**Software Design Specifications**

**Sistema di Video-Sorveglianza**

**SWATCHER**

**Università degli Studi del Sannio**

**Laurea Magistrale in Ingegneria Informatica**

**Corso di Ingegneria del Software**

**2015 I°A-II°S Prof. Massimiliano Di Penta**

**Studenti:**

**Antonio De Simone**

**Enrico Emanuele**

**Wilmer Ciasullo**

Sommario

[1 Introduzione 3](#_Toc468206925)

[1.1 Obiettivo 3](#_Toc468206926)

[1.2 Scopo del Sistema 3](#_Toc468206927)

[1.3 Definizioni Acronimi Abbreviazioni 3](#_Toc468206928)

[2 Architettura del Sistema 5](#_Toc468206929)

[2.1 Struttura e collegamenti delle componenti del sistema 5](#_Toc468206930)

[2.2 Sottosistema Client 6](#_Toc468206931)

[2.3 Sottosistema Server 7](#_Toc468206932)

[2.3.1MWS – Motion Webcam Server 8](#_Toc468206933)

[2.3.2 AWS – Apache Web Server 9](#_Toc468206934)

[2.3.2 File System 9](#_Toc468206935)

[2.4 Sottosistema di Notifica 10](#_Toc468206936)

[2.5 Sottosistema di Comunicazione 10](#_Toc468206937)

[2.6 Diagrammi di Interazione 13](#_Toc468206938)

[2.6.1 Configurazione Connessione 13](#_Toc468206939)

[2.6.2 Visualizzazione Streaming Video 15](#_Toc468206940)

[2.6.3 Snapshot 17](#_Toc468206941)

[2.6.4 Consultazione Gallery 19](#_Toc468206942)

[2.6.5 Visualizzazione del singolo media 21](#_Toc468206943)

[2.6.6 Configurazione Parametri Motion 22](#_Toc468206944)

[2.7 Diagrammi delle Classi 23](#_Toc468206945)

[3 Mapping Hardware/Software 23](#_Toc468206946)

# 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.

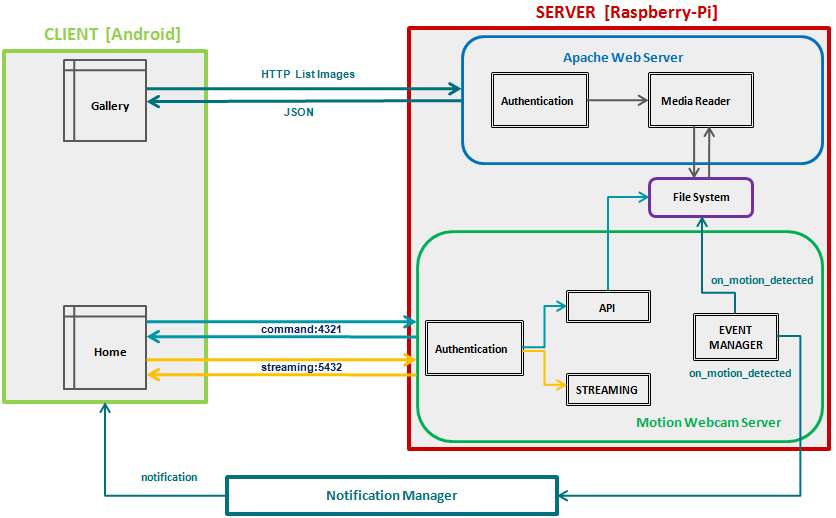


Figura : Una prima vista Architetturale del Sistema Swatcher

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 smartphone 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.

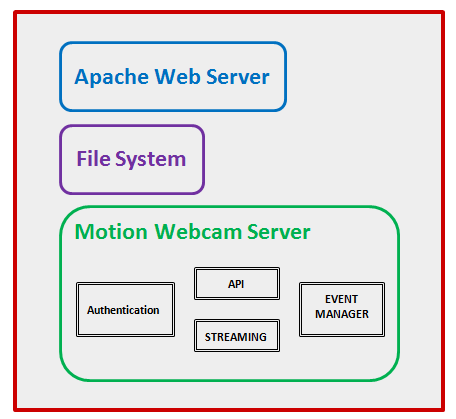


Figura : Sottosistema Server residente sulla Piattaforma Raspberry-Pi

### 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

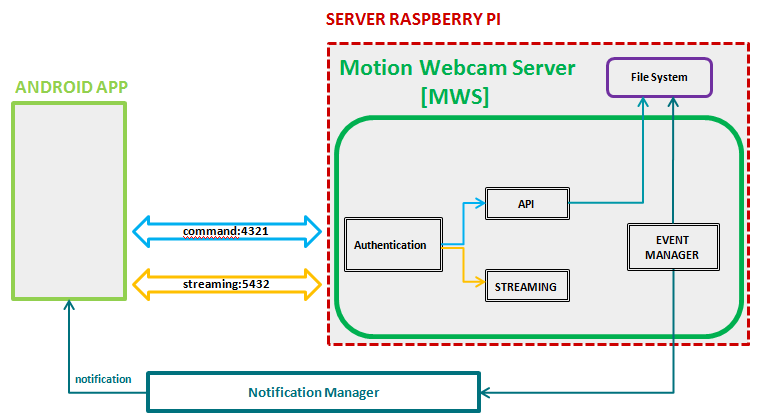


Figura : Composizionedell'MWS e loro interazioni interne e esterne al modulo

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.

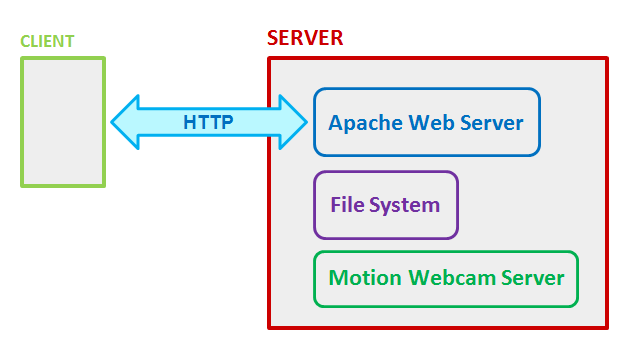


Figura : Interfaccia Client/Server e sottosistema server

### 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.

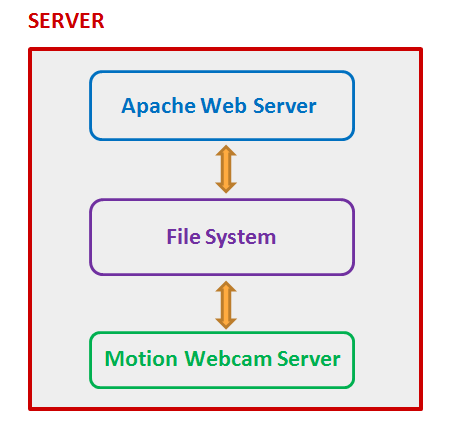


Figura : File System e interazione MWS/AWS

## 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.

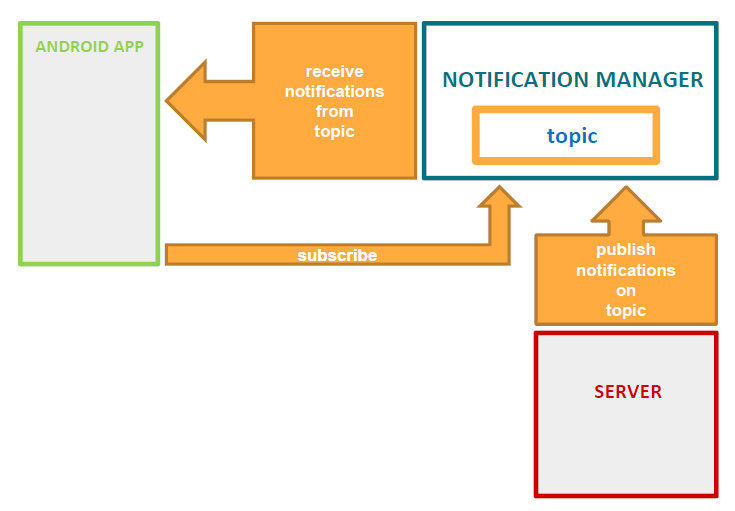


Figura : sottosistema di Notifica in Swatcher

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”.

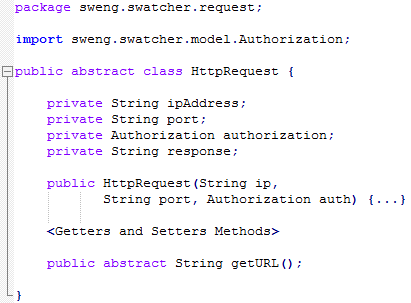


Figura : Classe Astratta HttpRequest

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”.

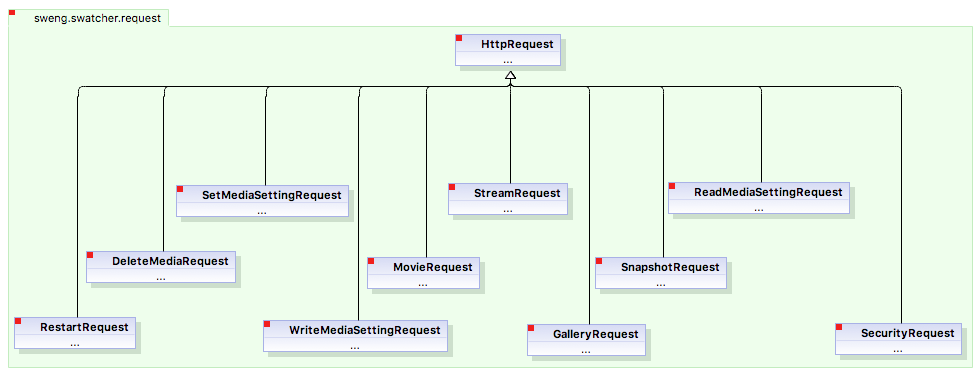


Figura : Famiglia delle classi HttpRequest

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”.

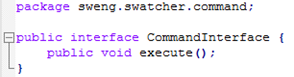


Figura : CommandInterface

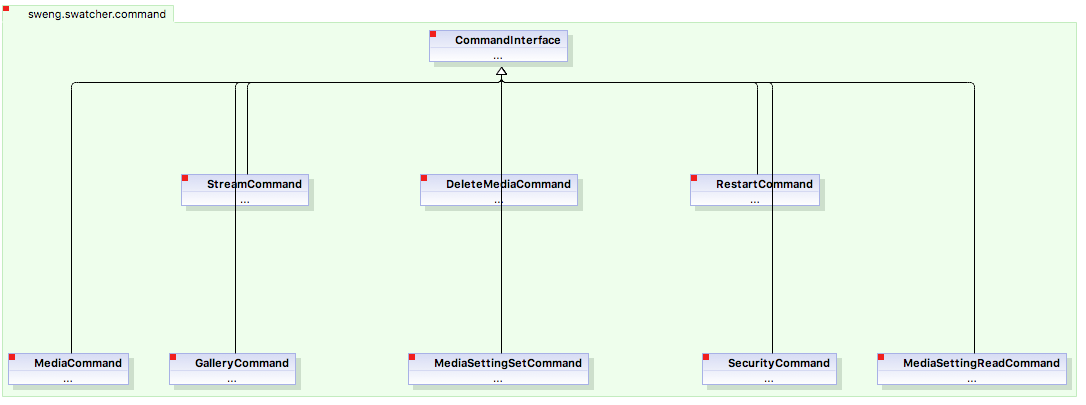


Figura : Interfaccia Command e classi che la implementano

L’interfaccia “CommandInterface” e le relative classi che la implementano sotto contenute nel package “command”.

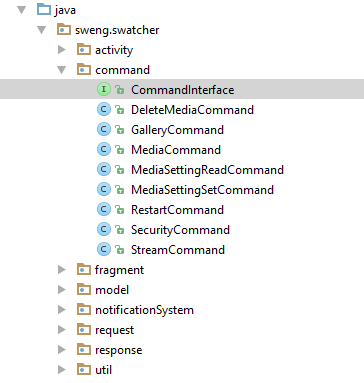


Figura : package command

## 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)

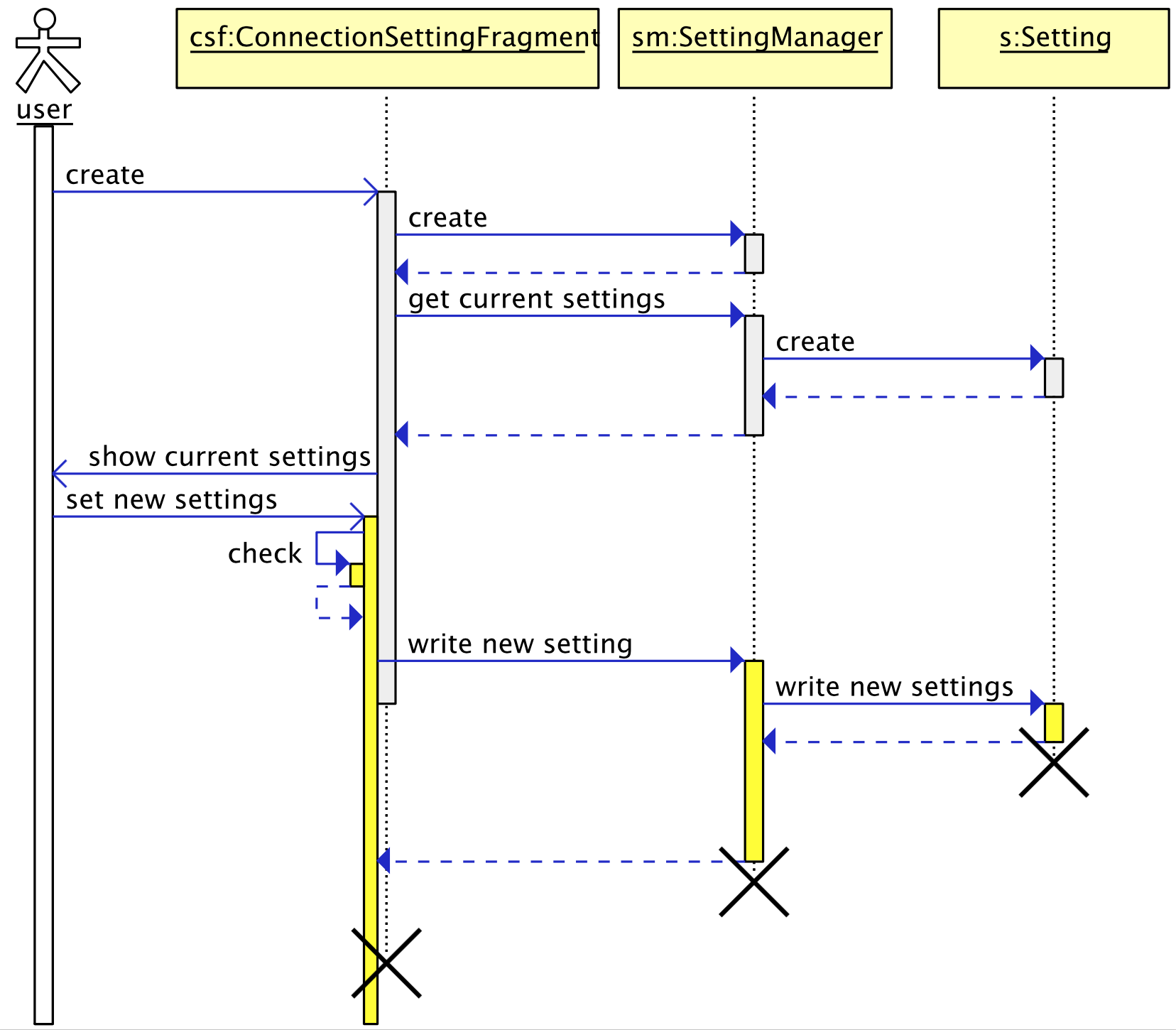


Figura : Sequence Diagram del requisito funzionale FR-02

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

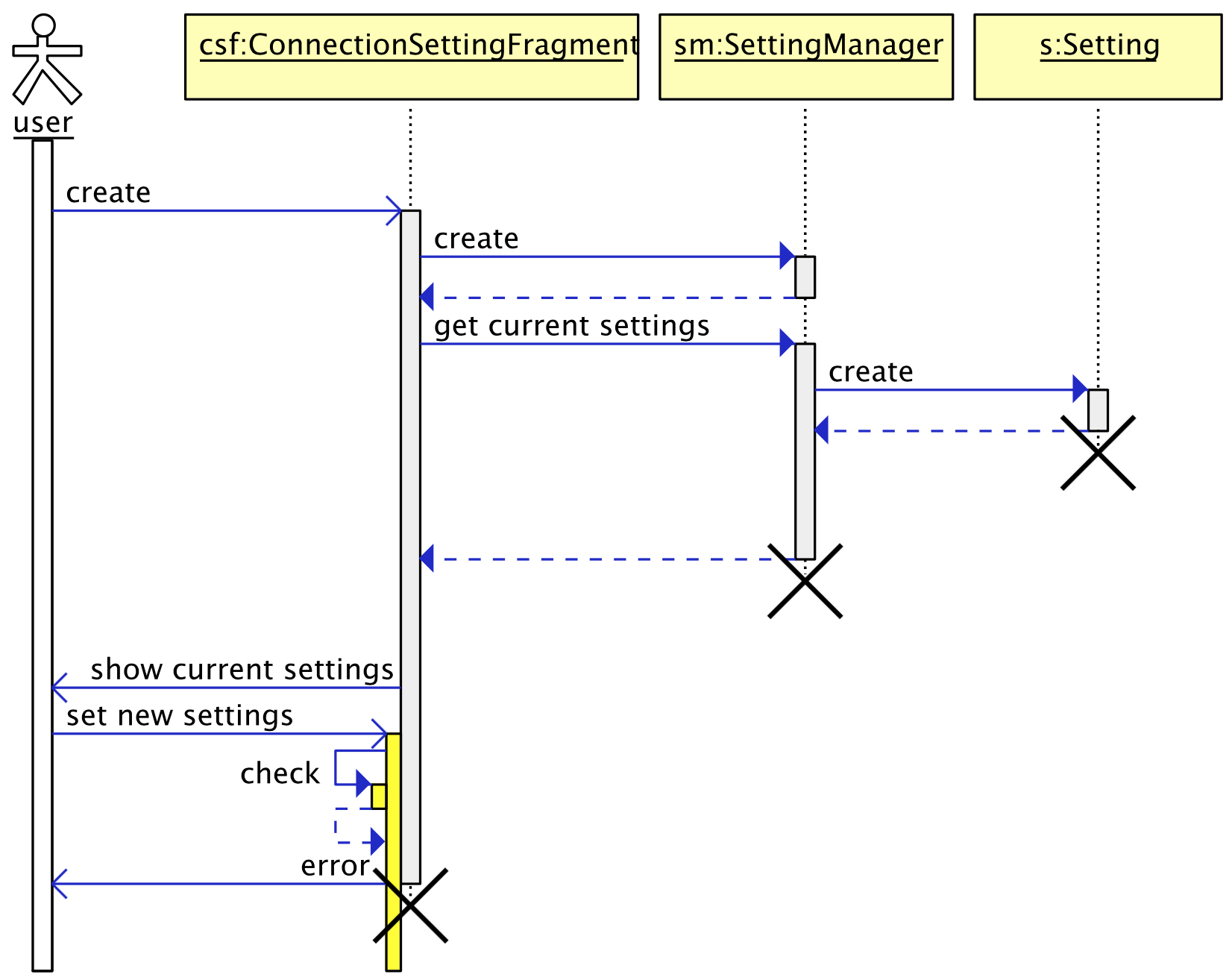


Figura : Sequence Diagram del requisito funzionale FR-02-A

### 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)

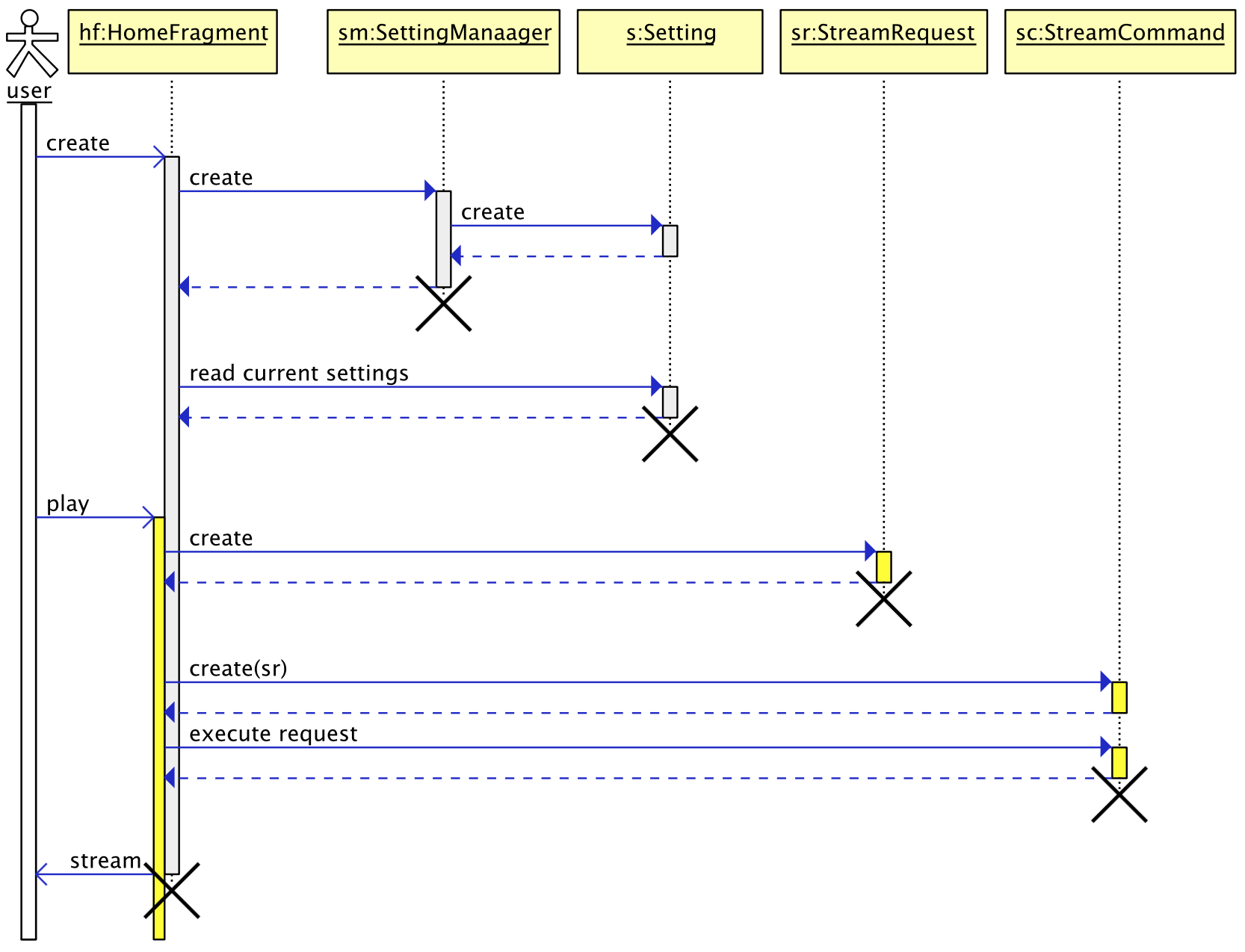


Figura : Sequence Diagram del requisito funzionale FR-03

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

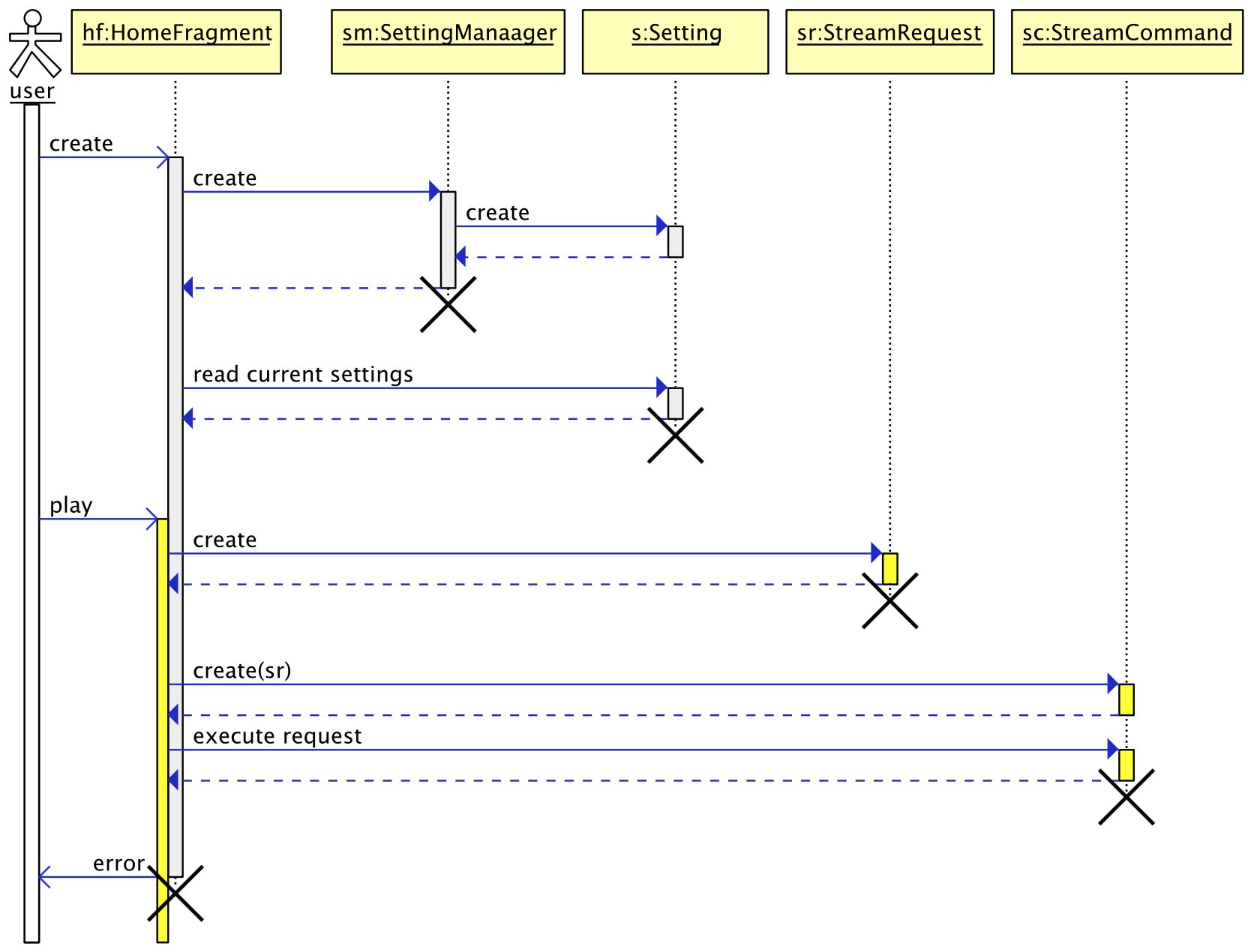


Figura : Sequence Diagram del requisito funzionale FR-03-A

### 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)

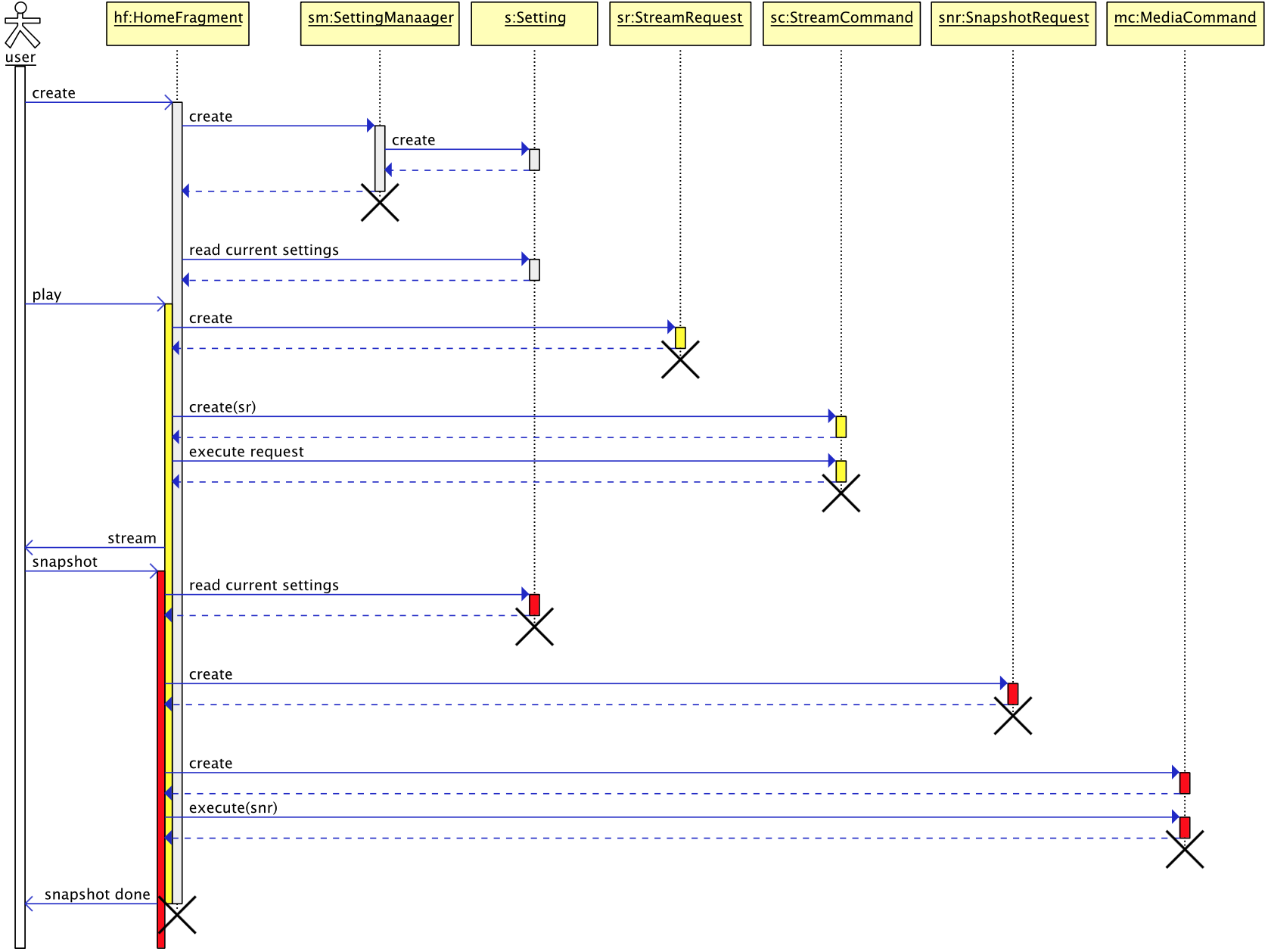


Figura : Sequence Diagram del requisito funzionale FR-04

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

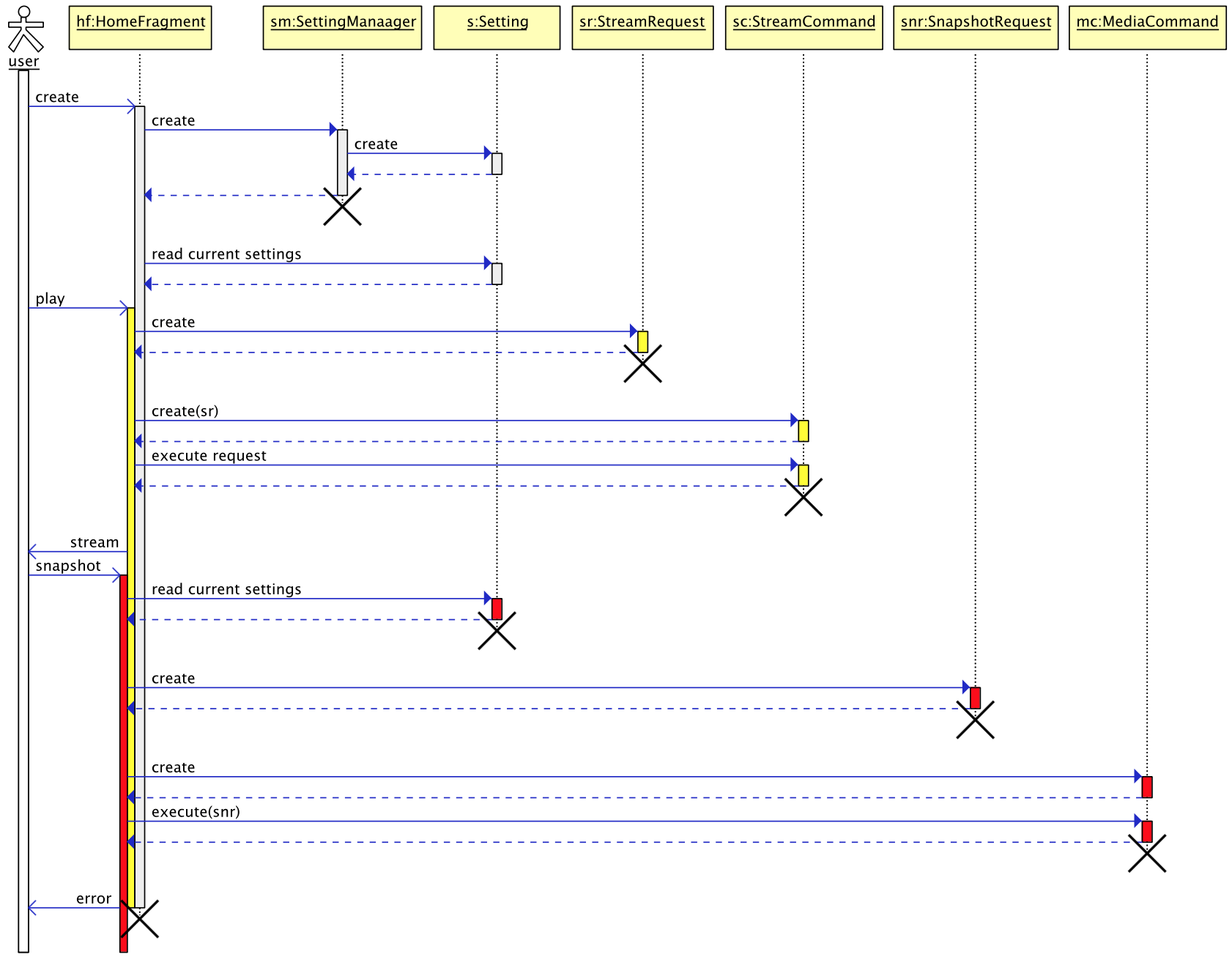


Figura : Sequence Diagram del requisito funzionale FR-04-A

### 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)

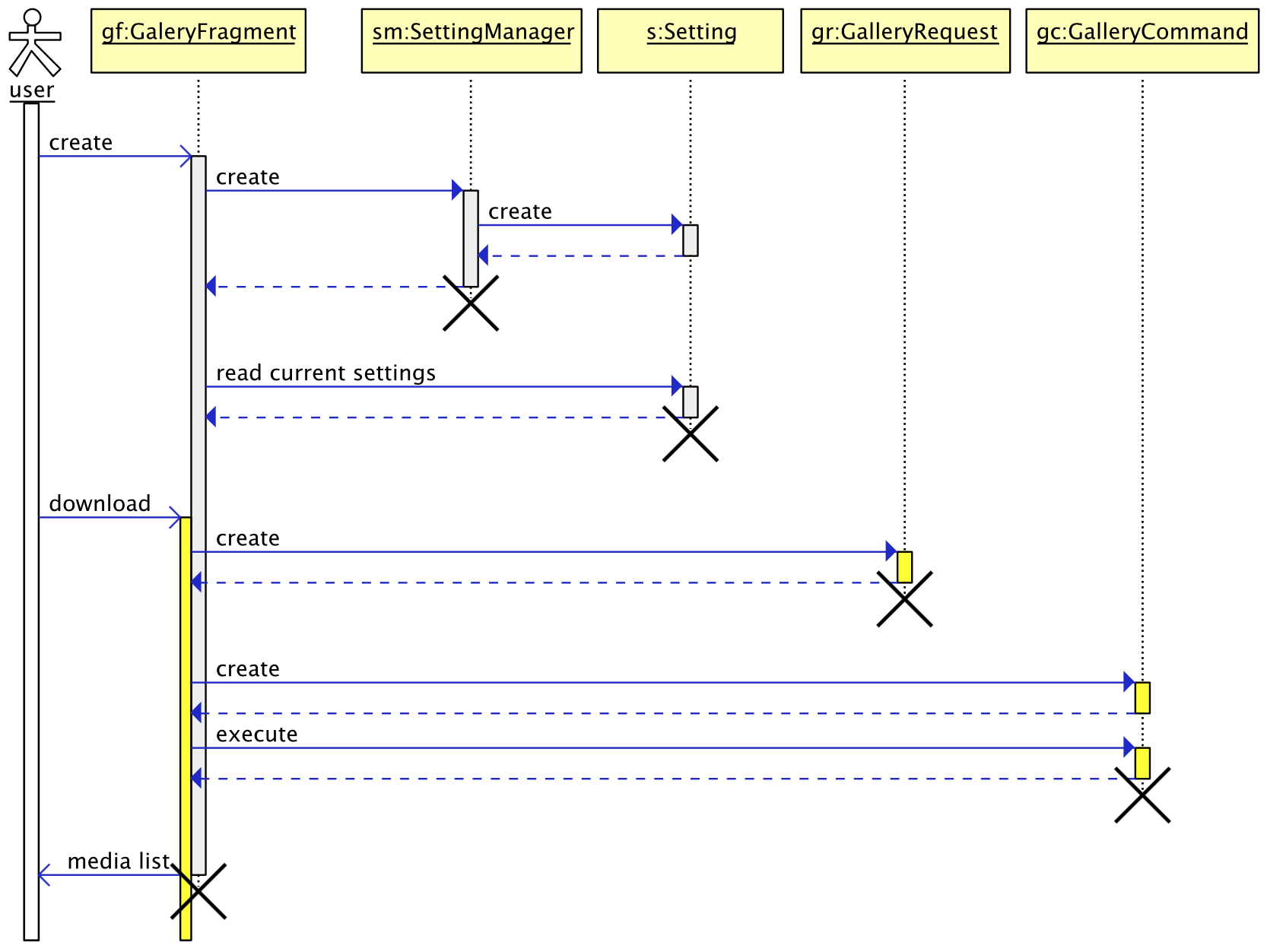


Figura : Sequence Diagram del requisito funzionale FR-05

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

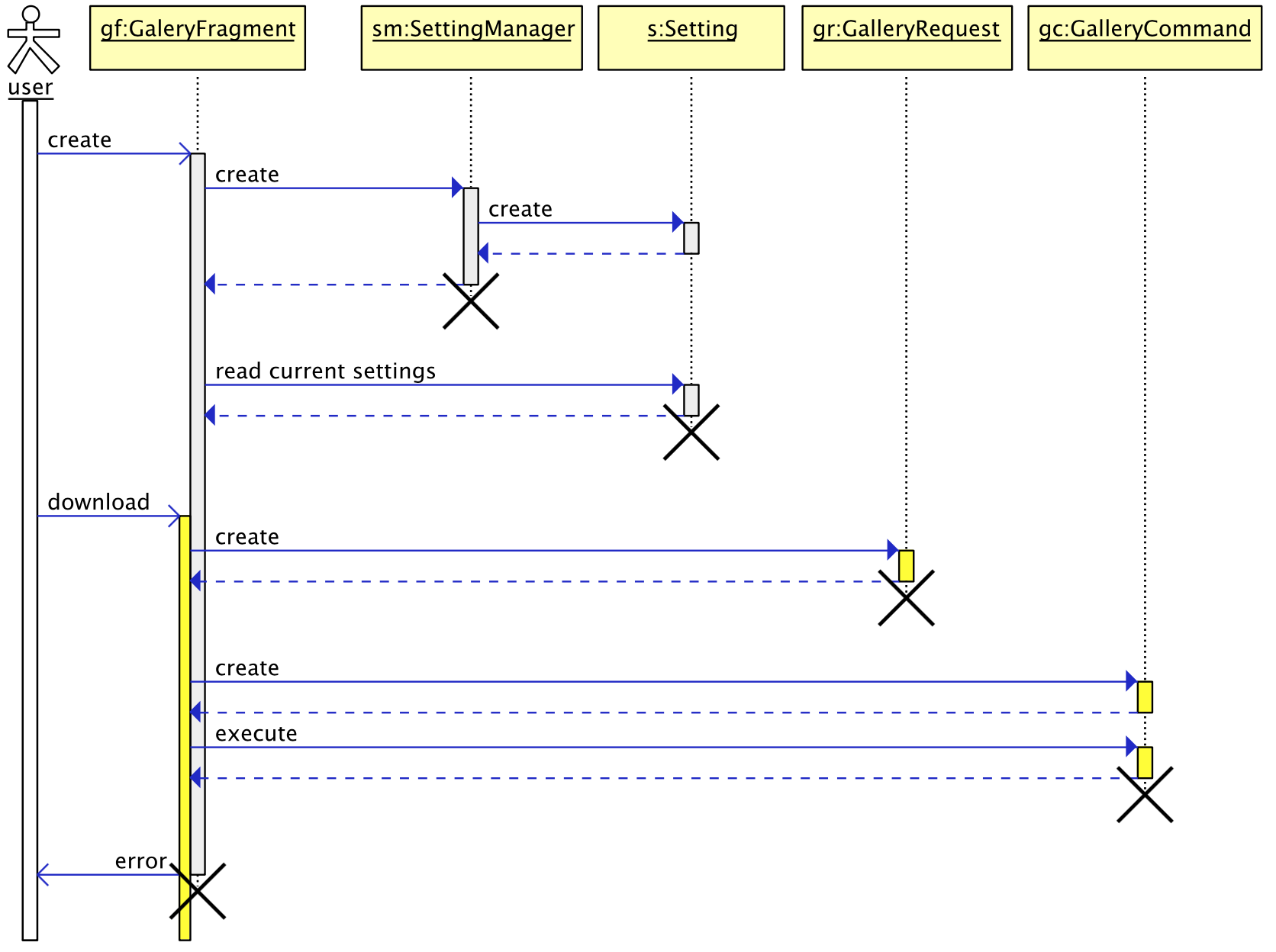


Figura : Sequence Diagram del requisito funzionale FR-05-A

### 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)

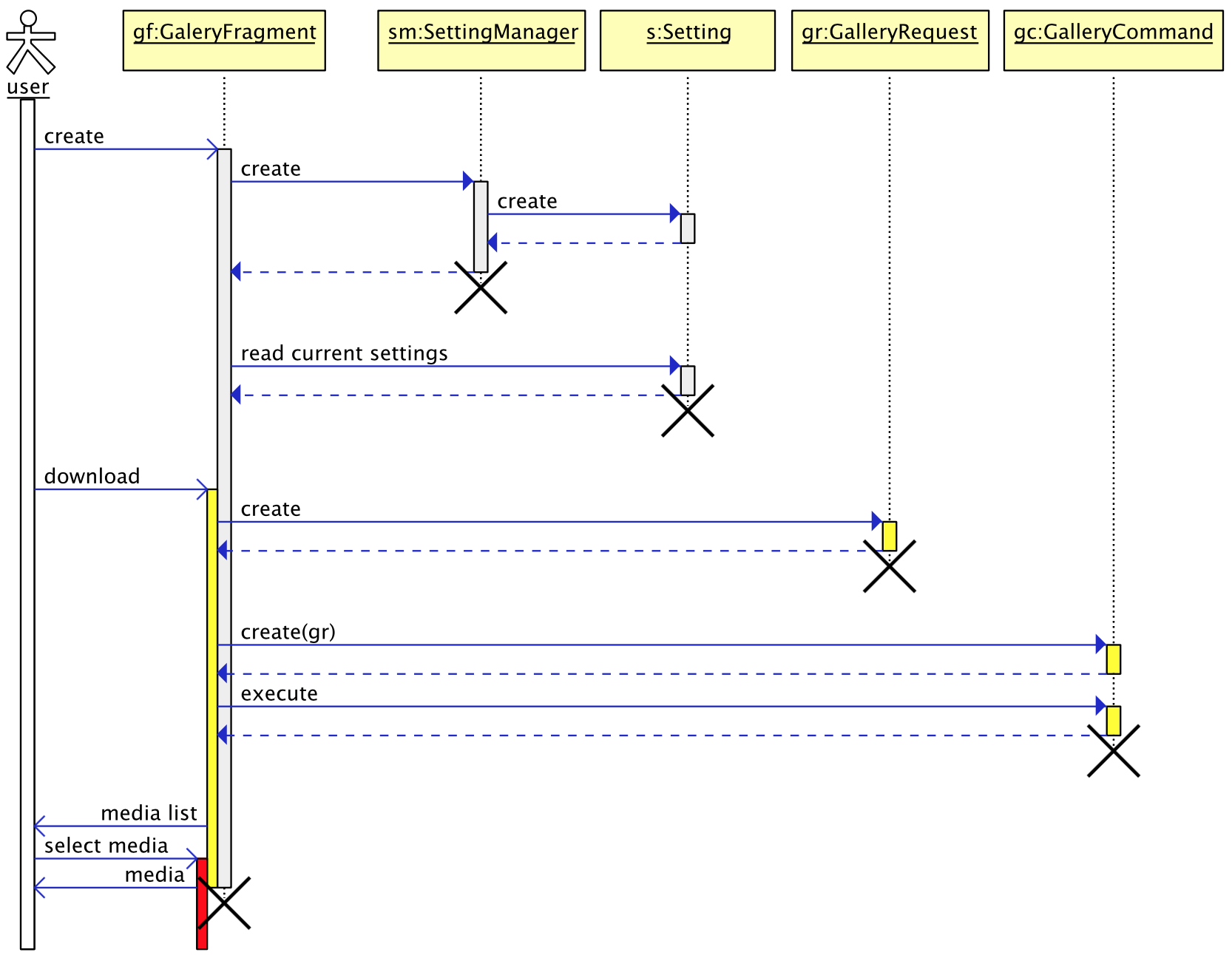


Figura : Sequence Diagram del requisito funzionale FR-06

### 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)

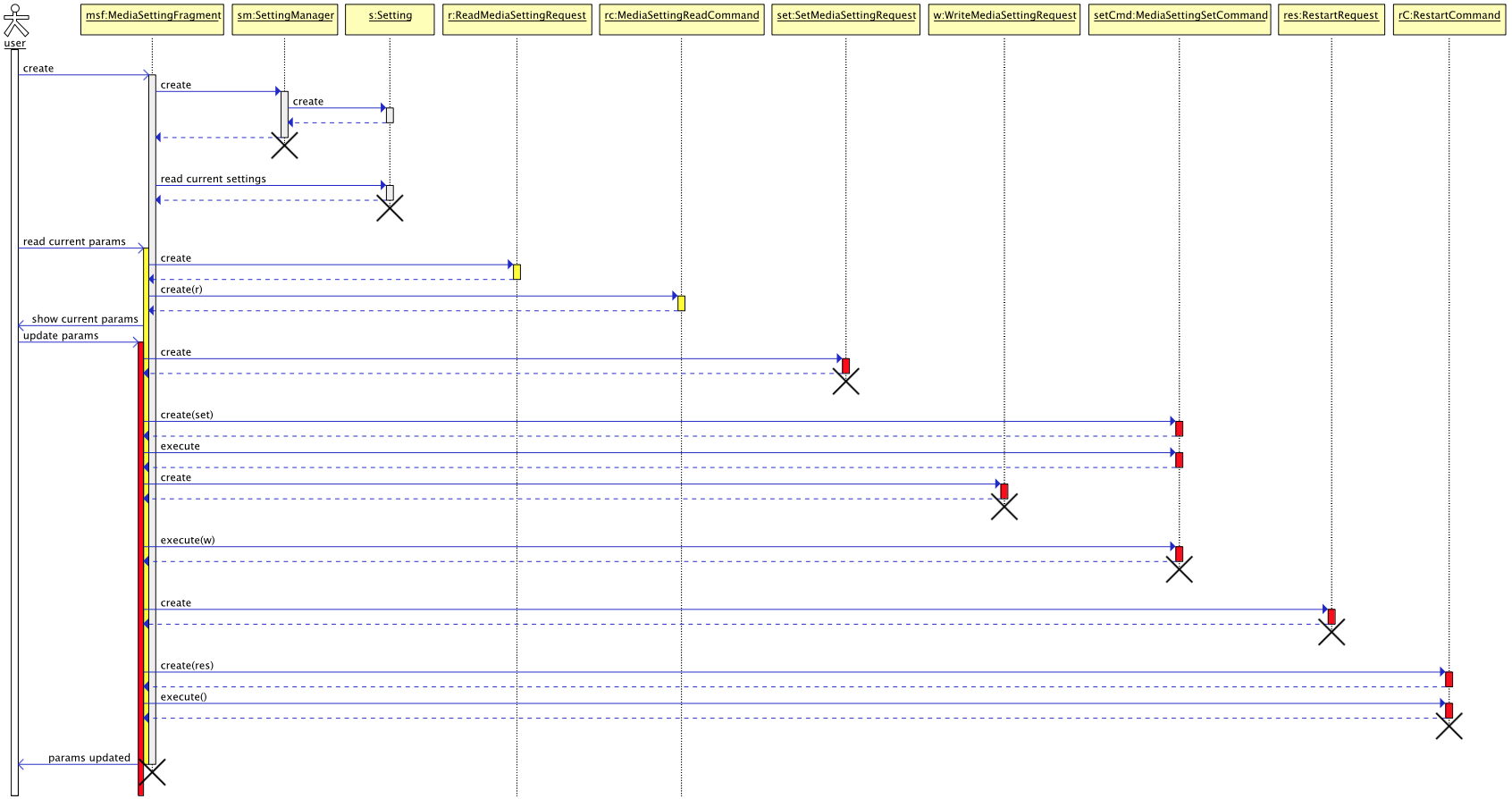


Figura : Sequence Diagram del requisito funzionale FR-08

## 2.7 Diagrammi delle Classi

In seguito è riportato il diagramma delle classi del sistema.

# 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