Ingegneria del Software A.A. 2018/2019 Esame 2019-05-17

Esercizio 1 (6 punti)

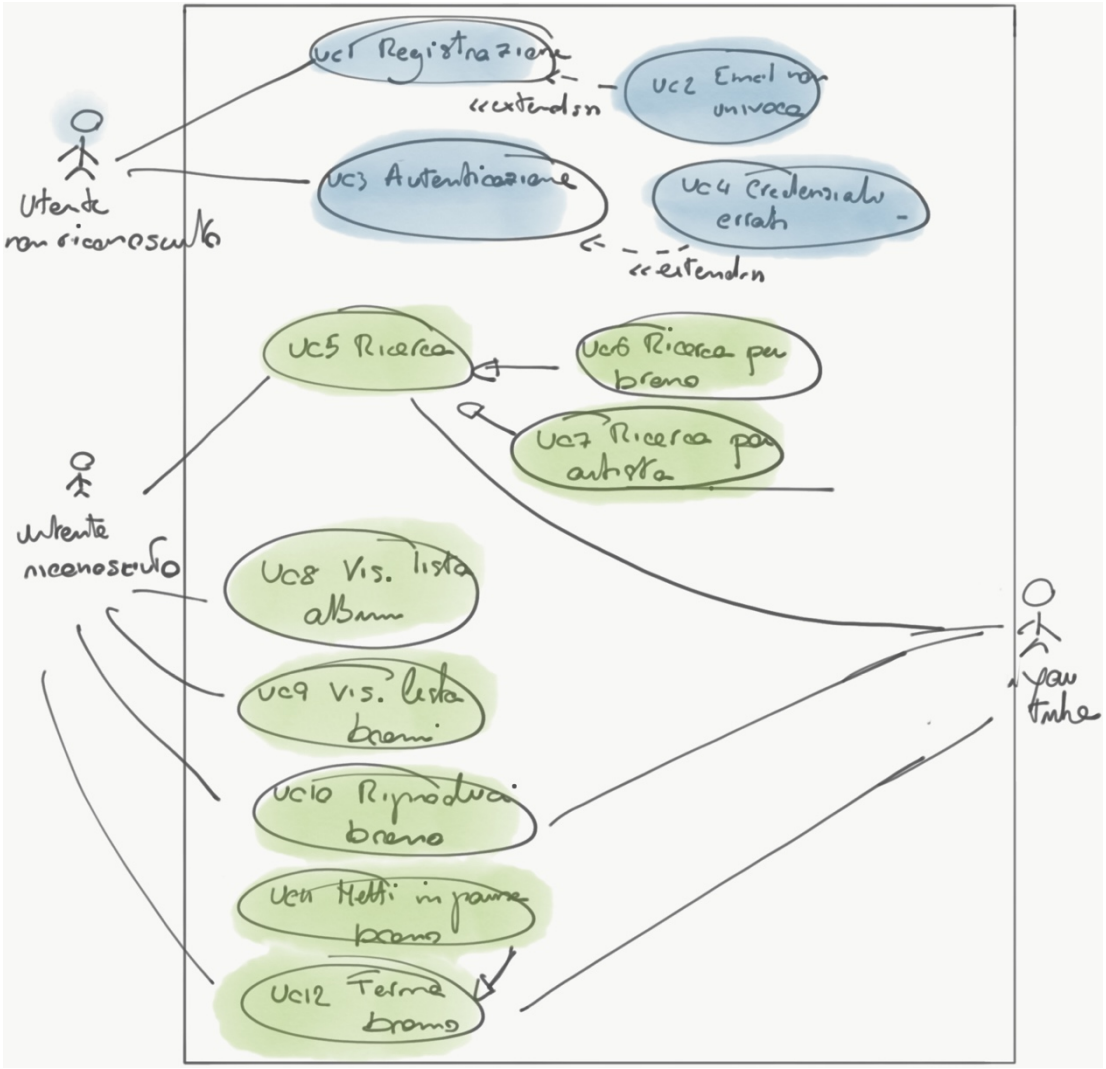
Descrizione

Netmus avrebbe dovuto essere il software per l'ascolto di brani musicali in streaming del futuro. Ideato nel 2010, venne velocemente soppiantato dall'arrivo di concorrenti quali Spotify e Google Music. Il servizio di streaming richiedeva dapprima una registrazione, tramite un'e-mail ed una password. L'e-mail doveva essere unica. La successiva fase di autenticazione avrebbe richiesto i medesimi dati. L'ascolto dei brani avveniva sfruttando la riproduzione dei video musicali su YouTube. Era possibile ricercare un brano sia per artista, sia per titolo del brano stesso. La prima ricerca ritornava una lista di album, la seconda i brani che corrispondevano al titolo inserito. Nel primo caso, di ogni album in lista erano visualizzati la copertina, il nome, e la durata complessiva. Le copertine erano recuperate all'occorrenza da un servizio esterno. Ogni brano della lista poteva essere riprodotto, messo in pausa o fermato. Inoltre, era possibile creare delle *playlist*, fornendo un nome ed aggiungendovi brani. Ogni playlist poteva essere condivisa su Facebook.

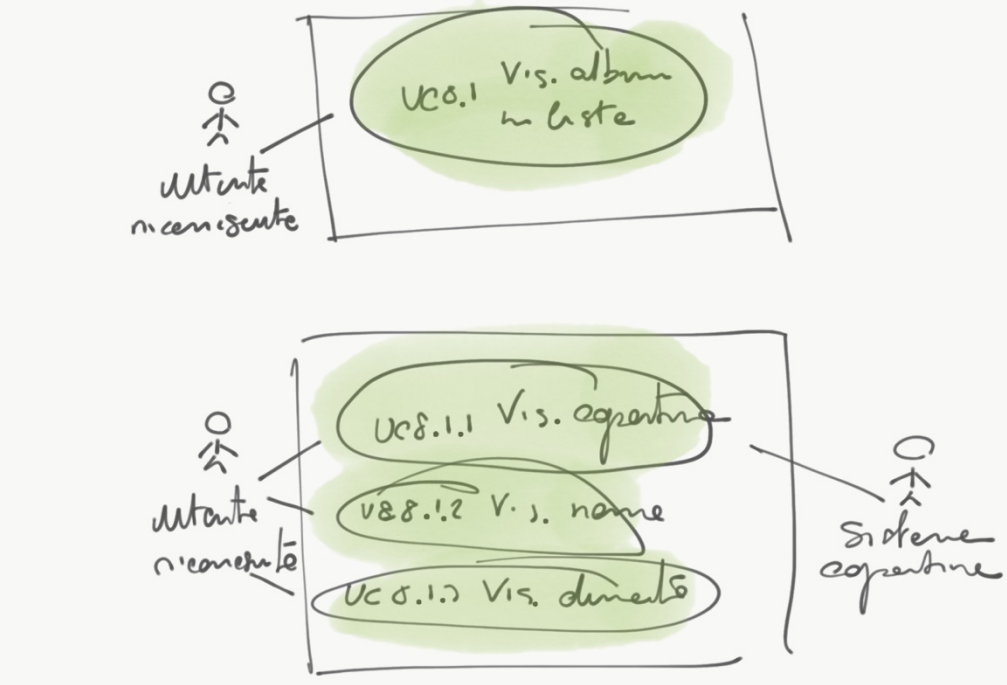
Si utilizzino i diagrammi dei casi d'uso per modellare gli scenari sopra descritti. Non ne è richiesta la descrizione testuale.

Soluzione

Di seguito la soluzione.



In dettaglio, la visualizzazione delle informazioni di un album nella lista proveniente dalla ricerca è la seguente.



Esercizio 2 (7 punti)

Descrizione

Netmus era un sistema molto avanzato per il suo tempo dal punto di vista architetturale. Per limitare l'utilizzo di risorse, anticipando i tempi, faceva uso della *reactive programming*. In particolare, la riproduzione di un brano utilizzando YouTube avveniva attraverso un *listener* sulla disponibilità di nuove informazioni provenienti dal sito di streaming video. Ogni video veniva suddiviso in *chunk*, e la disponibilità di un nuovo *chunk* immediatamente notificata al riproduttore musicale. Questo, avvalendosi di un algoritmo di decodifica audio/video, separava i due canali, restituendo unicamente quello audio. L'algoritmo era in continuo miglioramento, quindi la struttura pensata per la decodifica permetteva la sua estensione in modo agevole. Per semplicità, un brano musicale era comunque visto come un file completo localmente, contenendo l'informazione audio, il titolo e l'immagine della copertina dell'album associato.

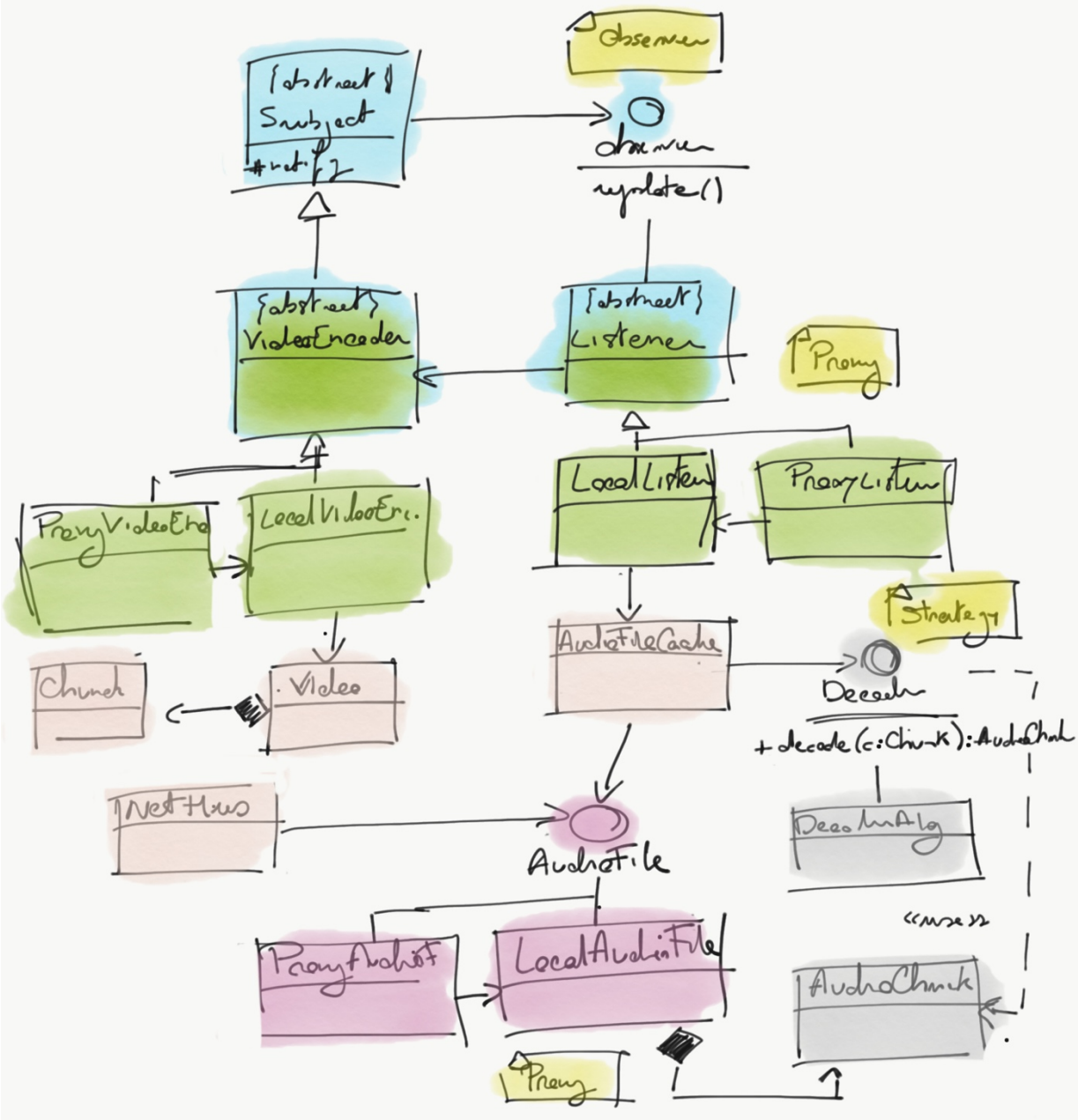
Si modelli tale sistema mediante un diagramma delle classi e i *design pattern* a esso pertinenti. Immaginando una richiesta di ascolto del brano "Bohemian Rhapsody" dei Queen, si utilizzi un diagramma di sequenza per descrivere l'interazione fra le componenti individuate.

Soluzione

I pattern individuati sono i seguenti:

- Observer
- Remote proxy
- Virtual proxy
- Strategy

Il diagramma delle classi è il seguente.



Il diagramma di sequenza è facilmente implementabile, di conseguenza.

Esercizio 3 (3 punti)

Descrizione

Il *framework* delle *Collection* in Java permette di ordinare una lista in modo arbitrario utilizzando il metodo della classe `Collections`, `static <T> void sort(List<T> list, Comparator<? super T> c)`.

L'interfaccia `Comparator` contiene un unico metodo `int compare(T o1, T o2)`.

Tale *framework* utilizza un noto *design pattern* della GoF. Si fornisca il diagramma delle classi che lo contestualizza, per una lista di oggetti della classe `Album`, utilizzando un comparatore che ordina per `title`.

Soluzione

Il pattern richiesto è lo Strategy pattern e il diagramma delle classi relativo all'esercizio è il seguente.