

Università degli Studi di Napoli Federico II

**Corso di Laurea in Ingegneria Informatica**

**Corso di Ingegneria del Software**

**Prof. S. Russo - A.A. 2020 - 21**

***Progetto***

APPLICAZIONE WEB AGENZIA TELEVISIVA

Gruppo: TeamBallo

Studenti:

Vito Romano N46004509 vito.romano2@studenti.unina.it

Stefano Licinio N46002779 s.licinio@studenti.unina.it

Paolo Russo N46002315 paolo.russo16@studenti.unina.it

Versione 1 del *14/12/2020*

**Indice**

[1. Requisiti 3](#_Toc59053823)

[1.1 Specifiche informali d’utente 3](#_Toc59053824)

[1.2 Requisiti software 4](#_Toc59053825)

[2. Analisi e specifica dei requisiti 8](#_Toc59053826)

[2.1 Modellazione dei casi d’uso 8](#_Toc59053827)

[2.2 Diagramma delle classi 10](#_Toc59053828)

[2.3 Diagrammi di sequenza 12](#_Toc59053829)

[3. Stima dei costi 15](#_Toc59053830)

[4. Piano di test funzionale 22](#_Toc59053831)

[5. Progettazione 28](#_Toc59053832)

[5.1 Diagramma delle classi 28](#_Toc59053833)

[5.2 Diagrammi di sequenza 29](#_Toc59053834)

[6. Implementazione 31](#_Toc59053835)

[7. Testing 34](#_Toc59053836)

[7.1 Test funzionale 34](#_Toc59053837)

[7.2 Test strutturale 37](#_Toc59053838)

[7.2.1 Complessità ciclomatica 37](#_Toc59053839)

[7.2.2 Test di unità 41](#_Toc59053840)

[7.3 Test di unità con JUnit 42](#_Toc59053841)

# Requisiti

## Specifiche informali d’utente

Si vuole realizzare una applicazione web per l'azienda VTV, una agenzia televisiva che registra

filmati di eventi di attualità e li diffonde verso giornalisti e agenzie di stampa sue clienti.

L'applicazione deve permettere ai cameraman di caricare video sul sistema, ai montatori di

creare dei servizi-tv, e ai clienti di ricercare i contenuti e di ricevere notifiche sulle nuove risorse. I cameraman effettuano l'upload di filmati sulla piattaforma che sono caratterizzati da un id, un nome, una data di registrazione, una durata, una dimensione.

I montatori creano servizi-tv, che sono il risultato della composizione di uno, due o più filmati: i servizi-tv possono essere standard oppure premium e sono associati ad un evento di attualità. Gli eventi di attualità, identificati da un nome, da un luogo e da una descrizione, possono essere definiti dai cameraman, in maniera indipendente dall'upload di filmati, oppure dai montatori, durante le operazioni di montaggio, se non esiste sulla piattaforma l'evento d'attualità oggetto di un servizio-tv. Esistono due categorie di utenti, standard e premium: gli utenti standard possono solo visualizzare servizi-tv standard, mentre gli utenti premium possono anche acquistare i servizi-tv premium. Di tutti gli utenti occorre memorizzare nome, cognome, data di nascita, indirizzo e casella email; per gli utenti premium bisogna aggiungere la data di scadenza dell'abbonamento. Gli utenti possono ricercare i servizi-tv in base ad una ricerca testuale sul nome dell'evento e possono registrarsi al servizio di notifiche di nuovi upload: i risultati delle ricerche degli utenti standard non devono includere i servizi-tv premium. Per finalità di business intelligence, VTV richiede inoltre che il reparto di marketing ottenga dati sulle visualizzazioni e sugli acquisti dei servizi-tv da parte di utenti standard e premium: per le visualizzazioni degli utenti occorre registrare la data; per gli acquisti dei servizi-tv la data e il numero di licenze d'uso comprate.

## Requisiti software

Con gli incontri avvenuti con il committente si è cercato di disambiguare alcuni aspetti del sistema. Sono stati chiariti i seguenti punti:

• Il cameraman, deve essersi autenticato, mediante username e password, all’interno della piattaforma per poter accedere alle rispettive funzionalità che riguardano il caricamento di un nuovo filmato, sul sistema, composto da un id, un nome, una data di registrazione, una durata, ed una dimensione. Inoltre, può definire un evento di attualità, a partire dai filmati già presenti, come servizio tv.

• Il montatore deve autenticarsi, mediante username e password, all’interno della piattaforma per poter accedere alle rispettive funzionalità come la creazione di un nuovo servizio tv, standard o premium, risultato della composizione di uno, due o più filmati. Il servizio tv sarà caratterizzato da un titolo ed una lista di uno o più filmati, Infine può definire un evento di attualità a partire dai servizi già presenti.

• Il cliente, per poter accedere deve autenticarsi sulla piattaforma mediante le proprie credenziali. Esistono due categorie di cliente, standard e premium:

* il cliente standard può solo visualizzare servizi-tv standard. Può ricercare i servizi-tv in base ad una ricerca testuale sul nome dell'evento ed abilitare il servizio di notifiche di nuovi upload di filmati e servizi tv.
* Il cliente premium può anche acquistare i servizi-tv premium.

• Per finalità di business intelligence, l’addetto marketing può generare un report sulle visualizzazioni da parte dei clienti standard e gli acquisti dei servizi tv da parte degli utenti premium.

I requisiti funzionali definiti all’interno del sistema sono:

1. CaricaFilmato: tramite questa funzionalità, il cameraman effettua l’upload di un nuovo

filmato

1. DefinisciEventoDiAttualità: tramite questa funzionalità, il cameraman o il montatore possono definire un evento di attualità a partire dai servizi TV già presenti
2. CreaServizioTv: tramite questa funzionalità, il montatore crea un servizio TV attraverso la composizione di uno o più filmati
3. VisualizzaDati: tramite questa funzionalità, l’addetto del reparto marketing può visualizzare i dati inerenti le visualizzazioni da parte dei clienti standard e, nel caso dei clienti abbonati, anche i dati sui servizi TV acquistati.
4. AbilitaNotifiche: tramite questa funzionalità, i clienti decidono se ricevere una notifica quando viene caricato sulla piattaforma un nuovo filmato o creato un nuovo servizio TV
5. RicercaContenuto: tramite questa funzionalità, i clienti possono ricercare un servizio TV già presente sulla piattaforma
6. VisualizzaServizioTv: tramite questa funzionalità, i clienti dopo aver ricercato un servizio TV possono visualizzarlo
7. AcquistaServizioPremium: tramite questa funzionalità, i clienti premium possono decidere di acquistare un servizio TV premium tra quelli disponibili
8. InviaNotifiche: tramite questa funzionalità, il sistema invia un messaggio con cui avvisa i clienti abilitati che è presente un nuovo servizio TV o filmato all’interno della piattaforma

I requisiti sui dati definiti all’interno del sistema sono:

1. Filmato è identificato da un ID, un nome, una data, una dimensione ed una durata
2. Cameraman è identificato da uno username, una password
3. ServizioTv è definito da un titolo
4. EventoDiAttualità è definito da un nome, un luogo ed una descrizione
5. Montatore è identificato da uno username ed una password
6. Cliente è identificato da un nome, un cognome, una data di nascita, un indirizzo ed una email; qualora il cliente fosse premium si aggiunge la data di scadenza dell’abbonamento

*“Codifica dei requisiti funzionali”*

|  |  |  |
| --- | --- | --- |
| RF01 | CaricaFilmato | L'applicazione deve permettere ai cameraman di caricare video sul sistema |
| RF02 | DefinisciEventoAttualità | Gli eventi di attualità, identificati da un nome, da un luogo e da una descrizione, possono essere definiti dai cameraman, in maniera indipendente dall'upload di filmati, oppure dai montatori, durante le operazioni di montaggio, se non esiste sulla piattaforma l'evento d'attualità oggetto di un servizio-tv. |
| RF03 | CreaServizioTv | I montatori creano servizi-tv, che sono il risultato della composizione di uno, due o più filmati |
| RF04 | VisualizzaDati | Per finalità di business intelligence, VTV richiede inoltre che il reparto di marketing ottenga dati sulle visualizzazioni e sugli acquisti dei servizi-tv da parte di utenti standard e premium: per le visualizzazioni degli utenti occorre registrare la data; per gli acquisti dei servizi-tv la data e il numero di licenze d'uso comprate |
| RF05 | AbilitaNotifiche | Gli utenti possono ricercare i servizi-tv in base ad una ricerca testuale sul nome dell'evento e possono registrarsi al servizio di notifiche di nuovi upload |
| RF06 | RicercaContenuto | Gli utenti possono ricercare i servizi-tv in base ad una ricerca testuale sul nome dell'evento e possono registrarsi al servizio di notifiche di nuovi upload: i risultati delle ricerche degli utenti standard non devono includere i servizi-tv premium. |
| RF07 | VisualizzaServizioTv | gli utenti standard possono solo visualizzare servizi-tv standard |
| RF08 | AcquistaServizioPremium | gli utenti premium possono anche acquistare i servizi-tv premium |

*“Codifica dei requisiti sui dati”*

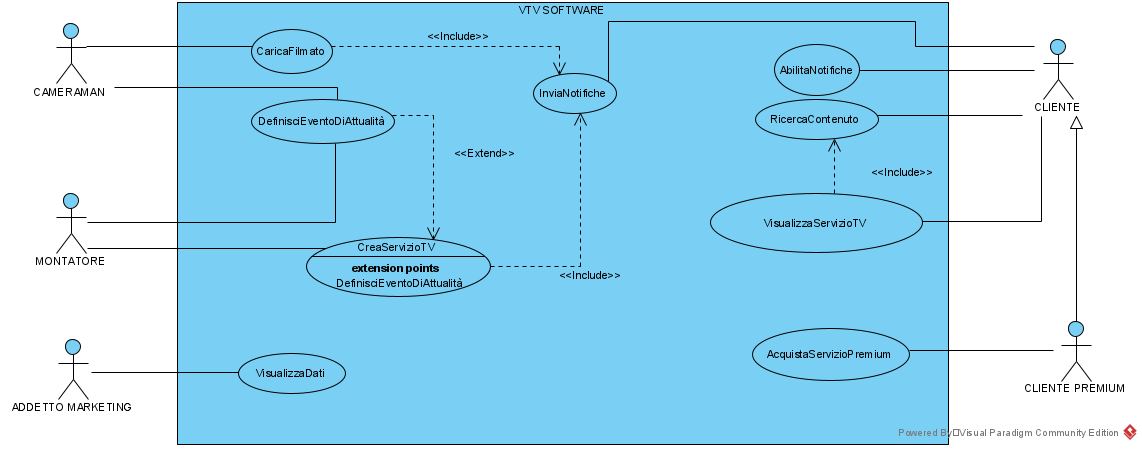
|  |  |  |
| --- | --- | --- |
| RD01 | Filmato | I cameraman effettuano l'upload di filmati sulla piattaforma che sono caratterizzati da un id, un nome, una data di registrazione, una durata, una dimensione. |
| RD02 | Cameraman | Username, password |
| RD03 | ServizioTv | Titolo |
| RD04 | EventoDiAttualità | Gli eventi di attualità, identificati da un nome, da un luogo e da una descrizione |
| RD05 | Montatore | Username, password |
| RD06 | Cliente | Di tutti gli utenti occorre memorizzare nome, cognome, data di nascita, indirizzo e casella email |
| RD07 | ClientePremium | per gli utenti premium bisogna aggiungere la data di scadenza dell'abbonamento. |

# Analisi e specifica dei requisiti

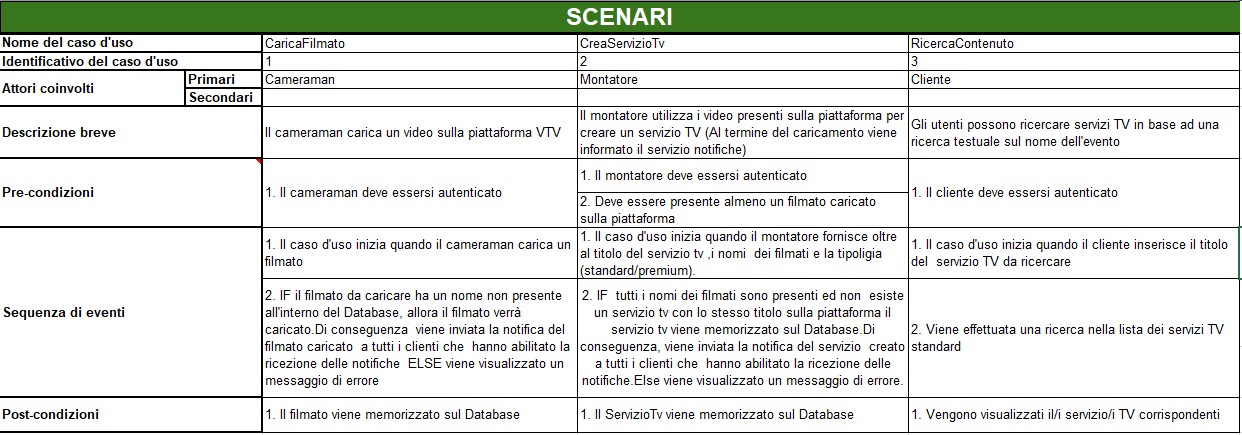
## Modellazione dei casi d’uso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attore Primario | Attore Secondario | Caso d’uso | Caso d’uso di inclusione | Caso d’uso di estensione |
| 1.Cameraman  2.Montatore  3.Addetto Marketing  4.Cliente  5.Cliente Premium | 1.Cliente | 1. CaricaFilmato  2. DefinisciEventoDiAttualità  3.CreaServizioTv  4.VisualizzaDati  5.AbilitaNotifiche  6.RicercaContenuto  7.VisualizzaServizioTv  8.AcquistaServizioPremium | 1.InviaNotifche  2.RicercaContenuto | 1.CreaServizioTv |

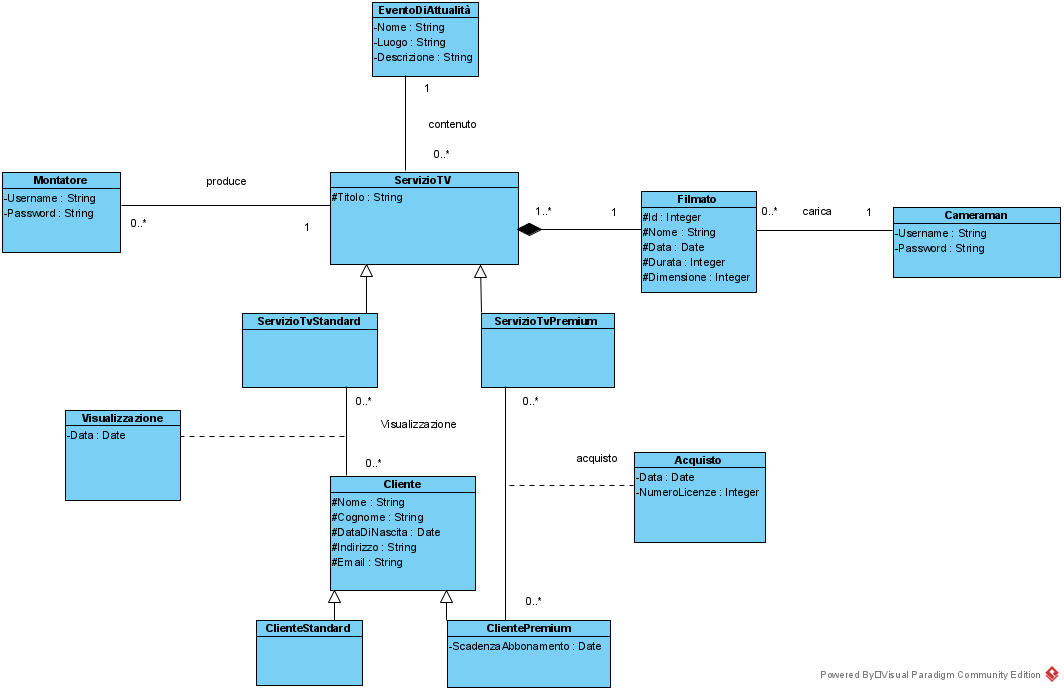
*“Diagramma dei Casi d’Uso”*

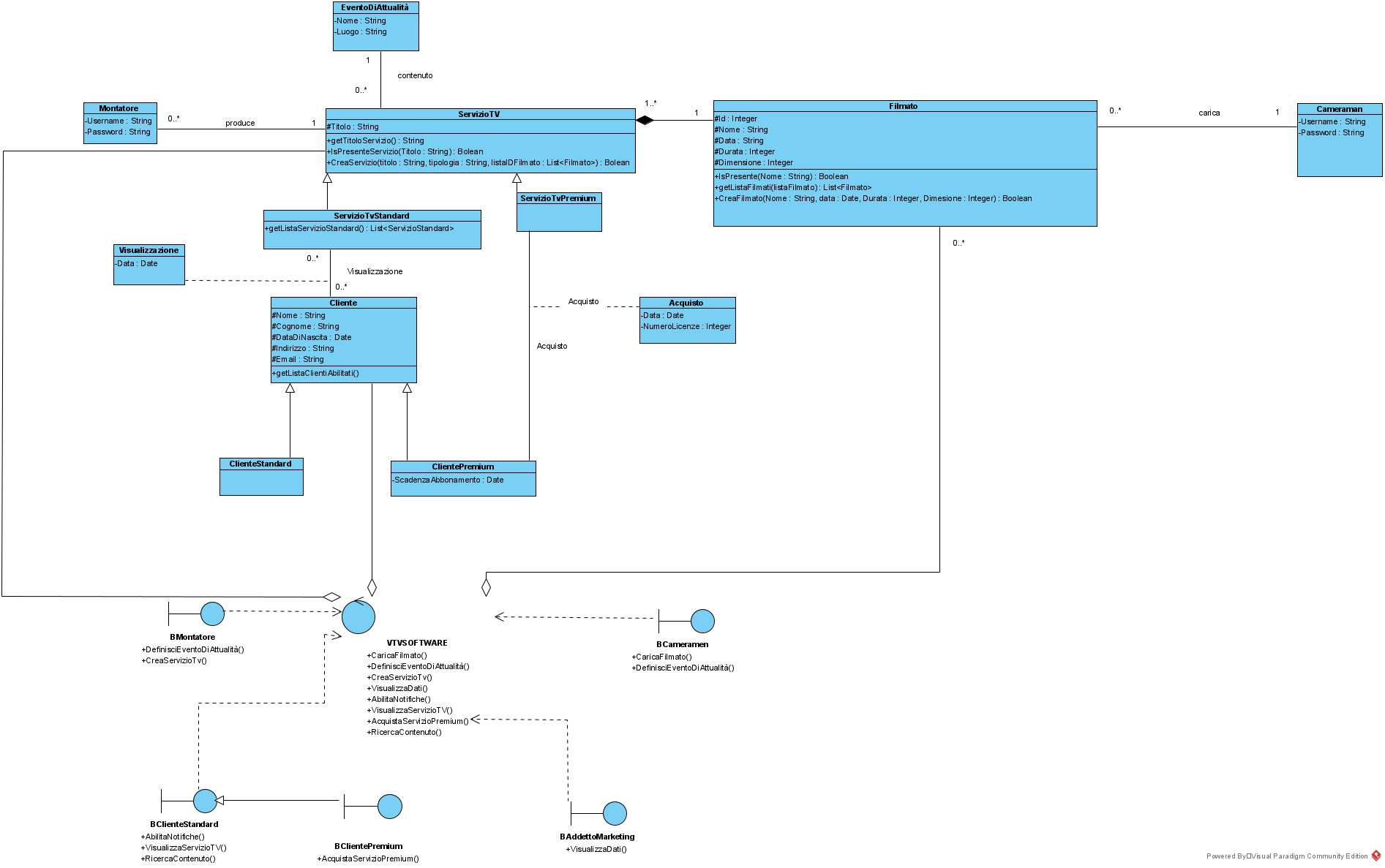


Gli scenari principali legati ai casi d’uso “CaricaFilmato”, “RicercaContenuto” e “CreaServizioTv” sono i seguenti:



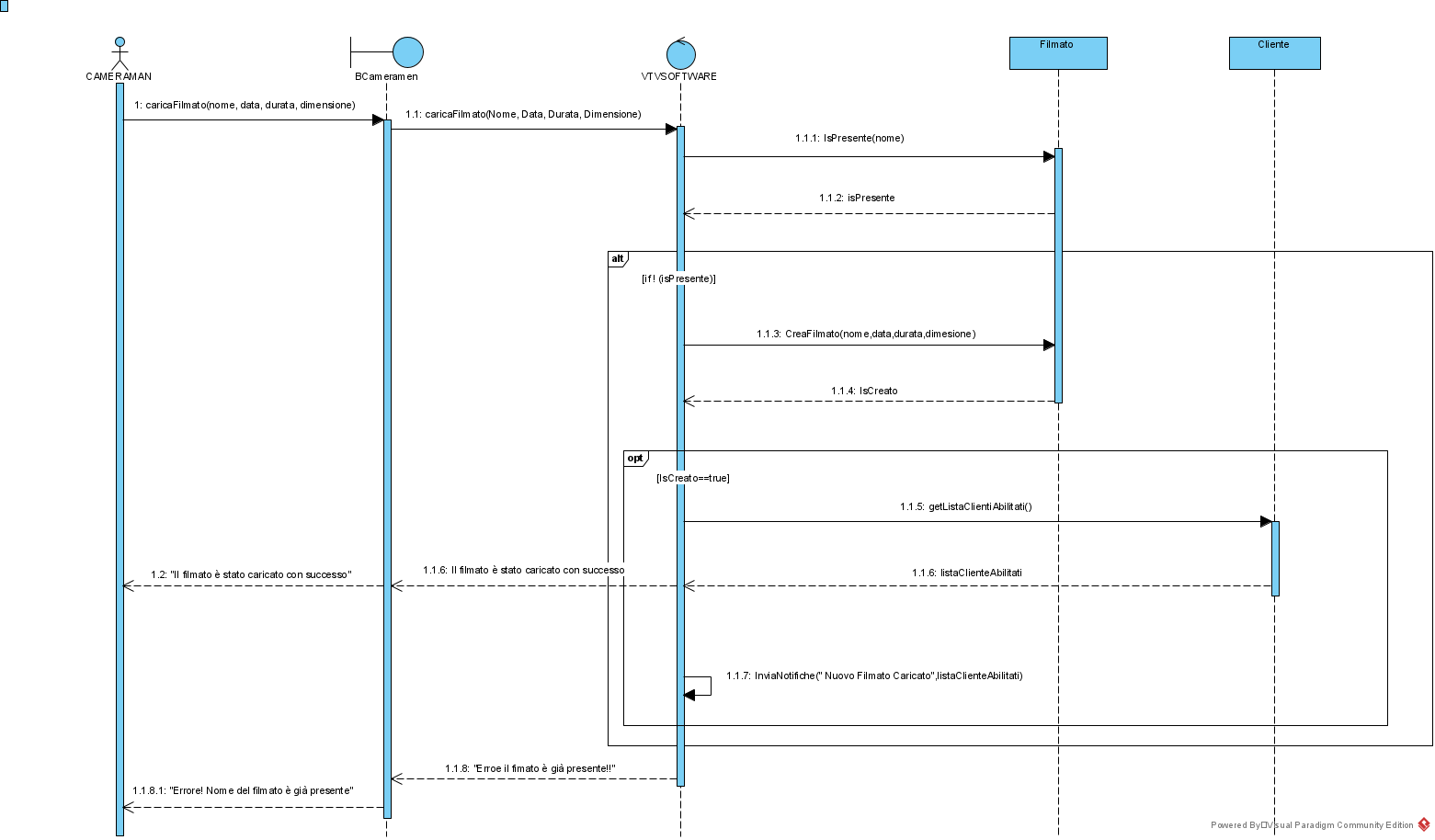
## Diagramma delle classi



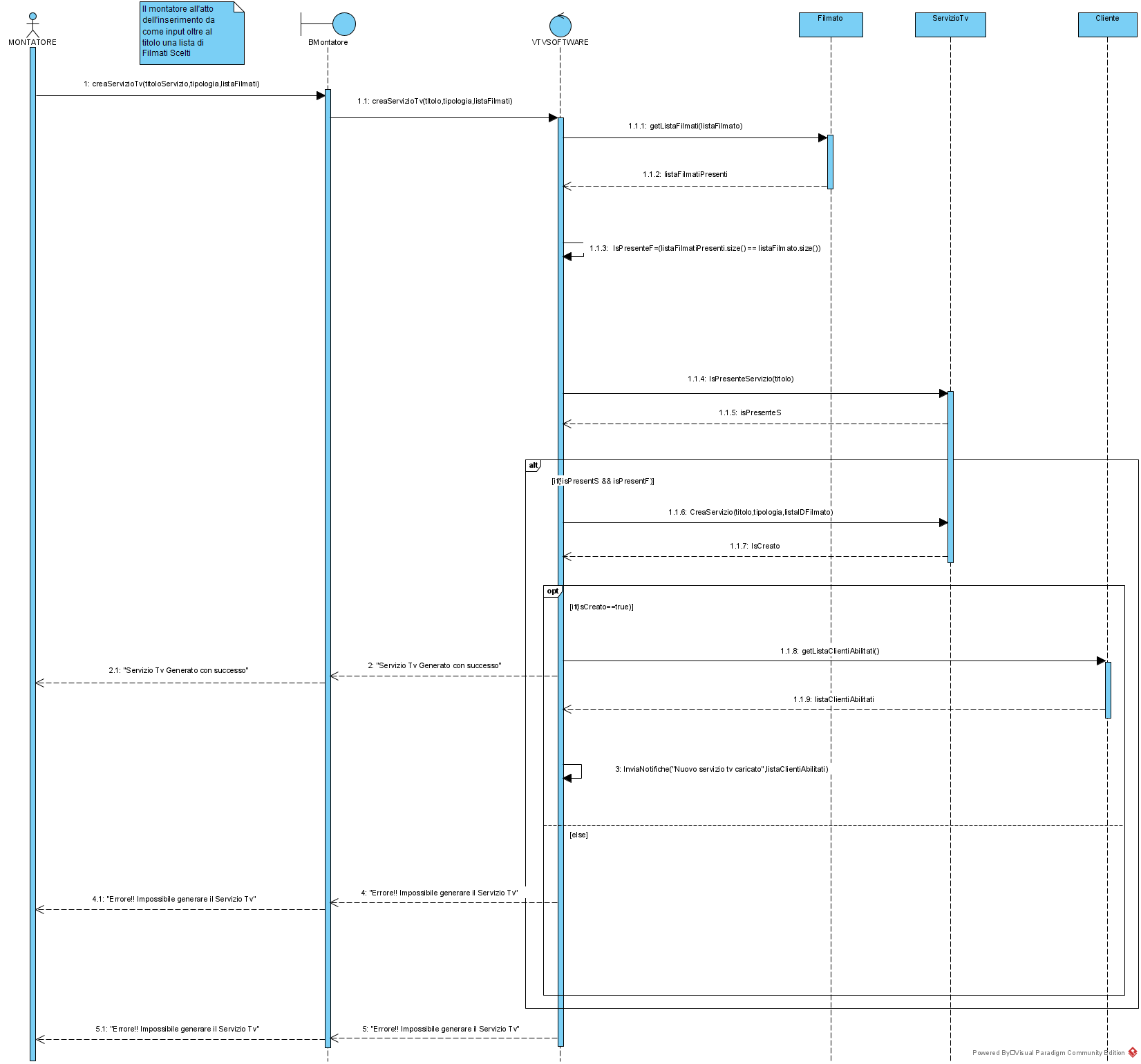


## Diagrammi di sequenza

**“CaricaFilmato”**

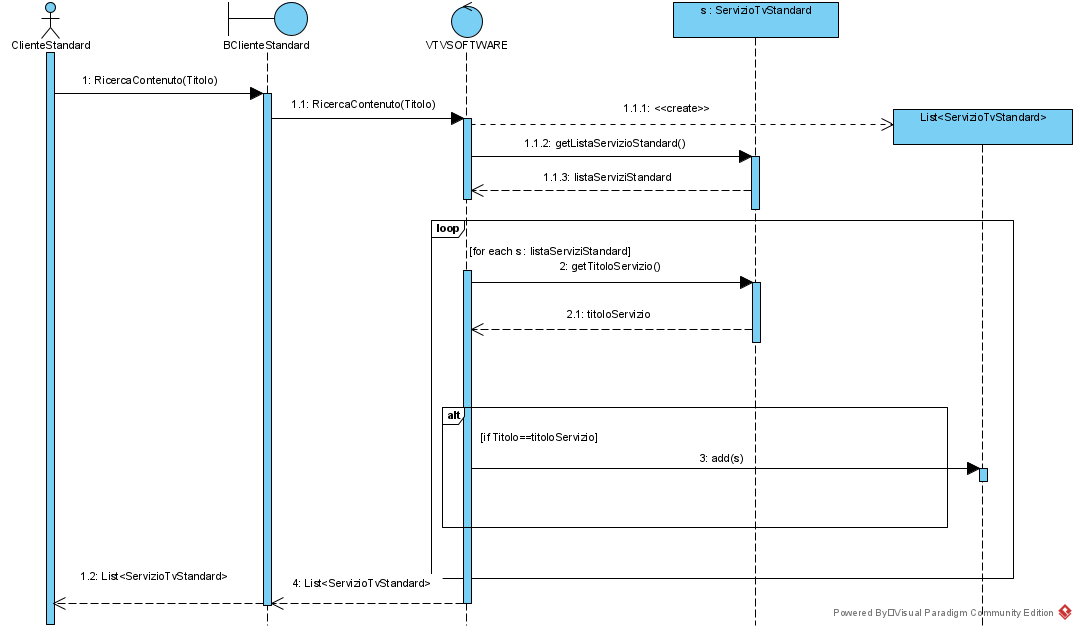
****

**“CreaServizioTV”**



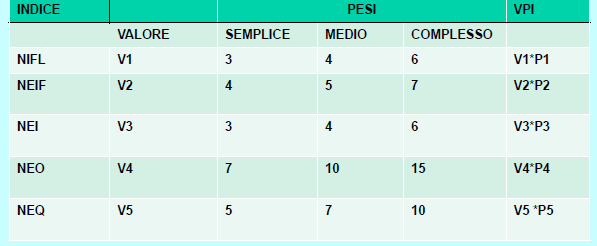
**“*RicercaContenuto*”**





# Stima dei costi

Andiamo ad analizzare la stima dei costi delle funzionalità considerando la seguente tabella:



**“CaricaFilmato”**

**External Input (EI)** - Informazioni distinte fornite dall’utente o da altre parti del sistema usate come dati di ingresso. Nel nostro caso, come parametro di input vado a considerare nome, data, durata e dimensione. Pertanto EI=4;

**External Output (EO)** – output distinti che il sistema restituisce all’utente come risultato delle proprie elaborazioni. In questo caso, non vado ad effettuare nessuna logica elaborativa per il calcolo di dati derivati. pertanto EO=0;

**External Inquiry (EQ)** – interrogazioni in linea che producono una risposta immediata del sistema. Pertanto EQ=0;

**Internal Logical File (ILF)** - aggregati logici di dati generati, usati e gestiti internamente dal sistema. Esempi tipici sono: tabelle di un database. Nel nostro caso mi aspetto come dati nome, data, durata e dimensione. Pertanto ILF=4;

**External Interface File (EIF)** - aggregati logici di dati scambiati o condivisi con altre applicazioni. In questo caso specifico EIF=0;

Supponendo che il peso per NEI sia medio mentre il peso dei rimanenti indicatori sia semplice si ottiene:

UFP = 1\*4 + 4\*3 = 28

Il numero di linee codice stimato se l’applicazione

è sviluppata in **C: 28\* 128 = 3544**

Se si adotta **Java: 28\*6 = 168**

**FATTORI CORRETTIVI**

**PRESTAZIONI** – Misura gli obbiettivi prestazionali dell’applicazione, richiesti o approvati dall’utente, sia in termine di risposta che di elaborazioni. Nel nostro caso il valore assegnato è 0: l’utente non ha specificato particolari requisiti prestazionali.

**EFIICIENZA PER L’UTENTE FINALE** – Esprime il grado di considerazione per i fattori umani e la facilità di utilizzo pe l’utente dell’applicazione misurata. Nel nostro caso il valore assegnato è 0: non sono state previste particolari tecniche legate all’userfriendliness.

**RIUSABILITÀ** – L’applicazione e il codice dell’applicazione sono appositamente progettati per essere utilizzati in altre applicazioni. Nel nostro caso il valore assegnato è 1: viene usato del codice riusabile nell’applicazione.

**STIMA FINALE**

Dove la sommatoria è uguale a 0+0+1 = 1, per cui i nostri FP saranno uguali a:

Il numero di linee codice stimato se l’applicazione

è sviluppata in **C: 10\* 128 = 1280**

Se si adotta **Java: 10\*6 = 60**

**“CreaServizioTV”**

**External Input (EI)** - Informazioni distinte fornite dall’utente o da altre parti del sistema usate come dati di ingresso. Nel nostro caso, come parametro di input vado a considerare il nome del Servizio TV da creare ed una lista di nomi che riguardano il filmato usato per creare il Servizio TV, pertanto EI = 2.

**External Output (EO)** – output distinti che il sistema restituisce all’utente come risultato delle proprie elaborazioni. In questo caso, non vado ad effettuare nessuna logica elaborativa per il calcolo di dati derivati. pertanto EO=0.

**External Inquiry (EQ)** – interrogazioni in linea che producono una risposta immediata del sistema. Pertanto EQ=0;

**Internal Logical File (ILF)** - aggregati logici di dati generati, usati e gestiti internamente dal sistema. Esempi tipici sono: tabelle di un database. Nel nostro caso mi aspetto come dati Id, nome, data, durata e dimensione. Pertanto ILF=1;

**External Interface File (EIF)** - aggregati logici di dati scambiati o condivisi con altre applicazioni. In questo caso specifico EIF=0;

I valori precedentemente conteggiati sono pesati secondo la tabella seguente:

Supponendo che il peso per NEI sia medio mentre il peso dei rimanenti indicatori sia semplice si ottiene:

UFP = 2\*4 + 1\*6 = 14

Il numero di linee codice stimato se l’applicazione

è sviluppata in **C: 14\* 128 = 1792.**

Se si adotta **Java: 14\*6 = 84.**

**FATTORI CORRETTIVI**

**FACILITA’ DI INSTALLAZIONE** – 4, poiché il software deve essere facilmente installabile, sia dai clienti che dagli operatori dell’emittente;

**PRESTAZIONI** - 3 è importante che il software sia reattivo durante l’arco di tutta la giornata, senza però le necessità imposte da un sistema hard real time;

**EFFICIENZA PER L’UTENZA FINALE** -5 è importante che il software disponga di tutti gli strumenti atti a renderne l’utilizzo quanto più user friendly possibile;

**RIUSABILITA’** – 2 ,con qualche modifica, potrebbero essere usati in altre applicazioni i moduli da noi implementati in questo progetto;

**STIMA FINALE**

Dove la sommatoria è uguale a 4+3+5+2 = 14, per cui i nostri FP saranno uguali a:

Il numero di linee codice stimato se l’applicazione

è sviluppata in **C: 11\* 128 = 1408**

Se si adotta **Java: 11\*6 = 66**

**“*RicercaContenuto*”**

**External Input (EI)** - Informazioni distinte fornite dall’utente o da altre parti del sistema usate come dati di ingresso. Nel nostro caso, come parametro di input vado a considerare Nome dell’evento, pertanto EI=1;

**External Output (EO)** – output distinti che il sistema restituisce all’utente come risultato delle proprie elaborazioni.In questo caso ,poiché vado a ricercare un contenuto nel mio Db ,non vado ad effettuare nessuna logica elaborativa per il calcolo di dati derivati. Pertanto EO=0;

**External Inquiry (EQ)** – interrogazioni in linea che producono una risposta immediata del sistema. In questo caso vado ad effettuare un ricerca attraverso una query nel mio DB. Pertanto EQ=1;

**Internal Logical File (ILF)** - aggregati logici di dati generati, usati e gestiti internamente dal sistema. Esempi tipici sono: tabelle di un database. Nel nostro caso mi aspetto come dati id, nome , dimensione, descrizione, durata. Pertanto ILF=1;

**External Interface File (EIF)** - aggregati logici di dati scambiati o condivisi con altre applicazioni.In questo caso specifico EIF=0;

Supponendo che il peso per NEI sia medio mentre il peso dei rimanenti indicatori sia semplice si ottiene:

UFP = 1\*6 + 1\*5+ 1\*4 = 15

Il numero di linee codice stimato se l’applicazione

è sviluppata in **C: 15\* 128 = 1920**

Se si adotta **Java: 15\*6 = 90**

**FATTORI CORRETTIVI**

**PRESTAZIONI** – Misura gli obbiettivi prestazionali dell’applicazione, offerti all’utente, sia in termine di risposta che di elaborazioni. Nel nostro caso il valore assegnato è 4: in quanto all’atto della ricerca l’utente si aspetta una risposta immediata.

**EFIICIENZA PER L’UTENTE FINALE** – Esprime il grado di considerazione per i fattori umani e la facilità di utilizzo pe l’utente dell’applicazione misurata. Nel nostro caso il valore assegnato è 4.

**RIUSABILITÀ** – L’applicazione e il codice dell’applicazione sono appositamente progettati per essere utilizzati in altre applicazioni. Nel nostro caso il valore assegnato è 3: viene usato del codice riusabile nell’applicazione.

**STIMA FINALE**

Dove la sommatoria è uguale a 4+4+3= 11, per cui i nostri FP saranno uguali a:

Il numero di linee codice stimato se l’applicazione

è sviluppata in **C: 11\* 128 = 2.304**

Se si adotta **Java: 11\*6 = 66**

# Piano di test funzionale

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI “*CaricaFilmato*”.**

|  |
| --- |
| **Nome** |
| * Nome Inesistente * Nome esistente [SINGLE] |

Abbiamo 1 categorie con 2classi quindi il Numero di test senza vincoli è pari a (1\*2) =2

Numero di test senza vincoli:2

Abbiamo nessun vincolo *error* e 1 vincolo *single*. Il metodo utilizzato per calcolare il numero dei casi di test è stato il seguente:

1)Considero tutte le combinazioni escludendo i vincoli SINGLE ed ERROR, ma contando i vincoli PROPERTY Ottengo: 1

2)Aggiungo le combinazioni precedentemente escluse con i vincoli ERROR, SINGLE 1 + 1 = 2(Ogni vincolo error o single

richiede un unico caso di test, con una sola – qualsiasi - combinazione di tutti gli altri; perciò non si moltiplica ma si aggiunge).

**TEST SUITE**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** | **Output Ottenuti** | **Post-condizioni Ottenute** | **Esito**  **(*FAIL*, *PASS*)** |
| 1 | Il cameraman vuole caricare un filmato non presente nel DataBase. | Nome filmato Inesistente | Il cameraman ha eseguito con successo il login. Il nome del filmato non deve essere presente all’interno del DataBase. | Nome: **Ricordo di Maradona** | **Visualizzato un messaggio che indica: “Il filmato è stato caricato con successo”.** | **Filmato correttamente caricato.** |  |  |  |
| 2 | Il cameraman vuole caricare un filmato già presente nel DataBase. | Nome filmato esistente | Il cameraman ha eseguito con successo il login.  Il filmato con nome “Ricordo di Maradona” è già presente nel DataBase | Nome: **Ricordo di Maradona** | **Visualizzato un messaggio che indica che il filmato è già presente.** | **Il filmato non viene caricato** |  |  |  |

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI “*RicercaServioTvStandard*”.**

|  |
| --- |
| **Nome** |
| * Nome Inesistente * Nome esistente [SINGLE] |

Abbiamo 1 categorie con 2 classi quindi il Numero di test senza vincoli è pari a (1\*2)=2

Abbiamo nessun vincolo *error* e 1 vincolo *single*. Il metodo utilizzato per calcolare il numero dei casi di test è stato il seguente:

1)Considero tutte le combinazioni escludendo i vincoli SINGLE ed ERROR,ma contando i vincoli PROPERTY Ottengo: 1

2)Aggiungo le combinazioni precedentemente escluse con i vincoli ERROR, SINGLE 1 + 1 = 2(Ogni vincolo error o single

richiede un unico caso di test, con una sola – qualsiasi - combinazione di tutti gli altri; perciò non si moltiplica ma si aggiunge).

**TEST SUITE**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** | **Output Ottenuti** | **Post-condizioni Ottenute** | **Esito**  **(*FAIL*, *PASS*)** |
| 1 | Il cameraman vuole caricare un filmato non presente nel DataBase. | Nome filmato Inesistente | Il cameraman ha eseguito con successo il login. Il nome del filmato non deve essere presente all’interno del DataBase. | Nome: **Ricordo di Maradona** | **Visualizzato un messaggio che indica: “Il filmato è stato caricato con successo”.** | **Filmato correttamente caricato.** |  |  |  |
| 2 | Il cameraman vuole caricare un filmato già presente nel DataBase. | Nome filmato esistente | Il cameraman ha eseguito con successo il login.  Il filmato con nome “Ricordo di Maradona” è già presente nel DataBase | Nome: **Ricordo di Maradona** | **Visualizzato un messaggio che indica che il filmato è già presente.** | **Il filmato non viene caricato** |  |  |  |

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI “*CreaServizioTV*”.**

|  |  |
| --- | --- |
| **Titolo** | **ListaFilmati** |
| * Titolo inesistente * Titolo esistente [SINGLE] | * Lista Filmati esistente * Lista Filmati inesistente [SINGLE] |

Abbiamo 1 categorie con 2 classe ed 1 categoria con 3 classi, quindi il Numero di test senza vincoli è pari a (1\*2)\*(3\*1) =6

Numero di test senza vincoli: 6

Abbiamo 3 vincoli *single*. Il metodo utilizzato per calcolare il numero dei casi di test è stato il seguente:

1)Considero tutte le combinazioni escludendo i vincoli SINGLE ed ERROR, ma contando i vincoli PROPERTY Ottengo: 1\*1=1

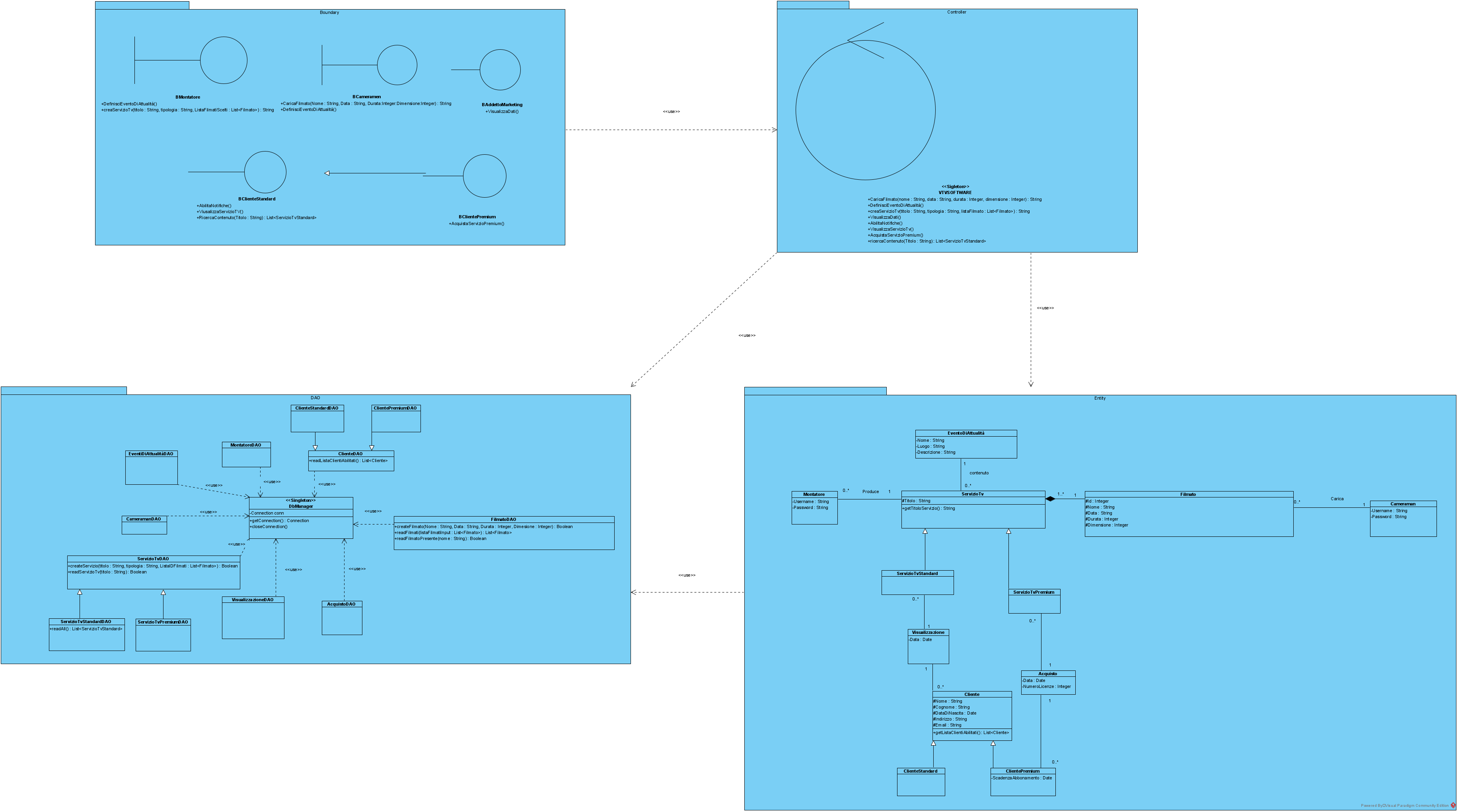
2)Aggiungo le combinazioni precedentemente escluse con i vincoli ERROR, SINGLE 1 +3= 4(Ogni vincolo error o single richiede un unico caso di test, con una sola – qualsiasi - combinazione di tutti gli altri; perciò non si moltiplica ma si aggiunge).

**TEST SUITE**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** | **Output Ottenuti** | **Post-condizioni Ottenute** | **Esito**  **(*FAIL*, *PASS*)** |
| **1** | Il montatore vuole creare un Servizio Tv non esistente nel DataBase. | Titolo inesistente  Lista Filmati esistente | Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all’interno del DataBase, mentre i filmati scelti sono presenti. | Nome: **Il dio del calcio**  **(inesistente)**  Lista Filmati: **Maradona1**  **Maradona2**  **(esistenti)** | **Visualizzato un messaggio che indica che il**  **Servizio TV è stato creato con successo** | **Servizio TV correttamente creato** |  |  |  |
| **2** | Il montatore vuole creare un Servizio Tv, che è già esistente nel DataBase. | Titolo esistente  Lista Filmati esistente | Il montatore si è autenticato con successo. Egli vuole creare un servizio Tv, che risulta già presente all’interno del DataBase. | Nome: **Il dio del calcio** (**esistente)**  Lista Filmati:  **Maradona1**  **Maradona2**  **(esistenti)** | **Visualizzato un messaggio che indica la mancata creazione del Servizio TV, poiché risulta già presente all’interno del DataBase.** | **Servizio TV non creato** |  |  |  |
| **3** | Il montatore vuole creare un Servizio Tv, in cui il titolo non è presente all’interno del Database, mentre, i filmati definiti non sono stati caricati nel DB. | Titolo inesistente  Lista Filmati inesistente | Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all’interno del DataBase, così come i filmati scelti. | Nome: **Il dio del calcio** (**inesistente)**  Lista Filmati:    **Maradona1**  **Maradona2**  **(inesistenti)** | **Visualizzato messaggio di errore.** | **Attesa indefinita di un parametro** |  |  |  |
| **4** | Il montatore vuole creare un Servizio Tv, in cui il titolo non è presente all’interno del Database, mentre, è presente almeno un filmato scelto dal montatore nel DB. | Titolo inesistente  Lista Filmati parziale | Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all’interno del DataBase. E’ presente almeno un filmato nella lista dei filmati definito dall’utente all’interno del DataBase. | Nome: **Il dio del calcio** (**inesistente)**  Lista Filmati:  **Maradona1**  **(esistenti)**  **Maradona2**  **(inesistenti)** | **Visualizzato messaggio di errore.** | **Servizio TV non creato** |  |  |  |

# Progettazione

## Diagramma delle classi

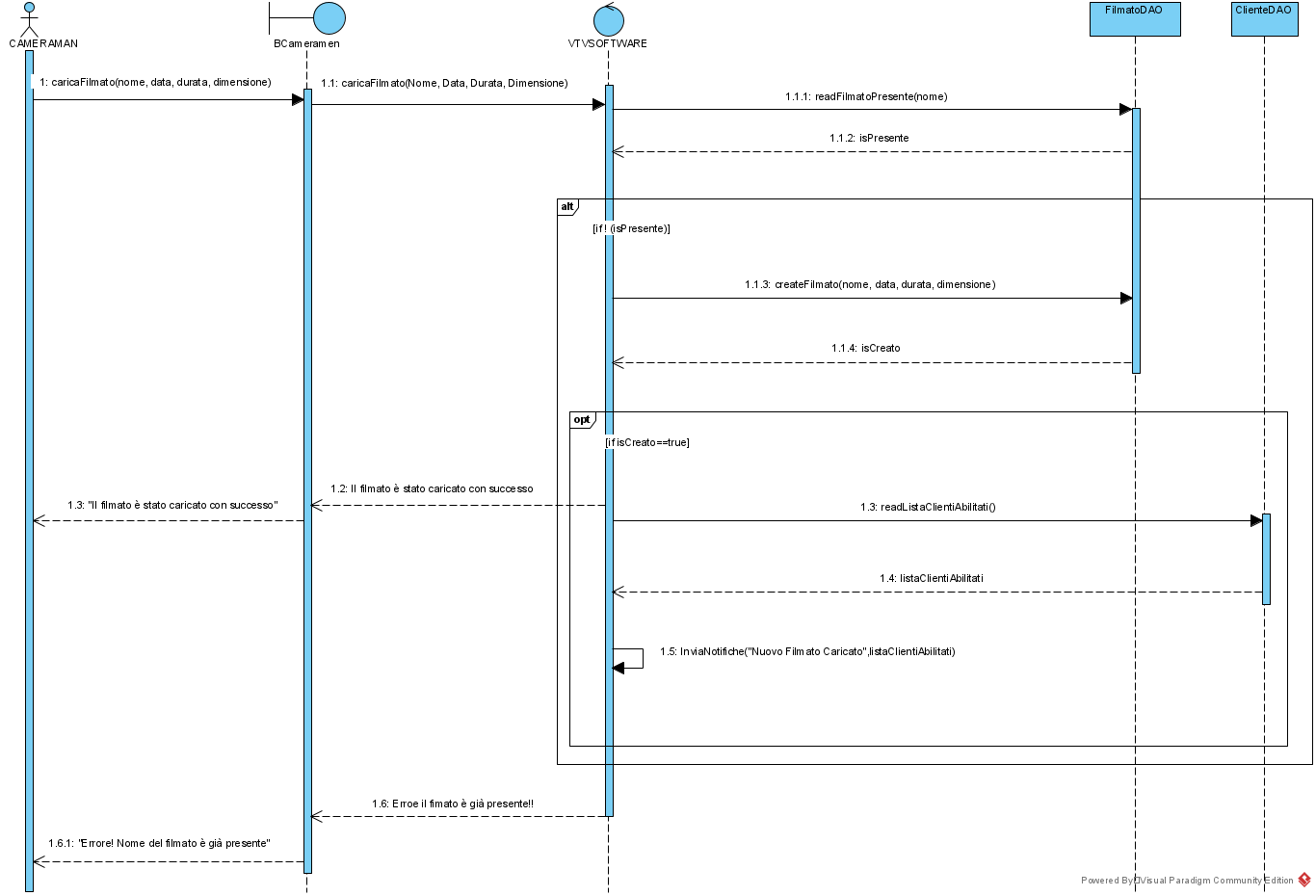


## 

## Diagrammi di sequenza

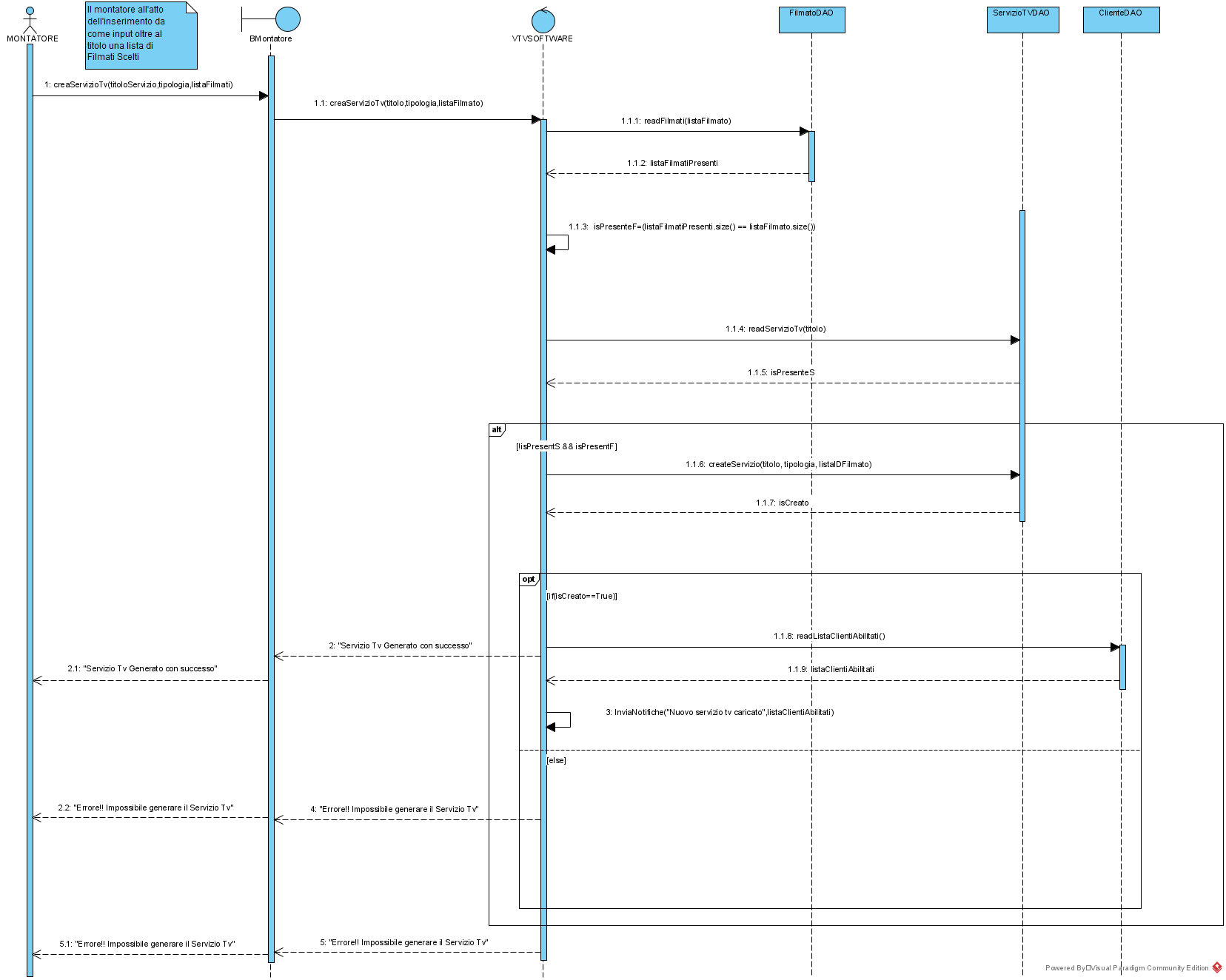
**“CaricaFilmato”**





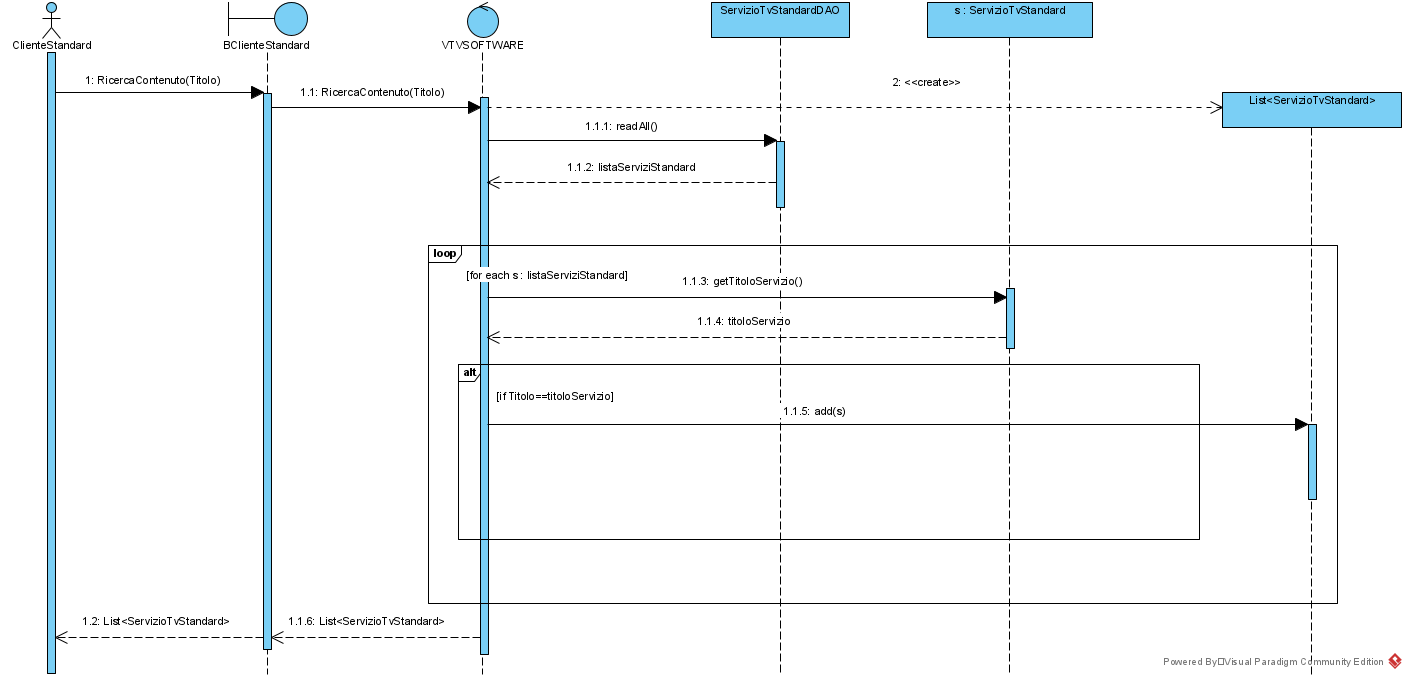
**“CreaServizioTV”**



****

**“*RicercaContenuto*”**

****



# Implementazione

Verranno adesso presentati gli artefatti di codifica:

* Package, Classi , Eccezzioni e Test

1. Boundary

* BCameramen
* BClienteStandard
* BMontatore

1. Controller

* VTVSOFTWARE

1. DAO

* ClienteDAO
* ClientePremiumDAO
* ClienteStandardDAO
* DbManager
* FilmatoDAO
* ServizioTvDAO
* ServizioTvPremiumDAO
* ServizioTvStandardDAO

1. Entity

* Cliente
* ClientePremium
* ClienteStandard
* Filmato
* ServizioTv
* ServizioTvPremium
* ServizioTvStandard

1. Exception

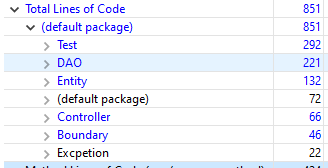
* DAOException
* VTVException

1. Test

* CaricamentoFilmato1Test
* CaricamentoFilmato2Test
* RicercaContenutoTest
* RicercaContenutoTest1
* Test\_CreazioneServizioTV
* Test\_FilmatoEsistenteServizioEsistente
* Test\_ServzioEsistenteFilmatoInesistente
* Artefatti necessari

1. JRE Systeme Library [JDK]
2. JUnit 4
3. DB H2 -1.4.199

* Numero Linee di codice VTV(definite mediate il plugin il plugin offerto dalla piattaforma Eclipse http://metrics.sourceforge.net/”. ) :



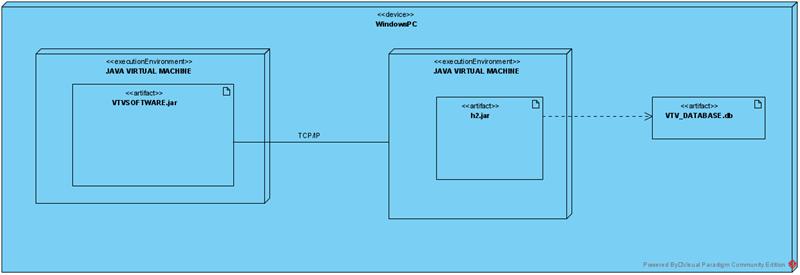
Per quanto riguarda la funzionalità RicercaContenuto, abbiamo prodotto un numero di righe di codice pari a 72. Per la funzionalità CaricaFilmato, abbiamo prodotto un numero di righe di codice pari a 59. Infine, per CreaServizioTv, abbiamo prodotto un numero di righe di codice pari a 60.

In tutti e tre i casi, il numero di linee di codice è risultato vicino al caso ideale, definito mediante la stima dei costi associata ad ogni singola funzionalità.

* Documentazione Javadoc:

[JAVADOC](https://github.com/IS-unina/canale_j_z-teamballo/tree/master/VTVSOFTWARE/Javadoc)

* Diagramma di deployment



# Testing

## Test funzionale

**“CaricaFilmato”**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** | **Output Ottenuti** | **Post-condizioni Ottenute** | **Esito**  **(*FAIL*, *PASS*)** |
| **1** | Il cameraman vuole caricare un filmato non presente nel DataBase. | Nome filmato Inesistente | Il cameraman ha eseguito con successo il login. Il nome del filmato non deve essere presente all’interno del DataBase. | Nome: **Ricordo di Maradona** | **Visualizzato un messaggio che indica: “Il filmato è stato caricato con successo”.** | **Filmato correttamente caricato.** | **Visualizzato un messaggio che indica: “Il filmato è stato caricato con successo”.** | **Filmato correttamente caricato.** | **PASS** |
| **2** | Il cameraman vuole caricare un filmato già presente nel DataBase. | Nome filmato esistente | Il cameraman ha eseguito con successo il login.  Il filmato con nome “Ricordo di Maradona” è già presente nel DataBase | Nome: **Ricordo di Maradona** | **Visualizzato un messaggio che indica che il filmato è già presente.** | **Il filmato non viene caricato** | **Visualizzato un messaggio che indica:** Errore il filmato è già presente!! | **Il filmato non viene caricato** | **PASS** |

**“*RicercaContenuto*”**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** | **Output Ottenuti** | **Post-condizioni Ottenute** | **Esito**  **(*FAIL*, *PASS*)** |
| **1** | Il cliente vuole ricercare un Servizio Tv esistente all’interno del DataBase | Nome esistente | Il cliente ha eseguito con successo il login. Il database deve contenere al proprio interno il servizio Tv ricercato dall’utente | Titolo: **CAPRI** | Lista servizi tv ricercati in  base al titolo | Il sistema non apporta modifiche al database | Lista servizi tv ricercati in  base al titolo | Il sistema non apporta modifiche al database | **PASS** |
| **2** | Il cliente vuole ricercare un Servizio Tv inesistente all’interno del DataBase | Nome inesistente | Il cliente ha eseguito con successo il login. Il database non deve contenere al proprio interno il servizio Tv ricercato dall’utente | Titolo: **CAPRI** | La lista servizi tv risulterà vuota | Il sistema non apporta modifiche al database | La lista servizi tv risulterà vuota | Il sistema non apporta modifiche al database | **PASS** |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** | **Output Ottenuti** | **Post-condizioni Ottenute** | **Esito**  **(*FAIL*, *PASS*)** |
| **1** | Il montatore vuole creare un Servizio Tv non esistente nel DataBase. | Titolo inesistente  Lista Filmati esistente | Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all’interno del DataBase, mentre i filmati scelti sono presenti. | Nome: **Il dio del calcio**  **(inesistente)**  Lista Filmati: **Maradona1**  **Maradona2**  **(esistenti)** | **Visualizzato un messaggio che indica che il**  **Servizio TV è stato creato con successo** | **Servizio TV correttamente creato** | **Visualizzato un messaggio che indica:**  “**Servizio Tv Generato con successo”** | **Servizio TV correttamente creato** | PASS |
| **2** | Il montatore vuole creare un Servizio Tv, che è già esistente nel DataBase. | Titolo esistente  Lista Filmati esistente | Il montatore si è autenticato con successo. Egli vuole creare un servizio Tv, che risulta già presente all’interno del DataBase. | Nome: **Il dio del calcio** (**esistente)**  Lista Filmati:  **Maradona1**  **Maradona2**  **(esistenti)** | **Visualizzato un messaggio che indica la mancata creazione del Servizio TV.** | **Servizio TV non creato** | **Visualizzato un messaggio che indica:** “ Errore!! Impossibile generare il Servizio Tv**”** | **Servizio TV non creato** | PASS |
| **3** | Il montatore vuole creare un Servizio Tv, in cui il titolo non è presente all’interno del Database, mentre, i filmati definiti non sono stati caricati nel DB. | Titolo inesistente  Lista Filmati inesistente | Il montatore si è autenticato con successo. Il titolo del servizio Tv non è presente all’interno del DataBase, così come i filmati scelti. | Nome: **Il dio del calcio** (**inesistente)**  Lista Filmati:    **Maradona1**  **Maradona2**  **(inesistenti)** | **Visualizzato messaggio di errore.** | **Servizio TV non creato** | **Visualizzato un messaggio che indica:** “ Errore!! Impossibile generare il Servizio Tv**”** | **Servizio TV non creato** | PASS |

***“CreaServizioTv”***

## Test strutturale

## Complessità ciclomatica

public String creaServizioTv(String titolo, String tipologia, List<Filmato> listaFilmato) throws VTVException {

/\*1/ Boolean isCaricato = false;

/\*2\*/ if (!tipologia.equals("s") && !tipologia.equals("p")) {

/\*3\*/ throw new VTVException("Tipologia può essere 's' oppure 'p'");

}

/\* List<Filmato> listaIdFilmato = filmatoDAO.readID(listaFilmato);

ServizioTvDAO servizioTvDAO = new ServizioTvDAO();

4

Boolean isPresentS = servizioTvDAO.readServizioTv(titolo);

\*/ Boolean isPresentF = (listaIdFilmato.size() == listaFilmato.size());

/\*5\*/ if (!isPresentS && isPresentF) {

/\*6\*/ isCaricato = servizioTvDAO.createServizio(titolo, tipologia, listaIdFilmato);

/\*7\*/ if (isCaricato) {

/\* listaClientiAbilitati = clienteDAO.*readListaClientiAbilitati*();

8 InviaNotifiche("Nuovo Servizio Tv Caricato", listaClientiAbilitati);

\*/

return "Servizio Tv Generato con successo";

} else {

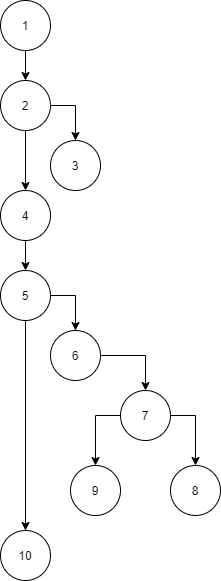
/\*9\*/ return "Errore!! Impossibile generare il Servizio Tv";

}

}

/\*10\*/ return "Errore!! Impossibile generare il Servizio Tv";

}



Cammino 1: 1-2-3

Cammino 2: 1-2-4-5-10

Cammino 3: 1-2-4-5-6-7-9

Cammino 3: 1-2-4-5-6-7-8

I nodi predicato sono 3 ovvero il nodo “2”, il nodo “5” ed il nodo “7”.

Il numero ciclomatico di MC Cabe sarà: 3+1 = 4

public List<ServizioTVStandard> ricercaContenuto (String titoloServizio) {

/\*1\*/ ArrayList<ServizioTvStandard> listaRisultati = new ArrayList<>();

/\*2\*/ for (ServizioTvStandard s : listaServiziStandard) {

/\*3\*/ if (s.getTitoloServizio().contains(titoloServizio)) {

/\*4\*/ listaRisultati.add(s);

}

}

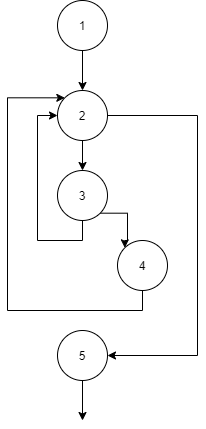
/\*5\*/ retrun listaRisultati;

}

Cammino 1: 1-2-3-4-2-5

Cammino 2: 1-2-3-2-5

Cammino 3: 1-2-5



I nodi predicato sono 2 ovvero il nodo “2” ed il nodo “3”.

Il numero ciclomatico di MC Cabe sarà: 2+1 = 3

public String caricaFilmato(String nome, String data, Integer durata, Integer dimensione) throws VTVException {

/\*1\*/ isPresente = filmatoDAO.readFilmatoPresente(nome);

/\*2\*/ if (!isPresente) {

/\*3\*/ isCreato = filmatoDAO.createFilmato(nome, data, durata, dimensione);

/\*4\*/ if (Boolean.***TRUE***.equals(isCreato)) {

/\* listaClientiAbilitati = clienteDAO.*readListaClientiAbilitati*();

5 InviaNotifiche("Nuovo Filmato Caricato", listaClientiAbilitati);

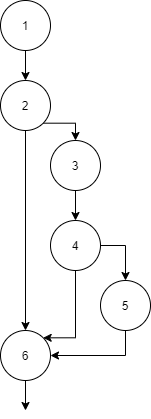
\*/ return "Il filmato è stato caricato con successo";

}

}

/\*6\*/ return "Erroe il fimato è già presente!!";

}



Cammino 1: 1-2-6

Cammino 2: 1-2-3-4-6

Cammino 3: 1-2-3-4-5

I nodi predicato sono 2 ovvero il nodo “2” ed il nodo “4”.

Il numero ciclomatico di MC Cabe sarà: 2+1 = 3

## Test di unità

*“CreaServizioTv”*

|  |  |  |
| --- | --- | --- |
| Cammini | Input | Output atteso |
| C1 | (!tipologia.equals("s") && !tipologia.equals("p") | throw new VTVException("Tipologia può essere 's' oppure 'p'"); |
| C2 | (tipologia.equals("s") || tipologia.equals("p") ;  (isPresentS || !isPresentF) | Errore!! Impossibile generare il Servizio Tv |
| C3 | (tipologia.equals("s") || tipologia.equals("p") ;  (!isPresentS && isPresentF);  (!isCaricato) | Errore!! Impossibile generare il Servizio Tv |
| C4 | (tipologia.equals("s") || tipologia.equals("p") ;  (!isPresentS && isPresentF);  (isCaricato) | Servizio Tv Generato con successo |

*“ricercaContenuto”*

|  |  |  |
| --- | --- | --- |
| Cammini | Input | Output atteso |
| C1 | s.getTitoloServizio().contains(titoloServizio) | return listaRisultati |
| C2 | !s.getTitoloServizio().contains(titoloServizio) | return listaRisultati |
| C3 | listaServiziTv vuota | return listaRisultati |

*“caricaFilmato”*

|  |  |  |
| --- | --- | --- |
| Cammini | Input | Output atteso |
| C1 | (isPresente) | “Erroe il fimato è già presente!!" |
| C2 | (!isPresente);  (Boolean.***FALSE***.equals(isCreato)) | “Erroe il fimato è già presente!!" |
| C3 | (!isPresente);  (Boolean.***TRUE***.equals(isCreato)) | “Il filmato è stato caricato con successo” |

## Test di unità con JUnit

## 

