @param error
         *            näytettävä virheilmoitus
         */
        public void showError(String error);

        // Lataus
        /**
         * Näyttää käyttäjälle että latautuminen tapahtuu
         *
         * @param reason
         */
        public void showLoading(String reason);

        /**
         * Piiloittaa latautumis indikaattorin
         */
        public void hideLoading();

        // Tiedot

        /**
         * @return kyseisen näkymän/ulkoasun nimi
         */
        public String getViewName();

        /**
         * @return valittuna oleva genre
         */
        public String getGenre();
        /**
         * @return valittuna oleva näyttelijä tai muu tuotantoryhmän jäsen
         */
        public String getCast();
        /**
         * @return valittuna oleva ohjaaja
         */
        public String getDirector();
        /**
         * @return valittuna oleva vuosi
         */
        public String getYear();
        /**
         * @return valittuna oleva elokuva
         */
        public Movie getMovie();


}
```

## *Käsittelijä/Ohjain*

Käsittelijän/ohjaimen tehtävänä on toimia käyttäjän käskyjen perusteella ja muuttaa mallia ja/tai näkymää niiden mukaisesti. Ohjain ei itsessään sisällä toiminnallisuutta vaan toimii eräänlaisena siltana mallin ja näkymän välillä. Tämä mahdollistaa vaivattomamman jatkokehityken ja käyttöliittymän tai tietolähteen vaihtamisen.

### Ohjaimen julkinen rakenne

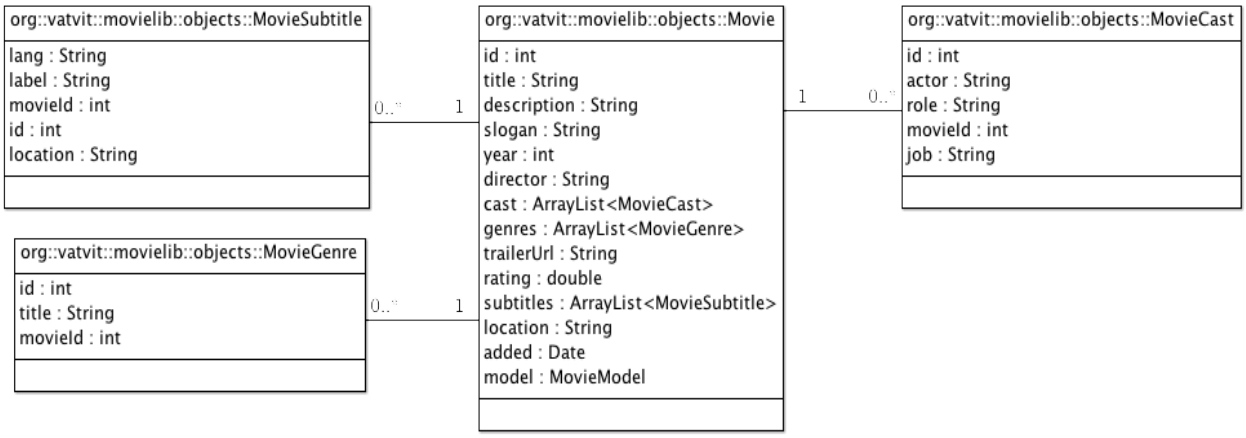
```
public MovieController(MovieModel model, MovieView view);
public MovieModel getModel();
public void setModel(MovieModel model);
public MovieView getView();
```

```
public void setView(MovieView view);
```

# Tietosisältö

Tässä osiossa käsitellään sovelluksen tietosisältöä.

Sovellus on tietosisällöltään yksinkertainen. Tietokantaan säilötään elokuvan (**Movie**) tiedot, elokuvan näyttelijöiden (**MovieCast**) tiedot, elokuvan luokitukset (**MovieGenre**) sekä elokuvana liitetyt tekstitykset (**MovieSubtitle**).

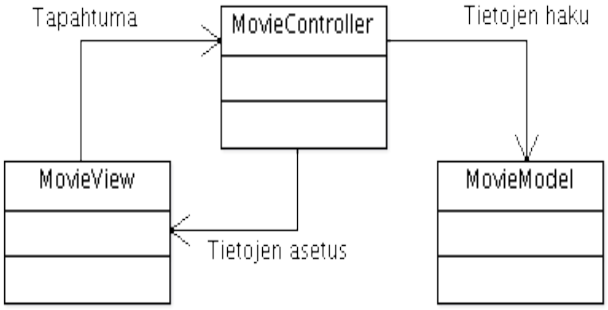



*Kuva 2: Tietosisältö*

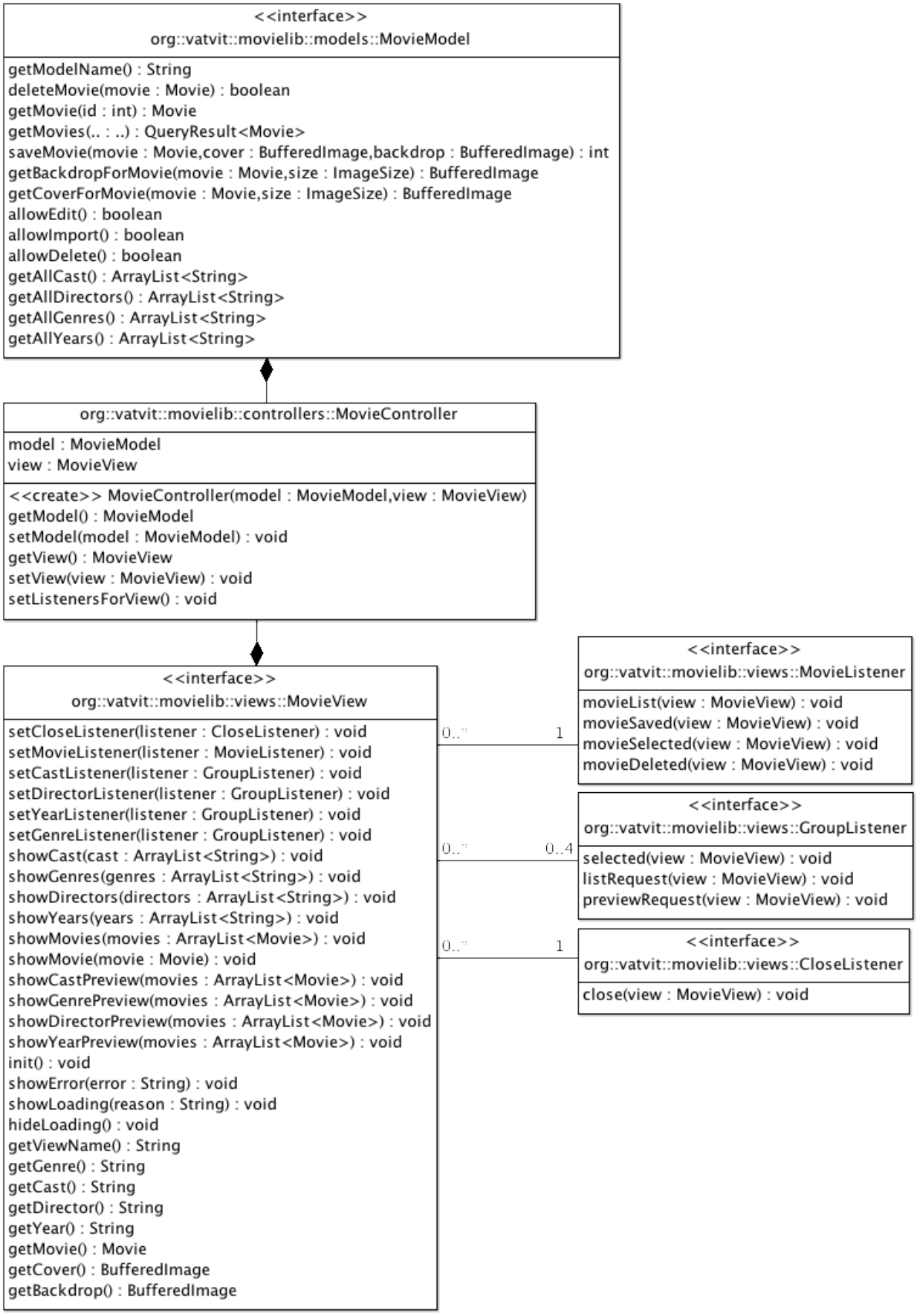
# Arkkitehtuurikuvaus

Tässä kappaleessa käsitellään sovelluksen arkkitehtuuria.

Sovelluksen arkkitehtuurimalliksi on valittu MVC. Malli valittiin sen tuoman muokattavuuden ja modulaarisuuden takia. Modulaarisuuden maksimoimiseksi MVC:n toteuttamiseksi valittiin niin sanottu välittäjästrategia (Mediator Strategy), jossa mallin ja näkymän toiminta ei ole sidottu toisiinsa tai sovelluksen muuhun toimintaan vaan tiedonvälitys tapahtuu ohjaimen asettamien kuuntelijoiden kautta.




*Kuva 3: MVC -arkkitehtuuri*

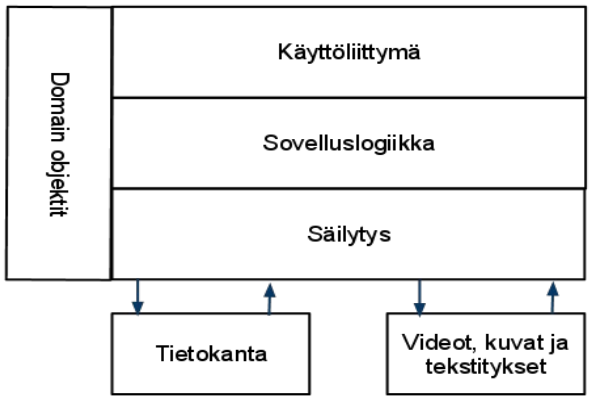



*Kuva 4: Luokkakaavio ohjaimen (MovieController) ja mallin (MovieModel) ja näkymän*

*(MovieView) välisistä riippuvuuksista.*

## *Kerrokset*

Arkkitehtuurissa on pyritty kerrosmaisuuteen. Kerrokset hyödyntävät aina alemman kerroksen tarjoamia palveluita. Kerroksisuus parantaa sovelluksen muokattavuutta ja selkeyttää vaihdettavien moduulien rakennetta. Kerroksine rakenne estää myös ylimääräisten riippuvuuksien syntymisen kerrosten välille.




*Kuva 5: Arkkitehtuurin kerroskuvaus*

## Käyttöliittymäkerros

Käyttöliittymäkerros sisältää sovelluksen käyttöliittymät. Käyttöliittymissä tapahtuvat tapahtumat välitetään sovelluslogiikka kerrokselle sovelluslogiikkakerroksen asettamien kuuntelijoiden avulla. Käyttöliittymäkerros koostuu **MovieView -rajapinnasta** ja sen toteuttavista käyttöliittymistä **MovieViewImpl** ja **MovieEditorViewImpl**.

## Sovelluslogiikkakerros

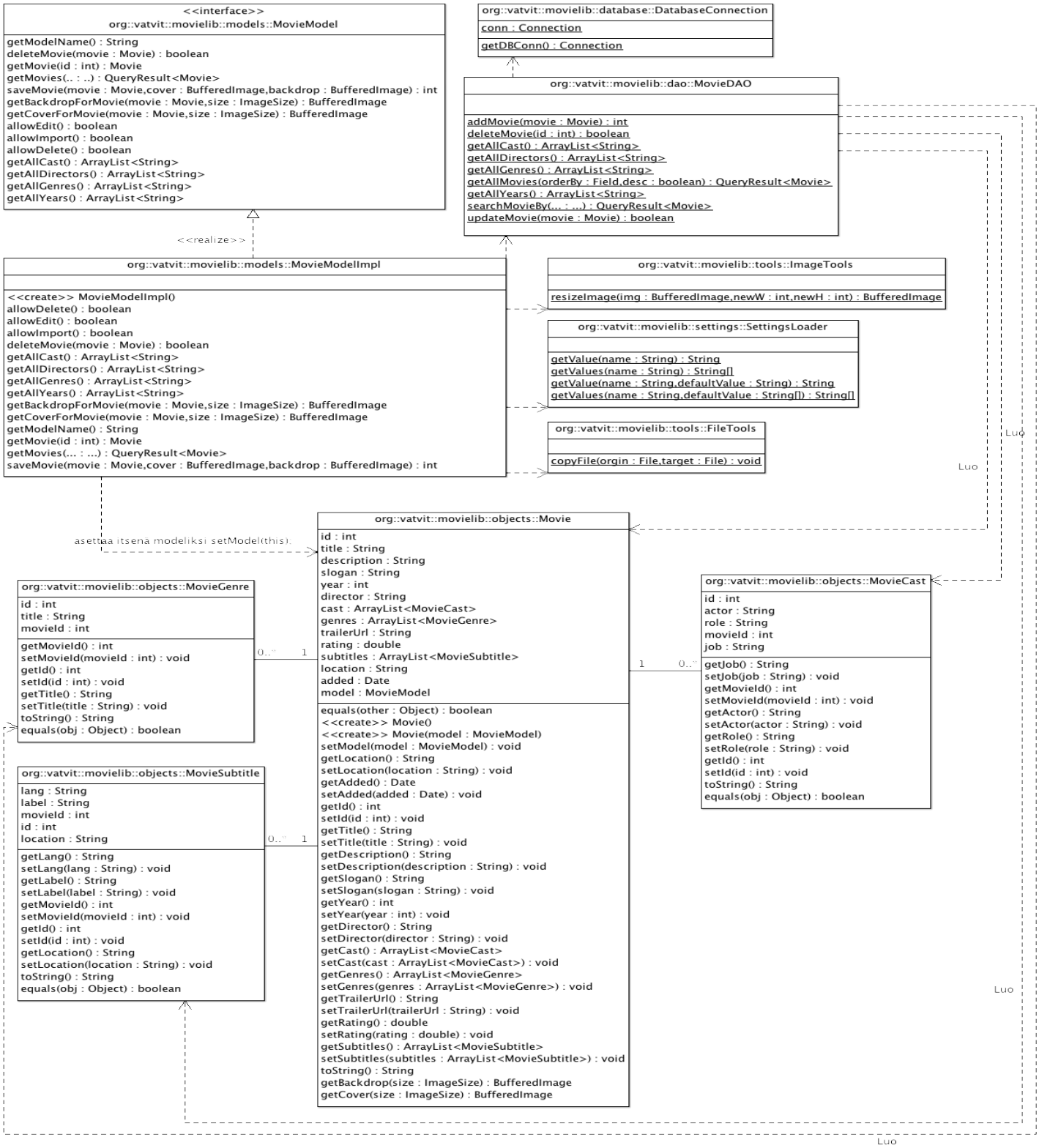
Sovelluslogiikkakerros sisältää sovelluksen käyttöliittymän ja säilytyskerroksen yhdistävät toiminnot. Yleisestä käytännöstä poiketen monet sovelluslogiikkakerroksen toiminnot on siirretty säilytyskerroksen toteutettavaksi. Tähän on päädytty, jotta sovelluksen riippuvuus käytetystä tietolähteestä saadaan rajattua säilytyskerrokseen. Sovelluslogiikkakerros koostuu **MovieController -luokasta**.

## Säilytyskerros

Säilytyskerros sisältää sovelluksen tietovarastojen hallintaan tarkoitetun toiminnallisuuden. Koska sovelluksen toiminta painottuu suurelta osin sovelluksen tietovarastojen selaamiseen ja hallitsemiseen, toteutetaan säilytyskerroksessa myös monet normaalisti sovelluslogiikkakerroksessa toteutettavat toimenpiteet liittyen elokuvien lisäyksen ja poiston yhteydessä tapahtuviin toimenpiteisiin.

Elokuvan perustiedot, näyttelijät, kategoriat ja tekstityksien tiedot säilötään SQLite -tietokantaan ja media-tiedostot (videot, kannet, taustat ja tekstit) säilötään, joko sovelluksen data-kansioon tai asetuksissa määritettyyn muuhun kansioon. Säilytyskerros koostuu **MovieModel -rajapinnasta** ja

sen toteuttavasta **MovieModelImpl**:sta.



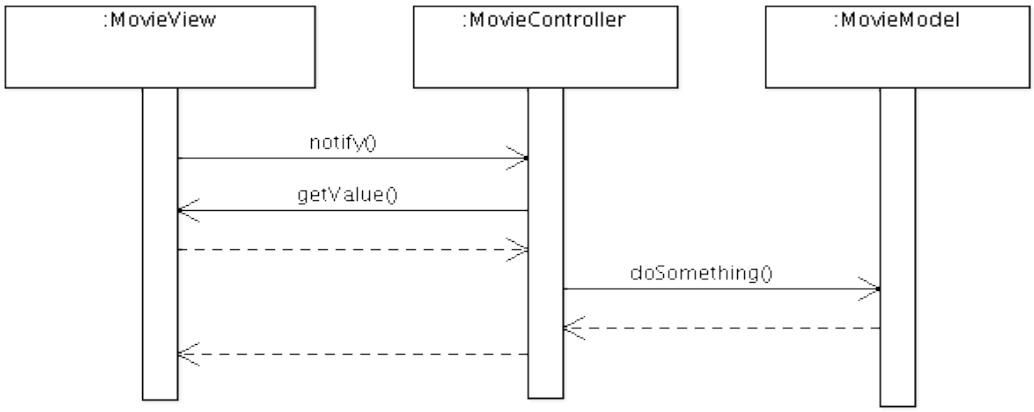

*Kuva 6: Mallin rajapinnan implementaation luokkakaavio. Luokkakaaviossa kuvattu vain julkiset metodit ja getMovie() -metodi on lyhennetty parametrien suuren lukumäärän takia.*

### *Esimerkit*

Tässä osiossa käsitellään sovelluksen toimintaan yleispätevän sekvenssikaavio avulla.

Sovelluksessa käytettävä MVC -arkkitehtuurimalli tekee kaikista sovelluksen toimenpiteistä rakenteeltaan samankaltaisia. Tästä syystä sovelluksen toiminnan kuvaamiseksi riittää yksi sekvenssikaavio.

Sovelluksen käyttöliitymässä **(MovieView)** tapahtuneesta tapahtumasta ilmoitetaan ohjaimelle (**MovieController**) ohjaimen asettaman tapahtumakohtaisen kuuntelijan kautta. Tämän jälkeen ohjain hakee käyttöliittymältä tarvitsemansa tiedot. Saatuja tietoja käyttäen ohjain keskustelee mallin (**MovieModel**) kanssa. Mallin palauttamien tietojen perusteella ohjain muuttaa käyttöliittymää vastaamaan mallin tilaa.




*Kuva 7: Sekvenssikaavio sovelluksen toiminnasta*

## Käyttöliittymä

Tässä kappaleessa käsitellään sovelluksen käyttöliittymiä.

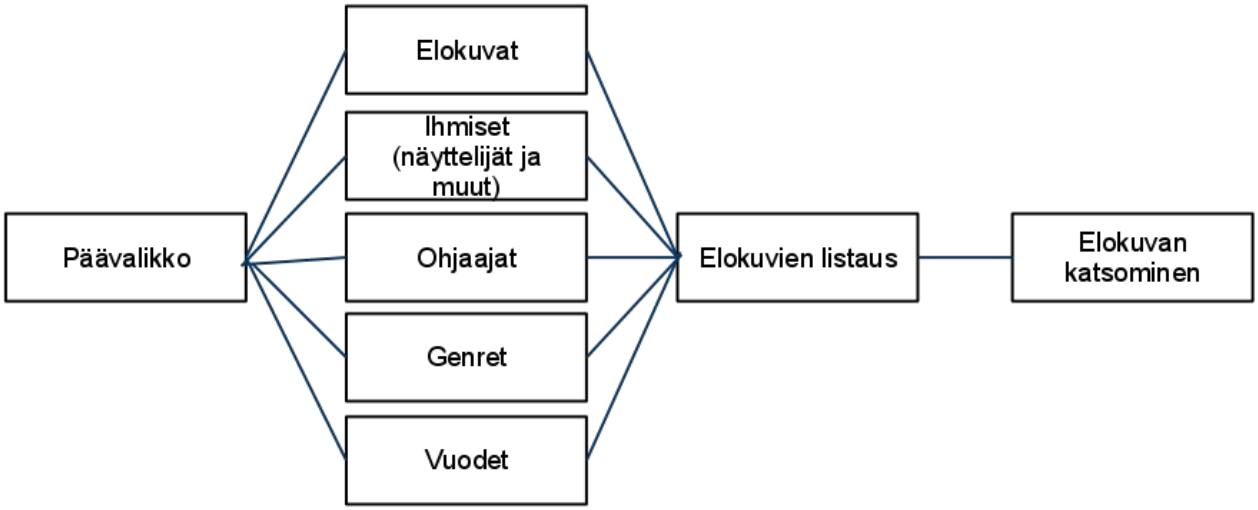
Sovelluksen käyttöliittymänä voi toimia mikä tahnsa **MovieView -rajapinnan** toteuttava luokka. Sovelluksen määrittelyn mukaisesti sovellukselle toteutetaan yksi elokuvien katsomista varten tarkoitettu käyttöliittymä **(MovieViewImpl)** ja yksi elokuvien muokkaamista ja lisäämistä varten tarkoitettu käyttöliittymä **(MovieEditorViewImpl)**.

### *MovieViewImpl*

**MovieViewImpl** on kokoruutuun aukeva elokuvien selaamista ja katsomista varten tarkoitettu käyttöliittymä. **MovieViewImpl** laajentaa Swingin Jframea. Kokoruutu tilassa hyödynnetään Javan uusi kokoruutu -toimintoja ja elokuvien katsomiseen käytetään VLC -kirjasto.

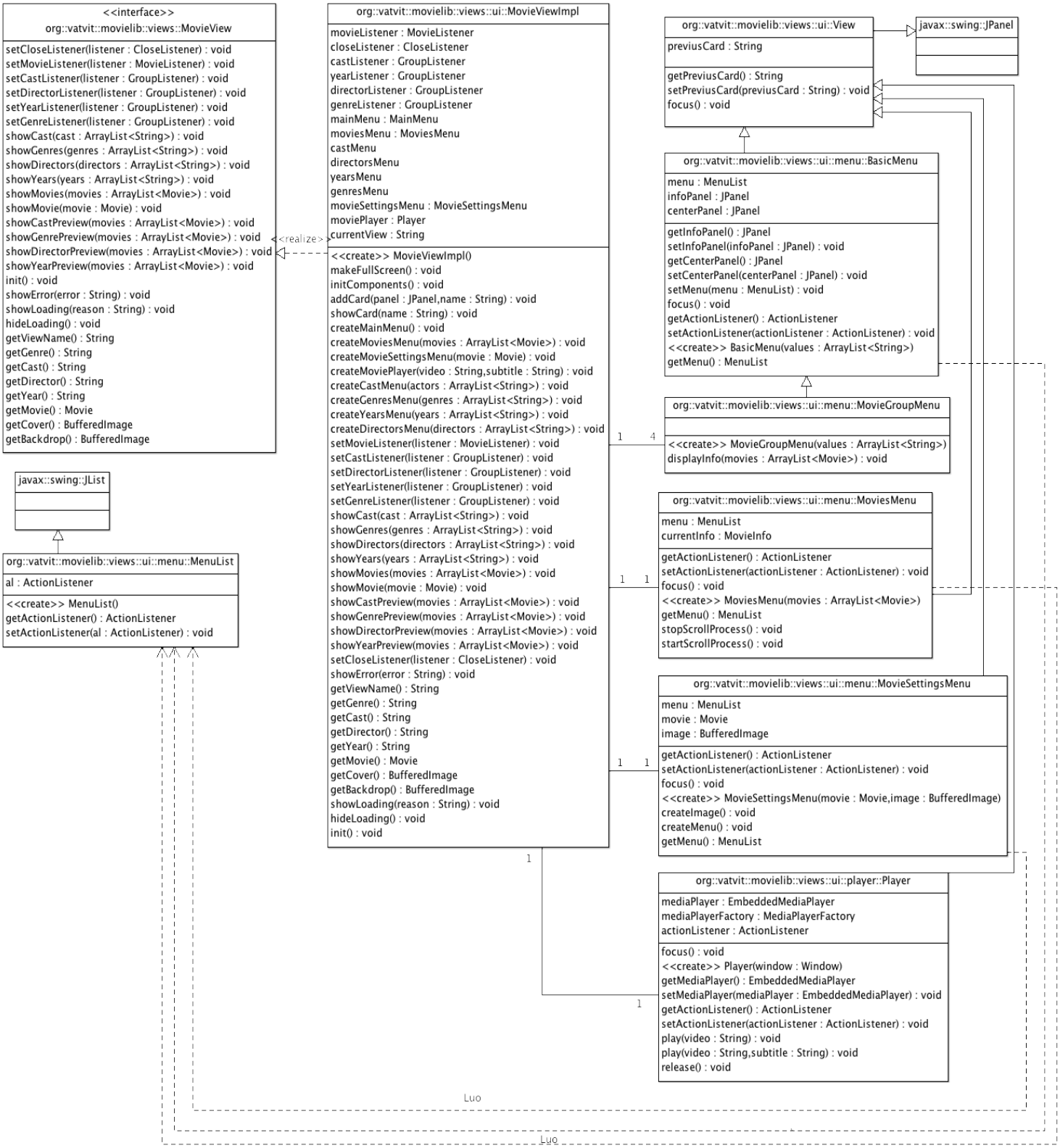
Käyttöliittymän valikoissa liikkuminen tapahtuu nuolinäppäimillä. Valikon kohdan voi valita painamalla oikealle tai ENTER -painiketta. Takaisin palaaminen tapahtuu painamalla vasemmalle

tai ESC -painiketta. Valikoissa voi myös liikkua käyttäen hiirtä.




*Kuva 8: MovieViewImpl -valikkorakenne*

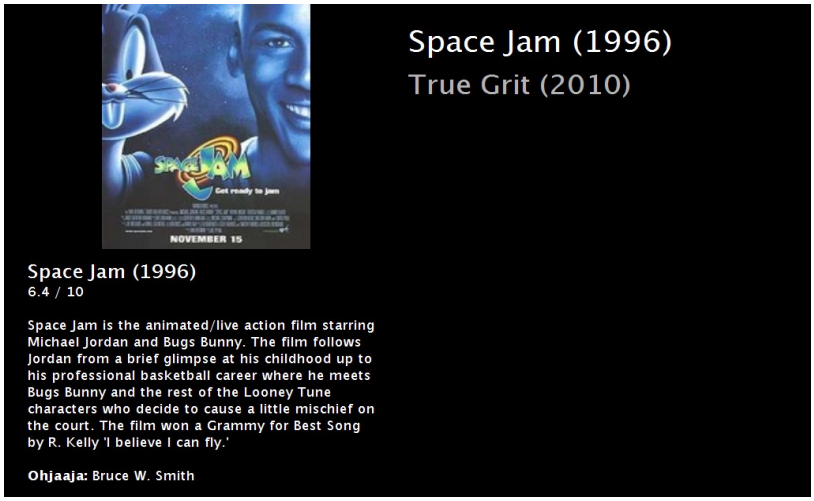
Valikkonäkymien pinoamisessa hyödynnetään **CardLayout**:ia, jonka avulla **Jpanel**it on kasattu päällekkäin niin, että vain haluttu valikko tai näkymä näytetään. Näkymät on jaettu kahteen osaan: lista/menu- ja info -puolleen. Jaossa käytetään **BoxLayout**:ia. Valikoiden keskittämiseen käytetään **GridBagLayout**:ia. Valikko toteutetaan laajentamalla **Jlisti**ä ominaisuuksilla, jotka mahdollistavat sen käyttämisen näppäimistön kautta tapahtuvassa navigoinnissa.



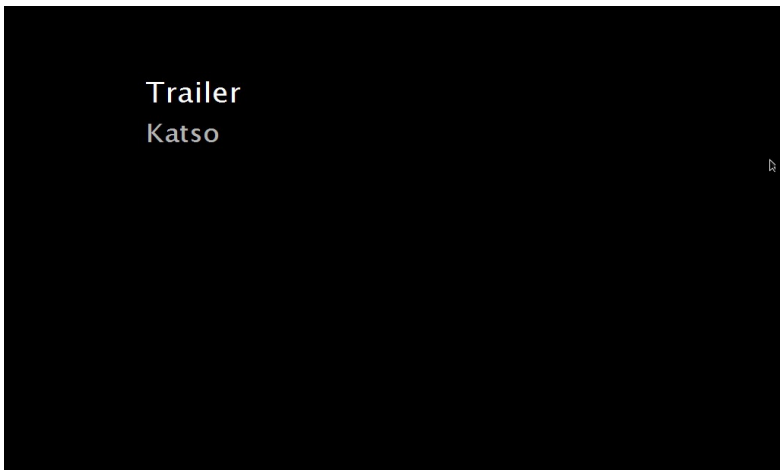

*Kuva 9: MovieViewImpl -käyttöliittymän luokkakaavio. Luokkakaaviossa kuvattu vain käyttöliittymää varten tehdyt komponentit. Swing -komponenttien (Jpanelin, JscrollPane ja layoutit) käyttö jätetty pois kaaviosta.*


*Kuva 10: Käyttöliittymän päävalikko. Vasemmalle puolelle ladataan valinnan nimeä vastaava kuva.*


*Kuva 11: Genre-valikko. Vasemmalla puolella näytetään kooste genreen kuuluvien elokuvien kansista. Muiden ryhmien (ohjaajat, näyttelijät ja vuodet) näkymissä käytetään samaa ulkoasua/valikkoa.*




*Kuva 12: Elokuva-listaus. Oikealla lista ryhmään kuuluvista elokuvista ja vasemmalla elokuvan tiedot. Elokuvan tiedot on asetettu **JscrollPane:n** sisälle. JscrollPane:sta tehdään automaattisesti ylöspäin vierivä omassa **thread**issa suoritettavan prosessin avulla. Tämä mahdollistaa suurienkin tietomäärien näyttämisen näkymässä.*



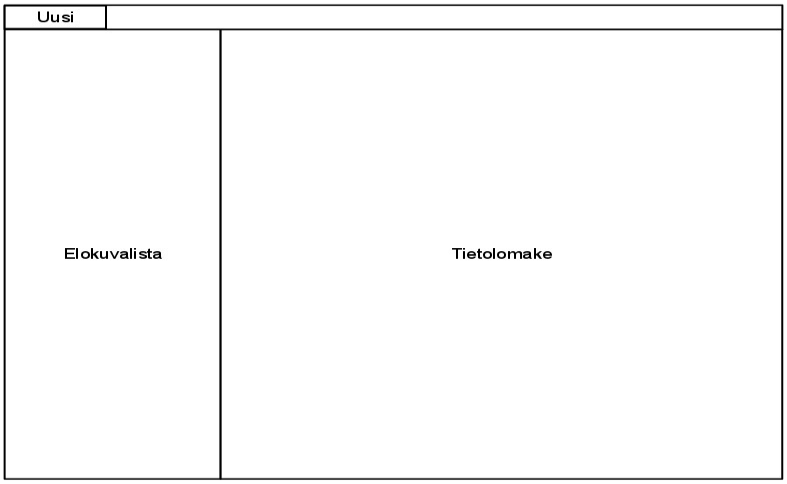

*Kuva 13: Elokuvan katsomisasetukset. Valikosta valitaan halutaanko elokuvasta katsoa esittelyvideo eli trailer tai millä tekstityksillä elokuva halutaan katsoa.*

*Kuva 14: Esittelyvideon katsominen. Näkymällä ei ole eroa elokuvien normaaliin katsomiseen. Videon kelaaminen tapahtuu nuolinäppäimillä ja toistotilasta voi poistua ESC -näppäimellä.*

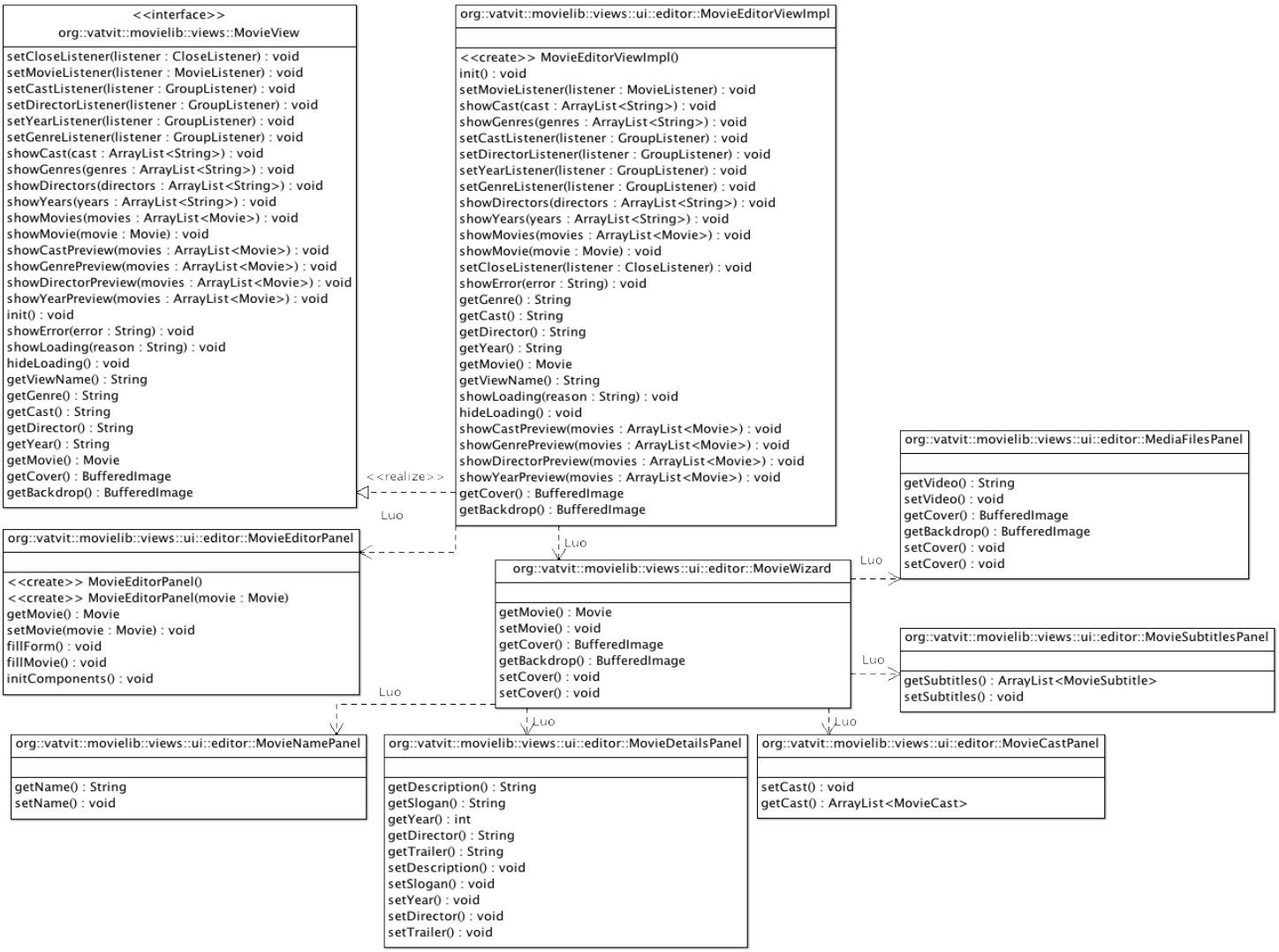
## *MovieEditorViewImpl*

**MovieEditorViewImpl** on elokuvien lisäämiseen ja muokkaamiseen tarkoitettu käyttöliittymä. Käyttöliittymä koostuu elokuvien listauksesta, valitun elokuvan tietoja esittävästä lomakkeesta ja wizard -tyylisestä uuden elokuvan lisäykseen tarkoitetusta näykmästä. Elokuvien listaus toteutetaan **Jlist**:llä. Lomake toteutetaan käytten kahta rinnakkaista **FlowLayout**ia ja uusien elokuvien lisäys näkymässä käytetään **CardLayout**ia.

Elokuvien lisäys tapahtuu vaiheissa alkaen elokuvan nimen täyttämisellä ja mahdollisilla esitietojen haulla **TMDb** -palvelusta. Koska esitiedoista voi tulla monia eri vastauksia on käyttäjän valittava saaduista vastauksista oikea elokuva. Tämän jälkeen käyttäjä siirtyy täyttämään tai tarkistamaan muut elokuvan perustiedot (slogan, tuotantovuosi, ohjaaja, kuvaus, trailer ja arvostelut). Seuraavaksi käyttäjä tarkistaa listauksen elokuvan näyttelijöistä ja muusta tuotantoryhmästä. Seuraava näkymässä asetetaan tai tarkitetaan elokuvan kansikuva ja taustakuva. Seuraavassa näkymässä asetetaan käyttäjän haluamat tekstitykset elokuvalle. Viimeisessä vaiheessa käyttäjä asettaa videotiedoston.




*Kuva 15: Päänäkymän asettelu*




*Kuva 16: MovieEditorViewImpl -käyttöliittymän luokkakaavio. Luokkakaaviossa kuvattu vain käyttöliittymää varten tehdyt komponentit.*

# Kolmannen osapuolen kirjastot

Kolmannen osapuolen komponentteja on käytetty niissä tilanteissa, joissa käyttötarkoitusta varten sopiva komponentti tai kirjasto on jo valmiiksi olemassa, eikä toimintojen uudelleen toteuttamisesta olisi käytettävyyden tai tehokkuuden kannalta merkittävää hyötyä.

## *SQLiteJDBC*

SQLite tietokannan ohjaamiseksi käytetään SQLiteJDBC ajuria. JDBC ajurin käyttäminen mahdollistaa tietokannan vaihtamisen esimerkiksi MySQL tietokannaksi vaivattomasti mikäli siihen on tulevaisuudessa tarvetta.

**Lisenssi:**
BSD
**Lisätietoa ajurista:**
http://www.zentus.com/sqlitejdbc/

## *JSON -kirjasto*

JSON eli JavaScript Object Notation on yksinkertainen tiedonsiirtomuoto. JSON -muotoista tietoa käytetään pääasiallisesti JavaScript sovelluksissa. JSON on myös yleisesti tuettu lukuisissa eri ohjelmointikielissä. JSON:ia MovieLib:ssä käytetään TMDb:stä saatavan tiedon parsimiseen ja asetustiedostoissa. JSON:n avulla on mahdollista tallentaa tietoa puumallisesti ja helppolukuistasti.

**Lisenssi:**
The JSON License
**Lisätietoa JSON:sta:**
http://www.json.org/

## *TMDb*

TMDb on yhteisövoimin ylläpidettävä elokuvatietokanta. Palvelusta on mahdollista hakea tiedot liittyen elokuvan perustietoihin, näyttelijöihin, kategorioihin ja kansikuviin. JTMDb on TMDb:n apin toteutus Java-kielelle.

**Käytetettävä versio:**

1.9.14

**Lisenssi:**

GNU Library or Lesser General Public License (LGPL)

## *Libvlc ja vlcj*

Sovellus käyttää videotiedostojen näyttämiseen vlc -kirjastoja. Kirjastojen hyödyntämiseen käytetään vlcj (Java sidos libvlc:lle).

**Lisenssit:**

libvlc ja vlc - GNU GPL v3

vlcj - GNU GPL v3

**Lisätietoa:**

vlc - http://www.videolan.org/

vlcj - http://code.google.com/p/vlcj/