

UNIVERSITÀ DEGLI  
STUDI DI NAPOLI  
FEDERICO II

Dipartimento di Ingegneria Elettrica  
e delle Tecnologie dell'Informazione  
Corso di Studi in Informatica

---

# Documentazione

## CineMates20

Insegnamento “Ingegneria del Software”  
Anno Accademico 2020/2021

---

### AUTORI

Francesco Borzacchiello N86002541

Carmine Grimaldi N86002551

ID GRUPPO: INGSW2021\_V\_03

### DOCENTE

Prof. Sergio Di Martino

# Indice

<b>PRESENTAZIONE PROGETTO.....</b>	<b>7</b>
<b>DESCRIZIONE .....</b>	<b>7</b>
<b>SORGENTI.....</b>	<b>8</b>
<b>DOCUMENTO DEI REQUISITI SOFTWARE .....</b>	<b>9</b>
<b>USE CASES.....</b>	<b>9</b>
<b>Mobile.....</b>	<b>9</b>
<b>Desktop.....</b>	<b>10</b>
<b>MOCKUPS.....</b>	<b>11</b>
<b>Mobile.....</b>	<b>11</b>
<b>Desktop.....</b>	<b>25</b>
<b>TABELLE DI COCKBURN.....</b>	<b>30</b>
<b>Mobile.....</b>	<b>30</b>
Aggiunge film ad una lista .....	30
Rimuove film da una lista .....	31
Effettua login .....	32
Effettua registrazione .....	33
Cerca film .....	35
Invia richiesta di amicizia .....	36
Rimuove amico .....	37
Visualizza film in comune .....	38
Segnala film .....	39
Segnala utente .....	40
<b>Desktop.....</b>	<b>41</b>
Effettua login .....	41
Visualizza segnalazioni gestite .....	42
Risolve segnalazione pendente .....	44
<b>MODELLI DI DOMINIO.....</b>	<b>46</b>
<b>Class Diagrams .....</b>	<b>46</b>
<b>Mobile .....</b>	<b>46</b>
Utente loggato aggiunge film ad una lista.....	46
Utente loggato rimuove film da una lista .....	47
Utente effettua login .....	48
Utente effettua registrazione .....	49
Utente loggato cerca film.....	50
Utente loggato invia richiesta di amicizia .....	51
Utente loggato rimuove amico .....	52
Utente loggato visualizza film in comune .....	53
Utente loggato segnala film.....	54
Utente loggato segnala utente .....	55
<b>Desktop .....</b>	<b>56</b>
Amministratore effettua login .....	56
Amministratore loggato visualizza segnalazioni gestite .....	57
Amministratore loggato risolve segnalazione pendente .....	58
<b>Sequence Diagrams .....</b>	<b>59</b>
<b>Mobile .....</b>	<b>59</b>
Utente loggato aggiunge film ad una lista .....	59
Utente loggato rimuove film da una lista .....	59
Utente effettua login .....	60
Utente effettua registrazione .....	62
Utente loggato cerca film.....	63
Utente loggato invia richiesta di amicizia .....	64
Utente loggato rimuove amico .....	65
Utente loggato visualizza film in comune .....	66
Utente loggato segnala film.....	67

Utente loggato segnala utente.....	68
<b>Desktop</b>	69
Amministratore effettua login .....	69
Amministratore loggato visualizza segnalazioni gestite .....	70
Amministratore loggato risolve segnalazione pendente .....	71
<b>Statechart Diagram</b> .....	74
Utente rimuove amico .....	74
<b>Activity Diagram</b> .....	75
Amministratore risolve segnalazione pendente .....	75
<b>Diagramma di Gantt</b> .....	76
<b>Organizzazione del team</b> .....	76
<b>DOCUMENTO DI DESIGN</b> .....	77
<b>ANALISI DELL'ARCHITETTURA</b> .....	77
<b>Architettura Android</b> .....	78
Panoramica dei servizi esterni Android .....	79
<b>Architettura Desktop</b> .....	80
Panoramica dei servizi esterni Desktop .....	80
<b>Architettura Server</b> .....	81
Panoramica dei servizi esterni Server .....	82
<b>CLASS DIAGRAMS DI DESIGN</b> .....	83
<b>Android</b> .....	83
Adapter .....	83
Fragment .....	83
Activity .....	84
Controller .....	85
Model .....	86
Util .....	86
<b>Desktop</b> .....	87
Controller .....	87
View .....	88
Model .....	88
Utils .....	89
<b>Server</b> .....	90
Controller .....	90
Dao .....	91
Entity .....	92
Enums .....	92
<b>SEQUENCE DIAGRAMS DI DESIGN</b> .....	93
<b>Android</b> .....	93
Utente loggato aggiunge film ad una lista .....	93
Utente loggato rimuove film da una lista .....	94
Utente effettua login .....	95
Utente effettua registrazione .....	96
Utente loggato cerca film .....	97
Utente loggato invia richiesta di amicizia .....	98
Utente loggato rimuove amico .....	99
Utente loggato visualizza film in comune .....	100
Utente loggato segnala film .....	101
Utente loggato segnala utente .....	102
<b>Funzioni Sequence Diagrams Android</b> .....	103
#FunzioneMostraAmico .....	103
#FunzioneMostraUtente .....	103
#FunzioneRecuperaListeFilm .....	104
#FunzioneSegnala .....	105
#FunzioneVisualizzaAmici .....	106
#FunzioneVisualizzaUtenteDesiderato .....	107
<b>Desktop</b> .....	108

Amministratore effettua login	108
Amministratore loggato visualizza segnalazioni gestite	109
Amministratore loggato risolve segnalazione pendente	110
<b>Funzioni Sequence Diagrams Desktop</b>	111
#FunzioneRisolviSegnalazionePendenteFilm	111
#FunzioneRisolviSegnalazionePendenteUtente	112
#FunzioneVisualizzaSegnalazioniPendentiFilm	113
#FunzioneVisualizzaSegnalazioniPendentiUtenti	114
#FunzioneVisualizzaSegnalazioniGestiteUtenti	115
<b>STATECHART DIAGRAM DI DESIGN</b>	116
<b>CRC CARDS</b>	117
<b>Mobile</b>	117
Controller	117
RegistrationController	117
LoginController	117
ReportController	117
SettingsController	117
UserController	118
PersonalProfileController	118
MainController	118
ResetPasswordController	118
HomeController	119
ShowDetailsMovieController	119
NotificationController	119
MoviesListsController	120
SearchMovieController	120
JoinedMoviesController	120
SearchFriendsController	120
FriendsController	121
Activity	121
MoviesListActivity	121
RegistrationActivity	121
HomeActivity	121
JoinedMoviesActivity	122
ShowDetailsMovieActivity	122
ResetPasswordActivity	122
NotificationActivity	122
PersonalProfileActivity	122
HomeActivity	123
FriendsActivity	123
SettingsActivity	123
InformationActivity	123
UserActivity	124
LoginActivity	124
SearchFriendsActivity	124
SearchMovieActivity	124
Fragment	125
EmptySearchFragment	125
NotEmptyFriendsSearchFragment	125
SettingsFragment	125
PendingFriendsNotificationFragment	125
ReportNotificationFragment	125
RegistrationFragment	125
NotEmptyMovieSearchFragment	126
ConfirmRegistrationCodeFragment	126
Adapter	126
HomeStyleMovieAdapter	126
FriendsAdapter	126
PendingFriendsRequestsAdapter	126

ReportNotificationAdapter.....	127
SearchMovieAdapter .....	127
ActorMovieAdapter.....	127
NotificationPagerAdapter .....	127
<b>ViewHolder</b>	<b>127</b>
HomeStyleMovieAdapter MovieHolder .....	127
UserHolder .....	127
FriendNotificationHolder .....	128
ReportNotificationHolder.....	128
SearchMovieAdapter MovieHolder.....	128
ActorHolder.....	128
<b>Model</b>	<b>128</b>
User .....	128
ListaFilmDB.....	128
UserDB .....	129
ReportMovieDB.....	129
ReportUserDB .....	129
ReportHttpRequests.....	129
UserHttpRequests .....	129
S3Manager .....	129
<b>Punto d'ingresso dell'applicativo</b>	<b>130</b>
EntryPoint .....	130
<b>Util</b>	<b>130</b>
Utilities .....	130
<b>Desktop</b>	<b>131</b>
<b>Controller</b>	<b>131</b>
Controller .....	131
ReportController .....	131
HomeController .....	131
ReportUserContainerDetailsController .....	132
ReportMoviesContainerDetailsController .....	132
ReportUserItemController .....	132
LoginController.....	133
ReportMovieItemController .....	133
ManagedReportsMoviesContainerController .....	133
ReportMoviesContainerController .....	133
ManagedReportUsersContainerController .....	134
NavigationMenuController .....	134
ReportUsersContainerController .....	134
<b>Model</b>	<b>134</b>
S3Manager .....	134
LoginModel .....	135
ReportHttpRequests.....	135
ReportMovieDB.....	135
ReportUserDB .....	135
UserDB .....	135
<b>View</b>	<b>136</b>
GridPaneGenerator .....	136
MessageDialog .....	136
<b>Utils</b>	<b>136</b>
FXMLUtils .....	136
Resources .....	136
NameResources .....	137
<b>Punto d'ingresso dell'applicativo</b>	<b>137</b>
App .....	137
<b>Server</b>	<b>138</b>
<b>Controller</b>	<b>138</b>
ServerSpringSegnalazioneController .....	138
ServerSpringAmministratoriController.....	138
ServerSpringUserController .....	138

ServerSpringListaFilmController .....	139
<b>Dao</b> .....	
Dao<E, I> .....	139
ListaFilmDao<E, I> .....	139
SegnalazioneFilmDao<E, I> .....	139
SegnalazioneUtenteDao<E, I> .....	140
AdministratorDao<E, I> .....	140
UserDao<E, I> .....	140
ListaFilmDaoImplementation .....	140
SegnalazioneFilmDaoImplementation .....	140
SegnalazioneUtenteDaoImplementation .....	141
AdministratorDaoImplementation .....	141
UserDaoImplementation .....	141
<b>Entity</b> .....	141
SegnalazioneUtenteEntity .....	141
SegnalazioneFilmEntity .....	141
UtenteEntity .....	142
ListaFilmEntity .....	142
CredenzialiAmministratoriEntity .....	142
<b>Enums</b> .....	142
TipologiaUtente .....	142
TipoSegnalazione .....	142
<b>Punto d'ingresso dell'applicativo</b> .....	142
DatabaseApplication .....	142
<b>DOCUMENTO DI TESTING</b> .....	143
<b>TEST PLAN</b> .....	143
<b>Mobile</b> .....	143
Test_UC_1 .....	143
Test_UC_2 .....	143
Test_UC_3 .....	144
Test_UC_4 .....	144
Test_UC_5 .....	145
Test_UC_6 .....	145
Test_UC_7 .....	146
Test_UC_8 .....	146
Test_UC_9 .....	147
Test_UC_10 .....	147
<b>Desktop</b> .....	148
Test_UC_1 .....	148
Test_UC_2 .....	148
Test_UC_3 .....	149
<b>TESTING DEL METODO GETBYID</b> .....	150
Testing Black-Box .....	151
Testing White-Box .....	153
<b>TESTING DEL METODO ISFRIENDREQUESTPENDING</b> .....	154
Testing Black-Box .....	155
Testing White-Box .....	159
<b>TESTING DEL METODO ISUSERNAMEVALID</b> .....	161
Testing Black-Box .....	162
Testing White-Box .....	166
<b>TESTING DEL METODO ISPASSWORDVALID</b> .....	168
Testing Black-Box .....	169
Testing White-Box .....	173

# Presentazione progetto

## Descrizione

CineMates20 è un social network composto da più moduli. Il suo obiettivo è mettere a disposizione un angolo virtuale dove gli appassionati di film e cinema si possono ritrovare.

Questo sistema è composto da tre moduli:

- L'applicazione per dispositivi mobile, riservata agli utenti, essi sono il target principale a cui è destinato il social network;
- Un applicazione desktop, riservata agli amministratori, che hanno il compito di garantire l'ordine ed il rispetto sulla piattaforma, assicurandosi che gli utenti ed i contenuti<sup>1</sup> della piattaforma rispettino delle regole prestabilite;
- Il terzo modulo è il server, esso viene utilizzato sia dall'applicativo mobile che dall'applicativo desktop, ha il compito di archiviare tutte le informazioni riguardanti gli utenti, gli amministratori ed i contenuti della piattaforma. Questi dati verranno poi forniti ai client quando saranno richiesti.

Di seguito viene riportato un elenco delle funzionalità che questo innovativo social network offre.

Per gli utenti:

- Come ogni social network che si rispetti l'utente ha la possibilità di stringere amicizie, in particolare può inviare richieste di amicizia ad altri utenti e a sua volta può riceverne, queste richieste possono essere accettate e quindi si diventa amici a tutti gli effetti o possono essere respinte, nel momento in cui un utente non desidera più essere amico di un altro utente ha la possibilità di rimuoverlo dalla propria lista di amici;
- Un utente può comodamente consultare la lista dei propri amici, senza dover ricordare quali essi sono, da questa lista può accedere ai profili dei propri amici per vedere le loro informazioni. Nel caso in cui l'utente è interessato a visionare il profilo di altri utenti che non fanno parte della propria cerchia di amici è possibile effettuare una ricerca, che può essere fatta per username, e-mail, nome e cognome;
- Oltre alla parte social il focus di questa piattaforma riguarda i film, la home page accoglie l'utente mostrando alcuni film che potrebbero essere di suo interesse, questi sono suddivisi in diverse categorie, ovvero i film usciti di recente, quelli che vedremo prossimamente nelle sale cinematografiche, i più apprezzati dagli utenti ed infine i film che sono attualmente di tendenza. Ma oltre a quelli proposti dalla home page, l'utente ha la possibilità di poter cercare qualunque film a suo piacimento mediante il titolo del film, dal risultato della ricerca è possibile poi visualizzare maggiori dettagli;

---

<sup>1</sup> Contenuti: in queste due pagine di presentazione del sistema con la parola contenuti attualmente si fa riferimento ai film, ma non si esclude che con futuri aggiornamenti gli utenti avranno la possibilità di pubblicare dei loro contenuti inerenti ai film o aggiungere dei commenti ai film o ad altri contenuti della piattaforma.

- Partendo dalla ricerca, l'utente può salvare alcuni film di maggiore interesse in delle liste personali, in particolare ogni utente ha una lista di film preferiti e di film da vedere in un secondo momento, ma dalla ricerca è anche possibile rimuovere dei film precedentemente inseriti nelle liste, queste liste hanno anche una propria schermata dedicata, da cui possono essere comodamente consultate e gestite, ovvero se nel consultare queste liste si volesse eliminare uno o più film è possibile farlo selezionando i film da voler eliminare, senza dover ricorrere alla ricerca;
- Gli utenti che sono amici, visualizzando il profilo di un amico di loro interesse, tramite un apposito tasto possono vedere se hanno interessi in comune verso gli stessi film, in particolare viene mostrata una lista di film in comune che contiene film presenti in entrambe le liste dei due amici;
- Nel caso in cui un utente non rispettasse regole morali di base o la politica della piattaforma, gli altri utenti possono segnalarlo in modo che gli amministratori possano prendere provvedimenti, lo stesso vale per i contenuti.

Per gli amministratori:

- Gli amministratori hanno il compito di gestire le segnalazioni fatte dagli utenti, decidendo se siano fatte a sproposito e quindi ignorarle o se siano fondate e quindi prendere dei provvedimenti oscurando i contenuti e quindi accogliendo le segnalazioni. Essi possono consultare le segnalazioni ricevute suddivise per utenti e film;
- Visto che gli amministratori potrebbero avere la necessità di consultare le segnalazioni da loro gestite, è presente una sezione dedicata alle segnalazioni precedentemente gestite, anche queste suddivise per utenti e film;
- Al crescere del numero di utenti si potrebbe andare incontro ad un'esplosione del numero di segnalazioni, per agevolare il lavoro degli amministratori è stata messa a loro disposizione una ricerca per poter filtrare le segnalazioni (sia da gestire che già gestite), ed un ordinamento per numero di segnalazioni o per nome (dei film o degli utenti).

Due parole sulla gestione degli accessi al sistema:

Gli utenti hanno la possibilità di registrarsi utilizzando la schermata di registrazione progettata dal team di CineMates20, al termine riceveranno un codice di conferma sulla mail inserita in fase di registrazione per poter confermare che quella mail gli appartenga, oppure possono optare per una registrazione più comoda è veloce utilizzando i loro profili esistenti su altri canali social. Discorso analogo per l'accesso.

Gli amministratori per poter essere inseriti nel sistema devono aspettare che gli vengano fornite le credenziali dai responsabili del progetto, poiché essi ricoprono un ruolo delicato è stato deciso che non possono registrarsi autonomamente, ma sarà la dirigenza di CineMates20 a selezionare gli amministratori e creargli un account per accedere al sistema.

Nel caso in cui gli utenti dovessero dimenticare la password possono reimpostarla in completa autonomia, mentre discorso a parte per gli amministratori che devono rivolgersi ad un responsabile.

## Sorgenti

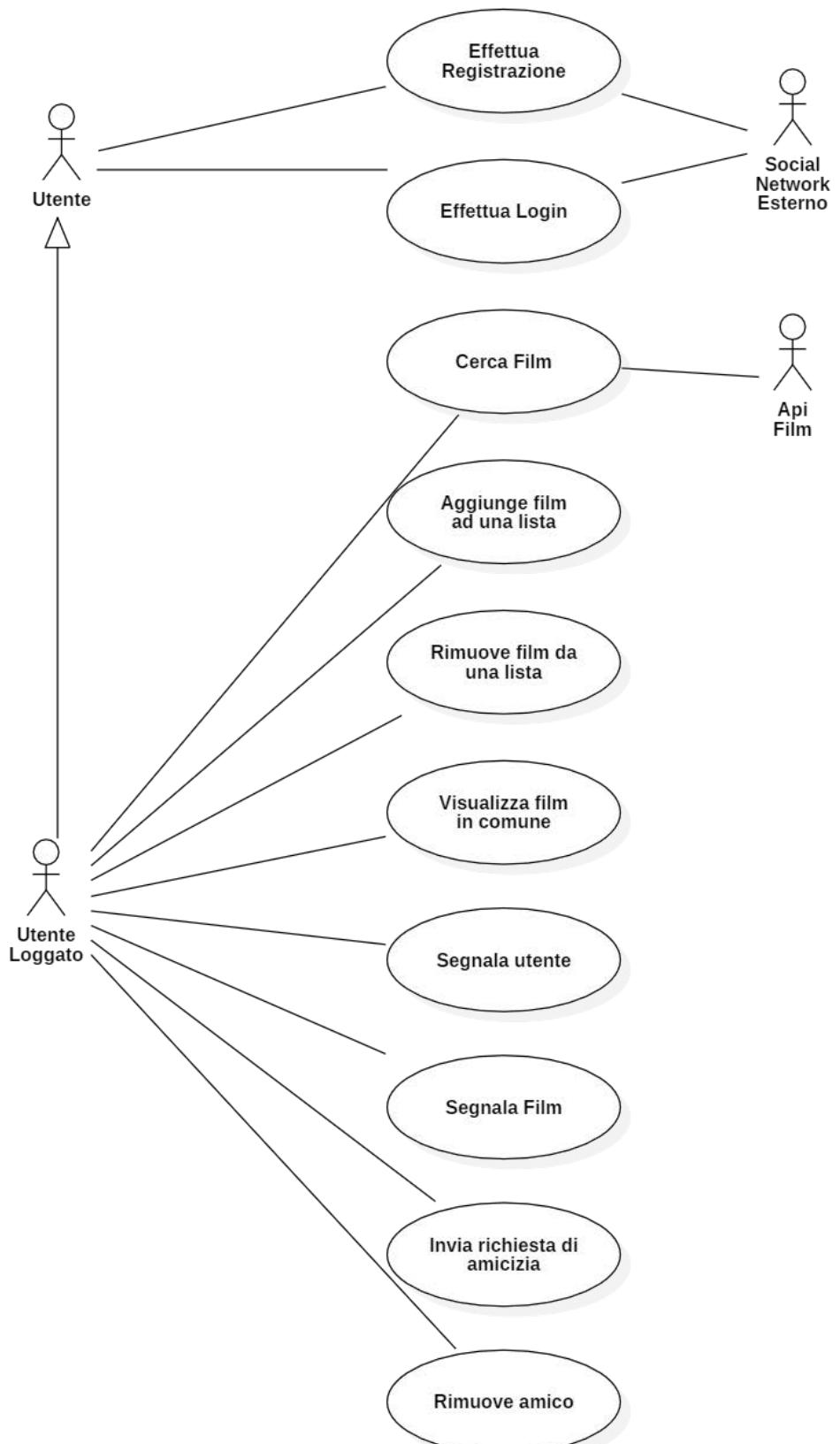
Per la visione dei sorgenti si rimanda ai seguenti repository GitHub:

- [Repository Server](#)
- [Repository Android](#)
- [Repository Desktop](#)

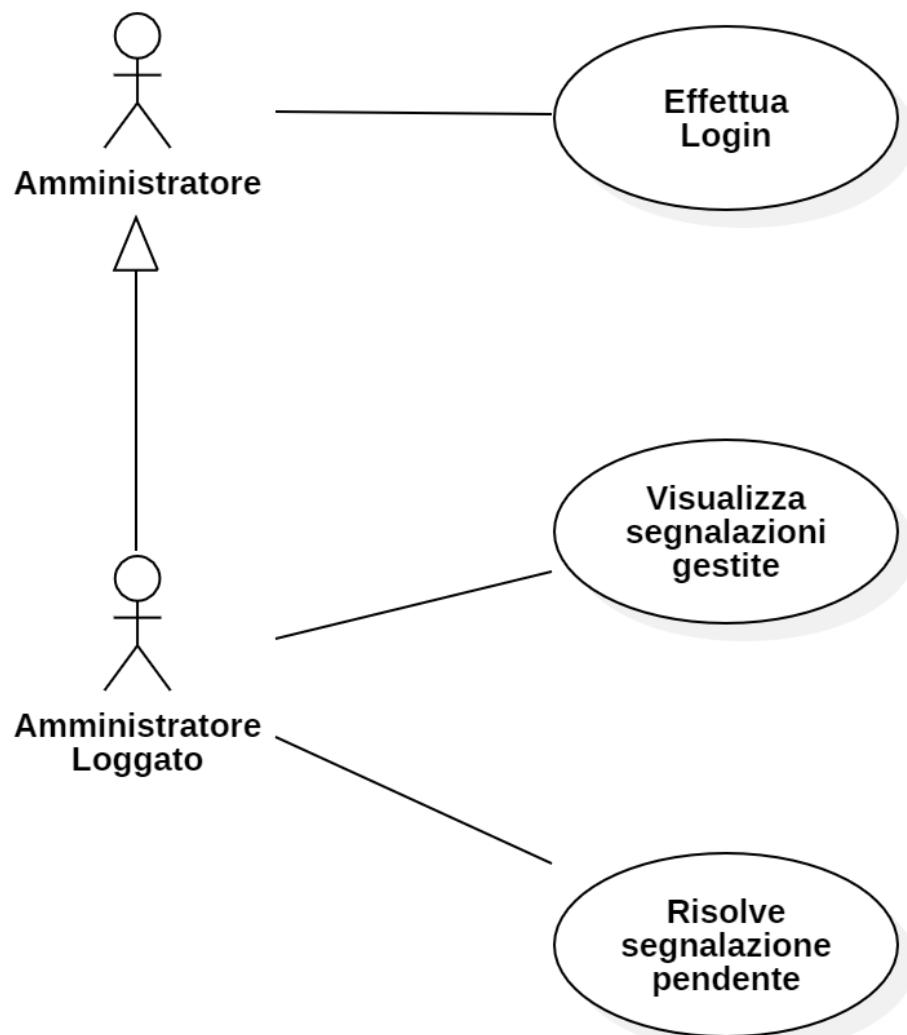
# Documento dei Requisiti Software

## Use cases

### Mobile



## Desktop



## Mockups

### Mobile

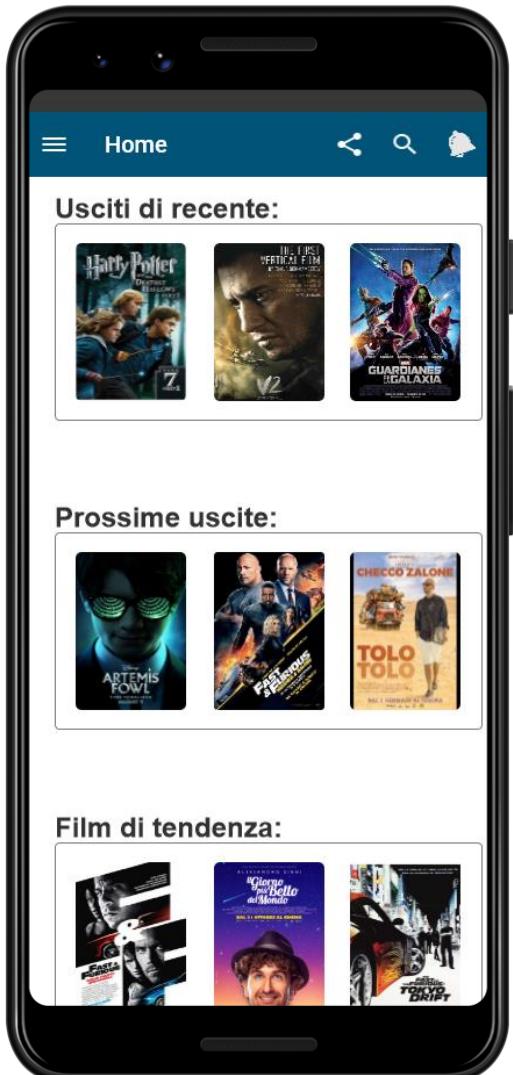


Figura 1 - M0

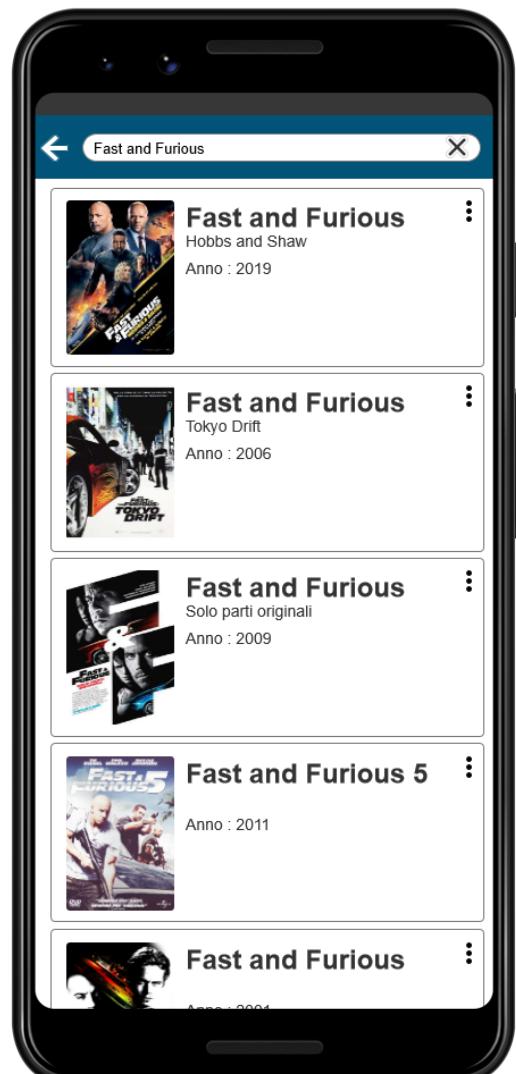


Figura 2 – M1



Le voci aggiungi a lista "Film da vedere" / "Film preferiti", non saranno presenti nel menù se il film appartiene già a tale lista



Figura 4 – M3



Figura 5 – M4

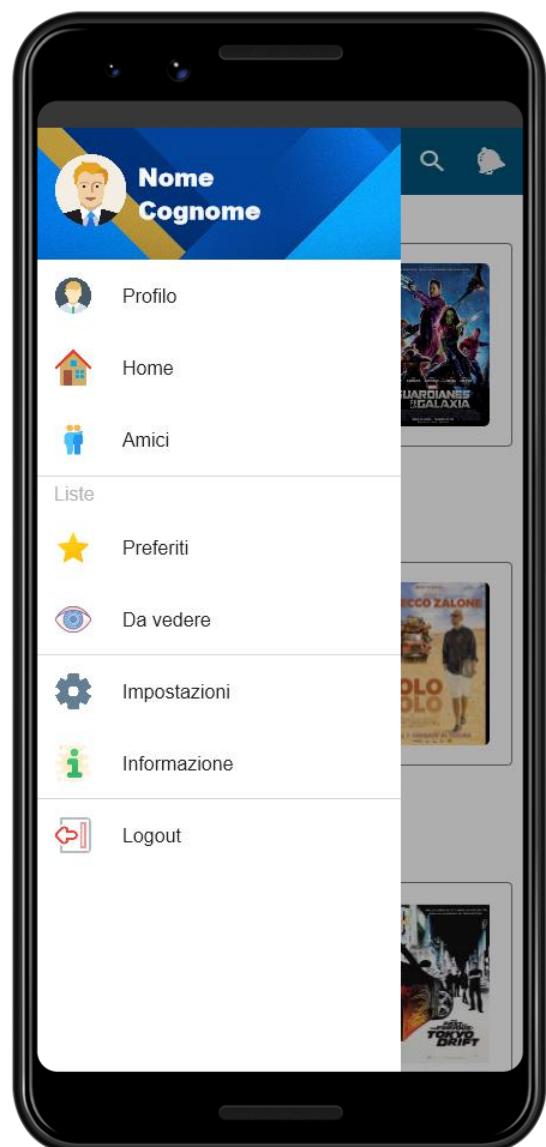


Figura 6 – M5

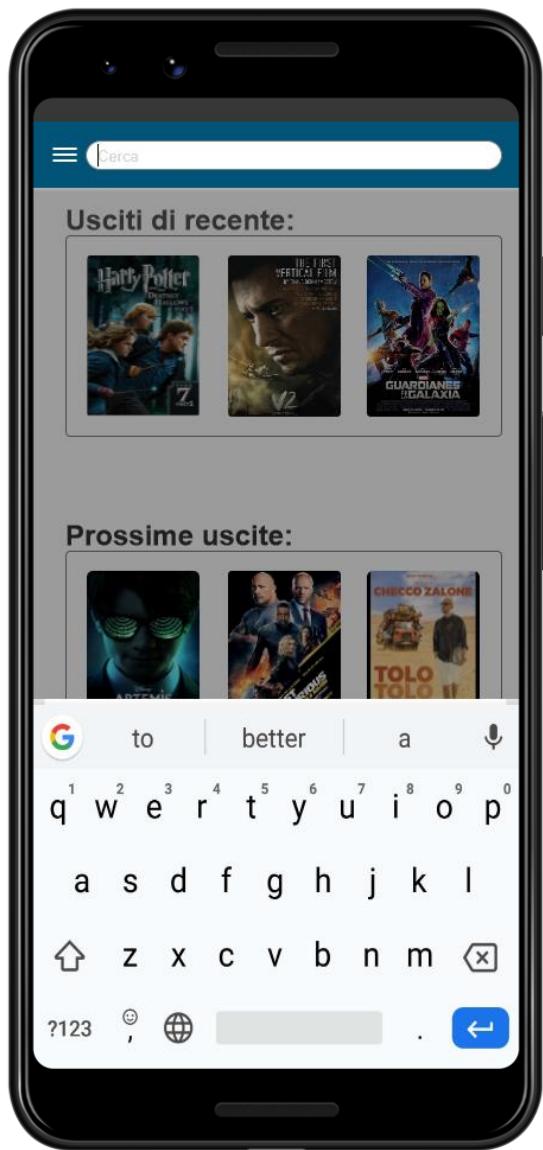


Figura 7 – M6



Figura 8 – M7

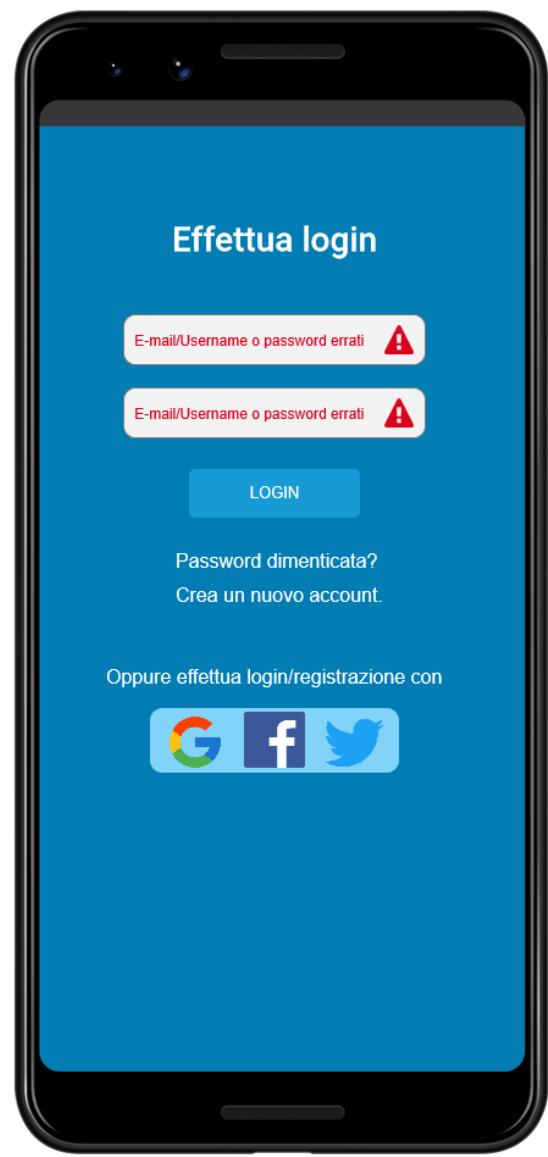
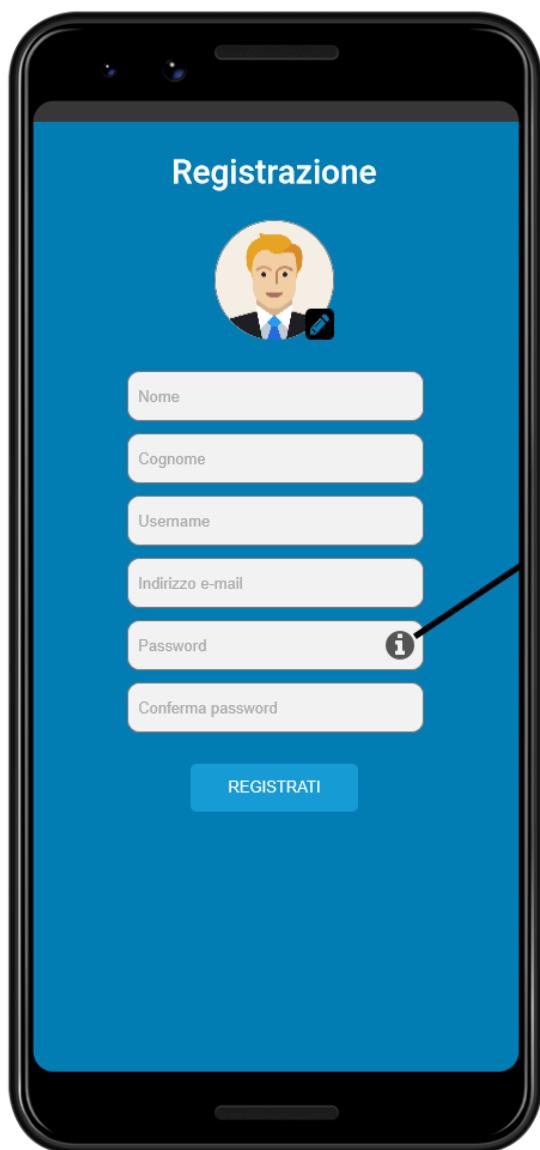


Figura 9 – M8



La password deve contenere almeno 8 caratteri, una lettera maiuscola e un carattere speciale.

Figura 10 – M9

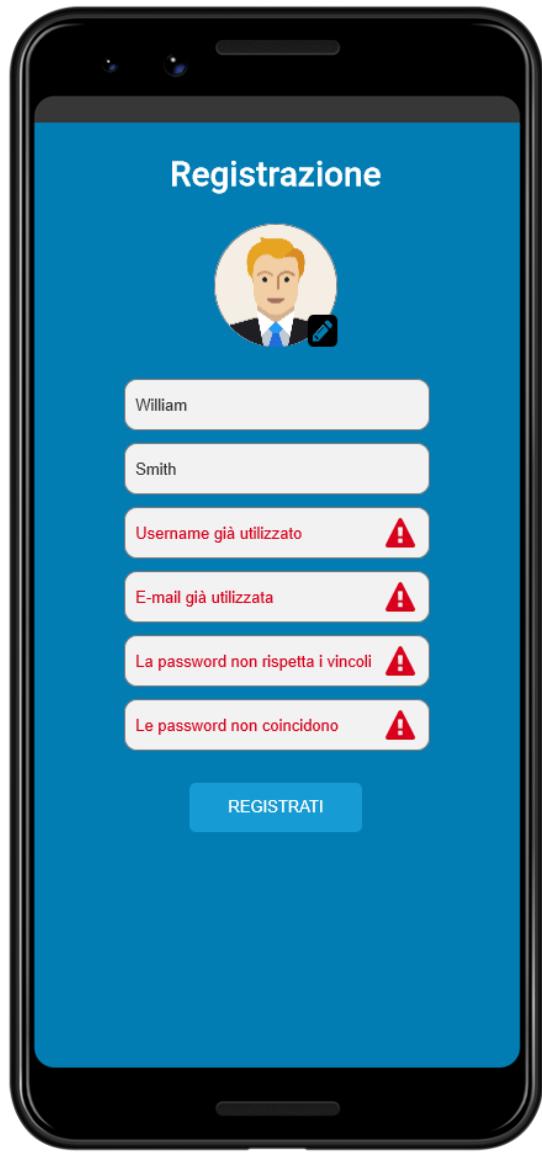


Figura 11 – M10

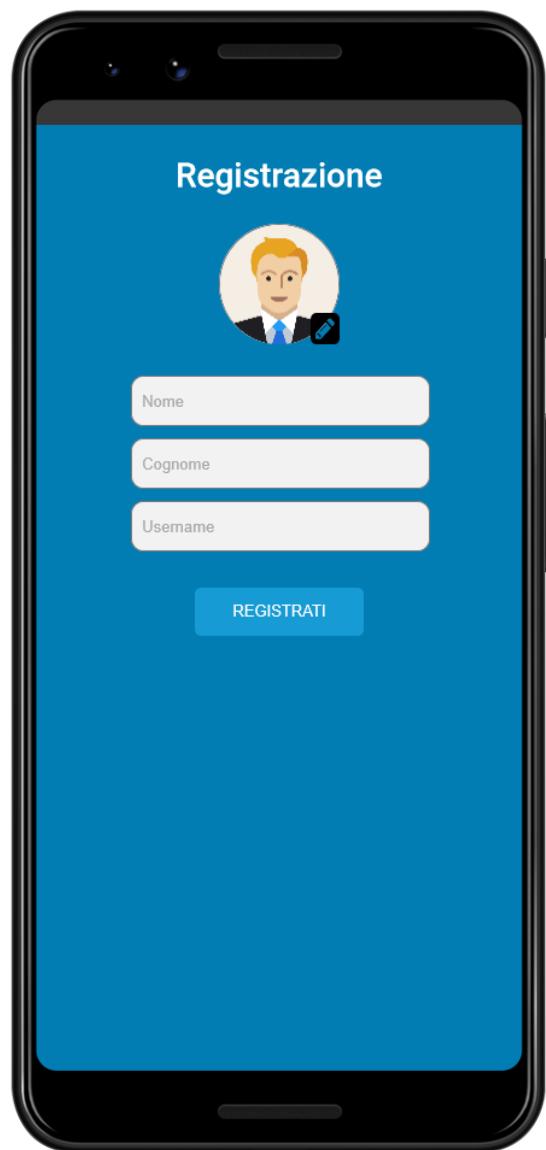


Figura 12 – M11

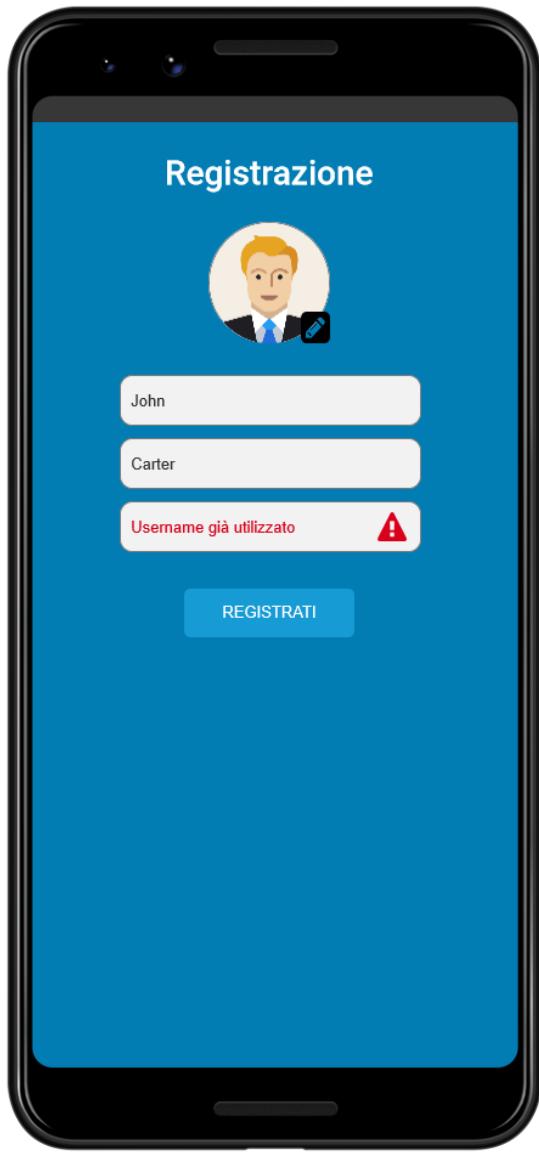


Figura 13 – M12

A seconda della lista da visualizzare scelta all'interno del Navigation Menu, comparirà la scritta "Preferiti" o "Da vedere".

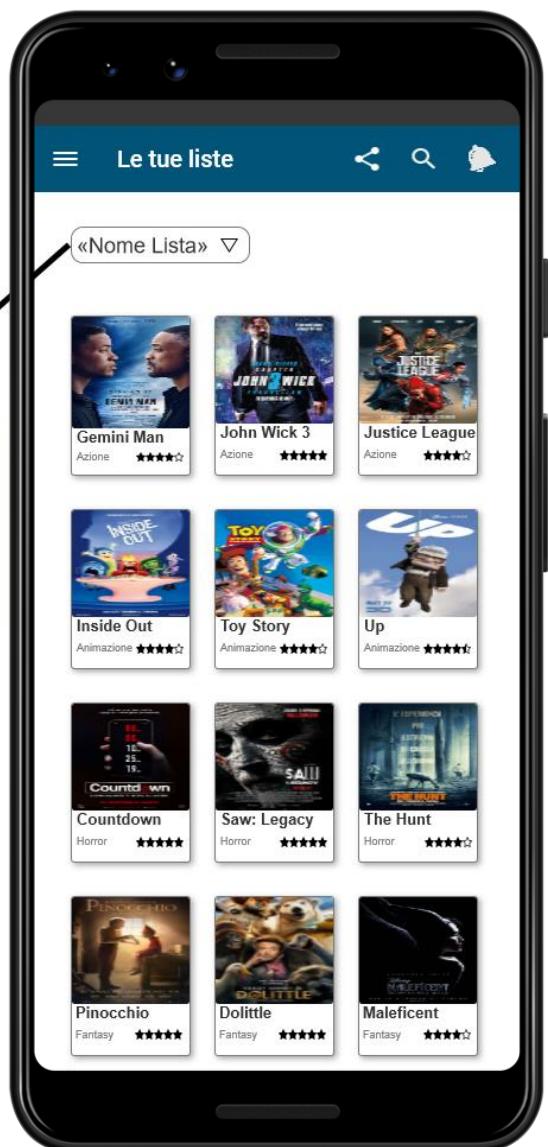


Figura 14 – M13

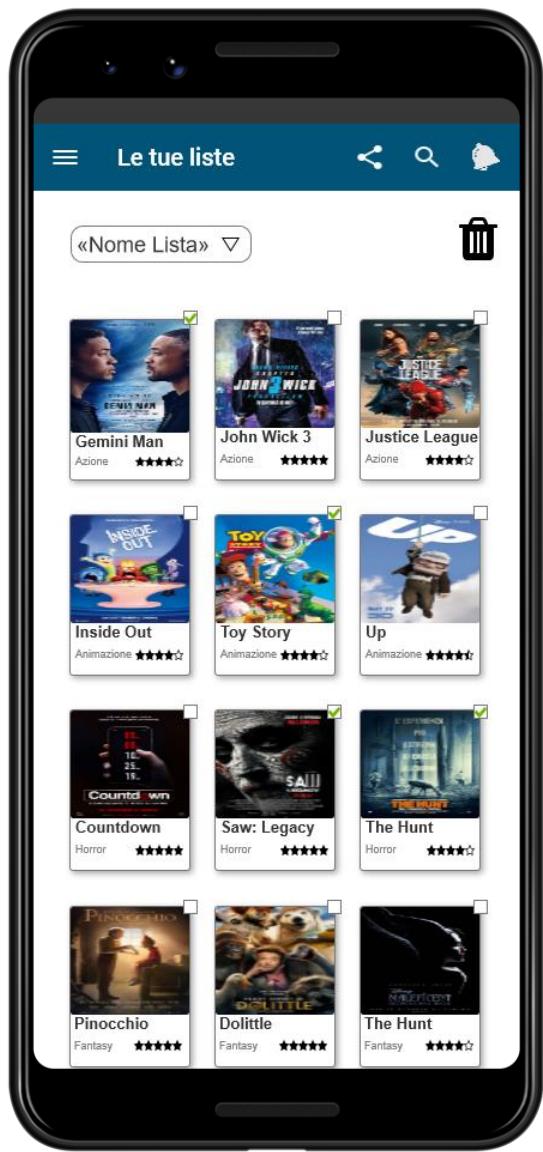


Figura 15 – M14



Figura 16 – M15



Figura 17 – M16

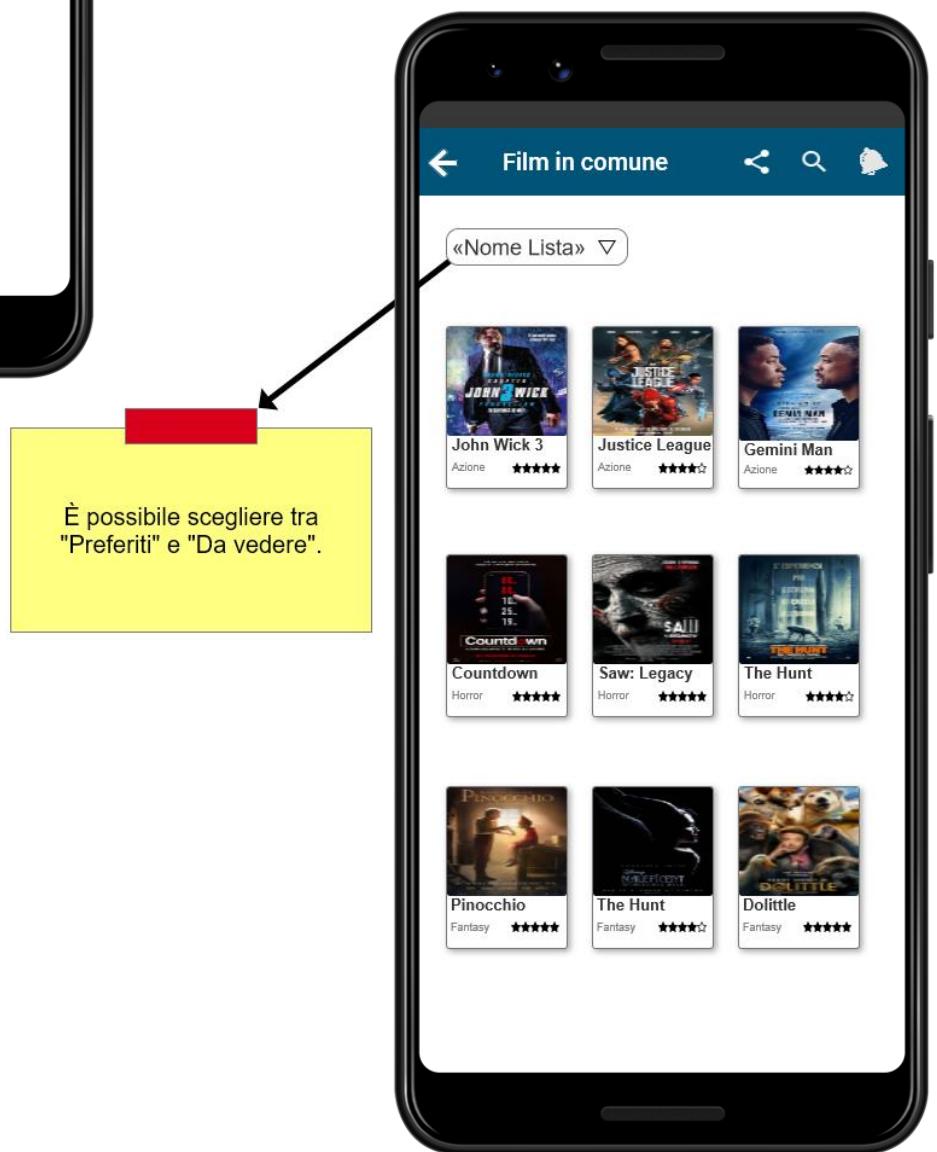


Figura 18 – M17

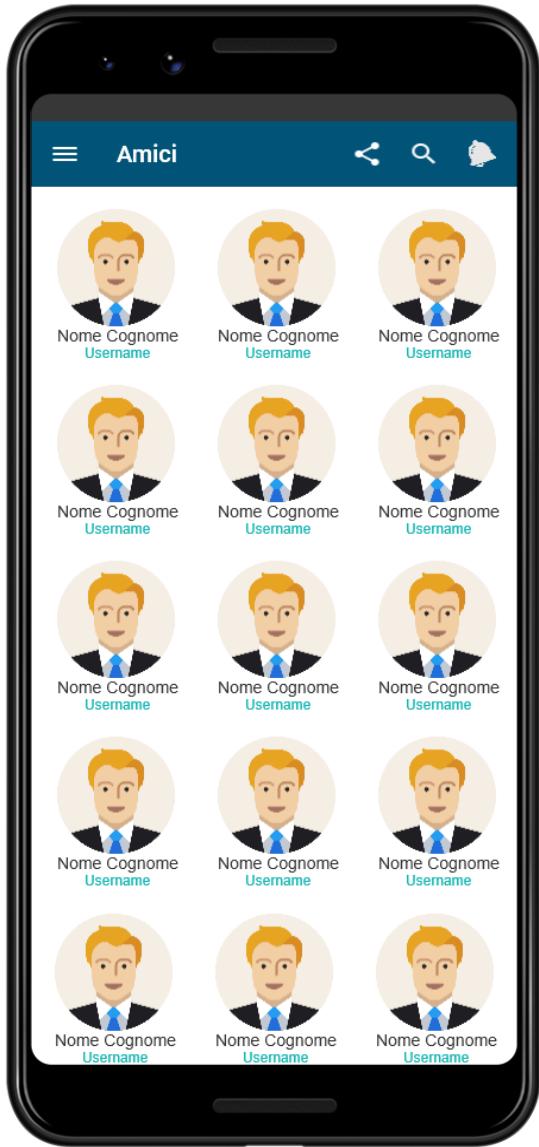


Figura 19 – M18



Figura 20 – M19

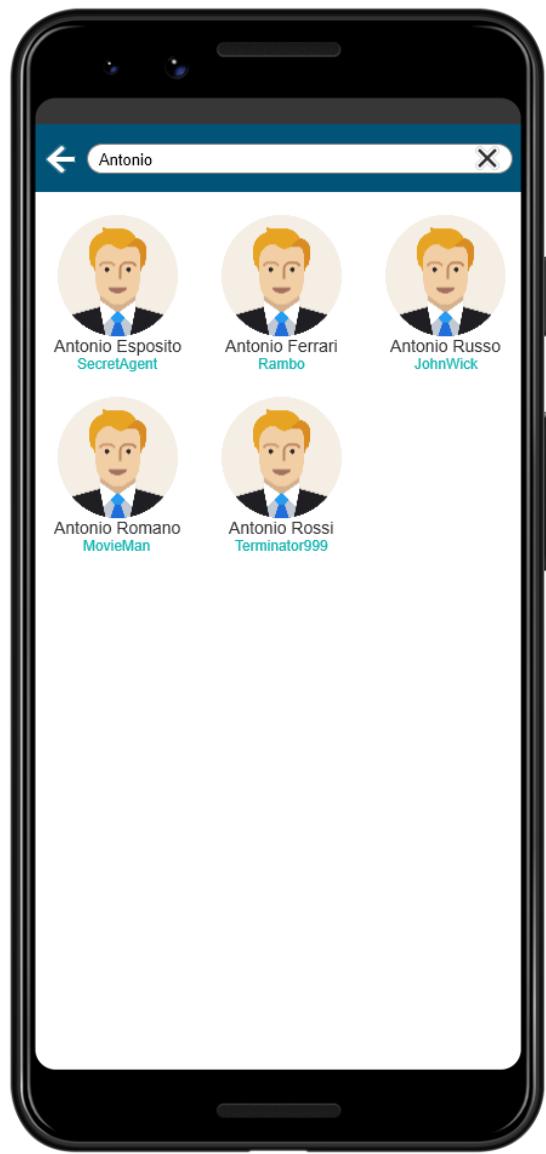


Figura 21 – M20



Figura 22 – M21

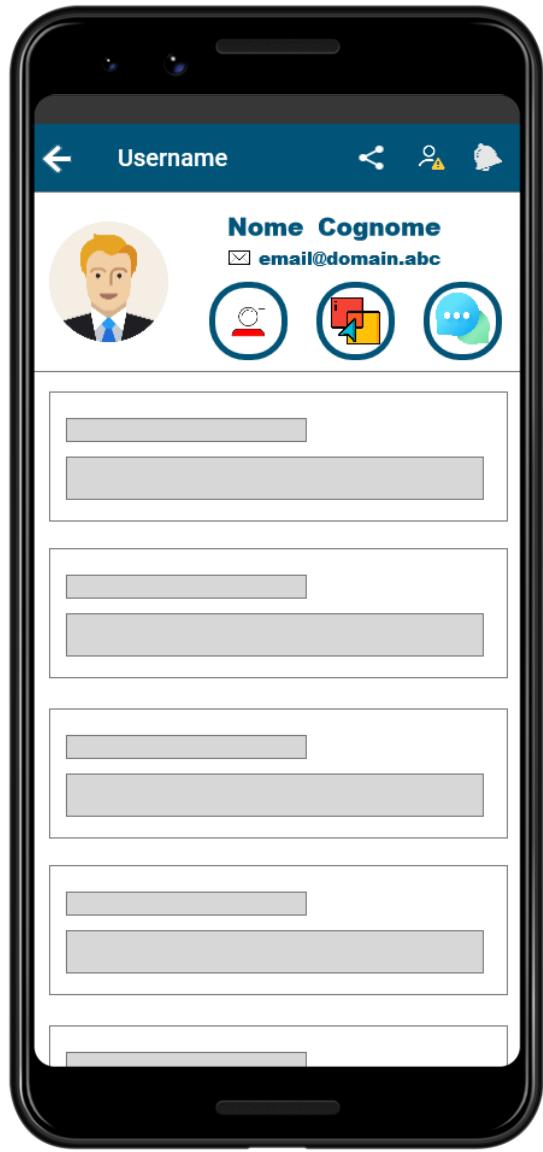


Figura 23 – M22



Figura 24 – M23



Figura 25 – M24



Figura 26 – M25



Figura 27 – M26



Figura 28 – M27

## Desktop

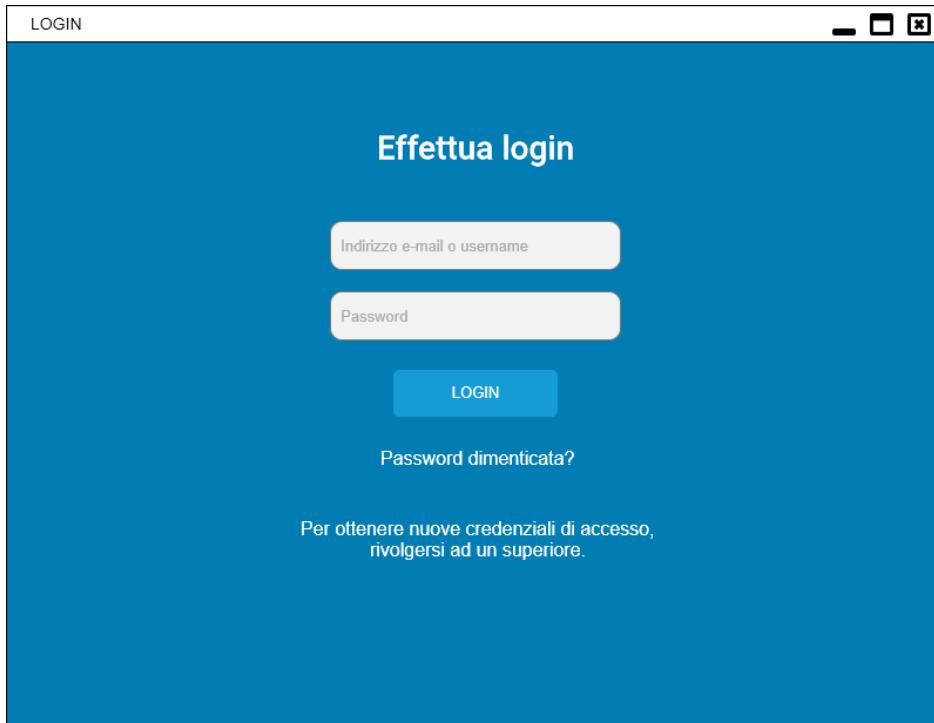


Figura 29 – D0

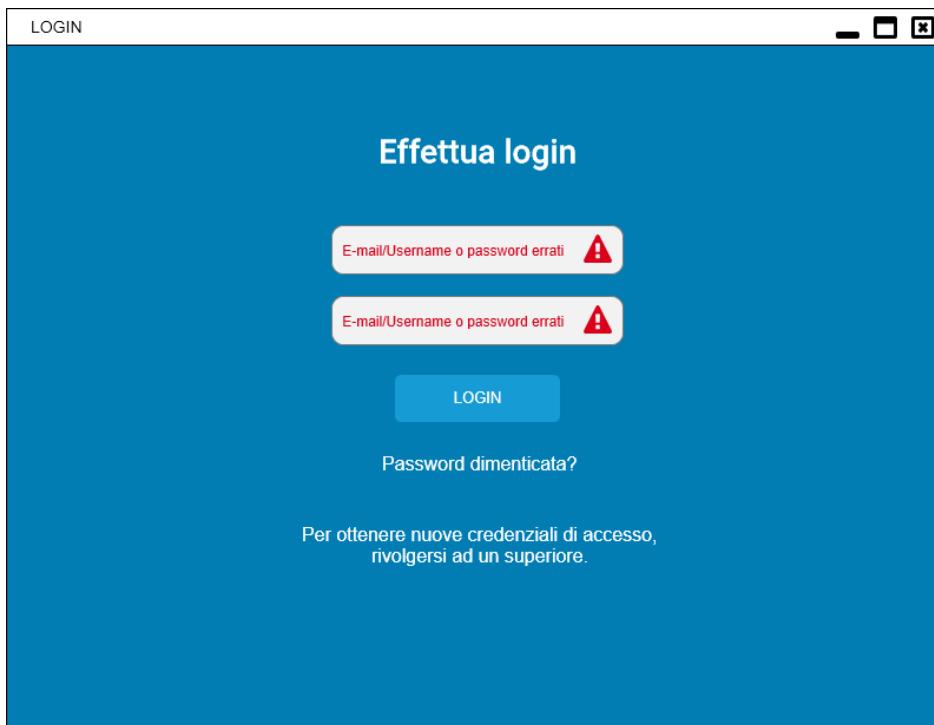


Figura 30 – D1

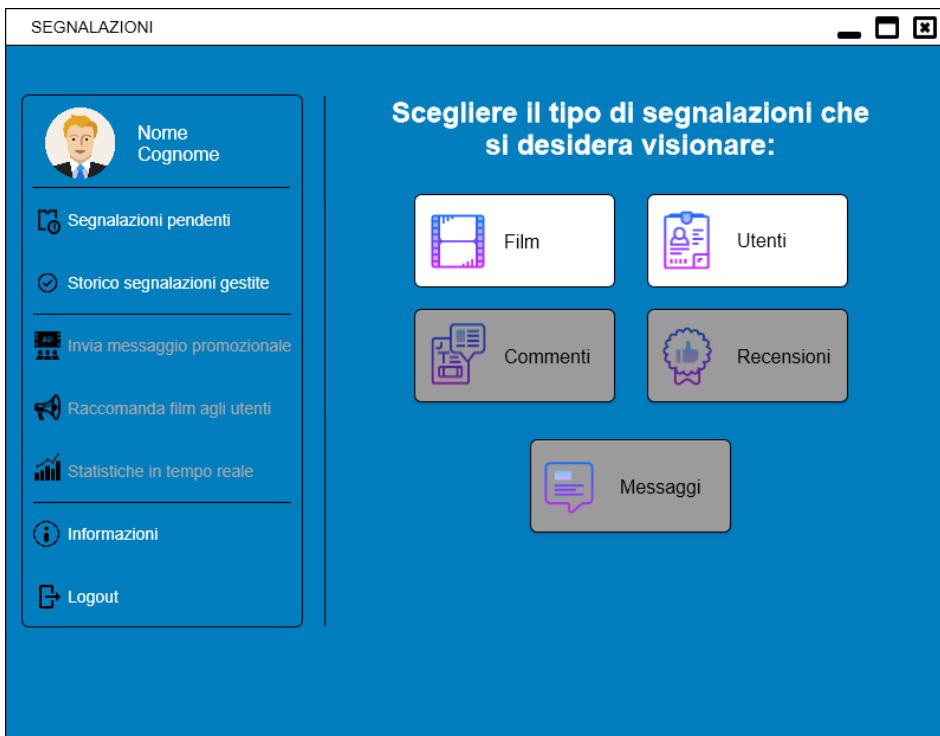


Figura 31 – D2

Nome Cognome @Username Segnalazioni: 47	Nome Cognome @Username Segnalazioni: 41
Nome Cognome @Username Segnalazioni: 35	Nome Cognome @Username Segnalazioni: 32
Nome Cognome @Username Segnalazioni: 30	Nome Cognome @Username Segnalazioni: 24
Nome Cognome @Username Segnalazioni: 14	Nome Cognome @Username Segnalazioni: 12
Nome Cognome @Username Segnalazioni: 3	Nome Cognome @Username Segnalazioni: 1

Figura 32 – D3

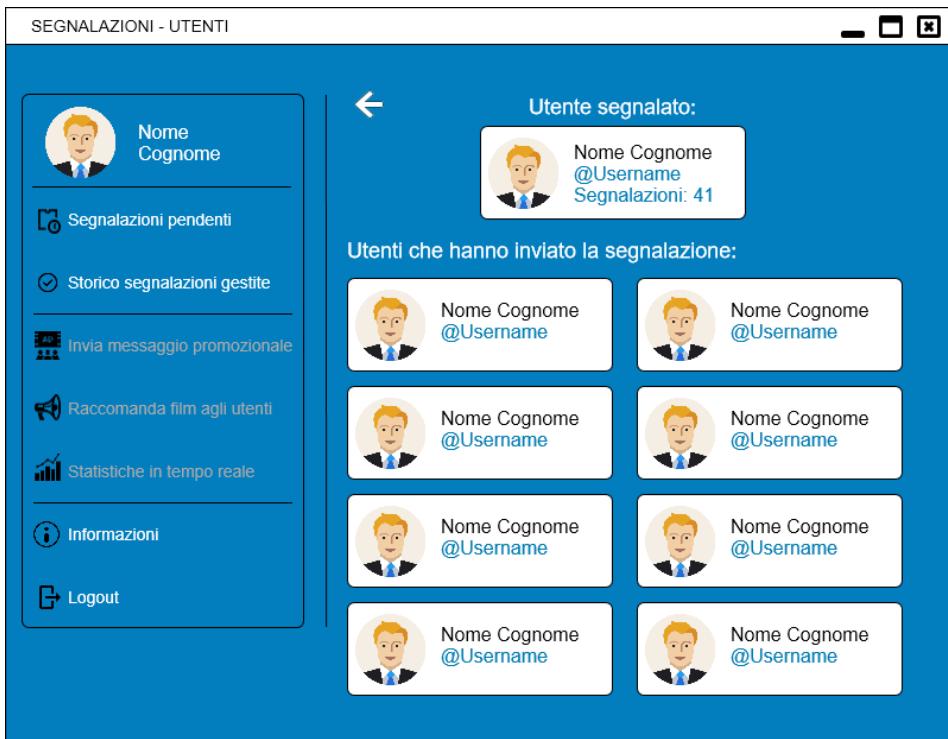


Figura 33 - D4

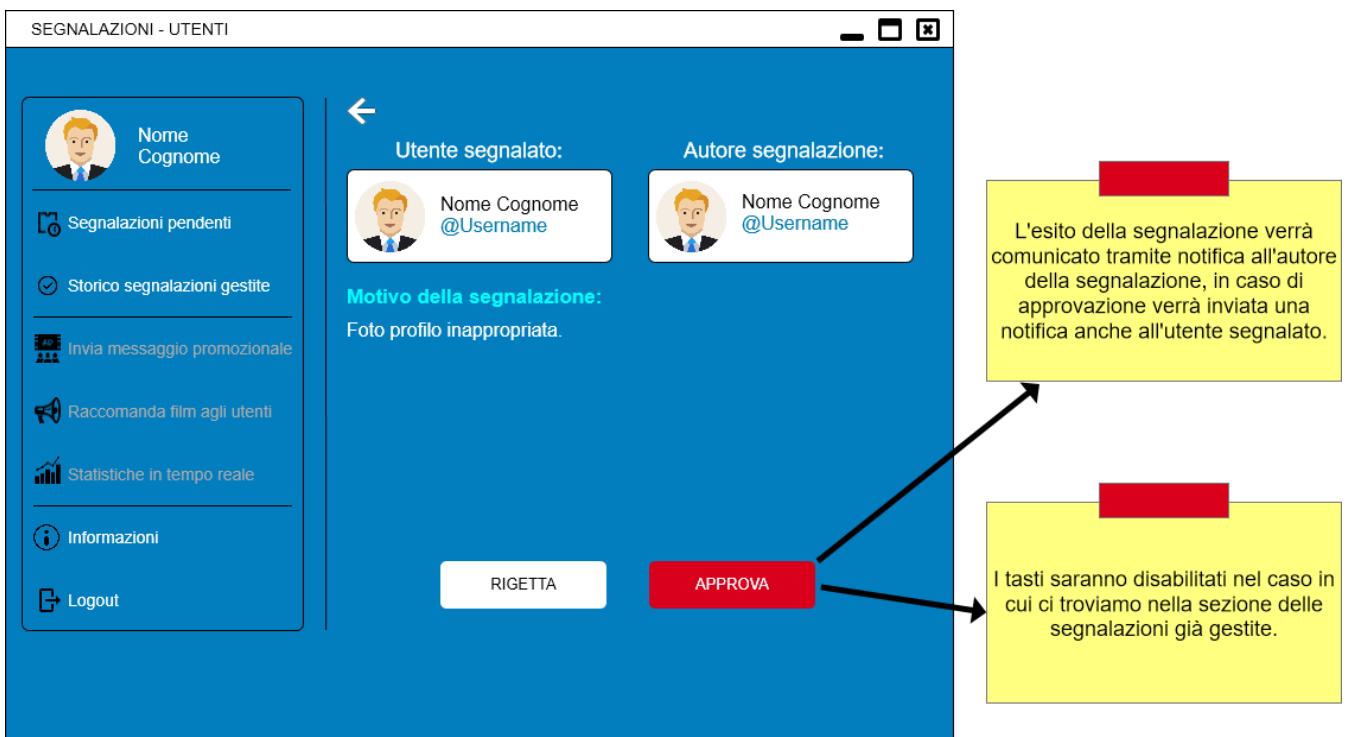


Figura 34 - D5

SEGNALAZIONI - FILM

Cerca

Artemis Fowl Segnalazioni: 13	Harry Potter e i doni della morte - Parte 1 Segnalazioni: 11
John Wick 3 Parabellum Segnalazioni: 4	Maleficent Segnalazioni: 3
Saw: Legacy Segnalazioni: 1	The hunt Segnalazioni: 17

Nome Cognome

- Segnalazioni pendenti
- Storico segnalazioni gestite
- Invia messaggio promozionale
- Raccomanda film agli utenti
- Statistiche in tempo reale
- Informazioni
- Logout

Ordina la lista per ordine alfabetico oppure in ordine decrescente di numero di segnalazioni.

Figura 35 – D6

Fast and Furious

Anno: 2001  
Regista: Rob Cohen  
Genere: Azione, poliziesco, thriller  
Durata: 1h 46m  
Età consigliata: 10+

Segnalazioni: 3

**Descrizione:**  
Per le strade di Los Angeles, gli amanti di motori e velocità competono in gare clandestine, sfrecciando su automobili dalle carene fiammanti e dai carburanti pompati a protossido di azoto. Incaricato di indagare su un clan di rapinatori che trafigano camion in corsa, il giovane agente Brian O'Conner si addenta sotto copertura nel mondo delle corse illegali entrando in contatto con il nerboruto Dominic Toretto, un esperto meccanico ed eccezionale pilota. Dopo aver perso una corsa contro di lui, Brian riesce a metterlo in salvo dalla polizia e da quel momento si aprono per lui le porte del garage e della famiglia di Toretto, dove non proprio tutti sono bendisposti ad accoglierlo.

**Lista utenti che hanno segnalato questo film**

- Nome Cognome  
@Username
- Nome Cognome  
@Username
- Nome Cognome  
@Username

Figura 36 – D7

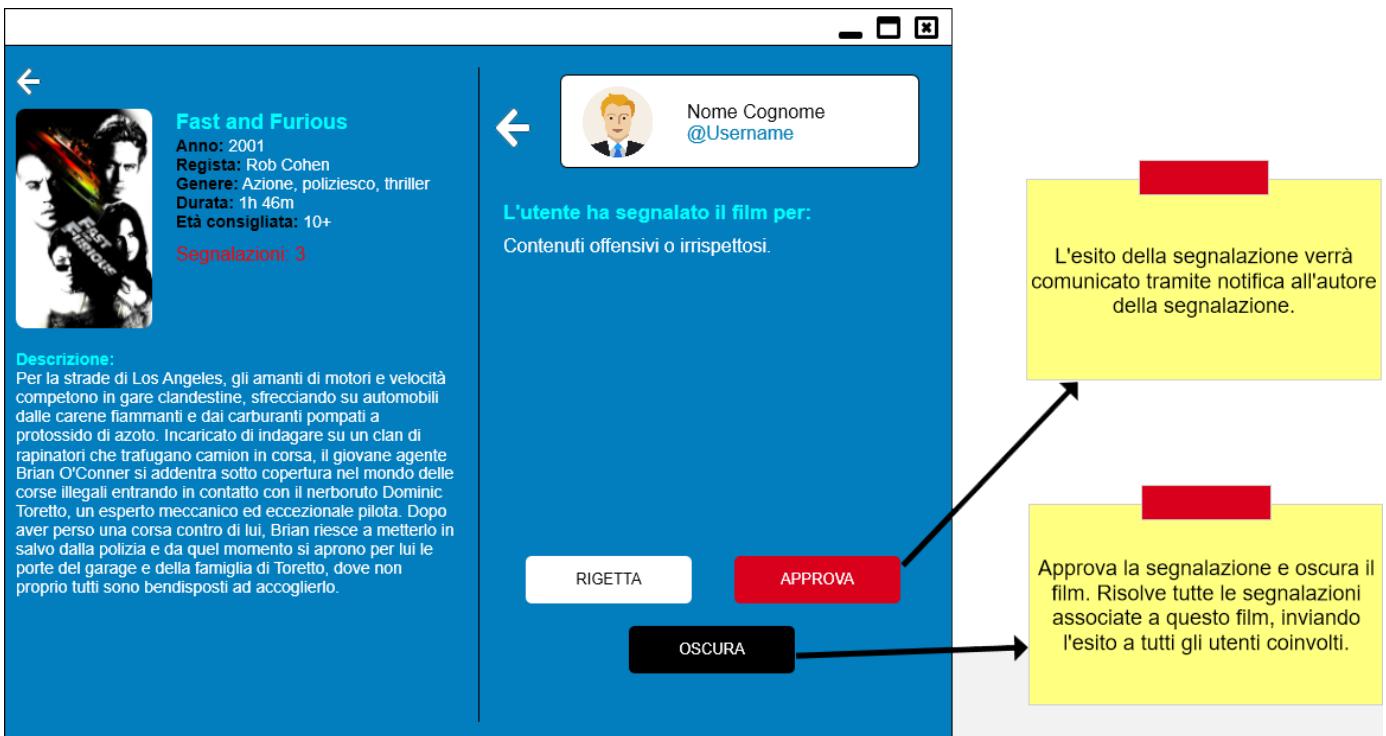


Figura 37 – D8

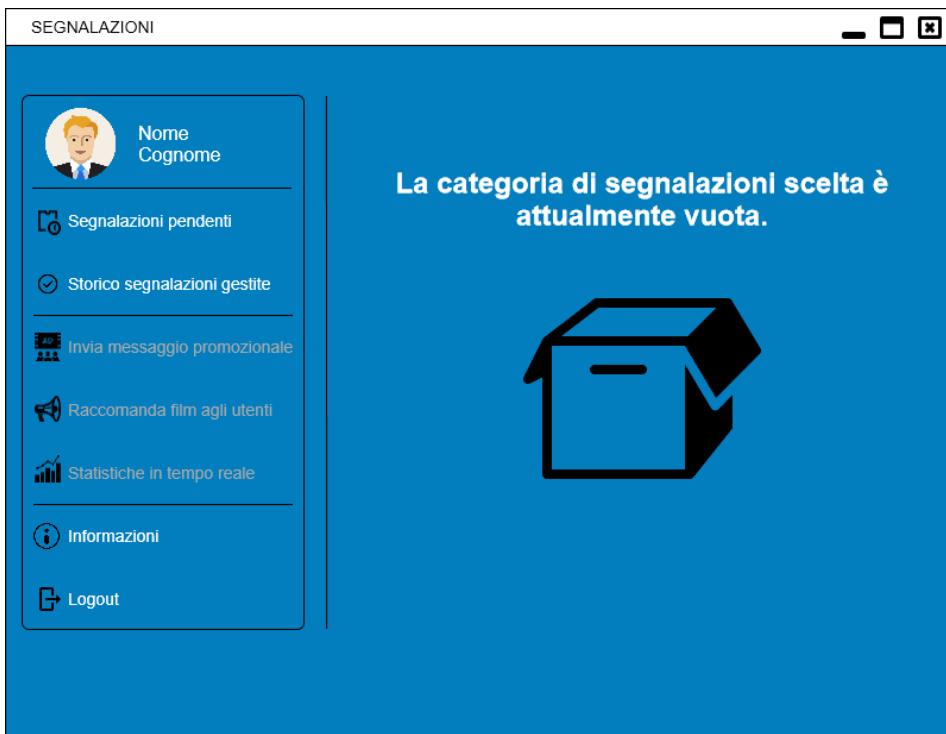


Figura 38 – D9

## Tabelle di Cockburn

Le seguenti tabelle illustrano il funzionamento che è stato previsto per i prototipi presenti nelle pagine precedenti. Per comodità, le scritte che indicano il nome di un mockup sono cliccабili, e consentono di essere reindirizzati al relativo mockup rappresentato dal nome.

### Mobile

<b>USE CASE #1</b>	<i>Aggiunge film ad una lista</i>		
<b>Goal in Context</b>	L'utente aggiunge il film ad una lista (da vedere/preferiti).		
<b>Preconditions</b>	L'utente ha effettuato il login al sistema e ha effettuato una ricerca che ha prodotto un insieme di film non vuoto.		
<b>Success End Condition</b>	Il film è stato aggiunto correttamente alla lista desiderata.		
<b>Failed End Condition</b>	Nessuna.		
<b>Primary Actor</b>	Utente Loggato		
<b>Trigger</b>	Pressione dell'icona identificata dai tre pallini verticali di <b>M1</b> , associata al film da aggiungere.		
<b>DESCRIPTION</b> Aggiunta alla lista film da vedere.	<b>Step n°</b>	<b>Utente Loggato</b>	<b>Sistema</b>
	1	Pressione dell'icona denotata dai tre pallini verticali di <b>M1</b> .	
	2		Mostra <b>M2</b> .
	3	Preme il tasto «Aggiungi a lista “Film da vedere”».	
	4		Mostra <b>M4</b> , con messaggio “Aggiunta avvenuta con successo”.
	5		Mostra <b>M1</b> .
<b>EXTENSION #1:</b> Aggiunta alla lista dei preferiti.	3.1	Preme il tasto «Aggiungi a lista “Film preferiti”».	
	4.1		Mostra <b>M4</b> , con messaggio “Aggiunta avvenuta con successo”.
	5.1		Mostra <b>M1</b> .

<b>USE CASE #2</b>	<i>Rimuove film da una lista</i>	
<b>Goal in Context</b>	L'utente rimuove il film da una lista (da vedere/preferiti).	
<b>Preconditions</b>	L'utente ha effettuato il login al sistema.	
<b>Success End Condition</b>	Il film è stato rimosso correttamente dalla lista desiderata.	
<b>Failed End Condition</b>	Nessuna	
<b>Primary Actor</b>	Utente Loggato	
<b>Trigger</b>	<p>Nel caso della ricerca, pressione dell'icona identificata dai tre pallini verticali, associata al film da rimuovere.</p> <p>Nel caso della sezione relativa alle liste: pressione del pulsante "Preferiti" / "Da vedere" nel navigation menu di <b>M5</b>, dopo quest'azione il sistema mostra <b>M13</b>.</p>	
<b>DESCRIPTION</b> Rimozione dalla lista dei film da vedere.	<b>Step n°</b>	<b>Utente Loggato</b>
	1	Effettua una ricerca del film desiderato da <b>M6</b> .
	2	Recupera una lista non vuota di film e mostra <b>M1</b> .
	3	Pressione dell'icona, denotata dai tre pallini verticali, associata ad un film.
	4	Mostra <b>M3</b> .
	5	Preme il tasto «Rimuovi da lista "Film da vedere"».
	6	Mostra <b>M4</b> , con messaggio "Rimozione avvenuta con successo".
	7	Mostra <b>M1</b> .
<b>EXTENSION #1:</b> Rimozione dalla lista dei preferiti.	5.1	Preme il tasto «Rimuovi da lista "Film preferiti"».
	6.1	Mostra <b>M4</b> , con messaggio "Rimozione avvenuta con successo".
	7.1	Mostra <b>M1</b> .
<b>EXTENSION #2:</b> Rimozione film tramite la sezione relativa alle liste. Cancellare uno o più film da <b>M13</b> .	1.2	Tenendo premuto su un film, in <b>M13</b> , lo seleziona.
	2.2	Mostra <b>M14</b> .
	3.2	Seleziona altri zero o più film.
	4.2	Clicca sull'icona in alto a destra del cestino.
	5.2	Aggiorna <b>M13</b> , rimuovendo i film selezionati.

<b>USE CASE #3</b>	<i>Effettua login</i>			
<b>Goal in Context</b>	Identificarsi nell'applicazione.			
<b>Preconditions</b>	L'utente deve avere già un account.			
<b>Success End Condition</b>	L'utente effettua con successo il login.			
<b>Failed End Condition</b>	1. Credenziali errate. 2. Username/Indirizzo e-mail e/o password non inseriti.			
<b>Primary Actor</b>	Utente			
<b>Trigger</b>	L'utente clicca sul Navigation Menu di <b>M0</b> e successivamente sul bottone "Logout", oppure l'utente avvia per la prima volta l'applicazione.			
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>	<b>Social Network Esterno</b>
	1	Apre l'applicazione per la prima volta.		
	2		Mostra <b>M7</b> .	
	3	Inserisce username/indirizzo e-mail e password, dopodiché clicca sul bottone "Login".		
	4		Mostra <b>M0</b> .	
<b>EXTENSION #1:</b> Credenziali errate.	4.1		Mostra <b>M8</b> .	
<b>EXTENSION #2:</b> Username/Indirizzo e-mail e/o password non inseriti.	3.2	Clicca il tasto "Login".		
	4.2		Mostra <b>M8</b> .	
<b>EXTENSION #3:</b> L'utente effettua il login attraverso un account social.	3.3	Clicca su una delle icone relative all'account social con il quale desidera accedere.		
	4.3			Mostra schermata di accesso.
	5.3	Digita credenziali di accesso social ed effettua il login.		
	6.3		Mostra <b>M0</b> .	

<b>EXTENSION #4:</b> L'utente effettua il logout dall'applicazione.	1.4	Clicca sul Navigation Menu di <b>M0</b> e successivamente sul bottone “Logout”.		
	2.4		Mostra <b>M7</b> .	
	3.4	Inserisce username/indirizzo e-mail e password, dopodiché clicca sul bottone “Login”.		
	4.4		Mostra <b>M0</b> .	

<b>USE CASE #4</b>	<i>Effettua registrazione</i>			
<b>Goal in Context</b>	Ottenere credenziali di accesso per identificarsi nell'applicazione.			
<b>Preconditions</b>	L'utente deve possedere un indirizzo e-mail.			
<b>Success End Condition</b>	L'utente effettua con successo la registrazione.			
<b>Failed End Condition</b>	1. Username non disponibile. 2. E-mail già registrata. 3. Password non coincidenti. 4. La password non rispetta i vincoli. Almeno un campo non è stato compilato.			
<b>Primary Actor</b>	Utente			
<b>Trigger</b>	L'utente clicca sul Navigation Menu di <b>M0</b> e successivamente sul bottone “Logout”, oppure l'utente avvia per la prima volta l'applicazione.			
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>	<b>Social Network Esterno</b>
	1	Apre l'applicazione per la prima volta.		
	2		Mostra <b>M7</b> .	
	3	Clicca sulla scritta “Crea un nuovo account”.		
	4		Mostra <b>M9</b> .	
	5	Inserisce nome, cognome, username, e-mail, password e conferma password e clicca sul bottone “Registrati”.		
	6		Mostra <b>M7</b> .	

<b>EXTENSION #1:</b> Username non disponibile.	6.1		Mostra <b>M10</b> con la scritta “Username già utilizzato”.	
<b>EXTENSION #2:</b> E-mail non disponibile.	6.2		Mostra <b>M10</b> con la scritta “E-mail già utilizzata”.	
<b>EXTENSION #3:</b> Password non coincidenti.	6.3		Mostra <b>M10</b> con la scritta “Le password non coincidono”.	
<b>EXTENSION #4:</b> La password non rispetta i vincoli.	6.4		Mostra <b>M10</b> con la scritta “La password non rispetta i vincoli”.	
<b>EXTENSION #5:</b> Almeno un campo non è stato compilato.	6.5		Mostra <b>M10</b> con i campi non inseriti in rosso, e i relativi messaggi di errore	
<b>EXTENSION #6:</b> L'utente effettua la registrazione attraverso un account social.	3.6	Clicca su una delle icone relative all'account social con il quale desidera creare il nuovo account.		
	4.6			Mostra schermata di registrazione.
	5.6	Inserisce i dati per registrarsi.		
	6.6			Chiede i permessi necessari ai fini dell'associazione dell'account.
	7.6	Concede i permessi.		
	8.6		Mostra <b>M0</b> .	
<b>EXTENSION #7:</b> L'utente effettua il logout.	1.7	Clicca sul Navigation Menu di <b>M0</b> e successivamente sul bottone “Logout”.		
	2.7		Mostra <b>M7</b> .	
	3.7	Clicca sulla scritta “Crea un nuovo account”.		
	4.7		Mostra <b>M9</b> .	
	5.7	Inserisce nome, cognome, username, e-mail, password e conferma password e clicca sul bottone “Registrati”.		
	6.7		Mostra <b>M7</b> .	

<b>USE CASE #5</b>	<i>Cerca film</i>			
<b>Goal in Context</b>	Trovare e visualizzare i dettagli di un film all'interno dell'applicazione.			
<b>Preconditions</b>	L'utente si trova all'interno di una qualunque delle schermate in cui è presente l'icona a forma di lente d'ingrandimento o la barra di ricerca, eccetto <b>M13</b> .			
<b>Success End Condition</b>	L'utente ha trovato il film ricercato.			
<b>Failed End Condition</b>	1. Il film non è presente nell'archivio.			
<b>Primary Actor</b>	Utente Loggato			
<b>Trigger</b>	L'utente clicca sull'icona a forma di lente d'ingrandimento o sulla barra di ricerca se già presente a schermo.			
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente</b>	<b>Sistema</b>	<b>API Film</b>
	1	Clicca sulla barra di ricerca/icona identificata da una lente d'ingrandimento e digita il titolo del film.		
	2			Recupera un insieme non vuoto di film.
	3		Mostra <b>M1</b> con la lista di film ottenuta.	
	4	Clicca su una delle icone di <b>M1</b> che rappresenta il film desiderato.		
	5		Mostra <b>M15</b> .	
<b>EXTENSION #1:</b> Il film non è presente nell'archivio.	2.1			Recupera un insieme vuoto di film.
	3.1		Mostra <b>M16</b> con la scritta "La ricerca non ha prodotto risultati!".	

<b>USE CASE #6</b>	<i>Invia richiesta di amicizia</i>		
<b>Goal in Context</b>	Inviare una richiesta di amicizia ad un particolare utente.		
<b>Preconditions</b>	L'utente che al quale si desidera inviare la richiesta di amicizia, non è già presente nella propria lista di amici.		
<b>Success End Condition</b>	L'utente ricercato è stato trovato e gli è stata inviata una richiesta di amicizia.		
<b>Failed End Condition</b>	1. L'utente ricercato non esiste.		
<b>Primary Actor</b>	Utente Loggato		
<b>Trigger</b>	Pressione del tasto denotato da una lente d'ingrandimento di <b>M18</b> .		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente Loggato</b>	<b>Sistema</b>
	1	Preme il tasto denotato da una lente d'ingrandimento di <b>M18</b> .	
	2		Mostra <b>M19</b> .
	3	Digita nome e cognome, username oppure indirizzo e-mail dell'utente che vuole ricercare, e preme invio.	
	4		Mostra <b>M20</b> con la lista dei risultati trovati in base all'input dell'utente.
	5	Preme sull'avatar della persona che corrisponde ai criteri di ricerca.	
	6		Mostra <b>M21</b> .
	7	Preme l'icona dell'avatar contrassegnato da un "+" in alto a destra.	
<b>EXTENSION #1:</b> L'utente ricercato non esiste.	4.1		Mostra <b>M16</b> con la scritta "La ricerca non ha prodotto risultati!".

<b>USE CASE #7</b>	<i>Rimuove amico</i>		
<b>Goal in Context</b>	Rimuovere un determinato utente dalla propria lista di amici.		
<b>Preconditions</b>	La lista degli amici dell'utente loggato non è vuota. L'utente che si desidera rimuovere dalla propria lista di amici, appartiene a tale lista.		
<b>Success End Condition</b>	L'utente è stato correttamente rimosso dalla propria lista di amici.		
<b>Failed End Condition</b>	Nessuna.		
<b>Primary Actor</b>	Utente Loggato		
<b>Trigger</b>	Pressione del tasto “Amici” di <b>M5</b> .		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente Loggato</b>	<b>Sistema</b>
	1	Preme il tasto “Amici” di <b>M5</b> .	
	2		Mostra <b>M18</b> .
	3	Scorre <b>M18</b> alla ricerca dell'amico che desidera rimuovere e clicca sull'avatar dell'utente trovato.	
	4		Mostra <b>M22</b> .
	5	Preme l'icona dell'avatar contrassegnato da un “–” in alto a destra.	
	6		Mostra <b>M21</b> .

<b>USE CASE #8</b>	<i>Visualizza film in comune</i>		
<b>Goal in Context</b>	Visualizzare la lista dei film in comune con un determinato amico.		
<b>Preconditions</b>	L'utente con il quale si desidera visualizzare la lista dei film in comune, appartiene alla propria lista di amici.		
<b>Success End Condition</b>	Viene mostrata la lista dei film in comune con l'amico selezionato.		
<b>Failed End Condition</b>	1. Non ci sono film in comune con l'amico selezionato.		
<b>Primary Actor</b>	Utente Loggato		
<b>Trigger</b>	Pressione del tasto "Amici" di <b>M5</b> .		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente Loggato</b>	<b>Sistema</b>
	1	Preme il tasto "Amici" di <b>M5</b> .	
	2		Mostra <b>M18</b> .
	3	Scorre <b>M18</b> alla ricerca dell'amico con il quale desidera visualizzare la propria lista di film <b>preferiti</b> in comune e clicca sull'avatar dell'utente trovato.	
	4		Mostra <b>M22</b> .
	5	Preme il secondo tasto contraddistinto da due quadrati che s'intersecano e un puntatore azzurro di <b>M22</b> .	
	6		Mostra <b>M17</b> con la lista dei film preferiti in comune con l'amico selezionato e con al posto di «Nome Lista» la scritta «Preferiti» di default.
<b>EXTENSION #1:</b> Non ci sono film in comune con l'amico selezionato.	6.1		Mostra <b>M24</b> .
<b>EXTENSION #2:</b> L'utente desidera visualizzare la lista di film <b>da vedere</b> in comune con l'amico selezionato.	7.2	Preme il tasto con la scritta "Preferiti" e seleziona "Da vedere".	
	8.2		Mostra <b>M17</b> con la lista dei film da vedere in comune con l'amico selezionato e con al posto di «Nome Lista» la scritta «Da vedere».

<b>USE CASE #9</b>	<i>Segnala film</i>		
<b>Goal in Context</b>	Inviare la segnalazione in merito ad un film.		
<b>Preconditions</b>	Il film deve essere disponibile nel catalogo. L'utente si trova nella schermata <b>M1</b> .		
<b>Success End Condition</b>	Viene inoltrata una segnalazione agli amministratori in merito al film scelto.		
<b>Failed End Condition</b>	1. Text field associata alla voce “Altro” vuota.		
<b>Primary Actor</b>	Utente Loggato		
<b>Trigger</b>	Pressione dell'icona identificata dai tre pallini verticali di <b>M1</b> .		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente Loggato</b>	<b>Sistema</b>
	1	Pressione dell'icona denotata dai tre pallini verticali di <b>M1</b> , associata al film da segnalare.	
	2		Mostra <b>M2</b> .
	3	Preme il tasto “Segnala contenuti inappropriati”.	
	4		Mostra <b>M23</b> .
	5	Preme il tasto associato al tipo di segnalazione desiderato, nel caso di “Altro” descrive la natura della segnalazione, e infine preme il tasto “Invia”.	
	6		Viene mostrato il messaggio “Segnalazione inviata” e tutti i componenti d’input di <b>M23</b> vengono disabilitati
<b>EXTENSION #1:</b> Text field associata alla voce “Altro” vuota.	5.1	Preme il tasto “Altro” senza digitare nulla nella text field associata alla segnalazione e preme il tasto “Invia”.	
	6.1		Mostra <b>M27</b> .

<b>USE CASE #10</b>	<i>Segnala utente</i>		
<b>Goal in Context</b>	Segnalare un particolare utente registrato.		
<b>Preconditions</b>	L'utente che si desidera segnalare deve essere registrato nella piattaforma.		
<b>Success End Condition</b>	È stata inoltrata agli amministratori una segnalazione riguardante l'utente scelto.		
<b>Failed End Condition</b>	1. Text field associata alla voce “Altro” vuota.		
<b>Primary Actor</b>	Utente Loggato		
<b>Trigger</b>	Pressione dell'icona di <b>M21/M22</b> denotata da un utente accanto ad un'icona triangolare gialla.		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Utente Loggato</b>	<b>Sistema</b>
	1	Preme l'icona di <b>M21 o M22</b> denotata da un utente accanto ad un'icona triangolare gialla.	
	2		Mostra <b>M25</b> .
	3	Preme il tasto associato al tipo di segnalazione desiderato, nel caso di “Altro” descrive la natura della segnalazione, e infine preme il tasto “Invia”.	
	4		Viene mostrato il messaggio “Segnalazione inviata” e tutti i componenti d’input di <b>M25</b> vengono disabilitati
<b>EXTENSION #1:</b> Text field associata alla voce “Altro” vuota.	3.1	Preme il tasto “Altro” senza digitare nulla nella text field associata alla segnalazione e preme il tasto “Invia”.	
	4.1		Mostra <b>M26</b> .

## Desktop

<b>USE CASE #1</b>	<i>Effettua login</i>		
<b>Goal in Context</b>	Identificarsi nell'applicativo.		
<b>Preconditions</b>	L'amministratore deve avere già un account.		
<b>Success End Condition</b>	L' amministratore effettua con successo il login.		
<b>Failed End Condition</b>	1. Credenziali errate. 2. Username/Indirizzo e-mail e/o password non inseriti.		
<b>Primary Actor</b>	Amministratore		
<b>Trigger</b>	L'amministratore avvia per la prima volta l'applicativo, oppure clicca sul tasto "Logout" di una delle schermate in cui è presente il menu laterale sulla sinistra, come <b>D2</b> .		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Amministratore</b>	<b>Sistema</b>
	1	Apre l'applicativo per la prima volta.	
	2		Mostra <b>D0</b> .
	3	Inserisce e-mail e password, dopodiché clicca sul bottone "Login".	
	4		Mostra <b>D2</b> con la scritta "Segnalazioni pendenti" evidenziata.
<b>EXTENSION #1:</b> Credenziali errate.	4.1		Mostra <b>D1</b> .
<b>EXTENSION #2:</b> Username/Indirizzo e-mail e/o password non inseriti.	3.2	Clicca il tasto "Login".	
	4.2		Mostra <b>D1</b> .
	1.3	Clicca sul tasto "Logout" presente nel menu laterale a sinistra in una delle schermate come <b>D2</b> .	
	2.3		Mostra <b>D0</b> .
	3.3	Inserisce username/indirizzo e-mail e password, dopodiché clicca sul bottone "Login".	
	4.3		Mostra <b>D2</b> con la scritta "Segnalazioni pendenti" evidenziata.

<b>USE CASE #2</b>	<i>Visualizza segnalazioni gestite</i>		
<b>Goal in Context</b>	Visualizzare nel sistema lo storico delle segnalazioni gestite.		
<b>Preconditions</b>	Nessuna.		
<b>Success End Condition</b>	Il sistema mostra una schermata con la segnalazione gestita scelta.		
<b>Failed End Condition</b>	1. Non sono presenti segnalazioni gestite per la categoria scelta.		
<b>Primary Actor</b>	Amministratore Loggato		
<b>Trigger</b>	L'amministratore loggato clicca sul tasto “Storico segnalazioni gestite” di una delle schermate in cui è presente il menu laterale sulla sinistra, come <b>D2</b> .		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Amministratore</b>	<b>Sistema</b>
	1	Clicca sul tasto “Storico segnalazioni gestite” di una delle schermate in cui è presente il menu laterale sulla sinistra, come <b>D2</b> .	
	2		Mostra <b>D2</b> con la scritta “Storico segnalazioni gestite” evidenziata.
	3	Clicca sul tasto “Film”.	
	4		Mostra <b>D6</b> .
	5	Clicca sul film in merito al quale desidera visualizzare le segnalazioni.	
	6		Mostra <b>D7</b> .
	7	Clicca su uno degli utenti che ha effettuato la segnalazione indirizzata al film selezionato allo step 5.	
	8		Mostra <b>D8</b> , con il tasto associato alla risposta data evidenziato, e i restanti due tasti disabilitati.
<b>EXTENSION #1:</b> Non sono presenti segnalazioni gestite per la categoria scelta.	4.1		Mostra <b>D9</b> .

<b>EXTENSION #2:</b> L'amministratore loggato visualizza le segnalazioni gestite riguardanti gli utenti.	3.2	Clicca sul tasto “Utenti”	
	4.2		Mostra <b>D3</b> .
	5.2	Clicca sull'utente in merito al quale desidera visualizzare le segnalazioni.	
	6.2		Mostra <b>D4</b> .
	7.2	Clicca su uno degli utenti che ha effettuato la segnalazione indirizzata all'utente selezionato allo step 5.2.	
	8.2		Mostra <b>D5</b> , con il tasto associato alla risposta data evidenziato, e il restante tasto disabilitato.

<b>USE CASE #3</b>	<i>Risolve segnalazione pendente</i>		
<b>Goal in Context</b>	Visualizzare e risolvere una segnalazione pendente.		
<b>Preconditions</b>	Nessuna.		
<b>Success End Condition</b>	Il sistema mostra una schermata con la segnalazione da risolvere scelta.		
<b>Failed End Condition</b>	1. Non sono presenti segnalazioni pendenti per la categoria scelta.		
<b>Primary Actor</b>	Amministratore Loggato		
<b>Trigger</b>	L'amministratore loggato clicca sul tasto “Segnalazioni pendenti” di una delle schermate in cui è presente il menu laterale sulla sinistra, come <b>D2</b> .		
<b>DESCRIPTION</b>	<b>Step n°</b>	<b>Amministratore</b>	<b>Sistema</b>
	1	Clicca sul tasto “Segnalazioni pendenti” di una delle schermate in cui è presente il menu laterale sulla sinistra, come <b>D2</b> .	
	2		Mostra <b>D2</b> con la scritta “Segnalazioni pendenti” evidenziata.
	3	Clicca sul tasto “Film”.	
	4		Mostra <b>D6</b> .
	5	Clicca sul film in merito al quale desidera visualizzare le segnalazioni.	
	6		Mostra <b>D7</b> .
	7	Clicca su uno degli utenti che ha effettuato la segnalazione indirizzata al film selezionato allo step 5.	
	8		Mostra <b>D8</b> .
	9	Clicca sul tasto “Approva” o sul tasto “Rigetta”.	
	10		Notifica l'autore dell'esito della segnalazione.
	11		Nel caso in cui sia presente almeno un'altra segnalazione pendente, mostra <b>D7</b> con il numero di segnalazioni aggiornato e senza l'icona associata all'utente scelto allo step 7. Nel caso in cui non siano presenti altre segnalazioni, mostra <b>D6</b> privato del film scelto allo step 5.

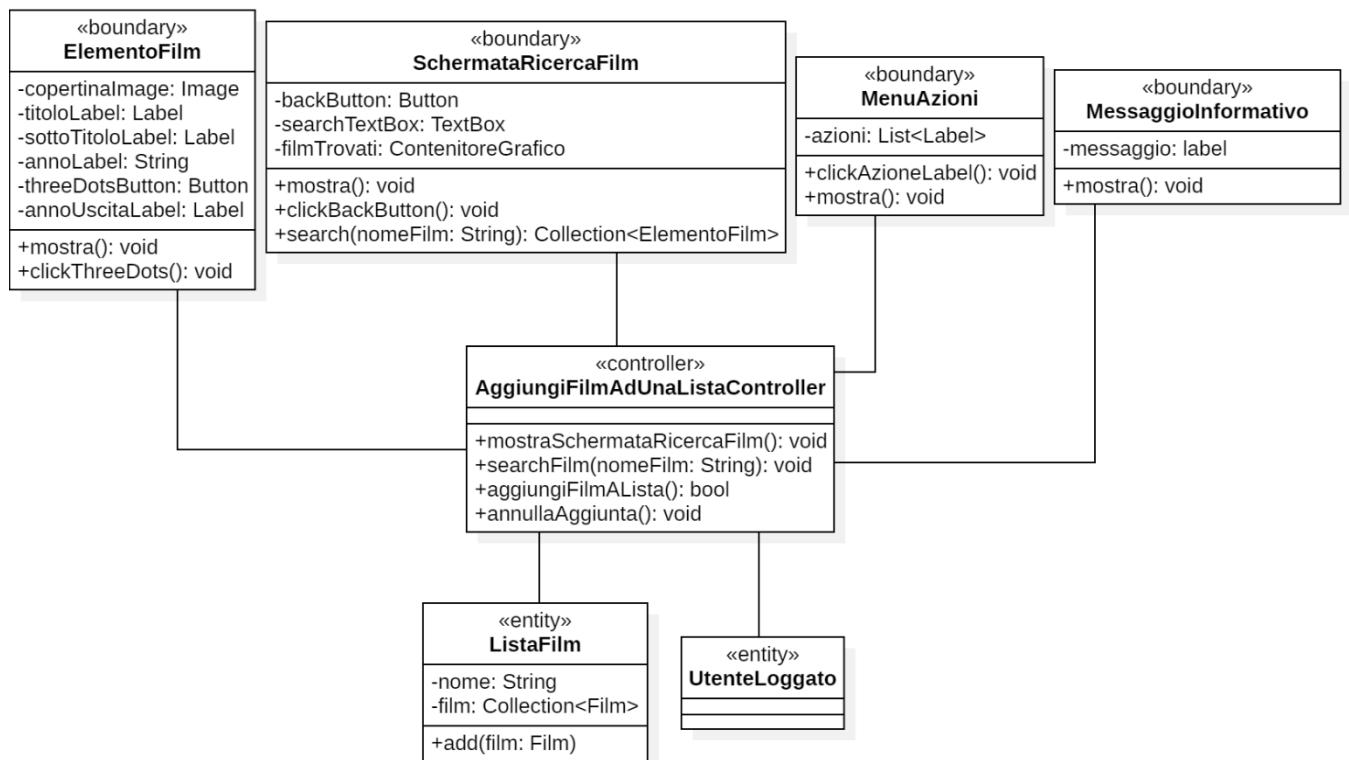
<b>EXTENSION #1:</b> Non sono presenti segnalazioni pendenti per la categoria scelta.	4.1		Mostra <b>D9</b> .
<b>EXTENSION #2:</b> L'amministratore loggato visualizza le segnalazioni pendenti riguardanti gli utenti.	3.2	Clicca sul tasto “Utenti”	
	4.2		Mostra <b>D3</b> .
	5.2	Clicca sull'utente in merito al quale desidera visualizzare le segnalazioni.	
	6.2		Mostra <b>D4</b> .
	7.2	Clicca su uno degli utenti che ha effettuato la segnalazione indirizzata all'utente selezionato allo step 5.2.	
	8.2		Mostra <b>D5</b> .
	9.2	Clicca sul tasto “Approva” o sul tasto “Rigetta”.	
	10.2		Notifica l'autore dell'esito della segnalazione.
<b>EXTENSION #3:</b> L'amministratore loggato approva la segnalazione del film e lo oscura.	9.3	Clicca sul tasto “Oscura”.	
	10.3		Oscura il film selezionato nello scenario principale allo step 5.
	11.3		Risolve a cascata tutte le segnalazioni pendenti degli utenti associate a quel film, memorizzandole come “Oscurata”.
	12.3		Notifica tutti gli utenti coinvolti riguardo l'esito della segnalazione.
	13.3		Mostra <b>D6</b> privato del film scelto nello scenario principale allo step 5.

# Modelli di dominio

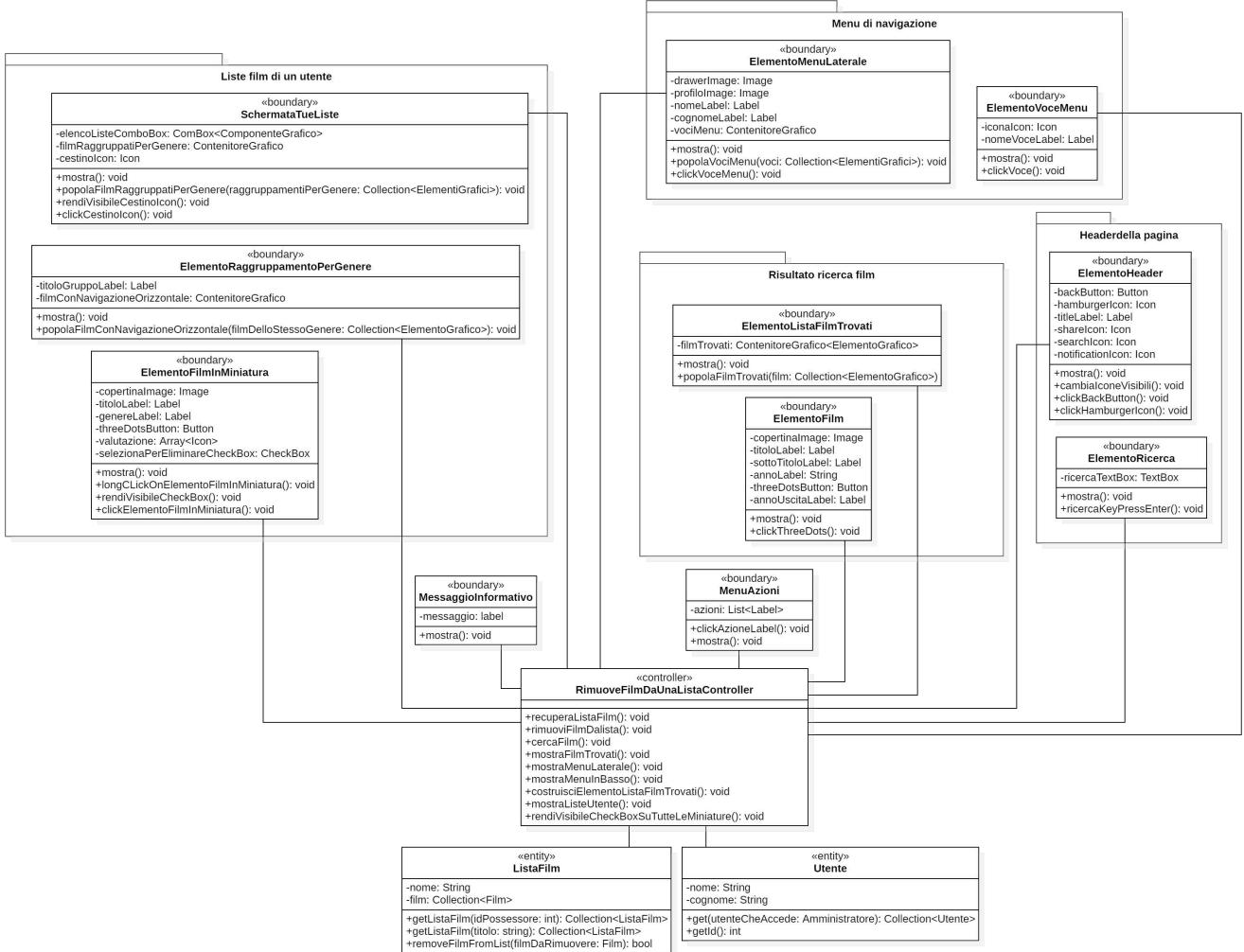
## Class Diagrams

### Mobile

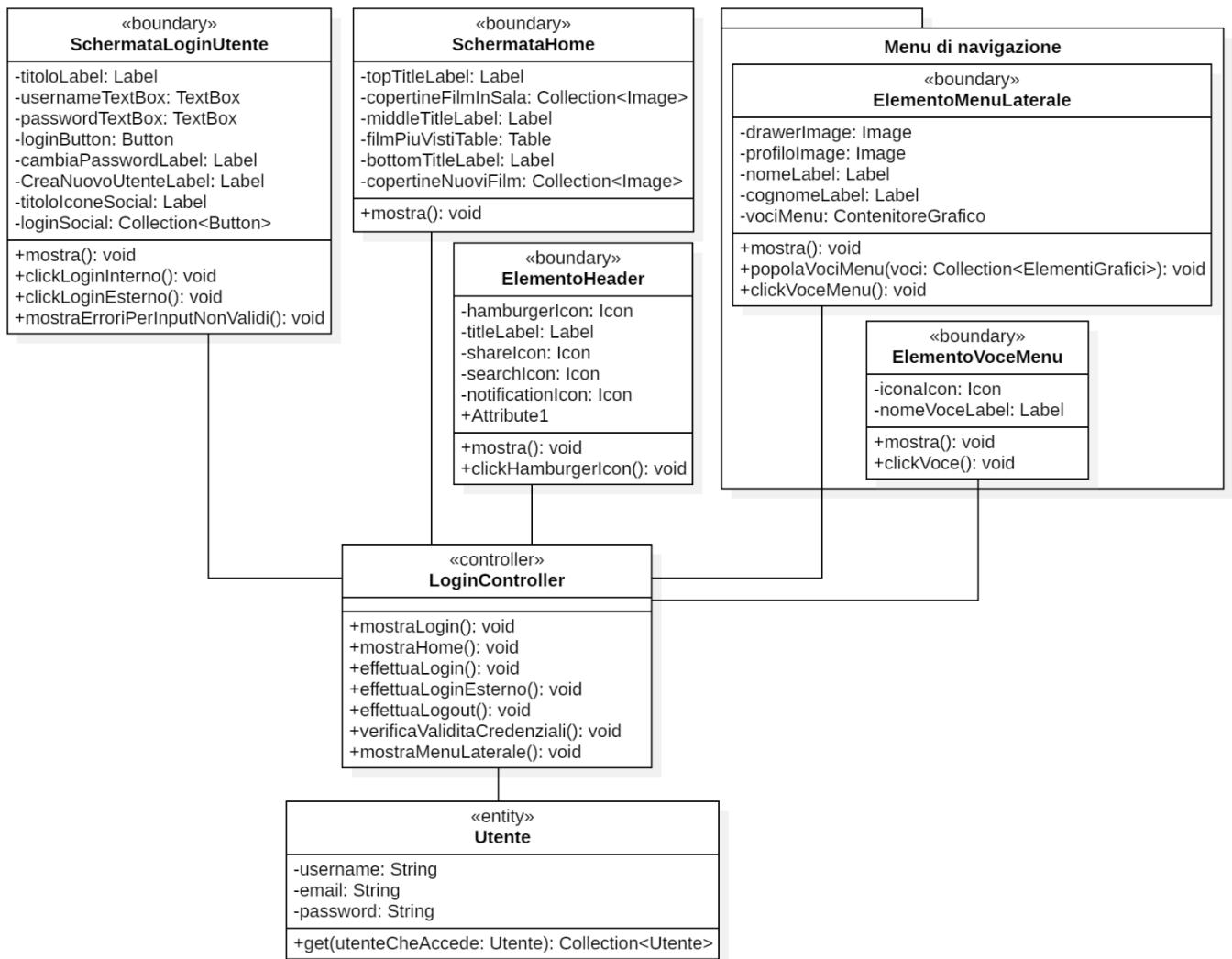
Utente loggato aggiunge film ad una lista



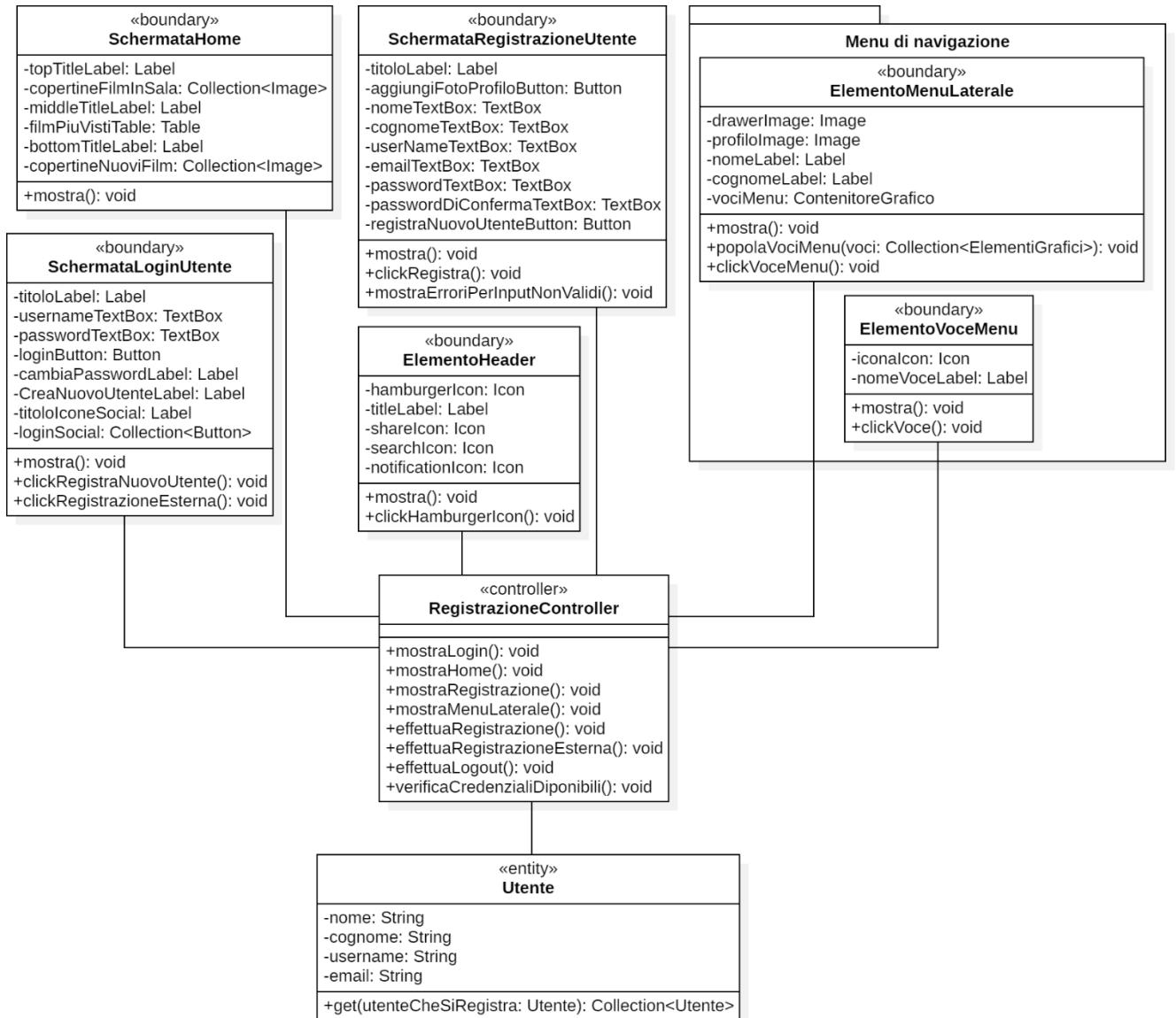
## Utente loggato rimuove film da una lista



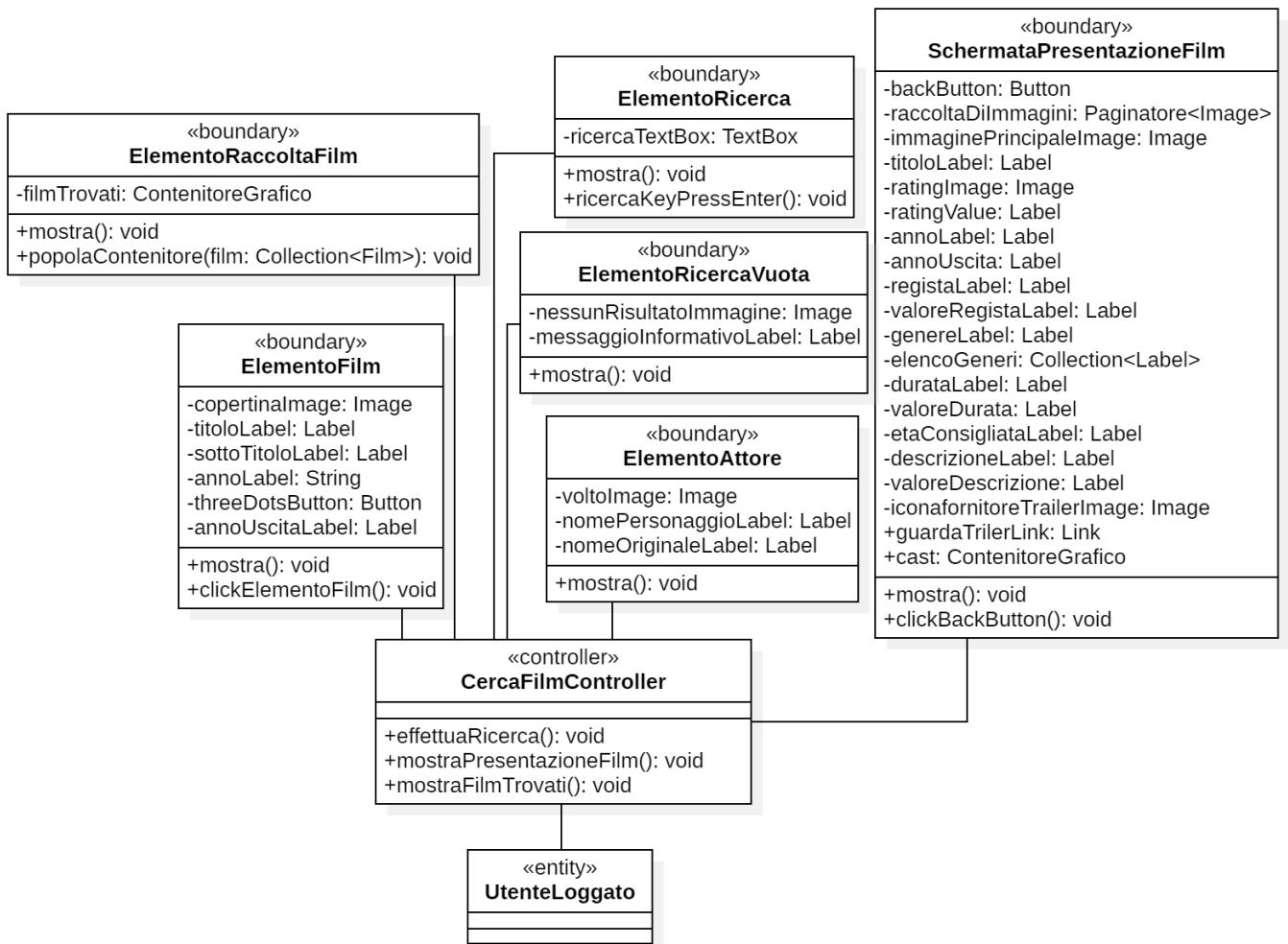
## Utente effettua login



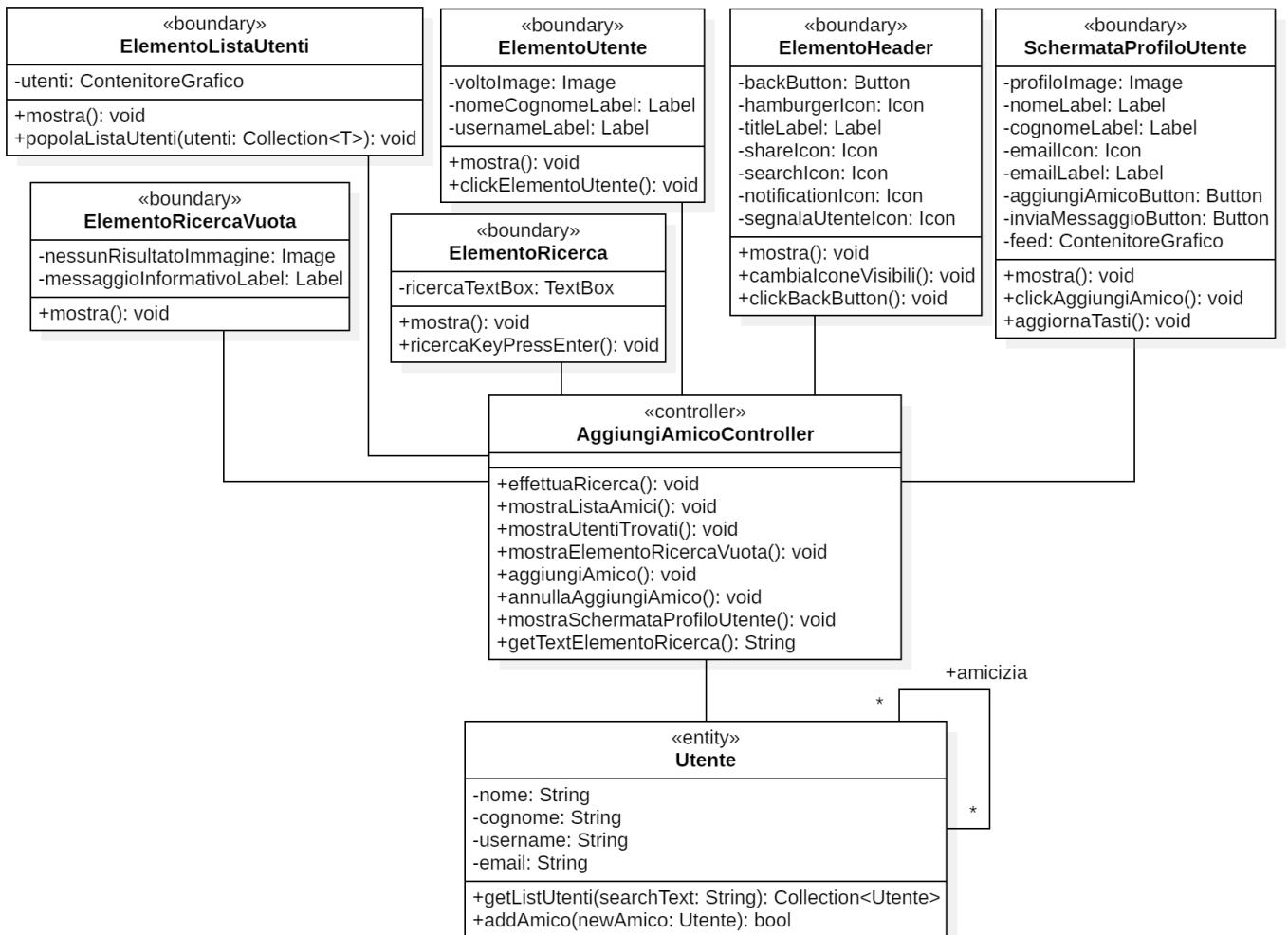
## Utente effettua registrazione



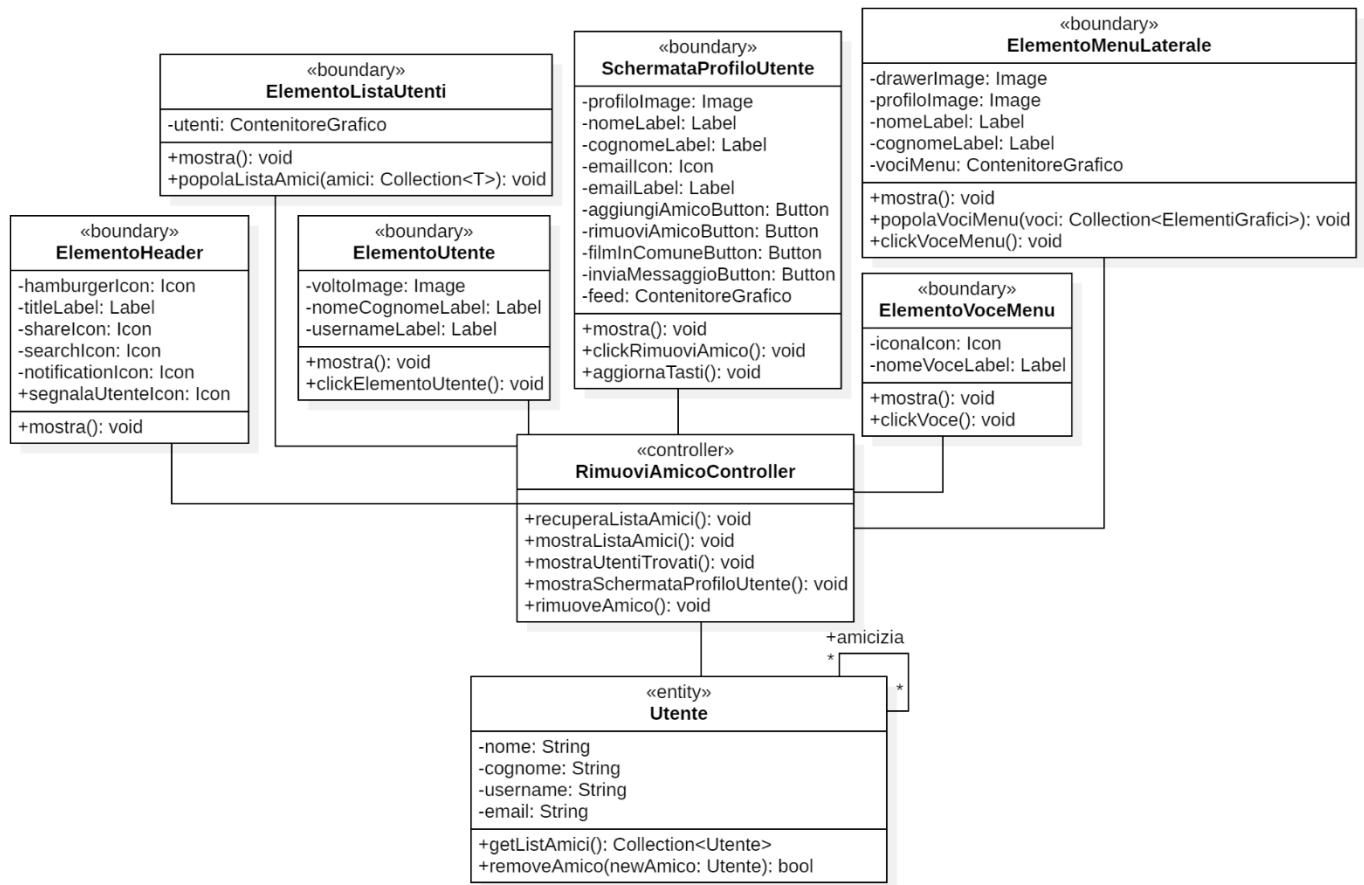
## Utente loggato cerca film



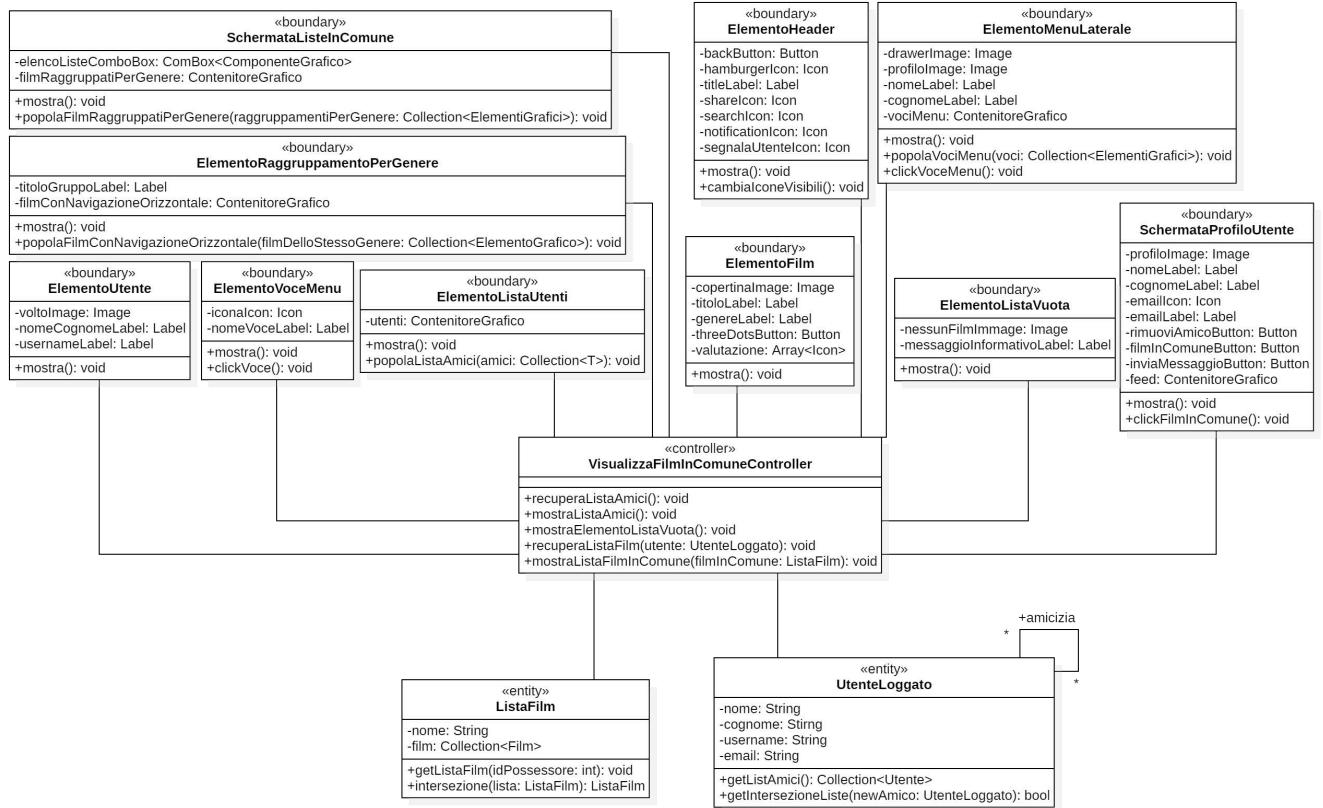
## Utente loggato invia richiesta di amicizia



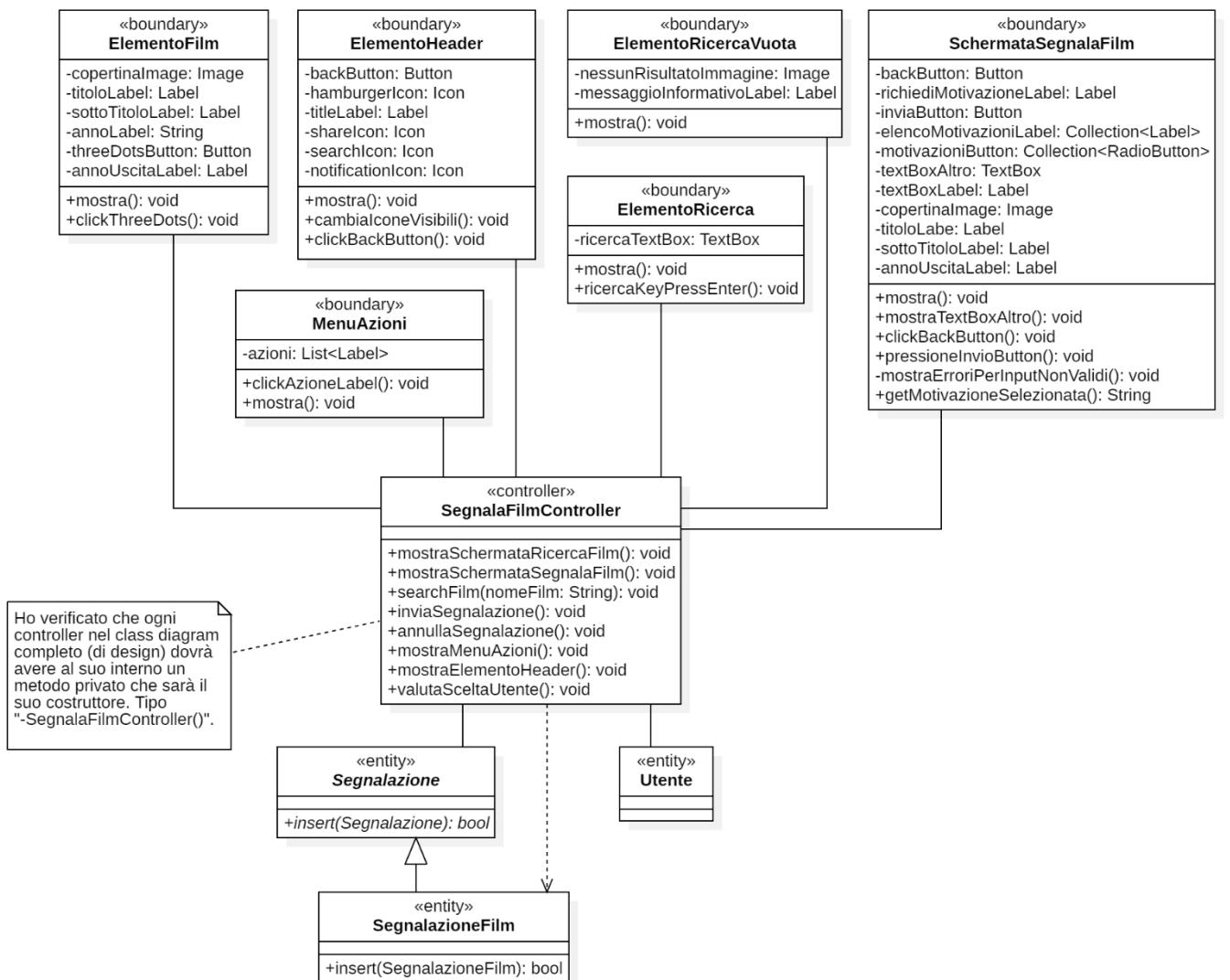
## Utente loggato rimuove amico



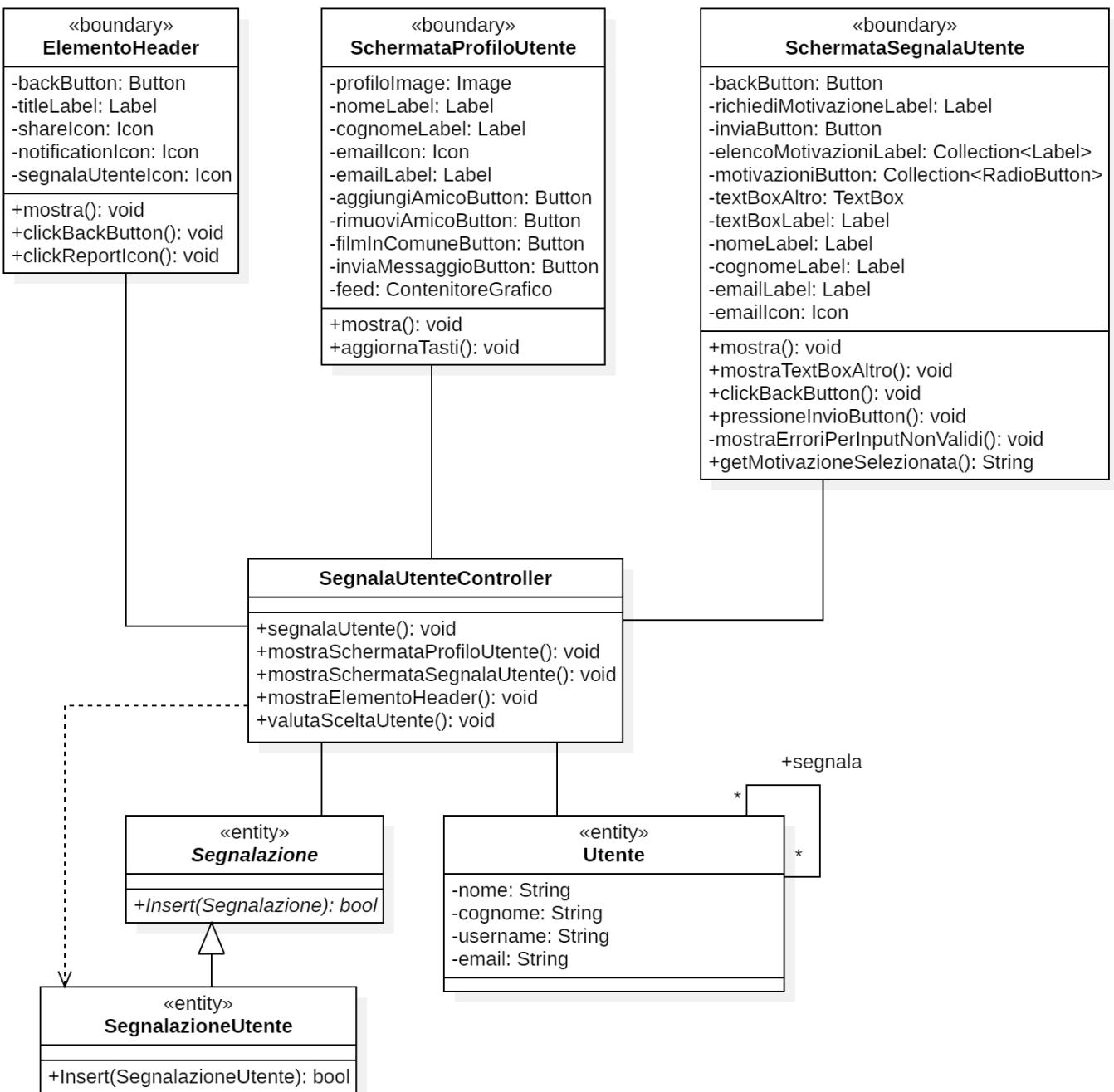
## Utente loggato visualizza film in comune



## Utente loggato segnala film

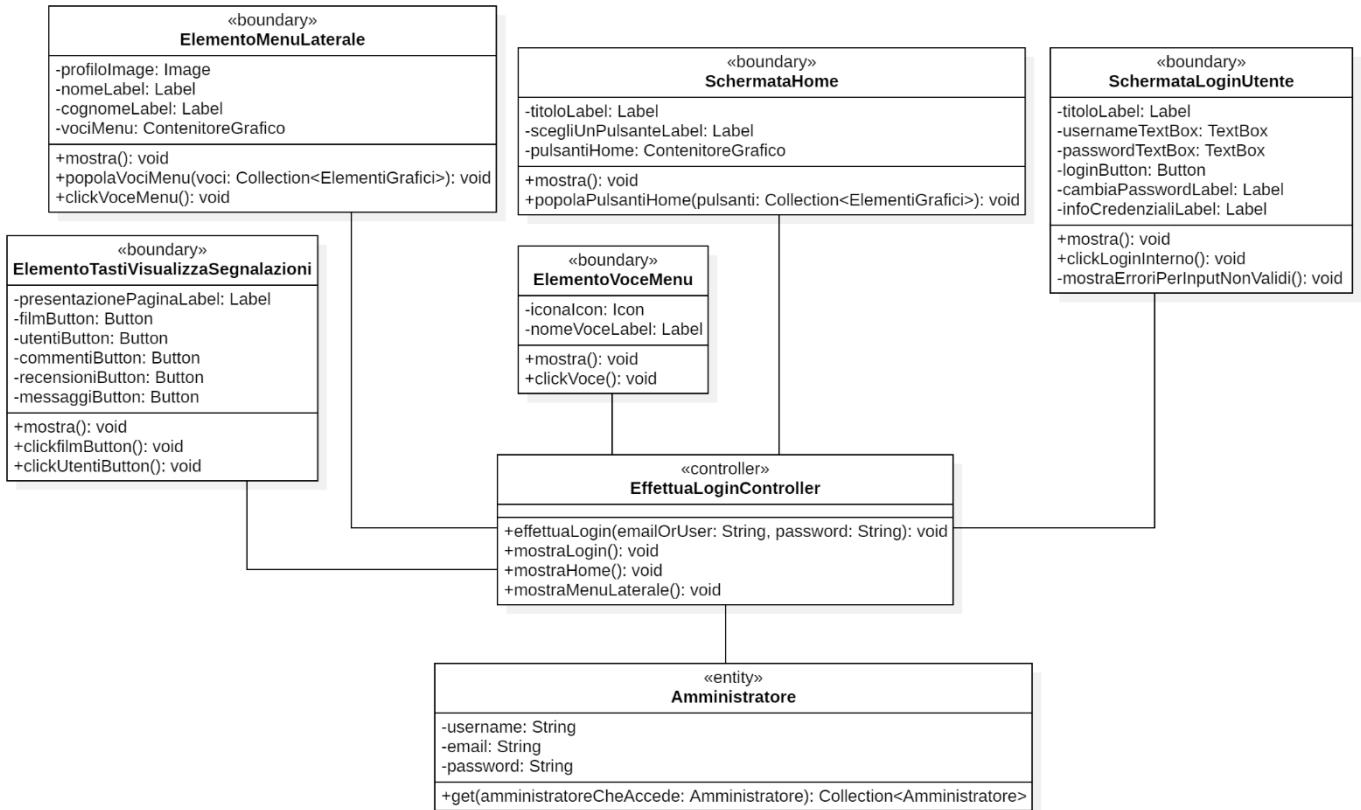


## Utente loggato segnala utente

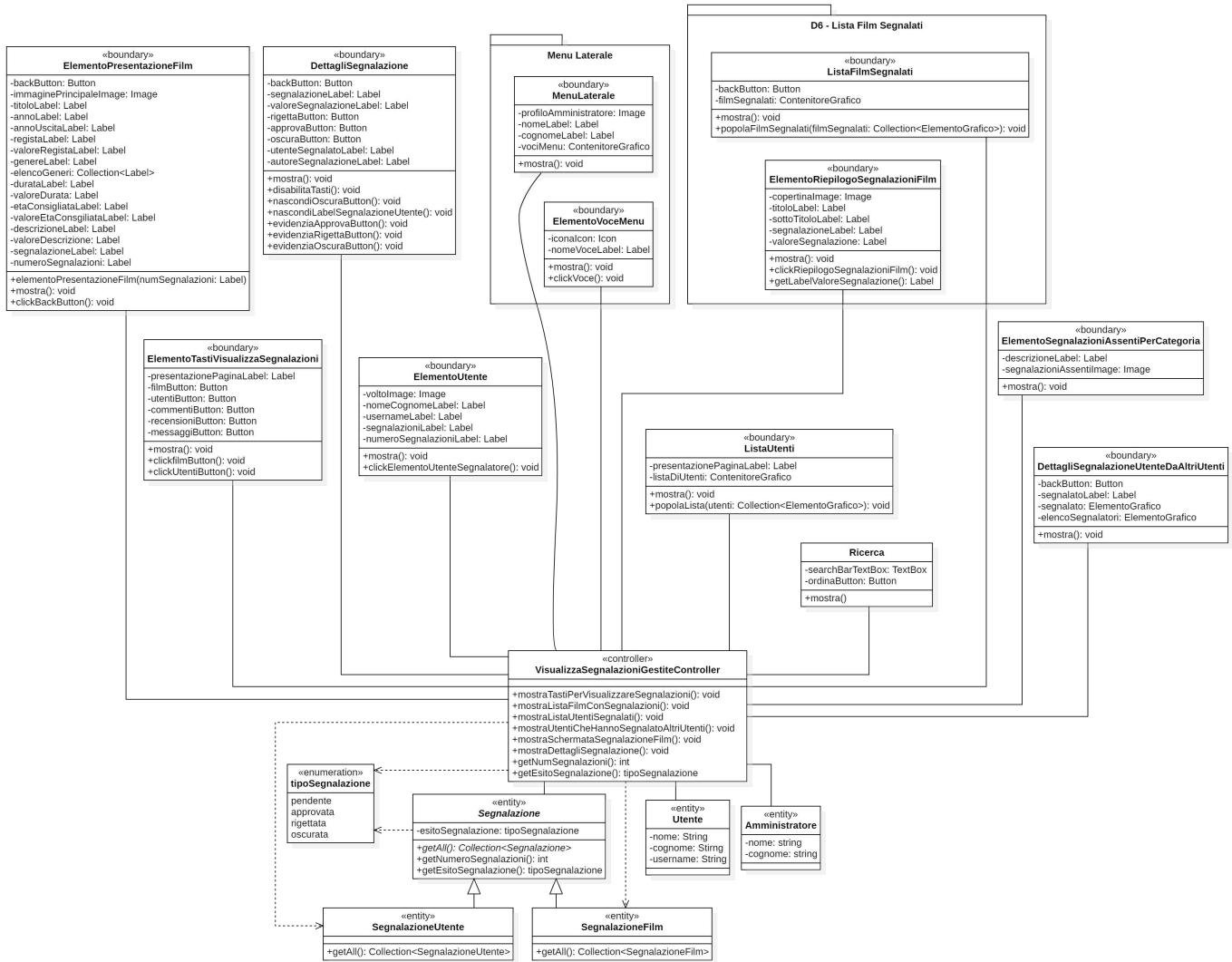


## Desktop

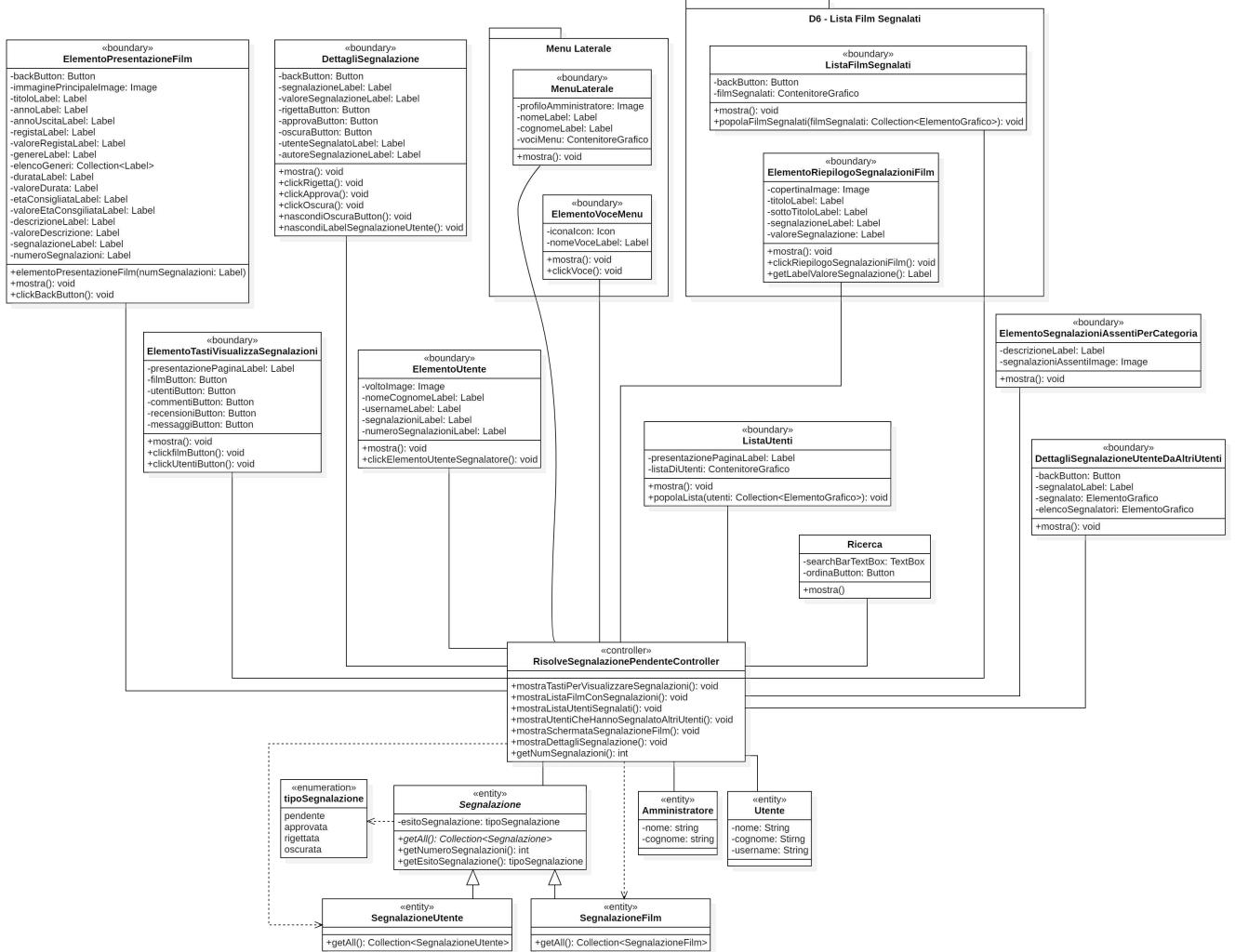
Amministratore effettua login



## Amministratore loggato visualizza segnalazioni gestite



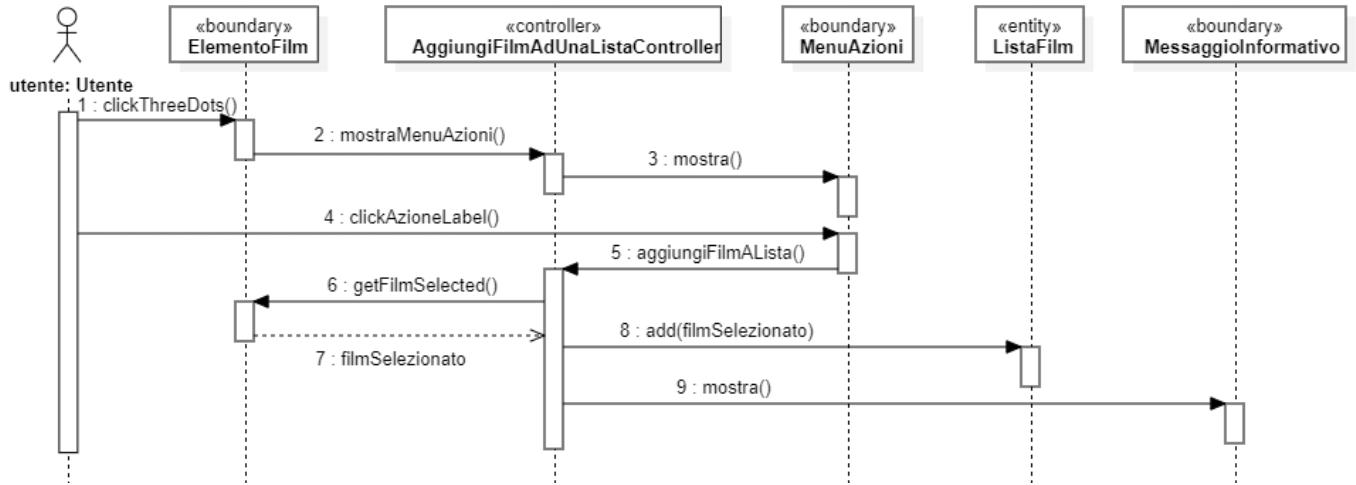
## Amministratore loggato risolve segnalazione pendente



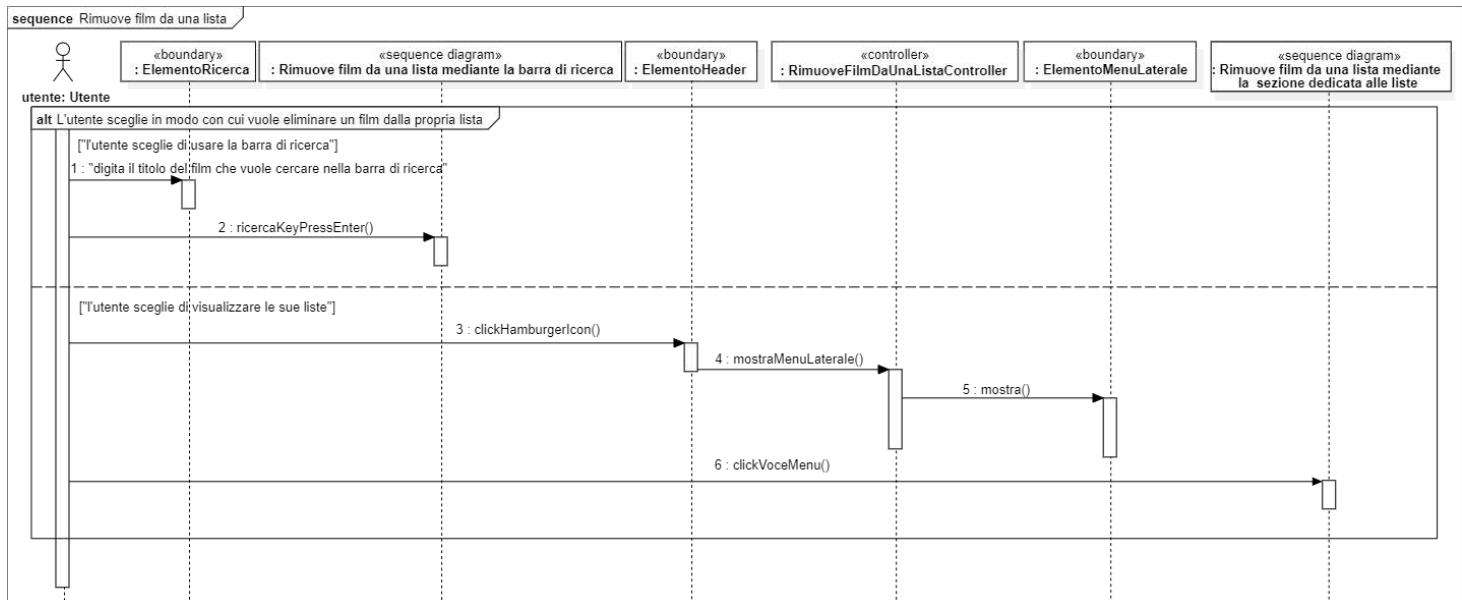
# Sequence Diagrams

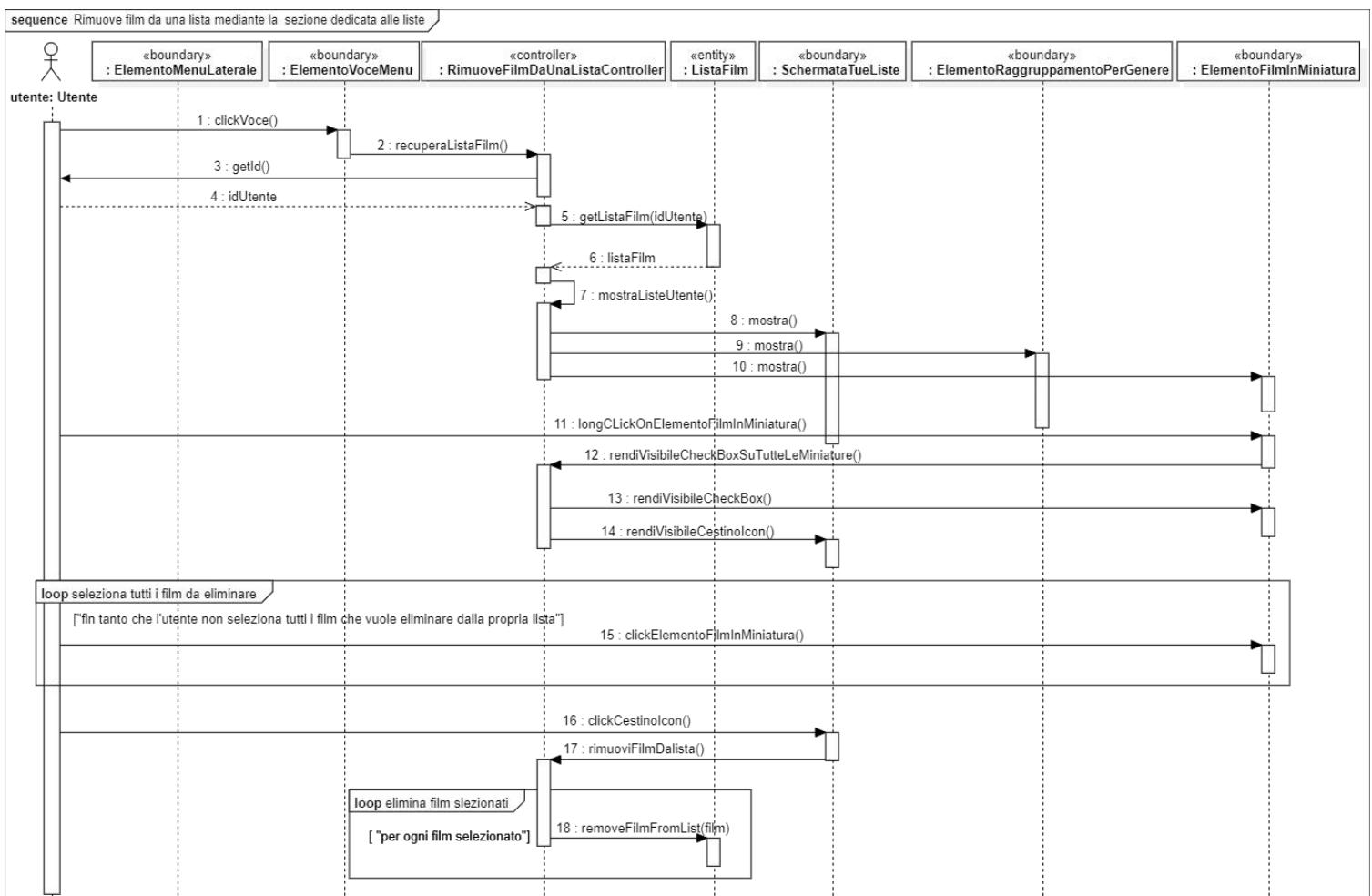
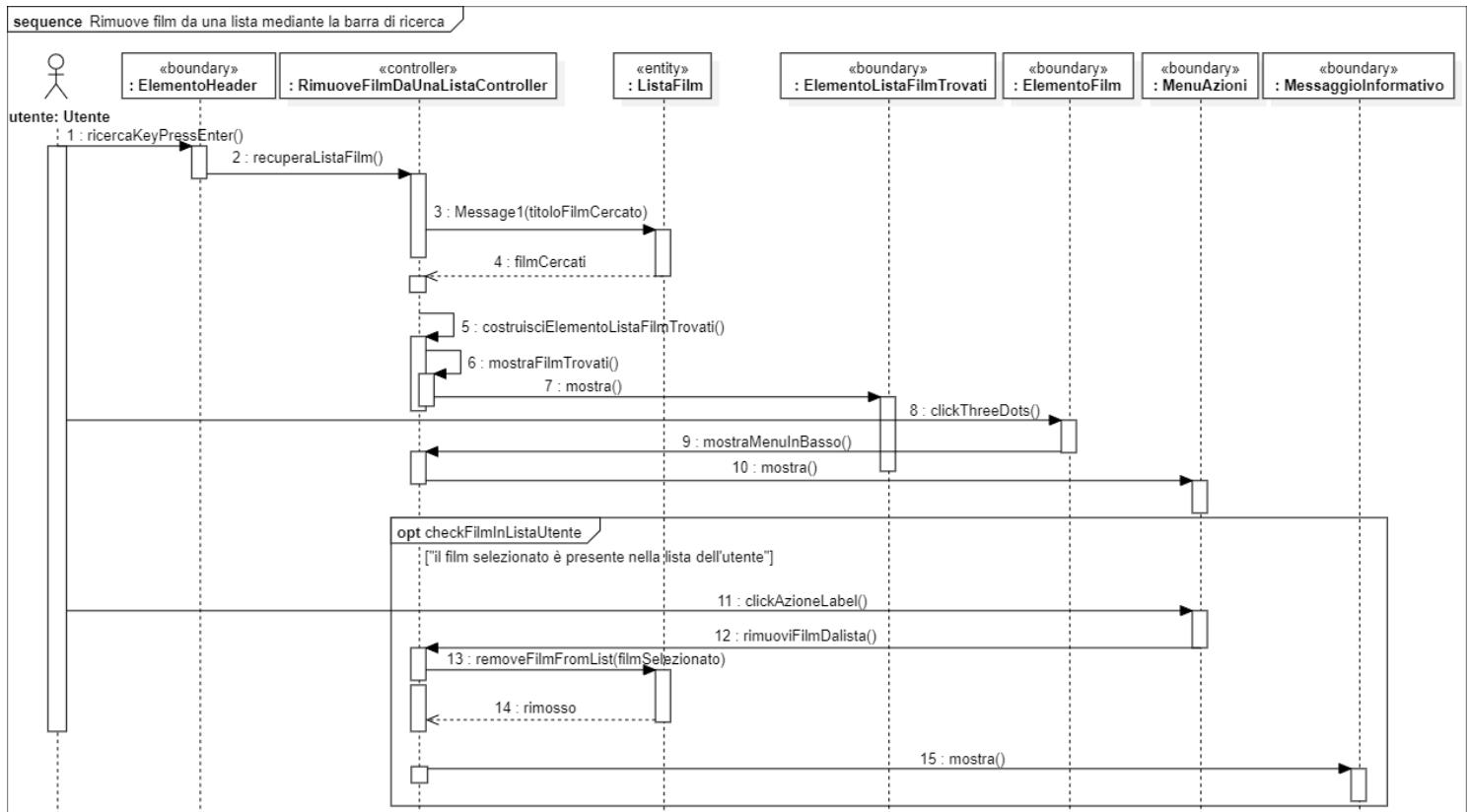
## Mobile

Utente loggato aggiunge film ad una lista

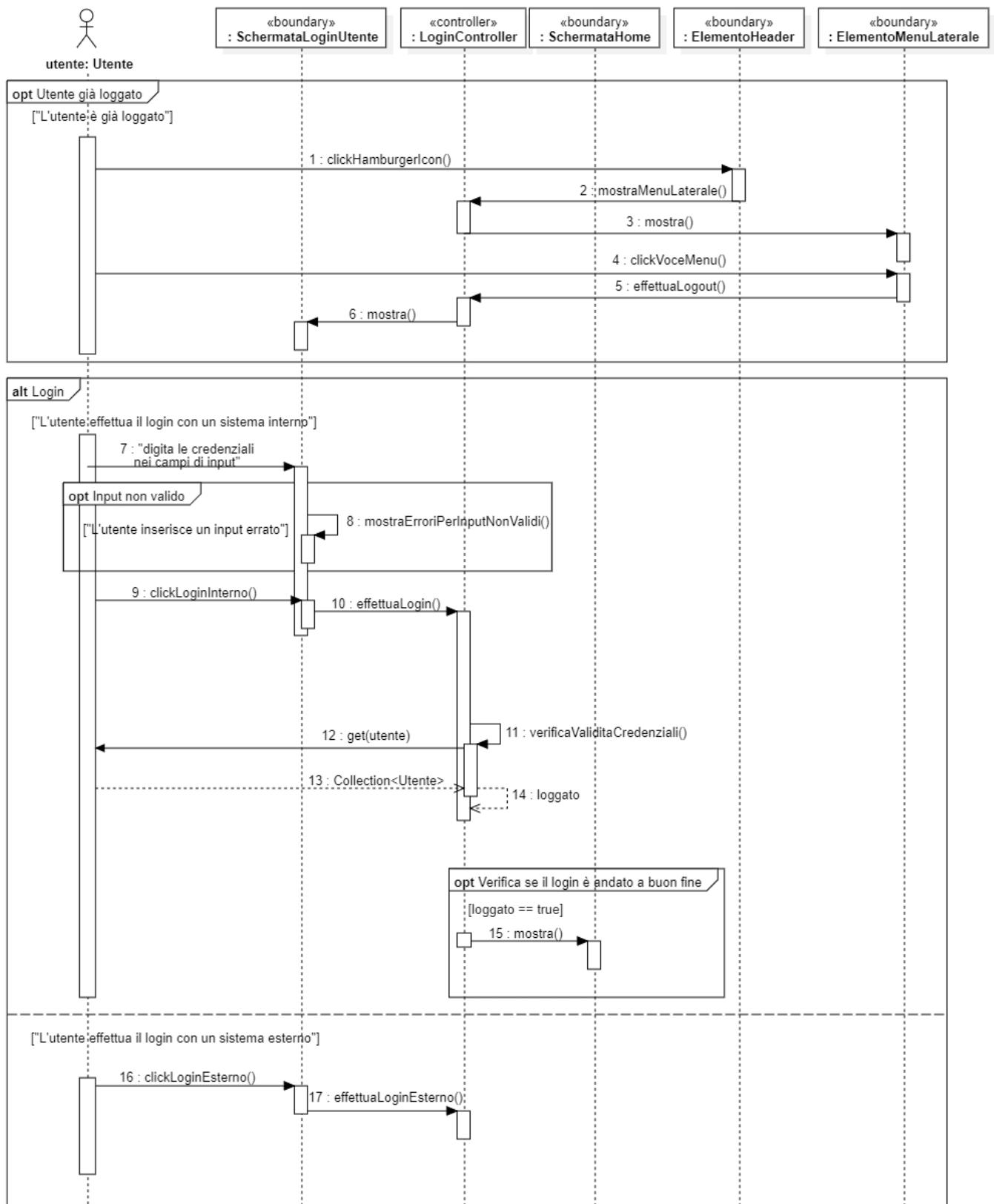


Utente loggato rimuove film da una lista

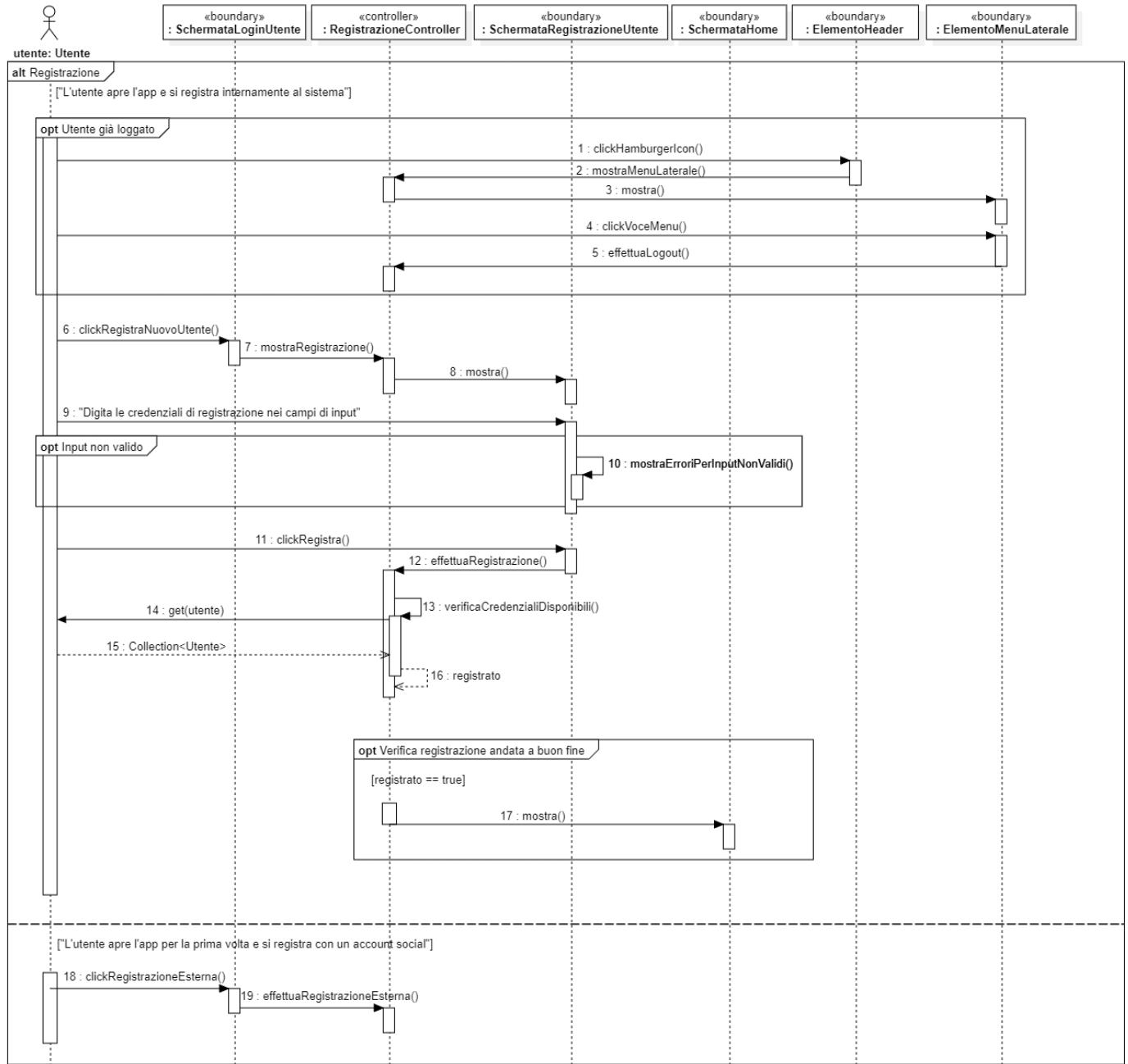




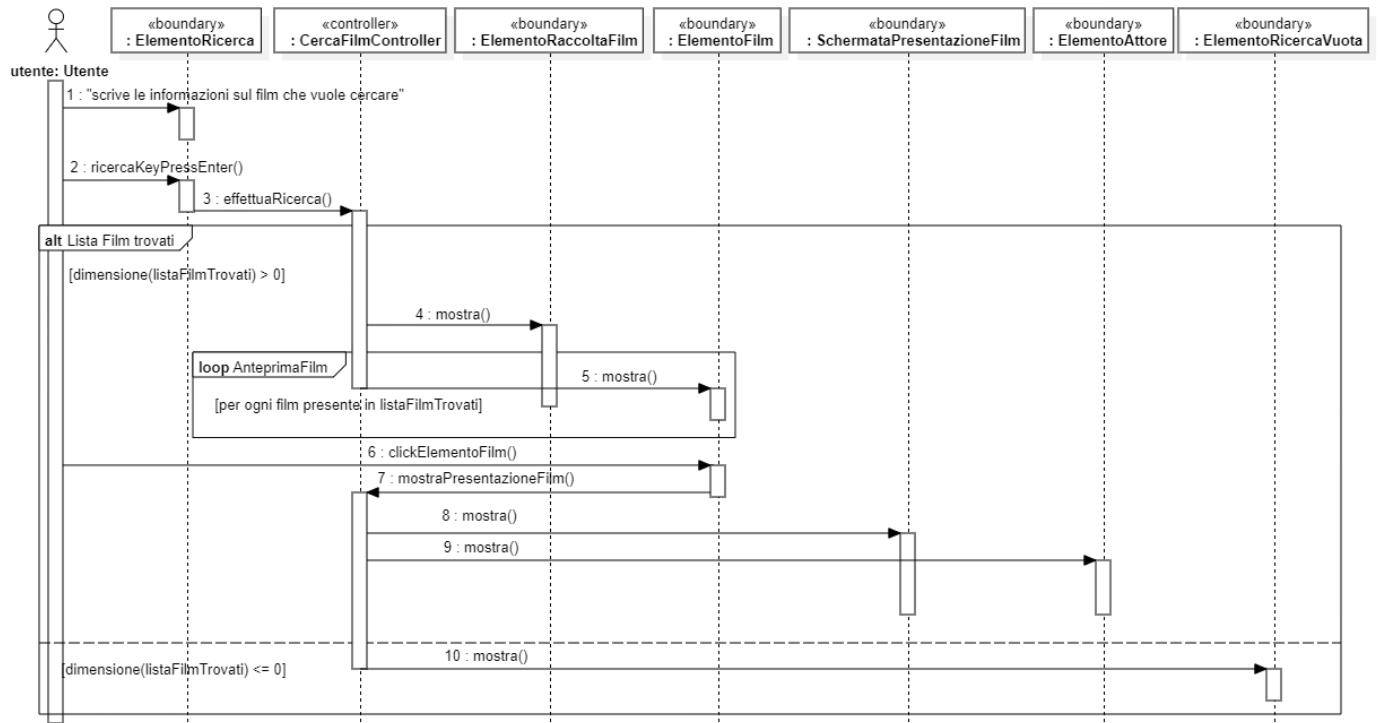
## Utente effettua login



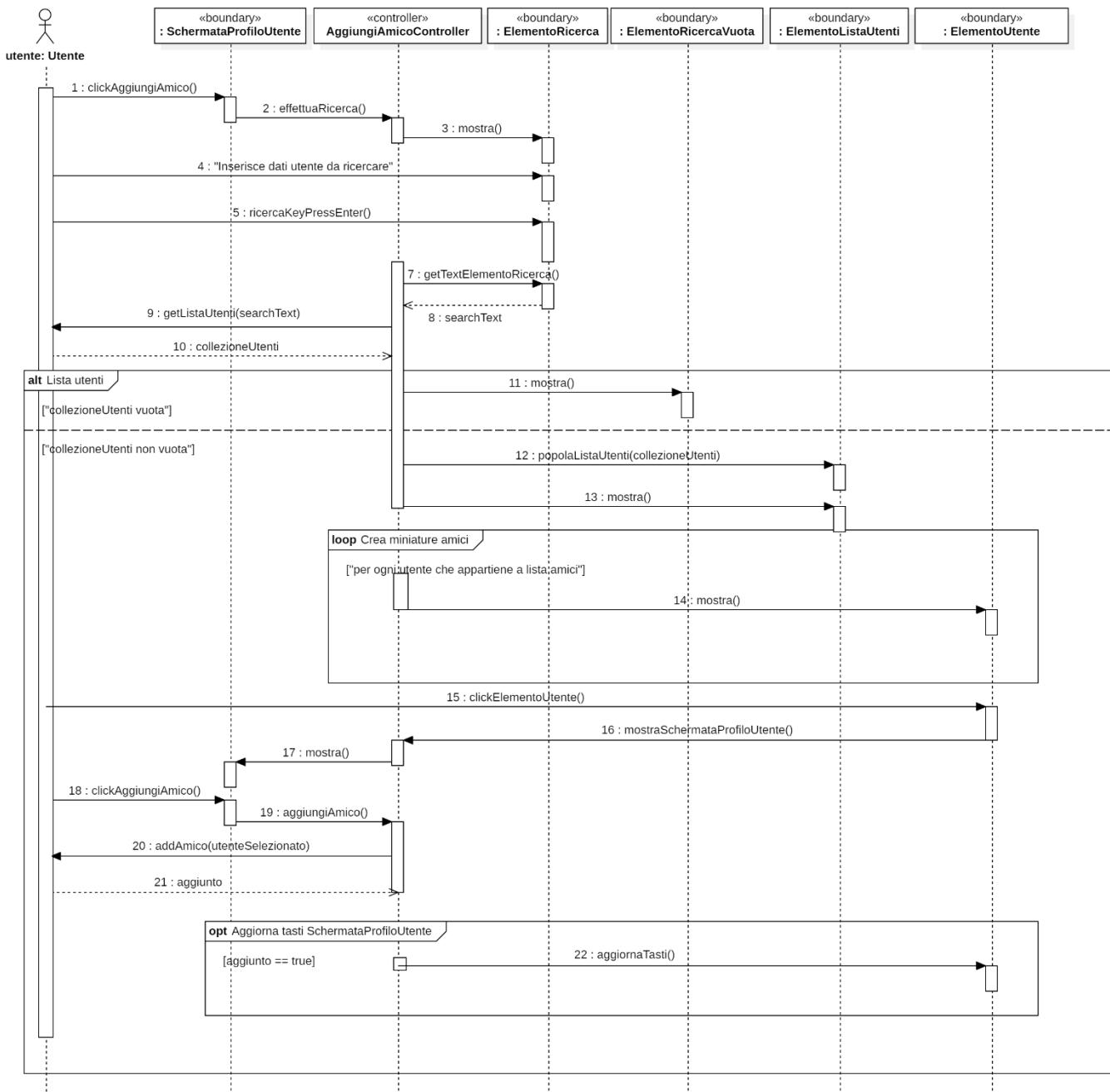
## Utente effettua registrazione



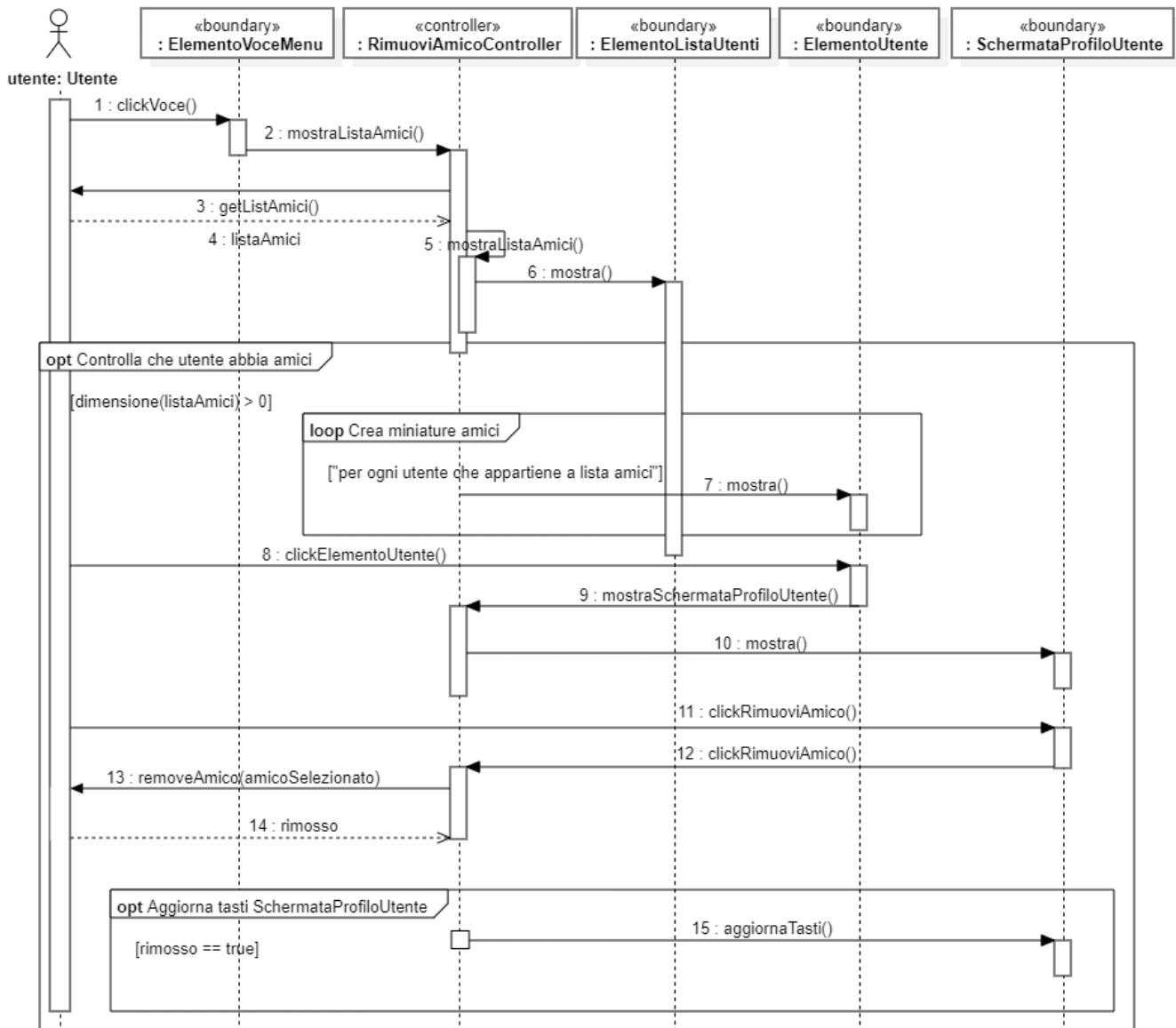
## Utente loggato cerca film



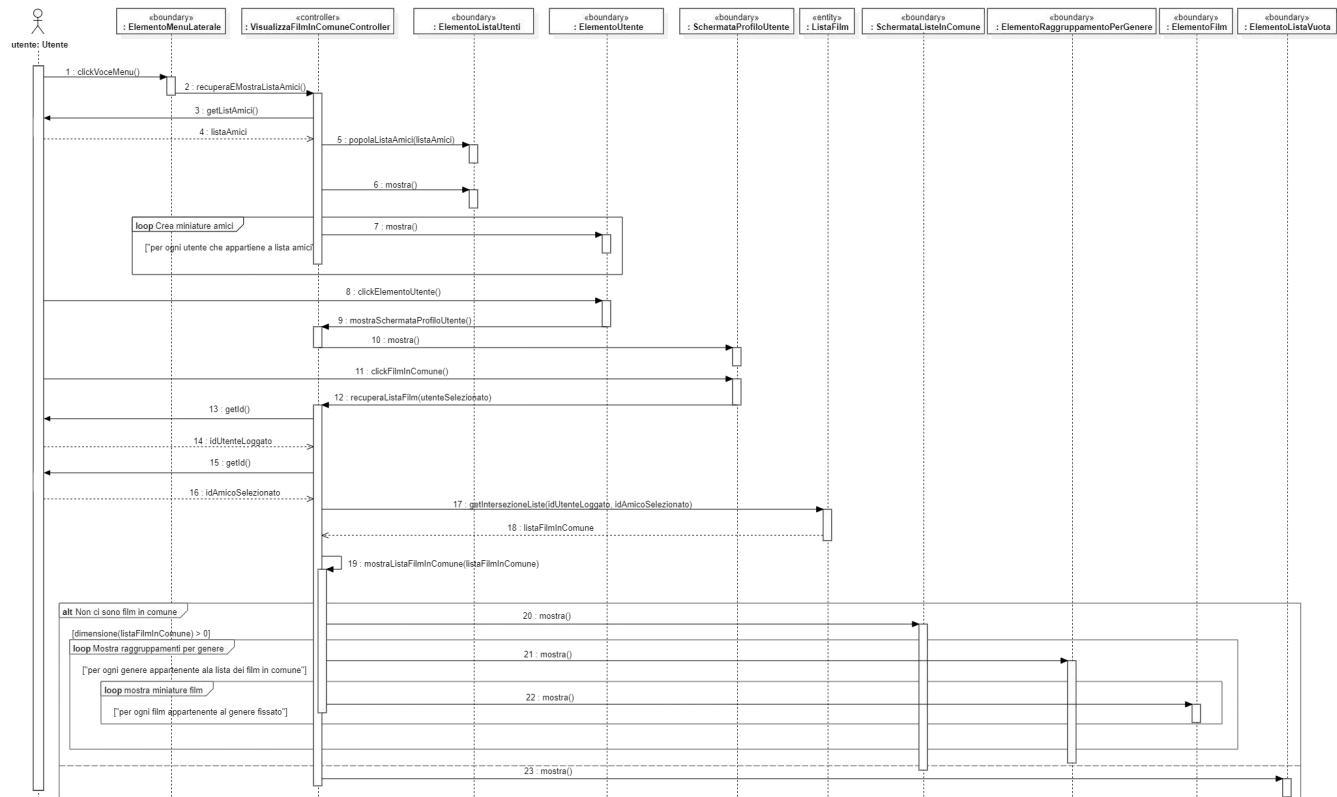
## Utente loggato invia richiesta di amicizia



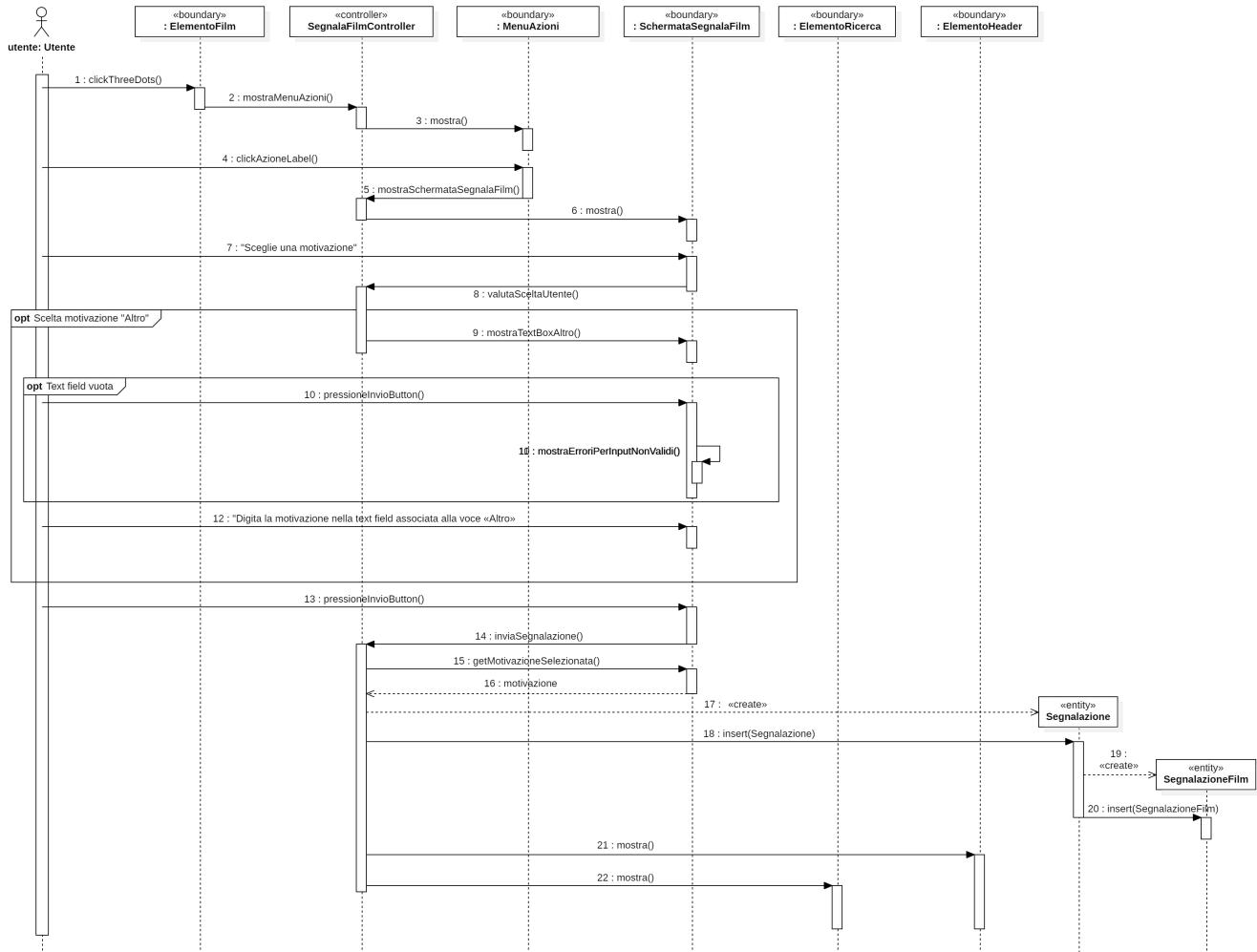
## Utente loggato rimuove amico



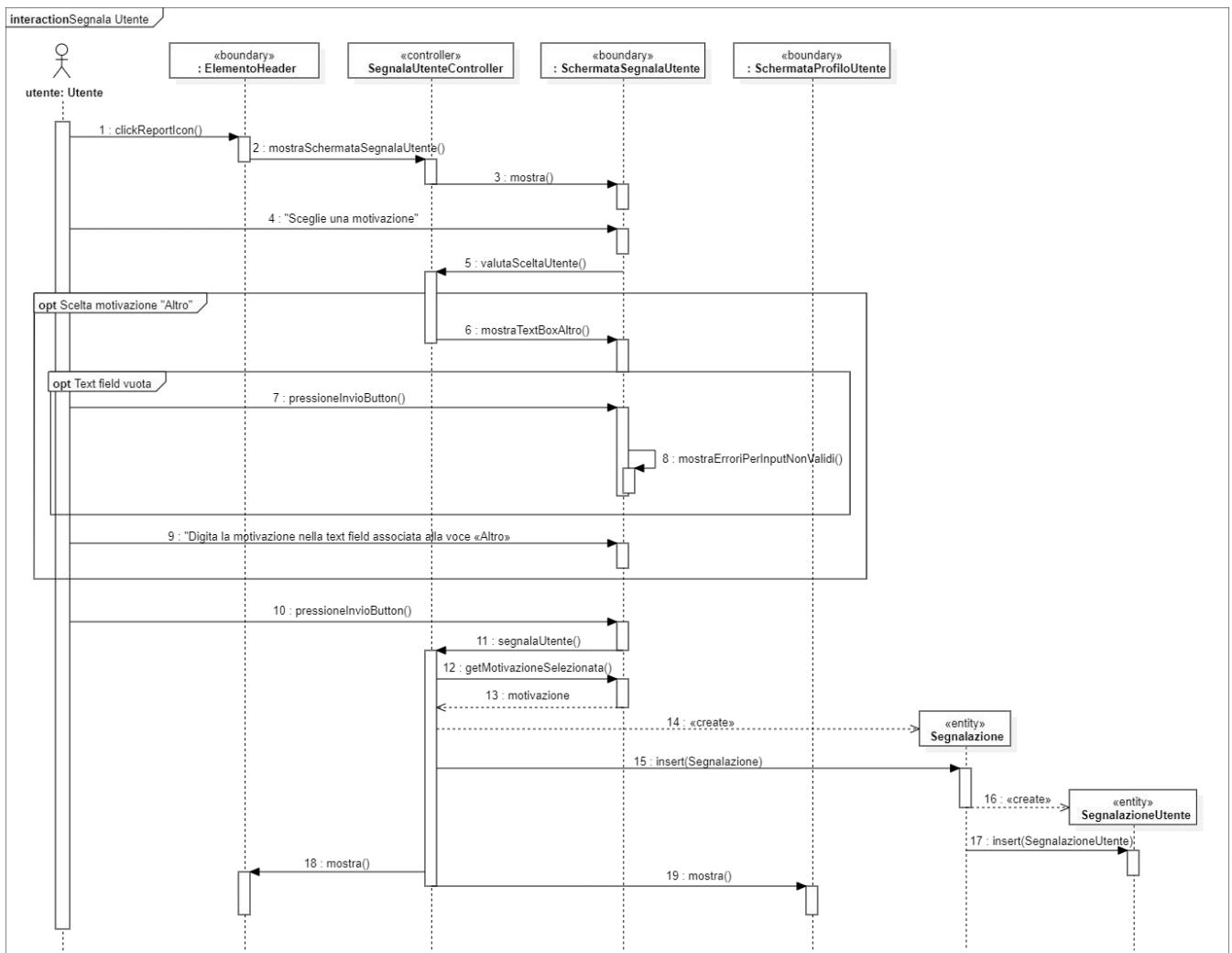
## Utente loggato visualizza film in comune



## Utente loggato segnala film

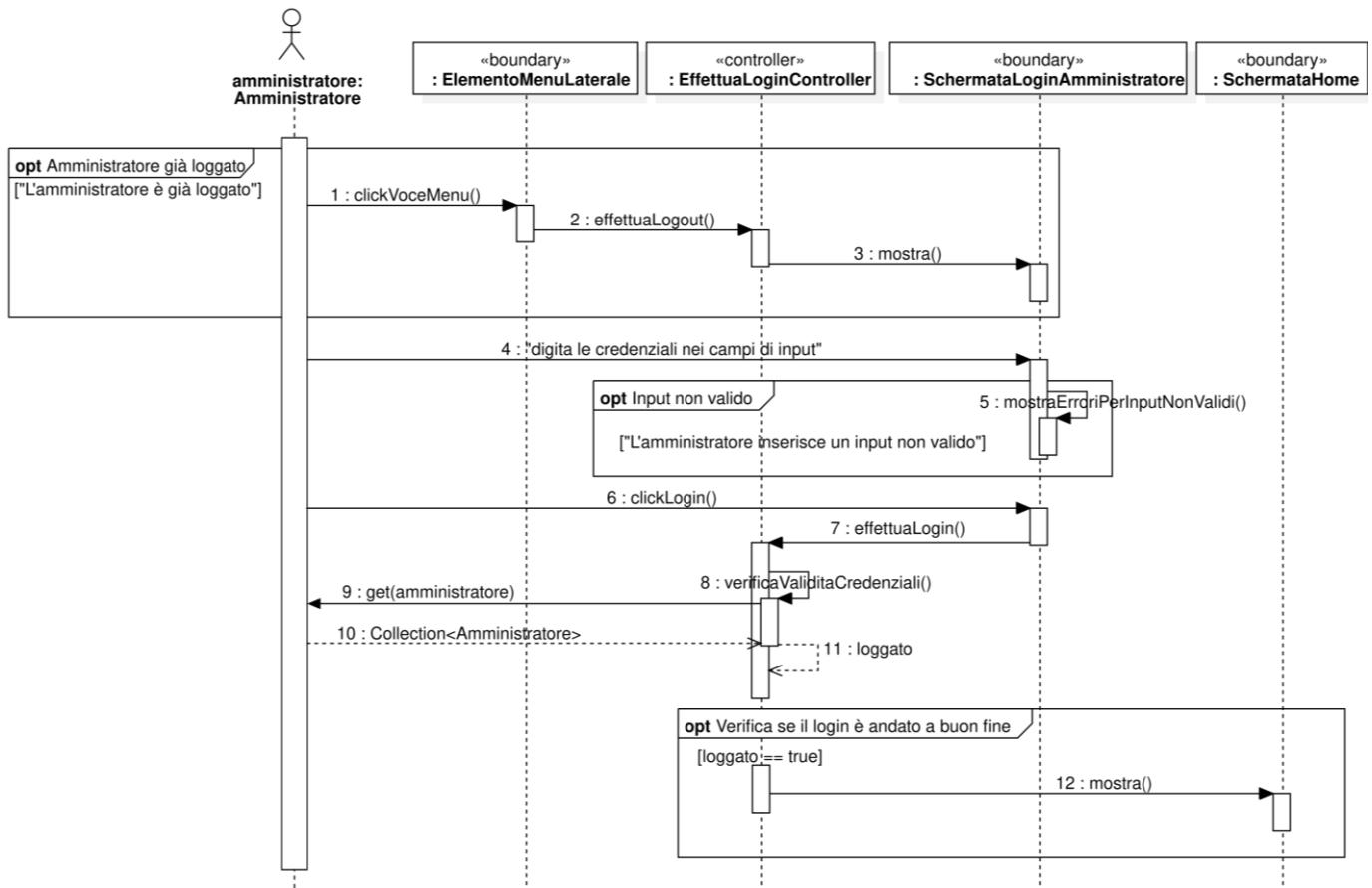


## Utente loggato segnala utente

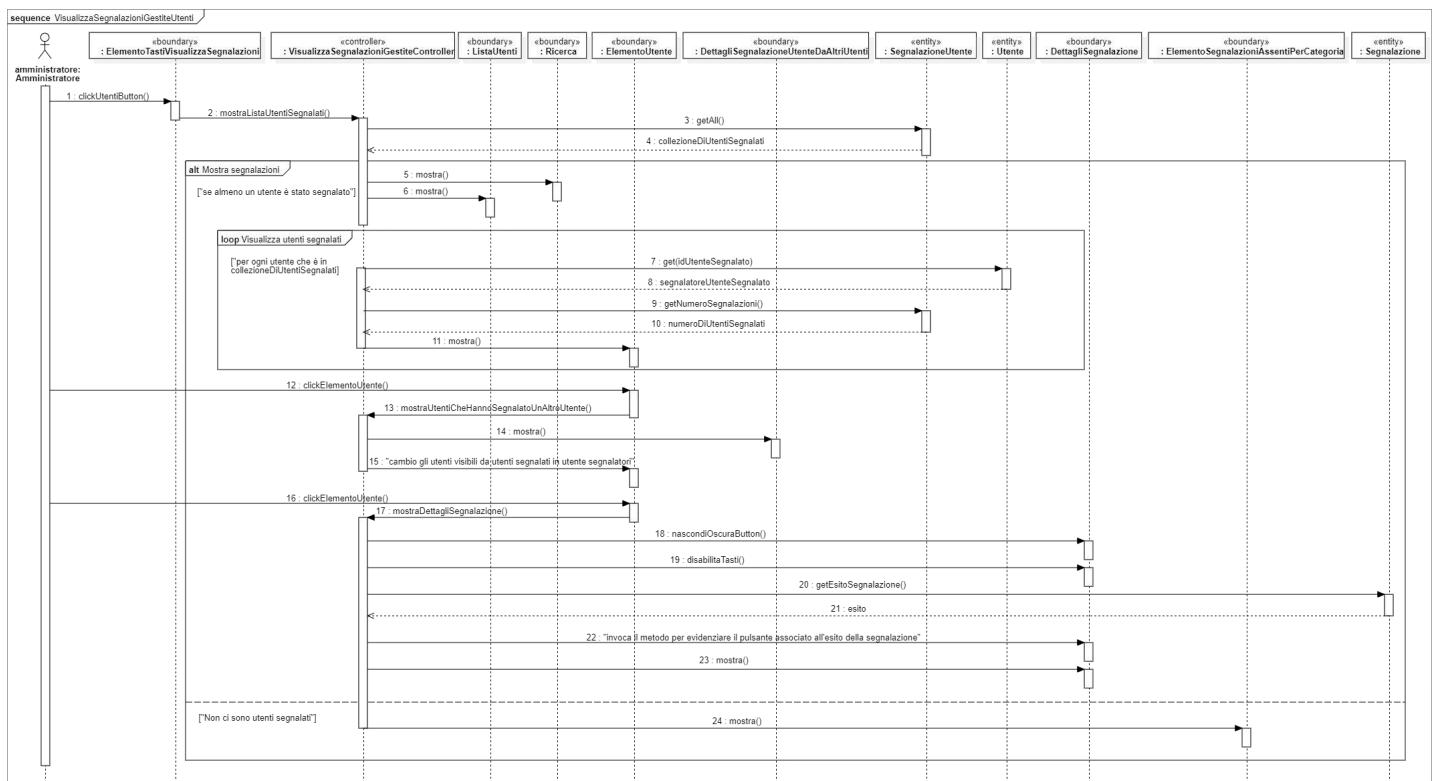
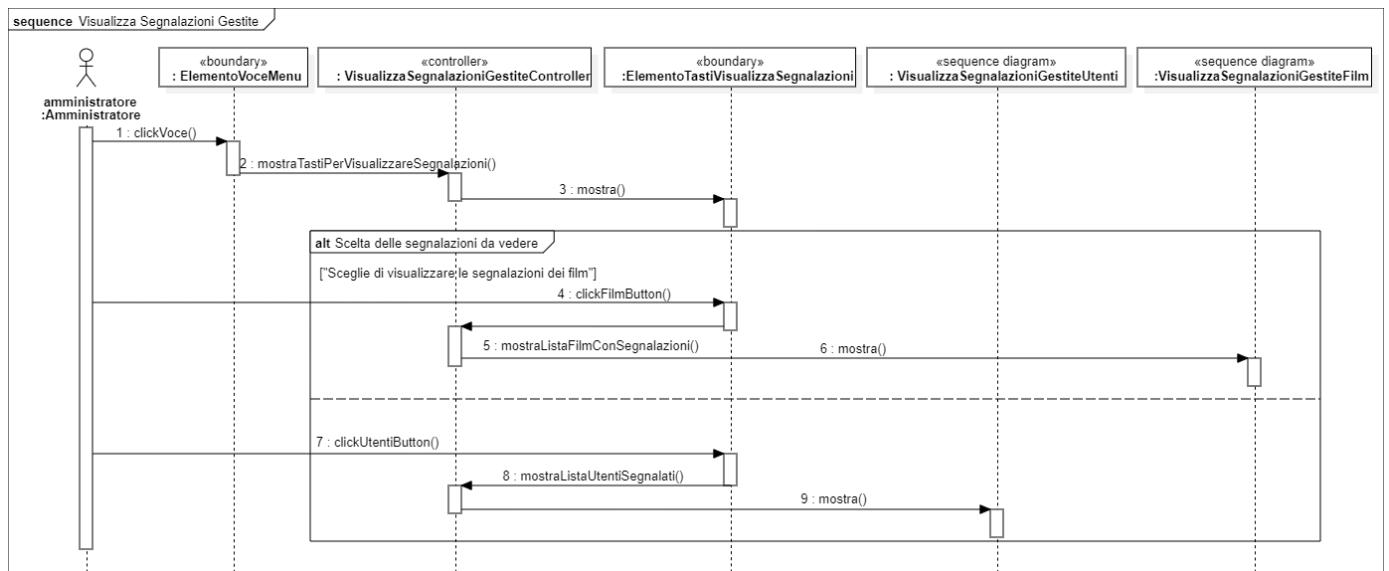


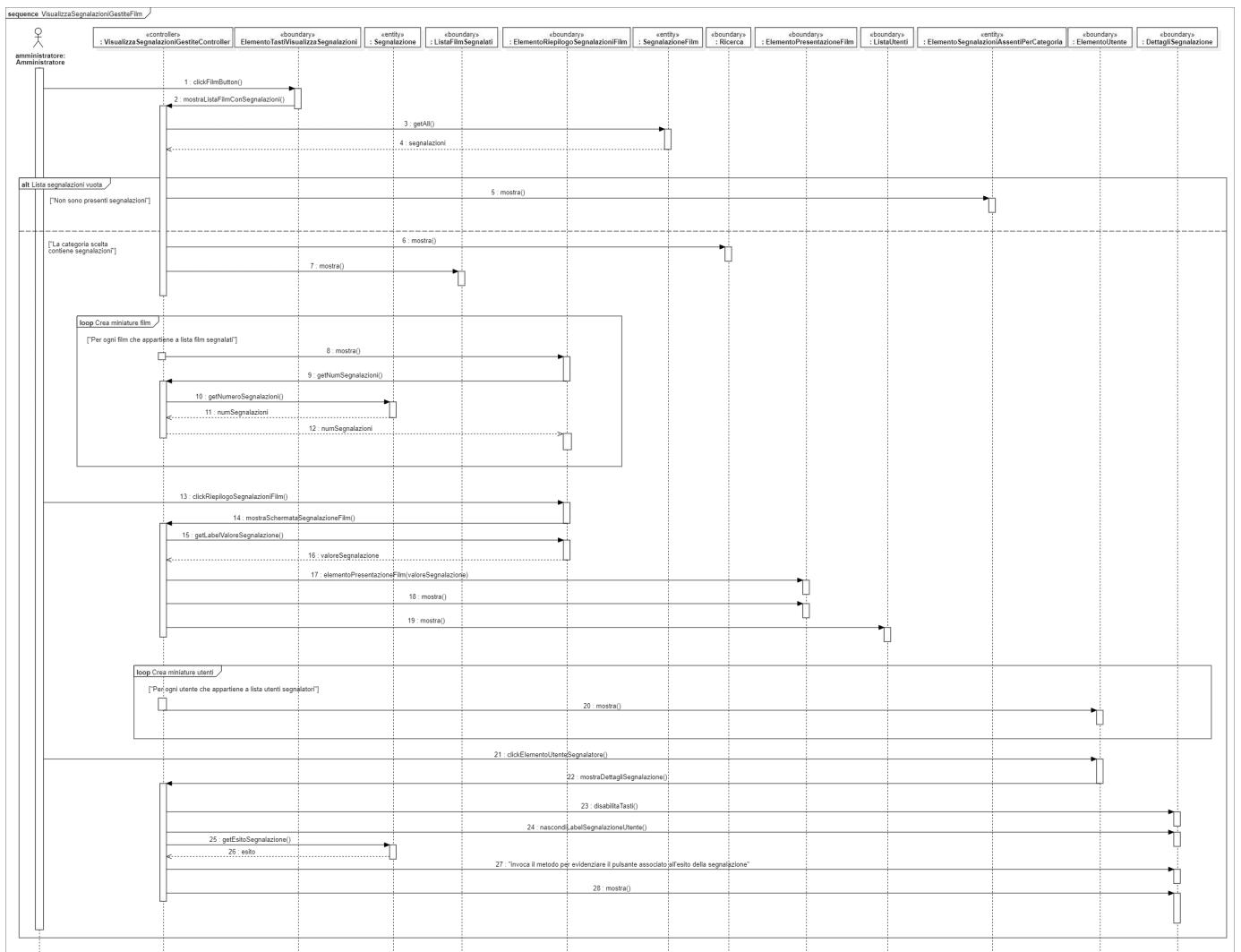
## Desktop

Amministratore effettua login

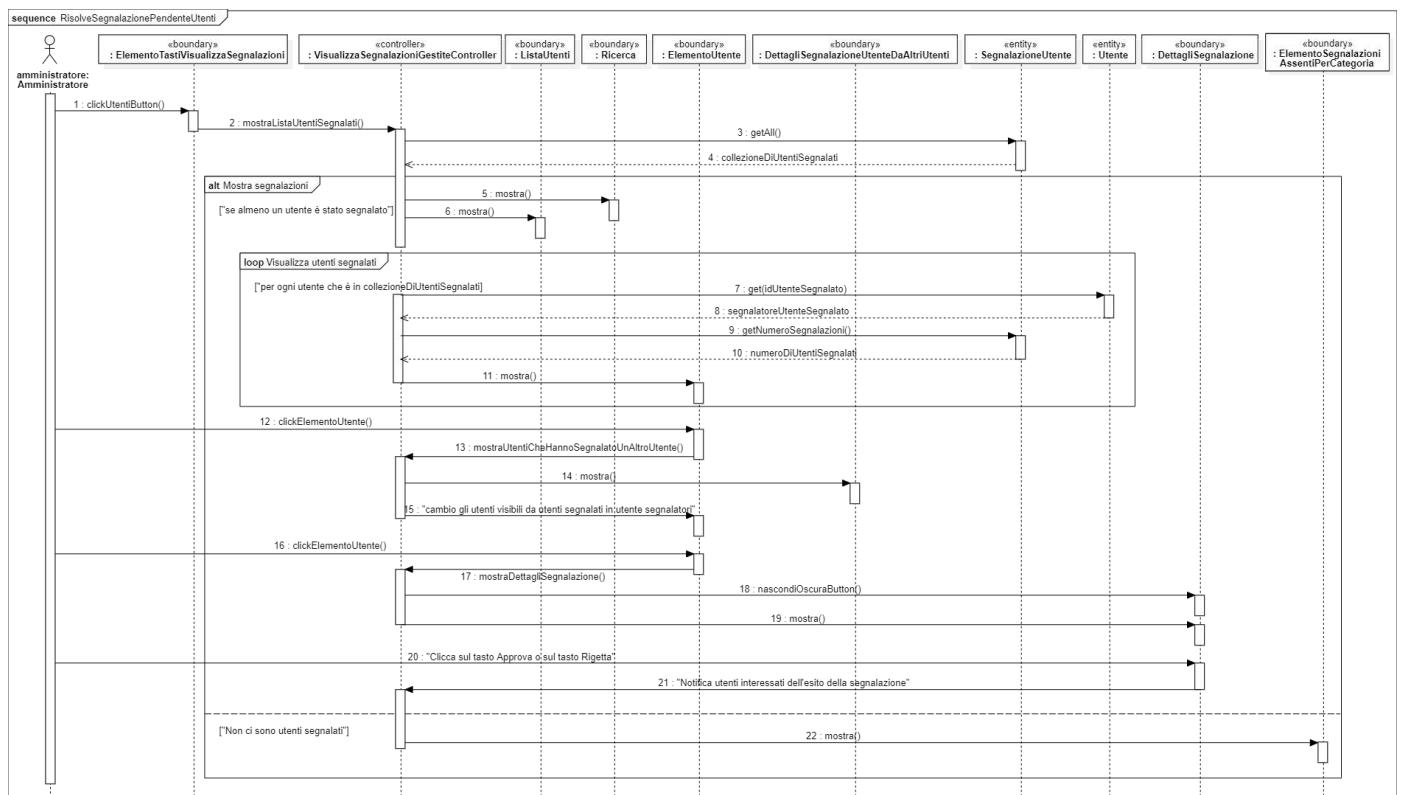
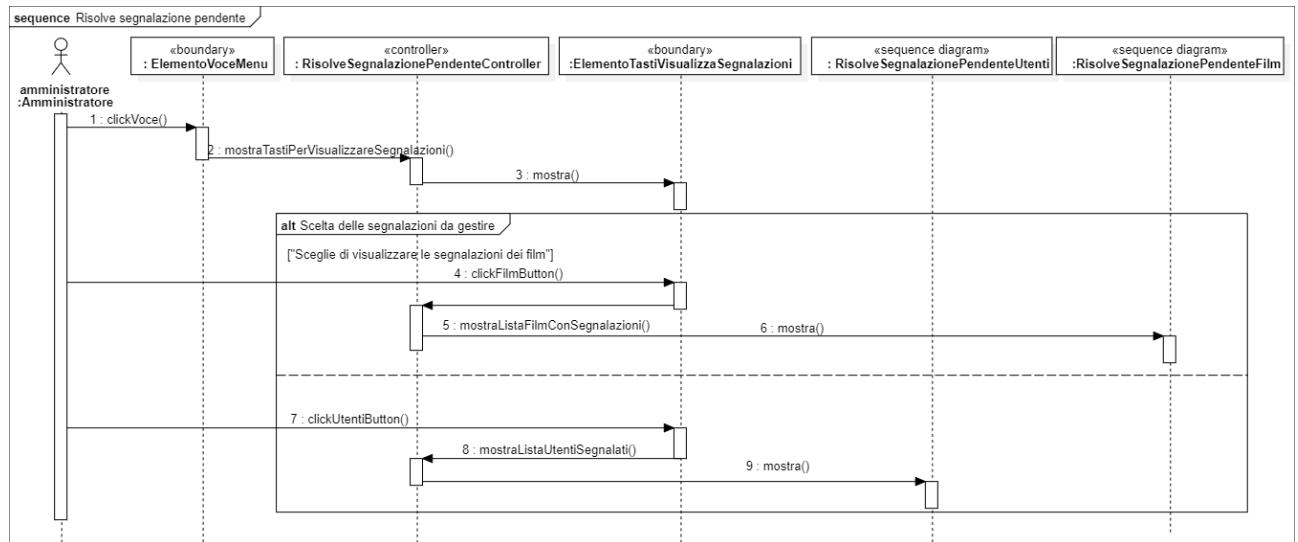


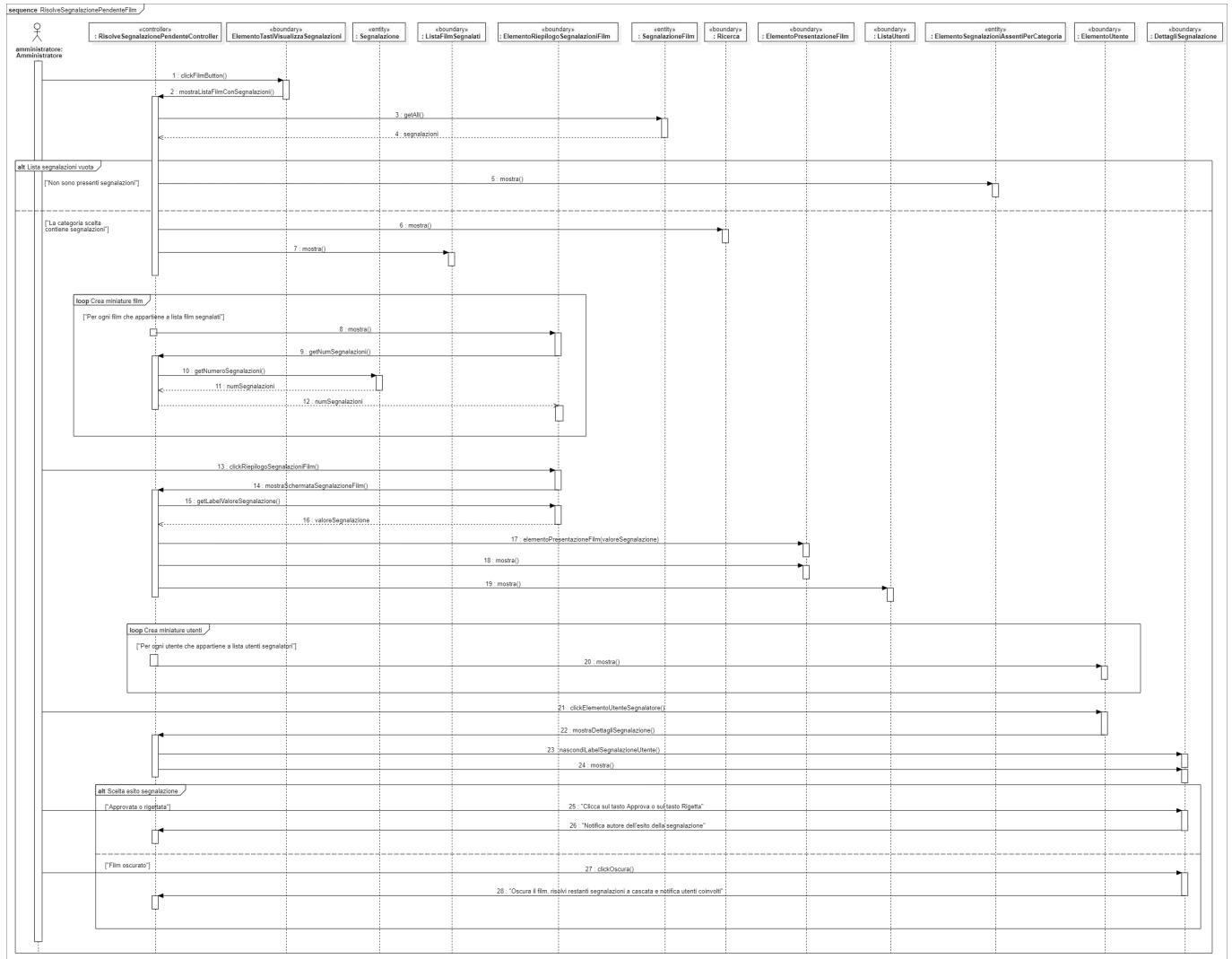
## Amministratore loggato visualizza segnalazioni gestite





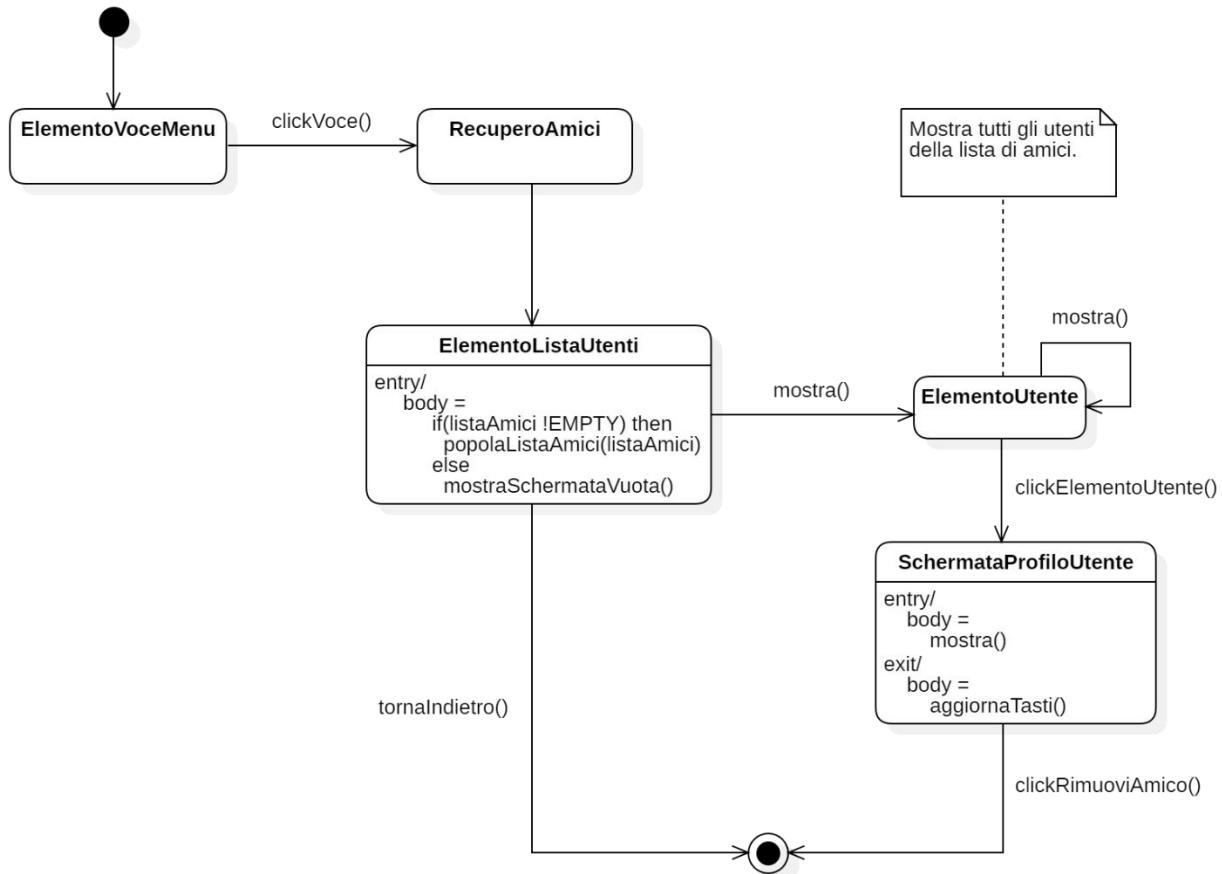
## Amministratore loggato risolve segnalazione pendente





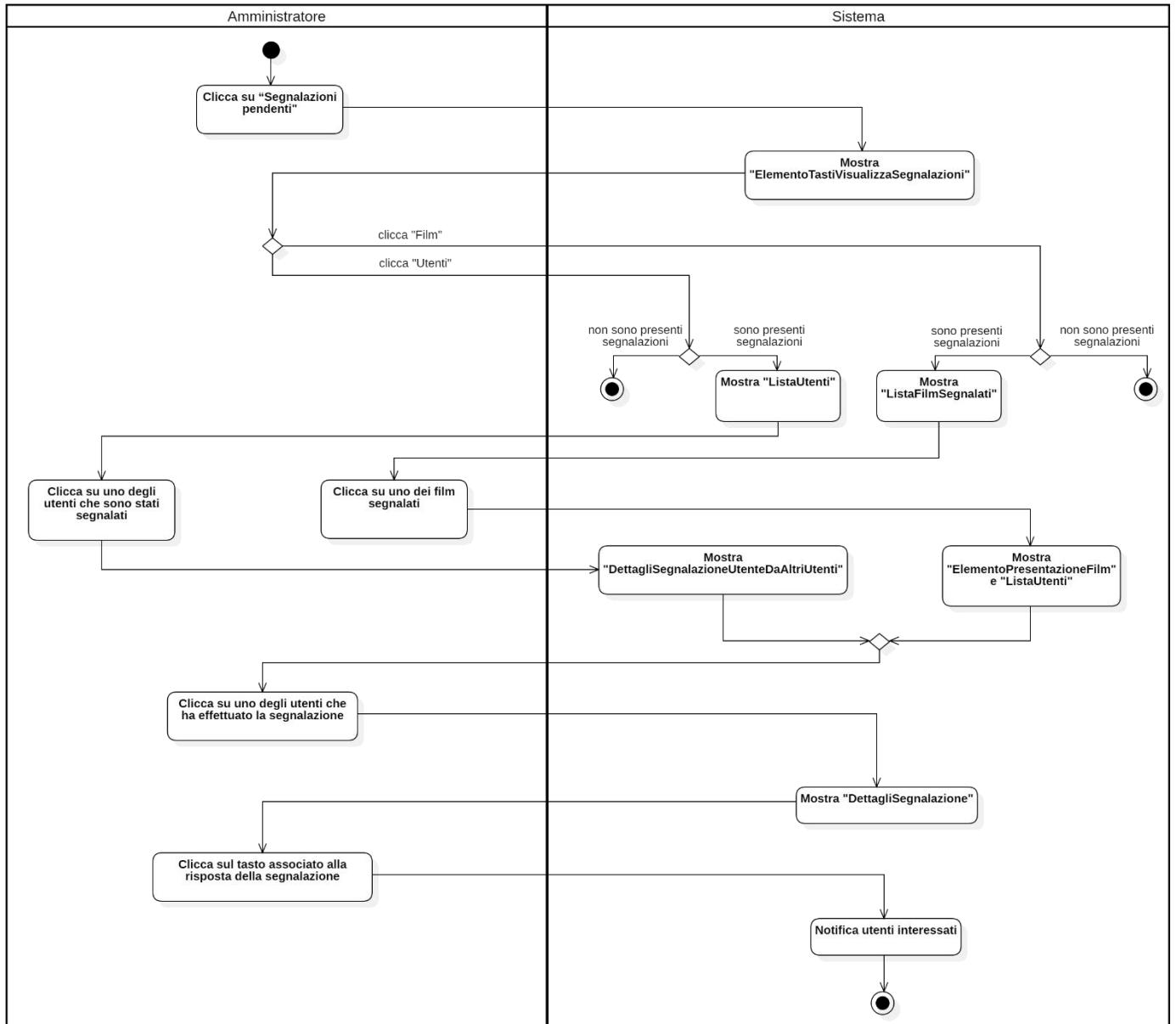
## Statechart Diagram

Utente rimuove amico

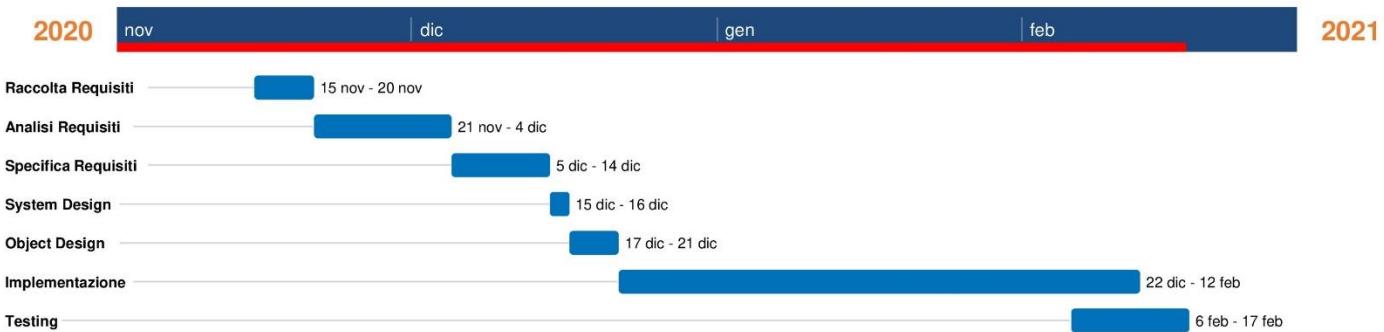


# Activity Diagram

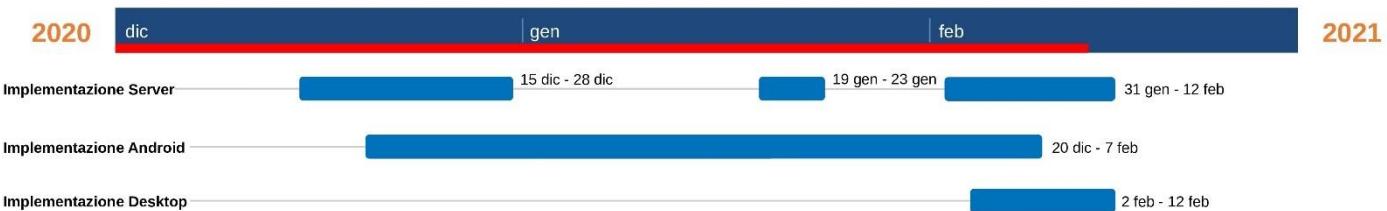
Amministratore risolve segnalazione pendente



## Diagramma di Gantt



Fase di implementazione nel dettaglio:



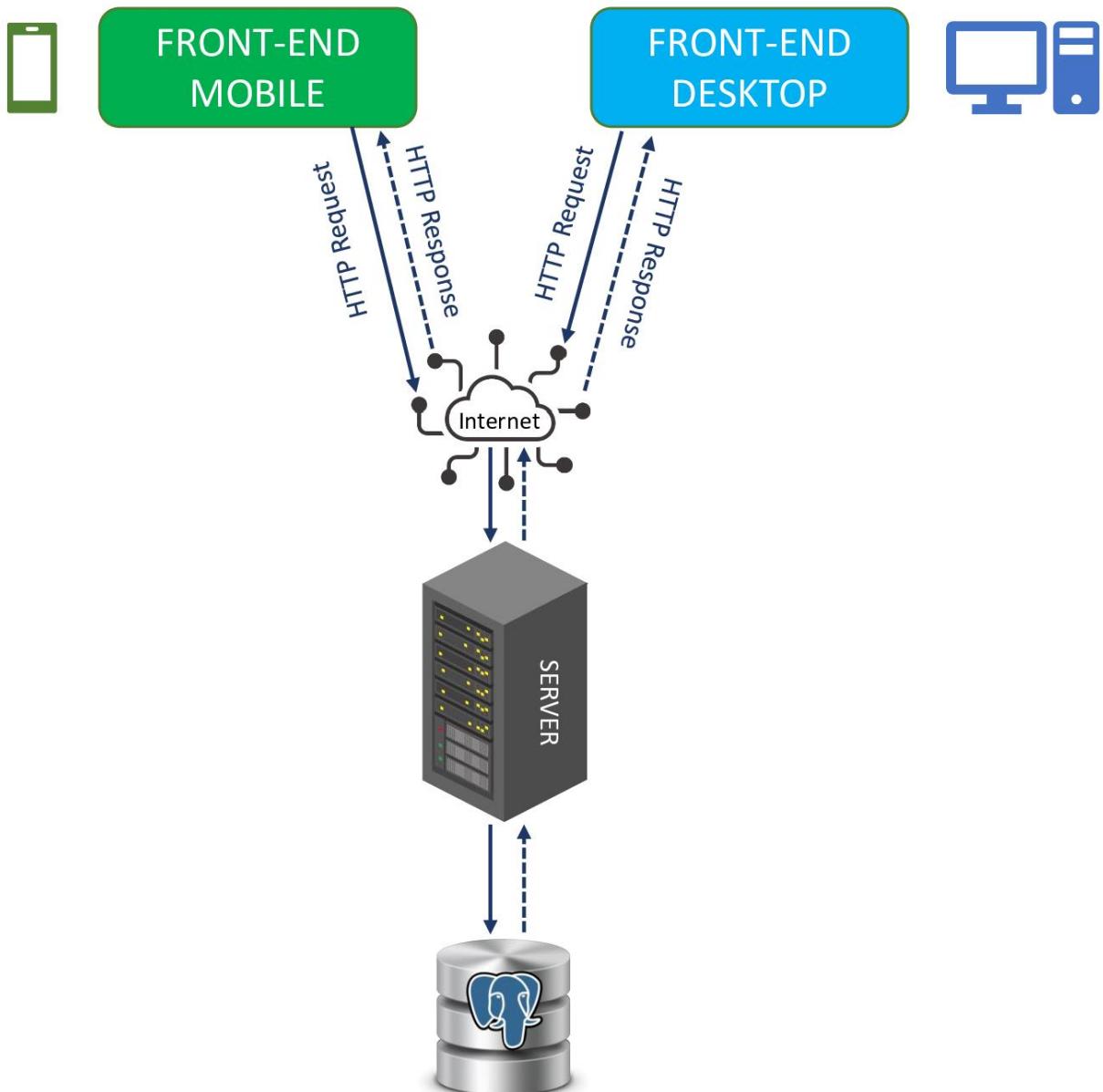
## Organizzazione del team

Data l'esigenza di poter lavorare solo ed esclusivamente a distanza, c'è stata un'accurata organizzazione del come portare avanti il progetto e come condividersi i file. Oltre ad un classico strumento di VSC, tutti i file che compongono il progetto sono stati condivisi su OneDrive, poiché in questo modo si avevano tutti i file condivisi in real-time, inoltre questo cloud è ben integrato con le piattaforme utilizzate da tutti i componenti del team. Questo strumento, oltre a fornire supporto per la condivisione dei file sorgenti, ha consentito di versionare qualsiasi altro tipo di file. In merito ai file sorgenti, è stato utilizzato un doppio versionamento, quello di OneDrive che ad ogni aggiornamento aggiungeva una voce al versionamento, ed il classico versionamento su GitHub, in questo modo, in caso di mal funzionamento di una delle due piattaforme, si è avuto comunque a disposizione uno storico nell'eventualità di fare dei rollback, o di visionare le versioni precedenti.

# Documento di Design

## Analisi dell'architettura

*CineMates20* è un sistema composto da 3 componenti: un server, che funge da tramite per le richieste al database, un front-end desktop e un front-end mobile (Android).



I client inviano richieste al back-end il quale, dopo averle elaborate, le propaga al database. Il back-end offre dunque un modo sicuro per indicizzare e recuperare i dati, in quanto non consente chiamate dirette sulla base di dati. Con questo tipo di architettura è possibile, inoltre, modificare il metodo di persistenza dei dati sostituendo il vecchio modulo con quello nuovo, oppure aggiungere in futuro ulteriori client, senza dover riprogettare interamente la struttura attuale.

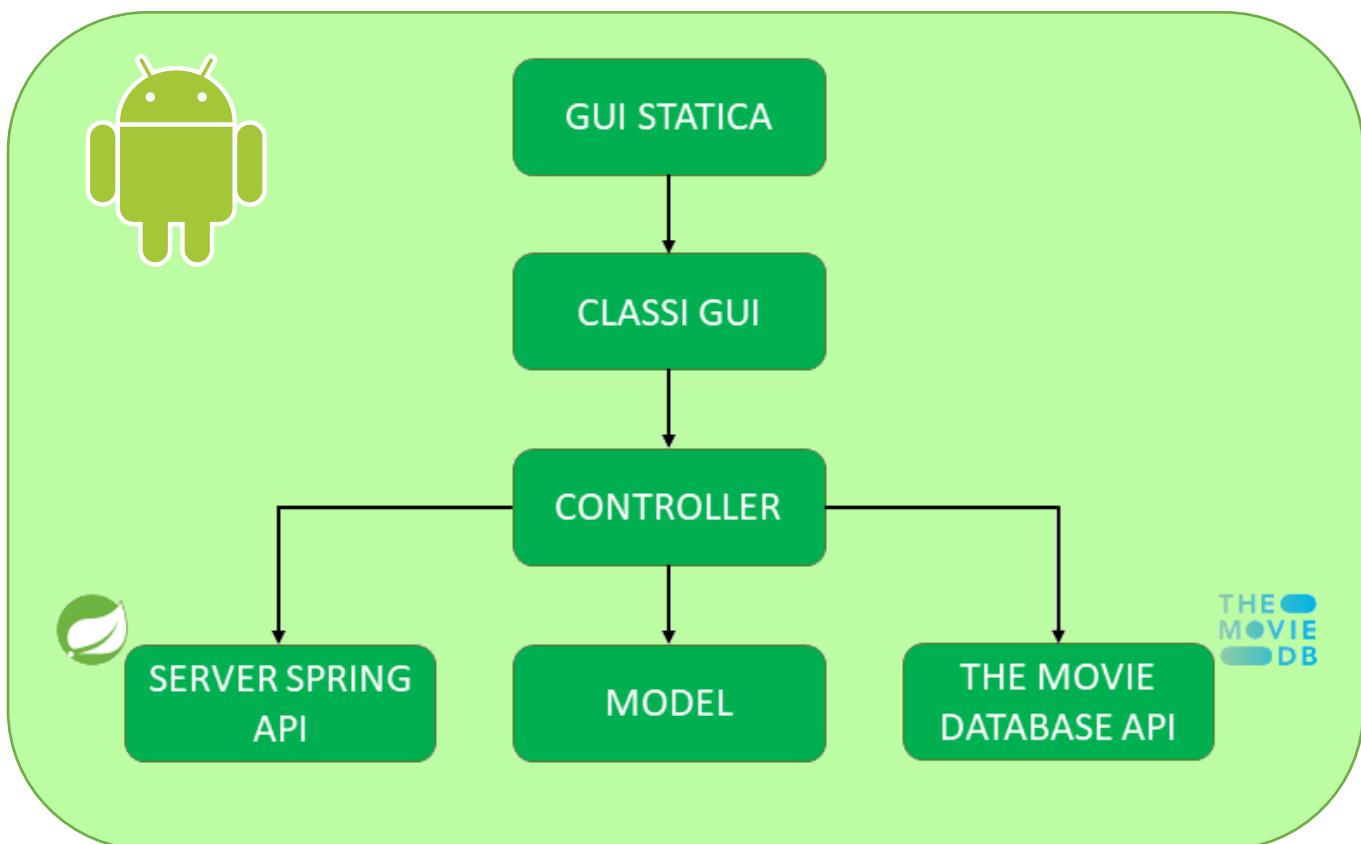
Tutti e tre gli applicativi sono stati scritti in linguaggio Java. Di seguito vengono illustrate le descrizioni delle tecnologie adottate per ciascun componente.

## Architettura Android

L'applicativo mobile è stato realizzato per il sistema operativo Android con API di versione minima 22 (Android 5.1). Gli unici permessi richiesti, per il recupero e l'invio dei dati al server sono:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Per la resa grafica, sono stati sfruttati i layout nativi di Android in formato XML, quindi anche in questo caso, come per l'applicativo Desktop, è garantita la separazione tra il layer di presentazione statica, il layer di presentazione dinamica e la logica di business.



Dettagli sui singoli moduli che compongono l'applicativo Android:

- **GUI statica** : è l'insieme dei layout XML.
- **Classi GUI** : carica i layout e cattura gli eventi scatenati dall'utente, si compongono di Activity e Fragment.
- **Controller** : è responsabile del controllo della trasmissione dei dati tra il model e la vista. Mappa le azioni dell'utente in aggiornamenti del model.
- **Model** : implementa la logica di business.

## *Panoramica dei servizi esterni Android*

Segue una breve sintesi dei servizi utilizzati a supporto dell'intera architettura:

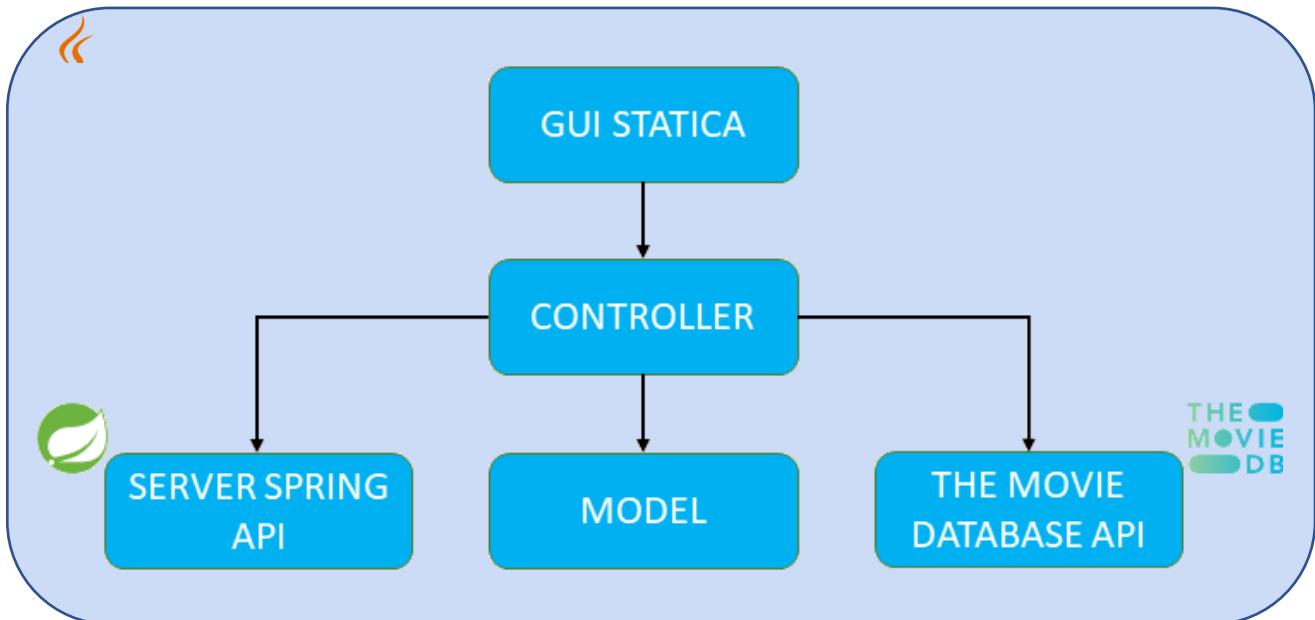
- **Amplify Framework<sup>2</sup>** per la gestione del pool di utenti, la loro registrazione ed autenticazione.
- **Facebook API** per l'autenticazione degli utenti con il loro account Facebook.
- **Google API** per l'autenticazione degli utenti con il loro account Google.
- **Amazon Elastic Beanstalk**, composto da **EC2**, per l'hosting del server.
- **Amazon RDS** con motore **PostgreSQL** per la persistenza dei dati.
- **Amazon S3** per la persistenza delle foto profilo degli utenti.
- **The Movie Database** per il recupero delle informazioni relative ai film.
- **Spring for Android** per la comunicazione con il web server.

---

<sup>2</sup> Amplify Framework: questo framework internamente fa uso di **Cognito**, un servizio di AWS.

## Architettura Desktop

L'applicativo Desktop è stato realizzato attraverso l'ausilio del framework JavaFX per la resa grafica che fornisce benefici in termini di responsabilità, dal momento che la maggior parte della grafica (statica) è stata realizzata tramite documenti FXML.



Dettagli sui singoli moduli che compongono l'applicativo Desktop:

- **GUI statica** : è l'insieme dei layout FXML.
- **Controller** : è responsabile del controllo della trasmissione dei dati tra il model e la vista. Essendo la gestione della grafica relativamente semplice, si occupa di gestire i layout grafici attraverso i metodi forniti dal framework di JavaFX. Mappa quindi le azioni dell'utente in aggiornamenti del model.
- **Model** : implementa la logica di business.

### Panoramica dei servizi esterni Desktop

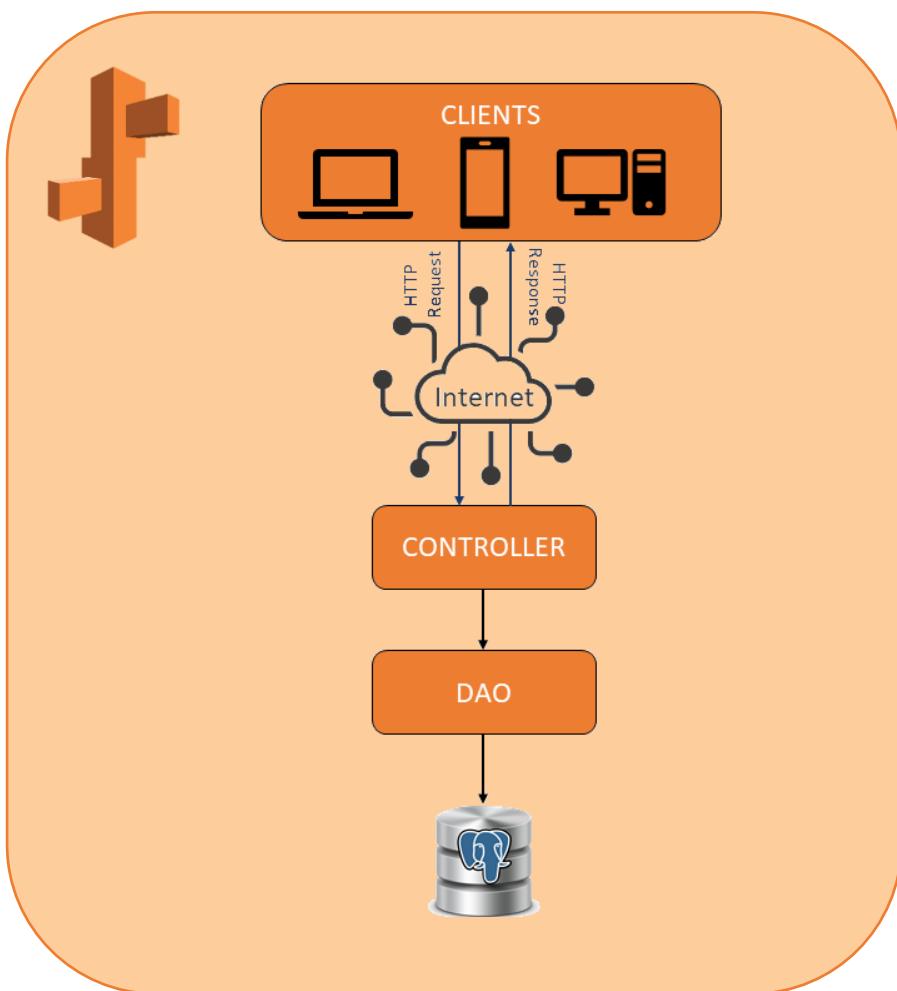
Segue una breve sintesi dei servizi utilizzati a supporto dell'intera architettura:

- **Amazon Elastic Beanstalk**, composto da **EC2**, per l'hosting del server.
- **Amazon RDS** con motore **PostgreSQL** per la persistenza dei dati.
- **Amazon S3** per la persistenza delle foto profilo degli utenti.
- **The Movie Database** per il recupero delle informazioni relative ai film.
- **Spring Boot** per la comunicazione con il web server.

## Architettura Server

Il server è stato realizzato con il supporto del framework **Spring Boot** che offre i seguenti vantaggi:

- Possibilità di incorporare direttamente applicazioni web server/container come Apache Tomcat, per cui non è necessario l'uso di file WAR (**Web Application Archive**);
- **Dependency injection** a supporto del pattern MVC, la quale prevede che tutti gli oggetti all'interno dell'applicazione accettino le dipendenze tramite costruttore o metodi setter. Non sono quindi gli stessi oggetti a creare le proprie dipendenze, ma esse vengono iniettate dall'esterno;
- Configurazione automatica di tutta una serie di servizi offerti dal framework.



Dettagli sui singoli moduli che compongono l'applicativo Server:

- **Controller** : è responsabile del controllo della trasmissione dei dati tra il client e i dao.
- **Dao** : è il responsabile delle comunicazioni con il database.

## *Panoramica dei servizi esterni Server*

Segue una breve sintesi dei servizi utilizzati a supporto dell'intera architettura:

- **Amazon Elastic Beanstalk** per l'hosting.
- **Amazon RDS** con motore **PostgreSQL** per la persistenza dei dati.
- **Spring Boot Security**, in particolare **BCrypt** per l'hashing delle e-mail e delle password gestite per gli amministratori.

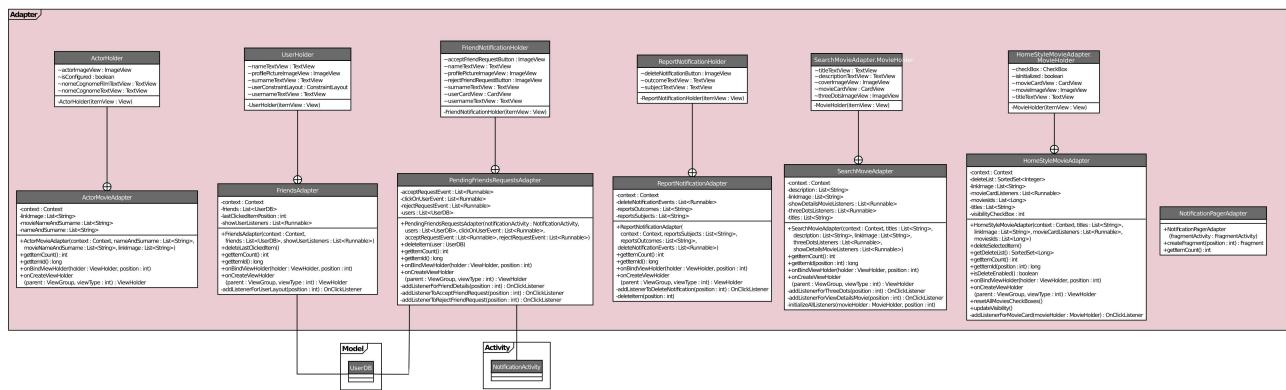
## Class Diagrams di Design

Nelle pagine seguenti si illustrano nel dettaglio i diagrammi delle classi relative agli applicativi Desktop e Mobile. I vari layer dell'architettura sono stati evidenziati con colori diversi per motivi di chiarezza. È stato inoltre adottato l'orientamento top-down per indicare che i livelli superiori sfruttano i servizi di quelli inferiori.

# Android

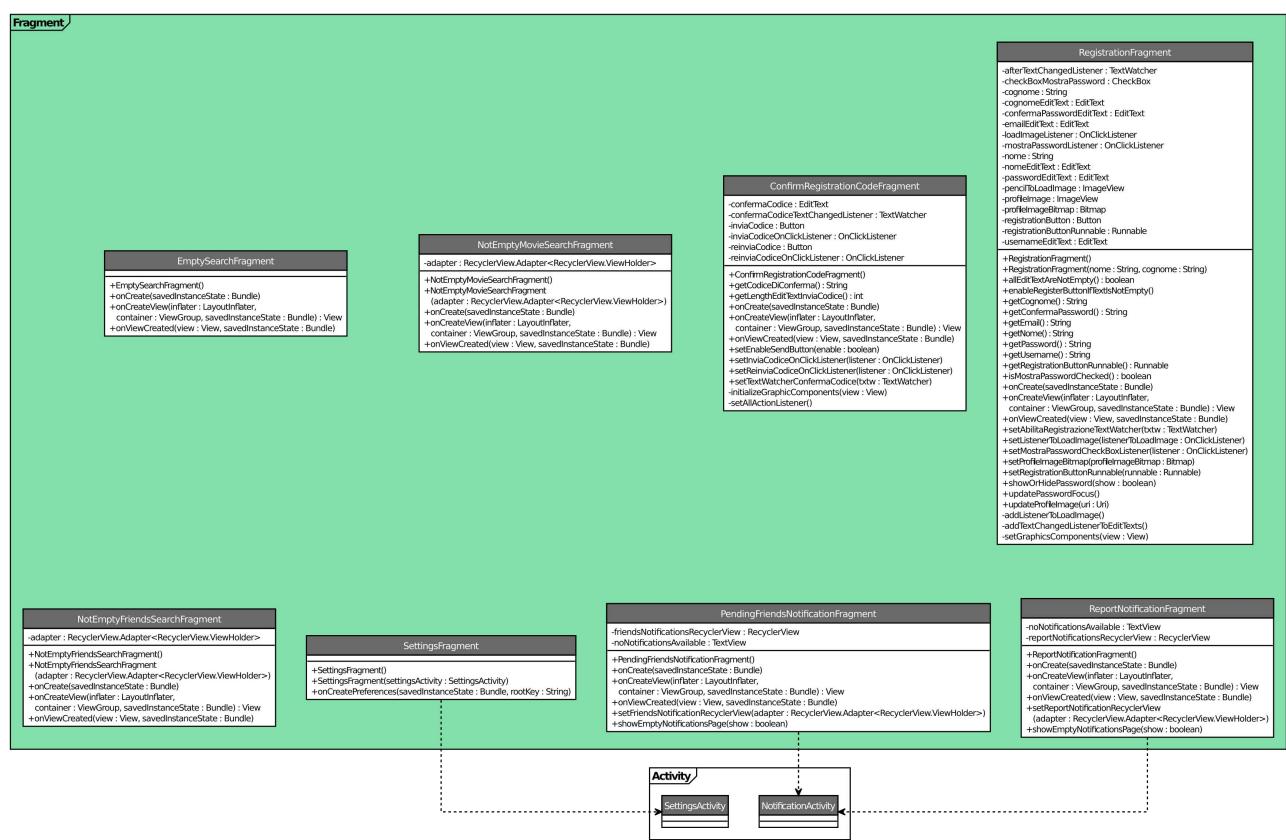
## Adapter

## Class Diagram Adapter

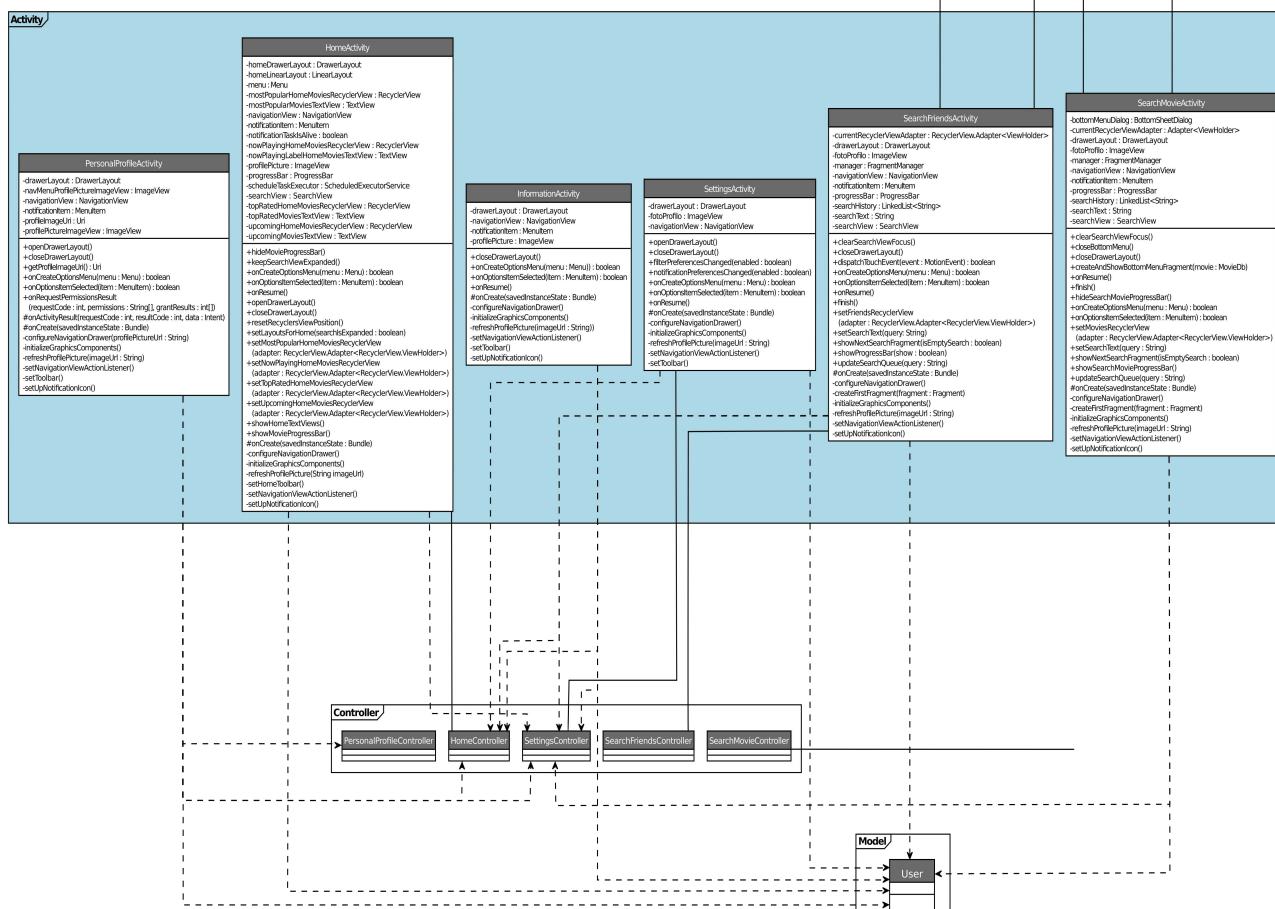
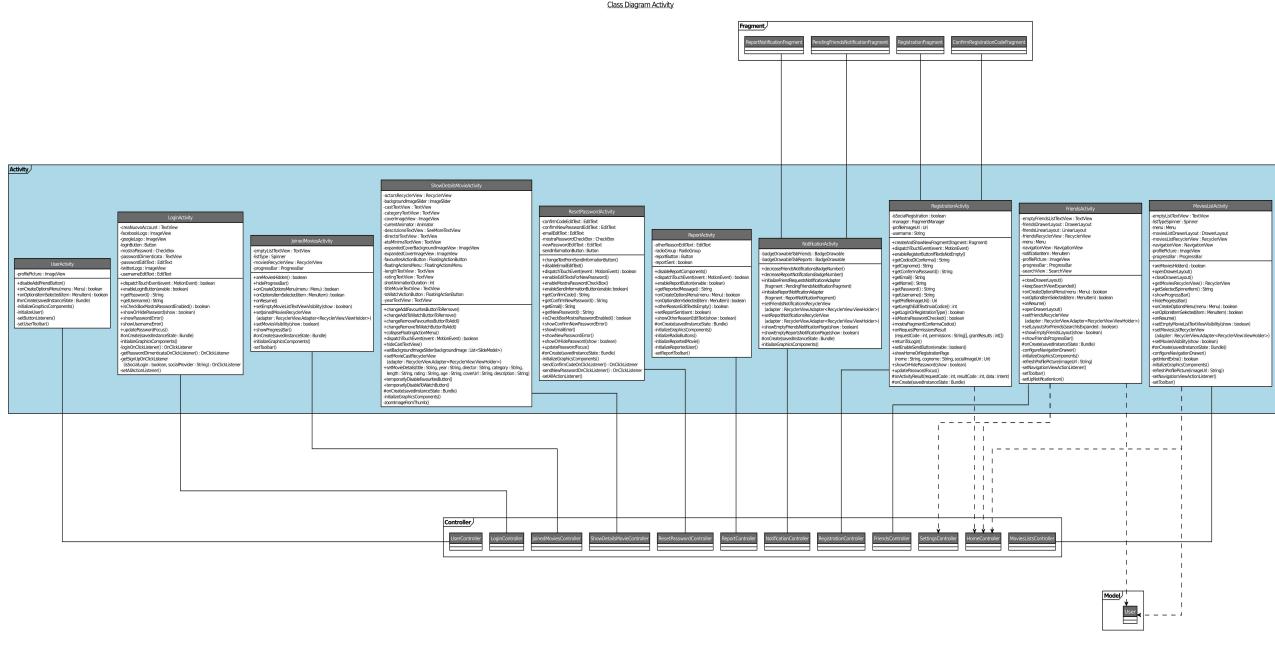


## Fragment

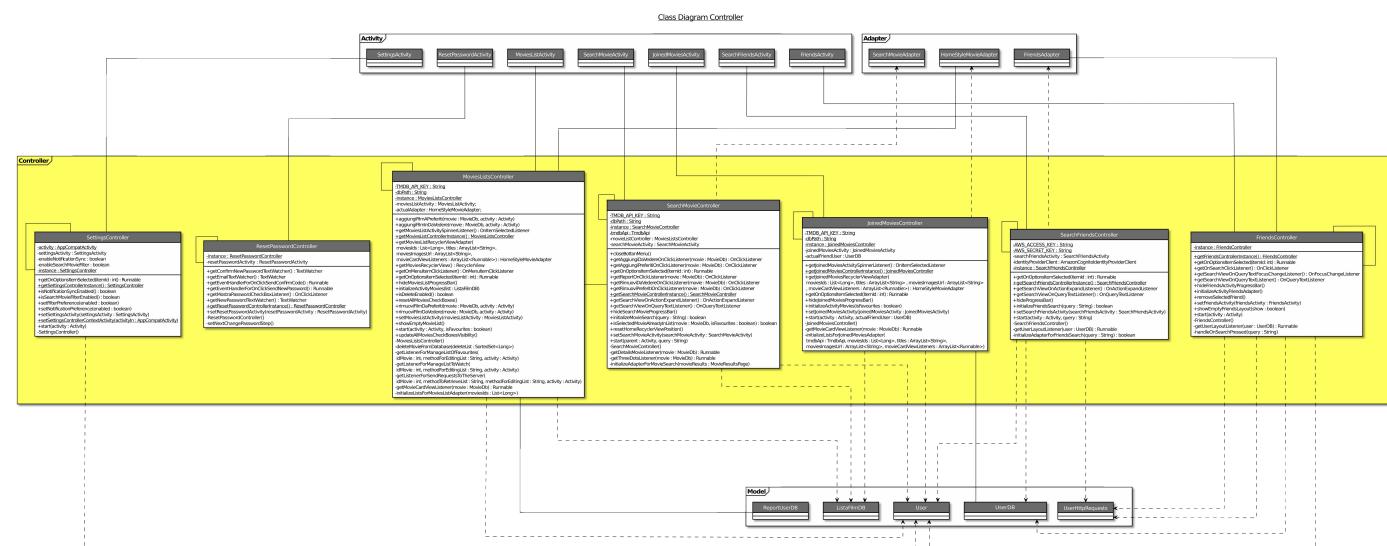
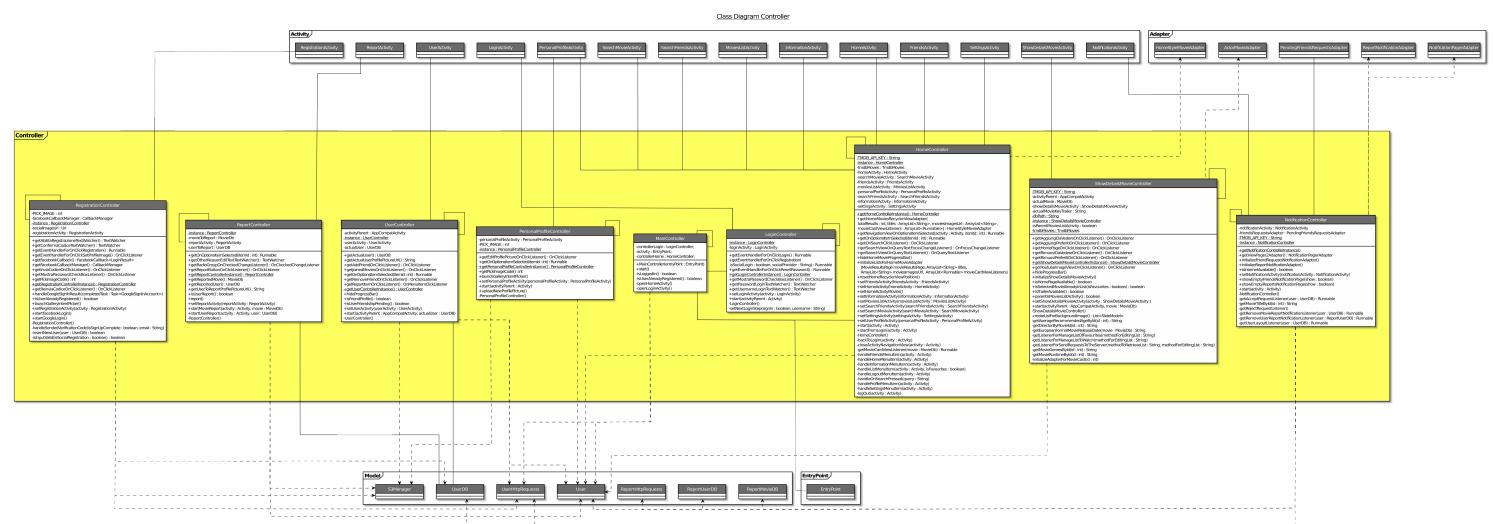
## Class Diagram Fragment



# Activity

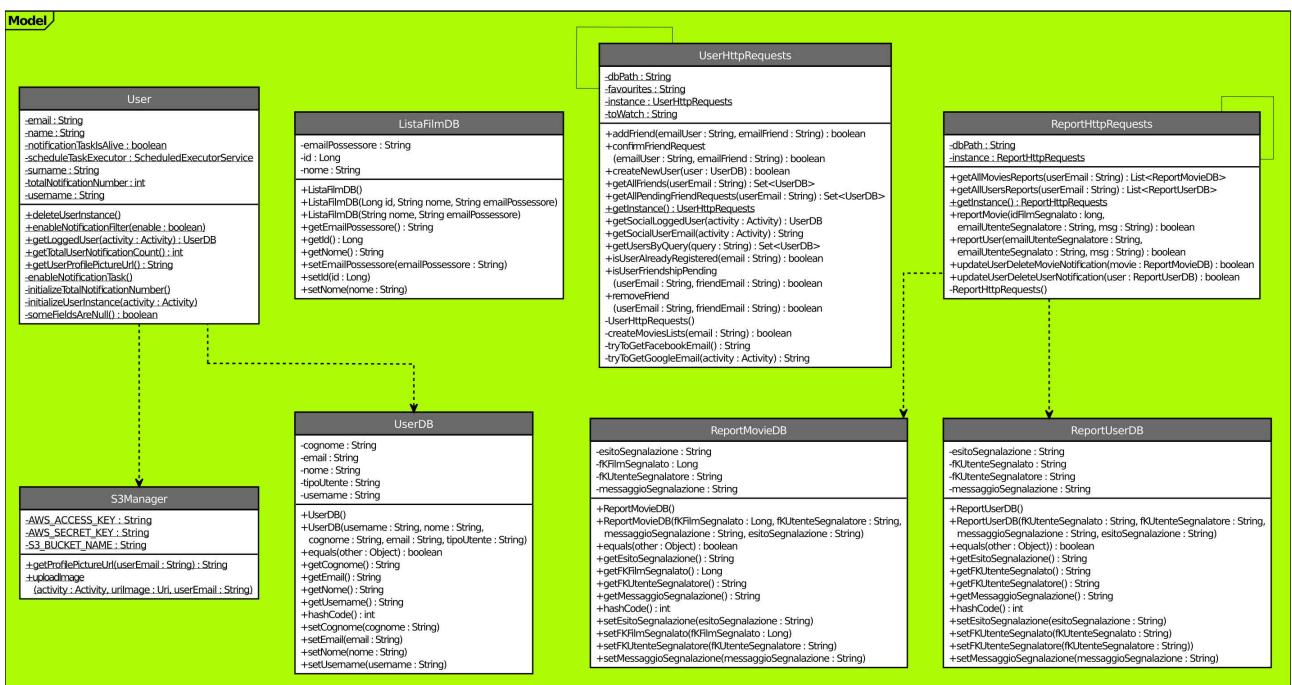


## Controller



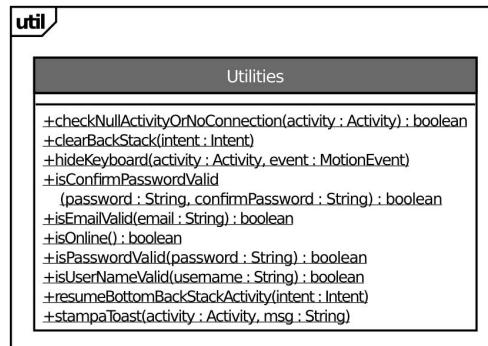
# Model

Class Diagram Model



# Util

La seguente classe è stata utilizzata per raggruppare varie funzioni di utility

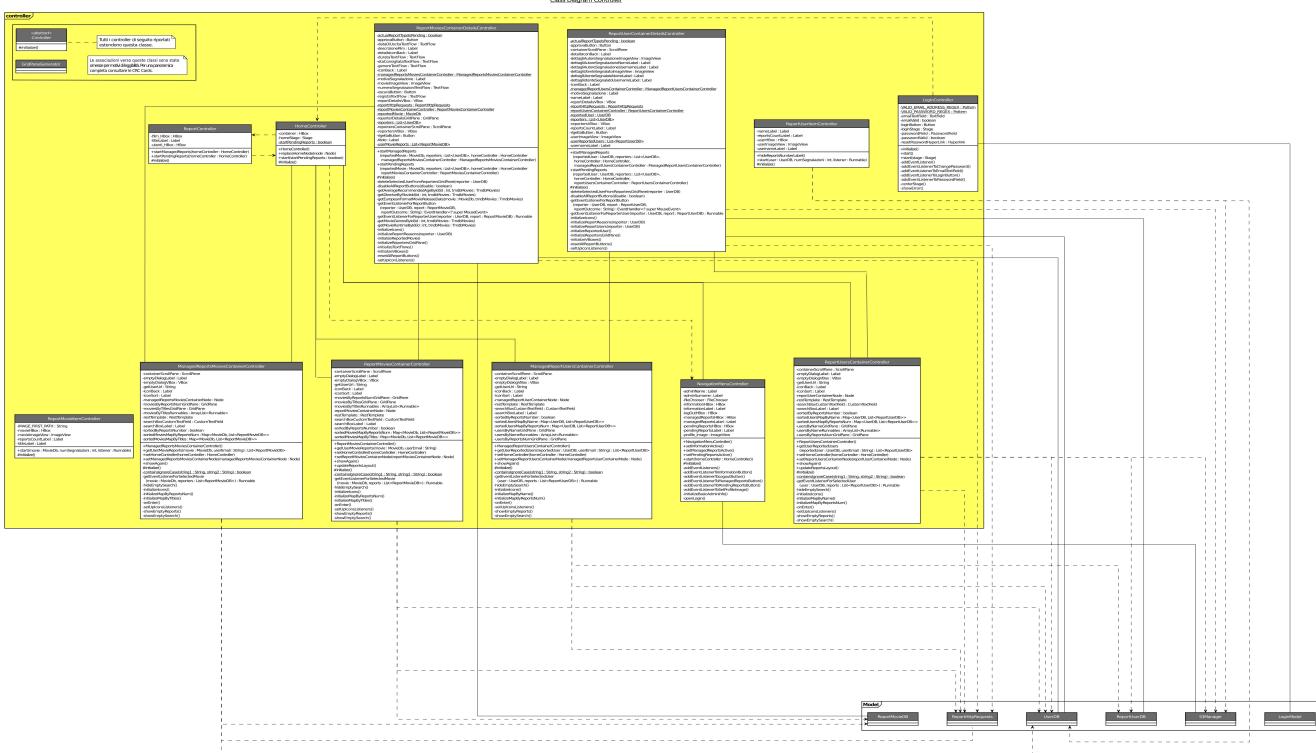


# Desktop



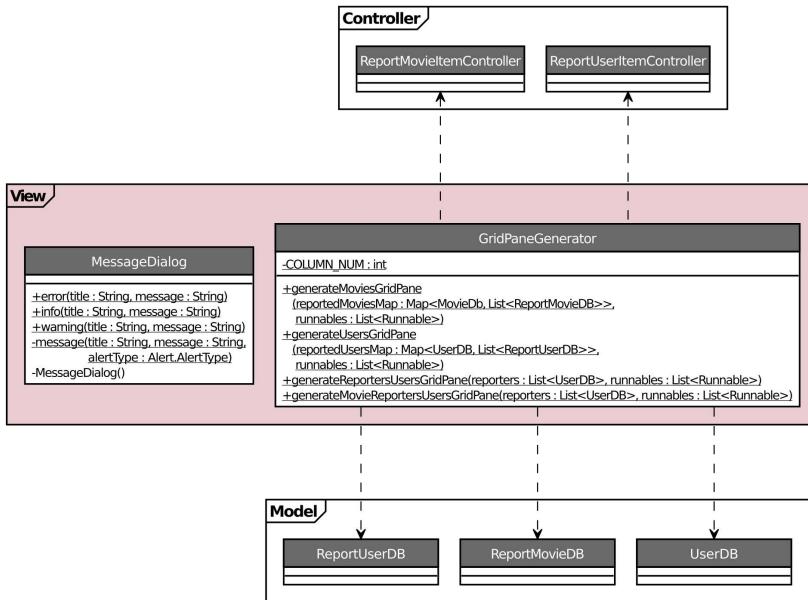
La seguente classe funge da punto di partenza dell'applicativo.

## Controller



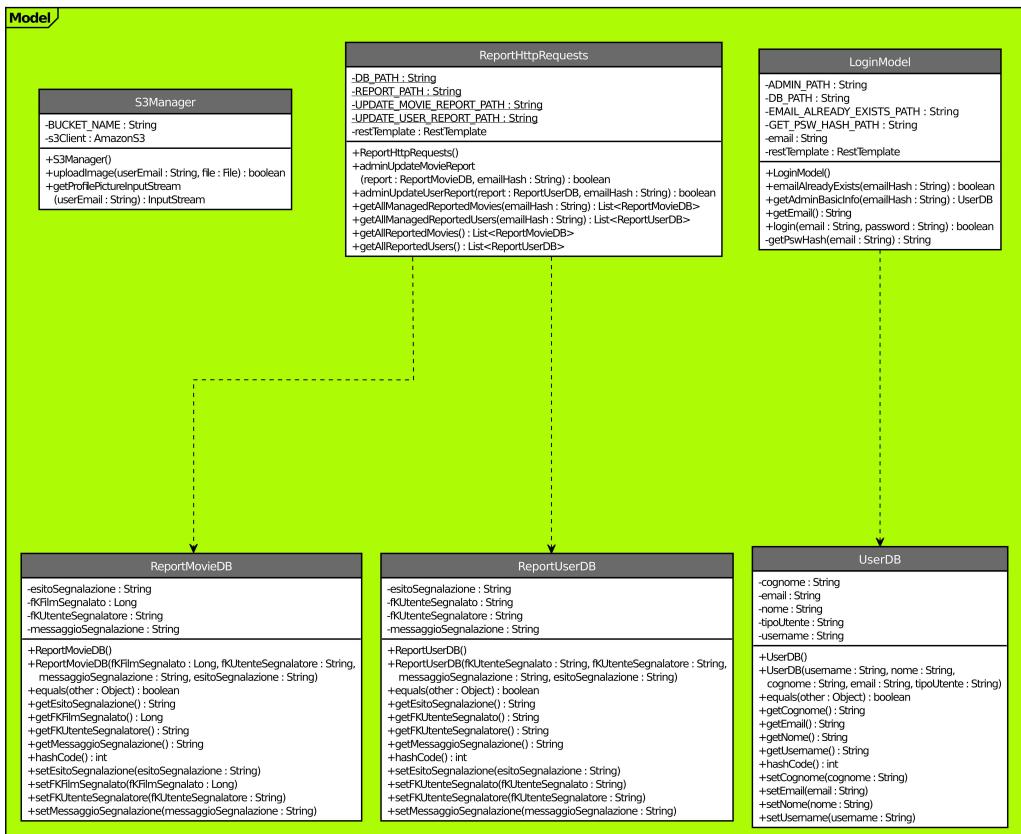
## View

Class Diagram View



## Model

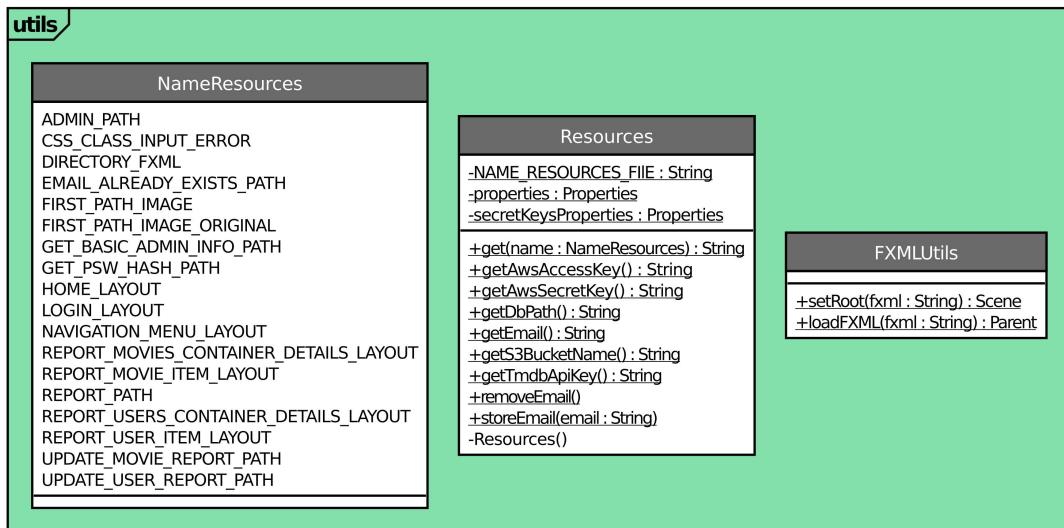
Class Diagram Model



## Utils

Questo package contiene classi che sono state utilizzate per raggruppare varie funzioni di utility.

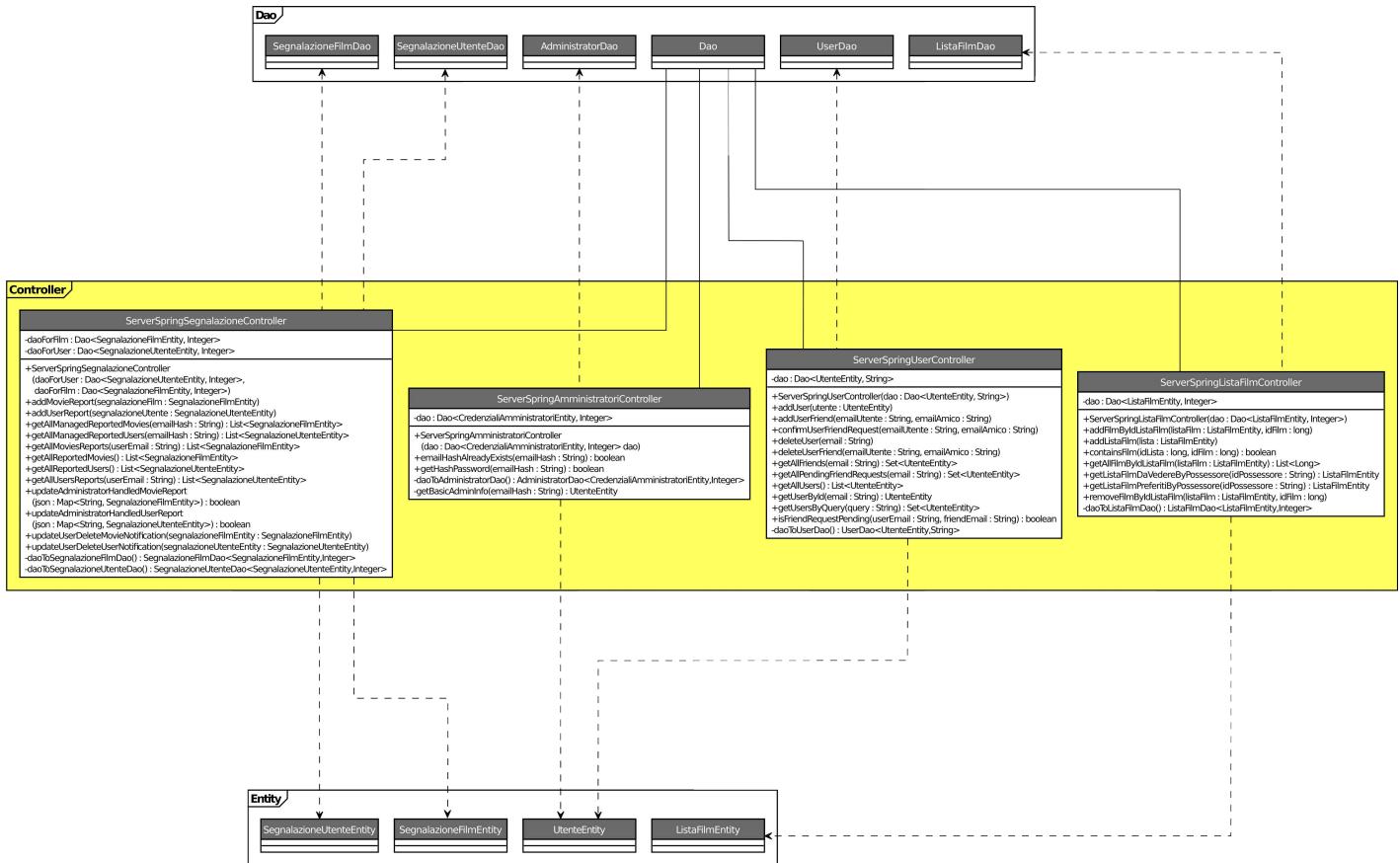
Class Diagram Utils



# Server

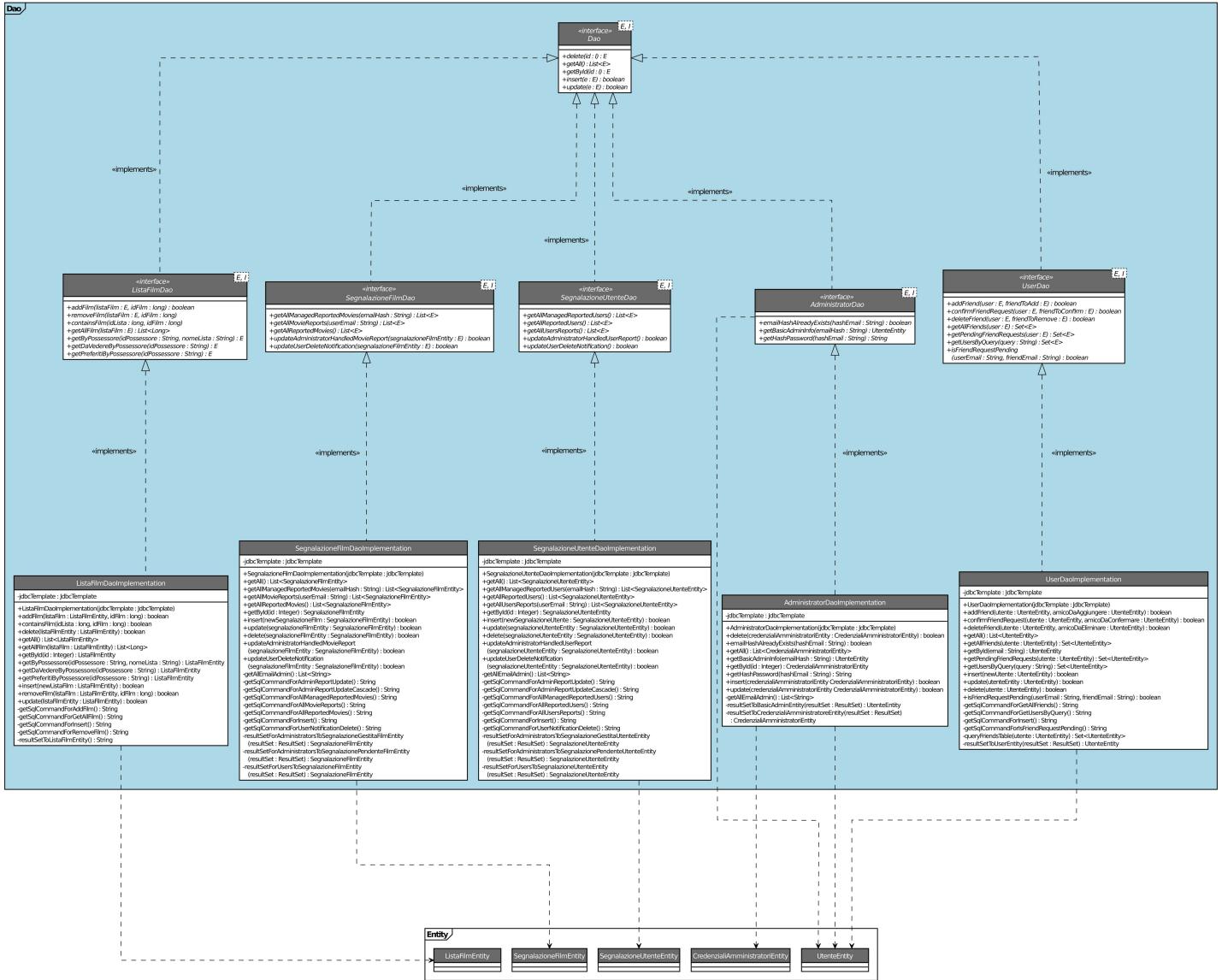
## Controller

Class Diagram Controller

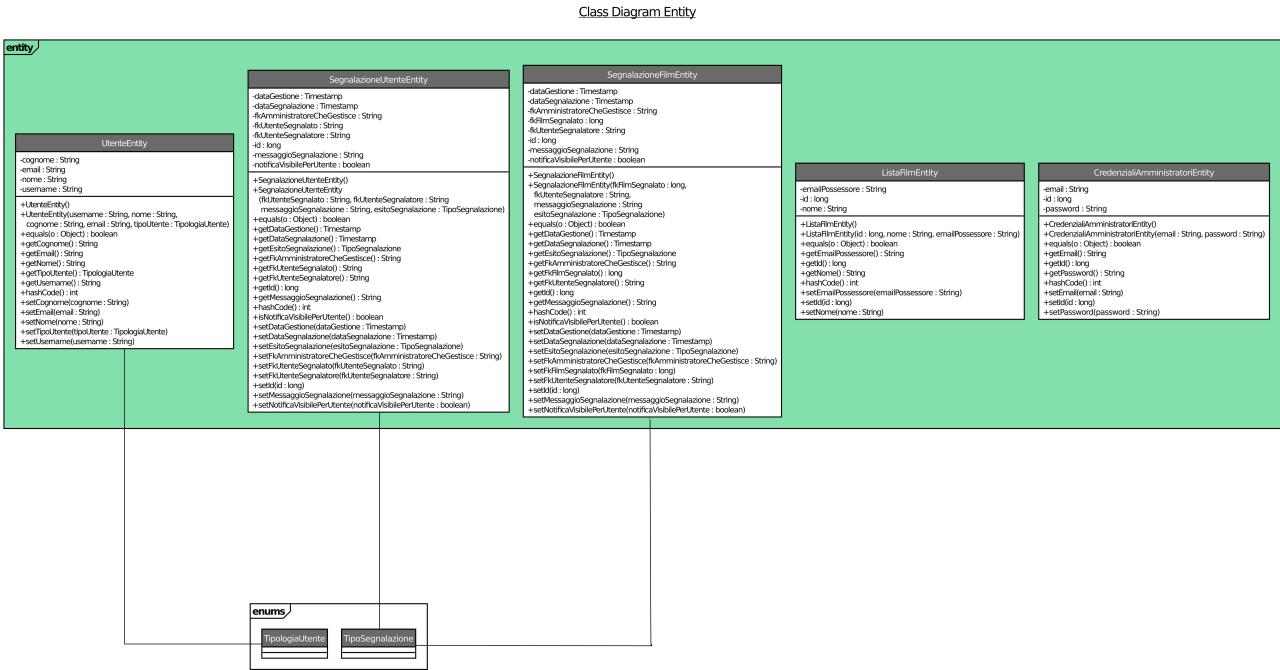


Dao

## Class Diagram DAC

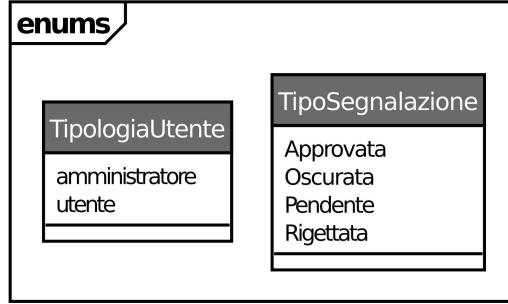


## Entity



## Enums

Class Diagram ENUM

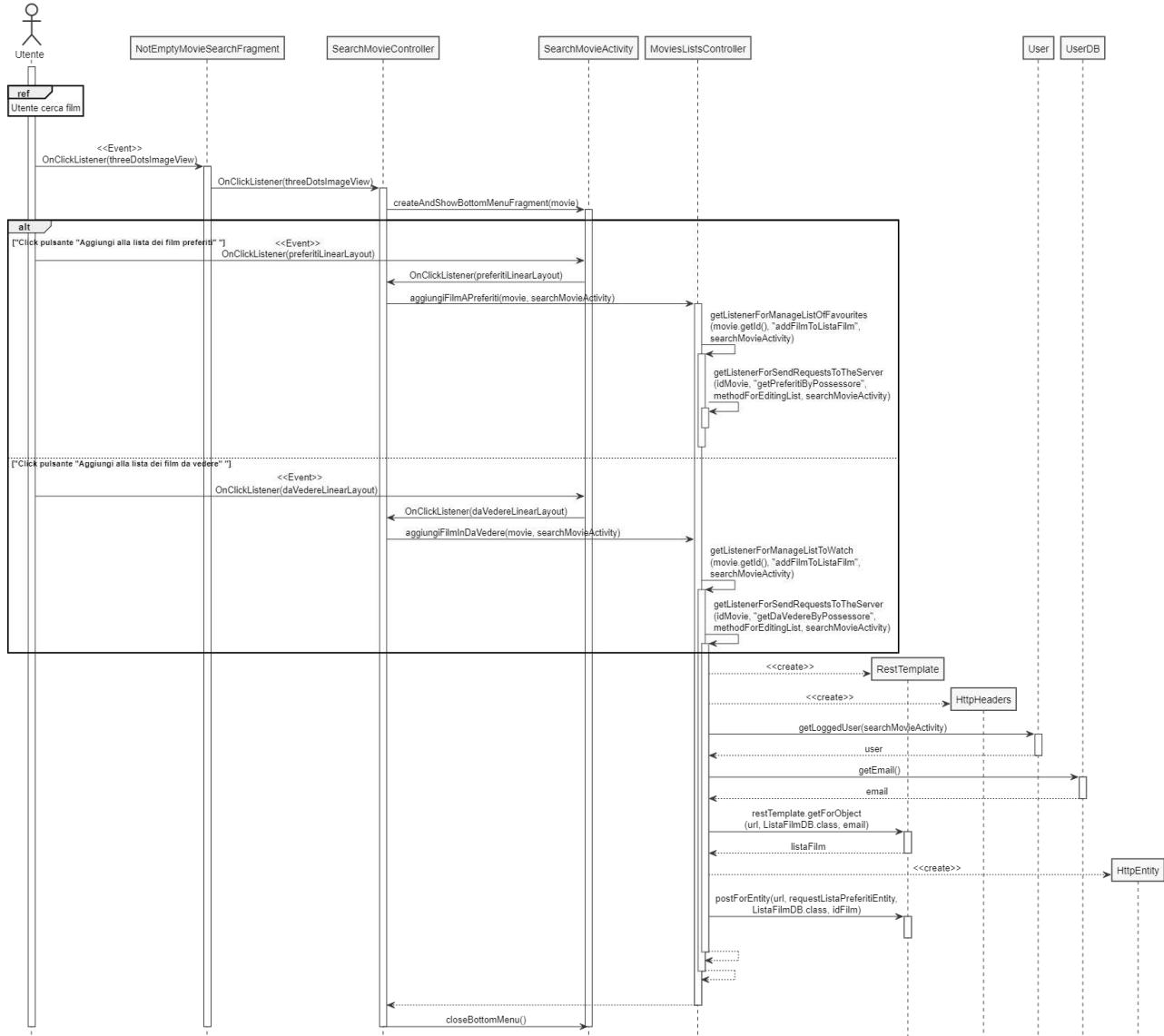


# Sequence Diagrams di Design

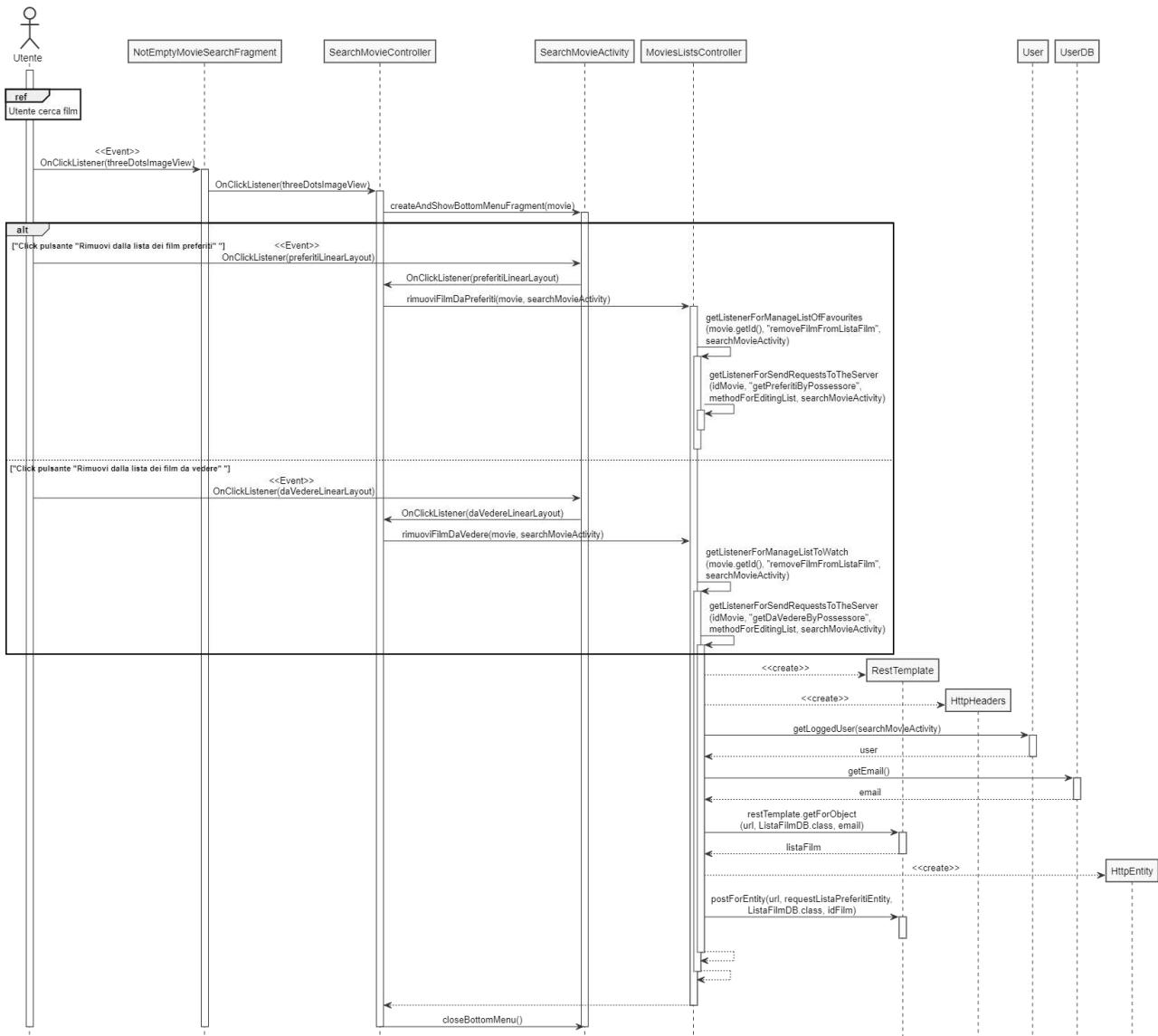
Per motivi di leggibilità, alcuni dei sequence diagrams di seguito illustrati fanno riferimento a delle funzioni, visibili di seguito in questo paragrafo.

## Android

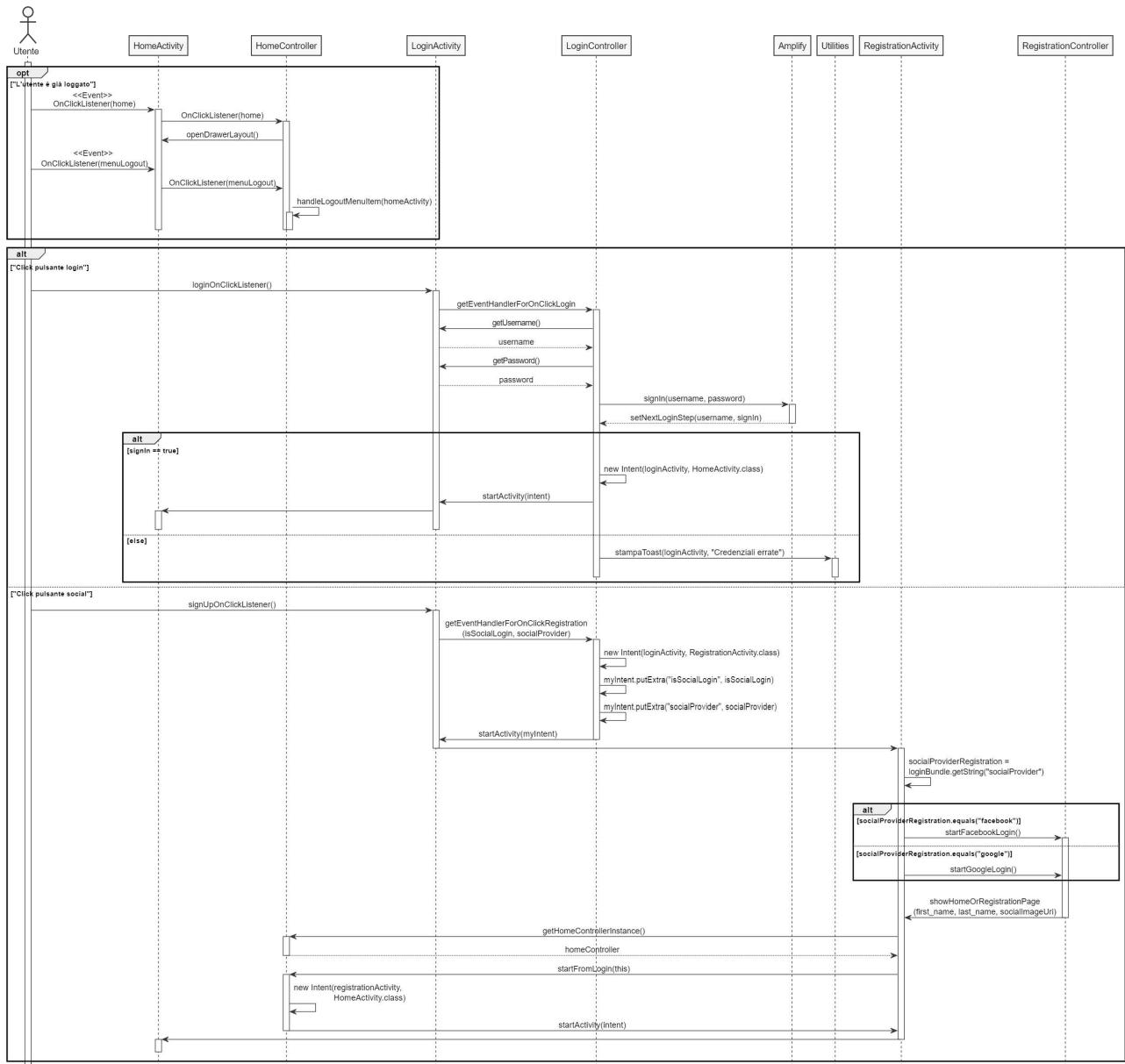
### Utente loggato aggiunge film ad una lista



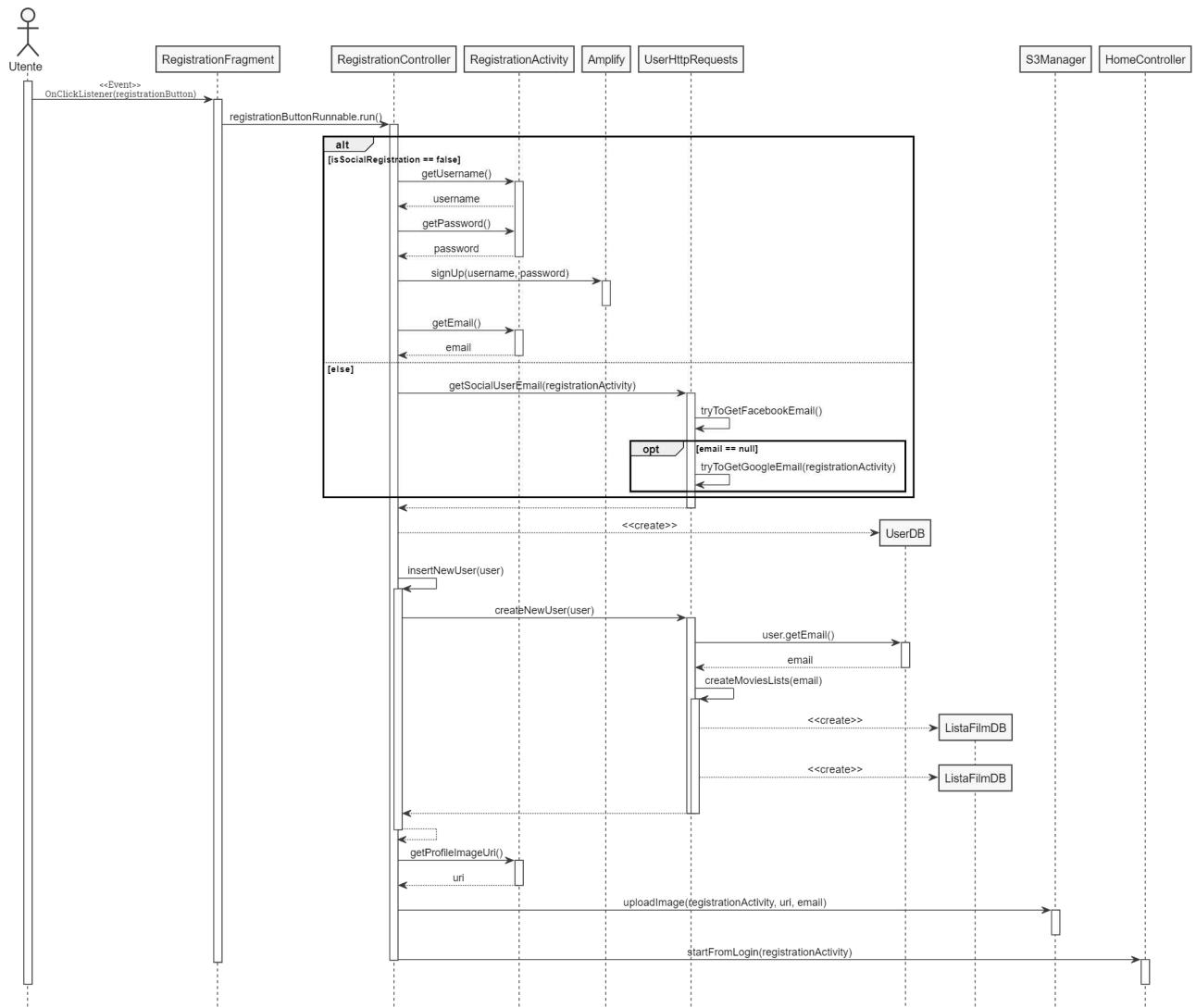
## Utente loggato rimuove film da una lista



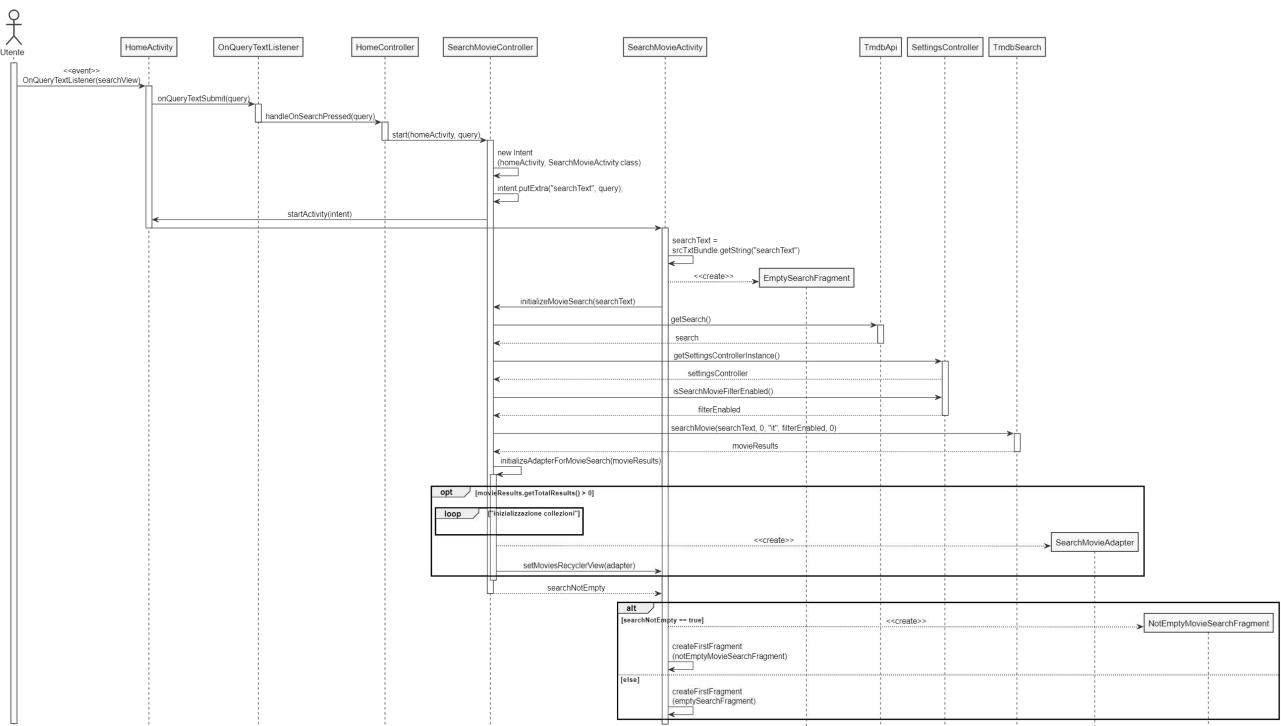
## Utente effettua login



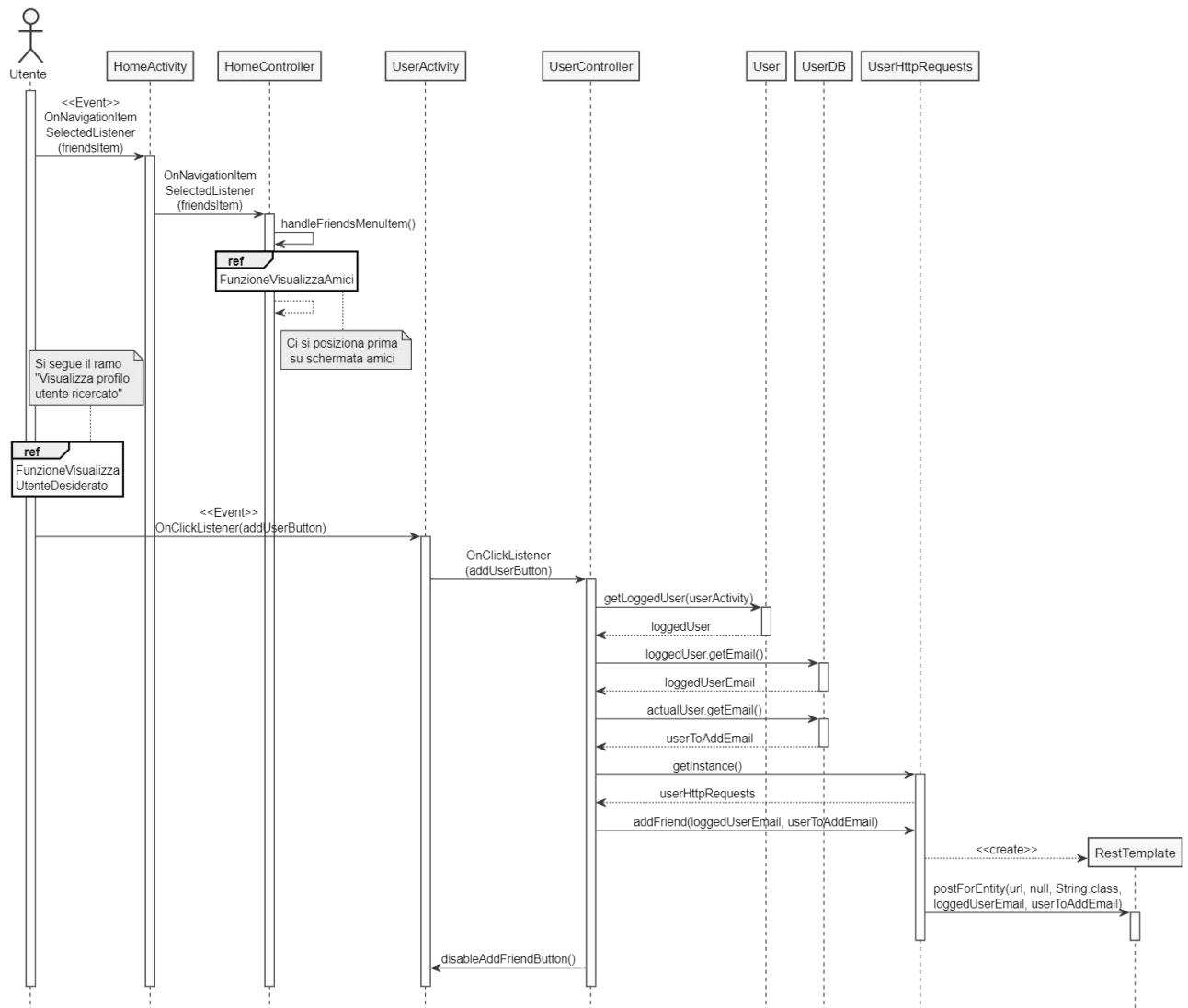
## Utente effettua registrazione



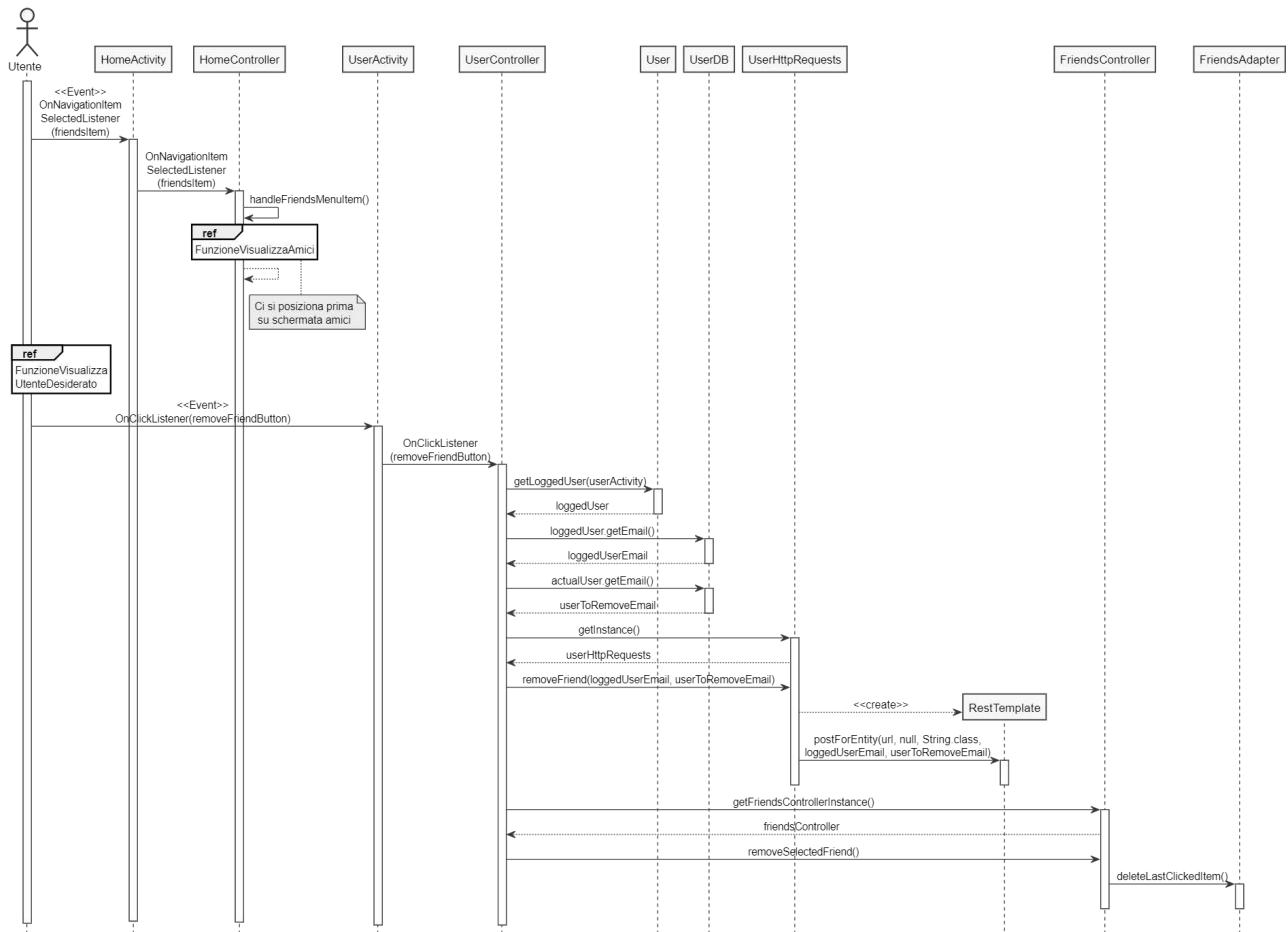
## Utente loggato cerca film



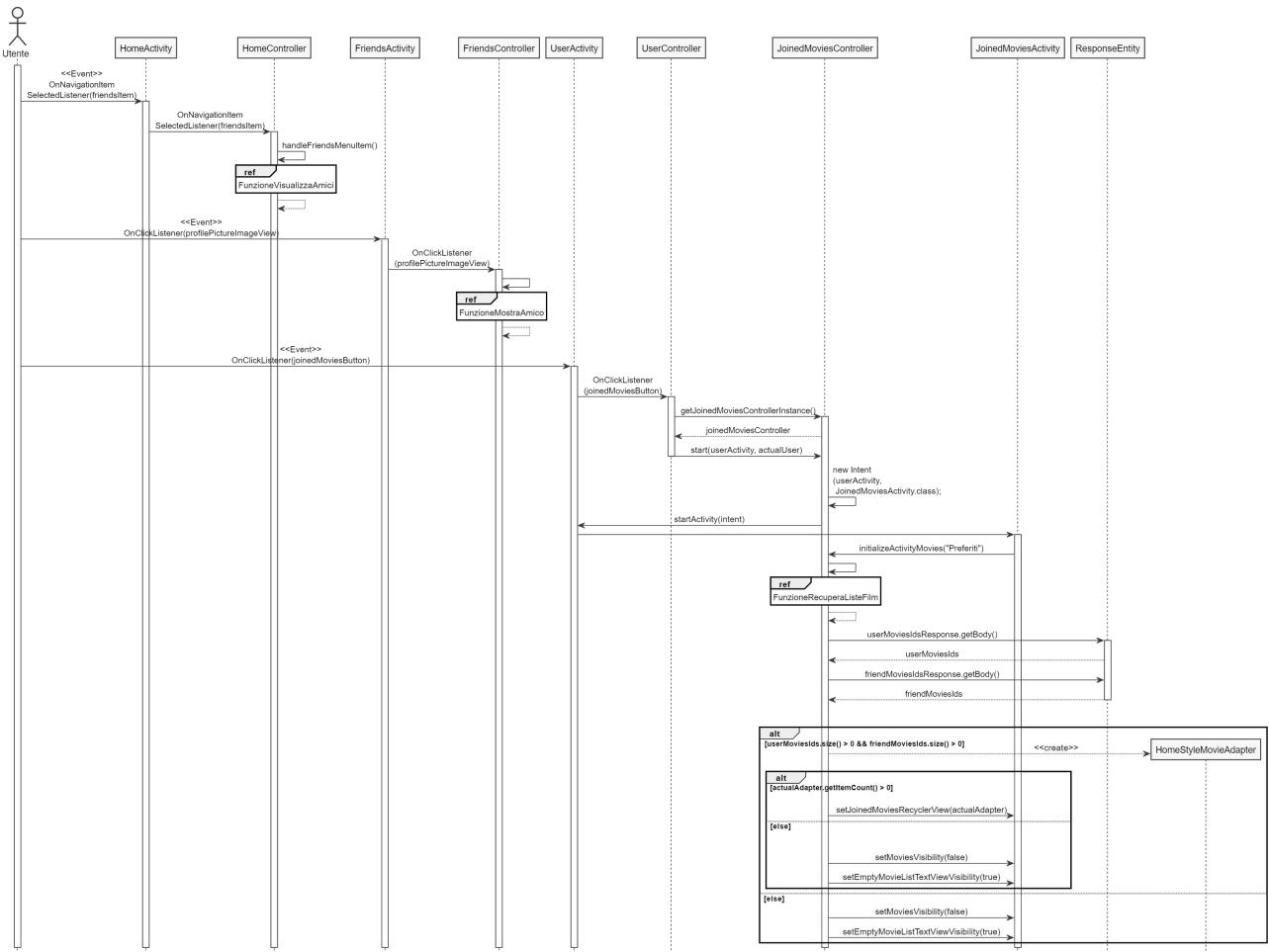
## Utente loggato invia richiesta di amicizia



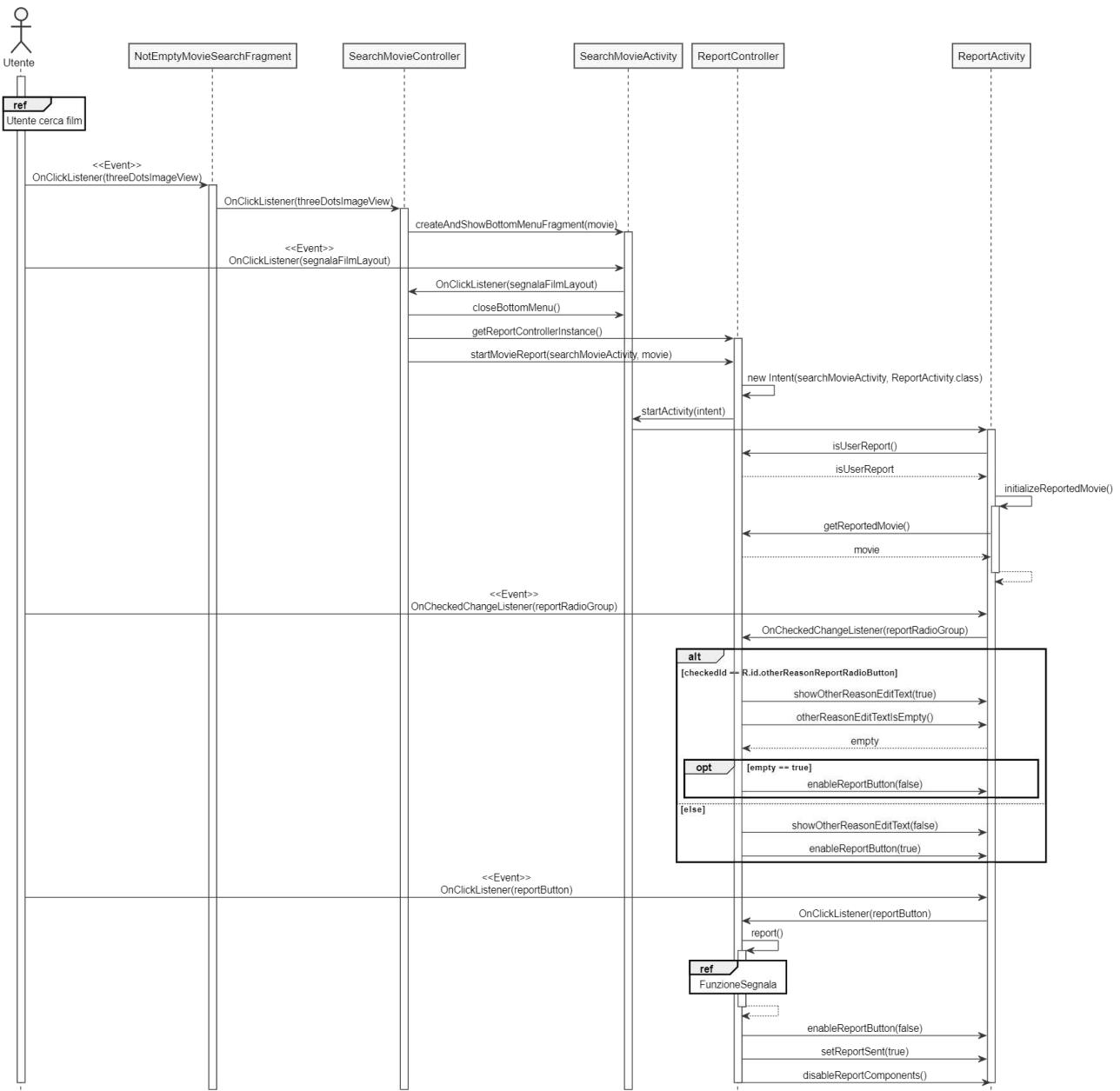
## Utente loggato rimuove amico



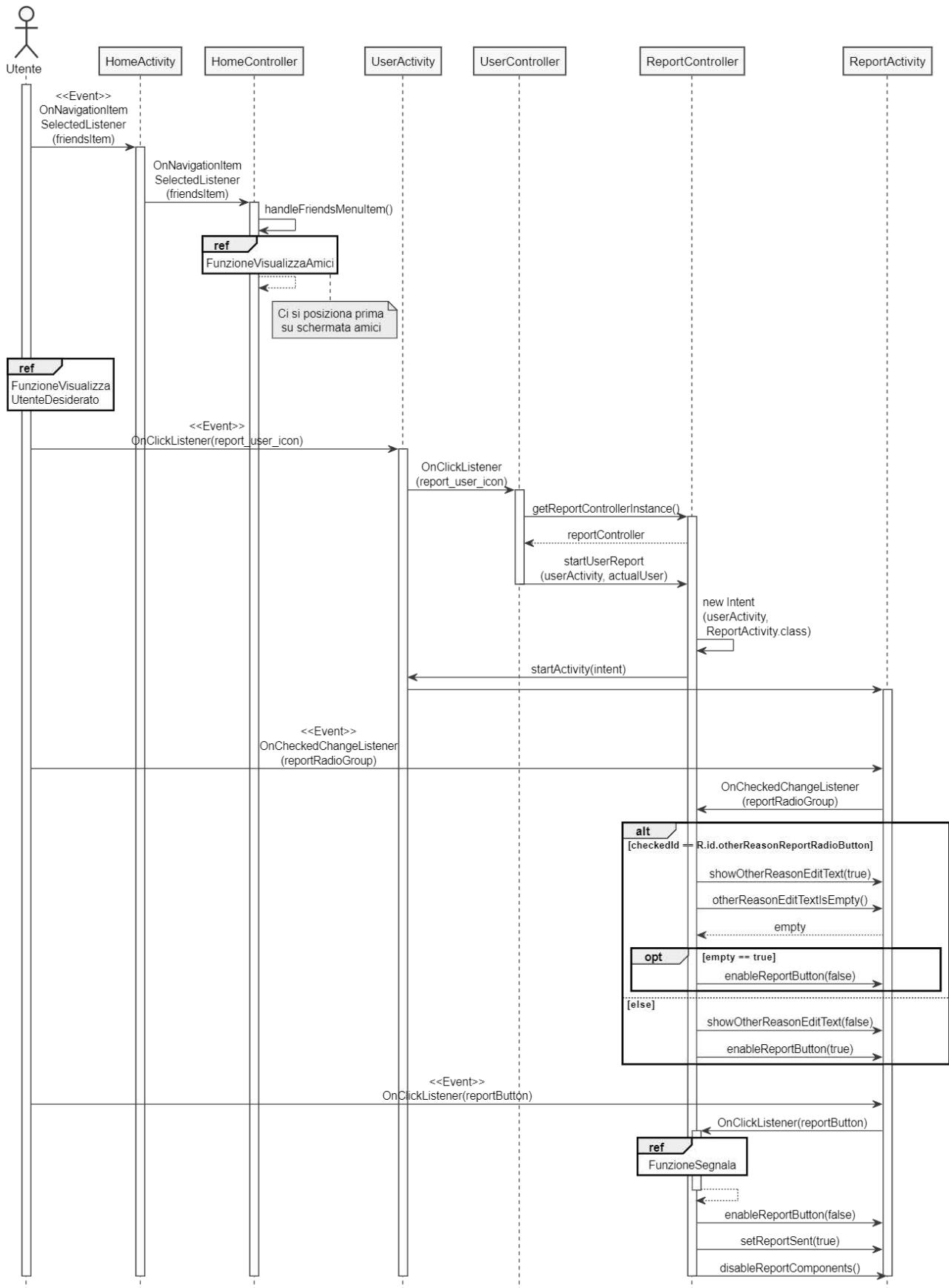
## Utente loggato visualizza film in comune



## Utente loggato segnala film

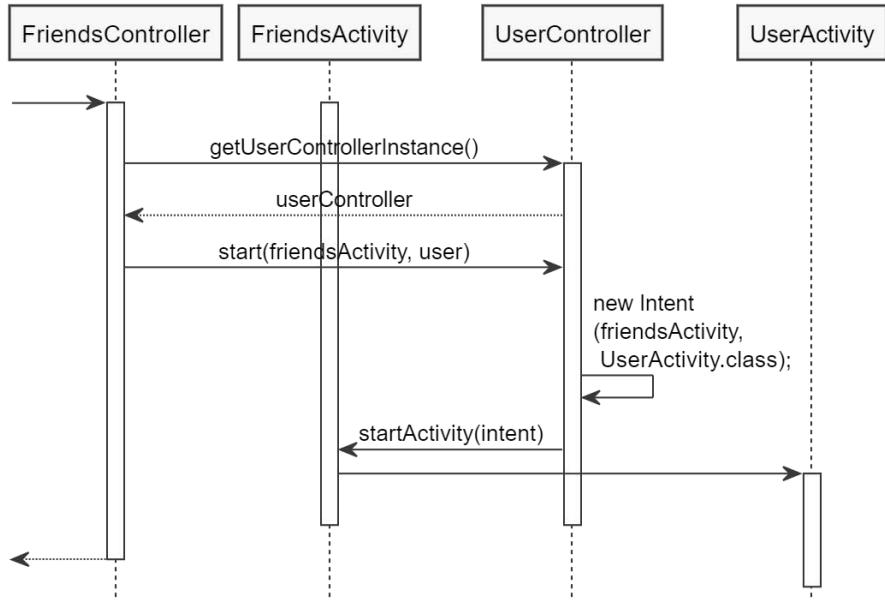


## Utente loggato segnala utente

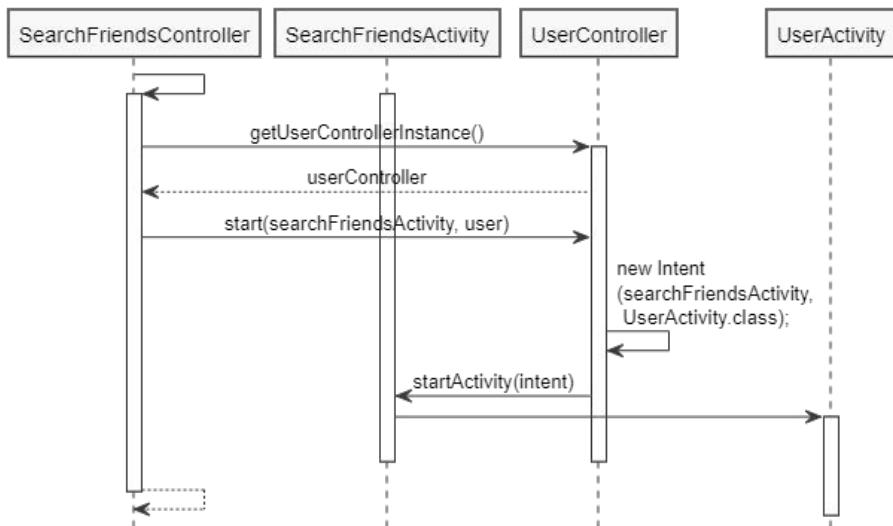


## Funzioni Sequence Diagrams Android

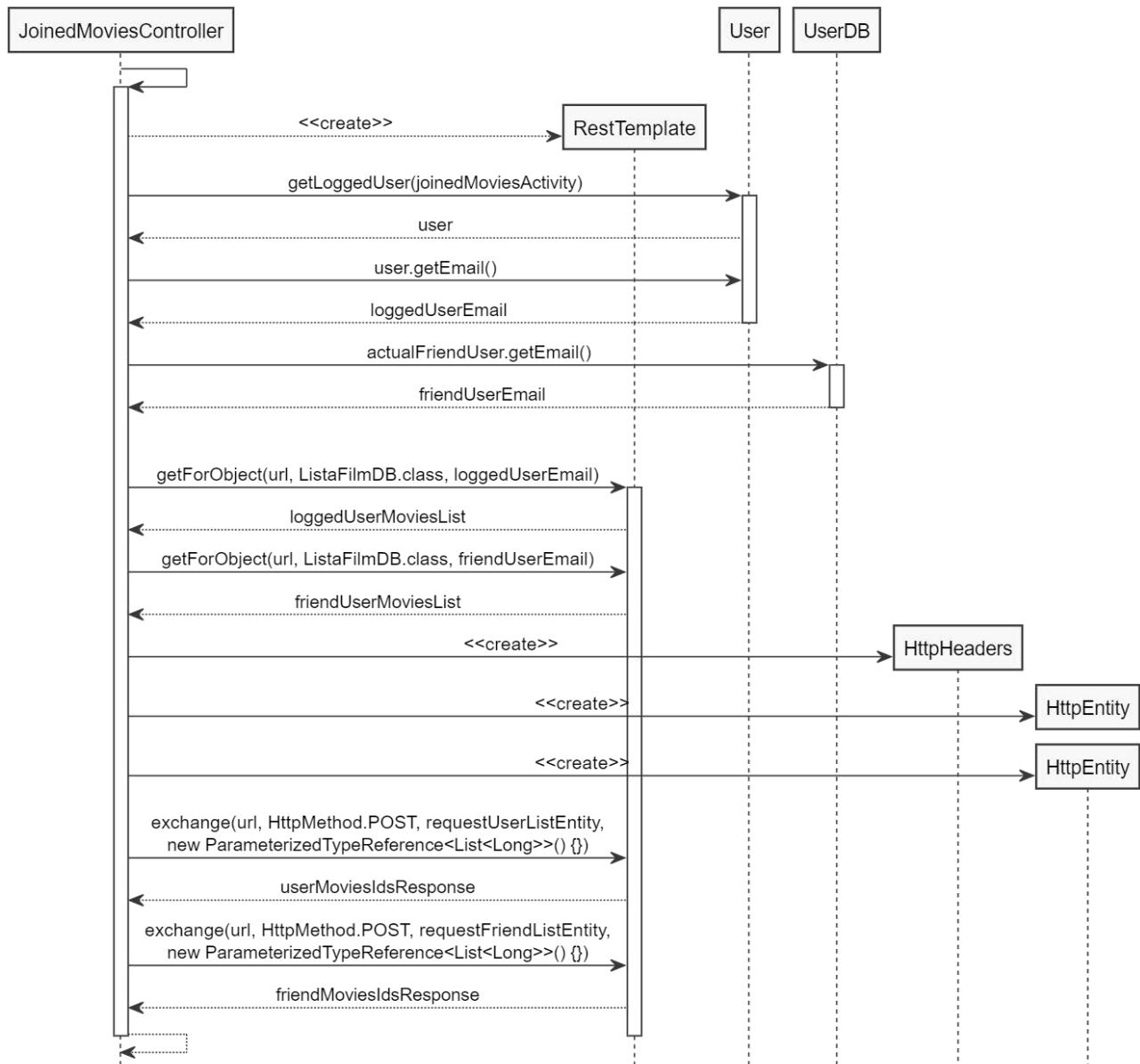
#FunzioneMostraAmico



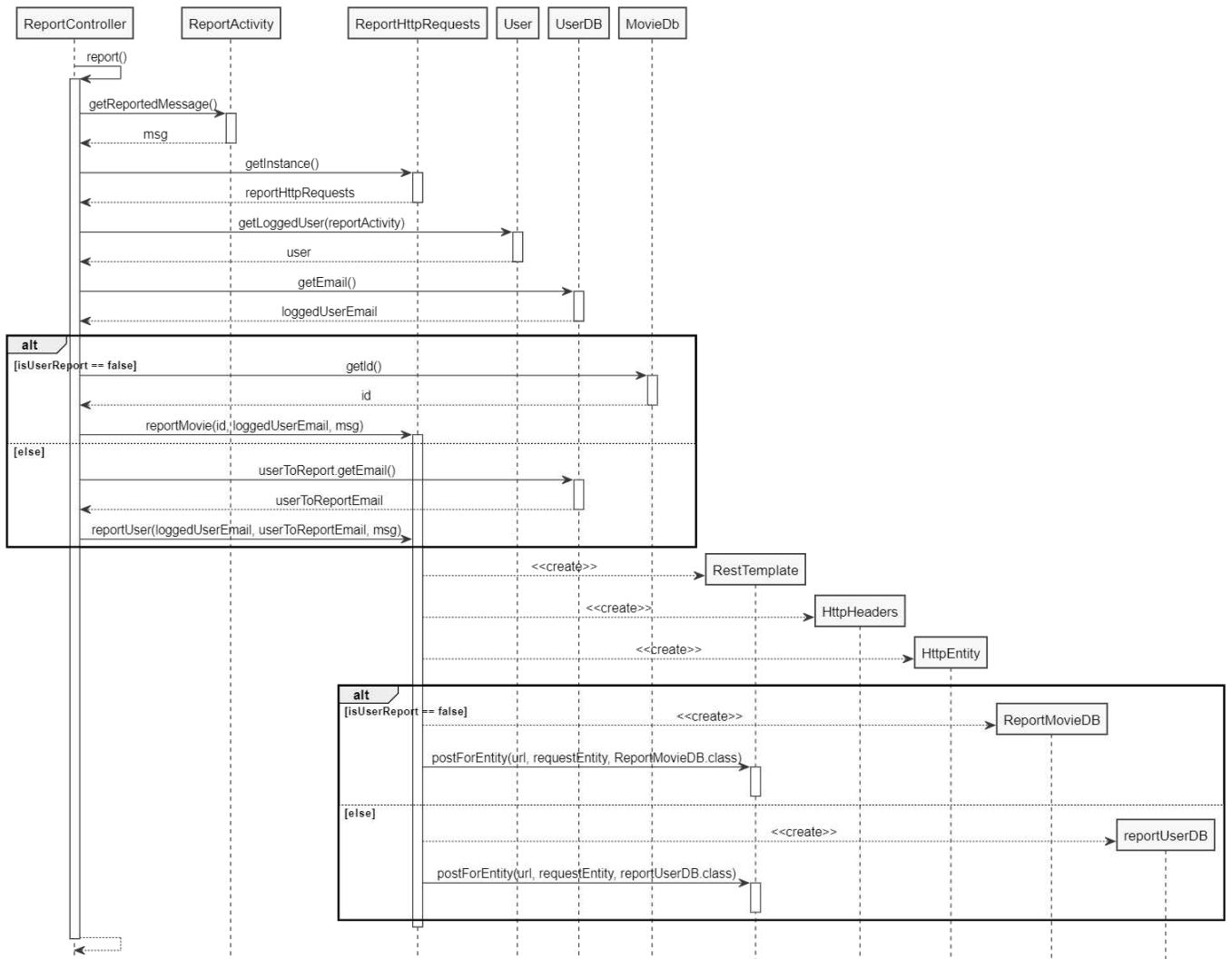
#FunzioneMostraUtente



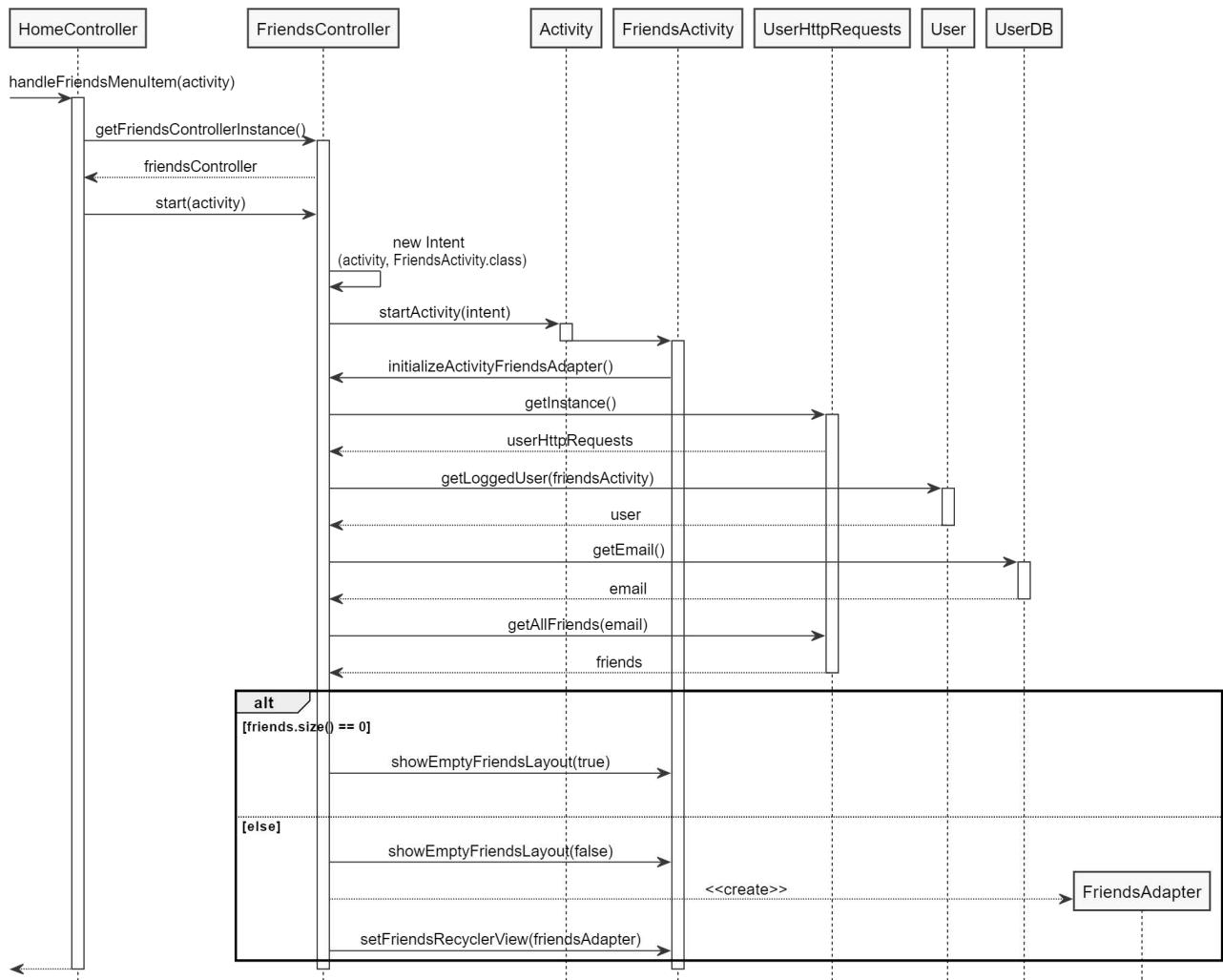
## #FunzioneRecuperaListeFilm



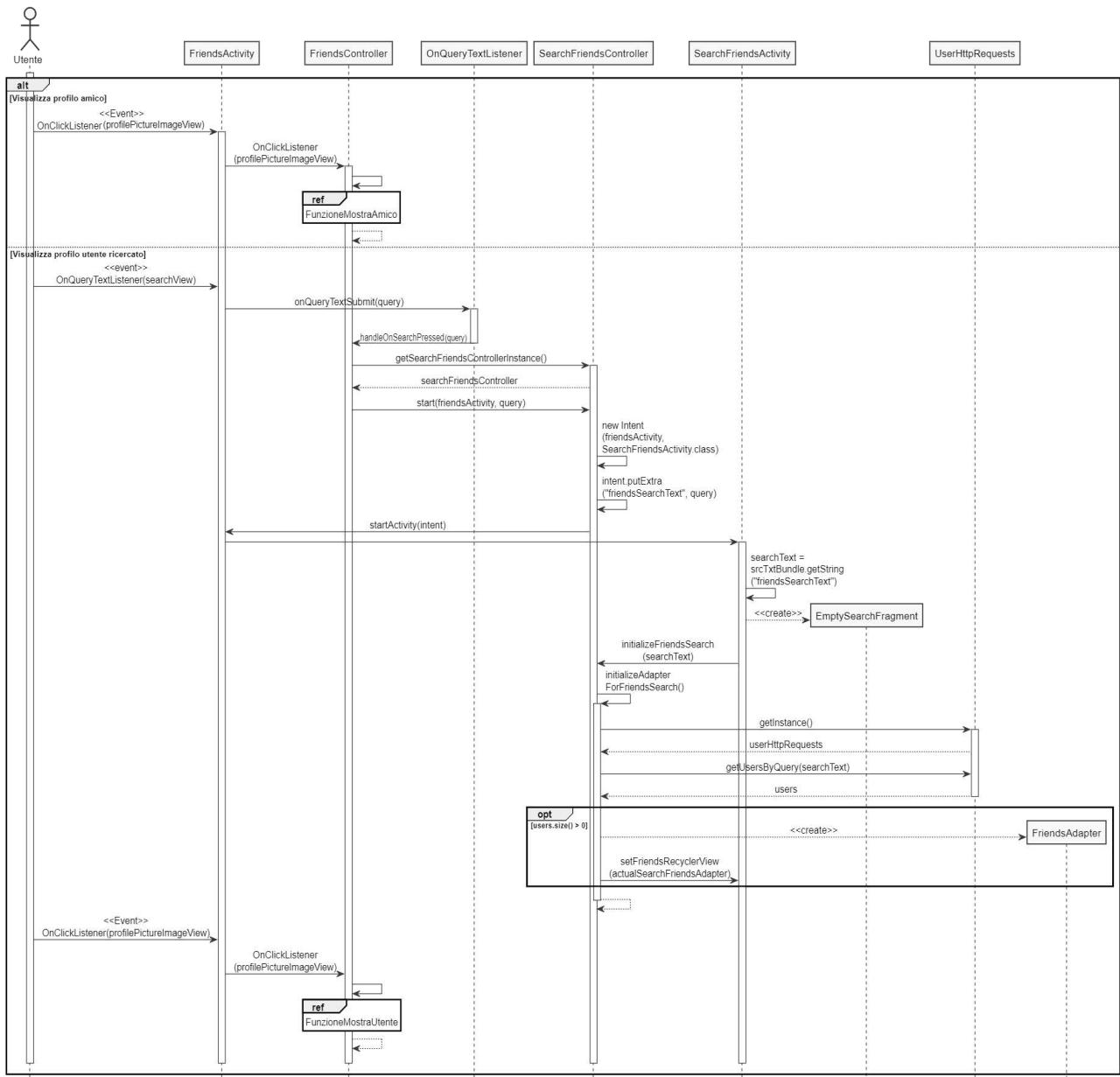
## #FunzioneSegnala



## #FunzioneVisualizzaAmici

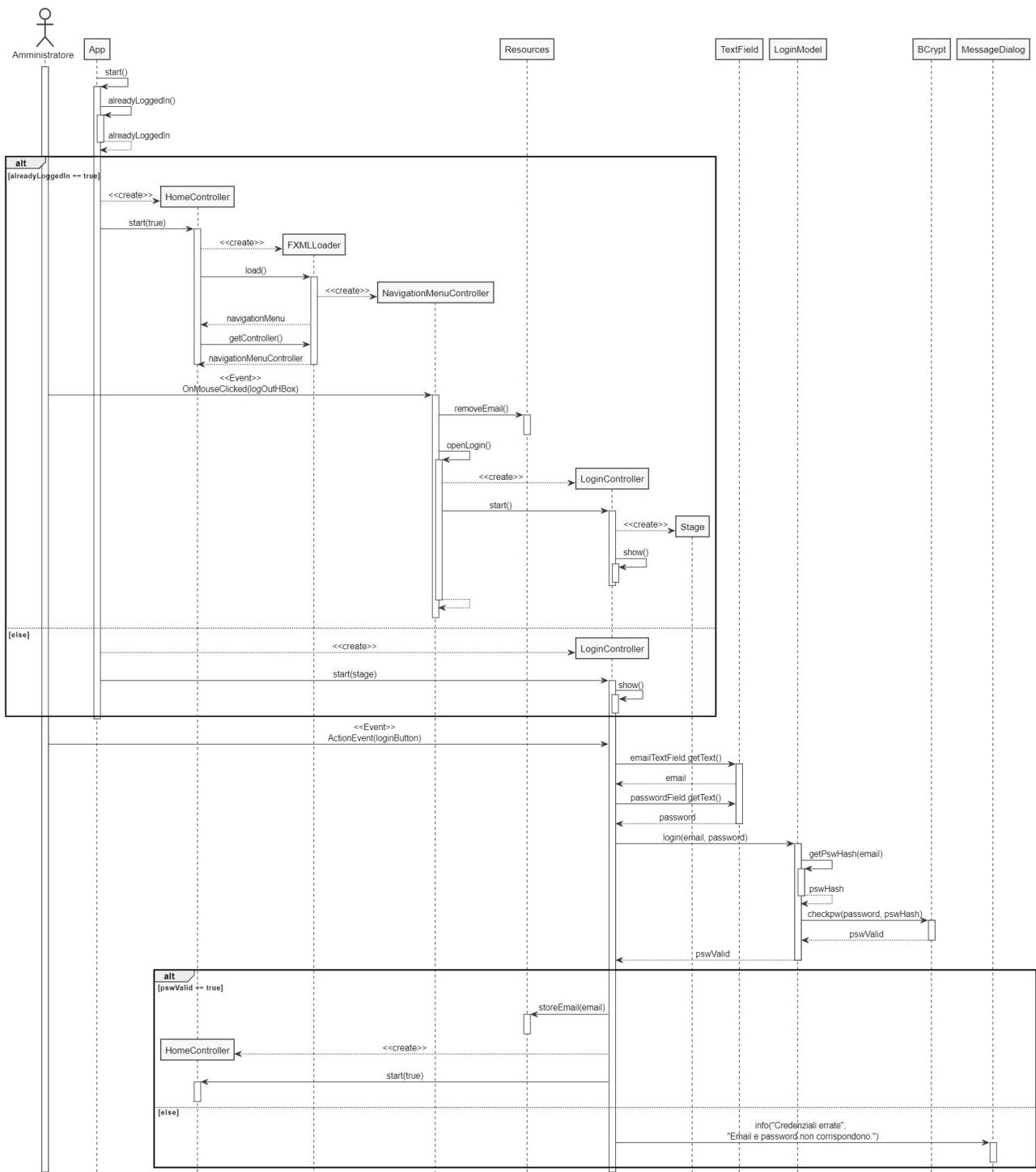


## #FunzioneVisualizzaUtenteDesiderato

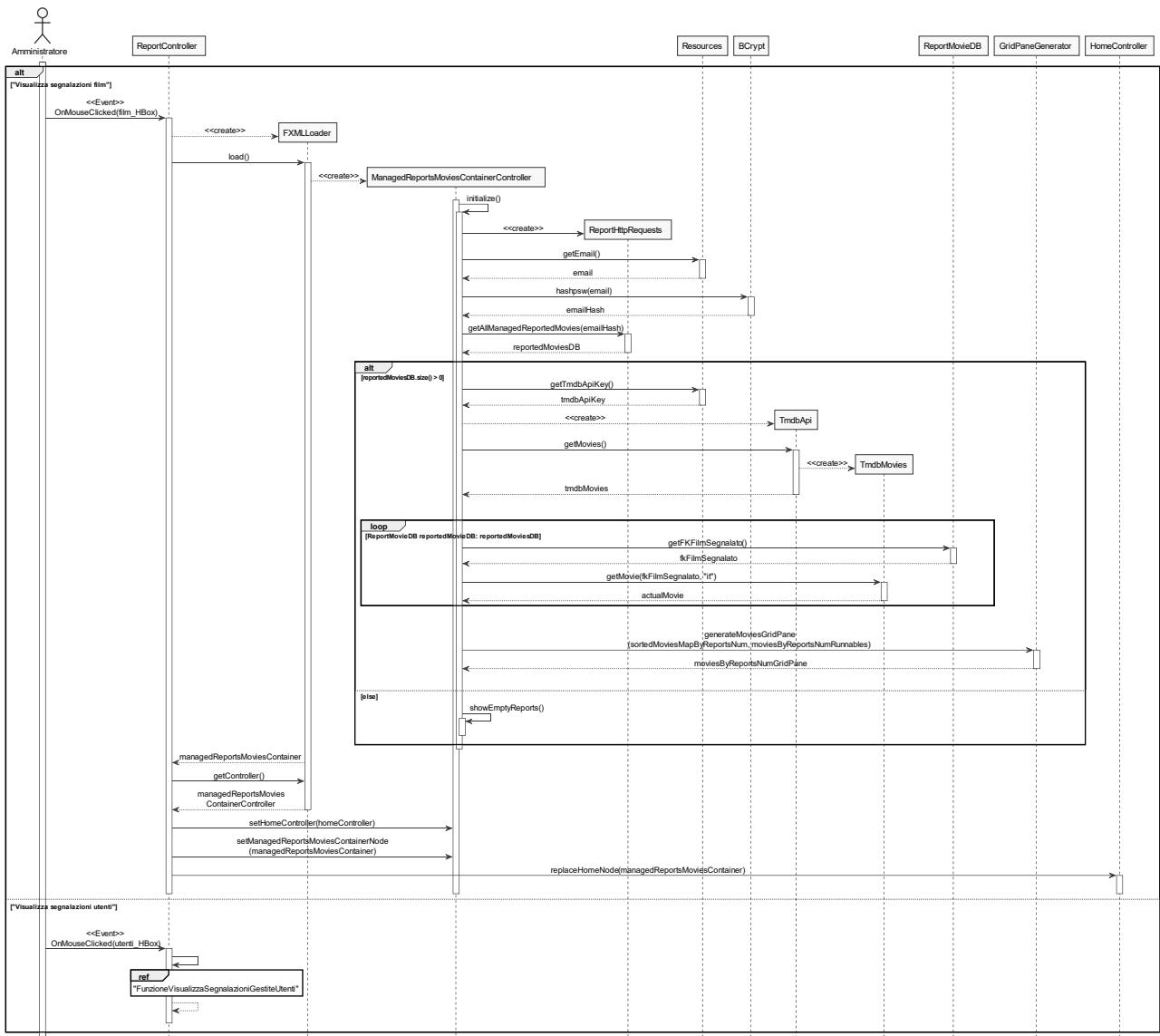


# Desktop

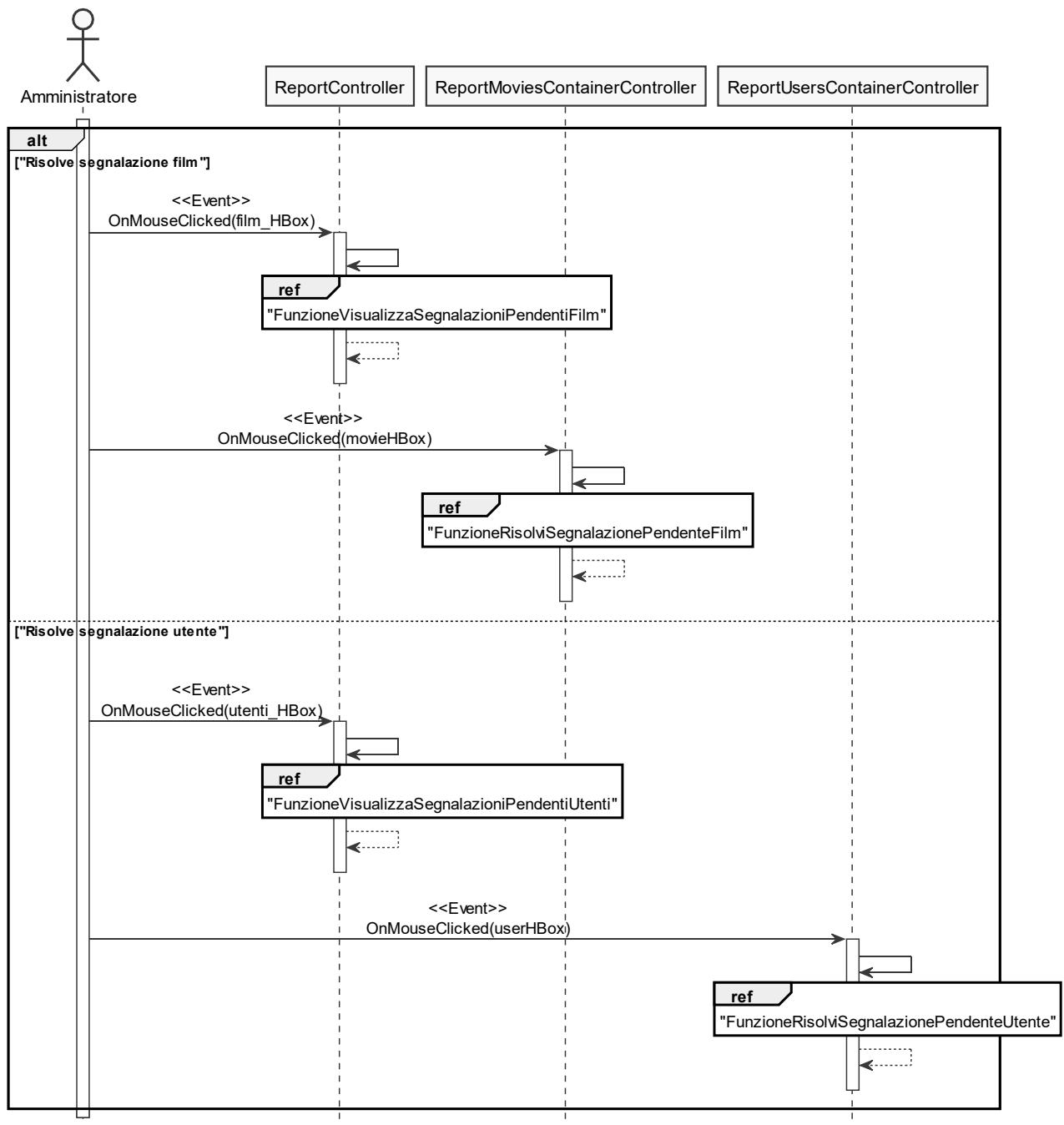
Amministratore effettua login



## Amministratore loggato visualizza segnalazioni gestite

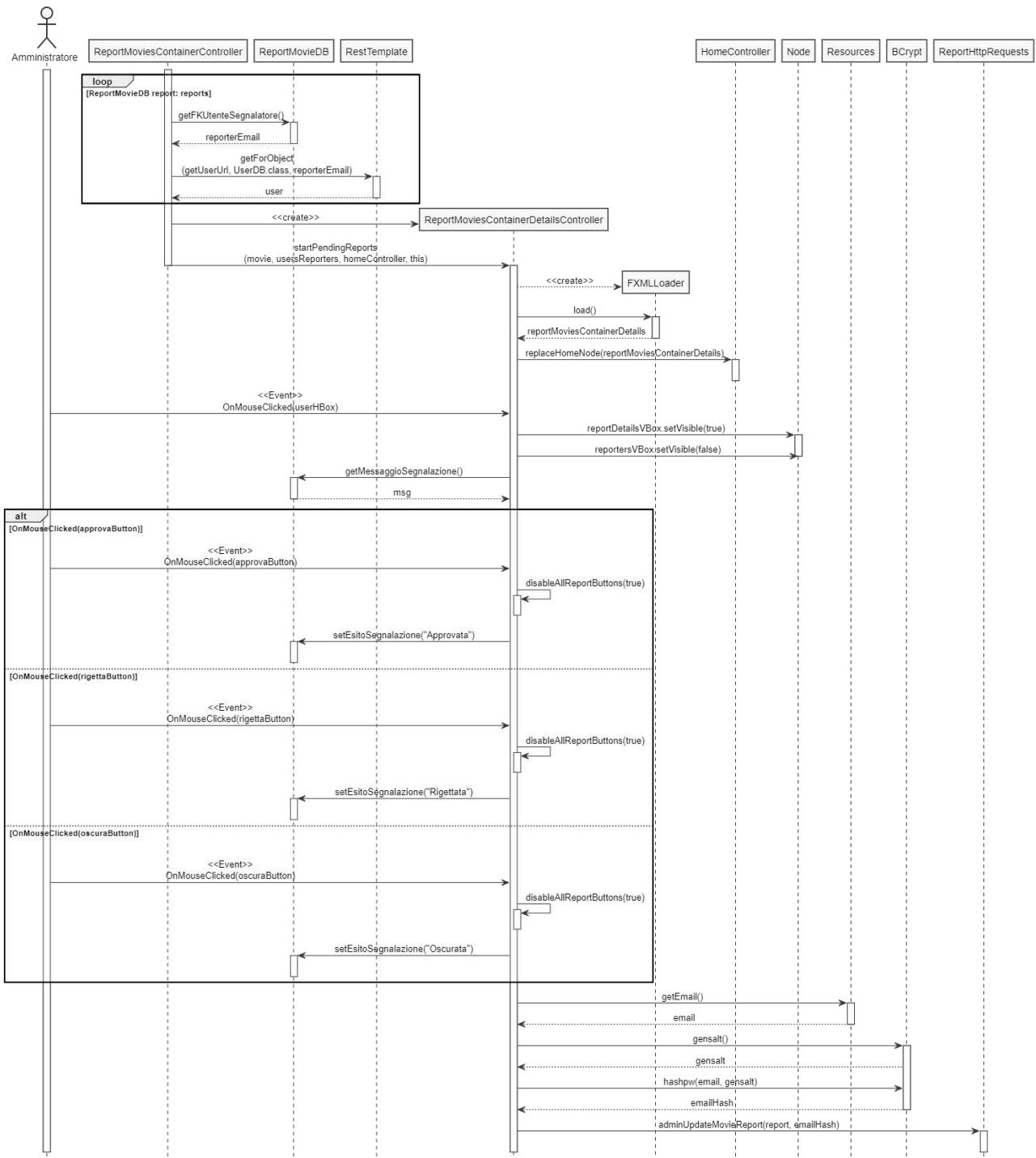


## Amministratore loggato risolve segnalazione pendente

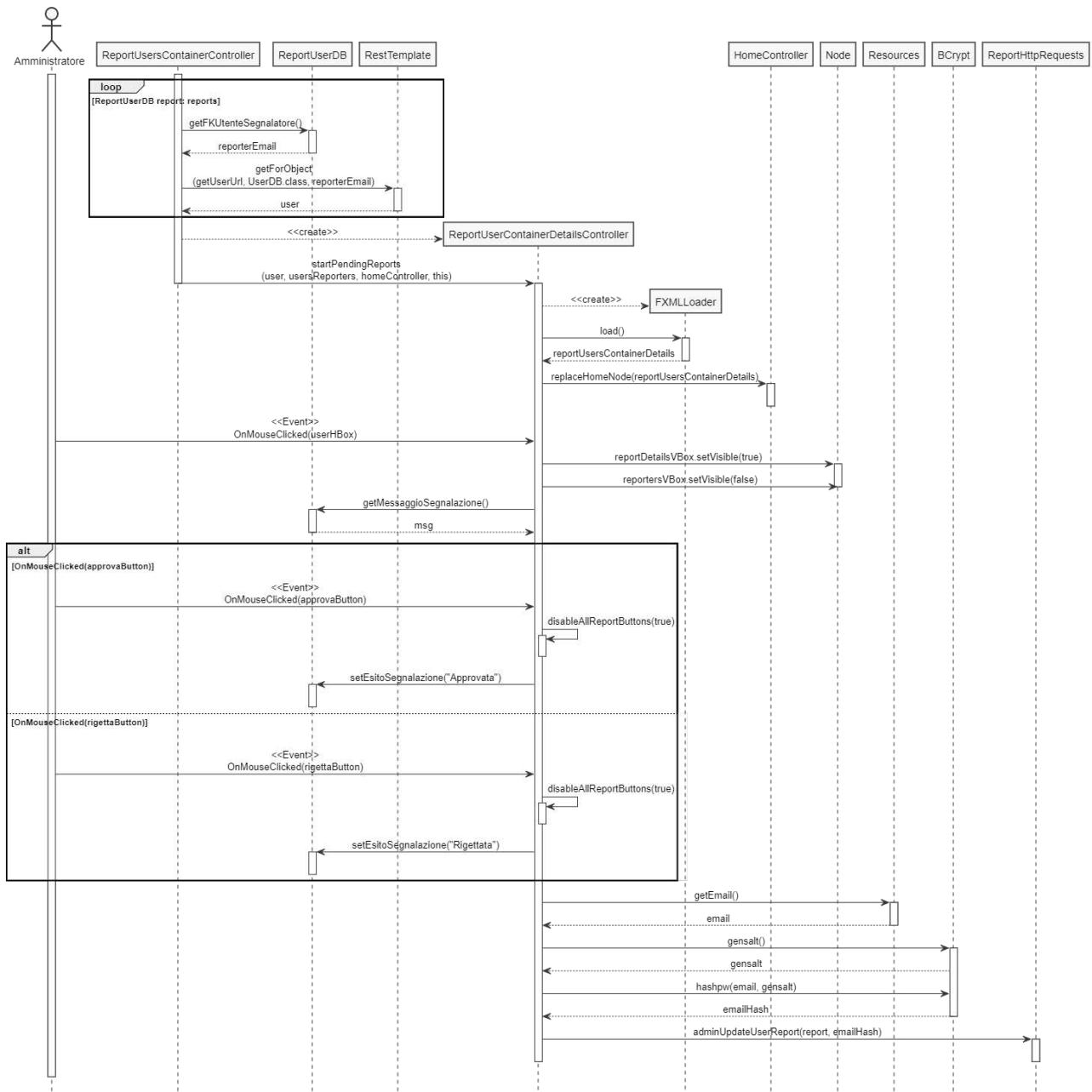


## Funzioni Sequence Diagrams Desktop

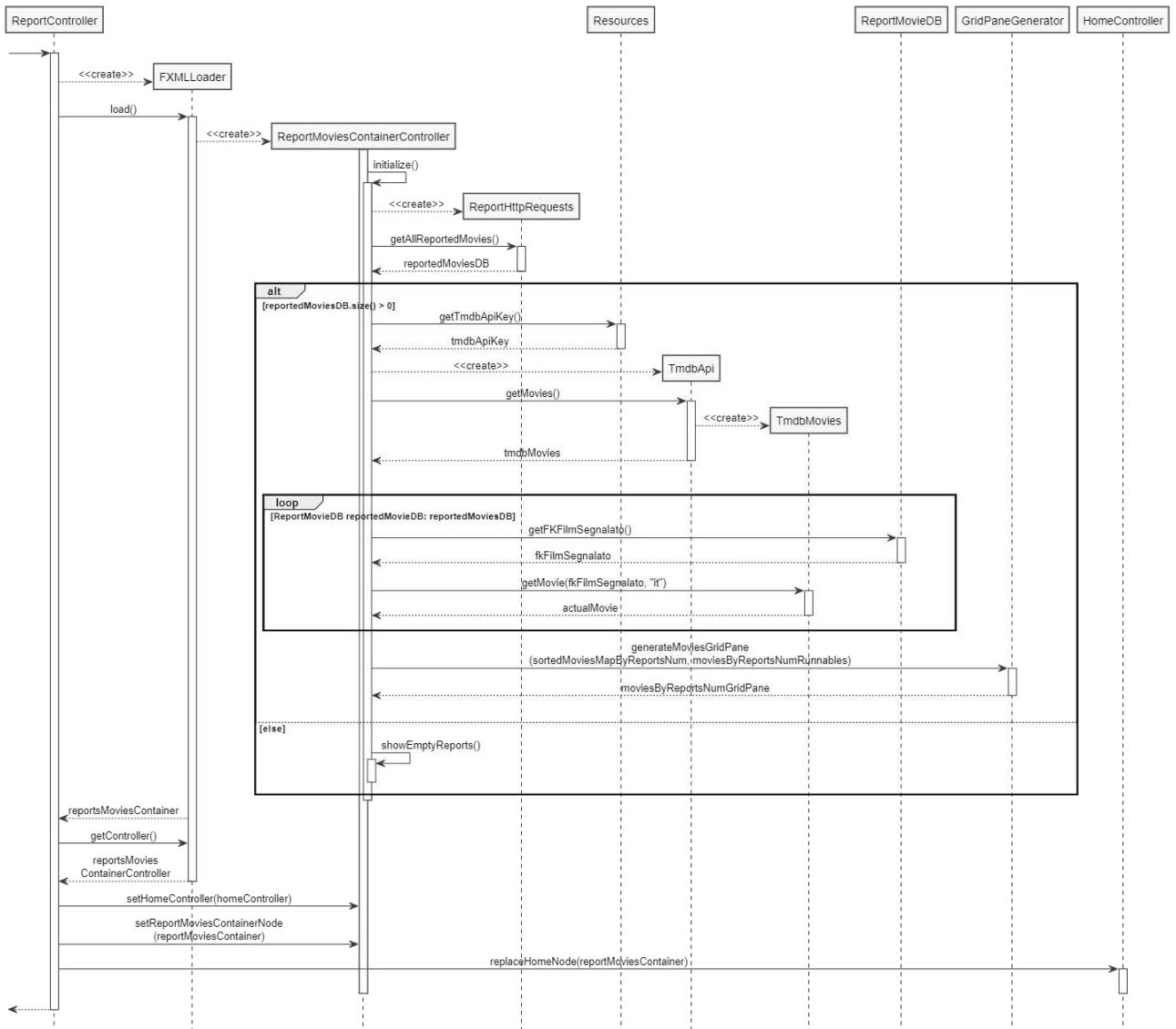
#FunzioneRisoviSegnalazionePendenteFilm



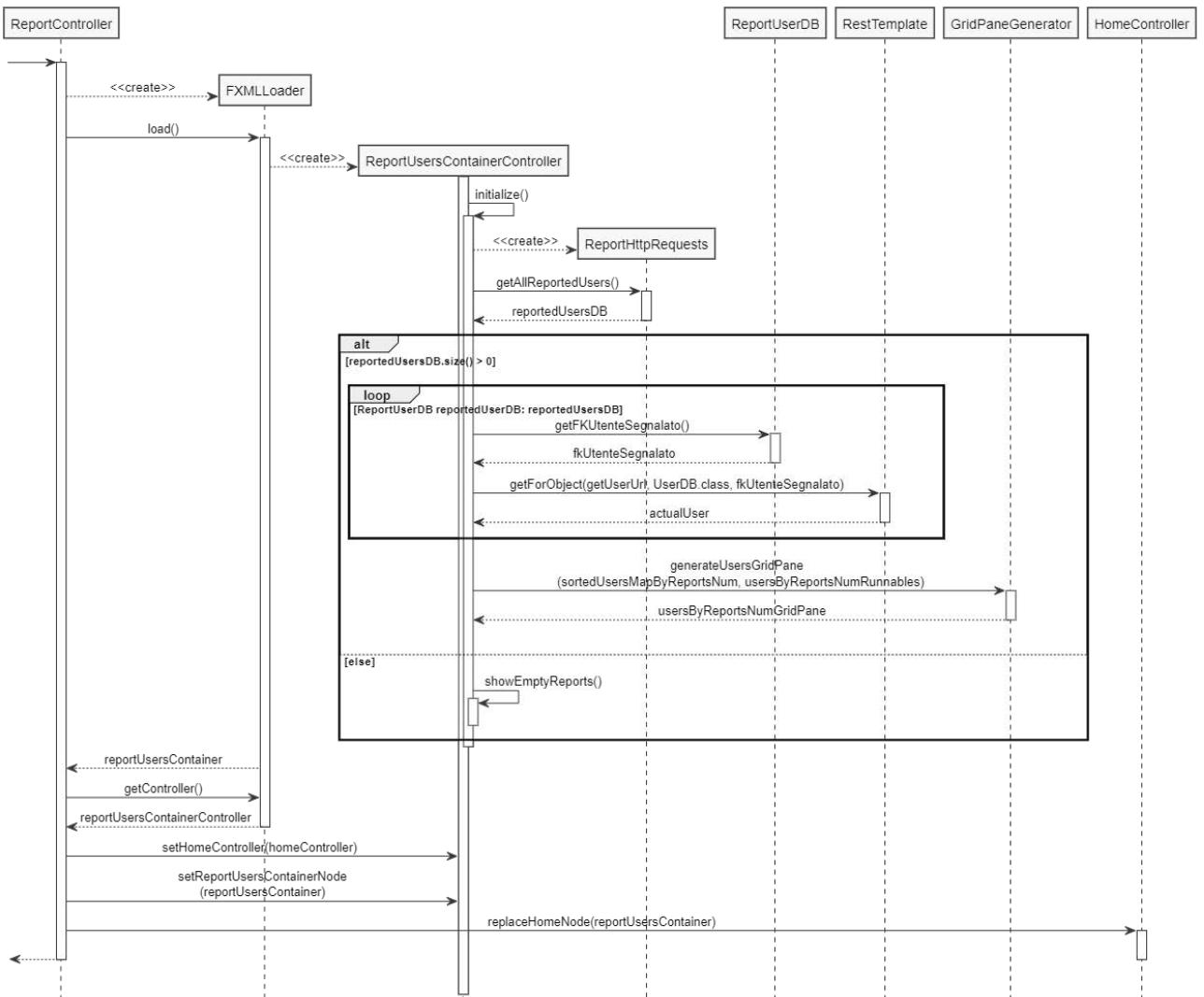
## #FunzioneRisolvisegnalazionePendenteUtente



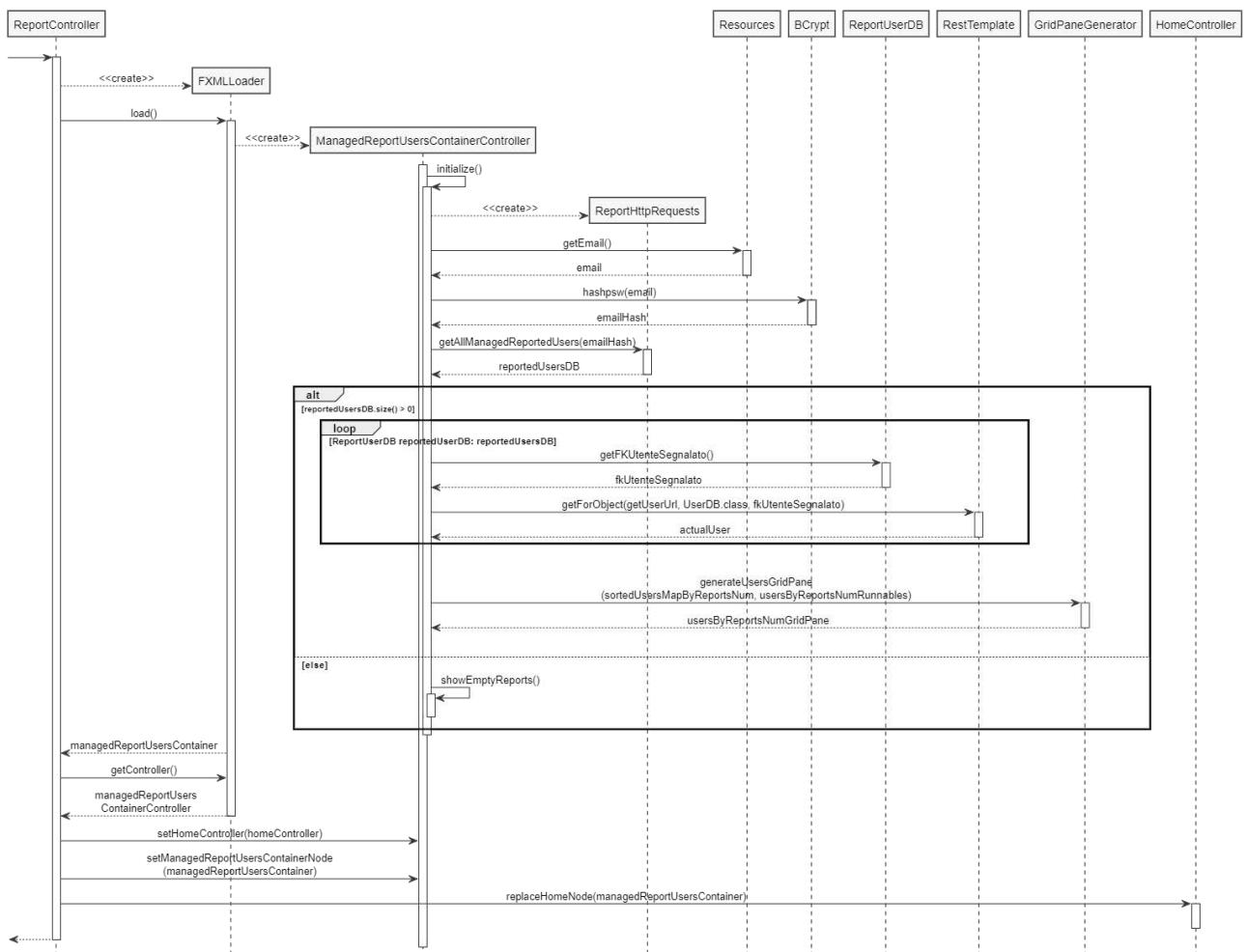
## #FunzioneVisualizzaSegnalazioniPendentiFilm



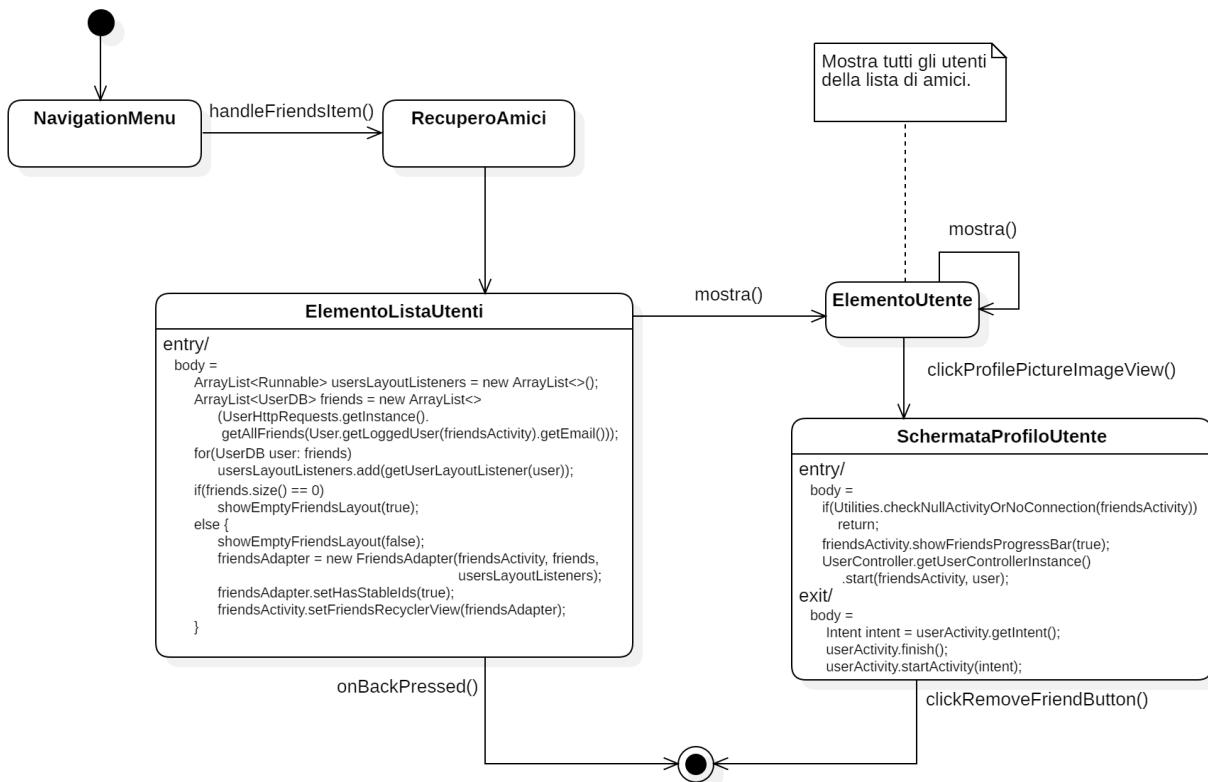
## #FunzioneVisualizzaSegnalazioniPendentiUtenti



## #FunzioneVisualizzaSegnalazioniGestiteUtenti



# Statechart Diagram di Design



# CRC Cards

## Mobile

Controller

<b>Nome Classe</b>	RegistrationController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento dello use case “Utente effettua registrazione”.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• S3Manager</li> <li>• UserDB</li> <li>• UserHttpRequests</li> <li>• RegistrationActivity</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	LoginController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento dello use case “Utente effettua login”.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeActivity</li> <li>• LoginActivity</li> <li>• RegistrationActivity</li> <li>• ResetPasswordActivity</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	ReportController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento degli use case “Utente loggato segnala utente” e “Utente loggato segnala film”.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• MovieDb</li> <li>• ReportHttpRequests</li> <li>• S3Manager</li> <li>• User</li> <li>• UserDB</li> <li>• ReportActivity</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	SettingsController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento delle impostazioni dell'applicativo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• SettingsActivity</li> </ul>

<b>Nome Classe</b>	UserController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento degli use case " <i>Utente loggato invia richiesta di amicizia</i> " e " <i>Utente loggato rimuove amico</i> ".
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• S3Manager</li> <li>• User</li> <li>• UserDB</li> <li>• UserHttpRequests</li> <li>• FriendsActivity</li> <li>• SearchFriendsActivity</li> <li>• UserActivity</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	PersonalProfileController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento per mostrare la schermata del proprio profilo e per modificare la foto di profilo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• S3Manager</li> <li>• User</li> <li>• PersonalProfileActivity</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	MainController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di avvio dell'app, se l'utente è loggato mostra la schermata home, altrimenti la schermata di login.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• EntryPoint</li> <li>• User</li> <li>• UserHttpRequests</li> </ul>

<b>Nome Classe</b>	ResetPasswordController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento per cambiare la password del proprio account, nel caso in cui non si riesce più ad accedere.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ResetPasswordActivity</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	HomeController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento del menu che permette di navigare tra le varie schermate dell'applicativo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• EntryPoint</li> <li>• User</li> <li>• FriendsActivity</li> <li>• HomeActivity</li> <li>• InformationActivity</li> <li>• MoviesListActivity</li> <li>• PersonalProfileActivity</li> <li>• SearchFriendsActivity</li> <li>• SearchMovieActivity</li> <li>• SettingsActivity</li> <li>• HomeStyleMovieAdapter</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	ShowDetailsMovieController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento per mostrare i dettagli di un film.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ListaFilmDB</li> <li>• User</li> <li>• HomeActivity</li> <li>• JoinedMoviesActivity</li> <li>• MoviesListActivity</li> <li>• SearchMovieActivity</li> <li>• ShowDetailsMovieActivity</li> <li>• ActorMovieAdapter</li> </ul>

<b>Nome Classe</b>	NotificationController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento delle notifiche.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ReportMovieDB</li> <li>• ReportUserDB</li> <li>• User</li> <li>• UserDB</li> <li>• UserHttpRequests</li> <li>• NotificationActivity</li> <li>• NotificationPagerAdapter</li> <li>• PendingFriendsRequestsAdapter</li> <li>• ReportNotificationAdapter</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	MoviesListsController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento degli use case “Utente loggato aggiunge film ad una lista” e “Utente loggato rimuove film da una lista”. Inoltre gestisce la logica per visualizzare le proprie liste.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ListaFilmDB</li> <li>• User</li> <li>• MoviesListActivity</li> <li>• HomeStyleMovieAdapter</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	SearchMovieController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento dello use case “Utente loggato cerca film”.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ListaFilmDB</li> <li>• User</li> <li>• SearchMovieActivity</li> <li>• SearchMovieAdapter</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	JoinedMoviesController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento dello use case “Utente loggato visualizza film in comune”.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ListaFilmDB</li> <li>• User</li> <li>• UserDB</li> <li>• JoinedMoviesActivity</li> <li>• HomeStyleMovieAdapter</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	SearchFriendsController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento per la ricerca di un qualsiasi utente (compresi gli amici).
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• UserDB</li> <li>• UserHttpRequests</li> <li>• SearchFriendsActivity</li> <li>• FriendsAdapter</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	FriendsController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento per la visualizzazione della lista dei propri amici
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• User</li> <li>• UserDB</li> <li>• UserHttpRequests</li> <li>• FriendsActivity</li> <li>• FriendsAdapter</li> <li>• Utilities</li> </ul>

## Activity

<b>Nome Classe</b>	MoviesListActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica che permettere ad un utente di visualizzare le proprie liste di film, da questa schermata è anche possibile rimuovere uno o più film da una lista.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• MoviesListsController</li> <li>• HomeController</li> <li>• User</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	RegistrationActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica che permettere ad un utente di registrarsi al sistema.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• RegistrationController</li> <li>• ConfirmRegistrationCodeFragment</li> <li>• RegistrationFragment</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	HomeActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica principale da cui è possibile cercare un film ed accedere alle altre schermate dell'applicativo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• NotificationController</li> <li>• SettingsController</li> <li>• User</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	JoinedMoviesActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica per permettere ad un utente di visualizzare la lista di film in comune con un amico.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• LoginController</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	ShowDetailsMovieActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica per permettere ad un utente di visualizzare i dettagli di un film, guardare il trailer o aggiungerlo ad (rimuoverlo da) una lista .
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ShowDetailsMovieController</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	ResetPasswordActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica per permettere ad un utente di cambiare la password nel caso sia stata dimenticata.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ResetPasswordController</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	NotificationActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica che permettere ad un utente di visualizzare le proprie notifiche.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• NotificationController</li> <li>• PendingFriendsNotificationFragment</li> <li>• ReportNotificationFragment</li> </ul>

<b>Nome Classe</b>	PersonalProfileActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica che permettere ad un utente di visualizzare il proprio profilo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• PersonalProfileController</li> <li>• HomeController</li> <li>• SettingsController</li> <li>• User</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	HomeActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica che mostra le informazioni sui progettisti e sviluppatori dell'applicativo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• SettingsController</li> <li>• User</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	FriendsActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica che permette ad un utente di visualizzare la lista dei propri amici.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• FriendsController</li> <li>• HomeController</li> <li>• SettingsController</li> <li>• User</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	SettingsActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica che permette all'utente di modificare le impostazioni dell'applicativo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• SettingsFragment</li> <li>• HomeController</li> <li>• SettingsController</li> <li>• User</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	InformationActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica che mostra le informazioni sui progettisti e sviluppatori dell'applicativo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• NotificationController</li> <li>• HomeController</li> <li>• SettingsController</li> <li>• User</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	UserActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica del profilo di un utente, permettendo l'aggiunta o la rimozione di un amico o eventualmente la segnalazione dell'utente.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• UserController</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	LoginActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica per permettere ad un utente di accedere al sistema, anche mediante l'utilizzo di social network esterni.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• LoginController</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	SearchFriendsActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica per permettere ad un utente di cercare un altro utente.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• SettingsController</li> <li>• SearchFriendsController</li> <li>• EmptySearchFragment</li> <li>• NotEmptyFriendsSearchFragment</li> <li>• User</li> <li>• Utilities</li> </ul>

<b>Nome Classe</b>	SearchMovieActivity
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la schermata grafica per permettere ad un utente di cercare un film, dal risultato della ricerca è possibile aggiungere un film ad una lista, eventualmente eliminarlo da una lista o segnalarlo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• SettingsController</li> <li>• SearchMovieController</li> <li>• EmptySearchFragment</li> <li>• NotEmptyMovieSearchFragment</li> <li>• User</li> <li>• Utilities</li> </ul>

## Fragment

<b>Nome Classe</b>	EmptySearchFragment
<b>Superclasse</b>	Fragment
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Modulo grafico che mostra all'utente che una ricerca non ha prodotto risultati.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	NotEmptyFriendsSearchFragment
<b>Superclasse</b>	Fragment
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Modulo grafico che mostra all'utente una lista di utenti, questa lista è il risultato di una ricerca.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	SettingsFragment
<b>Superclasse</b>	Fragment
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Modulo grafico che contiene le diverse opzioni che possono essere settate dalle impostazioni.
<b>Collaboratori</b>	SettingsActivity

<b>Nome Classe</b>	PendingFriendsNotificationFragment
<b>Superclasse</b>	Fragment
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Modulo grafico che contiene una lista di notifiche riguardanti le richieste d'amicizia.
<b>Collaboratori</b>	NotificationActivity

<b>Nome Classe</b>	ReportNotificationFragment
<b>Superclasse</b>	Fragment
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Modulo grafico che contiene una lista di notifiche riguardanti le segnalazioni fatte dall'utente.
<b>Collaboratori</b>	NotificationActivity

<b>Nome Classe</b>	RegistrationFragment
<b>Superclasse</b>	Fragment
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Modulo grafico contenente un form che permette all'utente d'inserire le proprie generalità per registrarsi al sistema, nel caso di registrazione mediante un social network esterno il form viene mostrato precompilato. Da questo componente grafico è anche possibile cambiare la foto di profilo
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	NotEmptyMovieSearchFragment
<b>Superclasse</b>	Fragment
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Modulo grafico che mostra all'utente una lista di film, questa lista è il risultato di una ricerca.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	ConfirmRegistrationCodeFragment
<b>Superclasse</b>	Fragment
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Modulo grafico che permette all'utente d'inserire il codice per confermare la registrazione o eventualmente chiedere un nuovo codice.
<b>Collaboratori</b>	Nessuno

## Adapter

<b>Nome Classe</b>	HomeStyleMovieAdapter
<b>Superclasse</b>	Adapter<ViewHolder>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Elemento grafico che permette la visualizzazione del singolo film, sia dalla home che dalle proprie liste.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• MoviesListsController</li> <li>• MovieHolder</li> </ul>

<b>Nome Classe</b>	FriendsAdapter
<b>Superclasse</b>	Adapter<ViewHolder>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Elemento grafico che permette la visualizzazione del singolo utente, che eventualmente può essere un amico.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• FriendsController</li> <li>• S3Manager</li> <li>• UserDB</li> <li>• UserHolder</li> </ul>

<b>Nome Classe</b>	PendingFriendsRequestsAdapter
<b>Superclasse</b>	Adapter<ViewHolder>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Elemento grafico che permette la visualizzazione della singola notifica di una richiesta d'amicizia.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• NotificationController</li> <li>• NotificationActivity</li> <li>• S3Manager</li> <li>• UserDB</li> <li>• FriendNotificationHolder</li> </ul>

<b>Nome Classe</b>	ReportNotificationAdapter
<b>Superclasse</b>	Adapter<ViewHolder>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Elemento grafico che permette la visualizzazione della singola notifica dell'esito di una segnalazione.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• NotificationController</li> <li>• ReportNotificationHolder</li> </ul>

<b>Nome Classe</b>	SearchMovieAdapter
<b>Superclasse</b>	Adapter<ViewHolder>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Elemento grafico che permette la visualizzazione del singolo film, risultato di una ricerca.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• SearchMovieController</li> <li>• MovieHolder</li> </ul>

<b>Nome Classe</b>	ActorMovieAdapter
<b>Superclasse</b>	Adapter<ViewHolder>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Elemento grafico che permette la visualizzazione del singolo attore.
<b>Collaboratori</b>	ActorHolder

<b>Nome Classe</b>	NotificationPagerAdapter
<b>Superclasse</b>	FragmentStateAdapter
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Elemento grafico che permette la visualizzazione del singolo attore.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• PendingFriendsNotificationFragment</li> <li>• ReportNotificationFragment</li> </ul>

## ViewHolder

<b>Nome Classe</b>	HomeStyleMovieAdapter MovieHolder
<b>Superclasse</b>	ViewHolder
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Memorizza i riferimenti delle componenti dell'elemento grafico creato da HomeStyleMovieAdapter.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	UserHolder
<b>Superclasse</b>	ViewHolder
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Memorizza i riferimenti delle componenti dell'elemento grafico creato da FriendsAdapter.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	FriendNotificationHolder
<b>Superclasse</b>	ViewHolder
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Memorizza i riferimenti delle componenti dell'elemento grafico creato da PendingFriendsRequestsAdapter.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	ReportNotificationHolder
<b>Superclasse</b>	ViewHolder
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Memorizza i riferimenti delle componenti dell'elemento grafico creato da ReportNotificationAdapter.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	SearchMovieAdapter MovieHolder
<b>Superclasse</b>	ViewHolder
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Memorizza i riferimenti delle componenti dell'elemento grafico creato da SearchMovieAdapter.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	ActorHolder
<b>Superclasse</b>	ViewHolder
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Memorizza i riferimenti delle componenti dell'elemento grafico creato da ActorMovieAdapter.
<b>Collaboratori</b>	Nessuno

## Model

<b>Nome Classe</b>	User
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Memorizza le informazioni dell'utente attualmente loggato.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• SettingsController</li> <li>• UserDB</li> <li>• S3Manager</li> </ul>

<b>Nome Classe</b>	ListaFilmDB
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappare il convenuto di un Json come lista di film, per poter comunicare col server
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	UserDB
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappare il convenuto di un Json come utente, per poter comunicare col server.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	ReportMovieDB
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappare il convenuto di un Json come segnalazione riguardante un film, per poter comunicare col server.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	ReportUserDB
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappare il convenuto di un Json come segnalazione riguardante un utente, per poter comunicare col server.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	ReportHttpRequests
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Invia richieste al server per poter effettuare operazioni riguardante le segnalazioni, ad esempio inviare una nuova segnalazione o recuperare un insieme di segnalazioni.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ReportMovieDB</li> <li>• ReportUserDB</li> </ul>

<b>Nome Classe</b>	UserHttpRequests
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Invia richieste al server per poter effettuare operazioni riguardanti l'utente, ad esempio creare un nuovo utente o recuperare un insieme di utenti.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	S3Manager
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Archiviare e recuperare le immagini di profilo degli utenti dal server.
<b>Collaboratori</b>	Utilities

## Punto d'ingresso dell'applicativo

<b>Nome Classe</b>	EntryPoint
<b>Superclasse</b>	AppCompatActivity
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	All'avvio dell'applicazione mostra il logo dell'app ed inizializza la configurazione del provider che gestisce gli accessi e le registrazioni al sistema.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• MainController</li> <li>• Utilities</li> </ul>

## Util

<b>Nome Classe</b>	Utilities
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Metodi di utilità che vengono utilizzati in buona parte del progetto, come la verifica della disponibilità della connessione, o la possibilità di visualizzare dei messaggi a comparsa.
<b>Collaboratori</b>	Nessuno

## Desktop

### Controller

<b>Nome Classe Astratta</b>	Controller
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• LoginController</li> <li>• ManagedReportsMoviesContainerController</li> <li>• ManagedReportUsersContainerController</li> <li>• NavigationMenuController</li> <li>• ReportController</li> <li>• ReportMovieItemController</li> <li>• ReportMoviesContainerController</li> <li>• ReportMoviesContainerDetailsController</li> <li>• ReportUserContainerDetailsController</li> <li>• ReportUserItemController</li> <li>• ReportUsersContainerController</li> </ul>
<b>Responsabilità</b>	Definisce la firma del metodo per inizializzare i componenti dell'interfaccia grafica.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	ReportController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica della schermata da cui è possibile gestire o visualizzare le segnalazioni gestite da un amministratore.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• App</li> <li>• HomeController</li> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	HomeController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica della schermata principale, in questa schermata verranno inserite le altre schermate che verranno aperte ed inserite come nodi figli.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• App</li> <li>• ReportController</li> <li>• MessageDialog</li> <li>• FXMLUtils</li> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ReportUserContainerDetailsController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica della visualizzazione della schermata che mostra l'utente segnalato con la lista di utenti che l'hanno segnalato, relativa ad una segnalazione pendente oppure gestita.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• App</li> <li>• ReportUsersContainerController</li> <li>• ManagedReportUsersContainerController</li> <li>• ReportHttpRequests</li> <li>• ReportUserDB</li> <li>• UserDB</li> <li>• S3Manager</li> <li>• GridPaneGenerator</li> <li>• MessageDialog</li> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ReportMoviesContainerDetailsController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la parte della logica dello use case <i>“Amministratore loggato visualizza segnalazioni gestite”</i> , relativa ai dettagli della segnalazione di un film, con la lista di utenti che l'hanno segnalato, relativa ad una segnalazione pendente oppure gestita.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• App</li> <li>• ReportMoviesContainerController</li> <li>• ManagedReportsMoviesContainerController</li> <li>• ReportHttpRequests</li> <li>• ReportMovieDB</li> <li>• UserDB</li> <li>• GridPaneGenerator</li> <li>• MessageDialog</li> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ReportUserItemController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica della visualizzazione dell'elemento grafico che permette la visualizzazione dell'utente con le relative informazioni ed il numero di segnalazioni ricevute.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• UserDB</li> <li>• S3Manager</li> </ul>

<b>Nome Classe</b>	LoginController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di funzionamento dello use case “Amministratore effettua login”.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• LoginModel</li> <li>• MessageDialog</li> <li>• FXMLUtils</li> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ReportMovieItemController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica della visualizzazione dell’elemento grafico che permette la visualizzazione del film con le relative informazioni ed il numero di segnalazioni ricevute.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ManagedReportsMoviesContainerController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce parte della logica dello use case “Amministratore loggato visualizza segnalazioni gestite”, relativa ai film.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• ReportHttpRequests</li> <li>• ReportMovieDB</li> <li>• UserDB</li> <li>• GridPaneGenerator</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ReportMoviesContainerController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce parte della logica dello use case “Amministratore loggato risolve segnalazione pendente”, relativa ai film.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• ReportHttpRequests</li> <li>• ReportMovieDB</li> <li>• UserDB</li> <li>• GridPaneGenerator</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ManagedReportUsersContainerController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce parte della logica dello use case “Amministratore loggato visualizza segnalazioni gestite”, relativa agli utenti.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• ReportHttpRequests</li> <li>• ReportMovieDB</li> <li>• UserDB</li> <li>• GridPaneGenerator</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	NavigationMenuController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la navigazione tra le varie schermate dell'applicativo e gestisce la logica del logout di un amministratore.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• App</li> <li>• HomeController</li> <li>• LoginModel</li> <li>• UserDB</li> <li>• S3Manager</li> <li>• MessageDialog</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ReportUsersContainerController
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce parte della logica dello use case “Amministratore loggato risolve segnalazione pendente”, relativa agli utenti.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• ReportHttpRequests</li> <li>• ReportUserDB</li> <li>• UserDB</li> <li>• GridPaneGenerator</li> <li>• Resources</li> </ul>

## Model

<b>Nome Classe</b>	S3Manager
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Archiviare e recuperare le immagini di profilo degli amministratori dal server.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	LoginModel
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Invia richieste al server, per gestire gli accessi degli amministratori all'applicativo.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• UserDB</li> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ReportHttpRequests
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Invia richieste al server, per la gestione delle segnalazioni, sia per gli utenti che per i film.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ReportUserDB</li> <li>• ReportMovieDB</li> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	ReportMovieDB
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappare il convenuto di un Json come segnalazione riguardante un film, per poter comunicare col server.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	ReportUserDB
<b>Superclasse</b>	Controller
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappare il convenuto di un Json come segnalazione riguardante un utente, per poter comunicare col server.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	UserDB
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappare il convenuto di un Json come utente, per poter comunicare col server.
<b>Collaboratori</b>	Nessuno

## View

<b>Nome Classe</b>	GridPaneGenerator
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce la logica di creazione delle griglie che conterranno gli elementi grafici riguardanti le segnalazioni.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• ReportMovieItemController</li> <li>• ReportUserItemController</li> <li>• ReportMovieDB</li> <li>• ReportUserDB</li> <li>• UserDB</li> <li>• NameResources</li> <li>• Resources</li> </ul>

<b>Nome Classe</b>	MessageDialog
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce i messaggi a comparsa, per notificare l'utente.
<b>Collaboratori</b>	Nessuno

## Utils

<b>Nome Classe</b>	FXMLUtils
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce il caricamento dei file FXML.
<b>Collaboratori</b>	App

<b>Nome Classe</b>	Resources
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce un file XML contenente le varie risorse utilizzate nel progetto.
<b>Collaboratori</b>	MessageDialog

<b>Nome Enumerazione</b>	NameResources
<b>Valori</b>	<ul style="list-style-type: none"> <li>• CSS_CLASS_INPUT_ERROR</li> <li>• DIRECTORY_FXML</li> <li>• LOGIN_LAYOUT</li> <li>• HOME_LAYOUT</li> <li>• NAVIGATION_MENU_LAYOUT</li> <li>• REPORT_MOVIE_ITEM_LAYOUT</li> <li>• REPORT_USER_ITEM_LAYOUT</li> <li>• REPORT_USERS_CONTAINER_DETAILS_LAYOUT</li> <li>• REPORT_MOVIES_CONTAINER_DETAILS_LAYOUT</li> <li>• ADMIN_PATH</li> <li>• REPORT_PATH</li> <li>• GET_PSW_HASH_PATH</li> <li>• EMAIL_ALREADY_EXISTS_PATH</li> <li>• GET_BASIC_ADMIN_INFO_PATH</li> <li>• UPDATE_MOVIE_REPORT_PATH</li> <li>• UPDATE_USER_REPORT_PATH</li> <li>• FIRST_PATH_IMAGE</li> <li>• FIRST_PATH_IMAGE_ORIGINAL</li> </ul>
<b>Responsabilità</b>	Va a mappare le chiavi utilizzate nel file delle risorse mediante la quale queste vengono recuperate.
<b>Collaboratori</b>	Nessuno

## Punto d'ingresso dell'applicativo

<b>Nome Classe</b>	App
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Classe da cui viene avviato l'applicativo desktop, in essa si verifica se l'amministratore è autenticato per stabilire se avviare la schermata di accesso o se avviare la schermata principale.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• HomeController</li> <li>• LoginController</li> <li>• LoginModel</li> <li>• Resources</li> </ul>

## Server

### Controller

<b>Nome Classe</b>	ServerSpringSegnalazioneController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce le richieste dei client riguardanti gli use case <i>"Utente loggato segnala utente"</i> , <i>"Utente loggato segnala film"</i> , <i>"Amministratore Loggato visualizza segnalazioni gestite"</i> , <i>"Amministratore Loggato risolve segnalazione pendente"</i> .
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• Dao</li> <li>• SegnalazioneFilmDao</li> <li>• SegnalazioneUtenteDao</li> <li>• SegnalazioneFilmEntity</li> <li>• SegnalazioneUtenteEntity</li> </ul>

<b>Nome Classe</b>	ServerSpringAmministratoriController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce le richieste dei client riguardanti la gestione degli accessi al sistema e del recupero delle informazioni degli amministratori.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• Dao</li> <li>• AdministratorDao</li> <li>• CredenzialiAmministratoriEntity</li> <li>• UtenteEntity</li> </ul>

<b>Nome Classe</b>	ServerSpringUserController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce le richieste dei client riguardanti la gestione degli utenti e della relazione di amicizia fa parte degli use case <i>"Utente loggato invia richiesta di amicizia"</i> e <i>"Utente loggato rimuove amico"</i> .
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• Dao</li> <li>• UserDao</li> <li>• UtenteEntity</li> <li>• TipologiaUtente</li> </ul>

<b>Nome Classe</b>	ServerSpringListaFilmController
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce le richieste dei client riguardanti la gestione delle liste di film degli utenti, compito menzionato negli use case “Utente loggato aggiunge film ad una lista” e “Utente loggato rimuove film da una lista”.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• Dao</li> <li>• ListaFilmDao</li> <li>• ListaFilmEntity</li> </ul>

Dao

<b>Nome Interfaccia</b>	Dao<E, I>
<b>Superclasse</b>	Nessuna
<b>Sottoclassi che implementano</b>	<ul style="list-style-type: none"> <li>• ListaFilmDao</li> <li>• SegnalazioneFilmDao</li> <li>• SegnalazioneUtenteDao</li> <li>• AdministratorDao</li> <li>• UserDao</li> </ul>
<b>Responsabilità</b>	Definisce le firme dei metodi mediante la quale si andrà ad interagire col livello di persistenza dei dati.
<b>Collaboratori</b>	Nessuno
<b>Parametri di tipo</b>	<ul style="list-style-type: none"> <li>▪ <b>E</b>: rappresenta la classe che andrà a mappare l'entità sugli oggetti del linguaggio</li> <li>▪ <b>I</b>: tipo di identificativo utilizzato</li> </ul>

<b>Nome Interfaccia</b>	ListaFilmDao<E, I>
<b>Superclasse</b>	Dao<E, I>
<b>Sottoclassi che implementano</b>	ListaFilmDaoImplementation
<b>Responsabilità</b>	Definisce le firme dei metodi per la gestione del livello di persistenza per le liste di film degli utenti.
<b>Collaboratori</b>	Nessuno

<b>Nome Interfaccia</b>	SegnalazioneFilmDao<E, I>
<b>Superclasse</b>	Dao<E, I>
<b>Sottoclassi che implementano</b>	SegnalazioneFilmDaoImplementation
<b>Responsabilità</b>	Definisce le firme dei metodi per la gestione del livello di persistenza per le segnalazioni riguardanti i film.
<b>Collaboratori</b>	Nessuno

<b>Nome Interfaccia</b>	SegnalazioneUtenteDao<E, I>
<b>Superclasse</b>	Dao<E, I>
<b>Sottoclassi che implementano</b>	SegnalazioneFilmDaoImplementation
<b>Responsabilità</b>	Definisce le firme dei metodi per la gestione del livello di persistenza per le segnalazioni riguardanti gli utenti.
<b>Collaboratori</b>	Nessuno

<b>Nome Interfaccia</b>	AdministratorDao<E, I>
<b>Superclasse</b>	Dao<E, I>
<b>Sottoclassi che implementano</b>	AdministratorDaoImplementation
<b>Responsabilità</b>	Definisce le firme dei metodi per la gestione del livello di persistenza per gli accessi degli amministratori e il recupero delle loro generalità.
<b>Collaboratori</b>	UtenteEntity

<b>Nome Interfaccia</b>	UserDao<E, I>
<b>Superclasse</b>	Dao<E, I>
<b>Sottoclassi che implementano</b>	UserDaoImplementation
<b>Responsabilità</b>	Definisce le firme dei metodi per la gestione del livello di persistenza per la parte dedicata agli utenti e alle loro amicizie.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	<i>ListaFilmDaoImplementation</i>
<b>Superclasse implementata</b>	<i>ListaFilmDao&lt;ListaFilmEntity, Integer&gt;</i>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce l'interazione con la parte di persistenza dei dati, memorizzati su un database, riguardanti le liste dei film degli utenti.
<b>Collaboratori</b>	<i>ListaFilmEntity</i>

<b>Nome Classe</b>	<i>SegnalazioneFilmDaoImplementation</i>
<b>Superclasse implementata</b>	<i>SegnalazioneFilmDao&lt;SegnalazioneFilmEntity, Integer&gt;</i>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce l'interazione con la parte di persistenza dei dati, memorizzati su un database, riguardanti le segnalazioni fatte a dei film da parte degli utenti.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• <i>SegnalazioneFilmEntity</i></li> <li>• <i>TipoSegnalazione</i></li> </ul>

<b>Nome Classe</b>	SegnalazioneUtenteDaoImplementation
<b>Superclasse implementata</b>	SegnalazioneUtenteDao<SegnalazioneUtenteEntity, Integer>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce l'interazione con la parte di persistenza dei dati, memorizzati su un database, riguardanti le segnalazioni fatte a degli utenti da parte di utenti.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• SegnalazioneUtenteEntity</li> <li>• TipoSegnalazione</li> </ul>

<b>Nome Classe</b>	AdministratorDaoImplementation
<b>Superclasse implementata</b>	AdministratorDao<CredenzialiAmministratoriEntity, Integer>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce l'interazione con la parte di persistenza dei dati, memorizzati su un database, riguardanti gli accessi al sistema e il recupero delle generalità degli amministratori.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• CredenzialiAmministratoriEntity</li> <li>• UtenteEntity</li> <li>• TipoSegnalazione</li> </ul>

<b>Nome Classe</b>	UserDaoImplementation
<b>Superclasse implementata</b>	UserDao< UtenteEntity, String>
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Gestisce l'interazione con la parte di persistenza dei dati, memorizzati su un database, riguardanti la gestione degli utenti e delle loro amicizie.
<b>Collaboratori</b>	<ul style="list-style-type: none"> <li>• UtenteEntity</li> <li>• TipologiaUtente</li> </ul>

## Entity

<b>Nome Classe</b>	SegnalazioneUtenteEntity
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappa la tabella del database, che gestisce le informazioni riguardanti le segnalazioni fatte ad un utente.
<b>Collaboratori</b>	TipoSegnalazione

<b>Nome Classe</b>	SegnalazioneFilmEntity
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappa la tabella del database, che gestisce le informazioni riguardanti le segnalazioni fatte ad un film.
<b>Collaboratori</b>	TipoSegnalazione

<b>Nome Classe</b>	UtenteEntity
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappa la tabella del database, che gestisce le informazioni degli utenti.
<b>Collaboratori</b>	TipologiaUtente

<b>Nome Classe</b>	ListaFilmEntity
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappa la tabella del database, che gestisce le informazioni riguardanti le liste di film.
<b>Collaboratori</b>	Nessuno

<b>Nome Classe</b>	CredenzialiAmministratoriEntity
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Mappa la tabella del database, che gestisce le credenziali di accesso degli amministratori.
<b>Collaboratori</b>	Nessuno

## Enums

<b>Nome Enumerazione</b>	TipologiaUtente
<b>Valori</b>	<ul style="list-style-type: none"> <li>• utente</li> <li>• amministratore</li> </ul>
<b>Responsabilità</b>	Definisce delle costanti per distinguere le diverse tipologie di utenti.
<b>Collaboratori</b>	Nessuno

<b>Nome Enumerazione</b>	TipoSegnalazione
<b>Valori</b>	<ul style="list-style-type: none"> <li>• Pendente</li> <li>• Approvata</li> <li>• Rigettata</li> <li>• Oscurata</li> </ul>
<b>Responsabilità</b>	Definisce delle costanti per distinguere lo stato in cui si trova una segnalazione.
<b>Collaboratori</b>	Nessuno

## Punto d'ingresso dell'applicativo

<b>Nome Classe</b>	DatabaseApplication
<b>Superclasse</b>	Nessuna
<b>Sottoclassi</b>	Nessuna
<b>Responsabilità</b>	Classe da cui viene avviato il server.
<b>Collaboratori</b>	Nessuno

# Documento di Testing

## Test Plan

Come per le tabelle di Cockburn, per comodità, le scritte che indicano il nome di un mockup sono cliccabili, e consentono di essere reindirizzati al relativo mockup rappresentato dal nome.

### Mobile

Id Test Case	Test_UC_1	
Nome	Utente loggato aggiunge film ad una lista	
Descrizione	L'obiettivo di questo test case è di verificare lo use case "Utente loggato aggiunge film ad una lista"	
Input	Oracolo	Esito
L'utente aggiunge un film alla lista dei film da vedere	Il film selezionato viene correttamente aggiunto alla lista dei film da vedere	✓
L'utente aggiunge un film alla lista dei film preferiti	Il film selezionato viene correttamente aggiunto alla lista dei film preferiti	✓
L'utente aggiunge un film ad una lista che ha raggiunto la capienza massima	Il film selezionato non viene aggiunto alla lista e viene mostrato un messaggio d'errore, "La lista è piena, rimuovi qualche film per poterne aggiungere altri"	✓
L'utente aggiunge un film già presente nella lista da vedere	L'interfaccia grafica impedisce quest'azione	✓
L'utente aggiunge un film già presente nella lista preferiti	L'interfaccia grafica impedisce quest'azione	✓

Id Test Case	Test_UC_2	
Nome	Utente loggato rimuove film da una lista	
Descrizione	L'obiettivo di questo test case è di verificare lo use case "Utente loggato rimuove film da una lista"	
Input	Oracolo	Esito
L'utente rimuove un film dalla lista dei film da vedere	Il film selezionato viene correttamente rimosso alla lista dei film da vedere	✓
L'utente rimuove un film dalla lista dei film preferiti	Il film selezionato viene correttamente aggiunto alla lista dei film preferiti	✓
L'utente rimuove un film dalla lista dei film da vedere, il film scelto non è presente nella lista	L'interfaccia grafica impedisce quest'azione	✓
L'utente rimuove un film dalla lista dei film preferiti, il film scelto non è presente nella lista	L'interfaccia grafica impedisce quest'azione	✓

L'utente rimuove più film dalla lista dei film da vedere	<b>M13</b> viene aggiornato rimuovendo dalla lista dei film i film eliminati	✓
L'utente rimuove più film dalla lista dei film preferiti	<b>M13</b> viene aggiornato rimuovendo dalla lista dei film i film eliminati	✓

<b>Id Test Case</b>	<a href="#">Test_UC_3</a>	
<b>Nome</b>	Utente effettua Login	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Utente effettua login"	
Input	Oracolo	Esito
L'utente inserisce username/e-mail e password validi	Il login avviene con successo e l'utente viene reindirizzato alla schermata <b>M0</b>	✓
L'utente inserisce credenziali non valide (username/e-mail non registrata o password errata)	Il login avviene fallisce e viene mostrata la schermata <b>M8</b>	✓
L'utente lascia uno o più campi incompleti	Il tasto login non viene abilitato, quindi il processo di login non può proseguire	✓

<b>Id Test Case</b>	<a href="#">Test_UC_4</a>	
<b>Nome</b>	Utente effettua registrazione	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Utente effettua registrazione"	
Input	Oracolo	Esito
L'utente Inserisce tutte le informazioni richieste rispettando tutti i vincoli	La registrazione va a buon fine e l'utente viene reindirizzato alla schermata <b>M7</b>	✓
L'utente inserisce uno username già in uso	La registrazione fallisce e viene mostrato <b>M10</b> con il messaggio "Username già utilizzato"	✓
L'utente inserisce una mail già in uso	La registrazione fallisce e viene mostrato <b>M10</b> con il messaggio "Email già utilizzata"	✓
L'utente inserisce una password di conferma che non corrisponde con la prima password inserita	La registrazione fallisce e viene mostrato <b>M10</b> con il messaggio "Le password non coincidono"	✓
L'utente inserisce una password che non rispetta i vincoli (lunghezza minima 8 caratteri, almeno una lettera maiuscola, almeno un carattere numerico, almeno un simbolo)	La registrazione fallisce e viene mostrato <b>M10</b> con il messaggio "La password non rispetta i vincoli"	✓
L'utente lascia uno o più campi incompleti	La registrazione fallisce e viene mostrato <b>M10</b> con i campi non inseriti in rosso e i relativi messaggi di errore.	✓

<b>Id Test Case</b>	Test_UC_5	
<b>Nome</b>	Utente loggato cerca film	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Utente loggato cerca film"	
Input	Oracolo	Esito
L'utente digita il titolo di un film che vuole cercare	La ricerca va a buon fine, e viene mostrata la schermata <b>M15</b> , con una lista di film col titolo che contengono la parola chiave cercata dall'utente	✓
L'utente cerca un film non disponibile	La ricerca non produce risultati e viene mostrata la schermata <b>M16</b>	✓
L'utente cerca un film lasciando il campo dedicato al titolo del film vuoto	L'interfaccia grafica non permette di effettuare ricerche vuote	✓

<b>Id Test Case</b>	Test_UC_6	
<b>Nome</b>	Utente loggato invia richiesta d'amicizia	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Utente loggato invia richiesta d'amicizia"	
Input	Oracolo	Esito
L'utente clicca sul tasto contrassegnato da un "+" in alto a destra sul profilo di un altro utente	L'inoltro della richiesta d'amicizia va a buon fine	✓
L'utente invia una richiesta d'amicizia ad un utente che è già un suo amico	Se un utente è già amico sul suo profilo non sarà disponibile il tasto per poterlo aggiungere, quindi il processo di aggiunta termina	✓
L'utente prova ad aggiungere sé stesso agli amici	L'interfaccia grafica non lo permette, sul profilo personale non è presente il tasto per inviare una richiesta d'amicizia	✓

<b>Id Test Case</b>	<a href="#">Test_UC_7</a>	
<b>Nome</b>	Utente loggato rimuove amico	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Utente loggato invia rimuove amico"	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente clicca sul tasto contrassegnato da un “–” in alto a destra sul profilo di un amico	La rimozione dell'amico va a buon fine	✓
L'utente rimuove dagli amici un utente che ancora non è amico	Se un utente non è amico sul suo profilo non sarà presente il tasto per rimuoverlo dagli amici, quindi il processo di rimozione s'interrompe	✓
L'utente prova a rimuovere sé stesso dagli amici	L'interfaccia grafica non lo permette, sul profilo personale non è presente il tasto per rimuoversi dalla lista degli amici	✓

<b>Id Test Case</b>	<a href="#">Test_UC_8</a>	
<b>Nome</b>	Utente loggato visualizza film in comune	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Utente loggato visualizza film in comune"	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente clicca sul secondo tasto contraddistinto da due quadrati che s'intersecano e un puntatore azzurro	Viene mostrato <b>M17</b> con la lista dei film presenti sia nella lista dell'utente loggato che dell'amico	✓
L'utente clicca sul tasto contraddistinto da due quadrati che s'intersecano e un puntatore azzurro sul profilo di un amico con cui non sono presenti amici film in comune	Viene mostrato <b>M24</b>	✓
L'utente cerca di visualizzare i film in comune con un utente che non è suo amico	L'operazione non è consentita dall'interfaccia grafica, poiché se un utente non è amico il tasto dedicato a questa funzionalità non è disponibile	✓

<b>Id Test Case</b>	<b>Test_UC_9</b>	
<b>Nome</b>	Utente loggato segnala film	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Utente loggato segnala film"	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente segnala un film scegliendo una motivazione diversa da "Altro"	La segnalazione viene inviata, si mostra il messaggio "Segnalazione inviata" e tutti i componenti d'input di <b>M23</b> vengono disabilitati	✓
L'utente sceglie "Altro" come motivo della segnalazione e digita la sua motivazione	La segnalazione viene inviata, si mostra il messaggio "Segnalazione inviata" e tutti i componenti d'input di <b>M23</b> vengono disabilitati	✓
L'utente sceglie "Altro" e lascia il campo per la motivazione vuota	Il tasto per effettuare la segnalazione non viene abilitato, di conseguenza non è possibile segnalare	✓
L'utente non seleziona alcun motivo per la segnalazione	Il tasto per effettuare la segnalazione non viene abilitato, di conseguenza non è possibile segnalare	✓

<b>Id Test Case</b>	<b>Test_UC_10</b>	
<b>Nome</b>	Utente loggato segnala utente	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Utente loggato segnala utente"	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'utente segnala un utente scegliendo una motivazione diversa da "Altro"	La segnalazione viene inviata, si mostra il messaggio "Segnalazione inviata" e tutti i componenti d'input di <b>M25</b> vengono disabilitati	✓
L'utente sceglie "Altro" come motivo della segnalazione e digita la sua motivazione	La segnalazione viene inviata, si mostra il messaggio "Segnalazione inviata" e tutti i componenti d'input di <b>M25</b> vengono disabilitati	✓
L'utente sceglie "Altro" e lascia il campo per la motivazione vuota	Il tasto per effettuare la segnalazione non viene abilitato, di conseguenza non è possibile segnalare	✓
L'utente non seleziona alcun motivo per la segnalazione	Il tasto per effettuare la segnalazione non viene abilitato, di conseguenza non è possibile segnalare	✓
L'utente segnala sé stesso	L'interfaccia grafica non lo permette, sul profilo personale non è presente il tasto per segnalare	✓

## Desktop

<b>Id Test Case</b>	Test_UC_1	
<b>Nome</b>	Amministratore effettua login	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Amministratore effettua login"	
Input	Oracolo	Esito
L'amministratore inserisce e-mail e password validi	Il login avviene con successo e l'amministratore viene reindirizzato alla schermata D2	✓
L'amministratore inserisce credenziali non valide (e-mail non registrata o password errata)	Il login avviene fallisce e viene mostrata la schermata D1	✓
L'amministratore lascia uno o più campi incompleti	Il tasto login non viene abilitato, quindi il processo di login non può proseguire	✓

<b>Id Test Case</b>	Test_UC_2	
<b>Nome</b>	Amministratore visualizza segnalazioni gestite	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Amministratore visualizza segnalazioni gestite"	
Input	Oracolo	Esito
L'amministratore clicca sul tasto "film" quando sono presenti segnalazioni gestite	Viene mostrata la schermata D6	✓
L'amministratore clicca sul tasto "film" quando non sono presenti segnalazioni gestite	Viene mostrata la schermata D9	✓
L'amministratore clicca sul tasto "utenti" quando sono presenti segnalazioni gestite	Viene mostrata la schermata D3	✓
L'amministratore clicca sul tasto "utenti" quando non sono presenti segnalazioni gestite	Viene mostrata la schermata D9	✓
L'amministratore clicca su un utente che ha segnalato un film	Viene mostrata la schermata D8, con il tasto associato alla risposta data evidenziato, e i restanti due tasti disabilitati	✓
L'amministratore clicca su un utente che ha segnalato un altro utente	Viene mostrata la schermata D5, con il tasto associato alla risposta data evidenziato, e il restante tasto disabilitato	✓

<b>Id Test Case</b>	<b>Test_UC_3</b>	
<b>Nome</b>	Amministratore risolve segnalazione pendente	
<b>Descrizione</b>	L'obiettivo di questo test case è di verificare lo use case "Amministratore risolve segnalazione pendente"	
<b>Input</b>	<b>Oracolo</b>	<b>Esito</b>
L'amministratore clicca sul tasto "film" quando sono presenti segnalazioni pendenti	Viene mostrata la schermata <b>D6</b>	✓
L'amministratore clicca sul tasto "film" quando non sono presenti segnalazioni pendenti	Viene mostrata la schermata <b>D9</b>	✓
L'amministratore clicca sul tasto "utenti" quando sono presenti segnalazioni pendenti	Viene mostrata la schermata <b>D3</b>	✓
L'amministratore clicca sul tasto "utenti" quando non sono presenti segnalazioni pendenti	Viene mostrata la schermata <b>D9</b>	✓
L'amministratore clicca su un utente che ha segnalato un film e clicca sul tasto "Approva" o sul tasto "Rigetta"	Nel caso in cui sia presente almeno un'altra segnalazione pendente, mostra <b>D7</b> con il numero di segnalazioni aggiornato, altrimenti mostra <b>D6</b> privato del film appena gestito	✓
L'amministratore clicca su un utente che ha segnalato un film e clicca sul tasto "Oscura"	Risolve a cascata tutte le segnalazioni pendenti degli utenti associate a quel film, memorizzandole come "Oscurata" e mostra <b>D6</b> privato del film gestito	✓
L'amministratore clicca su un utente che ha segnalato un altro utente e clicca sul tasto "Approva" o sul tasto "Rigetta"	Notifica l'autore dell'esito della segnalazione	✓

## Testing del metodo getById

Il metodo appartiene alla classe dao *UserDaoImplementation*, presente all'interno dell'applicativo server. Il suo scopo è, data in input l'e-mail di un utente, restituire l'utente associato ad essa, mappato sotto forma di *UtenteEntity*, una classe entity facente parte ancora una volta dell'applicativo server. Nel caso in cui l'e-mail in input non corrisponda ad un utente registrato nel sistema, restituisce *null*.

La firma del metodo è la seguente:

```
public UtenteEntity getById(String email)
```

Parametro di input:

- String email : una stringa con lo scopo di rappresentare l'e-mail di un utente registrato nel sistema.

Parametro di output:

- *UtenteEntity* che mappa gli attributi dell'utente recuperato tramite l'e-mail data in input.

Esempio d'uso

CHIAMATA	OUTPUT
getById("existing@email.com")	@UtenteEntity
getById("unexisting@email.com")	null
getById(null)	null
getById("")	null
getById(" ")	null

Come setup prima dei test viene creato il seguente utente per il testing del suo recupero dal database:

```
utente = new UtenteEntity("TestUsername", "test", "test", "test@test.com", TipologiaUtente.utente);
```

## Testing Black-Box

Le classi di equivalenza individuate sono le seguenti:

- CE1: {"e-mail registrate"}      valida
- CE2: String \ {CE1}      non valida
- CE3: {null}      non valida
- CE4: {""}      non valida

Trattandosi di un metodo che prende in input un solo parametro, le strategie WECT e SECT coincidono, inoltre ogni metodo copre una sola classe di equivalenza.

Le classi di equivalenza coperte dai metodi di seguito illustrati sono le seguenti:

- `getByIdWithValidEmail()` :      copre CE1
- `getByIdWithWrongEmail()`,  
`getByIdWithBlankSpace()` :      coprono CE2
- `getByIdNullEmail()` :      copre CE3
- `getByIdWithEmptyString()` :      copre CE4

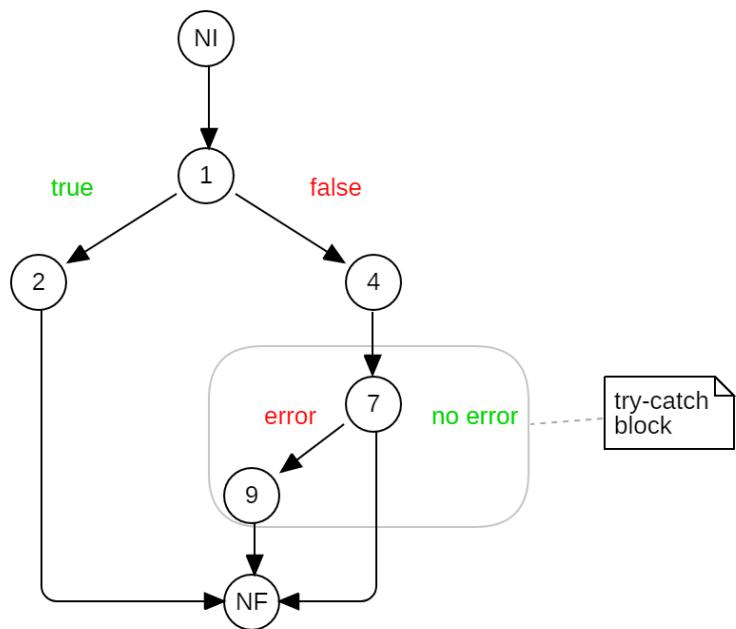
```
. class UserDaoImplementationTest {
.     @Test
.     void getByIdNullEmail() {
.         Assertions.assertNull(dao.getById(null));
.     }
.
.     @Test
.     void getByIdWithValidEmail() {
.         Assertions.assertEquals(utente, dao.getById("test@test.com"));
.     }
.
.     @Test
.     void getByIdWithWrongEmail() {
.         Assertions.assertNull(dao.getById("testErrore@test.com"));
.     }
.
.     @Test
.     void getByIdWithEmptyString() {
.         Assertions.assertNull(dao.getById(""));
.     }
.
.     @Test
.     void getByIdWithBlankSpace() {
.         Assertions.assertNull(dao.getById("  "));
.     }
}
```

## Testing White-Box

Il metodo ha il seguente corpo:

```
0  public UtenteEntity getById(String email) {  
1      if(email == null)  
2          return null;  
3  
4      final String sql = "SELECT * FROM public.\"Utente\" WHERE public.\"Utente\".email = ?;";  
5  
6      try {  
7          return jdbcTemplate.queryForObject(sql, new String[]{email},  
8                                              (resultSet,i)->resultSetToUserEntity(resultSet));  
9      }catch(DataAccessException e){  
10         return null;  
11     }  
11 }
```

Il Grafo del Flusso di Controllo ottenuto dal codice è il seguente:



In questo specifico caso, si noti che i test cases utilizzati per il test Black-Box di fatto compiono già una Node e Branch Coverage del grafo. Nello specifico:

- `getByIdNullEmail()` copre il path : 1..2 → NF
- `getByIdWithValidEmail()` copre il path : 1 → 4 → 7 → NF
- `getByIdWithWrongEmail()`, `getByIdWithBlankSpace()` e `getByIdWithEmptyString()` coprono il path : 1 → 4 → 7 → 9 → NF

## Testing del metodo isFriendRequestPending

Il metodo appartiene alla classe dao *UserDaoImplementation*, presente all'interno dell'applicativo server. Il suo scopo è, date in input le e-mail di due utenti, restituire vero nel caso in cui tra i due utenti esista una richiesta di amicizia in attesa di conferma, falso altrimenti. Il metodo lancia *IllegalArgumentException* nel caso in cui una o entrambe le due e-mail in input siano null.

La firma del metodo è la seguente:

```
public boolean isFriendRequestPending(String userEmail, String friendEmail)  
throws IllegalArgumentException
```

Parametri di input:

- `String userEmail`: una stringa con lo scopo di rappresentare l'e-mail di un utente registrato nel sistema che ha ricevuto o inviato una richiesta di amicizia verso `friendEmail` in attesa di conferma.
- `String friendEmail`: una stringa con lo scopo di rappresentare l'e-mail di un utente registrato nel sistema che ha ricevuto o inviato una richiesta di amicizia verso `userEmail` in attesa di conferma.

Essendo la relazione di amicizia *simmetrica*, non ha importanza quale delle due e-mail venga passata in input prima dell'altra.

Parametro di output:

- `boolean` che vale *true* se le e-mail date in input corrispondono ad utenti reali e se tra i due utenti è presente una richiesta di amicizia in attesa di conferma, *false* altrimenti.

Esempio d'uso

Si considerino tre utenti di esempio:

`utente1`: test@mail.com, `utente2`: test2@mail.com, `utente3`: test3@mail.com

Supponiamo che utente1 abbia inviato una richiesta di amicizia ad utente2, e che tale richiesta non abbia ancora ricevuto risposta da parte di utente2, e supponiamo che tra utente3 e gli altri due utenti non ci sia nessuna relazione:

CHIAMATA	OUTPUT
<code>isFriendRequestPending(null, "test@email.com")</code>	<code>IllegalArgumentException</code>
<code>isFriendRequestPending("test@mail.com", "test2@mail.com")</code>	<code>true</code>
<code>isFriendRequestPending("test2@mail.com", "test@mail.com")</code>	<code>true</code>
<code>isFriendRequestPending("test@mail.com", "test3@mail.com")</code>	<code>false</code>
<code>isFriendRequestPending("test3@mail.com", "test2@mail.com")</code>	<code>false</code>
<code>isFriendRequestPending("test@mail.com", "unexisting@mail")</code>	<code>false</code>
<code>isFriendRequestPending("", "")</code>	<code>false</code>

Come setup prima dei test vengono creati i seguenti utenti:

```
utente = new UtenteEntity("TestUsername", "test", "test", "test@test.com", TipologiaUtente.utente);
amicoPendente = new UtenteEntity("TestAmicizia", "mario", "rossi", "test1@test.com", TipologiaUtente.utente);
amico = new UtenteEntity("TestAmicizia2", "mario2", "rossi2", "test2@test.com", TipologiaUtente.utente);
```

Tra *utente* e *amico* esiste una relazione di amicizia, tra *utente* e *amicoPendente* esiste una relazione di attesa conferma richiesta di amicizia, tra *amico* e *amicoPendente* non esiste nessuna relazione. Le istruzioni che consentono di ottenere ciò sono:

```
Assertions.assertTrue(userDao.addFriend(utente, amicoPendente));
Assertions.assertTrue(userDao.addFriend(utente, amico));
Assertions.assertTrue(userDao.confirmFriendRequest(utente, amico));
```

## Testing Black-Box

Le classi di equivalenza individuate sono le seguenti:

- Per **userEmail**:
  - CE1: {"email registrate"} valida
  - CE2: String \ {CE1} non valida
  - CE3: {null} non valida
  - CE4: {""} non valida
- Per **friendEmail**:
  - CE5: {"email registrate ed appartenenti alla lista degli utenti che hanno ricevuto una richiesta di amicizia da userEmail"} valida
  - CE6: String \ {CE5} non valida
  - CE7: {null} non valida
  - CE8: {""} non valida

La strategia di testing adottata è WECT (**Weak Equivalence Class Testing**).

Le classi di equivalenza coperte dai metodi di seguito illustrati sono le seguenti:

- `isFriendRequestPendingUserIsNullAndFriendIsNull()`: copre CE3 e CE7
- `isFriendRequestPendingUserIsNotNullAndFriendIsNull()`: copre CE1 e CE7
- `isFriendRequestPendingUserIsNullAndFriendIsNotNull()`: copre CE3 e CE5
- `isFriendRequestPendingUserIsNotNullAndFriendIsNotNull()` e `isFriendRequestPendingFriendIsNotNullAndUserIsNotNull()`: coprono CE1 e CE5
- `isFriendRequestPendingInvalidFriend()`: copre CE1 e CE6
- `isFriendRequestPendingInvalidUserWithFriendEmail()` e `isFriendRequestPendingInvalidUserWithEmail()`: coprono CE2 e CE5
- `isFriendRequestPendingUserIsBlankAndFriendIsBlank()`: copre CE4 e CE8

```
. class UserDaoImplementationTest {
.     @Test
.     void isFriendRequestPendingUserIsNullAndFriendIsNull() {
.         Assertions.assertThrows(IllegalArgumentException.class,
.             () -> userDao.isFriendRequestPending(null, null));
.     }
.
.     @Test
.     void isFriendRequestPendingUserIsNotNullAndFriendIsNull() {
.         Assertions.assertThrows(IllegalArgumentException.class,
.             () -> userDao.isFriendRequestPending("test@test.com", null));
.     }
.
.     @Test
.     void isFriendRequestPendingUserIsNullAndFriendIsNotNull() {
.         Assertions.assertThrows(IllegalArgumentException.class,
.             () -> userDao.isFriendRequestPending(null, "test@test.com"));
.     }
.
.     @Test
.     void isFriendRequestPendingUserIsNotNullAndFriendIsNotNull() {
.         Assertions.assertTrue(userDao.isFriendRequestPending
.             ("test@test.com", "test1@test.com"));
.     }
}
```

```
.    @Test
.        void isFriendRequestPendingFriendIsNotNullAndUserIsNotNull() {
.            Assertions.assertTrue(userDao.isFriendRequestPending
.                ("test1@test.com", "test@test.com"));
.        }
.
.    @Test
.        void isFriendRequestPendingInvalidFriend() {
.            Assertions.assertFalse(userDao.isFriendRequestPending
.                ("test@test.com", "testFail@test.com"));
.        }
.
.    @Test
.        void isFriendRequestPendingInvalidUserWithValidFriend() {
.            Assertions.assertFalse(userDao.isFriendRequestPending
.                ("testFail@test.com", "test1@test.com"));
.        }
.
.    @Test
.        void isFriendRequestPendingInvalidUserWithEmail() {
.            Assertions.assertFalse(userDao.isFriendRequestPending
.                ("testFail@test.com", "test@test.com"));
.        }
.
.    @Test
.        void isFriendRequestPendingUserIsBlankAndFriendIsBlank() {
.            Assertions.assertFalse(userDao.isFriendRequestPending
.                ("", ""));
.        }
.
.    }
```

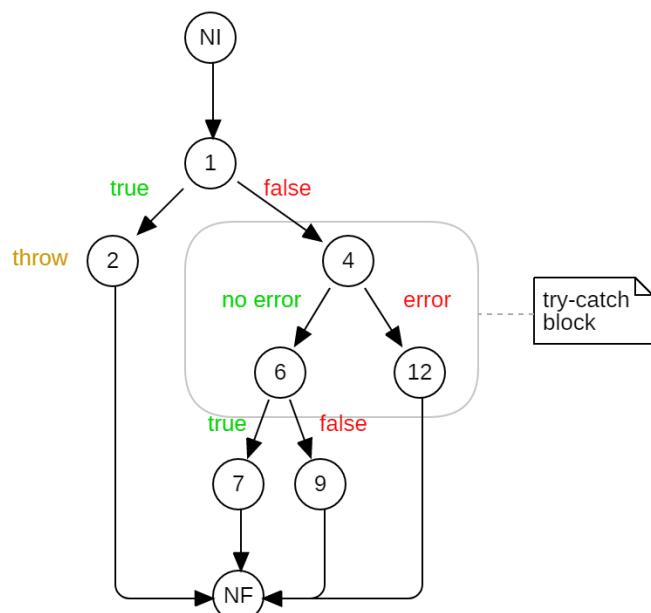
## Testing White-Box

Il metodo ha il seguente corpo:

```
0  public boolean isFriendRequestPending(String userEmail, String friendEmail)
     throws IllegalArgumentException {
1      if (userEmail == null || friendEmail == null)
2          throw new IllegalArgumentException("Passare un'email non nulla");
3      try {
4          Boolean confermata = jdbcTemplate.queryForObject(SqlCommandForIsFriendRequestPending(),
5                  new String[]{userEmail, friendEmail, friendEmail, userEmail}, Boolean.class);
6          if(confermata != null)
7              return !confermata; //Se confermata == false, allora è pendente
8          else
9              return false;
10     }
11     catch(Exception e) {
12         return false;
13     }
14 }
```

```
.  private String getSqlCommandForIsFriendRequestPending() {
.      return "SELECT confermata " +
.             "FROM \"Amici\" " +
.             "WHERE ((\"Amici\".\"Email_Utente\" = ? AND \"Amici\".\"Email_Amico\" = ?) " +
.                   " OR (\"Amici\".\"Email_Utente\" = ? AND .\"Amici\".\"Email_Amico\" = ?))" +
.             " AND confermata = 'false';";
. }
```

Il Grafo del Flusso di Controllo ottenuto dal codice è il seguente:



Di seguito vengono illustrati i metodi necessari ai fini di una Branch Coverage totale del **GFC** precedentemente illustrato.

```
.    @Test
.    void isFriendRequestPending_1b_2b() {
.        Assertions.assertThrows(IllegalArgumentException.class,
.            () -> userDao.isFriendRequestPending("test1@test.com", null));
.    }
.
.
.    @Test
.    void isFriendRequestPending_1b_4b_12b() {
.        Assertions.assertFalse(userDao.isFriendRequestPending
.            ("testFail@test.com", "testError@test.com"));
.    }
.
.
.    @Test
.    void isFriendRequestPending_1b_4b_6b_7b_isPending() {
.        Assertions.assertTrue(userDao.isFriendRequestPending
.            ("test@test.com", "test1@test.com"));
.    }
.
.
.    @Test
.    void isFriendRequestPending_1b_4b_12b_isNotPending() {
.        Assertions.assertFalse(userDao.isFriendRequestPending
.            ("test@test.com", "test2@test.com"));
.    }
.
.
.    @Test
.    void isFriendRequestPending_1b_4b_9b() {
.        /*
.            Questo ramo non è testabile poiché il metodo
.            jdbcTemplate.queryForObject() del nodo 4,
.            non restituisce mai null.
.        */
.        Assertions.assertTrue(true);
.    }
```

## Testing del metodo isUserNameValid

Il metodo appartiene alla classe view *Utilities*, presente all'interno dell'applicativo Android.

Il suo scopo è, data in input una stringa rappresentante un possibile username, restituire vero nel caso in cui tale stringa rispetti i seguenti vincoli:

- Minimo 3 caratteri
- @ non deve essere presente
- La stringa non deve essere vuota
- La stringa non deve contenere spazi bianchi nel mezzo

In caso di violazione di almeno uno di questi vincoli, il metodo restituisce falso.

La firma del metodo è la seguente:

```
public static boolean isUserNameValid(String username)
```

Parametro di input:

- `String username` : stringa rappresentante un username di un utente.

Parametro di output:

- `boolean` che vale *true* se l'username dato in input rispetta i vincoli richiesti, *false* altrimenti.

Esempio d'uso

CHIAMATA	OUTPUT
<code>isUserNameValid(null)</code>	<code>false</code>
<code>isUserNameValid("@uSer")</code>	<code>false</code>
<code>isUserNameValid("username")</code>	<code>true</code>
<code>isUserNameValid("user name")</code>	<code>false</code>

## Testing Black-Box

Le classi di equivalenza individuate sono le seguenti:

- CE1: {null} non valida
- CE2: {"Stringhe che rispettano i vincoli"} valida
- CE3: String \ {CE2} non valida

Trattandosi di un metodo che prende in input un solo parametro, le strategie WECT e SECT coincidono, inoltre ogni metodo copre una sola classe di equivalenza.

Le classi di equivalenza coperte dai metodi di seguito illustrati sono le seguenti:

- `isUserNameValidEmptyUsername()` : copre CE3
- `isUserNameValidNullUsername()` : copre CE1
- `isUserNameValidStartsWithBlankSpace()` : copre CE2
- `isUserNameValidStartsWithMultipleBlankSpaces()` : copre CE2
- `isUserNameValidEndsWithBlankSpace()` : copre CE2
- `isUserNameValidEndsWithMultipleBlankSpaces()` : copre CE2
- `isUserNameValidStartsAndEndsWithBlankSpace()` : copre CE2
- `isUserNameValidStartsAndEndsWithMultipleBlankSpaces()` : copre CE2
- `isUserNameValidContainsSingleBlankSpace()` : copre CE3
- `isUserNameValidContainsMultipleInnerBlankSpaces()` : copre CE3
- `isUserNameValidContainsMultipleBlankSpaces()` : copre CE3
- `isUserNameValidStartsWithAtChar()` : copre CE3
- `isUserNameValidContainsOnlyAtChar()` : copre CE3
- `isUserNameValidStartsWithMultipleAtChar()` : copre CE3
- `isUserNameValidEndsWithAtChar()` : copre CE3
- `isUserNameValidEndsWithMultipleAtChar()` : copre CE3
- `isUserNameValidContainsAtChar()` : copre CE3

- `isUserNameValidContainsMultipleAtChar()` : copre CE3
- `isUserNameValidWithTwoCharactersLength()` : copre CE3
- `isUserNameValidWithThreeCharactersLength()` : copre CE2
- `isUserNameValidWithFourCharactersLength()` : copre CE2

```
.
.   class UtilitiesTest {
.     @Test
.     void isUserNameValidEmptyUsername() {
.       assertFalse(Utilities.isUserNameValid(""));
.     }
.
.     @Test
.     void isUserNameValidNullUsername() {
.       assertFalse(Utilities.isUserNameValid(null));
.     }
.
.     @Test
.     void isUserNameValidStartsWithBlankSpace() {
.       assertTrue(Utilities.isUserNameValid(" Username"));
.     }
.
.     @Test
.     void isUserNameValidStartsWithMultipleBlankSpaces() {
.       assertTrue(Utilities.isUserNameValid("      Username"));
.     }
.
.     @Test
.     void isUserNameValidEndsWithBlankSpace() {
.       assertTrue(Utilities.isUserNameValid("Username "));
.     }
.
.     @Test
.     void isUserNameValidEndsWithMultipleBlankSpaces() {
.       assertTrue(Utilities.isUserNameValid("Username      "));
.     }
}
```

```
. @Test
. void isUserNameValidStartsAndEndsWithBlankSpace() {
.     assertTrue(Utilities.isUserNameValid(" Username "));
. }
.

. @Test
. void isUserNameValidStartsAndEndsWithMultipleBlankSpaces() {
.     assertTrue(Utilities.isUserNameValid("      Username      "));
. }
.

. @Test
. void isUserNameValidContainsSingleBlankSpace() {
.     assertFalse(Utilities.isUserNameValid("User name"));
. }
.

. @Test
. void isUserNameValidContainsMultipleInnerBlankSpaces() {
.     assertFalse(Utilities.isUserNameValid("U    ser na   me"));
. }
.

. @Test
. void isUserNameValidContainsMultipleBlankSpaces() {
.     assertFalse(Utilities.isUserNameValid("      U    ser na   me      "));
. }
.

. @Test
. void isUserNameValidStartsWithAtChar() {
.     assertFalse(Utilities.isUserNameValid("@Username"));
. }
.

. @Test
. void isUserNameValidContainsOnlyAtChar() {
.     assertFalse(Utilities.isUserNameValid("@@@"));
. }
```

```

.   @Test
.     void isUserNameValidStartsWithMultipleAtChar() {
.       assertFalse(Utilities.isUserNameValid("@@@@@@@@Username"));
.     }
.

.   @Test
.     void isUserNameValidEndsWithAtChar() {
.       assertFalse(Utilities.isUserNameValid("Username@"));
.     }
.

.   @Test
.     void isUserNameValidEndsWithMultipleAtChar() {
.       assertFalse(Utilities.isUserNameValid("Username@@@@@@@"));
.     }
.

.   @Test
.     void isUserNameValidContainsAtChar() {
.       assertFalse(Utilities.isUserNameValid("Usern@ame"));
.     }
.

.   @Test
.     void isUserNameValidContainsMultipleAtChar() {
.       assertFalse(Utilities.isUserNameValid("Us@@@ern@am@@e"));
.     }
.

.   @Test
.     void isUserNameValidWithTwoCharactersLength() {
.       assertFalse(Utilities.isUserNameValid("us"));
.     }
.

.   @Test
.     void isUserNameValidWithThreeCharactersLength() {
.       assertTrue(Utilities.isUserNameValid("use"));
.     }
.

.   @Test
.     void isUserNameValidWithFourCharactersLength() {
.       assertTrue(Utilities.isUserNameValid("user"));
.     }
.
}

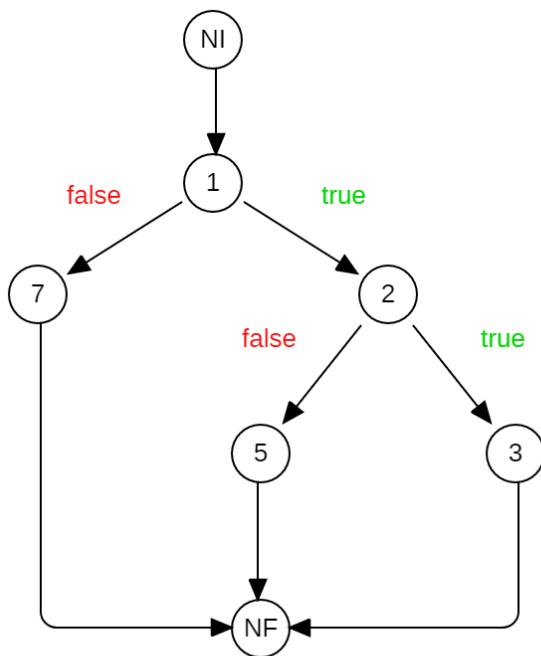
```

## Testing White-Box

Il metodo ha il seguente corpo:

```
0  public static boolean isUserNameValid(String username) {  
1      if(username != null) {  
2          if(username.trim().isEmpty() || username.trim().length() < 3 || username.contains("@"))  
3              return false;  
4  
5          return !Pattern.compile("\\s+").matcher(username.trim()).find();  
6      }  
7      return false;  
8  }
```

Il Grafo del Flusso di Controllo ottenuto dal codice è il seguente:



Di seguito vengono illustrati i metodi necessari ai fini di una Node e Branch Coverage totale del **GFC** precedentemente illustrato.

```
.    @Test
.    void isUserNameValid_1b_7b() {
.        assertFalse(Utilities.isUserNameValid(null));
.    }
.
.    @Test
.    void isUserNameValid_1b_2b_3b() {
.        assertFalse(Utilities.isUserNameValid("u"));
.    }
.
.    @Test
.    void isUserNameValid_1b_2b_5b() {
.        assertTrue(Utilities.isUserNameValid("User"));
.    }
```

## Testing del metodo isPasswordValid

Il metodo appartiene alla classe view Utilities, presente all'interno dell'applicativo Android.

Il suo scopo è, data in input una stringa rappresentante una possibile password, restituire vero nel caso in cui tale stringa rispetti i seguenti vincoli:

- La stringa contiene almeno un carattere speciale
- La stringa contiene almeno un numero
- La stringa contiene almeno una lettera maiuscola
- La stringa contiene almeno una lettera minuscola
- La stringa ha una lunghezza minima di otto caratteri
- La stringa non deve contenere spazi bianchi nel mezzo

In caso di violazione di almeno uno di questi vincoli, il metodo restituisce falso.

La firma del metodo è la seguente:

```
public static boolean isPasswordValid (String password)
```

Parametro di input:

- `String password` : stringa rappresentante la password di un utente.

Parametro di output:

- `boolean` che vale *true* se la password data in input rispetta i vincoli richiesti, *false* altrimenti.

Esempio d'uso

CHIAMATA	OUTPUT
<code>isPasswordValid(null)</code>	<code>false</code>
<code>isPasswordValid("@"uSer")</code>	<code>false</code>
<code>isPasswordValid("P4ssword!")</code>	<code>true</code>
<code>isPasswordValid("pass word")</code>	<code>false</code>
<code>isPasswordValid(" Pass_word123 ")</code>	<code>false</code>

# Testing Black-Box

Le classi di equivalenza individuate sono le seguenti:

- CE1: {"Stringhe che rispettano i vincoli"} valida
  - CE2: String \ {CE1} non valida
  - CE3: {null} non valida

Trattandosi di un metodo che prende in input un solo parametro, le strategie WECT e SECT coincidono, inoltre ogni metodo copre una sola classe di equivalenza.

Le classi di equivalenza coperte dai metodi di seguito illustrati sono le seguenti:

- `isPasswordValidNullPassword()`: copre CE3
  - `isPasswordValidEmptyPassword()`: copre CE2
  - `isPasswordValidBlankPassword()`: copre CE2
  - `isPasswordValidWithSevenCharactersLength()`: copre CE2
  - `isPasswordValidWithEightCharactersLength()`: copre CE1
  - `isPasswordValidWithNineCharactersLength()`: copre CE1
  - `isPasswordValidWithoutUppercase()`: copre CE2
  - `isPasswordValidWithoutLowercase()`: copre CE2
  - `isPasswordValidWithoutNumber()`: copre CE2
  - `isPasswordValidWithoutSpecialCharacter()`: copre CE2
  - `isPasswordValidStartsWithBlankSpace()`: copre CE2
  - `isPasswordValidStartsWithMultipleBlankSpaces()`: copre CE2
  - `isPasswordValidEndsWithBlankSpace()`: copre CE2
  - `isPasswordValidEndsWithMultipleBlankSpaces()`: copre CE2
  - `isPasswordValidStartsAndEndsWithBlankSpace()`: copre CE2
  - `isPasswordValidStartsAndEndsWithMultipleBlankSpaces()`: copre CE2
  - `isPasswordValidContainsSingleBlankSpace()`: copre CE2
  - `isPasswordValidContainsMultipleInnerBlankSpaces()`: copre CE2

- `isPasswordValidContainsMultipleBlankSpaces()`: copre CE2
- `isPasswordValidLongPassword()`: copre CE2

```
.
.   class UtilitiesTest {
.
.     @Test
.     void isPasswordValidNullPassword() {
.       assertFalse(Utilities.isPasswordValid(null));
.     }
.
.     @Test
.     void isPasswordValidEmptyPassword() {
.       assertFalse(Utilities.isPasswordValid(""));
.     }
.
.     @Test
.     void isPasswordValidBlankPassword() {
.       assertFalse(Utilities.isPasswordValid("    "));
.     }
.
.     @Test
.     void isPasswordValidWithSevenCharactersLength() {
.       assertFalse(Utilities.isPasswordValid("P@ssw0r"));
.     }
.
.     @Test
.     void isPasswordValidWithEightCharactersLength() {
.       assertTrue(Utilities.isPasswordValid("P@ssw0rd"));
.     }
.
.     @Test
.     void isPasswordValidWithNineCharactersLength() {
.       assertTrue(Utilities.isPasswordValid("P@ssw0rd_"));
.     }
.
.     @Test
.     void isPasswordValidWithoutUppercase() {
.       assertFalse(Utilities.isPasswordValid("p@ssw0rd"));
.     }
.
```

```
•     @Test
•         void isPasswordValidWithoutLowercase() {
•             assertFalse(Utilities.isPasswordValid("P@SSW0RD"));
•         }
•
•     @Test
•         void isPasswordValidWithoutNumber() {
•             assertFalse(Utilities.isPasswordValid("P@ssword"));
•         }
•
•     @Test
•         void isPasswordValidWithoutSpecialCharacter() {
•             assertFalse(Utilities.isPasswordValid("Passw0rd"));
•         }
•
•     @Test
•         void isPasswordValidStartsWithBlankSpace() {
•             assertFalse(Utilities.isPasswordValid(" P@ssw0rd"));
•         }
•
•     @Test
•         void isPasswordValidStartsWithMultipleBlankSpaces() {
•             assertFalse(Utilities.isPasswordValid("      P@ssw0rd"));
•         }
•
•     @Test
•         void isPasswordValidEndsWithBlankSpace() {
•             assertFalse(Utilities.isPasswordValid("P@ssw0rd "));
•         }
•
•     @Test
•         void isPasswordValidEndsWithMultipleBlankSpaces() {
•             assertFalse(Utilities.isPasswordValid("P@ssw0rd      "));
•         }
•
•     @Test
•         void isPasswordValidStartsAndEndsWithBlankSpace() {
•             assertFalse(Utilities.isPasswordValid(" P@ssw0rd "));
•         }
•
```

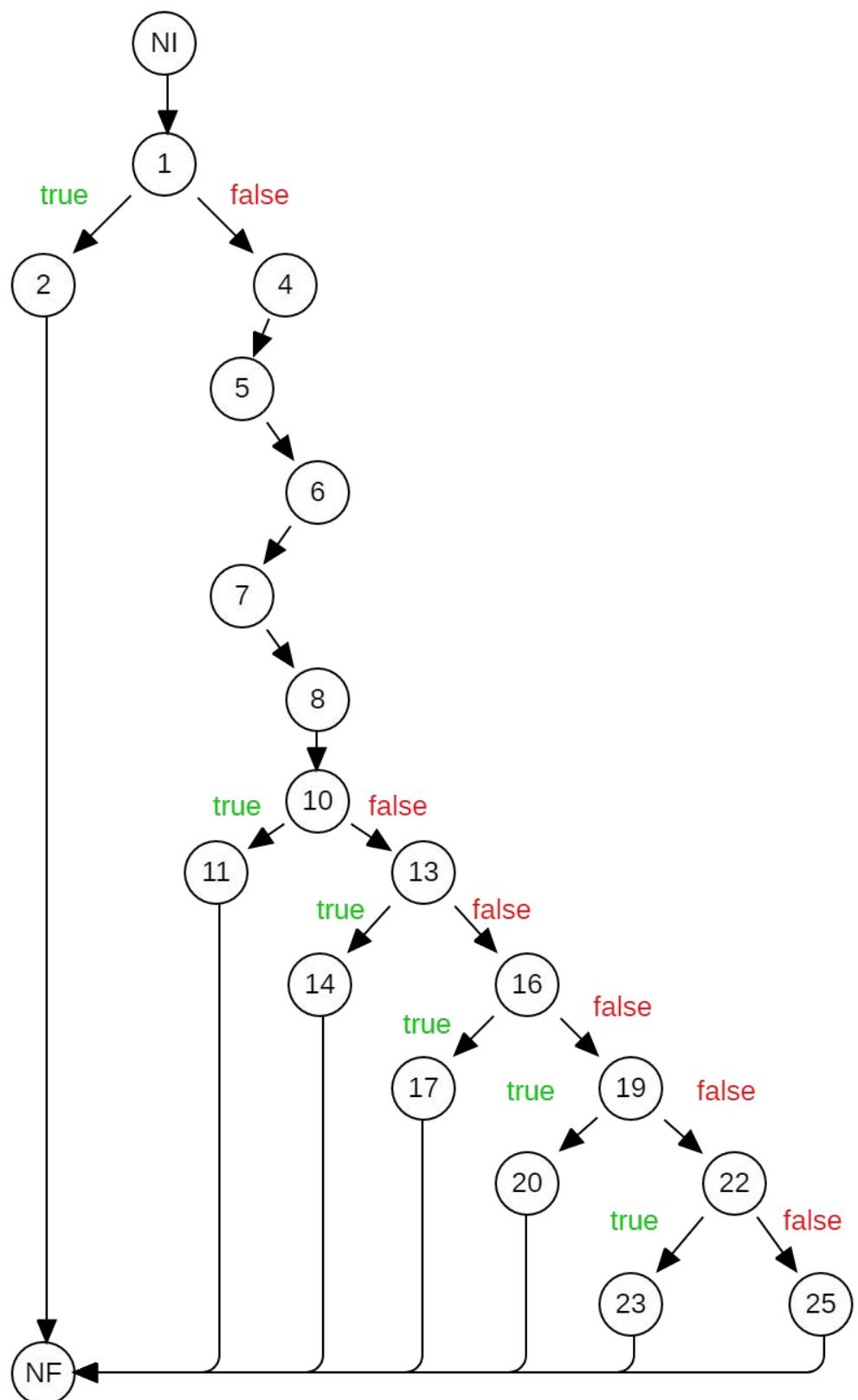


## Testing White-Box

Il metodo ha il seguente corpo:

```
0  public static boolean isPasswordValid(String password) {  
1      if(password == null) {  
2          return false;  
3  
4      Pattern specialCharPattern = Pattern.compile("[^a-z0-9 ]", Pattern.CASE_INSENSITIVE);  
5      Pattern upperCasePattern = Pattern.compile("[A-Z ]");  
6      Pattern lowerCasePattern = Pattern.compile("[a-z ]");  
7      Pattern digitCasePattern = Pattern.compile("[0-9 ]");  
8      Pattern whiteSpacePattern = Pattern.compile("\\s+");  
9  
10     if (password.trim().length() < 8)  
11         return false;  
12  
13     if(whiteSpacePattern.matcher(password.trim()).find())  
14         return false;  
15  
16     if (!specialCharPattern.matcher(password).find())  
17         return false;  
18  
19     if (!upperCasePattern.matcher(password).find())  
20         return false;  
21  
22     if (!lowerCasePattern.matcher(password).find())  
23         return false;  
24  
25     return digitCasePattern.matcher(password).find();  
26 }
```

Il Grafo del Flusso di Controllo ottenuto dal codice è il seguente:



Di seguito vengono illustrati i metodi necessari ai fini di una Node e Branch Coverage totale del **GFC** precedentemente illustrato.

```
. @Test
. void isPasswordValid_1b_2b() {
.     assertFalse(Utilities.isPasswordValid(null));
. }

.

. @Test
. void isPasswordValid_1b_4b_5b_6b_7b_8b_10b_11b() {
.     assertFalse(Utilities.isPasswordValid("string"));
. }

.

. @Test
. void isPasswordValid_1b_4b_5b_6b_7b_8b_10b_13b_14b() {
.     assertFalse(Utilities.isPasswordValid("str _ing"));
. }

.

. @Test
. void isPasswordValid_1b_4b_5b_6b_7b_8b_10b_13b_16b_17b() {
.     assertFalse(Utilities.isPasswordValid("str123ing"));
. }

.

. @Test
. void isPasswordValid_1b_4b_5b_6b_7b_8b_10b_13b_16b_19b_20b() {
.     assertFalse(Utilities.isPasswordValid("nu0vap@ss"));
. }

.

. @Test
. void isPasswordValid_1b_4b_5b_6b_7b_8b_10b_13b_16b_19b_22b_23b() {
.     assertFalse(Utilities.isPasswordValid("NU0VAP@SS"));
. }

.

. @Test
. void isPasswordValid_1b_4b_5b_6b_7b_8b_10b_13b_16b_19b_22b_25b_invalid() {
.     assertFalse(Utilities.isPasswordValid("NuVAP@SS"));
. }

.

. @Test
. void isPasswordValid_1b_4b_5b_6b_7b_8b_10b_13b_16b_19b_22b_25b_valid() {
.     assertTrue(Utilities.isPasswordValid("Nu0VAP@SS"));
. }
```