Accordo di Programma Quadro in materia di "e-Government e Società dell'Informazione" nella Regione Puglia

SAUSSC: Sistema di Accesso Unificato dei Servizi Sanitari per il Cittadino

Architettura Software Gestione Richieste Segnalazioni

Indice

Indice delle Figure

1. Prefazione

1.1. Storia del Documento

Revisione	Descrizione delle modifiche
V1	30/04/2010 – Versione finale

1.2. Allegati

Codice	Nome allegato
[Rif. 1]	SAUSSC_ArchitetturaSoftwareRichiesteSegnalazioni_integrazione

1.3. Lista degli Acronimi

Acronimo	Descrizione
UML	**U**nified **M**odeling **L**anguage
IoC	**I**nversion **O**f **C**ontrol

2. Scopo del documento

Il presente documento, nell'ambito della fornitura del sistema "SAUSSC: Sistema di Accesso Unificato dei Servizi Sanitari per il Cittadino", fornisce le specifiche tecniche del sistema "Gestione Richieste e Segnalazioni" in termini di UML diagram.

3. Architettura

3.1. Architettura software applicativo gestione richieste e segnalazioni

Di seguito verrà descritta l'architettura software delle componenti dell'applicativo. In particolare verranno mostrati gli use case i sequence diagram e i class diagram delle funzionalità del sistema.
Va precisato che visto l'utilizzo del framework spring tutte le operazioni CRUD possono essere rappresentate da due scenari derivanti da due classi astratte di conseguenza, prima di proseguire, verrà mostrato il class diagram delle classi astratte con dettaglio sui metodi definiti da queste classi.
Per le operazioni CRUD viene utilizzata una implementazione dell'interfaccia IGenericEntityManager; quindi oltre alle classe astratte verrà descritta anche questa interfaccia.
Per maggiori dettagli ci si può riferire al Javadoc allegato
Per maggiori dettagli ci si può riferire al javadoc consegnato insieme ai documenti. Definite le classi astratte nel prosieguo saranno indicate solo indicando il nome della classe.

3.1.1. Menu delle funzionalità

La tabella seguente riporta il dettaglio delle funzionalità presenti nel menù dell'area gestionale, raggruppate per voci di I livello.

Voce di I livello	Voce di II livello	Descrizione	Elenco Ruoli
GESTIONE SEGNALAZIONI	Registrazione segnalazione	Permette all'utente collegato l'inserimento di una segnalazione per conto di un cittadino	• rs_operatore • rs_supervisore • rs_redattore
	Assegnazione segnalazioni	Permette l'assegnazione delle segnalazioni agli operatori o supervisori o redattori o esperti del sistema	• rs_supervisore
	Gestione segnalazioni	Permette all'utente loggato la gestione delle segnalazioni ad esso assegnate	• rs_operatore • rs_supervisore • rs_redattore

	Situazione segnalazioni	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_supervisore • rs_redattore
	Ricerca segnalazioni	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_operatore • rs_supervisore • rs_redattore
GESTIONE RICHIESTE	Registra richiesta	Permette all'utente loggato la registrazione di una richiesta a nome di un cittadino	• rs_supervisore • rs_operatore • rs_redattore
	Assegnazione richieste	Permette l'assegnazione delle segnalazioni agli operatori o supervisori o redattori o esperti del sistema	• rs_supervisore
	Richieste esperto	Permette la gestione delle richieste all'esperto loggato sul sistema	• rs_supervisore • rs_esperto
	Gestione richieste	Permette la gestione delle richieste all'operatore loggato sul sistema	• rs_supervisore • rs_operatore • rs_redattore
	Situazione richieste	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_supervisore
	Ricerca richieste	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_supervisore • rs_operatore
GESTIONE TABELLE SISTEMA	Gestione comuni	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) dei comuni	• rs_supervisore
	Gestione tempi di risposta	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) dei tempi risposta	• rs_supervisore
	Gestione modalità ricezione/risposta	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) della modalità ricezione risposta	• rs_supervisore
	Gestione professioni	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) delle professioni	• rs_supervisore

Voce di I livello	Voce di II livello	Descrizione	Elenco Ruoli
GESTIONE SEGNALAZIONI	Registrazione segnalazione	Permette all'utente collegato l'inserimento di una segnalazione per conto di un cittadino	• rs_operatore • rs_supervisore • rs_redattore
	Assegnazione segnalazioni	Permette l'assegnazione delle segnalazioni agli operatori o supervisori o redattori o esperti del sistema	• rs_supervisore
	Gestione segnalazioni	Permette all'utente loggato la gestione delle segnalazioni ad esso assegnate	• rs_operatore • rs_supervisore • rs_redattore
	Situazione segnalazioni	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_supervisore • rs_redattore
	Ricerca segnalazioni	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_operatore • rs_supervisore • rs_redattore
	Gestione titoli di studio	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) dei titoli di studio	• rs_supervisore
	Gestione categorie	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) delle categorie	• rs_supervisore
	Gestione sub categorie	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) delle sub categorie	• rs_supervisore
	Gestione URP	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) degli URP	• rs_supervisore

Voce di I livello	Voce di II livello	Descrizione	Elenco Ruoli
GESTIONE SEGNALAZIONI	Registrazione segnalazione	Permette all'utente collegato l'inserimento di una segnalazione per conto di un cittadino	• rs_operatore • rs_supervisore • rs_redattore
	Assegnazione segnalazioni	Permette l'assegnazione delle segnalazioni agli operatori o supervisori o redattori o esperti del sistema	• rs_supervisore
	Gestione segnalazioni	Permette all'utente loggato la gestione delle segnalazioni ad esso assegnate	• rs_operatore • rs_supervisore • rs_redattore
	Situazione segnalazioni	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_supervisore • rs_redattore
	Ricerca segnalazioni	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_operatore • rs_supervisore • rs_redattore
	Gestione tipo struttura	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) dei tipi struttura	• rs_supervisore
	Gestione tipo segnalazione	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) dei tipi segnalazione	• rs_supervisore
	Gestione tipo richiesta	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) dei tipi richiesta	• rs_supervisore
	Gestione stato segnalazione	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) degli stati della segnalazione	• rs_supervisore

Voce di I livello	Voce di II livello	Descrizione	Elenco Ruoli
GESTIONE SEGNALAZIONI	Registrazione segnalazione	Permette all'utente collegato l'inserimento di una segnalazione per conto di un cittadino	• rs_operatore • rs_supervisore • rs_redattore
	Assegnazione segnalazioni	Permette l'assegnazione delle segnalazioni agli operatori o supervisori o redattori o esperti del sistema	• rs_supervisore
	Gestione segnalazioni	Permette all'utente loggato la gestione delle segnalazioni ad esso assegnate	• rs_operatore • rs_supervisore • rs_redattore
	Situazione segnalazioni	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_supervisore • rs_redattore
	Ricerca segnalazioni	Permette la ricerca ed il controllo delle segnalazioni registrate nel sistema	• rs_operatore • rs_supervisore • rs_redattore
	Gestione tipo richiesta	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) dei tipi richiesta	• rs_supervisore
	Gestione stato segnalazione	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) degli stati segnalazione	• rs_supervisore
	Gestione stato richiesta	Permette al supervisore loggato sul sistema la gestione (creazione, modifica, cancellazione) degli stati richiesta	• rs_supervisore

Tabella 1 Menu delle funzionalità

3.1.2. Classi astratte

3.1.2.1. AbstracFormController

Questa classe astratta viene utilizzata per operazioni semplici quali un submit o il rendering di form

3.1.2.1.1. Class Diagram

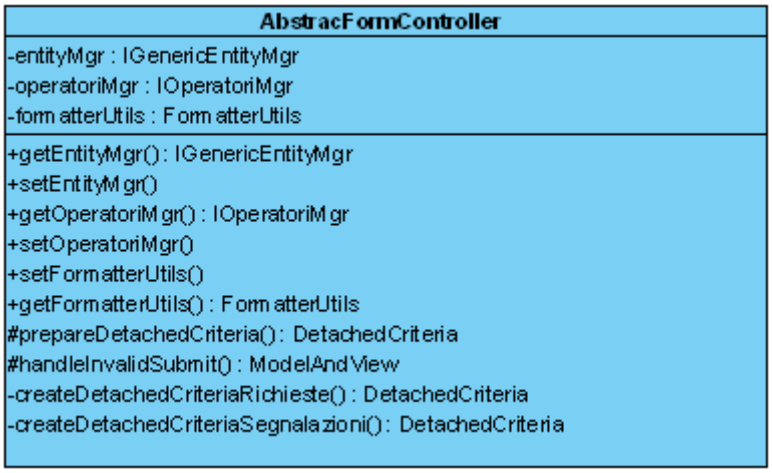

Figura 1 Class Diagramm AbstractFormController

3.1.2.1.2. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **entityMgr:** oggetto che si preoccupa di effettuare tutte le operazioni CRUD sul database.
- **operatoriMgr:** oggetto che permette di ricercare su OID di oracle gli utenti con opportuni ruoli.
- **formatterUtils:** oggetto che ha delle utility per la formattazione delle date e per la paginazione.
- **getEntityMgr():** restituisce l'entityMgr per effettuare le operazioni CRUD
- **setEntityMgr(IGenericEntityMgr entityMgr):** valorizza l'entityMgr
- **getOperatoriMgr():** restituisce l'operatoriMgr per effettuare la lettura sull'OID oracle
- **setOperatoriMgr(IOperatoriMgr operatoriMgr):** valorizza l'operatoriMgr
- **setFormatterUtils(FormatterUtils formatterUtils):** valorizza il formatterUtils
- **getFormatterUtils():** restituisce il formatterUtils
- **prepareDetachedCriteria(String tipoRichiesta, String username):** Costruisce un DetachedCriteria in base alla tipologia di oggetto che si vuole. Se il tipo è TIPOLOGIA_RICHIESTA viene preparato un DetachedCriteria per l'oggetto di tipo

Richiesta; se il tipoRichiesta è TIPOLOGIA_SEGNALAZIONE viene preparato un DetachedCriteria per l'oggetto di tipo Segnalazione

- **handleInvalidSubmit(HttpServletRequest request, HttpServletResponse response) throws Exception:** gestisce I submit invalidi (ad esempio refresh della pagina) per evitare ulteriori operazioni sul database.
- **createDetachedCriteriaRichieste(String username):** Prepara il DetachedCriteria per gli oggetti di tipo Richiesta Se username è valorizzata si filtra sul campo operatoreBackOffice
- **createDetachedCriteriaSegnalazioni(String username):** Prepara il DetachedCriteria per gli oggetti di tipo Segnalazione Se username è valorizzata si filtra sul campo assegnataA

3.1.2.2. SAUSSCAbstracWizardFormController

Questa classe astratta viene utilizzata nel caso in cui si debbano gestire dei wizard (ad esempio nel caso delle ricerche con paginazioni) ed offre tutte le utility per gestire i vari step del wizard.

3.1.2.2.1. Class diagram

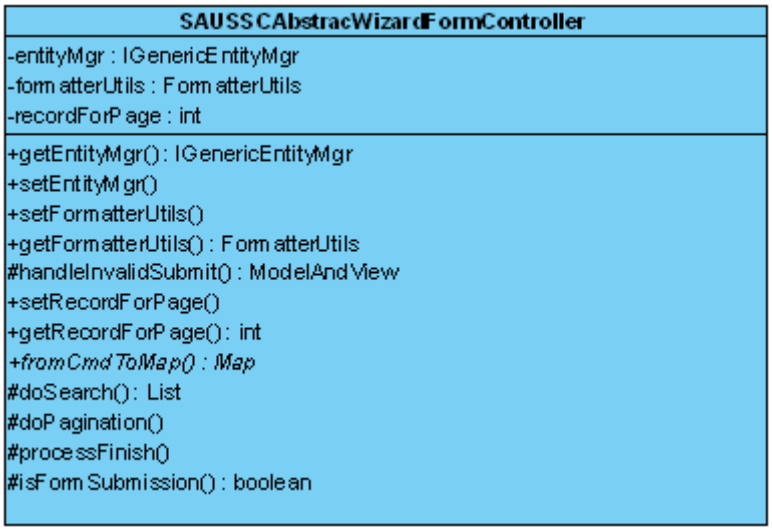

Figura 2 class diagram SAUSSCAbstracWizardFormController

3.1.2.2.2. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **entityMgr:** oggetto che si preoccupa di effettuare tutte le operazioni CRUD sul database.
- **formatterUtils:** oggetto che ha delle utility per la formattazione delle date e per la paginazione.
- **recordForPage:** numero di record per pagina
- **getEntityMgr():** restituisce l'entityMgr per effettuare le operazioni CRUD
- **setEntityMgr(IGenericEntityMgr entityMgr):** valorizza l'entityMgr

- **setFormatterUtils(FormatterUtils formatterUtils):** valorizza il formatterUtils
- **getFormatterUtils():** restituisce il formatterUtils
- **setRecordForPage(int recordForPage):** valorizza il numero di record per pagina
- **getRecordForPage():** restituisce il numero di record per pagina.
- **handleInvalidSubmit(HttpServletRequest request, HttpServletResponse response) throws Exception:** gestisce I submit invalidi (ad esempio refresh della pagina) per evitare ulteriori operazioni sul database.
- **fromCmdToMap:** passato un oggetto che rappresenta il form HTML di riferimento restituisce in uscita un Map opportunamente valorizzato per eseguire le queries sul database. Tutte le classi che estendono questa classe astratta devono implementare questo metodo
- **doSearch(Class clazz, Object o, int from, int to, String orderByFieldName):** metodo che effettua la ricerca sul DB.
- **doPagination(HttpServletRequest request, Errors errors, int page, Map model, AbstractRicercaCommand cmd, String paramEncoderName, Class clazz, String orderByFieldName,boolean inizioNew):** In questo metodo viene eseguita

- la logica per la paginazione con display tag; se paramencodername è nullo o vuoto viene sollevata una IllegalArgumentException
- **processFinish(HttpServletRequest request, HttpServletResponse response, Object command, BindException errors) throws Exception:** metodo che fa capire a Spring di essere arrivati all'ultimo passo del Wizard
- **isFormSubmission(HttpServletRequest request):** fa capire al framework che c'è stato un submit da parte del client.

3.1.3. Interfaccia IGenericEntityManager

3.1.3.1. Class diagram

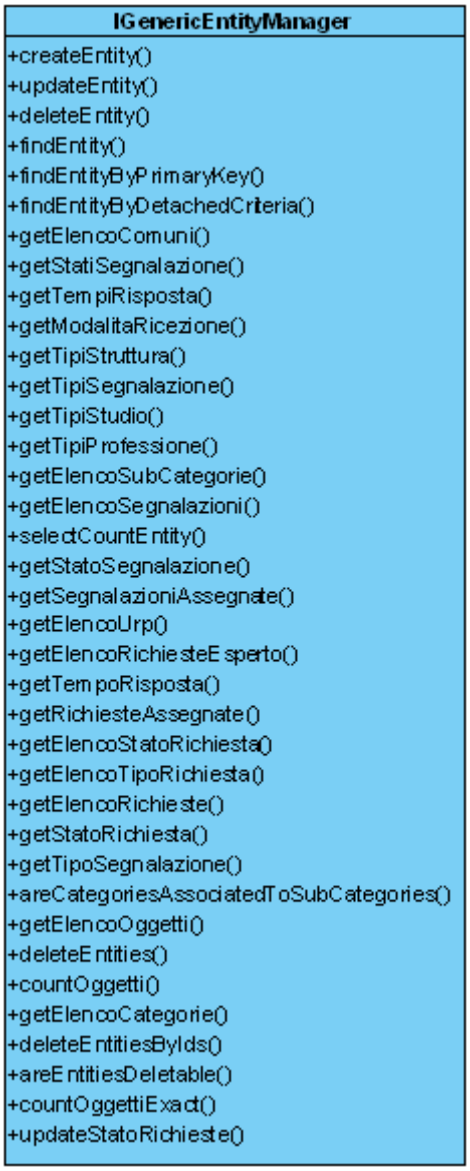

Figura 3 Class diagram IGenericEntityManager

3.1.3.2. Descrizione metodi

- **createEntity(GenericEntity entity):** si preoccupa di rendere persistente l'entità passata.
- **updateEntity(GenericEntity entity):** si preoccupa di aggiornare l'entità passata
- **deleteEntity(GenericEntity entity):** si preoccupa di cancellare l'entità passata
- **findEntity(Class entityClass, Map criteria, String orderBy):** ricerca tutte le entità del tipo passato in ingresso eventualmente filtrando in base al Map passato in ingresso ed ordinando in base al valore della stringa or derBy
- **findEntityByPrimaryKey(Class entityClass, Serializable key, String idPropertyName):** ricerca l'entità individuate dalla primaryKey passata in ingresso
- **findEntityByDetachedCriteria(DetachedCriteria criteria):** ricerca in base al detachedCriteria passato
- **getElencoComuni:** restituisce l'elenco di comuni
- **getStatiSegnalazione:** restituisce l'elenco degli stati della segnalazione
- **getTempiRisposta:** restituisce i tempi di risposta
- **getModalitaRicezione:** restituisce le modalità di ricezione delle segnalazioni/richieste
- **getTipiStruttura:** restituisce i tipi di struttura
- **getTipiSegnalazione:** restituisce i tipi di segnalazione
- **getTipiStudio:** restituisce i titoli di studio
- **getTipiProfessione:** restituisce i tipi di professione
- **getElencoSubCategorie:**restituisce le sub categorie
- **getElencoSegnalazioni(Map criteria, int from, int to):** restituisce l'elenco delle segnalazioni dal punto iniziale specificato dall'intero from al punto finale specificato da to; se il map di criteri è valorizzato vengono effettuati i filtri indicati
- **selectCountEntity (Class clazz, Map criteria):**effettua una select count sull'oggetto persistente specificato dalla classe; se il map di criteri è valorizzato vengono effettuati i filtri indicati
- **getStatoSegnalazione(String descrizione):** restituisce lo stato segnalazione specificata dalla descrizione breve passata in ingresso.
- **getSegnalazioniAssegnate(String username):** restituisce l'elenco delle segnalazioni assegnate all'utente individuato dalla username passata in ingresso.
- **getElencoUrp:** restituisce l'elenco degli URP
- **getElencoMomentiGiornata:** restituisce l'elenco dei momenti della giornata
- **getElencoRichiesteEsperto(String usernameEsperto):** restituisce l'elenco delle richieste assegnate all'esperto individuato dalla username passata
- **getTempoRisposta(String descrizione):** restituisce il tempo di risposta definito dalla descrizione breve passata in ingresso.
- **getRichiesteAssegnate(String username):** restituisce le richieste assegnate all'operatore individuato dalla username passata.
- **getElencoStatoRichiesta:** restituisce l'elenco degli stati richiesta
- **getElencoTipoRichiesta:** restituisce l'elenco dei tipi di richiesta
- **getElencoRichieste(Map criteria, int from, int to):** restituisce l'elenco delle richieste dal punto iniziale specificato dall'intero from al punto finale specificato da to; se il map di criteri è valorizzato vengono effettuati i filtri indicati

- **getStatoRichiesta(String descrBreve):** restituisce lo stato richiesta definito dalla descrizione breve passata in ingresso

- **getTipoSegnalazione(String descrBreve):**restituisce il tipo segnalazione definito dalla descrizione breve passata in ingresso
- **areCategoriesAssociatedToSubCategories(List categoryIds):** controlla se le categorie specificate dalla lista di id passata in ingresso sono associate a delle sub categorie
- **getElencoOggetti(Class clazz, Map criteria, int from, int to, String orderByFieldName):** ottengo l'elenco di oggetti persistenti descritti dalla classe passata in ingresso partendo dal punto iniziale specificato dall'intero from al punto finale specificato da to; se il map di criteri è valorizzato vengono effettuati i filtri indicati
- **deleteEntities(List entities):** cancella le entità persistenti passate in ingresso
- **countOggetti(Class clazz, Map criter):** effettua una select count sugli oggetti persistenti di tipo pari alla classe passata in ingresso; se il map di criteri è valorizzato vengono effettuati i filtri indicati
- **getElencoCategorie:** restituisce l'elenco delle categorie
- **deleteEntitiesByIds(List entitiesIds, Class clazz, String idPropertyName):** cancella tutte le entità persistenti di tipo pari alla classe passata individuate dalla lista di id passata andando ad agire sul campo primary key specificato dalla string idPripertyName
- **areEntitiesDeletable(List idEntitaDaCancellare, Class clazz, String alias, String newAliasName, String nomeProprieta):** controlla se le entità del tipo definito dalla classe passata in ingresso sono cancellabili o meno
- **countOggettiExact(Class clazz, Map criter):** effettua una select count sugli oggetti persistenti di tipo pari alla classe passata in ingresso; se il map di criteri è valorizzato vengono effettuati i filtri indicati. A differenza di prima i criteri sono effettuati in EXACT_MODE
- **updateStatoRichieste(List idRichieste, StatoRichiesta nuovoStato):**aggiorna l'update dello stato richiesta di tutte le richieste indicate nella lista passata. Lo stato passa al nuovo stato passato

3.1.4. Architettura componente inserimento segnalazione

3.1.4.1. Obiettivi e vincoli funzionali

L'obiettivo della componente è di permettere ad un utente del portale l'inserimento di una segnalazione o in forma anonima o trasmettendo i propri dati anagrafici. L'utente è obbligato a valorizzare tutti i campi obbligatori della segnalazione.

3.1.4.2. Vista dei casi d'uso

Di seguito viene riportato lo Use Case della componente:

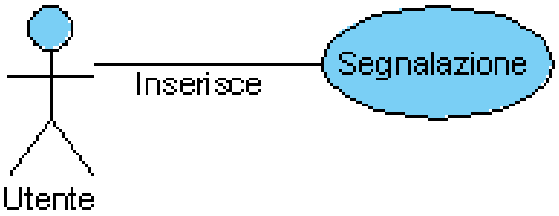

Figura 4 UDC: Inserimento segnalazione

Inserimento Segnalazione	Viene visualizzato il form con i dati da valorizzare. Viene creata la segnalazione e viene presentato all'utente l'esito dell'operazione

3.1.4.3. Vista Logica

3.1.4.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

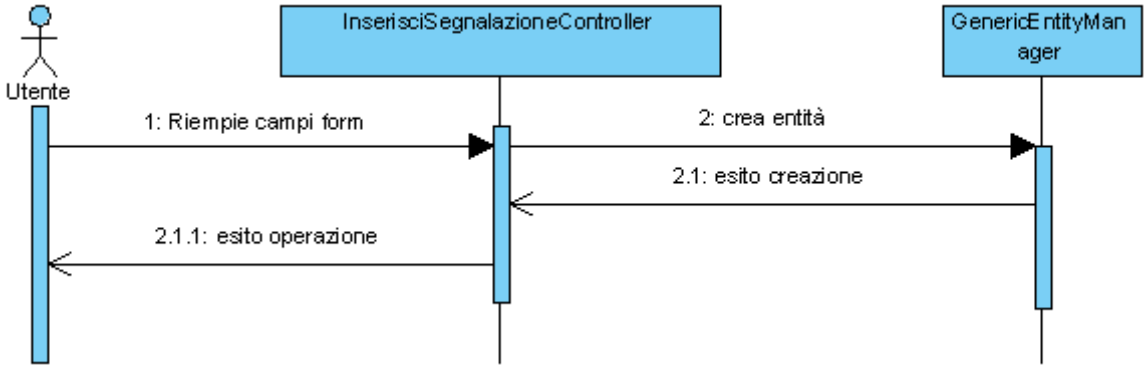

Figura 5 Sequence Diagram Inserimento Segnalazione

Class diagram

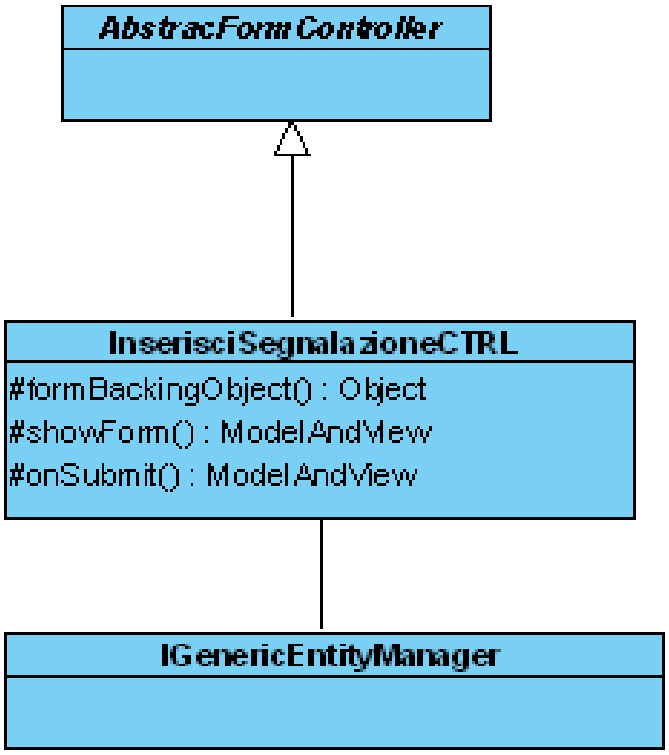

Figura 6 Class Diagram Inserimento Segnalazione

3.1.4.3.2. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **showForm(HttpServletRequest request, HttpServletResponse response, BindException errors, Map controlModel) throws Exception:** metodo richiamato subito prima del rendering del form HTML per effettuare altre operazioni non necessarie nel formBackingObject
- **onSubmit(HttpServletRequest request, HttpServletResponse response, Object command, BindException errors) throws Exception:** chiamato al submit del form; in questo metodo viene effettuata tutta la logica di business della funzionalità.

3.1.5. Architettura componente assegnazione segnalazione

3.1.5.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è i permettere all'utente con opportuni ruoli l'assegnazione delle segnalazioni agli utenti selezionati. Il vincolo è che ci siano effettivamente segnalazioni da assegnare.

3.1.5.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

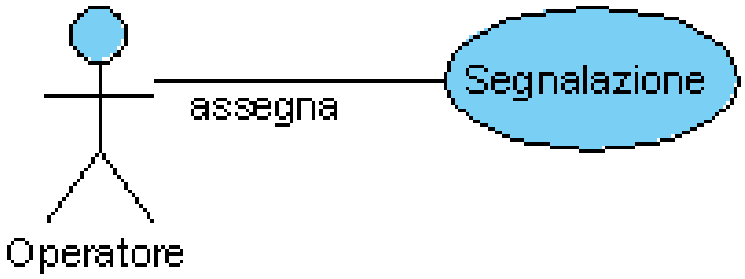

Figura 7 Use case assegnazione segnalazione

Assegnazione Segnalazioni	Viene visualizzato il form il numero di segnalazioni aperte e non assegnate e l'elenco degli operatori a assegnare la segnalazione. Scelti gli operatori vengono assegnate le segnalazioni e viene presentato il form con il risultato all'utente. Le segnalazioni vengono distribuite uniformemente tra gli utenti selezionati.

3.1.5.3. Vista logica

3.1.5.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

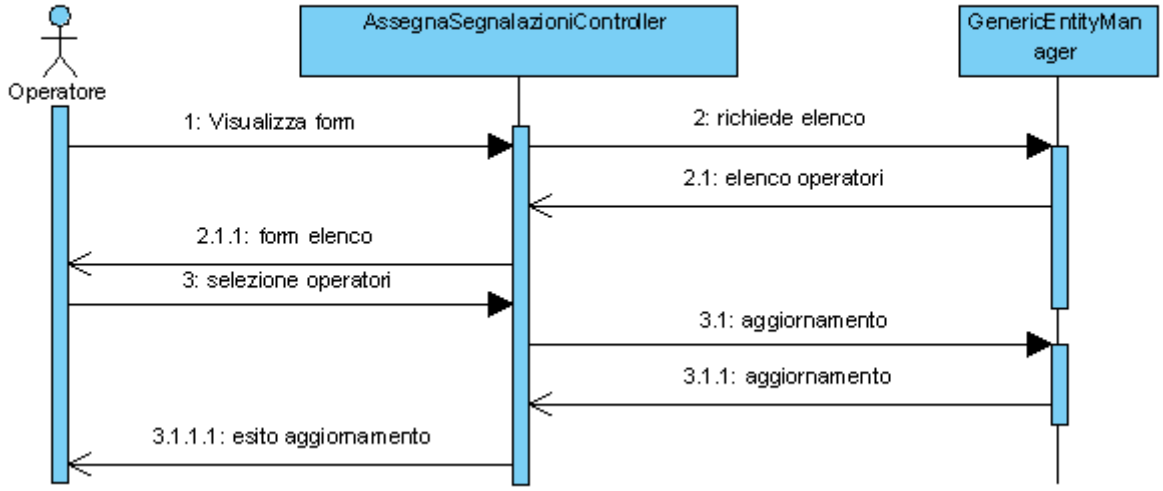

Figura 8 Sequence diagram assegnazione segnalazioni

3.1.5.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

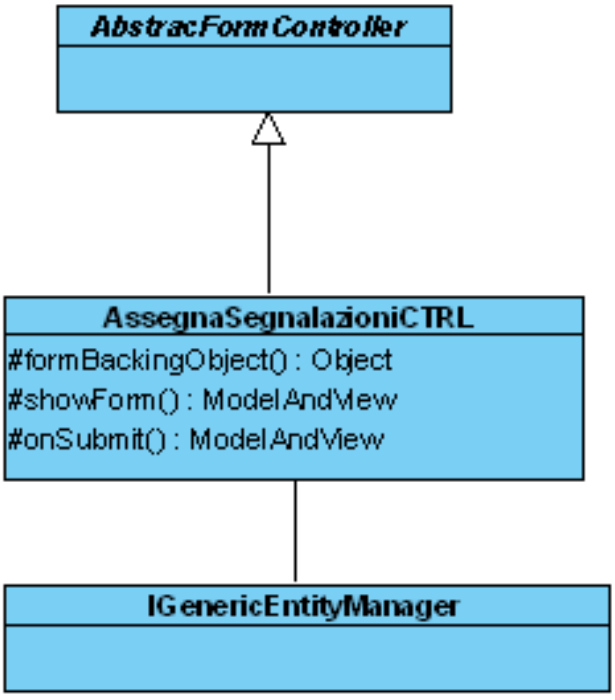

Figura 9 Class diagram assegnazione segnalazione

3.1.5.3.3. Descrizione proprietà e metodi

3.1.5.3.4. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **showForm(HttpServletRequest request, HttpServletResponse response, BindException errors, Map controlModel) throws Exception:** metodo richiamato subito prima del rendering del form HTML per effettuare altre operazioni non necessarie nel formBackingObject
- **onSubmit(HttpServletRequest request, HttpServletResponse response, Object command, BindException errors) throws Exception:** chiamato al submit del form; in questo metodo viene effettuata tutta la logica di business della funzionalità.

3.1.6. Architettura componente gestione segnalazione

3.1.6.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con opportuni ruoli la gestione delle segnalazioni.

3.1.6.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

Figura 10 Use case gestione segnalazione

Gestione Segnalazioni	Viene visualizzato l'elenco delle segnalazioni assegnate all'utente L'utente seleziona una segnalazione e la gestisce

3.1.6.3. Vista logica

3.1.6.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

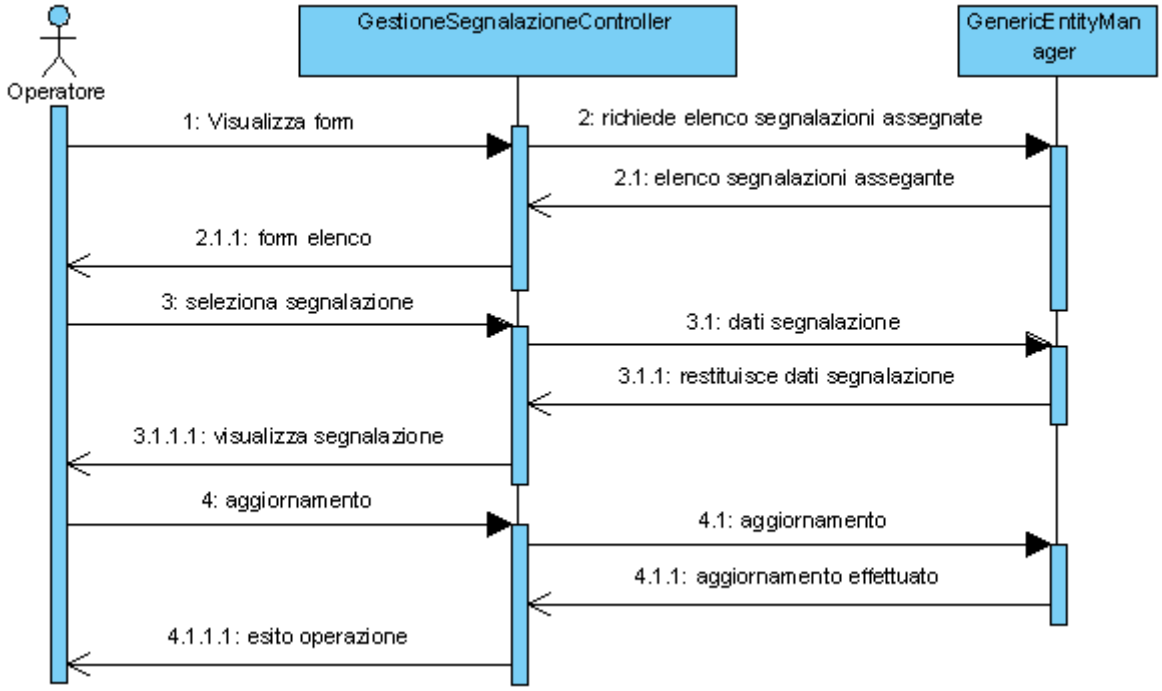

Figura 11 Sequence diagram gestione segnalazione

3.1.6.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

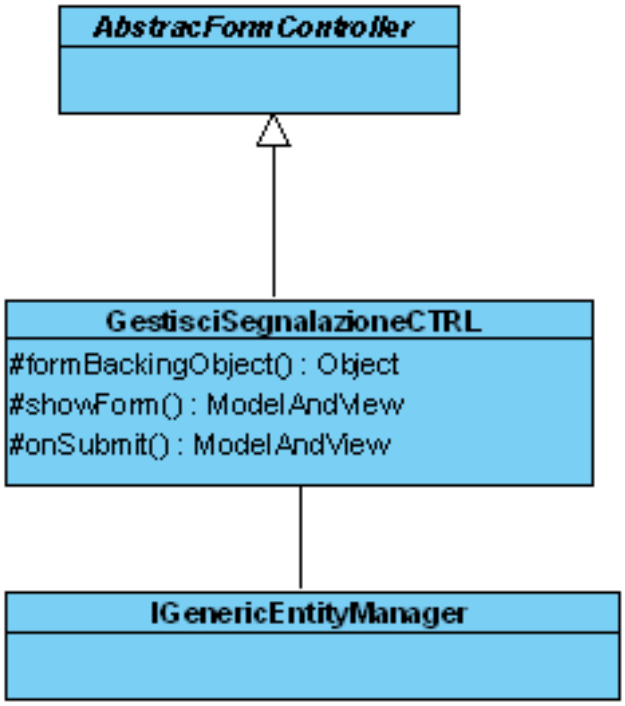

Figura 12 Class diagram gestione segnalazione

3.1.6.3.3. Descrizione proprietà e metodi

3.1.6.3.4. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **showForm(HttpServletRequest request, HttpServletResponse response, BindException errors, Map controlModel) throws Exception:** metodo richiamato subito prima del rendering del form HTML per effettuare altre operazioni non necessarie nel formBackingObject
- **onSubmit(HttpServletRequest request, HttpServletResponse response, Object command, BindException errors) throws Exception:** chiamato al submit del form; in questo metodo viene effettuata tutta la logica di business della funzionalità.

3.1.7. Architettura componente situazione segnalazioni

3.1.7.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con opportuni ruoli di ricercare le segnalazioni e di visualizzare il dettaglio di una di esse.

3.1.7.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

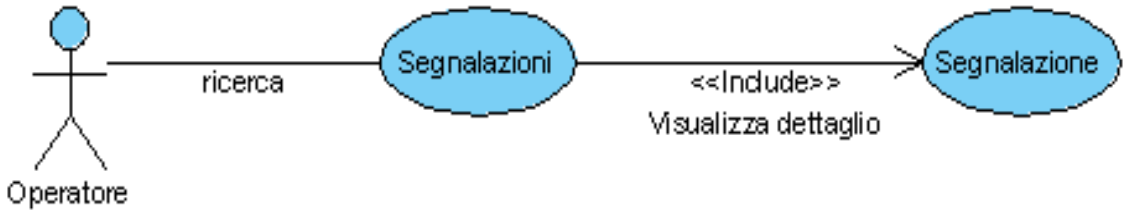

Figura 13 Use case situazione segnalazione

Situazione Segnalazioni	Si ricercano le segnalazioni L'utente seleziona una segnalazione e la visualizza

3.1.7.3. Vista logica

3.1.7.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

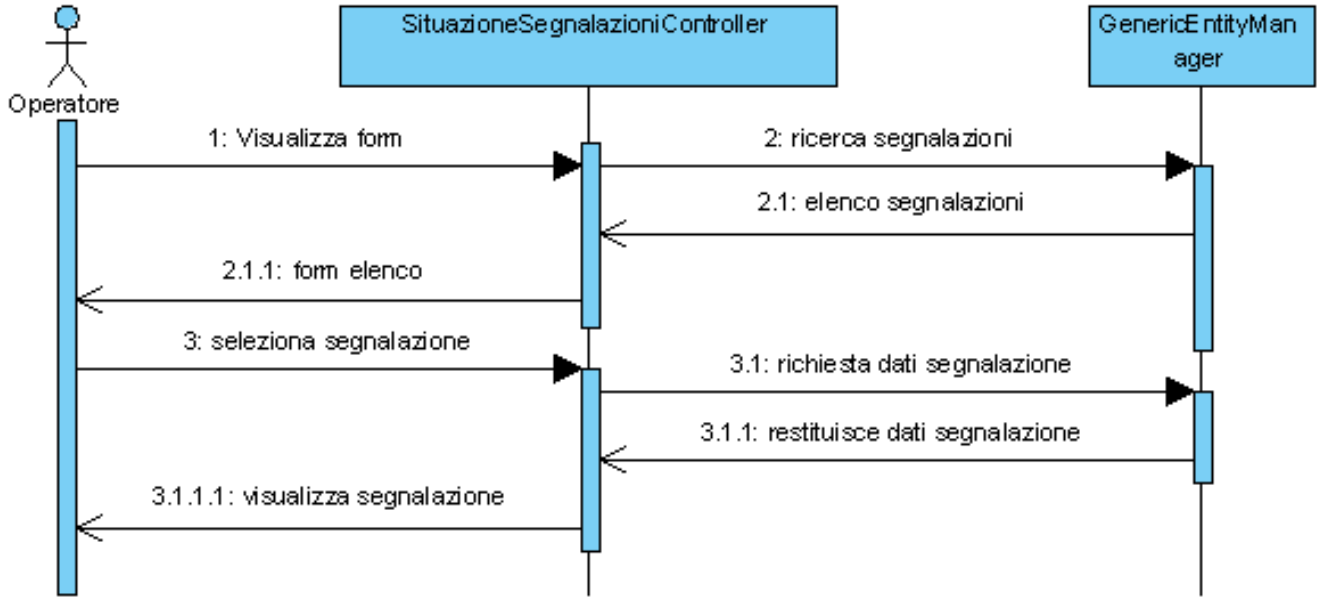

Figura 14 Sequence diagram situazione segnalazioni

3.1.7.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

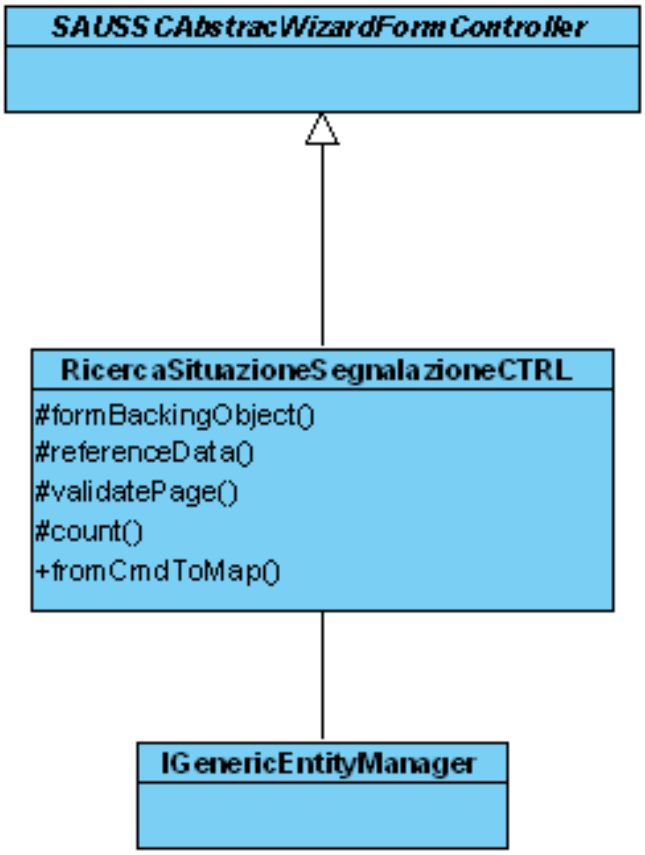

Figura 15 Class diagram situazione segnalazione

3.1.7.3.3. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **referenceData(HttpServletRequest request, Object command, Errors errors, int page):** metodo richiamato per effettuare la logica di business più opportuna durante la navigazione del wizard
- **validatePage(Object command, Errors errors, int page, boolean finish):** metodo chiamato durante il passaggio da uno step all'altro del wizard; si preoccupa di validare il contenuto del form HTML.
- **count(Map criteria):** una select count in base ai criteri passati se criteria è non valorizzato non viene applicato nessun filtro
- **fromCmdToMap:** si veda quando descritto al paragrafo 3.1.2.2.2

3.1.8. Architettura componente ricerca segnalazioni

3.1.8.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con opportuni ruoli di ricercare le segnalazioni e di visualizzare il dettaglio di una di esse.

3.1.8.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

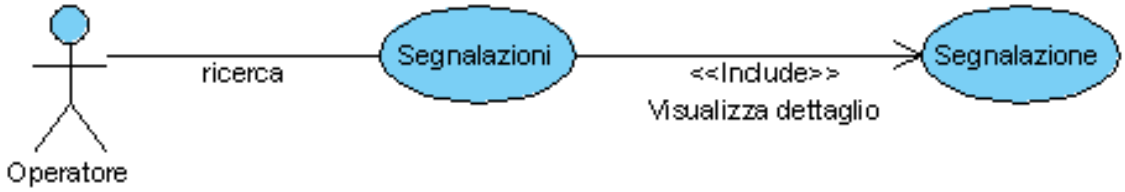

Figura 16 Use case ricerca segnalazione

Ricerca Segnalazioni	Si ricercano le segnalazioni. L'utente seleziona una segnalazione e la visualizza

3.1.8.3. Vista logica

3.1.8.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

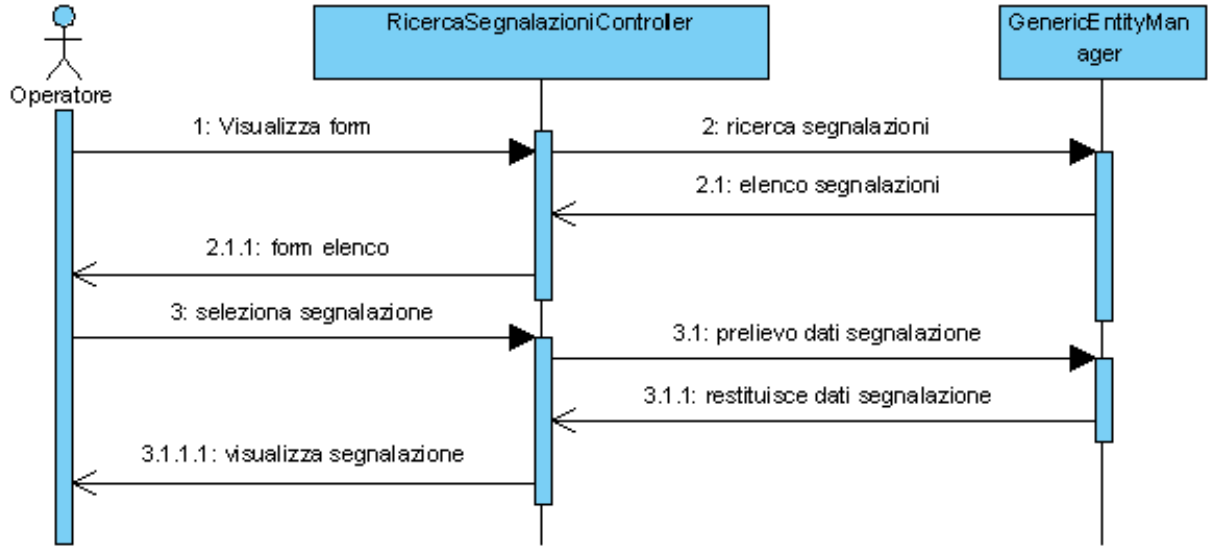

Figura 17 Sequence diagram ricerca segnalazioni

3.1.8.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

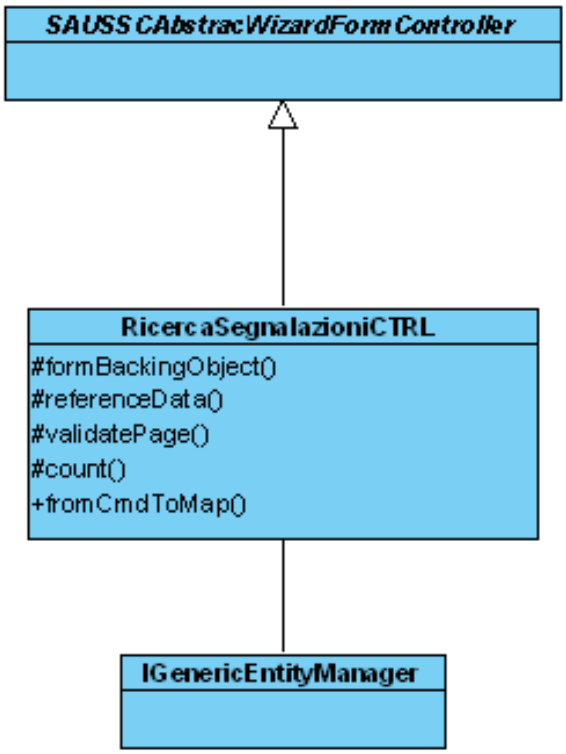

Figura 18 Class diagram ricerca segnalazioni

3.1.8.3.3. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **referenceData(HttpServletRequest request, Object command, Errors errors, int page):** metodo richiamato per effettuare la logica di business più opportuna durante la navigazione del wizard
- **validatePage(Object command, Errors errors, int page, boolean finish):** metodo chiamato durante il passaggio da uno step all'altro del wizard; si preoccupa di validare il contenuto del form HTML.
- **count(Map criteria):** una select count in base ai criteri passati se criteria è non valorizzato non viene applicato nessun filtro
- **fromCmdToMap:** si veda quando descritto al paragrafo 3.1.2.2.2

3.1.9. Architettura componente registrazione richiesta

3.1.9.1. Obiettivi e vincoli funzionali

L'obiettivo della componente è di permettere ad un utente, con opportuni ruoli, la registrazione di una richiesta. L'utente è obbligato a valorizzare tutti i campi obbligatori della richiesta.

3.1.9.2. Vista dei casi d'uso

Di seguito viene riportato lo Use Case della componente:

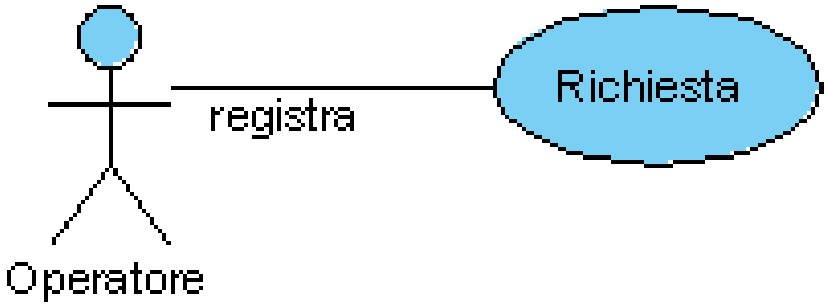

Figura 19 UDC: Registrazione richiesta

Registrazione richiesta	Viene visualizzato il form con i dati da valorizzare. Viene creata la richiesta e viene presentato all'utente l'esito dell'operazione

3.1.9.3. Vista Logica

3.1.9.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

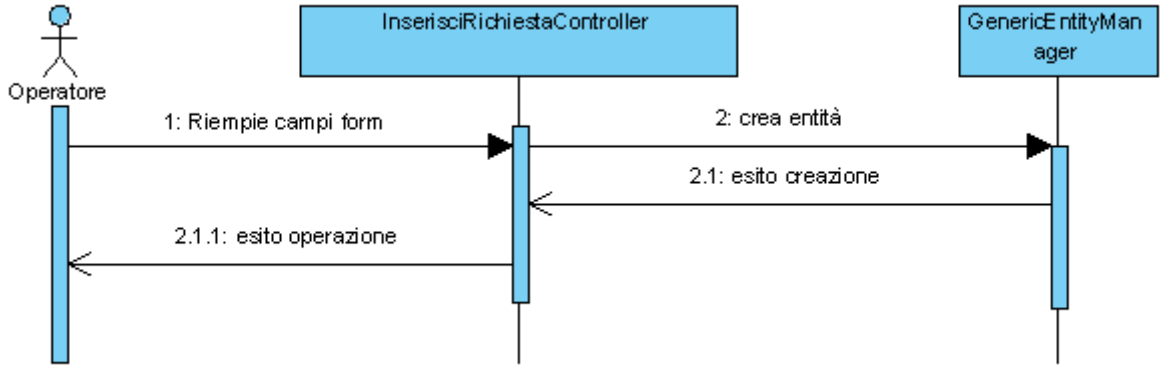

Figura 20 Sequence Diagram Registrazione richiesta

Class diagram

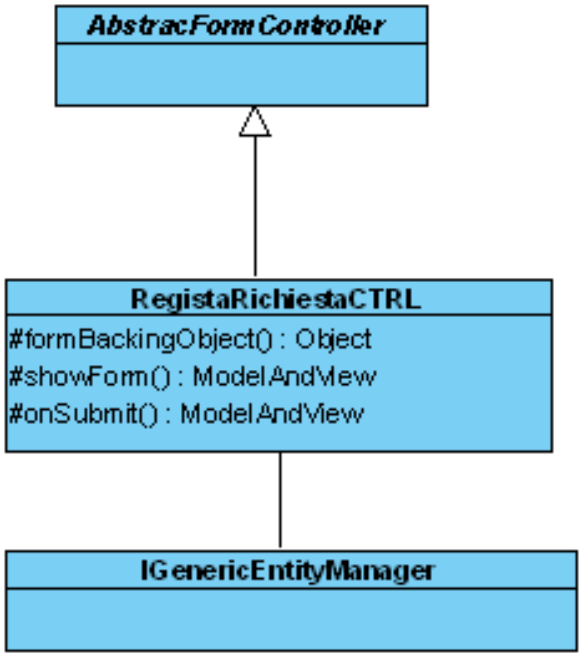

Figura 21 Class Diagram Registrazione richiesta

3.1.9.3.2. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **showForm(HttpServletRequest request, HttpServletResponse response, BindException errors, Map controlModel) throws Exception:** metodo richiamato subito prima del rendering del form HTML per effettuare altre operazioni non necessarie nel formBackingObject
- **onSubmit(HttpServletRequest request, HttpServletResponse response, Object command, BindException errors) throws Exception:** chiamato al submit del form; in questo metodo viene effettuata tutta la logica di business della funzionalità.

3.1.10. Architettura componente assegnazione richiesta

3.1.10.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è i permettere all'utente con opportuni ruoli l'assegnazione delle richieste agli utenti selezionati. Il vincolo è che ci siano effettivamente richieste da assegnare.

3.1.10.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

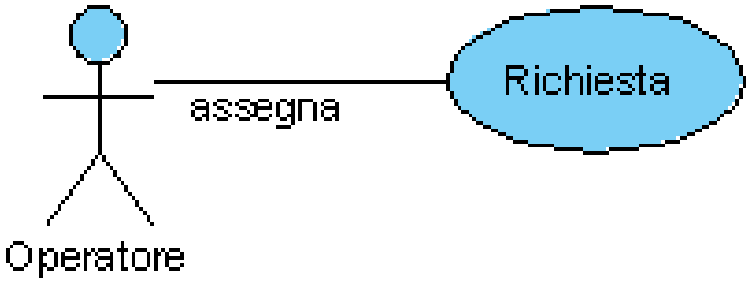

Figura 22 Use case assegnazione richiesta

Assegnazione Richieste	Viene visualizzato il form il numero di segnalazioni aperte e non assegnate e l'elenco degli operatori a assegnare la segnalazione. Scelti gli operatori vengono assegnate le richieste e viene presentato il form con il risultato all'utente. Le richieste vengono distribuite uniformemente tra gli utenti selezionati.

3.1.10.3. Vista logica

3.1.10.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

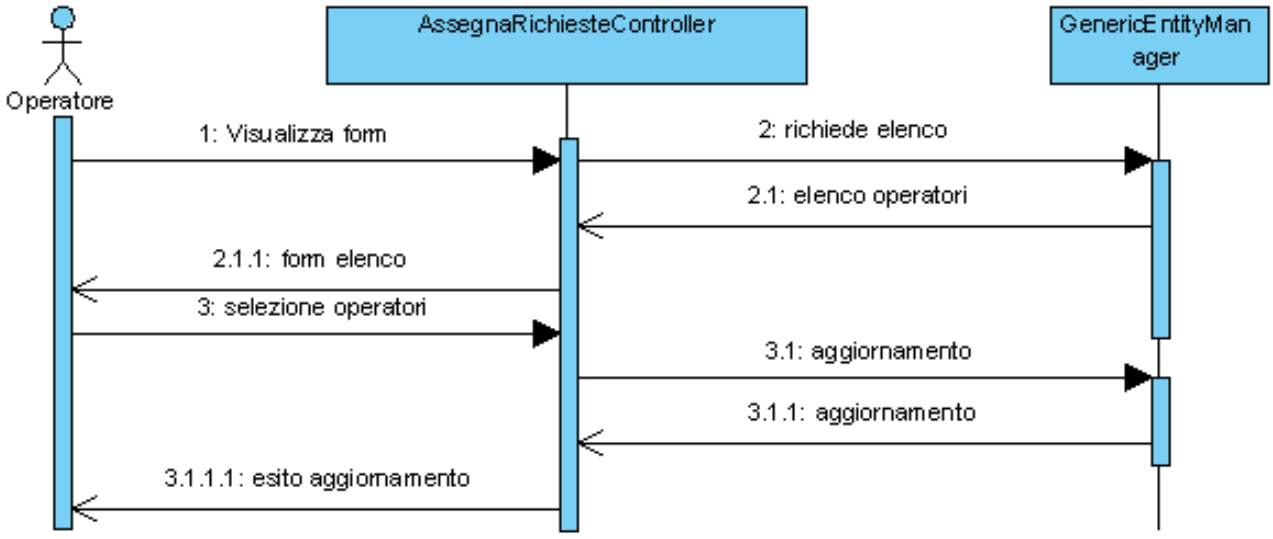

Figura 23 Sequence diagram assegnazione richieste

3.1.10.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

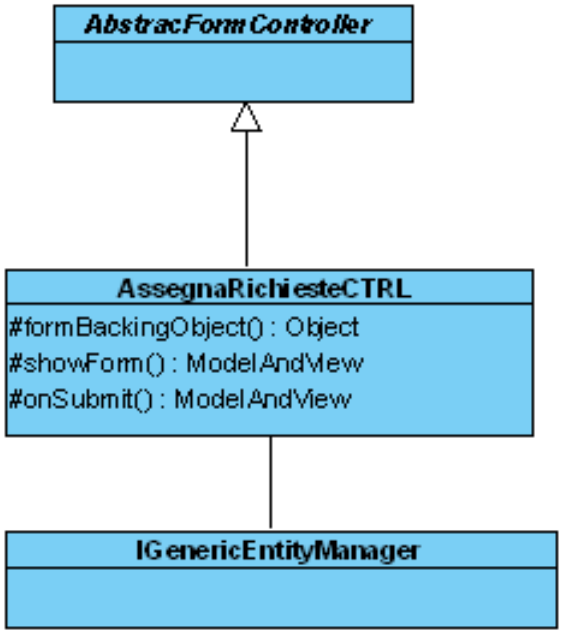

Figura 24 Class diagram assegnazione segnalazione

3.1.10.3.3. Descrizione proprietà e metodi

3.1.10.3.4. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **showForm(HttpServletRequest request, HttpServletResponse response, BindException errors, Map controlModel) throws Exception:** metodo richiamato subito prima del rendering del form HTML per effettuare altre operazioni non necessarie nel formBackingObject
- **onSubmit(HttpServletRequest request, HttpServletResponse response, Object command, BindException errors) throws Exception:** chiamato al submit del form; in questo metodo viene effettuata tutta la logica di business della funzionalità.

3.1.11. Architettura componente gestione richieste

3.1.11.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con opportuni ruoli la gestione delle richieste assegnate.

3.1.11.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

Figura 25 Use case gestione richieste

Gestione Richieste	Viene visualizzato l'elenco delle richieste assegnate all'utente L'utente seleziona una richiesta e la gestisce

3.1.11.3. Vista logica

3.1.11.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

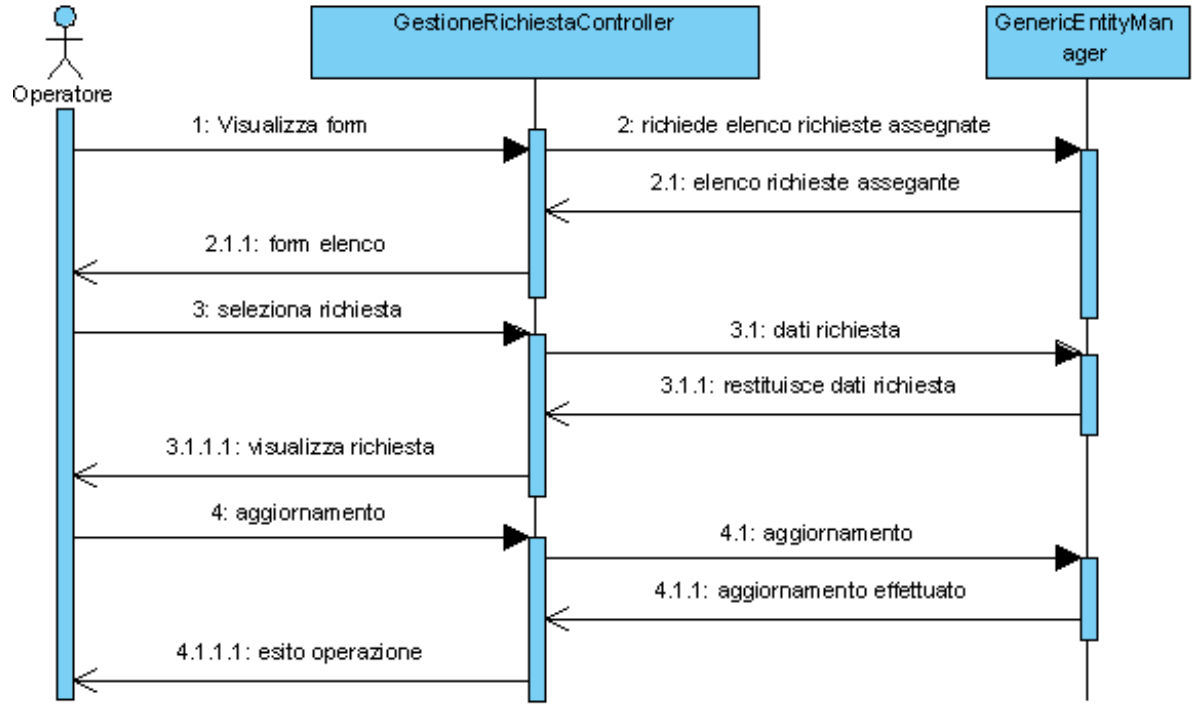

Figura 26 Sequence diagram gestione richieste

3.1.11.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

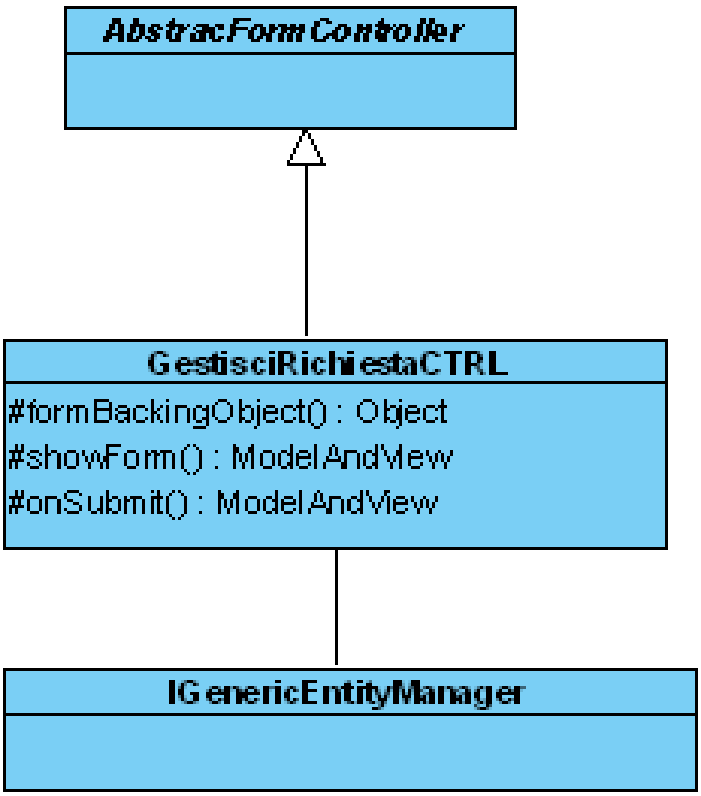

Figura 27 Class diagram gestione segnalazione

3.1.11.3.3. Descrizione proprietà e metodi

3.1.11.3.4. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **showForm(HttpServletRequest request, HttpServletResponse response, BindException errors, Map controlModel) throws Exception:** metodo richiamato subito prima del rendering del form HTML per effettuare altre operazioni non necessarie nel formBackingObject
- **onSubmit(HttpServletRequest request, HttpServletResponse response, Object command, BindException errors) throws Exception:** chiamato al submit del form; in questo metodo viene effettuata tutta la logica di business della funzionalità.

3.1.12. Architettura componente situazione richieste

3.1.12.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con opportuni ruoli di ricercare le richieste e di visualizzare il dettaglio di una di esse.

3.1.12.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

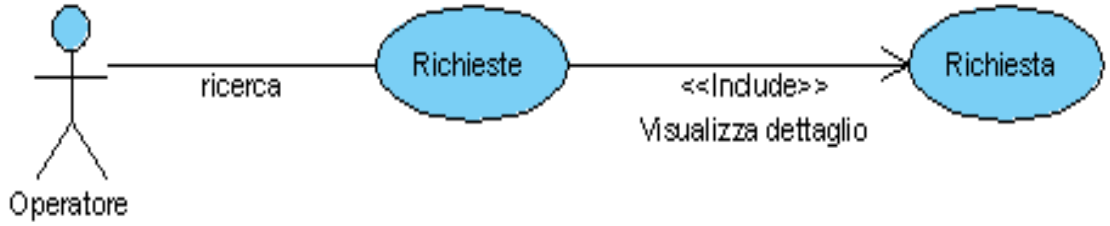

Figura 28 Use case situazione richieste

Situazione richieste	Si ricercano le richieste L'utente seleziona una richiesta e la visualizza

3.1.12.3. Vista logica

3.1.12.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

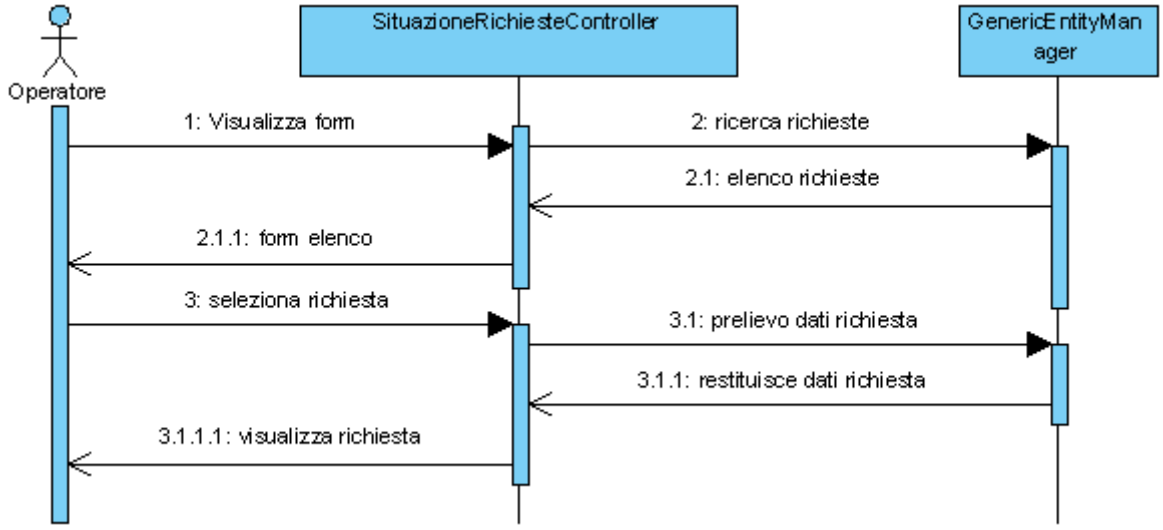

Figura 29 Sequence diagram situazione richieste

3.1.12.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

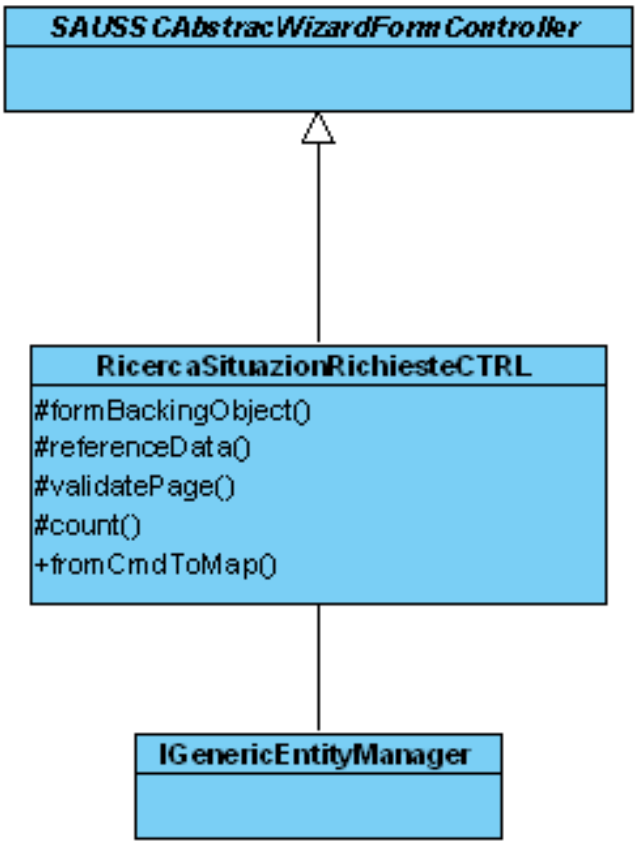

Figura 30 Class diagram situazione richieste

3.1.12.3.3. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **referenceData(HttpServletRequest request, Object command, Errors errors, int page):** metodo richiamato per effettuare la logica di business più opportuna durante la navigazione del wizard
- **validatePage(Object command, Errors errors, int page, boolean finish):** metodo chiamato durante il passaggio da uno step all'altro del wizard; si preoccupa di validare il contenuto del form HTML.
- **count(Map criteria):** una select count in base ai criteri passati se criteria è non valorizzato non viene applicato nessun filtro
- **fromCmdToMap:** si veda quando descritto al paragrafo 3.1.2.2.2

3.1.13. Architettura componente ricerca richieste

3.1.13.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con opportuni ruoli di ricercare le richieste e di visualizzare il dettaglio di una di esse.

3.1.13.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

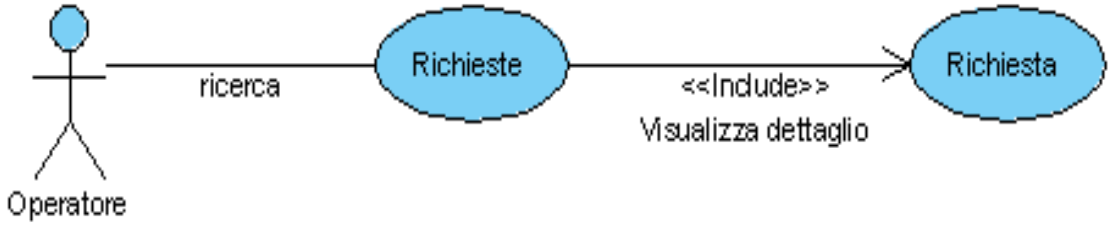

Figura 31 Use case ricerca richieste

Ricerca richieste	Si ricercano le richieste L'utente seleziona una richiesta e la visualizza

3.1.13.3. Vista logica

3.1.13.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

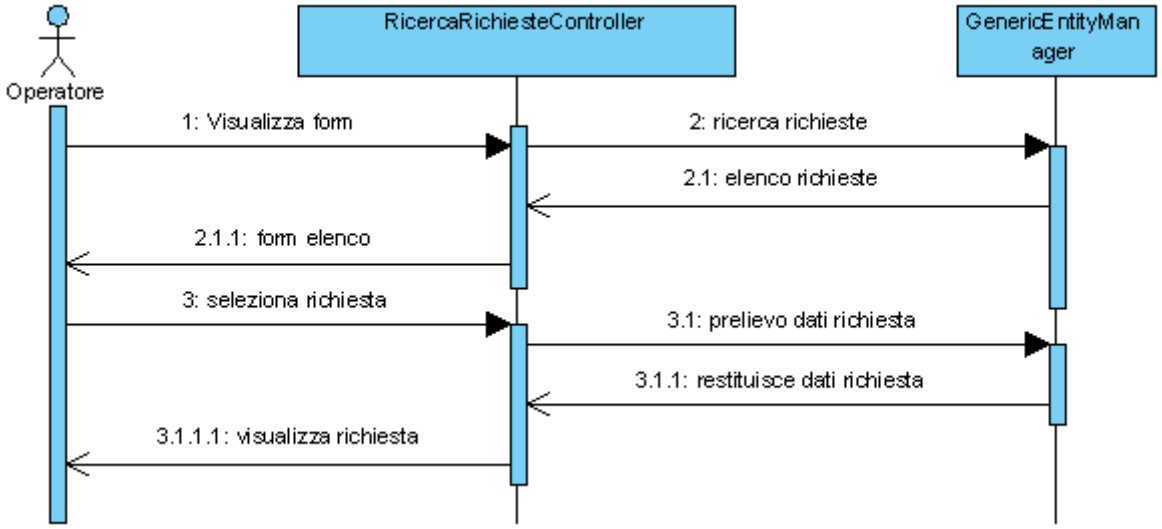

Figura 32 Sequence diagram ricerca richieste

3.1.13.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

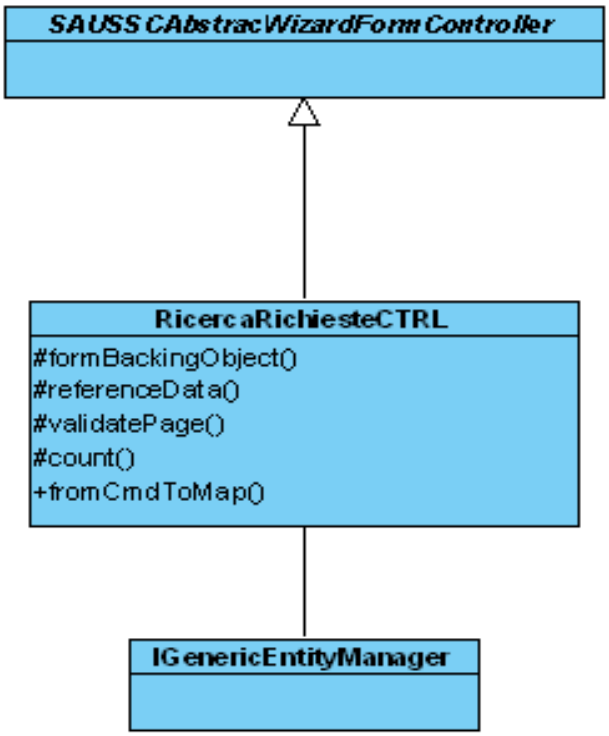

Figura 33 Class diagram ricerca richieste

3.1.13.3.3. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **referenceData(HttpServletRequest request, Object command, Errors errors, int page):** metodo richiamato per effettuare la logica di business più opportuna durante la navigazione del wizard
- **validatePage(Object command, Errors errors, int page, boolean finish):** metodo chiamato durante il passaggio da uno step all'altro del wizard; si preoccupa di validare il contenuto del form HTML.
- **count(Map criteria):** una select count in base ai criteri passati se criteria è non valorizzato non viene applicato nessun filtro
- **fromCmdToMap:** si veda quando descritto al paragrafo 3.1.2.2.2

3.1.14. Architettura componente richieste esperto

3.1.14.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con ruolo "esperto" la gestione delle richieste assegnate.

3.1.14.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

Figura 34 Use case richieste esperto

Richieste esperto	Viene visualizzato l'elenco delle richieste assegnate all'esperto L'esperto seleziona una richiesta e la gestisce

3.1.14.3. Vista logica

3.1.14.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

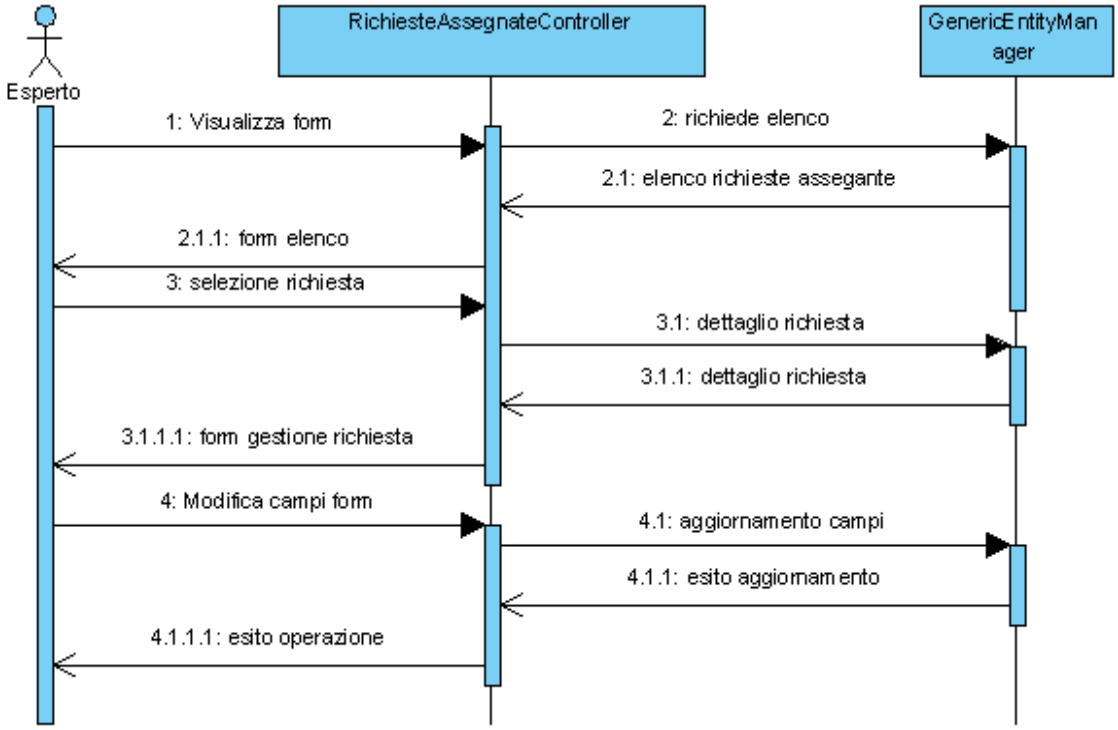

Figura 35 Sequence diagram richieste esperto

3.1.14.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

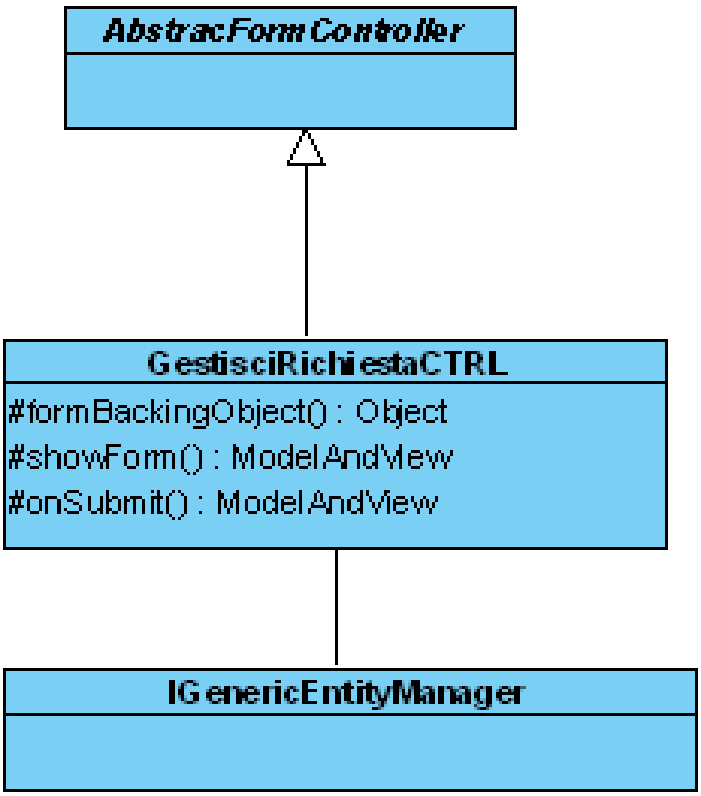

Figura 36 Class diagram gestione segnalazione

3.1.14.3.3. Descrizione proprietà e metodi

3.1.14.3.4. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **showForm(HttpServletRequest request, HttpServletResponse response, BindException errors, Map controlModel) throws Exception:** metodo richiamato subito prima del rendering del form HTML per effettuare altre operazioni non necessarie nel formBackingObject
- **onSubmit(HttpServletRequest request, HttpServletResponse response, Object command, BindException errors) throws Exception:** chiamato al submit del form; in questo metodo viene effettuata tutta la logica di business della funzionalità.

3.1.15. Architettura componente visualizzazione report

3.1.15.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con ruolo "supervisore" la visualizzazione dei vari report.

3.1.15.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

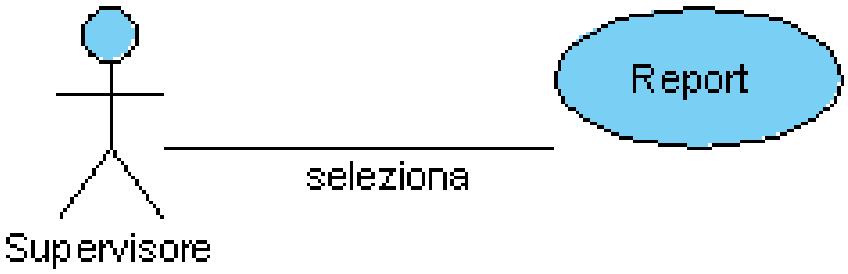

Figura 37 Use case visualizzazione report

Visualizzazione report	Viene visualizzato l'elenco dei report erogabili. Il supervisore ne seleziona uno e lo visualizza

3.1.15.3. Vista logica

3.1.15.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente:

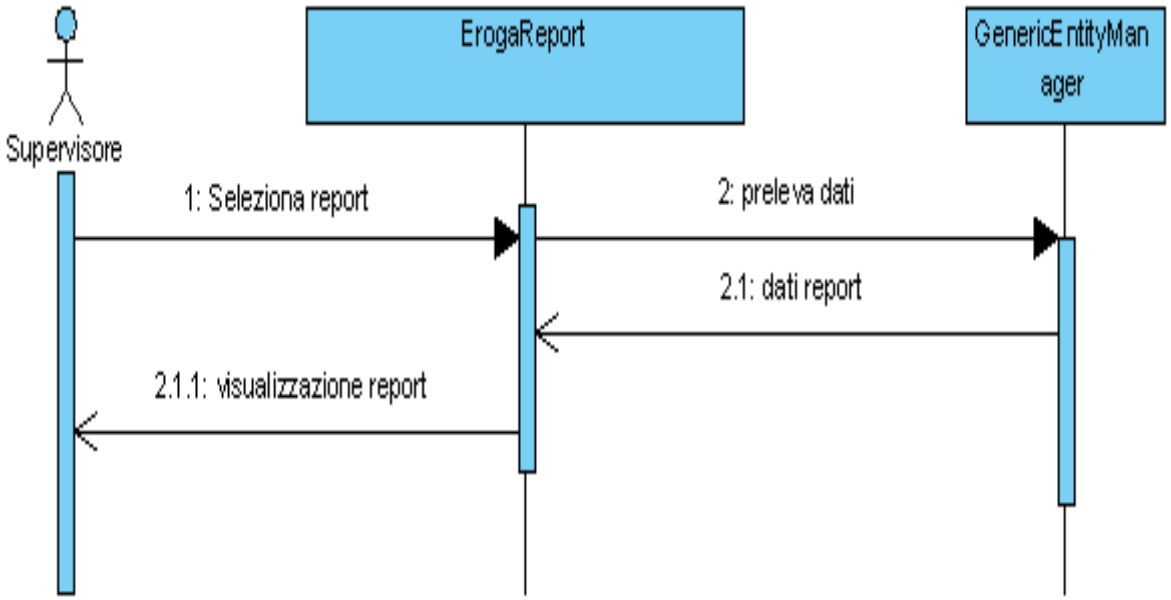

Figura 38 Sequence diagram visualizzazione report

3.1.15.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente:

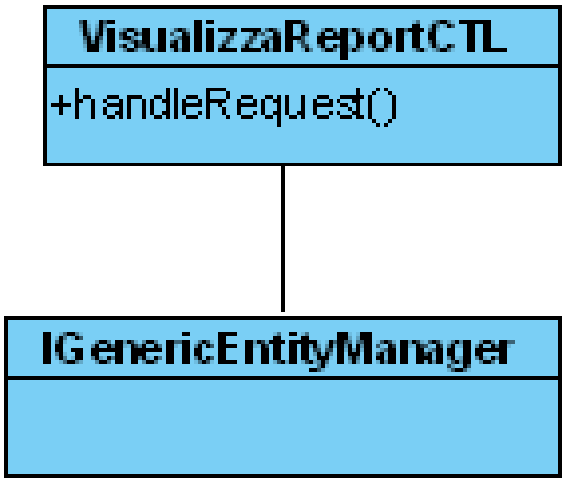

Figura 39 Class diagram visualizzazione report

3.1.15.3.3. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **handleRequest(HttpServletRequest request, HttpServletResponse response):** metodo che si preoccupa di erogare il report selezionato

3.1.15.3.4. Report erogati

Di seguito vengono elencati i report erogati dal sistema. Tutti i report sono stati sviluppati utilizzando il prodotto open source JasperReport.

- **Distribuzione Segnalazioni per categoria:** indica la suddivisione delle segnalazioni in base alla categoria
- **Telefonate ricevute nel terzo quadrimestre:** indica la percentuale di telefonate ricevute nel terzo quadrimestre
- **Telefonate mensili ricevute:** indica la distribuzione di telefonate mensili ricevute
- **Richieste mensili:** indica la distribuzione mensile di richieste
- **Richieste per quadrimestre:** indica la distribuzione delle richieste nel terzo quadrimestre
- **Segnalazioni mensili:** indica la distribuzione mensile di segnalazioni
- **Segnalazioni quadrimestre:** indica la distribuzione delle segnalazioni nel terzo quadrimestre
- **Segnalazioni per fascia di età:** indica la distribuzione delle segnalazioni in base alla fascia di età
- **Richieste per fascia di età:** indica la distribuzione delle richieste in base alla fascia di età
- **Segnalazioni per sesso:** indica la distribuzione delle segnalazioni in base al sesso
- **Richieste per sesso:** indica la distribuzione delle richieste in base al sesso
- **Segnalazioni per titolo di studio:** indica la distribuzione delle segnalazioni in base al titolo di studio
- **Richieste per titolo di studio:** indica la distribuzione delle richieste in base al titolo studio
- **Segnalazioni per professione:** indica la distribuzione delle segnalazioni in base alla professione
- **Richieste per professione:** indica la distribuzione delle richieste in base alla professione
- **Segnalazioni per tempo risposta:** indica la distribuzione delle segnalazioni in base al tempo risposta
- **Richieste per tempo risposta:** indica la distribuzione delle richieste in base alla professione
- **Segnalazioni per tipo segnalazione:** indica la distribuzione delle segnalazioni in base al tipo segnalazione
- **Richieste per tipo richiesta:** indica la distribuzione delle richieste in base al tipo richiesta
- **Segnalazioni per mese:** indica la distribuzione delle segnalazioni al mese
- **Richieste per mese:** indica la distribuzione delle richieste al mese
- **Andamento richieste non evase:** indica l'andamento delle richieste non evase nei vari anni
- **Telefonate per Fascia Oraria:** indica la distribuzione delle telefonate ricevute in base alla fascia oraria

3.1.16. Architettura componente gestione entità

3.1.16.1. Obiettivi e vincoli funzionali

L'obiettivo della funzionalità è di permettere all'utente con ruolo "supervisore" la gestione delle tabelle di sistema dell'applicativo. Sfruttando l'ereditarietà del linguaggio Java tutte le classi persistenti coinvolte estendono una superclasse GenerityEntity; poiché le funzionalità offerte sono le stesse qualsiasi sia l'entità coinvolta tutte le gestioni, per comodità, possono essere rappresentate in un unico capitolo riferendoci alla GenericEntity

3.1.16.2. Vista dei casi d'uso

Di seguito viene riportato il caso d'uso della componente:

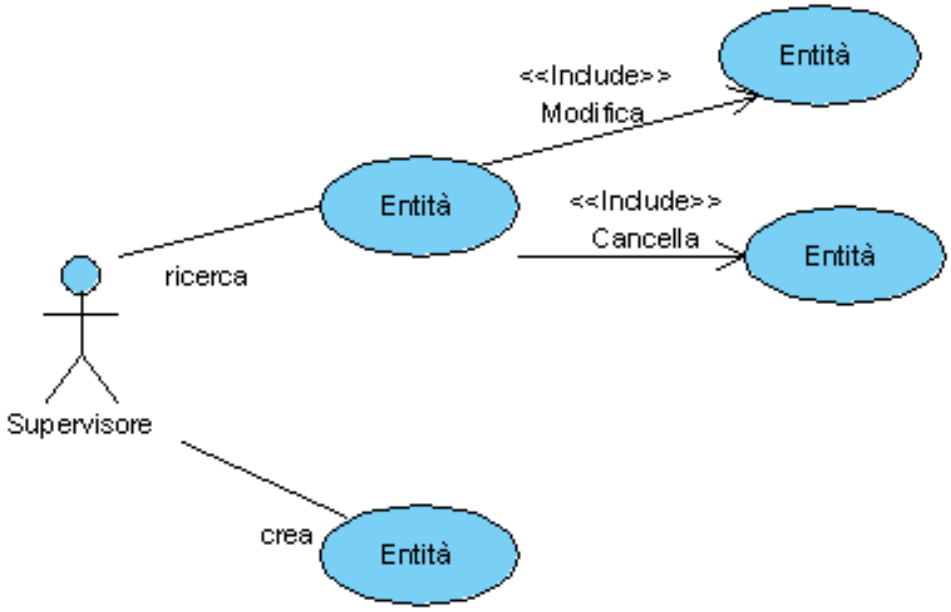

Figura 40 Use case gestione entità

Gestione entità	L'utente può: • Ricercare entità • Aggiungere una nuova entità Se ricerca l'entità il supervisore potrà o modificare l'entità oppure selezionare le entità e cancellarle.

3.1.16.3. Vista logica

3.1.16.3.1. Sequence diagram

Di seguito è riportato il sequence diagram della componente.

Verranno presentati tre sequence diagram a seconda delle scelte dell'utente
Caso cancellazione di una o più entità

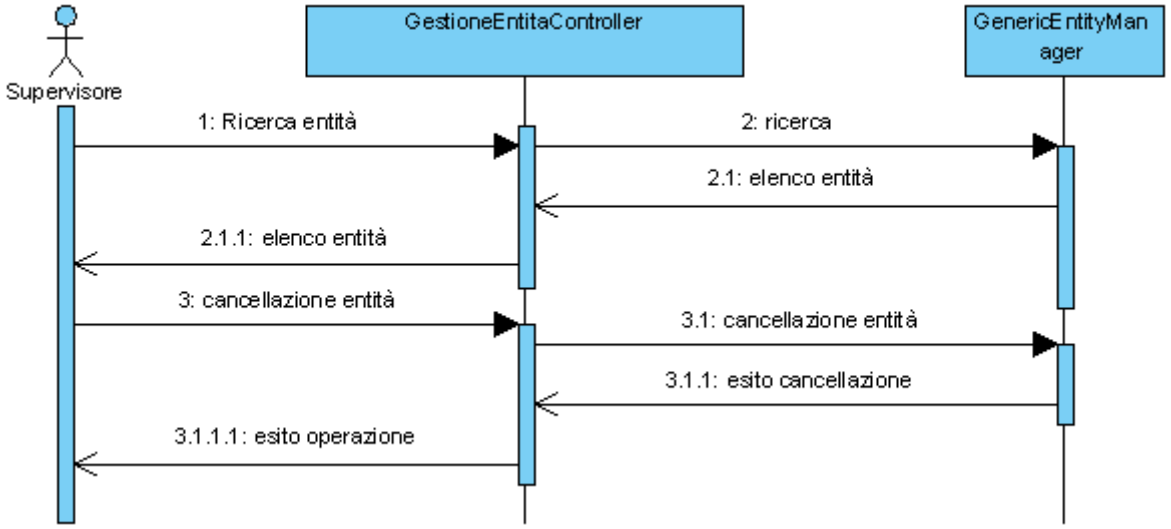

Figura 41 Sequence diagram gestione entità: cancellazione delle entità

Caso modifica dell'entità

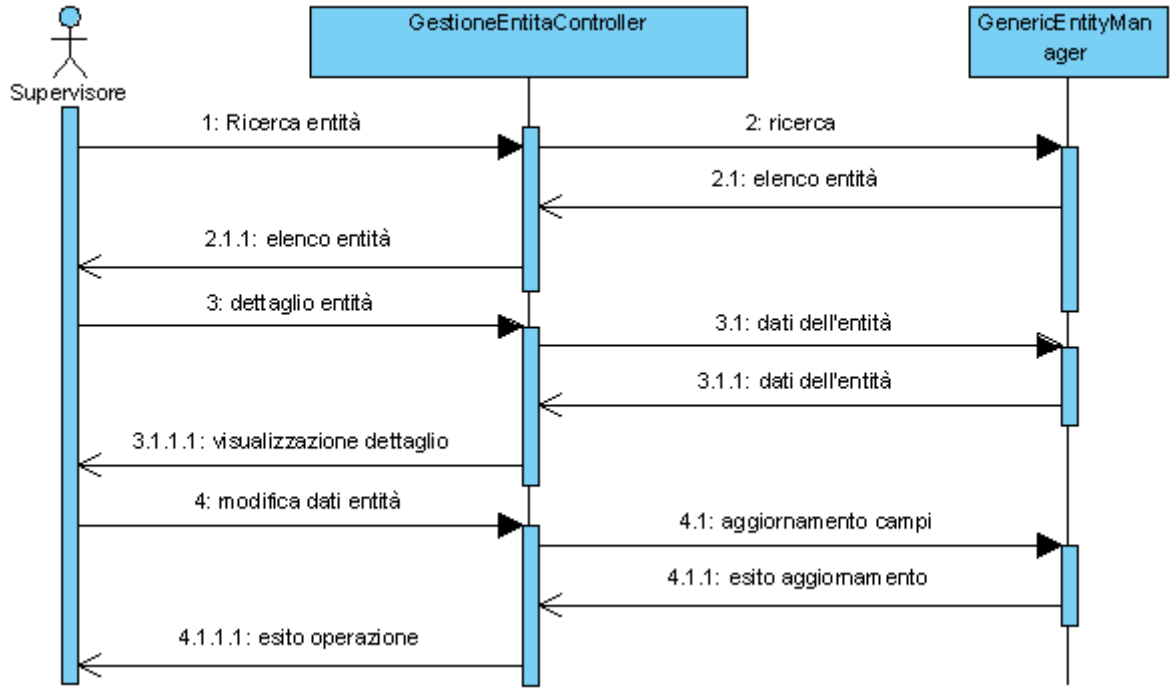

Figura 42 Sequence diagram gestione entità: modifica entità

Caso aggiunta entità

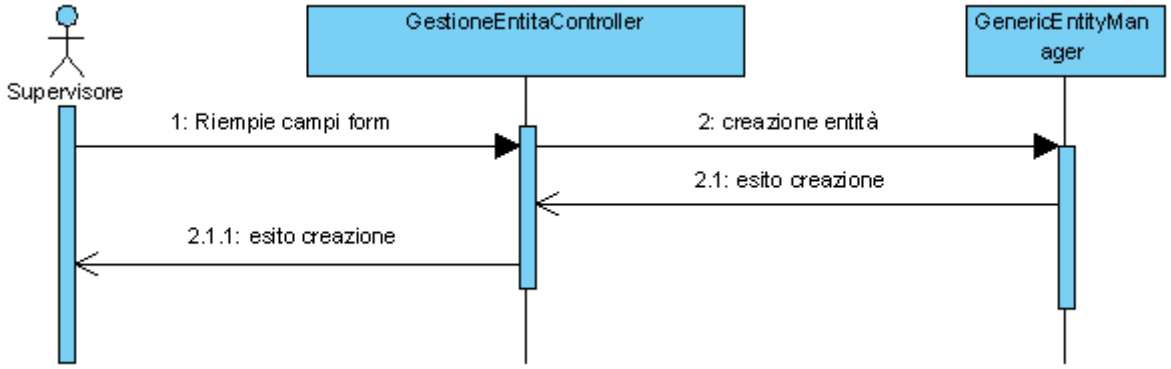

Figura 43 Sequence diagram gestione entità: creazione entità

3.1.16.3.2. Class diagram

Di seguito viene rappresentato il class diagram della componente. Le classi coinvolte sono sempre le stesse. A seconda dell'operazione scelta cambia la logica di business da applicare.

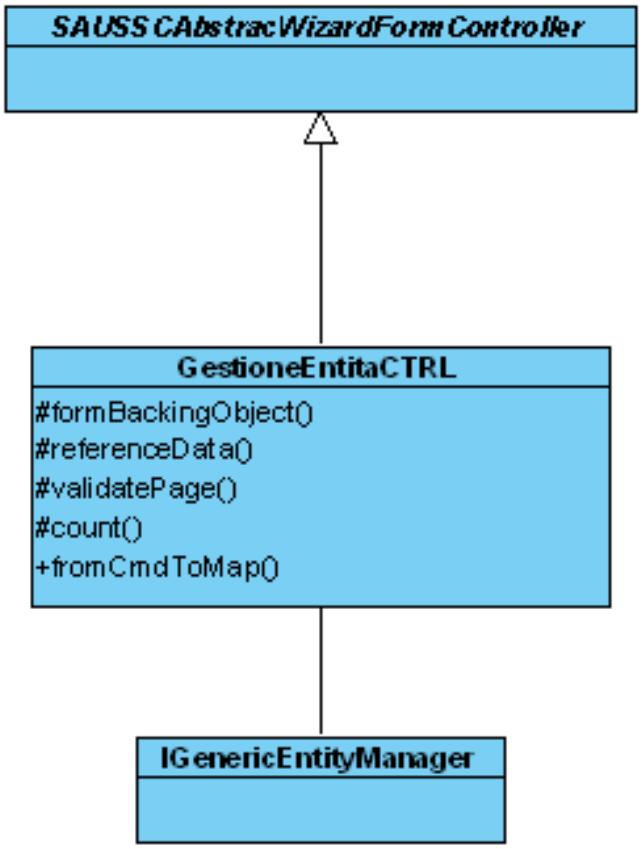

Figura 44 Class diagram gestione entità

3.1.16.3.3. Descrizione proprietà e metodi

Di seguito viene data una descrizione degli oggetti e dei metodi utilizzati all'interno della classe; per un maggior dettaglio ci si può riferire al Javadoc allegato al documento

- **formBackingObject(HttpServletRequest request) :** metodo che si preoccupa di valorizzare correttamente il command associato al form HTML presentato all'utente
- **referenceData(HttpServletRequest request, Object command, Errors errors, int page):** metodo richiamato per effettuare la logica di business più opportuna durante la navigazione del wizard
- **validatePage(Object command, Errors errors, int page, boolean finish):** metodo chiamato durante il passaggio da uno step all'altro del wizard; si preoccupa di validare il contenuto del form HTML.
- **count(Map criteria):** una select count in base ai criteri passati se criteria è non valorizzato non viene applicato nessun filtro
- **fromCmdToMap:** si veda quando descritto al paragrafo 3.1.2.2.2