UNIVERSITÀ DEGLI STUDI DEL SANNIO
Laurea Magistrale in Ingegneria Informatica

SWATCHER
Sistema di Video-Sorveglianza

SOFTWARE DESIGN SPECIFICATION

Corso di Ingegneria del Software
a.a.2014/2015
I A - II S

Studenti:
Antonio De Simone
Enrico Emanuele
Wilmer Ciasullo

Sommario

1 Introduzione

Il presente documento contiene l'SDD (Software Design Description) del sistema "Swatcher".

1.1 Obiettivo

L'obiettivo di questo documento è quello di illustrare l'architettura del sistema "Swatcher" fornendo una descrizione dettagliata delle componenti dalle quali è composto e le loro modalità di interconnessione.

La descrizione del sistema qui presentata è di supporto ad eventuali attività di sviluppo e manutenzione.

1.2 Scopo del Sistema

Swatcher è un sistema di video-sorveglianza remoto composto da un'applicazione Android e da una webcam.

L'intera applicazione realizzata è un tipico sistema Client/Server dove la parte Client è rappresentata dall'app. Android e la parte Server è costituita dalla webcam (device) e un opportuno sottosistema Server che la gestisce.

L'applicazione consente all'utente un'interazione in tempo reale costituita dalla visualizzazione dello streaming video, dalla cattura di foto e registrazione di video.

Il sistema, inoltre, è in grado di rilevare movimenti nell'ambiente sorvegliato ed avvertire l'utente dell'evento rilevato mediante notifica alla componente client.

1.3 Definizioni Acronimi Abbreviazioni

Termine	Definizione
Swatcher	Alias Security-Watcher, Nome del sistema
Utente	End-User che interagisce lato client con l'app mobile su un sistema Android
Sistema Client	Applicazione Android deployata/installata su uno smartphone
Sistema Server o Web-cam Server	Intero sottosistema lato server che comprende le seguenti componenti: Apache WebServer, WebCam Server, Raspberry, Notification System.
Snapshot	Instantanea fotografica richiesta on-demand lato client
Gallery	Intera collezione di Media presente sul Server
Media	Elemento multimediale (Immagine o Video) archiviato nella memoria del sistema.
Schermata Gallery o Media Gallery	GUI dell'app che mostra l'anteprima su mobile di tutti i media presenti nel sistema di memorizzazione lato server.
SAA o S2A	Swatcher Android Application (o modulo SAA/S2A) è l'applicazione Android deployata sul sistema client (Smartphone).
SSS o S3P	Swatcher Sub-System Server Platform è l'intero modulo Server del sistema Swatcher residente sulla scheda Raspberry-Pi.
AWS	Apache Web Server integrato all'interno di Raspberry-Pi
FS	File System di archiviazione dei contenuti lato server. È contenuto all'interno del modulo S3M
MWS	Motion WebCam Server. Sottosistema lato server che si occupa di gestire i comandi provenienti dall'App Client. Contiene al suo interno il modulo EVM.
EVM	Event Manager. Modulo presente all'interno del sottosistema MWS. L'EVM è in grado di intercettare gli eventi provenienti dalla WebCam di gestirli inviando una notifica al NOM o salvando su disco i media prodotti a seguito dell'evento intercettato(Snapshot e/o Video).
NOM	Notification Manager

2 Architettura del Sistema

Swatcher è stato realizzato partendo completamente da zero senza prendere come riferimento alcun sistema pre-esistente (Greenfield Project Engineering) pertanto la sua struttura è stata definita a valle di una raccolta di requisiti che sono presenti nell'apposito documento di "Requirement Specification".

2.1 Struttura e collegamenti delle componenti del sistema

Il sistema Swatcher è composto da 3 sottosistemi dislocati su 3 nodi differenti ed interagenti tra di loro:

1. Un sottosistema denotato come “Client” costituito da uno Smartphone dotato di sistema operativo Android.
2. Un sottosistema denotato come “Server” costituito da un single-board computer Raspberry-Pi collegato ad una webcam remota.
3. Un sottosistema di “Notifica”utilizzato in modalità SaS (Software as a Service) attraverso il servizio Firebase Cloud Messaging di Google.

I due sottosistemi appena elencati nei punti 1. e 2. sono stati definiti come “Client” e “Server” anche se il funzionamento complessivo del sistema non sempre rispetta un modello Client/Server puro: esistono determinate condizioni in cui la componente deployata sulla piattaforma Raspberry-Pi si comporta come un Peer interagente inviando una notifica al client attraverso il sottosistema di Notification Manager.

Lo scenario appena descritto si manifesta al verificarsi di un evento che viene intercettato e gestito autonomamente dall’Event Manager, un modulo o sottosistema appartenente alla componente ‘Server’. In tutti gli altri scenari di funzionamento il sottosistema Raspberry-Pi si comporta effettivamente come server dell’Applicazione Android residente sullo Smartphone.

La fruizione dei servizi resi disponibili dalla piattaforma server è possibile soltanto attraverso dispositivi dotati di sistema operativo Android; non è stato previsto uno sviluppo del sistema per poter interagire anche con dispositivi dotati di sistema operativo iOS.

Nell’immagine seguente (Figura 1) viene riportata una prima descrizione dell’architettura complessiva del sistema indicando come l’intero sistema è decomposto nelle varie componenti e comunicano tra di loro nelle loro principali funzionalità.

La comunicazione attraverso i 3 moduli che compongono il sistema avviene ovviamente sempre per mezzo del protocollo http.

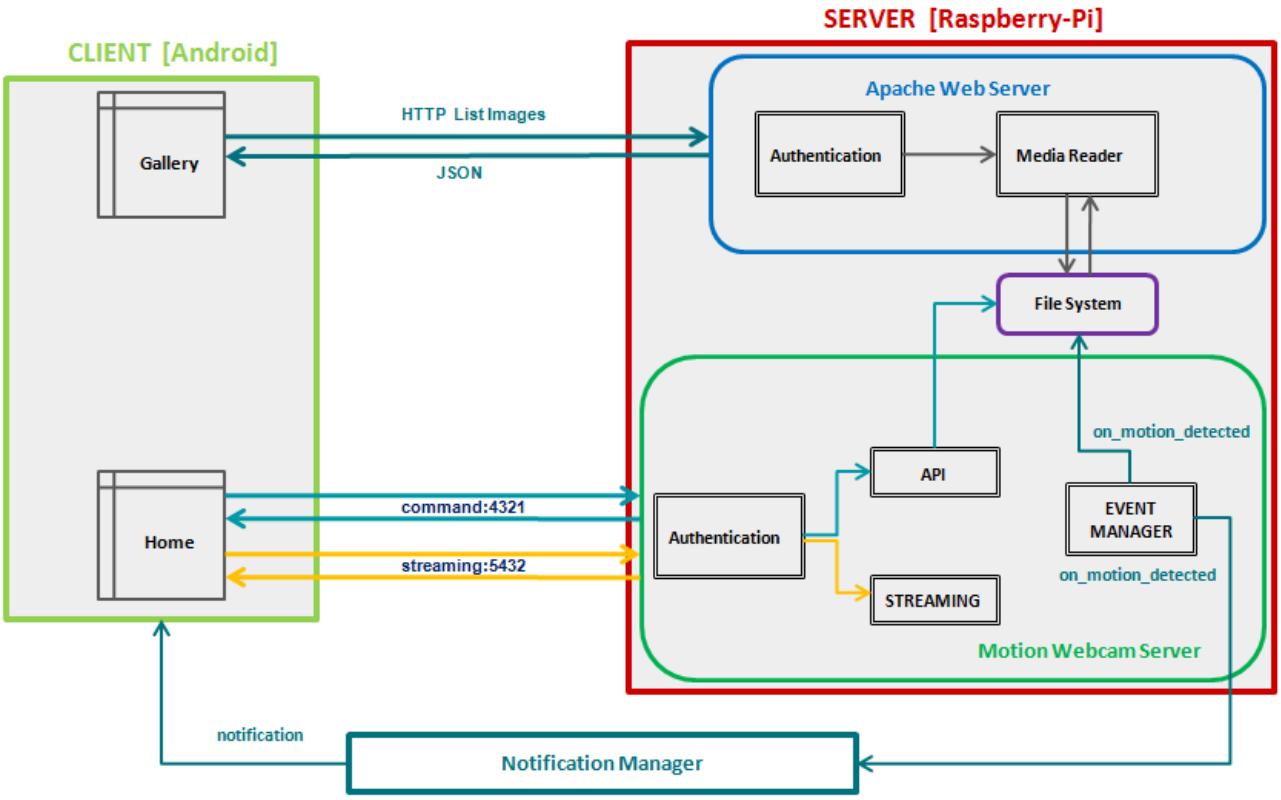

Figura 1: Una prima vista Architetturale del Sistema Swatcher

Ai fini di un agevole identificazione e classificazione di tutte le componenti e sottocomponenti del sistema ad ognuno di queste è stato associato uno specifico identificatore/acronimo come riportato nel paragrafo delle definizioni 1.3.

2.2 Sottosistema Client

Il sottosistema Client è costituito da un app. deployata su un qualsiasi smart-phone Android.

Nella prima Overview del sistema (Figura 1) sono state esplicitate le GUI "Home" e "Gallery" che sono due videate fondamentali dell'App di Swatcher quindi usate per comprendere le principali interazioni e modalità di comunicazione che il sottosistema client intraprende con la restante parte del sistema (Server).

La videata "Gallery" permette di richiedere al server l'intero listato dei file multimediali che sono stati archiviati nel file system della componente server; ciò avviene attraverso una richiesta http inoltrata dall'App al sottosistema Server.

A seguito di tale richiesta l'anteprima di dettaglio dell'intera collezione presente in remoto viene passata all'App in formato JSON.

La videata “Home” permette l’esecuzione di due funzionalità principali:

1. La visualizzazione dello streaming proveniente dalla Webcam
2. La possibilità di scattare istantanee (Snapshot) memorizzandole opportunamente in remoto lato server.

Entrambe le funzionalità sopra elencate sono realizzate mediante l’opportuno scambio di comandi tra l’App Mobile e l'MWS.

2.3 Sottosistema Server

Il Server è costituito da 3 sottocomponenti residenti sul single-board computer RaspBerry-Pi:

1. Motion Web-cam Server (MWS)
2. Apache Web Server (AWS)
3. File System (FS)

Dei tre componenti appena elencati l'MWS e l'AWS sono costituiti da ulteriori moduli interni che verranno descritti di seguito.

Un esemplificazione di ciò che è stato appena detto relativamente alla componente server è presente nell’immagine seguente.

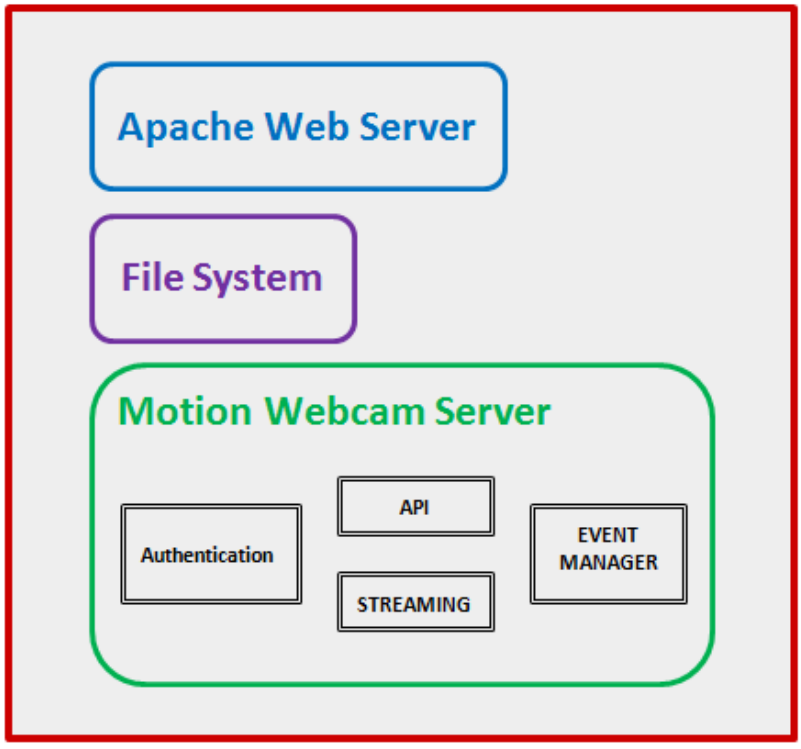

Figura 2: Sottosistema Server residente sulla Piattaforma Raspberry-Pi

2.3.1MWS – Motion Webcam Server

Il Motion Webcam Server è composto dalle seguenti componenti fondamentali:

1. Authentication o Modulo di Autenticazione
2. API o Modulo delle Funzionalità
3. Streaming
4. Event Manager

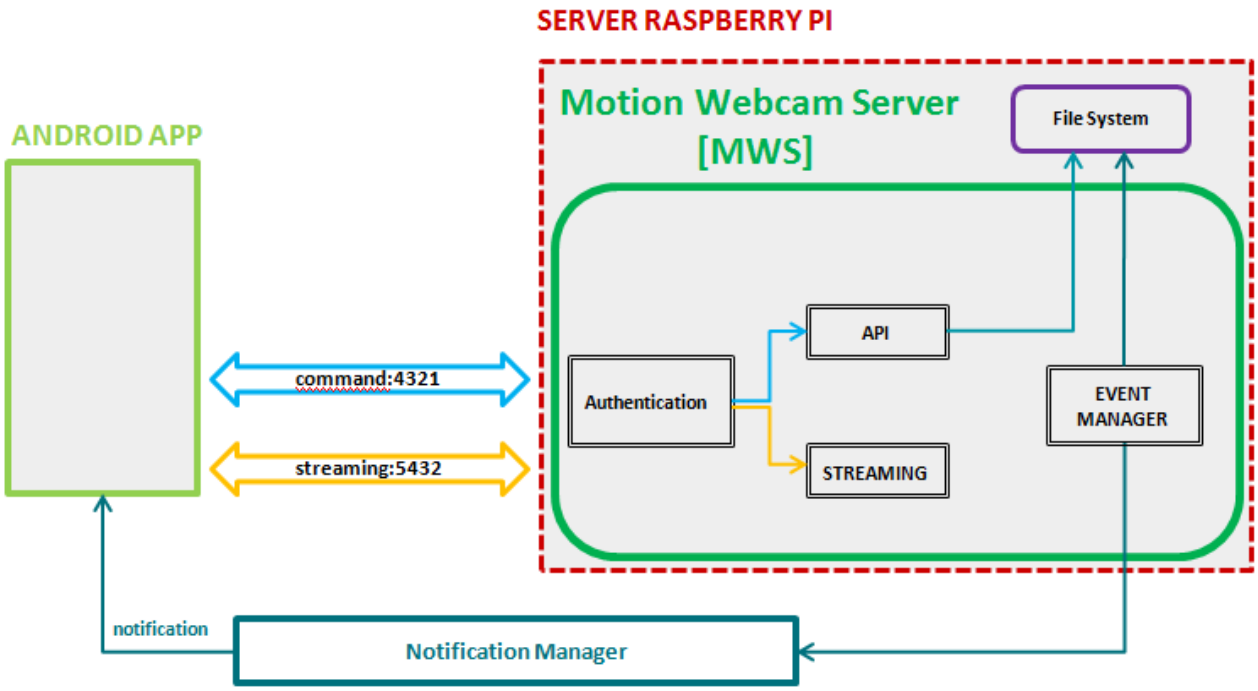

Figura 3: Composizionedell'MWS e loro interazioni interne e esterne al modulo

Il modulo di "Autenticazione" fa da Buffer separatore tra i comandi ricevuti dall'App-Android e gli altri due moduli interni di Streaming e API al fine di filtrare le richieste provenienti dal client, validarle e dirigerle al Modulo API o Streaming

Il modulo "Streaming" si occupa della gestione dello stream video proveniente dalla webcam, in particolare abilita la trasmissione dopo aver ricevuto una richiesta dal client e si occupa quindi di redigere il flusso dati in maniera opportuna all'app mobile.

Il modulo API gestisce i command provenienti dall'App mobile relativi al prelievo dei media richiesti.

L'Event Manager è l'unico componente attivo del sistema: in corrispondenza del rilevamento di un movimento (on_motion_detected) invia un messaggio al Notification Manager che a sua volta invia una notifica all'App mobile.

2.3.2 AWS – Apache Web Server

Sulla scheda Raspberry-Pi è installato un Web-Server Apache che mette a disposizione della componente client il set di funzionalità previsto per il sistema e l'accesso ai media, l'AWS rappresenta quindi l'interfaccia software tra il client e l'intero sottosistema server.

Tutte le richieste sottomesse all'AWS (e.g. richiesta di un media presente nel file-system del server) sono ovviamente richieste Http: in corrispondenza di una richiesta l'AWS si fa carico di prendere il media richiesto dal client e inviarglielo tramite uno script PHP.

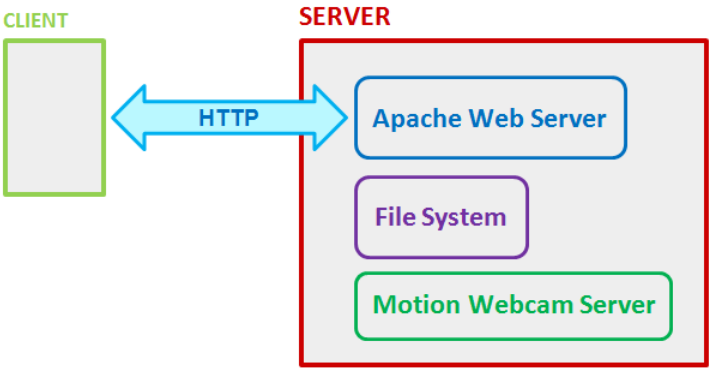

Figura 4: Interfaccia Client/Server e sottosistema server

2.3.2 File System

I media (foto e video) registrati dal Motion Webcam Server vengono inseriti nel File System e recuperati successivamente dall'AWS.

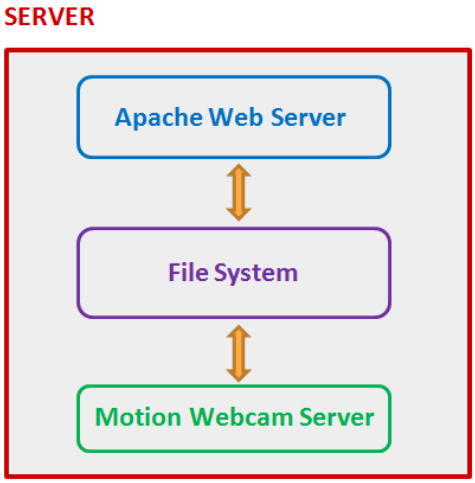

Figura 5: File System e interazione MWS/AWS

2.4 Sottosistema di Notifica

Il sottosistema di notifica è implementato attraverso un servizio esterno al sistema presso cui il client si registra rispetto ad un 'topic'effettuando un'opportuna autenticazione.

Il server pubblica sullo stesso 'topic' nel momento in cui si verifica un evento di motion detection.

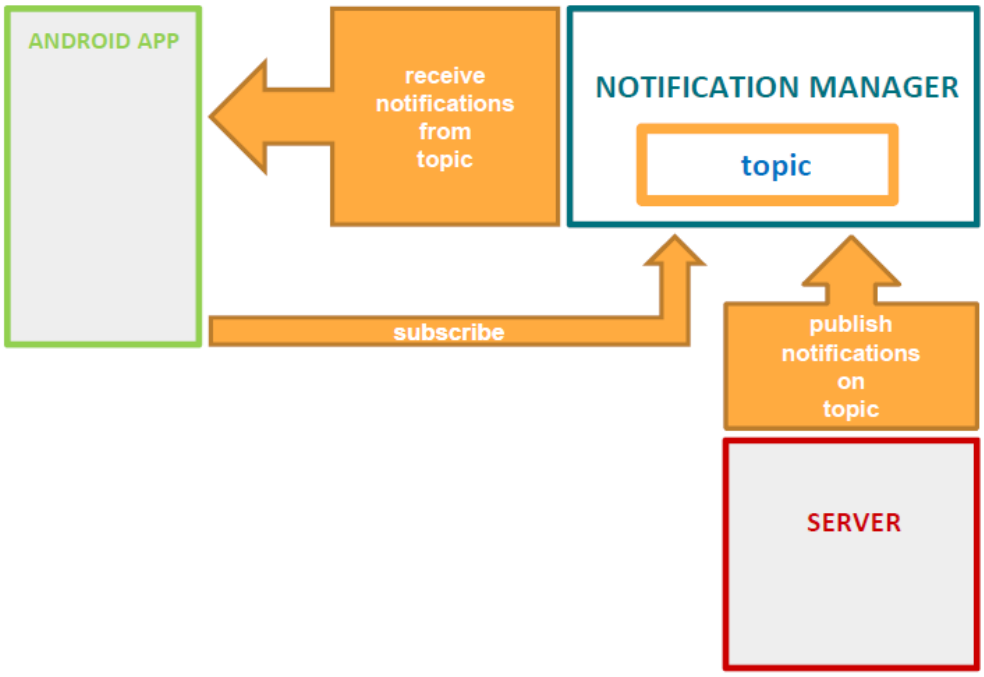

Figura 6: sottosistema di Notifica in Swatcher

Il sottosistema di notifica è quindi implementato secondo un tipico modello Publish/Subscribe.

2.5 Sottosistema di Comunicazione

In Swatcher la comunicazione tra le componenti è interamente basata su scambio di messaggi http. Nel codice sorgente dell'applicazione client (App. Android) due sono le Classi/Interfacce fondamentali su cui si basa l'interazione con il resto del sistema: HttpRequest e CommandInterface.

HttpRequest è una classe Astratta caratterizzata da:

- 4 proprietà accessibili attraverso gli opportuni metodi getters/setters e da
- 1 metodo 'getURL' vuoto di cui è definita soltanto la signature e che viene implementato dalle sottoclassi classi che estendono concretamente HttpRequest

L’immagine seguente riporta uno skeleton della classe astratta “HttpRequest”.

```
package sweng.swatcher.request;

import sweng.swatcher.model.Authorization;

public abstract class HttpRequest {

    private String ipAddress;
    private String port;
    private Authorization authorization;
    private String response;

    public HttpRequest(String ip,
            String port, Authorization auth) {...}

    <Getters and Setters Methods>

    public abstract String getURL();

}
```

Figura 7: Classe Astratta HttpRequest

L’immagine seguente riporta l’intero set di classi “concrete” che estendono/implementano la classe astratta “HttpRequest” e sono contenute all’intero del package “request”.

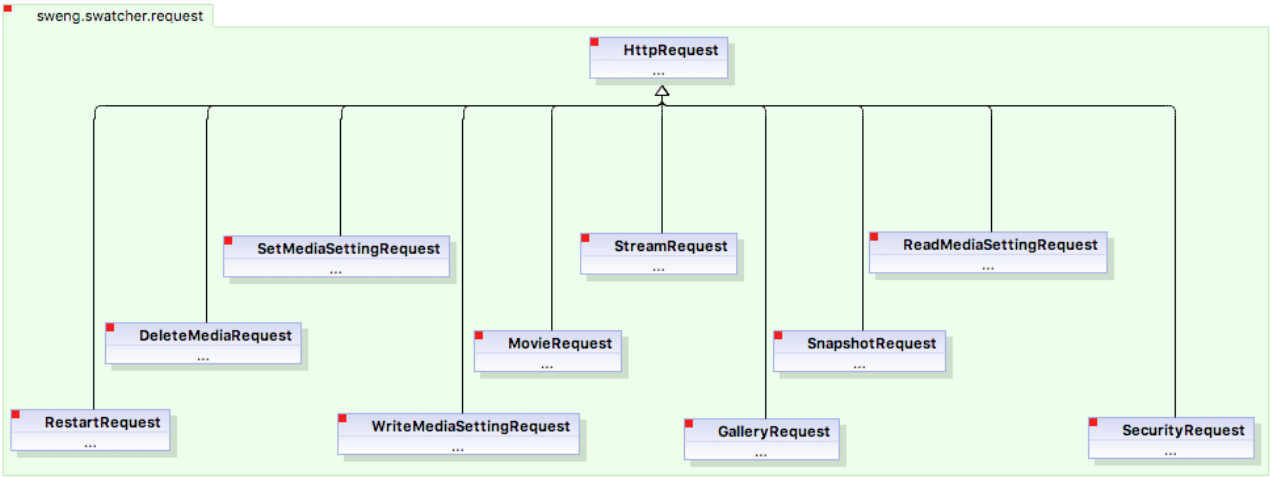

Figura 8: Famiglia delle classi HttpRequest

Tutti i comandi avvengono solo ed esclusivamente attraverso l’interfaccia “CommandInterface”che è caratterizzata dal metodo “execute”.

Tutte le specifiche azioni che il client compie nei confronti del resto del sistema sono incapsulate nelle diverse classi “command” implementando l’interfaccia “CommandInterface” e fornendo una specifica realizzazione del metodo “execute”.

```java
package sweng.swatcher.command;

public interface CommandInterface {
    public void execute();
}
```

Figura 9: CommandInterface

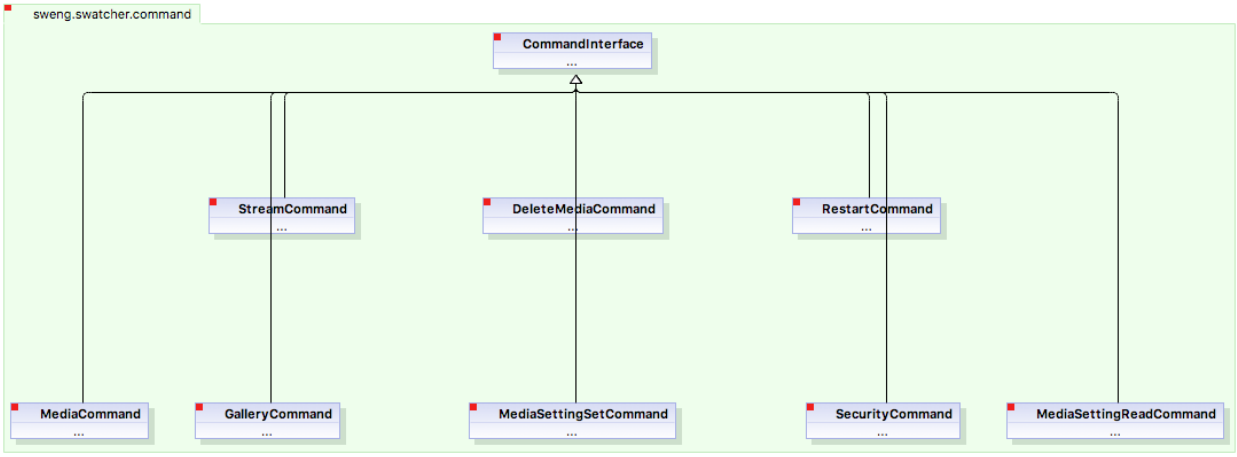

Figura 10: Interfaccia Command e classi che la implementano

L'interfaccia "CommandInterface" e le relative classi che la implementano sotto contenute nel package "command".

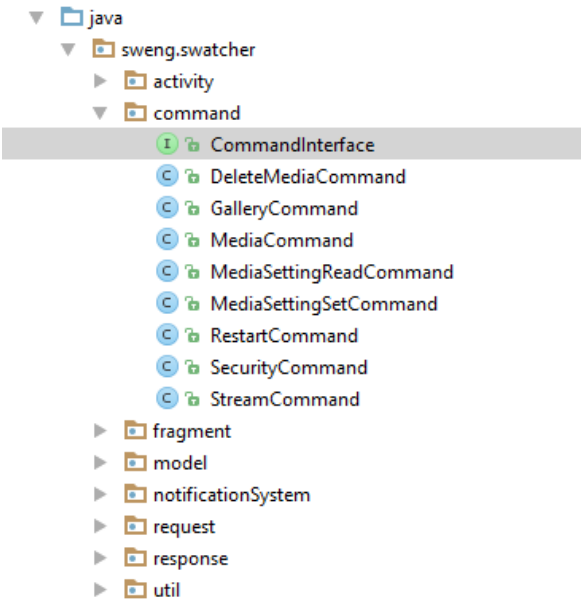

Figura 11: package command

2.6 Diagrammi di Interazione

In questa sezione vengono presentati alcuni sequence diagram che illustrano i principali meccanismi di interazioni tra le istanze del sistema relativamente ai requisiti funzionali più rilevanti.

2.6.1 Configurazione Connessione

Le immagini seguenti mostrano i sequence diagram relativi al requisito funzionale FR-02 riportando sia lo scenario primario FR-02 che lo scenario alternativo FR-02-A.

2.6.1.1 Diagramma di sequenza relativo allo scenario principale (SFR-02)

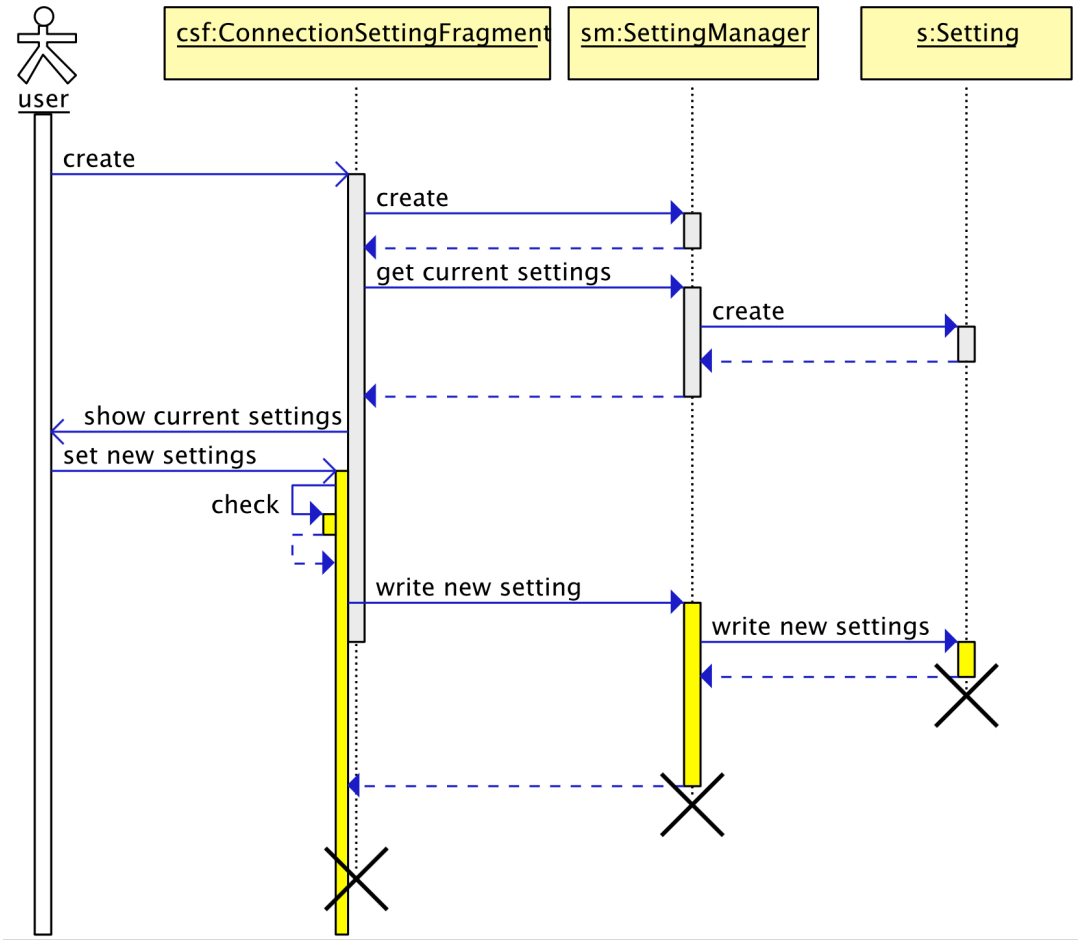

Figura 12: Sequence Diagram del requisito funzionale FR-02

2.6.1.2 Diagramma di sequenza relativo allo scenario alternativo (SFR-02-A)

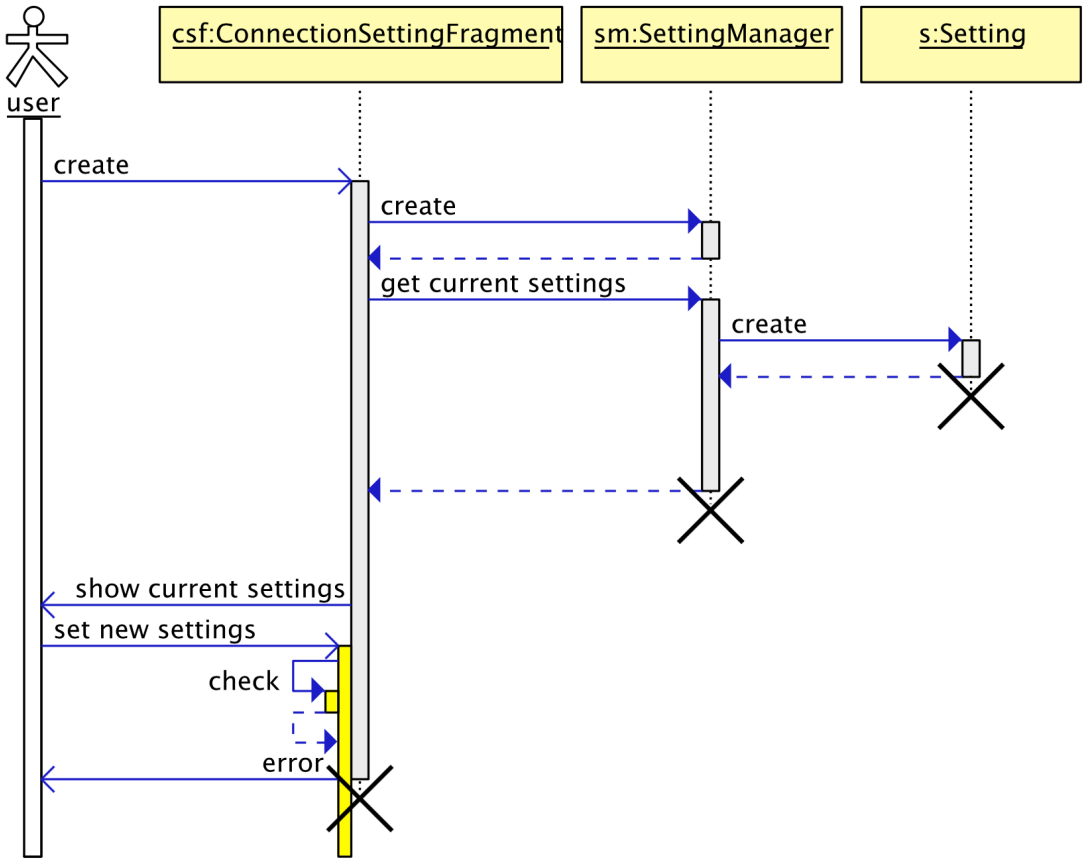

Figura 13: Sequence Diagram del requisito funzionale FR-02-A

2.6.2 Visualizzazione Streaming Video

Le immagini seguenti mostrano i sequence diagram relativi al requisito funzionale FR-03 riportando sia lo scenario primario FR-03 che lo scenario alternativo FR-03-A.

2.6.2.1 Diagramma di sequenza relativo allo scenario primario (SFR-03)

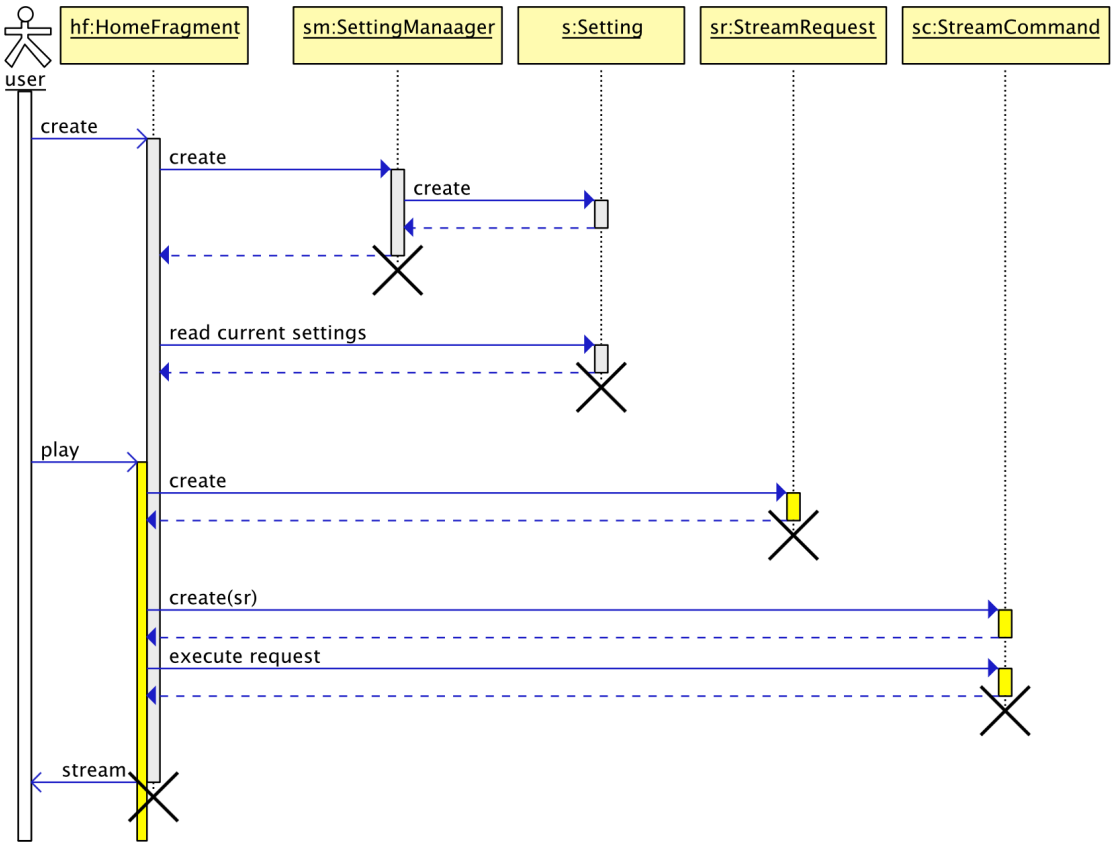

Figura 14: Sequence Diagram del requisito funzionale FR-03

2.6.2.2 Diagramma di sequenza relativo allo scenario alternativo (SFR-03-A)

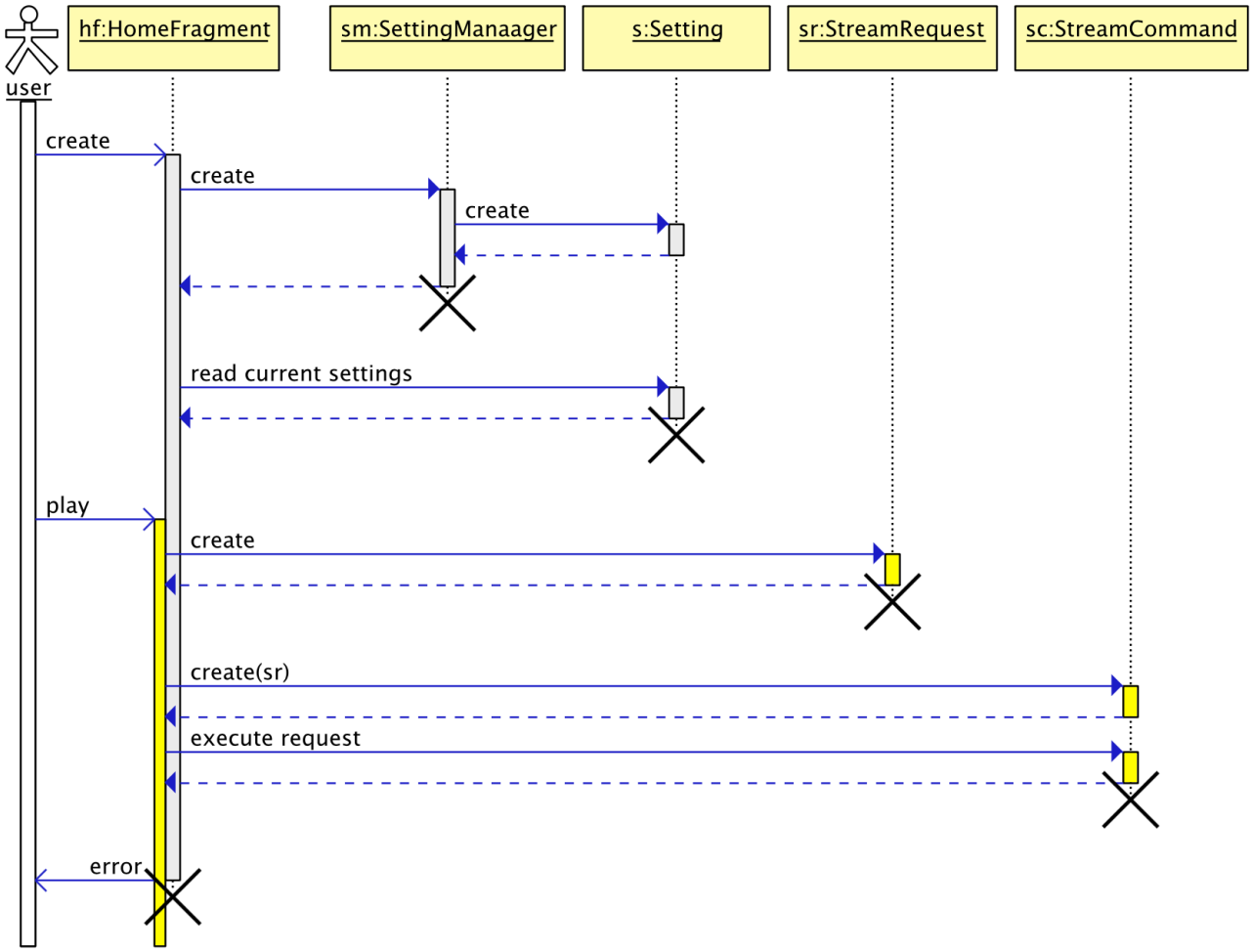

Figura 15: Sequence Diagram del requisito funzionale FR-03-A

2.6.3 Snapshot

Le immagini seguenti mostrano i sequence diagram relativi al requisito funzionale FR-04 riportando sia lo scenario primario FR-04 che lo scenario alternativo FR-04-A.

2.6.3.1 Diagramma di sequenza relativo allo scenario primario (SFR-04)

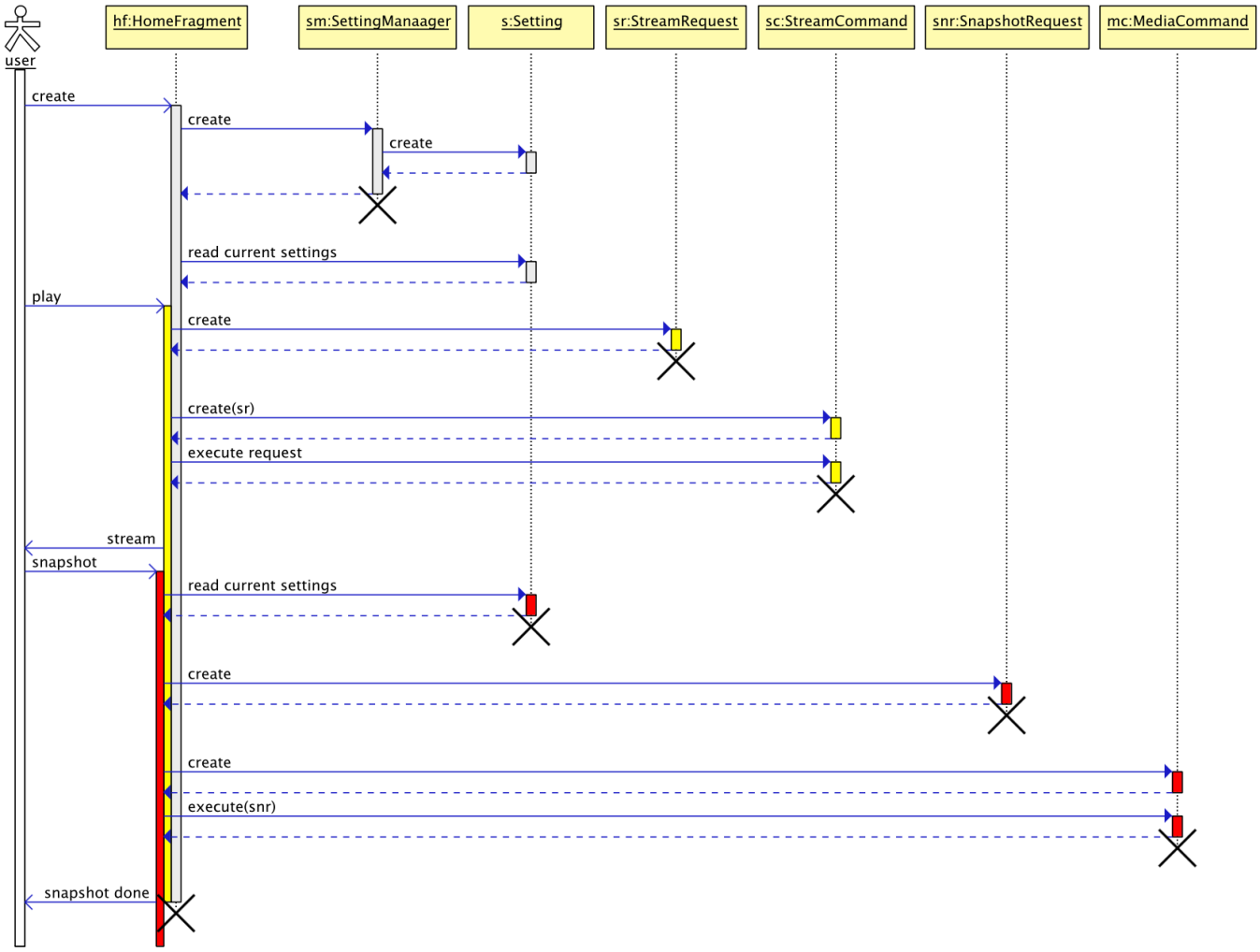

Figura 16: Sequence Diagram del requisito funzionale FR-04

2.6.3.2 Diagramma di sequenza relativo allo scenario primario (SFR-04-A)

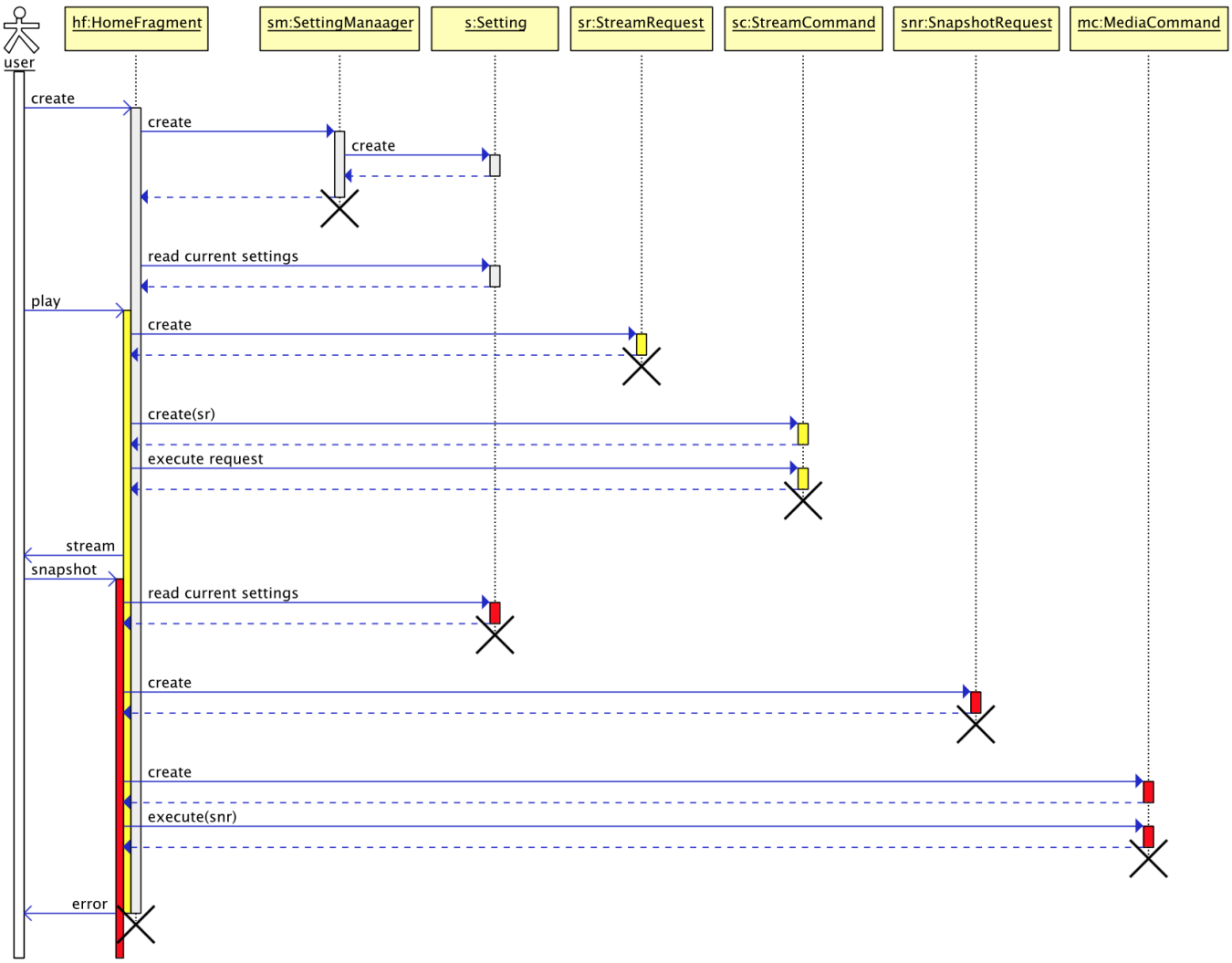

Figura 17: Sequence Diagram del requisito funzionale FR-04-A

2.6.4 Consultazione Gallery

Le immagini seguenti mostrano i sequence diagram relativi al requisito funzionale FR-05 riportando sia lo scenario primario FR-05 che lo scenario alternativo FR-05-A.

2.6.4.1 Diagramma di sequenza relativo allo scenario primario (SFR-05)

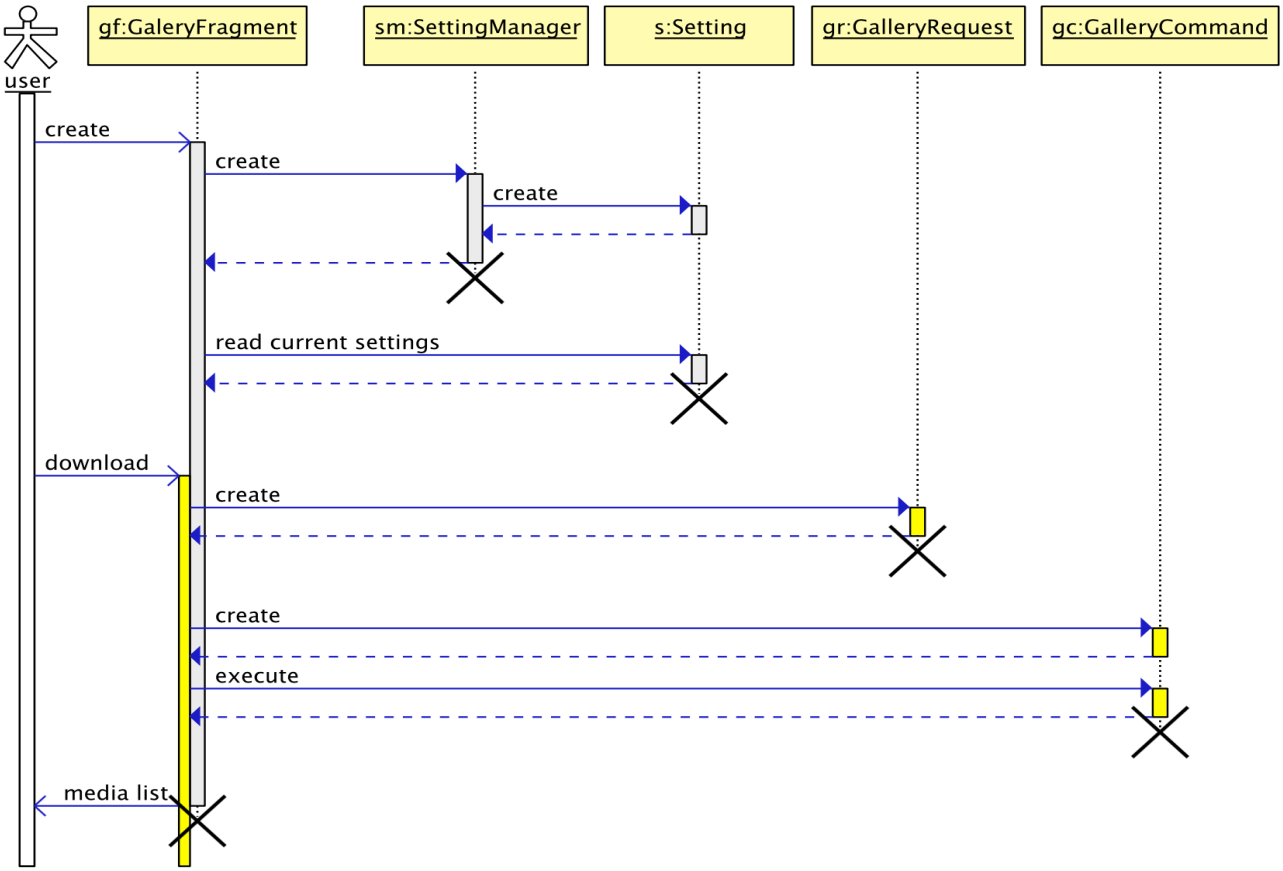

Figura 18: Sequence Diagram del requisito funzionale FR-05

2.6.4.2 Diagramma di sequenza relativo allo scenario alternativo (SFR-05-A)

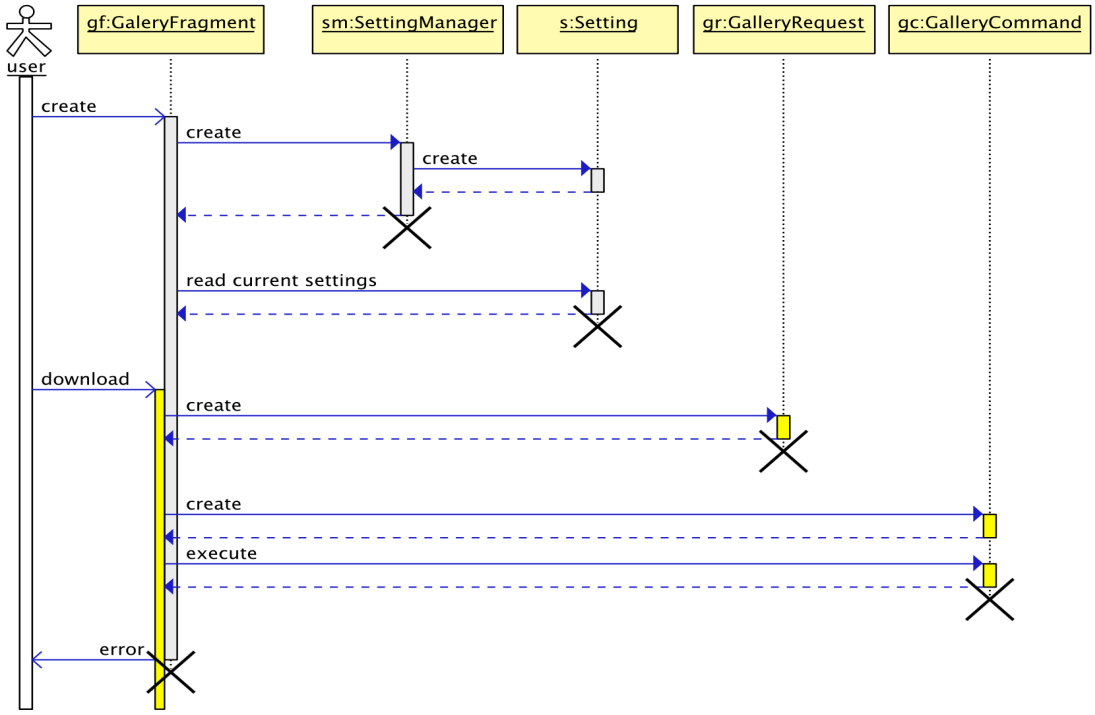

Figura 19: Sequence Diagram del requisito funzionale FR-05-A

2.6.5 Visualizzazione del singolo media

Le immagini seguenti mostrano i sequence diagram relativi al requisito funzionale FR-06.

2.6.5.1 Diagramma di sequenza relativo allo scenario primario (SFR-06)

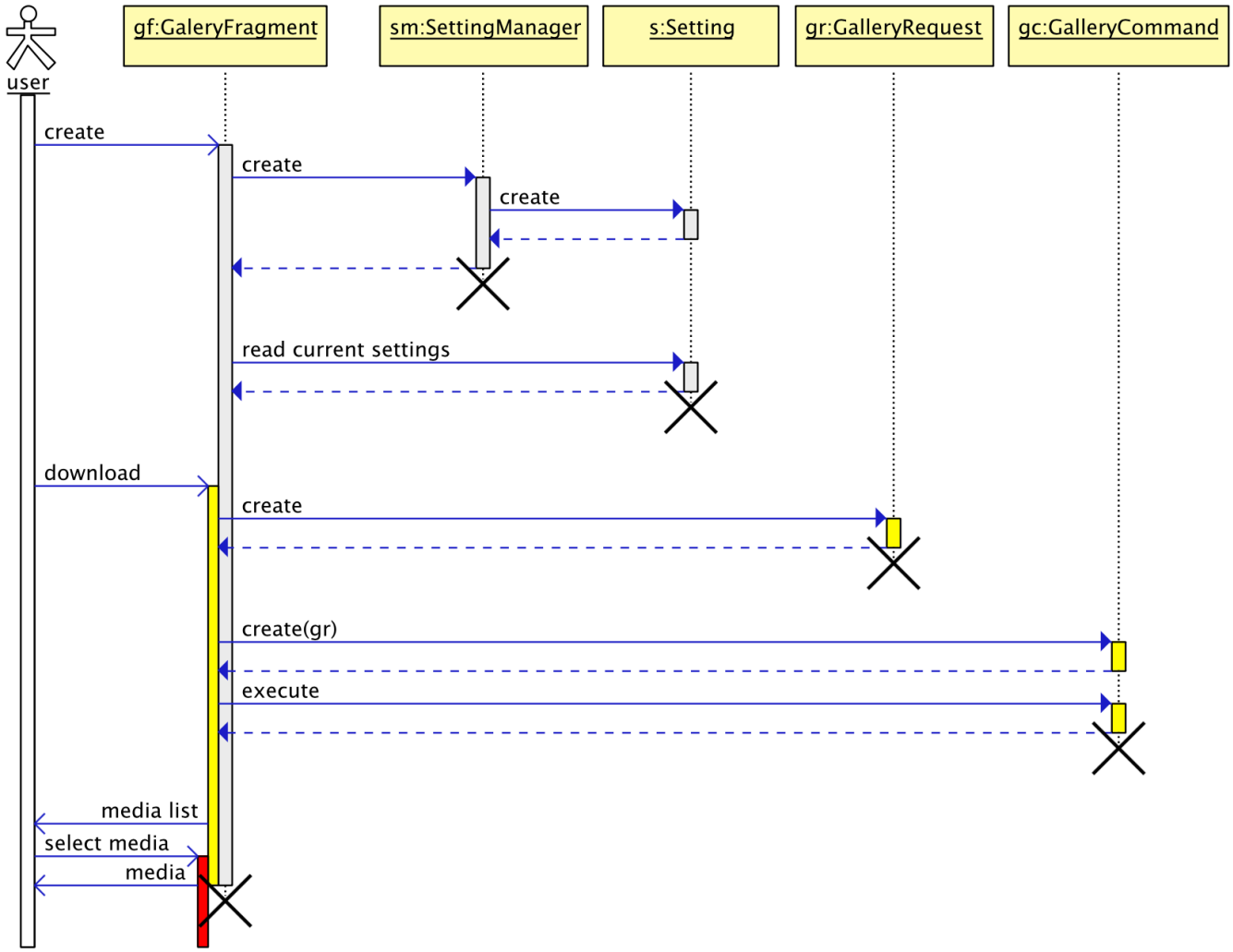

Figura 20: Sequence Diagram del requisito funzionale FR-06

2.6.6 Configurazione Parametri Motion

Le immagini seguenti mostrano i sequence diagram relativi al requisito funzionale FR-08.

2.6.6.1 Diagramma di sequenza relativo allo scenario primario (SFR-08)

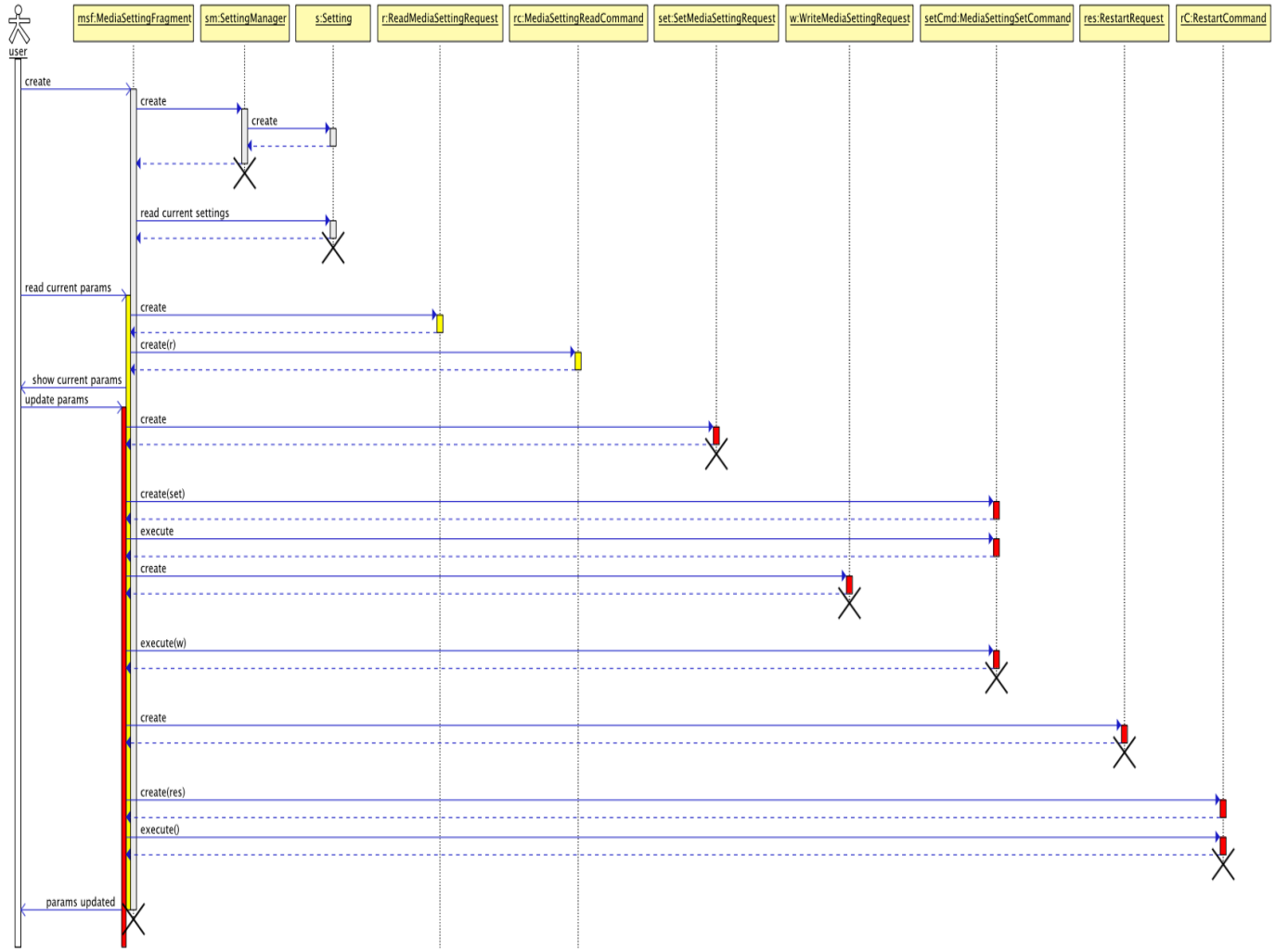

Figura 21: Sequence Diagram del requisito funzionale FR-08

2.7 Diagrammi delle Classi

In questa sezione vengono presentati i principali diagrammi delle Classi che illustrano la struttura e l'organizzazione del codice sorgente dell'app mobile Android.

In allegato nell'immagine seguente il diagramma delle classi complessivo, tutti gli altri diagrammi verranno trattati nei paragrafi successivi.

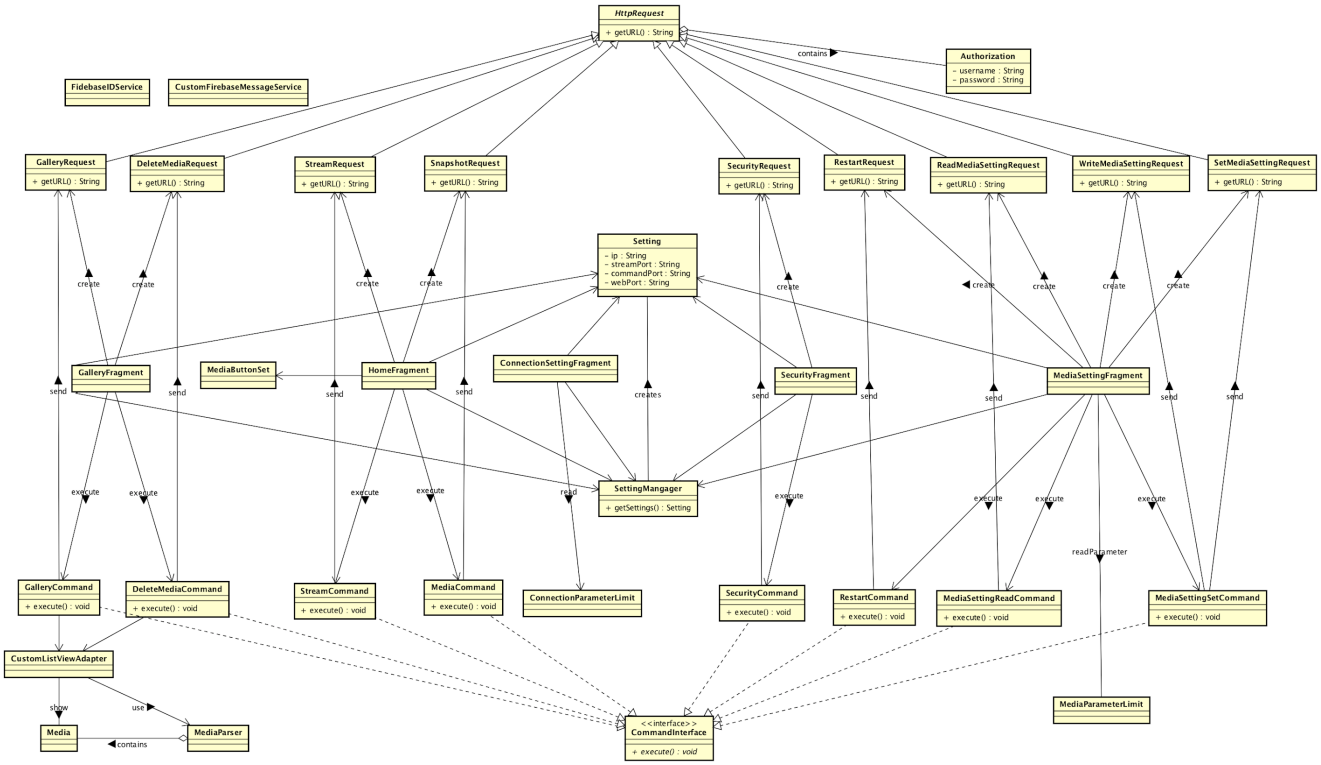

Figura 22: Diagramma delle classi complessivo

2.7.1 Package Command

In allegato il diagramma delle classi del package Command.

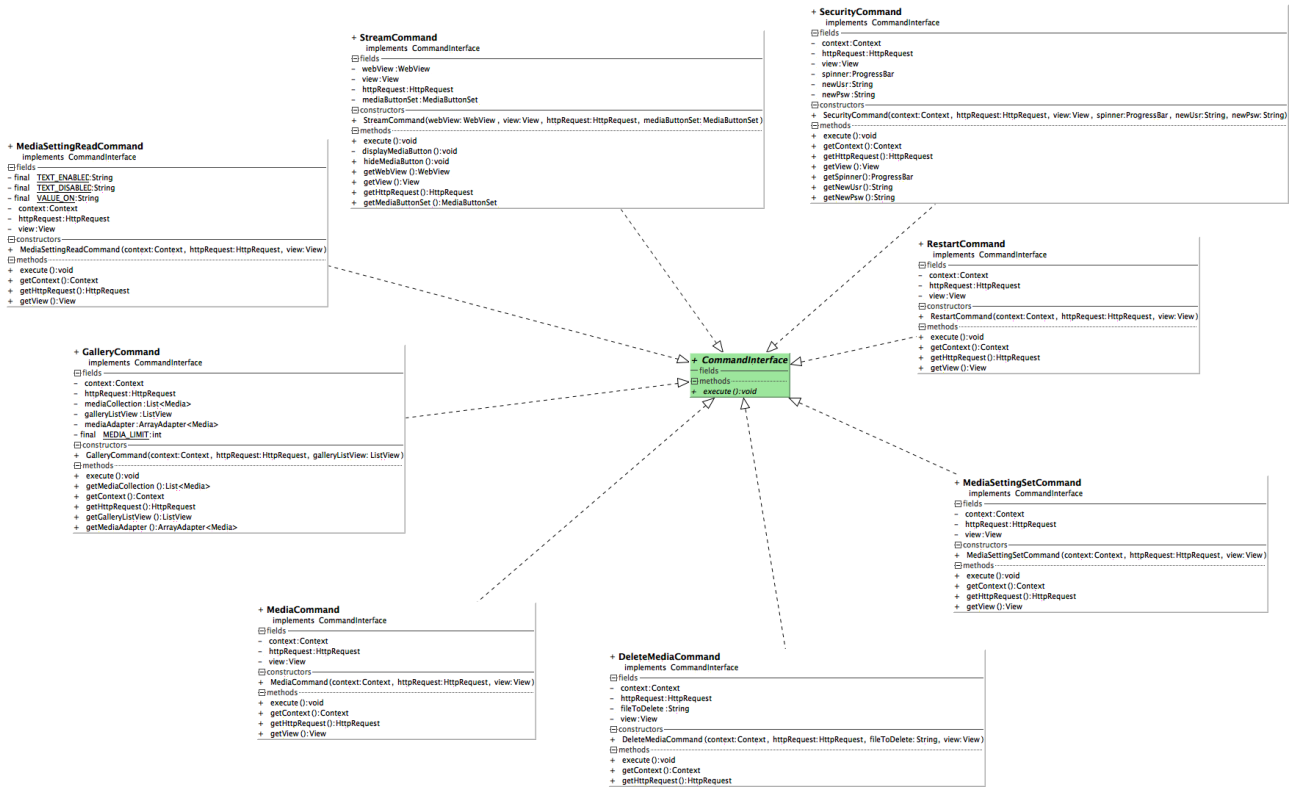

Figura 23: Diagrammi delle classi del Package Command

Tutte le classi command implementano l'interfaccia CommandInterface fornendo una propria realizzazione del metodo execute.

2.7.2 Package Request

In allegato il diagramma delle classi del package Request.

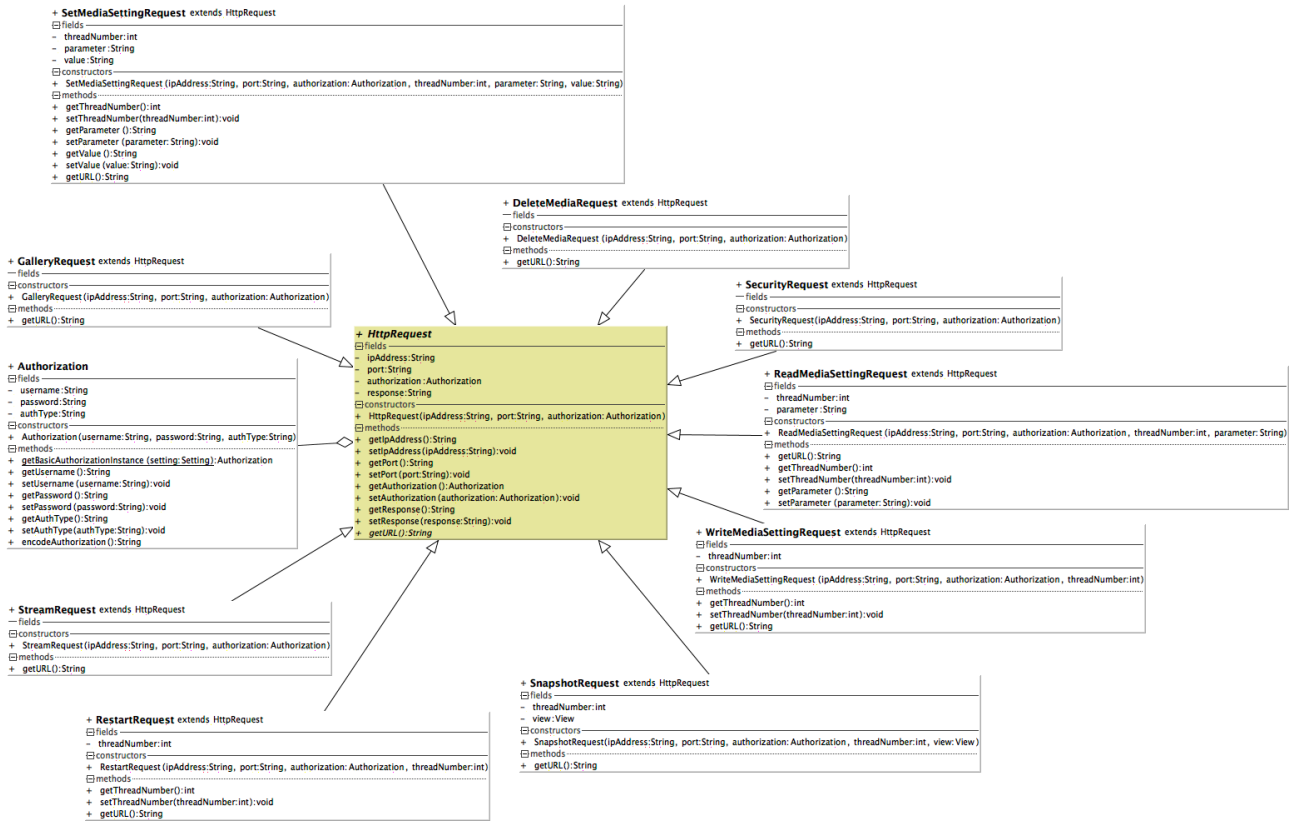

Figura 24: Diagramma delle classi del package Request.

Tutte le classi del package estendono la classe astratta HTTPRequest ereditando le proprietà:

- ipAddress
- port
- authorization
- response

ed implementando il metodo astratto:

- public abstract String getURL();

2.7.3 Package Fragment

In allegato i diagrammi delle principali classi appartenenti al package Fragment.

2.7.3.1 ConnectionSettingFragment

Nell'immagine seguente è illustrato il diagramma delle classe ConnectionSettingFragment

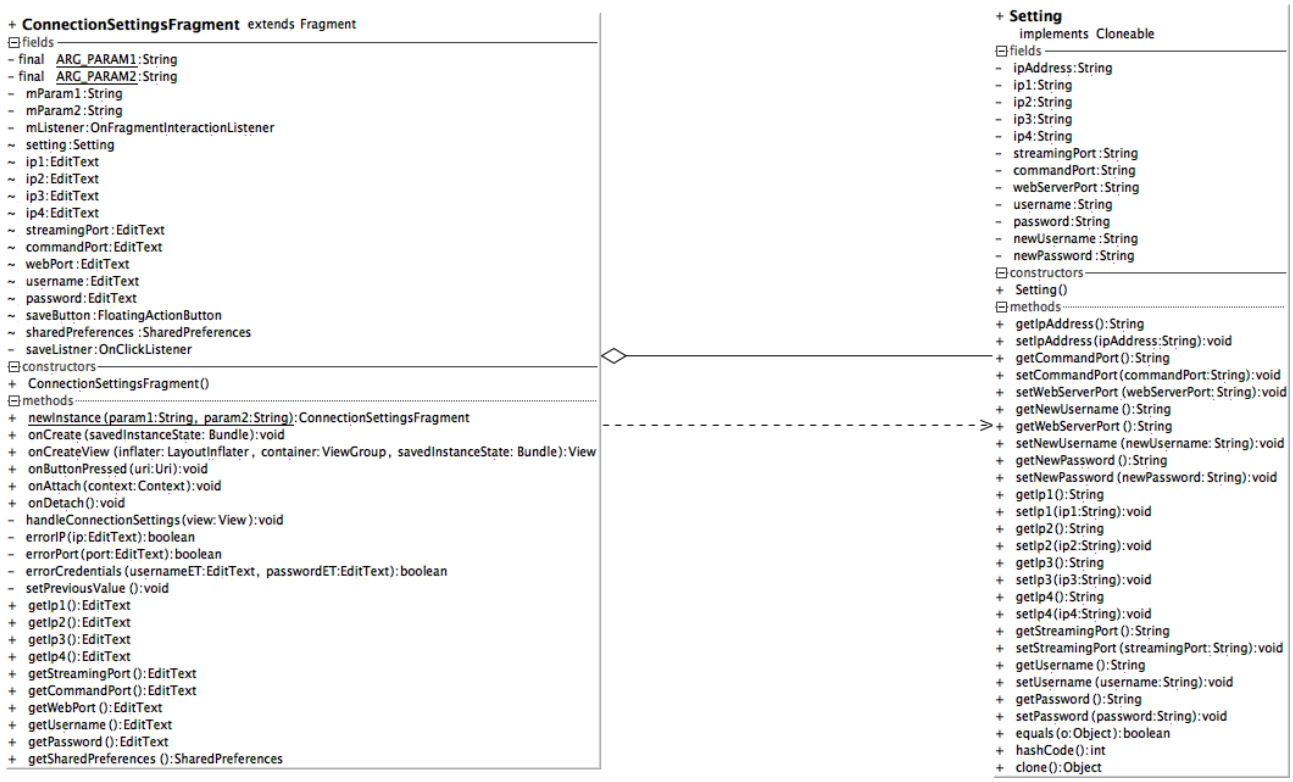

Figura 25: Diagramma della classe ConnectionSettingFragment

2.7.3.2 GalleryFragment

In allegato il diagramma della classe GalleryFragment.

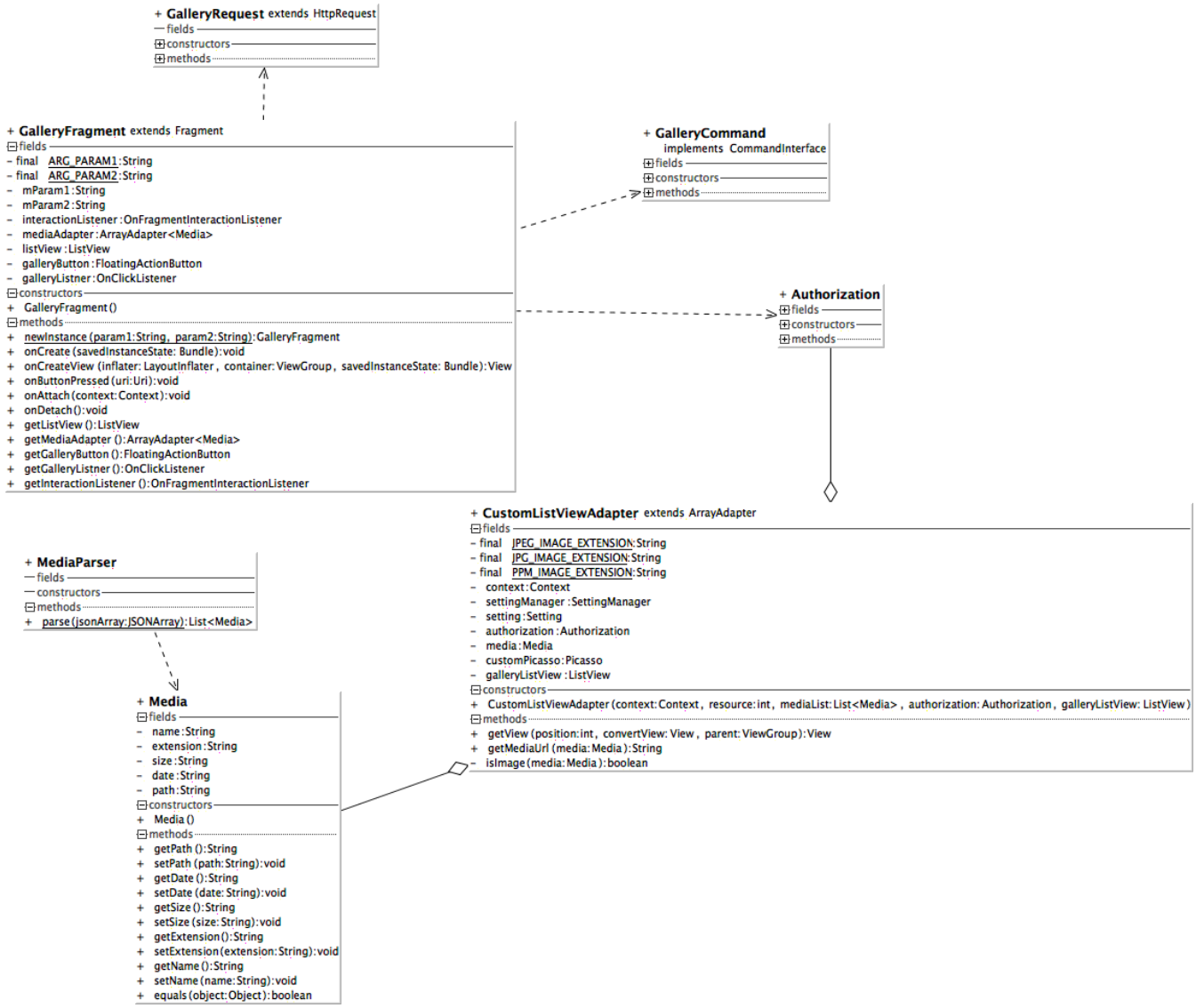

Figura 26: Diagramma della classe GalleryFragment

2.7.3.3 HomeFragment

In allegato il diagramma della classe HomeFragment.

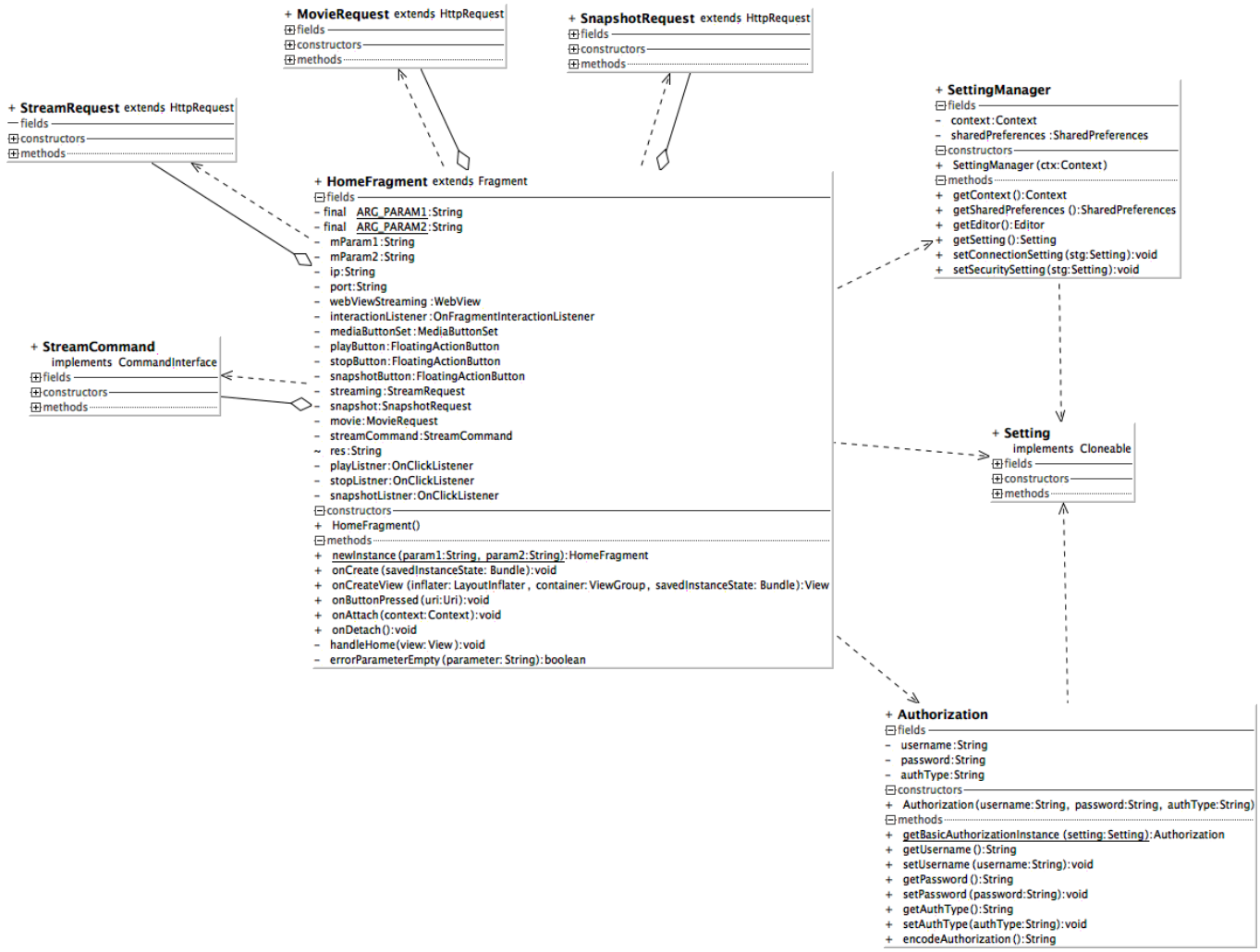

Figura 27: Diagramma della classe HomeFragment

2.7.3.4 MediaSettingFragment

In allegato il diagramma della classe MediaSettingFragment.

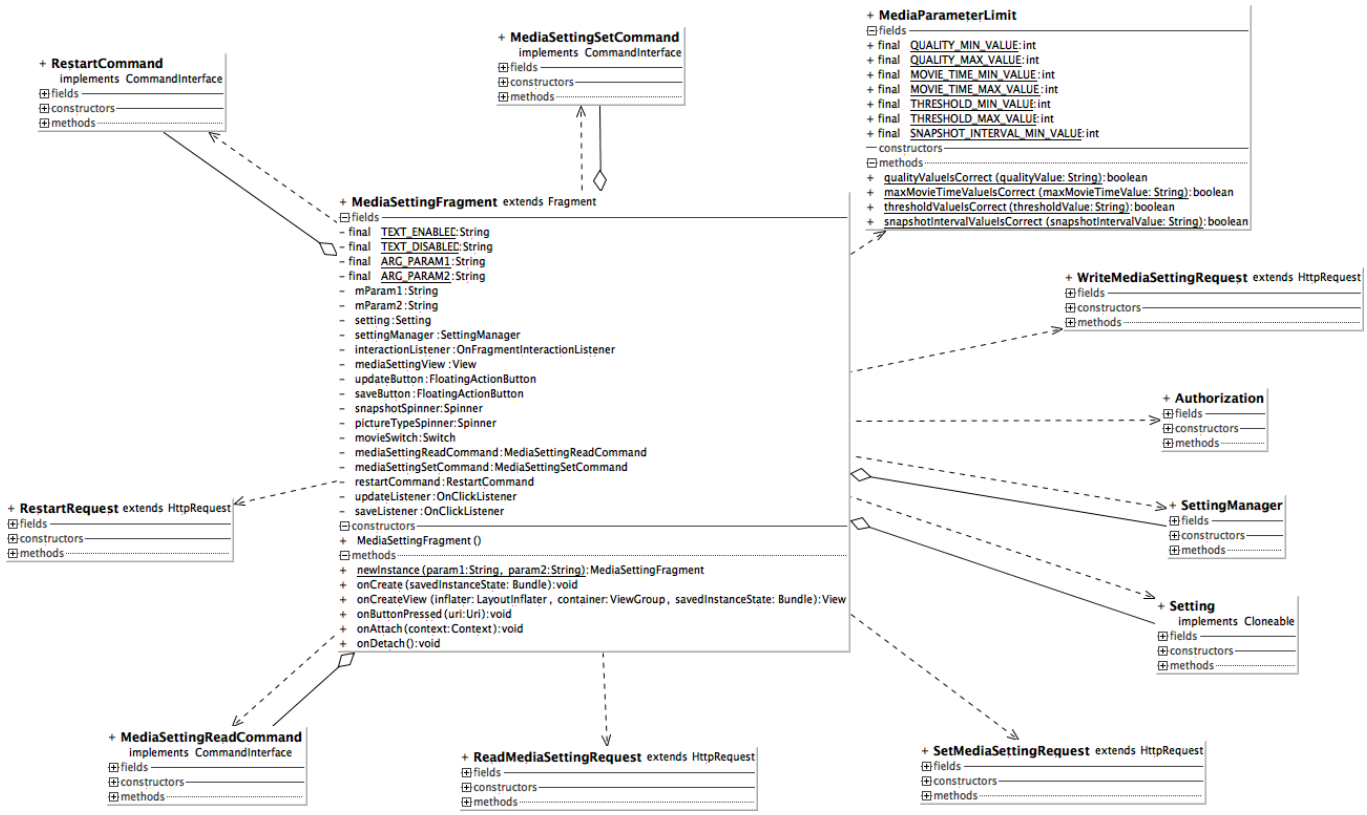

Figura 28: Diagramma della classe MediaSettingFragment

2.7.3.5 SecurityFragment

In allegato il diagramma della classe SecurityFragment.

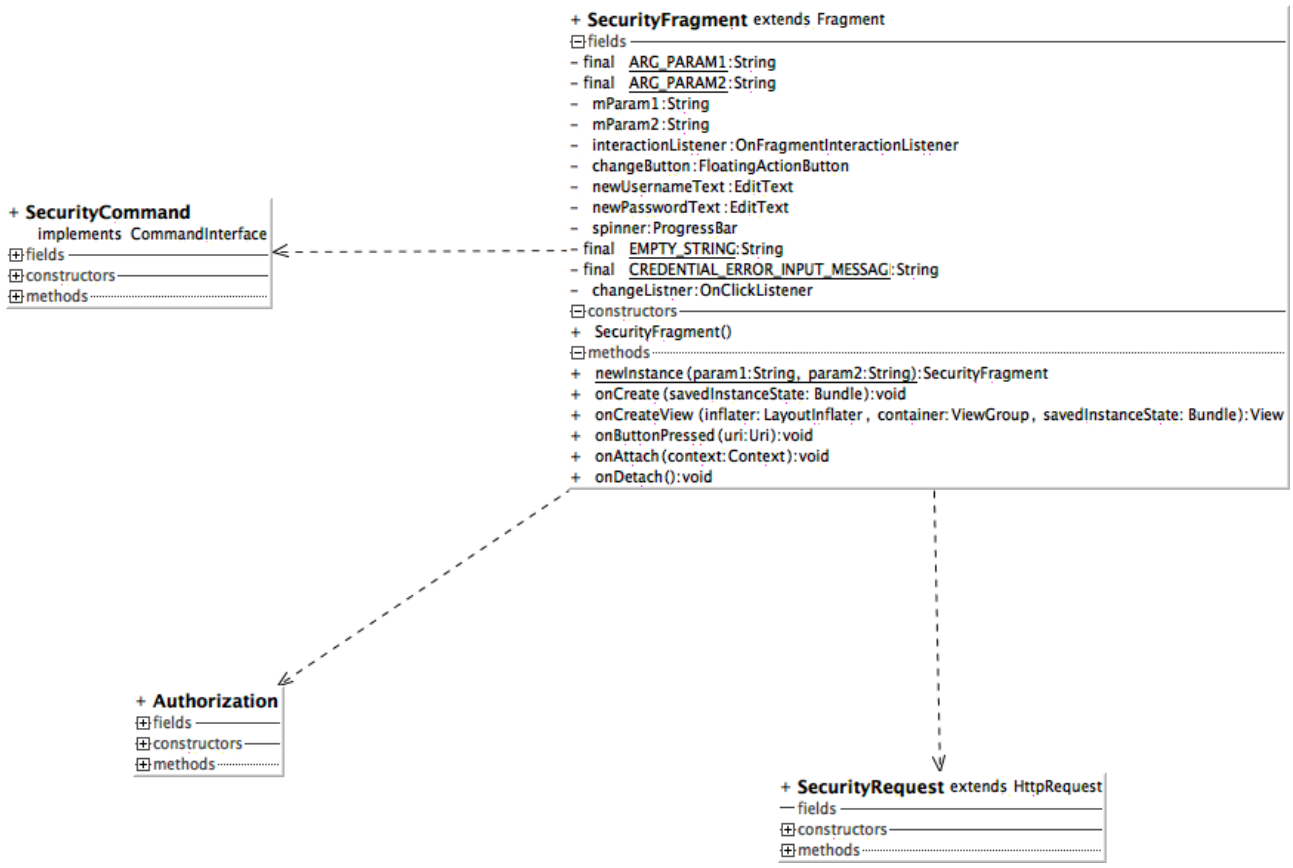

Figura 29: Diagramma della classe SecurityFragment

2.7.4 Package Notification

In Allegato il diagramma delle classi del package Notification composto dalle due classi:

- FirebaseIDService
- CustomFirebaseMessageService

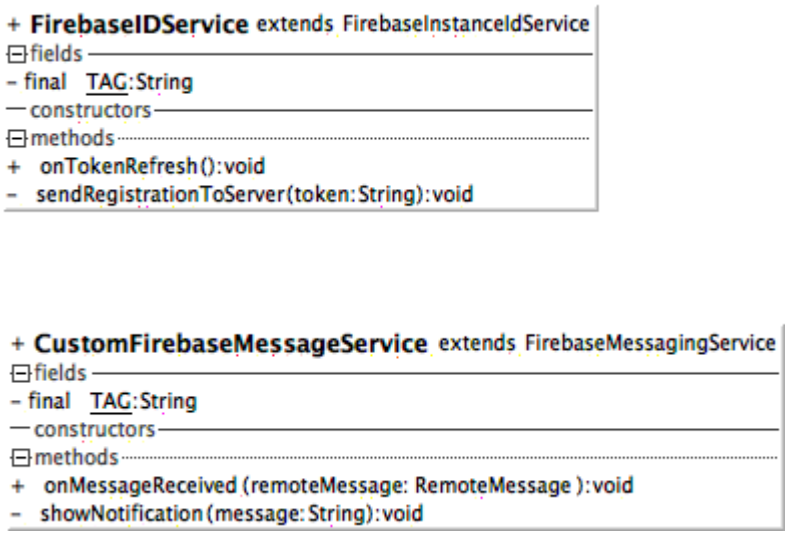

Figura 30: Diagramma delle classi del package Notification

3 Mapping Hardware/Software

L'intero sistema Swatcher è distribuito e configurato nel seguente modo:

- Piattaforma Client: dispositivo Android con versione del sistema operativo non inferiore alla 5.0 (Android Lollipop)
- Protocollo di Rete: HTTP
- Piattaforma Server: Web Server Apache, Webcam Server Motion che si interfaccia con una webcam
- Gestione della persistenza sul server: File System
- Gestione della persistenza sul Client: Shared-Preferences
- Notification System: servizio di push-notification fornito da Google (FirebaseCloudMessaging)
- Tecnologia per lo scambio di informazioni tra Client e Server: JSON
- Gestione dei Media: Libreria Picasso