TuTubi

1 Specifica delle Classi

1.1 Classe Video

```
Operazioni
```

```
durataSec() : Intero >= 0
    pre: nessuna
    post:
        result = durataSec(this.flusso)
n_visualizzazioni() : Intero >= 0
    pre: nessuna
    post:
        result è il numero di oggetti v:Visualizzazione tali che:
            (this, v): vid_vis
valutazioneMedia() : float >= 0
    pre: nessuna
    post:
        sia valutazioniVideo un insieme che contiene tutti gli oggetti
        di tipo Visualizzazione per cui esiste un link (this, visualizzazione): vid_vis
        sia vTot la somma di tutti i visualizzazione.valutazione in valutazioniVideo
        return vTot / |valutazioneVideo|
num_risposte() : int >= 0
    pre: nessuna
    post:
        sia insiemeRisposte un insieme contentente tutti gli oggetti
        di tipo Video per cui esiste un link (v, Video): risposta_a_video
        ritorna | insiemeRisposte |
```

1.2 Attenzione

Tutte le altre classi, poichè prive di operazioni sono state omesse per semplictità.

2 Specifica Tipi di Dato

FileVideo:

```
- sequenza di byte che codifica un flusso video
- operazioni del tipo di dato
  durataSec(f:FileVideo): Intero >= 0
  pre: nessuna
  post: result è la durata di 'f'
```

3 Specifica dei vincoli esterni

```
[V.Video.autore_non_risponde_a_se_stesso]
```

```
Per ogni r:Video, per cui esiste v:Video tale che (r,v): risponde_a_video sia:
- u:Utente tale che:
(u, r): pubblica
Deve che (u,v) non è un link di pubblica.
Formalmente:
```

```
\forall r, v, u \quad Video(r) \land Video(v) \land Utente(u) \land \\ pubblica(u, r) \land \\ risponde\_a\_video(r, v) \quad \rightarrow \\ \neg pubblica(u, v)
```

[V.Visualizzazione.valutare_proprio_video]

Per ogni utente u, sia VIS l'insieme degli oggetti di tipo Visualizzazione tale che:

- $\exists (u, vis) : ut_vis \text{ con } vis \in VIS$
- $\exists (vis, v) : vid_vis \text{ con } vis \in VIS$
- $\exists (u, v) : utente_pubblica_video$

Deve essere che $\forall vis \in Vis$, vis.valutazione = NULLFormalmente

```
\forall u, \ v, vis \ Utente(u) \land Video(v) \land Visualizzazione(vis) \\ \land ut\_vis(u, vis) \land vid\_vis(vis, v) \\ \land utente\_pubblica\_video(u, v) \rightarrow \\ valutazione(vis, NULL)
```

[V.Playlist_no_visualizza_privata]

Per ogni utente u, se esiste (u, p) : $puo_visualizzare$ e non esiste (u, p) : crea, allora deve essere che p.pubblica = True

[V.Video_censurati]

Solo i video non censurati possono essere visualizzati, commentati, valutati o aggiunti a playlist. Formalmente

 $\forall v, \ vis, \ e \ \ Video(v) \land Visualizzazione(vis) \land Elemento(e)elem_video(v,e) \lor vid_vis(vis,v) \rightarrow censura(v,False)$

4 Specifica Use Case

4.1 Registrazione

registrazione(n: Stringa): Utente

- Attori: this, Utente Non Regirato
- Pre Condizioni: Nessuna.
- Post Condizioni: Viene Creata una nuovo oggetto di Classe Utente con: Utente.nome =n (UNIVOCO) e Utente.iscrizione = now.

4.2 Pubblicazione Video

pubblicazioneVideo(t:Stringa, d: Stringa, f: FileVideo, c: Stringa, tag: Stringa):

- Attori: Utente Registrato
- Pre Condizioni: Nessuna
- Post Condizioni: Viene creato un nuovo oggetto v di Classe Video con:
 - v.titolo= t
 - v.descrizione= d
 - v.flusso= f
 - v.istante=now

viene creato un link (this, video) : utente_pubblica_video viene creato un link (video, categoria) : cat_video con la categoria c del video vengono creati tanti link (video, tag) : tag_video quanti sono i tag del video

4.3 Cronologia

cronologia(): insieme videoVisualizzati

- Attori: Utente Registrato
- Pre Condizioni: Nessuna
- Post Condizioni: Viene restituito un insieme videoVisualizzati contentente tutti i video per cui esiste un link (this, visualizzazione): ut_vis e (visualizzazione, video): vid_vis Tutti i video v per cui v.Censurato=True sono esclusi dalla cronologia.

4.4 Ricerca

```
ricerca(c: Stringa, t:Stringa[0..*], v:int 0..5): video[0..*]
```

- Attori: Utente Registrato
- Pre Condizioni: Nessuna
- Post Condizioni: Viene restituito un insieme video[0..*] contentente tutti i video per cui esiste un link (video, categoria): cat_video con categoria c e per cui esiste un link (video, tag): tag_video con tag t[i] per ogni i=0..* e per video.valutazioneMedia() >= v. Tutti i video v per cui v.Censurato=True sono esclusi dalla ricerca.

4.5 Ricerca Discussione

ricercaDscussione(c: Stringa, t:Stringa[0..*], v:int 0..5): video[0..*]

- Attori: Utente Registrato
- Pre Condizioni: Nessuna
- Post Condizioni: Viene restituito un insieme video[0..*] contentente tutti i video per cui esiste un link (video, categoria) : cat_video con categoria c e per cui esiste un link (video, tag) : tag_video con tag t[i] per ogni i=0..* e per video.valutazioneMedia() >= v ordinati per video.num_risposte() decrescente.

Tutti i video v per cui v.Censurato=True sono **esclusi** dalla ricerca.

4.6 Gestione Playlist

creaPlaylist(n: Stringa, p: Bool): Playlist

- Attori: Utente Registrato
- Pre Condizioni: Nessuna
- Post Condizioni: Viene creato un nuovo oggetto di Classe Playlist con: Playlist.nome=n, Playlist.pubblica=p e Playlist.Data Creazione= now.

aggiungiVideoPlaylist(p: Playlist, v: Video)

- Attori: Utente Registrato
- Pre Condizioni: v.Censurato=False.
- Post Condizioni:

Viene creato un link (p, elem): $playlist_elem$ con p playlist e elem un oggetto di tipo ElementoPlaylist con elem.NumeroVid un intero, tale intero è uguale al numero dei link (p, elem): $playlist_elem + 1$.

Viene inoltre creato un link (elem, v): $elem_vid$ con v un oggetto di tipo Video.

rimuoviVideoPlaylist(p: Playlist, v: Video)

- Attori: Utente Registrato
- Pre Condizioni: Esiste l'oggeto p di tipo Playlist video creato da this(Utente Registrato che invoca l'UseCase)
- Post Condizioni:

Viene eliminato il link (p, elem): $playlist_elem$ con p playlist e elem un oggetto di tipo ElementoPlaylist e aggiornati tutti i successivi video della playlist sottraendo un numero da ElementoPlaylist.NumeroVid.

Viene inoltre eliminato il link (elem, v): $elem_vid$ con v un oggetto di tipo Video.

4.7 Gestione Valutazioni

valuta(v: Video, val: int, c: Stringa): Valutazione

• Attori: Utente Registrato

• Pre Condizioni: this ha visualizzato il video v e v.Censurato=False.

• Post Condizioni: Viene creato un nuovo oggetto Valutazione di Classe Valutazione con: Valutazione.valutazione=val e Valutazione.istante=now.

Viene creato un link (this, valutazione): ut_vis con this Utente Registrato.

Viene creato un link (valutazione, video): vid_val con v Video.

Viene creato un link (*valutazione*, *commento*) : *val_commento* Commento.commento=c e Commento.Istante = now.

4.8 Censura

censura(v: Video,m : Stringa)

• Attori: Redazione

• Pre Condizioni: Nessuna

• Post Condizioni: Sia v un oggetto di classe Video.

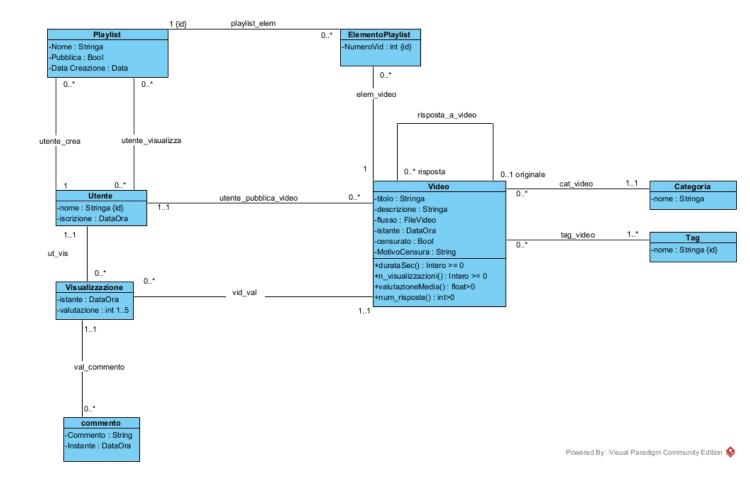
Viene impostato v.Censurato = true e v.MotivoCensura = m.

Viene eliminato l'oggetto v da tutte le Playlist in cui è presente.

Viene eliminato l'oggetto v da tutte le Cronologie degli Utenti.

5 Diagrammi

5.1 Diagramma UML



5.2 Use Case

