**Object Design Document**

1. **Introduzione**
   1. **Object Design trade-offs**

**Manutenibilità:** Abbiamo utilizzato un’architettura chiusa, ovvero ogni livello può comunicare solo con il livello immediatamente sotto di lui, che ci ha permesso di separare le diverse funzionalità del sistema.

**Riuso:** Sono stati usati i concetti di interface inheritance e delegation per favorire il riutilizzo i componenti.

**Usabilità:** L’interfaccia sarà realizzata in modo tale da consentire all’utente di interagire e navigare facilmente nel sistema.

* 1. **Linee guida per la documentazione**

Dopo un ciclo while, for e qualunque struttura di controllo del flusso come: *if conditions*, *else if*  e *else* seguirà la parentesi graffa e il blocco di istruzioni sarà indentato di un **tab** alla riga successiva, anche nel caso in cui il corpo contenga una singola istruzione aumentando così la leggibilità del codice.

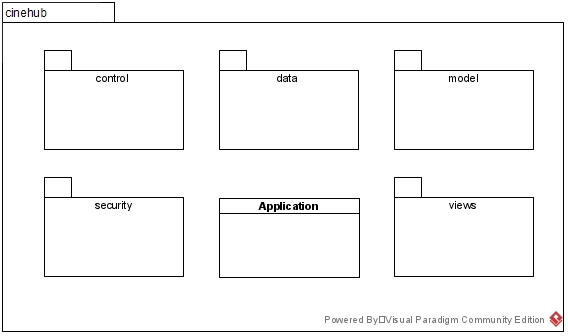
La scrittura dei metodi segue esattamente la stessa struttura di indentazione descritta nel punto precedente.

I nomi delle variabili e delle classi sono significati ed evocativi rispetto al ruolo che svolgono.

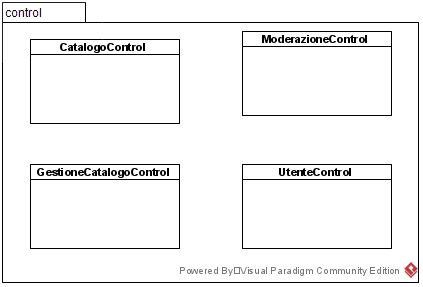
* 1. **Componenti off the shelf**

Andremo ad utilizzare due componenti off the shelf:

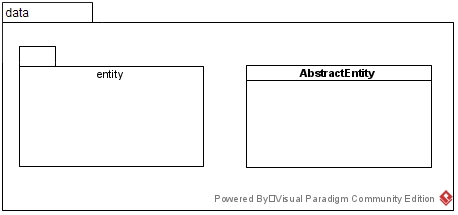
1. **Spring Boot**: un framework open source basato su Apache Tomcat e JavaEE che permette di scrivere in modo semplice applicazioni web. Abbiamo scelto di utilizzare questo componente perché rende di facile utilizzo componenti propri di Java EE come JPA, JPQL, Web Services aggiungendo livelli di astrazione e semplificando la configurazione.
2. **Vaadin**: un framework open source che consente di creare interfacce web responsive in linguaggio Java. Risulta efficiente utilizzare questo componente perché ci consente di utilizzare lo stesso linguaggio (Java) su tutta l’applicazione, sia front-end che back-end.
3. **Packages**
   1. **Package cinehub**



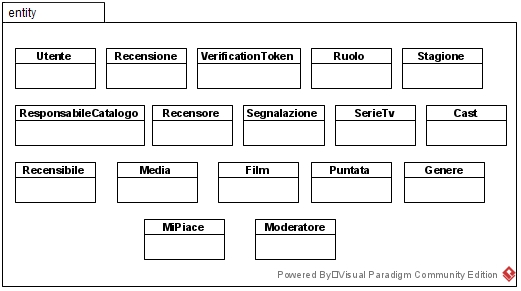
* + 1. **Package control**



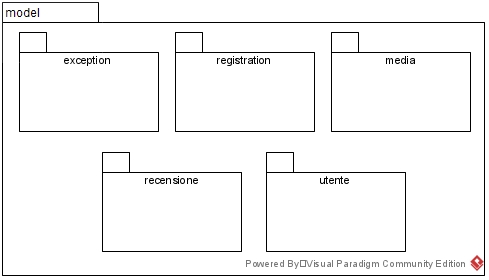
* + 1. **Package data**



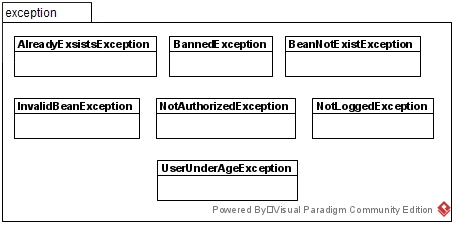
* + - 1. **Package entity**



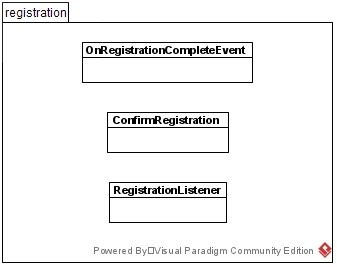
* + 1. **Package model**



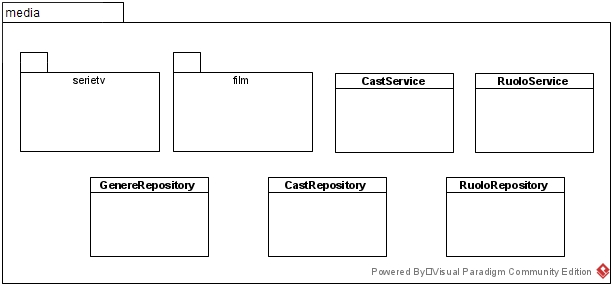
* + - 1. **Package exception**



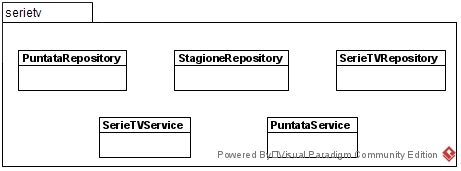
* + - 1. **Package registration**



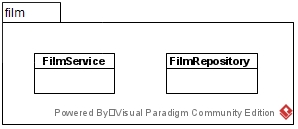
* + - 1. **Package media**



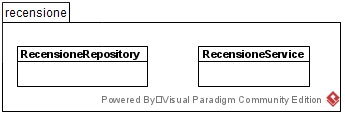
* + - * 1. **Package serietv**



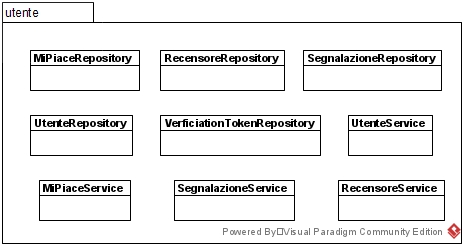
* + - * 1. **Package film**



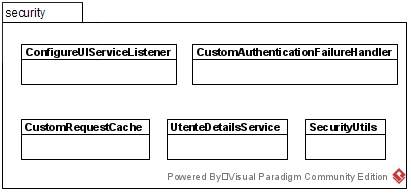
* + - 1. **Package recensione**



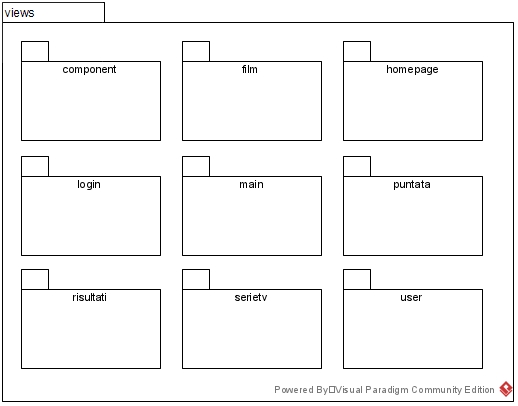
* + - 1. **Package utente**



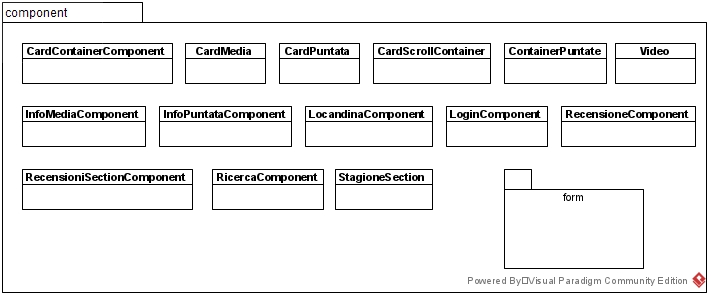
* + 1. **Package security**



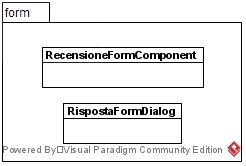
* + 1. **Package views**



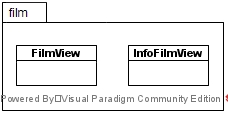
* + - 1. **Package component**

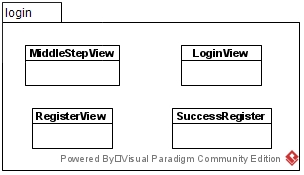


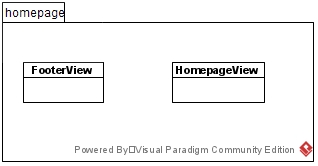
* + - * 1. **Package form**



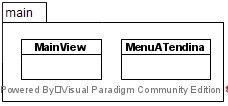
* + - 1. **Package film**



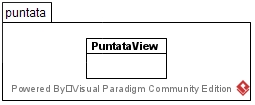
* + - 1. **Package homepage**
      2. **Package login**



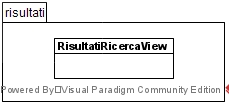
* + - 1. **Package main**



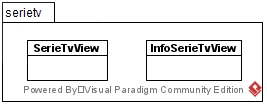
* + - 1. **Package puntata**



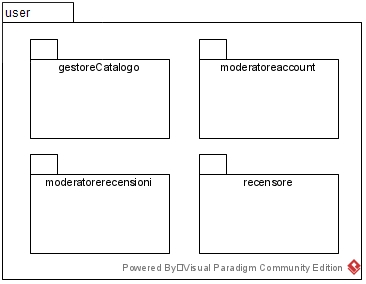
* + - 1. **Package risultati**



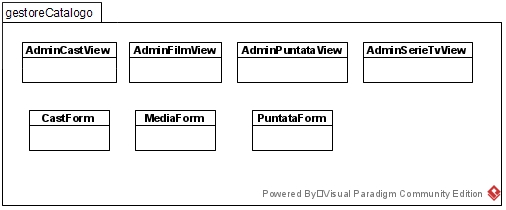
* + - 1. **Package serietv**



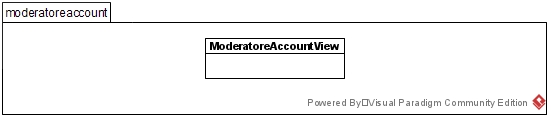
* + - 1. **Package user**



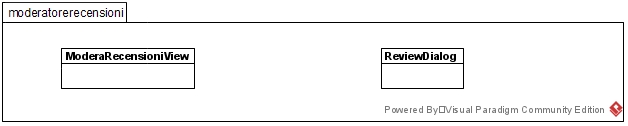
* + - * 1. **Package gestoreCatalogo**



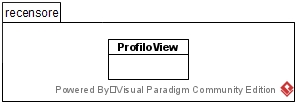
* + - * 1. **Package moderatoreaccount**



* + - * 1. **Package moderatorerecensioni**



* + - * 1. **Package recensore**



1. **Interfacce delle classi**
   1. **AbstractEntity**

|  |
| --- |
| **AbstractEntity** |
| - id: Long |
| + getId(): Long  + setId(id: Long): void |

* 1. **Media**

|  |
| --- |
| **Media** |
| - titolo: String  - annoUscita: Integer  - sinossi: String  - linkTrailer: String  - linkLocandina: String  - mediaVoti: Double  - generi: Set<Genere>  - ruoli: Collection<Ruolo>  - visibile : Boolean |
| + getTitolo(): String  + getAnnoUscita(): Integer  + getSinossi(): String  + getLinkTrailer(): String  + getLinkLocandina(): String  + getMediaVoti(): Double  + getGeneri(): Set<Genere>  + getRuoli(): Collection<Ruoli>  + getVisibile(): Boolean  + setTitolo(titolo: String): void  + setAnnoUscita(annoUscita: Integer): void  + setSinossi(sinossi: String): void  + setLinkTrailer(linkTrailer: String): void  + setLinkLocandina(linkLocandina: String): void  + setMediaVoti(mediaVoti: Double): void  + setGeneri(generi: Set<Genere>): void  + setRuoli(ruoli: Collection<Ruolo>): void  + setVisibile(Boolean visibile):void  + checkMedia(media: Media): boolean |

* 1. **Genere**

|  |
| --- |
| **Genere** |
| - nomeGenere: NomeGenere  - mediaCollegati: Set<Media> |
| + getNomeGenere(): NomeGenere  + getMediaCollegati(): Set<Media>  + setNomeGenere(nomeGenere: NomeGenere): void  + setMediaCollegati(mediaCollegati: Set<Media>): void  + getTuttiGeneri(): Set<Genere> |

* 1. **Ruolo**

|  |
| --- |
| **Ruolo** |
| - tipo: Tipo  - castId: Long  - mediaId: Long  - cast: Cast  - media: Media |
| + getTipo(): Tipo  + getCastId(): Long  + getMediaId(): Long  + getCast(): Cast  + getMedia(): Media  + setTipo(tipo: Tipo): void  + setCastId(castId: Long): void  + setMediaId(mediaId: Long): void  + setCast(cast: Cast): void  + setMedia(media: Media): void |

* 1. **Cast**

|  |
| --- |
| **Cast** |
| - nome: String  - cognome: String  - ruoli: Collection<Ruolo> |
| + getNome(): String  + getCognome(): String  + getRuoli(): Collection<Ruolo>  + setNome(nome: String): void  + setCognome(cognome: String): void  + setRuoli(ruoli: Collection<Ruoli>): void |

* 1. **Recensibile**

|  |
| --- |
| **Recensibile** |
|  |
| + getMediaVoti(): Double  + aggiungiRecensione(recensione: Recensione): void  + rimuoviRecensione(recensione: Recensione): void  + getListaRecensioni(): List<Recensioni> |

* 1. **Film**

|  |
| --- |
| **Film** |
| - listaRecensioni: List<Recensioni> |
| + getListaRecensioni(): List<Recensioni>  + setListaRecensioni(listaRecensioni: List<Recensioni>): void  + getMediaVoti(): Double  + aggiungiRecensione(recensione: Recensione): void  + rimuoviRecensione(recensione: Recensione): void  + getListaRecensioni(): List<Recensioni> |

* 1. **SerieTv**

|  |
| --- |
| **SerieTv** |
| - stagioni: Collection<Stagione> |
| + getStagioni(): Stagione stagione  + setStagioni(stagioni: Stagione): void  + calcolaMediaVoti(): void  - calcolaMediaVotiStagione(Stagione stagione): Double |

* 1. **Stagione**

|  |
| --- |
| **Stagione** |
| - numeroStagione: Integer  - serieTvId: Long  - puntate: Collection<Puntata>  - serieTv: SerieTv |
| + getNumeroStagione(): Integer  + getSerieTvId(): Long  + getPuntate(): Collection<Puntata>  + getSerieTv(): SerieTv  + setNumeroStagione(numeroStagione: Integer): void  + setSerieTvId(serieTvId: Long): void  + setPuntate(puntate: Collection<Puntata>): void  + setSerieTv(serietv: SerieTv): void |

* 1. **Puntata**

|  |
| --- |
| **Puntata** |
| - titolo: String  - numeroPuntata: Integer  - sinossi: String  - stagioneId: StagioneID  - stagione: Stagione  - listaRecensioni: List<Recensione>  - mediaVoti: Double  - visibile: Boolean |
| + getTitolo(): String  + getNumeroPuntata(): Integer  + getSinossi(): String  + getStagioneID(): StagioneID  + getStagione(): Stagione  + getListaRecensioni(): List<Recensione>  + getMediaVoti(): Double  + getVisibile(): Boolean  + setTitolo(titolo: String): void  + setNumeroPuntata(numeroPuntata: Integer): void  + setSinossi(sinossi: String): void  + setStagioneID(stagioneID: StagioneID): void  + setStagione(stagione: Stagione): void  + setListaRecensioni(listaRecensioni: List<Recensione>): void  + setMediaVoti(mediaVoti: Double): void  + aggiungiRecensione(recensione: Recensione): void  + rimuoviRecensioen(recensione: Recensione): void  + setVisibile(visibile: Boolean): void  + checkPuntata(p: Puntata): boolean  - calcolaMediaVoti(): Double |

* 1. **Recensione**

|  |
| --- |
| **Recensione** |
| - createdAt: Timestamp  - contenuto: String  - punteggio: Integer  - recensore: Recensore  - listaSegnalazioni: List<Segnalazione>  - listaMiPiace: List<MiPiace>  - listaRisposte: List<Recensione>  - padre: Recensione  - film: Film  - puntata: Puntata |
| + getCreatedAt(): Timestamp  + getContenuto(): String  + getPunteggio(): Integer  + getRecensore(): Recensore  + getListaSegnalazioni(): List<Segnalazione>  + getListaMiPiace(): List<MiPiace>  + getListaRisposte(): List<Recensione>  + getPadre(): Recensione  + getFilm(): Film  + getPuntata(): Puntata  + setCreatedAt(createdAt: Timestamp): void  + setContenuto(contenuto: String): void  + setPunteggio(punteggio: Integer): void  + setRecensore(recensore: Recensore): void  + setListaSegnalazioni(listaSegnalazioni: List<Segnalazione>): void  + setListaMiPiace(listaMiPiace: List<MiPiace>): void  + setListaRisposte(listaRisposte: List<Recensioen>): void  + setPadre(padre: Recensione): void  + setFilm(film: Film): void  + setPuntata(puntata: Puntata): void |

* 1. **MiPiace**

|  |
| --- |
| **MiPiace** |
| - tipo: boolean  - createdAt: Timestamp  - recensoreId: String  - recensioneId: Long  - recensore: Recensore  - recensione: Recensione |
| + isTipo(): boolean  + getCreatedAt(): Timestamp  + getRecensoreId(): String  + getRecensioneId(): Long  + getRecensore(): Recensore  + getRecensione(): Recensione  + setTipo(tipo: Tipo): void  + setCreatedAt(createdAt: Timestamp): void  + setRecensoreId(recensoreId: String): void  + setRecensioneId(recensioneId: Long): void  + setRecensore(recensore: Recensore): void  + setRecensione(recensione: Recensione): void |

* 1. **Segnalazione**

|  |
| --- |
| **Segnalazione** |
| - createdAt: Timestamp  - segnalatoreId: String  - recensoreId: String  - recensioneId: Long  - recensore: Recensore  - recensione: Recensione |
| + getCreatedAt(): Timestamp  + getSegnalatoreId(): String  + getRecensoreId(): String  + getRecensioneId(): Long  + getRecensore(): Recensore  + getRecensione(): Recensione  + setCreatedAt(createdAt: Timestamp): void  + setSegnalatoreId(segnalatoreId: String): void  + setRecensoreId(recensoreId: String): void  + setRecensioneId(recensioneId: Long): void  + setRecensore(recensore: Recensore): void  + setRecensione(recensione: Recensione): void |

* 1. **Utente**

|  |
| --- |
| **Utente** |
| - email: String  - nome: String  - cognome: String  - dataNascita: LocalDate  - username: String  - password: String  - isBannato: Boolean  - isActive: Boolean |
| + getEmail(): String  + getNome(): String  + getCognome(): String  + getDataNascita(): LocalDate  + getUsername(): String  + getPassword(): String  + getBannato(): Boolean  + getActive(): Boolean  + setEmail(email: String): void  + setNome(nome: String): void  + setCognome(cognome: String): void  + setDataNascita(dataNascita: LocalDate): void  + setUsername(username: String): void  + setPassword(password: String): void  + setBannato(bannato: Boolean): void  + setActive(active: Boolean): void |

* 1. **Recensore**

|  |
| --- |
| **Recensore** |
| - listaRecensioni: List<Recensione>  - listaSegnalazioni: List<Segnalazione>  - listaMiPiace: List<MiPiace> |
| + getListaRecensioni(): List<Recensione>  + getListaSegnalazioni(): List<Segnalazione>  + getListaMiPiace(): List<MiPiace>  + setListaRecensioni(listaRecensione: List<Recensione>): void  + setListaSegnalazioni(listaSegnalazioni: List<Segnalazione>): void  + setListaMiPiace(listaMiPiace: List<MiPiace>): void |

* 1. **Moderatore**

|  |
| --- |
| **Moderatore** |
| - tipo: Tipo |
| + getTipo(): Tipo |

* 1. **ResponsabileCatalogo**

|  |
| --- |
| **ResponsabileCatalogo** |
|  |
|  |

* 1. **Film Service**

|  |
| --- |
| **Nome Classe:** FilmService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante i film permettendo di aggiungere, rimuovere e modificare film nonché di ricercarli o prelevarli singolarmente o totalmente. |
| **Metodi:** + addFilm(Film film)  + removeFilm(Long id)  + retrieveAll()  + retrieveByKey(Long id)  + addGeneri(Collection<Genere> generi, Long id)  +addCast(Collection<Ruoli> ruoli, Long id)  + mergeFilm(Film film)  + searchByTitle(String titolo)  + searchByGenre(Genere genere)  + findMostRecentFilm(Integer howMany) |

|  |  |
| --- | --- |
| **Nome metodo:** addFilm(Film film): Film | |
| **Descrizione metodo:** dato un film lo aggiunge lo rende persistente | |
| **PreCondizioni** | **Context:**  FilmService::addFilm(Film film)  **Pre:** film != null &&  (!film.getTitolo.isBlank()) &&  (1895 <= film.getAnnoUscita <= annoCorrente) && (!film.getSinossi.isBlank()) && (!film.getLinkLocandina.isBlank()) && (!film.getLinkTrailer.isBlank()) &&  (!film.getGeneri.isEmpty()) && non deve esistere un altro film con stesso titolo e anno di uscita |
| **PostCondizioni** | **Context:**  FilmService::addFilm(Film film)  **Post:** film in Film su DBMS |
| **Eccezioni** | Se film == null ||  (film.getTitolo.isBlank()) || (1895 > film.getAnnoUscita) ||  (film.getAnnoUscita > annoCorrente) || (film.getSinossi.isBlank()) || (film.getLinkLocandina.isBlank()) || (film.getLinkTrailer.isBlank()) ||  (film.getGeneri.isEmpty()) throw **InvalidBeanException**  Se esiste un altro film con stesso titolo o anno di uscita allora lanciamo una **AlreadyExistException**. |

|  |  |
| --- | --- |
| **Nome metodo:** removeFilm(Long id): void | |
| **Descrizione metodo:** dato un id, se esiste un film associato a questo id allora viene eliminato | |
| **PreCondizioni** | **Context:** FilmService::removeFilm(Long id)  **Pre:** id != null && deve esistere un film con id: “id” |
| **PostCondizioni** | **Context:** FilmService::removeFilm(Long id)  **Post:** il film con id “id” è rimosso dalla tabella Film del DBMS |
| **Eccezioni** | Se id == null || non esiste film con id: “id” throw **InvalidBeanException** |

|  |  |
| --- | --- |
| **Nome metodo**: retrieveAll(): List<Film> | |
| **Descrizione metodo**: ritorna tutti i film presenti nel catalogo | |
| PreCondizioni | **Context**: FilmService::retriveAll()  **Pre**: // |
| PostCondizioni | **Context**: FilmService::retriveAll()  **Post**: // |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveByKey(Long id): Film | |
| **Descrizione metodo:** dato un id restituisce il film con quell’id se esiste | |
| **PreCondizioni** | **Context:** FilmService::retrieveByKey(Long id)  FilmService::retrieveByKey(Long id)  **Pre:** (id != null) && deve esistere un film con id: “id” |
| **PostCondizioni** | **Context:**  FilmService::retrieveByKey(Long id)  **Post:** // |
| **Eccezioni** | Se id == null throw **InvalidBeanException**  Se non esiste un film con id: “id” **BeanNotExistsException**. |

|  |  |
| --- | --- |
| **Nome metodo:** addGeneri(Collection<Genere> generi, Long id): Film | |
| **Descrizione metodo:** data una collezione di generi e l’id di un film appende alla lista di generi del film con tale id la collezione generi | |
| **PreCondizioni** | **Context:**  FilmService::addGeneri(Collection<Genere> generi, Long id)  **Pre:** (!generi.isEmpty()) && (id != null) && Deve esistere un film con id: “id” |
| **PostCondizioni** | **Context:**  FilmService::addGeneri(Collection<Genere> generi, Long id)  **Post:** film.getGeneri() = generiPrecedenti + generi |
| **Eccezioni** | Se (generi.isEmpty()) || (id == null) throw **InvalidBeanException**  Se non esiste un film con id: “id” throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** addCast(Collection<Ruoli>ruoli, Long id): Film | |
| **Descrizione metodo:** data una collezione di ruoli e l’id di un film associa a tale film la lista di ruoli | |
| **PreCondizioni** | **Context:**  FilmService::addCast(Collection<Ruoli> ruoli, Long id)  **Pre:** (!ruoli.isEmpty() && id != null)  il film con id: “id” deve esistere |
| **PostCondizioni** | **Context:**  FilmService::addCast(Collection<Ruoli> ruoli, Long id)  **Post:** film.getRuoli() = ruoli |
| **Eccezioni** | Se generi.isEmpty() || (id == null) throw **InvalidBeanException**  Se non esiste un film con id: “id” throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** mergeFilm(Film film): void | |
| **Descrizione metodo:** dato un film aggiorna le modifiche | |
| **PreCondizioni** | **Context:**  FilmService::mergeFilm(Film film)  **Pre:** film != null &&  (film.getTitolo != null) &&  (1895 <= !film.getAnnoUscita <= annoCorrente) && !film.getSinossi.isBlank() &&  (film.getLinkLocandina != null) &&  (film.getLinkTrailer != null) &&  il film con id: “id” deve esistere |
| **PostCondizioni** | **Context:**  FilmService::mergeFilm(Film film)  **Post:** il film Film ha lo stato del film passato in input. |
| **Eccezioni** | Se film == null ||  (film.getTitolo == null) ||  (1895 > film.getAnnoUscita) ||  film.getAnnoUscita > annoCorrente)|| film.getSinossi.isBlank() ||  film.getLinkLocandina == null || film.getLinkTrailer == null throw **InvalidBeanException**  Se non esiste un film con id: “id” throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** searchByTitle(String titolo): List<Film> | |
| **Descrizione metodo:** ricerca un film tramite titolo o parte di esso | |
| **PreCondizioni** | **Context:**  FilmService::searchByTitle(String titolo)  **Pre:** (!titolo.isBlank()) |
| **PostCondizioni** | **Context:**  FilmService::searchByTitle(String titolo)  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** searchByGenere(Collection<Genere> nomiGeneri): Collection<Film> | |
| **Descrizione metodo:** dato uno o più generi restituisce la lista di film con quei generi | |
| **PreCondizioni** | **Context:**  FilmService::searchByGenere(Collection<Genere> nomiGeneri)  **Pre:** !nomiGeneri.isEmpty() |
| **PostCondizioni** | **Context:**  FilmService::searchByGenere(Collection<Genere> nomiGeneri): List<Film>  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** List<Film> findMostRecentFilm(Integer howMany): List<Film> | |
| **Descrizione metodo:** dato un intero “howMany” restituisce gli “howMany” film più recenti | |
| **PreCondizioni** | **Context:**  FilmService:: findMostRecentFilm(Integer howMany)  **Pre:** howMany >= 0 |
| **PostCondizioni** | **Context:**  FilmService:: findMostRecentFilm(Integer howMany): List<Film>  **Post:** // |

* 1. **SerieTVService**

|  |
| --- |
| **Nome Classe:** SerieTVService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante le SerieTV, ne permette l’aggiunta, la rimozione e la modifica, nonché la ricerca per titolo e per genere. |
| **Metodi:** + addSerieTV(SerieTV serieTV)  + removeSerieTV(Long id)  + retrieveAll()  + retrieveByKey(Long id)  + addGeneri(Collection<Genere> generi, Long id)  + mergeSerieTV(SerieTv serietv)  + searchByTitle(String titolo)  + searchByGenere(Genere genere)  + addStagione(SerieTv serieTv, Stagione stagione)  + removeStagione(SerieTv serieTv, Stagione stagione)  + getStagione(SerieTv serieTv, Integer numeroStagione)  + aggiornaStagione(Stagione stagione)  + findMostRecentSerieTv(Integer howMany) |

|  |  |
| --- | --- |
| **Nome metodo:** addSerieTv(SerieTv serietv):SerieTv | |
| **Descrizione metodo:** data una serie tv la aggiunge e la rende persistente | |
| **PreCondizioni** | **Context:** SerieTvService::addSerieTv(SerieTv serietv)  **Pre:** serietv != null &&  (serietv.getTitolo != null) &&  (1895 <= serietv.getAnnoUscita <= annoCorrente) &&  (!serietv.getSinossi.isBlank()) && (serietv.getLinkLocandina != null) && (serietv.getLinkTrailer != null) &&  Non deve esistere un’altra serietv con stesso titolo e anno di uscita |
| **PostCondizioni** | **Context:** SerieTvService::addSerieTv(SerieTv serietv)  **Post:** serieTV presente nella tabella SerieTV del DBMS |
| **Eccezioni** | Se serietv == null ||  (serietv.getTitolo == null) ||  (1895 > serietv.getAnnoUscita ||  (serietv.getAnnoUscita > annoCorrente) ||  (serietv.getSinossi.isBlank()) || (serietv.getLinkLocandina == null) || (serietv.getLinkTrailer == null) throw **InvalidBeanException**  Se esiste già una serietv con stesso titolo e anno uscita throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** removeSerieTv(Long id): SerieTv | |
| **Descrizione metodo:** Dato l’id di una serieTv la rimuove se questa esiste | |
| **PreCondizioni** | **Context:**  SerieTVService::removeSerieTV(Long id)  **Pre:** id != null && esiste serietv con id: “id” |
| **PostCondizioni** | **Context:** SerieTVService::removeSerieTV(Long id)  **Post:** la serieTV con id “id” è rimossa dalla tabella SerieTV del DBMS |
| **Eccezioni** | Se id == null throw **InvalidBeanException**  Se non esiste serietv con id “id” throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** retriveAll():List<SerieTv> | |
| **Descrizione metodo:** ritorna tutte le serie tv presenti nel database | |
| **PreCondizioni** | **Context:**  SerieTvService::retriveAll()  **Pre:** // |
| **PostCondizioni** | **Context:**  SerieTvService::retriveAll()  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveByKey(Long id): SerieTv | |
| **Descrizione metodo:** dato un id restituisce la serie tv con quell’id se esiste | |
| **PreCondizioni** | **Context:** SerieTvService::retrieveByKey(Long id)  **Pre:** (id != null) && deve esistere una serietv con id “id” |
| **PostCondizioni** | **Context:** SerieTvService::retrieveByKey(Long id)  **Post:** // |
| **Eccezioni** | Se id == null throw **InvalidBeanException**  Se non esiste serietv con id “id” throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** addGeneri(Collection<Genere> generi, Long id ):SerieTv | |
| **Descrizione metodo:** data una collection di generi e un id aggiunge alla lista di generi della serie tv con id “id” la collection di generi passata | |
| **PreCondizioni** | **Context:** SerieTvService::addGeneri(Collection<Genere> genri, Long id)  **Pre:** (!generi.isEmpty()) && (id != null) && deve esiste una serietv con id “id” |
| **PostCondizioni** | **Context:**  SerieTvService::addSerieTv(SerieTv serietv)  **Post:** serieTV.getGeneri() = generiPrecedenti + generi |
| **Eccezioni** | Se generi.isEmpty() || id == null throw **InvalidBeanException**  Se non esiste serietv con id “id” throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** addCast(Collection<Ruoli>ruoli, Long id): SerieTv | |
| **Descrizione metodo:** data una collezione di ruoli e l’id di una serie tv associa a tale serie tv la lista di ruoli | |
| **PreCondizioni** | **Context:**  SerieTvService::addCast(Collection<Ruoli> ruoli, Long id)  **Pre:** (!ruoli.isEmpty() && id != null)  La serie tv con id: “id” deve esistere |
| **PostCondizioni** | **Context:**  FilmService::addCast(Collection<Ruoli> ruoli, Long id)  **Post:** film.getRuoli() = ruoli |
| **Eccezioni** | Se generi.isEmpty() || (id == null) throw **InvalidBeanException**  Se non esiste una serie tv con id: “id” throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** mergeSerieTv(SerieTv serietv): SerieTv | |
| **Descrizione metodo:** data una serie tv aggiorna le modifiche | |
| **PreCondizioni** | **Context:**  SerieTvService::mergeSerieTv(SerieTv serietv)  **Pre:** serietv != null &&  (serietv.getTitolo != null) &&  (1895 <= serietv.getAnnoUscita <= annoCorrente) &&  ! serietv.getSinossi.isBlank() && (serietv.getLinkLocandina != null) && (serietv.getLinkTrailer != null) &&  La serie tv con id: “id” deve esistere |
| **PostCondizioni** | **Context:**  SerieTvService::mergeSerieTV(SerieTv serietv)  **Post:** la serietv Serietv ha lo stato della serie tv passata in input. |
| **Eccezioni** | Se serietv == null ||  (serietv.getTitolo == null) ||  (1895 > serietv.getAnnoUscita ||  (serietv.getAnnoUscita > annoCorrente) ||  (serietv.getSinossi.isBlank()) ||  (serietv.getLinkLocandina == null) || (serietv.getLinkTrailer == null)  throw **InvalidBeanException**  Se esiste già una serietv con stesso titolo e anno uscita throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** searchByTitle(String titolo): List<SerieTv> | |
| **Descrizione metodo:** ricerca una serie tv tramite titolo o parte di esso | |
| **PreCondizioni** | **Context:** SerieTvService::searchByTitle(String titolo)  **Pre:** (!titolo.isBlank()) |
| **PostCondizioni** | **Context:** SerieTvService::searchByTitle(String titolo)  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** searchByGenere(Collection<Genere> nomiGeneri): Collection<SerieTv> risultati | |
| **Descrizione metodo:** dato uno o più generi restituisce la lista di serie tv che possiedono quei generi | |
| **PreCondizioni** | **Context:**  SerieTvService::searchByGenere(Collection<Genere> nomiGeneri)  **Pre:** !nomiGeneri.isEmpty() |
| **PostCondizioni** | **Context:**  SerieTvService::searchByGenere(Collection<Genere> nomiGeneri): List<SerieTv>  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** addStagione(SerieTv serieTv, Stagione stagione): Stagione | |
| **Descrizione metodo:** data una serietv ed una stagione aggiunge alla lista delle stagioni della serietv la stagione passata | |
| **PreCondizioni** | **Context:**  SerieTvService:: addStagione(SerieTv serieTv, Stagione stagione)  **Pre:** serietv != null &&  (serietv.getTitolo != null) &&  (1985 <= serietv.getAnnoUscita <= annoCorrente) &&  !serietv.getSinossi.isBlank() &&  (serietv.getLinkLocandina != null) && (serietv.getLinkTrailer != null) && stagione.getNumeroStagione() >= 1 &&  serietv in DB |
| **PostCondizioni** | **Context:**  SerieTvService:: addStagione(SerieTv serieTv, Stagione stagione)  **Post:** serieTv.getStagioni() == precedentiStagioni + stagione |
| **Eccezioni** | Se serietv == null ||  (serietv.getTitolo == null) ||  (serietv.getAnnoUscita == null) ||  (serietv.getSinossi.isBlank() || (serietv.getLinkLocandina == null) || (serietv.getLinkTrailer == null) ||  stagione.getNumeroStagione() < 1 throw **InvalidBeanException**  Se non esiste serietv in DB throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** removeStagione(SerieTv serieTv, Integer numeroStagione): Stagione | |
| **Descrizione metodo:** data una serietv ed un numero di una stagione elimina dalla lista delle stagioni della serietv quella stagione con numero “numeroStagione” | |
| **PreCondizioni** | **Context:**  SerieTVService::removeStagione(SerieTv serieTv, Integer numeroStagione)  **Pre:** serietv != null &&  (serietv.getTitolo != null) &&  (1985 <= serietv.getAnnoUscita <= annoCorrente) &&  (!serietv.getSinossi.isBlank()) &&  (serietv.getLinkLocandina != null) && (serietv.getLinkTrailer != null) &&  1 <= numeroStagione <= ultimaStagione &&  serieTv deve esistere |
| **PostCondizioni** | **Context:**  SerieTVService::removeStagione(SerieTv serieTv, Integer numeroStagione)  **Post:** for all Stagione s in serieTv.getStagioni() : s.getNumero() != numeroStagione |
| **Eccezioni** | Se serietv == null ||  (serietv.getTitolo == null) ||  (serietv.getAnnoUscita == null) ||  (serietv.getSinossi.isBlank() || (serietv.getLinkLocandina == null) || (serietv.getLinkTrailer == null) ||  (stagione.getNumeroStagione() < 1) ||  (stagione.getNumeroStagione() > ultimaStagione) throw **InvalidBeanException**  Se serietv non esiste throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** getStagione(SerieTv serieTv, Integer numeroStagione): Stagione | |
| **Descrizione metodo:** data una serietv ed un numero di una stagione restituisce la stagione, della serietv, con numeroStagione passato in input se questa esiste | |
| **PreCondizioni** | **Context:**  SerieTVService::getStragione(SerieTv serieTv, Integer numeroStagione): Stagione  **Pre:** serieTv != null && numeroStagione != null && stagione con numeroStagione “numeroStagione” deve esistere |
| **PostCondizioni** | **Context:**  SerieTVService::getStragione(SerieTv serieTv, Integer numeroStagione): Stagione  **Post:** // |
| **Eccezioni** | Se serieTv == null || numeroStagione == null throw **InvalidBeanException**  Se stagione con numeroStagione “numeroStagione” non esiste throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** aggiornaStagione(Stagione stagione): void | |
| **Descrizione metodo:** data una stagione aggiorna le modifiche | |
| **PreCondizioni** | **Context:** SerieTVService::aggiornaStagione(Stagione stagione)  **Pre:** stagione != null |
| **PostCondizioni** | **Context:** SerieTVService::aggiornaStagione(Stagione stagione)  **Post:** lo stato della Stagione stagione è stato aggiornato |
| **Eccezioni** | Se stagione == null throw **InvalidBeanException** |

|  |  |
| --- | --- |
| **Nome metodo:** List<SerieTv > findMostRecentSerieTv(Integer howMany): List<SerieTv> mostRecentSerieTv | |
| **Descrizione metodo:** dato un intero “howMany” restituisce le “howMany” serietv più recenti | |
| **PreCondizioni** | **Context:**  SerieTVService:: findMostRecentSerietv(Integer howMany)  **Pre:** howMany >= 0 |
| **PostCondizioni** | **Context:**  SerieTVService:: findMostRecentSerieTv(Integer howMany): List<SerieTv >  **Post:** // |

* 1. **PuntataService**

|  |
| --- |
| **Nome Classe:** PuntataService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante le Puntate delle serie tv, ne permette l’aggiunta, la rimozione e la modifica, nonché la ricerca per titolo e per genere. |
| **Metodi:** + addPuntata(Puntata puntata, Integer numeroStagione, Long idSerieTV)  + removePuntata(PuntataID id)  + retrieveAll()  + retrieveBySerieTV(Long idSerieTv)  + retrieveByStagione(Long idSerieTv, Integer numeroStagione)  + retrievePuntataByKey(PuntataID puntataID)  + mergePuntata(Puntata puntata)  + searchByTitle(String titolo) |

|  |  |
| --- | --- |
| **Nome metodo:** addPuntata(Puntata puntata, Integer numeroStagione, Long idSerieTV): Puntata | |
| **Descrizione metodo:** data una puntata, il numero di una stagione e l’id di una serie tv rende persistente la puntata associandola ad una stagione di una determinata serie tv | |
| **PreCondizioni** | **Context:** PuntataService::addPuntata(Puntata puntata, Integer numeroStagione, Long idSerieTV)  **Pre:** numeroStagione != null && numeroStagione > 0 && puntata != null &&  puntata.getNumeroPuntata > 0 && !puntata.getTitolo().isBlank() && !puntata.getSinossi().isBlank() &&  puntata non esiste |
| **PostCondizioni** | **Context:** PuntataService::addSerieTv(SerieTv serietv)  **Post:** puntata presenta nella tabella Puntata del DBMS, aggiunta stagione se non presente |
| **Eccezioni** | Se numeroStagione == null || numeroStagione <= 0 ||  puntata.getTitolo().isBlank() ||  puntata.getSinossi().isBlank()  throw **InvalidBeanException**  Se la puntata esiste già throw **BeanArleadyExistsException** |

|  |  |
| --- | --- |
| **Nome metodo:** removePuntata(PuntataID id): void | |
| **Descrizione metodo:** Dato l’id di una puntata la rimuove se questa esiste | |
| **PreCondizioni** | **Context:** PuntataService::removePuntata(PuntataID id)  **Pre:** id != null &&  la puntata con id: “PuntataID” deve esistere |
| **PostCondizioni** | **Context:** PuntataService::removePuntata(PuntataID id)  **Post:** la Puntata con id “PuntataID” è rimossa dalla tabella Puntata del DBMS |
| **Eccezioni** | Se id == null throw **InvalidBeanException**  Se la puntata non esiste throw **BeanNotExistsException** |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveAll():List<Puntata> | |
| **Descrizione metodo:** ritorna tutte le puntate presenti nel database | |
| **PreCondizioni** | **Context:** PuntataService::retrieveAll()  **Pre:** // |
| **PostCondizioni** | **Context:** PuntataService::retrieveAll()  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveBySerieTv(Long idSerieTv):List<Puntata> | |
| **Descrizione metodo:** dato l’id di una serietv restituisce tutte le puntate della serietv con quell’id se esiste | |
| **PreCondizioni** | **Context:** PuntataService:: retrieveBySerieTv(Long idSerieTv)  **Pre:** idSerieTv != null && la serietv con id “idSerieTv” deve esistere |
| **PostCondizioni** | **Context:** PuntataService:: retrieveBySerieTv(Long idSerieTv)  **Post:** // |
| **Eccezioni** | Se idSerieTv == null throw **InvalidBeanException**  Se la serietv non esiste throw **BeanNotExistExceptio** |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveByStagione(Long idSerieTv, Integer numeroStagione):List<Puntata> | |
| **Descrizione metodo:** dato l’id di una serietv ed il numero di una stagione restituisce tutte le puntate della stagione con quel numero della serietv con quell’id se esistono | |
| **PreCondizioni** | **Context:** PuntataService:: retrieveByStagione(Long idSerieTv, Integer numeroStagione)  **Pre:** idSerieTv != null && numeroStagione > 0 |
| **PostCondizioni** | **Context:** PuntataService:: retrieveByStagione(Long idSerieTv, Integer numeroStagione)  **Post:** // |
| **Eccezioni** | Se idSerieTv == null || numeroStagione <= 0 throw **InvalidBeanException** |

|  |  |
| --- | --- |
| **Nome metodo:** retrievePuntataByKey(PuntataID puntataID): Puntata | |
| **Descrizione metodo:** dato un id restituisce la puntata con quell’id se esiste | |
| **PreCondizioni** | **Context:** PuntataService::retrieveByKey(Long id)  **Pre:** (id != null) &&  La puntata con puntataID “puntataID” esiste |
| **PostCondizioni** | **Context:** PuntataService::retrieveByKey(Long id)  **Post:** // |
| **Eccezioni** | Se id == null throw **InvalidBeanException**  Se non esiste la puntata con puntataID “puntataID” **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** mergePuntata(Puntata puntata): Puntata | |
| **Descrizione metodo:** data una puntata aggiorna le modifiche | |
| **PreCondizioni** | **Context:**  PuntataService::mergePuntata(Puntata puntata)  **Pre:** puntata != null &&  puntata.getNumeroPuntata > 0 && !puntata.getTitolo().isBlank() && !puntata.getSinossi().isBlank() &&  puntata deve esistere |
| **PostCondizioni** | **Context:**  PuntataService::mergePuntata(Puntata puntata)  **Post:** la puntata Puntata ha lo stato della puntata passata in input. |
| **Eccezioni** | Se puntata == null ||  puntata.getNumeroPuntata <= 0 || puntata.getTitolo().isBlank() || puntata.getSinossi().isBlank()  throw **InvalidBeanException**  se la puntata non esiste throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** searchByTitle(String titolo): List<Puntata> | |
| **Descrizione metodo:** ricerca una puntata tramite titolo o parte di esso | |
| **PreCondizioni** | **Context:** PuntataService::searchByTitle(String titolo)  **Pre:** (!titolo.isBlank()) & (titolo != null) |
| **PostCondizioni** | **Context:** PuntataService::searchByTitle(String titolo)  **Post:** // |

* 1. **CastService**

|  |
| --- |
| **Nome Classe:** CastService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante il cast |
| **Metodi:** + addCast(Cast cast)  + removeCast(Long id)  + retriveAll()  + retrieveByKey(Long id)  + mergeCast(Cast cast) |

|  |  |
| --- | --- |
| **Nome metodo:** addCast(Cast cast):Cast | |
| **Descrizione metodo:** dato un membro del cast lo aggiunge al db e lo rende persistente | |
| **PreCondizioni** | **Context:** CastService::addCast(Cast cast)  **Pre:** cast != null &&  !cast.getNome.isBlank() && !cast.getCognome.isBlank() |
| **PostCondizioni** | **Context:** CastService::addCast(Cast cast)  **Post:** cast aggiunto alla tabella Cast del DBMS |
| **Eccezioni** | se cast == null ||  cast.getNome.isBlank() || cast.getCognome.isBlank()  throw **InvalidBeanException** |

|  |  |
| --- | --- |
| **Nome metodo:** removeCast(Long id): void | |
| **Descrizione metodo:** Dato l’id di un membro del cast, lo rimuove se questo esiste | |
| **PreCondizioni** | **Context:**  CastService::removeCast (Long id)  **Pre:** id != null &&  Cast con id: “id” deve esistere |
| **PostCondizioni** | **Context:** CastService::removeCast(Long id)  **Post:** il cast con id “id” è rimosso dalla tabella Cast del DBMS |
| **Eccezioni** | se id == nul throw **InvalidBeanException**  se cast con id “id” non esiste throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveAll():List<Cast> | |
| **Descrizione metodo:** ritorna tutti i membri del Cast presenti nel database | |
| **PreCondizioni** | **Context:** CastService::retrieveAll()  **Pre:** // |
| **PostCondizioni** | **Context:** CastService::retrieveAll()  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveByKey(Long id): Cast | |
| **Descrizione metodo:** dato un id restituisce il cast con quell’id se esiste | |
| **PreCondizioni** | **Context:** CastService::retrieveByKey(Long id)  **Pre:** id != null && cast con id: “id” deve esistere |
| **PostCondizioni** | **Context:** CastService::retrieveByKey(Long id)  **Post:** // |
| **Eccezioni** | se id == nul throw **InvalidBeanException**  se cast con id “id” non esiste throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** mergeCast(Cast cast): Cast | |
| **Descrizione metodo:** dato un membro del cast aggiorna le modifiche | |
| **PreCondizioni** | **Context:** CastService::mergeCast(Cast cast)  **Pre:** cast != null &&  !cast.getNome.isBlank() && !cast.getCognome.isBlank() && cast con id: “id” deve esistere |
| **PostCondizioni** | **Context:** CastService::mergeCast(Cast cast)  **Post:** il cast Cast ha lo stato del cast passato in input. |
| **Eccezioni** | se cast == null ||  cast.getNome.isBlank() || cast.getCognome.isBlank()  throw **InvalidBeanException**  se cast con id “id” non esiste throw **BeanNotExistException** |

* 1. **RecensioneService**

|  |
| --- |
| **Nome Classe:** RecensioneService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante le recensioni |
| **Metodi:** + addRecensione(Recensione recensione, Recensore recensore)  + removeRecensione(Recensione recensione)  + retrieveAll()  + addRisposta(Recensore recensore, Recensione recensione, Long idPadre)  + retrieveById(Long id) |

|  |  |
| --- | --- |
| **Nome metodo:** addRecensione(Recensione recensione, Recensore recensore): void | |
| **Descrizione metodo:** data una recensione ed il recensore che l’ha scritta rende persistente questa recensione e viene aggiunta alla lista delle recensioni del film o puntata a cui si riferisce | |
| **PreCondizioni** | **Context:** RecensioneService::addRecensione(Recensione recensione, Recensore recensore)  **Pre:** recensione != null &&  !recesione.getContenut.isBlank() &&  1 <= recensione.getPunteggio <= 5 &&  (recensione.getFilm() != null || recensione.getPuntata() != null) |
| **PostCondizioni** | **Context:** RecensioneService::addRecensione(Recensione recensione, Recensore recensore)  **Post:** recensione aggiunta alla tabella Recensione del DBMS |
| **Eccezioni** | Se recensione == null ||  !recesione.getContenut.isBlank() ||  Recensione.getPunteggio < 1 || recensione.getPunteggio > 5 ||  (recensione.getFilm() != null && recensione.getPuntata() != null)  throw **InvalidBeanException** |

|  |  |
| --- | --- |
| **Nome metodo:** removeRecensione(Recensione recensione): void | |
| **Descrizione metodo:** data una recensione questa viene cancellata. Sarà anche rimossa dalla lista di tutte le recensioni del media a cui si riferisce. A catena saranno eliminate tutte le sue risposte a questa recensione. | |
| **PreCondizioni** | **Context:** RecensioneService::removeRecensione(Recensione recensione)  **Pre:** recensione != null && recensione deve esistere |
| **PostCondizioni** | **Context:** RecensioneService::removeRecensione(Recensione recensione)  **Post:** recensione rimossa dalla tabella Recensione del DBMS |
| **Eccezioni** | se recensione == null throw **InvalidBeanException**  se recensione non esiste throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveAll(): List<Recensione> | |
| **Descrizione metodo:** restituisce tutte le recensioni. | |
| **PreCondizioni** | **Context:** RecensioneService::retrieveAll)  **Pre:** // |
| **PostCondizioni** | **Context:** RecensioneService::retrieveAll(): List<Recensione>  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveByKey(Long id): Recensione | |
| **Descrizione metodo:** restituisce la recensione dato l’id e questa esiste | |
| **PreCondizioni** | **Context:** RecensioneService::retrieveByKey  **Pre:** id != null &&  la recensione con id: “id” deve esistere |
| **PostCondizioni** | **Context:** RecensioneService::retrieveByKey():Recensione  **Post:** // |
| **Eccezioni** | se id == null throw **InvalidBeanException**  se recensione con id “id” non esiste throw **BeanNotExistException** |

|  |  |
| --- | --- |
| **Nome metodo:** addRisposta(Recensione recensione, Recensore recensore, Long idPadre): void | |
| **Descrizione metodo:** data una recensione ed il recensore che l’ha scritta e l’id di un’altra recensione rende persistente questa recensione e questa viene aggiunta alla lista delle risposte della recensione a cui si riferisce | |
| **PreCondizioni** | **Context:** RecensioneService::addRecensione(Recensione recensione, Recensore recensore)  **Pre:** !recesione.getContenuto.isBlank()&& recensione con id: “idPadre” deve esiste |
| **PostCondizioni** | **Context:** RecensioneService::addRecensione(Recensione recensione, Recensore recensore)  **Post:** recensione aggiunta alla tabella Recensione del DBMS |
| **Eccezioni** | se recensione.getContenuto.isBlank() throw **InvalidBeanException**  se recensione con id “idPadre non esiste” throw **BeanNotExistException** |

* 1. **RuoloService**

|  |
| --- |
| **Nome Classe:** RuoloService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante i ruoli |
| **Metodi:** + addRuolo(Ruolo ruolo, Long castId, Long mediaId)  + cleanRuolo(Long mediaId) |

|  |  |
| --- | --- |
| **Nome metodo:** addRuolo(Ruolo ruolo, Long castId, Long mediaId): void | |
| **Descrizione metodo:** Dato un ruolo, l’id di un cast e l’id di un media, aggiunge alla lista dei ruoli del media con id “mediaId” il membro del cast con id “castId” con il ruolo passato in input. | |
| **PreCondizioni** | **Context:** RuoloService:: addRuolo(Ruolo ruolo, Long castId, Long mediaId)  **Pre:** ruolo != null && castId != null && mediaId != null && ruolo.getTipo != null && media con id: “mediaId” deve esistere && cast con id: “castId” deve esistere |
| **PostCondizioni** | **Context:** RuoloService:: addRuolo(Ruolo ruolo, Long castId, Long mediaId)  **Post:** il ruolo è stato aggiunto nella tabella Ruolo del DBMS |
| **Eccezioni** | Se ruolo == nulll ||  castId == null ||  mediaId == null ||  ruolo.getTipo() == null  Thrown **InvalidBeanException**  Se il media non esiste o il cast non esiste thrown **BeanNotExistsException** |

|  |  |
| --- | --- |
| **Nome metodo:** cleanRuolo(Long mediaId): void | |
| **Descrizione metodo:** Dato l’id di un media rimuove tutti i ruoli associati a quel media | |
| **PreCondizioni** | **Context:** RuoloService:: cleanRuolo(Long mediaId)  **Pre:** deve esistere un media con id “mediaId” |
| **PostCondizioni** | **Context:** RuoloService:: cleanRuolo(Long mediaId)  **Post:** la lista dei ruoli del media con id “mediaId” è vuota |
| **Eccezioni** |  |

* 1. **SegnalazioneService**

|  |
| --- |
| **Nome Classe:** SegnalazioneService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante le segnalazioni |
| **Metodi:** + addSegnalazione(Recensione recensione, Recensore segnalatore)  + puoSegnalare(Recensione recensione, Recensore segnalatore)  + retrieveAll() |

|  |  |
| --- | --- |
| **Nome metodo:** addSegnalazione(Recensione recensione, Recensore segnalatore): Segnalazione | |
| **Descrizione metodo:** data una Recensione e un segnalatore aggiunge una segnalazione alla lista delle segnalazioni della recensione passata ed alla lista delle segnalazioni del recensore che ha scritto la recensione | |
| **PreCondizioni** | **Context:** SegnalazioneService:: addSegnalazione(Recensione recensione, Recensore segnalatore)  **Pre:** recensione !=null && segnalatore != null && !recensione.getRecensore.equals(segnalatore) |
| **PostCondizioni** | **Context:** SegnalazioneService:: addSegnalazione(Recensione recensione, Recensore segnalatore)  **Post:** la segnalazione è stata aggiunta alla tabella Segnalazione del DBMS |
| **Eccezione** | Se recensione == null || segnalatore == null throw **InvalidBeanException**  Se recensione.getRecensore.equals(segnalatore) throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome metodo:** puoSegnalare(Recensione recensione, Recensore segnalatore): Boolean | |
| **Descrizione metodo:** Data una recensione e un segnalatore controlla se il segnalatore è autorizzato a segnalare. | |
| **PreCondizioni** | **Context:** SegnalazioneService:: puoSegnalare(Recensione recensione, Recensore segnalatore)  **Pre:** recensione !=null && segnalatore != null && !recensione.getRecensore.equals(segnalatore) |
| **PostCondizioni** | **Context:** SegnalazioneService:: puoSegnalare(Recensione recensione, Recensore segnalatore)  **Post:** // |

|  |  |
| --- | --- |
| **Nome metodo:** retrieveAll(): List<Segnalazione> | |
| **Descrizione metodo:** restituisce tutte le segnalazioni | |
| **PreCondizioni** | **Context:** SegnalazioneService:: retrieveall()  **Pre:** // |
| **PostCondizioni** | **Context:** SegnalazioneService:: retrieveall(): List<Segnalazione>  **Post:** // |

* 1. **MiPiaceService**

|  |
| --- |
| **Nome Classe:** MiPiaceService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante i mipiace |
| **Metodi:** + handleMiPiace(boolean b, Recensione recensione, Recensore recensore)  - aggiungiMiPiace(Recensione recensione, Recensore recensore, MiPiace mipiace)  - modificaMiPiace(boolean b, MiPiace daDatabase)  - togliMiPiace(Recensione recensione, Recensore recensore, MiPiace daDatabase)  + findMiPiacxeById(Recensore recensore, Recensione recensione)  + getNumeroMiPiaceOfRecensione(Recensione recensione)  + getNumeroNonMiPiaceOfRecensione(Recensione recensione) |

|  |  |
| --- | --- |
| **Nome Metodo:** handleMiPiace(boolean b, Recensione recensione, Recensore recensore): MiPiace | |
| **Descrizione Metodo:** se non esiste un mipiace di tipo b da parte del recensore alla recensione lo aggiunge, altrimenti se esiste un mipiace di tipo b da parte del recensore alla recensione lo elimina, altrimenti se esiste un mipiace non di tipo b da parte del recensore alla recensione cambia il tipo del mipiace | |
| **PreCondizioni** | **Context:** MiPiaceService::handleMiPiace(boolean b, Recensione recensione, Recensore recensore)  **Pre**: recensione != null && recensore != null |
| **PostCondizioni** | **Context:** MiPiaceService::handleMiPiace(boolean b, Recensione recensione, Recensore recensore): MiPiace  **Post**: il mipiace sarà aggiunto o eliminato o cambiato |
| **Eccezioni** | Se recensione == null || recensore == null throw **InvalidBeanException** |

|  |  |
| --- | --- |
| **Nome Metodo:** aggiungiMiPiace(Recensione recensione, Recensore recensore, MiPiace mipiace): MiPiace | |
| **Descrizione Metodo:** | |
| **PreCondizioni** | **Context:** MiPiaceService::aggiungiMiPiace(Recensione recensione, Recensore recensore, MiPiace mipiace)  **Pre**: // |
| **PostCondizioni** | **Context:** MiPiaceService::aggiungiMiPiace(Recensione recensione, Recensore recensore, MiPiace mipiace): MiPiace  **Post**: il mipiace sarà aggiunto alla tabella MiPiace del DB ed alle liste dei mipiace di recensione e recensore |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** modificaMiPiace(boolean b, MiPiace daDatabase): MiPiace | |
| **Descrizione Metodo:** cambia il tipo del mipiace “daDatabase” in b | |
| **PreCondizioni** | **Context:** MiPiaceService::modificaMiPiace(boolean b, MiPiace daDatabase)  **Pre**: // |
| **PostCondizioni** | **Context:** MiPiaceService::modificaMiPiace(boolean b, MiPiace daDatabase): MiPiace  **Post**: daDatabase.getTipo() == b |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** togliMiPiace(boolean b, MiPiace daDatabase): MiPiace | |
| **Descrizione Metodo:** cambia il tipo del mipiace “daDatabase” in b | |
| **PreCondizioni** | **Context:** MiPiaceService::modificaMiPiace(boolean b, MiPiace daDatabase)  **Pre**: // |
| **PostCondizioni** | **Context:** MiPiaceService::modificaMiPiace(boolean b, MiPiace daDatabase): MiPiace  **Post**: daDatabase.getTipo() == b |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findMiPiaceById(Recensore recensore, Recensione recensione): MiPiace | |
| **Descrizione Metodo:** restituisce il mipiace del recensore alla recensione se esiste | |
| **PreCondizioni** | **Context:** MiPiaceService::findMiPiaceById(Recensore recensore, Recensione recensione)  **Pre**: recensore != null && recensione != null && mipiace deve esistere |
| **PostCondizioni** | **Context:** MiPiaceService::findMiPiaceById(Recensore recensore, Recensione recensione): MiPiace  **Post**: // |
| **Eccezioni** | Se recensione == null || recensore == null throw **InvalidBeanException**  Se il mipiace non esiste throw **BeanNotExsistException** |

|  |  |
| --- | --- |
| **Nome Metodo:** getNumeroMiPiaceOfRecensione(Recensione recensione): Integer | |
| **Descrizione Metodo:** restituisce il numero di mipiacedi tipo true della recensione | |
| **PreCondizioni** | **Context:** MiPiaceService::getNumeroMiPiaceOfRecensione(Recensione recensione)  **Pre**: // |
| **PostCondizioni** | **Context:** MiPiaceService::getNumeroMiPiaceOfRecensione(Recensione recensione): Integer  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** getNumeroNonMiPiaceOfRecensione(Recensione recensione): Integer | |
| **Descrizione Metodo:** restituisce il numero di mipiacedi tipo false della recensione | |
| **PreCondizioni** | **Context:** MiPiaceService::getNumeroNonMiPiaceOfRecensione(Recensione recensione)  **Pre**: // |
| **PostCondizioni** | **Context:** MiPiaceService::getNumeroNonMiPiaceOfRecensione(Recensione recensione): Integer  **Post**: // |
| **Eccezioni** |  |

* 1. **UtenteService**

|  |
| --- |
| **Nome Classe:** UtenteService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante gli utenti |
| **Metodi:** + signUp(Utente utente)  + findByEmail(String email)  + findUserDetailsByEmail(String email)  + deleteUtente(Utente utente)  + saveRegisteredUser(Utente utente)  + getUtenteByVerificationToker(String verificationToken)  + getVerificationToken(String token)  + createVerificationToken(Utente utente, String token)  + bannaRecensore(String email) |

|  |  |
| --- | --- |
| **Nome Metodo:** signUp(Utente utente):Utente | |
| **Descrizione Metodo:** Questo metodo permette di rendere persistente un utente non attivo. Inoltre cripta la password scelta dall’utente. | |
| **PreCondizioni** | **Context:** UtenteService::signUp(Utente utente): utente  **Pre**: annoCorrente – utente.getDataNascita().getYear() >= 13  && (L’utente non deve esistere || L’utente non deve essere bannato) |
| **PostCondizioni** | **Context:** UtenteService::signUp(Utente utente): utente  **Post**: utente risulta reso persistente con password criptata e non attivo per 24 ore. |
| **Eccezioni** | Se annoCorrente – utente.getDataNascita().getYear() < 13  Throw **UserUnderAgeException()**  Se utente esiste e non è bannato throw **AlreadyExistsException**  Se utente esiste ed è bannato throw **BannedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** findByEmail(String email):Utente | |
| **Descrizione Metodo:** Data una mail restituisce l’utente associato a quella email se esiste. | |
| **PreCondizioni** | **Context:** UtenteService::findByEmail(String email)  **Pre**: email != null && !email.isBlank() && l’utente deve esistere |
| **PostCondizioni** | **Context:** UtenteService::findByEmail(String email)  **Post**: // |
| **Eccezioni** | Se email == null || email.isBlank()  Throw **InvalidBeanException()**  Se utente non esiste throw **BeanNotExistsException** |

|  |  |
| --- | --- |
| **Nome Metodo:** deleteUtente(Utente utente):void | |
| **Descrizione Metodo:** Questo metodo permette di cancellare un utente se esiste | |
| **PreCondizioni** | **Context:** UtenteService::deleteUtente(Utente utente): utente  **Pre**: utente deve esistere |
| **PostCondizioni** | **Context:** UtenteService::deleteUtente(Utente utente): void  **Post**: utente viene cancellato dal DBMS |
| **Eccezioni** | Se utente non esiste throw **BeanNotExistsException** |

|  |  |
| --- | --- |
| **Nome Metodo:** saveRegisteredUser(Utente utente):Utente | |
| **Descrizione Metodo:** Questo metodo permette di modificare un utente se questo è attivo | |
| **PreCondizioni** | **Context:** UtenteService::saveRegisteredUser(Utente utente): utente  **Pre**: l’utente deve esistere e l’utente deve essere attivo |
| **PostCondizioni** | **Context:** UtenteService::saveRegisteredUser(Utente utente): utente  **Post**: utente risulta modificato all’interno del DBMS. |
| **Eccezioni** | Se utente non esiste o non è attivo throw **InvalidBeanException** |

|  |  |
| --- | --- |
| **Nome Metodo:** bannaRecensore(String email):Utente | |
| **Descrizione Metodo:** Questo metodo permette di bannare un recensore. | |
| **PreCondizioni** | **Context:** UtenteService::bannaRecensore(String email): utente  **Pre**: email != null && !email.isBlank() && l’utente deve esistere |
| **PostCondizioni** | **Context:** UtenteService::bannaRecensore(String email):Utente  **Post**: utente risulta bannato |
| **Eccezioni** | Se email == null || email.isBlank() **InvalidBeanException()**  Se utente non esiste throw **BeanNotExistsException** |

* 1. **RecensoreService**

|  |
| --- |
| **Nome Classe:** RecensoreService |
| **Descrizione Classe:** Classe che contiene la logica di business riguardante i recensori |
| **Metodi:** + findAll()  + findAllNotBanned() |

|  |  |
| --- | --- |
| **Nome Metodo:** findAll(): List<Recensore> | |
| **Descrizione Metodo:** restituisce tutti i recensori | |
| **PreCondizioni** | **Context:** RecensoreService::findAll()  **Pre**: // |
| **PostCondizioni** | **Context:** RecensoreService::findAll(): List<Recensore>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findAllNotBanned(): List<Recensore> | |
| **Descrizione Metodo:** restituisce tutti i recensori non bannati | |
| **PreCondizioni** | **Context:** RecensoreService::findAllNotBanned()  **Pre**: // |
| **PostCondizioni** | **Context:** RecensoreService::findAllNotBanned(): List<Recensore>  **Post**: // |
| **Eccezioni** |  |

* 1. **UtenteControl**

|  |
| --- |
| **Nome Classe:** UtenteControl |
| **Descrizione Classe:** Classe control che si occupa di smistare tutte le operazioni riguardanti l’utente alla logica di business |
| **Metodi:** + getAllRecensori()  + getAllNotBannedRecensori()  + registrazioneUtente(Utente utente)  + confermaRegistrazione(String token) |

|  |  |
| --- | --- |
| **Nome Metodo:** getAllRecensori():List<Recensore> | |
| **Descrizione Metodo:** Questo metodo restituisce tutti i recensori | |
| **PreCondizioni** | **Context:** UtenteControl::getAllRecensori():List<Recensori>  **Pre**: // |
| **PostCondizioni** | **Context:** UtenteControl::getAllRecensori():List<Recensori>  **Post**: // |

|  |  |
| --- | --- |
| **Nome Metodo:** getAllNotBannedRecensori():List<Recensore> | |
| **Descrizione Metodo:** Questo metodo restituisce tutti i recensori | |
| **PreCondizioni** | **Context:** UtenteControl:: getAllNotBannedRecensori ():List<Recensori>  **Pre**: // |
| **PostCondizioni** | **Context:** UtenteControl:: getAllNotBannedRecensori ():List<Recensori>  **Post**: // |

|  |  |
| --- | --- |
| **Nome Metodo:** registrazioneUtente(Utente utente):Utente | |
| **Descrizione Metodo:** Questo metodo permette di registrare un utente, e lancia l’evento che invia l’email con il link di conferma | |
| **PreCondizioni** | **Context:** UtenteControl:: registrazioneUtente():Utente  **Pre**: utente deve essere valido |
| **PostCondizioni** | **Context:** UtenteControl:: registrazioneUtente():Utente  **Post**: utente in dbms non attivo. |

|  |  |
| --- | --- |
| **Nome Metodo:** confermaRegistrazione(String token):void | |
| **Descrizione Metodo:** Questo metodo, dato un token valido permette di rendere attivo l’utente associato a quel token. | |
| **PreCondizioni** | **Context:** UtenteControl:: confermaRegistrazione():void  **Pre**: token != null && token non scaduto |
| **PostCondizioni** | **Context:** UtenteControl:: confermaRegistrazione():void  **Post**: utente associato al token in dbms attivo |

* 1. **ModerazioneControl**

|  |
| --- |
| **Nome Classe:** ModerazioneControl |
| **Descrizione Classe:** Classe control che si occupa di smistare tutte le operazioni riguardanti la moderazione |
| **Metodi:** + addSegnalazione(Recensione recensione)  + bannaRecensore(String email)  + findAllSegnalazioni()  + deleteRecensione(Recensione recensione)  + isSegnalted(Recensione recensione) |

|  |  |
| --- | --- |
| **Nome Metodo:** findAllSegnalazioni():List<Segnalazione> | |
| **Descrizione Metodo:** Restituisce tutte le segnalazioni del sistema | |
| **PreCondizioni** | **Context:** ModerazioneControl:: findAllSegnalazioni():List<Segnalazione>  **Pre**: // |
| **PostCondizioni** | **Context:** ModerazioneControl:: findAllSegnalazioni():List<Segnalazione>  **Post**: // |

|  |  |
| --- | --- |
| **Nome Metodo:** isSegnalated():Boolean | |
| **Descrizione Metodo:** Data una recensione restituisce un booleano che indica se l’utente loggato può segnalare | |
| **PreCondizioni** | **Context:** ModerazioneControl:: isSegnalated():Boolean  **Pre**: L’utente deve essere loggato e deve essere un recensore |
| **PostCondizioni** | **Context:** ModerazioneControl:: issSegnalated():Boolean  **Post**: // |
| **Eccezioni** | Se l’utente non è loggato throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** addSegnalazione(Recensione recensione):void | |
| **Descrizione Metodo:** Data una recensione fa in modo che le venga associata una segnalazione da parte dell’utente loggato | |
| **PreCondizioni** | **Context:** ModerazioneControl:: addSegnalazione(Recensione recensione):void  **Pre**: L’utente deve essere loggato e deve essere un recensore |
| **PostCondizioni** | **Context:** ModerazioneControl:: hasSegnalated():Boolean  **Post**: l’autore della recensione segnalata avrà una segnalazione in più nella sua lista segnalazioni, e la recensione avrà una segnalazione in più nella sua lista delle segnalazioni |
| **Eccezioni** | Se l’utente non è loggato o non è un recensore throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** bannaRecensore(String email):void | |
| **Descrizione Metodo:** Data la mail di un recensore lo banna | |
| **PreCondizioni** | **Context:** ModerazioneControl:: addSegnalazione(Recensione recensione):void  **Pre**: L’utente deve essere loggato e deve essere un moderatore account |
| **PostCondizioni** | **Context:** ModerazioneControl:: hasSegnalated():Boolean  **Post**: Tutte le recensioni dell’utente bannato saranno censurate e accanto alla username dell’utente bannato comparirà la scritta [utente bannato] |
| **Eccezioni** | Se l’utente non è loggato o non è un moderatore account throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** deleteRecensione(Recensione recensione):void | |
| **Descrizione Metodo:** Data la mail di un recensore lo banna | |
| **PreCondizioni** | **Context:** ModerazioneControl:: addSegnalazione(Recensione recensione):void  **Pre**: L’utente deve essere loggato e deve essere un moderatore recensioni |
| **PostCondizioni** | **Context:** ModerazioneControl:: hasSegnalated():Boolean  **Post**: La recensione risulta censurata |
| **Eccezioni** | Se l’utente non è loggato o non è un moderatore recensioni throw **NotAuthorizedException** |

* 1. **CatalogoControl**

|  |
| --- |
| **Nome Classe:** CatalogoControl |
| **Descrizione Classe:** Classe control che si occupa di smistare le operazioni relative al catalogo, quindi:   1. aggiunta di recensioni, risposte e mi piace 2. retrive di film, serietv e puntate |
| **Metodi:** + addRecensione(Recensione recensione)  + requestRecensioneById(Long id)  + addMiPiace(boolean b, Recensione recensione)  + findMiPiaceById(Recensione recensione)  + getNumeroMiPiaceOfRecensione(Recensione recensione)  + getNumeroNonMiPiaceOfRecensione(Recensione recensione)  + rispondiARecensione(Recensione risposta)  + findAllFilm()  + findAllSerieTv()  + findMostRecentMedia(Integer howMany)  + puntateByStagione(Long idSerieTv, Integer numeroStagione)  + puntateBySerie(Long idSerieTv)  + findFilmById(Long id)  + findSerieTvById(Long id)  + findPuntataById(PuntataID puntataID)  + searchFilmByGenere(Collection<Genere> generi)  + searchSerieTVByGenere(Collection<Genere> generi)  + searchFilmByTitle(String titolo)  + findAllPuntate()  + findPuntataByTitle(String titolo)  + searchSerieTvByTitle(String titolo)  + findMostVoted() |

|  |  |
| --- | --- |
| **Nome Metodo:** addRecensione(Recensione recensione): void | |
| **Descrizione Metodo:** aggiunge una recensione scritta dall’utente loggato | |
| **PreCondizioni** | **Context:** CatalogoControl::addRecensione(Recensione recensione)  **Pre**: l’utente deve essere loggato e deve essere un recensore |
| **PostCondizioni** | **Context:** CatalogoControl::addRecensione(Recensione recensione)  **Post**: la recensione sarà resa persistente e saranno aggiornate le liste delle recensioni del recensore loggato e del recensibile a cui si riferisce. Inoltre, saranno aggiornati il voto medio del recensibile |
| **Eccezioni** | Se l’utente non è loggato throw **NotLoggedException**  Se l’utente loggato non è un recensore throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** requestRecensioneById(Long id): Recensione | |
| **Descrizione Metodo:** restituisce la recensione con id “id” | |
| **PreCondizioni** | **Context:** CatalogoControl:: requestRecensioneById(Long id)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: requestRecensioneById(Long id): Recensione  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** addMiPiace(boolean b, Recensione recensione): void | |
| **Descrizione Metodo:** aggiunge una mipiace di tipo b alla recensione da parte del recensore loggato | |
| **PreCondizioni** | **Context:** CatalogoControl::addMiPiace (boolean b, Recensione recensione)  **Pre**: l’utente deve essere loggato e deve essere un recensore |
| **PostCondizioni** | **Context:** CatalogoControl::addRecensione(boolean b, Recensione recensione): void  **Post**: il mipiace di tipo b saranno aggiunto alla recensione e saranno aggiornate le liste dei mipiace delle recensione e del recensore loggato |
| **Eccezioni** | Se l’utente non è loggato throw **NotLoggedException**  Se l’utente loggato non è un recensore throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** findMiPiaceById(Recensione recensione): MiPiace | |
| **Descrizione Metodo:** restituisce il mipiace fatto dal recensore loggato alla recensione se esiste | |
| **PreCondizioni** | **Context:** CatalogoControl::findMiPiaceById (Recensione recensione)  **Pre**: l’utente deve essere loggato e deve essere un recensore |
| **PostCondizioni** | **Context:** CatalogoControl::findMiPiaceById (Recensione recensione): MiPiace  **Post**: // |
| **Eccezioni** | Se l’utente non è loggato throw **NotLoggedException**  Se l’utente loggato non è un recensore throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** getNumeroMiPiaceOfRecensione(Recensione recensione): Integer | |
| **Descrizione Metodo:** restituisce il numero di mipiace di tipo true della recensione | |
| **PreCondizioni** | **Context:** CatalogoControl::getNumeroMiPiaceOfRecensione (Recensione recensione)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl::getNumeroMiPiaceOfRecensione (Recensione recensione): Integer  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** getNumeroNonMiPiaceOfRecensione(Recensione recensione): Integer | |
| **Descrizione Metodo:** restituisce il numero di mipiace di tipo falsedella recensione | |
| **PreCondizioni** | **Context:** CatalogoControl::getNumeroNonMiPiaceOfRecensione (Recensione recensione)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl::getNumeroNonMiPiaceOfRecensione (Recensione recensione): Integer  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** rispondiARecensione(Recensione risposta, Long idPadre): void | |
| **Descrizione Metodo:** aggiunge una risposta da parte del recensore loggato alla recensione con id “idPadre” | |
| **PreCondizioni** | **Context:**  CatalogoControl:: rispondiARecensione(Recensione risposta, Long idPadre)  **Pre**: l’utente deve essere loggato e deve essere un recensore |
| **PostCondizioni** | **Context:**  CatalogoControl:: rispondiARecensione(Recensione risposta, Long idPadre):void  **Post**: la risposta è resa persistente e viene aggiunta alla lista delle recensioni del recensore loggato ed alla lista delle risposte della recensione con id “idPadre” |
| **Eccezioni** | Se l’utente non è loggato throw **NotLoggedException**  Se l’utente loggato non è un recensore throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** findAllFilm(): List<Film> | |
| **Descrizione Metodo:** restituisce tutti i Film | |
| **PreCondizioni** | **Context:** CatalogoControl:: findAllFilm()  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: findAllFilm(): List<Film>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findAllSerieTv(): List<SerieTv> | |
| **Descrizione Metodo:** restituisce tutti le SerieTv | |
| **PreCondizioni** | **Context:** CatalogoControl:: findAllSerieTv()  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: findAllSerieTv(): List<SerieTv>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findMostRecentMedia(Integer howMany): List<Media> | |
| **Descrizione Metodo:** restituisce gli howMany media più recenti | |
| **PreCondizioni** | **Context:** CatalogoControl:: findMostRecentMedia(Integer howMany)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: findMostRecentMedia(Integer howMany): List<Media>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** puntateByStagione(Long idSerieTv, Integer numeroStagione): List<Puntata> | |
| **Descrizione Metodo:** restituisce tutte le puntate della stagione con numeroStagione “numeroStagione” della serietv con id “idSerieTv” | |
| **PreCondizioni** | **Context:** CatalogoControl:: puntateByStagione(Long idSerieTv, Integer numeroStagione)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: puntateByStagione(Long idSerieTv, Integer numeroStagione): List<Puntata>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** puntateBySerie(Long idSerieTv): List<Puntata> | |
| **Descrizione Metodo:** restituisce tutte le puntate della serietv con id “idSerieTv” | |
| **PreCondizioni** | **Context:** CatalogoControl:: puntateBySerie(Long idSerieTv)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: puntateBySerie(Long idSerieTv): List<Puntata>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findFilmById(Long id): Film | |
| **Descrizione Metodo:** restituisce il film con id “id” se esiste | |
| **PreCondizioni** | **Context:** CatalogoControl:: findFilmById(Long id)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: findFilmById(Long id): Film  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findSerieTvById(Long id): SerieTv | |
| **Descrizione Metodo:** restituisce la serietv con id “id” se esiste | |
| **PreCondizioni** | **Context:** CatalogoControl:: findSerieTvById(Long id)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: findSerieTvById(Long id): SerieTv  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findPuntataById(PuntataID puntataID): Puntata | |
| **Descrizione Metodo:** restituisce la puntata con id “puntataID” se esiste | |
| **PreCondizioni** | **Context:** CatalogoControl:: findPuntataById(Long id)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl:: findPuntataById(Long id): Puntata  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** searchFilmByGenere(Collection<Genere> generi): Collection<Film> | |
| **Descrizione Metodo:** restituisce i Film che hanno almeno un genere in generi | |
| **PreCondizioni** | **Context:**  CatalogoControl::searchFilmByGenere(Collection<Genere> generi)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl::searchFilmByGenere(Collection<Genere> generi): Collection<Film>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** searchSerieTvByGenere(Collection<Genere> generi): Collection<SerieTv> | |
| **Descrizione Metodo:** restituisce le SerieTv che hanno almeno un genere in generi | |
| **PreCondizioni** | **Context:**  CatalogoControl::searchSerieTvByGenere(Collection<Genere> generi)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl::searchSerieTvByGenere(Collection<Genere> generi): Collection<SerieTv>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** searchFilmByTitle(String titolo): List<Film> | |
| **Descrizione Metodo:** restituisce i film il cui titolo contiene “titolo” | |
| **PreCondizioni** | **Context:** CatalogoControl::searchFilmByTitle(String titolo)  **Pre**: // |
| **PostCondizioni** | **Context:**  CatalogoControl::searchFilmByTitle(String titolo): List<Film>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findAllPuntate(): List<Puntata> | |
| **Descrizione Metodo:** restituisce tutte le Puntate | |
| **PreCondizioni** | **Context:** CatalogoControl::findAllPuntate()  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl::finAllPuntate(): List<Puntata>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findPuntataByTitle(String titolo): List<Puntata> | |
| **Descrizione Metodo:** restituisce le puntate il cui titolo contiene “titolo” | |
| **PreCondizioni** | **Context:**  CatalogoControl::findPuntataByTitle(String titolo)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl::findPuntataByTitle(String titolo): List<Puntata>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** searchSerieTvByTitle(String titolo): List<SerieTv> | |
| **Descrizione Metodo:** restituisce le serietv il cui titolo contiene “titolo” | |
| **PreCondizioni** | **Context:**  CatalogoControl::searchSerieTvByTitle(String titolo)  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl::searchSerieTvByTitle(String titolo): List<SerieTv>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** findMostVoted(): List<Media> | |
| **Descrizione Metodo:** restituisce tutti i media ordinati per voto medio decrescente | |
| **PreCondizioni** | **Context:** CatalogoControl::findMostVoted()  **Pre**: // |
| **PostCondizioni** | **Context:** CatalogoControl::findMostVoted(): List<Media>  **Post**: // |
| **Eccezioni** |  |

* 1. **GestioneCatalogoControl**

|  |
| --- |
| **Nome Classe:** GestioneCatalogoControl |
| **Descrizione Classe:** Classe control che si occupa di smistare le operazioni relative alla gestione del catalogo, quindi le operazioni di aggiunta, rimozione e modifica di Film, SerieTv, Puntata, Cast |
| **Metodi:** + addFilm(Film film)  + addSerieTV(SerieTV serieTv)  + addPuntata(Puntata puntata, Long idSerieTv, Integer numeroStagione)  + addCast(Cast cast)  + addRuolo(Ruolo ruolo, Long castId, Long mediaId)  + findAllCast()  + updateFilm(Film film)  + updateSerieTv(SerieTv serieTv)  + updatePuntata(Puntata puntata)  + removeFilm(Long id)  + removeSerieTV(Long id)  + removePuntata(PuntataID id)  + removeCast(Long id)  + addGeneriFilm(Collection<Genere> generi, Long id)  + addGeneriSerieTv(Collection<Genere> generi, Long id)  + findRecensioniByMiPiace(Recensibile recensibile)  + requestAllRecensioni() |

|  |  |
| --- | --- |
| **Nome Metodo:** addFilm(Film film): Film | |
| **Descrizione Metodo:** aggiunge un film al catalogo | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::addFilm(Film film)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::addFilm(Film film): Film  **Post**: il film è aggiunto alla tabella Film del DB |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** addSerieTv(SerieTv serietv): SerieTv | |
| **Descrizione Metodo:** aggiunge una serietv al catalogo | |
| **PreCondizioni** | **Context:**  GestioneCatalogoControl::addSerieTV(SerieTv serietv)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::addSerieTv(SerieTv serietv): SerieTv  **Post**: la serietv è aggiunta alla tabella SerieTv del DB |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** addPuntata(Puntata puntata, Long idSerieTv, Integer numeroStagione): void | |
| **Descrizione Metodo:** aggiunge una puntata alla stagione con numeroStagione “numeroStagione” della serietv con id “idSerieTv” | |
| **PreCondizioni** | **Context:**  GestioneCatalogoControl::addPuntata(Puntata puntata, Long idSerieTv, Integer numeroStagione)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:**  GestioneCatalogoControl::addPuntata(Puntata puntata, Long idSerieTv, Integer numeroStagione): void  **Post**: la puntata è aggiunta alla tabella Puntata del DB ed è stata aggiunta alla lista delle puntate della stagione con numeroStagione “numeroStagione” della serietv con id “idSerieTv” |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** addCast(Cast cast): void | |
| **Descrizione Metodo:** aggiunge un membro del cast al catalogo | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::addCast(Cast cast)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::addCast(Cast cast): void  **Post**: il cast è aggiunto alla tabella Cast del DB |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** addRuolo(Collection<Ruolo> ruoli, Long castId, Long mediaId): void | |
| **Descrizione Metodo:** aggiunge un ruolo del membro del cast con id “castId” al media con id “mediaId” | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::addRuolo(Collection<Ruolo> ruoli, Long castId, Long mediaId)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::addRuolo(Collection<Ruolo> ruoli, Long castId, Long mediaId): void  **Post**: i ruoli sono aggiunti alla tabella Ruolo del DB ed alle liste dei ruoli del Media e del Cast |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** findAllCast(): List<Cast> | |
| **Descrizione Metodo:** restituisce tutti i cast | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::findAllCast()  **Pre**: // |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::findAllCast(): List<Cast>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** updateFilm(Film film): void | |
| **Descrizione Metodo:** aggiorna lo stato di un Film | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::updateFilm(Film film)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::updateFilm(Film film): void  **Post**: lo stato di film è stato aggiornato |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** updateSerieTv(SerieTv serieTv): void | |
| **Descrizione Metodo:** aggiorna lo stato di una SerieTv | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::updateSerieTv(SerieTv serieTv)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::updateFilm(SerieTv serieTv): void  **Post**: lo stato della serietv è stato aggiornato |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** updatePuntata(Puntata puntata): void | |
| **Descrizione Metodo:** aggiorna lo stato di una Puntata | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::updatePuntata(Puntata puntata)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::updatePuntata(Puntata puntata): void  **Post**: lo stato della puntata è stato aggiornato |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** removeFilm(Long id): void | |
| **Descrizione Metodo:** rimuove il film con id “id” | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::removeFilm(Long id)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::removeFilm(Long id): void  **Post**: il film con id “id” è stato rimosso dal DB |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** removeSerieTv(Long id): void | |
| **Descrizione Metodo:** rimuove la serietv con id “id” | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::removeSerieTv(Long id)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::removeSerieTv(Long id): void  **Post**: la serietv con id “id” è stata rimossa dal DB |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** removePuntata(PuntataID id): void | |
| **Descrizione Metodo:** rimuove la puntata con id “id” | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::removePuntata(PuntataID id)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::removePuntata(PuntataID id): void  **Post**: la puntata con id “id” è stata rimossa dal DB |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

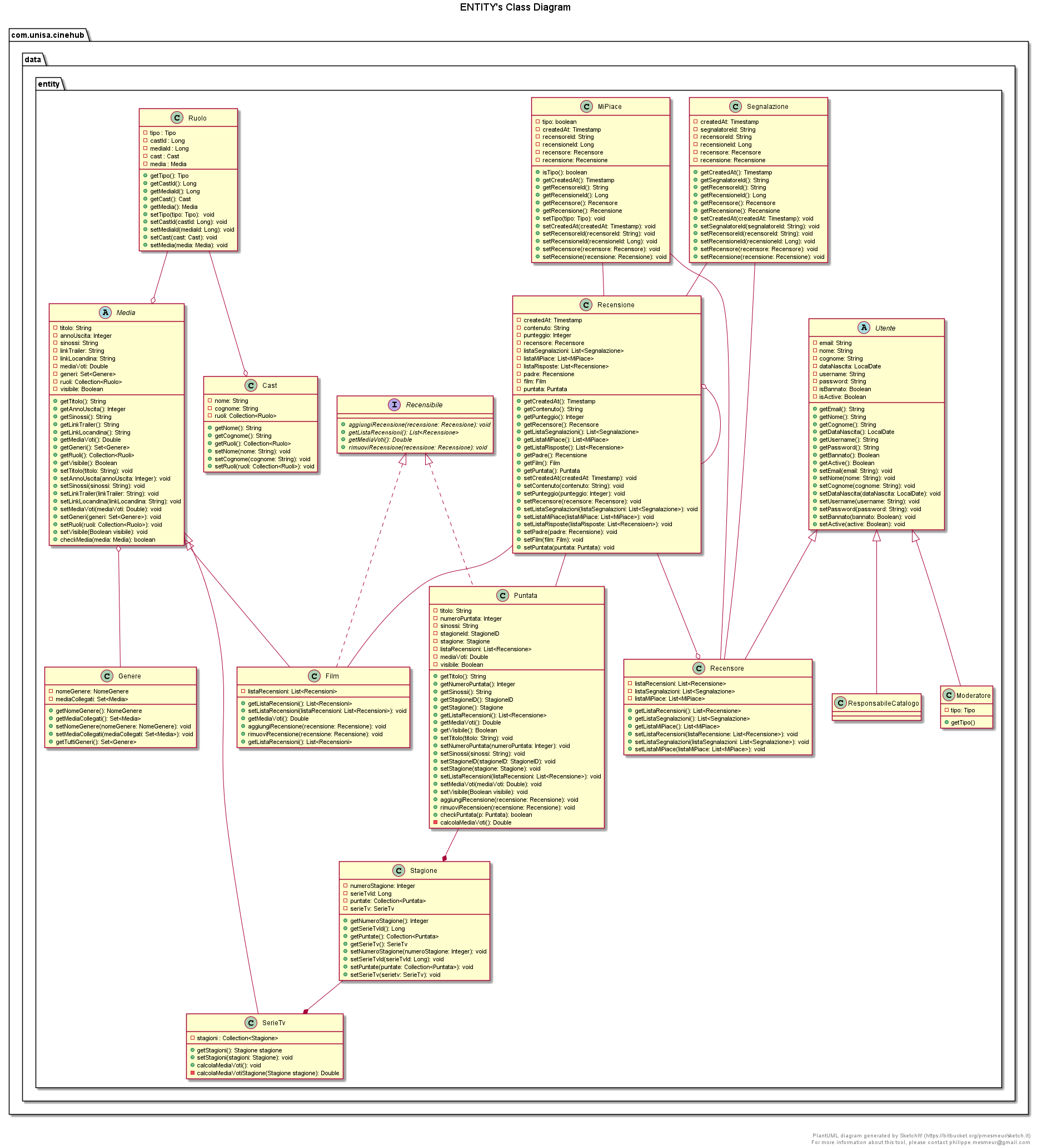
|  |  |
| --- | --- |
| **Nome Metodo:** removeCast(Long id): void | |
| **Descrizione Metodo:** rimuove il cast con id “id” | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::removeCast(Long id)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:**  GestioneCatalogoControl::removeCast(Long id): void  **Post**: il cast con id “id” è stato rimosso dal DB |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** addGeneriFilm(Collection<Genere> generi, Long id): void | |
| **Descrizione Metodo:** aggiunge i generi al film con id “id” | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::addGeneriFilm(Collection<Genere> generi, Long id)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:**  GestioneCatalogoControl::addGeneriFilm(Collection<Genere> generi, Long id): void  **Post**: il film con id “id” ha come generi quelli passati in input |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** addGeneriSerieTv(Collection<Genere> generi, Long id): void | |
| **Descrizione Metodo:** aggiunge i generi alla serietv con id “id” | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::addGeneriSerieTv(Collection<Genere> generi, Long id)  **Pre**: l’utente deve essere loggato e deve essere un gestore del catalogo |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::addGeneriSerieTv(Collection<Genere> generi, Long id): void  **Post**: la serietv con id “id” ha come generi quelli passati in input |
| **Eccezioni** | Se l’utente non è un gestore del catalogo throw **NotAuthorizedException** |

|  |  |
| --- | --- |
| **Nome Metodo:** findRecensioniByMiPiace(Recensibile recensibile): List<Recensione> | |
| **Descrizione Metodo:** dato un recensibile restituisce le recensioni ordinate per numero di mipiace | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::findRecensioniByMiPiace(Recensibile recensibile)  **Pre**: // |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::findRecensioniByMiPiace(Recensibile recensibile): List<Recensione>  **Post**: // |
| **Eccezioni** |  |

|  |  |
| --- | --- |
| **Nome Metodo:** requestAllRecensioni(): List<Recensione> | |
| **Descrizione Metodo:** restituisce tutte le recensioni | |
| **PreCondizioni** | **Context:** GestioneCatalogoControl::requestAllRecensioni()  **Pre**: // |
| **PostCondizioni** | **Context:** GestioneCatalogoControl::requestAllRecensioni(): List<Recensione>  **Post**: // |
| **Eccezioni** |  |

1. **Class Diagram Finale**

Altri diagrammi si trovano nella cartella “images” in “derivables”.