UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale in Ingegneria del Software

Sviluppo di una Piattaforma per la Fruizione di Contenuti Museali da Casa tramite Smart TV in Ambienti Virtuali Tridimensionali

Anno Accademico 2014-2015

relatore
Ch.mo prof. Porfirio Tramontana

correlatore aziendale
Ing. Fabio Coronato

candidato
Daniele Marotta
matr. M63 / 000107

Indice

Introduzione

L'obiettivo del progetto è quello di sviluppare una piattaforma tecnologica con servizi evoluti per la fruizione di "percorsi virtuali" direttamente da casa, attraverso Smart TV. Il progetto beneficia della sempre maggior diffusione di tali dispositivi nelle case degli utenti.

A tal fine risulta necessaria la definizione di:

1. Servizi Web per la distribuzione di contenuti multimediali di natura eterogenea.
2. Interfacce e meccanismi di "Human-Computer Interaction" ottimizzati per la navigazione dei percorsi virtuali attraverso Smart TV. Le soluzioni tecnologiche di ultima generazione consentono l'interazione attraverso riconoscimento dei gesti (Gesture Recognition) abilitando un'esperienza utente innovativa ma allo stesso tempo intuitiva.
3. Meccanismi di "rendering 3D" per la rappresentazione tridimensionale dei percorsi virtuali e dei Beni Culturali. I componenti hardware CPU e GPU disponibili nelle più recenti Smart TV consentono la creazione e visualizzazione di ambienti tridimensionali ad alto impatto visivo aumentando notevolmente l'immersione da parte dell'utente.

Capitolo 1: Stato dell'Arte

1.1 Tecnologie e Soluzioni Simili

1.1.1 Visita Virtuale dei Musei Capitoloni

Il Tour Virtuale dei Musei Capitolini [1] è realizzato mediante tecniche di elaborazione e montaggio fotografico. Il tour permette la visita del museo muovendosi all'interno degli ambienti. Offre la possibilità di ruotare a 360°, in alto e in basso e zoomare fino ad avvicinarsi ai diversi elementi presenti nelle sale: sculture, quadri, dettagli architettonici, rivestimenti, soffitti e pavimenti. Il Tour virtuale è costituito da immagini ad alta risoluzione opportunamente "montate" al fine di ricreare l'illusione di un ambiente reale. Inoltre, in ogni sala si accede ad una sezione multimedia contenente la galleria fotografica, l'audio guida e il video descrittivo dell'ambiente. Per muoversi all'interno del Museo è possibile utilizzare una mappa, un menù a tendina oppure cliccando sulle frecce posizionate all'interno degli ambienti.

I punti meno forti di questa soluzione risultano essere:

1. Il movimento è possibile solo seguendo percorsi prestabiliti.
2. Guardandosi attorno è abbastanza evidente una distorsione delle immagini, sopratutto nelle parti laterali dello schermo.
3. Non è possibile guardare le opere d'arte da più punti di vista essendo queste

presentate tramite fotografie.

1.1.2 Visita Virtuale dei Musei Vaticani

La Visita Virtuale dei Musei Vaticani [2] è costituita da una Mappa bidimensionale sulla quale è possibile cliccare con il mouse per accedere ai dettagli delle singole strutture presenti nell'insieme dei Musei Vaticani. Non è prevista alcuna forma di movimento realistico e lo spostamento avviene solo selezionando la struttura cui si desidera accedere dalla suddetta mappa o da un semplice elenco.

Per ognuno dei Musei è in genere presente una descrizione testuale e un insieme di fotografie delle varie opere. Non è prevista alcuna forma di interattività con le immagini visualizzabili.

In alcuni casi è prevista la possibilità di selezionare la voce "Visita Virtuale". Tale Visita Virtuale aggiunge, per alcune stanze, un riquadro che permette di guardarsi attorno.

Questa soluzione è chiaramente molto limitata in quanto non fornisce agli utenti un'immersione completa non permettendo loro piena libertà di movimento all'interno degli ambienti.

1.1.3 Google Art Project

Google Art Project [3] è una raccolta di immagini in alta risoluzione di opere d'arte esposte in vari musei in tutto il mondo, oltre che una Visita Virtuale delle gallerie in cui esse sono esposte. La visita virtuale permette di sfogliare le immagini delle opere d'arte e di esplorare i musei con la stessa tecnologia utilizzata dal progetto "Street View", sempre di Google. Le immagini presenti sul sito hanno una risoluzione di 7 gigapixel (7 miliardi di pixel).

1.1.4 Google Earth

Google Earth [4] è un software che genera ricostruzioni virtuali della Terra utilizzando immagini satellitari ottenute dal telerilevamento terrestre, fotografie aeree e dati topografici memorizzati in una piattaforma GIS.

Il programma è distribuito gratuitamente dalla società Google.

In aggiunta a questo Google Earth permette la visualizzazione di edifici tridimensionali. Proprio questa funzionalità viene utilizzata, in alcuni casi, per mostrare la ricostruzione di alcuni monumenti (ad esempio il Colosseo così com'era in origine o la Cattedrale di Notre Dame), luoghi d'interesse culturale e musei (come il Museo del Prado).

1.2 Piattaforme Smart TV

1.2.1 HTML5

Ad oggi non esiste ancora uno standard ben definito per quanto riguarda lo sviluppo di Applicazioni per le varie piattaforme Smart TV esistenti. Nella maggior parte dei casi tali Applicazioni sono eseguite in un Browser Web e sfruttano le tecnologie HTML5 e Javascript. Purtroppo restano molte le funzionalità "platform-dependent": in particolare quelle relative alla gestione di dispositivi fisici come telecomando, microfono e web-cam. Vari produttori mettono a disposizione SDK (Software Development Kit) più o meno complete che possono presentare differenze strutturali che rendono la portabilità delle Applicazioni sviluppate un obiettivo difficile da conseguire.

A tal proposito nasce il progetto Smart TV Alliance [5]. Trattasi di un framework HTML5 e Javascript il cui scopo è rendere agevole lo sviluppo di Applicazioni per Smart TV su piattaforme eterogenee. Attualmente i produttori affiliati sono [6]:

- LG
- Panasonic
- Philips
- Toshiba

Le Applicazioni che utilizzano la Smart TV Alliance SDK possono essere eseguite su diversi modelli di questi produttori senza alcuna modifica. Inoltre sono incluse

funzionalità avanzate non presenti nei comuni Browser Web HTML5 nonché alcuni metodi di gestione dei dispositivi e sensori (telecomando, web-cam ecc.) "portabili" e non dipendenti dal produttore. Purtroppo le funzionalità di Gesture Recognition non rientrano in questo ambito.

1.2.2 Java

Una seconda possibilità per lo sviluppo di Applicazioni Smart TV è il linguaggio di programmazione Java. Fra le varie alternative spicca la piattaforma Android TV [7] che rimpiazzerà a breve la precedente Google TV. Disponibili come "set-top box" da connettere a Smart TV, i dispositivi Android TV offrono oltre ad un vasto catalogo di contenuti multimediali on-demand, la possibilità di scaricare ed eseguire App. La piattaforma è basata su Sistema Operativo Android e permette lo sviluppo di Applicazioni Java come nel caso degli Smart Phone. Il vantaggio più importante consiste nel fatto le Applicazioni sono eseguibili da qualunque Smart TV connesso ad un box Android TV. Resta da valutare l'effettiva diffusione di tali dispositivi nel medio e lungo termine.

1.3 Approfondimento Tecnologico

1.3.1 Gesture Recognition

Le tecnologie di Gesture Recognition permettono il riconoscimento di gesti tramite web-cam. In generale i gesti possono provenire da qualsiasi parte del corpo ma nell'ambito delle Smart TV il caso più comune è quello di mani e volto. Il controllo dell'Applicazione tramite tale tecnologia rende l'esperienza più immersiva ed intuitiva rispetto al tradizionale utilizzo del telecomando.

In generale è possibile distinguere varie tipologie di Gesture ed i loro utilizzi tipici:

Movimento di una Mano

Muovere una mano può simulare il comportamento del mouse, ovvero lo spostamento di un cursore, ma anche gestire operazioni di scorrimento dei contenuti visibili sullo

schermo.

Apertura e Chiusura di una Mano

Gesti che possono essere utilizzati per simulare la pressione ed il rilascio di un pulsante del mouse e, più in generale, per operazioni di selezione o conferma di una scelta.

Agitare una Mano (Saluto)

Una "mano che saluta" può inviare un segnale di accendimento o spegnimento o comunque comunicare l'intenzione di iniziare o terminare un'attività.

Scorrimento Veloce di una Mano verso Destra o Sinistra

Gesti solitamente associati alla navigazione all'interno di un'App a più schermate che l'utente può scorrere. In particolare lo scorrimento verso sinistra è di frequente associato all'intenzione di tornare alla schermata precedente.

Allontanare o Avvicinare due Mani

L'utente pone entrambe le mani davanti allo schermo (in genere chiuse) e le avvicina o le allontana. Questo gesto è associato comunemente ad operazioni di ingrandimento o rimpicciolimento dei contenuti visibili sullo schermo, come, ad esempio, una fotografia.

Ruotare due Mani

L'utente pone entrambe le mani davanti allo schermo e le ruota come se stesse afferrando un manubrio. Il piano di rotazione è parallelo al pannello della Smart TV rendendo possibile la rotazione attorno ad un unico asse. Questo gesto può essere utilizzato per la rotazione dei contenuti visibili sullo schermo, come, ad esempio, una fotografia.

Al momento Samsung è l'unico produttore ad aver messo in commercio modelli di Smart TV che supportino il riconoscimento dei gesti (anno 2013 in poi). La SDK di sviluppo

Samsung fornisce un'API, nota come **Smart Interactions 2.0**, che permette il riconoscimento di tutti i gesti descritti in precedenza.

1.3.2 Rendering 3D

Il rendering 3D è il processo di trasformazione di una scena tridimensionale in un'immagine bidimensionale che viene poi generalmente mostrata su di uno schermo. Il concetto è simile a quello dello scatto di una fotografia.

Nell'ambito delle Smart TV, ed in particolare delle applicazioni HTML5, individuiamo le seguenti possibilità per effettuare il rendering di quelli che sono i percorsi virtuali di nostro interesse, ovvero di modelli tridimensionali rappresentanti le opere d'arte:

1.3.2.1 WebGL

WebGL [8] (Web Graphics Library) è un'API Javascript per il rendering di grafica 3D con il supporto dell'accelerazione hardware all'interno di un Browser Web. WebGL è basato su OpenGL ES 2.0 e sfrutta l'oggetto Canvas introdotto con HTML5. Inoltre WebGL consente (o meglio, necessita della) scrittura di Shader in codice GLSL (OpenGL Shading Language). Uno Shader è un programma eseguito dalla GPU della macchina client che descrive il modo in cui debba essere effettuato il rendering. Con la scrittura di programmi Shader è possibile realizzare effetti grafica avanzata e di alto impatto visivo.

WebGL è una tecnologia che può risultare complessa, specialmente a causa dei limiti del linguaggio Javascript: per questo motivo esistono numerosi framework che agiscono da intermediari per semplificarne l'utilizzo. In generale si possono descrivere tali framework come un insieme di oggetti di alto livello che, formando uno strato di astrazione fra l'applicazione client e la libreria WebGL (basso livello), rendono l'utilizzo di quest'ultima più semplice ed intuitivo.

Tipicamente una libreria di rendering nasconde completamente lo strato sottostante (in questo caso WebGL) esponendo oggetti dalle interfacce più semplici e user-friendly. In alcuni casi tale semplicità di utilizzo si paga con una minore flessibilità: questo si traduce nella necessità di utilizzare la libreria in modo non previsto dagli sviluppatori (cosa non

sempre possibile) per ottenere alcune funzionalità non supportate out-of-the-box. L'utilizzo di un'API di alto livello risulta nella maggior parte dei casi molto conveniente a patto di effettuare un'analisi preliminare al fine di selezionare quella che meglio si addice al progetto, cioè quella che offre tutte le caratteristiche necessarie (o quella che più vi si avvicina). Va detto infine che, trattandosi di librerie Javascript, queste sono quasi per forza di cose progetti open-source gratuiti, è dunque facilmente possibile apportare modifiche adattive.

1.3.2.2 Unity

Unity [9] è un sistema di creazione di applicazioni multimediali (principalmente video game) multipiattaforma che include un IDE particolarmente ricco di funzionalità. I progetti creati con Unity sono esportabili su più di 15 piattaforme: desktop, mobile e web. Per quanto riguarda quest'ultimo la versione attuale Unity 4 supporta l'incorporazione di Unity Web Player all'interno di un documento html.

La lista ufficiale dei browser supportati è la seguente [7]:

- Chrome
- Firefox
- Internet Explorer
- Opera
- Safari

La prossima versione di Unity (versione 5) supporterà la funzione di export WebGL e non farà uso di Unity Web Player [8]. Questo significa che dovrebbe essere possibile eseguire applicazioni create con Unity in qualunque browser che supporti WebGL, inclusi quelli presenti nei sistemi Smart TV. La data ufficiale di rilascio non è ancora stata annunciata.

1.3.2.3 Adobe AIR for TV

Adobe AIR for TV [9] è una piattaforma per la creazione di applicazione multimediali

dedicata allo sviluppo di applicazioni per Smart TV. La SDK AIR for TV permette la creazione di applicazioni 2D e 3D con accelerazione hardware specifica per questo tipo di dispositivi. Fra i produttori i cui televisori supportano questa tecnologia spiccano LG e Samsung.

Capitolo 2: Business Process Modelling

Questo capitolo descrive la fase iniziale della progettazione dell'architettura SOA (Service Oriented Architecture) della piattaforma Muse@Home.

In primo luogo sono evidenziati i "Business Goals", gli obiettivi della piattaforma, conoscenza dei quali è essenziale per poter procedere alla definizione dei Processi (Business Processes o in breve BP). I BP descrivono le attività svolte dalla e grazie alla piattaforma e dunque permettono di evincere le Capabilities (funzionalità) richieste.

La definizione dei Servizi veri e propri è riservata ai capitoli successivi.

2.1 Definizione dei Business Goals

Come anticipato nell'Introduzione la piattaforma può essere suddivisa in due macro aree funzionali. Da un lato dovrà essere possibile la Gestione dei Contenuti, vale a dire l'insieme di operazioni che permettono la creazione, eliminazione e modifica dei Contenuti (in questo contesto: opere d'arte) e dall'altro dovrà essere possibile ottenere questi Contenuti per permettere lo svolgersi delle Visite Virtuali. Una Visita Virtuale è definita come l'atto di navigare all'interno di un ambiente tridimensionale che ricostruisca (in maniera più che possibile fedele) un luogo o una struttura d'interesse artistico o culturale. Durante una Visita Virtuale gli utenti (i "visitatori") possono navigare all'interno dell'ambiente tridimensionale con visuale libera e osservarne in dettaglio i Contenuti. Un

esempio tipico di ambiente virtuale è la ricostruzione di un museo con tutte le opere che esso espone.

La fruizione delle Visite Virtuali è un servizio a pagamento per gli utenti finali e per questo motivo la piattaforma deve prevedere meccanismi atti all'esecuzione delle transazioni finanziarie necessarie.

2.1.1 Tipologie di Contenuti

Sebbene non strettamente correlata alla visione d'ampio respiro dei BP è piuttosto rilevante anche in questa fase la definizione delle tipologie di contenuti che la piattaforma è in grado di gestire. Si prevedono tre tipologie di contenuti:

- **IMMAGINE** un'immagine (esempio: un quadro).
- **VIDEO** un video (esempio: un documentario storico).
- **OGGETTO3D** un oggetto tridimensionale (esempio: una scultura).

2.1.2 Svolgimento di una Visita Virtuale

Una Visita Virtuale consente la navigazione all'interno di un ambiente tridimensionale. Come già detto gli ambienti contengono opere d'arte come quadri e sculture e gli utenti devono essere in grado di godere al massimo di tali contenuti. A tal fine il visitatore può "inquadrare" il singolo contenuto per visualizzarne diversi dettagli attraverso operazioni di zoom o di rotazione della visuale.

Sono dunque previste due modalità di funzionamento:

Modalità Navigazione - modalità "base" durante la quale l'utente naviga all'interno dell'ambiente. Le operazioni effettuabili sono:

- *Muovi Visitatore* - il visitatore può spostarsi liberamente all'interno dell'ambiente. L'operazione consiste dunque nel modificare la posizione del visitatore con un movimento lungo due possibile direzione:
  - *Avanti* - lo spostamento avviene nella direzione lungo la quale il visitatore

sta guardando.

- *Indietro* - lo spostamento avviene in direzione opposta.

- *Ruota Visitatore* - per potersi muover liberamente l'utente deve essere in grado di modificare la direzione lungo la quale egli "guarda". Le operazioni di rotazione supportate sono:

  - *Ruota a Destra* - la rotazione avviene in senso orario lungo l'asse verticale.
  - *Ruota a Sinisitra* - la rotazione avviene in senso antiorario lungo l'asse verticale.
  - *Guarda in Alto* - la rotazione avviene in senso antiorario lungo l'asse orizzontale.
  - *Guarda in Basso* - la rotazione avviene in senso orario lungo l'asse orizzontale.

  Le operazioni "Ruota a Destra" e "Ruota a Sinistra" modificano la direzione del movimento del visitatore, le altre due operazioni ("Guarda in Alto" e "Guarda in Basso") permettono di osservare l'ambiente in maniera più completa ma non influiscono sul movimento.

- *Inquadra Oggetto* - passa a **Modalità Oggetto** (vedi dopo).

Modalità Oggetto - modalità durante la quale l'utente ha deciso di visualizzare i dettagli di un singolo oggetto. L'utente può effettuare le seguente operazioni:

- **Esci dalla Modalità Oggetto** - torna a **Modalità Navigazione**

- **Immagine**

- *Zoom In/Out* - è possibile effettuare operazioni di Zoom In e Zoom Out per evidenziare un dettaglio dell'immagine
- *Panning* - il panning consiste nel muovere la porzione visibile dell'immagine (è effettuabile, dunque, solo nel caso in cui l'immagine sia stata ingrandita tramite Zoom In)

- **Video**
  - *Inizia Riproduzione* - inizia la riproduzione dell'oggetto video
  - *Ferma Riproduzione* - ferma la riproduzione dell'oggetto video e azzera l'istante di riproduzione
  - *Pausa* - mette in stato di pausa la riproduzione
  - *Riposizionamento* - modifica l'istante di riproduzione dell'oggetto video

- **Oggetto3D**
  - *Zoom In/Out* - avvicina (Zoom In) o allontana (Zoom Out) il punto di vista
  - *Ruota* - è possibile ruotare l'oggetto per visualizzarne diverse parti. Le operazioni di rotazione possibili sono:
    - *Ruota verso Destra*
    - *Ruota verso Sinistra*
    - *Ruota verso l'Alto*
    - *Ruota verso il Basso*

2.2 Business Processes

Questo paragrafo dettaglia i BP individuati e ne include i relativi diagrammi.

- **Visita Virtuale** definisce il modo in cui gli utenti fruiscono del servizio messo a disposizione dalla piattaforma. Include le interazioni con il servizio di gestione dei contenuti e con un eventuale servizio di pagamenti elettronici messo a disposizione da terzi.

- **Caricamento Contenuti** descrive il modo in cui i fornitori di contenuti (ad es. un museo) interagiscono con il servizio di gestione contenuti per caricare sulla piattaforma le risorse messe da loro a disposizione.

2.2.1 Notazione BPMN

Per la modellazione dei BP verrà utilizzata la Business Process Model Notation [10] [11] [12] (BPMN in breve). BPMN è una rappresentazione grafica per la specifica dei BP, la versione attuale è la 2.0. La notazione è basata su diagrammi di flusso simili ai diagrammi delle Attività UML. Lo scopo della notazione è fornire una rappresentazione facilmente comprensibile sia agli utenti esperti (analisti o tecnici addetti all'implementazione dei sistemi) sia agli utenti "business" senza specifiche competenze tecniche; questo senza rinunciare alla possibilità di modellare scenari anche molto complessi.

Gli elementi utilizzati nei paragrafi successivi sono qui riportati e brevemente descritti:

- Swimlane

  Le swimlanes sono utilizzate per organizzare e raggruppare le Attività. Le swimlane sono di due tipologie: una "Pool" rappresenta uno dei principali partecipanti al processo, nella maggior parte dei casi un'organizzazione o un intero sistema. Una Pool può essere suddivisa gerarchicamente in "Lanes" che dunque rappresentano una sottostruttura come un reparto o uno specifico ruolo

- Attività

  Un'attività rappresenta un compito da svolgere, un lavoro o un passo elaborativo da compiere all'interno del processo. Sono contenute all'interno di una Swimlane che ne indica l'esecutore. Sono previste alcune specializzazioni, tra le quali abbiamo utilizzato quella del "Sotto-Processo". Un'attività di tipo Sotto-Processo sta ad indicare che ciò che viene svolto è in genere complesso e modellato da un vero e proprio BP separato e di livello gerarchicamente inferiore. In questo modo è possibile mantenere diagrammi anche estremamente articolati leggibili e quindi più

agevolmente comprensibili.

- Eventi

  Gli eventi sono utilizzati per rappresentare fatti che possono verificarsi (in alcuni casi opzionalmente) durante il corso del processo. Gli eventi in genere determinano l'esecuzione di un'Attività e possono essere di categorizzati nel seguente modo:

  Evento Inziale è l'evento che da inizio al BP (che determina l'esecuzione della prima Attività)

  Evento Intermedio un evento generico che si verifica tra quello iniziale e quello finale. Un Evento Intermedio può anche essere di tipo *Throw* o *Catch*. Un evento *Throw* è un evento che rappresenta la generazione di una qualche entità, ad esempio un messaggio. Similmente, un evento di tipo *Catch* rappresenta l'evento di ricezione di una qualche entità (messaggio o altro).

  Evento Finale l'evento che termina il BP (vale a dire dopo il quale non viene svolta alcuna Attività)

- Connessioni

  Le connessioni sono utilizzate per collegare fra di loro gli elementi del diagramma, ad esempio attività ed eventi. Le connessioni sono orientate e descrivono dunque il flusso del processo specificando l'ordine di esecuzione dei vari elementi.

- Gateway

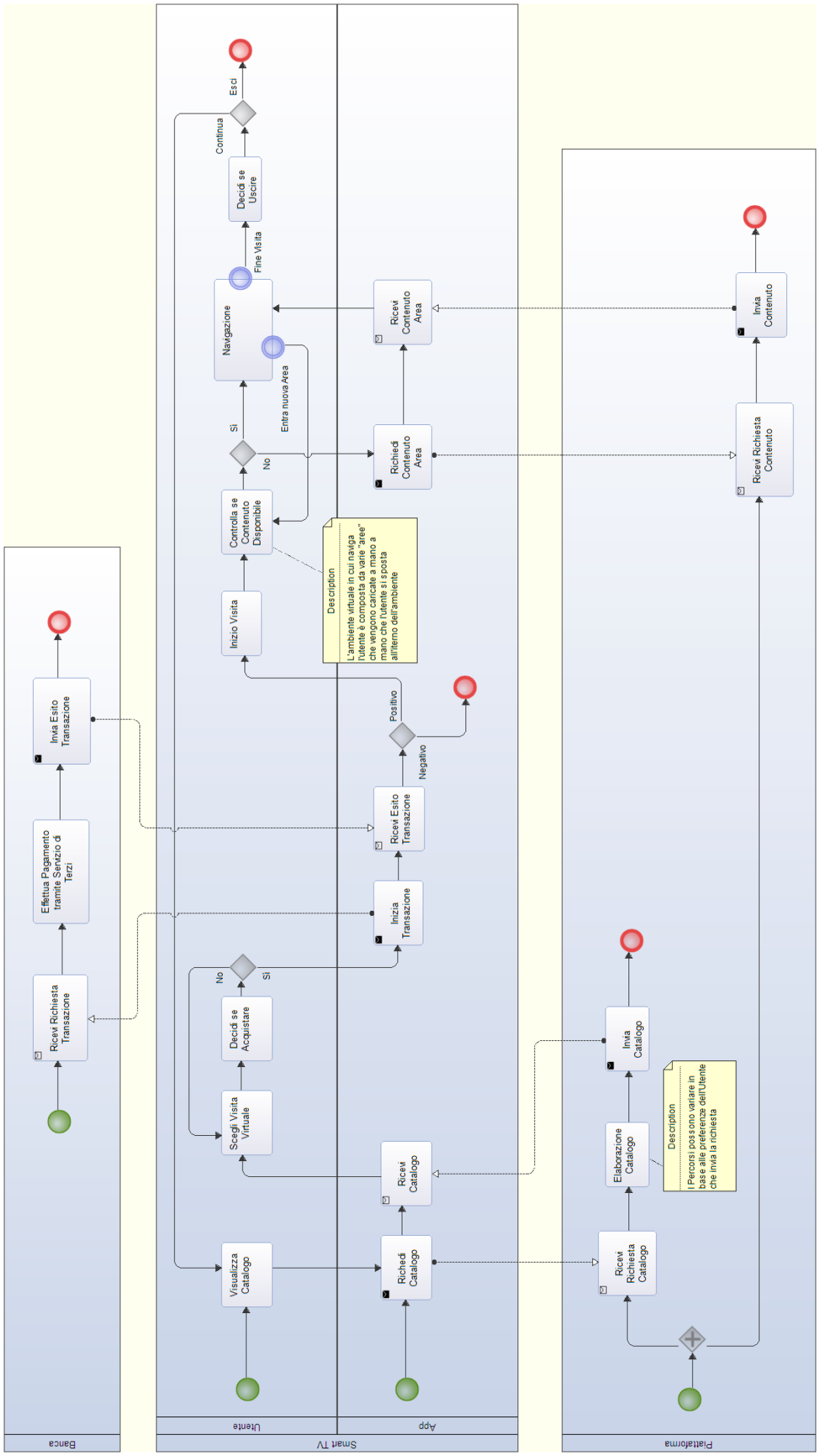
  Gli elementi Gateway sono utilizzati per differenziare la sequenza di attività di un processo in base ad una o più condizioni o per evidenziare come diversi *path* di esecuzione vengano eseguiti in parallelo. In generale un Gateway rappresenta la necessità di controllare una specifica condizione in base alla quale decidere come procedere.

2.2.2 Visita Virtuale

Il processo di Visita Virtuale coinvolge tre differenti entità:

- **Smart TV** descrive le attività dell'utente durante la fruizione del servizio
- **Banca** definisce il modo in cui vengono effettuati i pagamenti
- **Piattaforma** rappresenta l'entità che distribuisce i contenuti

Il fine ultimo delle attività descritte da questo processo è quello di mostrare come l'utente utilizza l'applicazione client per visualizzare i contenuti e come la suddetta applicazione comunichi con le altre entità del sistema al fine di realizzare le funzionalità necessarie.


Esci
Continua
Decidi se Uscire
Fine Visita
Navigazione
Entra nuova Area
Si
No
Controlla se Contenuto Disponibile
Description
L'ambiente virtuale in cui naviga l'utente è composta da varie "aree" che vengono caricate a mano a mano che l'utente si sposta all'interno dell'ambiente
Inizio Visita
Ricevi Contenuto Area
Richiedi Contenuto Area
Invia Contenuto
Ricevi Richiesta Contenuto
Positivo
Negativo
Ricevi Esito Transazione
Inizia Transazione
Invia Esito Transazione
Effettua Pagamento tramite Servizio di Terzi
Ricevi Richiesta Transazione
No
Si
Decidi se Acquistare
Scegli Vista Virtuale
Visualizza Catalogo
Ricevi Catalogo
Richiedi Catalogo
Invia Catalogo
Elaborazione Catalogo
Description
I Percorsi possono variare in base alle preferenze dell'Utente che invia la richiesta
Ricevi Richiesta Catalogo
Banca
Utente
App
Smart TV
Piattaforma

Inizialmente l'utente visualizza il catalogo dei contenuti disponibili. Tale catalogo è disponibile presso il gestore dei contenuti che l'applicazione contatta per ottenere il catalogo. Una volta che il catalogo è disponibile l'utente sfoglia i vari contenuti e ne sceglie uno. Con il termine "contenuto" si intende in questo contesto una **Visita Virtuale**, cioè un ambiente tridimensionale entro il quale gli utenti navigano per visualizzare gli oggetti (ad es., opere d'arte quali quadri e sculture) in esso presenti. Ad ogni visita è associato un costo e l'utente è libero di decidere se è disposto o meno ad effettuare il pagamento. In caso affermativo la visita ha inizio.

Come detto, la visita consiste nella navigazione all'interno di un ambiente virtuale. Per ottimizzare la godibilità della visita in termini di tempi di caricamento l'ambiente è suddiviso in diverse aree che vengono caricate "al volo" quando l'utente vi fa ingresso. Il diagramma del Business Process evidenzia questo aspetto mostrando come le varie aree vengono caricate con una richiesta al gestore dei contenuti al momento opportuno.

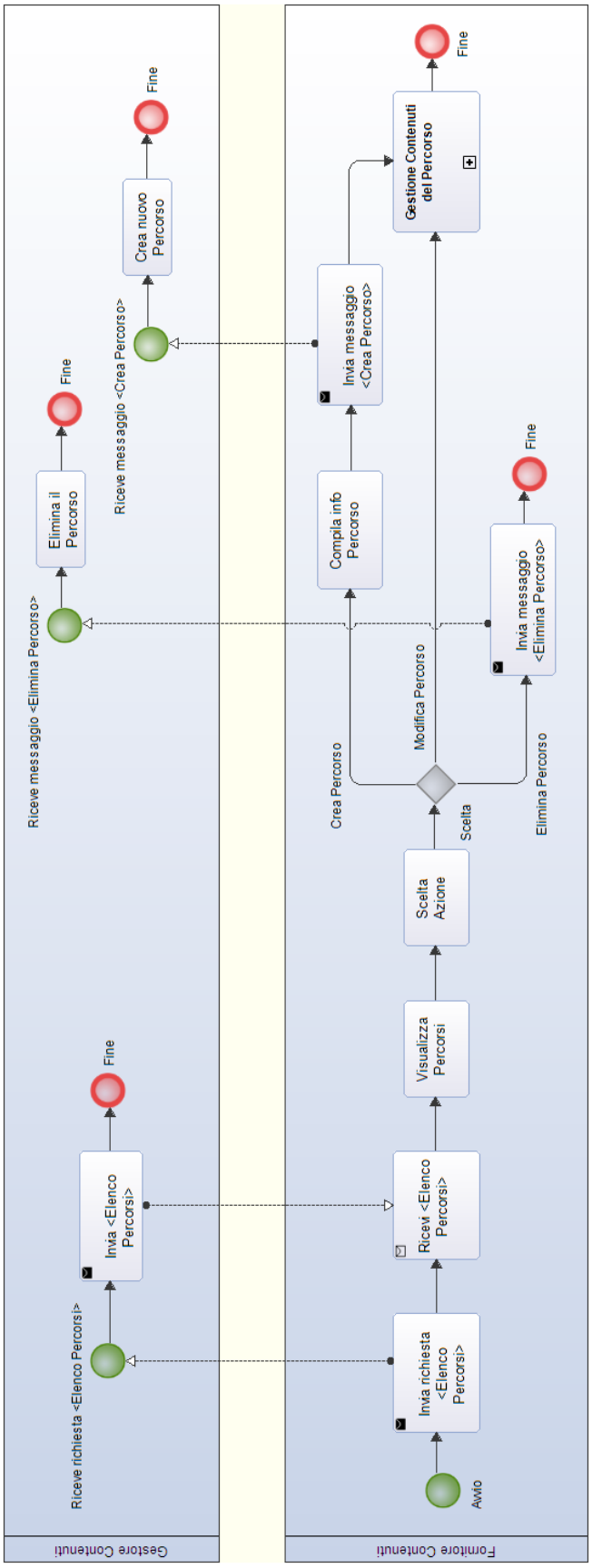
Quando l'utente lo ritiene opportuno può terminare la visita e decidere se chiudere l'applicazione o tornare al catalogo iniziale.

2.2.3 Caricamento Contenuti

Le entità coinvolte nel processo di Caricamento Contenuti sono:

- **Museo** rappresenta l'utente (ad es., operatore museo) che carica i contenuti e crea i percorsi
- **Piattaforma** è la stessa entità del processo precedente. In questo ambito accetta le richieste necessarie allo svolgimento di operazioni di tipo creazione, modifica ed eliminazione di percorsi e contenuti

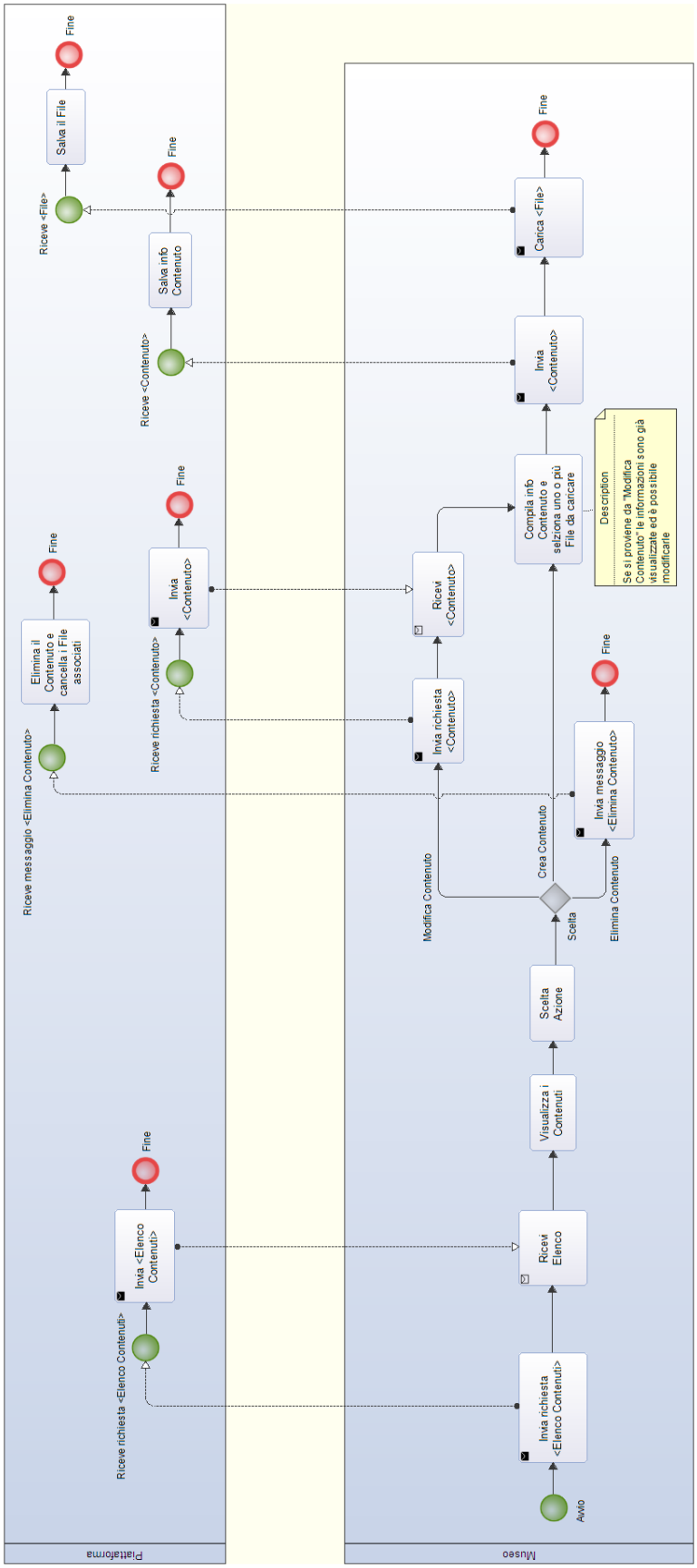
Dunque questo processo mostra come il museo si interfaccia alla piattaforma per eseguire le attività di caricamento.


Gestore Contenuti
Formtore Contenuti
Avvio
Invia richiesta <Elenco Percorsi>
Riceve richiesta <Elenco Percorsi>
Riceve <Elenco Percorsi>
Invia <Elenco Percorsi>
Fine
Visualizza Percorsi
Scelta Azione
Scelta
Crea Percorso
Modifica Percorso
Elimina Percorso
Compila info Percorso
Invia messaggio <Crea Percorso>
Gestione Contenuti del Percorso
Fine
Riceve messaggio <Crea Percorso>
Crea nuovo Percorso
Fine
Riceve messaggio <Elimina Percorso>
Elimina il Percorso
Fine
Invia messaggio <Elimina Percorso>
Fine

Il processo comincia con una richiesta dei percorsi associati al fornitore. Il sistema visualizza tale elenco e l'operatore decide se:

- Creare un percorso
- Eliminare un percorso
- Modificare (aggiungere/rimuovere contenuti) un percorso

Queste operazioni richiedono in ogni caso uno scambio di messaggi con il gestore di contenuti in quanto è questi che memorizza le varie informazioni.
I casi di creazione e modifica di un percorso portano al sottoprocesso di gestione dei contenuti, riportato nella seguente figura:


Piattaforma
Museo
Avvio
Invia richiesta <Elenco Contenuti>
Riceve richiesta <Elenco Contenuti>
Riceve Elenco
Visualizza i Contenuti
Scelta Azione
Scelta
Invia <Elenco Contenuti>
Fine
Riceve richiesta <Contenuto>
Invia <Contenuto>
Riceve messaggio <Elimina Contenuto>
Elimina il Contenuto e cancella i File associati
Modifica Contenuto
Invia richiesta <Contenuto>
Riceve <Contenuto>
Salva info Contenuto
Crea Contenuto
Compila info Contenuto e seleziona uno o più File da caricare
Description
Se si proviene da "Modifica Contenuto" le informazioni sono già visualizzate ed è possibile modificarle
Elimina Contenuto
Invia messaggio <Elimina Contenuto>
Invia <Contenuto>
Riceve <File>
Salva il File
Carica <File>

Lo svolgimento dell'attività di gestione dei contenuti è analogo a quello della gestione dei percorsi visto in precedenza. Infatti, dopo aver visualizzato l'elenco dei contenuti già caricati sulla piattaforma l'operatore potrà decidere se:

- Caricare un contenuto
- Eliminare un contenuto
- Modificare informazioni e risorse associate ad un contenuto

2.3 Capabilities

2.3.1 Business Competencies

Le Business Competencies derivanti dai processi individuati nel paragrafo precedente sono riportate nella tabella seguente:

	Business Competencies			
	Smart TV	Museo	Piattaforma	Banca
Direct				
Control				Verifica
Execute	Sfoglia Catalogo Acquista Biglietto Visita Virtuale	Gestisci Percorsi Gestisci Contenuti	Accesso Catalogo Accesso Contenuti Streaming Gestione Percorsi Gestione Contenuti	Transazione

La tabella evidenzia le componenti di business con le relative "aree funzionali". Ad ognuna di tali aree funzionali è associabile un livello operazionale:

- **Direct** decisioni strategiche
- **Control** funzioni di verifica e controllo
- **Execute** esecuzione di azioni

2.3.1.1 Smart TV

Aree funzionali

- **Sfoglia Catalogo (execute)** visualizzazione del catalogo e scelta della visita
- **Acquista Biglietto (execute)** acquisto del biglietto virtuale che garantisce l'accesso ad una delle visite disponibili
- **Visita Virtuale (execute)** fruizione della visita virtuale, consistente nella navigazione all'interno dell'ambiente e nella visualizzazione dei vari contenuti

2.3.1.2 Museo

Aree funzionali

- **Gestisci Percorsi (execute)** creazione ed eliminazione dei percorsi
- **Gestisci Contenuti (execute)** creazione, modifica ed eliminazione dei contenuti

2.3.1.3 Piattaforma

Aree funzionali

- **Accesso Catalogo (execute)** elaborazione e distribuzione del catalogo delle visite virtuali disponibili
- **Accesso Contenuti (execute)** distribuzione dei contenuti disponibili all'interno di una visita
- **Streaming (execute)** servizio di video streaming
- **Gestione Percorsi (execute)** elaborazione delle richieste di creazione ed eliminazione di contenuti
- **Gestione Contenuti (execute)** elaborazione delle richieste di creazione, modifica ed eliminazione di contenuti

2.3.1.4 Banca

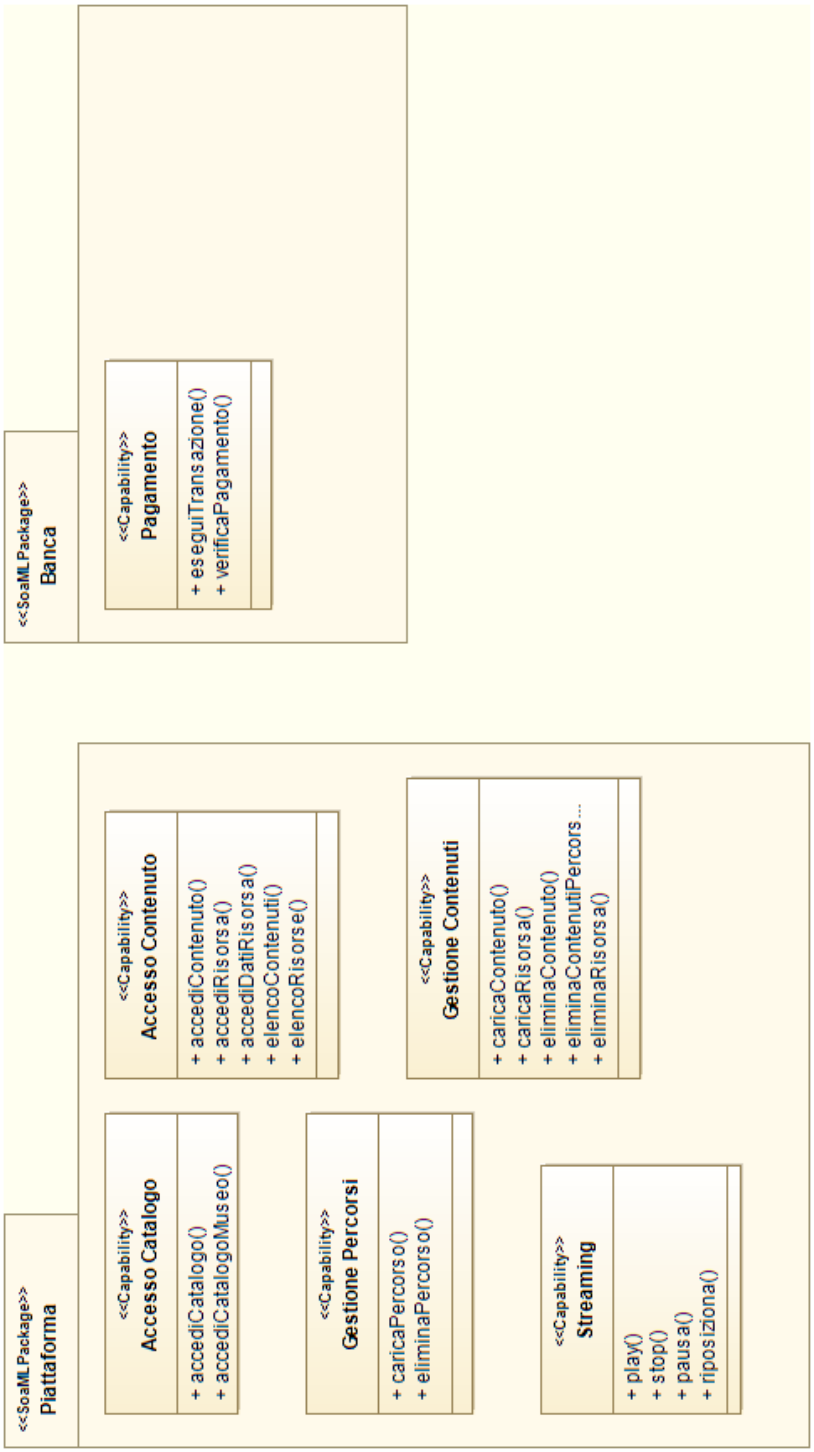
Aree funzionali

- **Verifica (control)** determinazione dell'effettuabilità di un pagamento (ad es., verifica della disponibilità economica per il metodo di pagamento selezionato)
- **Transazione (execute)** esecuzione del pagamento

2.3.2 Candidate Services

Le componenti e le relative capabilities e aree funzionali del paragrafo precedente

individuano i Candidate Services, come in figura:



Ognuna delle capability è racchiusa in un package che ne indica l'area di competenza. Tali capabilities rappresentano i servizi con le operazioni messe a disposizione dei service consumers.

Capitolo 3: Business Architecture Modelling

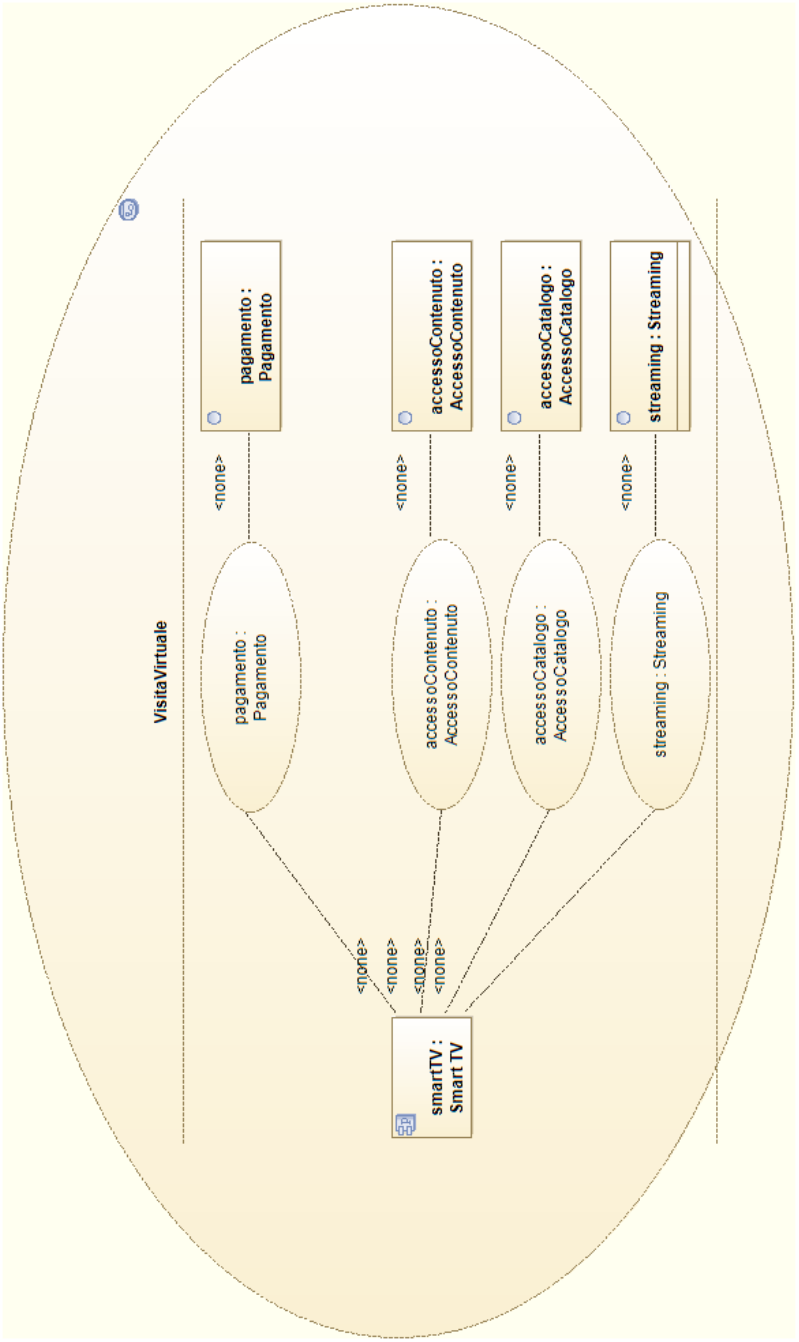
3.1 Service Architectures

Le Service Architectures mostrano le interazioni fra consumer e provider di servizi nell'ambito dei due scenari di Visita Virtuale e Caricamento Contenuti.

I diagrammi di Service Architecture fanno parte della notazione SoaML [13] (Service oriented architecture Modelling Language), un profilo UML per la modellazione di servizi nell'ambito di Architetture orientate ai Servizi.

3.1.1 Visita Virtuale

Questa Service Architecture mostra le interazioni fra l'applicazione client su Smart TV e i vari service providers durante una visita virtuale.

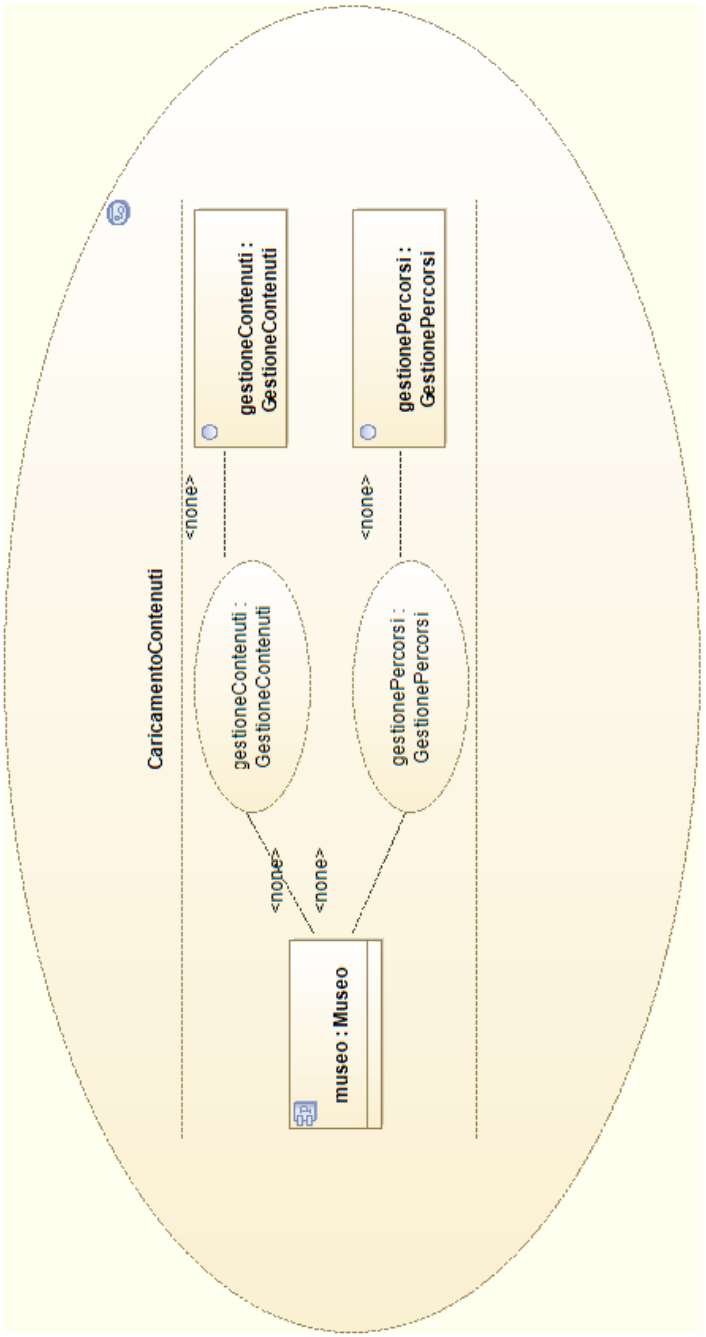

VisitaVirtuale
pagamento : Pagamento
accessoContenuto : AccessoContenuto
accessoCatalogo : AccessoCatalogo
streaming : Streaming
smartTV : Smart TV
<none>

I servizi "consumati" dal client sono:

- **accessoCatalogo** - per ottenere il catalogo delle visite disponibili
- **accessoContenuto** - per ottenere i contenuti di una visita
- **streaming** - per effettuare lo streaming dei contenuti di tipo video
- **pagamento** - per acquistare il diritto di accesso ad una visita

3.1.2 Caricamento Contenuti

La seconda Service Architecture mostra le come i musei interagiscono con la piattaforma per effettuare le operazioni di caricamento.

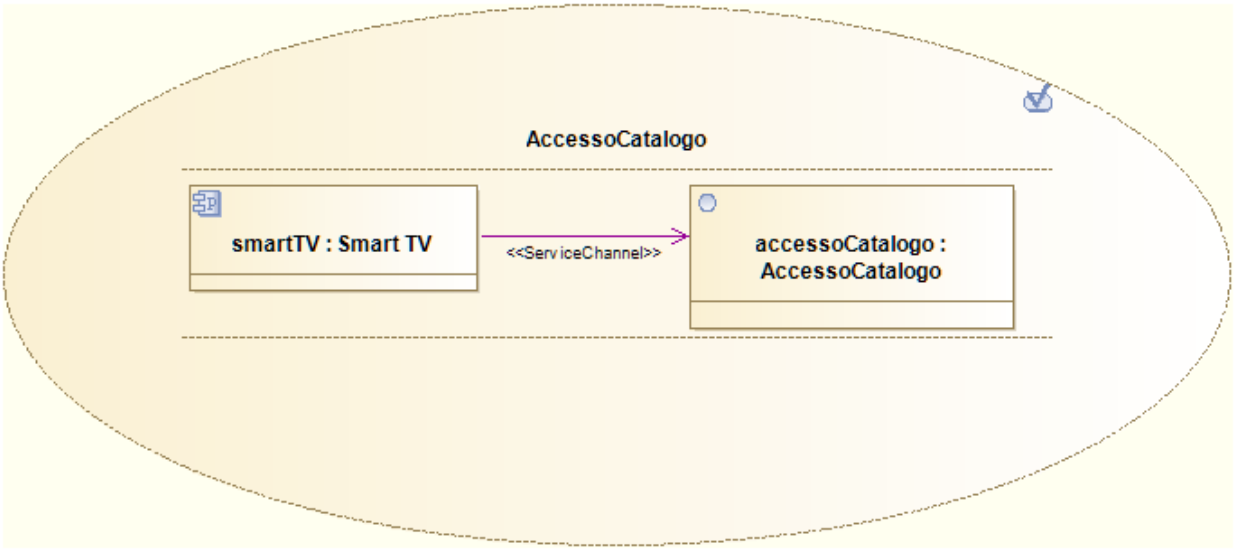

CaricamentoContenuti
gestioneContenuti : GestioneContenuti
gestionePercorsi : GestionePercorsi
<none>
<none>
gestioneContenuti : GestioneContenuti
gestionePercorsi : GestionePercorsi
<none>
<none>
museo : Museo

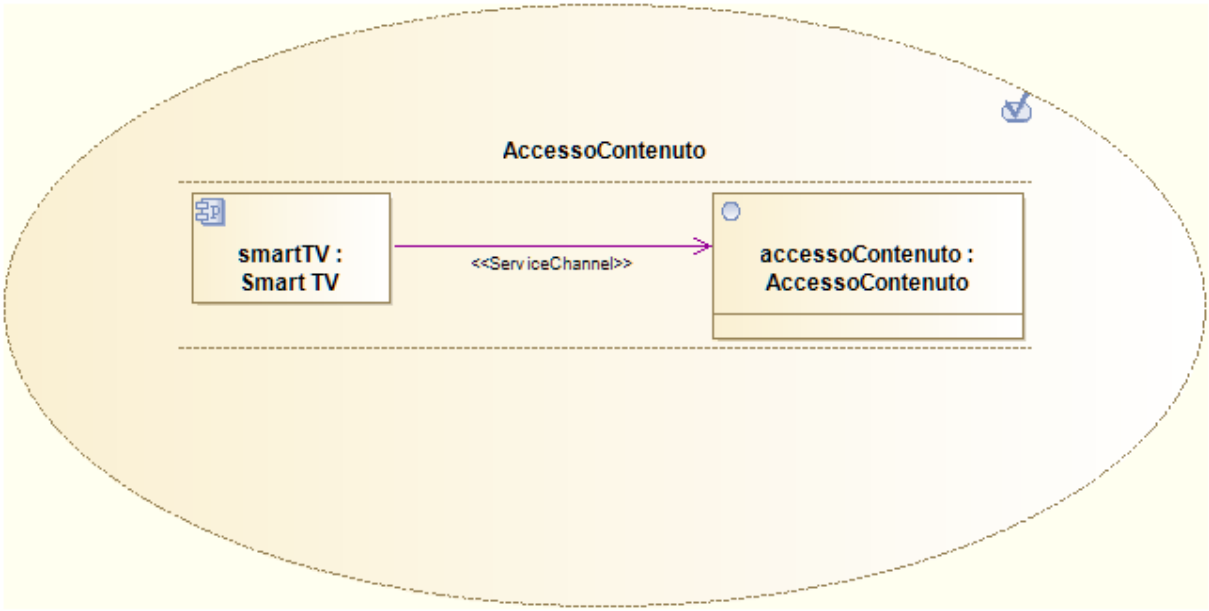
Dalla figura si evince l'utilizzo dei seguenti servizi:

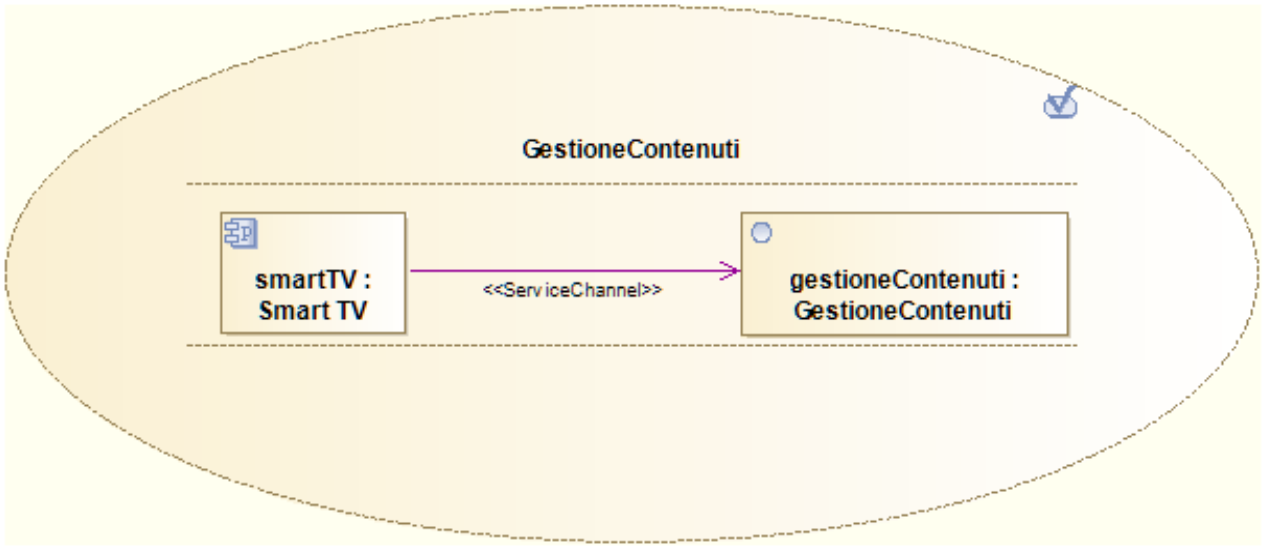
- **gestionePercorsi** - offre le operazioni inerenti al caricamento di percorsi
- **gestioneContenuti** - offre le operazioni inerenti al caricamento di contenuti

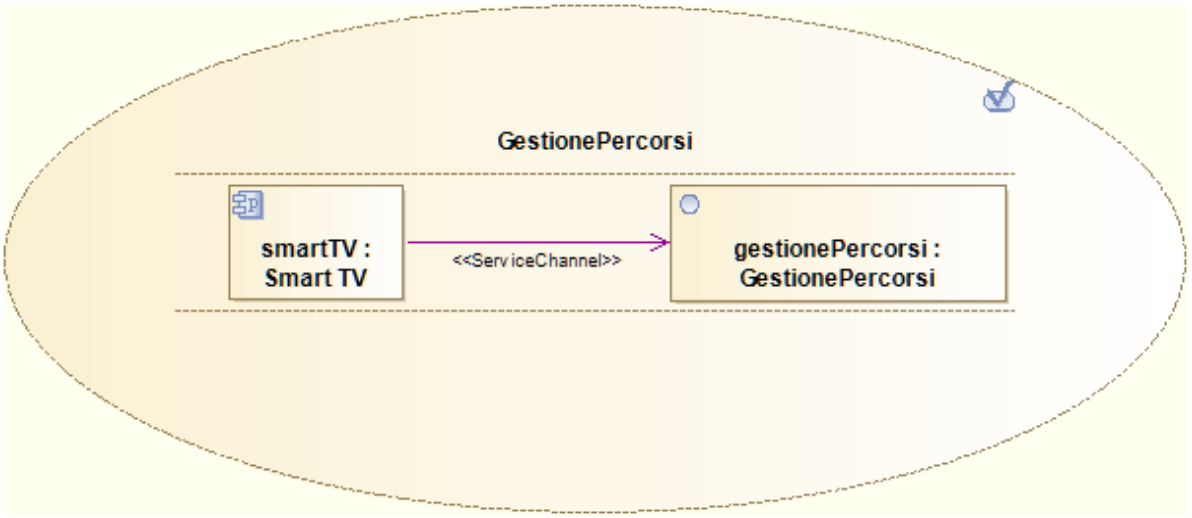
3.2 Service Contracts

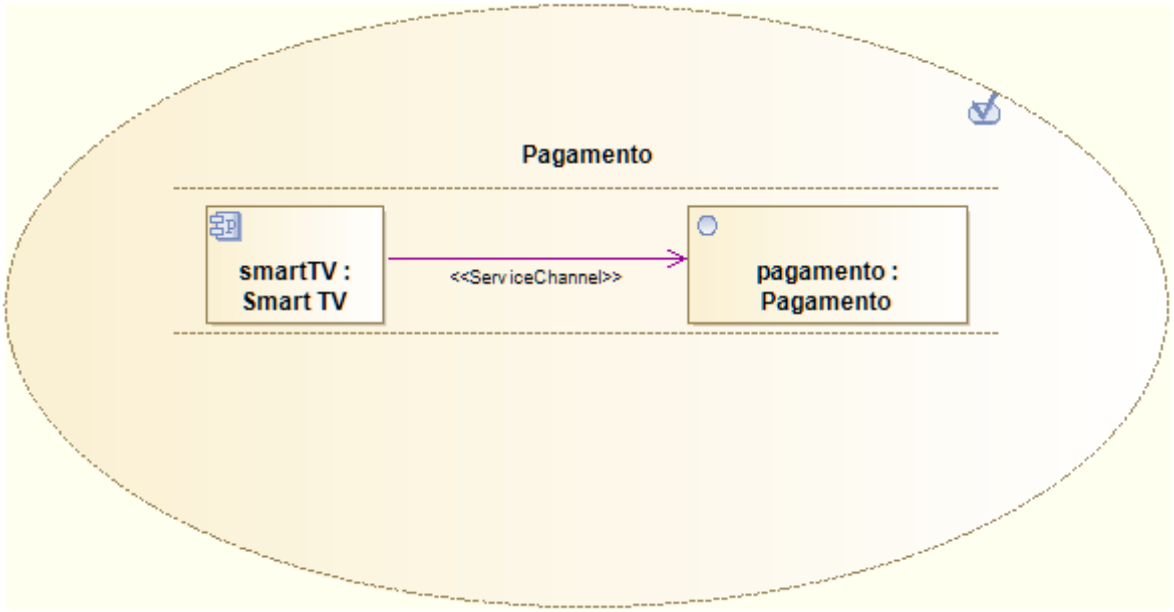
Ad ognuno dei servizi si associa un contratto che individua in maniera chiara i ruoli dei vari partecipanti in chiave di consumer e provider. I service contracts sono qui presentati come diagrammi:

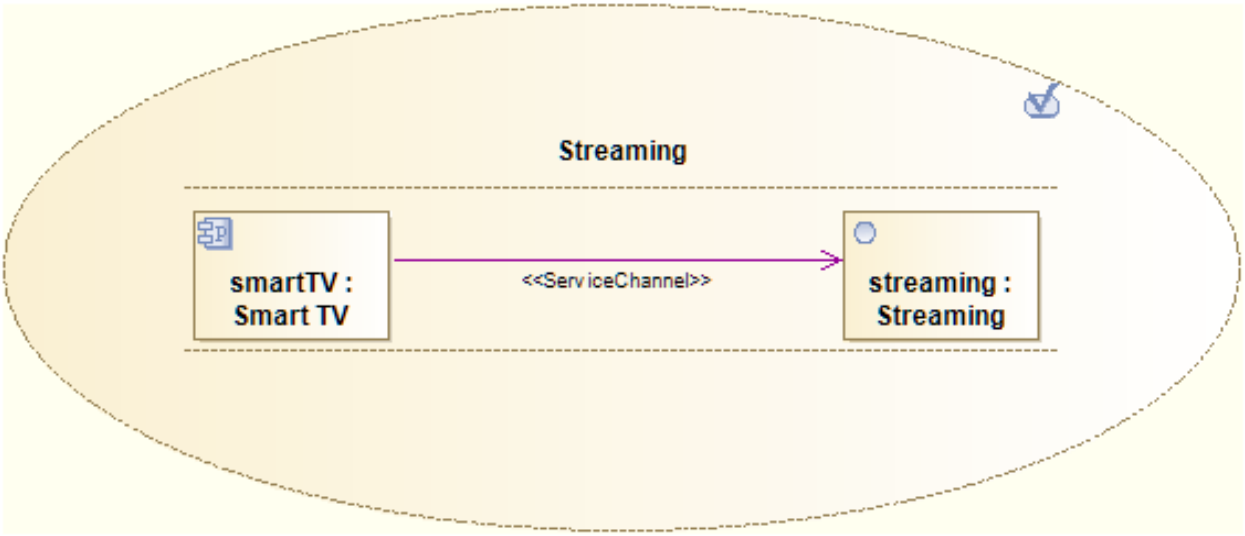

AccessoCatalogo
smartTV : Smart TV
<<ServiceChannel>>
accessoCatalogo : AccessoCatalogo


AccessoContenuto
smartTV : Smart TV
<<ServiceChannel>>
accessoContenuto : AccessoContenuto


GestioneContenuti
smartTV : Smart TV
<<ServiceChannel>>
gestioneContenuti : GestioneContenuti


GestionePercorsi
smartTV : Smart TV
<<ServiceChannel>>
gestionePercorsi : GestionePercorsi


Pagamento
smartTV : Smart TV
<<ServiceChannel>>
pagamento : Pagamento


Streaming
smartTV : Smart TV
<<ServiceChannel>>
streaming : Streaming

Capitolo 4: System Architecture Modelling

4.1 Structural Modelling

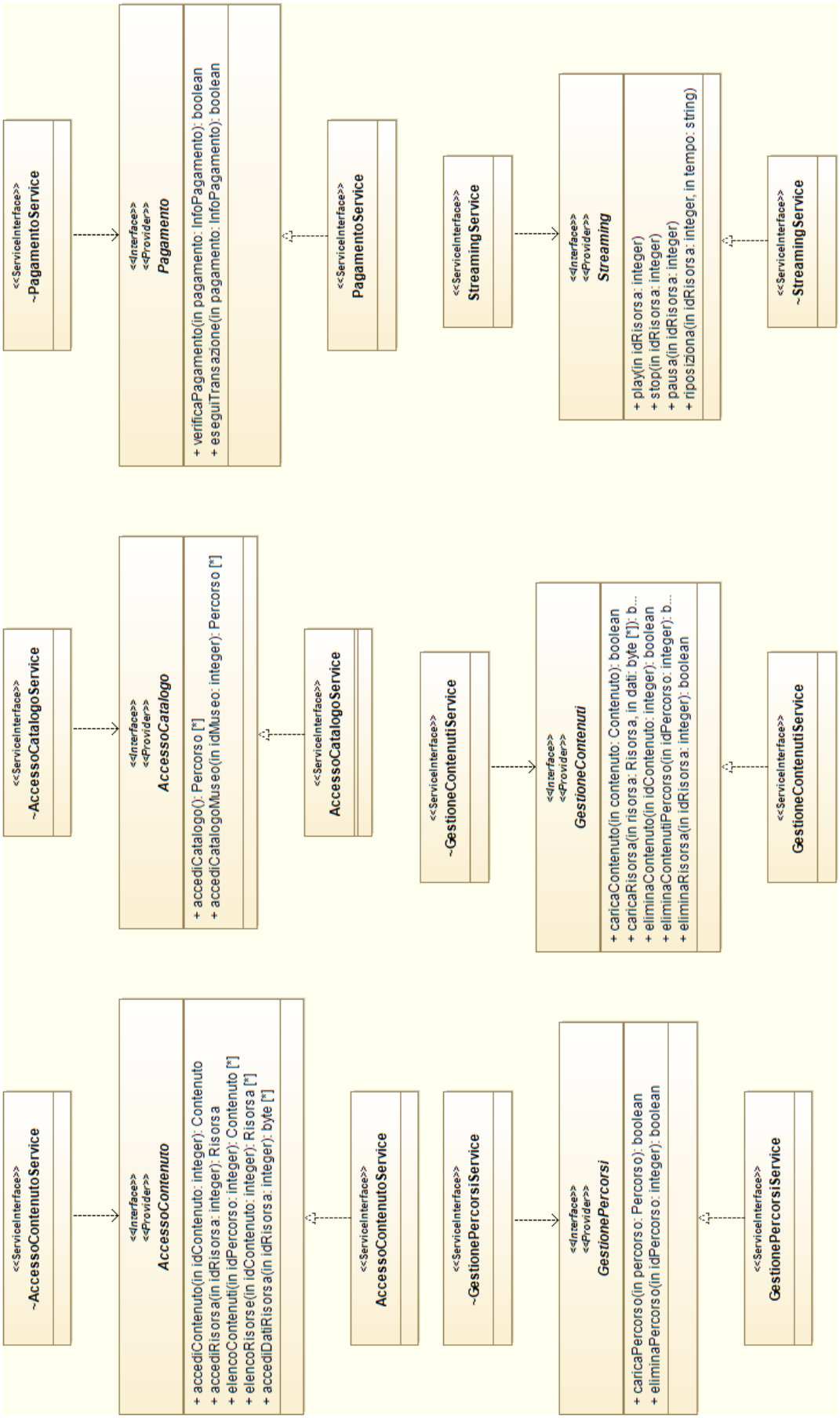
4.1.1 Service Interfaces

In questo paragrafo sono definite le service interfaces che consentono di descrivere servizi bidirezionali per gestire la comunicazione tra provider e consumer. Una service interface dichiara un insieme di operazioni e può utilizzare una o più interfacce.

In figura sono mostrare le interfacce semplici nonché quelle dei servizi individuati, elencate di seguito:

- **AccessoCatalogoService**
- **AccessoContenutiService**
- **PagamentoService**
- **GestionePercorsiService**
- **GestioneContenutiService**
- **StreamingService**

Per ogni interfaccia semplice, associata al provider, vi è una relazione di **Usage** e **Realization** con le rispettive interfacce di servizio.


«ServiceInterface»
~PagamentoService
«Interface»
«Provider»
Pagamento
+ verificaPagamento(in pagamento: InfoPagamento): boolean
+ eseguiTransazione(in pagamento: InfoPagamento): boolean
«ServiceInterface»
PagamentoService
«ServiceInterface»
StreamingService
«Interface»
«Provider»
Streaming
+ play(in idRisorsa: integer)
+ stop(in idRisorsa: integer)
+ pausa(in idRisorsa: integer)
+ riposiziona(in idRisorsa: integer, in tempo: string)
«ServiceInterface»
~StreamingService
«ServiceInterface»
~AccessoCatalogoService
«Interface»
«Provider»
AccessoCatalogo
+ accediCatalogo(): Percorso [*]
+ accediCatalogoMuseo(in idMuseo: integer): Percorso [*]
«ServiceInterface»
AccessoCatalogoService
«ServiceInterface»
~GestioneContenutiService
«Interface»
«Provider»
GestioneContenuti
+ caricaContenuto(in contenuto: Contenuto): boolean
+ caricaRisorsa(in risorsa: Risorsa, in dati: byte [*]): b...
+ eliminaContenuto(in idContenuto: integer): boolean
+ eliminaContenutiPercorso(in idPercorso: integer): b...
+ eliminaRisorsa(in idRisorsa: integer): boolean
«ServiceInterface»
GestioneContenutiService
«ServiceInterface»
~AccessoContenutoService
«Interface»
«Provider»
AccessoContenuto
+ accediContenuto(in idContenuto: integer): Contenuto
+ accediRisorsa(in idRisorsa: integer): Risorsa
+ elencoContenuti(in idPercorso: integer): Contenuto [*]
+ elencoRisorse(in idContenuto: integer): Risorsa [*]
+ accediDatiRisorsa(in idRisorsa: integer): byte [*]
«ServiceInterface»
AccessoContenutoService
«ServiceInterface»
~GestionePercorsiService
«Interface»
«Provider»
GestionePercorsi
+ caricaPercorso(in percorso: Percorso): boolean
+ eliminaPercorso(in idPercorso: integer): boolean
«ServiceInterface»
GestionePercorsiService

Interfaccia **AccessoCatalogo**

Distribuisce il catalogo delle visite virtuali disponibili

OPERAZIONI

- *accediCatalogo* - richiede l'invio del catalogo

  Parametri Input

  Nessuno

  Parametri Output

  Insieme di oggetti di tipo **Percorso**

- *accediCatalogoMuseo* - richiede l'invio del catalogo con riferimento ad uno specifico museo

  Parametri Input

  idMuseo - valore che identifica il museo i cui percorsi sono richiesti

  Parametri Output

  Insieme di oggetti di tipo **Percorso**

Interfaccia **AccessoContenuto**

Distribuisce i contenuti

OPERAZIONI

- *accediContenuto* - richiede l'invio di un contenuto. Le risorse associate ad esso (uno o più file) sono richieste in seguito con l'operazione *accediRisorsa*

Parametri Input

idContenuto - valore che identifica il contenuto richiesto

Parametri Output

Oggetto di tipo **Contenuto**

- *accediRisorsa* - richiede l'invio di una risorsa. L'oggetto così ottenuto contiene solo informazioni descrittive (quali nome del file risorsa). Per accedere al file risorsa è necessario invocare l'operazione accediDatiRisorsa

Parametri Input

idRisorsa - valore che identifica la risorsa richiesta

Parametri Output

Oggetto di tipo **Risorsa**

- *elencoContenuti* - richiede l'invio dei contenuti appartenenti ad uno specifico percorso

Parametri Input

idPercorso - valore che identifica il percorso i cui contenuti sono richiesti

Parametri Output

Insieme di oggetti di tipo **Contenuto**

- *elencoRisorse* - richiede l'invio delle risorse appartenenti ad uno specifico contenuto

Parametri Input

idContenuto - valore che identifica il contenuto le cui risorse sono richieste

Parametri Output

Insieme di oggetti di tipo **Risorsa**

- *accediDatiRisorsa* - richiede l'invio del file associato ad una risorsa

Parametri Input

idRisorsa - valore che identifica la risorsa richiesta

Parametri Output

Il file associato alla risorsa come flusso di byte

Interfaccia **GestionePercorsi**

Gestisce le operazioni di caricamento dei percorsi da parte dei fornitori

OPERAZIONI

- *caricaPercorso* - permette il caricamento di un nuovo percorso

Parametri Input

percorso - oggetto di tipo **Percorso** che definisce le informazioni relative al percorso da creare

Parametri Output

Un valore booleano che indica l'esito dell'operazione

- eliminaPercorso - elimina un percorso esistente e tutti i contenuti e risorse ad esso associati

  Parametri Input

  idPercorso - valore che identifica il percorso da eliminare

  Parametri Output

  Un valore booleano che indica l'esito dell'operazione

Interfaccia **GestioneContenuti**

Gestisce le operazioni di caricamento dei contenuti da parte dei fornitori

OPERAZIONI

- *caricaContenuto* - permette il caricamento di un nuovo percorso

  Parametri Input

  contenuto - oggetto di tipo Contenuto che definisce le informazioni relative al contenuto da creare

  Parametri Output

  Un valore booleano che indica l'esito dell'operazione

- caricaRisorsa - permette il caricamento di una nuova risorsa

  Parametri Input

  risorsa - oggetto di tipo Risorsa che definisce le informazioni relative alla risorsa da caricare

  dati - il file associato alla risorsa come flusso di byte

Parametri Output

Un valore booleano che indica l'esito dell'operazione

- *eliminaContenuto* - elimina un contenuto esistente e tutte le risorse ad esso associate

Parametri Input

idContenuto - valore che identifica il contenuto da eliminare

Parametri Output

Un valore booleano che indica l'esito dell'operazione

- *eliminaContenutiPercorso* - elimina tutti i contenuti appartenenti ad uno specifico percorso e le relative risorse

Parametri Input

idPercorso - valore che identifica il percorso i cui contenuti devono essere eliminati

Parametri Output

Un valore booleano che indica l'esito dell'operazione

- *eliminaRisorsa* - elimina una risorsa

Parametri Input

idRisorsa - valore che identifica la risorsa da eliminare

Parametri Output

Un valore booleano che indica l'esito dell'operazione

Interfaccia **Streaming**

Servizio di video streaming

OPERAZIONI

- *play* - inizia la riproduzione di una risorsa video

  Parametri Input

  idRisorsa - valore che identifica la risorsa video

  Parametri Output

  Nessuno

- *stop* - ferma la riproduzione di una risorsa video. Inoltre viene eseguita un'operazione di riposizionamento che "riavvolge" il video

  Parametri Input

  idRisorsa - valore che identifica la risorsa video

  Parametri Output

  Nessuno

- *pausa* - ferma temporaneamente la riproduzione di una risorsa video

  Parametri Input

  idRisorsa - valore che identifica la risorsa video

Parametri Output

Nessuno

- *riposiziona* - modifica l'istante di riproduzione di una risorsa video

Parametri Input

idRisorsa - valore che identifica la risorsa video

tempo - valore che indica l'istante di riproduzione da impostare

Parametri Output

Nessuno

Interfaccia **Pagamento**

Gestisce le funzionalità relative ai pagamenti da parte degli utenti

OPERAZIONI

- *verificaPagamento* - verifica l'effettuabilità di un pagamento

Parametri input

Oggetto di tipo **InfoPagamento**

Parametri output

Un valore booleano che indica l'effettuabilità dell'operazione di pagamento

- *eseguiTransazione* - effettua un pagamento

Parametri input

Oggetto di tipo **InfoPagamento**

Parametri output

Un valore booleano che indica l'esito dell'operazione

4.1.2 Software Components

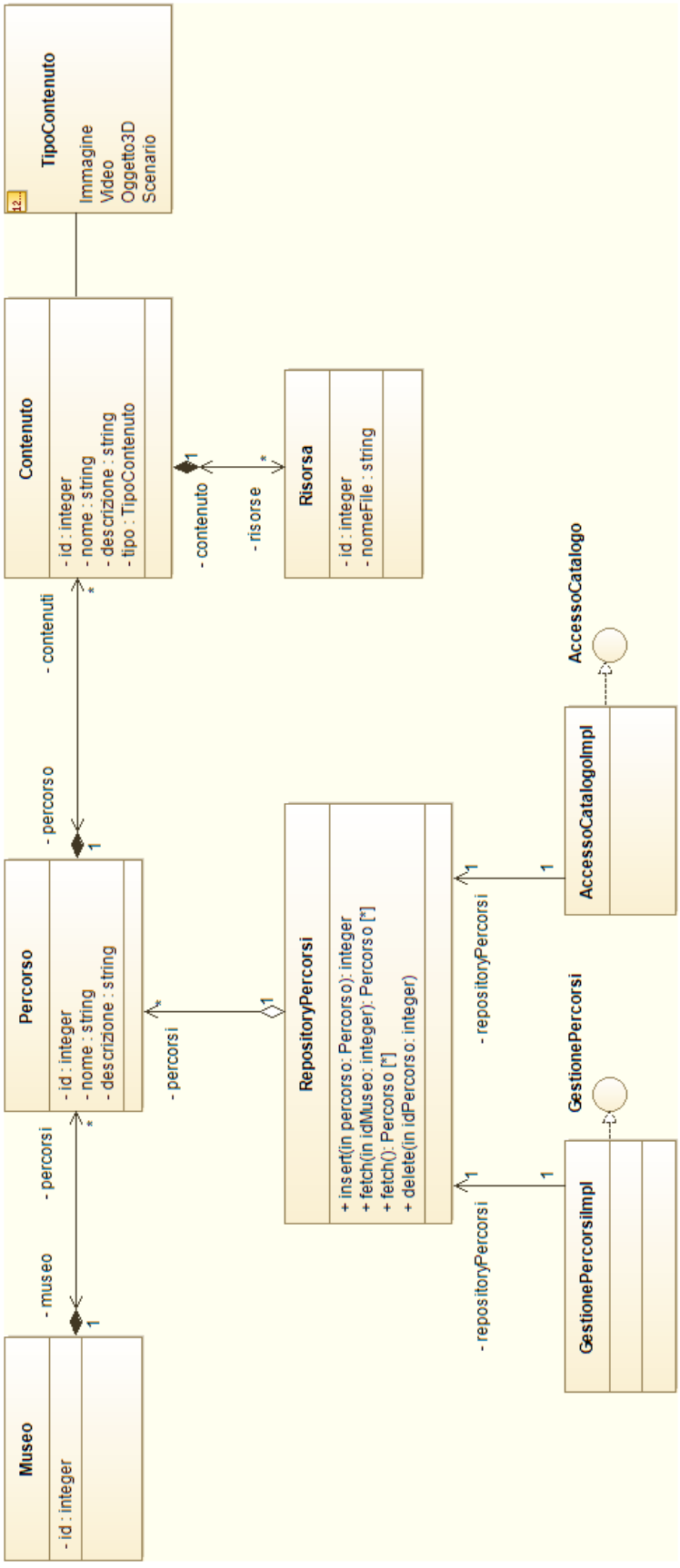
In questo paragrafo sono definiti i Software Components che implementano i servizi (di cui già sono note le Service Interfaces) visti in precedenza.

Sono stati individuati quattro componenti:

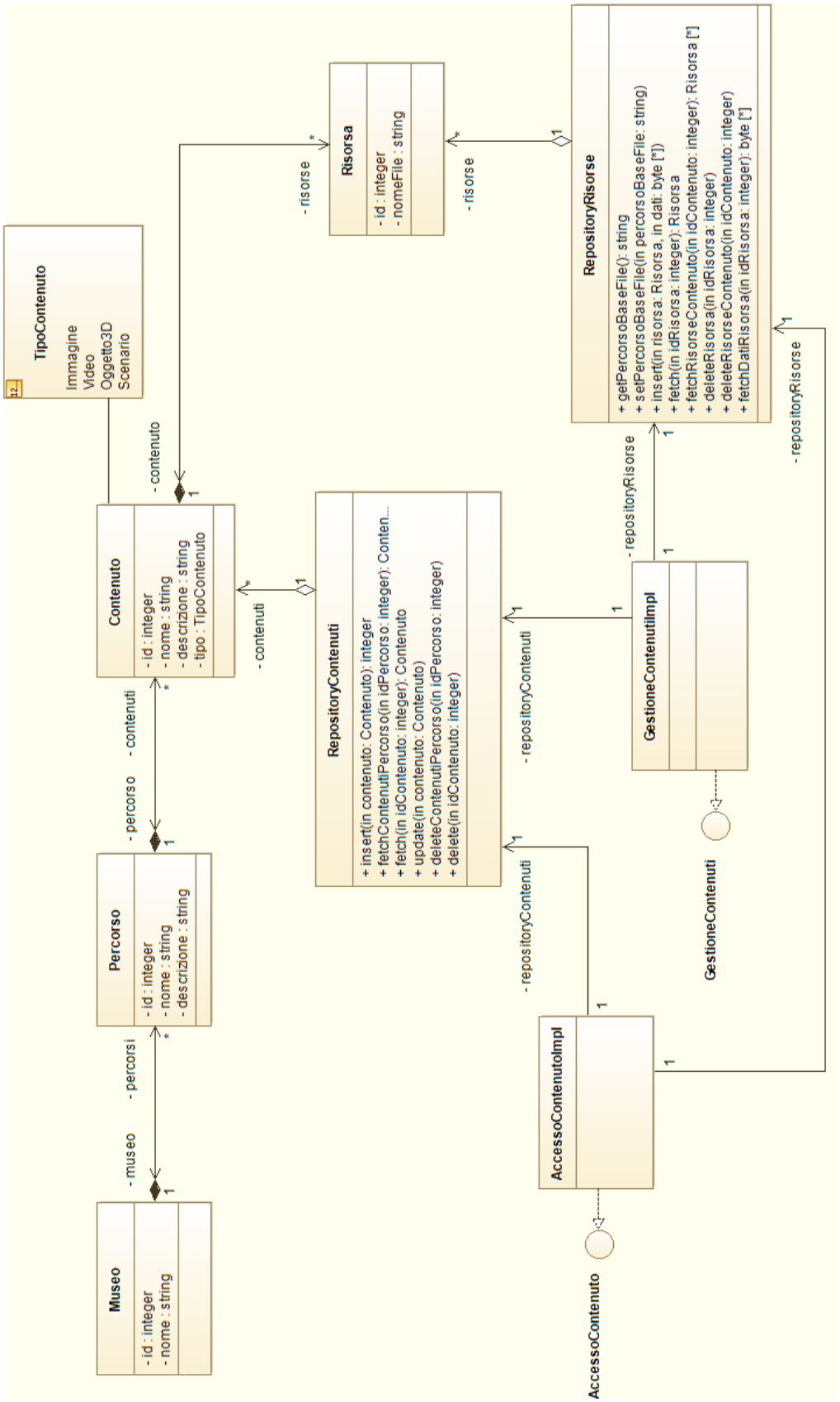
- **PercorsiComponent** - realizza le interfacce *AccessoCatalogo* e *GestionePercorsi*
- **ContenutiComponent** - realizza le interfacce *AccessoContenuto* e *GestioneContenuti*
- **PagamentoComponent** - realizza l'interfaccia *Pagamento*
- **StreamingComponent** - realizza l'interfaccia *Streaming*

Di seguito sono riportati i Class Diagram che descrivono gli oggetti utilizzati nella realizzazione dei componnti.

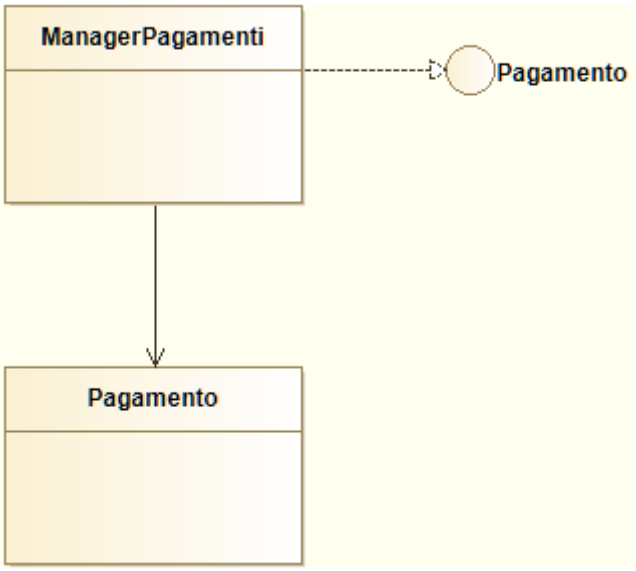
4.1.2.1 PercorsiComponent



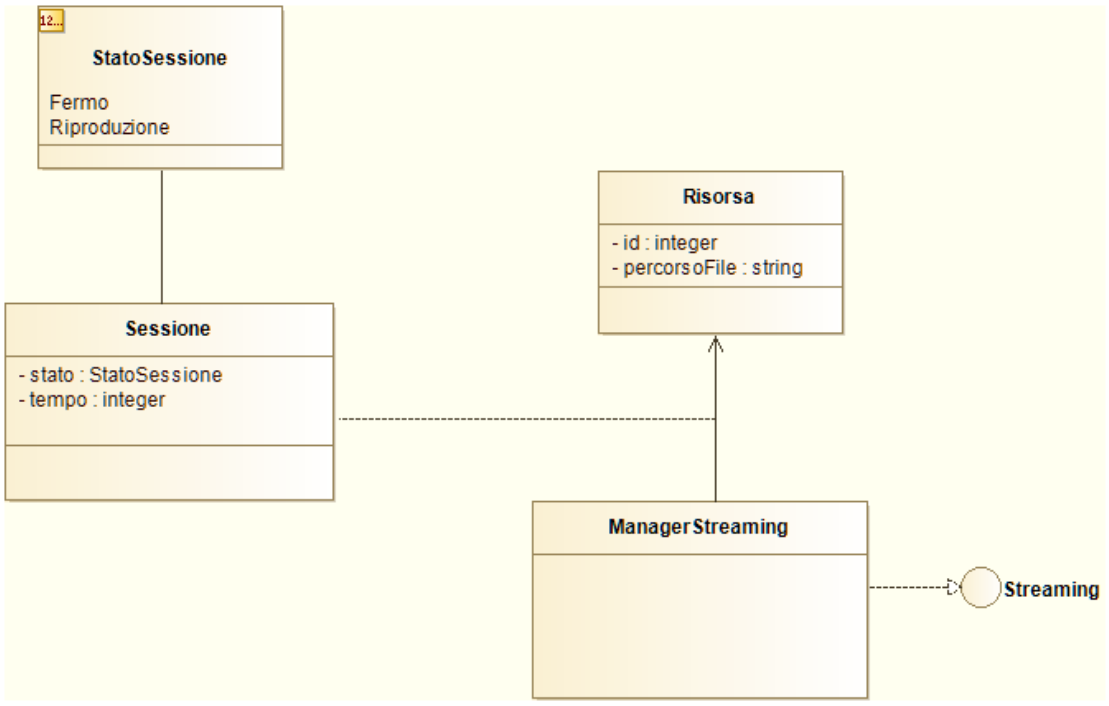
4.1.2.2 ContenutiComponent



4.1.2.3 PagamentoComponent

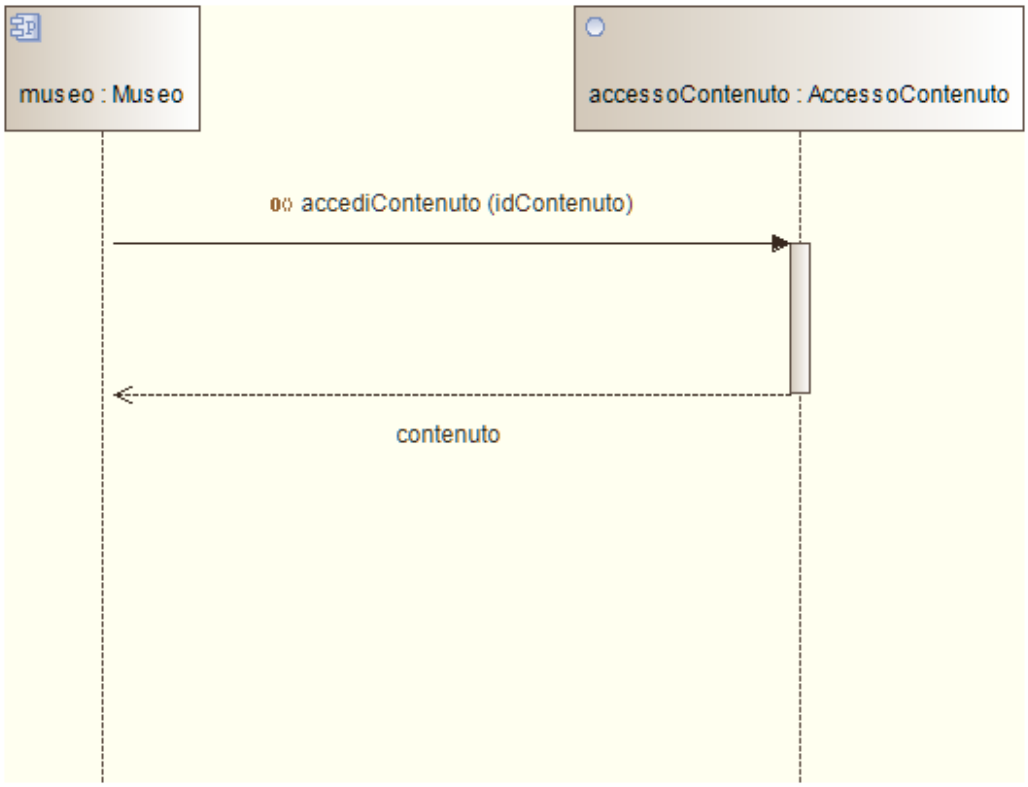


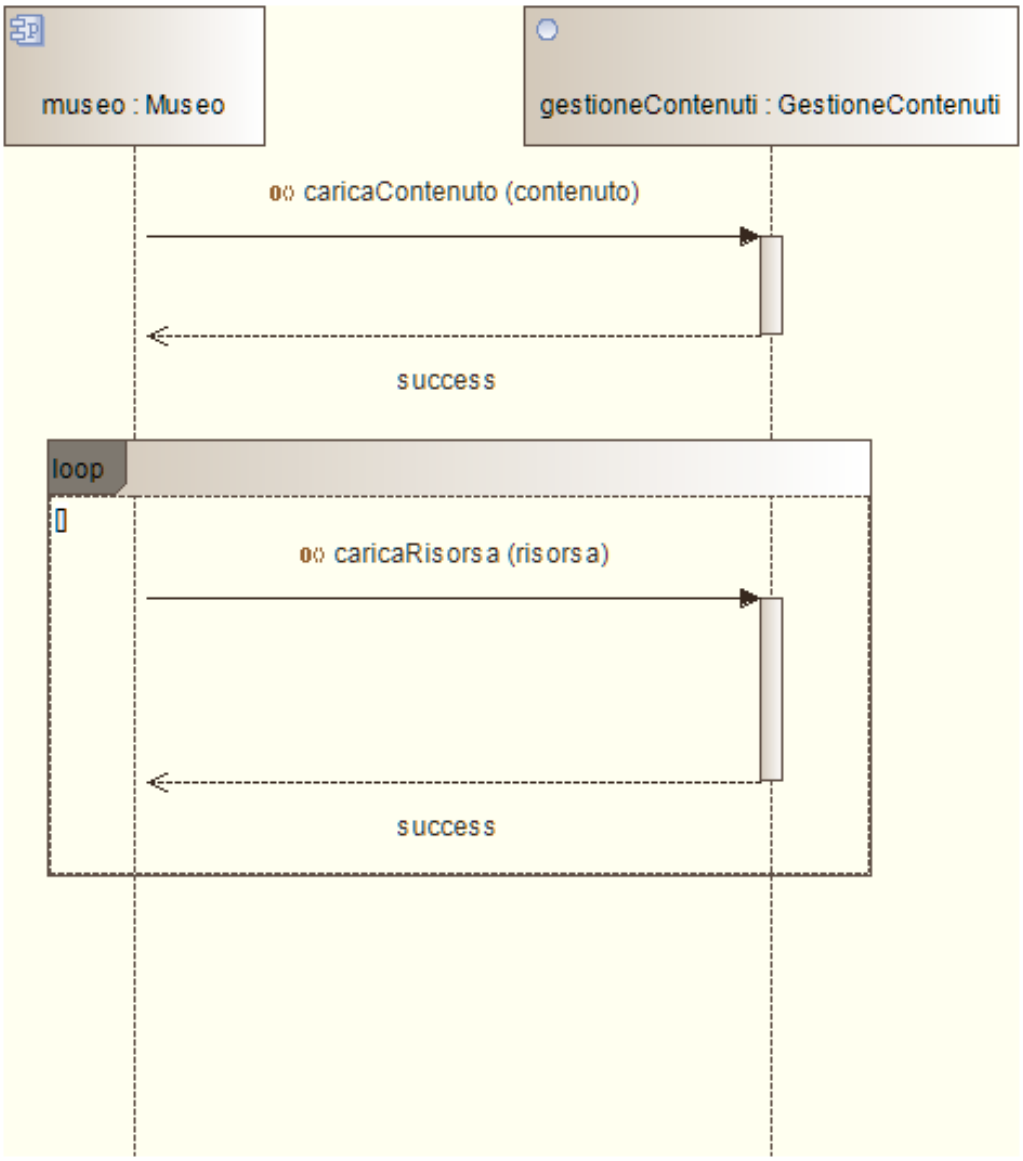
3.1.2.4 StreamingComponent

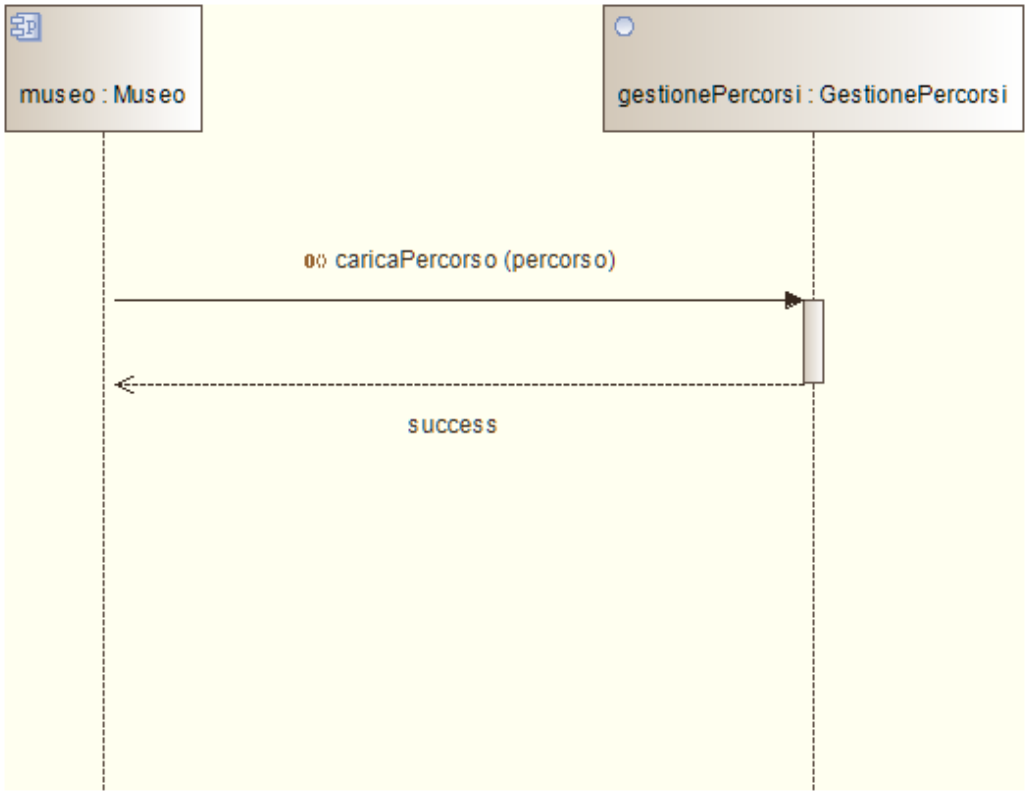


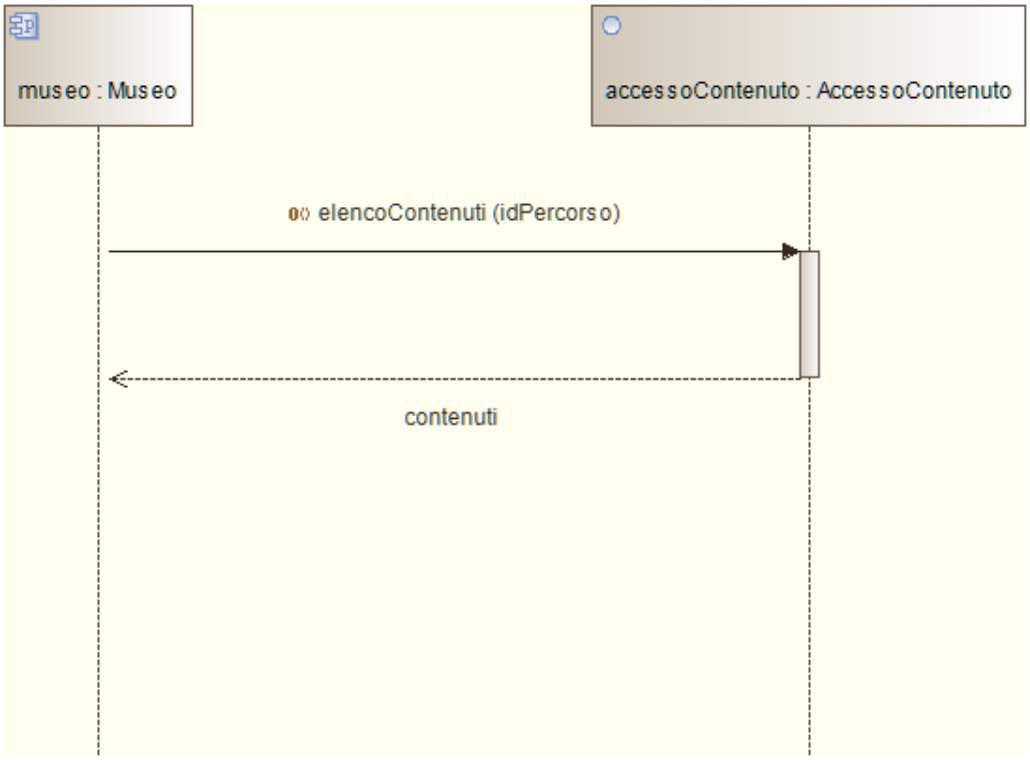
4.2 Services Architectures

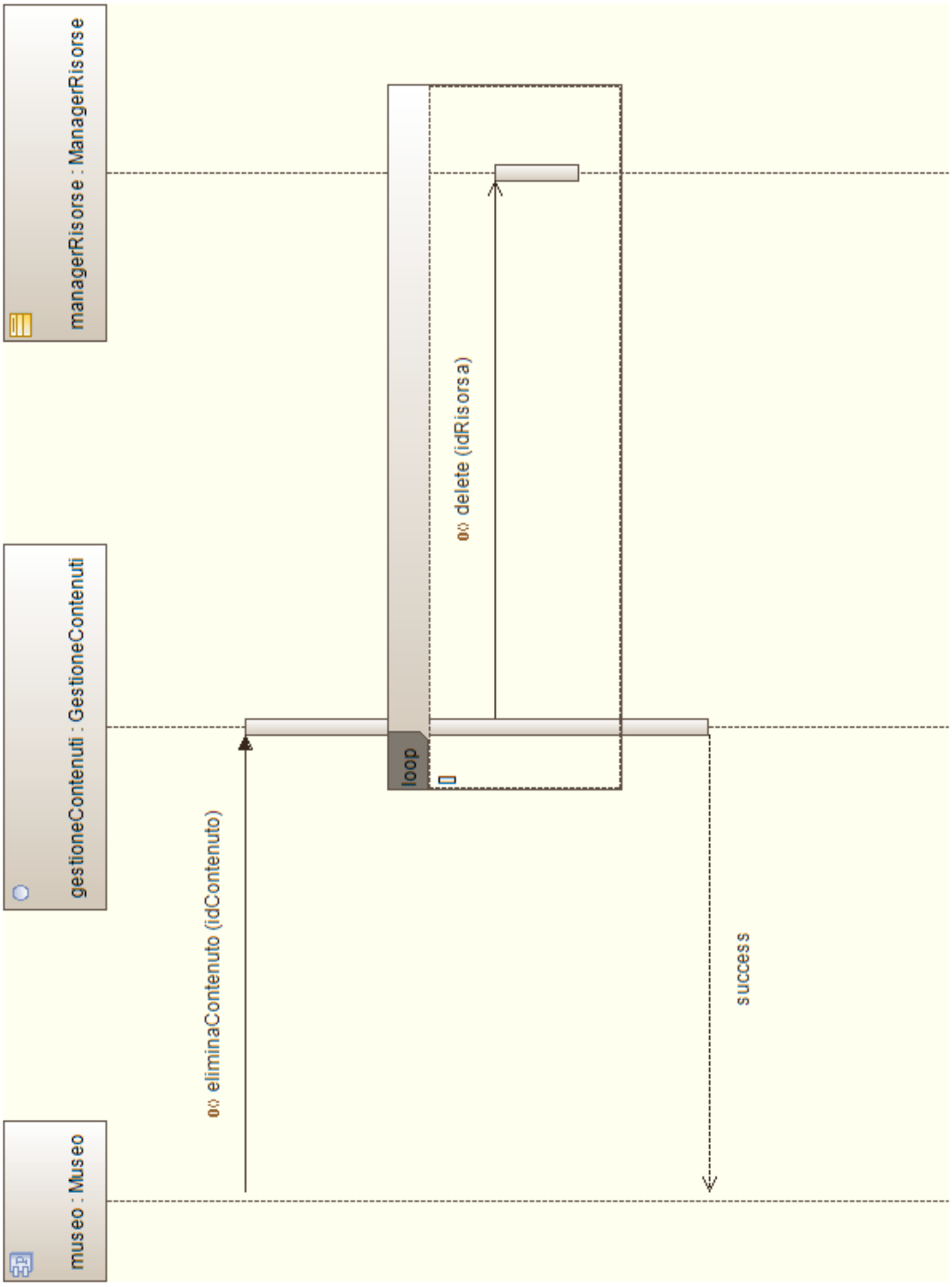
4.2.1 Service Choreography

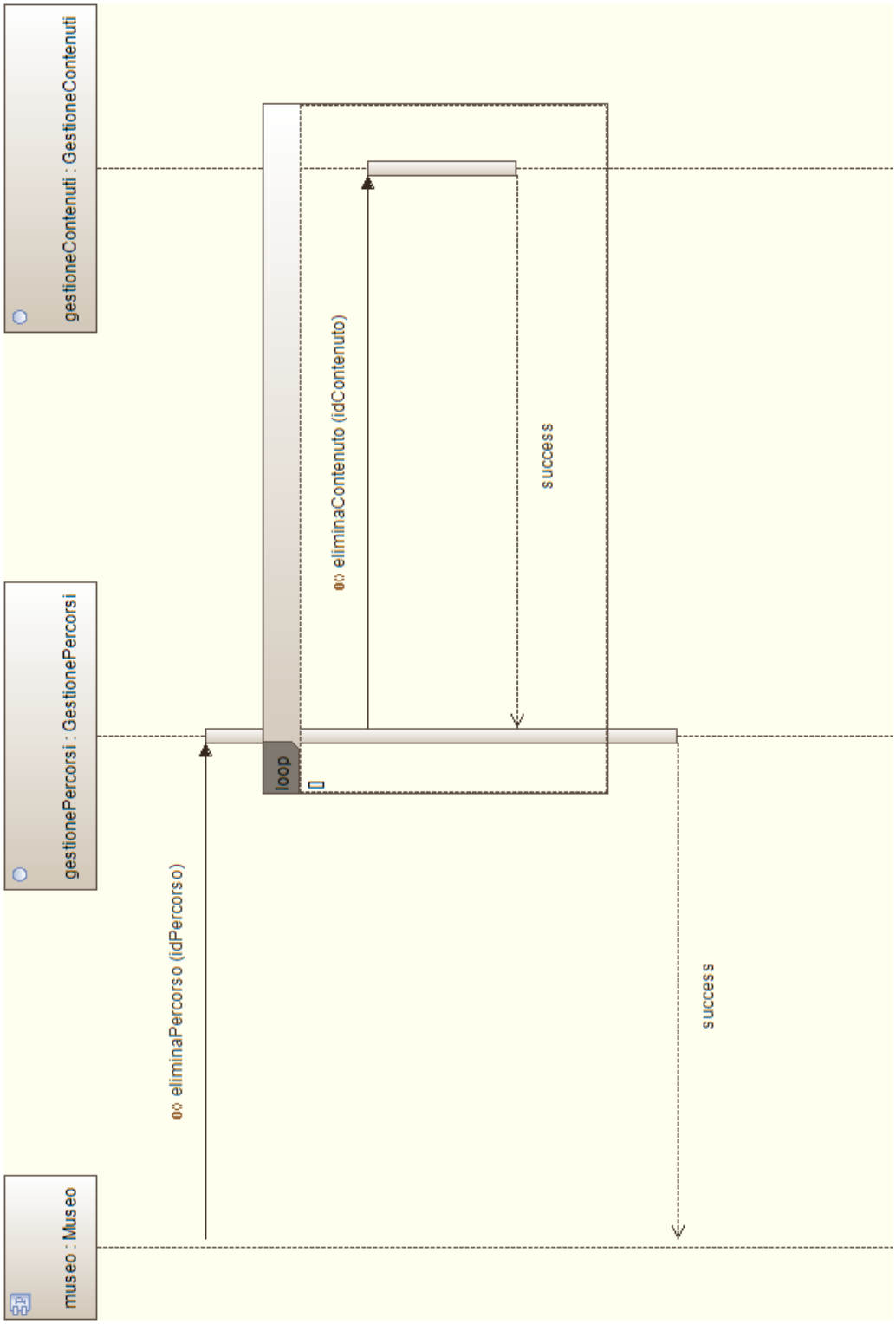


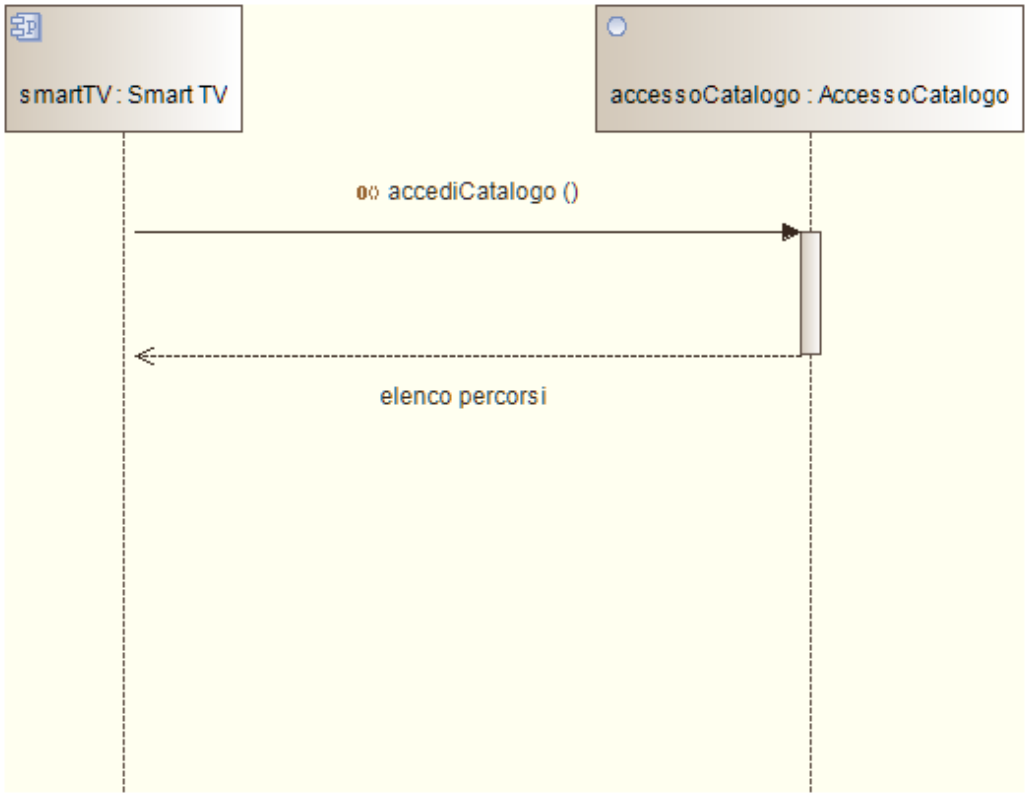

museo : Museo
gestioneContenuti : GestioneContenuti
caricaContenuto (contenuto)
success
loop
caricaRisorsa (risorsa)
success

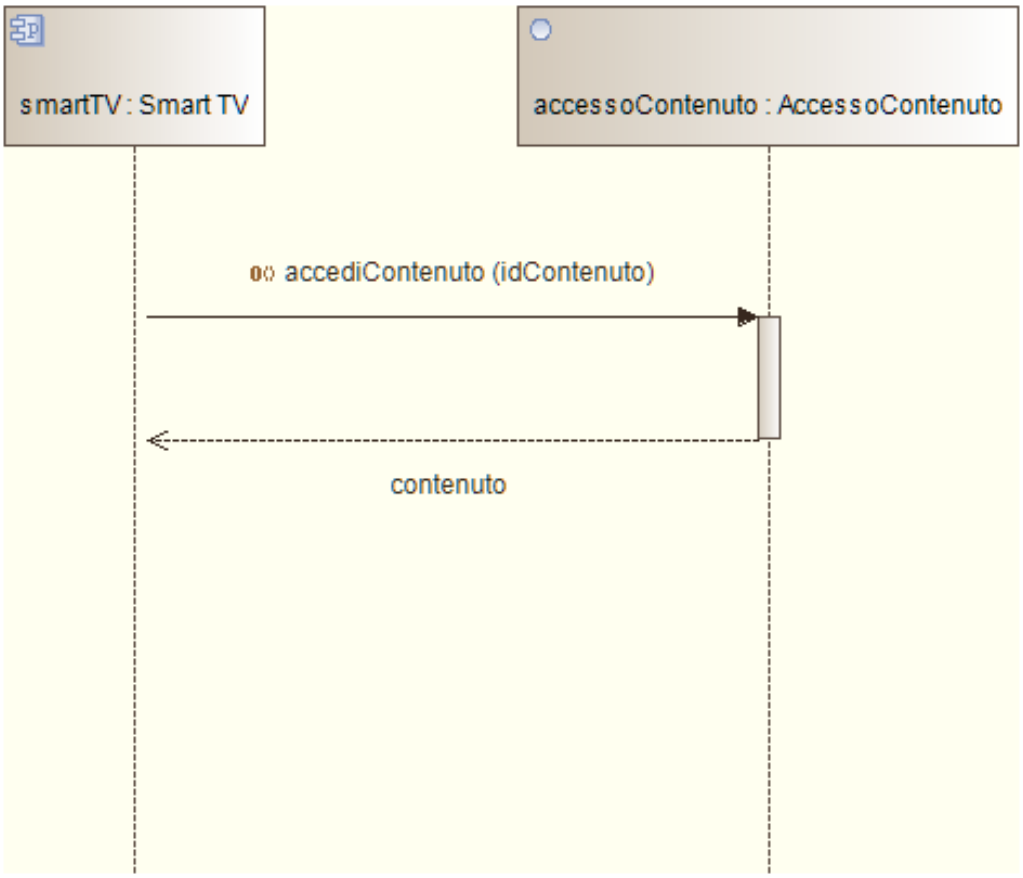

museo : Museo
gestionePercorsi : GestionePercorsi
caricaPercorso (percorso)
success

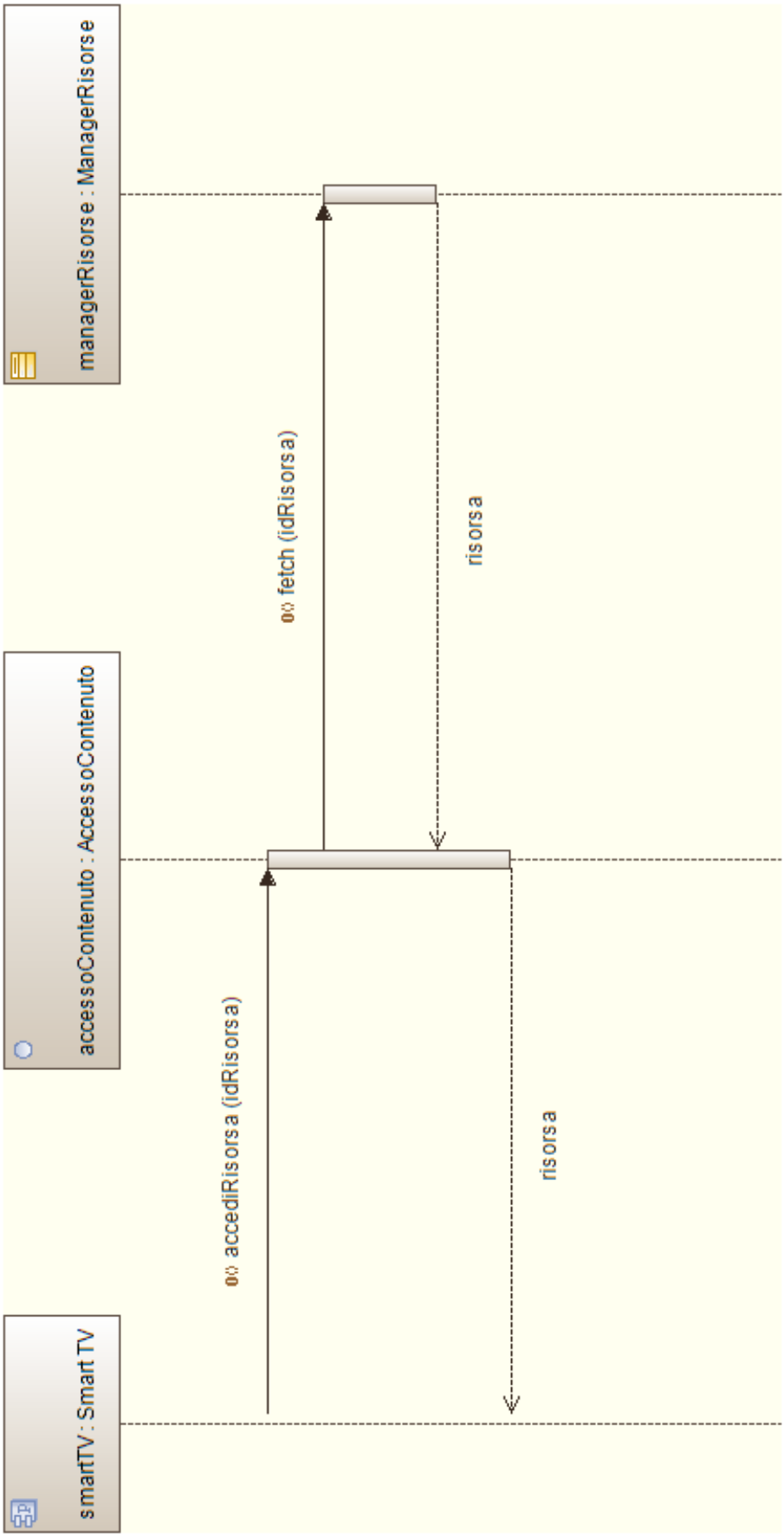

museo : Museo
accessoContenuto : AccessoContenuto
elencoContenuti (idPercorso)
contenuti

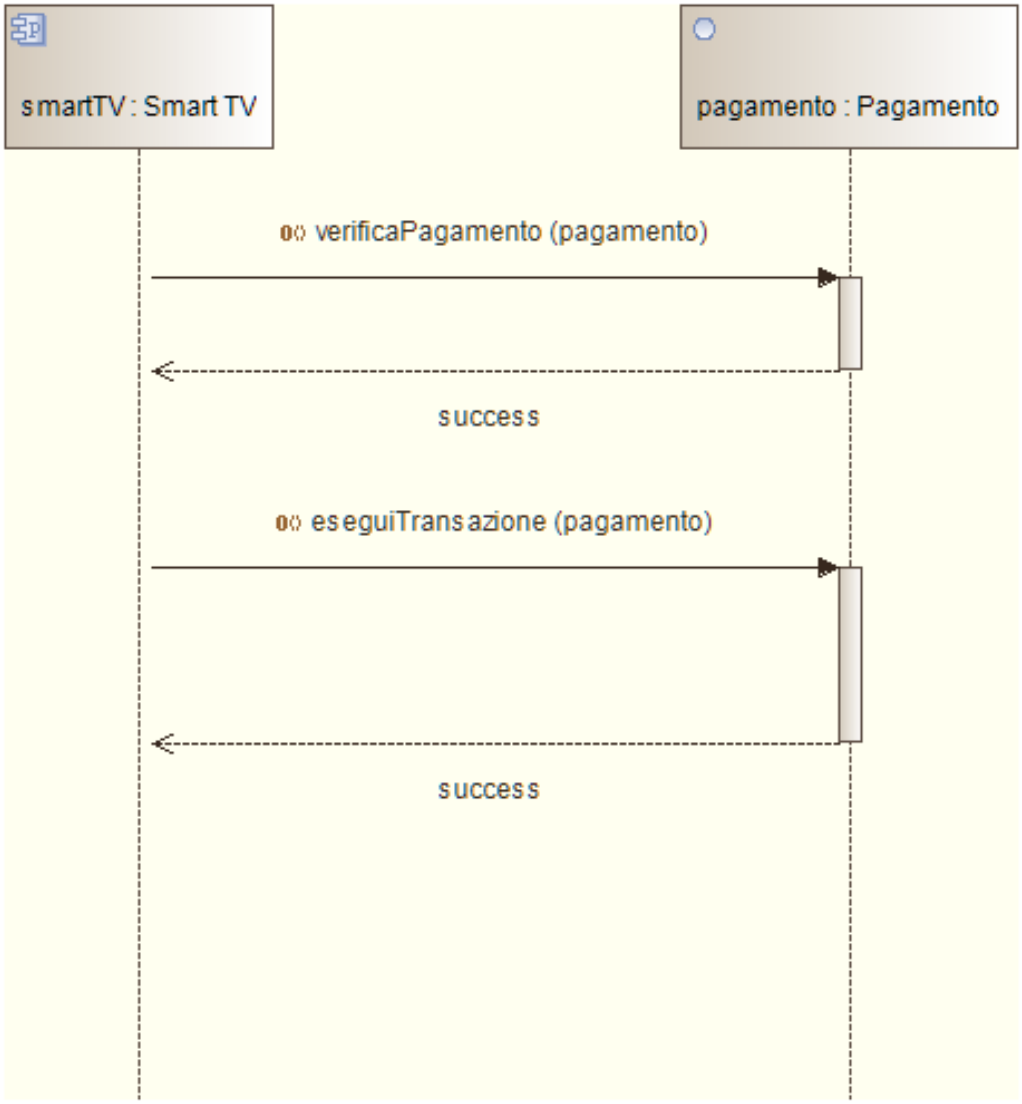

managerRisorse : ManagerRisorse
gestioneContenuti : GestioneContenuti
museo : Museo
loop
delete (idRisorsa)
eliminaContenuto (idContenuto)
success

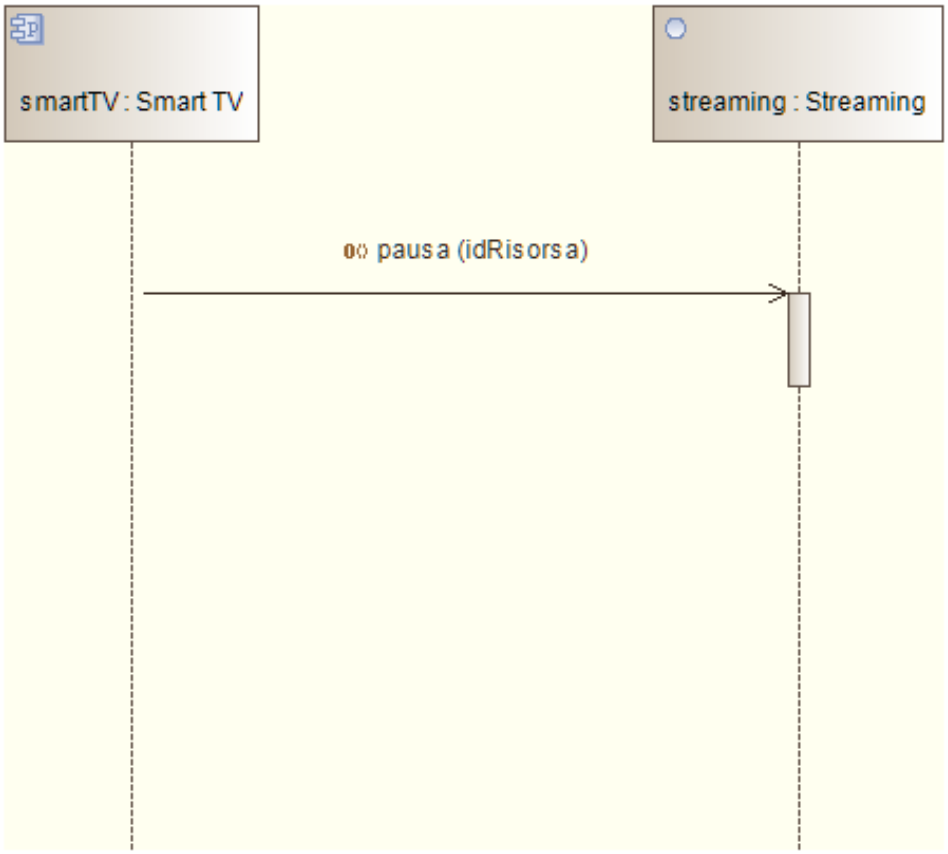

gestioneContenuti : GestioneContenuti
gestionePercorsi : GestionePercorsi
museo : Museo
eliminaContenuto (idContenuto)
success
loop
eliminaPercorso (idPercorso)
success

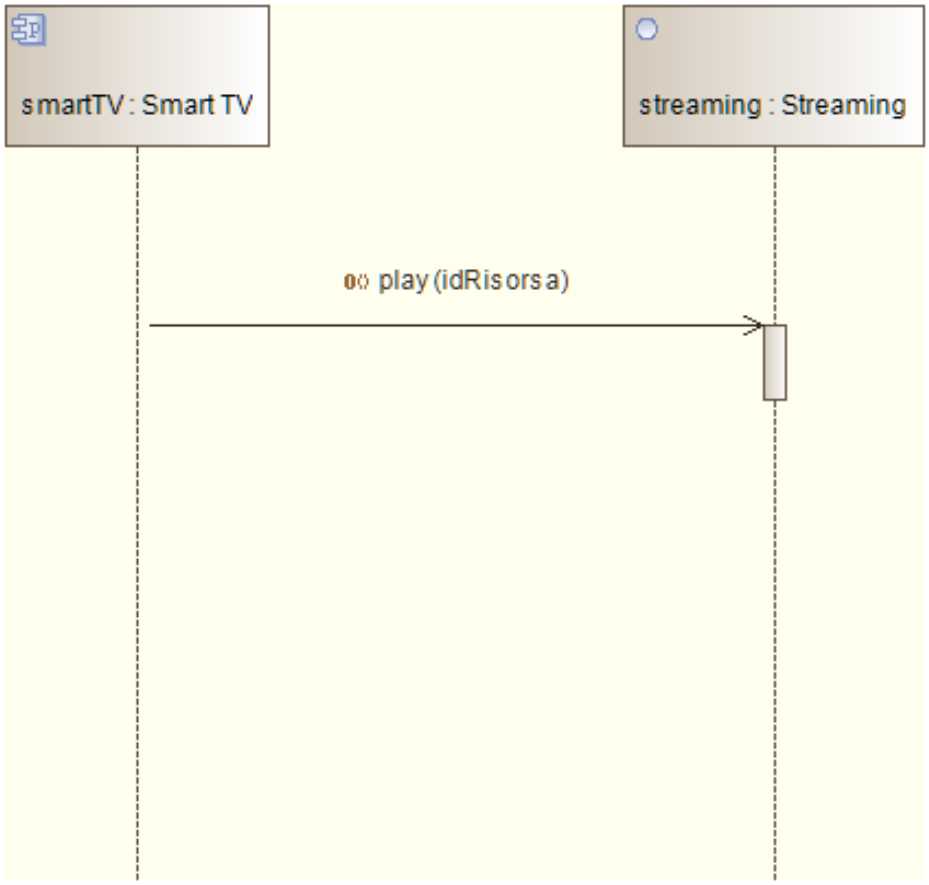

smartTV : Smart TV
accessoCatalogo : AccessoCatalogo
accediCatalogo ()
elenco percorsi

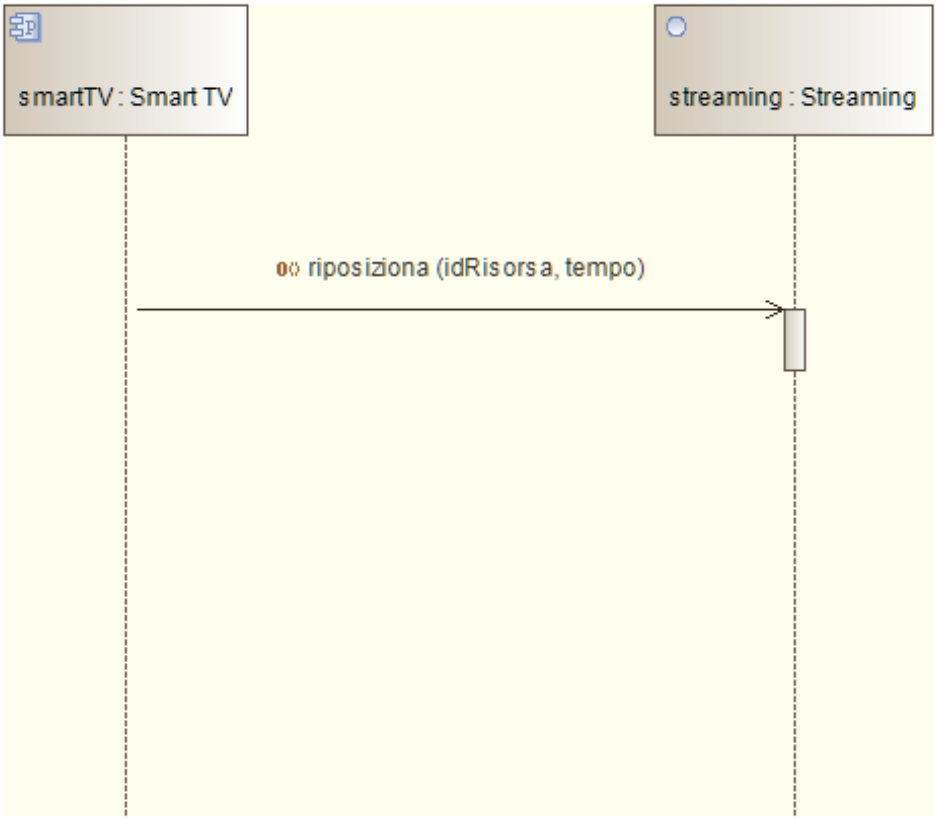

smartTV : Smart TV
accessoContenuto : AccessoContenuto
accediContenuto (idContenuto)
contenuto

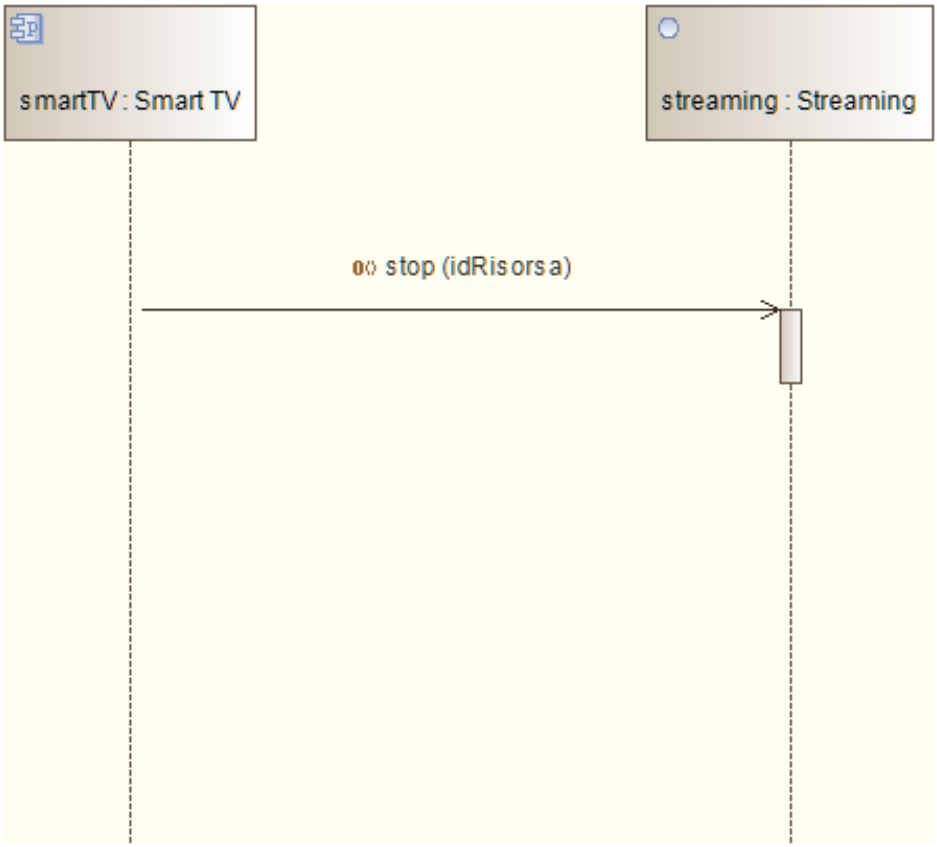

managerRisorse : ManagerRisorse
accessoContenuto : AccessoContenuto
smartTV : Smart TV
fetch (idRisorsa)
risorsa
accediRisorsa (idRisorsa)
risorsa


smartTV : Smart TV
pagamento : Pagamento
verificaPagamento (pagamento)
success
eseguiTransazione (pagamento)
success


smartTV : Smart TV
streaming : Streaming
pausa (idRisorsa)


smartTV : Smart TV
streaming : Streaming
play (idRisorsa)


smartTV : Smart TV
streaming : Streaming
riposiziona (idRisorsa, tempo)

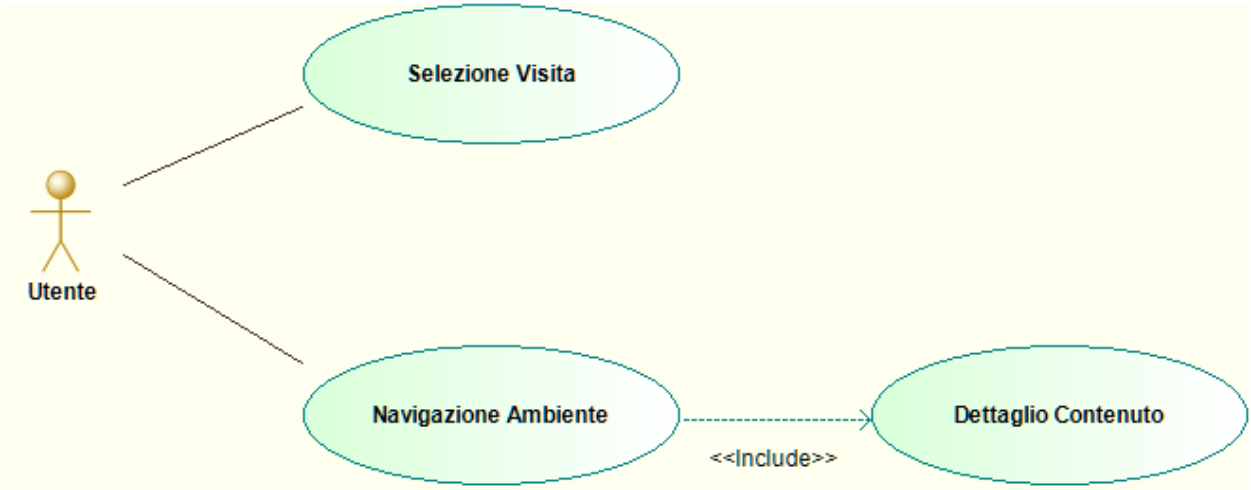

smartTV : Smart TV
streaming : Streaming
stop (idRisorsa)

Capitolo 5: Applicazione Client

5.1 Analisi

5.1.1 Casi d'Uso

I **Casi d'Uso** descrivono le interazioni dell'utente con l'applicazione.



- **Selezione Visita**

  L'utente visualizza un menù (una lista) contenente le visite fruibili. Viene utilizzato il telecomando del televisore per l'interazione con la schermata.

  Dettaglio Interazioni

  1. Tasto "Freccia Su" o tasto "Freccia Giù" per cambiare la selezione
  2. Tasto "OK" per iniziare la visita

- **Navigazione Ambiente**

  L'utente ha scelto una visita e adesso naviga (si sposta) all'interno di un ambiente virtuale entro il quale sono visibili i vari contenuti (opere d'arte quali tele e sculture). I comandi vengono impartiti tramite gesti.

  Dettaglio Interazioni

  1. Con il pugno chiuso (e fermo) l'utente si sposta in avanti
  2. Muovendo il pugno chiuso è possibile guardarsi attorno (cambiando dunque la direzione del movimento)
  3. Quando si è nei pressi di un'opera (un contenuto) sullo schermo vengono mostrati alcuni dettagli (titolo e descrizione)
  4. Con un gesto di tipo "Like" (pollice in su) è possibile aprire una nuova schermata con i dettagli del contenuto (Caso d'Uso III)

- **Dettaglio Contenuto**

  L'utente visualizza un singolo contenuto con la possibilità di effettuare operazioni come ingrandimento, scorrimento e rotazione. Le operazioni disponibili dipendono dal tipo di contenuto. Come nel Caso d'Uso II (Visita) tutti i comandi vengo impartiti con dei gesti.

  Dettaglio Interazioni

  1. Con il movimento del pugno chiuso è possibile spostare il contenuto
  2. Allargando le mani aperte ci si avvicina al contenuto per visualizzarne i dettagli (operazione di "zoom-in")
  3. Avvicinando le mani aperte ci si allontana dal contenuto (operazione di "zoom-out")
  4. Ruotando le due mani si gira attorno al contenuto

5.1.2 Stati dell'Applicazione

L'applicazione può essere modellata come un Automa a Stati Finiti (FSM).

- **Stato Caricamento**

L'applicazione si trova in questo stato durante un'operazione di caricamento.

- **Stato Errore Gesture Recognition**

  Il modello di Smart TV utilizzato non supporta le funzionalità di Gesture Recognition.

- **Stato Selezione Visita**

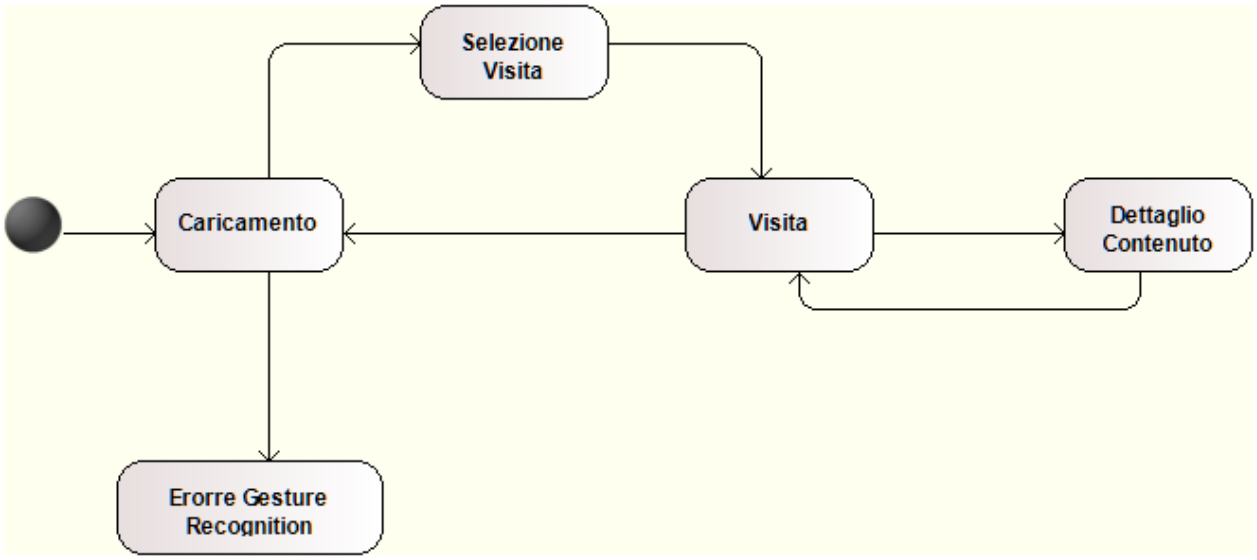
  In questo stato viene visualizzata la lista delle visite disponibili e l'utente può selezionarne una (vedi anche Caso d'Uso I).

- **Stato Visita**

  Stato che rappresenta la navigazione all'interno dell'ambiente virtuale (vedi anche Caso d'Uso II).

- **Stato Dettaglio Contenuto**

  L'utente ha scelto di visualizzare un singolo contenuto nel dettaglio. L'utente può effettuare varie operazioni che gli permettono di osservare l'oggetto da vari punti di vista (vedi anche Caso d'Uso III).



5.2 Progettazione

5.2.1 Architettura

Il modello architetturale utilizzato è un adattamento del paradigma Model-View-Controller (MVC).

Come visto l'applicazione è strutturata come una FSM. Gli stati prendono il nome di "Scene", dal punto di vista del pattern MVC le scene fungono da Controller.

Ad ogni scena è associata una vista (View) costituita da un documento html e da un foglio di stile css. Il layer View si complica nel caso di scene con rendering 3D (stati "Visita" e "Dettaglio Contenuto") nelle quali sono adoperati diversi oggetti del package View.

Il modello (Model) è costituito da oggetti semplici e da un servizio che consente il caricamento, tramite Web Services SOAP, delle varie entità che formano il dominio applicativo (come percorsi e contenuti).

5.2.2 Descrizione dei Package

In questo paragrafo sono descritti i package nei quali sono organizzati gli oggetti. Per ogni package è fornita una breve descrizione di delle classi in esso definite. I dettagli relativi a tali oggetti sono riportati nei paragrafi successivi.

- **Package Core**

  Questo package costituisce il "cuore MVC" dell'applicazione. La classe **Scene** funge da base per gli oggetti che rappresentano le varie scene. Una scena altro non è che uno stato in cui può trovarsi l'applicazione, ad esempio il "Menù Principale", la "Schermata di Caricamento" o la "Visita Virtuale". Ogni scena ha un proprio **ciclo di vita** il cui responsabile è l'oggetto **SceneManager** che mantiene uno **stack** di scene. La scena in cima allo stack è detta **scena attiva** ed è quella che l'utente visualizza.

  Al ciclo di vita di una scena (schematizzato in figura) sono associate diverse operazioni. Sarà lo SceneManager ad orchestrare il tutto secondo il seguente schema:

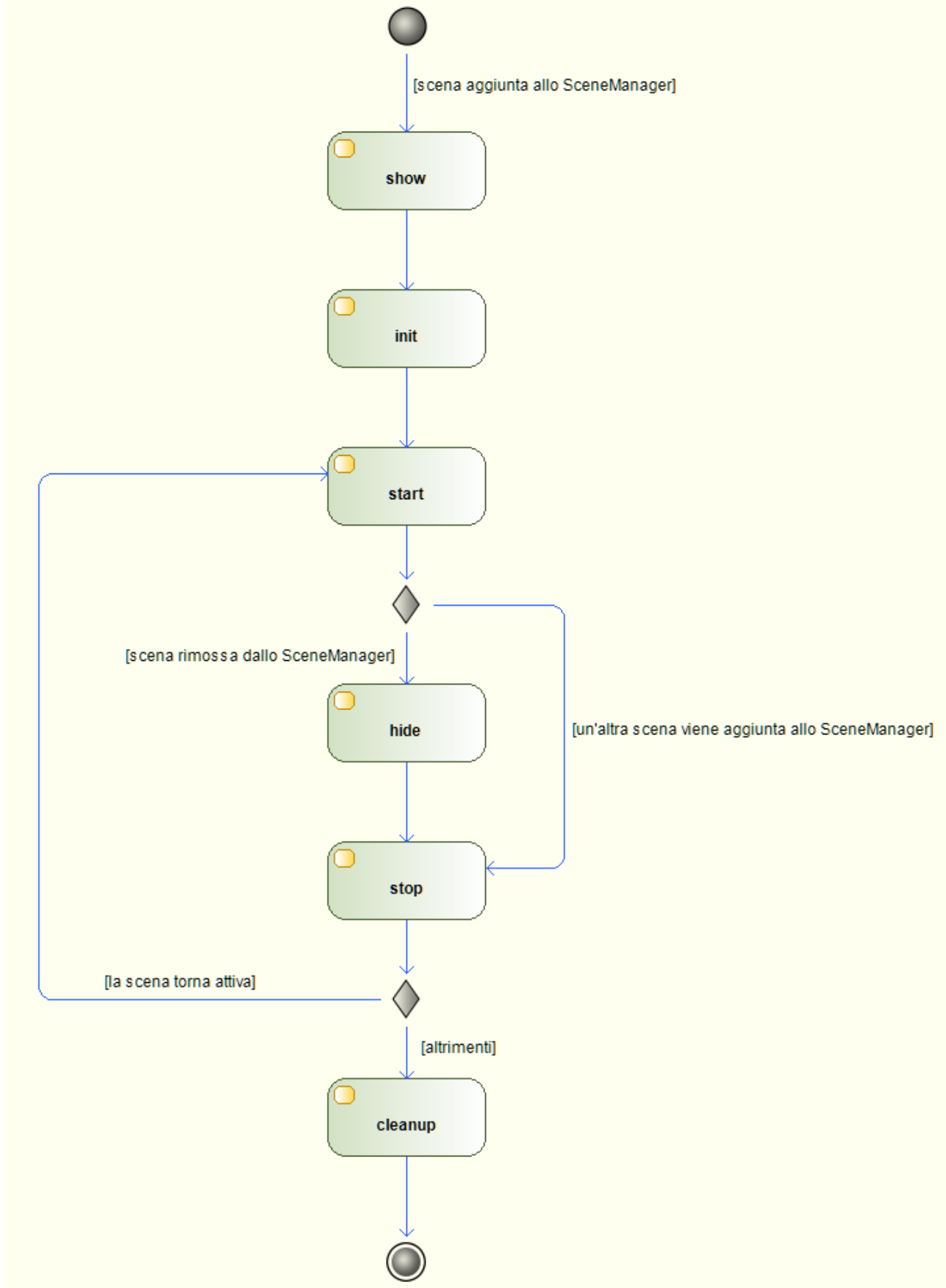
  - Operazione di aggiunta (push o replace):

1. Se esiste una scena nello stack ne viene invocato il metodo **stop**
2. Il metodo **show** della nuova scena viene eseguito. Questo metodo **crea la vista** iniettando il markup HTML nel documento[1]
3. Vengono richiamati i metodi **init** e **start**

- Operazione di rimozione (pop):
  1. Il metodo **hide** della scena rimossa viene eseguito. Questo metodo rimuove il markup HTML dal documento
  2. Vengono richiamati i metodi **stop** e **cleanup**
  3. Se esiste una scena nello stack ne viene invocato il metodo **start**

L'oggetto App funge da punto d'ingresso al framework eseguendo alcune operazioni di inizializzazione fondamentali e aggiungendo la prima scena allo SceneManager. La classe App definisce inoltre alcuni metodi "handler" per la gestione degli eventi di input (telecomando della Smart TV, mouse e gesti). In generale questi metodi delegano la propria responsabilità alla scena attiva.

[1] L'App viene eseguita all'interno di un Browser e dunque ciò che l'utente visualizza è un documento HTML


[scena aggiunta allo SceneManager]
show
init
start
[scena rimossa dallo SceneManager]
hide
[un'altra scena viene aggiunta allo SceneManager]
stop
[la scena torna attiva]
[altrimenti]
cleanup

- Package **Scenes**

  Questo package contiene le scene:

  - **MainMenu** la scena iniziale dove l'utente può scegliere la visita virtuale che preferisce.
  - **LoadingOverlay** una semplice schermata di caricamento con un messaggio.
  - **NoGestureRecognitionError** schermata di errore visualizzata in caso di mancato supporto delle funzionalità di Gesture Recognition.
  - **Tour** la vera e propria visita virtuale. L'utente può spostarsi all'interno di un ambiente 3D utilizzando le funzionalità di Gesture Recognition per impartire comandi all'applicazione.
  - **Contenuto** visualizzazione dei dettagli di un contenuto. In questa schermata l'utente visualizza un singolo contenuto e può manipolare la visuale utilizzando la Gesture Recognition.

- Package **Model**

  In questo package sono definiti oggetti che rappresentano le entità di business, in particolare:

  - **Percorso** rappresenta un Percorso, cioè un'insieme di Contenuti presenti all'interno di un Museo.
  - **Contenuto** rappresenta uno dei Contenuti all'interno di un Percorso. I Contenuti sono opere d'arte come quadri e sculture.
  - **Posizionamento** una struttura dati semplice che descrive il posizionamento di un Contenuto.
  - **Risorsa** una o più Risorse sono associate ad un Contenuto e ne costituiscono i dati veri e propri. Ad un Contenuto di tipo "IMMAGINE", ad esempio, è associato un file di tipo JPEG.

Esiste un servizio, l'oggetto **ModelService**, il cui scopo è la creazione di questi oggetti. Il servizio fornisce un'astrazione utile in quanto nasconde i dettagli relativi alla strategia di creazione/caricamento delle entità. Si può facilmente creare, ad esempio, un "mock" della classe ModelService per fornire oggetti fittizi utilizzabili nel caso in cui il servizio "reale" non sia disponibile. L'implementazione corrente accede ai Web Services della piattaforma MuSE @ Home per ottenere i dati necessari alla creazione degli oggetti richiesti.

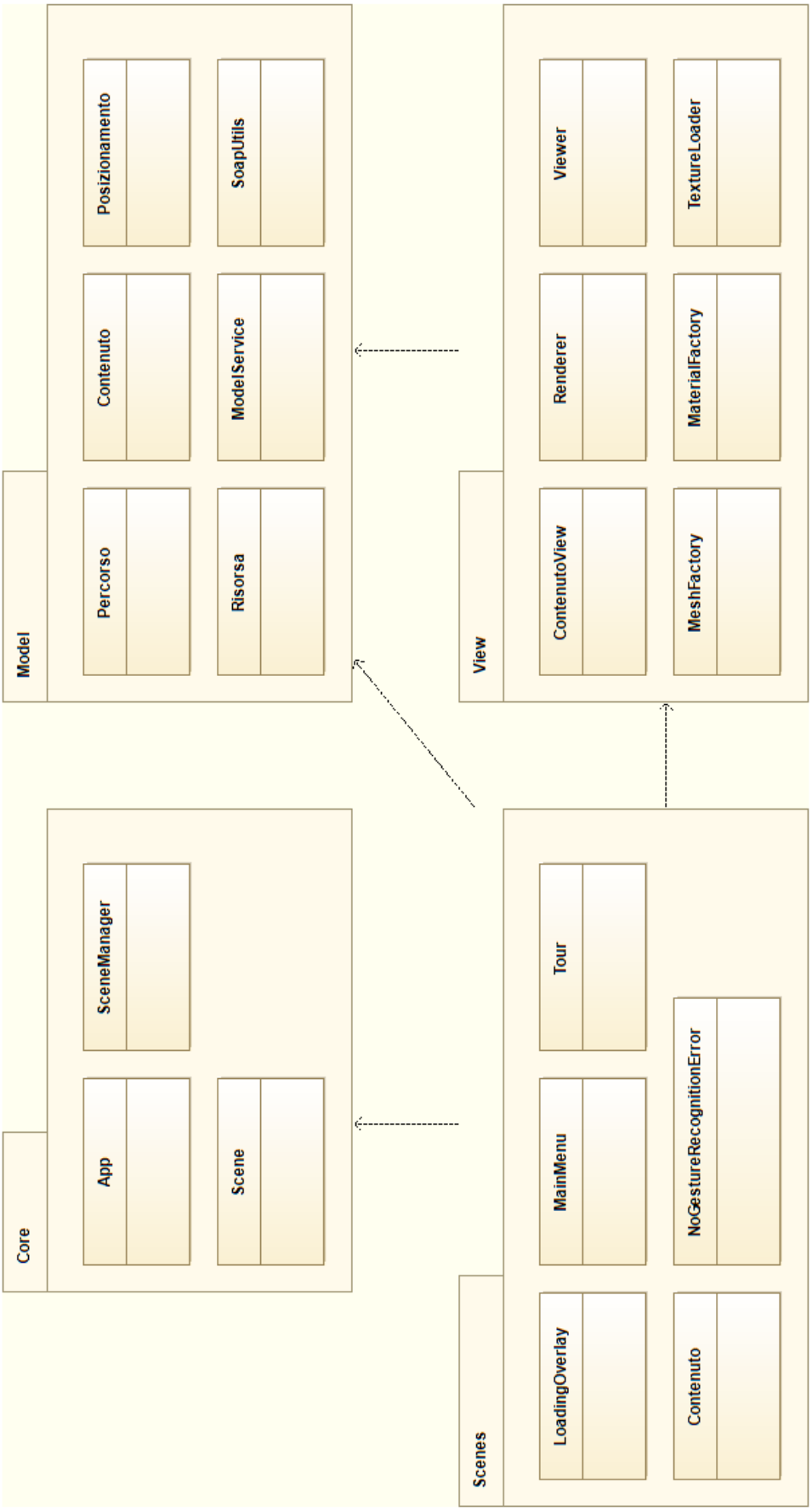
- Package **View**

  Contiene gli oggetti che realizzano il rendering delle scene 3D.

  L'elemento fondamentale è il **Renderer** che ha la responsabilità di effettuare il rendering dei Contenuti. La classe **ContenutoView** incapsula un Contenuto e ne costituisce la "vista", cioè un "modello 3D".

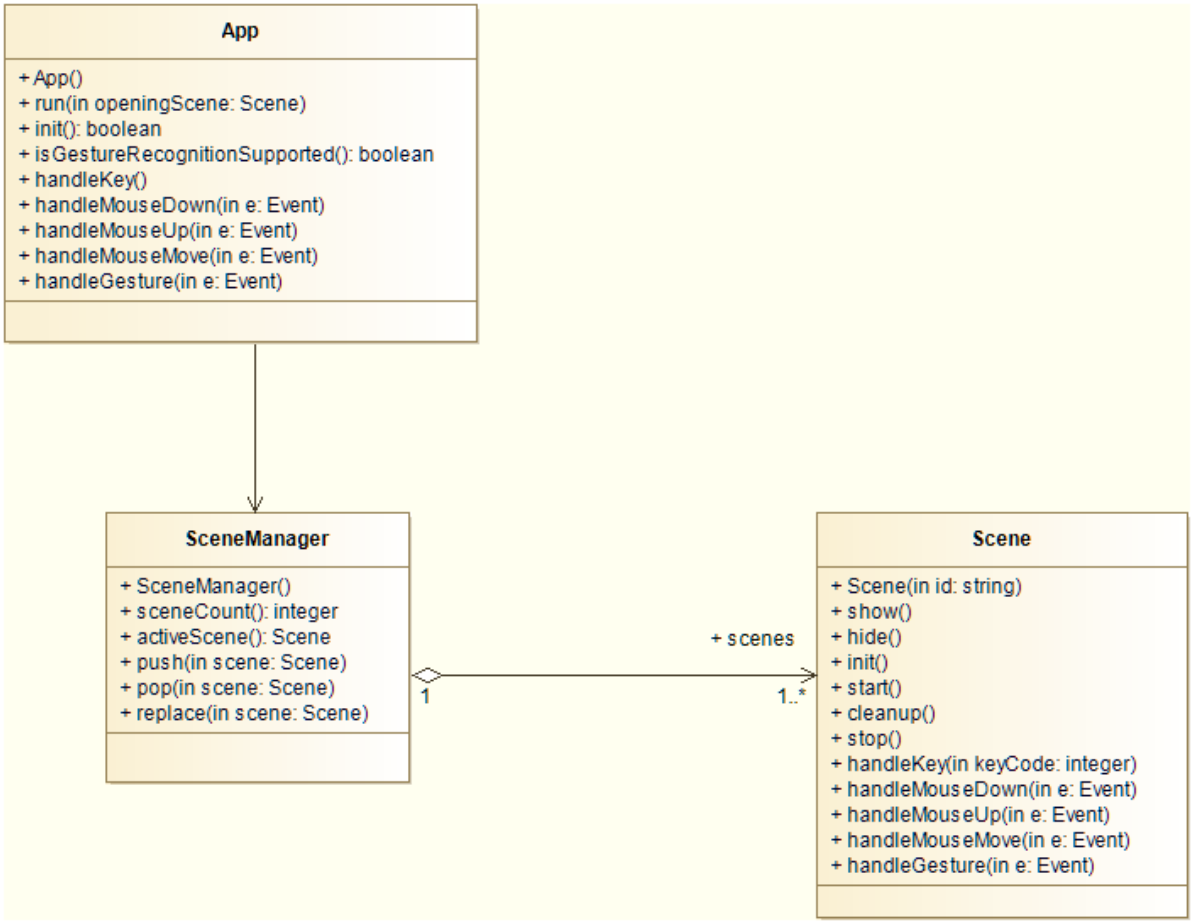
  L'oggetto Viewer rappresenta il visitatore e definisce metodi per "guardarsi attorno" e per spostarsi all'interno di un ambiente.

  Infine nel package sono presenti alcune classi "ausiliarie" per la creazione degli oggetti usati dalla libreria ThreeJs: MeshFactory, MaterialFactory e TextureLoader.


Model
Posizionamento
SoapUtils
Contenuto
ModelService
Percorso
Risorsa
View
Viewer
TextureLoader
Renderer
MaterialFactory
ContenutoView
MeshFactory
Core
SceneManager
App
Scene
Scenes
Tour
MainMenu
NoGestureRecognitionError
LoadingOverlay
Contenuto

5.2.3 Descrizione Dettagliata degli Oggetti

5.2.3.1 Package Core



- Classe **App**

OPERAZIONI

- run - inizializza ed esegue l'applicazione.

Parametri Input

openingScene - la scena iniziale

Parametri Output

Nessuno

- init - Inizializza l'applicazione

  Parametri Input

  Nessuno

  Parametri Output

  Un valore booleano che indica l'esito dell'operazione

- isGestureRecognitionSupported - Restituisce un valore che indica se il modello di Smart TV utilizzato per eseguire l'applicazione supporta la Gesture Recognition.

  Parametri Input

  Nessuno

  Parametri Output

  Un valore booleano che indica l'esito dell'operazione

- handleKey - Metodo invocato quando viene premuto un tasto del telecomando della Smart TV. Delega la gestione dell'evento alla scena attiva.

  Parametri Input

  e - oggetto contenente informazioni relative all'evento

  Parametri Output

  Nessuno

- handleMouseDown - Metodo invocato quando viene premuto il tasto sinistro del mouse o quando l'utente chiude il pugno. Delega la gestione dell'evento alla scena attiva.

Parametri Input

e - oggetto contenente informazioni relative all'evento

Parametri Output

Nessuno

- handleMouseUp - Metodo invocato quando viene rilasciato il tasto sinistro del mouse o quando l'utente apre il pugno. Delega la gestione dell'evento alla scena attiva.

Parametri Input

e - oggetto contenente informazioni relative all'evento

Parametri Output

Nessuno

- handleMouseMove - Metodo invocato quando viene mosso il puntatore del mouse o quando l'utente muove la mano. Delega la gestione dell'evento alla scena attiva.

Parametri Input

e - oggetto contenente informazioni relative all'evento

Parametri Output

Nessuno

- handleGesture - Metodo invocato quando il sistema di Gesture Recognition riconosce un gesto. Delega la gestione dell'evento alla scena attiva.

Parametri Input

e - oggetto contenente informazioni relative all'evento

Parametri Output

Nessuno

- Classe **SceneManager**

OPERAZIONI

- sceneCount - restituisce il numero di scene presenti nello stack

Parametri Input

Nessuno

Parametri Output

Il numero di scene presenti nello stack

- activeScene - restituisce la scena attiva (quella in cima allo stack).

Parametri Input

Nessuno

Parametri Output

La scena attiva (quella in cima allo stack)

- push - Aggiunge una scena in cima allo stack. La scena aggiunta diviene la nuova scena attiva.

Parametri Input

scene - la scena da aggiungere

Parametri Output

Nessuno

- pop - Rimuove la scena attiva dallo stack. La prossima scena nello stack diviene la nuova scena attiva.

Parametri Input

Nessuno

Parametri Output

Nessuno

- replace - rimuove tutte le scene dallo stack e aggiunge la scena indicata. La scena aggiunta diviene la nuova scena attiva.

Parametri Input

scene - la scena da aggiungere

Parametri Output

Nessuno

- Classe **Scene**

OPERAZIONI

- show - invocato quando la scena viene aggiunta allo SceneManager. Crea la scena nel documento.

Parametri Input

Nessuno

Parametri Output

Nessuno

- hide - invocato quando la scena viene rimossa dallo SceneManager. Elimina la scena dal documento.

Parametri Input

Nessuno

Parametri Output

Nessuno

- init - eseguito una sola volta quando la scena viene aggiunta allo SceneManager.

Parametri Input

Nessuno

Parametri Output

Nessuno

- start - eseguito ogni volta che la scena viene attivata.

Parametri Input

Nessuno

Parametri Output

Nessuno

- cleanup - eseguito una sola volta quando la scena viene rimossa dallo SceneManager.

Parametri Input

Nessuno

Parametri Output

Nessuno

- stop - eseguito ogni volta che la scena viene deattivata.

Parametri Input

Nessuno

Parametri Output

Nessuno

- handleKey - metodo invocato quando viene premuto un tasto del telecomando della Smart TV.

Parametri Input

keyCode - numero identificativo del tasto premuto

Parametri Output

Nessuno

- handleMouseDown - metodo invocato quando viene premuto il tasto sinistro del mouse o quando l'utente chiude il pugno.

Parametri Input

e - oggetto contenente informazioni relative all'evento

Parametri Output

Nessuno

- handleMouseUp - metodo invocato quando viene rilasciato il tasto sinistro del mouse o quando l'utente apre il pugno.

  Parametri Input

  e - oggetto contenente informazioni relative all'evento

  Parametri Output

  Nessuno

- handleMouseMove - metodo invocato quando viene mosso il puntatore del mouse o quando l'utente muove la mano.

  Parametri Input

  e - oggetto contenente informazioni relative all'evento

  Parametri Output

  Nessuno

- handleGesture - metodo invocato quando viene il sistema di Gesture Recognition riconosce un gesto.

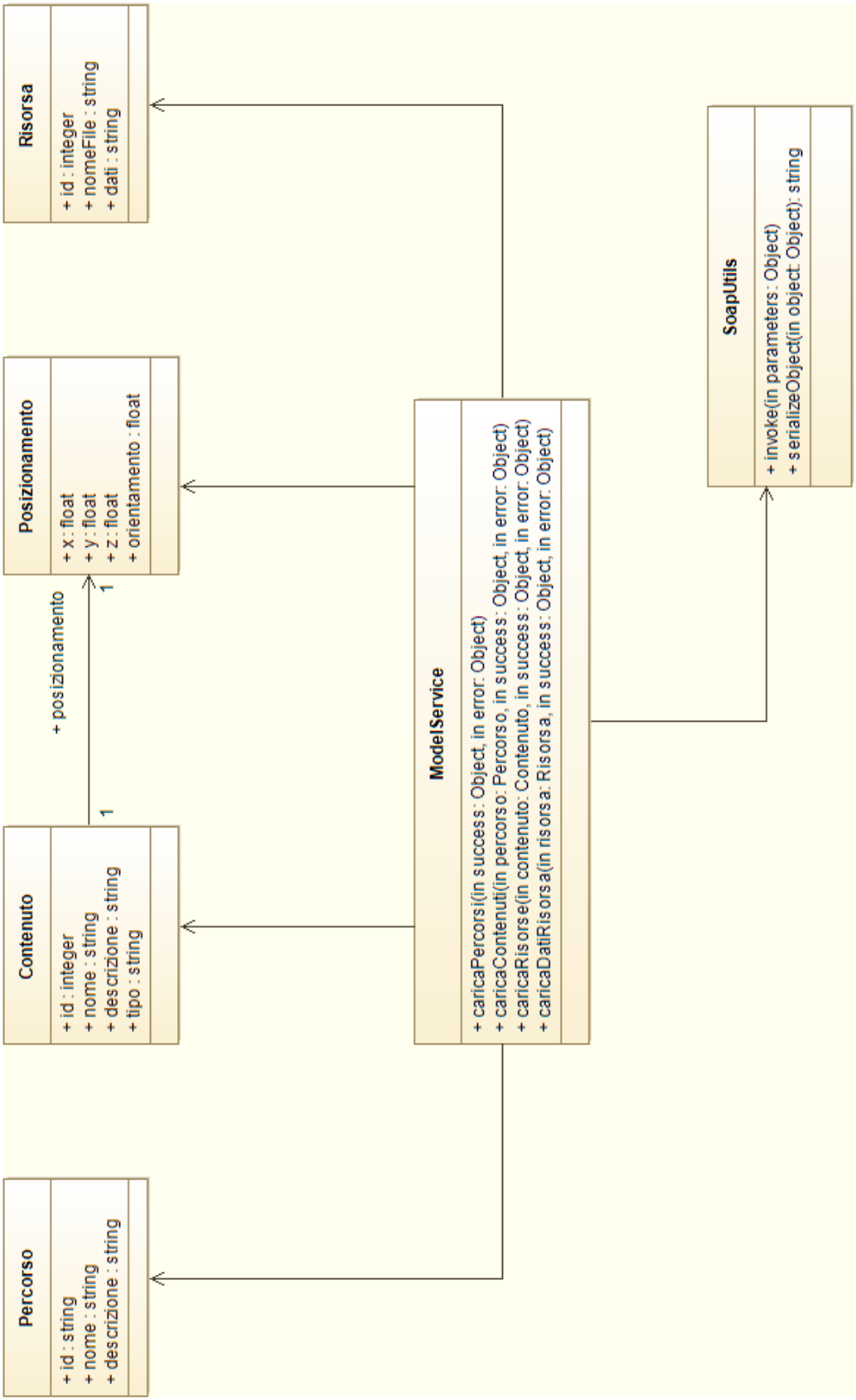
  Parametri Input

  e - oggetto contenente informazioni relative all'evento

  Parametri Output

  Nessuno

5.2.3.2 Package Model



- Classe **ModelService**

OPERAZIONI

- caricaPercorsi - carica tutti i Percorsi disponibili.

Parametri Input

success - callback invocata in caso di esito positivo dell'operazione. Tale callback accetta come parametro un array contenente i Percorsi caricati

error - callback invocata in caso di errore

Parametri Output

Nessuno

- caricaContenuti - carica i Contenuti di un Percorso.

Parametri Input

percorso - il Percorso i cui Contenuti sono richiesti

success - callback invocata in caso di esito positivo dell'operazione. Tale callback accetta come parametro un array contenente i Contenuti caricati

error - callback invocata in caso di errore

Parametri Output

Nessuno

- caricaRisorse - carica le Risorse di un Contenuto.

Parametri Input

contenuto - il Contenuto le cui Risorse sono richieste

success - callback invocata in caso di esito positivo dell'operazione. Tale callback accetta come parametro un array contenente le Risorse caricate

error - callback invocata in caso di errore

Parametri Output

Nessuno

- caricaDatiRisorsa - carica i dati di una Risorsa. Tali dati sono costituiti dal contenuto del file associato alla Risorsa, e sono rappresentati da una stringa codificata in base64.

Parametri Input

risorsa - la Risorsa di cui è richiesto caricare i dati

success - callback invocata in caso di esito positivo dell'operazione. Tale callback accetta come parametro la Risorsa i cui dati sono stati caricati

error - callback invocata in caso di errore

Parametri Output

Nessuno

- Classe **SoapUtils**

OPERAZIONI

- invoke - invoca un'operazione SOAP

Parametri Input

parameters - un oggetto le cui proprietà forniscono un'insieme di parametri di configurazione della richiesta SOAP da invocare. Le proprietà che devono essere definite sono:

- **url** l'url a cui inviare la richiesta
- **namespace** il namespace del Web Service SOAP
- **operation** il nome dell'operazione da invocare
- **arguments [opzionale]** un oggetto le cui proprietà costituiscono gli argomenti dell'operazione da invocare
- **success** callback invocata in caso di esito positivo dell'operazione. Tale callback accetta come parametro la risposta ricevuta dal Web Service
- **error** callback invocata in caso di errore

Parametri Output

Nessuno

- serializeObject - trasforma un oggetto in una stringa XML.

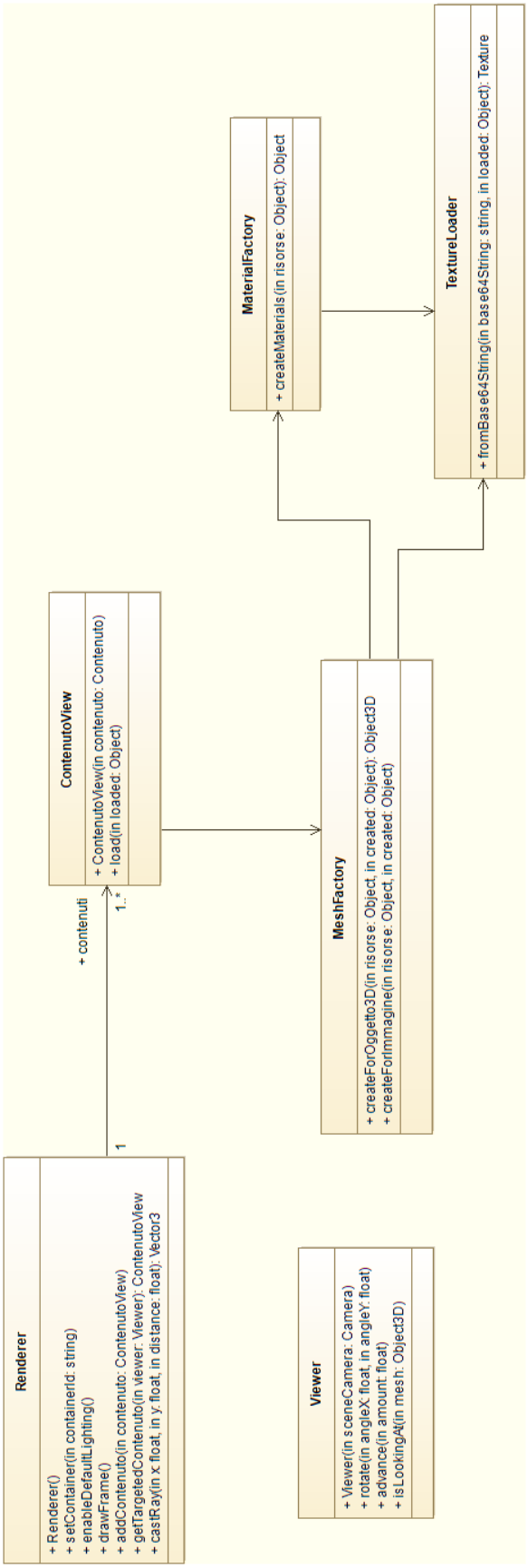
Parametri Input

object - l'oggetto da serializzare

Parametri Output

L'oggetto serializzato come stringa XML

5.2.3.3 Package View



- Classe **Renderer**

OPERAZIONI

- setContainer - imposta l'elemento DOM all'interno del quale verrà effettuato il rendering della scena 3D.

  Parametri Input

  containerId - valore id dell'elemento DOM da impostare come contenitore

  Parametri Output

  Nessuno

- enableDefaultLighting - aggiunge alcune luci predefinite alla scena.

  Parametri Input

  Nessuno

  Parametri Output

  Nessuno

- drawFrame - esegue il rendering della scena 3D.

  Parametri Input

  Nessuno

  Parametri Output

  Nessuno

- addContenuto - Aggiunge un Contenuto al Renderer.

Parametri Input

contenutoView - oggetto di tipo ContenutoView da aggiungere

Parametri Output

Nessuno

- getTargetedContenuto - restituisce il Contenuto (oggetto ContenutoView) inquadrato nella scena 3D. Se non è inquadrato alcun Contenuto viene restituito il valore "null".

Parametri Input

viewer - l'oggetto Viewer utilizzato durante il rendering della scena 3D

Parametri Output

Il Contenuto inquadrato, se esiste, il valore "null" altrimenti

- castRay - ad ogni punto bidimensionale sullo schermo si associa un "raggio" di lunghezza infinita che attraversa la scena 3D. Questo metodo restituisce un punto nella scena 3D lungo uno di questi raggi.

Parametri Input

x - coordinata X del punto sullo schermo

y - coordinata Y del punto sullo schermo

distance - distanza lungo il raggio del punto richiesto

Parametri Output

Il punto nella scena 3D (oggetto THREE.Vector3).

- Classe **Viewer**

OPERAZIONI

- rotate - ruota la direzione lungo la quale il Viewer "osserva" la scena 3D.

  Parametri Input

  yaw - angolo di rotazione attorno all'asse Y dello spazio tridimensionale. L'angolo è espresso in radianti.

  pitch - angolo di rotazione attorno all'asse perpendicolare all'asse Y a alla direzione del Viewer. L'angolo è espresso in radianti.

  Parametri Output

  Nessuno

- advance - muove il Viewer "in avanti".

  Parametri Input

  distance - lunghezza dello spostamento.

  Parametri Output

  Nessuno

- isLookingAt - determina se il Viewer sta inquadrando un Contenuto.

  Parametri Input

  contenutoView - l'oggetto ContenutoView da esaminare

  Parametri Output

  Un valore booleano che indica l'esito dell'operazione

- Classe **ContenutoView**

OPERAZIONI

- load - inizializza l'oggetto.

  Parametri Input

  loaded - callback invocata quando l'oggetto è stato inizializzato

  Parametri Output

  Nessuno

- Classe **MeshFactory**

  OPERAZIONI

  - createForOggetto3D - crea un oggetto di tipo THREE.Object3D che costituisce la "rappresentazione 3D" di un contenuto di tipo "OGGETTO3D". La classe THREE.Object3D è definita all'interno della libreria ThreeJs.

    Parametri Input

    risorse - le risorse associate ad un Contenuto di tipo "OGGETTO3D". L'unico formato supportato è OBJ (incluso file MTL eventualmente associato)

    created - callback invocata quando l'oggetto THREE.Object3D è stato creato

    Parametri Output

    Nessuno

- createForImmagine - crea un oggetto di tipo THREE.Object3D che costituisce la "rappresentazione 3D" di un contenuto di tipo "IMMAGINE". La classe THREE.Object3D è definita all'interno della libreria ThreeJs.

Parametri Input

risorse - le risorse associate ad un Contenuto di tipo "IMMAGINE". L'unico formato supportato è JPEG

created - callback invocata quando l'oggetto THREE.Object3D è stato creato

Parametri Output

Nessuno

5.3 Note sull'Implementazione

5.3.1 Librerie di Terzi

L'implementazione prevede l'utilizzo di alcune librerie (javascript) di terzi.

- **jQuery**

Url del progetto: http://jquery.com/

jQuery è una libreria di funzioni Javascript, per le applicazioni web, che si propone come obiettivo quello di semplificare la programmazione lato client delle pagine HTML. La sintassi di jQuery è studiata per semplificare la navigazione dei documenti e la selezione degli elementi DOM, creare animazioni, gestire eventi e implementare funzionalità AJAX. Il framework mira a mantenere la compatibilità tra browser diversi standardizzando gli oggetti messi a disposizione dall'interprete Javascript del browser.

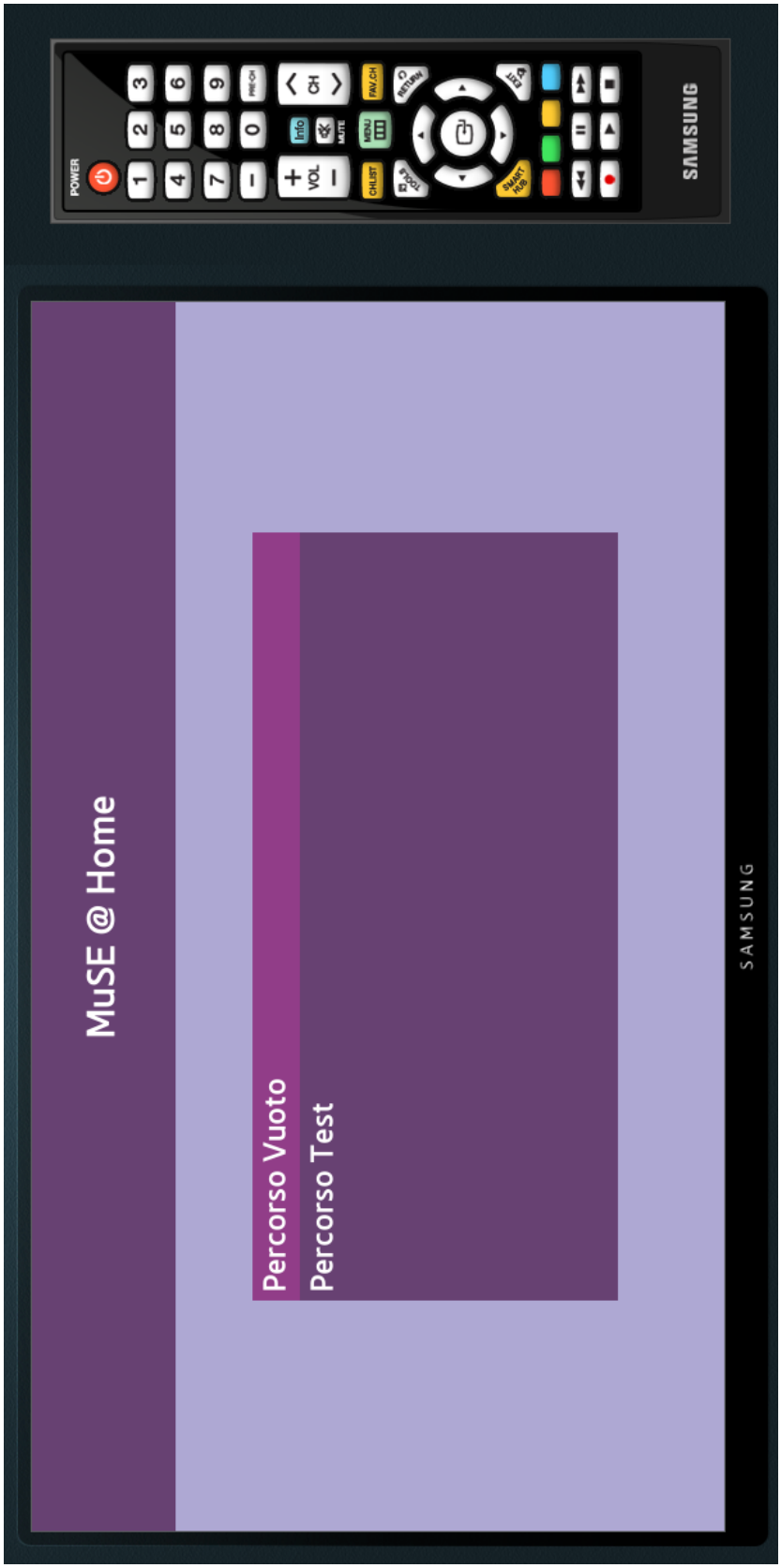
jQuery è un software liberamente distribuibile e gratuito, come previsto dalla licenza MIT.

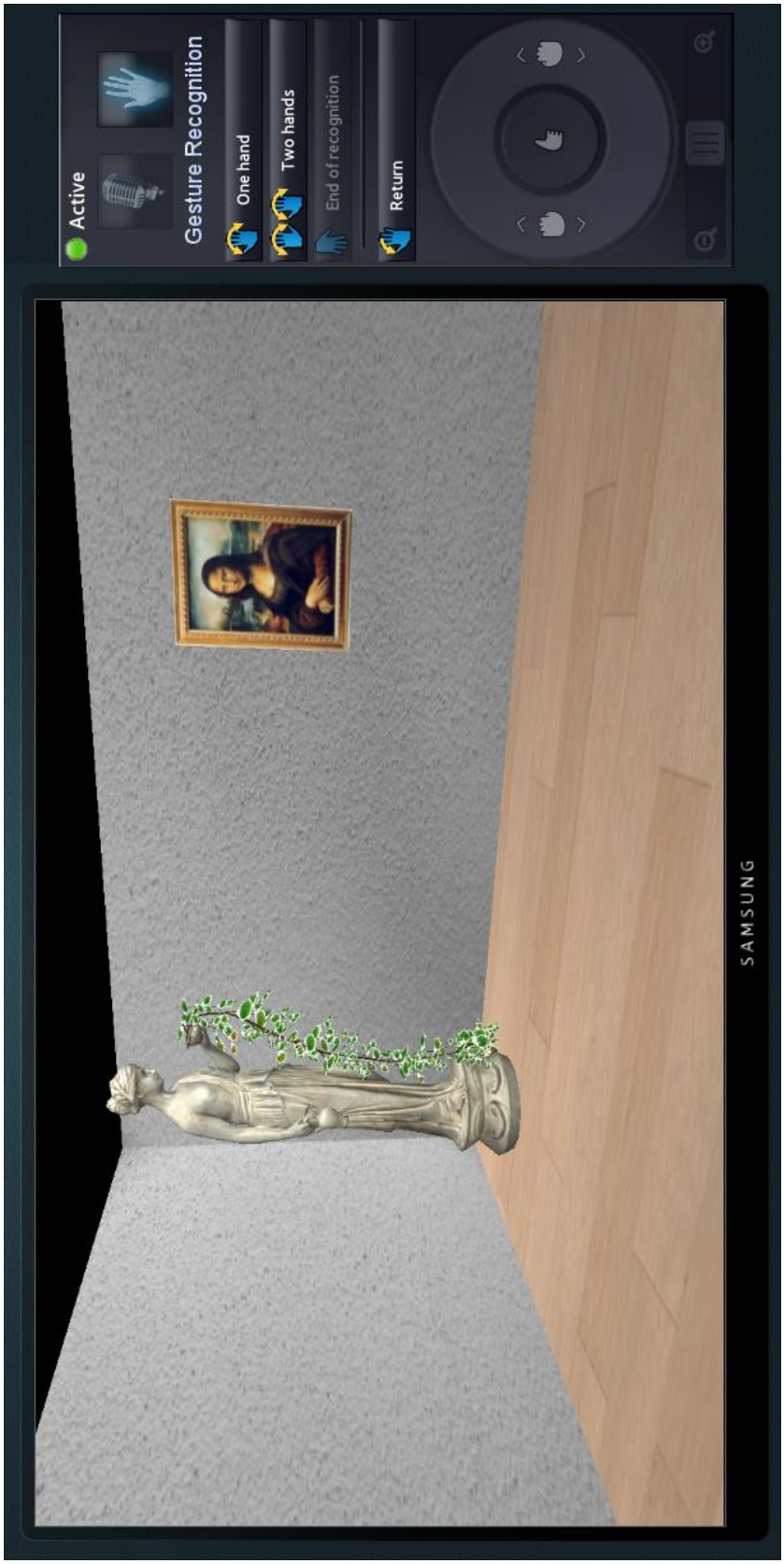
- **Three.js**

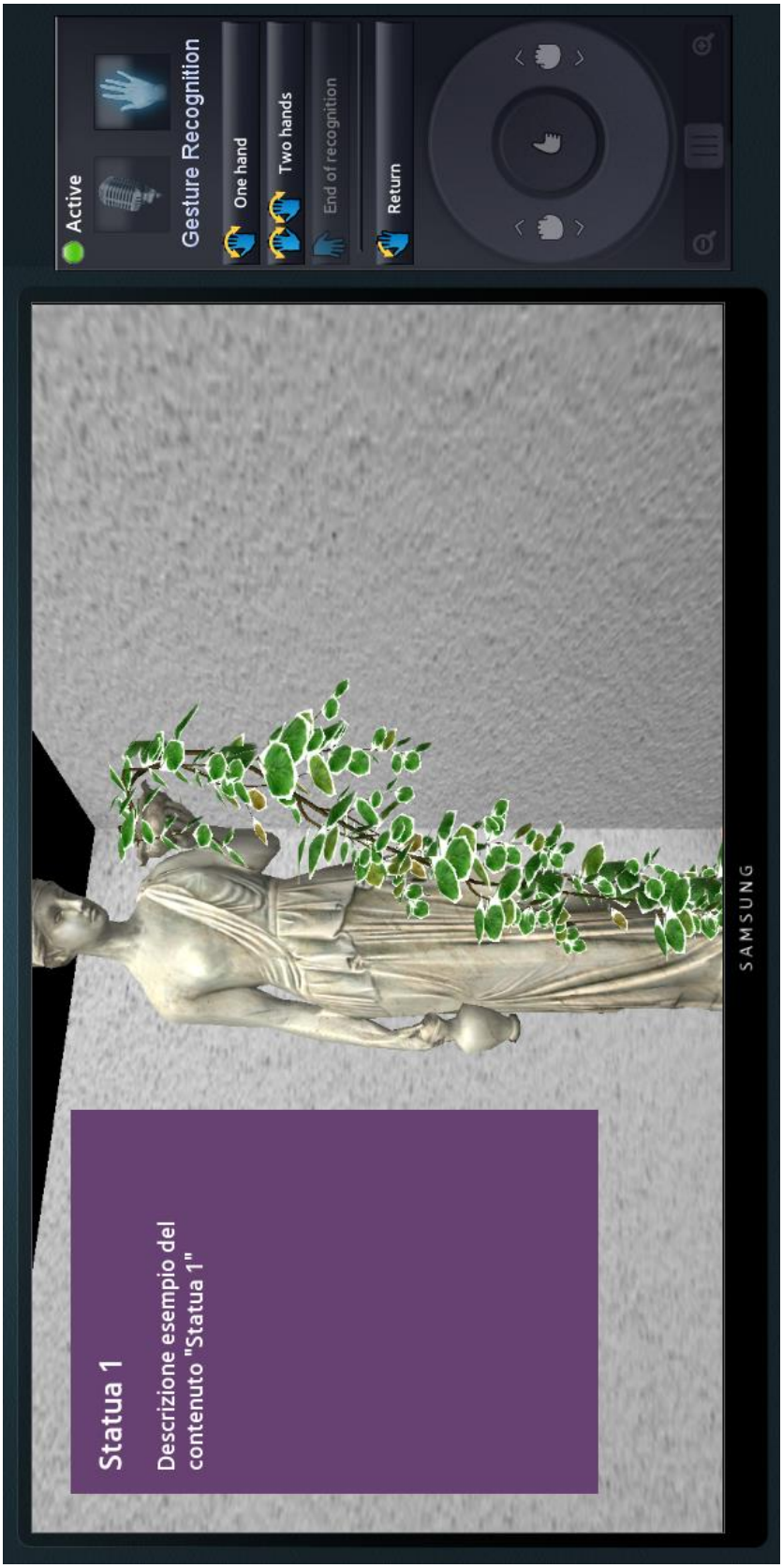
Url del progetto: http://threejs.org/

Three.js è una libreria Javascript usata per creare animazioni in computer-grafica 3D all'interno di un Browser Web. La API Three.js offre la possibilità di scegliere se utilizzare per il rendering l'elemento Canvas HTML5 o WebGL. Nel caso in cui si scelga WebGL Three.js farà pieno utilizzo dell'accelerazione grafica della GPU.
La libreria Three.js semplifica la creazione di Scene 3D di complessità arbitraria utilizzando il solo linguaggio Javascript senza dipendere da plugin esterni del browser.

5.4 Schermate


SAMSUNG
MuSE @ Home
Percorso Vuoto
Percorso Test
SAMSUNG


Active
Gesture Recognition
One hand
Two hands
End of recognition
Return
SAMSUNG


Active
Gesture Recognition
One hand
Two hands
End of recognition
Return
Statua 1
Descrizione esempio del contenuto "Statua 1"
SAMSUNG


Active
Gesture Recognition
One hand
Two hands
End of recognition
Return
Statua 1
SAMSUNG


Active
Gesture Recognition
One hand
Two hands
End of recognition
Return
La Gioconda
SAMSUNG

Capitolo 6: Web Application di Caricamento dei Contenuti

6.1 Analisi

In questo capitolo si affronta lo sviluppo della Web Application per il Caricamento dei Contenuti.

Lo scopo dell'Applicazione è quello di gestire l'insieme di dati che costituiscono le Visite Virtuali sfruttando i Web Services della piattaforma Muse @ Home la cui realizzazione è stata descritta nei capitoli precedenti.

In particolare, come più dettagliamente riporta il paragrafo che segue, è necessario interagire con la piattaforma al fine di gestire:

- **Percorsi** in questo contesto sono semplici strutture dati cui si associano i Contenuti
- **Contenuti** rappresentano le opere d'arte di un museo, ad essi sono associati uno o più file denominati "Risorse", possono essere di varia natura come ad esempio Immagini e Oggetti tridimensionali
- **Risorse** sono i file che costituiscono la rappresentazione grafica dei contenuti, possono dungue essere file Immagine, Video o Modelli Tridimensionali (come file con estensione .OBJ)

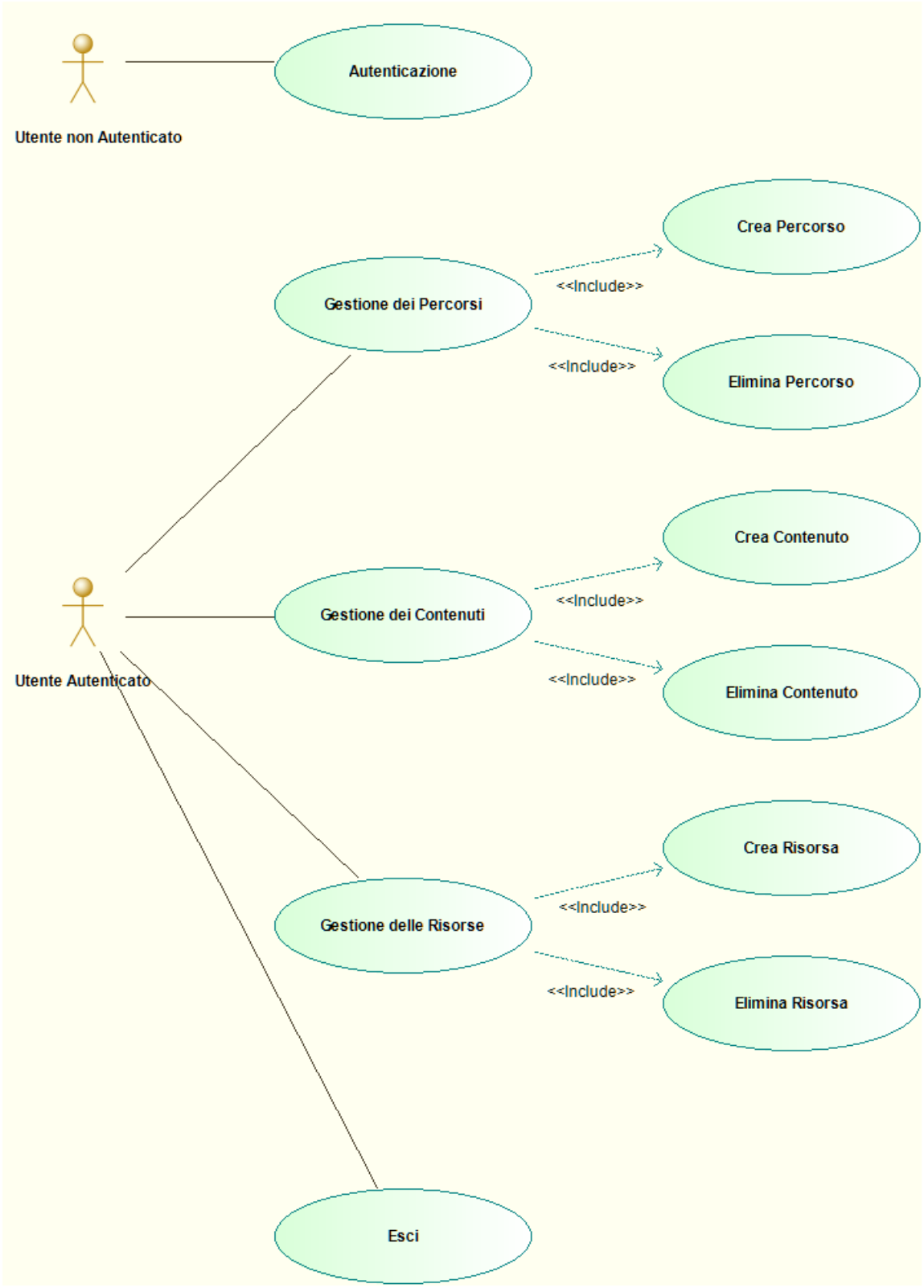
Come detto l'Applicazione, al fine di prelevare i dati e i file esistenti e di caricarne di

nuovi, sfrutta i Web Services messi a disposizione dalla piattaforma. Questo implica che nessun tipo di tecnologia per la memorizzazione di dati (ad esempio Database relazionale) dovrà essere impiegata durante la realizzazione dell'Applicazione.

6.1.1 Casi d'Uso

In questa sezione sono riportati il diagramma dei Casi d'Uso dell'Applicazione nonché la descrizione di ciascuno di essi.

Per tutte le operazioni l'Applicazione richiede l'autenticazione dell'identità dell'Utente. Esiste dunque una differenziazione fra "Utente *non* Autenticato" ed "Utente Autenticato".


Utente non Autenticato
Autenticazione
Utente Autenticato
Gestione dei Percorsi
Crea Percorso
Elimina Percorso
Gestione dei Contenuti
Crea Contenuto
Elimina Contenuto
Gestione delle Risorse
Crea Risorsa
Elimina Risorsa
Esci
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>
<<Include>>

Casi d'Uso per **Utente non Autenticato**

- **Autenticazione**

  L'utente effettua l'autenticazione con l'Applicazione tramite modulo web.

  Dettaglio Interazioni

  - L'utente compila il modulo inserendo le proprie credenziali
  - L'utente invia il modulo ed attende il risultato dell'operazione
  - **Successo** le credenziali vengono verificate e l'utente è reindirizzato alla pagina principale dell'Applicazione (Caso d'Uso **Gestione Percorsi**)
  - **Errore** il modulo contiene dati non validi (campi vuoti) o le credenziali non vengono riconosciute, per proseguire l'utente deve ripetere l'operazione

Casi d'Uso per **Utente Autenticato**

- **Gestione Percorsi**

  Vengono visualizzati i Percorsi esistenti associati all'Utente. Si possono effettuare le operazioni **Crea Percorso** ed **Elimina Percorso**.

- **Crea Percorso**

  La creazione di un Percorso prevede l'inserimento dei vari attributi che lo descrivono attraverso il riempimento di un modulo web.

  Dettaglio Interazioni

  - L'utente riempie il modulo inserendo: Nome, Descrizione
  - L'utente invia il modulo ed attende il risultato dell'operazione
  - **Successo** i dati inseriti sono validati ed un nuovo Percorso viene creato e visualizzato nella lista dei Percorsi esistenti
  - **Errore** il modulo contiene dati non validi, l'utente deve ripetere l'operazione inserendo dati corretti seguendo le indicazioni fornite

- **Elimina Percorso**

L'utente può eliminare uno o più dei Percorsi esistenti.

Dettaglio Interazioni

- L'utente seleziona uno o più Percorsi da eliminare
- L'utente conferma l'operazione premendo un pulsante
- I Percorsi vengono eliminati

- **Gestione Contenuti**

  Vengono visualizzati i Contenuti associati ad un Percorso. Si possono effettuare le operazioni **Crea Contenuto** ed **Elimina Contenuto**.

- **Crea Contenuto**

  La creazione di un Contenuto prevede l'inserimento dei vari attributi che lo descrivono attraverso il riempimento di un modulo web.

  Dettaglio Interazioni

  - L'utente riempie il modulo inserendo: Nome, Tipo (Immagine, Oggetto3d, Scenario, Video), Descrizione, Posizione (tripletta di coordinate [X, Y, Z]) all'interno dell'Ambiente, Rotazione (attorno all'asse Y, valore espresso in gradi)
  - L'utente invia il modulo ed attende il risultato dell'operazione
  - **Successo** i dati inseriti sono validati ed un nuovo Contenuto viene creato e visualizzato nella lista dei Contenuti esistenti
  - **Errore** il modulo contiene dati non validi, l'utente deve ripetere l'operazione inserendo dati corretti seguendo le indicazioni fornite

- **Elimina Contenuto**

  L'utente può eliminare uno o più dei Contenuti esistenti.

  Dettaglio Interazioni

  - L'utente seleziona uno o più Contenuti da eliminare

- L'utente conferma l'operazione premendo un pulsante
- I Contenuti vengono eliminati

- **Gestione Risorse**

  Vengono visualizzate le Risorse associati ad un Contenuto. Si possono effettuare le operazioni **Crea Risorsa** ed **Elimina Risorsa**.

- **Crea Risorsa**

  La creazione di un file Risorsa prevedere l'utilizzo di un controllo specializzato che deve permettere l'upload di file di qualsiasi tipo.

  Dettaglio Interazioni

  - L'utente indica i file da caricare, utilizzando un'apposita finestra di dialogo o tramite operazioni di tipo "drag & drop"
  - L'operazione viene onfermata premendo un pulsante
  - Dopo un periodo di attesa che dipende dalla dimensione dei file caricati le nuove Risorse vengono visualizzate correttamente nella schermata di **Gestione delle Risorse** (omonimo Caso d'Uso)

- **Elimina Risorsa**

  L'utente può eliminare una o più delle Risorse esistenti.

  Dettaglio Interazioni

  - L'utente seleziona una o più Risorse da eliminare
  - L'utente conferma l'operazione premendo un pulsante
  - Le Risorse vengono eliminate

- **Esci**

  Uscita dall'applicazione. L'utente entra nello stato di **Utente non Autenticato** e per tornare ad utilizzare l'applicazione dovrà nuovamente effettuare l'Autenticazione

(omonimo Caso d'Uso).

6.2 Progettazione

6.2.1 Architettura e Framework

Per la realizzazione del progetto Web si è scelta un'architettura di tipo MVC. Le motivazioni che hanno spinto a compiere tale scelta sono semplici:

1. Ad oggi è praticamente una metodologia standard per lo sviluppo di applicazioni web di ragionevole complessità

2. I benefici della separazione tra dati, logica applicativa e presentazione sono più che mai significativi in un ambiente come quello Web nel quale tali aspetti sono fortemente dinamici (soggetti a cambiamento). In particolare la possibilità di apportare modifiche in isolamento facilita l'integrazione dei vari componenti che costituiscono il software

3. Esistono numerosi framework open source di livello professionale per diverse piattaforme e linguaggi di programmazione che agevolano e quindi velocizzano lo sviluppo

A supporto dello sviluppo si è scelto di utilizzare il framework PHP open source **Zend Framework 2** [14]. Fra le più rilevanti caratteristiche di questo framework notiamo:

1. Basato sull'architettura MVC

2. Moderno, supporta tutte le features dell'ultima versione del linguaggio di scripting PHP

3. Nuove versioni con migliorie o correzioni di bug rilasciate di frequente

4. Documentazione esaustiva, ricca di esempi

5. Comunità molto vasta ed attiva

Una volta scelta architettura e framework di riferimento è possibile scomporre il processo di progettazione in varie fasi. Inizialmente è prevista la definizione di quegli oggetti che costituiscono i dati dell'applicazione (Model) e della logica che ne abilita il caricamento da e verso una fonte esterna della quale l'applicazione sarà idealmente agnostica.

La fase successiva, tenendo conto del framework selezionato e delle tecnologie a disposizione definirà gli oggetti appartenenti allo strato Controller. In questo strato risiede la logica che elabora le richieste utente (richieste HTTP) eseguendo (eventualmente) operazioni sui dati e fornendo una risposta. Nella maggior parte dei casi l'applicazione invia una risposta HTTP il cui contenuto è costituito da un documento HTML. Possono ovviamente esistere eccezioni.

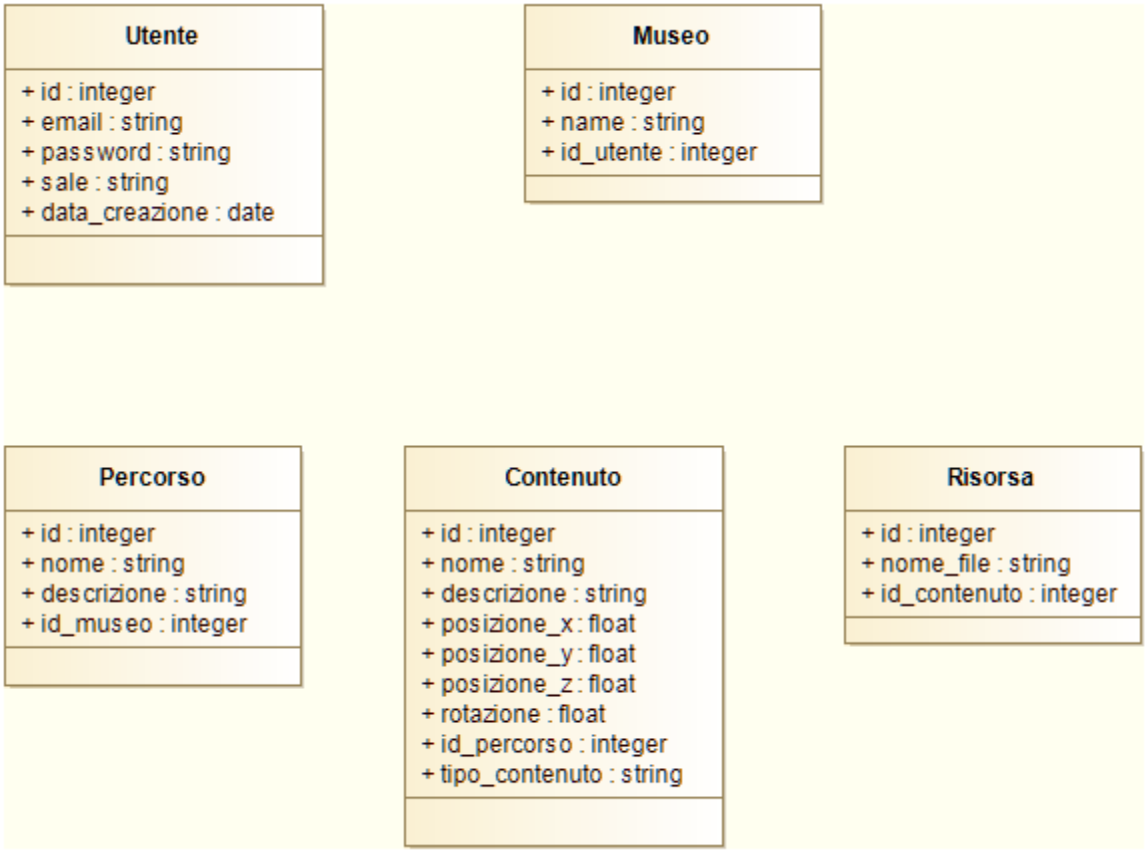
Infine, il layer View è costituito dall'insieme di oggetti che genera il contenuto delle risposte HTTP inviate dall'Applicazione.

6.2.2 Modello dei Dati

Come accennato, in questo paragrafo si descrive lo strato Model dell'architettura MVC. In particolare il seguente diagramma mostra gli oggetti che rappresentano i dati utilizzati dall'Applicazione.

Il concetto di **Utente** e **Museo** sono unici dell'Applicazione Web. Un Utente rappresenta un insieme di credenziali che permettono l'autenticazione e quindi l'utilizzo del sistema. Un Museo è associato ad un Utente ed è una semplice struttura dati che contiene dati descrittivi della struttura cui appartengono i Contenuti caricati. Al momento questi dati consistono unicamente in un nome. I restanti oggetti rispecchiano molto da vicino quelli già visti nei capitoli precedenti e sembra inutile ripeterne la descrizione per l'ennesima volta.

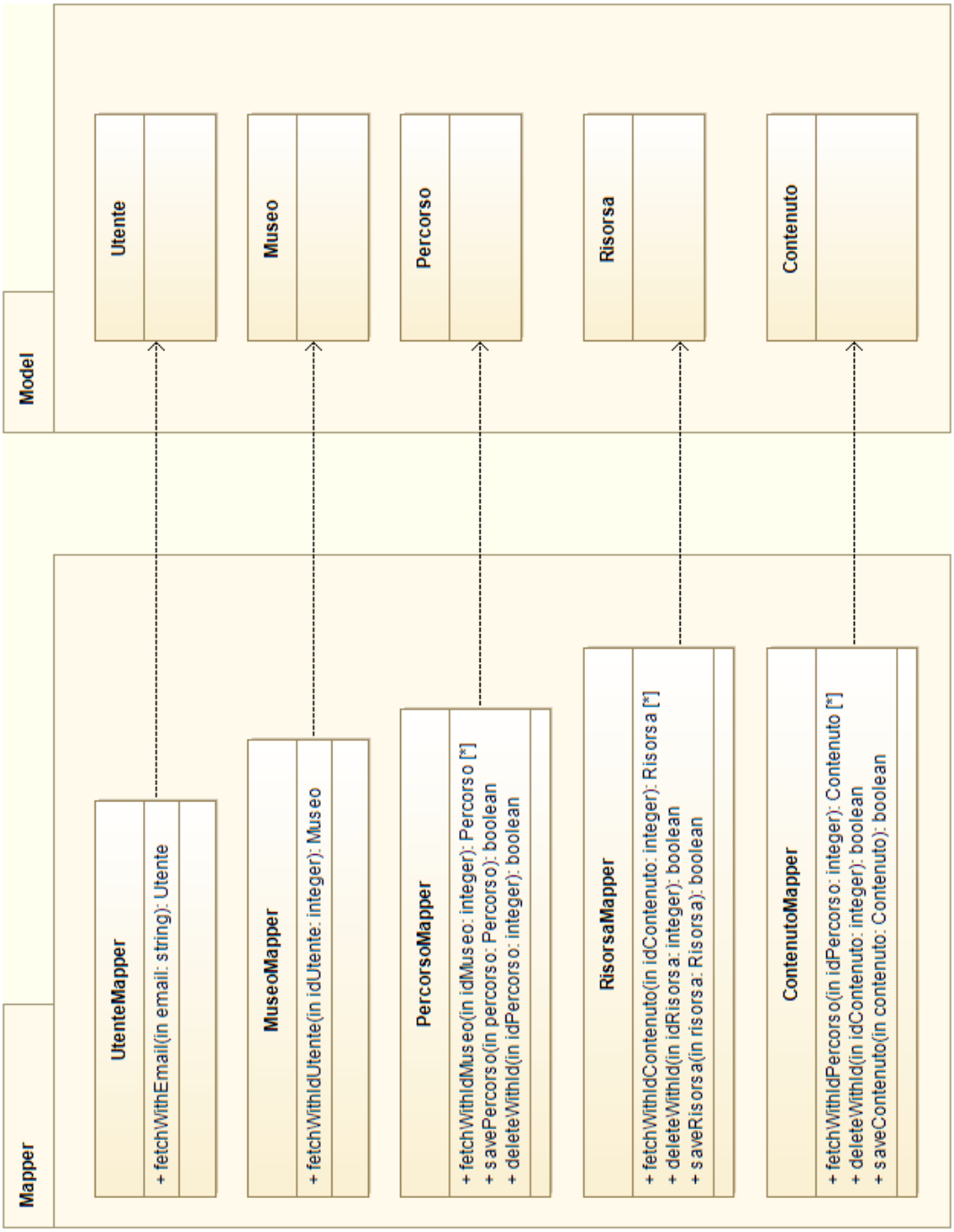
La semplicità del modello è dovuta al fatto che questi oggetti non contengono alcuna logica ma sono semplici contenitori di dati. La ragione di ciò è la scelta dell'utilizzo del pattern *Data Mapper*. Questo pattern rende possibile eseguire le operazioni di caricamento dei dati in modo trasparente all'applicazione tramite la creazione di oggetti appositi (i Data Mappers) la cui responsabilità e caricare/salvare le entità. In altri termini, l'Applicazione n



e

nza del modo in cui i dati sono effettivamente memorizzati e resi disponibili ma dovrà utilizzare gli appositi Mappers. Data un'interfaccia pubblica qualunque modifica all'implementazione di un oggetto Mapper non verrà percepita dal codice che lo utilizza. Ad esempio diviene possibile migrare da una fonte di dati di tipo Database relazionale (come Oracle Db o MySql) verso una fonte totalmente diversa (come un Web Service SOAP o una REST Api) senza dover modificare alcuna parte del codice client. Immediati benefici sono anche la facilità di testing in quanto è banale implementare Mock dei Data Mappers che restituiscano oggetti predefiniti.

Il diagramma seguente mostra le classi Mapper.



- **UtenteMapper**

  Mapper dell'entità Utente.

  OPERAZIONI

  - fetchWithEmail - Restituisce l'Utente con l'indirizzo email specificato.

    Parametri Input

email - indirizzo email dell'Utente richiesto

Parametri Output

L'Utente richiesto o null se tale oggetto non esiste

- **MuseoMapper**

  Mapper dell'entità Museo.

  OPERAZIONI

  - fetchWithIdUtente - Restituisce il Museo associato all'Utente con il numero id specificato

    Parametri Input

    idUtente - numero id dell'Utente cui è associato il Museo richiesto

    Parametri Output

    Il Museo richiesto o null se tale oggetto non esiste

- **PercorsoMapper**

  Mapper dell'entità Percorso.

  OPERAZIONI

  - fetchWithIdMuseo - Restituisce i Percorsi associati al Museo con il numero id specificato

    Parametri Input

    idMuseo - numero id del Museo cui sono associati i Percorsi richiesti

    Parametri Output

    Insieme dei Percorsi richiesti (può essere vuoto)

  - savePercorso - Salva il Percorso specificato

    Parametri Input

percorso - entità Percorso da salvare

Parametri Output

Un valore booleano che indica l'esito dell'operazione

- deleteWithId - Elimina il Percorso con il numero id specificato

  Parametri Input

  idPercorso - numero id del Percorso da eliminare

  Parametri Output

  Un valore booleano che indica l'esito dell'operazione

- **ContenutoMapper**

  Mapper dell'entità Contenuto.

  OPERAZIONI

  - fetchWithIdPercorso - Restituisce i Contenuti associati al Percorso con il numero id specificato

    Parametri Input

    idPercorso - numero id del Percorso cui sono associati i Contenuti richiesti

    Parametri Output

    Insieme dei Conteuti richiesti (può essere vuoto)

  - saveContenuto - Salva il Contenuto specificato

    Parametri Input

    contenuto - Entità Contenuto da salvare

    Parametri Output

    Un valore booleano che indica l'esito dell'operazione

- deleteWithId - Elimina il Contenuto con il numero id specificato

  Parametri Input

  idContenuto - numero id del Contenuto da eliminare

  Parametri Output

  Un valore booleano che indica l'esito dell'operazione

- **RisorsaMapper**

  Mapper dell'entità Risorsa.

  OPERAZIONI

  - fetchWithIdContenuto - Restituisce le Risorse associate al Contenuto con il numero id specificato

    Parametri Input

    idContenuto - numero id del Contenuto cui sono associate le Risorse richieste

    Parametri Output

    Insieme delle Risorse richieste (può essere vuoto)

  - saveRisorsa - Salva la Risorsa specificata

    Parametri Input

    risorsa - Entità Risorsa da salvare

    Parametri Output

    Un valore booleano che indica l'esito dell'operazione

  - deleteWithId - Elimina la Risorsa con il numero id specificato

    Parametri Input

    idRisorsa - numero id della Risorsa da eliminare

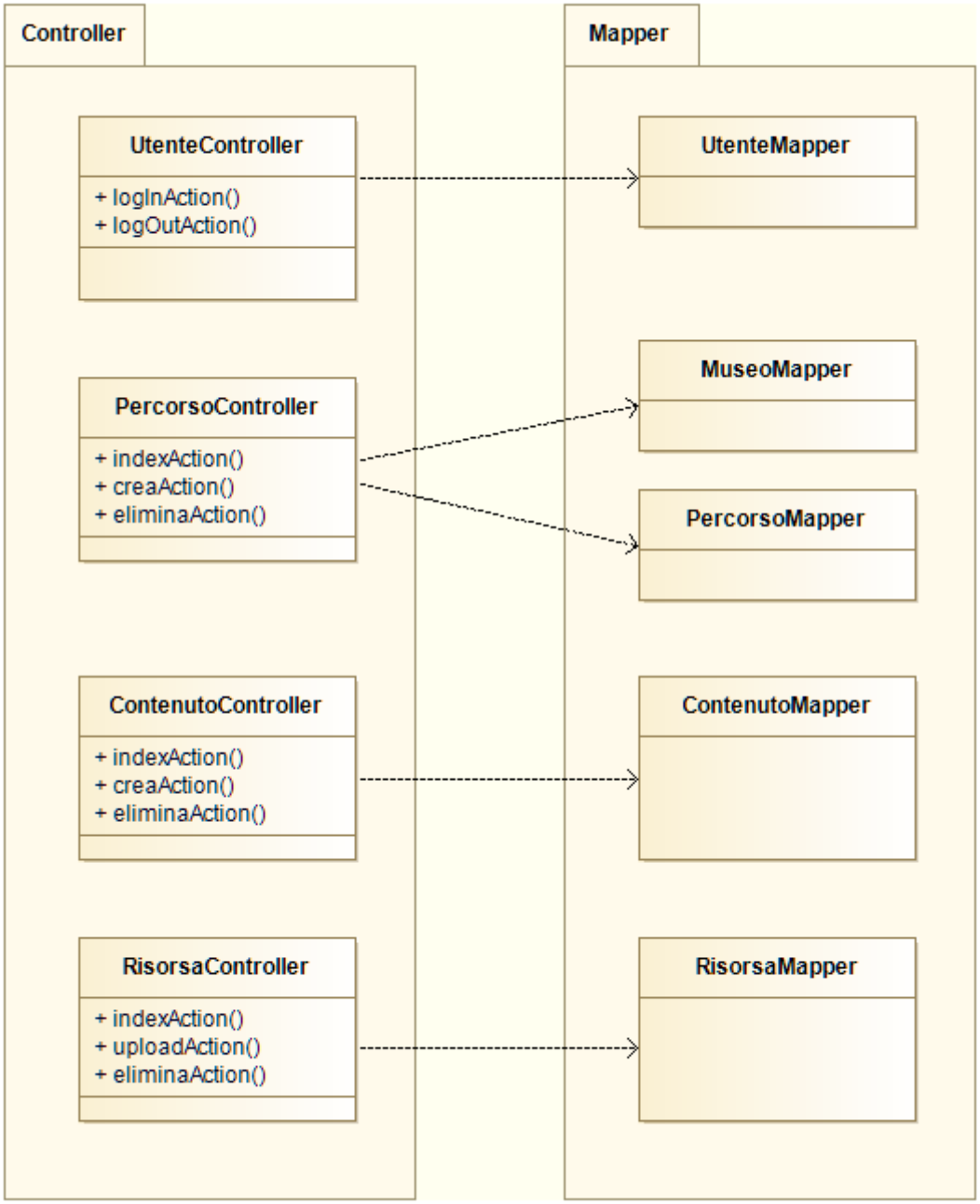
    Parametri Output

Un valore booleano che indica l'esito dell'operazione

6.2.3 Controller

Nello strato Controller risiede la logica applicativa che elabora le richieste dell'utente. Nel caso delle applicazioni web questo significa di frequente manipolare oggetti dello strato Model e inviare risposte HTTP contenenti documenti HTML. Il caso dell'Applicazione in esame non è diverso. Le diverse azioni che l'utente può eseguire (che si rifanno ai Casi d'Uso evidenziati in fase di analisi) sono raggruppate logicamente e implementate tramite i Controller. Un Controller dunque rappresenta un insieme di azioni logicamente correlate e che, potenzialmente, richiedono elaborazioni differenti ma simili o che comunque interagiscono con oggetti Model dello stesso tipo. Ad esempio un Controller può gestire tutte le operazioni inerenti la gestione degli utenti di un sito web, come log-in, log-out, registrazione e password recovery. Un secondo approccio vede i Controller come Risorse manipolabili attraverso alcune azioni. Nell'esempio precedente la risorsa è l'utente del sito web mentre le azioni rimangono le stesse.

Nel caso dell'Applicazione di Caricamento dei Contenuti della piattaforma Muse @ Home verrà realizzato un Controller per ogni entità di business.



Osservando il diagramma si può facilmente notare come le operazioni eseguite dai Controller siano di fatto i Casi d'Uso dell'Applicazione. In particolare:

- **UtenteController**

  logInAction - Caso d'Uso **Autenticazione**

  logOutAction - Caso d'Uso **Esci**

- **PercorsoController**

  indexAction - Caso d'Uso **Gestione dei Percorsi**

  creaAction - Caso d'Uso **Crea Percorso**

  eliminaAction - Caso d'Uso **Elimina Percorso**

- **ContenutoController**

  indexAction - Caso d'Uso **Gestione dei Conteuti**

  creaAction - Caso d'Uso **Crea Contenuto**

  eliminaAction - Caso d'Uso **Elimina Contenuto**

- **RisorsaController**

  indexAction - Caso d'Uso **Gestione delle Risorse**

  uploadAction - Caso d'Uso **Crea Risorsa**

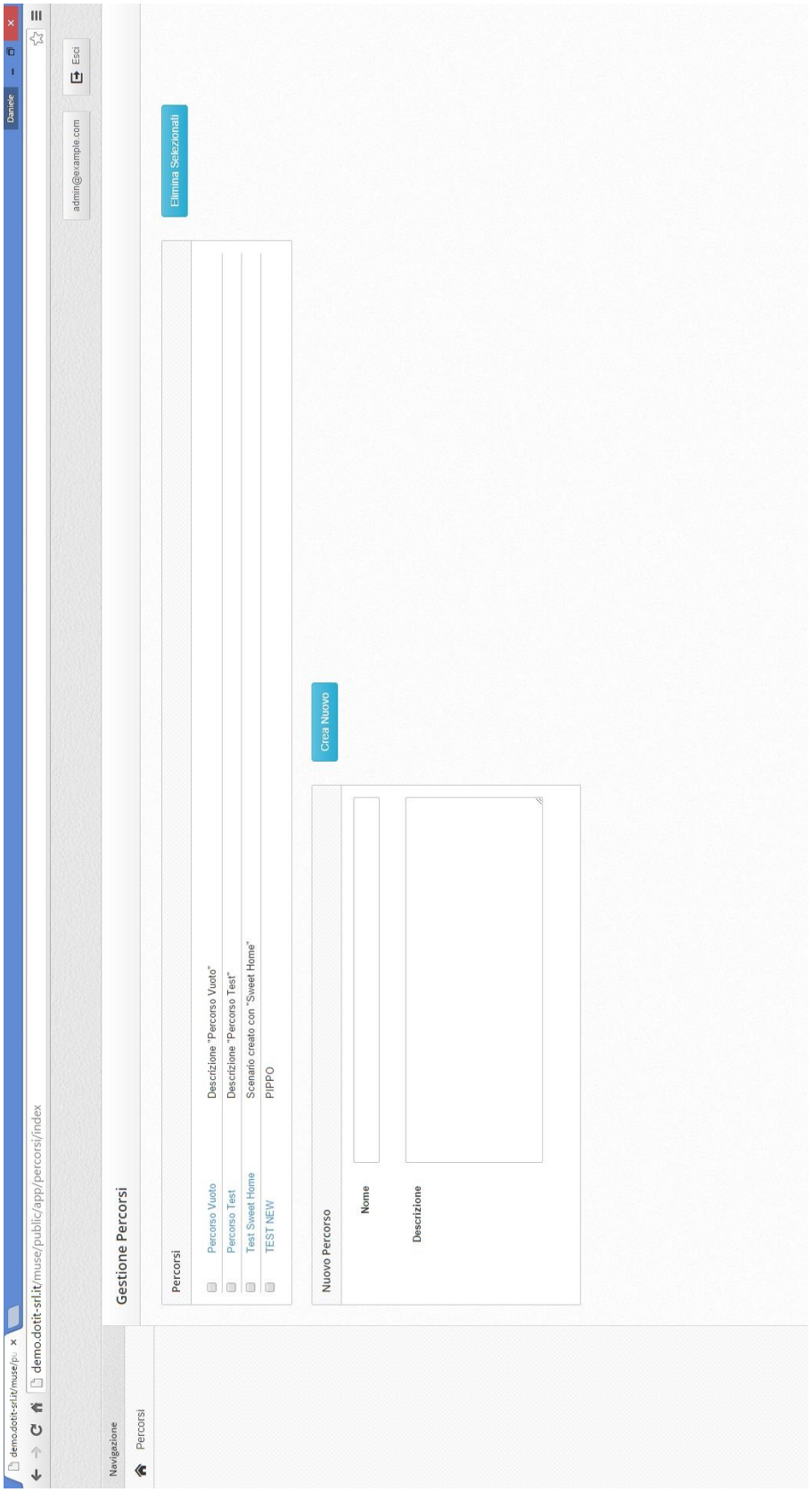
  eliminaAction - Caso d'Uso **Elimina Risorsa**

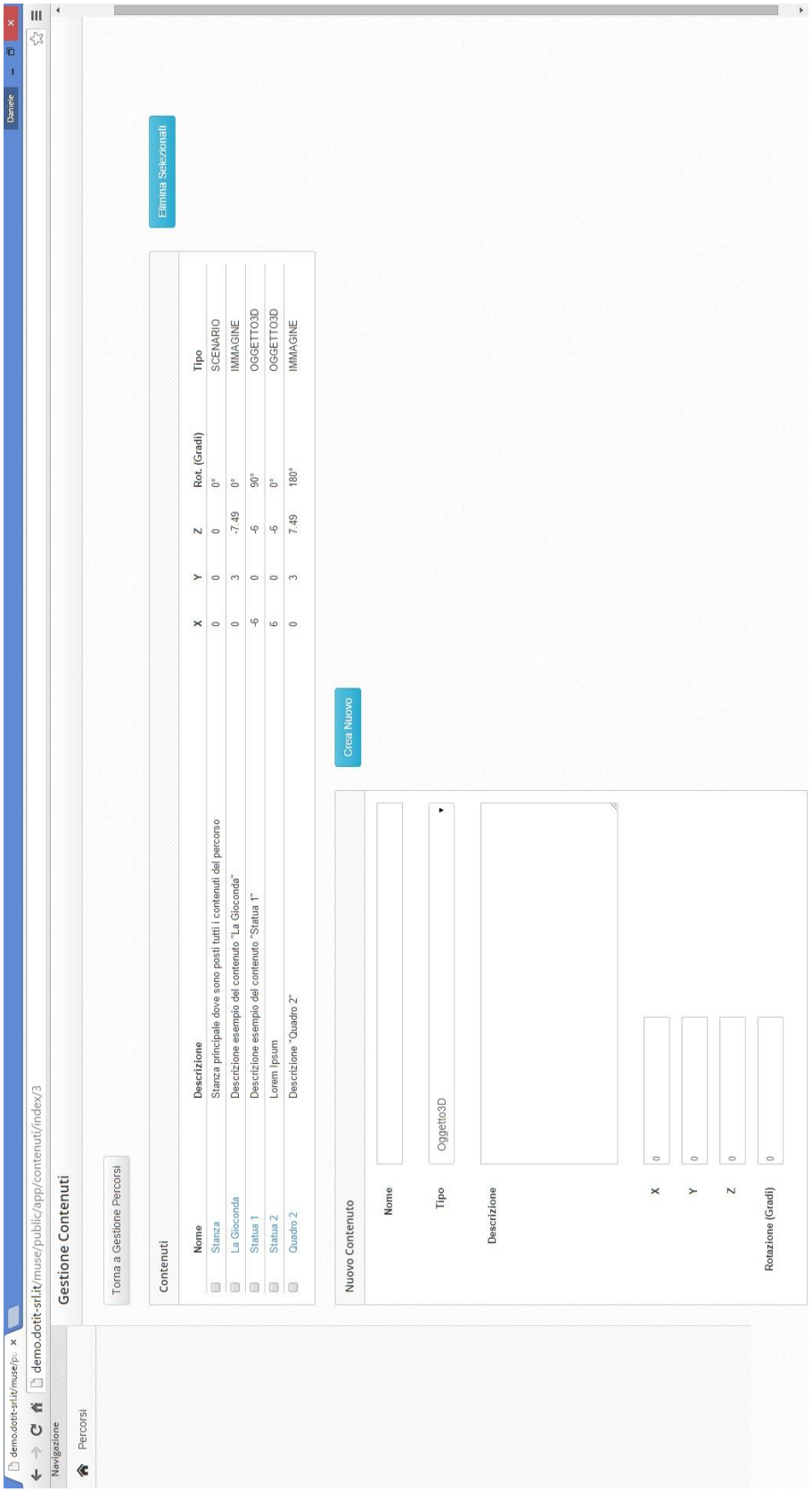
6.2.4 View

Il layer View dell'Applicazione è costituito da script PHP il cui scopo è generare i documenti HTML inviati in risposta alle richieste dell'utente. Quest'operazione viene detta rendering. In generale un View-Script ha accesso ad alcuni oggetti dello strato Model (tipica dipendenza dell'architettura MVC tra i package View e Model) che vengono iniettati dal Controller che sta elaborando la richiesta HTTP.

Data la relativa semplicità dell'Applicazione è poco rilevante, in fase di progettazione, esaminare più in dettaglio questo strato dell'architettura.

6.3 Schermate


Esci
admin@example.com
Gestione Percorsi
Elimina Selezionati
Percorsi
Percorso Vuoto
Descrizione "Percorso Vuoto"
Percorso Test
Descrizione "Percorso Test"
Test Sweet Home
Scenario creato con "Sweet Home"
TEST NEW
PIPPO
Crea Nuovo
Nuovo Percorso
Nome
Descrizione
Navigazione
Percorsi


demo.dotit-srl.it/muse/pu
demo.dotit-srl.it/muse/public/app/contenuti/index/3
Navigazione
Percorsi
Gestione Contenuti
Torna a Gestione Percorsi
Contenuti
Elimina Selezionati
Nome
Descrizione
X
Y
Z
Rot. (Gradi)
Tipo
Stanza
Stanza principale dove sono posti tutti i contenuti del percorso
0
0
0
0°
SCENARIO
La Gioconda
Descrizione esempio del contenuto "La Gioconda"
0
3
-7.49
0°
IMMAGINE
Statua 1
Descrizione esempio del contenuto "Statua 1"
-6
0
-6
90°
OGGETTO3D
Statua 2
Lorem Ipsum
6
0
-6
0°
OGGETTO3D
Quadro 2
Descrizione "Quadro 2"
0
3
7.49
180°
IMMAGINE
Crea Nuovo
Nuovo Contenuto
Nome
Tipo
Oggetto3D
Descrizione
X
Y
Z
Rotazione (Gradi)


Esci
admin@example.com
Gestione Risorse
Torna a Gestione Contenuti
Elimina Selezionati
Risorse
FLOOR.jpg
ROOM.mtl
ROOM.obj
WALL.jpg
Caricamento File
Select files
Add files to the upload queue and click the start button.
Filename
Size
Status
Drag files here.
Add files
Start upload
0 b
0%
Navigazione
Percorsi

Conclusioni

L'obiettivo finale del progetto prevedeva lo sviluppo in forma prototipale di una piattaforma integrata per la fruizione di Percorsi Virtuali Museali da casa. Da quanto proposto in questo lavoro è possibile affermare che tale obiettivo sia stato perseguito in modo soddisfacente. Si è visto come la fase di realizzazione sia suddivisa in diversi momenti.

In prima istanza sono stati individuati progetti esistenti di natura simile e le tecnologie utilizzabili al fine di sviluppare il sistema oggetto di questo lavoro. Tra le varie proposte presenti attualmente sul mercato si descrive come nessuna offra le funzionalità che il progetto Muse@Home si propone di offrire all'utente finale.

A questa fase segue la modellazione dei Business Processes che incarnano gli aspetti fondamentali del sistema: le Visite Virtuali vere e proprie e la creazione delle stesse da parte di utenti specifici (vale a dire non gli stessi utenti finali domestici). La realizzazione dei processi ha ricoperto un ruolo delicato durante la fase di analisi in quanto descrive il comportamento del sistema e aiuta ad individuare le interazioni tra i vari partecipanti.

Data la scelta quasi naturale di un'Architettura orientata ai Servizi (SoA) si è proceduto con la progettazione della stessa e dunque la realizzazione di Web Services utilizzati estensivamente per interazioni di natura eterogenea con la piattaforma. Tale fase progettuale culmina con la produzione di artefatti che specificano in maniera esaustiva tali

servizi.

Una volta che i servizi sono stati definiti (e realizzati) è stato possibile procedere con la progettazione e implementazione di una semplice (e volutamente limitata per quanto riguarda le funzionalità) Applicazione Web che permettesse di inviare alla piattaforma i contenuti multimediali che gli utenti finali avrebbero visualizzato durante le Visite Virtuali. Tale applicazione è stata realizzata utilizzando tecnologie standard largamente impiegate per progetti web di molteplici tipologie. Sebbene sia adatta allo scopo di rendere il sistema utilizzabile e testabile in fase prototipale questa parte del progetto è sicuramente quella che richiede maggiori sviluppi e attenzione.

La parte finale del lavoro consiste nell'analisi, progettazione ed implementazione dell'Applicazione Client per piattaforme Smart TV. Di fatto il progetto realizzato costituisce un prototipo il cui scopo è dimostrare le potenzialità (e possibilità) delle Smart TV in questo ambito. L'applicazione sviluppata presenta le caratteristiche principali che il prodotto finale dovrebbe possedere, vale a dire:

- La possibilità di visualizzare un elenco delle Visite Virtuali disponibili e la selezione di una di queste

- La navigazione all'interno dell'ambiente tridimensionale utilizzando solo la Gesture Recognition (riconoscimento del movimento delle mani e dei gesti)

- La visualizzazione in dettaglio di una singola opera d'arte con la possibilità di ruotare e ingrandire o rimpicciolire il modello tridimensionale che la rappresenta ancora una volta utilizzando la Gesture Recognition

Sviluppi Futuri

Il prototipo sviluppato e la piattaforma nella sua interezza presentano alcune limitazioni e

mancano di alcune delle funzionalità che il prodotto finale dovranno possedere.

Per prima cosa bisogna notare come in futuro dovrà essere prevista la possibilità di effettuare pagamenti direttamente dal client Smart TV per poter acquistare un "Biglietto Virtuale" che garantirà l'accesso a determinate visite non gratuite. Il progetto prevede un servizio apposito di cui però è stato implementato solo uno Stub. La realizzazione di tale servizio necessita la collaborazione di terzi (ad esempio un istituto di credito) e comunque esula dallo scopo di questo lavoro. Ciò nonostante l'assenza di una reale implementazione di tale servizio merita di essere menzionata.

Per quanto riguarda la realizzazione del client Smart TV è importante considerare le seguenti limitazioni:

- Il prototipo è pensato per funzionare con televisori prodotti dall'azienda Samsung e solo con i modelli presenti sul mercato dall'anno 2013 in poi. Al momento della stesura di questa Tesi Samsung è l'unico produttore ad offrire un supporto completo alla funzionalità di Gesture Recognition.

- Essendo l'utilizzo della Gesture Recognition come sistema di controllo all'avanguardia uno dei punti chiave del progetto non è prevista la possibilità di "comandare" l'applicazione tramite telecomando. In caso in cui il modello in utilizzo non abbia le funzionalità richieste per supportare questa tecnologia l'applicazione semplicemente mostrerà un messaggio di errore all'utente che non potrà dunque fruire della stessa. Sarebbe opportuno che il prodotto finale potesse essere utilizzato anche con un normale telecomando Smart TV in modo da allargare il bacino d'utenza (sebbene quest'opzione vada a degradare la qualità dell'esperienza).

- Supporto per il solo formato .OBJ per quanto riguarda i modelli tridimensionali. I limiti più importanti di questo formato sono le grandi dimensioni dei file (in quanto

i dati sono scritti in formato testuale leggibile e modificabile dagli esseri umani) e il supporto non proprio sofisticato nella definizione dei materiali (un materiale definisce il modo in cui un oggetto 3d appare, ad esempio ne indica la "texture" o il modo in cui la luce interagisce con esso).

- In fase di testing sono stati considerati ambienti di dimensioni contenute e non si è dunque potuto procedere con un'analisi approfondita delle prestazioni legate al rendering 3d in tempo reale. Con una certa probabilità gli ambienti virtuali utilizzati realmente dovranno essere suddivisi in parti più piccole in modo da ridurre in numero di poligoni che arrivano alla fase di rastering della pipeline della gpu (tramite il cosiddetto "Frustum Culling" che è già implementato nell'applicazione). Va detto che questo aspetto non è strettamente legato al client ma più che altro al modo in cui vengono realizzate le ricostruzioni tridimensionali degli ambienti in cui è possibile navigare.

- L'implementazione un meccanismo di "Collision Detection" più sofisticato permetterebbe la navigazione all'interno di ambienti più interessanti. In particolare sarebbe possibile per l'utente salire scale o piccoli "slope" rendendo il tutto particolarmente più realistico. Ancora una volta questa limitazione è stata prevista in fase di progetto ma la realizzazione di questa funzionalità, nemmeno particolarmente onerosa, è quasi d'obbligo nel prodotto finale.

Queste le principali limitazione dell'applicazione client.

Il problema più significativo del progetto è il modo in cui vengono caricati i contenuti delle visite. Attualmente viene utilizzata una semplice applicazione web che tramite i tradizionali "form" permette di inserire nome e descrizione delle opere d'arte e di effettuare l'upload di file risorsa (immagini e modelli tridimensionali) ad esse legati. Si intuisce facilmente come la definizione di un ambiente tridimensionale richiederebbe un

feedback visivo di ciò che si sta facendo piuttosto che semplici indicazioni testuali. Per risolvere questo problema si è pensato alla realizzazione di un vero e proprio software "desktop" che permetta la definizione di architetture e l'inserimento di oggetti 3d prefabbricati all'interno di esse tramite operazioni di "drag & drop". La realizzazione di tale software costituisce chiaramente un progetto a parte ed è immediato comprendere come sia anche particolarmente onerosa. Per tale motivo si è ritenuto opportuno scavalcarla completamente durante la fase prototipale e dunque esula dallo scopo di questo lavoro.

Bibliografia

[1] Musei Capitolini | Virtual Tour, http://tourvirtuale.museicapitolini.org/

[2] Visite Online, http://mv.vatican.va/2_IT/pages/MV_Visite.html

[3] Google Cultural Institute, https://www.google.com/culturalinstitute/project/art-project?hl=it

[4] Google Earth, https://www.google.it/intl/it/earth/

[5] Smart TV Alliance Home, http://www.smarttv-alliance.org/

[6] Smart TV Alliance - Manufacturers, http://smarttv- alliance.org/Markets/Manufacturers.aspx

[7] È in arrivo Android TV, http://www.android.com/tv/

[8] WebGL - OpenGL ES 2.0 for the Web, https://www.khronos.org/webgl/

[9] Unity - Game Engine, http://unity3d.com/

[10] BPMN 2.0, http://www.omg.org/spec/BPMN/2.0/

[11] Adriano Comai, Introduzione a BPMN, 2011

[12] OMG, BPMN 2.0 By Example, 2010

[13] OMG, Service oriented architecture Modeling Language (SoaML) Specification, 2012

[14] Zend Framework, http://framework.zend.com/